

Heterogeneous Multi-Robot Person-Following in Constrained Environments

A Thesis

Presented to

the Faculty of the Daniel Felix Ritchie School of Engineering and Computer

Science

University of Denver

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Ori A. Miller

August 2024

Advisor: Dr. Christopher Reardon

©Copyright by Ori A. Miller 2024

All Rights Reserved

Author: Ori A. Miller
Title: Heterogeneous Multi-Robot Person-Following in Constrained
Environments
Advisor: Dr. Christopher Reardon
Degree Date: August 2024

Abstract

Maintaining visibility of a person requires effective systems. Security cameras or ground robots might be ideal, but they often fail in uncontrolled or unknown environments. A single ground robot struggles to navigate and track an agent at the same time. This work addresses the challenge by developing a multi-robot system with a slow ground robot and an agile aerial robot. Three methods are evaluated: FORWARD-PF, RL-Person Following (RL), and a baseline closed-loop method. FORWARD-PF proved the most reliable, completing all nine paths and reaching targets nearly twice as fast as RL. Despite completing seven paths, RL faltered on complex tasks. The closed-loop method succeeded only 33% of the time, managing just the easiest paths. These findings underscore FORWARD-PF's efficiency and reveal both the potential and limitations of RL, demonstrating the promise of heterogeneous multi-robot systems for person-following in unpredictable settings.

Acknowledgments

I express my deep gratitude to all who supported me throughout this journey. Special thanks to my advisor, Dr. Christopher Reardon, for invaluable guidance and mentorship. Thanks to Dr. Jason Gregory and the Army Research Lab for their exceptional mentorship. Acknowledgement to the University of Denver and Sandia National Lab for essential resources. Thanks to my friends and lab mates at the University of Denver for their support. Grateful to my defense committee for their feedback. Heartfelt thanks to my family for their unwavering support.

Table of Contents

1	Introduction	1
1.1	Ground-Aerial Robot Team Person-Following	3
1.2	Contributions	4
2	Literature Review	6
2.1	Visibility-based Pursuit-Evasion	6
2.2	Person Following	7
2.3	Heterogeneous Multi-robot Teaming	9
2.4	Multi-Agent Reinforcement Learning	10
3	Methods	12
3.1	Problem Definition	12
3.2	Motion Strategies	13
3.2.1	Forecasting Optimization of Robotic Way-point Assignment for Robust and Dynamic Person Following (FORWARD-PF) .	13
3.2.2	RL-Person Following using MADDPG	17
3.2.3	Closed-Loop Baseline	20
4	Experiments	23
4.1	Exploratory Experimentation	23
4.1.1	Ground Robot Target Localization	23
4.1.2	Spatially Tasking Micro-Aerial Platforms	28
4.1.3	Air-Ground Robot Team Target Following	29
4.2	Multi-robot Person Following Experiments	31
4.2.1	Experiment Design	31
4.2.1.1	Metrics	34
4.2.1.2	Simulator	35
4.2.2	Closed-Loop Method	37
4.2.3	FORWARD-PF Experiment Implementation	39
4.2.4	RL-Person Following Experiment Implementation	41
4.2.4.1	Model Training	43
4.2.4.2	Adapting Agent Actions	45
4.2.4.3	Adapting Observation of Obstacles	45
4.2.4.4	Adapting Observation of Positions	46
4.3	Real-world Validation	48
5	Results	51
5.1	Simulation Experiments	51
5.1.1	Success Rate	51

5.1.2	Arrival Time Delta	54
5.1.3	Time in View	55
5.1.4	Cumulative Distance Travelled	56
5.1.5	Max Distance Between Target and UGV	58
5.1.6	Observed Behaviour	60
6	Discussion	64
6.1	Limitations	65
6.1.1	Domain Adaptation	65
6.1.2	Development Time	65
6.2	Future Work	66
7	Conclusion	68
	Bibliography	69
A	Experiment Results Grouped by Method	74
A.1	Closed-Loop Method	74
A.2	FORWARD-PF	75
A.3	RL-Person Following	76

List of Tables

5.1	Summary of simulation experiment success rates grouped by path difficulty. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL)	51
5.2	Summary of simulation experiment, targets time in robot systems view. Time in view for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL). * Experiment configuration failed.	55
5.3	Summary of simulation experiment UGV cumulative distance traveled. UGV cumulative distance for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL).	56
5.4	Summary of simulation experiment UGV max distance from target. UGV max distance for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL).	58
A.1	Closed-loop Method. ET, elapsed time for UGV to reach target end position. $E\Delta$, difference in time between target and UGV reaching the end position. CD, cumulative distance UGV travelled to reach target end position. SR, success rate, TIV, percent of time target in view of the robot system. *Failures due to collision. **Failures due to losing sight of target.	74
A.2	FORWARD-PF Method. ET, elapsed time for UGV to reach target end position. $E\Delta$, difference in time between target and UGV reaching the end position. CD, cumulative distance UGV travelled to reach target end position. SR, success rate, TIV, percent of time target in view of the robot system. *Failures due to collision. **Failures due to losing sight of target.	75
A.3	RL-Person Following Method. ET, elapsed time for UGV to reach target end position. $E\Delta$, difference in time between target and UGV reaching the end position. CD, cumulative distance UGV travelled to reach target end position. SR, success rate, TIV, percent of time target in view of the robot system. *Failures due to collision. **Failures due to losing sight of target.	76

List of Figures

3.1	Algorithm for Forecasting Optimization of Robotic Waypoint Assignment for Robust and Dynamic Person Following (FORWARD-PF)	15
3.2	A diagram of the MADDPG algorithm showing centralized training and decentralized execution [1].	18
3.3	Closed-loop Ground Robot-only Method: Ground robot adjusts heading to face person.	21
4.1	This figure shows the person localization system for the TurtleBot iteration. The grey circle represents the area which the LIDAR covers and in which the distance of a detected person can be accurately measured. The orange cone shows the field of view of the forward facing camera that is used to detect a person and when they are outside of the radius of the LIDAR, used to approximate the distance.	24
4.2	TurtleBot equipped with a 2D LIDAR and a forward-facing monocular camera	25
4.3	The monocular person localization works by computing an estimate of the person's distance by analyzing the size of their bounding box in the camera frame. By extrapolating this data, the person's distance can be approximated as they move further from the camera. This graph shows the distance output of this method from an experiment where a person moved towards the robot in a stepwise manner.	26
4.4	This figure shows the occupancy grid built of the lab, the TF frame positions of the TurtleBot and person detected and localized, as well as the TurtleBots forward facing camera	27
4.5	Bitcraze Crazyflie 2.1 equipped with their first-party AI Deck with a forward-facing color camera. Also mounted with Optitrack markers for external position and orientation sensing in the lab space.	28
4.6	A TurtleBot acting as the ground-robot detects, tracks, and localizes a person in the environment (labeled <i>adversarial_agent</i> in RVIZ), then spatially tasks a Crazflie (micro-aerial platform) to the position of the person. The dotted red line shows the path the micro-aerial platform took. The yellow circle in the camera image highlights the micro-aerial platform.	29

4.7	Unity-based simulation environment containing an Clearpath Robotics Warthog [2], circled in red, a generic quad-rotor aerial robot, circled in blue, and a target (e.g., soldier model) circled in yellow.	32
4.8	Top-down view of the Unity environment where the experiments take place. The numbered blue circles show the experiment starting position for the neighboring paths. The numbered white circles indicate the end of a path and its number. The paths pattern illustrates the tortuosity/difficulty of the path.	33
4.9	This figure shows the Phoenix autonomy stack and Unity simulator used for the experimentation. Developed by the Army Research Lab, it encapsulates all the capabilities needed for ground autonomy, specifically in our case: person detection, tracking, localization, online simultaneous localization and mapping (SLAM), along with global and local planners.	36
4.10	This shows a two snapshots from the FORWARD-PF method being evaluated for the simulation experiment. Candidate goal scoring and goal selection for an observer (UAV). Candidate goal locations are colored on a green-red spectrum to display their score. Solid green is the maximum score out of the set while red is the minimum. The blue circle shows the goal location that was selected. A yellow arrow is overlaid on the image to show where the UAV is and where it is going. Also, a red arrow and target icon are overlaid to show the current position of the target and in which direction its headed.	40
4.11	This figure shows the Multi Particle Environment (MPE) used for training MADDPG models for the MARL experiment.	42
4.12	This shows a snapshot from the MARL method being evaluated for the simulation experiment. The left side is the follower (UGV), with the right being the observer (UAV). The purple, green, and orange sphere markers represent the closest obstacle in each of the 10 evenly divided angular sections around the robot. Those along with the current position of the other agents in the environment are passed to the trained model. The model then returns an action for all agents at 10 hz. For the UGV, the actions are averaged over a rolling 1 second period. This is converted to a vector which is represented by the green vector coming off the robot in the left image. For the UAV, because it can move holonomically, the actions are interpreted directly.	44

4.13	Real-world experiment side-by-side with RVIZ containing an autonomous ground vehicle, circled in red, an autonomous micro-aerial vehicle, circled in blue, and a target agent (e.g., a person) circled in yellow.	48
4.14	Metric results from real-world validation of FORWARD-PF method	50
5.1	Success rate is the percentage of successful runs per experiment path.	52
5.2	The Red circle and arrow indicate the current position and direction of movement of the UGV, which the Yellow circles illustrates this for the target. The Orange circle is the last known location of the target according to the UGV. This large difference in the UGV's knowledge of the target position vs the true position would result in a failure for this run as the possibility of the UGV reacquiring view of the target is minimal.	53
5.3	Time delta between target reaching end position and UGV arriving at the targets end position. The performance of the methods are grouped by path difficulty. Where no bar is displayed, that method was unable to complete the experiment for the given path.	54
5.4	Transparent light blue area visualizes the UAV's current area of visibility. Dark blue circle and area visualize goal location and the UAV's area of visibility at that goal location. Black dots represent the possible future locations of the target. Red circle visualizes a UGV goal location that minimizes UGV distance to the target. . . .	56
5.5	Percentage of time target is visible to robot team. The performance of the methods are grouped by path difficulty.	57
5.6	Cumulative distance travelled by UGV . The performance of the methods are grouped by path difficulty. The blue illustrates the total distance travelled by the person. Where no bar is displayed, that method was unable to complete the experiment for the given path. The FORWARD-PF and RL-Person Following methods resulted in cumulative distances of the UGV less than the person's path. This is due to the methods exploiting the robot team and by the UAV maintaining visibility of the person, the UGV navigates a	59
5.7	Maximum distance between UGV and target during length of the experiment. The performance of the methods are grouped by path difficulty. Where no bar is displayed, that method was unable to complete the experiment for the given path.	61

5.8	Target (Yellow), UAV (Blue), UGV (Red). Paths showing method that follows the path of the target either through design or the UGV having equal velocity to the target which would naturally lead it to take a similar path.	62
5.9	Target (Yellow), UAV (Blue), UGV (Red). Paths showing the UGV taking a shortcut to navigate to the target with a decreased path length. The UAV is seen still following a similar path to the target as to maintain visibility.	63

1 Introduction

As robots become increasingly prevalent in industrial and everyday settings, they frequently interact with other agents in shared environments. Whether in unstructured environments like construction sites handling complex tasks, disaster sites assisting in search and rescue operations, or remote areas conducting field research, these interactions are growing more common.

This increase in interactions adds complexity to navigating and operating in dynamic, shared spaces. Robots must perform their tasks efficiently while adapting to the unpredictable behaviors of humans. This requires sophisticated perception systems and advanced algorithms capable of real-time decision-making and obstacle avoidance. The challenge is particularly pronounced when human behavior is erratic or not fully aligned with robotic expectations, necessitating robust and flexible approaches to ensure seamless and safe interactions [3].

In more complex and constrained environments, tasks can become too difficult to run optimally with a single robot. Implementing heterogeneity in systems can improve the outcome of robotics tasks [4][5][6] .

For example, a builder on a construction site might lead a supply-carrying robot to the other side of the property, or a first responder might lead a robot carrying a disaster survivor to an exit. In these scenarios, heterogeneity can split the mission into sub-tasks, such as using a fast aerial robot to maintain localization of the person while a slow-moving ground robot navigates towards them.

The problem addressed in this research is developing methods for a heterogeneous multi-robot team, consisting of one aerial robot and one ground robot, to follow a person. Specifically, the objective is to develop and evaluate strategies for coordinating aerial and ground robots to achieve this goal, considering their distinct capabilities and limitations.

When a robot system is tasked with following a person, generalized to a target agent, the assumption that their behaviors will be compliant with a robotic system's perception and navigation limitations does not always hold. To this end, I assume there is no *a-priori* information available to precisely predict the target's motion, and that the target's trajectory is non-ergodic, i.e. even under identical circumstances their behavior may not be deterministic.

A robotic system will have a specific set of constraints, such as payload, sensing, maneuverability, and speed limitations. Micro-aerial robots are extremely payload-limited, which in turn limits sensing and computational power and can prevent fully independent autonomous operation. For this reason, my system pairs a micro-aerial robot or unmanned aerial vehicle (UAV) with a ground robot or unmanned ground vehicle (UGV) containing greater payload capacity for more powerful sensing and compute capabilities such as mapping, navigation, and person detection. This complements the simple UAV with the ability to localize itself and the person in the same space as the UGV.

Finally, the environment may present its own limitations. In this application, the UAV is assumed to maintain a fixed height above the ground. Where in some outdoor environments, a UAV may have the opportunity to gain altitude and direct its sensors downward to get a birds-eye view of the target in the environment, this research seeks to address the case where this might not be possible for a variety of reasons, such as indoor settings, intermittent

height obstacles such as tree canopies, or airspace restrictions. This research specifically explores the use of UGV and UAV teams within these constrained spaces.

Robot agents can be constrained by the nature of their desired tasks and environments they are working in. Other agents related to the robot mission can be motivated in a way to require the robots to ignore any assumptions they might have made. To address these specific velocity, height, and agent related constraints, I developed a heterogeneous multi-robot system, equipped with a predictive goal searching algorithm, named Forecasting Optimization of Robotic Way-point Assignment for Robust and Dynamic Person Following (FORWARD-PF), motion-strategy, capable of following a target agent through a height restricted environment. The FORWARD-PF method is compared with a multi-agent reinforcement learning (MARL) method and a UGV-only closed-loop method as a baseline.

1.1 Ground-Aerial Robot Team Person-Following

The robot systems objective is for the UGV to navigate to the position of the target, while the UAV assists. The UGV utilizes its perception capabilities to construct a motion model of the target, which facilitates anticipation of the target's future movements within a certain time horizon. This motion model enables the UGV to effectively follow or approach the target.

In parallel, the UAV's objective is to identify optimal positions in real-time to maximize its visibility of the target for the longest duration. For example, if the target is navigating around a corner, the aerial robot aims to position itself offset from the corner's outside, ensuring continuous visibility throughout the target's movement, rather than being obstructed as the target rounds the corner.

This research explores and compares three different methods. The first method, FORWARD-PF, the environment is systematically searched, and locations are scored based on the requirements of each robot. The locations with the highest scores are then assigned to the respective robots.

The second method employs a reinforcement learning framework, specifically a multi-agent reinforcement learning (MARL) approach. In this method, agents are trained using reward structures designed to promote emergent behaviors that align with the goals of each robot. This approach enables the robots to learn optimal strategies through interaction with the environment and feedback mechanisms.

Finally, a UGV-only approach is also evaluated as a baseline. This straightforward method involves directing the UGV in the current direction of the target and moving towards it.

By comparing these three approaches, this research aims to identify the strengths and weaknesses of each method and determine which approach yields superior performance in achieving the desired behaviors for both the ground and aerial robots.

1.2 Contributions

This work introduces a novel algorithm for identifying goal locations in a multi-robot person-following task, utilizing a two-robot system where one robot follows the target and the other observes from a vantage point, and incorporates a novel scoring mechanism to maintain line-of-sight with the moving target, enhancing real-time tracking, coordination, and performance in complex environments; another contribution is a method where a model trained for person-following, effectively employs transfer learning from 2D simulations

to 3D environments, and demonstrates the robustness of simplified models in real-world applications.

2 Literature Review

The research problem for the heterogeneous multi-robot system composed of an aerial platform and a ground platform can be viewed as the integration of two distinct problems. The aerial platform’s objective aligns with the visibility-based pursuit-evasion problem, where the goal is to maintain visibility of a moving target. Conversely, the ground robot’s objective corresponds to the person-following problem, which focuses on tracking and following a human target. This section will review past work on both visibility-based pursuit-evasion and person-following to contextualize the research and identify gaps that this work aims to address.

2.1 Visibility-based Pursuit-Evasion

The visibility-based pursuit-evasion problem was defined by Guibas et al. [7] and involves designing strategies for pursuers to maintain line-of-sight with evaders within a two-dimensional environment containing obstacles. This problem is essential in applications such as surveillance, search and rescue, and autonomous security, where continuous visual contact with a target is crucial. The challenge arises from the need for pursuers to navigate around obstacles that can obscure their view while anticipating and countering the evaders’ efforts to break line-of-sight.

This problem assumes robots observing an arbitrarily fast agent around an obstacle filled environment. The the objective is to observe a target, the robots speed is less important as line of sight is all that required. In tasks where

a robot is required to follow the target closely, methods for this problem will most likely fail to adequately achieve this new objective.

Current approaches to this problem revolve around rapidly-exploring random trees (RRT) [8] [9] and graph-based algorithms [10].

The problem differs when looking at the targets behavior being evasive versus this works problem that assumes a non-cooperative person whose behavior is totally agnostic to the actions of the robot system. Along with this, these solutions plan for searching for the evasive target versus following a target. While there are significant differences in the problem and solutions compared to this work, the problem is inspiration for the Aerial robots role in my robot team.

2.2 Person Following

Many solutions to person following, some involving robot teaming and/or heterogeneous systems, have been proposed. There is no one solution that optimally fits all person following applications because each application has its own unique set of objectives, context, and team dynamics. Existing solutions from literature show a vast variety of solutions for different use cases [11], these include general use single ground robots [12], general use single aerial robots [13], social ground single-robots [14], [15], social ground homogeneous multi-robot teams [16], and outdoor heterogeneous multi-robot teams [17] [18].

According to Islam et al. [11], person-following approaches can be classified into three main categories: position-based, image-based, and end-to-end learning-based.

- **Position-based approaches** rely on predicting the person’s motion and using that information for planning [19].

- **Image-based approaches** make decisions for person-following based on images captured by the robot’s camera [20].
- **End-to-end learning-based approaches** are commonly used in self-driving cars, where models are learned from input observations of the environment to produce actions that the robot then executes [21].

Orita and Fukao [22] developed a person following robot specifically for fire-fighters. They make the assumption that the target they are following will take the optimal path through what could be a cluttered environment. They argue this is because when a firefighter is searching through a building they must do so with great care. Due to this, in [22] they also assumed the target will almost never move at high-speed or change positions in an exaggerated manner. With these assumptions, in [22] they developed a robot system that follows the path of the target. While the robot’s task is the same, I assume no prior knowledge about the target other than its max velocity being bound between the aerial and ground robots. The work in [22] is capable of handling partial occlusion of the target, but was not shown to handle complete occlusion, while this work handles complete occlusion by relying on the aerial robot to maintain visibility of target even if the target is completely occluded to the ground robot.

From looking at previous work in person following, it’s clear that the constraints, assumptions, and robot systems vary greatly between use cases, which puts robustness and adaptability as the current main challenges in the field. Additionally, there is a gap in researching air-ground robot teams performing person following with specific constraints such as height, speed (walking at a reasonable speed), avoiding obstacles, taking smooth turns, and

handling complete occlusion using an aerial robot to maintain visibility, without any prior knowledge about the target other than maximum velocity bounds.

2.3 Heterogeneous Multi-robot Teaming

In related work, Reardon et al. focused on utilizing human gaze to spatially task autonomous micro UAVs [23]. Their method involved establishing coordinate frame alignment between a headset worn by the person and the UAVs. By deriving location information from the mobile eye tracker worn by the human, they could send corresponding goals to the UAVs. This study highlighted the challenges of coordinating heterogeneous robots and enabling effective interaction with humans in shared environments, including issues related to coordinate frame alignment and goal determination based on human interactions.

Another study explored the collaboration of a heterogeneous air-ground team in surveillance operations within a 3D environment [24]. Their research aimed to perform surveillance over a multi-story building, leveraging the advantages of using a combination of aerial and ground platforms. In a 3D space where verticality is crucial, an aerial platform can efficiently inspect upper floors or rooftops, while in confined spaces, a ground robot with superior perception capabilities may navigate more effectively. This approach recognized that no single platform excels in all situations; by combining them, a spectrum of surveillance scenarios could be efficiently covered.

The concept of leveraging the strengths of each platform is directly applicable to this research, which addresses the limitations faced by ground robots in keeping pace with humans and the computational constraints of aerial robots. This study aims to fill the research gap by developing novel strategies for the coordination of a heterogeneous robot team, comprising a slow-moving

but computationally capable ground robot and a fast but computationally constrained micro aerial robot. This approach seeks to enhance the robot team’s ability to follow a person in a shared environment, overcoming the challenges identified in previous works.

2.4 Multi-Agent Reinforcement Learning

This paper shows the use of an MARL method to learn pursuit-evasion tactics [1]. This work by Kouzeghar et al. inspired my use of Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and PettingZoo’s Multi-Agent Particle Environment (MPE) to train and then deploy on real robots. While their research explores multiple robots split by roles, working collaboratively to pursue an evader, my research removes the assumption of similarly mobile agents and significantly increases the complexity of the obstacles.

Simulation to real learning transfer for robots faces several challenges, including the significant gap between simulation and reality, making it hard to transfer models trained in one environment to another [25]. Additionally, discrepancies between simulated and real data often lead to failure when transferring policies trained in simulation to physical robots [26]. Moreover, simulators imperfectly model real-world dynamics, resulting in unsatisfactory performance of policies trained in simulation when deployed on actual hardware [27]. Determining the readiness of policies for transfer is crucial, as deploying under-trained policies can lead to unreliable and dangerous behaviors, while over-training can cause policies to over-fit to the simulator’s dynamics and appearance [28]. These challenges highlight the complexity of achieving successful simulation-to-reality transfer in robotic learning tasks.

Work by Kadian et al. [29] show an example of this where deteriorated performance when transferring a learned model from simulation to real-world

was attributed to AI agents exploiting simulator imperfections, such as abusing collision dynamics to take shortcuts, which do not work in the real world.

3 Methods

3.1 Problem Definition

One Differential Drive Robot (DDR), one holonomic robot, and one holonomic agent are modeled as oriented discs and move in an unexplored 3D environment with obstacles. Without loss of generality, the DDR is called the follower with the holonomic robot being its companion, named the observer. The holonomic agent is the target. The follower and observer are equipped with a sensor that detects the target if a clear line of sight connecting the robots centers does not intersect an obstacle. In this case, the sensors have an angular-bound.

The target is tasked with navigating agnostic to the pursuing teams actions to its end location, while the follower is tasked with navigating to the target. The target is assumed to have a maximum velocity greater than the follower, but less than the observer. With this, the observer is tasked with maintaining visibility of the target.

The goal of this work is to find motion-strategies for the pursuer team, follower and observer, to coordinate them for the person following task. The agents do not know the target's future movements in this work. Their motion strategies are computed on the basis of the target's current pose. In the case where the target is not visible to the pursuing team, their motion strategies are computed with the target's last known position.

3.2 Motion Strategies

3.2.1 Forecasting Optimization of Robotic Way-point Assignment for Robust and Dynamic Person Following (FORWARD-PF)

The FORWARD-PF approach aims to systematically search the environment and score candidate locations based on the objective of each robot. The locations with the highest utility are then assigned to the respective robots. This approach is outlined in Algorithm 3.1 is explained in this section.

In this formulation, time-horizon refers to the number of seconds a motion model of a target is predicted into the future. Position history refers to the number of seconds in the past over which position data is collected of a target and used to calculate a motion model. Motion prediction refers to the process of estimating the future motion or trajectory of an object or agent based on its current state and past observations.

The goal of motion prediction is to anticipate the future behavior of the target in order to make informed decisions. This approach uses a third degree polynomial to model the target's motion, $P(t)$. Using position data over the position history, a polynomial fitting algorithm is used to calculate the coefficients. The input is the time in seconds the position was observed and the output is a set of coordinates, (x, y). Points are then generated with this polynomial by getting coordinates between current time and the time-horizon in one second intervals. Each point is then used as the input for a normal Gaussian distribution with a σ of 0.9 generating 10 points each.

A sigma of 0.9 was chosen due to environmental factors regarding the distance between obstacles that the person would be navigating through. In tighter spaces, a higher sigma should be used because the likely paths are

more constrained, enhancing motion prediction accuracy. The opposite is true in more open areas where the person’s potential path is less constrained.

These points, F_p , are meant to characterize the potential motion of the target over the time-horizon.

The set of candidate goal locations, C , is generated as a grid around the target’s current position. Prior to the experiment execution, two values need to be defined. The value n which defines the size of the grid as the order value of a square matrix, meaning the grid would be of size $n \times n$, I will refer to this as C_{rows} . The other value is the distance in meters that each cell of the grid is spaced out by, $C_{spacing}$. For example, if $C_{spacing} = 1.0$, for all cells c in C , the center of cell c is 1 meter distance from all of its neighboring cells. From these two values, our set of candidate goal locations are defined. These values should be tweaked based on the environment in which this system is running. An obstacle-free outdoor space can afford a higher values for both since staying close to the target isn’t as necessary when compared to an indoor environment with lots of hallways where the threshold for an optimal goal location could be small. Once the set of candidate goal locations are generated, scoring begins.

There are two goal location scoring formulas, one for each agent of the pursuing team. Eq. 3.1, where goal locations are scored with the objective of the follower to achieve and maintain a close proximity distance to the target. Similarly, Eq. 3.2, where goal locations with the objective of the observer maintaining view of the target over some time-horizon, while also minimizing travel time to a new goal location.

Each candidate goal location is scored for the follower by the distance between the goal location c in C and the value of the predictive motion

Algorithm 1 FORWARD-PF Algorithm

```
1: Input: Time-horizon  $z$ , Observer optimal distance from target  $d_s$ 
2: while do
3:   Initialize candidate goal locations  $C$  around current position of target
4:   Generate motion model  $P(t)$  using past position observations of the target

5:   Generate motion prediction points  $F_p(z)$  using  $P(t)$  over time-horizon  $z$ 
6:   for each candidate goal location  $c$  in  $C$  do
7:     Score  $c$  using follower_score  $\rightarrow$  distance between  $c$  and average of  $F_p(z)$ 

8:     Score  $c$  using observer_score  $\rightarrow$  number of points in  $F_p(z)$  with line-of-
       sight to  $c$  minus  $d_s$  minus distance between  $c$  and average of  $F_p(z)$ 
9:   end for
10:  Select  $c_{f\_best}$  with the lowest follower_score in  $C$ 
11:  Select  $c_{o\_best}$  with the highest observer_score in  $C$ 
12:  Score current robot position
13:  if Current robot position is better than  $c_{f\_best}$  or  $c_{o\_best}$  respectively then
14:    Don't move the robot
15:  else
16:    Task  $c_{f\_best}$  to follower and  $c_{o\_best}$  to observer
17:  end if
18: end while
```

Figure 3.1: Algorithm for Forecasting Optimization of Robotic Way-point Assignment for Robust and Dynamic Person Following (FORWARD-PF)

polynomial at the time-horizon z :

$$follower_{score} \rightarrow ||c - avg(F_p(z))||_2 \quad (3.1)$$

Each candidate goal location's score is computed for the observer as utility minus cost. The utility is the line-of-sight score, which is the number of predicted motion points, F_p , visible to candidate goal location c . Function $los(c, F_p)$ returns the number of predicted motion points f in F_p that are visible to candidate goal location c in C . Costs includes the straight-line distance from the robot platform to the person, and the difference in distance between the preferred distance around the person to maintain, d_s , and the distance the candidate goal location is from the person.:

$$observer_{score} \rightarrow los(c, F_p) - |d_s - ||c - avg(F_p(z))||_2| \quad (3.2)$$

For the observer, the robots heading must direct towards the current position of the target. This orientation is calculated as the angle between the observer's current position and the position of the target:

$$\theta_{goal} = \arctan 2(T_y - U_y, T_x - U_x)$$

where (T_x, T_y) represents the coordinates of the current position of the target and (U_x, U_y) represents the coordinates of the current position of the companion.

Scoring is computed for both robots independently. The set of candidate goal locations, C , is copied for both the follower and observer, C_f and C_o respectively. For the ground platform C_f is iterated through and scored with

Eq. 3.1. Similarly, for the observer, C_o is iterated through where each candidate goal location is scored with Eq. 3.2

The next step is to select the target location and assign tasks. At each time step, the goal location in C_f with the lowest score c_{f_best} is selected for the follower, similarly the goal location in C_o with the highest score c_{o_best} is selected for the observer. If neither robot has been tasked a goal yet the current best candidate goal location from their respective candidate goal set is tasked to them. The observer is tasked c_{o_best} and the follower is tasked c_{f_best} . If a robot has already been tasked, they must first determine if they need re-tasking or if their current goal scores better than the more recently scored best candidate goal location. To do this, the current goal is re-scored with the most up-to-date map of the environment and position of the target. If a robots more recently scored candidate goal is greater than their re-scored current goal, that robot is tasked with the new candidate goal.

This process operates iteratively. When the target is in motion, the robots are continuously assigned goal locations that align with their objectives. Upon the target reaching its final destination and ceasing movement, the robots should maintain their goal positions. Given the stationary status of the target and the unchanging environment, the evaluation of potential goal locations will yield consistent scores at each time step.

3.2.2 RL-Person Following using MADDPG

For this approach, Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is utilized. MADDPG follows an actor-critic style and addresses the critical issue of non-stationarity in swarm learning by deploying a centralized critic mechanism that has access to both observations and actions of all agents [1] [30]. This centralized critic mechanism makes

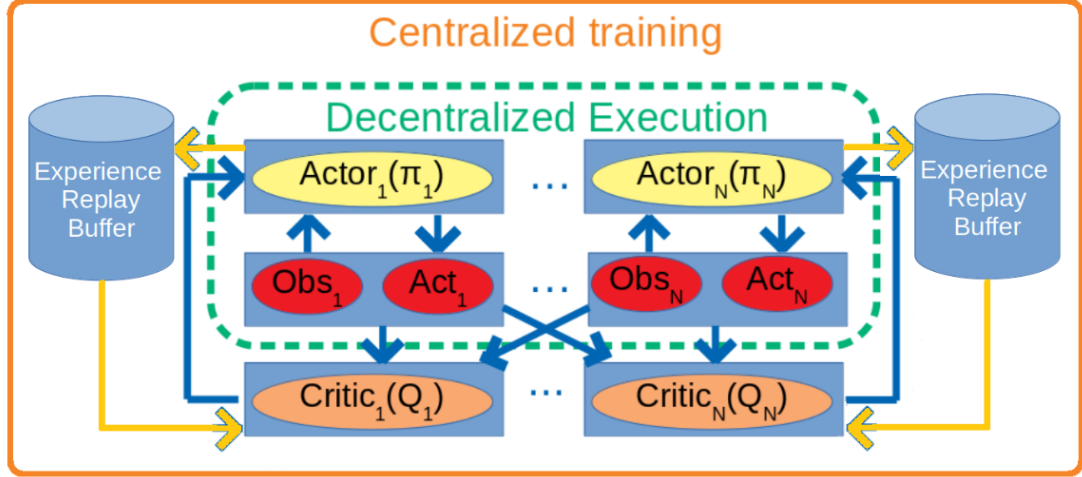


Figure 3.2: A diagram of the MADDPG algorithm showing centralized training and decentralized execution [1].

MADDPG a centralized training decentralized execution (CTDE) algorithm; see Ref. 3.2. The centralized critic evaluated the actions taken by each agent and provided rewards based on their performance. For the follower, the reward was based on its proximity to the target. For the observer, the reward was based on maintaining visual contact with the target and avoiding obstacles. CTDE is particularly suitable for multi-agent systems because it allows for decentralized execution with distributed control while also accommodating opposing teams of agents in competitive scenarios and enabling coordination within a team.

In our setup, I trained three agents simultaneously:

1. **Target:** This agent acts as the target. During training, the agent moves intelligently to evade the pursuing team rather than following a random or predictable path, which would result in less effective training for the pursuer team.

2. **Follower:** The follower's objective is to maintain a set proximity to the target. It receives rewards based on its proximity to the target, simulating a scenario where it must follow a moving target while accounting for its slower speed compared to the target.
3. **Observer:** The observer's goal is to maintain visual contact with the target while avoiding obstacles. The observer is rewarded for keeping the target in sight and staying within an optimal distance, prioritizing clear lines of sight without obstructions.

Each agent provides the following observations to the MADDPG algorithms:

- **Current Position:** The agent's location within the environment.
- **Current Velocity:** The agent's speed and direction of movement.
- **Relative Positions of Other Agents:** The locations of the other agents in the environment.
- **Velocities of Other Agents:** The speeds and directions of movement of the other agents.
- **Relative Position of Obstacles:** The locations of obstacles relative to the agent.

These observations enable each agent to make informed decisions based on the positions and movements of the other agents and obstacles within the environment.

The action space for each agent includes:

- **No Action**

- **Move Up**
- **Move Down**
- **Move Left**
- **Move Right**

This trained model is then deployed on the robot system where the observation space is computed in real time from the current environment, and the returned actions are processed by the robot team.

3.2.3 Closed-Loop Baseline

The closed-loop method works by maintaining the followers heading directed towards the target. This is a single-robot method designed to stand as a baseline motion strategy against the previous two which each utilize two robots on the pursuer team.

The process operates as follows:

1. **Line-of-Sight:** The target is assumed to be visible to the follower at the beginning of experiment.
2. **Angular Centering:** Calculate the angular offset of the target from follower's x-axis. Use this offset to rotate the follower so its heading directs towards the target
3. **Forward Motion:** Simultaneously, the follower is given a fixed forward velocity, ensuring it continually moves towards the target.

Let x_c be the horizontal coordinate of the centroid of the target's position relative to the follower's coordinate frame, and let x_{center} be the horizontal coordinate of the center of the follower's coordinate frame.

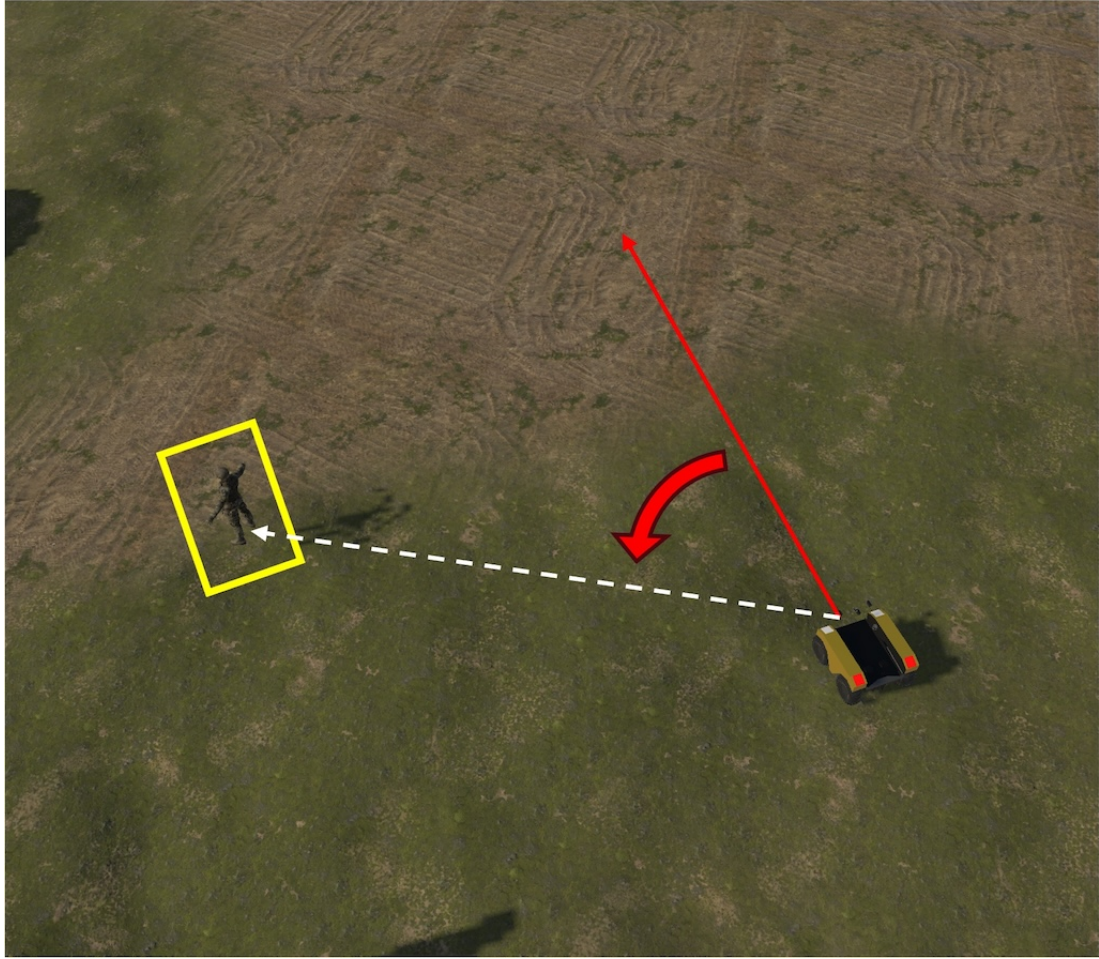


Figure 3.3: *Closed-loop Ground Robot-only Method: Ground robot adjusts heading to face person.*

The closed-loop method thus allows for the robot to adjust its heading and maintain a steady approach towards the target by consistently re-centering the follower so its heading is towards the target.

4 Experiments

4.1 Exploratory Experimentation

Exploratory experiments were conducted prior to the development of this current problem as defined in Section 3.1. These experiments inspired the defined problem and the various motion strategies that will be examined in Section 4.2.

4.1.1 Ground Robot Target Localization

This research initially stemmed from the idea of creating a robot capable of playing tag with a person. This required a robot that was capable of positioning itself and a person in the same coordinate frame, then making informed decisions to navigate towards the person. This was split into two objectives: the detection, tracking, and localization of the person in the environment, and subsequently, determining the robot's motion to approach and follow the person.

The TurtleBot, Fig. 4.2, used in this iteration of the project is equipped with a 2D LIDAR and a forward-facing monocular camera. The LIDAR has a range of 4 meters, beyond which it returns no values. When the person or agent being localized is within 4 meters of the robot, it utilizes the distances from the LIDAR in the direction corresponding to the detected bounding box of the person in the camera frame. This allows localizing the person by combining the angle derived from the bounding box in the camera frame with the distance from the LIDAR scan.

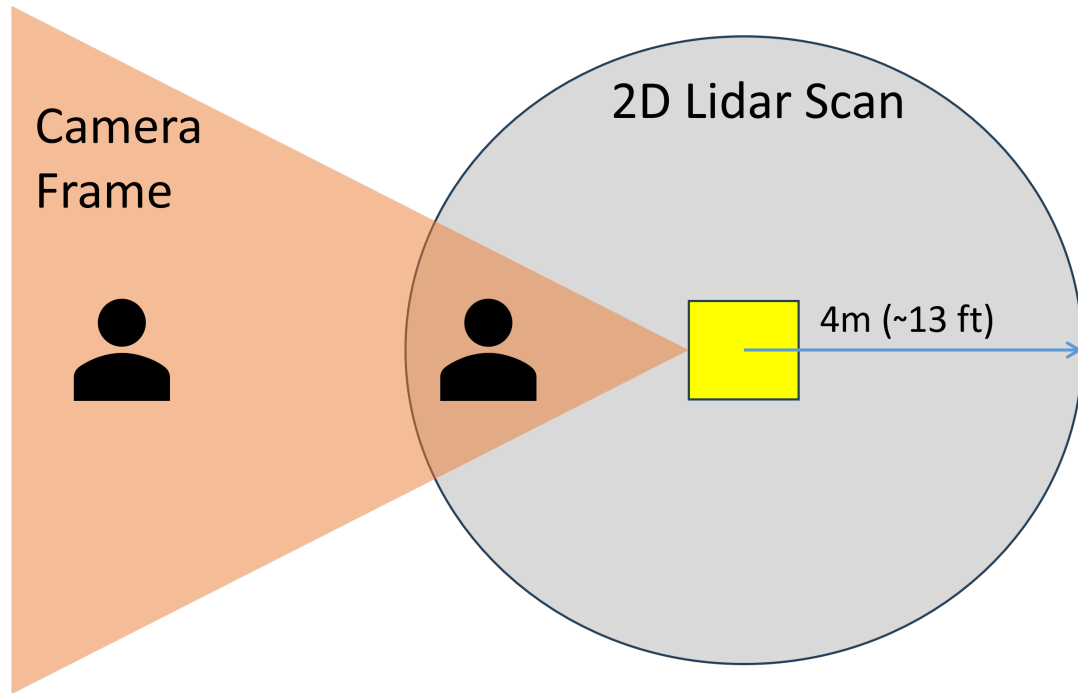


Figure 4.1: This figure shows the person localization system for the TurtleBot iteration. The grey circle represents the area which the LIDAR covers and in which the distance of a detected person can be accurately measured. The orange cone shows the field of view of the forward facing camera that is used to detect a person and when they are outside of the radius of the LIDAR, used to approximate the distance.

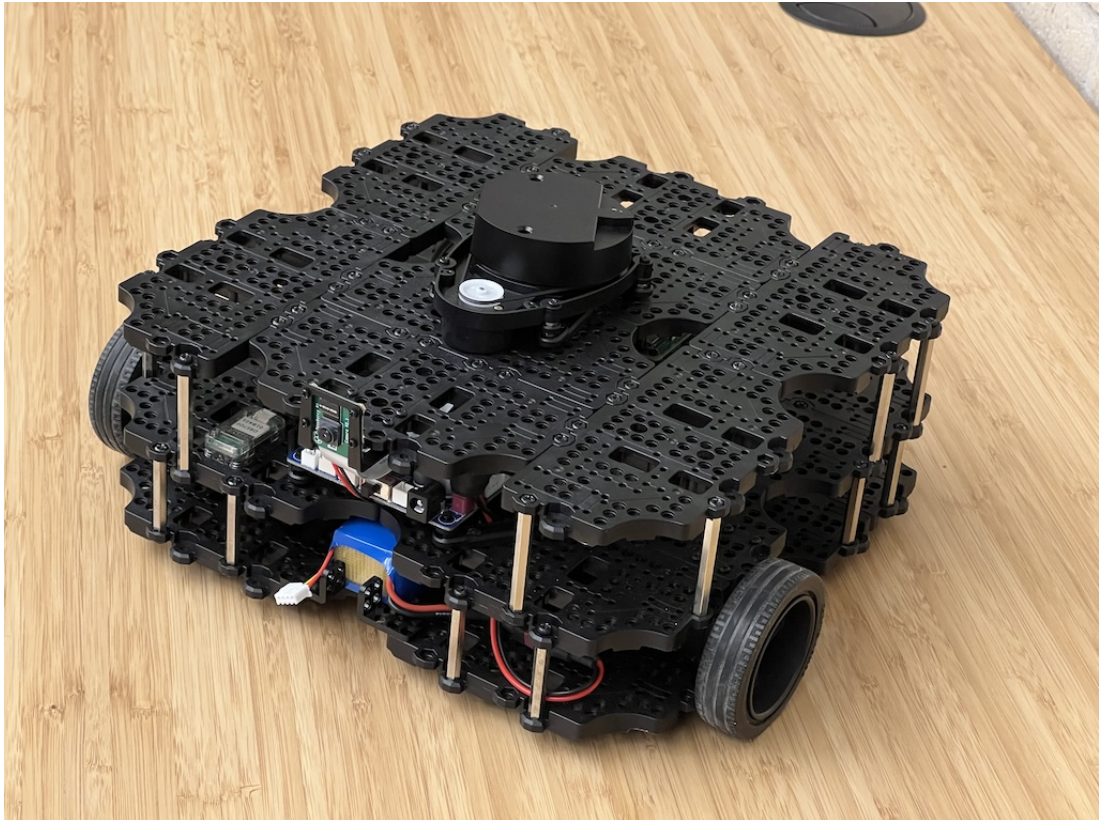


Figure 4.2: *TurtleBot equipped with a 2D LIDAR and a forward-facing monocular camera*

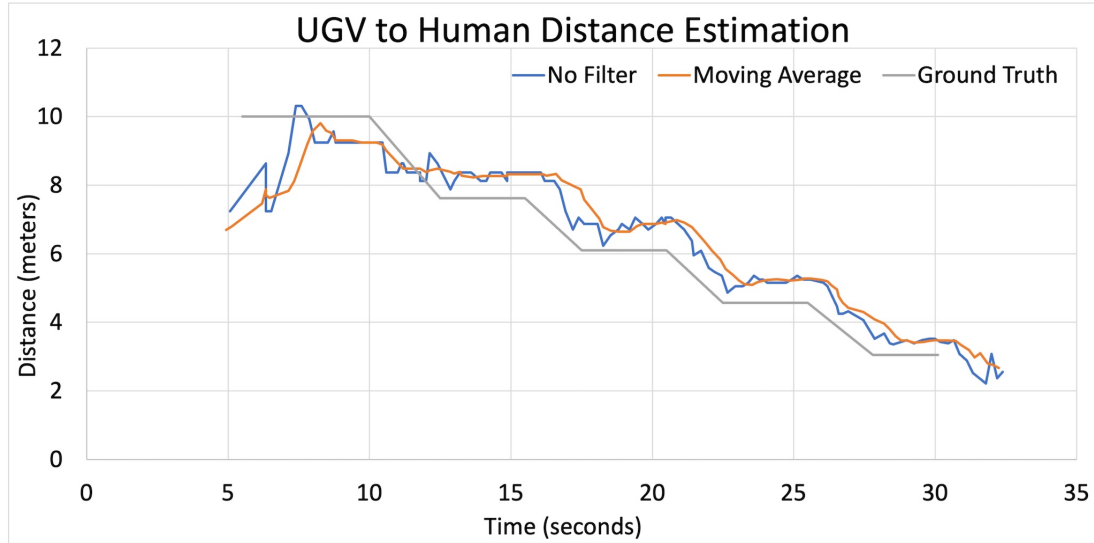


Figure 4.3: *The monocular person localization works by computing an estimate of the person's distance by analyzing the size of their bounding box in the camera frame. By extrapolating this data, the person's distance can be approximated as they move further from the camera. This graph shows the distance output of this method from an experiment where a person moved towards the robot in a stepwise manner.*

However, when the person is detected in the camera frame but is outside the 4-meter range of the LIDAR, an alternative method is needed. My solution involves approximating using only the single camera, see Ref. 4.1. Given that the extrinsic camera parameters were already being used to determine the angle between the person and the robot in the world frame, the person's distance can be estimated by analyzing the size of their bounding box in the camera frame. By extrapolating this data, it can approximate the person's distance as they move further from the camera. Results from this can be seen in Fig. 4.3.

This method relies on assumptions about the dimensions of the person being tracked, making it a backup method to the more reliable LIDAR readings.

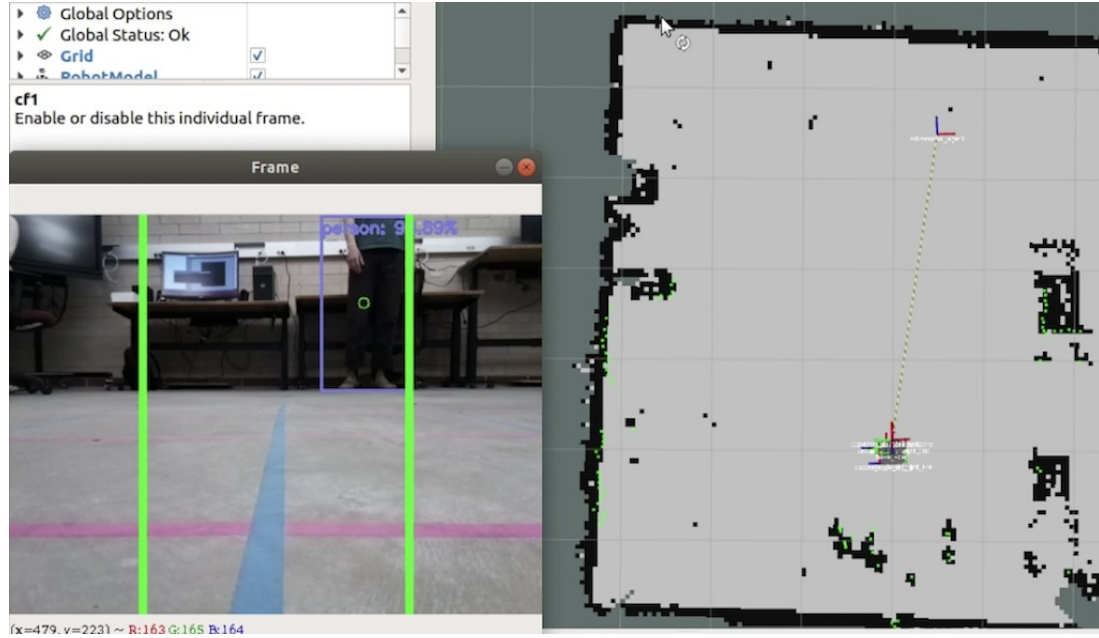


Figure 4.4: *This figure shows the occupancy grid built of the lab, the TF frame positions of the TurtleBot and person detected and localized, as well as the TurtleBots forward facing camera*

The subsequent step involved determining a motion strategy for the robot based on the person's position. Here, a simplistic approach was adopted, segmenting the video frame into left, middle, and right segments, see Ref. 4.4. The robot's movement was dictated by the relative position of the person within these segments, with the goal of keeping the person's centroid within the middle segment.

Although the localization information wasn't directly utilized for tag-playing behavior, it paved the way for the integration of a UAV into the research. Initially lacking onboard cameras, the UAV relied on positional information relayed by the ground robot. This introduction marked a transition into exploring multi-robot systems, particularly heterogeneous ones, which required addressing challenges like frame alignment to ensure seamless coordination between robots.



Figure 4.5: *Bitcraze Crazyflie 2.1 equipped with their first-party AI Deck with a forward-facing color camera. Also mounted with Optitrack markers for external position and orientation sensing in the lab space.*

4.1.2 Spatially Tasking Micro-Aerial Platforms

The decision to incorporate the micro aerial platform was inspired by research involving the spatial tasking of UAVs [23] [24], aligning with the concept of using one agent to guide another in a different coordinate frame. A Bitcraze Crazyflie was added to the robot team, Fig. 4.5. Aligning the coordinate frames between the ground robot and the UAV facilitated the spatial tasking process. Despite the initial lack of sensors on the UAV, learning to control it and integrate it into the system was a significant milestone.

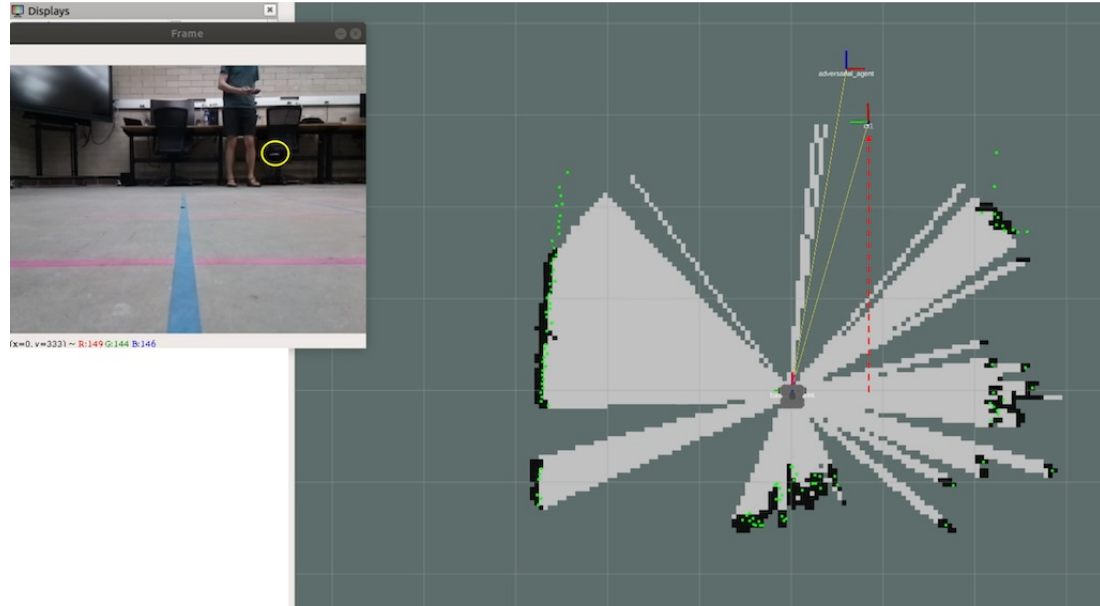


Figure 4.6: A TurtleBot acting as the ground-robot detects, tracks, and localizes a person in the environment (labeled `adversarial_agent` in RVIZ), then spatially tasks a Crazzfly (micro-aerial platform) to the position of the person. The dotted red line shows the path the micro-aerial platform took. The yellow circle in the camera image highlights the micro-aerial platform.

During this iteration of the research I was able to demonstrate a TurtleBot spatially tasking aCrazyflie to the position of a person dynamically detected and localized in the environment, see Fig. 4.6.

As the research progressed, plans were made to enhance the system by adding sensors to the aerial platform and transitioning to a more resource-rich ground robot platform.

4.1.3 Air-Ground Robot Team Target Following

In the next iteration of the research, the TurtleBot ground robot was replaced with a Clearpath Jackal differential drive ground robot, which provided significantly more resources. This platform was equipped with an RGBD camera and a 3D LIDAR, enhancing its perception capabilities. The

micro aerial platform remained a Bitcraze Crazyflie, but this time, a single gray-scale camera was utilized, adding perception capabilities to our aerial platform.

With these more capable platforms in place, the focus shifted towards the developing a more intelligent person-following solution, where the robot system endeavors to track and follow a person as they move through an environment. This posed a new set of challenges and prompted a literature review to explore existing solutions. While some previous work focused on utilizing top-down views from UAVs in outdoor environments, my aim was to develop a system suitable for indoor and GPS-denied environments.

Defining the problem more precisely involved identifying the three key agents: the ground robot, the aerial robot, and the person being tracked. Assumptions were made regarding the relative speeds of these agents, with the ground robot assumed to be slower than the person and the aerial robot faster. These assumptions were grounded in safety and the possibility of a ground robot carrying a significant payload. Constraints were also established, including the aerial platform's limitations in maneuverability and perception capabilities, as well as altitude restrictions imposed by the environment.

This assumption about agent speed plays a crucial role in shaping the behavior of the system. By designating the ground robot as slower and the aerial robot as faster, a scenario is created where the ground robot faces significant challenges in navigating to a moving target, particularly in obstacle-filled environments. This assumption underscores the complexity of the tracking task and highlights the need for efficient coordination between the ground and aerial robots to overcome obstacles and effectively track the person.

One significant constraint was the vertical limit imposed by the environment, whether it be the ceiling of a room or airspace restrictions outdoors. This constraint dictated that the aerial robot maintain an altitude close to the other agents in the environment, typically within a few meters from the ground. While some flexibility existed in altitude, there were practical limits to how high or low the aerial robot could operate.

4.2 Multi-robot Person Following Experiments

Those exploratory experiments led to the definition of the problem and development of the FORWARD-PF approach, Section 3.2.1, the multi-agent reinforcement method, Section 3.2.2, and the UGV-only closed-loop method, Section 3.2.3.

This next section explains the experimental design for the simulated experiments that were used to evaluate the three motion strategies. I also go in-depth on the technical aspects of evaluating these methods on simulated hardware as well as a real-world hardware validation for the FORWARD-PF method.

4.2.1 Experiment Design

The simulated experiment occurs in an environment rendered in a Unity-based ROS simulator, discussed in Section 4.2.1.2. Fig. 4.7 displays the agents and environment.

The experiment commences with both the ground robot and aerial robot spawning in the environment in close proximity to each other. Meanwhile, the target spawns out of sight of both robots. Upon initiation, the target begins movement, following a predefined experiment path. The robots remain stationary until the target enters their field of view, marking the official start of the experiment.



Figure 4.7: *Unity-based simulation environment containing an Clearpath Robotics Warthog [2], circled in red, a generic quad-rotor aerial robot, circled in blue, and a target (e.g., soldier model) circled in yellow.*

To assess the performance of our multi-agent system in following a target through paths of varying difficulty, an experiment was designed that gradually increases the complexity of the paths traversed by the target agent, see Fig. 4.8.

For the experimental setup, the environment was prepared with obstacles to simulate three levels of path complexity: easy, medium, and hard. For all of these paths, the robots start in close together with the same heading, while the target starts close by but out of the robot(s) camera view. The experiment begins when the target enters the UGV’s camera frame for the first time and ends either when the UGV reaches the target once it is stationary at its end location, or if the robot system fails which is defined as the robot(s) colliding with an object or the robot(s) losing visual of the target while heading in a direction it won’t be able to find the target again.

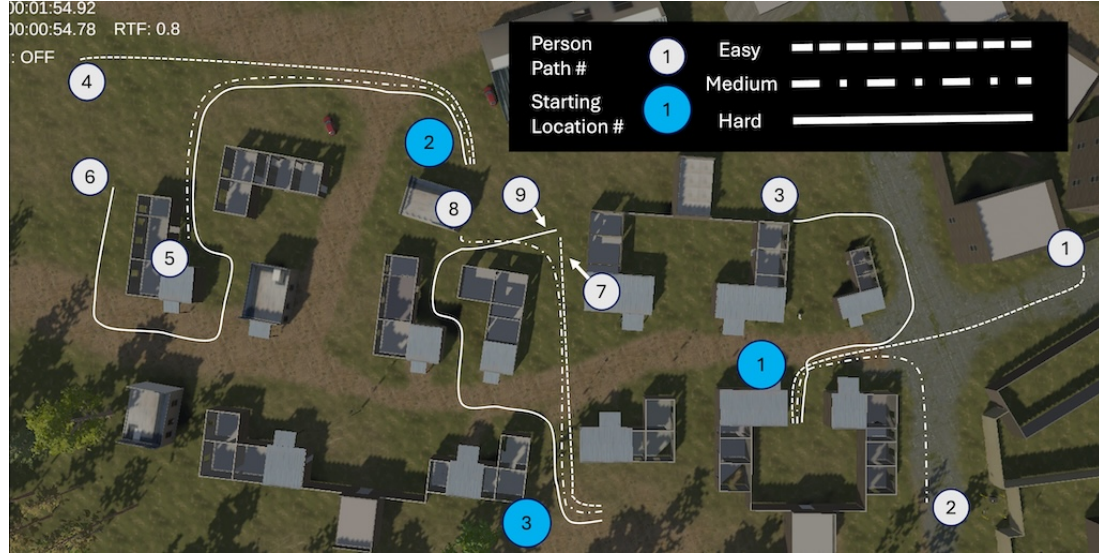


Figure 4.8: Top-down view of the Unity environment where the experiments take place. The numbered blue circles show the experiment starting position for the neighboring paths. The numbered white circles indicate the end of a path and its number. The paths pattern illustrates the tortuosity/difficulty of the path.

The difficulty of a path is quantitatively measured by its tortuosity, which is measured as the ratio of the paths length to the distance between the start and end positions. Easy paths average a tortuosity of ~ 1.0 , medium is ~ 1.3 , and hard is ~ 2.0 .

The easy path consists of predominantly straight trajectories with minimal turns and obstacles..

As the complexity increases, the medium path introduces several turns around corners and additional obstacles, challenging the system's ability to maintain visual contact with the target during maneuvers.

The hard path presents a high tortuosity trajectory with frequent and sharp turns, requiring navigation between closely spaced obstacles and testing the system's robustness in complex environments.

During the experiment execution, the target traverses the predefined paths while the closed-loop, FORWARD-PF, and multi-agent reinforcement learning (MARL) approaches are evaluated separately for each path in several different environments. So with 3 difficulties in 3 different locations, 9 total experiment paths and with 3 different approaches, there are 27 experiment configurations. Evaluating each configuration over 3 runs leads to 81 total responses.

4.2.1.1 Metrics

The following metrics were collected to evaluate the success, efficiency, and robustness of each method.

Success Rate: The metric "success rate" is defined as the percentage of experiments in which the UGV navigated to the target's stationary end position after the target had arrived, under each method and path configuration. A failure is defined by one of two criteria: the robot(s) colliding with an object, or the robot(s) losing visual contact with the target while heading in a direction that precludes reacquiring the target. The latter is determined by the experimenter. There is no time limit set on the duration of an experiment. If the experimenter assesses that, given the current state of the observation space, the robot is unlikely to reacquire visibility of the target, the experiment is terminated, and that run is marked as a failure.

Time in View: Inspired by [31], which formulated the metric Following Success which represents the amount of the experiment the system was able to follow the target, I define the metric, Time in View, which shows the percentage of the experiment the target was visible to the robot system. If the either the UGV or UAV have a view of the target, the UGV will be able to navigate there, making this a good metric for determining the performance of the methods ability to maintain visibility of the moving target.

Cumulative Distance: The UGV cumulative distance is a metric representing the total length of the path the UGV traveled during the experiment, from start to reaching the target’s end position. A lower value during a successful run may indicate more efficient navigation and decision-making.

Maximum Distance: The UGV maximum distance is a metric representing the farthest distance between the UGV and the target during the experiment. A higher value may indicate less efficient navigation and decision-making by the method.

I expect the closed-loop method to encounter difficulties in maintaining visual contact around turns and obstacles, particularly on the medium and hard paths.

In contrast, the FORWARD-PF and MARL approaches are anticipated to demonstrate improved performance by leveraging learned policies to adapt to the complexities of the environment and coordinate between ground and aerial robots.

4.2.1.2 Simulator

The experimentation was conducted using Phoenix Ground Autonomy Stack and Simulator [32]. Phoenix is developed and maintained by Army Research Lab along with a number of collaborating Universities. The Ground Autonomy stack is designed to be deployed on platforms such as the Clearpath Robotics Warthog, Husky, and Jackal. Along with the ability to deploy the autonomy stack on hardware, a Unity-based simulator can be used with custom environments and Clearpath Robotics modeled robots [33]. This allows for realistic simulated execution of robots and software in custom Unity maps that are modeled on real locations.

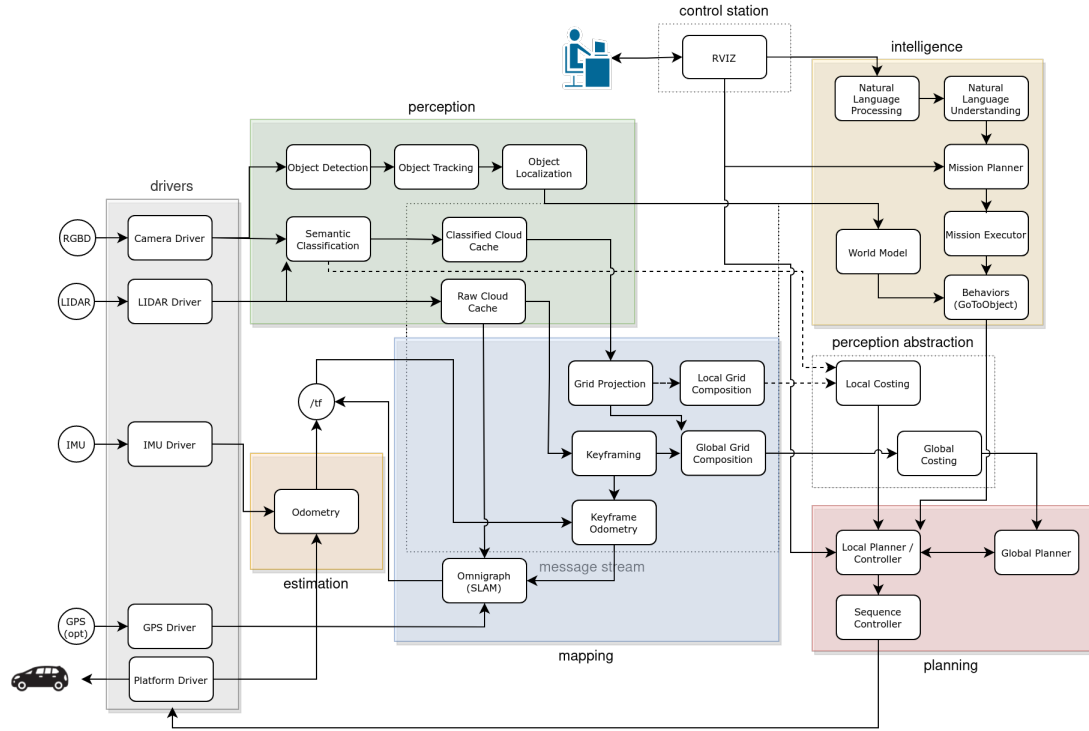


Figure 4.9: This figure shows the Phoenix autonomy stack and Unity simulator used for the experimentation. Developed by the Army Research Lab, it encapsulates all the capabilities needed for ground autonomy, specifically in our case: person detection, tracking, localization, online simultaneous localization and mapping (SLAM), along with global and local planners.

For this experimentation, a custom Unity map is used a location for testing these methods, see Fig. 4.8. In this environment, a simulated Clearpath Robotics Warthog and generic quad-rotor aerial robot are deployed as they would be in a real-world setting. A simulated person capable of being detected by the simulated robots cameras and of following predefined paths is present in the environment during the experiments.

Each robot is equipped with a simulated forward-facing camera used for object detection, tracking, and localization. In the simulator, when an object (i.e. person) is in the field of view and 20 meter range of a camera, the position of the object relative to the camera is returned and used to localize the object in the robots shared map for the following experiments.

The reasoning for using this autonomy stack and simulator is a familiarity with developing with it as well as its realism is adequate for simulation-based experimentation.

4.2.2 Closed-Loop Method

In the simulation experiment, a closed-loop method is employed as a baseline comparison utilizing the simulated forward-facing camera and object detection within the Unity simulator. This enables extracting a bounding box of the target in the camera frame, which is then used as feedback for the closed-loop method. The method calculates the required angular velocity to be commanded to the robot, ensuring the target remains centered within the camera frame. This procedure is repeated across all nine paths.

The camera captures images that are processed through an object detector, which identifies and locates the target. The object detector outputs a bounding box around the detected target in the camera frame.

The primary goal is to maintain the centroid of the bounding box at the center of the camera frame along the horizontal axis. This is achieved by calculating the offset of the centroid from the center of the frame and using proportional control to determine the necessary angular velocity command for the robot. This adjustment aligns the robot's orientation to keep the target in the middle of the camera frame.

Simultaneously, the robot is given a fixed forward velocity, ensuring it continually moves towards the target. This straightforward approach enables the robot to follow the person based on real-time visual feedback.

Let x_c be the horizontal coordinate of the centroid of the bounding box in the camera frame, and let x_{center} be the horizontal coordinate of the center of the camera frame. The offset Δx is given by:

$$\Delta x = x_c - x_{center}$$

The angular velocity ω needed to re-center the target can be computed using proportional control:

$$\omega = -k_p \Delta x$$

where k_p is the proportional gain.

The forward velocity v is set to a constant value v_f :

$$v = v_f$$

Combining these, the velocity commands for the robot are:

$$v = v_f$$

$$\omega = -k_p \Delta x$$

The closed-loop method thus allows for the robot to adjust its heading and maintain a steady approach towards the target by consistently re-centering the bounding box of the detected person in the camera frame.

4.2.3 FORWARD-PF Experiment Implementation

As with all the experiments, the robots operate within the simulation environment equipped with a full autonomy stack capable of generating a map and path planning.

Using the simulated camera, the target’s position within the camera frame is identified and projected onto the map. Subsequently, motion prediction is employed by integrating the target’s position and velocity into an extended Kalman filter to mitigate sensor noise. The filtered position data is then utilized to fit a third-degree polynomial to the position history buffer, facilitating the projection of the target’s future position. This projection is determined based on the target’s distance from the robot, governed by a fixed proportion coefficient.

The set of candidate goal locations are generated as a grid around the target’s current position. Each candidate goal location undergoes scoring, with the ground robot’s scoring based on proximity to the target, see Fig. 4.10. This proximity is assessed by a buffer zone of five meters around the target, simplifying the goal to reaching within this buffer zone. For the aerial robot, scoring considers both its line of sight to the target’s current and predicted motion. To achieve this, Gaussian spread is added to the predicted motion vector, generating points along the trajectory for raycasting against the occupancy grid.

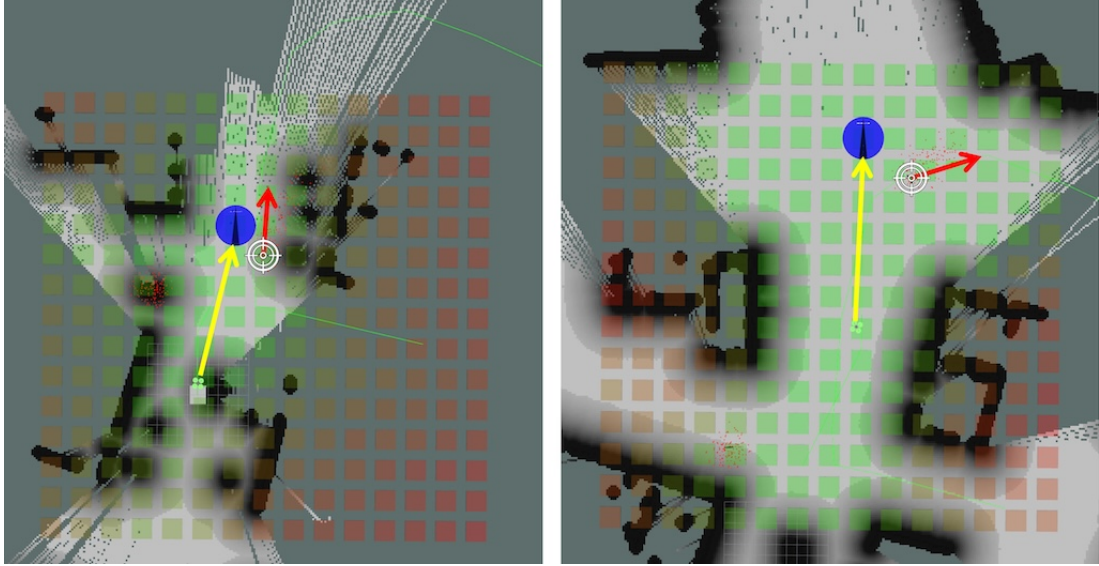


Figure 4.10: This shows a two snapshots from the FORWARD-PF method being evaluated for the simulation experiment. Candidate goal scoring and goal selection for an observer (UAV). Candidate goal locations are colored on a green-red spectrum to display their score. Solid green is the maximum score out of the set while red is the minimum. The blue circle shows the goal location that was selected. A yellow arrow is overlaid on the image to show where the UAV is and where it is going. Also, a red arrow and target icon are overlaid to show the current position of the target and in which direction its headed.

During scoring, each candidate goal location is evaluated for its feasibility. A cost is assigned based on the presence of obstacles encountered on a line drawn on the robots occupancy grid between the targets predicted motion vector and a candidate goal location. Additionally, the distance from the ideal ten-meter distance between the aerial platform and the target is factored into the scoring mechanism. The candidate goal locations are scored accordingly, considering the absence of obstacles and the proximity to the ideal distance. The location with the highest score is selected for navigation.

Furthermore, to ensure optimal tracking, the aerial platform maintains its heading towards the tracking target while navigating to the chosen location. This ensures continuous alignment of the aerial platform’s camera with the target, enhancing tracking accuracy.

4.2.4 RL-Person Following Experiment Implementation

For the multi-agent reinforcement learning approach in this experiment, PettingZoo was used, a Python library that facilitates multi-agent reinforcement learning (MARL) by providing a simple interface for interacting with OpenAI’s Gym environments. PettingZoo was combined with the Multi Particle Environment (MPE) for training. The MPE is a two-dimensional space populated with particles of varying sizes and functions, as illustrated in the Fig. 4.11.

In the environment, gray particles represent landmarks, which are obstacles that agents cannot pass through but can detect by proximity or collision. The red particle, a medium-sized entity, serves as the target agent. This agent, which the robot system aims to follow, is also trained to simulate a target attempting to evade the robots. The smaller blue particle represents the observer, analogous to an aerial robot. It is faster than the red target agent,

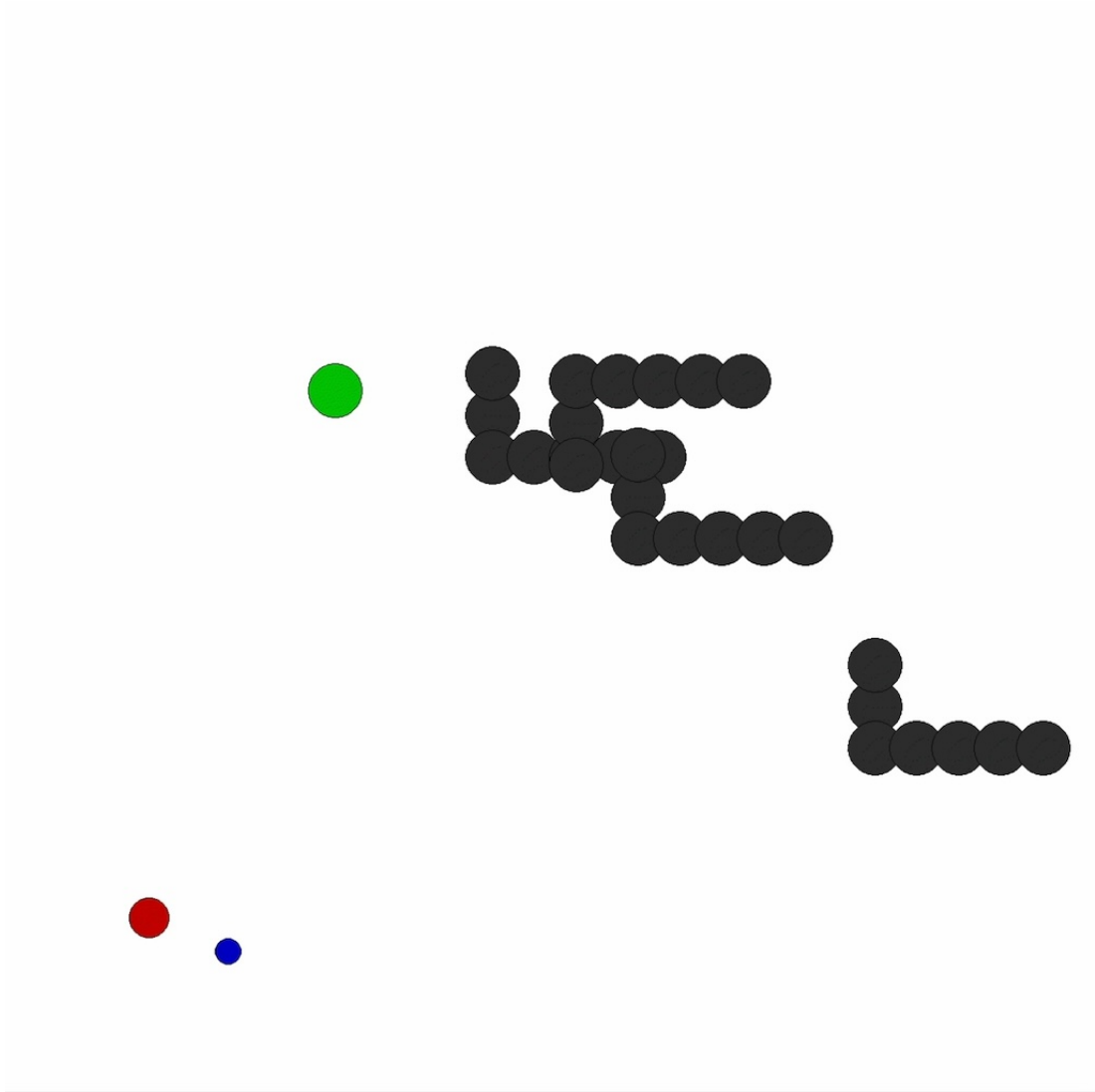


Figure 4.11: *This figure shows the Multi Particle Environment (MPE) used for training MADDPG models for the MARL experiment.*

enabling it to catch up in a straight line, reflecting the speed advantage of aerial platforms.

The larger, slower green particle represents the follower agent, akin to our ground robot. This agent, being the slowest, cannot catch up to the target agent in a straight line, mimicking the slower speed of ground robots compared to humans or aerial robots.

4.2.4.1 Model Training

The training setup leverages the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm, a centralized training decentralized execution (CTDE) approach. During training, a centralized critic, which has access to the observations and actions of all agents, rewards each agent based on their roles. However, the actors operate decentrally, with individual observations guiding their actions.

In this experiment, the follower agent was rewarded for minimizing its distance to the target agent, despite its slower speed. The observer agent was rewarded for maintaining a line of sight to the target agent and staying within an optimal distance, avoiding obstacles in the process. The target agent was trained to move in ways that challenge the follower and observer agents, simulating a realistic target evasion scenario.

Each agent’s observations include their current position, velocity, the positions and velocities of other agents, and the relative positions of obstacles in the environment. Actions for each agent are discrete and include moving up, down, left, or right. The agents’ performances were evaluated based on their ability to follow the target agent across varying levels of path complexity, designed to test the robustness of the trained policies in increasingly challenging environments. Adapting the 2D simplistically trained model to work

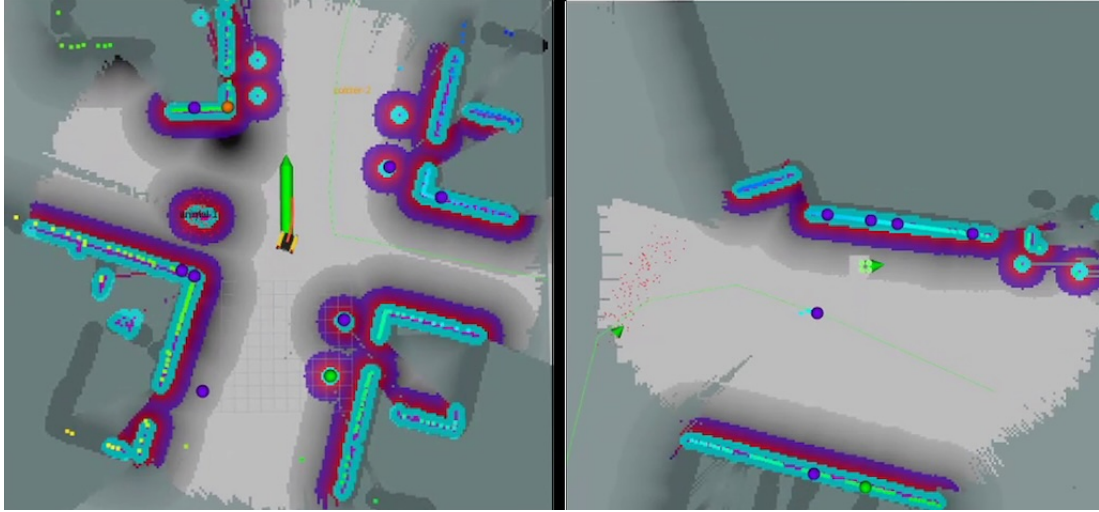


Figure 4.12: This shows a snapshot from the MARL method being evaluated for the simulation experiment. The left side is the follower (UGV), with the right being the observer (UAV). The purple, green, and orange sphere markers represent the closest obstacle in each of the 10 evenly divided angular sections around the robot. Those along with the current position of the other agents in the environment are passed to the trained model. The model then returns an action for all agents at 10 hz. For the UGV, the actions are averaged over a rolling 1 second period. This is converted to a vector which is represented by the green vector coming off the robot in the left image. For the UAV, because it can move holonomically, the actions are interpreted directly.

in a complex 3D environment led to converting the robots observations from the more complex world to one the model would comprehend, see Fig. 4.12.

The next step involves transitioning from the training environment in the Multi-Agent Particle Environment (MPE) to deploying the trained models on real robots in a 3D space. Given the discrepancy between the simulated action space, which allows for discrete up, down, left, and right movements, and the actual control requirements of our ground robot, a Clearpath Warthog with differential drive, there is a need to adapt the action execution appropriately.

4.2.4.2 Adapting Agent Actions

To bridge this gap, a method to convert the discrete actions into a continuous motion command suitable for a differential drive vehicle was implemented. This was achieved by averaging the actions received over the last one to two seconds. For example, if the model commands "move right" and "move up" alternately, interpreting this as a command to move in the "up-right" direction in the world frame. By maintaining the world frame as the parent frame for all agent locations, consistent directionality (up, down, left, right) regardless of the robot's orientation can be ensured.

Additionally, small obstacles in the 3D environment were being inconsistently observed by the LIDAR which resulted in a different than true observation state to be provided during model deployment, this resulted in regular collisions between these small obstacles and the robot. To remedy this, for the UGV the actions are averaged over the past 1 second and the vector projected to 5 meters is given to a local planner to enable collision-avoidance.

4.2.4.3 Adapting Observation of Obstacles

During the adaptation, challenges related to the observation space were encountered. The original environment expected the relative positions of all landmarks or obstacles, which presented two issues. First, if an environment with a fixed number of obstacles (e.g., 10) was trained in, deploying the model in an environment with a different number of obstacles (e.g., 11 or 9) could cause a mismatch in the observation space. Second, increasing the number of obstacles would significantly increase the training time due to the larger observation space.

To address these issues, several workarounds were experimented with. Initially, the observation was limited to the 10 closest obstacles, but this approach proved problematic. The list of closest obstacles changed dynamically as the agent moved, causing inconsistencies in the observation space and potentially increasing training time.

Ultimately, a more effective solution was developed by dividing the 360° space around the agent into 10 segments. Within each segment, the closest obstacle was identified and a state associated with the distance to that obstacle was created. This segmentation allowed us to maintain a consistent observation space. 10 Segements was decided because, I found that as the number of obstacles in the observation space increased, the training time increased exponentially. Therefore, 10 segments seemed like enough information for the agent to learn obstacle avoidance without significantly hampering training time.

Distances were categorized into states, such as one through five, which corresponded to varying levels of proximity to obstacles. For example, the aerial platform was designed to maintain a state of three relative to the target agent, ensuring that it did not approach obstacles closer to state two.

4.2.4.4 Adapting Observation of Positions

The trained model was initially trained in a controlled environment, specifically a 2 x 2 unit space with randomized obstacle positions. The primary objective was to adapt this pre-trained model for deployment in varying environments rather than training a new model for each specific setting, which would be resource and time intensive. To achieve this, it is crucial that the observations fed to the model closely resemble those encountered during training.

During experiment, the robot operates within a substantially larger $\sim 100 \times \sim 100$ meter space. The model expects the positional data for the X and Y coordinates to normalize between -1 and 1 , and the relative values between agents to also fall within this range, consistent with the maximum values encountered in a 2×2 unit box. To adapt the robot’s positional data to this format, the X and Y values are scaled down to fit into the 2×2 unit square, and an offset is applied based on the starting location of the experiment. This scaled and offset data forms the observation space, which is then used by the model.

While this method generally provides observations compatible with the trained model, it requires fine-tuning. Minor adjustments of $\sim 10\%$ to the initial scaled values were necessary increase similarity of model behaviour between simulation environments. It was observed that the density and proximity of obstacles in the environment significantly influenced the model’s performance. In highly cluttered environments, it was beneficial to reduce the scaling of positional values. Excessive scaling down made the model perceive objects as being closer than they were, thus diminishing the perceived consequences of actions. For example, scaling the environment by a factor of 10 would result in the model perceiving a unit movement as only one-tenth of its actual displacement.

This approach of adapting a generalized model to different environments presents certain limitations. Specifically, there is a threshold for the maximum area size and obstacle density beyond which the model’s performance degrades. Future research should involve a comparative analysis between optimizing a generalized model for various environments and developing specialized models for each environment.



Figure 4.13: *Real-world experiment side-by-side with RVIZ containing an autonomous ground vehicle, circled in red, an autonomous micro-aerial vehicle, circled in blue, and a target agent (e.g., a person) circled in yellow.*

Given the current constraints of machine learning techniques, particularly in multi-agent reinforcement learning, the time required to train a new model for each unexplored environment is impractical [34]. Thus, at present, using a generalized model and adapting it to specific environments remains the more feasible strategy.

4.3 Real-world Validation

Real-world validation was performed to demonstrate the practical functionality of these methods on actual hardware. A Crazyflie 2.1 from Bitcraze [35] was used as the UAV for real-world experimentation. Originally equipped with a 320x320 grayscale resolution forward facing camera for object recognition and localization, and a downward facing camera for optical flow localization. The ground robot is a differential drive robot from Clearpath Robotics, known as Jackal [36]. It is equipped with the Army Research Lab’s ground autonomy stack, Phoenix (see Fig. 4.9). Additionally, the robot features a Velodyne VLP-16 3D LiDAR [37] for SLAM (Simultaneous Localization and Mapping), and a forward-facing Intel RealSense camera used for object

detection, tracking, and localization. This configuration enables the Jackal to navigate and interact with its environment efficiently during real-world experimentation.

However, due to the limitations of the current equipment, only a validation was performed rather than a full experiment, requiring numerous assumptions for successful operation.

The primary hardware challenge was the aerial platform. The Bitcraze Crazyflies, while cost-efficient and effective in demonstrating swarm capabilities with an external sensing system such as OptiTrack, faced significant problems. Extensive efforts to enhance their sensing, localization, and navigation capabilities were unsuccessful. The localization drift made simultaneous rotation and translation inaccurate, adversely affecting navigation. Additionally, the 320x320 resolution grayscale forward-facing camera was unreliable for object detection.

To mitigate these deficiencies, the OptiTrack external sensing system was used for localization, and the ground-truth position of the target agent was utilized. However, using OptiTrack confined experimentation to the lab space, limiting the number of possible configurations and the insights derived from the results.

In contrast, the UGV was capable of running a full-autonomy stack with robust mapping, navigation, and object detection. Future work will involve replacing the aerial platform with a more capable one.

Given these constraints, a real-world hardware validation was performed. The person, UGV, and UAV were all equipped with Optitrack markers for ground-truth positioning. Due to the unreliability for object detection with the Crazyflie, the robot team receives an updated position of the person if there is

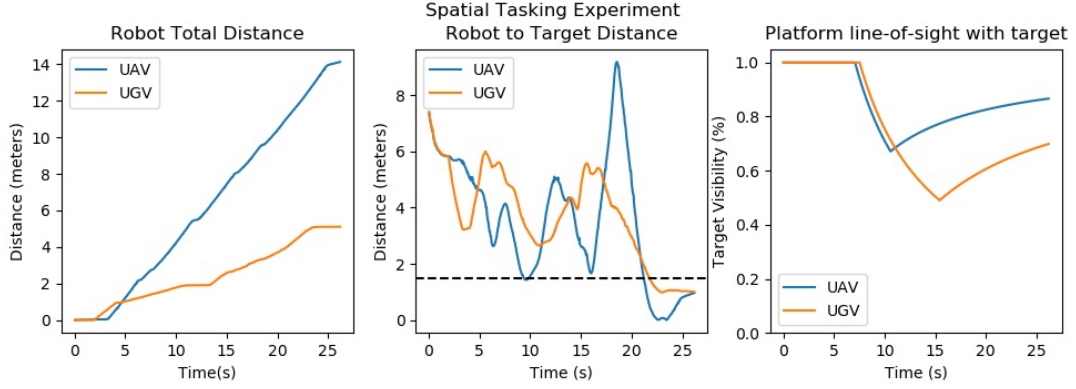


Figure 4.14: Metric results from real-world validation of FORWARD-PF method

line of sight as determined by raycasting over the occupancy grid being built by the Jackal.

Given these constraints, only the FORWARD-PF method was demonstrated on real hardware, see Fig. 4.13, as its robustness ensured safe operation in a space with people. Metric results can were also collected, see Fig. 4.14. The RL-Person Following method was excluded because of its unpredictability. The closed-loop method was not demonstrated here, as it had already been tested during exploratory experimentation, see Fig. 4.4.

Given the significant issues that needed to be addressed and the deviations from the original experiment, demonstrating the approaches on real hardware was deemed reasonable. However, due to the numerous assumptions required, drawing conclusions from the real-world results would be unreasonable.

5 Results

5.1 Simulation Experiments

This section looks at the results from the 108 runs collect during the simulation experiment. Metrics analysed are success rate, arrival time delta, time in view, cumulative distance travelled, and max distance between robot and target. Observed results look at emergent behaviours from the methods.

5.1.1 Success Rate

	Easy	Medium	Hard
CL	1.0	0.0	0.0
PF	1.0	1.0	0.89
RL	1.0	1.0	0.33

Table 5.1: Summary of simulation experiment success rates grouped by path difficulty. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL)

Table 5.1 summarizes the success rate for each method grouped by path difficulty. The values represent the percentage of runs in which the UGV arrived at the target end location, indicating that the robot team did not fail.

Failures occurred for two primary reasons: collisions with the environment or other agents, and the robot team losing sight of the target, with the last known position of the target exceeding the robot’s object detection range. Specifically, for the closed-loop method, a failure was noted if the unmanned ground vehicle (UGV) lost sight of the target and was on a trajectory that increased the distance from the target, see Fig. 5.2.

The FORWARD-PF method proved to be the most reliable, successfully completing all nine paths irrespective of their complexity, indicating its

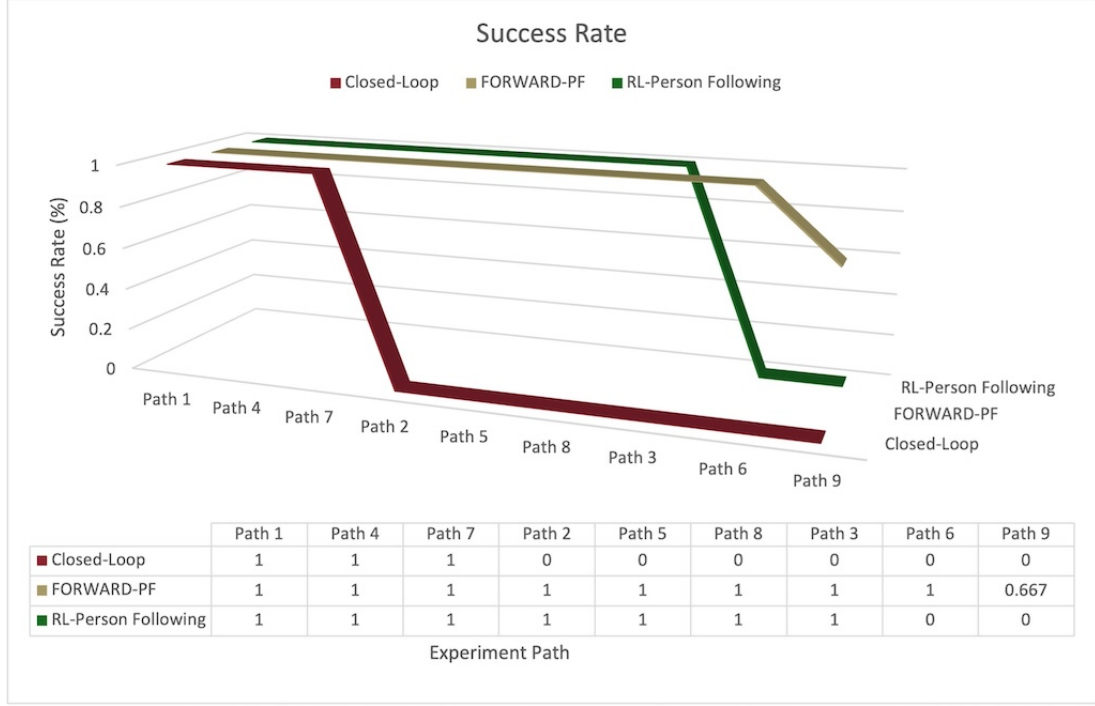


Figure 5.1: Success rate is the percentage of successful runs per experiment path.

robustness in handling different levels of path tortuosity. The RL method exhibited a moderate success rate, completing seven out of nine paths, with failures occurring in the most complex paths. The primary cause for failure with the RL approach was collisions with obstacles in the environment. This suggests that while RL can handle simpler and moderately complex paths, it struggles with highly tortuous environments. Meanwhile, the closed-loop method only completed three of the nine paths, all of which were the easiest, low-tortuosity paths.

From these results, it can be postulated that the FORWARD-PF method performed better in areas with tightly cluttered obstacles compared to the RL method, likely due to the challenges associated with transferring learning from a simple 2D simulator to a significantly more complex 3D simulator.



Figure 5.2: The Red circle and arrow indicate the current position and direction of movement of the UGV, which the Yellow circles illustrates this for the target. The Orange circle is the last known location of the target according to the UGV. This large difference in the UGV's knowledge of the target position vs the true position would result in a failure for this run as the possibility of the UGV reacquiring view of the target is minimal.

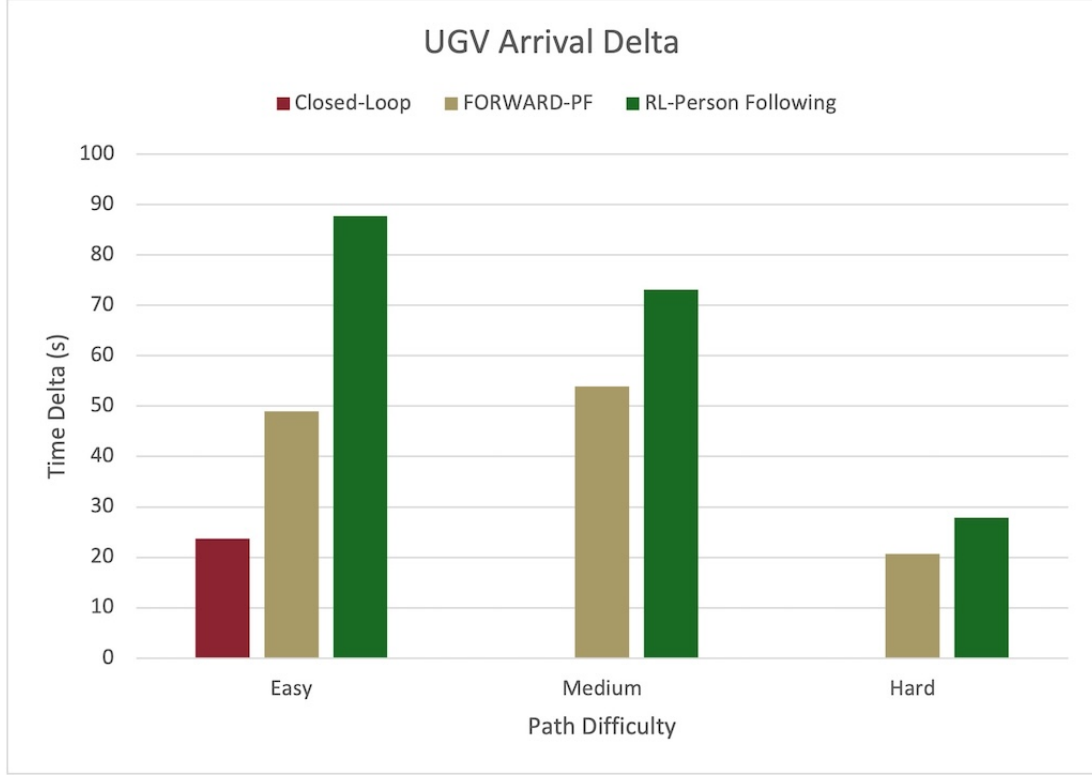


Figure 5.3: Time delta between target reaching end position and UGV arriving at the targets end position. The performance of the methods are grouped by path difficulty. Where no bar is displayed, that method was unable to complete the experiment for the given path.

Additionally, both the RL and FORWARD-PF methods demonstrated better adaptability to different target paths compared to the closed-loop method.

5.1.2 Arrival Time Delta

Arrival Time Delta is a metric representing the time difference between the target stopping at its end position and the UGV reaching the target, as shown in Fig. 5.3. This metric differs from cumulative distance travelled because it takes into account other factors, for example rotating in place and time to plan, making a more realistic metric for analysis of the technical approach.

The FORWARD-PF method consistently outperformed the RL method, arriving at the target 1.95 times faster on average. The more difficult the path,

the less of an advantage the FORWARD-PF method has over the RL in terms of arrival time. The closed-loop method demonstrated the smallest Arrival Time Delta on the paths it was able to complete. This is likely due to its low-level velocity control of the robot, compared to the FORWARD-PF and RL methods, where the UGV is driven by both a global and local planner, which requires additional time for path planning.

5.1.3 Time in View

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
CL	0.61	0.85*	0.48*	0.51	0.35*	0.38*	0.69	0.24*	0.34*
PF	0.98	1.0	1.0	1.0	1.0	0.98	1.0	0.97	0.72
RL	1.0	0.71	1.0	1.0	1.0	0.14*	1.0	0.96	0.20*

Table 5.2: Summary of simulation experiment, targets time in robot systems view. Time in view for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL).
* Experiment configuration failed.

Fig. 5.4 illustrates the area visible to the UAV at a given moment. The Time in View metric represents the duration the target remains within the UAV's visibility area. A higher Time in View value indicates that the target is visible to the UAV for a greater percentage of time. Consequently, this allows the UGV to have the most up-to-date location of the target, enabling it to make more informed decisions. Table 5.2 and Fig. 5.5 show that both the RL and FORWARD-PF methods averaged significantly higher target visibility percentages compared to the closed-loop method. For all successful experiment configurations, the target visibility percentage for both the RL and FORWARD-PF methods was at least 71%. In contrast, the two configurations where the RL method failed had target visibility percentages under 20% during the experiment.

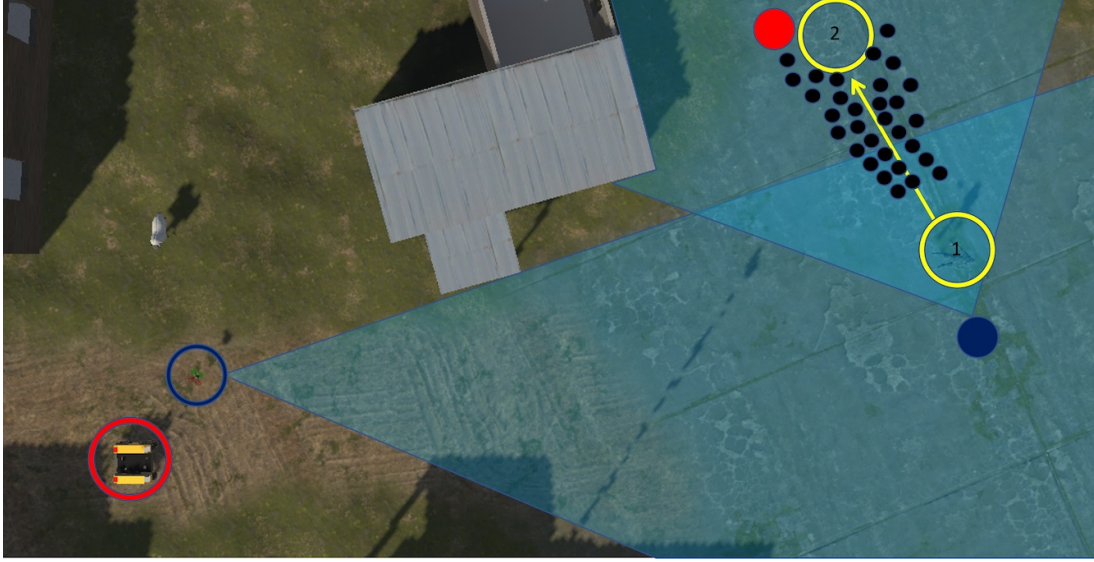


Figure 5.4: Transparent light blue area visualizes the UAV's current area of visibility. Dark blue circle and area visualize goal location and the UAV's area of visibility at that goal location. Black dots represent the possible future locations of the target. Red circle visualizes a UGV goal location that minimizes UGV distance to the target.

Future work could investigate the minimum target visibility percentage required for a successful run in air-ground heterogeneous robot teams. Understanding this threshold will help in optimizing the performance and reliability of these systems.

5.1.4 Cumulative Distance Travelled

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
CL	65.95	N/A	N/A	84.29	N/A	N/A	62.77	N/A	N/A
PF	67.93	58.06	38.17	88.25	78.58	98.22	64.68	106.08	55.46
RL	78.69	72.62	45.34	106.04	78.95	N/A	79.46	92.75	N/A

Table 5.3: Summary of simulation experiment UGV cumulative distance traveled. UGV cumulative distance for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL).

Fig. 5.8 and Fig. 5.9 illustrate the advantage of a lower cumulative distance travelled in the best case where taking a shortcut is the optimal path in this

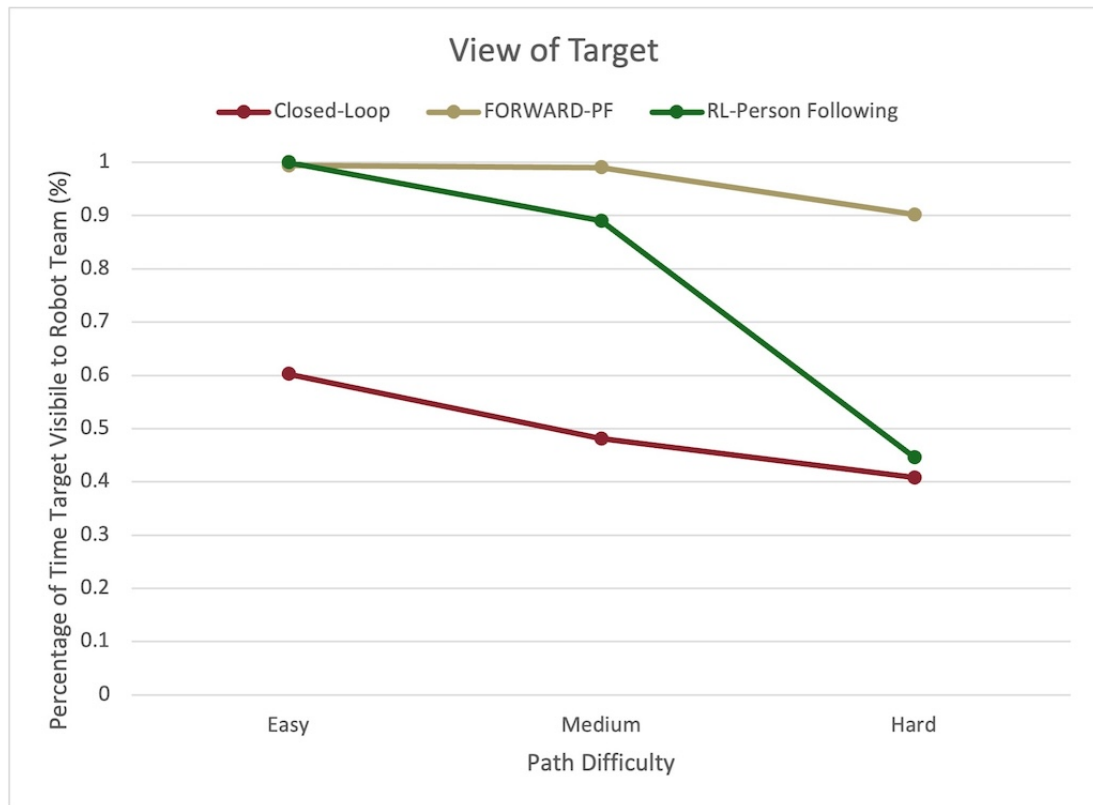


Figure 5.5: *Percentage of time target is visible to robot team. The performance of the methods are grouped by path difficulty.*

case compared to following the path of the target which is longer in length. The closed-loop method demonstrated the lowest cumulative distance traveled across all experimental configurations it successfully completed. This is shown in Table 5.3 and Fig. 5.6. Meanwhile, the FORWARD-PF method averaged a slightly lower cumulative distance compared to the RL method. Notably, the cumulative distance traveled by the FORWARD-PF method is only marginally longer than that of the closed-loop method, which serves as the baseline. This observation suggests that the FORWARD-PF method approaches the efficiency of the baseline closed-loop method, which is a significant and interesting finding.

5.1.5 Max Distance Between Target and UGV

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
CL	44.03	N/A	N/A	56.38	N/A	N/A	39.98	N/A	N/A
PF	52.95	38.65	33.39	68.41	44.18	71.68	54.81	58.51	49.48
RL	53.58	40.75	32.93	68.31	50.04	N/A	54.35	56.28	N/A

Table 5.4: Summary of simulation experiment UGV max distance from target. UGV max distance for the three methods evaluated through nine experiment paths. closed-loop (CL), FORWARD-PF (PF), and RL-Person Following (RL).

Examining the maximum distance, Table 5.4 and Fig. 5.7, which indicates the farthest distance between the UGV and the target during the experiment, the closed-loop method exhibits the lowest maximum distance for all the configurations it successfully completed. This is expected, as the closed-loop method is purely reactive. Unlike the FORWARD-PF and RL methods, which rely on a planner to navigate, the closed-loop method allows the robot to move immediately upon detecting the target.

For the FORWARD-PF and RL methods, once the robot detects the target, it must determine the appropriate direction or goal location. The goal location is

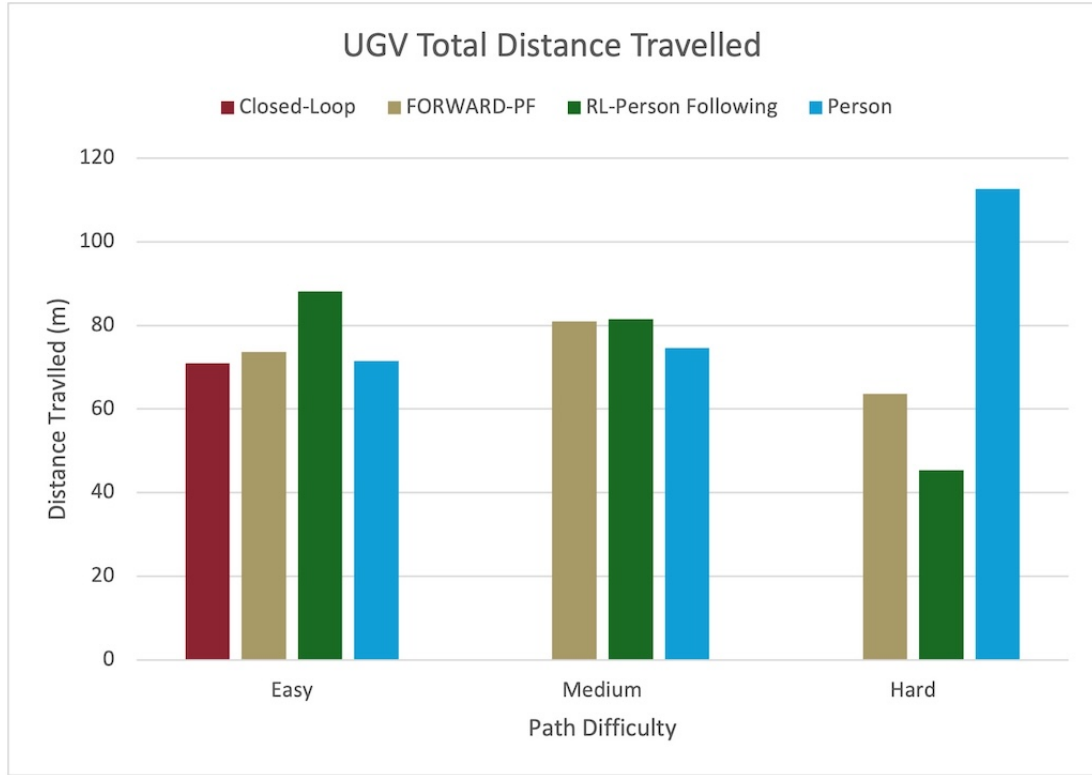


Figure 5.6: Cumulative distance travelled by UGV . The performance of the methods are grouped by path difficulty. The blue illustrates the total distance travelled by the person. Where no bar is displayed, that method was unable to complete the experiment for the given path. The FORWARD-PF and RL-Person Following methods resulted in cumulative distances of the UGV less than the person's path. This is due to the methods exploiting the robot team and by the UAV maintaining visibility of the person, the UGV navigates a

then sent to the planner, which requires some planning time before the robot can execute the plan. This additional planning time explains why the closed-loop method achieves the lowest maximum distance.

When comparing the FORWARD-PF method to the RL method, the FORWARD-PF method averages a lower maximum distance to the target. This suggests that the FORWARD-PF method may be making better decisions than the RL method. However, it is important to note that the FORWARD-PF method likely makes fewer path plans compared to the RL method. The FORWARD-PF method continuously provides a goal position based on the target's current position, while the RL method frequently changes direction based on the model's current directives, leading to more frequent adjustments.

5.1.6 Observed Behaviour

An observed advantage of the aerial-ground robot team is that the slow-moving UGV can take shortcuts when appropriate. Without prior knowledge of the target's intentions or path, it is impossible to predict the shortest route to the target's position. However, as the UGV is continuously updated with the target's position by the UAV, it can plan the most efficient path to the current target location without needing to maintain line-of-sight. This behavior is illustrated in Fig. 5.8 and Fig. 5.9. Both the FORWARD-PF and RL methods exhibited this behavior during experiments for path 3, and the FORWARD-PF method showed this behavior again for path 9.

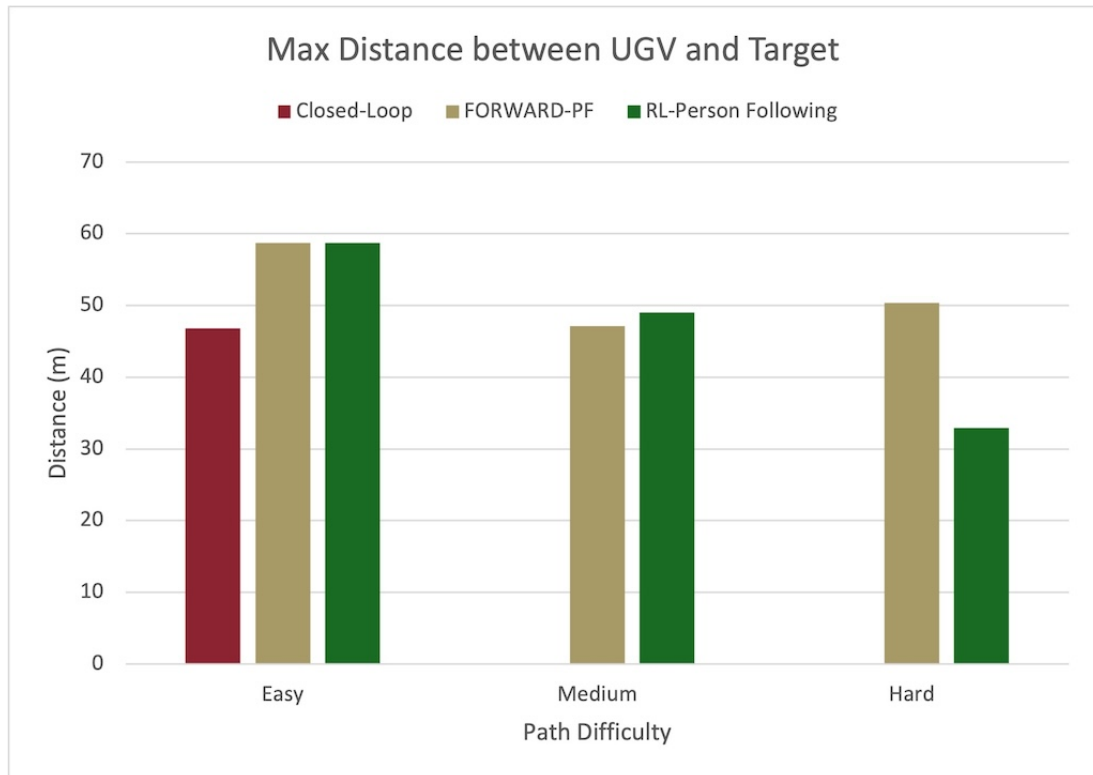


Figure 5.7: Maximum distance between UGV and target during length of the experiment. The performance of the methods are grouped by path difficulty. Where no bar is displayed, that method was unable to complete the experiment for the given path.

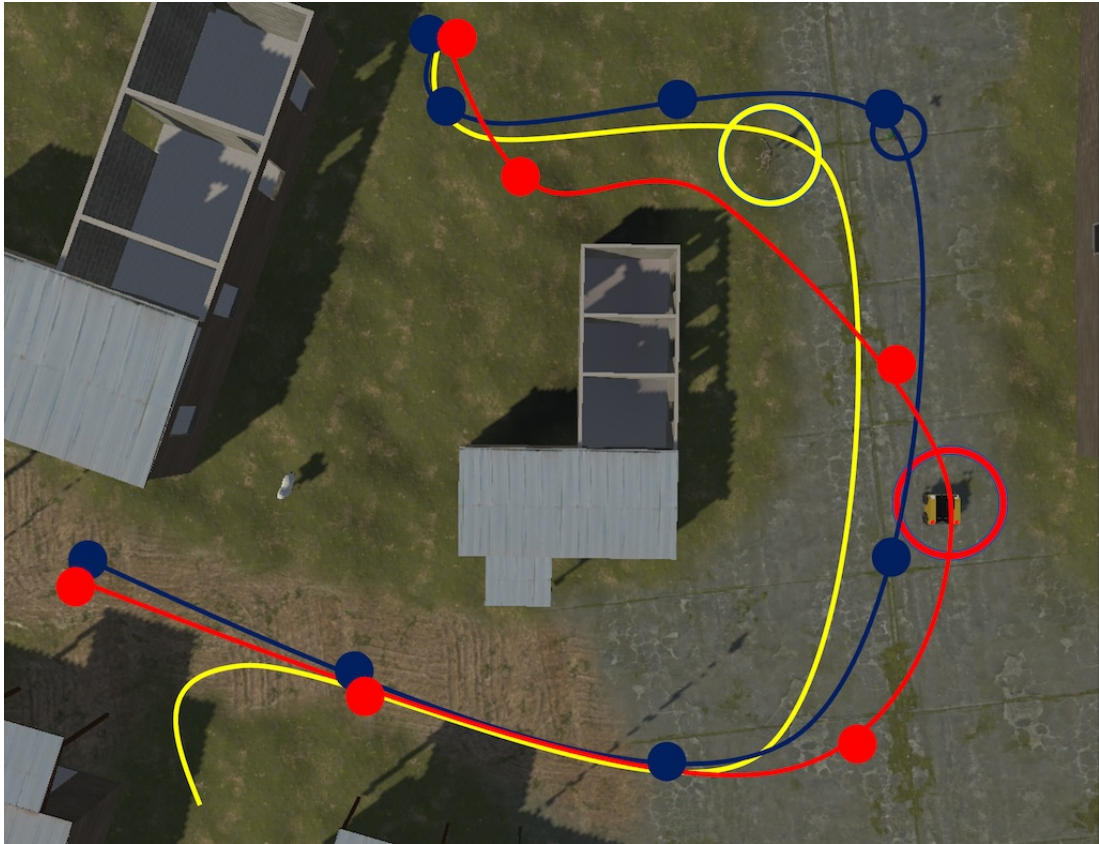


Figure 5.8: Target (Yellow), UAV (Blue), UGV (Red). Paths showing method that follows the path of the target either through design or the UGV having equal velocity to the target which would naturally lead it to take a similar path.

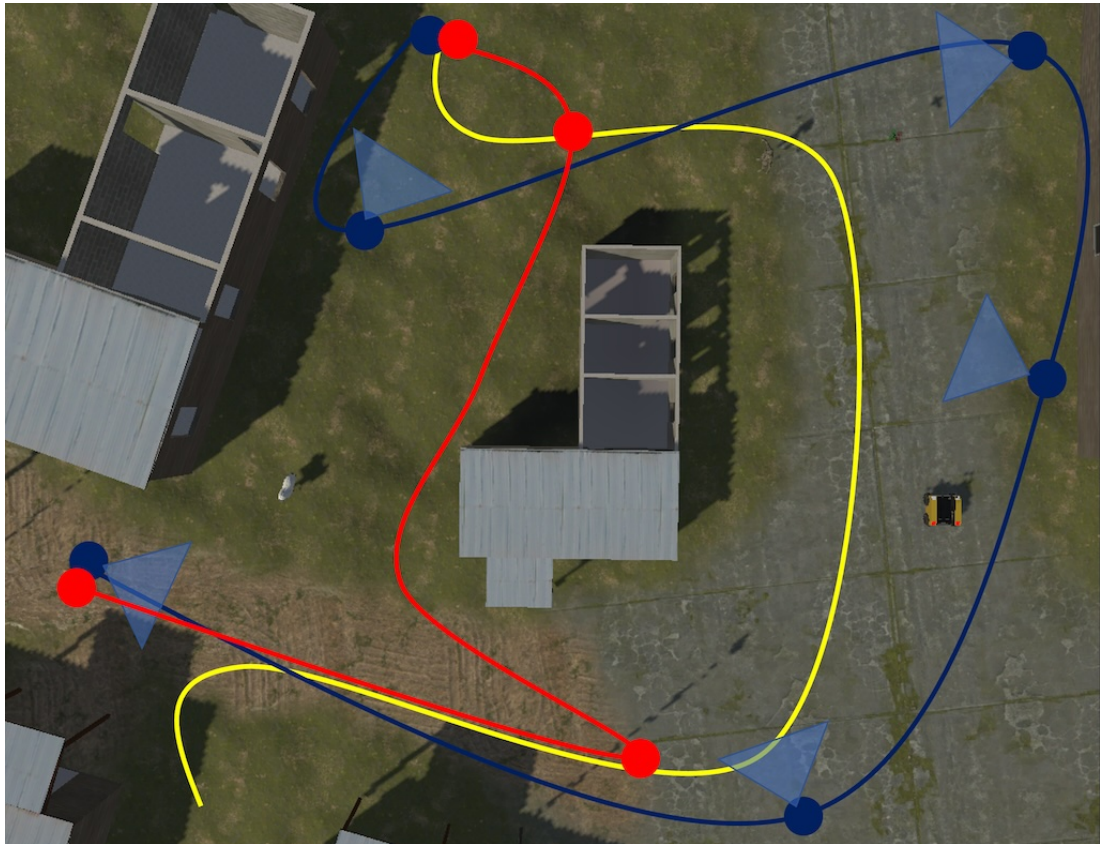


Figure 5.9: Target (Yellow), UAV (Blue), UGV (Red). Paths showing the UGV taking a shortcut to navigate to the target with a decreased path length. The UAV is seen still following a similar path to the target as to maintain visibility.

6 Discussion

The experiments conducted aimed to evaluate the effectiveness of different methods for a heterogeneous multi-robot system, comprising an aerial robot and a ground robot, in a person-following task. Three methods were investigated: My novel algorithm named FORWARD-PF, a MARL approach, and a closed-loop method for the UGV. The results from these methods provide insights into their strengths, limitations, and potential areas for future improvement.

The MARL and FORWARD-PF methods were both significantly more adaptable than the baseline closed-loop method. The closed-loop method only successfully completed 3 out of the 9 total paths, all of which were classified as easy path difficulties. In terms of adaptability, the FORWARD-PF method performed marginally better than the MARL method, completing all 9 paths compared to the MARL's 7 out of 9. The failures of the MARL method might be attributed to the difficulties and limitations associated with adapting the pre-trained model to the complex 3D environment of the simulator in which the experiments were conducted.

For the paths it was able to complete, the closed-loop method achieved the shortest arrival times, outperforming the other methods. However, with a success rate of only 33%, this advantage is overshadowed by its inability to succeed in most environments. Meanwhile, the FORWARD-PF method reached the target almost twice as fast as the MARL method. This can be attributed to the FORWARD-PF method's design, which provides goal locations around

the target’s current position, allowing the planner to create an optimal path for the robot at any given moment. In contrast, the MARL approach provides instantaneous actions that need to be interpreted, resulting in a lower time horizon and less efficient paths.

6.1 Limitations

6.1.1 Domain Adaptation

One significant limitation of this research is the difficulty associated with adapting a pre-trained model to a new environment. The models were initially trained in a simplistic 2D environment, but experiment deployment required operating in a complex 3D environment. This transition introduced numerous challenges, including differences in sensor data interpretation, environmental dynamics, and agent interactions, which the pre-trained models were not optimized for.

These adaptation difficulties limit the significance of the conclusions drawn from the results. The performance observed in the new environment may not fully reflect the model’s true capabilities, as it is influenced by the model’s ability to generalize from the training environment to the testing environment. This limitation underscores the need for developing methods that can robustly transfer learning across varied environments or for training models directly within the operational environment to ensure more accurate and reliable performance assessments.

6.1.2 Development Time

Another important point of discussion is the amount of time it takes to develop a tuned algorithm like the FORWARD-PF method compared to training and deploying an MARL model. This difference is significant and it is important to look at the results with that perspective. Development of FORWARD-PF

took significantly more time and effort compared to the MARL approach shown in this study. While FORWARD-PF outperformed the MARL method in the simulated experiments, I am hesitant to conclude it is a better method. Instead, I would argue the main value from the system is the advantage a heterogeneous system can bring to complex tasks. Also, the method one would choose is very dependent on their use case. FORWARD-PF performed in a very dependable and consistent manner compared to the more inconsistent MARL method. Use case will aid in deciding what method to use.

6.2 Future Work

Future work should investigate the minimum target visibility percentage required for successful operation in air-ground heterogeneous robot teams. Understanding this threshold will help optimize the performance and reliability of these systems.

Additionally, future evaluations should be conducted on real hardware, necessitating a more capable aerial platform. During attempts to run the real-world validation, the micro-aerial platform's limitations imposed significant constraints, restricting the experiment space to a small indoor lab. Acquiring a more capable aerial platform capable of performing mapping and object detection would expand the operational space for this robot system.

Future work should encompass an in-depth exploration of domain adaptation from both utilization and evaluation standpoints. It is crucial to ascertain the threshold at which pre-trained models can be effectively deployed across various platforms. Additionally, evaluating the factors that constrain robotic performance, whether limitations stem from the model itself or its adaptation to a particular platform is of paramount importance.

The primary goal of this work was to develop and evaluate person-following methods in terms of robustness and efficiency. Evaluating the social acceptance and perception of these methods from the perspective of the person being followed is outside the scope of this body of work [38]. However, this is an important area that needs to be explored as human-robot interactions become more prevalent. Future directions include developing more human-friendly methods and evaluating performance with the human in mind.

7 Conclusion

This research addresses the challenge of developing methods for a heterogeneous multi-robot team, consisting of one aerial robot and one ground robot, to effectively follow a person. The primary objective was to develop and evaluate methods for coordinating these robots, considering their distinct capabilities and limitations.

In person-following tasks, assumptions about the target’s compliance with the robotic system’s perception and navigation limitations often do not hold. This research assumes no *a-priori* information is available to predict the target’s motion, and that the target’s trajectory is non-ergodic, meaning their behavior is not deterministic even under identical circumstances.

Robotic systems are constrained by factors such as payload, sensing, maneuverability, and speed limitations. Micro-aerial robots are extremely payload-limited, restricting their sensing and computational power, which can prevent fully independent autonomous operation. To address this, the system pairs a micro-aerial robot (UAV) with a ground robot (UGV) that has greater payload capacity for more powerful sensing and computational capabilities, such as mapping, navigation, and person detection. This combination allows the UAV to localize itself and the person in the same space as the UGV.

The environment also presents limitations. In this application, the UAV is assumed to maintain a fixed height above the ground. Unlike some outdoor environments where a UAV can gain altitude for a bird’s-eye view, this research focuses on scenarios where this is not possible due to factors such

as indoor settings, intermittent height obstacles like tree canopies, or airspace restrictions. This research specifically explores the use of UGV and UAV teams within these constrained spaces.

Robot agents are often constrained by their tasks and the environments in which they operate. To address these specific velocity, height, and agent-related constraints, a heterogeneous multi-robot system equipped with my novel algorithm named FORWARD-PF was developed. This system is capable of following a target agent through a height-restricted environment. FORWARD-PF was compared with a UGV-only closed-loop method and a multi-agent reinforcement learning (MARL) method.

From evaluating these methods through simulated experimentation, FORWARD-PF proved the most adaptable to the various environments and paths which the robot system was tasked with following a target through. FORWARD-PF successfully completed all paths, while the MARL method completed 7 out of 9 only failing during 2 of the 3 most difficult paths. Meanwhile, the baseline comparison closed-loop method was only able to successfully complete 3 of the 9 paths, all of which were minimal difficulty.

By leveraging the strengths of each platform, this research addresses the limitations faced by ground robots in keeping pace with humans. It fills the research gap by developing novel strategies for coordination and task allocation, enhancing the robots' ability to follow a person in a shared environment.

Bibliography

- [1] M. Kouzeghar, Y. Song, M. Meghjani, and R. Bouffanais, "Multi-Target Pursuit by a Decentralized Heterogeneous UAV Swarm using Deep Multi-Agent Reinforcement Learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 3289–3295.
- [2] "Clearpath robotics warthog," <https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/>.
- [3] M. Natarajan, E. Seraj, B. Altundas, R. Paleja, S. Ye, L. Chen, R. Jensen, K. C. Chang, and M. Gombolay, "Human-Robot Teaming: Grand Challenges," *Current Robotics Reports*, vol. 4, no. 3, pp. 81–100, Aug. 2023.
- [4] S. S. O V, R. Parasuraman, and R. Pidaparti, "Impact of Heterogeneity in Multi-Robot Systems on Collective Behaviors Studied Using a Search and Rescue Problem," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2020, pp. 290–297.
- [5] Q. Yang and R. Parasuraman, "Needs-driven Heterogeneous Multi-Robot Cooperation in Rescue Missions," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2020, pp. 252–259.
- [6] M. Pólka, S. Ptak, and Ł. Kuziora, "The Use of UAV's for Search and Rescue Operations," *Procedia Engineering*, vol. 192, pp. 748–752, Jan. 2017.
- [7] L. J. Guibas, J.-C. Latombe, S. M. Lavalley, D. Lin, and R. Motwani, "A visibility-based pursuit-evasion problem," *International Journal of Computational Geometry & Applications*, vol. 09, no. 04n05, pp. 471–493, Aug. 1999.
- [8] E. Lozano, I. Becerra, U. Ruiz, L. Bravo, and R. Murrieta-Cid, "A visibility-based pursuit-evasion game between two nonholonomic robots in environments with obstacles," *Autonomous Robots*, vol. 46, no. 2, pp. 349–371, Feb. 2022.
- [9] E. Lozano, U. Ruiz, I. Becerra, and R. Murrieta-Cid, "Surveillance and Collision-Free Tracking of an Aggressive Evader With an Actuated Sensor Pursuer," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6854–6861, Jul. 2022.
- [10] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3045–3051.

- [11] M. J. Islam, J. Hong, and J. Sattar, "Person-following by autonomous robots: A categorical overview," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1581–1618, Dec. 2019.
- [12] G. Xing, S. Tian, H. Sun, W. Liu, and H. Liu, "People-following system design for mobile robots using kinect sensor," in *2013 25th Chinese Control and Decision Conference (CCDC)*, May 2013, pp. 3190–3194.
- [13] F. Vasconcelos and N. Vasconcelos, "Person-following UAVs," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Lake Placid, NY, USA: IEEE, Mar. 2016, pp. 1–9.
- [14] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*. Arlington Virginia USA: ACM, Mar. 2007, pp. 17–24.
- [15] B.-J. Lee, J. Choi, C. Baek, and B.-T. Zhang, "Robust Human Following by Deep Bayesian Trajectory Prediction for Home Service Robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 7189–7195.
- [16] M. R. Batista, D. G. Macharet, and R. A. F. Romero, "Socially Acceptable Navigation of People with Multi-robot Teams," *Journal of Intelligent & Robotic Systems*, vol. 98, no. 2, pp. 481–510, May 2020.
- [17] M. Owen, H. Yu, T. McLain, and R. Beard, "Moving ground target tracking in urban terrain using air/ground vehicles," in *2010 IEEE Globecom Workshops*. Miami, FL, USA: IEEE, Dec. 2010, pp. 1816–1820.
- [18] S. Minaeian, J. Liu, and Y.-J. Son, "Vision-Based Target Detection and Localization via a Team of Cooperative UAV and UGVs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 1005–1016, Jul. 2016.
- [19] J. Ahn, M. Kim, S. Kim, S. Lee, and J. Park, "Formation-Based Tracking Method for Human Following Robot," in *2018 15th International Conference on Ubiquitous Robots (UR)*, Jun. 2018, pp. 24–28.
- [20] M. J. Islam, J. Hong, and J. Sattar, "Person-following by autonomous robots: A categorical overview," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1581–1618, Dec. 2019.
- [21] B. S. B. Dewantara and J. Miura, "Generation of a socially aware behavior of a guide robot using reinforcement learning," in *2016 International Electronics Symposium (IES)*, Sep. 2016, pp. 105–110.

- [22] Y. Orita and T. Fukao, "Robust Human Tracking of a Crawler Robot," *Journal of Robotics and Mechatronics*, vol. 31, no. 2, pp. 194–202, Apr. 2019.
- [23] L. Yuan, C. Reardon, G. Warnell, and G. Loianno, "Human Gaze-Driven Spatial Tasking of an Autonomous MAV," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1343–1350, Apr. 2019.
- [24] C. Reardon and J. Fink, "Air-ground robot team surveillance of complex 3D environments," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. Lausanne, Switzerland: IEEE, Oct. 2016, pp. 320–327.
- [25] L. Schwendeman, A. SaLoutos, E. Stanger-Jones, and S. Kim, "Improving Domain Transfer of Robot Dynamics Models with Geometric System Identification and Learned Friction Compensation," in *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, Dec. 2023, pp. 1–8.
- [26] R. Garcia, R. Strudel, S. Chen, E. Arlaud, I. Laptev, and C. Schmid, "Robust Visual Sim-to-Real Transfer for Robotic Manipulation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2023, pp. 992–999.
- [27] R. Juan, H. Ju, J. Huang, R. Gomez, K. Nakamura, and G. Li, "Sim-to-Real Policy and Reward Transfer with Adaptive Forward Dynamics Model," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 7212–7218.
- [28] S. Sharma, E. Novoseller, V. Viswanath, Z. Javed, R. Parikh, R. Hoque, A. Balakrishna, D. S. Brown, and K. Goldberg, "Learning Switching Criteria for Sim2Real Transfer of Robotic Fabric Manipulation Policies," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*, Aug. 2022, pp. 1116–1123.
- [29] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, Oct. 2020.
- [30] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with Non-Stationarity in Multi-Agent Deep Reinforcement Learning," Jun. 2019.
- [31] A. Ghimire, X. Zhang, S. Javed, J. Dias, and N. Werghi, "Robot Person Following in Uniform Crowd Environment," May 2022.

- [32] J. Gregory, J. Fink, E. Stump, J. Twigg, J. Rogers, D. Baran, N. Fung, and S. Young, "Application of Multi-Robot Systems to Disaster-Relief Scenarios with Limited Communication," in *Field and Service Robotics: Results of the 10th International Conference*, D. S. Wettergreen and T. D. Barfoot, Eds. Cham: Springer International Publishing, 2016, pp. 639–653.
- [33] Unity Technologies, "Unity game engine," 2022, computer software. [Online]. Available: <https://unity.com>
- [34] C. Yi, H. Chen, X. Hu, and Y. Xu, "Domain Adaptation from Public Dataset to Robotic Perception Based on Deep Neural Network," in *2020 Chinese Automation Congress (CAC)*, Nov. 2020, pp. 6218–6222. [Online]. Available: <https://ieeexplore.ieee.org/document/9327112>
- [35] "Bitcraze crazyflie 2.1," <https://www.bitcraze.io/products/crazyflie-2-1/>.
- [36] "Clearpath robotics jackal," <https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/>.
- [37] "Velodyne vlp-16," <https://ouster.com/products/hardware/vlp-16>.
- [38] R. Möller, A. Furnari, S. Battiato, A. Härmä, and G. M. Farinella, "A survey on human-aware robot navigation," *Robotics and Autonomous Systems*, vol. 145, p. 103837, Nov. 2021.

A Experiment Results Grouped by Method

A.1 Closed-Loop Method

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
ET	66.33	N/A	N/A	85.0	N/A	N/A	64	N/A	N/A
$E\Delta$	21.21	N/A	N/A	30.61	N/A	N/A	19.35	N/A	N/A
CD	65.95	N/A	N/A	84.29	N/A	N/A	62.77	N/A	N/A
SR	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
TIV	0.61	0.85*	0.48*	0.51	0.35*	0.38*	0.69	0.24*	0.34*

Table A.1: Closed-loop Method. *ET*, elapsed time for UGV to reach target end position. *$E\Delta$* , difference in time between target and UGV reaching the end position. *CD*, cumulative distance UGV travelled to reach target end position. *SR*, success rate, *TIV*, percent of time target in view of the robot system.

*Failures due to collision.

**Failures due to losing sight of target.

A.2 FORWARD-PF

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
ET	89.0	76.0	57.67	111.33	109.33	145.67	90.67	142.0	89.66
$E\Delta$	43.88	30.86	7.55	56.94	55.0	43.12	46.01	75.89	15.57
CM	67.93	58.06	38.17	88.25	78.58	98.22	64.68	106.08	55.46
SR	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.75
TIV	0.98	1.0	1.0	1.0	1.0	0.98	1.0	0.97	0.72**

Table A.2: FORWARD-PF Method. *ET*, elapsed time for UGV to reach target end position. $E\Delta$, difference in time between target and UGV reaching the end position. *CD*, cumulative distance UGV travelled to reach target end position. *SR*, success rate, *TIV*, percent of time target in view of the robot system.

*Failures due to collision.

**Failures due to losing sight of target.

A.3 RL-Person Following

	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8	Path 9
ET	130.0	110.33	78.0	141.33	132.0	N/A	136.0	142.5	N/A
E Δ	84.88	65.19	27.89	86.94	77.67	N/A	91.35	76.39	N/A
CD	78.69	72.62	45.34	106.04	78.95	N/A	79.46	92.75	N/A
SR	1.0	1.0	1.0	1.0	1.0	0.0	1.0	1.0	0.0
TIV	1.0	0.71	1.0	1.0	1.0	0.14*	1.0	0.96	0.20*

Table A.3: *RL-Person Following Method. ET, elapsed time for UGV to reach target end position. E Δ , difference in time between target and UGV reaching the end position. CD, cumulative distance UGV travelled to reach target end position. SR, success rate, TIV, percent of time target in view of the robot system.*

**Failures due to collision.*

***Failures due to losing sight of target.*