Reinforcement Learning with Reasoning for Long-horizon Robotic Tasks

by

Zhitao Yu

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 6, 2023

Keywords: Ground Robot, UAV, Long-horizon tasks, Path planning, Reinforcement learning

Approved by

Shiwen Mao, Professor of Electrical and Computer Engineering
Thaddeus Roppel, Associate Professor of Electrical and Computer Engineering
Xiaowen Gong, Assistant Professor of Electrical and Computer Engineering
Mark Nelms, Professor and Chair of Electrical and Computer Engineering

Abstract

Recent developments in Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) have attracted intensive research interest from both academia and industrial areas. Although Unmanned Aerial Vehicles (UAV) are usually deployed outdoors, there is increasing interest in applying UAVs for indoor applications. It is a highly attractive and challenging task to precisely localize a UAV in an indoor environment where Global Positioning System (GPS) service is absent. To achieve high accuracy and low cost for localization, a Radio-frequency Identification (RFID) enhanced UAV system that provides a precise 6 degrees of freedom (6-DoF) pose for UAVs. Moreover, UGVs are good compliments for UAVs which made them be widely used for various tasks. Therefore, the control commands communication between UAVs and UGVs is crucial as well. Furthermore, they are both constrained by some essential features that make them incapable of completing complicated tasks in many scenarios. For example, the UGV cannot reach high altitudes, while the UAV is limited by its power supply and smaller payload capacity. In my dissertation, I want to present a deep reinforcement learning(DRL)-based network that could generate an optimal strategy to make a UGV and UAV form a coalition that is complementary and cooperative for the completion of tasks that they are incapable of achieving alone. At the same time, I also would like to discuss the challenge and solutions when using DRL methods for solving such long-horizon robotic tasks. DRL methods usually suffer when the state and action spaces are very large. So the way we handle the observations during training is essential. In the last section, a reasoning scheme that enables robots better understand their tasks in the environment is investigated to promote the intelligence and robustness of the cooperation system.

Acknowledgments

This work is supported in part by the US National Science Foundation under grants CNS-1702957, CNS-2107190, CNS-2148382, CNS-1822055, ECCS-1923163, ECCS-1923717, Auburn University RFID Lab, and by the Wireless Engineering Research and Education Center (WEREC) at Auburn University, Auburn, AL, USA.

I would like to thank my advisor, Dr. Shiwen Mao, for all the help and guidance that he has given me over the past several years. Additionally, I would like to express my sincere gratitude to Dr. Jian Zhang for the continuous support of my Ph.D. research, his patience, and his motivation. Besides my advisor, I am extremely grateful to my committee members, Dr. Xiaowen Gong, Dr. Thaddus Roppel, Dr.Yang Zhou, and Dr. Mark Nelms, for their insightful comments and support. My sincere appreciation also goes to my friends, Dr. Xiangyu Wang, Dr. Yibo Lyu, Dr. Chao Yang, and Junwei, for their advice and inspiration. Finally, I gratefully acknowledge the assistance of my parents and family, who give me understanding and love during my study period at Auburn University.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last decade has witnessed significant developments in the unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) technologies, which have enabled their wide deployment for various applications, such as surveillance, search and rescue, inspection [1], inventory counting [2, 3], and more [4, 5, 6, 7, 8, 9]. Recently, With the development of wireless communication technology such as 5G network [10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23], researchers have shown a growing interest to deploy them for more complex tasks that require multiple UAVs or UGVs to work together to improve efficiency [24] cooperatively. Most existing research focuses on cooperation in a multi-agent (or multi-robot) system that consists of a group of UAVs or UGVs.The topic of this thesis is to build an intelligent cooperation system consisting of multiple UAVs and UGVs for understanding and completing long-horizon tasks efficiently in dynamic indoor environments [25]. Basically, there are three challenges when we build this system: localization, cooperation, and enabling robots with reasoning abilities. Thus, deep reinforcement learning(DRL) techniques join the field of robotic policy exploring methods. Secondly, DRL methods suffer from the huge state and action space. In such cases, it becomes difficult for the algorithm to explore the entire space and find an optimal policy. Furthermore, another challenge is how to leverage the reinforcement learning method to enable robots with rudimentary reasoning abilities, such as understanding tasks in the environment and selecting reasonable actions. In my dissertation, I would focus on these challenges to improve the current intelligent multiple UAV-UGV intelligent cooperation system.

1.1 Summary of Contributions

1.1.1 RFUAV: Robust RFID Based 6-DoF Localization for Unmanned Aerial Vehicles

In this paper, indoor localization with RFID technology would be investigated because the accurate pose is crucial for UAV landing on the Ground Robot and command control. It is a highly challenging task to precisely localize a UAV in an indoor environment where Global Positioning System (GPS) service is absent [26, 27, 28]. Moreover, due to the limited load-bearing capacity of UAVs, the positioning method used cannot leverage large onboard equipment. Therefore, Three light weighted Passive RFID tags or more ultra-high frequency (UHF) RFID tags are attached to the UAV and interrogated by a Commercial Off-The-Shelf (COTS) RFID reader with multiple antennas. The main contributions of this work are summarized as follows:

1. We developed a real-time RFID tag tracing system that incorporates a Bayesian filter [29]. Based on the phase measurements of RFID responses from a COTS reader, the tag tracker can track the motion of multiple UHF RFID tags simultaneously.

2. We propose a real-time UAV pose estimator. Based on the positions of the attached tags, the pose estimator can compute precise 6-DoF poses for the UAV in a 3D space with a singular value decomposition method.

3. We tested the RFUAV system with COTS RFID tags and reader and demonstrate its performance in a representative indoor environment. Experimental results demonstrate that RFUAV can achieve precise poses with only 0.04m error in position and 2° error in orientation. Such performance enables a UAV to autonomously navigate in an indoor environment.

1.1.2 IADRL: Imitation Augmented Deep Reinforcement Learning Enabled UGV-UAV Coalition for Tasking in Complex Environments

We propose an Imitation Augmented Deep Reinforcement Learning Network (IADRL) that enables a UGV and UAV to form a coalition that is complementary and cooperative for the

completion of tasks that they are incapable of achieving alone. IADRL learns the underlying complementary behaviors of UGVs and UAVs from a demonstration dataset that is collected from some simple scenarios with non-optimized strategies. Based on observations from the UGVs and UAVs, IADRL provides an optimized policy for the UGV-UAV coalitions to work in an complementary way while minimizing the cost. We evaluate the IADRL approach in an visual game-based simulation platform, and conduct experiments that show how it effectively enables the coalition to cooperatively and cost-effectively accomplish tasks.The main contributions of this work are summarized as follows:

1. The proposed network enables a UGV and UAV to form a coalition to complement and enhance each other during complicated tasks that either agent alone could not complete. It also optimizes the complementary coordination strategy among those agents to accomplish various tasks with the lowest cost (e.g. minimum power consumption, optimized navigational trajectory with shortest steps, etc.).

2. We develop an imitation network to learn the complicated complementary behavior of UGVs and UAVs in the coalition using demonstration data that was collected from simple scenarios with non-optimized strategies. This will greatly reduce the effort of modeling the complementary behaviors of agents in the coalition.

3. We test IADRL in a visual game-based simulated environment, and show that our network enables the complementary behaviors of UGVs and UAVs during searching tasks.

1.1.3 RIRL: A Recurrent Imitation and Reinforcement Learning Method for Long-Horizon Robotic Tasks

In this paper, we propose Recurrent Imitation and Reinforcement Learning (RIRL) to address the challenges and enable robots for such tasks. The proposed RIRL incorporates a long short-term memory (LSTM) network to retain long-term memories, which could be an effective and efficient method to tackle the long dependency problem raised in long-horizon robotic tasks. To assess the performance of the RIRL, we test it with an optimized path planning problem for a robot to perform a Radio-frequency identification (RFID) inventory in dynamic and previously

unknown environments. We experimentally validate RIRL's feasibility and effectiveness in a visual game-based simulation platform, where the proposed RIRL model outperforms three baseline schemes with considerable gains. The main contributions of our work are summarized as follows:

1. To the best of our knowledge, this is the first work to develop an LSTM-embedded imitation and reinforcement learning network to enhance the action prediction ability of the agent by exploiting historical observations.

2. The proposed model allows the agent to explore the unknown environment in a continuous action space, to deal with the exponentially boosted complexity and uncertainty.

3. We experimentally validate the feasibility of RIRL in a visual game-based simulated environment and demonstrate that the proposed model enables the robot to perform inventory tasks in a dynamic environment.

1.1.4 Multi-state-space Reasoning Reinforcement Learning for Long-horizon RFID-based Robotic Searching and Planning Tasks

we propose a novel learning framework, called Multiple State Spaces Reasoning Reinforcement Learning (SRRL), to endow the agent with the primary reasoning capability. First, we abstract the implicit and latent links between multiple state spaces. Then, we embed historical observations through an LSTM network to preserve long-term memories and dependencies. The proposed SSRL's ability of abstraction and long-term memory enables agents to execute long-horizon robotic searching and planning tasks more quickly and reasonably by exploiting the correlation between RFID sensing properties and the environment occupation map. We experimentally validate the efficacy of SRRL in a visual game-based simulation environment. Our methodology outperforms three state-of-the-art baseline schemes by significant margins. The main contributions of our work could be summarized in the following:

1. To the best of our knowledge, this is the first study to integrate a reasoning scheme abstracted from various state spaces in a DRL network, allowing the agent to comprehend the latent correlation across state spaces with different dimensions and bases.

2. Incorporating the reasoning scheme and recurrent networks, the proposed framework enables the agent to achieve long-term goals despite exponentially increasing complexity and unpredictability (e.g., exploring a wide area of an unknown environment in a continuous action space).

3. By experimentally validating SRRL's viability in a visual game-based simulation environment, we prove that the proposed model enables the robot to execute long-horizon inventory management tasks in a dynamic environment.

Chapter 2

Robust RFID based 6-DoF Localization for Unmanned Aerial Vehicles

The last decade witnessed a tremendous growth of interest in unmanned aerial vehicles (UAV) [30]. Thanks to its outstanding maneuverability, small size, and low cost, UAVs have been widely adopted for surveillance, entertainment, search and rescue, inspection, and maintenance applications. These applications occur mostly in outdoor environments with existing navigation systems that rely on inertial sensors and Global Positioning System (GPS). Due to the low positioning resolution and the absence of GPS signal in an indoor environment (i.e.,warehouses, retail stores, etc.), most existing UAVs are infeasible for operation indoors. Consequently, recent research has investigated the problem of UAV indoor localization. The most popular indoor UAV localization methods can be categorized by measurement into three groups: vision-/laser-based, inertial navigation system (INS)-based, and wireless signals-based solutions.

The vision-based solutions are proposed to exploit the visual information provided by one or two cameras [31, 32, 33, 34, 35] for UAV indoor localization and navigation. Most of the vision-based solutions are leveraged with simultaneous localization and mapping (SLAM) technologies and use an Iterative Closest Point (ICP) algorithm to achieve real-time indoor localization. One of the first real-time, monocular SLAM methods based on nonlinear filtering was proposed by Chiuso *et al.* [36]. Most laser-based approaches employ a similar architecture to tackle the indoor UAV location. Instead of visual signals, they rely on laser beams to estimate the location of the UAV. Nevertheless, the SLAM technologies are easily challenged with issues related to the use and collection of feature points and their inability to provide stable and highly accurate localization in a complex indoor environment.

The INS is a navigation system that uses the Inertial Measurement Unit (IMU) to track the speed, position, and orientation of a device. With the development of Microelectromechanical systems (MEMS) technology, researchers can equip a small and low-cost IMU on a UAV, while many modern mini-UAVs have integrated the IMU internally. However, because of unavoidable, inherent hardware error and the error accumulated during the drift, accuracy will decrease after the UAV has been flying for a certain period of time [37].

With the astonishing growth of wireless systems and applications, many researches now focus on RF-based indoor localization [38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]. The basic concepts of indoor wireless localization are suitable for UAVs. Up to now, many systems based on received signal strength indicator (RSSI) and Ultra-wideband (UWB) were designed for localizing UAV in GPS-denied indoor environments [51, 52]. RSSI is a cheap and efficient way to measure distance and position, but its accuracy is unsatisfactory. In [53], the authors presented an algorithm based on measurements of the distance between a UAV and the existing infrastructure consisting of Wi-Fi Access Points (APs). With known locations of APs, a UAV is deployed to collect RSSI from the Wi-Fi APs during flight, transform these measurements into distances, and draw the flight trajectory with 33 points. The ultra-large bandwidth of UWB technology enables highly precise time measurements using Time Difference of Arrival (TDOA). Tiemann *et al.* proposed a cooperative UWB positioning system that enabled autonomous flying of a UAV [54]. This state-of-the-art UAV indoor localization work integrated UWB and INS technology. For example, Li *et al.* utilized UWB, INS, and 3D laser scanner data fusion, which is based on a Kalman filter, to achieve precise UAV indoor localization [55].

Meanwhile, radio-frequency identification (RFID) technology, especially the passive UHF RFID, has been widely deployed in retail environments [56]. RFID was developed as a cost-effective wireless technique for item serialization and has been widely recognized as a promising solution for indoor localization [57, 58, 59, 60, 61, 62, 63, 64]. Due to lightweight and low-cost RFID tags, RFID technology offers a promising method for UAV indoor localization. Choi *et al.* first proposed the concept of using passive RFID tags [65] for indoor UAV localization in [66]. However, they only demonstrated their concept and design of the system but lacked experimental validation. Recently, RFID technology was used in 3D reconstruction. For

example, Bu *et al.* presented a new theory based on the phase difference of RFID tags for 3D reconstruction of standard cubes [67]. Most existing RFID-based 3D reconstruction methods adopt an architecture of finding optimized results among multiple potential poses [67, 68, 69], which limits their applicability for UAV indoor localization, where six degrees of freedom (6-DoF) poses are needed in real-time.

In this chapter, we present the **RFUAV** – a low-cost **RF**ID based system to localize a **UAV** and enable it to autonomously navigate in complex indoor environments, such as warehouses, retail stores, hazmat storage facilities, and factories. Usually, such environments are crowded with racks, shelves, furniture, and other items of various sizes and layouts. With the increase in popularity of UAVs, there has come an increased concern with UAVs and public safety, leading to a compelling need for accurately locate an UAV in such 3-D indoor space [70, 71]. The proposed RFUAV will be deployed in an indoor environment to maintain the UAV's precise positioning, prevent collisions with other objects, and, hence, reduce the safety risks while flying in target environments. Our idea for RFUAV was motivated by existing RFID-based 3D reconstruction work [61]. However, the proposed method can provide precise 6-DoF poses, including both position and orientation in a 3D space, in real-time. In RFUAV, N ($N \geq 3$) UHF passive RFID tags are attached to a UAV, the position of each tag against the built-in coordinate of the UAV is measured first. This position is denoted as a *local* position. Then, a COTS (Commercial Off-The-Shelf) RFID reader with multiple antennas is deployed to collect observations of the tags. Based on the phase measurement of each RFID tag's response at multiple antennas, we can precisely track the position of the tags in the global coordinates of the 3D space. We denote this position as a *global* position. With the known local position of each tag and the global position of the N tags, the 6-DoF pose of the UAV is determined. Note that the reader and antennas are installed on the ground and powered from the target environment, while only the passive UHF RFID tags are attached to the UAV in the proposed scheme. Thus, the RFUAV system does not incur any extra power consumption to the UAV. Furthermore, the RFID infrastructure is already deployed in most of our target environments and the proposed system can be seamlessly integrated without much extra financial investment. The remainder of this chapter is organized as follows. The preliminaries are discussed in Section 2.1. We

present the proposed approach and the analysis of the RFUAV system in Section 2.2 and our experimental study in Section 2.3. Section 2.4 concludes this chapter.

2.1 Preliminaries

2.1.1 Phase Model for an UHF RFID System

To interrogate RFID tags, continuous-wave (CW) signals are transmitted by an RFID reader. The phase value of a tag response measured by the reader describes the phase difference between the transmitted signal and the corresponding received signal, which ranges from $-\pi$ to π. Nowadays phase values can be read by many commercial RFID readers, such as Impinj R420 and Zebra FX7500. Specifically, the phase value depends on the spatial distance between the tag and the reader's antenna. Letting d denote the tag-antenna distance, the measured phase value θ can be expressed as:

$$\theta = \left(2\pi \left(\frac{2d}{\lambda}\right) + \theta' + \theta_{noise}\right) \bmod 2\pi, \tag{2.1}$$

where λ is the wavelength of the channel, *mod* represents the modulo operation, and θ_{noise} is the phase offset caused by thermal noise and is a normal random variable. θ' is the phase offset caused by the reader's transmit/receive circuits and the tag circuits, which is expressed as:

$$\theta' = \theta_T + \theta_R + \theta_{TAG}, \tag{2.2}$$

where θ_T, θ_R, θ_{TAG} are the RF phase rotation caused by the reader's transmit circuits, the reader's receive circuits, and the tag's reflection characteristics, respectively.

Even though θ' is unknown, it dependents on the given hardware and is quite stable over time. We first conducted a benchmark experiment, as follows, to demonstrate that the phase offset θ' is stable while the tag moves throughout the environment. As the tag moves within the detectable range of the antenna, the measurement θ and the associated distance d to the antenna are recorded at several positions. According to (2.1), the theoretical phase is calculated as $\theta_T = 2\pi \left(\frac{2d}{\lambda}\right)$. The phase remainder is determined by $\theta' + \theta_{noise} = unwrap(\theta) - \theta_T$, where

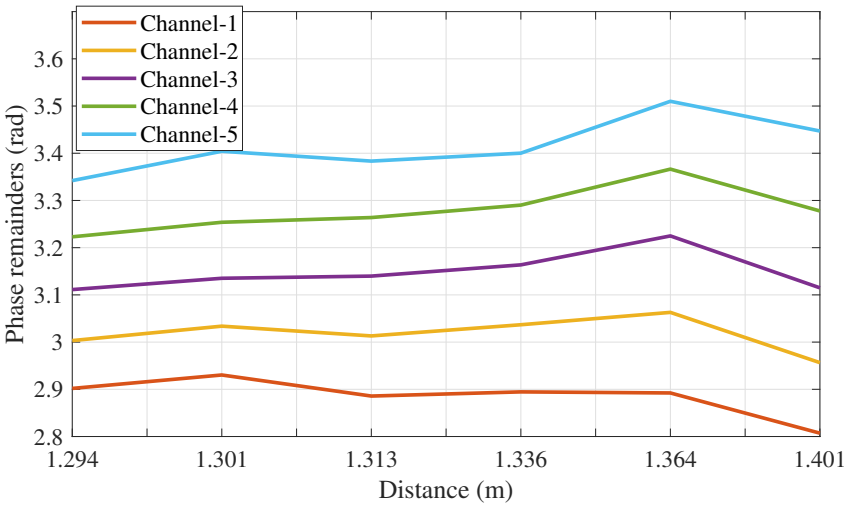


Figure 2.1: Phase remainders ($\theta' + \theta_{noise}$) at 6 sampling positions.

$unwrap(\cdot)$ adjusts the radian phases by adding multiple $\pm 2\pi$ to remove the phase discontinuities introduced by the round operation, while the distance between the tag and the antenna continuously increases Fig. 2.1 shows the experimental results of phase remainders that are measured at 6 positions when the tag moves. It shows that the phase remainders of 5 channels at the 6 locations are quite stable. The maximum phase remainder change occurs on channel-5, which is 0.15 rad. Considering that θ_{noise} is about 0.1 rad, θ' remains quite stable as the tag moves.

Therefore, θ' could be easily removed from the phase observation. As shown in Fig. 2.2, the measured phase value repeats from $-\pi$ to π with a period of a half wavelength. Within a half wavelength and for a given channel, the phase value exhibits a linear relationship with the distance between the tag and the reader's antenna.

2.1.2 Coordinates of the UAV

In general, the UAV is flying in a three-dimensional environment which has six degrees of freedom (6-DoF). Three translational degrees of freedom (DoFs) represent the *position* and three rotational DoFs represent the *orientation*. A pose of a UAV is depicted by the combination of position and orientation. A common means to describe the pose of a UAV is to attach it to a frame coordinate system. After a frame coordinate system is defined, the pose can be described by the origin and orientations of the axes of the frame. A pose $\mathbf{P}$ referring to given frame

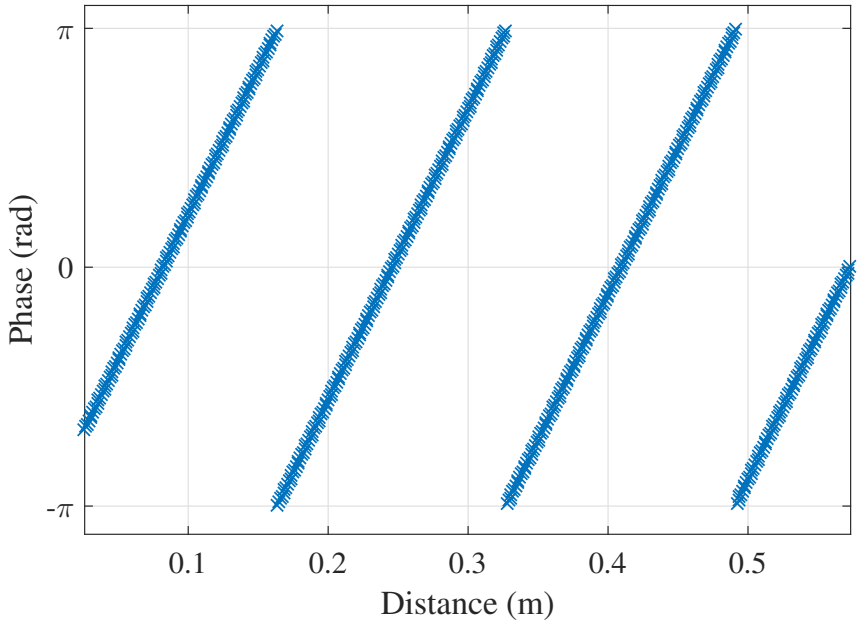


Figure 2.2: The linear relationship between the phase of RFID tag response and tag-antenna distance on a given channel.

coordinates $\mathcal{A}$ can be denoted as:

$$\mathbf{P} = [\mathbf{T}, \mathbf{R}]^T, \tag{2.3}$$

where $\mathbf{T}$ and $\mathbf{R}$ are the position and orientation against frame $\mathcal{A}$, respectively, and $(\cdot)^T$ is the transpose operation.

In our proposed RFUAV system, two frame coordinates are created to depict the UAV position. The first one is called the global frame coordinate system, which is denoted by g, representing the experimental environment coordinates. The second one is the UAV's built-in frame coordinate system, which is denoted by c, representing the UAV reference coordinate system. To translate a position from the UAV's built-in coordinates to the global coordinates, a rigid transformation, ${}^g_c\mathbf{T}$, is calculated and consists of a translation matrix, $\mathbf{T}$, and a rotation matrix, $\mathbf{R}$. Here, $\mathbf{T}$ and $\mathbf{R}$ are the position and orientation of the UAV against the global coordinates, which are given in (2.3). When we obtain a position $\bar{\mathbf{l}}_n = (\bar{x}_n, \bar{y}_n, \bar{z}_n)^T$ in the UAV's built-in coordinate system, the corresponding global position $\mathbf{l}_n = (x_n, y_n, z_n)^T$ can be

derived by:

$$\mathbf{l}_n = \mathbf{R} \cdot \bar{\mathbf{l}}_n + \mathbf{T}. \tag{2.4}$$

The translation matrix $\mathbf{T}$ describes the position shift of the UAV's built-in frame c with reference to the global frame g. Let $[o_x^g\ , o_y^g,\ o_z^g]$ and $[o_x^c,\ o_y^c,\ o_z^c]$ be the origin point positions of frame g and c, respectively. Then $\mathbf{T}$ can be expressed as:

$$\mathbf{T} = [o_x^g - o_x^c, o_y^g - o_y^c, o_z^g - o_z^c]^T. \tag{2.5}$$

Rotation matrix $\mathbf{R}$ describes the relative relationship of orientations between the two frames. The orientation of the UAV's built-in frame c with a reference to the global frame g is expressed by $\mathbf{R}$ as:

$$\mathbf{R} = \begin{bmatrix} \hat{X}_c \cdot \hat{X}_g & \hat{Y}_c \cdot \hat{X}_g & \hat{Z}_c \cdot \hat{X}_g \\ \hat{X}_c \cdot \hat{Y}_g & \hat{Y}_c \cdot \hat{Y}_g & \hat{Z}_c \cdot \hat{Y}_g \\ \hat{X}_c \cdot \hat{Z}_g & \hat{Y}_c \cdot \hat{Z}_g & \hat{Z}_c \cdot \hat{Z}_g \end{bmatrix}, \tag{2.6}$$

where $\hat{X}_c$, $\hat{Y}_c$, and $\hat{Z}_c$ are the unit vectors of the axes of the UAV's built-in coordinate frame, and $\hat{X}_g$, $\hat{Y}_g$, and $\hat{Z}_g$ are the unit vectors of the axes of the global coordinate system. The relationship between the two coordinate systems is illustrated in Fig. 2.3.

2.2 The Proposed Approach and Analysis

The RFUAV system is a low-cost, RFID-based system that enables a UAV to autonomously navigate in a complex indoor environment. This is achieved by providing precise 6-DoF poses of the UAV in a 3D space, which includes both position and orientation. Specifically, we attach N ($N \geq 3$) UHF passive RFID tags to a UAV. Through tracking the RFID tags with phase measurements, we can estimate the 6-DoF pose of the UAV. In this section, we introduce the system model, architecture, and analysis of RFUAV.

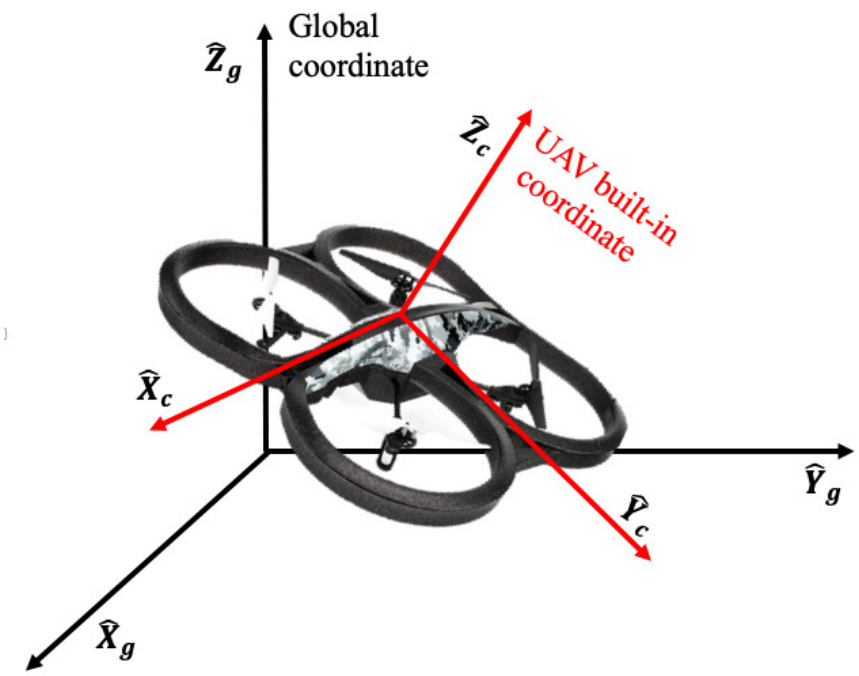


Figure 2.3: The global coordinate system and the UAV's built-in local coordinate system.

2.2.1 System Architecture

The architecture of RFUAV is illustrated in Fig. 2.4. The proposed system consists of two main components.

- **RFID tracker**: Based on phase measurements from the reader, a Bayesian filter is used to track the position of the tags against the global coordinates in a 3D space.
- **Pose estimator**: After the RFID tracker provides the global position of N ($N \geq 3$) tags, with the known local position of the tags compared to the UAV's built-in coordinates, the pose of the UAV can be estimated by an SVD-based algorithm.

2.2.2 RFID Tracker

In the RFUAV system, an RFID reader with M antennas is deployed to obtain phase measurements from responses of the attached tags. The positions of all antennas are already known. Hereafter, we use h_m^g to denote the position of the mth antenna in the global coordinate.

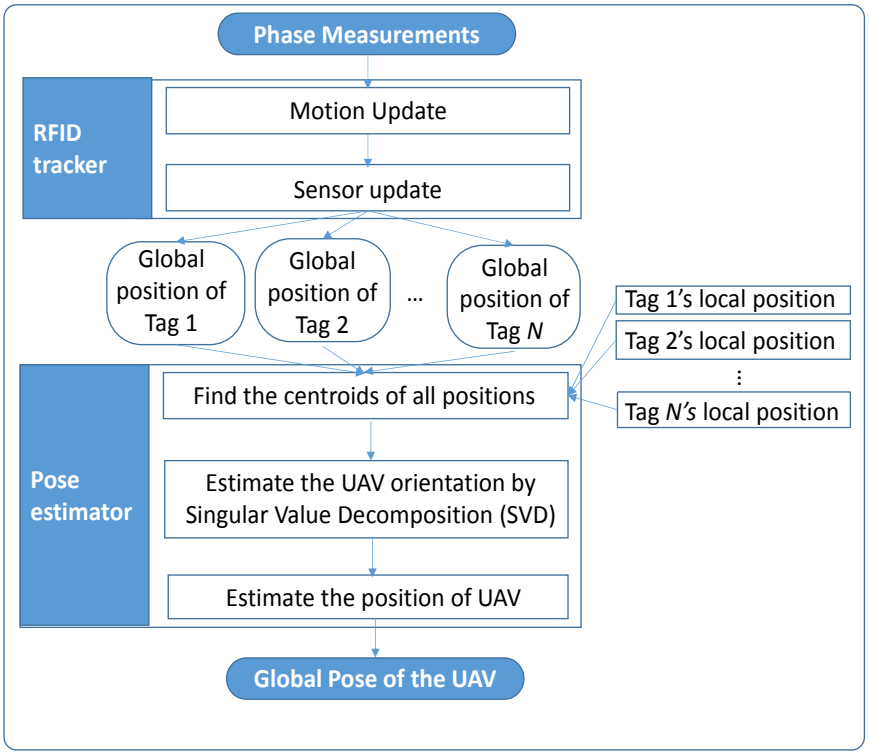


Figure 2.4: The system architecture of RFUAV, including the RFID tracker and pose estimator.

Bayesian Filter Updates for Tag Tracking

The RFID tracker utilizes a Bayesian filter to estimate (or track) tag locations. The Bayesian filter addresses the problem of estimating belief over the hypothetical posterior state l of a dynamic system from sensor observations. Here, state l denotes the location of the tag in the global coordinate system. The Bayesian filter recursively updates the belief $bel\left(l_t\right)$, which denotes the probability of the system in state l at time t. The $bel\left(l_t\right)$ is calculated from control u_t, observation z_t, and prior belief $bel\left(l_{t-1}\right)$ at time $t-1$, which is calculated previously.

There are two essential steps for the updating cycle of a typical Bayesian filter. The first step is called control update or prediction, which is given by:

$$\overline{bel}\left(l_t\right) = \int P\left(l_t|u_t, l_{t-1}\right) bel\left(l_{t-1}\right) dl_{t-1}, \tag{2.7}$$

where $P\left(l_t|u_t, l_{t-1}\right)$ is a motion model and provides the probability for a tag to move from state l_{t-1} to l_t after control u_t is applied, and $\overline{bel}\left(l_t\right)$ denotes the state probability distribution of the tag after control u_t is applied. We deploy a constant speed mobility model for the RFID tracker,

i.e., we assume that in a very short time interval, the speed of a tag will remain constant. We do not assume that the tag moves at a constant velocity over all time, but rather that it maintains an average speed, u_t, with an undetermined and negligible amount of acceleration within a short time frame. Its movement can be described mathematically by a Gaussian distribution as [29]:

$$P\left(l_t|u_t, l_{t-1}\right) = \frac{1}{\sqrt{2\pi}\xi} \int_0^{\Delta t} e^{-\frac{\left(l_t-\left(l_{t-1}+u_t\cdot\tau\right)\right)^2}{2\xi^2}} d\tau, \tag{2.8}$$

where u_t is the speed of the item at time t (i.e., the control), Δt is the time interval between $t-1$ and t, and ξ^2 is the variance to model the movement of the item satisfying a typical Gaussian distribution.

A commercial RFID reader can interrogate tags at a rate of about 500Hz. To demonstrate this, we provide an experimental setting of a reader connected with 3 antennas to read 10 tags in an environment where there are hundreds of tags. Fig. 2.5 shows how many times each of the 10 tags were read by three antennas within 60 seconds. This benchmark experiment was conducted in a mock apparel store, where hundreds of RFID tags were deployed in the environment. During the experiment, we enabled the filter function, which is available for most COTS readers, of the reader to only interrogate the 10 given tags. Each tag was read for about 3000 times (1000 times per antenna) in a period of 60 seconds. Thus, each tag can be interrogated by the reader for about 50 times per second. Considering N ($10 > N \geq 3$) tags will be attached to the UAV, the practical reading frequency should be larger than 50Hz. Thus, the interval of two continuous observations of the tag is about $10 \sim 20$ milliseconds. In such a short period, our constant speed model in (2.8) should be suitable, since for indoor deployment of UAVs, the speed is usually lower than that in outdoor applications. Therefore, for practical applications, we can program the filter function to ensure the assumption of (2.8) be maintained.

The second step is measurement update, which is given by:

$$bel\left(l_t\right) = \eta \cdot \overline{bel}\left(l_t\right) \cdot P\left(z_t|l_t\right). \tag{2.9}$$

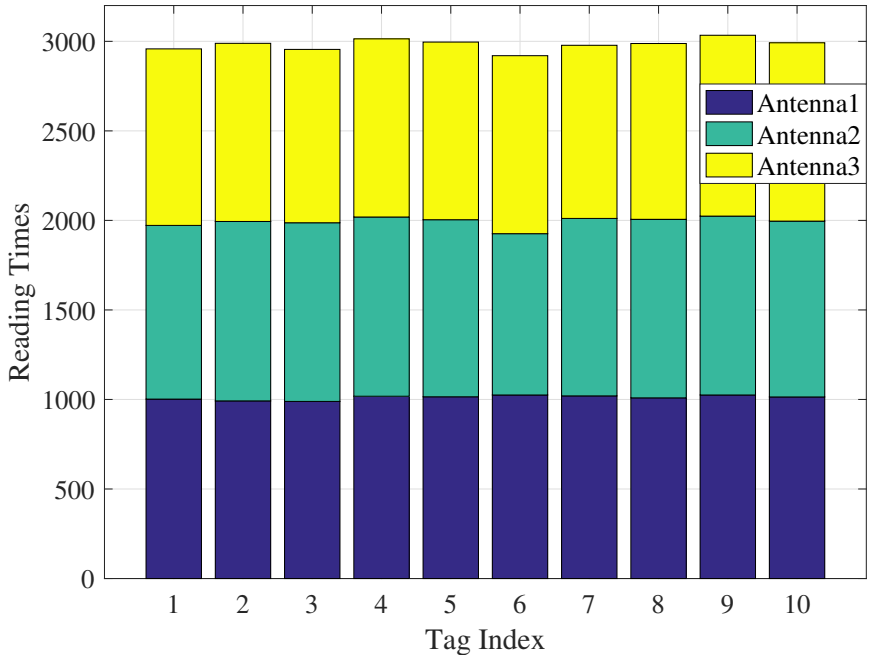


Figure 2.5: Reading occurrences of 10 tags by 3 antennas in a period of 60 seconds.

In (2.9), η is a constant that helps to normalize the sum of all $bel\,(l_t)$ to 1, and $P\,(z_t|\,l_t)$ is called the *observation model*. In the RFID tracker, M reader antennas are deployed. Therefore, $P\,(z_t\,|\,l_t)$ is given by:

$$P\,(z_t|\,l_t) = \prod_{m=1}^{M} P\,(z_t|l_t, h_m^g). \tag{2.10}$$

We can thus rewrite (2.9) as:

$$bel\,(l_t) = \prod_{m=1}^{M} \eta \cdot \overline{bel}\,(l_t) \cdot P\,(z_t|l_t, h_m^g). \tag{2.11}$$

where $P\,(z_t|\,l_t, h_m^g)$ is the observation model for the mth antenna, which provides the probability for the tag to be located in position l_t, and the measurement of z_t is observed by the mth antenna that is in a known position h_m^g. More details of the observation model are given in the next subsection.

The Observation Model of RFID Phase Measurement

The relationship of the RF phase shift between the sent and received signals is given by (2.1) and (2.2). The experimental results in Fig. 2.1 show that θ_T, θ_R, θ_{TAG} are relatively constant when the reader antenna, the RFID tag, and the radio frequency are fixed. When we consider the RF phase for the same antenna, the same RFID tag, and under the same RF frequency, and ignore θ_{noise},[1] (2.1) can be rewritten as:

$$\theta = \left(2\pi \cdot \left(\frac{2d}{\lambda}\right) + \theta'\right) \bmod 2\pi. \tag{2.12}$$

Assume a reader antenna is set in position h_m^g, a tag is in position l_{t-1}, and the RF phase θ_1 for the tag is observed. When the tag moves to position l_t, it generates an RF phase θ_2 under the same frequency. The differential RF phase between these two positions satisfies the following relationship.

$$\Delta\theta_{12} = (\theta_1 - \ \theta_2) \bmod 2\pi \tag{2.13}$$

$$\Delta\theta_{12} = \left(\left(2\pi\left(\frac{2\,|l_{t-1}, h_m^g|}{\lambda}\right) + \theta'\right) \bmod 2\pi - \left(2\pi\left(\frac{2\,|l_t, h_m^g|}{\lambda}\right) + \theta'\right) \bmod 2\pi\right) \bmod 2\pi \tag{2.14}$$

$$\Delta\theta_{12} = \left(\frac{4\pi}{\lambda} \cdot (|l_{t-1}, h_m^g| - \ |l_t, h_m^g|)\right) \bmod 2\pi, \tag{2.15}$$

where $|\cdot,\cdot|$ measures the Euclidean distance between two positions. Equation (2.15) shows that the differential RF phase, under the same frequency, the same antenna, and the same RFID tag, can be determined by the difference of distances when the tag moves from one position to another. In other words, $\Delta\theta_{12}$ is not affected by the constant phase offset θ' and is only related to the distance between the two positions. Hereafter, we assume that all the RF phases are measured by the same reader antenna for the same RFID tag at the same RF frequency.

[1] The modeling of θ_{noise} will be introduced later.

The antenna of the RFID reader is stationary in a known position h_m^g. The tag will be located in a series of positions denoted as $\{l_1, l_2, \ldots, l_t\}$, and the corresponding phase shifts for these positions are $\{\theta_1, \theta_2, \ldots, \theta_t\}$. It follows that

$$\begin{cases} |l_i, h_m^g| - |l_j, h_m^g| = \frac{\lambda}{4\pi} \cdot \Delta\theta_{ij} + n \cdot \frac{\lambda}{2} \\ \Delta\theta_{ij} = (\theta_i - \theta_j) \bmod 2\pi, \end{cases} \tag{2.16}$$
$$n = \{1, 2, ...\}, i, j \, \epsilon \{1, 2, ..., t\} \text{ and } i \neq j.$$

We next update the observation model $P(z_t | l_t, h_m^g)$ by (2.16), which gives the probability that a tag moves from l_t to l_{t-1} to achieve the differential RF phase shift $\Delta\theta_{t,t-1}$. The model of differential RF phase is given by:

$$P(\Delta\theta_{t,t-1} | l_{t-1}, l_t, h_m^g) = \begin{cases} 1, & \text{if (2.16) is satisfied} \\ 0, & \text{otherwise.} \end{cases} \tag{2.17}$$

The RF phase is measured by the reader antenna, and usually it is distorted by thermal noise, denoted by θ_{noise} in (2.1). Experiments reveal that θ_{noise} satisfies a typical Gaussian distribution. Therefore, the RF phase containing this random error can be modeled as $\theta \sim \mathcal{N}(\mu, \delta^2)$, where μ is the mean of the RF phase without thermal noise and δ^2 is the variance. It follows that the phase difference, as the difference of two Gaussian random variables, is also Gaussian as $\Delta\theta_{ij} \sim \mathcal{N}(\mu_i - \mu_j, 2\delta^2)$. Incorporating the thermal noise to (2.17), we have

$$P(\Delta\theta_{t,t-1} | l_{t-1}, l_t, h_m^g) = \frac{1}{\sqrt{2\pi}\delta} \int_0^{\Delta\theta_{t,t-1}} e^{-\frac{(y-(\mu_t - \mu_{t-1}))^2}{2\delta^2}} dy, \tag{2.18}$$

where

$$\mu_t - \mu_{t-1} = \left(\frac{2|l_t, h_m^g|}{\lambda} - \frac{2|l_{t-1}, h_m^g|}{\lambda} \right) \bmod 2\pi. \tag{2.19}$$

To consider thermal noise when estimating the location of RFID tags by (2.11), we can use (2.18) instead of (2.17).

2.2.3 Pose Estimator

The RFID tracker provides the tag position in the global coordinate system. We use $\mathbf{l}_n^t = (x_n^t, y_n^t, z_n^t)^T$ to denote the global position of tag n at time t. With the UAV be located at $\mathbf{T}_t$, with orientation $\mathbf{R}_t$ in the given global coordinate system, while $\mathbf{T}_t$ and $\mathbf{R}_t$ together provide the pose of the UAV at time t. The position of each attached tag in the UAV built-in coordinate is known and fixed. We indicate this local position for the nth tag as $\bar{\mathbf{l}}_n = (\bar{x}_n, \bar{y}_n, \bar{z}_n)^T$. The relationship between local and global positions is given by (2.4), which can be rewritten as:

$$\mathbf{l}_n^t = \mathbf{R}_t \cdot \bar{\mathbf{l}}_n + \mathbf{T}_t, \tag{2.20}$$

where $\mathbf{R}_t$ and $\mathbf{T}_t$ are the orientation and position of the UAV in the global coordinate system at time t, respectively; and $\mathbf{l}_n^t$ and $\bar{\mathbf{l}}_n$ are the locations of the nth tag in the global coordinate system and the UAV built-in coordinate system, respectively.

When the RFID tracker localizes three or more tags simultaneously, we can use (2.20) to obtain an optimal transformation ${}_c^g\mathbf{T}_t$, which consists of $\mathbf{R}_t$ and $\mathbf{T}_t$. The method to solve (2.20) will be introduced later in this section. In practice, the RFID reader cannot query multiple tags simultaneously. However, we can assume the three consecutive queries happen at the same time. This assumption is reasonable for most indoor UAV applications. Unlike the scenario of moving rigid body localization discussed in [72], the UAV usually moves at a much lower speed (e.g., 1 m/s) in an indoor environment. As discussed previously, the current RFID reader can conduct 500 queries per second, which is only 2 ms per query. In such a short time period the displacement of the UAV is only about several millimeters (e.g., 2 mm when a UAV moves at 1 m/s) and can be ignored. Therefore, when the N tags are located by the RFID tracker, it

follows (2.20) that

$$\begin{cases} \mathbf{l}_1^t = \mathbf{R}_t \cdot \bar{\mathbf{l}}_1 + \mathbf{T}_t \\ \mathbf{l}_2^t = \mathbf{R}_t \cdot \bar{\mathbf{l}}_2 + \mathbf{T}_t \\ \quad \cdots \\ \mathbf{l}_N^t = \mathbf{R}_t \cdot \bar{\mathbf{l}}_N + \mathbf{T}_t, \end{cases} \tag{2.21}$$

where $\mathbf{l}_1^t$, $\mathbf{l}_2^t$, and $\mathbf{l}_3^t$ are the global positions for tags 1, 2, and 3, respectively; and $\bar{\mathbf{l}}_1$, $\bar{\mathbf{l}}_2$, and $\bar{\mathbf{l}}_3$ are the local positions (measured in the built-in coordinate of the UAV) for tags 1, 2, and 3, respectively. The goal is to find the optimal transform ${}_c^g\mathbf{T}_t$, which includes rotation $\mathbf{R}_t$ and translation $\mathbf{T}_t$, between two sets of corresponding 3D data points. The task can be formulated as a least squares minimization problem as:

$$\min_{\{\mathbf{R}_t, \mathbf{T}_t\}} \sum_{i=1}^{N} ||\mathbf{l}_i^t - (\mathbf{R}_t \cdot \bar{\mathbf{l}}_i + \mathbf{T}_t)||, \tag{2.22}$$

where $||\cdot||$ is the norm of a vector. Determining the rotation and translation relationship between two sets of data points at different coordinates is a typical problem in pattern analysis [73, 74]. Based on the method introduced in [73], the proposed pose estimator is developed to find the optimal ${}_c^g\mathbf{T}_t$ with the following procedure:

1. Find the centroids of all the positions in both the global coordinate system and the UAV's built-in coordinate system.

2. Use the centroids as the new origin of the two coordinate systems, and transforming the positions into these two coordinates. Then based on these transformed positions to find the optimal rotation $\mathbf{R}_t$ with the singular value decomposition (SVD) method.

3. Solve for the translation $\mathbf{T}_t$ using rotation $\mathbf{R}_t$.

In Step 1, the centroids are computed as:

$$\begin{cases} \mathbf{O}_g = \frac{1}{N} \sum_{i=1}^{N} \mathbf{l}_n^t \\ \mathbf{O}_c = \frac{1}{N} \sum_{i=1}^{N} \bar{\mathbf{l}}_n, \end{cases} \tag{2.23}$$

where $\mathbf{O}_g$ and $\mathbf{O}_c$ are the centroids of all the positions in the global and the UAV's built-in coordinate systems, respectively.

In Step 2, we use $\mathbf{O}_g$ and $\mathbf{O}_c$ as new origins to shift the global and UAV's built-in coordinates to create two new coordinate systems, which are called the *shifted* global and *shifted* UAV's built-in coordinate systems, respectively. The positions in these two coordinates, which are denoted as $\bar{\mathbf{P}}_g^i$ and $\bar{\mathbf{P}}_c^i$, are given by:

$$\begin{cases} \bar{\mathbf{P}}_g^i = \mathbf{l}_i^t - \mathbf{O}_g \\ \bar{\mathbf{P}}_c^i = \bar{\mathbf{l}}_i - \mathbf{O}_c, \end{cases} \text{ for } i \in [1, 2, ..., N]. \tag{2.24}$$

We then apply the SVD method to find the optimal rotation between the two sets of positions in the shifted global and shifted UAV built-in coordinate systems. First, we create a matrix $\mathbf{H}$, which is given by:

$$\mathbf{H} = \sum_{i=1}^{N} \bar{\mathbf{P}}_c^i \cdot (\bar{\mathbf{P}}_g^i)^T. \tag{2.25}$$

Note that the position in each coordinate system is 3-dimensional, and $\bar{\mathbf{P}}_g^i$ and $\bar{\mathbf{P}}_c^i$ are each represented by a 3×1 vector. Hence, the $\mathbf{H}$ given by (2.25) is a 3×3 matrix. We decompose or factorize matrix $\mathbf{H}$ by the SVD method as:

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = SVD(\mathbf{H}). \tag{2.26}$$

Then the optimal rotation $\mathbf{R}_t$ can be derived as:

$$\mathbf{R}_t = \mathbf{V} \cdot \mathbf{U}^T. \tag{2.27}$$

A special case must be considered here. When the determinant of $\mathbf{V}$ is -1, we must multiply the third column of $\mathbf{R}_t$ by -1 to obtain the correct rotation.

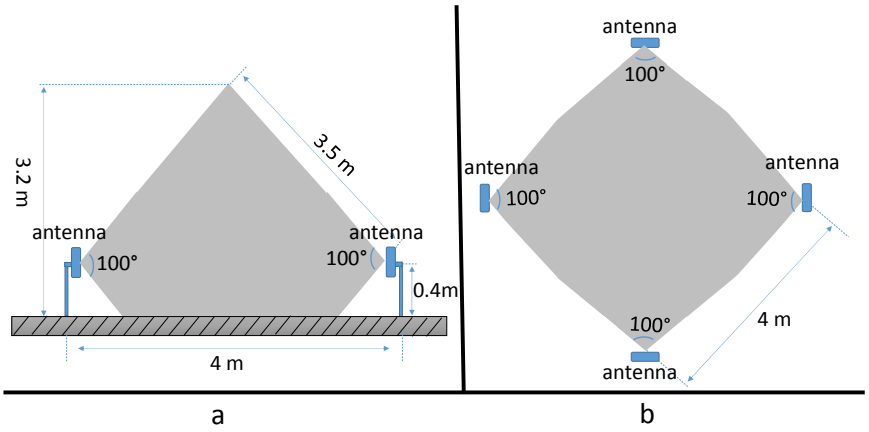


Figure 2.6: Antennas setup for the RFUAV prototype: (a) Side view of the RFID detectable field; (b) Top view of the RFID detectable field.

In Step 3, after obtaining the rotation $\mathbf{R}_t$, we can easily derive the translation $\mathbf{T}_t$ by the following equation.

$$\mathbf{T}_t = \mathbf{O}_g - \mathbf{R}_t \cdot \mathbf{O}_c. \tag{2.28}$$

Thus we derive the orientation $\mathbf{R}_t$ and position $\mathbf{T}_t$ of the UAV in the given global coordinate system.

2.3 Experimental Study and Discussions

2.3.1 Experiment Setup

To validate the performance of the RFUAV system, we conduct a set of experiments in a representative indoor environment at the RFID Laboratory of Auburn University, Auburn, AL. To build a prototype of RFUAV, we employ a Zebra FX7500 RFID reader and four Zebra AN720 Antennas to collect observation of the RFID tags. The entire RFID system operates in the $902 \sim 928$ MHz band, which is the frequency range allocated by Federal Communications Commission (FCC) in the USA. The Zebra FX7500 reader is one of the most widely used RFID products in the market. It is compatible with EPC Gen2 standard, and provides the Low-Level Reader Protocol (LLRP) through an Ethernet port to report the RFID readings. The reader interrogates the RFID tags and sends query reports that includes the information on EPC, RSSI, phase, time stamp and channel index. The Zebra AN720 Antennas provide a

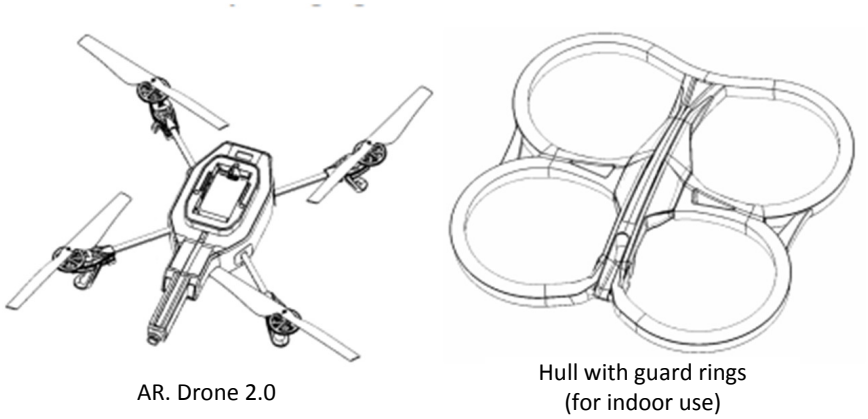


Figure 2.7: Illustration of the Parrot AR. Drone 2.0 schematic.

$5.5 \sim 6$ dB gain and a left circular polarization with 100° beamwidth. The size of each antenna is 132.8×132.8×18 mm^3. Each antenna is mounted on a holder of 0.4 m high. The four antennas with their holders are deployed at the corners of a square of 4×4 m^2. During our experiment, the reader is operated at the maximum RF transmission power of about 33 dBm. This allows the reading range of the antenna up to 6 m. Four antennas create a detectable field and can interrogate an RFID tag simultaneously. The configuration of our experiment is shown in Fig. 2.6.

The Parrot AR Drone2.0 Elite Edition, a low-cost platform with good maneuverability, is employed as our indoor UAV platform. It consists of a drone shell, hull, and battery, as shown in Fig. 2.7. A fully charged battery can support the UAV in continuous flight for 15 minutes. The AR Drone is equipped with a front and bottom camera, a sonar, and an IMU. With readings from these sensors, it can localize itself by a sensor fusion method, such as Parallel Tracking and Mapping (PTAM) [75] that estimates a 3D pose of the UAV in an unknown environment.

Three UHF passive RFID tags are attached to the UAV as illustrated in Fig. 2.8. Our experimental RFID tag is Smartrac Dogbone - Impinj Monza R6, which is widely used in the retail market. It is equipped with an Impinj Monza R6 chip that provides up to -22.1 dBm read wake-up sensitivity and up to -18.8 dBm write wake-up sensitivity. Our proposed RFUAV is not restricted to any specific tag layout, and a detailed experiment will be presented later to demonstrate the effect of various tag layouts. In our experiments, the Electronic Product Code (EPC) of each tag serves as its identity to consistently and accurately distinguish the received

Table 2.1: Eperiment Configuration and Parameters

Parameter	*Value*
RFID reader	Zebra FX7500 RFID reader
Antenna	Zebra AN720 Antennas
Number of antennas	4
Antenna frequency	$902 \sim 928$ MHz
Antenna gain	$5.5 \sim 6.0$ dB
Antenna beamwidth	100°
Antenna height	0.4 m
Transmission power	33 dBm
RFID tag	Smartrac Dogbone - Impinj Monza R6
Tag read sensitivity	-22.1 dBm
Tag write sensitivity	-18.8 dBm
UAV	Parrot AR Drone2.0 Elite Edition
UAV battery	15 min
Dynamic ground-truth positioning system	Ultra-Wideband (UWB) positioning system from PLUSLocation.LLC

readings from that of other RFID tags. During the experiments, to achieve accurate localization and orientation estimation, the initial position of tags in the three-dimensional global system and the UAV's built-in coordinate system are given. The configuration for the experiments reported in this section is summarized in Table 2.1.

To precisely collect the ground-truth for the poses and trajectories of the UAV, we design two experimental settings, a confined and a dynamic setup. In the confined setup, the UAV was mounted to an adjustable rolling rack, as illustrated in Fig. 2.9(a). The UAV-mounted rolling rack is easily maneuverable throughout our experimental field, and the height of the UAV can be adjusted from 0.8m to 1.6m. During the experiments, we manually moved the rolling rack instead of flying the UAV, as shown in Fig 2.9(b). In this setting, the ground truth of the moving trajectories can be represented by a set of discrete sample poses, including positions and orientations, which are precisely and manually measured while the rolling rack is at a sampling point. Considering the errors usually introduced by taking measurements

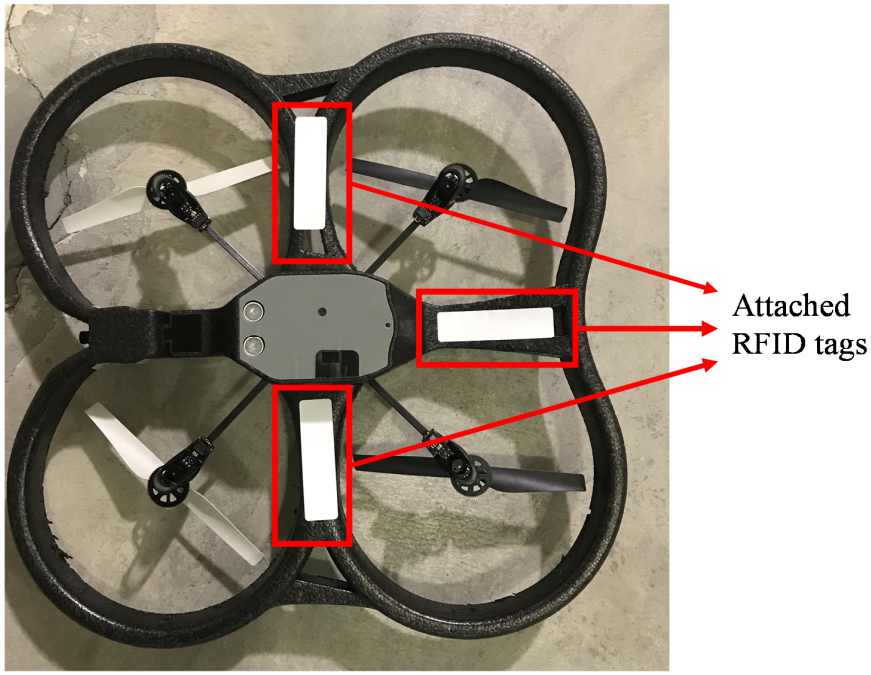


Figure 2.8: The Parrot AR Drone2.0 UAV with three attached RFID tags.

manually, these ground truth data can provide sub-centimeter accuracy. However, the confined setup enables us to provide extremely precise ground-truth trajectories and poses in a semi-static manner. To evaluate our RFUAV in a dynamic manner and obtain ground truth while it is flying, we designed the dynamic setup. An Ultra-Wideband (UWB) positioning system from PLUSLocation.LLC was installed to cover the entire space of the RFID Laboratory in the Auburn University campus. We attached a UWB tag to the UAV, which is shown in Fig 2.9(c), in such a way that while the UAV is flying, its positions can be read by the system in real-time. The localization accuracy of the UWB system in the experimental field was 3cm with a limited area of 4×4 m^2. Although we are able to track the UAV in a dynamic way in this setup, it provides us with position information but no orientation information, so the UAV's position accuracy is compromised. Therefore, we utilized the confined setup for quantitative experiments and the dynamic setup for qualitative experiments or experiments where the UAV must fly.

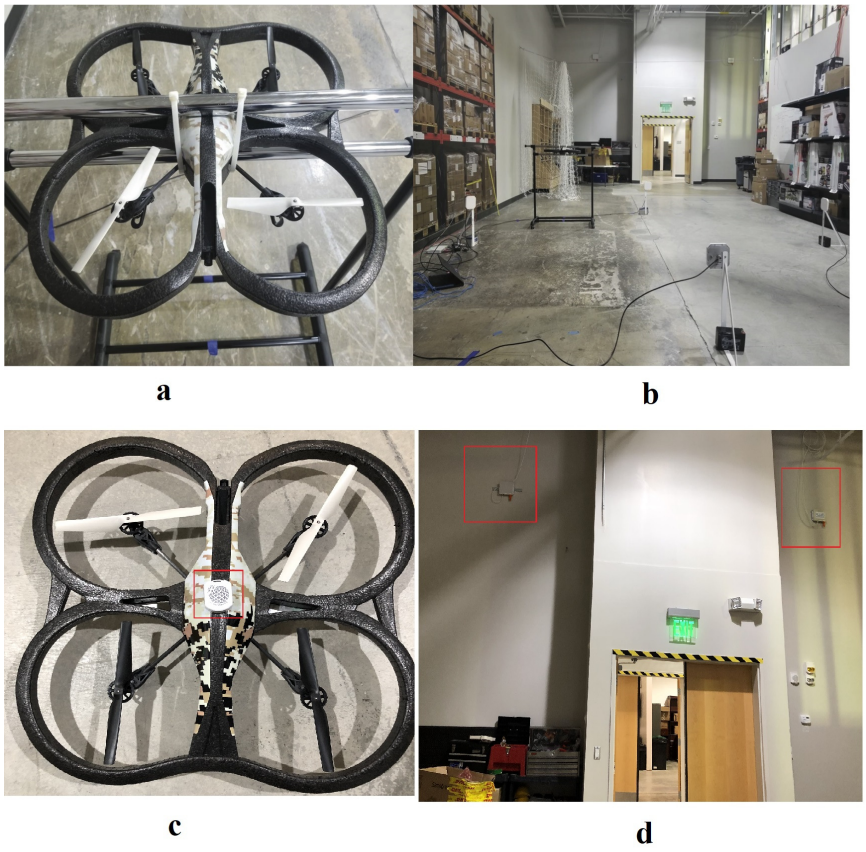


Figure 2.9: (a) UAV carried by a rolling rack in the confined setup, (b) The UAV confined rolling rack moves in the experimental field, (c) An UWB tag is attached to the UAV in dynamic setup, the UWB tag is marked by a red rectangle, (d) Two nodes that are marked in red of the UWB positioning system, there are 6 nodes are installed in the experimental field.

2.3.2 Accuracy of RFID Tag Tracking

We first launched an experiment to evaluate the performance of the RFID tracker of RFUAV by comparing its accuracy with that ofthe state-of-art approach Tagoram [61]. To guarantee the fairness of comparison, the same equipment is utilized for both approaches. The confined setup was used for this experiment and three UHF-passive RFID tags were attached to the UAV. These tags were moved around the experimental field by manually pushing the UAV-mounted rolling rack. The ground-truth positions of each tag were manually measured at the sampling positions of the trajectories. The moving trajectories were unknown to the proposed RFID tracker or to Tagoram. Therefore, the Tagoram functioned in uncontrollable mode where the trajectory function is unknown. Tracker performance was evaluated by assessing the amount of errors between the estimated and ground-truth positions of the sampled points.

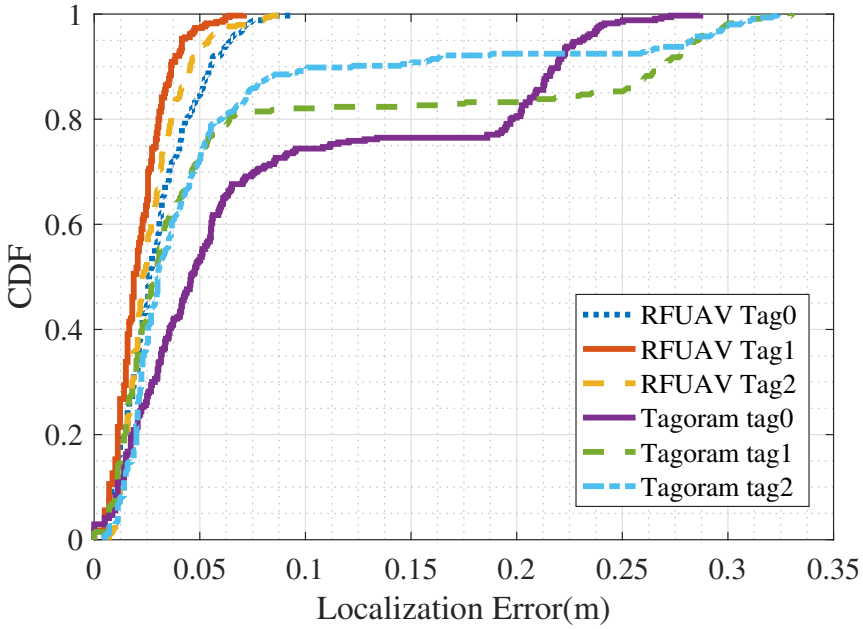


Figure 2.10: CDFs of multiple tags' localization errors for RFUAV and Tagoram.

The experiment results are presented in Fig. 2.10 and Fig. 2.11. As shown in Fig. 2.10, about 80% of all the localization errors for RFUAV are less than 0.04 m. In addition, the RFUAV maximum location error is 0.085 m, while the maximum error for Tagoram is 0.33 m. Obviously, the proposed RFID tracker of RFUAV is more suitable for a complex indoor environment. Fig. 2.11 presents the average localization error and standard deviation (see error bars) of RFUAV and Tagoram. The average localization errors of RFUAV are much less than those of Tagoram. Furthermore, the RFUAV's standard deviations are also much smaller, indicating more robust performance by our proposed scheme.

2.3.3 Accuracy of Pose Estimation

In this section, we investigate the pose accuracy of the RFUAV system by conducting a set of experiments to evaluate the impact of two important system configurations: the layout and number of attached tags on the UAV.

Effect of the Layout of Tags

The design of our proposed pose estimator allows for RFUAV to not be restricted by any specific tag layout. To demonstrate this advantage, we attached three tags in four representative

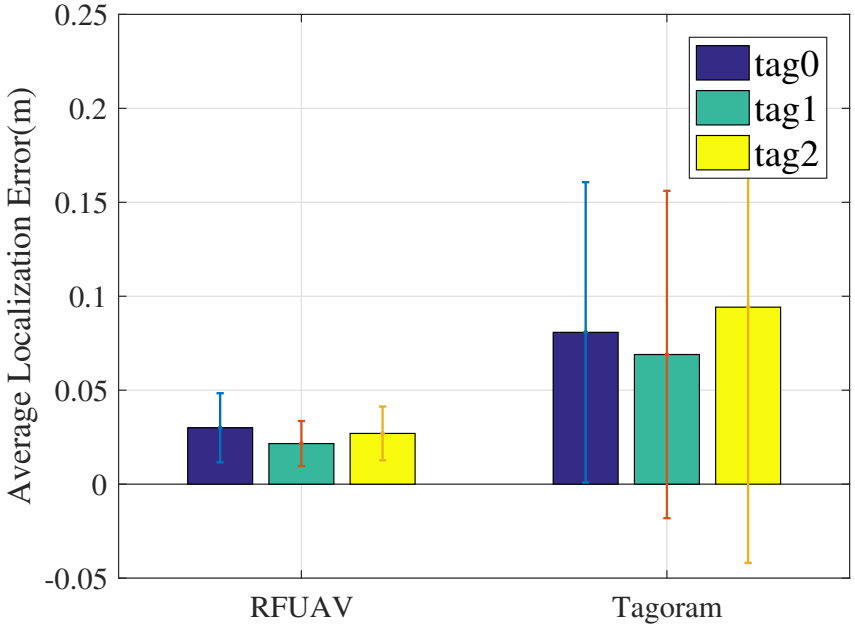


Figure 2.11: Average distance error and standard deviation of RFUAV and Tagoram.

layouts, which is illustrated in Fig. 2.12. In Layout 1 in Fig. 2.12(a) shows three tags arranged in a plane with two identical coordinate values against the UAV's built-in coordinate system. In Fig. 2.12(b), Layout 2 shows three tags arranged in a plane, but not on a straight line. For Layout 3 in Fig. 2.12(c), three tags are arranged in a plane, but on a straight line. As shown in Fig. 2.12(d), Layout 4 consists of three tags arranged neither in a plane nor on a straight line. This experiment was conducted in the confined setup, and we moved the UAV-mounted rolling rack throughout the environment in the same small-scale trajectory (with a length of 0.5 m) for each representative tag layout. Each trajectory is sampled in every 2cm, that is 25 points for every trajectory. We compared the pose accuracy for these four layouts, and those results are presented in Fig. 2.13.

Fig. 2.13 shows that the average position errors for the four layouts are 0.038 m, 0.016 m, 0.019 m, and 0.019 m, respectively. The average orientation errors for the four layouts are 56.4°, 2.0°, 2.3°, and 2.2°, respectively. Clearly, all layouts achieve a small error (less than 0.04 m) on positioning of the UAV. However, the orientation of Layout 1 yields a relatively greater error of 56.4°, due to the arrangement of tags being in an extremely adversarial layout; two coordinate values in the UAV's built-in coordinate system are identical. This layout is vulnerable to small turbulence of estimated global tag locations. It may also cause the estimated

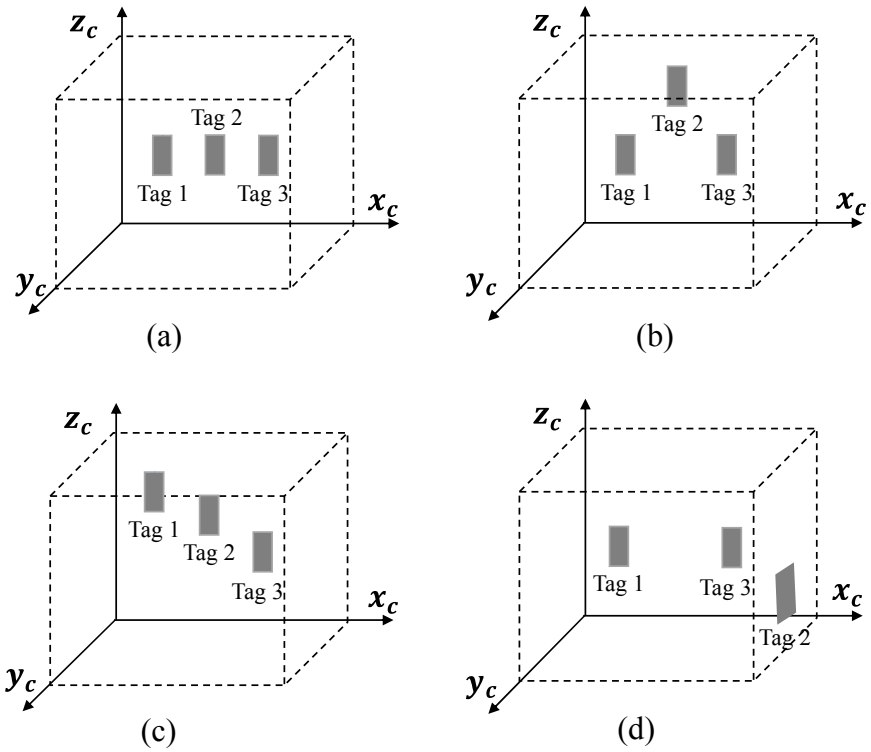


Figure 2.12: Four representative layouts of the attached tags in the UAV's built-in coordinate system: (a) Layout 1; (b) Layout 2; (c) Layout 3; (d) Layout 4.

orientation of the UAV to reverse. However, Fig. 2.13 shows that except for this extreme case, the other layouts of tags have an orientation error of less than 2.5° and do not affect the performance of RFUAV.

Effect of the Number of Tags

The RFUAV requires at least three tags to compute a 6-DoF pose in an indoor environment. We examined the effect of the number of attached tags on pose accuracy. We attached three, four, five, and six tags on the UAV in each experiment. According to the previous experiments, the layout of the tags does not affect the tracking precision (except for the extreme case). Based on the previous experiment and to guarantee a fair comparison in this experiment, Layout 4 is used. Fig. 2.15 illustrates the experimental layout. For each trial of the experiment, the same trajectory (a confined setup with a length of 0.5 m and 25 sampling points) is followed by the UAV. This experiment's results are presented in Fig. 2.14.

Fig. 2.14 presents the relationship between pose error and the number of tags that are used in RFUAV. From Fig. 2.14(a) we can see that the highest position error, 0.019 m, is achieved

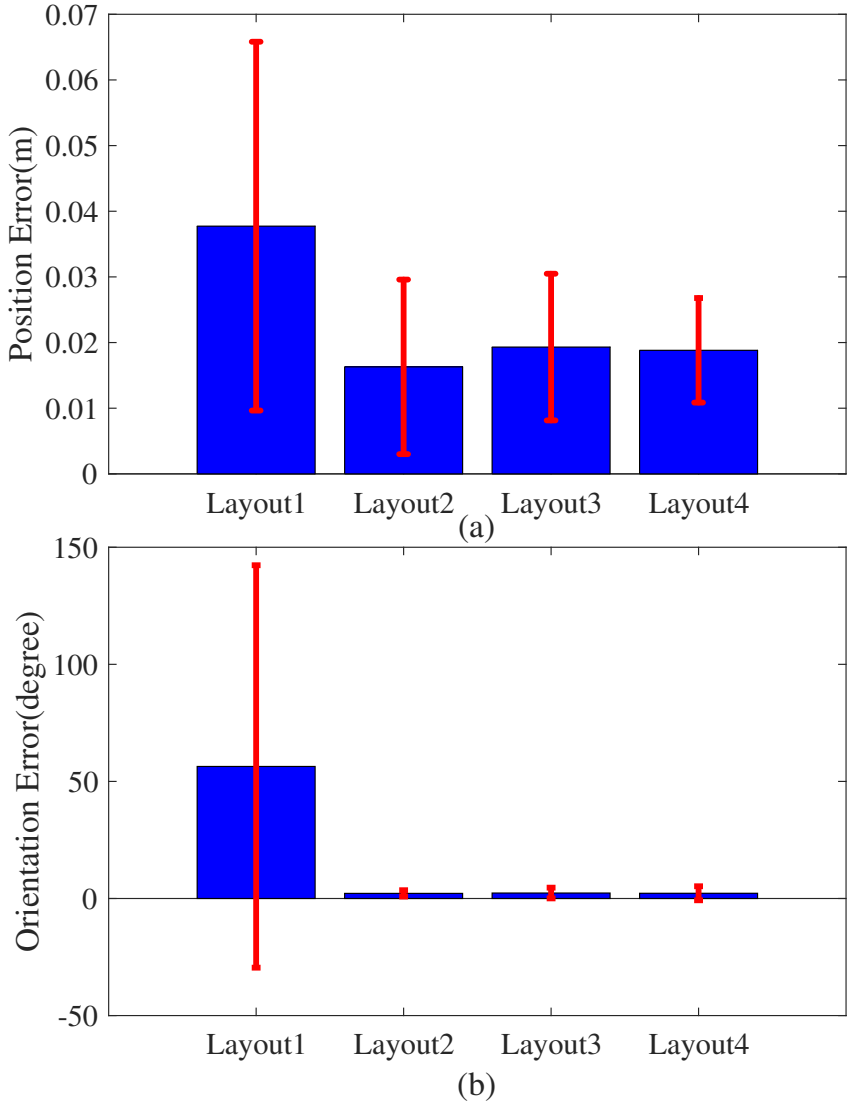


Figure 2.13: (a) Position errors of different layouts of attached RFID tags; (b) Orientation errors of different layouts of attached RFID tags.

when 3 tags are used in our system, while the lowest position error, 0.016 m, is achieved with 4 tags. The variation of the position errors among different sets of tags is less than 3 mm. Fig. 2.14(b) shows that the number of tags does not affect the orientation accuracy neither. All sets of tags provide a similar orientation error around 2°. Thus, it is safe to say that the RFUAV system does not exhibit an obvious difference in performance when different numbers of tags are deployed. For experiments herein, we attached three tags to the bottom of the UAV's hull, as shown in Fig. 2.8. Due to the uneven shape of the hull, the deployed layout is Layout 4 in Fig. 2.12(d), which we have discussed previously.

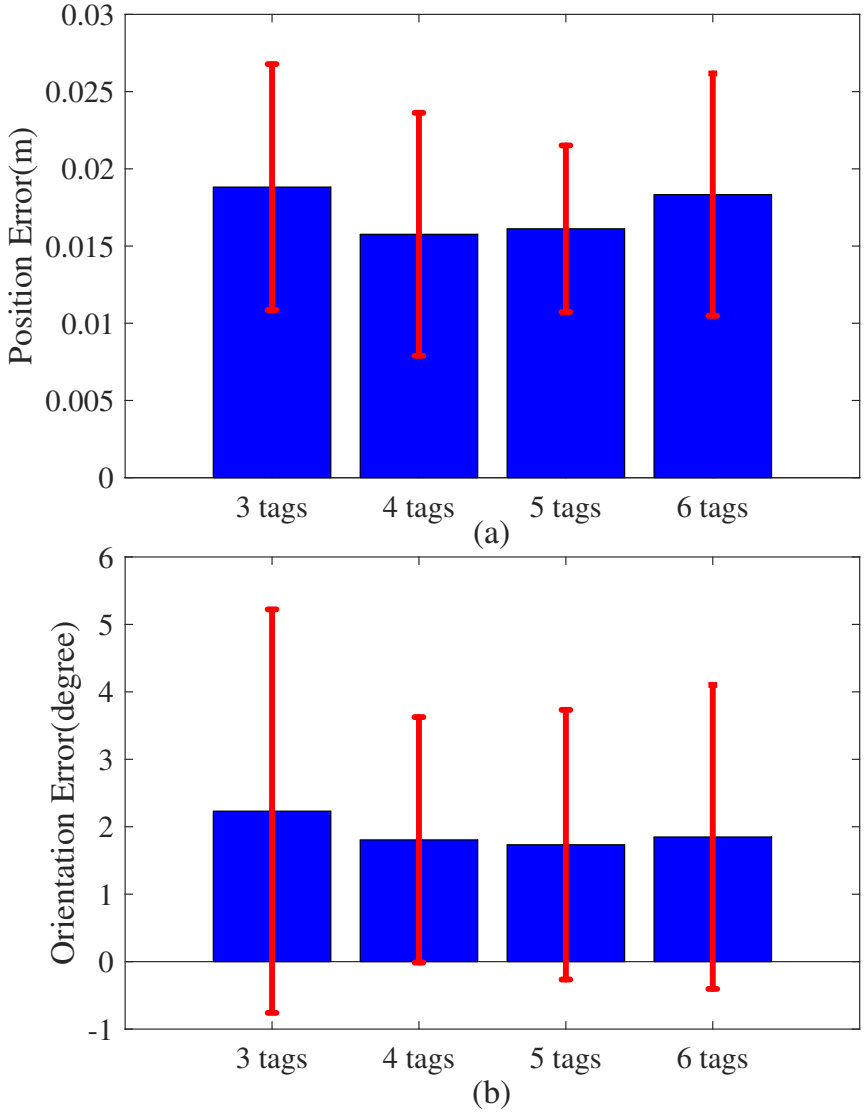


Figure 2.14: (a) Position errors of different numbers of attached RFID tags, (b) Orientation errors of different numbers of attached RFID tags.

Comparison with State-of-the-Art Method

Next, we compare our approach to the state-of-the-art UAV indoor localization method. We implement the recently developed Parallel Tracking and Mapping (PTAM) scheme [75] with our Parrot ARDrone 2.0 hardware. The PTAM based implementation utilizes data from a 2D camera, sonar, and an IMU to estimate a 3D pose in an unknown environment. We conducted this first experiment under the confined setup, and each trial followed the same trajectories in our experimental filed. We manually moved the UAV-mounted rolling rack back and forth in 4×4 m^2 field and adjusted the UAV's vertical height to make the total length of the trajectories

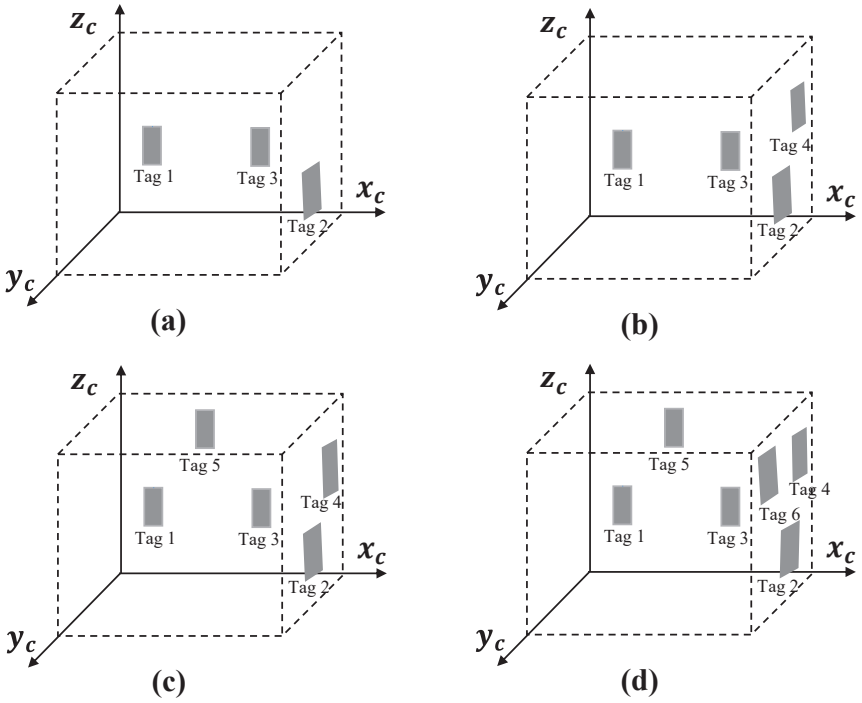


Figure 2.15: Layout 4 is deployed for evaluating the effect of the number of tags on pose accuracy: (a) three tags, (b) four tags, (c) five tags, and (d) six tags.

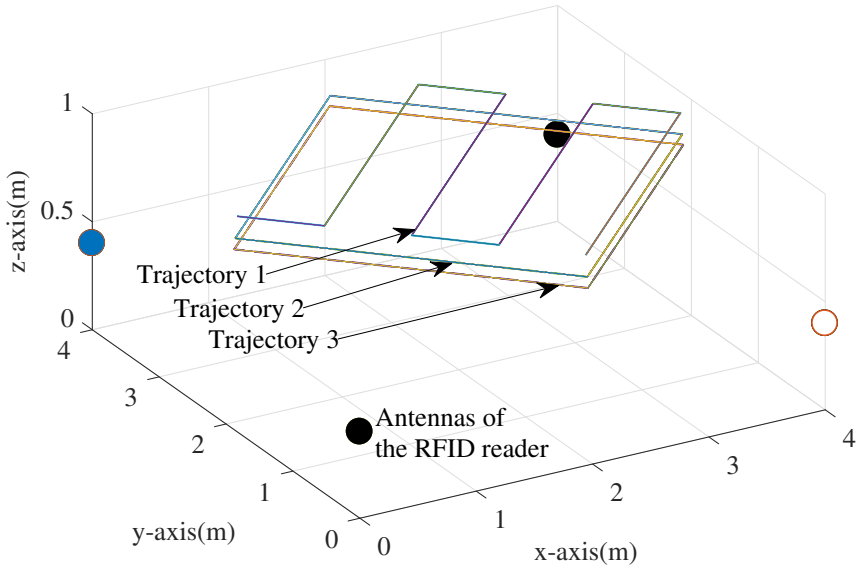


Figure 2.16: Examples of the experimental trajectories.

more than 10 m. An example of the trajectories is shown in Fig 2.16. The proposed RFUAV system localizes the UAV using readings from the RFID reader, while the PTAM localizes the UAV with multi-modal data fusion from the Parrot ARDrone2.0 platform.

The results of comparing the CDFs of position and orientation error are presented in Fig. 2.17. As shown in Fig. 2.17(a), RFUAV achieves a median position error of about 0.04

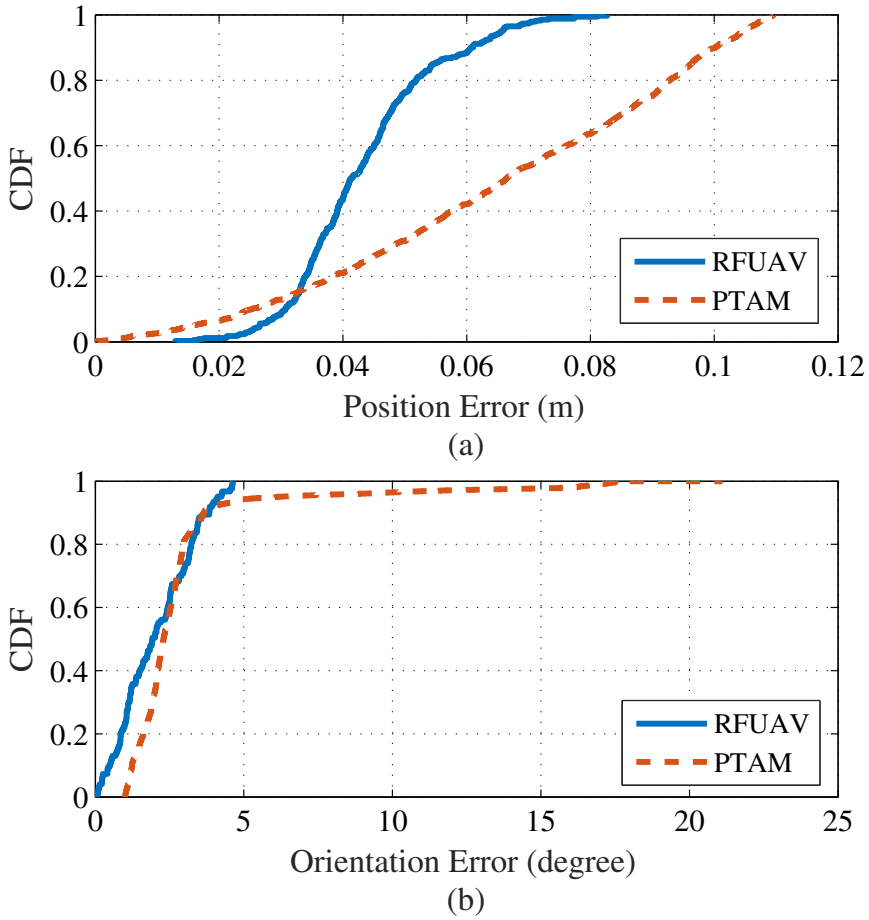


Figure 2.17: Comparison of localization accuracy: (a) CDFs of position errors of RFUAV and PTAM; (b) CDFs of orientation errors of RFUAV and PTAM.

m, and the 90th percentile error is about 0.06 m. The PTAM system achieves a median error about 0.067 m, and the 90th percentile error is slightly lower than 0.1 m. RFUAV outperforms PTAM with a great reduction of both the median error and 90th percentile error. Fig. 2.17(b) compares the orientation accuracy of RFUAV and PTAM. It shows that RFUAV can achieve a median error about 2°. On the other hand, PTAM has a median error about 2.5°. Obviously, RFUAV can provide a more reliable orientation estimation than PTAM, because the maximum orientation error of RFUAV is less than 5° , while the maximum orientation error of PTAM is up to 21°.

Our second experiment was conducted under the dynamic setup to compare the performance of the two methods, while the UAV is hovering in a position in the air. During the experiment, we sealed all air vents in the laboratory to create a windless environment for the UAV. Absolute position oscillations were about 10 cm in each direction of x, y, and z. Usually,

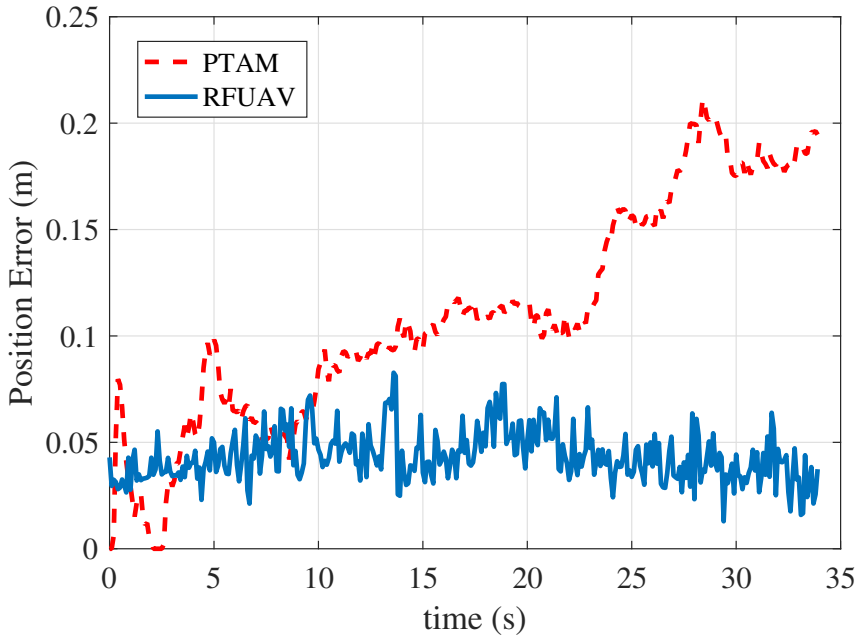


Figure 2.18: Cumulative positioning error of RFUAV and PTAM while the UAV hovers for 35 seconds.

vision-based algorithms, such as PTAM, suffer from significant position error when the UAV is hovering due to camera data noise and the slight position shift induced by IMU. For vision-based positioning systems, the current localization result depends on the estimation of its previous location. So the accumulative error will persistently increase as flight time goes on [76], especially, while the UAV is hovering. We compared the position error between RFUAV and PTAM while the UAV hovered in a fixed position for 35 seconds, the results of which are presented in Fig. 2.18. Results show that the position error of PTAM increases continuously; the error grows to 0.2 m by the end of the 35-second period. Whereas, RFUAV achieves a stable position error while the UAV hovers for the same period of time. The maximum position error of RFUAV is less than 0.08 m. From Fig. 2.18, we conclude that RFUAV is resilient to accumulative error, and it can provide a precise position for a hovering UAV. RFUAV localizes the UAV at each individual observation from the RFID reader. Even though the measurement noise in the observation will distort the estimated position, the observation model of the RFID does not accumulate error over time.

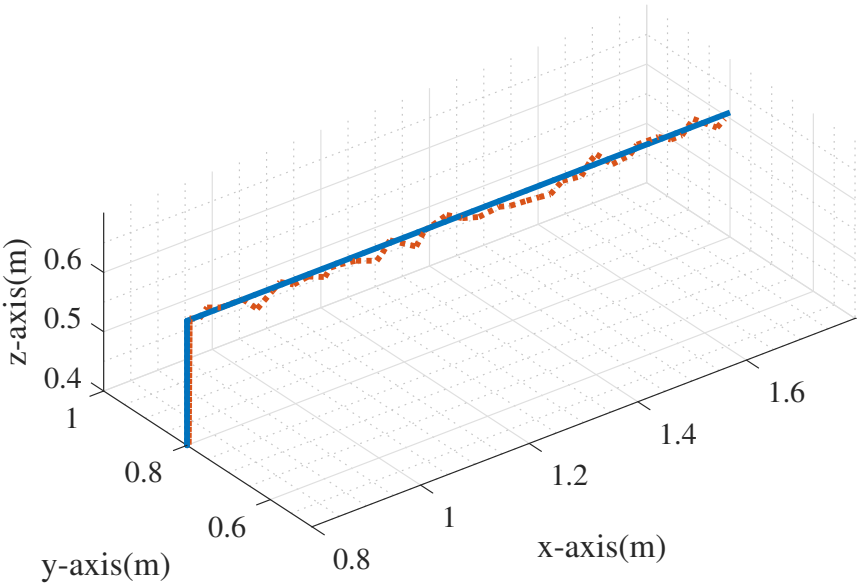


Figure 2.19: The UAV Trajectory as estimated by RFUAV (red dashed line) and ground truth (blue solid line).

Navigational Trajectory

To evaluate the RFUAV system's potential for indoor autonomous navigation, we conducted an experiment under the dynamic setup in our indoor laboratory. During the experiment, the UAV moved through a set of fixed destination waypoints. The trajectory of the movement is represented by the positions provided by RFUAV. The experiment's results are shown in Fig. 2.19, and illustrate that the estimated trajectory is highly accurate, with only a small disturbance around the ground-truth. Thus, control of the UAV becomes quite straightforward with RFUAV. As shown by other works, the pose estimation algorithm plays a critical role in the control strategy of autonomous UAV navigation [77, 78]. Thus, our proposed RFUAV system can greatly improve the performance of autonomous navigation of UAVs in indoor environments.

2.4 Conclusions

In this paper, we proposed an innovative indoor localization system for UAVs, termed RFUAV, which provides precise 6-DoF orientation and location estimation with a COTS RFID reader and tags. A Bayesian filter was leveraged to estimate the location of tags with phase difference. Then, we estimated pose with an SVD-based algorithm. To evaluate the performance

of our RFUAV system, we conducted exhaustive experiments in an indoor environment. The results demonstrated that our RFUAV system can achieve an accurate location estimation with a mean error of 0.04 m and an accurate orientation estimation with a mean error of 2.5°. To the best of our knowledge, this was the first feasible UHF passive RFID based localization system for UAVs. RFUAV is a promising method for indoor UAV navigation that is simple, computationally cost-effective, and not dependent on specific UAV architecture.

Chapter 3

RFHUI:An RFID based Human-Unmanned Aerial Vehicle Interaction System in an Indoor Environment

3.1 Introduction

The application of Unmanned Aerial Vehicle (UAV), which originated in the military arena, has rapidly expanded to other areas, such as agriculture, research, commerce, and so on. Due to its prominent maneuverability, small form factor, and low cost, the UAV is widely adopted for surveillance, entertainment, search and rescue, and inspection for maintenance. In terms of personal UAV applications, over the past few years, more and more advanced algorithms and sensors have been introduced, which make their use increasingly powerful and comprehensive. These personal UAVs are usually used for human entertainment activities, such as taking photos and videos. The mounting growth of demands makes the interaction between the user and UAV a research topic attracting considerable interests [5].

In this paper, we propose **RF**ID-based **H**uman **U**AV **I**nteraction (RFHUI), a low-cost, RFID-based system which provides an intuitive and easy-to-operate way to control and navigate a UAV in a complex indoor environment. The proposed method provides a means to precisely control a UAV to navigate it in a 3D space in a real-time manner. Specifically, we attach N ($N \geq 3$) Ultra high frequency (UHF) passive RFID tags to a small board to create a hand-held controller. We record the position of each tag against the built-in coordinates of the controller. This position is denoted as a local one. We then deploy a Commercial Off-The-Shelf (COTS) RFID reader with multiple antennas to gather the observation of the tags. The global position, which refers to the global coordinates in the 3D space, of an RFID tag can be precisely tracked by the channel state information (CSI) phase measurements of the RFID tag

responses from multiple antennas. A 6-DoF pose of the controller can be obtained from the known local position and estimated global position of the N attached tags. Finally, following the movement of the controller, the UAV responds and updates its pose and position in the air. The remainder of this paper is organized as follows. We review related work in Section 3.2. We present the design and analysis of the RFHUI system in Section 3.3 and our experimental study in Section 3.4. Section 3.5 concludes this paper.

3.2 Related Work

With the development of robotics and growing demands for civilian and industrial applications, the concept of interaction and collaboration between human and robots has received a lot of attention. The study of Human Robot Interaction (HRI) focuses on how their communication achieves better real-time performance. It can be approximately divided into three areas of applications: teleoperation in specific environments [79, 80, 81, 82, 83], human-centric social interaction [84], and industrial manufacturing [85]. For applications in social interaction, Santos et al. proposed a tour-guide robot which is capable of recognizing users hand gestures and providing voice feedback [86]. In the field of HRI teleoperation in a specific environment, urban search and rescue (USAR) is a high-interest research topic for deploying an HRI teleoperation in a specific environment. For example, Kohlbrecher et al. presented a human-robot system for rescue missions [87].

Compared to traditional robotic Unmanned Ground Vehicles (UGV), the UAV has significant differences, including flying freely, poor carrying capability, and being unsafe to touch. These demand a different and suitable new interaction method for human and UAV. The applications of Human Drone Interaction (HDI) are primarily focused on jogging companion UAVs involved in shooting videos, gesture recognition, and floating display. Muller et al. designed and built an outside jogging companion quadcopter system with GPS localization [88]. In [89], Scheible et al. proposed a system that combines a quadrocopter, a video projector, and a mobile phone for projecting contents onto walls or objects in an open space. Obviously, these UAVs are large and could only be used outdoors, thus prohibiting close interaction between human

and drones. For gesture control applications, Cauchard et al. investigated the problem of multiple participants and found that natural gesture control leads to a more intimate relationship between user and UAV [90]. In the current commercial UAV market, DJI announced a state-of-the-art small gesture control based UAV product, called Spark, in May 2017. This is the first time that gesture recognition technologies have been introduced for consumer-class UAVs, enabling the removal of a traditional remote controller.

Since the last decade, RFID technology has been widely recognized as a promising solution for item serialization and tracking. Due to its cost-effective, lightweight, small form factor, and power-free properties, the RFID has also been widely deployed for indoor localization [64, 91, 92, 93, 94, 95, 96, 57, 97, 98, 99]. A considerable number of studies have focused on accessing the phase measurement of RF signals for localization [58, 59, 100, 101, 102, 103] and vital sign monitoring [104, 105]. Making use of Angle of Arrival (AOA) is a classic solution, which is driven by measuring the phase difference of the signals received at different antennas. In [59], Azzouzi presented the new measurement results for an AOA approach to localize RFID tags. In addition to localization applications, RFID technology has also been employed for 3-D reconstruction. Bu et al. proposed an approach based on the phase differences of RF signals for the 3-D reconstruction of cubes [67], which is free of the limitation of line-of-sight and battery life constraints. Moreover, there are many other interesting scenarios that access RFID technology [106, 107, 108]. For example, in [109], the reading patterns of RFID tags are leveraged to detect customers' behaviors in a physical clothes store. In [110], RFID tags are attached to the clothes of a patient to measure his/her respiration rate.

Motivated by the research of the aforementioned RFID applications, we go beyond the above HRI and HDI works to design a practical HDI navigation system based on the RFID technology and test it in a real-world laboratory environment. Compared to traditional vision-based HDI systems, the proposed RFHUI does not have the line-of-sight limitation due to the penetrating characteristics of RF signals.

3.3 RFHUI Design and Analysis

RFHUI is a low-cost, RFID-based system aiming to offer flexible human-UAV interaction. It provides an intuitive and easy-to-operate means for controlling a UAV in a 3D space. The RFHUI system comprises N $(N \geq 3)$ UHF passive RFID tags and a COST RFID reader with M $(M \geq 2)$ antennas. The tags are attached to the controller, and, when tracking the RFID tags by querying the phase information of each tag, a 6-DoF pose of the controller can be obtained. Then, the UAV can be controlled by this pose. In this section, we will introduce the system model and RFHUI architecture and design. Table 3.1 shows the important notations used in this paper.

3.3.1 System Architecture

The system architecture of RFHUI is presented in Fig. 3.1. Our proposed RFHUI system consists of three main components as follows:

- *RFID Localizer*: We deploy a Bayesian filter to estimate the global location of the tags by utilizing the phase measurement from each tag, which is obtained by the reader.
- *Pose Tracker*: After the global location of N $(N \geq 3)$ tags are obtained by the RFID localizer and combined with the given local location of each tag, we can track the pose of the controller with an SVD based method. Here, the local location is given in the built-in coordinate of the controller.
- *Control Module*: It converts the pose of the controller into flying control commands, which are transmitted to the UAV. Thus, the UAV can be navigated following a trajectory that is guided by the movement of the controller.

We present the design of these three components in the remainder of this section.

Table 3.1: Important notations used in the paper

Notation	Description
N	Amount of implemented RFID tags
M	Amount of RFID antennas
l_m	The position of the mth antenna in the global coordinate
x_t	The hypothetical posterior state of a dynamic system at a given time t. In RFHUI system, it refers to the position of an RFID tag at time t
u_t	The received control at time t. In RFHUI system, it denotes the speed of an RFID tag at time t
z_t	Observation at time t
$\mathcal{B}(x_t)$	The belief that denotes the probability of the system is in state x at time t. In RFHUI system, it refers to the probability of a tag in position x_t
$P(z_t \mid x_t, l_m)$	The observation model of the mth antenna
θ	RF phase measured from the reader
R	The distance between the reader antenna and an RFID tag
λ	Wavelength of the RF radio signal
$\theta_T, \theta_R, \theta_{TAG}$	The RF phase distortion caused by the reader's transmit circuits, the reader's receiver circuits, and the tag's reflection characteristics, respectively
$\hat{X}_c, \hat{Y}_c, \hat{Z}_c$	Unit vectors of the axes of the built-in coordinates of the controller
$\hat{X}_g, \hat{Y}_g, \hat{Z}_g$	Unit vectors of the axes in the global coordinates
$\mathbf{T}_t, \mathbf{R}_t$	The position and orientation of the controller at time t
$\mathbf{p}_n^t$	The position $(x_n^t, y_n^t, z_n^t)^T$ of the nth tag at time t in the global coordinates
${}_c^g\mathbf{T}_t$	The rigid transform between the controller's built-in coordinates and the global coordinates at time t
$\mathbf{H}_t$	Pose of the controller at time t
$\mathbf{U}_t$	Pose of the UAV at time t

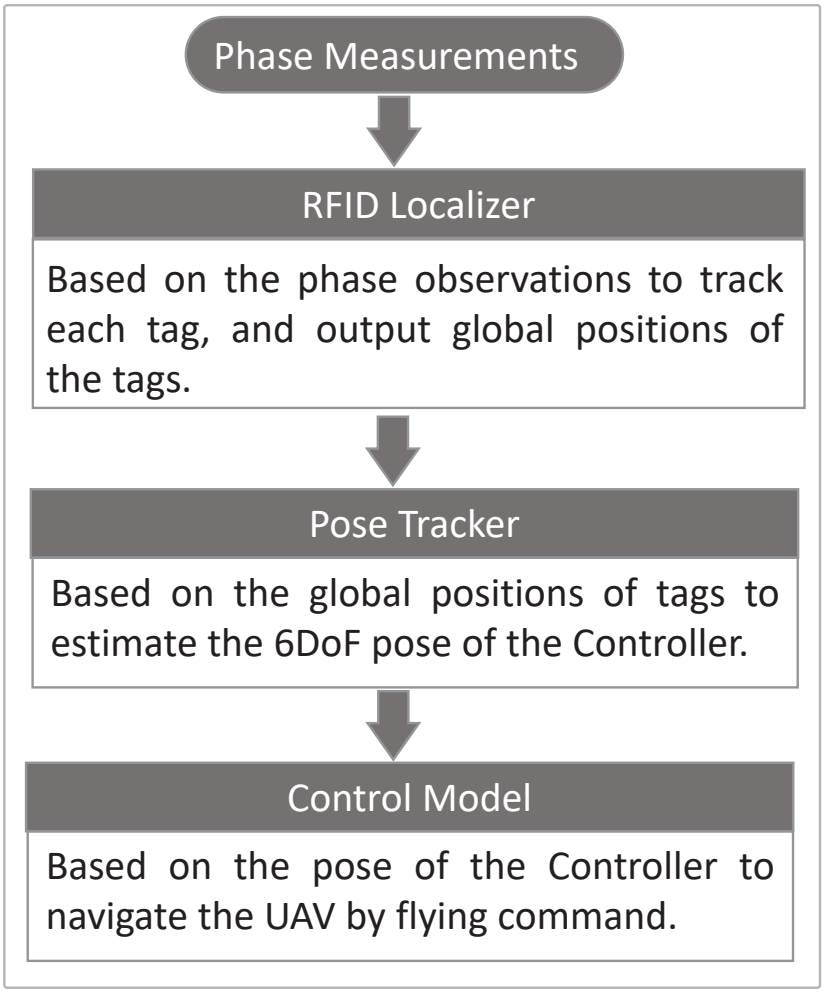


Figure 3.1: The system architecture of RFHUI, where the global coordinates are built in the real world.

3.3.2 RFID Localizer

In the RFHUI system, phase measurements of the tag responses are collected by an RFID reader with M antennas. We fix and measure the positions of all the antennas. Hereafter, let l_m denote the position of the mth antenna in the global coordinate.

Bayesian Filter Updates for Tag Localizing

In RFHUI, a Bayesian filter is deployed to localize the tags mounted on the controller. The Bayesian filter addresses the problem of estimating belief over the hypothetical posterior state x of a dynamic system by sensor observations. For the RFID localizer, the state x denotes the position of the tag against the global coordinate. The belief $\mathcal{B}(x_t)$, which denotes the

probability that the system is in state x at time t, is recursively updated by the Bayesian filter. The update is calculated from control u_t, observation z_t, and prior belief $\mathcal{B}(x_{t-1})$ at time $(t-1)$, which is calculated previously.

Usually, one updating cycle of a typical Bayesian filter can be divided into two essential steps. Control update or prediction is the first step of the process, which is given as:

$$\overline{\mathcal{B}}\left(x_t\right) = \int P(x_t \mid u_t, x_{t-1}) \mathcal{B}\left(x_{t-1}\right) dx_{t-1}, \tag{3.1}$$

where $P(x_t \mid u_t, x_{t-1})$ provides the probability of a tag moving from position x_{t-1} to x_t under the control of u_t, referred to as a motion model, and $\overline{\mathcal{B}}(x_t)$ represents the probability of the tag at position x_t after control u_t is executed. We assume that the speed of tags will remain constant for a very short time interval, and hence, a constant speed model can be deployed for the RFID localizer, which is expressed as:

$$\begin{aligned} & P(x_t \mid u_t, x_{t-1}) \\ = & \frac{1}{\sqrt{2\pi}\delta} \int_0^{\Delta t} \exp\left\{ -\frac{\left(x_t - \left(x_{t-1} + \ u_t \cdot y \ \right)\right)^2}{2\delta^2} \right\} dy, \end{aligned} \tag{3.2}$$

where u_t denotes the speed of the tag at time $t-1$ and Δt represents the time interval between $t-1$ and t.

Without loss of generality, we assume the movements of the tag satisfy a typical Gaussian distribution with standard deviation δ. The second step is the measurement update, which is written as:

$$\mathcal{B}(x_t) = \eta \cdot \overline{\mathcal{B}}(x_t) \cdot P(z_t \mid x_t). \tag{3.3}$$

In (3.3), η is a constant to integrate the sum of all $\mathcal{B}(x_t)$ into 1, and $P(z_t \mid x_t)$ represents the observation model. The RFID localizer is equipped with M reader antennas. Thus (3.3) can be

rewritten as:

$$\mathcal{B}\left(x_t\right) = \sum_{m=1}^{M} \eta \cdot \overline{\mathcal{B}}(x_t) \cdot P(z_t \mid x_t, l_m), \tag{3.4}$$

where $P(z_t \mid x_t, l_m)$ denotes the observation model for the mth antenna. It provides the probability when the mth antenna in position l_m observes measurement z_t of the tag, which is in position x_t. The detail of the model is presented in the following.

Model of RFID Phase Measurement

The relationship of the RF phase shift between transmitted and received signal is given by the following equation:

$$\theta = \left(2\pi \cdot \left(\frac{2R}{\lambda}\right) + \theta_T + \theta_R + \theta_{TAG}\right) \mod 2\pi, \tag{3.5}$$

where θ is the RF phase measured by the reader, R is the distance between the reader antenna and the RFID tag, λ is the wavelength of the RF radio signal, $\theta_T, \theta_R, \theta_{TAG}$ are the RF phase distortion caused by the reader's transmit circuits, by the reader's receiver circuits, and by the tag's reflection characteristics, respectively, and $\mod$ is the Modulo operation.

Experiments show that for the same reader antenna, the same RFID tag, and the same radio frequency, θ_T, θ_R, and θ_{TAG} are fixed, and can be denoted as $\theta' = \theta_T + \theta_R + \theta_{TAG}$. Thus (3.5) can be rewritten as:

$$\theta = \left(2\pi \cdot \left(\frac{2R}{\lambda}\right) + \theta'\right) \mod 2\pi. \tag{3.6}$$

We assume that a tag is in position x_{t-1} and a reader antenna in a position l_m observes the RF phase θ_1 from the tag's response. Under the same RF frequency, the tag moves to position x_t and the RF phase θ_2 is observed from the tag. The differential RF phase measurement between

the two positions satisfies the following conditions:

$$\Delta\theta_{12} = (\theta_1 - \theta_2) \mod 2\pi \tag{3.7}$$

$$\Delta\theta_{12} = \left(\left(2\pi\left(\frac{2\left|x_{t-1}\cdot l_m^g\right|}{\lambda}\right) + \theta'\right) \mod 2\pi \right. \tag{3.8}$$
$$\left. - \left(2\pi\left(\frac{2\left|x_t\cdot l_m^g\right|}{\lambda}\right) + \theta'\right) \mod 2\pi\right) \mod 2\pi$$

$$\Delta\theta_{12} = \left(\frac{4\pi}{\lambda}\cdot(|x_{t-1}\cdot l_m| - |x_t\cdot l_m|)\right) \mod 2\pi. \tag{3.9}$$

Equation (3.9) shows that under the same frequency for the same antenna and the same RFID tag, the differential RF phase $\Delta\theta_{12}$ is only determined by the distance the tag moves from x_{t-1} to x_t. In (3.9), $|x_t \cdot l_m|$ denotes the distance between the two positions. Hereafter, we assume that all the RF phases are measured for the same RFID reader and the same RFID tag under the same RF frequency. The tag moves in a discrete trajectory that is represented by a series of locations $x_1, x_2, ..., x_t$. The antenna, which is stationary in position l_m, collects the phase measurement for each location as $\theta_1, \theta_2, ..., \theta_m$. Then, the discrete trajectory of the movement of the tag should satisfy:

$$\begin{cases} |x_i\cdot l_m| - |x_j\cdot l_m| = \frac{\lambda}{4\pi}\cdot\Delta\theta_{ij} + n\cdot\frac{\lambda}{2} \\ \Delta\theta_{ij} = (\theta_i - \theta_j) \mod 2\pi \\ n \in \{1,\ 2,\ 3,\ ...\} \\ i, j \in \{1, 2, ..., t\} \text{ and } i \neq j. \end{cases} \tag{3.10}$$

The observation model $P(z_t \mid x_t, l_m)$ can be updated by (3.10) to provide the probability that if a tag moves from x_{t-1} to x_t, the differential RF phase $\Delta\theta_{t,t-1}$ is obtained by the reader. We model the differential RF phase by the following equation:

$$P(\Delta\theta_{t,t-1} \mid x_{t-1}, x_t, l_m) = \begin{cases} 1, & \text{if (10) is satisfied} \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

Let's consider the distortion of the RF phase that is caused by the thermal noise. Experiments reveal that the thermal noises introduces random errors to the phase measurement

following a typical Gaussian distribution. Thus, we denote the RF phase as $\theta \sim \mathcal{N}(\mu, \delta)$, where μ is the RF phase without the distortion of thermal noise and δ denotes the standard deviation. Hence, we can update the differential RF phase as $\Delta\theta_{ij} \sim \mathcal{N}(\mu_i - \mu_j, \sqrt{2}\delta)$, and (3.11) can be updated as:

$$
\begin{aligned}
&P(\Delta\theta_{t,t-1} \mid x_{t-1}, x_t, l_m) \\
&= \frac{1}{\sqrt{2\pi}\delta} \int_0^{\Delta\theta_{t,t-1}} \exp\left\{ -\frac{\big(y - \big(\mu_t - \mu_{t-1}\big)\big)^2}{2\delta^2} \right\} dy \qquad (3.12) \\
&\mu_t - \mu_{t-1} = \left(\frac{2\,|x_t \cdot l_m|}{\lambda} - \frac{2\,|x_{t-1} \cdot l_m|}{\lambda} \right) \mod 2\pi. \qquad (3.13)
\end{aligned}
$$

Based on (3.12), (3.13) and (3.4), we can estimate the locations of the RFID tags.

3.3.3 Pose Tracker

The location of a tag, which is denoted as $\mathbf{p}_n^t = (x_n^t, y_n^t, z_n^t)^T$ for the nth tag at time t, can be estimated by the RFID localizer. When the controller is located at $\mathbf{T}$, with orientation $\mathbf{R}$ in the given global coordinate, $\mathbf{T}$ and $\mathbf{R}$ together are called the pose of the controller. Here, we denote the position of the controller at time t as $\mathbf{T}_t = (x_t, y_t, z_t)^T$, and the orientation at time t as

$$
\mathbf{R}_t = \begin{bmatrix} \hat{X}_c \cdot \hat{X}_g & \hat{Y}_c \cdot \hat{X}_g & \hat{Z}_c \cdot \hat{X}_g \\ \hat{X}_c \cdot \hat{Y}_g & \hat{Y}_c \cdot \hat{Y}_g & \hat{Z}_c \cdot \hat{Y}_g \\ \hat{X}_c \cdot \hat{Z}_g & \hat{Y}_c \cdot \hat{Z}_g & \hat{Z}_c \cdot \hat{Z}_g \end{bmatrix}, \qquad (3.14)
$$

where $\hat{X}_c, \hat{Y}_c$, and $\hat{Z}_c$ represent the unit vectors of the axes of the built-in coordinates of the controller, and $\hat{X}_g$, $\hat{Y}_g$, and $\hat{Z}_g$ denote the unit vectors of the axes in the global coordinates. The relationship of the two coordinates is illustrated in Fig. 3.2.

We measure the location of each attached tag in the controller's built-in coordinates, and the local location for the nth tag is denoted as $\bar{\mathbf{p}}_n = (\bar{x_n}, \bar{y_n}, \bar{z_n})^T$. The transformation between

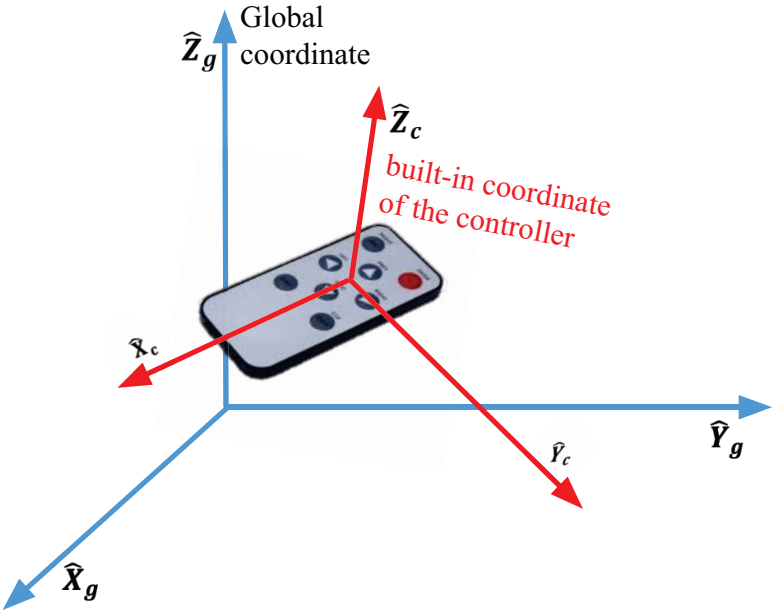


Figure 3.2: The global coordinates versus the built-in coordinates of the controller.

the global location and local location of the same tag is given by:

$$\begin{cases} \tilde{\mathbf{P}}_n^t = {}_c^g\mathbf{T}_t \cdot \bar{\mathbf{P}}_n \\ \tilde{\mathbf{P}}_n^t = (\mathbf{p}_n^t, 1)^T \\ \bar{\mathbf{P}}_n = (\bar{\mathbf{p}}_n, 1)^T, \end{cases} \tag{3.15}$$

where ${}_c^g\mathbf{T}_t$ denotes the rigid transform at time t, $\mathbf{p}_n^t$ and $\bar{\mathbf{p}}_n$ is the location of the nth tag in the global coordinates and the controller's built-in coordinates, respectively. In (3.15), ${}_c^g\mathbf{T}_t$ comprises the pose of the controller in the global coordinate:

$${}_c^g\mathbf{T}_t = \begin{bmatrix} \mathbf{R}_t & \mathbf{T}_t \\ 0\,0\,0 & \mathbf{1} \end{bmatrix}, \tag{3.16}$$

where $\mathbf{R}_t$ and $\mathbf{T}_t$ denote the global orientation and global position of the controller at time t, respectively. Based on (3.15) and (3.16), we can obtain the pose of controller by searching an optimal transform ${}_c^g\mathbf{T}_t$. When the global locations of all the tags of the controller are provided

by proposed RFID localizer, (3.15) can be updated as follows:

$$\begin{cases} \mathbf{p}_1^t = \mathbf{R}_t \cdot \bar{\mathbf{p}}_1 + \mathbf{T}_t \\ \mathbf{p}_2^t = \mathbf{R}_t \cdot \bar{\mathbf{p}}_2 + \mathbf{T}_t \\ \cdots \\ \mathbf{p}_n^t = \mathbf{R}_t \cdot \bar{\mathbf{p}}_n + \mathbf{T}_t, \end{cases} \tag{3.17}$$

where $\mathbf{p}_1^t$, $\mathbf{p}_2^t$, and $\mathbf{p}_n^t$ are the global locations for tag 1, 2, and n, respectively; and $\bar{\mathbf{p}}_1$, $\bar{\mathbf{p}}_2$, and $\bar{\mathbf{p}}_n$ are the local locations for tag 1, 2, and n, respectively. Therefore, the process of finding an optimal transform ${}_c^g\mathbf{T}_t$ can be formulated as a least square minimization problem as:

$$\min_{\{\mathbf{R}_t, \mathbf{T}_t\}} \sum_{i=1}^{N} \parallel \mathbf{p}_i^t - (\mathbf{R}_t \cdot \bar{\mathbf{p}}_i + \mathbf{T}_t) \parallel, \tag{3.18}$$

where N is the total number of tags and $|| \cdot ||$ is the norm of a vector. Problem (3.18) is a typical problem of determining the rotation and translation relationship between two sets of data points at different coordinates, and a variety of methods have been introduced to solve such problems [73, 74]. Based on the approach that is introduced in [73], our proposed pose tracker is developed to find the optimal ${}_c^g\mathbf{T}_t$ in three steps:

Step 1. Finding the centroids of all the locations in both the global coordinates and the local coordinates, which is denoted as $\mathbf{C}$ and $\bar{\mathbf{C}}$, respectively. Then, use the centroids as the new origins of two coordinates and transfer the locations into these two coordinates, as:

$$\begin{cases} \mathbf{p}_i^{t\prime} = \mathbf{p}_i^t - \mathbf{C}, \ \text{ for } i \in [1, 2, ..., N] \\ \bar{\mathbf{p}}_i' = \bar{\mathbf{p}}_i - \bar{\mathbf{C}}, \ \text{ for } i \in [1, 2, ..., N], \end{cases} \tag{3.19}$$

where N is the total number of tags.

Step 2. Determining the optimal rotation $\mathbf{R}_t$ with the singular value decomposition (SVD) method. First, cascade all the shifted locations of the tags in both global and local coordinates

to form two matrices:

$$\begin{cases} \mathbf{A} = [\mathbf{p}_1^{t\prime}, \mathbf{p}_2^{t\prime}, ..., \mathbf{p}_n^{t\prime}] \\ \mathbf{B} = [\bar{\mathbf{p}}_1', \bar{\mathbf{p}}_2', ..., \bar{\mathbf{p}}_n'], \end{cases} \tag{3.20}$$

where both $\mathbf{A}$ and $\mathbf{B}$ are $3 \times N$ matrices. Then, we decompose or factorize the matrix $\mathbf{AB}^T$ with the SVD method as:

$$[\mathbf{U}, \mathbf{D}, \mathbf{V}] = \mathit{SVD}(\mathbf{AB}^T), \tag{3.21}$$

where, $\mathbf{UU}^T = \mathbf{VV}^T = 1$, and $\mathbf{D} = \text{diag}(d_i), d_1 \geq d_2 \geq \cdots \geq d_n \geq 0$. Based on the result in [73], we obtain the optimal rotation $\mathbf{R}_t$ as:

$$\mathbf{R}_t = \mathbf{USV}^T, \tag{3.22}$$

where

$$\mathbf{S} = \begin{cases} \mathbf{I}, & \text{if } \det(\mathbf{U})\det(\mathbf{V}) = 1 \\ \text{diag}(1, 1, ..., 1, -1), & \text{if } \det(\mathbf{U})\det(\mathbf{V}) = -1. \end{cases} \tag{3.23}$$

In (3.23), $\mathbf{I}$ is an identity matrix, and diag$(\cdot)$ is a diagonal matrix.

Step 3. Obtaining the translation $\mathbf{T}_t$. After obtaining the rotation $\mathbf{R}_t$, $\mathbf{T}_t$ can be determined by the following equation:

$$\mathbf{T}_t = \mathbf{C} - \mathbf{R_t} \cdot \bar{\mathbf{C}}. \tag{3.24}$$

Therefore, based on the locations of the tags in both global and local coordinates, the proposed pose tracker can determine the controller's pose, including the orientation $\mathbf{R}_t$ and the position $\mathbf{T}_t$, referring to the global coordinates.

3.3.4 Human UAV Interaction Module

The human UAV interaction module primarily links the change of the controller's pose with UAV movement to achieve flexible remote control. We use the estimated pose of the controller to control the navigation of the UAV. To achieve real-time control, the UAV must react sensitively to the change of the controller's pose in a manner that follows the trajectory of the moving controller.

We use $\mathbf{H}_t$ to denote the pose of the controller, and $\mathbf{U}_t$ to denote the pose of the UAV at time t. The process of the module can be divided into four steps, which are detailed as follows:

1. Obtaining $\mathbf{H}_t$ and $\mathbf{H}_{t+1}$ from the pose tracker.

2. Calculating $\Delta\mathbf{H} = \mathbf{H}_{t+1} - \mathbf{H}_t$, which contains the change of position and orientation in the three-dimensional space.

3. Amplifying $\Delta\mathbf{H}$ as $\Delta\mathbf{H}' = \alpha \cdot \Delta\mathbf{H}$, where α is the parameter of the amplification, and we usually set $\alpha = 5$. We can make a slight movement of the controller to activate a large-scale movement of the UAV.

4. Converting the $\Delta\mathbf{H}'$ to flying control commands and send it to the UAV.

Step 4 cooperates with the specific UAV platform, and it usually relies on the API to communicate with the UAV. For example, in our experimental platform, an ROS based system is developed to communicate with the ARDrone2.0 platform. It updates the target position of the UAV by

$$\mathbf{U}_{t+1} = \mathbf{U}_t + \Delta\mathbf{H}', \tag{3.25}$$

and sends the $\mathbf{U}_{t+1}$ to the UAV through the ROS message service.

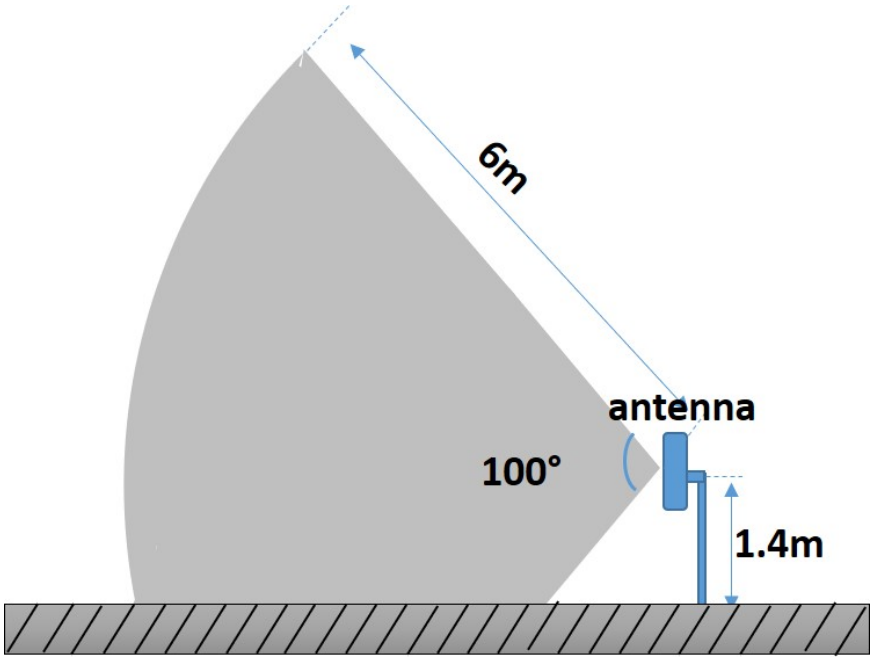


Figure 3.3: Side view of the RFID detectable field.

3.4 Experimental Validation and Results

3.4.1 Experiment Setup

We conduct a series of experiments to demonstrate the performance of the RFHUI system. We establish a prototype of RFHUI using a COTS reader and several UHF passive RFID tags. A Zebra FX7500 RFID reader with four Zebra AN720 antennas is incorporated to query the RFID tags. The Zebra FX7500 reader is widely deployed in retail, manufacturing factory, and warehouse applications, and meets the EPC Gen2 standard requirements.

In our prototype system, we use the Low-Level Reader Protocol (LLRP) through an Ethernet port to communicate with the reader and report the RFID measurements. The Zebra AN720 Antennas provide a left circular polarization with a 100° beam width and a $5.5 \sim 6$ dB gain. Each antenna is mounted on a holder of 1.4 m in height. The four antennas with their holders are deployed in front of the user. In all our experiments, we set the reader works at the maximum RF transmission power, i.e., 33 dBm, to enable each antenna to gain a detectable range of up to 6 m. Our experimental setting is illustrated in Fig. 3.3 (side view) and Fig. 3.4 (top view). The configuration of the four antennas created a detectable field, which allows the four antennas to interrogate an RFID tag simultaneously.

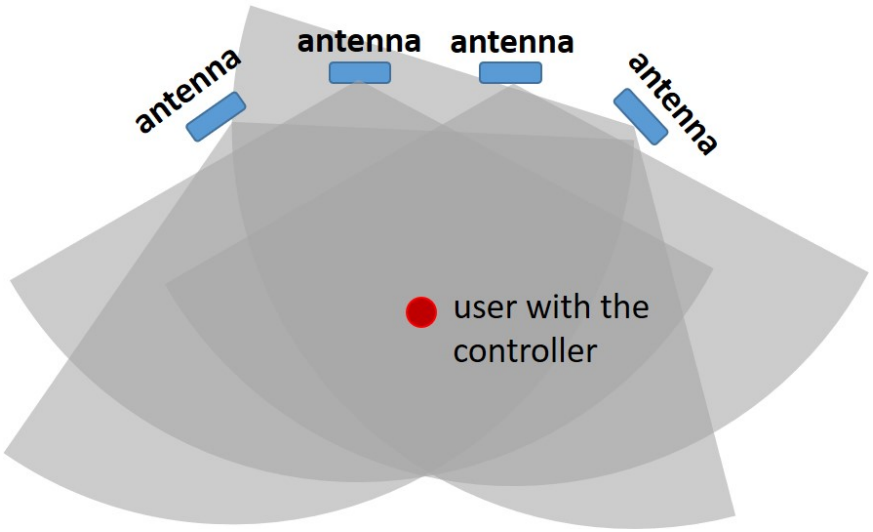


Figure 3.4: Top view of the RFID detectable field.

Three UHF passive RFID tags are attached to a foam board working as our prototype controller, which is shown in Fig. 3.5. Fig. 3.6 shows how the controller is operated by a user during the tests. Our experimental RFID tag is Smartrac Dogbone Monza R6, which is widely used in the retail business. We choose The Parrot ARDrone2.0 Elite Edition drone [75] as our UAV platform, which is shown in Fig. 3.7. It is equipped with a front camera, a bottom camera, a sonar, and an inertial measurement unit (IMU). Based on the measurements of the onboard sensors, it can localize itself by using a sensor fusion method. For example, the Parallel Tracking and Mapping (PTAM) technique can be implemented to estimate the 3D pose of the ARDrone2.0 drone.

3.4.2 Accuracy of RFID tracking and Pose Estimation

Effect of the Number of Antennas

Before revealing the performance of the proposed RFHUI system, we first conduct a set of benchmark experiments to discover the effect of the number of RFID antennas on the system performance. We configure the RFID reader with 1, 2, 3, and 4 antennas in each benchmark experiment, respectively. During every benchmark experiment, the controller is moved along the same trajectory, which is given in Fig. 3.8. We first moved the controller 20 centimeters in the direction of the x-axis, and then moved it for another 20 centimeters along the y-axis

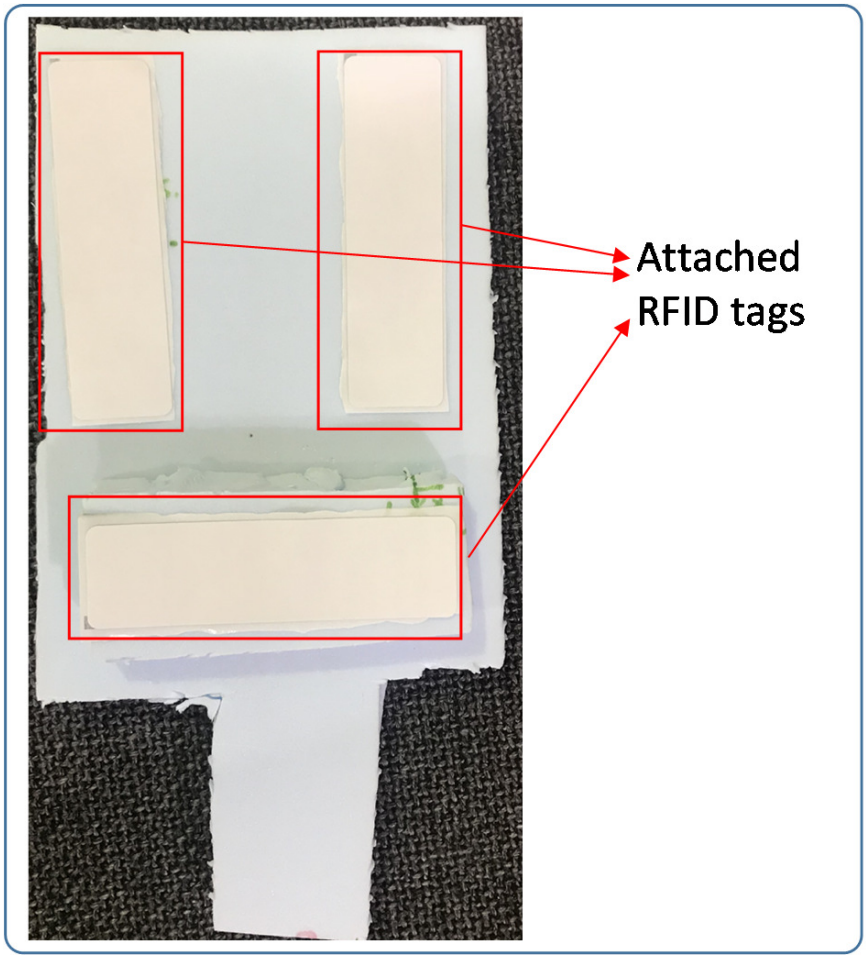


Figure 3.5: A prototype of our RFHUI controller.

direction. We sampled the trajectory every 2 centimeters, which is illustrated by the red points in Fig. 3.8. There was a total of 21 sampled points. At every sampled point we record the ground truth location and the estimated location that is provided by the RFID localizer for every tag and collect the ground truth and the estimated pose of the controller.

First, we evaluate the accuracy of the RFID localizer by comparing the estimated location to the ground truth location of every tag at all sample points. The average location error of each tag at different antenna configurations is shown in Fig. 3.9. We can see that the more antennas are deployed, the more accurate the estimated localization. The results are consistent with the conclusion of (3.4): the more antennas are deployed, the more accurate estimation can be made.

We also evaluate the accuracy of the controller's pose, including position and orientation, which is measured by our RFHUI system. The results in each antenna configuration are given

Figure 3.6: A user holds the controller in hand during an experiment.

in Fig. 3.10 and Fig. 3.11. From Fig. 3.10 and Fig. 3.11, the average errors of both position and orientation are reduced dramatically when the number of deployed antennas is increased. For the configuration with 4 antennas, the system achieves an average error of 0.021 m in position and 1.8° in orientation. Therefore, all hereafter experiments are executed with the 4-antenna configuration with the setup shown in Fig. 3.4.

RFID Tags Tracking

To evaluate the performance of the RFID localizer in RFHUI, we launch another experiment by attaching three UHF passive RFID tags to the controller. A user holds the controller and moves it following a given trajectory, which is inside the experiment field. In contrast to the simple trajectory in our benchmark experiments, we move the controller in a more complex and

Figure 3.7: The ARDrone2.0 Elite Edition drone used in our experiments.

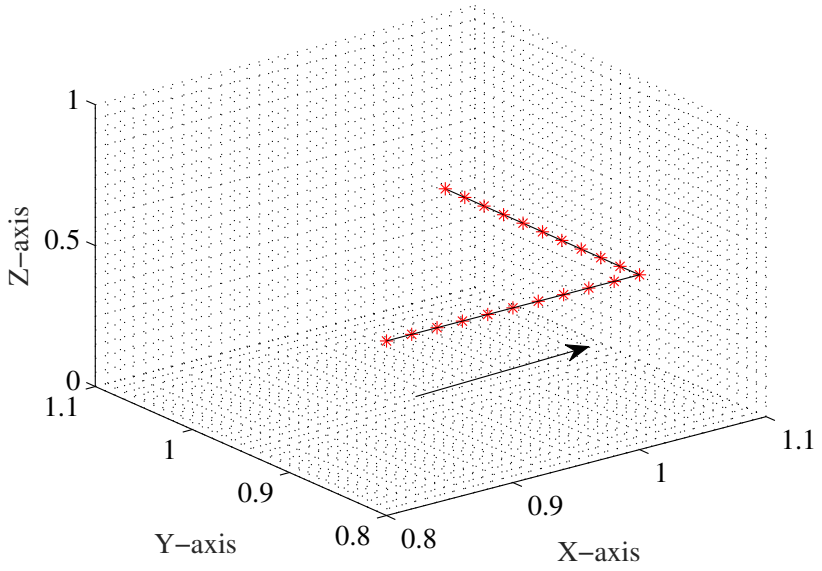


Figure 3.8: The moving trajectory of the benchmark experiments: the red points are the sampled locations.

longer trajectory with more variety in the moving direction, thus mimicking the actual user behavior while operating the UAV. During the experiment, the RFID localizer of RFHUI provides estimated locations for each tag while the controller is moving. We obtain the ground-truth locations by measuring the sampled points every 5 mm along the trajectory. The accuracy of

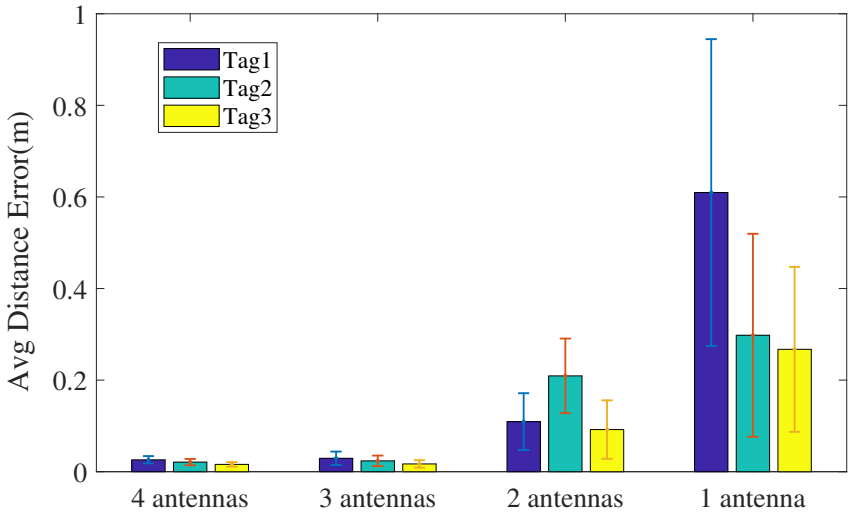


Figure 3.9: The average error and standard deviation of the localization error of the controller's tags for different antenna configurations.

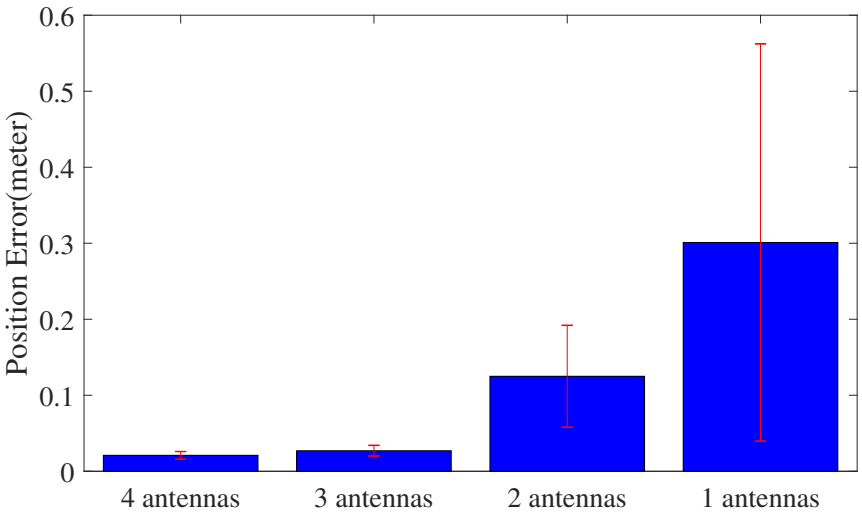


Figure 3.10: The average position error of the controller for different antenna configurations.

the proposed method is evaluated by calculating the errors between the ground-truth locations and the estimated locations of the sampled points.

We repeat the experiment several times, and the experimental results are presented in Fig. 3.12 in the form of the cumulative distribution function (CDF) of localization errors between estimated and ground-truth positions. We can see that the maximum error of the RFID localizer is less than 0.095 m for all the three tags. Moreover, with the RFID localizer of RFHUI, 80% of the localization errors are less than 0.045 m and 90% of them are under 0.06 m. Therefore, it is safe to state that the RFID localizer achieves very precise localization for tracking the moving RFID tags. Note that the average error of every tag is a little bit higher

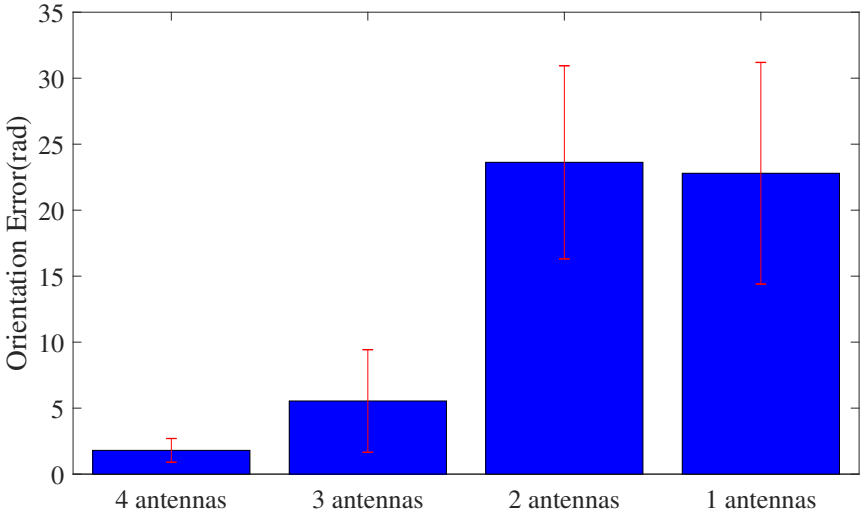


Figure 3.11: The average orientation error of the controller for different antenna configurations.

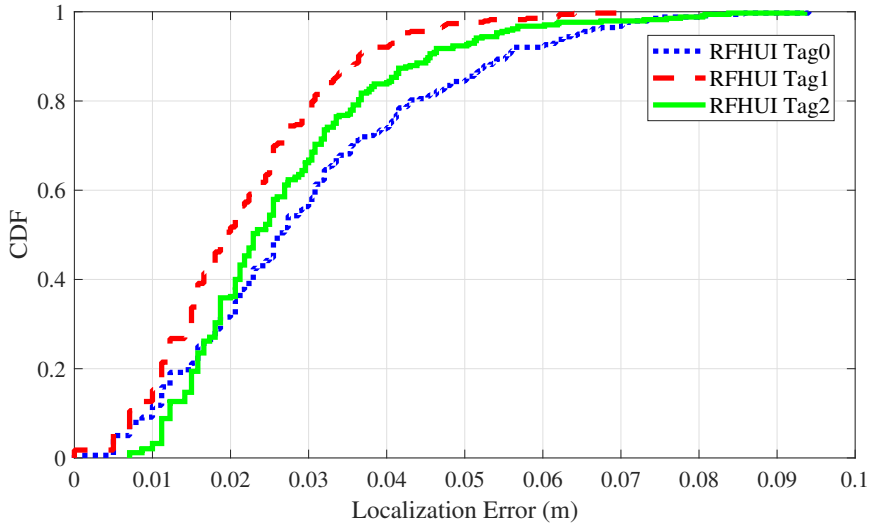


Figure 3.12: CDF of RFID tags tracking error with a more complex and longer trajectory.

than that in the benchmark experiments because we considered much more sample points and the controller moves along a more complex trajectory in this experiment.

Controller Pose Estimation

We next conduct an experiment to verify the feasibility and accuracy of our proposed pose tracker, including position and orientation estimation. The controller moves along a trajectory in our experiment field, held by a user.

The results are presented in Fig. 3.13. Fig. 3.13(a) shows that about 78% of the position errors of the proposed pose tracker are under 0.05 m, and the maximum error is less than 0.083

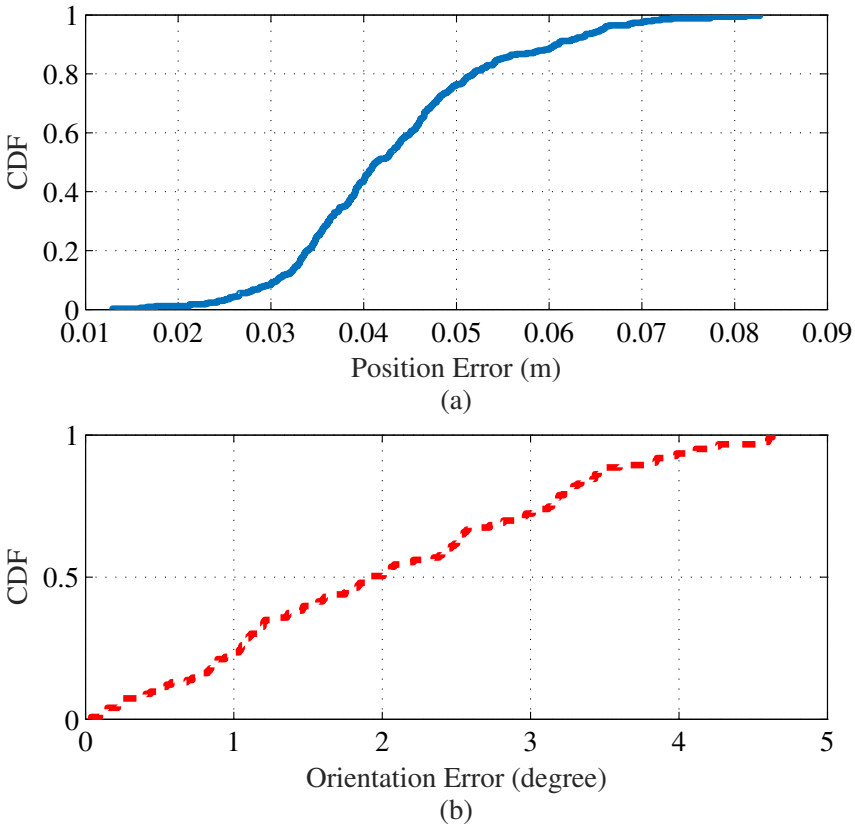


Figure 3.13: (a) CDF of the controller position estimation error; (b) CDF of the controller orientation error.

m. Additionally, as shown in Fig. 3.13(b), we can see that 60% the orientation errors are less than 2.5°. Moreover, for the pose tracker, almost 90% of the orientation estimations achieve an error under 3.5°. Obviously, regardless of position and orientation estimation, the proposed pose tracker of RFHUI is sufficiently accurate for most practical human-UAV interaction scenarios. Also note that similar to the result in the of RFID tags tracking experiment, due to the greater number of sample points and increased complexity of the moving trajectory in this experiment, the average error in both position and orientation are a little bit higher than that in the benchmark experiments.

3.4.3 Overall System Performance

Finally, we conducted an experiment in our indoor lab environment to demonstrate the feasibility of our system in a real-time manner. The typical experimental environments are shown in Fig. 3.14 and Fig. 3.15. The complex indoor environment, with intricate features and layouts

Figure 3.14: The empty lab environment.

with shelves, clothes stands, and furniture as shown in Fig. 3.15, requires our proposed RFHUI system to provide an accurate and robust control method to safely operate the UAV indoors. During this experiment, a user holds the controller, which is attached with 3 RFID tags, to control the UAV. We compared the ideal movement trajectory of the UAV, which is amplified by the trajectory of the controller, and the actual movement of the UAV to illustrate the performance of the proposed RFHUI.

A typical experiment result is presented in Fig. 3.16. The movement of the controller follows a random trajectory, which is illustrated by the black curve in Fig. 3.16. The blue curve represents the trajectory of the controller. The red curve denotes the ideal trajectory of the UAV, which is an amplified version of the trajectory of the controller. Clearly, we can tell that the UAV precisely follows the ideal trajectory, with only tiny disturbances around the ideal occurred. This is caused by the inherent errors of the UAV, especially, when the UAV is in a hovering mode. It is apparent that our RFHUI system achieves high accuracy in real-time navigation. This experiment validates that our RFID-based controller strategy is robust and practical. This is mainly due to the fact that our proposed RFHUI system can provide a highly accurate pose estimation, which plays a critical role in UAV navigation.

Figure 3.15: The cluttered lab environment.

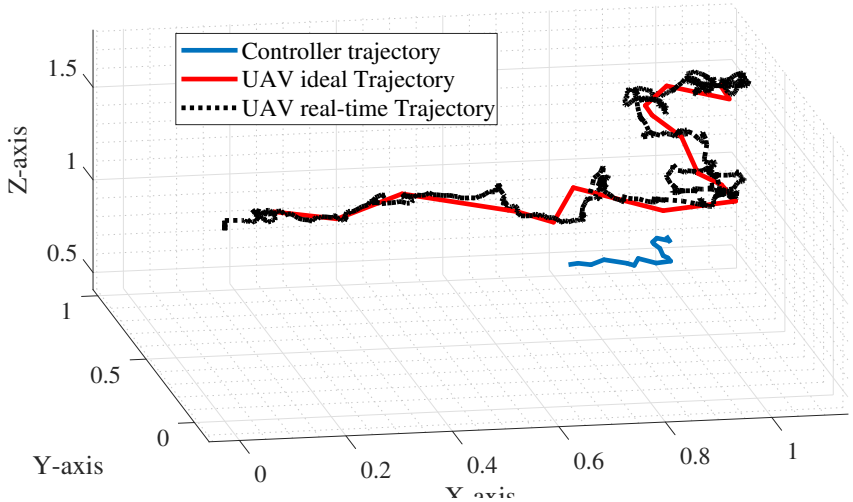


Figure 3.16: Trajectory comparison

3.5 Conclusions

In this paper, we proposed the RFHUI, an RFID based system for navigation control of a UAV using a COTS RFID reader. We experimentally validated the feasibility of utilizing an RFID localization-based method as the core of the UAV controller. We leveraged a Bayesian filter to estimate the location of RFID tags using the phase information in RFID tag responses. Then

an SVD algorithm was employed for data pre-processing to track the pose of the controller. Finally, the control module converted the pose data into flying control commands to achieve UAV navigation control in real-time. The extensive experiments in a representative lab environment demonstrated the capability of the proposed RFHUI system. To the best of our knowledge, the proposed RFHUI is the first practicable UHF passive RFID based UAV navigation control system, which provides a promising method for Human-UAV interaction.

Chapter 4

IADRL:Imitation Augmented Deep Reinforcement Learning Enabled UGV-UAV Coalition for Tasking in Complex Environments

4.1 Introduction

The last decade has witnessed significant developments in unmanned aerial vehicle (UAV) and unmanned ground vehicle (UGV) technologies, which have enabled their wide deployment for various applications, such as surveillance, search and rescue, inspection [1], inventory counting [2, 3], and more [4, 5, 111, 8, 9]. Recently, researchers have shown a growing interest to deploy them for more complex tasks that require multiple UAVs or UGVs to cooperatively work together to improve efficiency [24]. Most of the existing research focuses on cooperation in a multi-agent (or multi-robot) system that consists of a group of UAVs or UGVs. For example, Koubâ et al. introduced COROS [112], a high-level conceptual architecture for multi-agent UGV/robotic systems that represents a generic architecture for cooperative multi-agent applications. A cooperative architecture for the navigation of a swarm of robots based on Dynamic Fuzzy Cognitive Maps was introduced in [83, 81, 113], which allows for the development of homogeneous autonomous robot navigation without a global controller. A multi-UAV system was introduced in [24] to optimize target assignment and path planning. In addition to these homogeneous systems, some works went further to create a system that consists of heterogeneous agents/robots with different capabilities. For example, Das et al. in [114] introduced a distributed algorithm for task allocation in a system of multiple heterogeneous, autonomous robots deployed in a healthcare facility.

There are some essential limitations for both UGVs and UAVs. For example, a UGV has limited vertical detective/access capability, and a UAV is restrained by inadequate operation

range and time due to its limited power supply capacity. These limitations impede them in many applications. For instance, a ground robot proposed in [2] failed to perform inventory counting of items stored on high racks. Recently, UAVs have been expected to be widely deployed for disaster relief (e.g., survey, search and rescue, and providing network access). However, the authors of [115] found that a UAV's limited fight time (usually 20-30 minutes) greatly reduces their operating range. Obviously, for the above scenarios, we cannot solve the problem by simply deploying a swarm of UGVs or a swarm of UAVs alone. Alternatively, pairing them as a complementary team would help to overcome these constraints for tasks that UGVs or UAVs would be incapable of completing alone. However, an effective and low-cost strategy for implementing such a complementary UGV-UAV coalition is lacking.

To remedy these limitations, this paper presents an innovative method, named Imitation Augmented Deep Reinforcement Learning (IADRL), that enables a UGV and a UAV to form a coalition that can complement each other for complex tasks. The complementary UGV-UAV coalition can be deployed for applications that are usually incapable of being completed by a UGV or UAV alone. Using the disaster relief scenario as an example, an IADRL-enabled coalition can be deployed for autonomous search-and-rescue tasks. In the chaotic and hazardous environment following a disaster, a powerful UGV can autonomously carry a UAV to remote destinations usually out of the UAV's flight range. Additionally, the UGV provides communication and a power supply that greatly extends the operational range of a resource-constrained UAV, and the UAV helps the UGV with finding the best route and navigating through complex terrains that are out of the UGV's navigational capability (e.g., vertically unreachable or invisible to the UGV). To ensure that the coalition can successfully and effectively accomplish tasks, the cooperation of its agents (i.e., the UGV and UAV) must follow an underlying and complex model that varies depending on the task or operating environment.

The proposed IADRL model can learn the complementary features of UGV-UAV from a demonstration dataset that is collected from a simple and imperfect scenario. The model also learns a policy that responds to the environment, such as collision avoidance when around obstacles and other agents. Based on observations of the UAV and UGV, the IADRL model provides a series of actions for the UGV and UAV that ensures an optimized and complimentary

strategy for a given task. Additionally, we extend the IADRL to support multiple UGV-UAV coalitions working together within the same space. To the best of our knowledge, this is the first work to focus on creating such a coalition of robots with complementary capabilities for task completion, where a single agent in the team alone is incapable of completing. In a complex scenario, a task is executed by the first agent, and then another agent must continue the task based on the previous agent's success. Thus, the actions of all agents in the coalition are dependent upon each other, and agents must work as a complementary, cooperative team.

In the remainder of this paper, we discuss related work in Section 4.2, introduce and analyze the proposed IADRL model in Section 4.3, present our experimental study in Section 4.4, and conclude our work in Section 4.5.

4.2 Related Works

4.2.1 Imitation Learning

Imitation learning methods focus on the problem of learning and perform a task by learning from demonstration data. These methods can be roughly divided into three categories: Behavior Cloning (BC; or supervised learning) [116, 117], Inverse reinforcement learning (IRL) [118], and Generative Adversarial Network (GAN) imitation learning [119].

Behavior Cloning (BC)

This type of imitation learning was motivated by humans' tendency to learn skills by imitating the behaviors of others and has been widely used in autonomous driving [120, 121], wireless communication [122, 123, 124, 124, 125], and smart grids [126, 127]. In BC, agents receive instructions from a hand-crafted demonstrator (which serves as training data) and then replicate actions from the expert policy. BC is able to imitate the demonstrator immediately without any interaction with the environment. However, these agents cannot handle situations that are not included in the demonstrator. Furthermore, when the agents are limited in capacity, wrong or unnecessary behavior may be replicated. The method is simple but is useful only with large amounts of high-quality training data. Additionally, because agents merely learn single-step

decisions, the compounding error accumulation caused by the covariate shift problem could lead to a large learning deviation.

Inverse Reinforcement Learning (IRL)

In a classic Reinforcement Learning (RL) setting, the ultimate goal is for an agent to learn a decision process to generate behaviors that could maximize accumulated rewards by some predefined reward functions. As demonstrated by Ng et al. in [128], IRL is given the observed agent's behaviors and observations of the environment to infer the optimal reward function. IRL generally has a reward function that is difficult to accurately quantify, and another system has to be able to complete the tasks well to offer instructions for the model. The difference between IRL and BC is that IRL generates a reward function to infer an optimal policy instead of using a fixed replication policy.

GAN for Imitation Learning

Ho and Ermon proposed Generative Adversarial Imitation Learning (GAIL) in 2016 [119]. They introduced the idea of a GAN combined with imitation learning. Unlike GAN, GAIL does not have an explicit Generator that acts as the policy of agents. Learning in GAIL is divided into two steps. First, to train the Discriminator adversarially with the data obtained from the current policy sampling and expert data. Second, the Discriminator serves as the replaced reward function to train the policy. GAIL is superior for large-planning and high-dimensional problems as compared to BC and IRL.

4.2.2 Multi-Agent System Planning and Control

This is a hot topic that has attracted considerable research interest in recent years. The existing studies have mainly focused on operating multiple UGVs/robots and UAVs in the same environment. For example, Sariel-Talay et al. proposed a multi-robot cooperation framework to solve complex tasks in a cost-efficient manner [129]. Swarm intelligence is inspired by social animals and aims to form the behavior of many decentralized autonomous cooperative agents. For example, Wang et al. solved the multi-robot task allocation problem using an ant colony

algorithm [130]. In recent years, RL has become extremely trendy in the field of multi-agent systems. In [24], the author presented an innovative artificial intelligence method combined with a well-known RL method, the Multi-Agent Deep Deterministic Policy Gradient Algorithm, to solve path planning and task allocation problems in dynamic environments. However, these existing methods have never been applied to a coalition of multiple UGVs/robots and UAVs before.

Few studies have considered the use of multiple UGVs and UAVs simultaneously to solve complex tasks in dynamic environments. For example, Ghamry et al. proposed an algorithm that controls UAV's autonomous take-off, tracking, and landing with a UGV [131]. They also presented an interesting study on forming a team of cooperating UAVs-UGVs for forest monitoring and fire detection [132]. Khaleghi et al. studied the team formation approach of multiple UGVs and UAVs [133]. The author in [134] introduced an auction-based approach for applying an estimated utility to task assignment for heterogeneous, multi-agent teams. But these studies only focus on one area (i.e., team formation or task allocation) because of the huge computational cost and the communication difficulties between agents. Meanwhile, some companies (e.g., Quanser Inc.) provide a variety of mobile robots and UAV swarm systems, but none of them focus on creating a UGV-UAV coalition for complex tasks. Unlike these existing methods, our proposed approach creates a coalition that enables a UGV and a UAV to complement each other during complex tasks that are incapable of being completed by a single UGV or UAV or by a swarm of UGVs or UAVs alone. This approach not only concerns the optimization of path planning, but also learns an underlying complementary model for the agents from a set of non-optimized demonstration data.

4.3 The Proposed Approach

Our proposed IADRL approach enables a coalition consisting of a UGV and UAV to complement each other for complex tasks. Additionally, we extend IADRL to include a system of multiple UGV-UAV coalitions working together.

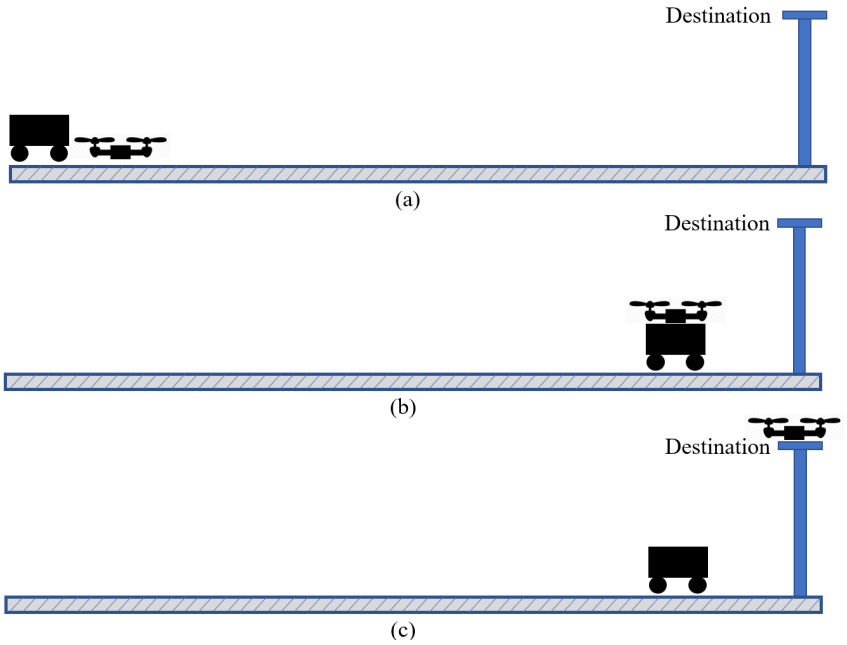


Figure 4.1: An example of a UGV-UAV complementary coalition for task completion: (a) the target destination is too far for the UAV to reach, while too high for the UGV alone, (b) the UGV carries the UAV closer to the destination, and, finally, (c) the UAV flies to the high-altitude destination.

4.3.1 IADRL Enabled UGV-UAV Coalition

Problem Definition and Challenges

There are several essential limitations of UGVs and UAVs that prevent them from being deployed for some tasks. Fig. 4.1 illustrates a motivating scenario where rescue teams must reach a high-altitude position. The UAV is capable of reaching that position; however, the destination is too far for it to fly from the starting point with its limited battery capacity. Alternatively, the UGV can move closer to the destination but is incapable of climbing up the high altitude. An intuitive idea to reach the destination is to pair the UGV and UAV together as a coalition that complements each other: the UGV can carry the UAV closer to the destination, and then the UAV launches from the UGV and flies to the target.

Motivated by the Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs) [135], this UGV-UAV complementary coalition for task completion with minimum cost can be described by the tuple $< \varepsilon, \mathbf{o}, \mathbf{a}, r, \gamma, \mathcal{M} >$, where ε denotes the environment the coalition will interact with; $\mathbf{o} = (o_1, o_2)$ is the joint observations of the coalition, and consists

of the UGV's observation, o_1, and the UAV's observation, o_2; $\mathbf{a} = (a_1, a_2)$ denotes the joint actions of the UGV, a_1, and UAV, a_2, in the coalition; r is the reward function of the coalition while joint actions $\mathbf{a}$ impose ε with joint observations $\mathbf{o}$; $\gamma \in [0, 1)$ is a discount factor for future rewards; and $\mathcal{M}$ defines the complementary cooperation model of the UGV and UAV. To achieve successful task completion, the UGV and UAV must collaborate with and complement each other; thus, their joint actions satisfy $\mathbf{a} = (a_1, a_2) \sim \mathcal{M}$.

The goal of IADRL is to learn a joint value-action function $Q_c^{\pi}(\mathbf{o}, \mathbf{a}; \theta)$ that enables a complementary UGV and UAV coalition to achieve maximum overall rewards (or minimal overall costs) while accomplishing various tasks. The equation for this complementary coalition is formulated as (4.1):

$$\underset{\mathbf{a} \sim \pi}{\operatorname{argmax}} \; Q_c^{\pi}(\mathbf{o}, \mathbf{a}; \theta) \tag{4.1}$$

$$\text{s.t. } \mathbf{a} = (a_1, a_2) \sim \mathcal{M}, \tag{4.2}$$

where θ is the parameter of the value-action function Q_c^{π}. Note that $\mathbf{o}$ and $\mathbf{a}$ represent the joint observations and actions in the coalition, and the joint actions follow an underlying model, $\mathcal{M}$, that complements each action during tasks. To explicitly model the underlying complementary cooperation model, $\mathcal{M}$, of the UGV-UAV coalition during tasks is difficult and, at least, requires significant effort and expertise.

We faced several challenges when creating the IADRL model under these requirements. For our method to successfully complete generic and complex tasks, we have to develop a straightforward way to represent the coalition's complementary cooperation model. Equation (4.1) shows that the proposed network has to learn an optimized policy, π, for UGV-UAV joint actions. Reference [136] suggests that the joint-action space increases exponentially with the number of agents. Consequently, it is difficult for deep reinforcement learning (DRL) methods to reach the optimized policy, π, in such huge searching space. Furthermore, the trained policy, π, not only needs to provide optimized actions for task execution, but also needs to follow the underlying model $\mathcal{M}$ to enable the UGV-UAV coalition to successfully complete tasks. State-of-the-art methods such as Value-Decomposition Networks (VDN) [137] and

QMIX [136] require that the actions of agents at the same time step are independent so they can be factorized. Obviously, this assumption does not hold true for the UGV-UAV coalition. Additionally, it is necessary to train the proposed model in a continuous-action space that empowers the UGV-UAV coalition's operation in complex environments. This further increases the size of the joint-action space and challenges the training of the IADRL model.

The IADRL Model

To tackle the above challenges, first, instead of explicitly modeling the collaboration between the UGV and UAV, we captured their complementary cooperation using a set of demonstration data. The dataset was collected by manually controlling the UGV-UAV coalition to complete several simple tasks. The demonstration data do not need optimization, but only a set of the most basic and important rules of the collaborative and complementary actions. As such, our method needs to teach the coalition just as one would teach a new sports skill to a team of kids, by showing them how to play through imitation.

Therefore, we design IADRL by combining an imitation model with a DRL model. The architecture of IADRL is presented in Fig. 4.2. The imitation model and the DRL model are contained in a pink block and green block, respectively. The imitation model learns the cooperative features, $\mathcal{M}$, of complementary cooperation from the non-optimized demonstration dataset and augments the DRL model's training to develop an optimized strategy. As such, we learn the optimized policy, π, while following the complementary cooperation model. Meanwhile, the DRL model also learns a strategy to respond to dynamic environments, such as avoiding collisions with obstacles and other UGVs and UAVs.

The Imitation Model The imitation model is inspired by the study of GAIL [119], and it is based on a GAN [138] architecture that comprises two basic entities: a discriminator, $\mathcal{D}$, and a generator, $\mathcal{G}$. Discriminator $\mathcal{D}$ is created to distinguish between the "expert" data and the data produced by generator $\mathcal{G}$. Additionally, $\mathcal{D}$ and $\mathcal{G}$ are simultaneously trained in an adversarial way: $\mathcal{G}$ is updated to produce "counterfeited" data that could pass the detection of $\mathcal{D}$, while $\mathcal{D}$ is improved to distinguish the "counterfeited" data from the true "expert" data. The resulting

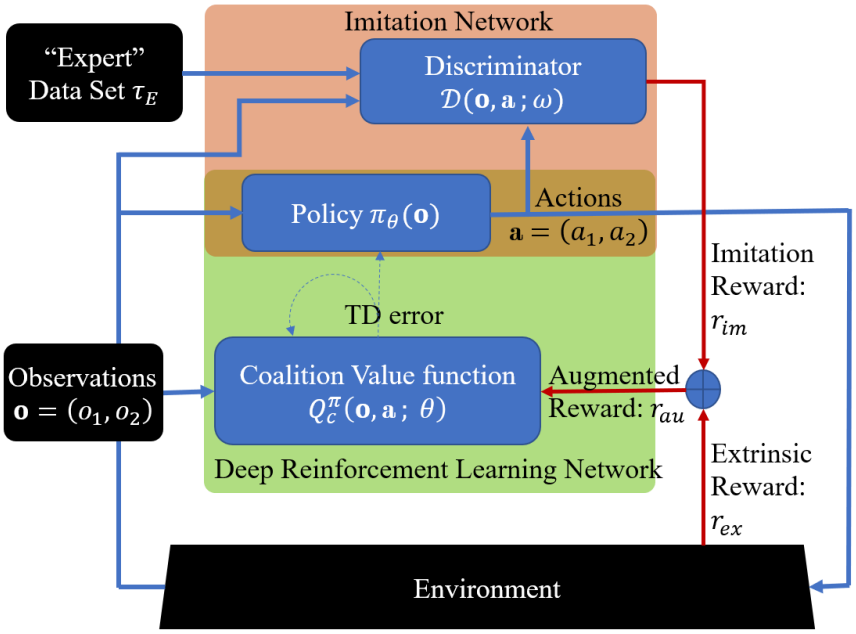


Figure 4.2: The architecture of the IADRL model.

competition drives both entities to improve their capabilities. Thus, a well-trained imitation model not only generates data with almost the same distribution of the "expert" data, but also precisely measures the similarity of any given data with the "expert" data.

Different from the original GAIL model, we replaced the Trust Region Policy Optimization-based [139] generator with the latest Proximal Policy Optimization (PPO)-based [140] generator, which also serves as the policy, π, of the DRL model. Thereby, the term generator $\mathcal{G}$ and policy π will be used interchangeably in the rest of this paper. Policy π has two roles in our IADRL, as it not only generates actions following the distribution of the "expert" data, but also reacts to the environment with an optimized strategy. The details of policy π will be introduced when we discuss the DRL model. Here, we focus on the discriminator, $\mathcal{D}(\mathbf{o}, \mathbf{a}; \omega)$, of the imitation model.

In our imitation model, $\mathcal{D} : \mathbf{O} \times \mathbf{A} \mapsto (0, 1)$ is a discriminator function with weight ω, and $\mathbf{O}$ and $\mathbf{A}$ are the observation and action space, respectively, of the UGV-UAV coalition. We implement the discriminator $\mathcal{D}$ with a deep neural network, which is a fully connected neural network with M_D hidden layers. Each hidden layer has the same number of N_D units. The size of the input layer is determined by the size of the concatenated input $(\mathbf{o}, \mathbf{a})$. The size of $\mathcal{D}$ can

be configured using N_D and M_D. Usually, a larger-sized network is required if the UGV-UAV coalition is deployed for more complex environments and tasks.

During the training process, we can improve the discriminator $\mathcal{D}$ by maximizing the following value function:

$$\begin{aligned}\mathcal{V}(\omega) = \mathbb{E}_{\pi}[\log(\mathcal{D}(\mathbf{o}, \mathbf{a}; \omega))] + \\ \mathbb{E}_{\tau_E}[\log(1 - \mathcal{D}(\mathbf{o}, \mathbf{a}; \omega))] - \lambda H(\pi),\end{aligned} \tag{4.3}$$

where $H(\pi)$ represents the causal entropy [141] of π defined as $H(\pi) \equiv \mathbb{E}_{\pi}[-\log \pi(\mathbf{a}|\mathbf{o})]$, and it severs as a policy regulator to make the distribution of policy as evenly as possible; $\lambda \geqslant 0$ is the discount factor of H; and τ_E refers to the "expert" policy provided by a demonstrated dataset with length N, i.e., $\tau_E = [\eta_1, \eta_2, ...\eta_N]$. Here $\eta_n = [(\mathbf{o}^0, \mathbf{a}^0), (\mathbf{o}^1, \mathbf{a}^1), ..., (\mathbf{o}^T, \mathbf{a}^T)]$ is the record of an episode with T steps. It represents the model of the complementary cooperation between the UGV and UAV; thus, $\tau_E \sim \mathcal{M}$. Again, τ_E is not a perfect policy, but is collected from a few sample scenarios in controlled settings navigated by manual control and is, therefore, considered to be the "expert."

Equation (4.3) is derived from the objective function of GAIL [119]. It shows that during the training process, as discriminator $\mathcal{D}$ is updated to increase $\mathcal{V}(\omega)$, its ability to detect the similarity of a policy and the "expert" data is improved. When it produces a lower value for a given action, $\mathbf{a}$, it indicates that the chance of action $\mathbf{a}$ is higher from the "expert" data, and thus, shows with higher confidence that it is following the underlying complementary model, $\mathbf{a} \sim \mathcal{M}$.

The DRL Model The proposed IADRL model must not only learn the complementary cooperation model, but must also react to the dynamics of an environment and provide an optimized navigation strategy for the UGV-UAV coalition. To this end, we created the DRL model based on a PPO network [140] with an actor-critic architecture, which enables the model to produce continuous actions for the UGV-UAV coalition during task completion in complex environments. The proposed DRL model consists of two separate components: an actor (i.e., policy π) and a critic (i.e., value function Q_c^{π}). Policy π is responsible for generating action $\mathbf{a}$ based on

the given observation $\mathbf{o}$. Additionally, policy π is learnt by a neural network from the training and history data. The value function, Q_c^π, processes the received rewards and evaluates the current action prescribed by policy π.

We implement Q_c^π and π using two deep neural networks that are both fully connected networks with M_π hidden layers for π and M_Q hidden layers for Q_c^π. Each hidden layer has the same number of N_π and N_Q units for the π and Q_c^π networks, respectively. The size of the input layers is determined by the size of the input vectors. The size of the output layer of π is determined by the size of the joint action, $\mathbf{a}$, of the coalition. As in the case of discriminator network $\mathcal{D}$, usually larger-sized networks are required for π and Q_c^π if the UGV-UAV coalition is deployed for more complex environments and tasks.

Ultimately, the goal of training the DRL model is to maximize the UGV-UAV coalition's state-value function Q_c^π for a given policy π, given by

$$Q_c^\pi(\mathbf{o}, \mathbf{a}; \theta) = \mathbb{E}[r_{au}(\mathbf{o}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi}[Q_c^\pi(\mathbf{o}', \mathbf{a}')]], \tag{4.4}$$

where θ is the parameter of function Q_c^π; $\gamma \in (0, 1]$ is the discount factor for future rewards; r_{au} is the augmented reward function, given by

$$r_{au}(\mathbf{o}, \mathbf{a}) = \beta \cdot r_{im}(\mathbf{o}, \mathbf{a}) + (1 - \beta) \cdot r_{ex}(\mathbf{o}, \mathbf{a}), \tag{4.5}$$

where $\beta \in (0, 1)$ represents the confidence weight of the "expert" demonstration data, and a larger β can be deployed if τ_E is closer to the optimized policy; r_{ex} is the reward function that comes from the environment, which is the same as a traditional Markov Decision Processes (MDPs) environment; additionally, r_{im} is the reward function of the imitation model and measures how similar the coalition's joint actions $\mathbf{a}$ are with the "expert policy," as

$$r_{im}(\mathbf{o}, \mathbf{a}) = \log(1 - \mathcal{D}(\mathbf{o}, \mathbf{a}; \omega)). \tag{4.6}$$

During the training, we aim to increase Q_c^π. Equations (4.3), (6.4), and (4.6) show that as we increase Q_c^π, we decrease the value of $V(\omega)$. Thus, from the results of [119], we increase the

similarity of the policy, π, and the "expert" dataset, τ_E, as we increase Q_c^π. Note that our goal is not to train the policy, π, to copy τ_E, but to learn the complementary cooperative model that underlays τ_E while maximizing the extrinsic reward. Therefore, we introduced the confidence parameter β to augment the learning process by trading-off between learning from expert data and the environment. Alternatively, r_{ex} guides the IADRL to learn a strategy that reacts to the environment. Its configuration is straightforward and lists several rules for the coalition when interacting with the extrinsic environment. Usually, we can assign a penalty for the coalition if any agent collides with either an obstacle or other agent. This way, a trained Q_c^π enables the UGV and UAV to choose the action that does not cause a collision. We can also assign a small penalty for every step taken by each agent, and this enables Q_c^π to provide the coalition's best navigational route for reaching a target. Here, the best route is the one with the lowest sum of navigational costs of the UGV and UAV. Note that the cost of operating a UAV is usually higher than that of the UGV. Additionally, an example of r_{ex} will be provided in the later experimental section. The value function Q_c^π of the proposed DRL network can be trained end-to-end by minimizing the following loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{a}\sim\pi}[y - Q_c^\pi(\mathbf{o}, \mathbf{a}; \theta)]^2, \tag{4.7}$$

where $y = r_{au} + \gamma \cdot \max_{\mathbf{a}'}[Q_c^\pi(\mathbf{o}', \mathbf{a}'; \theta^-)]$ and θ^- are parameters trained by the previous iteration. During the training, we try to decrease the stochastic gradient of (6.6) with respect to θ. Then, a trained state-value function Q_c^π precisely evaluates action $\mathbf{a}$ of the UGV-UAV coalition.

From (6.4) and (6.5), we know that as long as a policy, π, is found that guides the UGV-UAV coalition to achieve a higher cumulative Q value, the proposed IDARL network will enable the complementary cooperation between agents and find the best strategy to accomplish a given task. To better explain the process of updating the policy in our PPO-based DRL model, we introduce an additional objective function with respect to the φ weighted policy, π_φ, as:

$$J(\varphi) = \mathbb{E}_t \left[\min \left(\frac{\pi_\varphi(\mathbf{o})}{\pi_{\varphi_{old}}(\mathbf{o})} Q_c^{\pi_{\varphi_{old}}},\ f(\epsilon, Q_c^{\pi_{\varphi_{old}}}) \right) \right], \tag{4.8}$$

Algorithm 1: The Training Procedure of IADRL

```
Input: "Expert" dataset τ_E, and initial parameters ω_0 and θ_0 ;
for episode i = 1 to M do
    Sample training dataset π_i ;
    Update discriminator D by ascending the stochastic gradient of (4.3) with respect to ω ;
    Update value function Q_c^π of the DRL by decreasing the stochastic gradient of (6.6) with
      respect to θ;
    Update policy π_φ of the DRL by ascending the stochastic gradient of (4.8) with respect to
      φ;
end
```

where ϵ is a hyper-parameter set to 0.1 or 0.2; $\pi_{\varphi_{old}}$ and π_{φ} denote the policy before and after the training update, respectively; and $f(\cdot)$ is a clip function defined as:

$$\begin{cases} f(\epsilon, Q) = (1+\epsilon)Q, & \text{if } Q > 0 \\ f(\epsilon, Q) = (1-\epsilon)Q, & \text{if } Q < 0. \end{cases} \tag{4.9}$$

The training process aims to maximize $J(\varphi)$ by ascending the stochastic gradient of (4.8) with respect to φ. Thus, policy π tends to provide actions that can impose higher Q values. During the training, (4.9) limits the updated range of π_{φ} so that it remains close to the last policy, $\pi_{\varphi_{old}}$. This greatly improves training stability by avoiding too much of a policy update in one step.

We summarize the training process of IADRL in Algorithm 1. During the training process, we recursively update discriminator $\mathcal{D}$ of the imitation model to provide a more accurate evaluation of how good the complementary cooperation is between the UGV and UAV. Then, the value function, Q_c^{π}, is updated to enable the model to precisely assess the joint-action, $\mathbf{o}$, of the coalition as compared to the extrinsic environment and the intrinsic complementary cooperation model. Last, IADRL updates policy π that provides a series of actions to accomplish given tasks and to receive higher cumulative Q values. Thus, a well-trained IADRL model enables the UGV-UAV coalition to follow the complementary model, $\mathcal{M}$, and provides an optimized strategy when the coalition is deployed for various tasks.

4.3.2 Multi-Coalition Systems

Our IADRL model can be easily extended to support a system with multiple UGV-UAV coalitions. This system follows the traditional Dec-POMDPs, and the coordination among the coalitions is loose and satisfy the model of VDN [137]. Therefore, the global joint-action value function, denoted by Q_g, of a system with N coalitions can be represented as:

$$Q_g(\mathbf{s}, \mathbf{u}) = \sum_{i=1}^{N} Q_{c\,i}^{\pi}(\mathbf{o}_i, \mathbf{a}_i; \theta_i), \tag{4.10}$$

where $\mathbf{o}_i = (o_{1i}, o_{2i})$ and $\mathbf{a}_i = (a_{1i}, a_{2i})$ denote the joint observations and actions, respectively, of the UGV and UAV in coalition i. Additionally, $\mathbf{s} = (\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_N)$ and $\mathbf{u} = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_N)$ refer to the joint observations and actions, respectively, for all N coalitions in the system. A joint observation, $\mathbf{s}$, is created by concatenating all observations, $\mathbf{o}_i$, from all the coalitions. Equation (4.10) indicates that based on the current joint observation, $\mathbf{s}$, we find a best joint-action for the system, $\mathbf{u}$, by decomposing the problem and finding all of the best joint-action, $\mathbf{a}_i$, for each coalition, which is determined by the trained IADRL model based on its observation $\mathbf{o}_i$.

The UGV-UAV coalition requires wireless communications to function well. From (4.1), the optimized policy, π, of the coalition requires joint observation and joint action data, which are created by the observations and actions from both the UGV and the UAV. Thus, wireless communications within the coalition is essential for sharing this information. On the other hand, communications among UGV-UAV coalitions is not mandatory. In (4.10), it is shown that the global joint-action value function, Q_g, is the sum of individual coalition value functions, Q_c^{π}, which is conditional on the coalition's observations and actions. Therefore, a decentralized optimized policy for a system with multiple coalitions can be achieved when each coalition selects its own optimized policy, π, from a trained IADRL model without sharing information among coalitions.

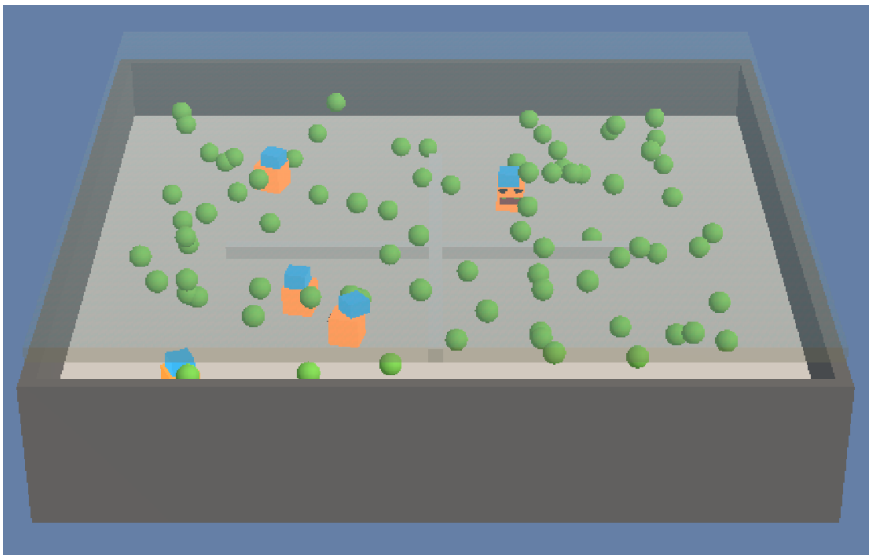


Figure 4.3: Basic simulation experimental setup for five UGV-UAV coalitions performing tasks cooperatively using the IADRL system. The UGV-UAV coalitions are marked as orange (UGV) and blue (UAV) block pairs; the tasks are marked as green balls.

4.4 Experimental Study and Discussions

4.4.1 Experiment Configuration

Simulation Platform

We designed a simulation training and evaluation platform for the IADRL system based on the Unity3D ML-Agents platform [142]. The platform is illustrated in Fig. 6.5. It is designed to simulate the scenario of deploying UGV-UAV coalitions in a giant, high-bay warehouse crowded with high racks and shelves. The coalitions are tasked with reaching given targets to mimic item scanning applications (i.e., RFID or barcode) in indoor spaces. The platform's dimension is $50 \times 50 \times 7$ m^3, and is divided into 4 sub-zones by cross shaped obstacles. As Fig. 6.5 depicts, orange agents represent UGVs, blue agents represent UAVs, and the green spheres suspended in air (they are actually on different levels of racks in this space) represent given targets.

We implemented our IADRL model using Tensorflow on a computer with an Intel 9900K CPU and two Nvidia 2080 GPUs. We conducted each experiment with the same IADRL configuration: the discriminator, $\mathcal{D}$, has $M_D = 2$ hidden layers and $N_D = 128$ units per layer; the

coalition value function, Q_c^{π}, has $M_Q = 3$ hidden layers and $N_Q = 512$ units per layer; the policy, π, has $M_\pi = 3$ hidden layers and $N_\pi = 512$ units per layer. In the following experiments, we deployed 5 UGV-UAV coalitions. Their initial positions and the positions of all targets were randomly generated.

The observations (or states of the environment) are collected by each agent's Ray-cast sensor, which is provided by Unity3D. Similar to a Lidar sensor (e.g., the RPLidar laser scanner), the Ray-cast sensor casts rays into the surrounding environment, and the feedback is a vector that provides the position of all detected objects and their distances. A UGV's Ray-cast sensor detects only in the horizontal direction (to identify obstacles on the floor), while a UAV casts rays towards the horizon, and upward and downward within 45 vertical degrees. The maximum detection range of all Ray-cast sensors is set to 20 meters with a 20-Hz refresh rate. A UGV-UAV coalition's observation, $\mathbf{o}$, is created by concatenating all of the observation vectors of its UGV and UAV agents to form a new vector. The UGV's action is represented by $a_1 = [a_x, a_y]$, and the UAV's action is represented by $a_2 = [a_x, a_y, a_z]$, where a_x, a_y, and a_z are accelerations in the x, y, and z direction, respectively. The UGV-UAV coalition's action, $\mathbf{a} = (a_1, a_2)$, is also created by concatenating a_1 and a_2 to form a new vector.

Extrinsic Rewards

The extrinsic rewards configuration is summarized in Table 4.1. They are designed to capture basically every condition that could be experienced when deploying UGV-UAV coalitions for item scanning tasks. Considering that the average battery life of a UGV is 5 to 10 times that of a UAV, we set the UAV's cost of each step to be 6 times that of the UGV. Thus, the UAV tends to ride on the UGV when transiting between positions, while simultaneously finding the best trajectory to reach the destination by trading-off from the ride-on to fly state. To encourage coalitions to complete tasks, we set the reward of reaching each target to 100 times that of the step cost for UAVs. Our intention is for the UGV to successfully scan all the targets within its reachable vertical height and define them as bad targets for the UAV. If the UAV mistakenly reaches a bad target, a penalty as big as the reward (i.e., 60) will be issued. Targets that are too high and out of the UGV's reach are considered good targets for the UAV.

Table 4.1: Extrinsic Rewards Configuration

Reward Items	*Reward Value*
UGV's step cost	-0.1
UAV's step cost	-0.6
UGV reaches a target	+60
UAV reaches a bad target	-60
UAV reaches a good target	+60
UAV collides with an obstacle	-60
UAV collides with another agent	-60
UGV collides with an obstacle	-30
UGV collides with another agent	-30
Final reward	+30

To ensure that the UGV and UAV avoid colliding into obstacles and other agents, the penalty for a collision is equal to the target reward (i.e., 60) for the UAV and half of that for the UGV. The reason for setting a lower penalty for the UGV is that UGVs are usually protected with anti-collision sensors or bumpers. When the coalition reaches all targets (or the given number of targets), it has completed the task and wins a final reward. We set the confidence weight β in (6.5) to 0.1 for the remaining experiments.

Demonstration Data Collection

The demonstration dataset τ_E is collected by manually controlling a UGV-UAV coalition through several simple scenarios that are displayed in Fig. 4.4. The dataset τ_E consists of 40 total episodes of completed tasks (10 tasks per scenario) according to the scenarios described in Fig. 4.4 (10 for each scenario). For the scenario shown in Fig. 4.4(a), a target is created within the reachable height of the UGV, and we controlled the coalition in a way that allowed the UGV to reach the target. In Fig. 4.4(b), a target at a higher place is generated for the UAV to reach. The UAV first rides on the UGV to move closer to the target, and then flies to the target to scan it. In Fig. 4.4(c), we create two targets, one for the UAV and the other for the UGV, in the same sub-zone. Again, we navigated the coalition so the UGV and UAV could reach their targets cooperatively. Fig. 4.4(d) is a scenario similar to the scenario in Fig. 4.4(c), but we

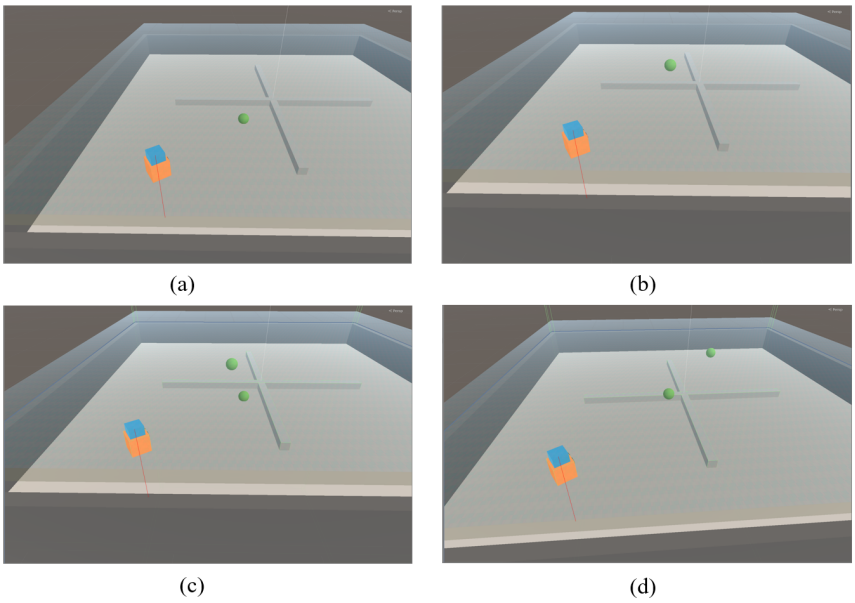


Figure 4.4: The scenarios that allow for collection of demonstration data τ_E: (a) a target (green ball) within reachable height of the UGV, (b) a target reachable only by the UAV, (c) one target each for the UAV and UGV to reach within the same sub-zone, (d) one target each for the UAV and UGV to reach, but within different sub-zones.

place the two targets in different sub-zones. Note that the targets in each scenario are generated randomly.

During this process, we manually controlled the coalition with some non-optimized strategies. For example, we do not optimize the route when moving towards any target. For the scenarios in Figs. 4.4(c) and 4.4(d), we do not consider the order of targets for optimizing the moving trajectory. Thus, τ_E serves as an instructor that guides all agents to learn complimentary behavior patterns rather than only copying the sample actions provided in the training stage.

4.4.2 Experimental Results

Training Process Results

In the training process, the maximum number of steps, st_{max}, for one episode is 1×10^5, which includes the steps of the UGV-UAV coalition. If the coalitions reach all of the targets, the training episode is terminated immediately and the final reward is received. Otherwise, it will

keep tasking until st_{max} is reached. As a baseline scheme for performance comparison, we implemented three existing models, including:

- the original GAIL model, termed GAIL, introduced in [119];
- the PPO model, termed PPO, presented in [140]; and
- a supervised learning method, termed BC (Behavior Cloning) from [117].

Moreover, to guarantee a fair comparison, we used the same training parameter (i.e., number of targets achieved, learning rate, maximum number of steps, etc.) for the three approaches.

First, we conducted several experiments with five coalitions using the four models. As shown in Fig. 6.6, the accumulated rewards of IADRL and PPO are convergent, while the GAIL and BC curves do not converge. Obviously, compared to the other three algorithms, the cumulative reward value of the IADRL approach is the highest and it is the most stable given the same reward settings. This result is consistent with our preliminary theoretical conjecture that GAIL only replicates the behaviors and policy offered by the demonstration dataset τ_E, rather than by the optimal policy for achieving higher rewards. Although the cumulative reward obtained by the PPO model is high and convergent, it cannot successfully complete all the cooperative tasks. This is because PPO is incapable of learning the complementary model between the UGV and UAV. Fig. 4.6 shows that all episodes of the PPO model are terminated when they reach the maximum number of steps, st_{max} = 10^5, and, thus, are incapable of successfully reaching all the targets. The task completion rate for the PPO model is consistently zero, indicating that the model is not able to provide an optimized policy that enables the UGV-UAV coalition to complete tasks exploiting complementary cooperation. Therefore, in the following section, we will not discuss the performance of PPO. Furthermore, Fig. 6.8 shows the training loss values during the training process. It is clear that the loss values of IADRL, GAIL, and BC are significantly minimized after st_{max} = 10^5 steps are completed.

Note that every training episode will be terminated if all targets are reached before the maximum number of steps are taken. Thus, the average steps to complete an episode varies for each models. To compare the models and better present the training process, the results in Figs. 6.6 and 6.8 are obtained with different numbers of steps for the models.

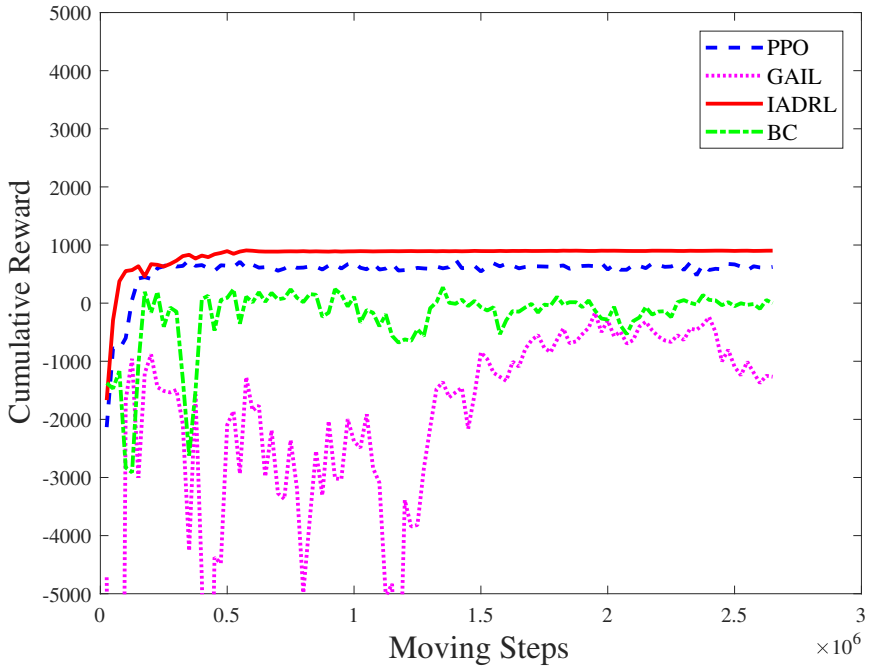


Figure 4.5: Accumulated training rewards values for PPO, GAIL, IADRL, and BC methods.

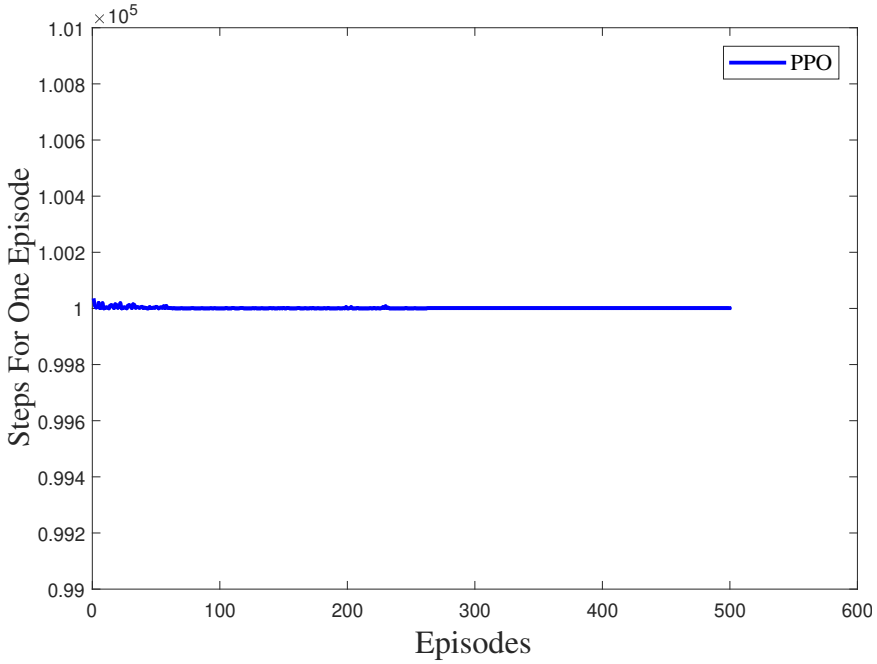


Figure 4.6: The number of steps taken before episodes are terminated for the PPO model.

Performance Analysis

To further prove the superiority of IADRL, we evaluate three additional indicators: (i) number of collisions in one episode, (ii) steps needed for completing one episode, and (iii) the overall task completion rate. Fig. 4.8 describes the task completion rate, denoted by $\Re_{task}$, for IADRL,

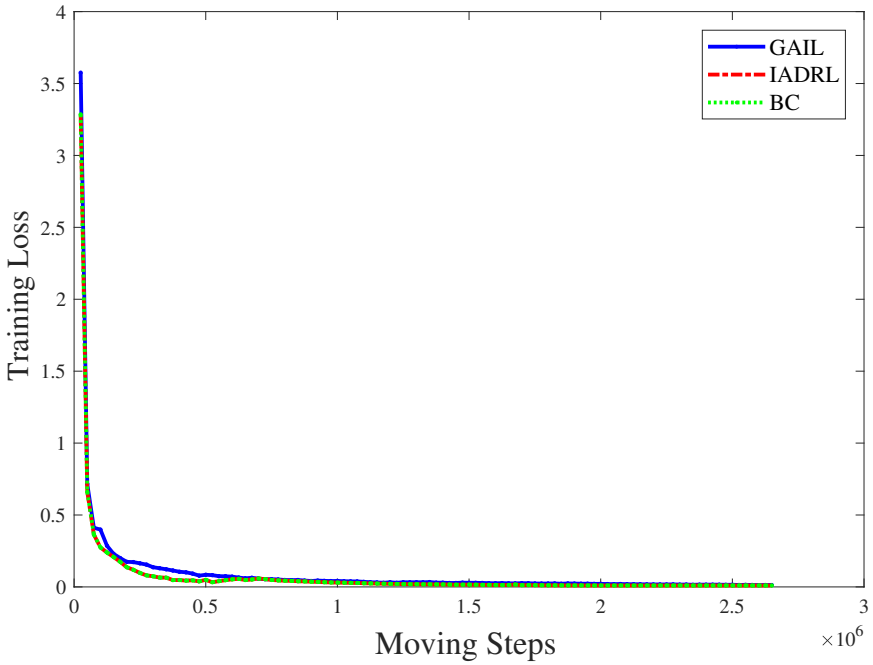


Figure 4.7: Training loss values of GAIL, BC, and IADRL methods.

GAIL, and BC. We defined the task completion rate as:

$$\Re_{task} = \frac{N_{failed}}{N_{total}}, \tag{4.11}$$

where N_{failed} is the number of episodes that the UGV-UAV coalitions fail to reach all targets, and N_{total} is the total number of completed episodes. For our task setting, the key point towards completing a mission is the complementary cooperation between UGVs and UAVs. Fig. 4.8 shows that the task completion rate $\Re_{task}$ of IADRL quickly converges to 1, which indicates that after it fails in the first several episodes, IADRL quickly learns the complementary model from τ_E and succeeds in all the subsequent episodes. The three curves close to each other illustrates that IADRL has a similar capability of learning a model from τ_E to that of GAIL and BC, which are designed to directly replicate the policy from demonstration data.

To evaluate the efficiency of tasking, we compare the number of steps taken to reach all the targets with these three schemes, and the results are presented in Fig. 4.9. Obviously, IADRL achieves the given tasks within 600 steps for each episode, which is far less than the number of steps GAIL and BC take given the same mission. Furthermore, the number of steps required for IADRL training is much more sustainable than that of GAIL and BC, as it reaches the

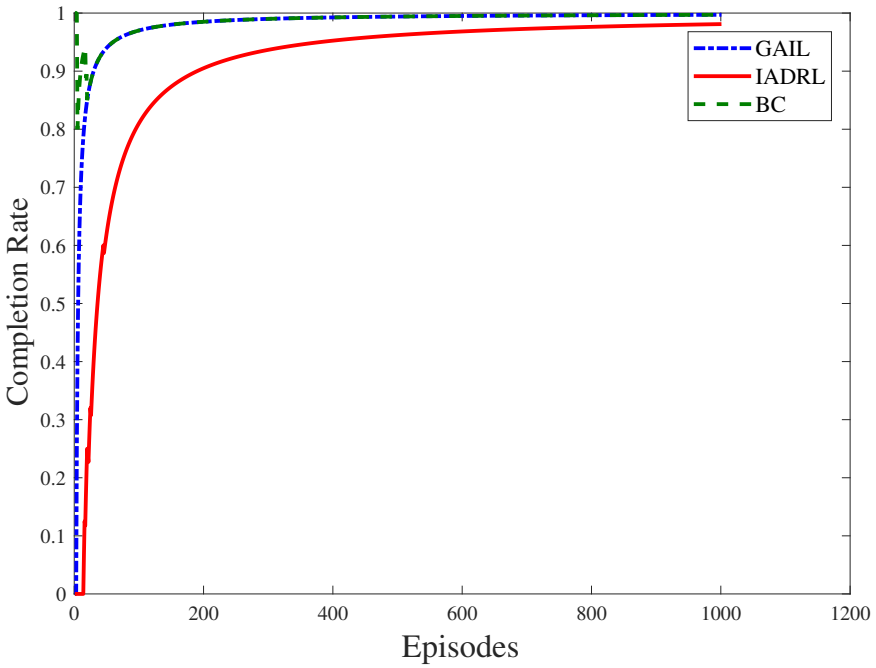


Figure 4.8: Task completion rate, $\Re_{task}$, for GAIL, BC, and IADRL methods.

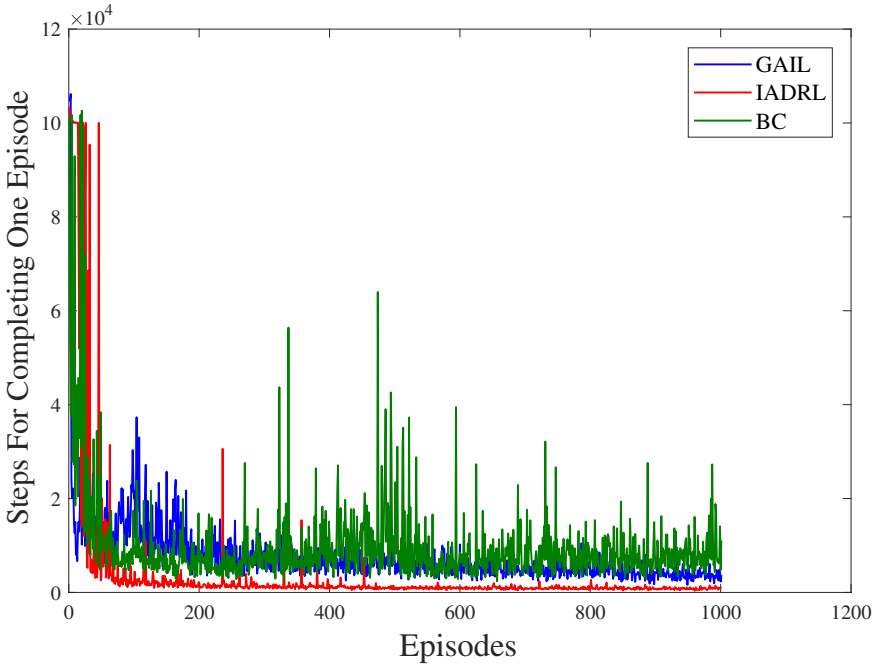


Figure 4.9: Number of steps needed to complete each training episode using IADRL, GAIL, and BC methods.

optimized policy within fewer episodes (around 200 training episodes). Furthermore, the BC method not only uses the most steps to complete tasks, but even at the end of the training, no convincing task completion strategies have been determined, as shown by the large fluctuations at the tail end of the BC curve.

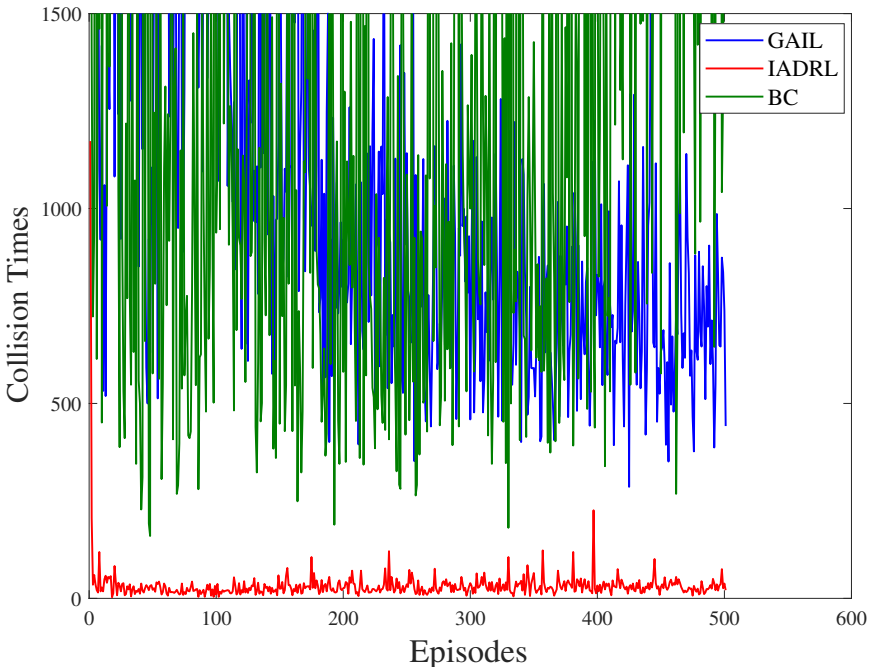


Figure 4.10: Total number of agent collisions with GAIL, BC, and IADRL methods.

Collision avoidance is a key factor when deploying UGV-UAV coalitions for many applications, and, therefore, the number of collisions for all agents is a critical gauge for measuring the quality of our work. According to Fig. 4.10, GAIL and BC perform poorly when avoiding collisions. To better present results, we limited the range of the y-axis to [0,1500]. In the early training stages of GAIL and BC, there are many poor performance results, and some even exceed 4000 times that of IADRL. After training convergence, the number of collisions of IADRL for all agents in each episode is reduced to very low levels compared to that of GAIL and BC.

Additionally, we plotted the total number of collisions, the number of UGV collisions, and the number of UAV collisions in Fig. 4.11. As shown, UGVs experience the most collisions, and the more vulnerable UAVs work safely in a majority of cases. This is consistent with our initial design that the penalty for UGV collision is only half of that of a UAV's, as established in Table 4.1. Note that in real deployment scenarios, UGVs are more robust to collisions than UAVs, as most UGVs are equipped with bumpers and bumper sensors that help them protect against and avoid collision. Furthermore, UGVs utilize collisions to detect and navigate around the surrounding environment (e.g., iRobot Roomba Vacuums).

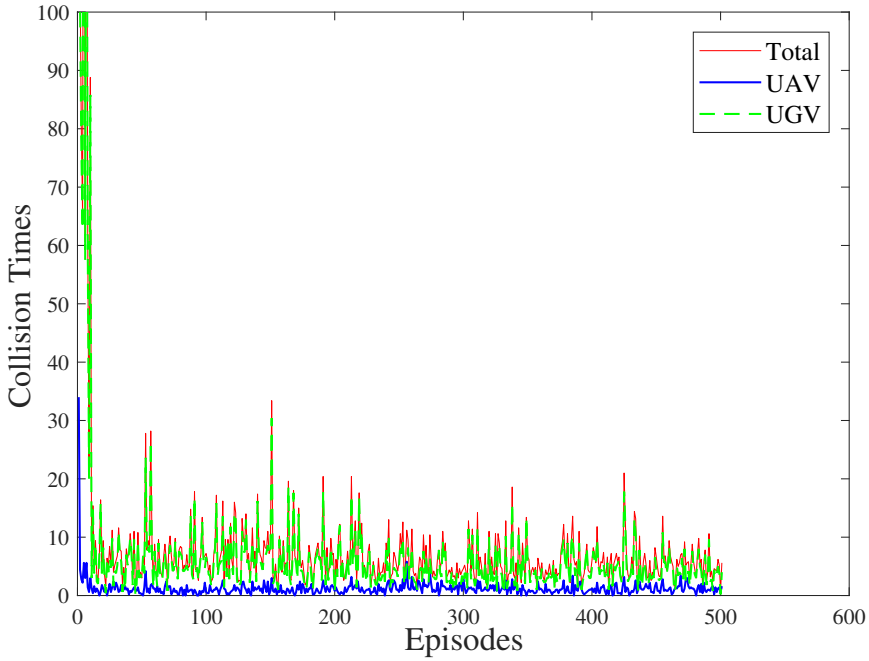


Figure 4.11: A composition of the number of collisions by UAVs and UGVs using the IADRL model, where the UGV collisions are dominant and UAV collisions are minimal.

After a further analysis, we find that the collisions are mainly caused by the sparse observation of UGV and UAV, as agents in IADRL are not able to detect obstacles and other agents. Although this result is already acceptable for many real-world robotic applications, we are confident that the addition of more sensory information to our system would allow for a much better performance on avoiding collisions.

To illustrate the path planning performance of each scheme, we designed a simple test with two targets, one located at (-5, -5, 5) and the other at (-5, -5, 1).[1] The planned paths for the three schemes obtained in five trials are plotted in Fig. 4.12. An optimized strategy for the coalition would have the lowest cost associated with reaching both targets. The UAV should ride on the UGV as close to the first target as possible, and then fly to reach the first target. The UGV should then continue on to reach the second target. Due to the physical size differences of UAVs and UGVs, we ploted their trajectories individually. For the trajectories generated by IADRL, we can see that the initial parts of the red lines (UAV) are parallel to the blue lines (UGV) because the UGV carries the UAV during this interval. The five lines for the UAV and UGV are for each of the five trials. Obviously, the path planning of our proposed algorithm

[1]Otherwise, the planned paths would be hard to plot and see.

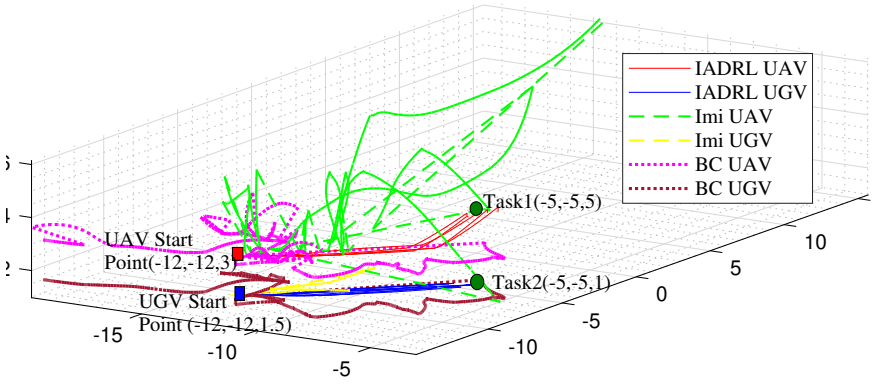


Figure 4.12: Planned paths for the UGV and UAV to reach two objects in five trials as computed by IADRL, GAIL, and BC schemes.

enables the UGV-UAV coalition to reach targets with an optimized route at a greatly reduced cost than that of GAIL or BC methods. Additionally, each IADRL planned route is almost identical in every trial, further proof of its stable performance.

Robustness in Different Environments

The proposed IADRL scheme is robust to changes in the environment and can be directly deployed in an environment different from where it was trained. As such, we train the model in an environment similar to Fig. 6.5 and deploy it in a more complex environment shown in Fig. 4.13. We add more obstacles, marked in red in Fig. 4.13, to simulate a warehouse with higher obstacle density. The same UGV-UAV coalitions with the well-trained IADRL model are deployed in this new environment to complete the same missions. We then compare the previous results with that in Fig. 6.5 using the three measurements introduced in the section 4.4.2. To guarantee the credibility of comparison results, all parameters, including reward settings, materials, and shape of agents, are kept identical in these two environments. In the rest of this section, we will refer to the results of experiments in Fig. 4.13 as the "Complex Environment," whereas the result in Fig. 6.5 will be referred to as the "Simple Environment."

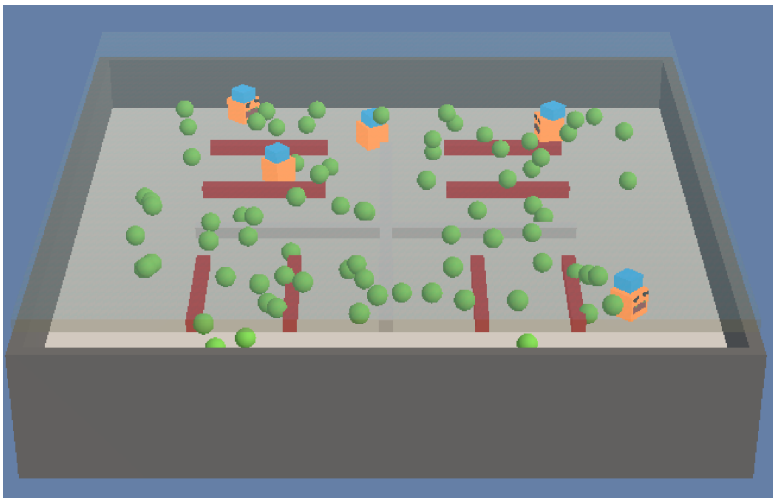


Figure 4.13: A complex simulation environment representing a high-density warehouse.

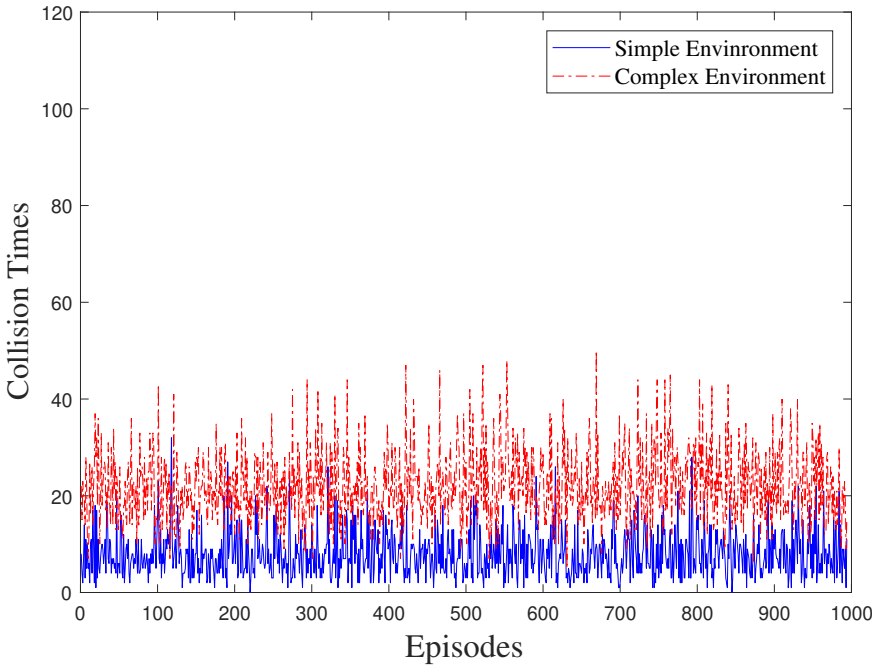


Figure 4.14: Number of collisions in the simple and complex environments.

Fig. 4.14 depicts the number of collisions during the testing process. Even challenged by higher environmental complexity, the number of collisions for each episode is only slightly increased due to the increased complexity of the environment. We also note that there is only a slight decrease in the accumulated reward values in the complex environment as compared to the simple environment, as illustrated in Fig. 4.15. Additionally, we investigate the amount of steps needed to complete the tasks in each episode. The results displayed in Fig. 4.16 show that it takes about 200 more steps for the coalitions to accomplish all tasks in the complex

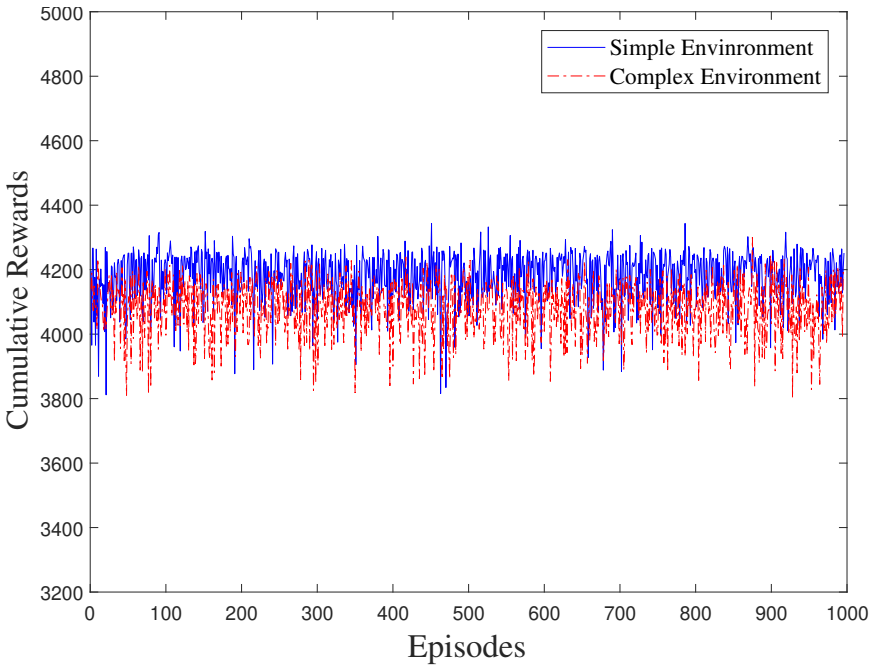


Figure 4.15: Accumulated rewards in the simple and complex environments.

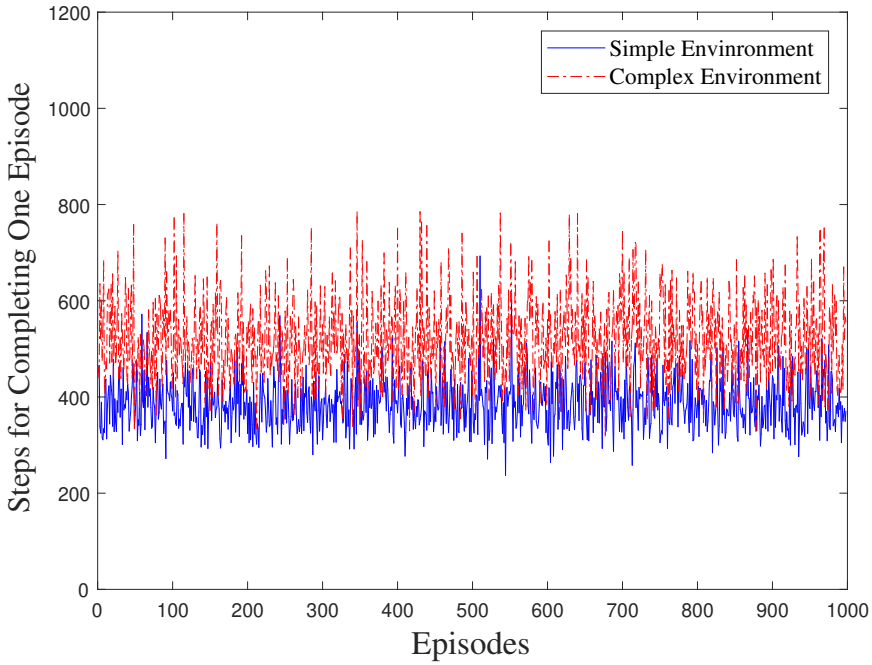


Figure 4.16: Number of steps needed to complete one episode in the simple and complex environments.

environment. These observations meet our expectations, as coalitions require more steps to bypass extra obstacles in the complex environment, and, thus, have a higher step cost and a decrease in accumulated rewards.

4.5 Conclusions

This paper presented IADRL, a novel method that enables UGVs and UAVs to form a coalition for the complementary accomplishment of tasks that neither the UAV or UGV could not complete independently. IADRL learns the complementary behavior features of the UGV-UAV coalition from a demonstration dataset that can be readily collected from some simple and imperfect settings alike. It also optimizes the strategy to achieve given goals with minimum overall costs required to complete task in dynamic environments. We also extended the IADRL model to facilitate the cooperation of multiple UGV-UAV coalitions deployed together for complex tasks. The experimental results proved that the proposed IADRL approach was effective for solving intricate tasks requiring heterogeneous agents to complement each other in dynamic environments.

Chapter 5

RIRL: A Recurrent Imitation and Reinforcement Learning Method for Long-Horizon Robotic Tasks

5.1 Introduction

With the considerable developments of deep learning methods in the past few years, reinforcement learning (RL), a method that has been proposed for over 20 years [143], is equipped with deep learning models and re-attracted the attention of academia and industry. It has been widely deployed to lead intelligent agents to interact with an environment to maximize the obtained cumulative rewards. Meanwhile, more and more robots are being deployed in various environments to accomplish tasks, such as inventory counting in retails and warehouses [2, 57]. In the past few years, researchers have shown a growing interest in applying deep reinforcement learning (DRL) methods to enable robotic systems to task in complex environments. For example, Kober et al. presented an automated meta-parameter acquirement for adjusting robots' movement by reinforcement learning [144]. The authors in [145] proposed a DRL-based motion planner that generates linear and angular velocities directly for navigation without an existing map. A model-based reinforcement learning framework for legged locomotion was introduced in [146], which allows the learned model to generalize to new tasks without any fine-tuning or using any extra collected data.

Currently, the RL methods applied to robotics could be roughly divided into two categories: traditional policy-based reinforcement learning and imitation learning methods. Some RL-based algorithms only utilize the received rewards to gain an approximated optimal policy, which makes them heavily rely on the effectiveness of the reward function [147, 148]. In [149], the author proposed two new dense reward functions to learn robust strategy in path planning

tasks. The reward sketching method was proposed in [150] that extracts human preferences to learn a reward function for a new task. However, manually designed reward functions that gratify the desirable agent actions are extremely complicated and usually not feasible, especially in the scenarios of dynamic and real environments with only sparse rewards. Moreover, even an expert cannot accurately quantify the reward of every behavior for various agents. A practical solution for the problems above is imitation learning (IL). Instead of manually designing a reward function, a set of well-prepared demonstrations provided by experts are followed and imitated by agents to learn the optimal policy for given tasks. An imitation learning method was introduced by Abbeel et al. in [151] to predict the agent's actions from a set of demonstrations and sequential states for another demonstration with different initial conditions. However, the agent must consume over 100,000 demonstrations to learn a simple strategy for reacting to different situations and initializations for the same task. Ho and Ermon proposed Generative Adversarial Imitation Learning (GAIL) [119], which combined Generative Adversarial Network(GAN) [138] with imitation learning. First, it trains the discriminator with the current policy's sampled data and expert data. Then, the discriminator plays the role of reward function to lead the agent to learn the optimal policy.

Long-horizon task planning is a challenging and open problem in robotics. Its complexity grows exponentially with increased numbers of acting steps and sub-tasks [152]. In many of these applications, the robotic agents need to execute a very large number of steps to reach the goal in unseen environments, considering an autonomous robot patrolling, searching, and retrieving objects in a giant unexplored building. In [153], the authors simplified the long-horizon policy learning problem to a hierarchical and goal-conditioned policy, where the low-level policy only requires a fixed, low number of steps to accomplish. Their simulation scenario was based on a simple kitchen environment, where tasks can be executed with a short sequence of discrete actions. Pitis et al. designed a strategy in that the agent pursues the maximization of the entropy of the historically achieved goal distribution instead of inaccessible goals when facing sparse extrinsic learning signals [154]. Similarly, in order to solve the sparse extrinsic reward problem of long-horizon tasks, the authors in [155] incorporated demonstrations into the RL method that is built on top of Deep Deterministic Policy Gradients (DDPG). They

focused on teaching agents to stack blocks with a robot arm by continuous multi-step control and generalize to varying goal states. However, the multi-step behavior needs far fewer steps for achieving the goal than in our target scenario, which requires thousands of operational steps for a robot to accomplish the task.

In addition, the training samples impose one more challenge for deploying RL methods in robotics. An RL-based method requires a great number of training samples. It may take millions of steps of experience to learn a strategy for a simple task. It is a great challenge and even infeasible to collect such a large amount of operational data in many robotic applications. Although with IL methods, robots are able to learn a robust strategy with a small number of expert demonstrations in short-term tasks, they still suffer from compounding errors and the massive growth of expert demonstration demands when facing longer and more complicated tasks. Some Pioneering works have exploited methods by combining the RL with demonstrations to overcome this challenge. For instance, demonstrations are used to initialize policy and accelerate training for reinforcement learning [156]. In addition to these challenges mentioned above, we find an additional difficulty of key interest: the long-term dependency problem, which some robotic tasks not only depend on the current observation but also rely on previous observations.

In this paper, we present a novel method called Recurrent Imitation and Reinforcement Learning (RIRL) to remedy the above limitations, which enables agents to leverage historical observations as well as environment feedback for more accurate action prediction. Our method exploits the robustness of Long Short-Term Memory(LSTM) in representing sequential information and solving the long-range dependency problem in long-horizon tasks.

The rest of this paper is organized as follows. In Section 5.2, we present the proposed approach. Our experimental study is discussed in Section 6.5. Section 6.6 concludes this paper.

5.2 Proposed Approach

5.2.1 Problem Statement and Challenges

We are interested in large-scale, long-horizon robotic tasks that require an agent to take thousands of steps to achieve the goal in unseen and dynamic environments. The problem can be formulated as an Partially Observable Markov Decision Process (POMDP) denoted by $(\mathcal{S}, \mathcal{A}, R, T, \mathcal{O}, \Omega)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, respectively; $T(s_{t+1}|s_t, a_t)$ is the state-transition probability for the agent to state s_{t+1} given it was at state s_t and takes action a_t; $R(s_t, a_t)$ is the reward function to provide an instant reward $r_t = R(s_t, a_t)$; Ω is a finite set of observations that the agent can experience of its world; $\mathcal{O}(o_t|s_t)$ is the probability for the agent to receive an observation o_t while it is at state s_t.

In a practical robotic application, the state s_t of the environment is hidden from the agent. Instead, it can only partially perceive the environment to attain an observation o_t by built-in sensors (e.g., cameras, sonar, LIDAR, etc.). Furthermore, to perform large-scale long-horizon tasks, the action a_t at step t not only depends on the current observation o_t, but also relies on the historical observations $o_0, o_1, ..., o_{t-1}$. We illustrate this long-range dependency problem in Fig. 5.1, w here a robot is deployed for a patrolling task. It needs to patrol the area and cycle around the two objects (A and B) to check the target features, e.g., to inspect the corrosion of power poles or towers. As shown in Fig. 5.1(a), the robot at state s_t gains observation o_t to represent the surrounding environment (such as its distance to the objects). However, o_t lacks the information for a fully understanding of the environment, such as which area it has already scanned. Based on the information from o_t, it is difficult to correctly choose the subsequent actions that could lead to a better navigational path. Fig. 5.1(b) and Fig. 5.1(c) illustrate that if we know the historical information, the next actions could be easily made to enable the robot with a better strategy to complete the task. Therefore, in the long-horizon task scenario, historical observations provide valuable information for predicting the action a_t. However, most of the existing RL based methods predict action $a \sim \pi(a_t|o_t)$ only depending on the current

observation, making them perform badly in the above described scenario. Thus, we formulate the objective policy as $\pi(a_t|\mathbf{O_t})$ to tackle the historical and instant observations, where $\mathbf{O_t} = (o_0, o_1, ..., o_t)$ represents a finite set including the historical and instant observations.

Our goal is to find such a policy π that maximizes the expected future discounted reward $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $0 \leq \gamma < 1$ is a discount factor. We adopt a value-action function $Q_\pi(a_t, \mathbf{O_t}; \theta) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t]$ to represent the above reward. Therefore, the goal of RIRL is to learn a policy π and Q_π that empowers the agent to achieve stable maximum overall rewards while performs various long-horizon tasks in dynamic environments. The objective function is given by

$$\underset{\mathbf{a}\sim\pi}{\operatorname{argmax}}\, Q_\pi(a_t, \mathbf{O_t}; \theta), \tag{5.1}$$

where θ serves as the parameter set of the value function Q_π; $a_t \sim \mathcal{A} \in \mathbb{R}^N$ denotes an action in a continuous space for the agent at time t; N is the dimension of the control space (usually, $N = 2$ for a ground robotic agent). The complexity of gaining such an optimized policy π grows exponentially with the increased amount of training data from historical observations. It is challenging and even infeasible for a traditional DRL method to converge quickly in the training processes for such tasks.

5.2.2 RIRL Network Architecture

The proposed RIRL is a model-free method that can learn a strategy from demonstrations to tackle the long-horizon task and subsequently fine-tune this strategy through the interaction with the environment for handling dynamics. As shown in Fig. 6.3, it deploys an architecture based on an Imitation Learning (IL) augmented Deep Reinforcement Learning (DRL) network that was introduced in [157]. The IL module adopts a Generative Adversarial Imitation Learning (GAIL) [119] network enhanced with an LSTM layer to enable it with the recurrent capability to retrain historical data and effectively learn the strategy from the demonstrations. The recurrent enabled IL module is the key to overcome the complexity challenge discussed in Section 5.2.1. The demonstration data provides a seed policy to greatly reduce the searching

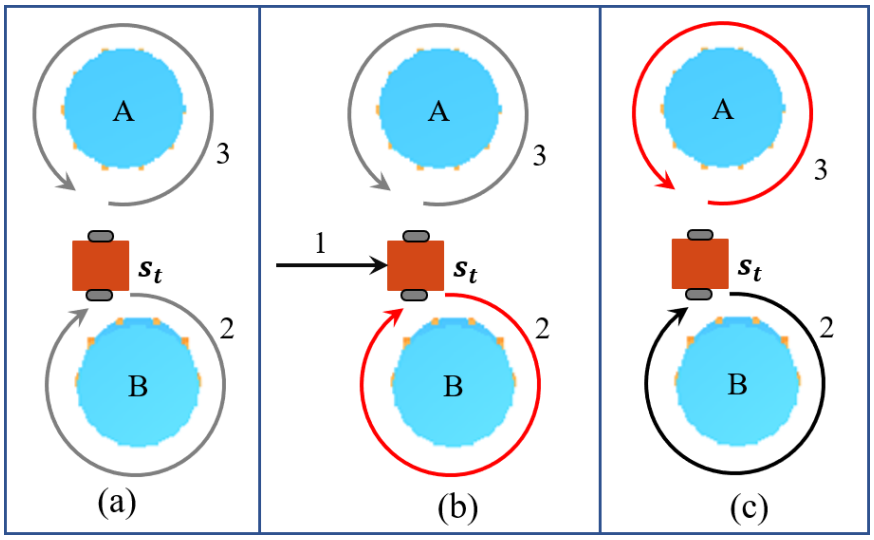


Figure 5.1: A brief scenario illustrates the long-range dependency problem of long-horizon robotic tasks: (a) the robot at state s_t, while two potential paths 2 and 3 are available; (b) if it has moved from path 3 to state s_t, the next actions lead to path 2 is a better choice; (c) if it has moved from path 2 to state s_t, the next actions lead to path 3 is a better choice.

space in the DRL training process to learn an optimized policy π. The DRL module is based on the Proximal Policy Optimization (PPO) [140] and is also enhanced by an LSTM layer that enables it to tackle both historical and instant data.

The IL and the DRL modules are marked in pink and green in Fig. 6.3, respectively. The IL module helps to learn a seed strategy from demonstration data and subsequently augments the training of the DRL module to build an optimized policy π to solve the long-dependency problem in long-horizon tasks.

The LSTM Network

As mentioned before, the key to tackling the long-dependency problem is to incorporating memory at the agent to exploit historical data for predicting actions. To this end, we adopt an LSTM network [158] layer to both the IL and DRL module in our proposed RIRL. LSTM is a popular Recurrent Neural Network (RNN) for effectively solving the long-term dependency problem. The architecture of its basic cell unit is shown in Fig. 5.3. There are three control gates: forget gate, input gate, and output gate. The forget gate determines which information from the last cell state could still continue to pass through the current cell. The input gate controls the new data to flow into the memory and update the cell state. The output gate selects

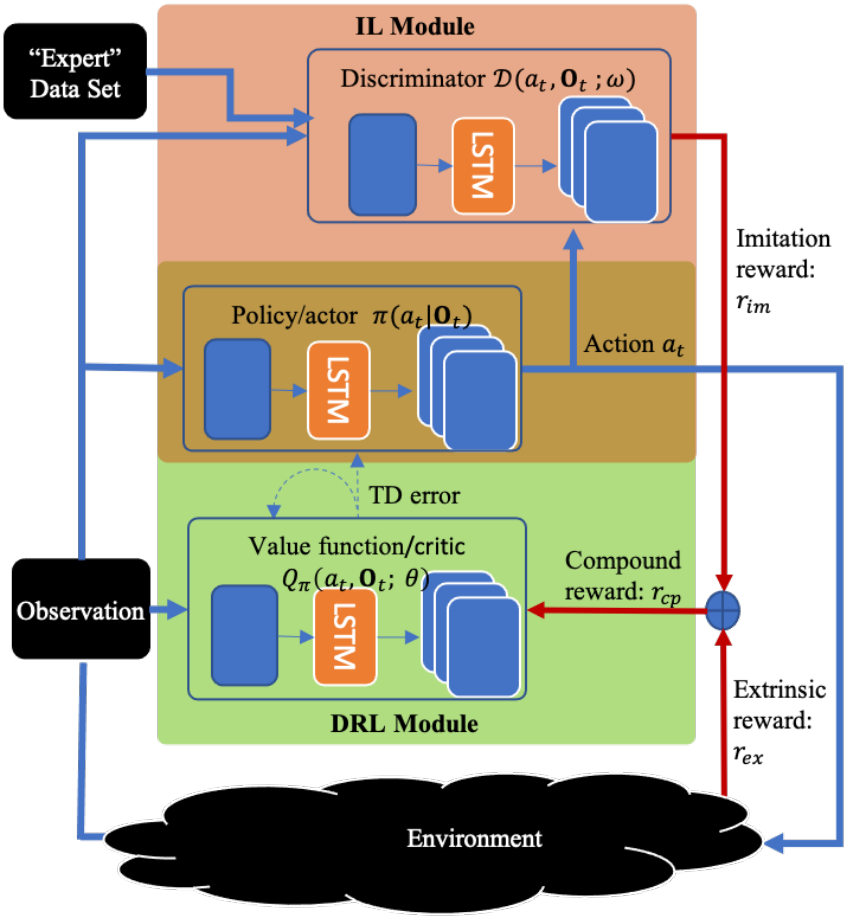


Figure 5.2: The architecture of the proposed RIRL method.

which part of the cell state can be exported as output. This structure helps to avoid the gradient exploding or vanishing problems of traditional RNN models. It exploits the temporal information of the observations by using the recursive hidden LSTM units. Important information over a long time horizon is stored by non-linear gate units of the built-in memory cell in each LSTM hidden unit. The last cell in the LSTM network outputs a vector h_t, which contains extracted features that not only include the information from the input data but also contain the important information from long-term historical observations. The memory built-in features h_t will be treated as the input of the subsequent network units to allow the proposed RIRL to obtain an optimized policy for tackling the long-dependency problem in long-horizon robotic tasks.

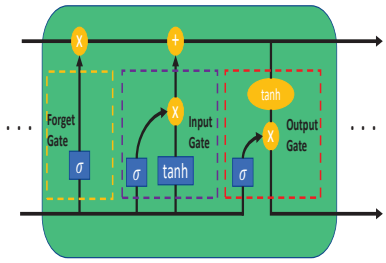


Figure 5.3: The LSTM structure.

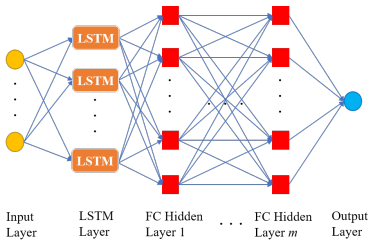


Figure 5.4: Architecture of Discriminator $\mathcal{D}$.

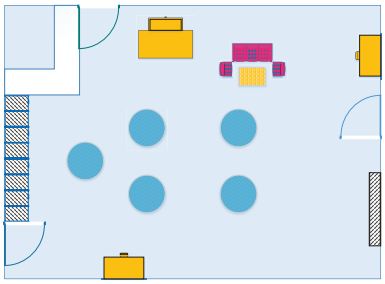


Figure 5.5: Layout of the simulated apparel store.

The Recurrent IL Module

The backbone of the proposed IL module is based on the GAIL framework [119] that adopts a network architecture developed from Generative Adversarial Network (GAN) [138]. It is composed of two basic parts, a Discriminator $\mathcal{D}$ and a Generator $\mathcal{G}$, which work in an adversarial and cooperative manner of learning a strategy from a set of demonstration data. Discriminator $\mathcal{D}$ is responsible for distinguishing the data produced by $\mathcal{G}$ from demonstration data, while both $\mathcal{D}$ and $\mathcal{G}$ are simultaneously trained in an adversarial and competitive way. During the training process, Discriminator $\mathcal{D}$ will be wiser to tell the generated data while Generator $\mathcal{G}$ gains more expertise in counterfeiting data. Eventually, the IL module converges when the generated "fake" data from $\mathcal{G}$ could pretend as demonstration data and pass the detection of $\mathcal{D}$. In our proposed network architecture, Generator $\mathcal{G}$ is shared by the IL and the DRL module. It also serves as the policy π of the DRL module; the terms Generator $\mathcal{G}$ and policy π are interchangeably in this paper. To enable the IL module with recurrent capability, we deploy an LSTM layer to enhance both Discriminator $\mathcal{D}$ and Generator $\mathcal{G}$.

The Recurrent Discriminator D

The recurrent enabled Discriminator $D(a_t, \mathbf{O_t}; \omega)$ of the IL module evaluates the data based on instant and historical observations to augment the process of predicting the next action. The discriminator $\mathcal{D} : \mathbf{O} \times \mathbf{A} \rightarrow (0, 1)$ is a function with weight ω, where $\mathbf{O}$ and $\mathbf{A}$ are the observation and action space, respectively. The Discriminator $\mathcal{D}$ model is shown in Fig. 5.4, which is a fully connected LSTM layer followed by m hidden layers. The LSTM and each hidden layer have the same number of units. The size of the input layer is decided by the

number of inputs. The LSTM layer maps the inputs to a feature vector h_t, which also carries the information from longer memories beyond the input vectors. Then the m fully connected hidden layer will convert the memory built-in features h_t to a score to measure the similarity of input data and "expert" data. By deploying an LSTM layer, any actions a_t will be evaluated on a large time scale to consider its performance for the long-horizon task. To train the Discriminator $\mathcal{D}$, we update and maximize the following value function, which is derived from the objective function of the GAIL network:

$$\mathcal{V}(\omega) = \mathbb{E}_{\pi}[\log(1 - \mathcal{D}(a_t, \mathbf{O_t}; \omega))] + \mathbb{E}_{\pi_e}[\log(\mathcal{D}(a_t, \mathbf{O_t}; \omega))] - \lambda H(\pi), \tag{5.2}$$

where π_e refers to the "expert" policy provided by a demonstration dataset, which provides seeds for subsequently training of the optimized policy π. Although referred as "expert," it is not a perfect policy the agent should take under any circumstance. In (6.3), π_e is collected by manipulating in several sample scenarios by manual settings with primitive strategies, which can only allow the agent to complete the long-horizon task in a non-optimized manner. $H(\pi)$ denotes the causal entropy of π and is defined as $H(\pi) \doteq \mathbb{E}_{\pi}[-\log \pi(a_t|\mathbf{O_t})]$. It serves as a policy regulator. It encourages the exploration behaviors and lets the learned strategy to be as random as possible while optimizing the objective, instead of quickly greedily converging to a local optimum. $\lambda \geqslant 0$ is the discount weight of H. The Discriminator $\mathcal{D}$ is updated to improve its ability to tell the similarity of a policy with expert data by increasing $\mathcal{V}(\omega)$. A well-trained $\mathcal{D}$ provides a higher score if the given data is more similar to the demonstrations. Therefore, by coordinating with Generator $\mathcal{G}$, the actions similar to the one provided in the demonstrations will get a higher selection rate. This way, the search space for the subsequent training can be narrowed to close to the seed behaviors and quickly converge to an optimized policy π.

The Recurrent DRL Module

In the proposed RIRL, the IL module could effectively learn a seed policy by imitating the behaviors from the demonstration dataset. This seed policy requires fine-tuning to allow the

agent to engage with a dynamic environment. Therefore, we deployed a DRL module to learn an optimized policy π by interacting with the dynamic and complex environment.

The DRL module is based on a PPO network and has an actor-critic architecture to exploit the environment with a continuous action space. It consists of an actor representing policy π and a critic shown as the value function Q_π. The policy π is responsible for generating the action, a_t, based on given observations $\mathbf{o}$. The value function Q_π processes the received rewards and evaluates the current action prescribed by policy π. As shown in Fig. 6.3, we implement these two components with two neural networks embedded with an LSTM layer for referencing historical observations recurrently. The architectures of these two neural networks are similar to that shown in Fig. 5.4, except for the number of layers. Let N_π and N_Q be the numbers of hidden layers for the actor and critic network, respectively. Each hidden layer has the same number of units. The size of the input layer is decided by the input vector dimension. The size of the generated action a_t determines the final output layer size of the actor.

The goal of training the DRL network is to maximize the value function Q_π for a given policy π, i.e.,

$$Q_\pi(a_t, \mathbf{O_t}; \theta) = \mathbb{E}[r_{cp}(o_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q_\pi(a_{t+1}, \mathbf{O_{t+1}})]], \tag{5.3}$$

where θ represents the parameter of the value function Q_π, γ is the discount factor for future reward, r_{cp} is a compound reward that consists of reward from the IL module and the extrinsic reward while interacting with the environment, i.e.,

$$r_{cp}(o_t, a_t) = \mu \cdot r_{ex}(o_t, a_t) + (1 - \mu) \cdot r_{im}(o_t, a_t), \tag{5.4}$$

where $\mu \in (0, 1)$ is a proportion parameter: a smaller μ will be implemented if the demonstration data is closer to the optimal policy. It helps the learning process to trade-off between demonstration data and engagement of the environment. In (6.5), $r_{im} = \log(\mathcal{D}(a_t, \mathbf{O_t}; \omega))$ is the reward that comes from the IL module, measuring how similar the action a_t is with the "expert policy" from demonstrations. The extrinsic rewards r_{ex} is provided as a sparse function

to represent the basic constraints and rules for the agent to interact with the environment and guide it to the desired goals. For example, we can give a penalty if the agent collides with an object in the environment and a positive reward if it reaches a goal position.

During the training process, policy π will be updated to choose the actions a_t to increase Q_π by gaining a larger r_{ex} and r_{im}. From (6.3), (5.3), and (6.5), increasing r_{im} by updating policy π, which also acts as the generator of the IL module, will decrease the value of $\mathcal{V}(\omega)$. Together with the training of the discriminator in the IL module, which tends to maximize $\mathcal{V}(\omega)$, the competitive processes end up with a policy π that roots with the strategies provided in the demonstration data. Furthermore, increasing extrinsic rewards r_{ex} will eventually lead the trained policy π to react to the environment with a better strategy. The built-in LSTM layers in the components of policy π, value function Q_π, and Discriminator $\mathcal{D}$ enhance the agent with memories for action predicting and evaluation. Therefore, the proposed RIRL provides an effective means to train an optimized policy π for long-horizon tasks, especially for the scenarios that require long-term memories.

5.3 Experimental Study

5.3.1 Experiment Setup

We deploy the proposed RIRL system in a simulated environment using the Unity3D platform to simulate an application of deploying a mobile robot for RFID-based inventory in an apparel store, which is the same scenario as in [2]. As shown in Fig. 5.5, we create a four-wall enclosed environment to simulate an apparel store of dimension $50 \times 50\text{m}^2$, which is crowded with racks (represented by blue circles) and obstacles (e.g., furniture). RFID tagged items are hosted on the racks. A simulated robotic agent is deployed to patrol the area to scan all the RFID tags. The positions of racks and the agent are randomly generated inside the room at the beginning of each episode. The agent simulates an RFID-enabled mobile robot controlled by action $a_t = (v_t, \phi_t)$, where $v_t \in [-v_{max}, v_{max}]$ represents the linear speed, while v_{max} is the maximum linear velocity, and $\phi_t \in [-\phi_{max}, \phi_{max}]$ denotes the rotation speed, while ϕ_{max} is the maximum rotation velocity.

The task of the agent is to scan all the RFID tags in the room with the shortest trajectory. The number of agent collisions is also considered as one of the criteria for measuring the quality of training results. The robotic agent is equipped with two sensors, a ray-cast sensor and a simulated RFID reader to collect observations. Unity3D provides the ray-cast sensor to simulate a widely deployed Lidar sensor. It detects the surrounding environment by casting rays and outputs a vector with the detected objects and the corresponding distances. We design a simulated RFID reader for the agent to mimic the characteristics of practical RFID applications. For example, it can only scan the tags within a detectable range, and the probability of reading a tag decreases as the distance between the reader and a tag increases.

The RIRL network is implemented with Tensorflow on a computer with an Intel 9900K CPU and two Nvidia 2080 GPUs. We conducted all our experiments with the same network: Discriminator $\mathcal{D}$ is implemented with one LSTM layer and two fully connected hidden layers, where each layer has 128 units. For the DRL network module, both the value function Q_π and policy π consist of an LSTM layer with 128 units and three hidden layers each with 512 units. In the experiments, the robotic agent will be deployed in the simulated apparel store, while its initial position is randomly generated at each episode.

5.3.2 Results and Analysis

We compare the proposed RIRL with three existing methods: the PPO network introduced in [140], the GAIL proposed by [119], and the IADRL scheme [157] that is an RL and IL combined method without the LSTM layers. We use the same basic reward settings and training parameters in the same environment set for these four approaches to guarantee a fair comparison. Fig. 5.6 plots the cumulative rewards acquired by the agent when it interacts with the environment. We only set several basic and sparse reward configurations: scanning a new RFID tag gains +10 rewards, collision results in a -1 punishment, and moving costs -0.001 for each step. Obviously, our approach, shown as the red solid line in Fig. 5.6, achieves the best reward in the training process, which is higher and more stable compared to the other three methods.

Fig. 5.7 presents the number of steps for finishing the tag scanning task in each episode. We set a maximum number of 20,000 steps for each training episode that aims to decrease the

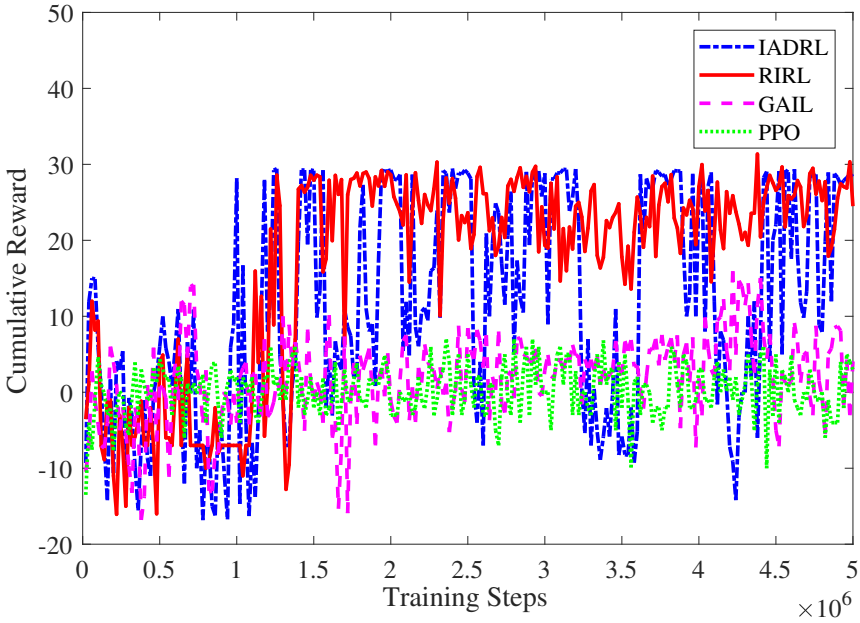


Figure 5.6: Accumulated training rewards values

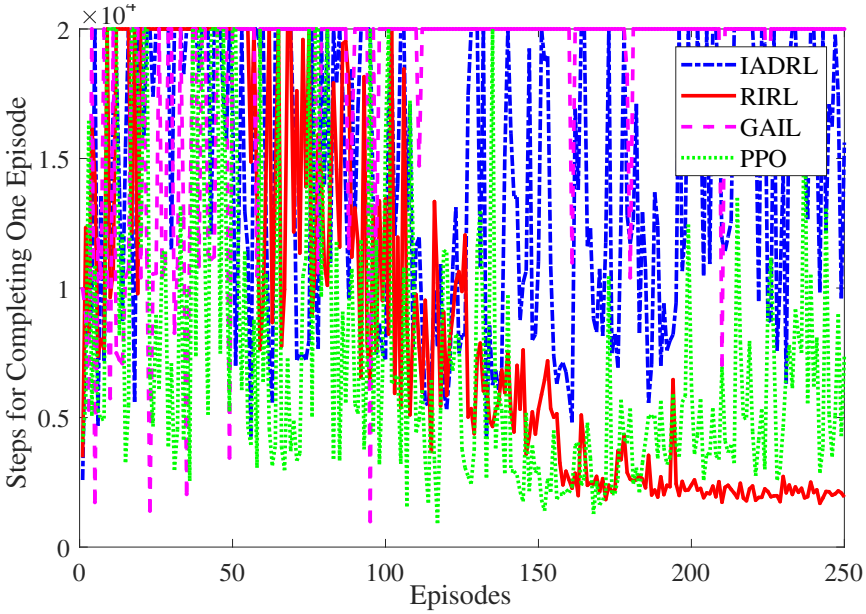


Figure 5.7: The number of steps for finishing tag scanning task

unnecessarily long training time. The red solid line depicts that our method result stabilizes after around 200 episodes. The agent implemented with RIRL could stably and consistently handle the given tasks within 2,000 steps. Apparently, as shown in Fig. 5.7, the other three methods cannot even converge with the same training process, indicating that the agent is unable to find a reliable strategy to accomplish the given task.

Fig. 6.9 presents the cumulative distribution function (CDF) of the percentage of unscanned tags in the testing stage. We test all the four trained models in 200 episodes within

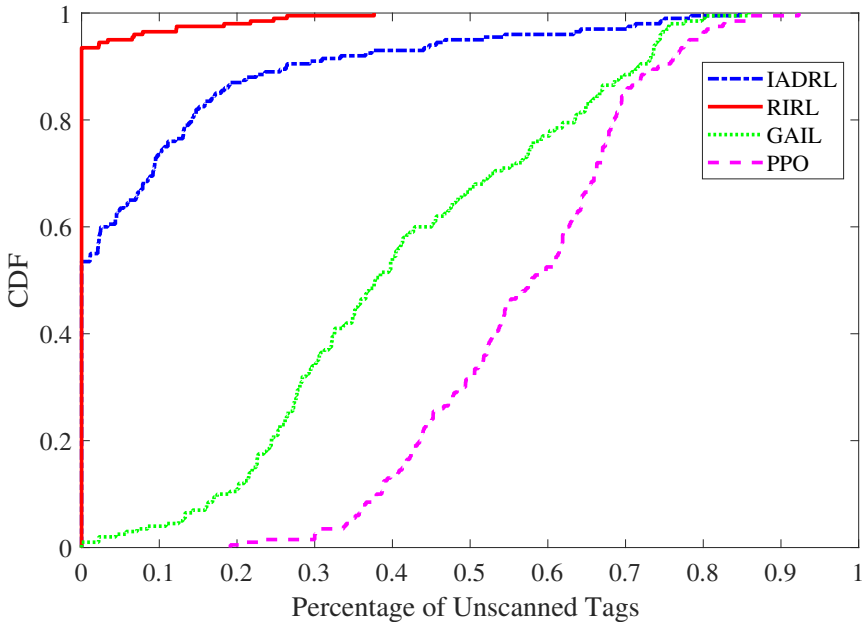


Figure 5.8: CDF of percentage of unscanned tags in total

the required 20,000 steps. Fig. 6.9 shows that in about 96% of the episodes, the proposed RIRL model scans all the tags, while the IADRL scans all the tags in about 55% of the episodes. Obviously, the GAIL and PPO model cannot even scan all the tags in an episode. Moreover, RIRL attains a maximum unscanned tag percentage of 28.2%, which is much lower than the almost 90% missing rate achieved by the other three methods. Apparently, the proposed RIRL exhibits considerably higher effectiveness and robustness for such long-horizon tasks in dynamic environments.

The experiments validate that our proposed RIRL outperforms the three benchmark approaches. It help the agent to accomplish the long-horizon tag scanning task with high efficiency and robustness. Moreover, it proves the feasibility of utilizing the LSTM network to enhance the agent performance by leveraging historical observations.

5.4 Conclusion

In this paper, we proposed RIRL, a deep recurrent imitation and reinforcement learning-based system, which enables agents to accomplish long-horizon tasks in dynamic and complicated environments. We also experimentally validated the feasibility of embedding an LSTM layer in the traditional IL and DRL methods. The outstanding result achieved by our method proved

the effectiveness of leveraging history observations for enhancing RIRL to solve the long-range dependency problem in various long-horizon robotic tasks.

Chapter 6

SRRL: Multi-state-space Reasoning Reinforcement Learning for Long-horizon RFID-based Robotic Searching and Planning Tasks

6.1 Introduction

Robotic task planning over long-time horizons, including navigation, path planning, tasks allocation, etc., has been a challenging, relevant, and hot topic in robotics since the last century. As the numbers of acting steps and subtasks are increased, so does the complexity. Many of these applications require robotic agents to complete a huge number of steps in unknown surroundings, such as an autonomous robot patrolling, searching, and recovering things in a massive uncharted structure. Robots are being used increasingly in various environments to perform activities, such as inventory counting in retail stores and warehouses [2, 57, 159]. In [153], the authors reduced the long-horizon policy learning problem to finding a hierarchical and goal-conditioned policy, in which the low-level policy takes only a fixed, small number of steps to complete. They used a kitchen as a simulation environment consisting of short sequences of discrete actions for completing tasks.

Meanwhile, Simultaneously Localization and Mapping (SLAM), first proposed by Durrant-Whyte in [160], is a method commonly used in numerous map-based path planning algorithms [161, 162]. SLAM allows the robot to start from an unknown position in an unknown environment, determine its own position and posture by repeatedly observing the characteristics of the environment during movement, and then draw an incremental surrounding environment map based on the position of the surrounding environment. The authors introduced a path planning algorithm paired with active SLAM in [162] that may continually increase the localization accuracy without disrupting the main task. The goal is to deal with the dynamic

changes in the environment, such as shifting obstacles and localizations that may arise while the robot is moving. However, once the environment has been altered significantly, map-based algorithms usually require to rebuild the map in the testing stage, which is unquestionably a time-consuming and difficult process.

Reinforcement learning (RL), a method that has been proposed for more than two decades, is now equipped with deep learning models and has re-attracted the attention of academia and industry [143, 163]. It has been frequently utilized to direct intelligent agents to interact with an environment so as to optimize their accumulated benefits. In recent years, academics have shown an increasing interest in adopting deep reinforcement learning (DRL) to enable robotic systems to function in complicated situations. In contrast to the map-based methods, the optimal policy trained by RL methods does not need a pre-built obstacle map or intensive environment features. For instance, an automated meta-parameter acquisition for altering robot movement via reinforcement learning was proposed in [144]. Kim and Pineau in [164] described a framework for socially adaptable path planning in dynamic environments that includes a feature extraction module, inverse reinforcement learning, and path planning module.

In terms of the application of DRL in long-horizon robotic tasks, when confronted with scarce extrinsic learning inputs, Pitis et al. [154] proposed a method in which the agent aims to maximize the entropy of the historically attained goal distribution rather than inaccessible objectives. Likewise, the authors of [155] included examples in the RL approach that is based on Deep Deterministic Policy Gradients in order to overcome the sparse extrinsic reward problem of long-horizon tasks (DDPG). The focus was placed on instructing agents to stack blocks with a robot arm using continuous multi-step control and generalization of goal states. Nevertheless, in comparison to our target scenario, this introduced multi-step behavior requires considerably less steps to achieve the goal. Many RL-based algorithms rely solely on the received external rewards to arrive at an approximated optimal policy, making them particularly reliant on the effectiveness of the reward function. Meanwhile, hand-crafted reward functions that fulfill the desired agent behaviors are exceedingly complex and typically infeasible, especially in dynamic, real-world contexts with sparse rewards. Furthermore, even an expert cannot precisely measure the payoff for each and every agent's behavior. For some tasks, the robots are

required to handle observations from multiple types of sensors and fuse the different observations to find optimized actions. The relations among the multiple observed spaces are usually intangible and implicit. It imposes additional challenges to DRL-based methods because the complexity to explore multiple spaces will be increased exponentially.

To address these issues, this paper proposes an approach of Multiple State Spaces Fusion and Reasoning Reinforcement Learning (SRRL), a novel method that allows the agent to abstract features and infer policy from multiple state spaces. Basically, we consider the robotic applications where a robot is used to scan the RFID tags in an unknown area (e.g., an apparel store or inventory area). Two state spaces are considered in SRRL: one for environment occupancy observations and the other for RFID sensing. An RFID reader, carried by the robot, transmits radio signals to interrogate RFID tags, and the surrounding radio intensity map determines the probability of tags being scanned. Generally speaking, the chance of tags being scanned diminishes steadily as the distance to the reader is increased. We record the approximated radio map while the agent moves and convert it into an image. Additionally, through the observations from a spinning Light Detection and Ranging (Lidar) sensor, we also built a gray-scale 2-D environment occupation map of the physical world. These two maps serve as the foundation of our multiple state spaces. We aim to let the agent learn abstract reasoning by continuously fed with multiple states during the process of solving long-horizon tasks in a dynamic and previously unknown environment.

The remainder of this paper is organized as follows. In Section 6.2, we introduce the related works. We then present the preliminaries and motivation in Section 6.3. The SRRL system design is described in Section 6.4. Our experimental study is presented in Section 6.5. Section 6.6 concludes this paper.

6.2 Related Work

Artificial General Intelligence (AGI) refers to the capacity of models or agents to behave like humans with cognitive abilities to comprehend and learn any intellectual job. With the rapid advances in deep learning, AGI has attracted increasing interest in the community. Briefly speaking, it is to solve tasks as human thinking, called the reasoning ability. This is a fairly

broad idea, but it is essential to people's daily life. For example, image recognition is a form of reasoning, although being one that is quite straightforward and more like prediction. Deep Learning has basically solved this type of prediction problems, and thus the next step is to handle more complex and more challenging reasoning problems. Here is a simple analogy to illustrate why reasoning is more complicated than prediction. Given all the necessary ingredients, including flour, sugar, eggs, yeast, utensils, cookware, and a set of instructions, then, baking a cake is just a matter of determining the correct proportions of each component by trial and error. This is identical to what conventional deep learning accomplishes, after several rounds of forward pass and backward weight updates, to identify the optimal parameter set to ensure high prediction accuracy. But imagine the case that one is merely provided with the raw materials and cooking utensils. In such a circumstance, one cannot bake bread by just placing flour in the oven, but also need to follow a series of correct procedures and use exact amounts of material, which is of an entirely different level of complexity.

6.2.1 Reasoning in Deep Learning

Deep learning has been highly effective in extracting useful representations from vast amounts of data. It creates possibilities to query and consciously reason about the extracted representations to develop understanding. Through a sequence of mathematical manipulations of the available information, reasoning machines may arrive at a conclusion about a new set of factors in response to a query. In recent years, an increasing body of research has focused on incorporating new types of inductive biases into deep neural networks in order to facilitate deliberative reasoning [165, 166, 167, 168], hence pushing deep learning systems towards the thinking mode. In [169], the authors illustrated a learning-to-reason framework, where reasoning is framed as a classification job in which it is necessary to assess if the knowledge base contains a conclusion. It leverages neural networks to execute a number of essential functions, such as abstraction, concept binding, attention [170], causal interplay estimation, and composition.

Moreover, some researchers have shown that the reasoning process is intricately tied to efforts on neural memories [171, 172], which is an intellectual capacity for memorizing, recovering, altering information, and simulating unobserved situations. Grave et al. [173], for

instance, developed a model consisting of a neural network that can read and write to an external memory matrix, akin to the random-access memory of a conventional computer. The model can utilize its memory to represent and manipulate complicated data structures like a conventional computer, while memory is a collection of slots connected to a neural network for storing intermediate outcomes or data. The authors in [172] utilized a unique memory to store controller weights, similar to the stored-program memory in contemporary computer architectures, where sub-programs are collected and stored, and to be leveraged to generate new programs on-the-fly based on a query. Despite the recent enormous advances in reasoning in deep learning, there are still many challenges, such as weak generalization. In addition, discovering and understanding the association between data pattern and query is crucial to its success. Therefore, reasoning tends to be particular to data patterns, resulting in inadequate systematic generalization ability.

6.2.2 Reasoning in Robotics

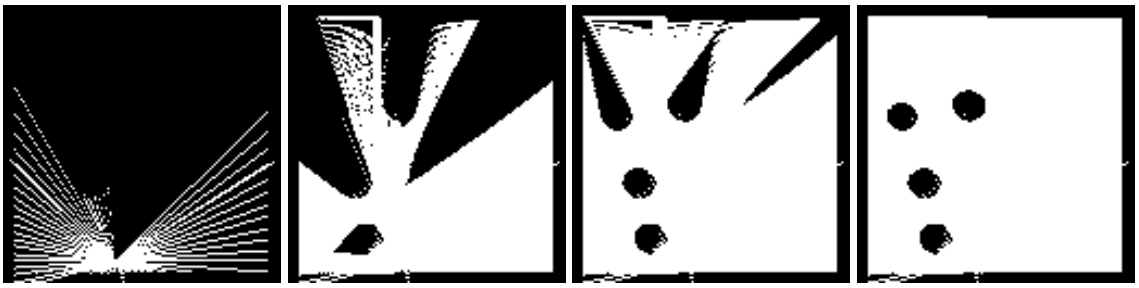
In addition to the above mentioned transition from deep learning to deep reasoning, the demands and applications of reasoning in the area of robotics are also becoming research hot spots, which is called cognitive robotics. This is the study of knowledge representation and reasoning posed by an autonomous robot (or, agent) in a dynamic and partially observable environment. For instance, combining robotic tasks with visual reasoning is quite prevalent. In [174], the authors showed that the Convolutional Neural Networks (CNNs) are unable to identify complicated attribute patterns within or across rows/columns of Raven's Progressive Matrices (RPM), since they rely solely on relation extraction at the matrix level. Therefore, inspired by human induction strategies, the introduced method extracts several coarse rules embedding at different levels, including cell-wise, individual-wise, and ecological embedding, from the two rows/columns provided. It deploys different levels of reasoning on different network components. The authors in [175] proposed a graph framework called Continuous Scene Representations (CSR), consisting of sets of nodes and edges, for capturing feature relationships among items. Nodes and edges in the form of continuous vectors of a graph are all

represented by a learnt feature. It firstly uses a Faster Region Based Convolutional Neural Networks (R-CNN) model to detect and segment nodes. A match function will provide a score for all features between the global and local scene graphs for updating purposes, including object nodes and related features. Edges are averaged into the representation if a new relationship is observed; otherwise, they are added to the representation.

With the rise in popularity of DRL in recent years, the combination of reasoning and DRL has also produced several innovative works [176, 177, 178]. An end-to-end DRL framework that combines the feature abstraction ability and Q-learning was presented in [177] to identify features in natural scenes that represent a particular event or interaction and then discover the relationship among the features in the form of generalized rules. This was motivated by the fact that humans can closely approximate rules, which are set by social norms or the goal of interaction, simply by observing several instances of the interaction. The proposed method, termed as Staged Social Behavior Learning (SBBL) in [178], is focused on using DRL in social human-robot interaction. This study employs a technique for learning a mapping between input pictures and reduced low-dimensional state representations. In this study, the authors focused on the first two steps required for a robot to acquire behaviors for approaching small groups. Additionally, several recent studies concentrate on merging knowledge graph reasoning with DRL in order to infer the required entity from the entities and relations currently present in the knowledge graph. The authors in [179] built a relational module that may be considered as a universal plug-in for a reasoning framework, with a self-attention mechanism that repeatedly infers the relations between things to steer a model-free strategy. The proposed model was shown to increase the efficiency and comprehensibility of conventional approaches through structure perception and relational reasoning. However, these prior studies rely solely on observations as the state space. In contrast, our proposed method decodes characteristic features from reasoning across a large number of independent state spaces.

6.3 Preliminaries

In this paper, we focus on large-scale, long-horizontal robot tasking in unknown and dynamic environments. We enable the robot with the reasoning ability by learning the correlation across



(a) Step 0 (b) Step 500 (c) Step 2000 (d) Step 4000

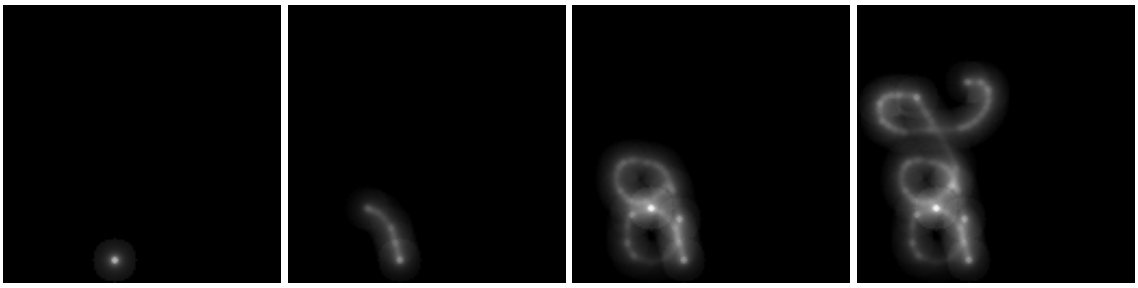
Figure 6.1: The environment occupation map after different steps.

multiple state spaces. In this paper, we consider the application scenario where an agent carries an RFID reader to swiftly and safely scan the RFID tags on all the racks in a retail store or warehouse. There are two particular goals for this agent. The first is to identify the racks as target points from the environment occupation map created by Lidar sensors quickly and efficiently. The second is to employ RFID radio signals to cover the target points as rapidly as possible. The key point for solving such long-horizontal robotic tasks is the agent's ability to fuse and reason with multiple state spaces.

6.3.1 Multi-state-spaces Feature Extraction and Reasoning

Environment Occupation State Space $\mathbf{S}_L$

In our project, we use occupation maps to represent the objects in a physical environment. An occupation map can be incrementally created from the observations of a robot. As depicted in Fig. 6.1, a Lidar sensor is deployed to continuously scan the surrounding space, aligned with the agent's motion, in order to construct a map including information on the environmental layout. Each subplot in Fig. 6.1 describes the environment that the agent partially observed at the moment, where the white area represents the observed free zone and the black area represents the occupied or unknown portion of the space. We build this occupation map at each step and use a set of maps to represent the environment occupation state space, denoted as $\mathbf{S}_L$.



(a) Step 0 (b) Step 500 (c) Step 2000 (d) Step 4000

Figure 6.2: The RFID sensing radio map after different steps.

RFID Sensing State Space $\mathbf{S}_R$

The proposed system also requires tackling a secondary state space that is created by the wireless signals from the robot's built-in RFID reader. To more precisely define the state space, we resort to the RFID model of fixed radio frequency (RF) transmit power, $P(o_t|x, d_t)$, proposed in [57]. The model calculates the probability that an RFID reader's antenna is located at d_t, and measures an observation o_t of a tag that is located at x. Fig. 6.2 illustrates the expansion of the RFID radio map while the agent explores the unknown space. As shown in Fig. 6.2a, RFID sensing has a limited range. The smaller the range, the greater the possibility of reading an RFID tag close by (as indicated by the brighter point in the figure). Obviously, if the agent has remained stationary for an extended period of time, the coverage of RF sensing will not vary, and the likelihood of reading tags within the range will be close to one hundred percent. Consequently, our objective is to enable the agent to gradually learn the environment, so that the areas around the target can be more efficiently covered by RF sensing. This is obviously a long-horizon robotic task. Similar to the environment occupancy state space $\mathbf{S}_L$, the RFID sensing state space $\mathbf{S}_R$ is defined by a series of RFID radio maps with the same size.

Feature Decoding By Deep Convolutional Neural Networks (DCNN)

The DCNN model consists of multiple convolutional and sub-sampling layers and one or more fully linked layers. It makes use of local correlations by sharing the same weights among neurons in adjacent layers, hence saving training time. DCNN is also capable of extracting local dependency and scale-invariant characteristics from input data. Importantly, it can derive

richer abstract representations of the input image data from the lower layers to the upper layers of the hierarchical design.

Following is a description of DCNN's primary components. Using linear convolutional filters followed by nonlinear activation functions, the convolutional layer can extract local feature maps from the previous layer's feature maps. Let μ_n^i be the nth feature decoding in layer i, defined as

$$\mu_n^i = \sigma\left(\sum_{m \in L_{i-1}} w_{nm}^i * \mu_m^{i-1} + b_n^i\right), \tag{6.1}$$

where $\sigma(t) = \frac{1}{1+\exp(-t)}$ represents the sigmoid function; S_{i-1} is the set of feature maps in layer $(i-1)$ connected to the current feature map; w_{nm}^i denotes the convolutional kernel to generate the nth feature decoding in layer i; μ_m^{i-1} represents the feature decoding of the last layer; and b_n^i is the bias of the nth feature decoding in layer i.

Since the input contains two maps per step, following the convolutional layer is typically a pooling layer, which decreases the size of the activation map and reduces the computational cost. From a small region termed a pooling window, the pooling layer picks the maximum of a representative feature. Furthermore, as previously stated, all state spaces have a fixed-length observation, leading to the adoption of two LSTM layers after the DCNN output. The goal is to improve the agent's ability to utilize historical data derived from extracted features. In summary, the reasoning ability derived from multiple state spaces could be written as $\mathcal{R}_t(\mu_{s_L}, \mu_{s_R}; \phi)$, where ϕ represents the parameter set of the network of multiple state spaces feature decoding and reasoning; μ_{s_L} and μ_{s_R} denote the encoded features of the state spaces S_L and S_R, respectively.

6.3.2 Reinforcement Learning

Motivated by the Partially Observable Markov Decision Process (POMDP), this problem could be described by $(\varepsilon, \mathbf{S}, A, T, R)$, where ε represents the environment the agent interacts with; $\mathbf{S} = (S_L, S_R)$ is the joint partially observed state space that consists of two state spaces with different dimensions and meanings: the real-time obstacle map S_M created by Lidar sensors,

and the RFID sensing signal state space S_R; $A = (a_1, a_2, ..., a_t)$ denotes the set of all available actions; $T(s_{t+1}|s_t, a_t)$ is the state-transition probability from state s_t to state s_{t+1} if the agent is in state s_t and performs action a_t. The reward function $R(s_t, a_t)$ provides a present reward given by $r_t = R(s_t, a_t)$.

The purpose of our method is to discover a policy $\pi(a_t|\mathcal{R}_t)$ that maximizes the predicted future discounted reward $E_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $0 \leq \gamma < 1$ is a discount factor. This reward is formulated as the value-action function: $Q_\pi(\mathcal{R}_t, a_t; \theta, \phi)$. Accordingly, the objective of our method is to identify a policy π and Q_π that enables the agent to attain reliable maximum overall rewards while carrying out a variety of long-horizon robotic searching and planning tasks in dynamic environments, which is defined as

$$\underset{\mathbf{a}\sim\pi}{\operatorname{argmax}} \; Q_\pi(\mathcal{R}_t, a_t; \theta, \phi), \tag{6.2}$$

where θ and ϕ serve as the parameter sets of the value function Q_π and the reasoning module, respectively. Note that $\mathcal{R}_t$ is the output of the reasoning module described above. $\mathcal{R}_t$ extracts the latent relations from multiple state spaces; it also provides an abstracted presentation of observations, which are sampled from multiple state spaces. Thus, $\mathcal{R}_t$ will be deployed to reduce the searching space of the subsequent training processing and retain the latent information in and among all state spaces. Developing such an optimal policy becomes exponentially more difficult as the quantity of training data derived from past observations increases.[1] A standard DRL technique may or may not be capable of achieving rapid convergence in the training stages. Still, it cannot even accomplish such long-term tasks without the reasoning ability.

[1]The complexity of a DRL-based method is determined by the size of its state space and action space. In our work, the number of fetches will increase exponentially with the precision, and the number of actions will grow exponentially with the increase in degrees of freedom. Moreover, in long horizon tasks, historical information is needed for better motion planning, which also increases the size of the state space. Finally, the computational complexity in dynamic environments is hard to quantify but does exist for DRL-based algorithms. The changing environment makes it impossible for the agent to use the successful experience of the previous round directly but requires it to be able to reason and generalize.

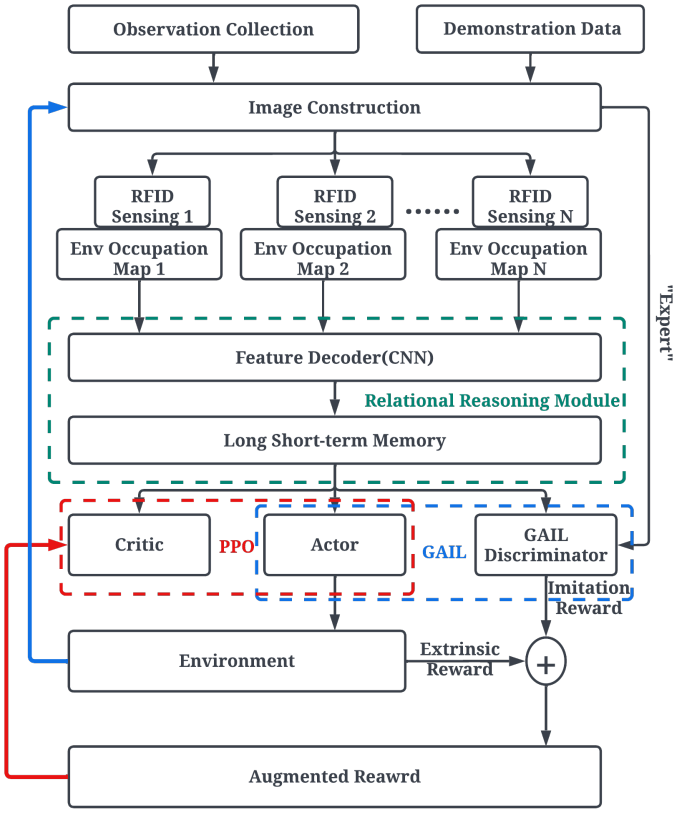


Figure 6.3: The architecture of the proposed method.

6.4 Overview of the Proposed System

Our proposed model can infer the optimal policy for long-horizon robotic searching and planning tasks from the latent interconnections of many distinct dimensional state spaces. Subsequently, this policy is optimized through interactions with the environment and data from manually provided demonstration data. As depicted in Fig. 6.3, the model consists of three parts: (i) the Relational Reasoning module, (ii) the Imitation Learning (IL) module, and (iii) the DRL module [157]. Additionally, all the networks are reinforced with an LSTM layer to retain historical data in a recurrent manner. In Fig. 6.3, the IL module and DRL module are marked by the red and blue dotted boxes, respectively.

The basic IL module utilizes Generative Adversarial Imitation Learning (GAIL) [119], so that agents can successfully adapt their strategies based on demonstrations. The recurrently enabled IL module is utilized to address the barrier of complexity by giving a seed policy

through the demonstration data to considerably reduce the searching space for the DRL to learn an optimal policy. The DRL module is based on Proximal Policy Optimization (PPO) [140] and is augmented with an LSTM layer that enables it to deal with both historical and present data.

6.4.1 The Multi-state-space Fusion and Reasoning Module

The reasoning module, discussed in Section 6.3.1, including the construction of multiple state spaces and feature decoding and reasoning, is the key to explore the environment for scanning RFID tags. In our case, there are two state spaces: one reflects the observation of the actual real world, and the other represents the property of the RF signal space, which varies in response to the agent's motion. Their underlying link is difficult to quantify or establish directly, yet it is crucial for efficiently completing long-horizon robotic searching and planning tasks. The output of the reasoning module is not only the extracted features containing historical information, but also the reasoning ability generalized from these implicit connections, which can help the agent choose actions more quickly and rationally, so as to be as close as possible to the capacity of humans to record expert demonstrations under fully observed conditions.

6.4.2 The Recurrent IL Module

The proposed IL module incorporates the GAIL framework [119], which employs a network design derived from the Generative Adversarial Network (GAN) [138]. It comes with two fundamental components, a Discriminator $\mathbf{D}$ and a Generator $\mathbf{G}$, which learns a strategy from a set of demonstration data in an adversarial and cooperative way, respectively. The Discriminator $\mathbf{D}$ is capable of identifying $\mathbf{G}$'s data from demonstration data, while both $\mathbf{D}$ and $\mathbf{G}$ are concurrently taught in a competitive and adversarial manner. During the training stage, Discriminator $\mathbf{D}$ will become more adept at identifying created data, while Generator $\mathbf{G}$ will be much more proficient at forging data. The IL module eventually converges when the generated "fake" data from $\mathbf{G}$ can masquerade as demonstration data and survive $\mathbf{D}$'s verification. Generator $\mathbf{G}$ is shared by the IL and DRL modules in our network design. It also functions as the DRL module's policy. Note that the concept Generator $\mathbf{G}$ and policy π are thought to

be interchangeable in this work. In other words, the policy π serves two functions: it not only creates actions based on the distribution of "expert" data, but also is responsible to react to the environment with an improved approach. The policy π will be thoroughly discussed in Section 6.4.3 when we present the DRL network. In this section, we just look at the imitation network's discriminator $\mathbf{D}(a_t, \mathcal{R}_t; \omega, \phi)$.

As previously stated, we employ an LSTM layer to augment both Discriminator $\mathbf{D}$ and Generator $\mathbf{G}$ in order to supply recurrent capabilities to the IL module. The recurrence-enabled Discriminator $\mathbf{D}(a_t, \mathcal{R}_t; \omega, \phi)$ assesses data based on real-time and historical data to improve the process of forecasting the next action. $\mathbf{D} : \mathcal{R} \times \mathbf{A} \mapsto (0, 1)$ is a function with weights, where $\mathcal{R}$ and $\mathbf{A}$ represent the combination of the extracted correlations of multiple state spaces and action spaces, accordingly. Fig. 6.4 illustrates the structure of the Discriminator, which is composed of a fully connected LSTM layer followed by m hidden layers with same amount of units. During the training stage, the discriminator $\mathbf{D}$ can be enhanced by optimizing the following value function:

$$\begin{aligned} \mathrm{V}(\omega) = \mathbb{E}_{\pi}[\log(1 - \mathbf{D}(a_t, \mathcal{R}_t; \omega, \phi))] + \\ \mathbb{E}_{\pi_E}[\log(\mathbf{D}(a_t, \mathcal{R}_t; \omega, \phi))] - \lambda H(\pi), \end{aligned} \tag{6.3}$$

where τ represents the "expert" policy given by a demonstration dataset, which serves as a seed for later training of the optimized policy. It is not a perfect policy, but based on a few sample instances in controlled circumstances navigated by manual control and is thus regarded as "expert." In (6.3), π_E is gathered by handling multiple sample scenarios with manual settings and basic methods, which can only allow the agent to execute long-horizon robotic searching and planning tasks in a sub-optimal approach. $H(\pi)$ is defined as $H(\pi) \equiv \mathbb{E}_{\pi}[-\log \pi(a_t|\mathcal{R}_t)]$ and indicates the causal entropy of policy and work as policy regulator. It also fosters exploratory behavior and allows the learned approach to remain as random as feasible while achieving the goal, rather than fast mindlessly converging to a local optimal. $\lambda \geqslant 0$ denotes the discount weight of H. $\mathrm{V}(\omega)$ is increased to improve the Discriminator's capacity to compare a policy's resemblance to the "expert" data. When it generates a lower value for a particular action a_t, it

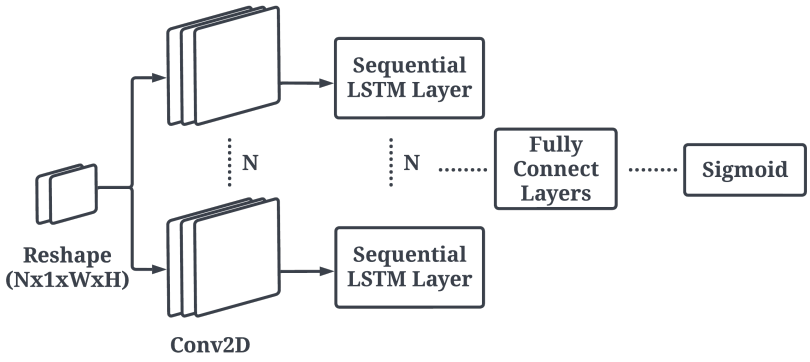


Figure 6.4: The architecture of the Discriminator.

suggests that the probability of action a_t is greater based on "expert" data, hence exhibiting a greater capacity for reasoning across various state spaces.

6.4.3 The Recurrent DRL Module

In our framework, the IL module could successfully acquire a seed policy by emulating the behaviors from demonstration datasets. To allow the agent to interact with a dynamic environment, this seed policy must be fine-tuned. To this end, we set up a DRL module to interact with the dynamic and complicated environment to develop an optimal policy π. This module is based on a PPO network [140] and is composed of two different components: actor π and critic as value function Q_π. The actor π is responsible for generating action a_t based on the relational observations $\mathcal{R}_t$. The above-mentioned relational reasoning module learns this by extracting correlations of a finite set of multiple state spaces data. The value function Q_π assesses the current action generated from the actor by processing the received rewards and evaluates.

Ultimately, the goal of training the DRL network is to maximize the value function Q_π defined in (6.2) for a given policy π, as

$$Q_\pi(\mathcal{R}_t, \mathbf{a}_t; \theta, \phi) = \mathbb{E}[r_{en}(\mathcal{R}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{a}_{t+1} \sim \pi}[Q_\pi(\mathcal{R}_{t+1}, \mathbf{a}_{t+1})]], \tag{6.4}$$

where θ and ϕ are the parameters of the value function Q_π and the relational reasoning module, respectively, γ represents the discount factor for future reward, r_{en} denotes an enhanced reward that combines the reward from IL module with the gained extrinsic reward when interacting

with the environment, as

$$r_{en}(o_t, a_t) = \alpha \cdot r_{im}(\mathcal{R}_t, a_t) + (1 - \alpha) \cdot r_{ex}(\mathcal{R}_t, a_t), \tag{6.5}$$

where α is a confidence weight parameter of the "expert" demonstration data, and a larger α means it is closer to the optimal policy; r_{im} and r_{ex} denote the reward comes from the IL module and external environment, respectively. The r_{im} evaluates how similar the action a_t is to the "expert policy" from demonstrations. The extrinsic rewards r_{ex} are provided as a sparse function to describe the fundamental limitations and rules that allow the agent to interact with the environment and steer it to the desired goals. The policy will be updated during the training process to choose the actions to raise Q_π by obtaining a larger r_{im} and r_{ex}.

As previously stated, the Discriminator training in the IL module aims to maximize the value $\mathrm{V}(\omega)$ in (6.3), but the updating policy π, which also serves as the generator of the IL module, tends to reduce it. This type of adversarial training results in a policy π that is rooted in the strategies offered in the demonstration data. Increasing extrinsic rewards r_ex will eventually lead to the trained policy π reacting to the environment with a better policy. The value functions Q_π of the proposed DRL network could be trained end-to-end by minimizing the following loss function:

$$\mathcal{L}(\theta, \phi) = \mathcal{L}^{Actor} - c_1\mathcal{L}^{Critic} + c_2\mathcal{H}, \tag{6.6}$$

where c_1 and c_2 are the discount factor for critic loss and entropy bonus, respectively. To better explain the process of updating policy, we introduce the above loss function more specifically as the objective function with respect to the $\varphi = (\theta, \phi)$ weighted policy π_φ:

$$\begin{aligned}\mathcal{J}(\varphi) = & -(\mathbb{E}_t\left[\min\left(f_t(\varphi)\bar{A}_t, \mathrm{Clip}\left(f_t(\varphi), 1-\epsilon, 1+\epsilon\right)\bar{A}_t\right)\right] \\ & + c_1\mathbb{E}_t\left[(\mathcal{V}_t^{\pi_\phi} - Q_\pi(\mathcal{R}, \mathbf{a}_t; \varphi))^2\right] - c_2\mathcal{H}),\end{aligned} \tag{6.7}$$

where ϵ denotes the amplitude of policy update, which is usually set to 0.1 or 0.2; $\mathcal{V}_t^{\pi_\phi}$ represents the reward returned by the current policy π_ϕ; $f_t(\varphi)$ is defined as

$$f_t(\varphi) = \frac{\pi_\varphi(\mathcal{R})}{\pi_{\varphi_{old}}(\mathcal{R})}, \tag{6.8}$$

where $\pi_{\varphi_{old}}$ and π_φ represent the policy prior to and after the training update, correspondingly; and $\bar{A}_t$ is the generalized advantage, which is an estimation that tells the agent whether the last decision is worth insisting on, which could be simply expressed as:

$$\hat{A}_t = \delta_t + \gamma\lambda\hat{A}_{t+1} \tag{6.9}$$

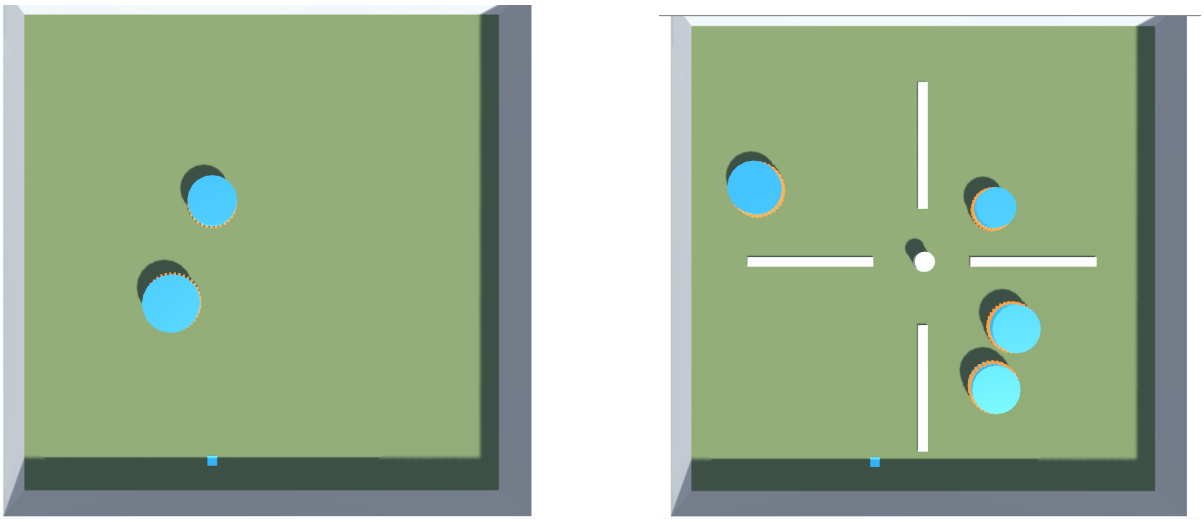
$$\delta_t = r_t + \gamma Q\left(\mathcal{R}_{t+1}\right) - Q\left(\mathcal{R}_t\right), \tag{6.10}$$

where γ is the discount factor for future reward; λ is a smoothing parameter used to lower training variance, hence making it more stable. For (6.7), the training process seeks to maximize $\mathcal{J}(\varphi)$ by ascending the stochastic gradient with regard to φ. As a result, based on the real-time multiple state spaces relational reasoning output, policy π_ϕ would tend to offer actions with the potential to impose greater Q values. The Clip function in (6.7) restricts the update range of π_ϕ, so that it would not update too greedy to fall into the local optimal trap, thus considerably improving the training stability.

6.5 Experiment Study

6.5.1 Experiment Setup

Using the Unity3D platform, we develop the proposed SRRL system in a simulated environment to mimic an application of deploying a mobile robot for RFID-based inventory management in an apparel store, which is the same scenario as in [2]. Unity3D is a robust game engine capable of rendering large, intricate 3D worlds. It also creates aesthetically realistic worlds with advanced mechanics and complicated interactions between agents of differing abilities. These features allow it to be frequently utilized as a simulation tool for study of various intelligent agents [142]. As depicted in Fig. 6.5, we construct two environments of four-walled area



(a) Simple inventory environment

(b) Complex inventory environment

Figure 6.5: Basic experimental setup for agent performing long-horizon RFID inventory tasks. The agent is represented as a blue cube, and the tags are orange strings attached to the blue cylinder-shaped racks.

of different complexity to imitate a $50 \times 50\text{m}^2$ apparel store: one with only racks represented by blue cylinders, and the other with more fixed-position obstacles (marked by the white lines and dot). Note that these obstacles do not impede the Lidar sensor's scanning but complicate the agent's action policy and path planning (i.e., they block the movement of the agent). Items with RFID tags are attached to the racks. A simulated robotic agent is sent out to scan all of the RFID tags in the region. At the start of each episode, the placements of the racks and the agent are created at random inside the enclosed area. Moreover, there will be a certain safety distance between the randomly generated racks and the obstacle in the complex environment scenario to ensure that the agent can pass smoothly and the task does not get. Indeed, the distances between racks also follow this requirement. Based on action $a_t = (v_t, \Delta_t)$, the agent simulates an RFID-equipped wheeled robot moving at a speed of $v_t \in [-v_{max}, v_{max}]$, while v_{max} represents the maximum velocity; $\Delta_t \in [-\Delta_{max}, \Delta_{max}]$ is the rotational velocity, where Δ_{max} denotes the maximum rotational velocity.

The agent's mission is to scan all the RFID tags in the the apparel store using the shortest possible path. Each scanning session terminates when all RFID tags are read or when the maximum number of steps is achieved. As one of the criteria for determining the quality of training results, the number of collisions and the step count for completing the given task

are also assessed. To collect observations, the robotic agent carries two sensors: a ray-cast sensor and a simulated RFID reader. Ray-casting is an optional sensor that Unity3D provides to simulate a common Lidar sensor. It detects the surrounding environment by projecting rays and returns a vector containing the observed items and their distances. We develop a virtual RFID reader for the agent to imitate the properties of real-world RFID applications. It can only scan tags within a detectable range and the probability of reading a tag reduces as the distance between the reader and the tag grows.

Pytorch is used to build the SRRL network on a computer with an Intel 9900K CPU and two Nvidia 2080 GPUs. Two CNNS with three convolutional layers and a stride of one make up the feature encoder. The Discriminator $\mathbf{D}$ is implemented with one LSTM layer and two fully connected hidden layers, each with 128 units. Both the value function Q_π and policy π in the DRL module have an LSTM layer with 128 units and three hidden layers, each with 256 units. The basic training configuration has been summarized in Table 6.1. For the remaining experiments, We set the proportion parameter μ in (6.5) to 0.1. This robotic agent will be put in the simulated apparel store during the experiments, with its starting location being randomly generated at each episode.

6.5.2 Results and Analysis

Training Results

In the training process, we train the agent in the simple environment with two racks, as shown in Fig. 6.5a, to quickly converge with the ability of reasoning the latent relationship among multiple state spaces. We will then test this well-trained model in the complex environment presented in Fig. 6.5b during the testing stage, which includes additional randomly created racks and certain obstacles that hinder Lidar scanning. The number of steps in each training episode is capped at 2×10^4 to avoid the unnecessarily long training time. In addition, once the number of collisions of the agent in an episode reaches 20, this episode will be instantly terminated and marked as a failure, and the amount of steps cost will be recorded as the maximum number of 2×10^4. Each training episode also ends immediately if the agent scans all

Table 6.1: Basic Training Configuration

RL Parameter	Value
learning rate	2.0e-4
gamma(discount factor)	0.99
hidden layer units	256
sequence length	64
batch size	1024
memory size	256
max steps	5e6

CNN Parameter	Value
convolutional layer num	3
stride	1
kernel size	[4×4];[3×3];[3×3]
FC layer num	2
hidden units	128

the RFID tags on the racks, and the agent will earn the "find" reward. Otherwise, it performs tasks until the maximum number of allowed steps is achieved. Our method is compared with the three state-of-the-art models: (i) the GAIL model proposed in [119], (ii) the PPO network proposed in [140], and (iii) the RIRL model proposed in our recent work [180], which is an RL and IL combined method but without a relational reasoning module. GAIL and PPO, two approaches without memory mechanisms, are used in our comparison study to show the influence of the recurrent network in solving long-horizon robotic tasks. To provide a fair comparison, we employ the same basic reward levels and training parameters (i.e., learning rate, number of targets achieved, the maximum number of steps, etc.) in the same environment for all the four approaches.

Cumulative rewards for each training episode are depicted in Fig. 6.6 as the agent interacts with the environment. We basically specified a few simple and sparse reward configurations: reading a new RFID tag earns $+0.01$ point, colliding is punished by -0.1 point, moving costs -0.0001 point for each step, and completing the task gets $+1$ point. The SRRL, represented by the red solid line in Fig. 6.6, achieves the best reward results after about 60 training episodes,

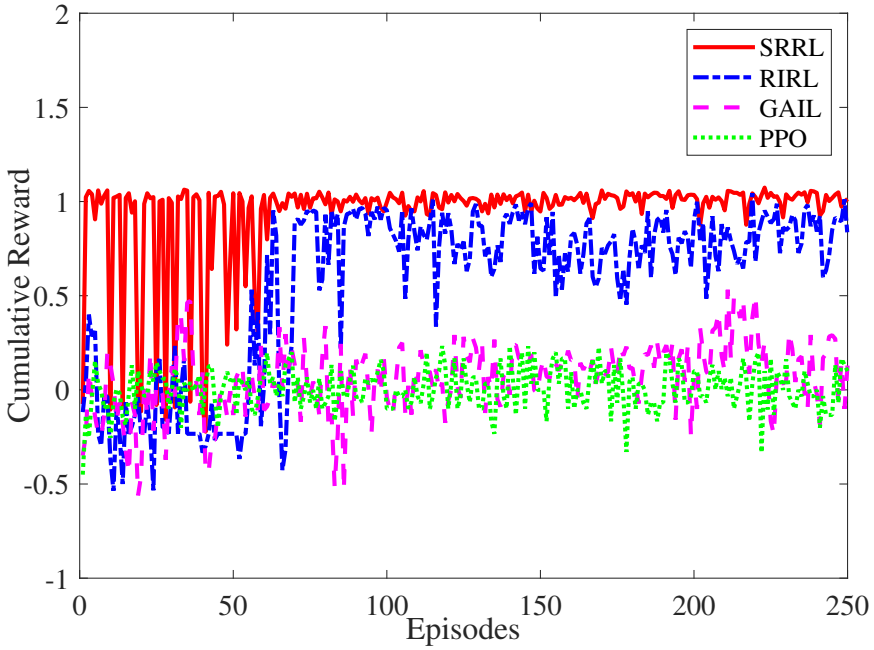


Figure 6.6: Accumulated training reward values for SRRL, RIRL, PPO, and GAIL methods.

which is both greater and more consistent than the other three methods. Although the RIRL approach also generates significantly superior results than GAIL and PPO, its convergence speed and stability are inferior to our proposed method. After the SRRL and RIRL model converges, there are still a certain degree of small-scale fluctuation since the environment generated each time is unique. These results validate that our proposed method and RIRL model with LSTM embedding can achieve higher cumulative reward faster and more consistently than GAIL and PPO, two networks without using the recurrent network.

Additionally, as described in Fig. 6.7, our method SRRL performs better than RIRL because it takes fewer steps to complete a task in one episode. As mentioned before, each training episode will be terminated if all the targets are reached prior to the maximum number of steps. We can tell that our proposed method's results stabilize much faster than that of the RIRL model, which almost takes about 160 training episodes to converge. The agent implemented with SRRL could stably and consistently handle the given tasks within 1,500 steps. This result indicates that the reasoning mechanism in our proposed approach, which has the abstraction ability within long-term memory, is well suited to help the robot understand the nature of the task in advance and perform the task consistently and efficiently. The other two methods without LSTM embedding cannot even complete the assigned task and usually takes the maximum

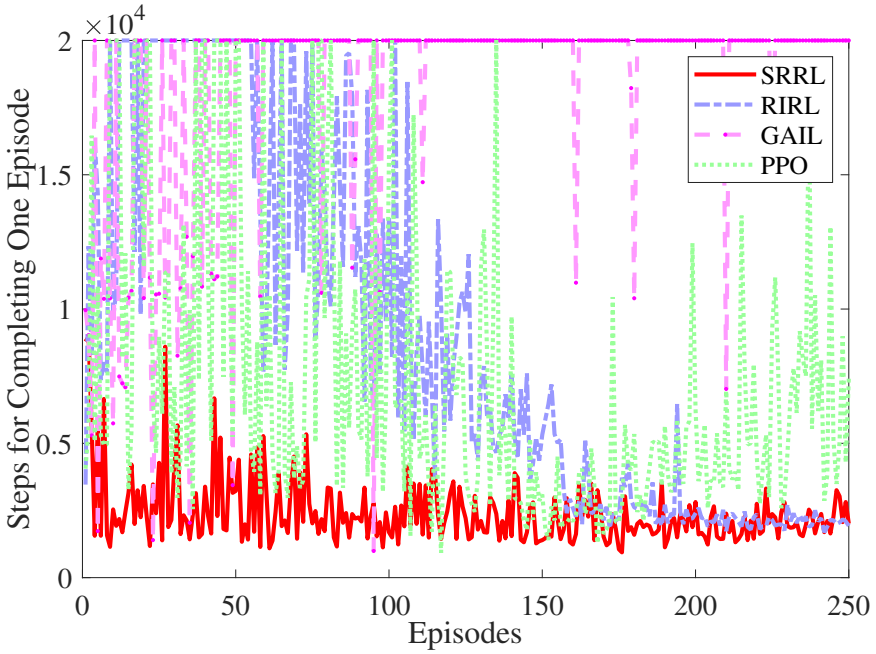


Figure 6.7: Steps for finishing the tag scanning task within one episode during the training phase.

number of steps. To further compare the models' performance during the training process, we plot the training loss values for each method in Fig. 6.8. Obviously, the loss values of SRRL and RIRL are convergent, whereas the loss curves for PPO and GAIL do not. This is because that PPO and GAIL are incapable of finding a reliable strategy to accomplish the long-horizon robotic searching and planning tasks reliably and efficiently.

Training these methods usually takes days. And even if our method SRRL leverages Imitation Learning to reduce some of the search time, it often takes more than 12 hours to make the agent understand and complete the task perfectly. In the testing stage, we usually do not change the configuration except for the environment, keeping it consistent with that in training. Typically, if the state space and action space are not too big, it takes an agent with our method about 1 ms to choose an action. This is also true for the other methods. But if the size of the map and the number of degrees of freedom of the action are very large, it will take about 10ms longer to make an action with our method, and about 1.5 seconds for the other methods to choose the next action.

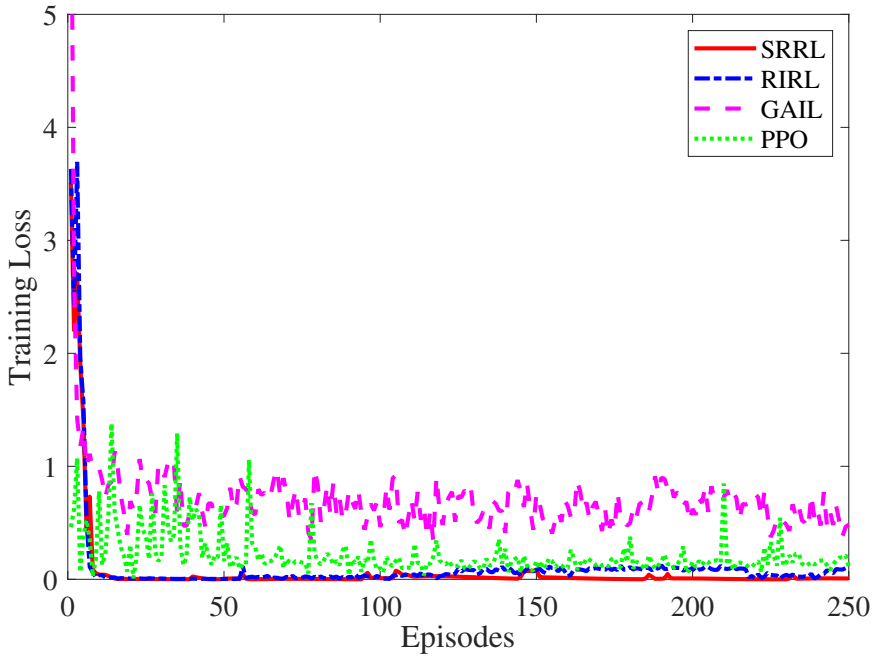


Figure 6.8: Training loss of SRRL, RIRL, GAIL, and PPO.

RFID Tag Scanning Results

The cumulative distribution function (CDF) of the percentage of unscanned tags in the testing stage is shown in Fig. 6.9. We test all four trained models in 100 episodes within the required 20,000 steps. The figure shows that the proposed SRRL model scans all tags in approximately 95% of episodes, while the RIRL without reasoning scheme scans all tags in about 84% of episodes. On the contrary, the GAIL and PPO models cannot consistently scan all the tags in an episode. In addition, SRRL achieves a maximum unscanned tag percentage of 20%, which is significantly lower than the 38% of RIRL, as well as the almost 90% attained by the other two approaches. Apparently, our proposed method is significantly more effective and robust for long-term search tasks in dynamic environments.

To further illustrate the superiority of our method, we evaluate the well-trained model of the four approaches mentioned above in both simple and complex environments using two indicators: (i) the average number of collisions in an episode, (ii) the average number of steps required to complete the given task in an episode. Moreover, in order to demonstrate the versatility and robustness of our model, we first train it in a simple environment as the one shown

in Fig. 6.5a and then deploy the well-trained model to both simple and complex environments shown in Fig. 6.5 with a varied number of racks in one hundred testing episodes.

Fig. 6.10 presents the average number of steps for completing the given tasks in a simple environment within 100 testing episodes. We can see that the SRRL method could cost-efficiently accomplish the tasks, as it only needs about 5,500 steps for the four-rack scenario, whereas the other three approaches struggle to do so. Note that the average step cost for PPO and GAIL almost nearly surpasses 2×10^4, which indicates they failed to complete the tasks after reaching the maximum number of steps in almost all episodes. Second, avoiding collisions is crucial when deploying a robot for various purposes. Hence, the frequency of collisions for the agent is an essential metric for evaluating the quality of our work. According to Fig. 6.11, when there are more racks in the simple environment, the average collision times rise as the number of racks is increased. It is obvious that GAIL and PPO perform poorly regardless of the number of racks while avoiding accidents. Although RIRL is relatively better at avoiding obstacles than GAIL and PPO, its average number of collisions increases by around eight times as the number of racks grows from two to four. While our method exhibits robust performance across all four cases, with a variance of the four average collision times less than 0.5. We also provide the error bars in the figures to show the robustness of each method during testing. A smaller error bar generally means that the method is more stable, while the opposite means that the method does not generalize well to dynamically changing unknown environments. For example, our proposed method in Fig. 6.10 and Fig. 6.11 is significantly better and more stable than the other methods in terms of the average step collision number. RIRL maintains high stability for a small number of racks but is not robust for scenarios with more than two racks that are not covered in training. Certainly, we can also observe that the results obtained by the two methods, PPO and GAIL, are less volatile because they have been completing their tasks very consistently poorly.

Similarly, the complete number of steps and collision times per testing episode for the complex environment are presented in Fig. 6.12 and Fig. 6.13, respectively. Although the quantity of these two measures for our method increases when confronted with a more complex environment, they remain within a respectable range and demonstrate that our SRRL model

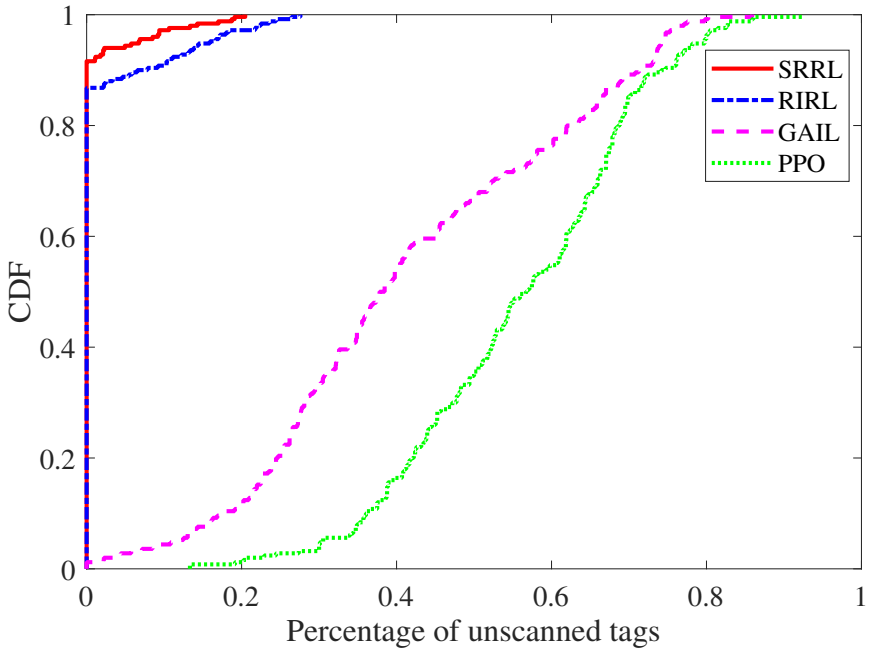


Figure 6.9: CDF of the percentage of unscanned tags in total in the testing stage.

has high transferability and robustness to dynamic, unknown environments. Note that in these two figures, the error bars of PPO and GAIL are zero in complex environments with multi-rack scenarios, which means that all the attempts of these two methods end in failure, i.e., the number of steps and collisions have all reached the error tolerance limit (thus no variance). From these two figures, we can draw the same conclusion for scenarios of both the simple environment and the complex environment that our method could tackle long-horizon robotic planning tasks much more effectively and efficiently with the incorporation of the relational reasoning module.

6.6 Conclusions

In this paper, we proposed SRRL, a deep recurrent imitation and reinforcement learning-based system augmented with a relational reasoning module that allows the agent to accomplish long-horizon robotic searching and planning tasks in dynamic, complex situations. To the best of our knowledge, this is the first work in the field of DRL that teaches agents how to reason from various state spaces to learn the optimal policy. Furthermore, using historical observations to create state spaces can improve the model and alleviate the long-range dependency problem. We experimentally validated the feasibility of incorporating a relational reasoning module in

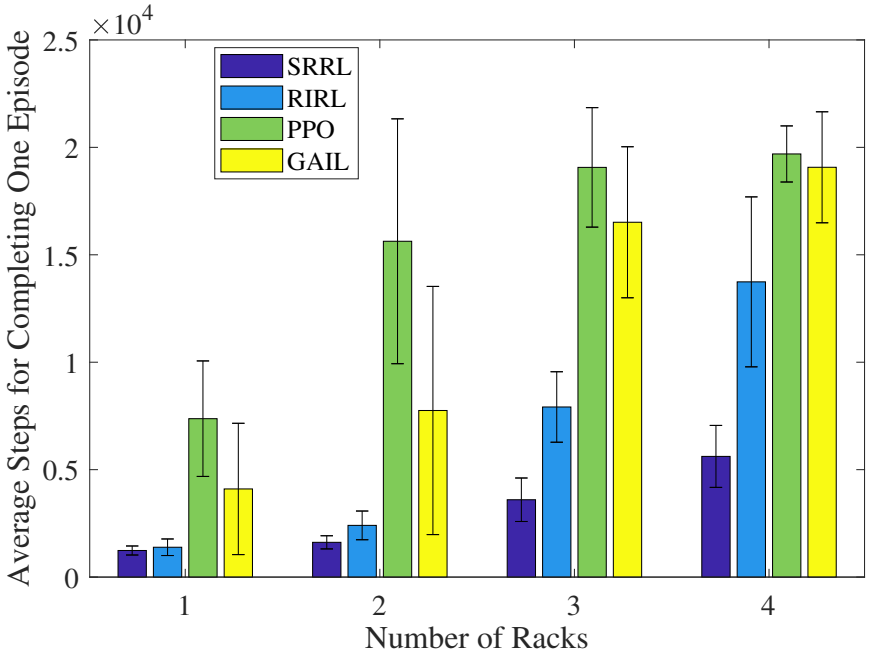


Figure 6.10: Average number of steps to complete the task in 100 episodes in the testing stage in the simple environment.

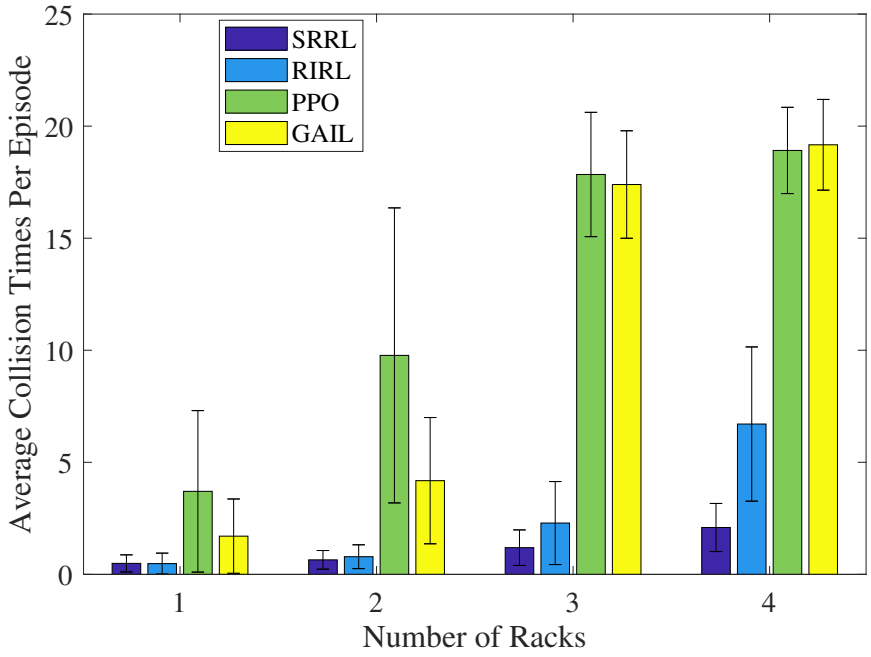


Figure 6.11: Average number of collisions of the agent per episode in the simple environment.

traditional DRL methods by performing in a visual game-based simulation environment. The excellent results demonstrated the effectiveness of encouraging agents to learn strategies from extracted latent correlations across multiple state spaces to complete such long-horizon tasks.

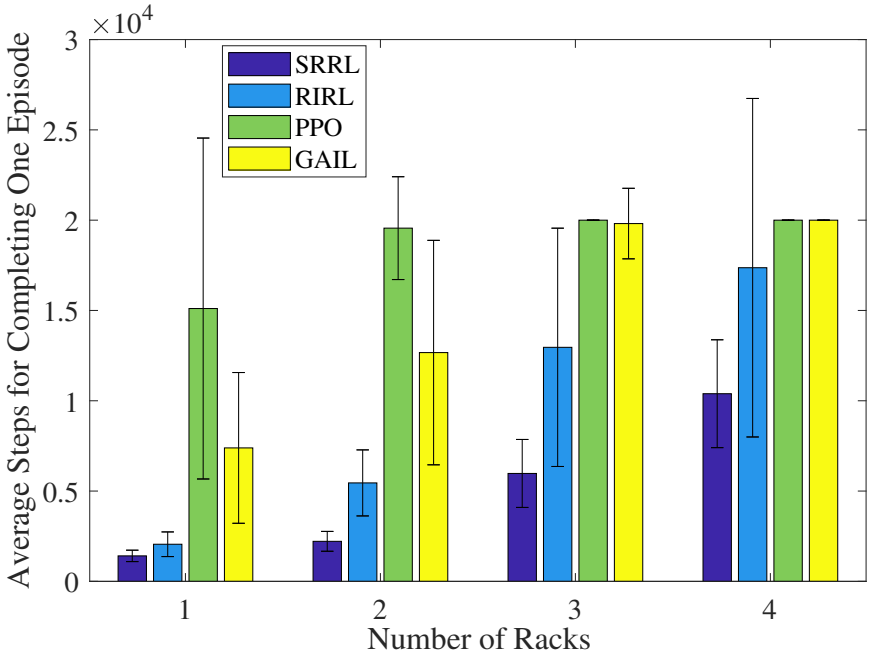


Figure 6.12: Average number of the task completion steps in 100 episodes of the testing stage in the complex environment.

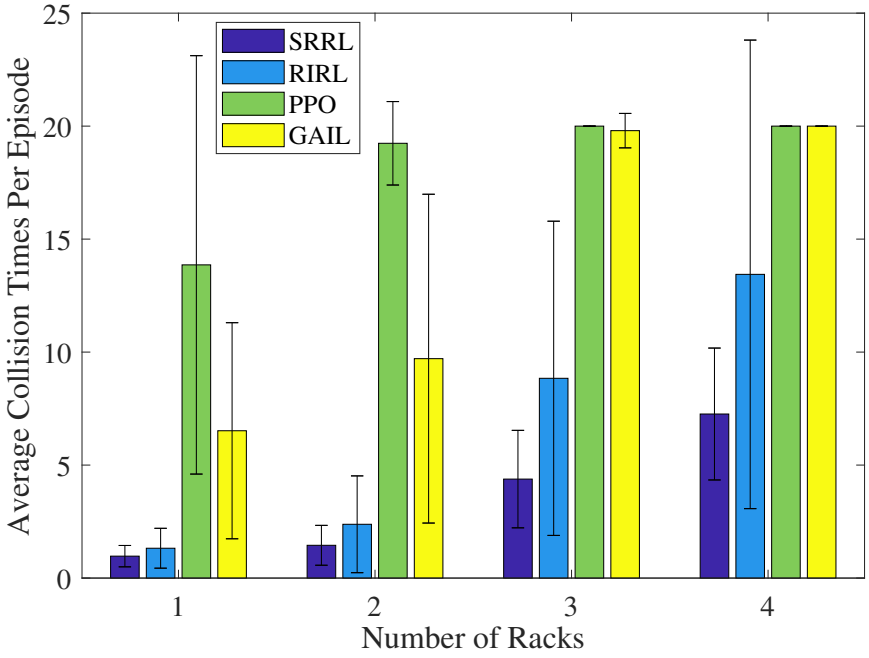


Figure 6.13: Average collision times that happened in agent per episode in the complex environment.

Chapter 7

Summary and Future Work

7.1 Summary

During my doctoral studies, I concentrated on creating workable solutions to the inherent issues with RL-based robotic systems. The main goal of this system is to empower ground robots and unmanned aerial vehicles (UAVs) to assign and accomplish tasks using reasoning, particularly when dealing with long-horizon tasks and dynamic environments. It not only focuses on robots but also RFID technology applications. To begin, RFUAV creates an indoor localization system for UAVs by combining precise 6-DoF orientation and location estimation with a commercially available RFID reader and tags. To estimate the location of tags with phase differences, a Bayesian filter is used. The pose of the UAV is then estimated using an SVD-based algorithm. In the work of RFHUI, we extended the work of RFUAV to leverage precise RFID tag positioning to assist in controlling the aerial pose of UAVs in 6-DOF. With the UAV localization system, we investigated the cooperation system that enables UGVs and UAVs to form coalitions for the complementary accomplishment of tasks that neither the UAVs or UGVs could not complete independently. IADRL learns the UGV-UAV coalition's complementing behavior traits from a demonstration dataset that can be easily obtained from both simple and problematic circumstances. It also optimizes the method for achieving given goals while experiencing the lowest overall costs to finish tasks in dynamic environments. Finally, we proposed the SRRL, a deep recurrent imitation and reinforcement learning-based system augmented with a relational reasoning module, to further investigate and improve the performance

of long-horizon robotic searching and planning tasks in dynamic and complicated environments. SRRL teaches agents how to reason from different state spaces in order to discover the best policy. Additionally, creating state spaces from previous observations helps strengthen the model and relieve the long-term dependency issue.

7.2 Future Work

Although we have investigated a lot of reinforcement learning-based methods, many interesting research topics are still open in the field of developing intelligent robotic systems.

7.2.1 Sim2Real Gap

The name ”Sim2Real” is an acronym of the phrase ”simulation to reality”. The majority of research on RL-based robotic applications is still in the simulation environment. The wide gap between simulation and reality makes it difficult to implement the reinforcement learning model. We are focusing on implementing well-trained RL models to guide real robots in solving long-horizon tasks in dynamic environments. Deploying these robots in continuous observation and action settings is quite difficult, especially when it comes to training an ideal RL model and realistic robotic mechanical manipulation.

7.2.2 Virtual Reality and Digital Twins

Virtual reality technology could simulate highly realistic scenarios, such as interior inventory warehouses, by utilizing computer modeling and some graphical software. We intend to imitate the observations that the robot can make in reality by creating a virtual scene that is more refinedly similar to the actual scene. This will narrow the gap of sim2real to a certain extent and help the robot to complete tasks according to the trained model in different real-world environments faster and smoother.

7.2.3 Privacy Security Issues Related to Robots

As robots become highly common in human daily life and manufacturing, privacy concerns relating to robot use have gained increasing attention. Since the robots need to observe its

surrounding and collect information for better decisions, sensors such as cameras,radars, and even microphones are required. These sensors are typically access the most sensitive privacy information of users, such as photos of private rooms, voice recordings among family members, etc.

List of Publications

1. Zhitao Yu, Jian Zhang, Shiwen Mao, Senthilkumar CG Periaswamy, and Justin Patton, "Multi-state-space reasoning reinforcement learning for long-horizon RFID-based robotic searching and planning tasks," Journal of Communications and Information Networks, vol.7, no.3, pp.239-251, Sept. 2022. DOI: 10.23919/JCIN.2022.9906938.

2. Zhitao Yu, Jian Zhang, Shiwen Mao, Senthilkumar CG Periaswamy, and Justin Patton, "RIRL: A recurrent imitation and reinforcement learning method for long-horizon robotic tasks," in Proc. IEEE CCNC 2022, Las Vegas, NV, Jan. 2022, pp.230-235. (The IEEE CCNC 2022 Runner-up of the Best Paper Award)

3. Xiangyu Wang, Zhitao Yu, Shiwen Mao, Jian Zhang, Senthilkumar CG Periaswamy and Justin Patton, "MapLoc: LSTM-based Location Estimation using Uncertainty Radio Maps," in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2023.3262619.

4. Jian Zhang, Zhitao Yu(co-first author), Shiwen Mao, Senthilkumar CG Periaswamy, Justin Patton, and Xue Xia, "IADRL: An imitation augmented deep reinforcement learning network enabled UGV-UAV complementary coalitions," IEEE Access Journal, Special Section on Advanced Communications and Networking Techniques for Wireless Connected Intelligent Robot Swarms, vol.8, no.1, pp.102335-102347, June 2020. DOI: 10.1109/ACCESS.2020.2997304.

5. Jian Zhang, Zhitao Yu, Xiangyu Wang, Yibo Lyu, Shiwen Mao, Senthilkumar C.G. Periaswamy, Justin Patton, and Xuyu Wang, "RFHUI: An intuitive and easy-to-operate human-UAV interaction system for controlling a UAV in a 3D space," in Proc. EAI MobiQuitous 2018, New York City, NY, Nov. 2018, pp.69-76.

6. Jian Zhang, Zhitao Yu, Xiangyu Wang, Yibo Lyu, Shiwen Mao, Senthilkumar CG Periaswamy, Justin Patton, and Xuyu Wang, "RFHUI: An RFID based human-unmanned aerial vehicle interaction system in an indoor environment," Elsevier Digital Communications and Networks Journal, vol.6, no.1, pp.14-22, Feb. 2020. DOI: 10.1016/j.dcan.2019.05.001. (included in DCN's High-influence Article Collection, Nov. 2020)

7. Xuyu Wang, Zhitao Yu, and Shiwen Mao, "DeepML: Deep LSTM for indoor localization with smartphone magnetic and light sensors," in Proc. IEEE ICC 2018, Kansas City, MO, May 2018.

8. Xuyu Wang, Zhitao Yu, and Shiwen Mao, "Indoor localization using magnetic and light sensors with smartphones: A deep LSTM approach," Springer Mobile Networks and Applications (MONET) Journal, Special Issue on Towards Future Ad Hoc Networks: Technologies and Applications, vol.25, no.2, pp.819-832, Apr. 2020. DOI: 10.1007/s11036-019-01302-x.

9. Jian Zhang, Xiangyu Wang, Zhitao Yu, Yibo Lyu, Shiwen Mao, Senthilkumar CG Periaswamy, Justin Patton, and Xuyu Wang, "Robust RFID based 6-DoF localization for unmanned aerial vehicles," IEEE Access Journal, Special Section on Network Resource Management in Flying Ad Hoc Networks: Challenges, Potentials, Future Applications, and Wayforward, vol.7, no.1, pp. 77348-77361, June 2019. DOI: 10.1109/ACCESS.2019.2922211.

10. Xiangyu Wang, Jian Zhang, Zhitao Yu, Shiwen Mao, Senthilkumar C.G. Periaswamy, and Justin Patton, "On remote temperature sensing using commercial UHF RFID tags," IEEE Internet of Things Journal, vol.6, no.6, pp. 10715-10727, Dec. 2019. DOI: 10.1109/JIOT.2019.2941023.

References

[1] T. Roppel, Y. Lyu, J. Zhang, X. Xia *et al.*, "Corrosion detection using robotic vehicles in challenging environments," in *CORROSION 2017*, Louisiana, USA, March 2017.

[2] J. Zhang, Y. Lyu, T. Roppel, J. Patton, and C. Senthilkumar, "Mobile robot for retail inventory using rfid," in *2016 IEEE international conference on Industrial technology (ICIT)*, Taipei, Taiwan, March 2016, pp. 101–106.

[3] X. Xia, T. Roppel, J. Zhang, Y. Lyu, S. Mao, S. C. Periaswamy, and J. Patton, "Enabling a mobile robot for autonomous rfid-based inventory by multilayer mapping and aco-enhanced path planning," *Online Journal of Robotics & Automation Technology*, Sept. 2019.

[4] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-uav enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, Jan. 2018.

[5] J. Zhang, Z. Yu, X. Wang, Y. Lyu, S. Mao, S. C. Periaswamy, J. Patton, and X. Wang, "Rfhui: An intuitive and easy-to-operate human-uav interaction system for controlling a uav in a 3d space," in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, New York, USA, Nov. 2018, pp. 69–76.

[6] X. Wang, S. Mao, and M. Gong, "A survey of lte wi-fi coexistence in unlicensed bands," *ACM GetMobile: Mobile Computing and Communications Review*, vol. 20, no. 3, pp. 17–23, July 2016.

[7] Y. Xu, G. Yue, and S. Mao, "User grouping for massive MIMO in FDD systems: New design methods and analysis," *IEEE Access Journal*, vol. 2, no. 1, pp. 947–959, Sept. 2014.

[8] J. Zhang, Z. Yu, X. Wang, Y. Lyu, S. Mao, S. C. Periaswamy, J. Patton, and X. Wang, "Rfhui: An rfid based human-unmanned aerial vehicle interaction system in an indoor environment," *Digital Communications and Networks*, vol. 6, no. 1, pp. 14–22, May 2019.

[9] J. Zhang, X. Wang, Z. Yu, Y. Lyu, S. Mao, S. C. Periaswamy, J. Patton, and X. Wang, "Robust rfid based 6-dof localization for unmanned aerial vehicles," *IEEE Access*, vol. 7, pp. 77 348–77 361, June 2019.

[10] S. Duan, D. Wang, J. Ren, F. Lyu, Y. Zhang, H. Wu, and X. Shen, "Distributed artificial intelligence empowered by end-edge-cloud computing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 591–624, 2023.

[11] S. Duan, F. Lyu, H. Wu, W. Chen, H. Lu, Z. Dong, and X. Shen, "Moto: Mobility-aware online task offloading with adaptive load balancing in small-cell mec," *IEEE Transactions on Mobile Computing*, pp. 1–16, 2022.

[12] N. Tang, S. Mao, and R. M. Nelms, "Adversarial attacks to solar power forecast," in *Proc. IEEE GLOBECOM 2021*, Madrid, Spain, Dec. 2021, pp. 1–6.

[13] Z. Bao, Y. Lin, S. Zhang, Z. Li, and S. Mao, "Threat of adversarial attacks on DL-based IoT device identification," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 9012–9024, June 2022.

[14] K. Xiao, S. Mao, and J. Tugnait, "Hierarchical radio resource allocation for network slicing in fog radio access networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3866–3881, Apr. 2019.

[15] M. Feng, S. Mao, and T. Jiang, "Dealing with link blockage in mmwave networks: A combination of d2d relaying, multi-beam reflection, and handover," *IEEE Transactions on Wireless Communications*.

[16] T. Zhang and S. Mao, "Energy-efficient federated learning with intelligent reflecting surface," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 2, pp. 845–858, June 2022.

[17] N. Tang, S. Mao, Y. Wang, and R. Nelms, "Solar power generation forecasting with a LASSO-based approach," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1090–1099, Apr. 2018.

[18] L. Wang, S. Mao, B. Wilamowski, and R. Nelms, "Pre-trained models for non-intrusive appliance load monitoring," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 56–68, Mar. 2022.

[19] M. Feng, S. Mao, and T. Jiang, "Boost: Base station on-off switching strategy for energy efficient massive mimo hetnets," in *Proc. IEEE INFOCOM 2016*, San Francisco, CA, Apr. 2016, pp. 1395–1403.

[20] T. Zhang and S. Mao, "Machine learning for end-to-end congestion control," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 52–57, June 2020.

[21] Z. He, S. Mao, S. Kompella, and A. Swami, "On link scheduling in dual-hop 60 ghz mmwave networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 11 180–11 192, Dec. 2017.

[22] K. Xiao, S. Mao, and J. Tugnait, "MAQ: A multiple model predictive congestion control scheme for cognitive radio networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 4, pp. 2614–2626, Apr. 2017.

[23] S. Mao, Y. T. Hou, X. Cheng, H. D. Sherali, S. F. Midkiff, and Y.-Q. Zhang, "On routing for multiple description video over wireless ad hoc networks," *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1063–1074, Oct. 2006.

[24] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146 264–146 272, Sept. 2019.

[25] Z. Yu, J. Zhang, S. Mao, S. C. Periaswamy, and J. Patton, "Multi-state-space reasoning reinforcement learning for long-horizon RFID-based robotic searching and planning tasks," *Journal of Communications and Information Networks*, vol. 7, no. 3, pp. 239–251, Sept. 2022.

[26] X. Wang, J. Zhang, S. Mao, S. C. Periaswamy, and J. Patton, "Locating multiple RFID tags with Swin Transformer-based RF hologram tensor filtering," in *Proc. IEEE VTC-Fall 2022*, London, UK, Sept. 2022.

[27] X. Wang, X. Wang, S. Mao, J. Zhang, S. Periaswamy, and J. Patton, "Adversarial deep learning for indoor localization with channel state information tensors," *IEEE Internet of Things Journal*, vol. 9, no. 19, pp. 18 182–18 194, Oct. 2022.

[28] X. Wang, J. Zhang, S. Mao, S. Periaswamy, and J. Patton, "MulTLoc: RF hologram tensor filteing and upscaling for indoor localization using multiple UHF passive RFID tags," in *Proc. ICCCN 2021*, Athens, Greece, July 2021.

[29] S. Thrun, "Probabilistic algorithms in robotics," *Ai Magazine*, vol. 21, no. 4, pp. 93–93, Dec. 2000.

[30] K. P. Valavanis, "Advances in unmanned aerial vehicles: state of the art and the road to autonomy," *Design and Control of a Miniature Quadrotor*, vol. 33, pp. 171–210, 2007.

[31] J. Courbon, Y. Mezouar, N. Guénard, and P. Martinet, "Vision-based navigation of unmanned aerial vehicles," *Control Engineering Practice*, vol. 18, no. 7, pp. 789–799, July 2010.

[32] B. Hérissé, T. Hamel, R. Mahony, and F.-X. Russotto, "A terrain-following control approach for a vtol unmanned aerial vehicle using average optical flow," *Autonomous robots*, vol. 29, no. 3, pp. 381–399, Sept. 2010.

[33] S. Klose, J. Wang, M. Achtelik, G. Panin, F. Holzapfel, and A. Knoll, "Markerless, vision-assisted flight control of a quadrocopter," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 5712–5717.

[34] K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani, "Monocular vision slam for indoor aerial vehicles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA, Oct. 2009, pp. 1566–1573.

[35] J. Engel, J. Sturm, and D. Cremers, "Camera-based navigation of a low-cost quadrocopter," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vilamoura-Algarve, Portugal, Oct. 2012, pp. 2815–2821.

[36] A. Chiuso, P. Favaro, H. Jin, and S. Soatto, "Structure from motion causally integrated over time," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 523–535, Aug 2002.

[37] A. Noureldin, T. B. Karamat, M. D. Eberts, and A. El-Shafie, "Performance enhancement of mems-based ins/gps integration for low-cost navigation applications," *IEEE Transactions on vehicular technology*, vol. 58, no. 3, pp. 1077–1096, May 2008.

[38] K. Pahlavan, X. Li, and J.-P. Makela, "Indoor geolocation science and technology," *IEEE communications magazine*, vol. 40, no. 2, pp. 112–118, Aug. 2002.

[39] X. Wang, L. Gao, S. Mao, and S. Pandey, "DeepFi: Deep learning for indoor fingerprinting using channel state information," in *Proc. WCNC'15*, New Orleans, LA, Mar. 2015, pp. 1666–1671.

[40] J. Xiao, Z. Zhou, Y. Yi, and L. M. Ni, "A survey on wireless indoor localization from the device perspective," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–31, June 2016.

[41] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, "Spotfi: Decimeter level localization using wifi," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, vol. 14, New York, NY, USA, Aug. 2015, pp. 269–282.

[42] X. Wang, L. Gao, and S. Mao, "CSI phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1113–1123, Dec. 2016.

[43] ——, "BiLoc: Bi-modality deep learning for indoor localization with 5GHz commodity Wi-Fi," *IEEE Access Journal*, vol. 5, no. 1, pp. 4209–4220, Mar. 2017.

[44] X. Wang, X. Wang, and S. Mao, "ResLoc: Deep residual sharing learning for indoor localization with CSI tensors," in *Proc. IEEE PIMRC 2017*, Montreal, Canada, Oct. 2017.

[45] X. Wang, Z. Yu, S. Mao, J. Zhang, S. C. Periaswamy, and J. Patton, "Maploc: Lstm-based location estimation using uncertainty radio maps," *IEEE Internet of Things Journal*, pp. 1–1, Mar. 2023.

[46] X. Wang, L. Gao, S. Mao, and S. Pandey, "CSI-based fingerprinting for indoor localization: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 1, pp. 763–776, Jan. 2017.

[47] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, LomBard,IL,USA, Apr. 2013, pp. 71–84.

[48] J. Wang, D. Vasisht, and D. Katabi, "Rf-idraw: Virtual touch screen in the air using rf signals," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 235–246, Aug. 2014.

[49] S. Pradhan, E. Chai, K. Sundaresan, L. Qiu, M. A. Khojastepour, and S. Rangarajan, "Rio: A pervasive rfid-based touch gesture interface," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, Snowbird, Utah, USA, Oct. 2017, pp. 261–274.

[50] L. Shangguan, Z. Yang, A. X. Liu, Z. Zhou, and Y. Liu, "Relative localization of {RFID} tags using spatial-temporal phase profiling," in *12th {USENIX} Symposium on*

Networked Systems Design and Implementation ({NSDI} 15), Oakland,CA,USA, May 2015, pp. 251–263.

[51] X. Wang, Z. Yu, and S. Mao, "Deepml: Deep lstm for indoor localization with smartphone magnetic and light sensors," in *Proc. ICC'18)*, Kansas City, MO, July 2018, pp. 1–6.

[52] X. Wang, X. Wang, S. Mao, J. Zhang, S. Periaswamy, and J. Patton, "Indoor radio map construction and localization with deep Gaussian Processes," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 11 238–11 249, Nov. 2020.

[53] B. R. Stojkoska, J. Palikrushev, K. Trivodaliev, and S. Kalajdziski, "Indoor localization of unmanned aerial vehicles based on rssi," in *IEEE EUROCON 2017-17th International Conference on Smart Technologies*, Ohrid, Macedonia, July 2017, pp. 120–125.

[54] J. Tiemann, F. Schweikowski, and C. Wietfeld, "Design of an uwb indoor-positioning system for uav navigation in gnss-denied environments," in *2015 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Banff, AB, Canada, Oct. 2015, pp. 1–7.

[55] K. Li, C. Wang, S. Huang, G. Liang, X. Wu, and Y. Liao, "Self-positioning for uav indoor navigation based on 3d laser scanner, uwb and ins," in *2016 IEEE International Conference on Information and Automation (ICIA)*, Ningbo, China, Aug. 2016, pp. 498–503.

[56] B. Hardgrave, "Try it you'll like it!-the rfid lab's annual state-of-adoption report of us retailers," *RFID Journal*, Aug. 2015.

[57] J. Zhang, Y. Lyu, J. Patton, S. C. Periaswamy, and T. Roppel, "Bfvp: A probabilistic uhf rfid tag localization algorithm using bayesian filter and a variable power rfid model," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 8250–8259, Feb. 2018.

[58] Y. Zuo, "Survivable rfid systems: Issues, challenges, and techniques," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 40, no. 4, pp. 406–418, Apr. 2010.

[59] S. Azzouzi, M. Cremer, U. Dettmar, R. Kronberger, and T. Knie, "New measurement results for the localization of uhf rfid transponders using an angle of arrival (aoa) approach," in *2011 IEEE International Conference on RFID*, Orlando, FL, USA, May 2011, pp. 91–97.

[60] X. Wang, S. Mao, S. Pandey, and P. Agrawal, "CA2T: Cooperative antenna arrays technique for pinpoint indoor localization," in *Proc. MobiSPC 2014*, Niagara Falls, Canada, Aug. 2014, pp. 392–399.

[61] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu, "Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, Maui, Hawaii, USA, Sept. 2014, pp. 237–248.

[62] X. Wang, C. Yang, and S. Mao, "TensorBeat: Tensor decomposition for monitoring multi-person breathing beats with commodity WiFi," *ACM Transactions on Intelligent Systems and Technology*, vol. 9, no. 1, pp. 8:1–8:27, Sept. 2017.

[63] X. Wang, X. Wang, S. Mao, J. Zhang, S. Periaswamy, and J. Patton, "DeepMap: Deep Gaussian Process for indoor radio map construction and location estimation," in *Proc. IEEE GLOBECOM 2018*, Abu Dhabi, United Arab Emirates, Dec. 2018.

[64] F. Gandino, B. Montrucchio, M. Rebaudengo, and E. R. Sanchez, "On improving automation by integrating rfid in the traceability management of the agri-food sector," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 7, pp. 2357–2365, Apr. 2009.

[65] J. Zhang, S. C. Periaswamy, S. Mao, and J. Patton, "Standards for passive uhf rfid," *GetMobile: Mobile Comp. and Comm.*, vol. 23, no. 3, pp. 10–15, Jan 2020.

[66] J. S. Choi, B. R. Son, H. K. Kang, and D. H. Lee, "Indoor localization of unmanned aerial vehicle based on passive uhf rfid systems," in *2012 9th international conference on ubiquitous robots and ambient intelligence (URAI)*, Daejeon, Korea (South), Nov. 2012, pp. 188–189.

[67] Y. Bu, L. Xie, J. Liu, B. He, Y. Gong, and S. Lu, "3-dimensional reconstruction on tagged packages via rfid systems," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, San Diego, CA, USA, June 2017, pp. 1–9.

[68] T. Wei and X. Zhang, "Gyro in the air: tracking 3d orientation of batteryless internet-of-things," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, New York City, New York, Oct. 2016, pp. 55–68.

[69] Q. Lin, L. Yang, Y. Sun, T. Liu, X.-Y. Li, and Y. Liu, "Beyond one-dollar mouse: A battery-free device for 3d human-computer interaction via rfid tags," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, Hong Kong, China, Apr. 2015, pp. 1661–1669.

[70] M. Z. Anwar, Z. Kaleem, and A. Jamalipour, "Machine learning inspired sound-based amateur drone detection for public safety applications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 3, pp. 2526–2534, Jan. 2019.

[71] Z. Kaleem and M. H. Rehmani, "Amateur drone monitoring: State-of-the-art architectures, key enabling technologies, and future research directions," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 150–159, May 2018.

[72] S. Chen and K. Ho, "Accurate localization of a rigid body using multiple sensors and landmarks," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6459–6472, Dec. 2015.

[73] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Computer Architecture Letters*, vol. 13, no. 04, pp. 376–380, Apr. 1991.

[74] D. W. Eggert, A. Lorusso, and R. B. Fisher, "Estimating 3-d rigid body transformations: a comparison of four major algorithms," *Machine vision and applications*, vol. 9, no. 5, pp. 272–290, Mar. 1997.

[75] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, Nara, Japan, Nov. 2007, pp. 225–234.

[76] Y. Chen, R. Huang, and Y. Zhu, "A cumulative error suppression method for uav visual positioning system based on historical visiting information." *Engineering Letters*, vol. 25, no. 4, pp. 1–7, Nov. 2017.

[77] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Robotics Research*, 2017, pp. 235–252.

[78] C. Teulière, L. Eck, E. Marchand, and N. Guenard, "3d model-based tracking for uav position control," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, Oct. 2010, pp. 1084–1089.

[79] J. L. Casper and R. R. Murphy, "Workflow study on human-robot interaction in usar," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, Washington, DC, May 2002, pp. 1997–2003.

[80] H. L. Jones, S. M. Rock, D. Burns, and S. Morris, "Autonomous robots in swat applications: Research, design, and operations challenges," *AUVSI'02*, pp. 1–15, July 2002.

[81] M. Li, K. Lu, H. Zhu, M. Chen, S. Mao, and B. Prabhakaran, "Robot swarm communication networks: architectures, protocols, and applications," in *2008 Third International Conference on Communications and Networking in China*, Hangzhou, P.R. China, Aug. 2008, pp. 162–166.

[82] M. Feng, S. Mao, and T. Jiang, "Joint frame design, resource allocation and user association for massive mimo heterogeneous networks with wireless backhaul," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1937–1950, Mar. 2018.

[83] M. Li, J. Harris, M. Chen, S. Mao, Y. Xiao, W. Read, and B. Prabhakaran, "Architecture and protocol design for a pervasive robot swarm communication networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 8, pp. 1092–1106, Aug. 2011.

[84] C. Bartneck and J. Forlizzi, "A design-centred framework for social human-robot interaction," in *RO-MAN 2004. 13th IEEE international workshop on robot and human interactive communication (IEEE Catalog No. 04TH8759)*, Kurashiki, Japan, Sept. 2004, pp. 591–594.

[85] O. Ogorodnikova, "Safe and reliable human-robot interaction in manufactory, within and beyond the workcell," in *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*, Budapest, Hungary, June 2010, pp. 65–70.

[86] V. Alvarez-Santos, R. Iglesias, X. M. Pardo, C. V. Regueiro, and A. Canedo-Rodriguez, "Gesture-based interaction with voice feedback for a tour-guide robot," *Journal of Visual Communication and Image Representation*, vol. 25, no. 2, pp. 499–509, Feb. 2014.

[87] S. Kohlbrecher, A. Romay, A. Stumpf, A. Gupta, O. Von Stryk, F. Bacim, D. A. Bowman, A. Goins, R. Balasubramanian, and D. C. Conner, "Human-robot teaming for rescue missions: Team vigir's approach to the 2013 darpa robotics challenge trials," *Journal of Field Robotics*, vol. 32, no. 3, pp. 352–377, May 2015.

[88] F. Mueller and M. Muirhead, "Jogging with a quadcopter," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, Seoul, Republic of Korea, Apr. 2015, pp. 2023–2032.

[89] J. Scheible, A. Hoth, J. Saal, and H. Su, "Displaydrone: a flying robot based interactive display," in *Proceedings of the 2nd ACM International Symposium on Pervasive Displays*, Mountain View, CA, June 2013, pp. 49–54.

[90] J. R. Cauchard, J. L. E, K. Y. Zhai, and J. A. Landay, "Drone & me: an exploration into natural human-drone interaction," in *Proceedings of the 2015 ACM international joint conference on pervasive and ubiquitous computing*, Osaka, Japan, Sept. 2015, pp. 361–365.

[91] L. Jing and P. Yang, "A localization algorithm for mobile robots in rfid system," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, Wuhan, China, Sept. 2007, pp. 2109–2112.

[92] P. Yang, W. Wu, M. Moniri, and C. C. Chibelushi, "Slam algorithm for 2d object trajectory tracking based on rfid passive tags," in *2008 IEEE International Conference on RFID*, Las Vegas, NV, Apr. 2008, pp. 165–172.

[93] A. Bekkali, H. Sanson, and M. Matsumoto, "Rfid indoor positioning based on probabilistic rfid map and kalman filtering," in *Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2007)*, White Plains, NY, Oct. 2007, pp. 21–21.

[94] W. Wang, X. Wang, and S. Mao, "Deep convolutional neural networks for indoor localization with CSI images," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 316–327, Jan./Mar. 2020.

[95] X. Wang, X. Wang, and S. Mao, "Cifi: Deep convolutional neural networks for indoor localization with 5 ghz wi-fi," in *Proc. IEEE ICC 2017*, Paris, France, May 2017, pp. 1–6.

[96] P. V. Nikitin, R. Martinez, S. Ramamurthy, H. Leland, G. Spiess, and K. Rao, "Phase based spatial identification of uhf rfid tags," in *2010 IEEE International Conference on RFID (IEEE RFID 2010)*, Orlando, FL, Apr. 2010, pp. 102–109.

[97] T.-M. Choi, "Coordination and risk analysis of vmi supply chains with rfid technology," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 3, pp. 497–504, Aug. 2011.

[98] C. Yang, X. Wang, and S. Mao, "Sparsetag: High-precision backscatter indoor localization with sparse rfid tag arrays," in *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, Boston, MA, USA, June 2019, pp. 1–9.

[99] C. Yang, S. Mao, and X. Wang, "An overview of 3GPP positioning standards," *ACM GetMobile*, vol. 26, no. 1, pp. 9–13, Mar. 2022.

[100] X. Wang, L. Gao, and S. Mao, "Csi phase fingerprinting for indoor localization with a deep learning approach," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1113–1123, Dec. 2016.

[101] ——, "Phasefi: Phase fingerprinting for indoor localization with a deep learning approach," in *2015 IEEE Global Communications Conference (GLOBECOM)*, San Diego, CA, Dec. 2015, pp. 1–6.

[102] X. Wang, X. Wang, and S. Mao, "Indoor fingerprinting with bimodal CSI tensors: A deep residual sharing learning approach," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4498–4513, Mar. 2021.

[103] X. Wang, L. Gao, and S. Mao, "Biloc: Bi-modal deep learning for indoor localization with commodity 5ghz wifi," *IEEE access*, vol. 5, no. 1, pp. 4209–4220, Mar. 2017.

[104] X. Wang, R. Huang, and S. Mao, "Sonarbeat: Sonar phase for breathing beat monitoring with smartphones," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, Vancouver, Canada, July/Aug. 2017, pp. 1–8.

[105] X. Wang, C. Yang, and S. Mao, "Phasebeat: Exploiting csi phase data for vital sign monitoring with commodity wifi devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Atlanta, GA, June 2017, pp. 1230–1239.

[106] C. Yang, X. Wang, and S. Mao, "Respiration monitoring with RFID in driving environments," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 2, pp. 500–512, Feb. 2021.

[107] ——, "Subject-adaptive skeleton tracking with RFID," in *Proc. The 16th IEEE International Conference on Mobility, Sensing and Networking (MSN 2020)*, Tokyo, Japan, Dec. 2020.

[108] ——, "RFID-Pose: Vision-aided 3D human pose estimation with RFID," *IEEE Transactions on Reliability*, vol. 70, no. 3, pp. 1218–1231, Sept. 2021.

[109] Z. Zhou, L. Shangguan, X. Zheng, L. Yang, and Y. Liu, "Design and implementation of an rfid-based customer shopping behavior mining system," *IEEE/ACM transactions on networking*, vol. 25, no. 4, pp. 2405–2418, Aug. 2017.

[110] C. Yang, X. Wang, and S. Mao, "Autotag: Recurrent variational autoencoder for unsupervised apnea detection with rfid tags," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–7.

[111] X. Wang, J. Zhang, Z. Yu, S. Mao, S. Periaswamy, and J. Patton, "On remote temperature sensing using commercial UHF RFID tags," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 715–10 727, Dec. 2019.

[112] A. Koubâa, M.-F. Sriti, H. Bennaceur, A. Ammar, Y. Javed, M. Alajlan, N. Al-Elaiwi, M. Tounsi, and E. Shakshuki, "Coros: a multi-agent software architecture for cooperative and autonomous service robots," in *Cooperative Robots and Sensor Networks 2015*. Springer, 2015, pp. 3–30.

[113] M. Mendonça, I. R. Chrun, F. Neves Jr, and L. V. Arruda, "A cooperative architecture for swarm robotic based on dynamic fuzzy cognitive maps," *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 122–132, Jan. 2017.

[114] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, "A distributed task allocation algorithm for a multi-robot system in healthcare facilities," *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 33–58, Nov 2014.

[115] M. Erdelj and E. Natalizio, "Uav-assisted disaster management: Applications and open issues," in *2016 international conference on computing, networking and communications (ICNC)*, Kauai, HI, USA, Feb. 2016, pp. 1–5.

[116] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995, pp. 103–129.

[117] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, Aug. 2018.

[118] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *arXiv preprint arXiv:1806.06877*, Aug. 2019.

[119] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, Barcelona, SPAIN, Dec. 2016, pp. 4565–4573.

[120] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, Apr. 2016.

[121] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, Seoul,Korea, Oct. 2019, pp. 9329–9338.

[122] Y. Sun, M. Peng, Y. Zhou, Y. Huang, and S. Mao, "Application of machine learning in wireless networks: Key techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3072–3108, Fourth Quarter 2019.

[123] Y. Li, H. Ma, L. Wang, and S. Mao, "Optimized content caching and user association for edge computing in densely deployed heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 6, pp. 2130–2142, June 2022.

[124] M. Chen, V. C. M. Leung, S. Mao, and M. Li, "Energy-efficient itinerary planning for mobile agents in wireless sensor networks," in *Proc. IEEE ICC 2009*, Dresden, Germany, June 2009, pp. 1–5.

[125] X. Wang, Z. Yu, and S. Mao, "Indoor localization using magnetic and light sensors with smartphones: A deep lstm approach," *Springer Mobile Networks and Applications (MONET) Journal*, vol. 25, no. 2, pp. 819–832, Apr. 2020.

[126] L. Wang, S. Mao, B. M. Wilamowski, and R. Nelms, "Ensemble learning for load forecasting," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 616–628, Apr. 2020.

[127] Y. Wang, Y. Shen, S. Mao, X. Chen, and H. Zou, "Lasso and lstm integrated temporal model for short-term solar intensity forecasting," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2933–2944, Apr. 2018.

[128] A. Y. Ng, S. J. Russell *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, Stanford,CA,USA, June 2000, pp. 663–670.

[129] S. Sariel-Talay, T. R. Balch, and N. Erdogan, "A generic framework for distributed multi-robot cooperation," *Journal of Intelligent & Robotic Systems*, vol. 63, no. 2, pp. 323–358, May 2011.

[130] J. Wang, Y. Gu, and X. Li, "Multi-robot task allocation based on ant colony algorithm," *Journal of Computers*, vol. 7, no. 9, pp. 2160–2167, Sept. 2012.

[131] K. A. Ghamry, Y. Dong, M. A. Kamel, and Y. Zhang, "Real-time autonomous take-off, tracking and landing of uav on a moving ugv platform," in *2016 24th Mediterranean conference on control and automation (MED)*, Athens, Greece, June 2016, pp. 1236–1241.

[132] K. A. Ghamry, M. A. Kamel, and Y. Zhang, "Cooperative forest monitoring and fire detection using a team of uavs-ugvs," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, Arlington, VA, USA, June 2016, pp. 1206–1211.

[133] A. M. Khaleghi, D. Xu, S. Minaeian, M. Li, Y. Yuan, J. Liu, Y.-J. Son, C. Vo, and J.-M. Lien, "A dddams-based uav and ugv team formation approach for surveillance and

crowd control," in *Proceedings of the Winter Simulation Conference 2014*, Savanah, GA, USA, 2014, pp. 2907–2918.

[134] C. E. Pippin and H. Christensen, "A bayesian formulation for auction-based task allocation in heterogeneous multi-agent teams," in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR II*, vol. 8047, Orlando,Florida,USA, May 2011, p. 804710.

[135] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 289–353, May 2008.

[136] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," *arXiv preprint arXiv:1803.11485*, Mar. 2018.

[137] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, Richland, SC, July 2018, p. 2085–2087.

[138] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, Montreal, Canada, Dec. 2014, pp. 2672–2680.

[139] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, Lille, France, July 2015, pp. 1889–1897.

[140] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, July 2017.

[141] M. Bloem and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," in *53rd IEEE Conference on Decision and Control*, Los Angeles, CA, USA, Feb. 2015, pp. 4911–4916.

[142] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *CoRR*, vol. abs/1809.02627, Sept. 2018.

[143] C. J. Watkins and P. Dayan, "Q-learning," *Springer Machine Learning J.*, vol. 8, no. 3-4, pp. 279–292, May 1992.

[144] J. Kober, E. Oztop, and J. Peters, "Reinforcement learning to adjust robot movements to new situations," in *Robotics: Science and Systems*, Y. Matsuoka, H. Durrant-Whyte, and J. Neira, Eds. Cambridge, MA: The MIT Press, 2011, pp. 33–40.

[145] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE IROS'17*, Vancouver, Canada, Sept. 2017, pp. 31–36.

[146] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Proc. 3rd Conf. Robot Learning*, Osaka, Japan, Nov. 2020, pp. 1–10.

[147] X. Xia, T. Roppel, J. Y. Hung, J. Zhang, S. C. Periaswamy, and J. Patton, "Balanced map coverage using reinforcement learning in repeated obstacle environments," in *Proc. IEEE ISIE'20*, Delft, The Netherlands, June 2020, pp. 41–48.

[148] M. Feng and S. Mao, "Dealing with limited backhaul capacity in millimeter wave systems: A deep reinforcement learning approach," *IEEE Communications*, vol. 57, no. 3, pp. 50–55, Mar. 2019.

[149] J. Xie, Z. Shao, Y. Li, Y. Guan, and J. Tan, "Deep reinforcement learning with optimized reward functions for robotic trajectory planning," *IEEE Access*, vol. 7, pp. 105 669–105 679, July 2019.

[150] S. Cabi *et al.*, "Scaling data-driven robotics with reward sketching and batch reinforcement learning," *arXiv preprint arXiv:1909.12200*, Sept. 2019. [Online]. Available: https://arxiv.org/abs/1909.12200

[151] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," *arXiv preprint arXiv:1703.07326*, Mar. 2017. [Online]. Available: https://arxiv.org/abs/1703.07326

[152] X. Xia, T. Roppel, J. Y. Hung, J. Zhang, S. C. Periaswamy, and J. Patton, "Environmental complexity measurement using shannon entropy," Raleigh, NC, USA, Mar. 2020, pp. 1–6.

[153] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proc. CORL'19*, Osaka, Japan, Oct. 2020, pp. 1025–1037.

[154] S. Pitis, H. Chan, S. Zhao, B. Stadie, and J. Ba, "Maximum entropy gain exploration for long horizon multi-goal reinforcement learning," in *Proc. PMLR ICML'20*, Virtual Conference, July 2020, pp. 7750–7761.

[155] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE ICRL'18*, Vancouver, Canada, Apr.-May 2018, pp. 6292–6299.

[156] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Elsevier Neural Networks*, vol. 21, no. 4, pp. 682–697, May 2008.

[157] J. Zhang, Z. Yu, S. Mao, S. Periaswamy, J. Patton, and X. Xia, "IADRL: Imitation augmented deep reinforcement learning enabled UGV-UAV coalition for tasking in complex environments," *IEEE Access*, vol. 8, no. 1, pp. 102 335–102 347, June 2020.

[158] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[159] C. Yang, X. Wang, and S. Mao, "AutoTag: Recurrent vibrational autoencoder for unsupervised apnea detection with RFID tags," in *Proc. IEEE GLOBECOM 2018*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–7.

[160] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Mmagazine*, vol. 13, no. 2, pp. 99–110, June 2006.

[161] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Robotics: Science and Systems*, S. Thrun, G. S. Sukhatme, and S. Schaal, Eds. Cambridge, MA: The MIT Press, 2005, pp. 65–72.

[162] I. Maurović, M. Seder, K. Lenac, and I. Petrović, "Path planning for active slam based on the d* algorithm with negative edge weights," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 8, pp. 1321–1331, Aug. 2017.

[163] K. Xiao, S. Mao, and J. Tugnait, "TCP-Drinc: Smart congestion control based on deep reinforcement learning," *IEEE Access Journal*, vol. 7, no. 1, pp. 11 892–11 904, Jan. 2019.

[164] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, Oct. 2016.

[165] T. M. Le, V. Le, S. Venkatesh, and T. Tran, "Dynamic language binding in relational visual reasoning," *arXiv preprint arXiv:2004.14603*, Apr. 2020. [Online]. Available: https://arxiv.org/abs/2004.14603

[166] P. Hohenecker and T. Lukasiewicz, "Deep learning for ontology reasoning," *arXiv preprint arXiv:1705.10342*, May 2017. [Online]. Available: https://arxiv.org/abs/1705.10342

[167] D. Hudson and C. D. Manning, "Learning by abstraction: The neural state machine," *arXiv preprint arXiv:1907.03950v4*, Nov. 2019. [Online]. Available: https://arxiv.org/abs/1907.03950

[168] S. Aditya, Y. Yang, and C. Baral, "Integrating knowledge and reasoning in image understanding," *arXiv preprint arXiv:1906.09954*, June 2019. [Online]. Available: https://arxiv.org/abs/1906.09954

[169] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko, "Learning to reason: End-to-end module networks for visual question answering," in *Proc. IEEE ICCV'17*, Venice, Italy, Oct. 2017, pp. 804–813.

[170] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS'17*, Long Beach, CA, June 2017, pp. 1–11.

[171] H. Le, T. Tran, and S. Venkatesh, "Self-attentive associative memory," in *Proc. ICML'20*, Vienna, Austria, July 2020, pp. 5682–5691.

[172] T. H. Le, T. Tran, and S. Venkatesh, "Neural stored-program memory," in *Proc. ICLR'20*, Virtual Conference, Apr. 2020, pp. 1–28.

[173] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, Oct. 2016.

[174] S. Hu, Y. Ma, X. Liu, Y. Wei, and S. Bai, "Stratified rule-aware network for abstract visual reasoning," *arXiv preprint arXiv:2002.06838*, Dec. 2020. [Online]. Available: https://arxiv.org/abs/2002.06838

[175] S. Y. Gadre, K. Ehsani, S. Song, and R. Mottaghi, "Continuous scene representations for embodied AI," *arXiv preprint arXiv:2203.17251*, Mar. 2022. [Online]. Available: https://arxiv.org/abs/2203.17251

[176] P. Tiwari, H. Zhu, and H. M. Pandey, "DAPath: Distance-aware knowledge graph reasoning based on deep reinforcement learning," *Elsevier Neural Networks*, vol. 135, pp. 1–12, Mar. 2021.

[177] M. Clark-Turner and M. Begum, "Deep reinforcement learning of abstract reasoning from demonstrations," in *Proc. ACM/IEEE HRI'18*, Chicago, IL, Mar. 2018, pp. 160–168.

[178] Y. Gao, F. Yang, M. Frisk, D. Hemandez, C. Peters, and G. Castellano, "Learning socially appropriate robot approaching behavior toward groups using deep reinforcement learning," in *Proc. 28th IEEE Int. Conf. Robot Human Interactive Commun.*, New Delhi, India, Oct. 2019, pp. 1–8.

[179] Q. Wang, Y. Hao, and J. Cao, "ADRL: An attention-based deep reinforcement learning framework for knowledge graph reasoning," *Elsevier Knowledge-Based Systems*, vol. 197, p. 105910, June 2020.

[180] Z. Yu, J. Zhang, S. Mao, S. C. Periaswamy, and J. Patton, "RIRL: A recurrent imitation and reinforcement learning method for long-horizon robotic tasks," in *Proc. IEEE CCNC'22*, Virtual Conference, Jan. 2022, pp. 230–235.