Multi-State-Space Reasoning Reinforcement Learning for Long-Horizon RFID-Based Robotic Searching and Planning Tasks

Zhitao Yu, Jian Zhang, Shiwen Mao, Senthilkumar C G Periaswamy, Justin Patton

***Abstract*—In recent years, reinforcement learning (RL) has shown high potential for robotic applications. However, RL heavily relies on the reward function, and the agent merely follows the policy to maximize rewards but lacks reasoning ability. As a result, RL may not be suitable for long-horizon robotic tasks. In this paper, 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 a long short-term memory (LSTM) network to preserve long-term memories and dependencies. The proposed SRRL'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 radio frequency identification (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.**

***Keywords*—reinforcement learning, multiple state spaces, abstract reasoning, long-horizon robotic task**

Manuscript received Jun. 02, 2022; revised Jul. 10, 2022; accepted Aug. 13, 2022. This work is supported in part by the NSF under Grants ECCS-1923163 and CNS-2107190, and through the RFID Lab and the Wireless Engineering Research and Education Center at Auburn University, Auburn, AL, USA. The associate editor coordinating the review of this paper and approving it for publication was L. Bai.

Z. T. Yu, S. W. Mao. Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201, USA (email: zzy0021@auburn.edu; smao@ieee.org).

J. Zhang. Department of Electrical and Computer Engineering, Kennesaw State University, Kennesaw, GA 30144, USA (email: jianzhang@ieee.org).

Z. T. Yu, S. C. G. Periaswamy, J. Patton. RFID Lab at Auburn University, Auburn, AL 36849, USA (email: zzy0021@auburn.edu; szc0089@auburn.edu; jbp0033@auburn.edu).

I. 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 number of acting steps and subtasks increases, 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[1,2]. In Ref. [3], 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 Ref. [4], is a method commonly used in numerous map-based path planning algorithms[5,6]. SLAM allows the robot to start from an unknown position in an unknown environment, determine its 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 Ref. [6] 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 locations that may arise while the robot is moving. However, once the environment has been altered significantly, map-based algorithms usually require rebuilding 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[7]. 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 Ref. [8]. Kim et al. in Ref. [9] described a framework for socially adaptable path planning in dynamic environments that include a feature extraction module, inverse reinforcement learning, and a 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.[10] 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 Ref. [11] included examples in the RL approach that is based on deep deterministic policy gradients (DDPG) in order to overcome the sparse extrinsic reward problem of long-horizon tasks. 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 fewer 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 infers policy from multiple state spaces. Basically, we consider the robotic applications where a robot is used to scan the radio frequency identification (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 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.

The remainder of this paper is organized as follows. In section II, we introduce the related works. We then present the preliminaries and motivation in section III. The SRRL system design is described in section IV. Our experimental study is presented in section V. Section VI concludes this paper.

II. 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 forwarding 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.

A. 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[12-15], hence pushing deep learning systems towards the thinking mode. In Ref. [16], 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[17], causal interplay estimation, and composition.

Moreover, some researchers have shown that the reasoning process is intricately tied to efforts on neural memories[18,19], which is an intellectual capacity for memorizing, recovering, altering information, and simulating unobserved situations. Grave et al.[20], 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 Ref. [19] utilized a unique memory to store controller weights, similar to the stored-program memory in contemporary computer architectures, where subprograms 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 the data pattern and query is crucial to its success. Therefore, reasoning tends to be particular to data patterns, resulting in inadequate systematic generalization ability.

B. 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 Ref. [21], 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 Ref. [22] 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 learned 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 the popularity of DRL in recent years, the combination of reasoning and DRL has also produced several innovative works[23-25]. An end-to-end DRL framework that combines the feature abstraction ability and Q-learning was presented in Ref. [24] 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 staged social behavior learning (SBBL) in Ref. [25], 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 Ref. [26] built a relational mod-

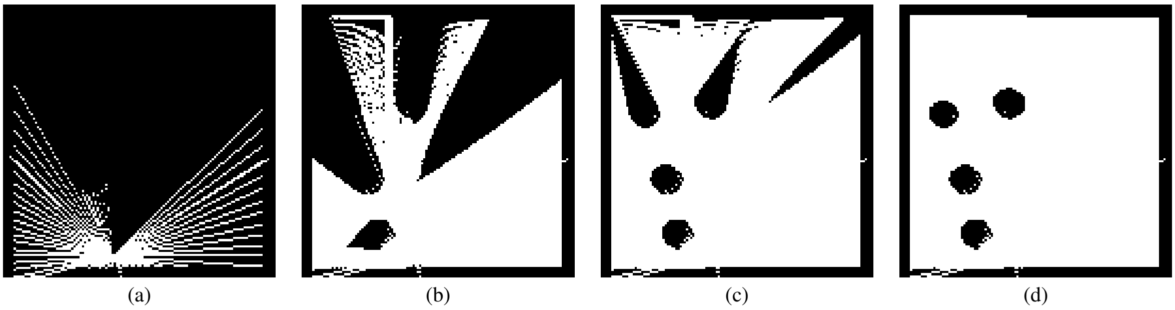

Fig. 1 The environment occupation map after different steps: (a) Step 0; (b) Step 500; (c) Step 2 000; (d) Step 4 000

ule 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.

III. 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 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.

A. Multi-State-Spaces Feature Extraction and Reasoning

1) Environment Occupation State Space $\boldsymbol{S_L}$: 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. 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. 1 describes the environment that the agent partially observed the moment, when 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 $\boldsymbol{S_L}$.

2) RFID Sensing State Space $\boldsymbol{S_R}$: 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 Ref. [2]. 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. 2 illustrates the expansion of the RFID radio map while the agent explores the unknown space. As shown in Fig. 2(a), 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 $\boldsymbol{S_L}$, the RFID sensing state space $\boldsymbol{S_R}$ is defined by a series of RFID radio maps of the same size.

3) 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

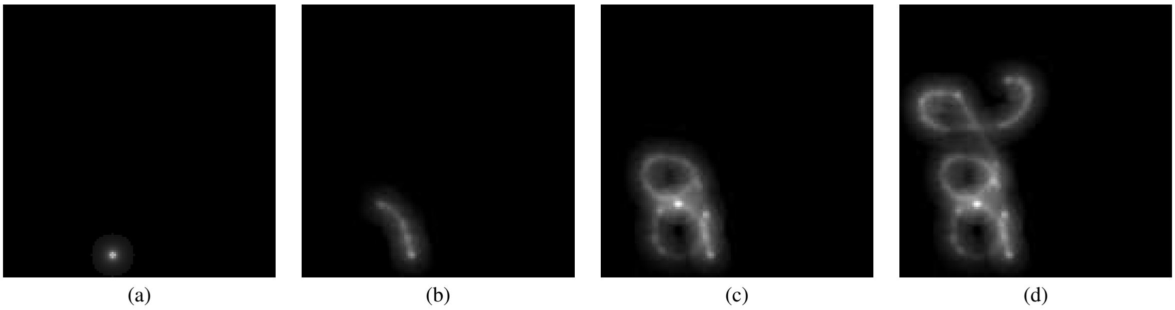

Fig. 2 The RFID sensing radio map after different steps: (a) Step 0; (b) Step 500; (c) Step 2 000; (d) Step 4 000

be the nth feature decoding in layer i, defined as

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

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 long short-term memory (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 $\mathscr{R}_t(\mu_{s_L}, \mu_{s_R}; \phi)$, 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 $\boldsymbol{S}_L$ and $\boldsymbol{S}_R$, respectively.

B. Reinforcement Learning

Motivated by the partially observable Markov decision process (POMDP), this problem could be described by $(\varepsilon, \boldsymbol{S}, A, T, R)$, where ε represents the environment in the agent interacts with; $\boldsymbol{S} = (\boldsymbol{S}_L, \boldsymbol{S}_R)$ is the joint partially observed state space that consists of two state spaces with different dimensions and meanings: the real-time obstacle map $\boldsymbol{S}_M$ created by Lidar sensors, and the RFID sensing signal state space $\boldsymbol{S}_R$; $\boldsymbol{a} = (a_1, a_2, \cdots, a_t)$ 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 $\pi(a_t|\mathscr{R}_t)$ that maximizes the predicted future discounted reward $E_\pi[\sum_{t=0}^{T} \gamma^t r_t]$, where $0 \leqslant \gamma < 1$ is a discount factor. This reward is formulated as the value-action function: $Q_\pi(\mathscr{R}_t, a_t; \theta, \phi)$. 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 are defined as

$$\underset{\boldsymbol{a}\sim\pi}{\operatorname{argmax}}\ Q_\pi(\mathscr{R}_t, a_t; \theta, \phi), \qquad (2)$$

where θ and ϕ serve as the parameter sets of the value function Q_π and the reasoning module, respectively. Note that $\mathscr{R}_t$ is the output of the reasoning module described above. $\mathscr{R}_t$ extracts the latent relations from multiple state spaces; it also provides an abstract presentation of observations, which are sampled from multiple state spaces. Thus, $\mathscr{R}_t$ 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.

IV. Overview of the Proposed System

Our proposed model can infer the optimal policy for long-horizon robotic searching and planning tasks from the

[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.

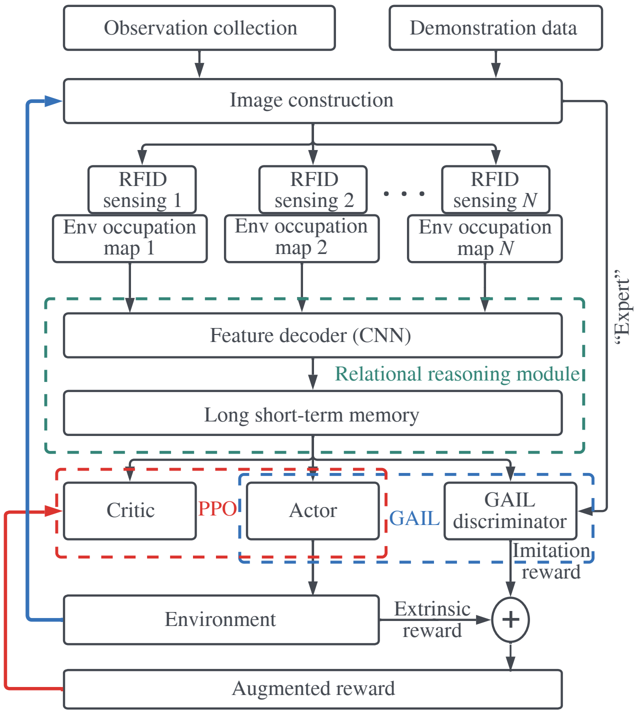

Fig. 3 The architecture of the proposed method

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. 3, the model consists of three parts: (i) the relational reasoning module, (ii) the imitation learning (IL) module, and (iii) the DRL module[27]. Additionally, all the networks are reinforced with an LSTM layer to retain historical data in a recurrent manner. In Fig. 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)[28], 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)[29] and is augmented with an LSTM layer that enables it to deal with both historical and present data.

A. The Multi-state-space Fusion and Reasoning Module

The reasoning module, discussed in section III.A, including the construction of multiple state spaces and feature decoding and reasoning, is the key to exploring 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.

B. The Recurrent IL Module

The proposed IL module incorporates the GAIL framework[28], which employs a network design derived from the generative adversarial network (GAN)[30]. It comes with two fundamental components, a discriminator $\boldsymbol{D}$ and a generator $\boldsymbol{G}$, which learns a strategy from a set of demonstration data in an adversarial and cooperative way, respectively. The discriminator $\boldsymbol{D}$ is capable of identifying $\boldsymbol{G}$'s data from demonstration data, while both $\boldsymbol{D}$ and $\boldsymbol{G}$ are concurrently taught in a competitive and adversarial manner. During the training stage, discriminator $\boldsymbol{D}$ will become more adept at identifying created data, while generator $\boldsymbol{G}$ will be much more proficient at forging data. The IL module eventually converges when the generated "fake" data from $\boldsymbol{G}$ can masquerade as demonstration data and survive $\boldsymbol{D}$'s verification. Generator $\boldsymbol{G}$ 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 $\boldsymbol{G}$ 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 IV.C when we present the DRL network. In this section, we just look at the imitation network's discriminator $\boldsymbol{D}(a_t, \mathscr{R}_t; \boldsymbol{\omega}, \phi)$.

As previously stated, we employ an LSTM layer to augment both discriminator $\boldsymbol{D}$ and generator $\boldsymbol{G}$ in order to supply recurrent capabilities to the IL module. The recurrence-enabled discriminator $\boldsymbol{D}(a_t, \mathscr{R}_t; \boldsymbol{\omega}, \phi)$ assesses data based on real-time and historical data to improve the process of forecasting the next action. $\boldsymbol{D}: \mathscr{R} \times \boldsymbol{a} \mapsto (0,1)$ is a function with weights, where $\mathscr{R}$ and $\boldsymbol{a}$ represent the combination of the extracted correlations of multiple state spaces and action spaces, accordingly. Fig. 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 $\boldsymbol{D}$ can be enhanced by optimizing the following value function.

$$\begin{aligned} V(\boldsymbol{\omega}) = \mathbb{E}_{\pi}[\log(1 - \boldsymbol{D}(a_t, \mathscr{R}_t; \boldsymbol{\omega}, \phi))] + \\ \mathbb{E}_{\pi_E}[\log(\boldsymbol{D}(a_t, \mathscr{R}_t; \boldsymbol{\omega}, \phi))] - \lambda H(\pi), \end{aligned} \tag{3}$$

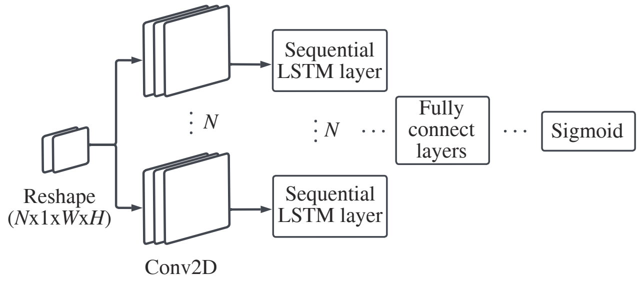

Fig. 4 The architecture of the discriminator

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 (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 $H(\pi) \equiv \mathbb{E}_\pi[-\log \pi(a_t|\mathscr{R}_t)]$ 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. $V(\omega)$ 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 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.

C. 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[29] 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 $\mathscr{R}_t$. 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 evaluating.

Ultimately, the goal of training the DRL network is to maximize the value function Q_π defined in (2) for a given policy π, as

$$Q_\pi(\mathscr{R}_t, a_t; \theta, \phi) = \mathbb{E}[r_{\text{en}}(\mathscr{R}_t, a_t) + \gamma \mathbb{E}_{a_{t+1}\sim\pi}[Q_\pi(\mathscr{R}_{t+1}, a_{t+1})]], \tag{4}$$

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

$$r_{\text{en}}(o_t, a_t) = \alpha r_{\text{im}}(\mathscr{R}_t, a_t) + (1-\alpha) r_{\text{ex}}(\mathscr{R}_t, a_t), \tag{5}$$

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 $V(\omega)$ in (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

$$\mathscr{L}(\theta, \phi) = \mathscr{L}^{\text{Actor}} - c_1 \mathscr{L}^{\text{Critic}} + c_2 \mathscr{H}, \tag{6}$$

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.

$$\mathscr{J}(\varphi) = -(\mathbb{E}_t\left[\min\left(f_t(\varphi)\bar{A}_t, \text{Clip}\left(f_t(\varphi), 1-\varepsilon, 1+\varepsilon\right)\bar{A}_t\right)\right] + c_1 \mathbb{E}_t\left[(\mathscr{V}_t^{\pi_\phi} - Q_\pi(\mathscr{R}, a_t; \varphi))^2\right] - c_2 \mathscr{H}), \tag{7}$$

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

$$f_t(\varphi) = \frac{\pi_\varphi(\mathscr{R})}{\pi_{\varphi_{\text{old}}}(\mathscr{R})}, \tag{8}$$

where $\pi_{\varphi_{\text{old}}}$ 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

$$\hat{A}_t = \delta_t + \gamma\lambda \hat{A}_{t+1}, \tag{9}$$

$$\delta_t = r_t + \gamma Q\left(\mathscr{R}_{t+1}\right) - Q\left(\mathscr{R}_t\right), \tag{10}$$

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 (7), the training process seeks to maximize $\mathscr{J}(\varphi)$ 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 (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.

V. Experiment Study

A. 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 Ref. [1]. 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 the study of various intelligent agents[31]. As depicted in Fig. 5, we construct two environments of a four-walled area of different complexity to imitate a 50 × 50 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 $\Delta_{\max}$ denotes the maximum rotational velocity.

The agent's mission is to scan all the RFID tags in 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.

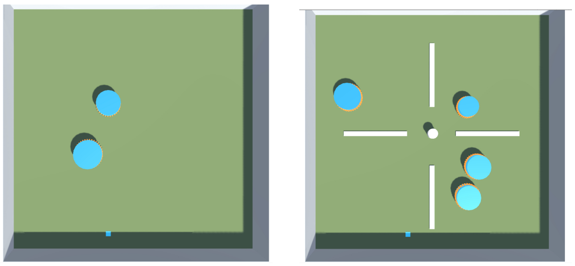

Fig. 5 Basic experimental setup for agent performing long-horizon RFID inventory tasks. The agent is represented as a blue cube, and the tags are orange strings attached to the blue cylinder-shaped racks: (a) Simple inventory environment; (b) Complex inventory environment

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 $\boldsymbol{D}$ 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 Tab. 1. For the remaining experiments, we set the proportion parameter μ in (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.

B. Results and Analysis

1) Training Results: In the training process, we train the agent in the simple environment with two racks, as shown in Fig. 5(a), 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. 5(b) 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 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

Tab. 1 Basic training configuration

RL Parameter	Value
Learning rate	0.000 2
Gamma (discount factor)	0.99
Hidden layer units	256
Sequence length	64
Batch size	1024
Memory size	256
Max steps	5 000 000
CNN Parameter	**Value**
Convolutional layer num	3
Stride	1
Kernel size	[4×4];[3×3];[3×3]
FC layer num	2
Hidden units	128

2×10^4. Each training episode also ends immediately if the agent scans all 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 Ref. [28], (ii) the PPO network proposed in Ref. [29], and (iii) the recurrent imitation and reinforcement learning (RIRL) model proposed in our recent work[32], 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 four approaches.

Cumulative rewards for each training episode are depicted in Fig. 6 as the agent interacts with the environment. We basically specified a few simple and sparse reward configurations: reading a new RFID tag earns $+0.01$ points, colliding is punished by -0.1 points, moving costs $-0.000\,1$ points for each step, and completing the task gets $+1$ point. The SRRL, represented by the red solid line in Fig. 6, achieves the best reward results after about 60 training episodes, 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 is 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.

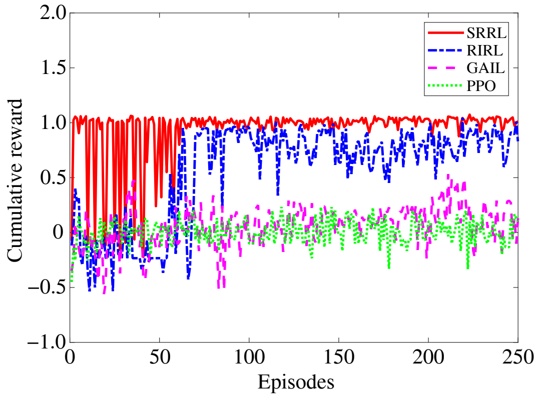

Fig. 6 Accumulated training reward values for SRRL, RIRL, PPO, and GAIL methods

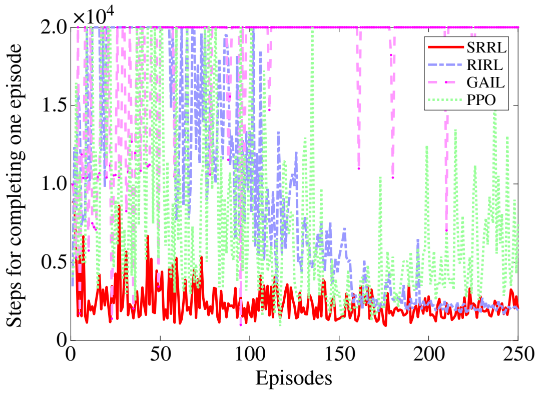

Fig. 7 Steps for finishing the tag scanning task within one episode during the training phase

Additionally, as described in Fig. 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 number of steps. To further compare the models' performance during the training process, we plot the training loss values for each method in Fig. 8. Obviously, the loss values of SRRL and RIRL are convergent, whereas the loss curves for PPO and GAIL do not. This is because PPO and GAIL are incapable of finding a reliable

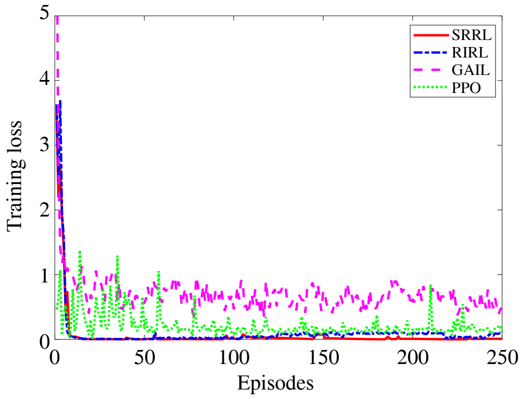

Fig. 8 Training loss of SRRL, RIRL, GAIL, and PPO

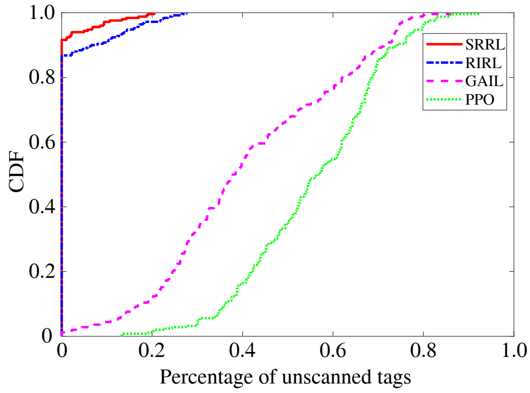

Fig. 9 CDF of the percentage of unscanned tags in total in the testing stage

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 h to make the agent to 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 1ms 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 is very large, it will take about 10 ms longer to make an action with our method, and about 1.5 s for the other methods to choose the next action.

2) RFID Tag Scanning Results: The cumulative distribution function (CDF) of the percentage of unscanned tags in the testing stage is shown in Fig. 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. 5(a), and then deploy the well-trained model to both simple and complex environments shown in Fig. 5 with a varied number of racks in one hundred testing episodes.

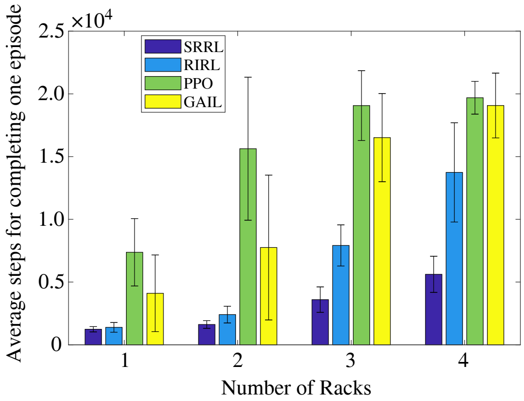

Fig. 10 Average number steps to complete the task in 100 episodes in the testing stage in the simple environment

Fig. 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. 11, when there are more racks in the simple environment, the average collision times rise as the number of racks is increasing. 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

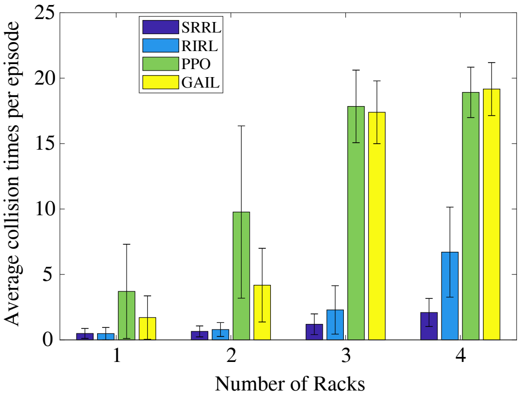

Fig. 11 Average number of collisions of the agent per episode in the simple environment

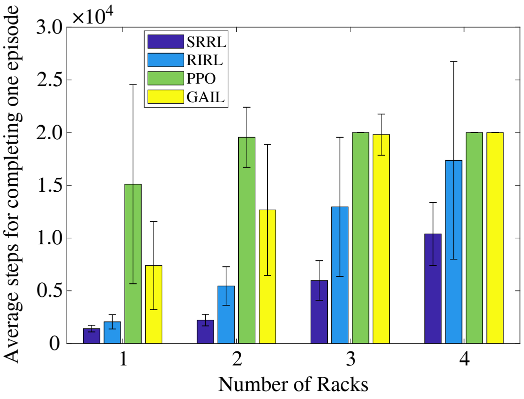

Fig. 12 Average number of the task completion steps in 100 episodes of the testing stage in the complex environment

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. 10 and Fig. 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 completion number of steps and collision times per testing episode for the complex environment are presented in Fig. 12 and Fig. 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 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.

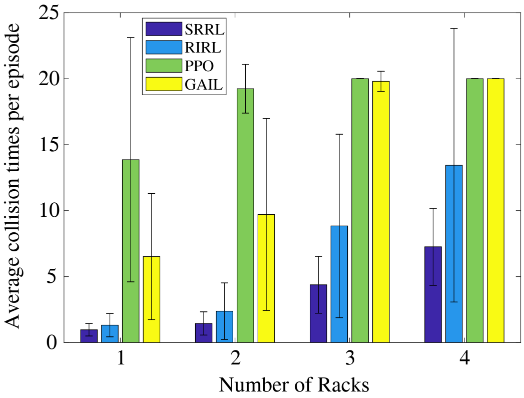

Fig. 13 Average collision times that happened in agent per episode in the complex environment

VI. 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, and 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 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.

References

[1] ZHANG J, LYU Y, ROPPEL T, et al. Mobile robot for retail inventory using RFID[C]//Proceedings of IEEE ICIT'16. Piscataway: IEEE Press, 2016: 101-106.

[2] ZHANG J, LYU Y, PATTON J, et al. BFVP: a probabilistic UHF RFID tag localization algorithm using Bayesian filter and a variable power RFID model[J]. IEEE Transactions on Industrial Electronics, 2018, 65(10): 8250-8259.

[3] GUPTA A, KUMAR V, LYNCH C, et al. Relay policy learning: solving long-horizon tasks via imitation and reinforcement learning[C]//Proceedings of CORL'19. [S.l.:s.n.], 2020: 1025-1037.

[4] DURRANT-WHYTE H, BAILEY T. Simultaneous localization and mapping: part I[J]. IEEE Robotics and Automation Mmagazine, 2006, 13(2): 99-110.

[5] STACHNISS C, GRISETTI G, BURGARD W. Information gain-based exploration using Rao-Blackwellized particle filters. Robotics: Science and Systems, S. Thrun, G. S. Sukhatme, and S. Schaal, Eds[R]. Cambridge: The MIT Press, 2005.

[6] MAUROVIĆ I, SEDER M, LENAC K, et al. Path planning for active slam based on the d* algorithm with negative edge weights[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017, 48(8): 1321-1331.

[7] WATKINS C J, DAYAN P. Q-learning[J]. Springer Machine Learning Journal, 1992, 8(3-4): 279-292.

[8] KOBER J, OZTOP E, PETERS J. Reinforcement learning to adjust robot movements to new situations[R]. Cambridge: The MIT Press, 2011.

[9] KIM B, PINEAU J. Socially adaptive path planning in human environments using inverse reinforcement learning[J]. International Journal of Social Robotics, 2016, 8(1): 51-66.

[10] PITIS S, CHAN H, ZHAO S, et al. Maximum entropy gain exploration for long horizon multi-goal reinforcement learning[C]//Proceedings of PMLR ICML'20, Virtual Conference. [S.l.:s.n.], 2020: 7750-7761.

[11] NAIR A, MCGREW B, ANDRYCHOWICZ M, et al. Overcoming exploration in reinforcement learning with demonstrations[C]//Proceedings of IEEE ICRL'18, Vancouver. Piscataway: IEEE Press, 2018: 6292-6299.

[12] LE T M, LE V, VENKATESH S, et al. Dynamic language binding in relational visual reasoning[J]. arXiv preprint arXiv:2004.14603.

[13] HOHENECKER P, LUKASIEWICZ T. Deep learning for ontology reasoning[J]. arXiv preprint arXiv:1705.10342.

[14] HUDSON D, MANNING C D. Learning by abstraction: the neural state machine[J]. arXiv preprint arXiv:1907.03950v4.

[15] ADITYA S, YANG Y, BARAL C. Integrating knowledge and reasoning in image understanding[J]. arXiv preprint arXiv:1906.09954.

[16] HU R, ANDREAS J, ROHRBACH M, et al. Learning to reason: end-to-end module networks for visual question answering[C]//Proceedings of IEEE ICCV'17. Piscataway: IEEE Press, 2017: 804-813.

[17] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need[C]//Proceedings of NIPS'17. [S.l.:s.n.], 2017: 1-11.

[18] LE H, TRAN T, VENKATESH S. Self-attentive associative memory[C]//Proceedings of ICML'20. [S.l.:s.n.], 2020: 5682-5691.

[19] LE T H, TRAN T, VENKATESH S. Neural stored-program memory[C]//Proceedings of ICLR'20, Virtual Conference. [S.l.:s.n.], 2020: 1-28.

[20] GRAVES A, WAYNE G, REYNOLDS M, et al. Hybrid computing using a neural network with dynamic external memory[J]. Nature, 2016, 538(7626): 471-476.

[21] HU S, MA Y, LIU X, et al. Stratified rule-aware network for abstract visual reasoning[J]. arXiv preprint arXiv:2002.06838.

[22] GADRE S Y, EHSANI K, SONG S, et al. Continuous scene representations for embodied AI[J]. arXiv preprint arXiv:2203.17251.

[23] TIWARI P, ZHU H, PANDEY H M. DAPath: distance-aware knowledge graph reasoning based on deep reinforcement learning[J]. Elsevier Neural Networks, 2021, 135: 1-12.

[24] CLARK-TURNER M, BEGUM M. Deep reinforcement learning of abstract reasoning from demonstrations[C]//Proceedings of ACM/IEEE HRI'18. Piscataway: IEEE Press, 2018: 160-168.

[25] GAO Y, YANG F, FRISK M, et al. Learning socially appropriate robot approaching behavior toward groups using deep reinforcement learning[C]//Proceedings of 28th IEEE International Conference on Robot Human Interactive Communications. Piscataway: IEEE Press, 2019: 1-8.

[26] WANG Q, HAO Y, CAO J. ADRL: an attention-based deep reinforcement learning framework for knowledge graph reasoning[J]. Elsevier Knowledge-Based Systems, 2020, 197: 105910.

[27] ZHANG J, YU Z, MAO S, et al. IADRL: imitation augmented deep reinforcement learning enabled UGV-UAV coalition for tasking in complex environments[J]. IEEE Access, 2020, 8(1): 102335-102347.

[28] HO J, ERMON S. Generative adversarial imitation learning[C]//Proceedings of ACM NIPS'16. New York: ACM, 2016: 4572-4580.

[29] SCHULMAN J, WOLSKI F, DHARIWAL P, et al. Proximal policy optimization algorithms[J]. arXiv preprint arXiv:1707.06347.

[30] GOODFELLOW I, POUGET-ABADIE J, MIRZA M, et al. Generative adversarial nets[C]//Proceedings of ACM NIPS'14. New York: ACM, 2014: 2672-2680.

[31] Juliani A, Berges V-P, Teng E, et al. Unity: a general platform for intelligent agents[J]. arXiv preprint arXiv:1809.02627.

[32] Yu Z, Zhang J, Mao S, et al. RIRL: a recurrent imitation and reinforcement learning method for long-horizon robotic tasks[C]//Proceedings of IEEE CCNC'22. Piscataway: IEEE Press, 2022: 230-235.

About the Authors

Zhitao Yu received the B.Sc. in Electrical Engineering from Nanjing University of Posts and Telecommunication, Nanjing, China, in 2016, and the M.Sc. in Electrical and Computer Engineering from Auburn University, Auburn, AL, USA, in 2018. He has been pursuing the Ph.D. in ECE at Auburn University since the Spring of 2019. His research interests include indoor localization, deep learning, and indoor UAV navigation. He is a co-recipient of the Runner-up of the Best Paper Award of IEEE CCNC 2022.

Jian Zhang received the B.Sc. and the M.Sc. degrees in Applied Physics from Sichuan University, Chengdu, China in 2001 and 2008, respectively, and the Ph.D. degree in Electrical and Computer Engineering from Auburn University, Auburn, AL, USA, in 2016. Currently, he is an Assistant Professor in the Department of Electrical and Computer Engineering at Kennesaw State University, USA. His main research interests include RFID technologies and applications, the Internet of Things, indoor localization, and intelligent robotics. His work focuses on improving the efficiency of supply chain management for industry and business. He is a co-recipient of the Runner-up of the Best Paper Award of IEEE CCNC 2022.

Shiwen Mao [corresponding author] (S'99-M'04-SM'09-F'19) received the Ph.D. degree in Electrical Engineering from Polytechnic University in 2004. Currently, he is a Professor and Earle C. Williams Eminent Scholar, and Director of the Wireless Engineering Research and Education Center at Auburn University, USA. Dr. Mao's research interests include wireless networks, multimedia communications, and smart grid. He received the IEEE ComSoc TC-CSR Distinguished Technical Achievement Award in 2019, the Auburn University Creative Research and Scholarship Award in 2018, the NSF CAREER

Award in 2010, and several service awards from IEEE ComSoc. He is a co-recipient of the 2021 Best Paper Award of Elsevier/KeAi Digital Communications and Networks Journal, the 2021 IEEE Internet of Things Journal Best Paper Award, the 2021 IEEE Communications Society Outstanding Paper Award, the IEEE Vehicular Technology Society 2020 Jack Neubauer Memorial Award, the 2018 Best Journal Paper Award and the 2017 Best Conference Paper Award from IEEE ComSoc MMTC, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems. He is a co-recipient of the Best Paper Awards from IEEE ICC 2022 and 2013, IEEE GLOBECOM 2019, 2016, and 2015, and IEEE WCNC 2015, and the Best Demo Awards from IEEE INFOCOM 2022 and IEEE SECON 2017.

Senthilkumar C G Periaswamy received the Ph.D. degree in Computer Science from the University of Arkansas Fayetteville in 2010. He is currently the Director of Technology for the RFID Lab at Auburn University, a unique collaboration platform that involves end users, suppliers, technology providers, standards organizations, industry groups, and academic institutions on a global scale. He has researched, advised, and executed projects that enable the efficient adoption of RFID and sensor fusion in retailing, aerospace, manufacturing, and transportation. His work has focused on the common goal of making the adaptation of RFID and related sensor technologies more secure, efficient, reliable, and useful.

Justin Patton is the Director of the Auburn University RFID Lab, a research institute focusing on the business case and technical implementation of emerging technologies in retail, supply chain, aerospace, and manufacturing. The RFID Lab is a unique private/academic partnership between users, technology vendors, standards organizations, and faculty. Justin has participated in business case research for advanced technology with Walmart, Target, Amazon, FedEx, Dillard’s, Macy’s, Delta Air Lines, and Boeing, and is currently researching upstream supply chain benefits of RFID in both retail and manufacturing. He is one of the primary developers of the ARC program, the first and most widely utilized international performance validation system for RFID, and is currently working to help standardize the process of testing and certifying RFID performance in all aspects of the supply chain.