MGRL: Graph neural network based inference in a Markov network with Reinforcement Learning for visual navigation[☆]

Yi Lu[a,b], Yaran Chen[a,b], Dongbin Zhao[a,b,*], Dong Li[a,b]

[a] *The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China*
[b] *University of Chinese Academy of Sciences, Beijing 101408, China*

Abstract

Visual navigation is an essential task for indoor robots and usually uses the map as assistance to providing global information for the agent. Because the traditional maps match the environments, the map-based and map-building-based navigation methods are limited in the new environments for obtaining maps. Although the deep reinforcement learning navigation method, utilizing the non-map-based navigation technique, achieves satisfactory performance, it lacks the interpretability and the global view of the environment. Therefore, we propose a novel abstract map for the deep reinforcement learning navigation method with better global relative position information and more reasonable interpretability. The abstract map is modeled as a Markov network which is used for explicitly representing the regularity of objects arrangement, influenced by people activities in different environments. Besides, a knowledge graph is utilized to initialize the structure of the Markov network, as providing the prior structure for the model and reducing the difficulty of model learning. Then, a graph neural network is adopted for probability inference in the Markov network. Furthermore, the update of the abstract map, including the knowledge graph structure and the parameters of the graph neural network, are combined into an end-to-end learning process trained by a reinforcement learning method. Finally, experiments in the AI2THOR framework and the physical environment indicate that our algorithm greatly improves the success rate of navigation in case of new environments, thus confirming the good generalization.

Keywords: `Visual navigation; graph neural network; Markov network; reinforcement learning; probabilistic graph model;`

[☆]This work is supported partly by the National Natural Science Foundation of China (NSFC) under Grants No. 61803371, the Beijing Science and Technology Plan under Grants Z191100007419002, and No. GJHZ1849 International Partnership Program of Chinese Academy of Sciences.

[*]Corresponding author

Email addresses: `luyi2017@ia.ac.cn` (Yi Lu), `chenyaran2013@ia.ac.cn` (Yaran Chen), `dongbin.zhao@ia.ac.cn` (Dongbin Zhao), `lidong2014@ia.ac.cn` (Dong Li)

1. Introduction

A visual navigation task has practical significance in various indoor robots [1, 2], in which the agent learns to make a series of action decisions for finding a given target only based on visual input. The visual input is the observation image of the current state by the camera on the agent, which is usually a partial observation of the environment. The agent makes a series of decisions to find the target by the partial observation, known as the partially observable Markov decision process (POMDP). When the target is not in the observation image, the agent should take a lot of trials and errors to explore the environment for locating the target.

Including the global location information of the environment, the map is regarded as a useful tool for the agent to understand the entire environment and locate the target [3, 4, 5]. Many map-based navigation methods use known maps as reference system to provide crucial location information for navigation [6, 7]. However, these methods fail in the new environment when the map is unknown. To solve this, the map-building-based navigation methods, such as simultaneous localization and mapping (SLAM) and concurrent mapping and localization (CML) [8, 9], building a map before navigation, are proposed. Although these methods allow an agent to navigate in the new environments, the complex map-building process, including exploration, localization, and map generalization processes [9], limits their application scenarios, especially the changeable indoor environments. Actually the matching of the traditional map and the environment location is the reason why the map-based and map-building-based navigation methods fail and limit in new environments.

In recent years, non-map-based navigation methods, especially the deep reinforcement learning based methods which are not using a map as an assist and often learn a deep network to make navigation decision [10, 11, 12]. These methods mainly utilize the excellent image representation ability of deep network and achieve state-of-the-art performance in navigation [11, 12]. However, without the help of map, these data-driven methods lack interpretability and the global view of the environment. So we consider to offer an abstract map to the deep reinforcement learning navigation method which can break the traditional map limitation and utilize the powerful representation ability of the deep network.

In our paper, we propose Markov network for modeling the environment as an abstract map. Because in different indoor environments, the object positions are arranged by people activities which imply a certain regularity such as chairs next to the desk, cups on the desk. As this regularity guides people's actions when searching in the new environment, it inspires us to build a model for the regularity to assist agent navigation in the new environments. Considering the regularity of relative position relationships and the uncertainty of the environments, the regularity can be supposed to subject to a certain probability distribution with the random vector to represent the object positions and the

joint probability to represent the regularity of relative position relationships. Obviously, the probability distribution of the object positions is complex and contains multiple random vectors which is hard to learn. Because Markov network is one of the probabilistic graphical models which can compactly encode a complex probability distribution over a high-dimensional space by a graph-based representation [13]. Therefore, we consider to build a Markov network for expressing the objects position distribution of different environments as an abstract map.

Different from traditional maps containing specific locations and visual features, the Markov network consists of random vectors as nodes and the direct probabilistic interactions between the nodes as edges. In our model, the Markov network represents relative distance between objects and describes the position in terms of a random distribution, therefore, the abstract map modeled by the Markov network is applicable to more varieties of environments than traditional maps.

By modeling the environment as a Markov network, we propose an intelligent system (hereafter, named as MGRL) to realize the intelligent navigation decision for agent. MGRL is named by the three critical components in constructing the intelligent system: representation as the Markov network, inference by a Graph neural network , and learning under a reinforcement learning framework.

For the representation, to further make learning easily, the structure of a knowledge graph is introduced as the prior of the Markov network. For the probability inference, the graph neural network (GNN) is selected as it has an excellent ability to process graph structured data. For the learning, considering that the deep reinforcement learning performs well in navigation, we propose a reinforcement learning method to determine the weights of the model and updating the graph structure on the basis of interaction with the environment.

By the environment modeled by a Markov network, inference with a GNN, learning conducted under the Reinforcement Learning algorithm, the intelligence system MGRL improves the generalizability of the baseline method and outperforms a benchmark method with a high success rate in new environments of the simulation environment AI2THOR [14].

In summary, the contributions are itemized as follows.

1) We propose a reinforcement learning method combined with a Markov network (MGRL), which represents the environment by an uncertain model.

2) MGRL method combines probability inference and structure learning by reinforcement learning, further improving the ability of the baseline method to adapt to different environments.

3) The interpretability of the proposed method is demonstrated through experimental data analysis and theoretical analysis.

This paper is structured as follows. Section 2 briefly describes the previous research on navigation and probability inference by GNNs. Section 3 illustrates the baseline advantage actor-critic (A2C) method for visual navigation. Section 4 details the construction of MGRL method. Section 5 presents the experiments

and the theoretical analysis to reveal the interpretability of the method in improving the generalization ability for various environments. Finally, Section 6 presents the conclusion and addresses the future work.

2. Related Work

Visual navigation is a fundamental task related to autonomous movement of robots, and it has been extensively researched for several decades [10, 15, 16, 11]. We present a brief review of related work in the following aspects: deep reinforcement learning methods in navigation, the Markov network in navigation, and GNNs for probability inference in navigation.

2.1. Deep Reinforcement Learning in Navigation

Reinforcement learning is suitable for sequential decision making in the partially observable Markov decision process and has been widely used in navigation [17, 18, 19, 20]. With the development of deep learning techniques, deep reinforcement learning effectively utilizes the end-to-end training process to extract visual features and learn navigation policy simultaneously [10, 15, 21]. In 2017, Zhu *et al.* use the generic Siamese network to process observation and target images to obtain the state of the reinforcement learning [10]. Gupta *et al.* combine a mapping module with a planning module to output a local 2D representation of the environment, which helps an agent obtain the environment location information [22]. In 2019, Mousavian *et al.* use more detailed visual informations such as objects detection masks, segmentation and the depth information to improve the success rate of navigation [11].

The aforementioned methods use deep network to extract effective features for navigation, but ignore the environment prior and lack a global view of the environment. Furthermore, many studies use memory mechanisms to build a topological structure data to store the visual features of an environment before navigation; these navigation methods with memory mechanisms can be incorporated with map-building-based methods [23]. The map-building-based methods focus on the problem of map building, which involves time-intensive processes of exploring the entire environment and generating maps before navigation [9, 24]. However, the common trait of these methods is that the maps contain absolute position information, which renders the model unsuitable for various environments.

2.2. Markov Network in Navigation

Markov network, a kind of probabilistic graph model, represents the probability distribution by the undirected graphical model which containing objects, characterized by node features, and edges as representing the direct probabilistic interactions [13]. Probabilistic graph model is a reasoning tool that is independent from the knowledge [13]. So it can be improved for different domain without revising the reasoning algorithms constantly. Besides, the probabilistic model

can tackle the uncertainty from the limited observation and knowledge. Furthermore, the graph-based representation makes the model succinctly describing a complex distribution on a high-dimensional domain .

In previous studies of navigation, the probabilistic graph model is mainly used to represent the uncertainty caused by the partial observation. For example, in 2013, to alleviate the effect of inaccurate location of objects in the map, the probabilistic graph model is introduced into the navigation to model the uncertainty [25]. In recent years, Bayesian model, Markov network, and conditional random field are used for modeling the localization in navigation [26, 27, 28]. However, the probabilistic model based on the relative position relationship for the object positions distribution in the whole environment has been rarely studied.

2.3. GNNs for probability inference

In 2005, Gori and Scarselli firstly propose GNN which is a kind of deep learning method for handling graph data [29]. In 2009, GNN is further illustrated and proved to be a continuously differentiable function [30]. In few years, it is starting to refocus people for its excellent handling of graph data. GNN is good at extracting the feature of data containing graph structure and achieves state-of-the-art results in many fields [31, 32]. GNN consists of two crucial modules, the aggregation and the combination, in extracting the node features.

In literature, the use of GNNs in probability inference has been reported. In 2018, Yoon *et al.* prove that GNN architecture is well suited for the probability inference task by demonstrating a better performance of GNN than the belief propagation method even for loopy graphs [33]. Moreover, graph Markov neural network (GMNN), proposed by Qu *et al.* in 2019, uses two GNNs for learning and inference and achieves state-of-the-art results in object classification [34]. However, these systems are trained by supervised learning with the fixed structure. In our method, a GNN combined with a cosine function is used to predict the joint probability and it is trained by the reinforcement learning with the structure updatincurr.

3. Problem Formulation and the Baseline method

In the navigation to a target with semantic labels, the agent is required to reach the target by visual observation. We propose a method that uses a deep reinforcement learning framework for helping agents make correct navigation decision and reach variable targets from sequential decisions in different types of indoor scenes. Reinforcement learning uses a policy function for the decision on "where to go", $p(a_t|\mathbf{s}_t, \theta)$, usually written as $\pi_\theta(a_t|\mathbf{s}_t)$, where a_t is the action of step t and $\mathbf{s}_t$ is the state of step t. The objective of reinforcement learning is to learn the optimal policy function that achieves the maximum expected return, which can be calculated as the total reward of the trajectory, written as

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \sum_{t=1}^{T} E_{(\mathbf{s}_t,a_t)\sim\pi_\theta(a_t|\mathbf{s}_t)}[r(\mathbf{s}_t,a_t)], \\ &= \arg\max_{\theta} E_{(\mathbf{s}_t,a_t)\sim\pi_\theta(a_t|\mathbf{s}_t)}[\sum_{t=1}^{T} r(\mathbf{s}_t,a_t)] \\ &= \arg\max_{\theta} E_{(\mathbf{s}_t,a_t)\sim\pi_\theta(a_t|\mathbf{s}_t)}[R]\end{aligned} \tag{1}$$

where $r(\mathbf{s}_t, a_t)$ is the reward function; $R = \sum_{t=1}^{T} r(\mathbf{s}_t, a_t)$ is the cumulative total return; θ are parameters to learn and θ^* are the optimal parameters.

In visual navigation methods, observation images are the inputs and the action decision is the output. The input observation images are obtained from three perspectives, namely, the front, right, and left views, and are respectively denoted as $\mathbf{o}_t = [o_t^0, o_t^1, o_t^2]$. The corresponding actions are moving forward, rotating right, and rotating left, denoted as a^0, a^1, a^2, respectively. In reinforcement learning, the method evaluates the action of current step by a reward. If an agent reaches the target, a positive reward of 10.0 is given for this step; otherwise, a penalty of -0.01 is assigned to each step of the agent:

$$r_t = \begin{cases} 10 & if\ \mathbf{s}_t = s_T \\ -0.01 & otherwise, \end{cases} \tag{2}$$

where s_T is the state of target reached. As in previous studies, when an agent arrives at an object within a fixed distance, the target is considered to be reached.

For the purpose of conducting a contrastive study, we use the A2C algorithm as the baseline method. The A2C algorithm, frequently used in navigation owing to its efficiency and ease of implementation, is a type of actor-critic policy network containing two modules, namely, an actor and a critic [35]. The actor and critic in the A2C method share input $\mathbf{s}_t$ and output the values of state $\mathbf{s}_t$ and action a_t, respectively. In the baseline A2C method for navigation, state $\mathbf{s}_t$ is the visual feature of the observation, obtained by the Resnet [36]. The Resnet contains a stacked residual block as the identical transformation, whose output vectors are deemed to hold the image information as much as possible [36]. With the pre-trained Resnet, each image can be transformed into a 2048-dimensional vector. Then, the observations $\mathbf{o}_t = [\mathbf{o}_t^0, \mathbf{o}_t^1, \mathbf{o}_t^2]$ can be replaced by a 6144-dimensional vector obtained by the Resnet, because the Resnet does not participate in the training. In the A2C method, the visual feature $\mathbf{s}_t^v$ is a 512-dimensional vector obtained by a fully connected layer and represented as

$$\mathbf{s}_t^v = F_\beta(\mathbf{o}_t),$$

where F is a fully connected layer, β are the parameters.

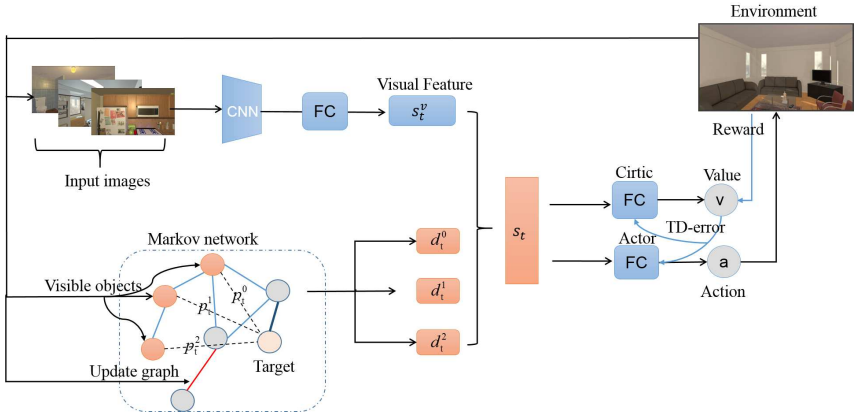

Figure 1: Structure of MGRL method. The state contains two parts, the visual feature and relational feature $\mathbf{s}_t^g = [d_t^0, d_t^1, d_t^2]$. p_t^0, p_t^1, p_t^2 are the joint probabilities on the Markov network which are obtained by a GNN-based model with the prior knowledge graph structure and semantic node features. Meanwhile, in the A2C training process, the knowledge graph structure is updated by new observations.

4. MGRL for Navigation

With the framework of A2C, the structure of MGRL is shown in Fig. 1. Different from the previous approach, besides the visual feature, MGRL considers a novel feature, named as the graph relational feature, which is deduced by the graph module based on the Markov network, shown in the dashed box of Fig. 1. The Markov network is modeled for an abstract map of the environment, which is with the probability inference process implemented by a GNN-based model.

As shown in Fig. 1, the inputs of the A2C method are the visual features $\mathbf{s}_t^v$ and the graph relational features $\mathbf{s}_t^g$, which is combined as the state $\mathbf{s}_t$ of the environment. The graph relational features measure the relative distance between the observation and the target, and thus provides a global view of the environment.

In MGRL, the graph relational features are the edge features of the direct probabilistic interactions between the target and the observation objects, which are obtained by probability inference. Probability inference for calculating the joint probability in the Markov network is conducted by a GNN-based model with the prior knowledge graph structure and semantic node features. In the training process, the structure is also updated on the basis of the interaction of the environment.

The following sections present detailed descriptions of how to build the environment abstract map by Markov network, how to use the abstract map, updating the graph structure and the learning process of the entire model.

4.1. The Abstract Map Built by Markov network

We describe the details of building the Markov network as the abstract map. The Markov network (also named the Markov random field) is a probabilistic

model that utilizes the graph topological structure to represent the joint probability; it contains an undirected graph as the structure with random variables as the nodes and direct probabilistic interactions as edges [37, 13]. In general, object positions have statistical regularity with complex relations among objects, which are useful for establishing navigation. The Markov network can suitably describe the statistical regularity of positions and the relation dependency of objects.

In our Markov network, the node set is denoted as N, with each node n_i corresponding to an object position.

In the navigation task, the direct probabilistic interactions between target n_T and the observation objects $n_O = (n_{o^0}, n_{o^1}, n_{o^2})$ is the probability of that the target appears where the observation objects are. In others words, the edge features of the Markov network are obtained by the joint probability of the adjacent relationships between the target and the observation objects, shown as

$$(p_t^0, p_t^1, p_t^2) = (p(n_T, n_{o^0}), p(n_T, n_{o^1}), p(n_T, n_{o^2})),$$

where $p(n_i, n_j)$ denotes the joint probability between node n_i and n_j. The edge features calculated by the joint probability can be taken as the measurement of the relative distance between the objects (nodes) in the Markov network.

The Markov network determines the direct probabilistic interactions between nodes, measured by the joint probability; the joint probability of that the target and the observation objects are adjacent has inversely proportional relationship with the nodes relative distances, and it is written as

$$Distance(n_i, n_j) \propto 1/p(n_i, n_j), \tag{3}$$

where n_i and n_j are the nodes of the graph. $Distance(n_i, n_j)$ is the value of the edge e_{ij}, so there is edge e_{ij} in the Markov network means the probability of that two objects are adjacent is not zero.

In a navigation task, the agent can only observe partial information of the environment and often lacks the target location information. By using the Markov network, the navigation is transformed into finding the shortest path between the current state to the target. A greedy strategy can be selected to determine where to go, which provides the optimal action where the agent moves to the nearest observed object. In MGRL, the strategy is realized by a neural network with one fully connected layer. For the navigation to an unknown location, unlike in the traditional map that uses the Euclidean distance between the objects as a metric, the Markov network uses the relative distance as the metric.

Although the Markov network has no absolute position, the relative distance calculated by the joint probability between the observation objects and the target can guide the agent move close to the goal. So in navigation, the Markov network containing the relative distance and the objects as nodes has the same function as the traditional navigation map and can be regarded as an abstract map.

To illustrate the probability inference process for obtaining the joint probability $p(n_i, n_j)$, we first describe how to establish the prior graph structure and node features.

4.1.1. Prior Graph Structure

A knowledge graph, built by Visual Genome dataset [38], is used to initialize the graph structure, which can facilitate the graph model to learn the Markov network quickly. Visual Genome dataset is proposed by Stanford University, which contains 3.8 million object instances and 2.3 million relationships. The relationships of the involved objects are extracted as the edges to build the knowledge graph structure.

4.1.2. Node Features

The vector representation of semantic labels obtained by GloVe [39] is used as the node features. GloVe, proposed by Pennington *et al.*, transforms words into vectors that can retain the semantic information of the words [39]. The distance (the Euclidean distance or cosine similarity) of these vectors can effectively measure the semantic similarity of the corresponding words; therefore, the closest words in the vector space are sometimes synonyms or within the same main category. In our method, the 300-dimensional GloVe vector is selected to represent the semantic labels. The vector similarity provides the object category relations; therefore, the GloVe vector representation can provide more object associations.

Thus, a graph is initialized with the structure from prior relations and node features from the GloVe vector representation.

4.1.3. Probability Inference by GNN-based Model

In this part, we describe the process of predicting the joint probability. The joint probability represents the probability that two nodes (objects) are adjacent. We intend to obtain the node embedding vectors between whose similarity is consistent with the joint probability, written as

$$p(n_i, n_j) \doteq similarity(h_i, h_j),$$

where h_i, h_j are the embedding vectors of node i and j, respectively.

To obtain the node embedding vectors we should combine the features of the object and the adjacent relationships between objects containing in the structure of the Markov network. We calculate the node embedding vectors by a GNN for its excellent performance in extracting features on graph data.

In 2017, Thomas *et al.* propose graph convolutional network (GCN), which is one of the spectral-based GNNs [40]. GCN simplifies the eigendecomposition of the Laplacian matrix and proposes a Laplacian transformation on the graph. GCN is used to obtain the hidden features of the nodes by performing message propagation to exchange information, as

$$h_{output} = \widetilde{D}^{-\frac{1}{2}}(A + I_{|N|})\widetilde{D}^{-\frac{1}{2}} h_{input}\omega^*,$$

where A is the adjacent matrix of the graph; $|N|$ is the number of the nodes; ω is the parameters of one layer to learn; $\widetilde{D}$ is a diagonal matrix with element $\widetilde{D}_{ii} = \sum_j (A + I_{|N|})_{i,j}$ [40].

With the non-linear activation functions, a two-layers GCN can be written as

$$G_\omega(A, X) = softmax(\widetilde{D}^{-\frac{1}{2}}(A + I_{|N|})\widetilde{D}^{-\frac{1}{2}} ReLU(\widetilde{D}^{-\frac{1}{2}}(A + I_{|N|})\widetilde{D}^{-\frac{1}{2}} X\omega^0)\omega^1),$$

where X is the node feature matrix as the input of the GCN.

Because GCN can effectively capture the correlation information embedded in the structure, we use GCN to learn the node embedding features for joint probability prediction, written as

$$\mathbf{h} = G_\omega(A, N),$$

where G represents the GCN; $\mathbf{h}$ is the node embedding obtained by the GCN; $N = [n_0, \ldots, n_{|N|}]$ is the node feature matrix as the input feature matrix of GCN.

Second, the cosine similarity function is used to calculate the joint probability, written as

$$p(n_i, n_j) \doteq similarity(h_i, h_j) = cos(h_i, h_j),$$

where $cos(h_i, h_j)$ ranges from 0 to 1 because $\mathbf{h} > 0$ which is outputting from the softmax function.

4.2. Using Markov Network as an Abstract Map to Guide the Navigation

We have obtained the joint probability on the Markov network. In this section we detail how the joint probability participates in navigation decisions. Because the joint probability represents the probability that two nodes (objects) are adjacent which is inversely proportional to the distance between them. We can predict the relative distance between the observed object i and the target, shown as (4).

$$\begin{aligned} Distance(n_i, n_T) &\doteq d(h_i, h_T) \\ &= 1 - cos(h_i, h_T) \\ &= 1 - cos(I^i \cdot \mathbf{h}, I^T \cdot \mathbf{h}) \\ &= 1 - cos(I^i \cdot G_\omega(A, N), I^T \cdot G_\omega(A, N)) \end{aligned} \tag{4}$$

where I^T and I^i are diagonal matrix where the Tth and the ith elements in the diagonal are 1, while the other elements are 0. I^k can be taken as an indicator function to obtain the kth hidden feature of the object.

$Distance(\cdot, \cdot)$ is a non-negative, reflexive, and with the value scope [0-1]. Because $Distance(\cdot, \cdot)$ does not satisfy the triangle inequality, it is a pseudo-metric that is used to evaluate the relative distance between the observed object and the target.

The objects are detected by a pre-training CNN [41] which does not participate in training. The distances between the target and the visible objects in one view are denoted as $d_t^i, (i = 0, 1, 2)$. For the visible objects from the left, right, and front views, we obtain a three-dimensional vector $\mathbf{s}_t^g = [d_t^0, d_t^1, d_t^2]$, named graph relational features, to represent the relative distance feature obtained from the Markov network. In particular, when there is no object in one view, the predict relative distance is 0; when there are multi-objects in one view, the predict distance is the nearest one.

The state $\mathbf{s}_t$ is obtained by the concatenation of visual features $\mathbf{s}_t^v$ and graph relation features $\mathbf{s}_t^g$, written as

$$\mathbf{s}_t = [\mathbf{s}_t^v, \mathbf{s}_t^g] = [F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2]. \tag{5}$$

Two approaches to obtain $\mathbf{s}_t$ are tested experimentally, direct concatenation and concatenation after a fully connected layer to obtain $\mathbf{s}_t^v$ with the dimension as that of $\mathbf{s}_t^g$. Because both approaches yield similar performance in the test, we finally choose direct concatenation.

The action decision is obtained by the policy function in the form of a neural network with one fully connected layer as

$$\pi_\theta(\mathbf{o}_t) = F_\eta([\mathbf{s}_t^v, \mathbf{s}_t^g]) = F_\eta([F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2]), \tag{6}$$

where η is the parameter of the fully connected layer and $\theta = \omega, \beta, \eta$.

By adding the graph relational features s_t^g, even the target is not observed, the global information of the relative distances to the target is considered which is usually provided by the maps. In MGRL, this global information can be obtained to assist navigation from the Markov network which acts as the role of an abstract map.

4.3. Graph Structure Updating

The prior knowledge graph structure represents the adjacent relations between objects. The more consistent the knowledge graph with the environment is, the more accurate prediction the joint probability make. In reinforcement learning, the agent can interact with the environment, and the observation contains new relations from the current environment. The close and visible objects in the adjacent steps can be as the new relations. The relations between close objects are the new relations that are added as the graph edges.

4.4. Learning Process

As the baseline method, the model is learned by the A2C algorithm. The actor and critic functions of A2C are both a neural network with one fully connected layer, outputting the action and the value, respectively. The actor is the function mentioned in (6). The two fully connected layers are denoted as $F_\gamma(\cdot)$ and $F_\eta(\cdot)$, where γ and η are the parameters to learn. Then, we obtain the critic function as

$$V_{\omega,\beta,\gamma}(\mathbf{o}_t) = F_\gamma([\mathbf{s}_t^v, \mathbf{s}_t^g]) = F_\gamma([F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2]). \tag{7}$$

where γ is the parameters of the fully connected layer in the critic.

The value loss is given by

$$V_{loss} = (R - V(\mathbf{o}_t))^2 = (R - F_\gamma([F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2]))^2. \tag{8}$$

The policy gradient is given by

$$\begin{aligned} &\nabla_\theta \ \pi_\theta(\mathbf{o}_t) \\ =&\nabla_\theta \ log(\pi(\mathbf{o}_t))(R - V(\mathbf{o}_t)) \\ =&\nabla_\theta \ log(F_\eta([F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2])))(R - F_\gamma([F_\beta(\mathbf{o}_t), d_t^0, d_t^1, d_t^2]). \end{aligned} \tag{9}$$

By the A2C reinforcement learning, all the parameters are determined.

5. Experiments

First, to evaluate the effectiveness of the proposed MGRL method, it is compared with the baseline A2C method and the random algorithm. Then, to investigate the generalizability of the proposed method, comparative experiments are conducted with Yang's benchmark method [12]. To further explore the mechanism of MGRL, ablation experiment are conducted with only the Markov network. Furthermore, the variable visualization of the intermediate output is analyzed to explore the interpretability of MGRL.

5.1. Experimental Environment

We evaluate our method using the AI2THOR framework, which is a high-fidelity home indoor simulation environment proposed by Kolve *et al.* [14]. The framework includes four categories of scenes (kitchen, living room, bedroom, and bathroom) and 105 categories of objects. Each category scene consists of 30 rooms with different decoration and layout styles. Many methods have used the AI2THOR framework to evaluate the navigation task owing to its high similarity to the physical environment in terms of factors such as lighting and interior decoration.

5.2. Comparison with the Baseline Methods

In this section, three groups of experiments are conducted to evaluate the proposed MGRL's effectiveness in comparison with the baseline methods A2C and random algorithm in term of success rate. Test 1 validates the learning ability of MGRL in the same environment of the navigation task. Test 2 validates the generalizability of MGRL in different scenes with the same category. Test 3 demonstrates the generalizability of MGRL for learning in multiple categories of scenes.

5.2.1. Experimental Implementation

The maximum step size of each episode is 100, which implies that the agent reaching the target in 100 steps is one successful episode. The first 20 scenes of each category scenes in the AI2THOR framework are adopted as the training set, scenes $21-26$ are taken as the test set, and the last five scenes are taken as the valuation set. The training and test targets are listed in Table A.5 (see Appendix A). The success rate is the average of 500 episodes on the condition that the target, scene, and start locations are all randomly selected with the same random seed.

To initialize the node features, the 300-dimensional word representation vectors obtained by the GloVe [39] are selected as the semantic node features of the 105 categories, and the corresponding relations are obtained from the Visual Genome dataset. Thus, the size of the node feature matrix is 105×300 and that of the adjacent matrix is 105×105. A three-layer GNN with the output size the same as the dimension of the initial node feature is applied in MGRL.

To evaluate the learning ability of MGRL in the same environment, Test 1 is conducted using the same scenes (here, kitchen) as the training scenes and the test targets (as shown in Table A.5). To investigate the ability of the proposed MGRL to be generalized in different environments, Test 2 is conducted using new scenes (scene type of bedroom) and test targets. To validate the extensibility of the proposed, Test 3 is conducted with training in four types of scenes (kitchen, living room, bedroom, and bathroom) and validation in new scenes of the four categories of scenes.

5.2.2. Results

Table 1: Results of comparison of the proposed MGRL with baseline methods.

Method(%)	Test 1	Test 2	Test 3
Random	50.4	36.4	55.67
A2C	59.9	58.8	58.89
MGRL(ours)	**61.4**	**64.4**	**61.22**

The results are shown in Table 1. In general, MGRL achieves the highest success rate of all the three methods in all tests. Thus, the feasibility and validity of our proposed method are verified. Concretely, the only difference between MGRL and the A2C baseline method is the inclusion of the Markov network in our method, which provides the relative distance prediction in a global manner. The improved success rate suggests that this global relative distance information helps in selecting the correct action. Furthermore, the improved success rate reflects that the Markov network benefits in generalization in multiple environments.

5.3. Comparison with the Benchmark Method

Yang *et al.* first introduce the knowledge graph as a scene prior in the AI2THOR framework; when the method is applied to new scenes, the improvement of the success rate is observed [12]. To further validate the generalizability of our method, we compare MGRL with Yang's method.

5.3.1. Experimental Implementations

In our experiment, MGRL is trained for four categories of scenes. In the test, the method searches for known target objects in the test set whose scenes are different from the training set.

5.3.2. Results

Table 2: Results of comparison of the proposed MGRL with the benchmark method.

Methods(%)	new scene, known objects
Yang [12] *	56.9
MGRL	**60.12**

* The result is an average of four models training and test in the same category scene independently.

The success rates of the two methods are shown in Table 2. Yang's method is trained with four models corresponding to each category scene, while one MGRL is trained for four categories of scenes in our method. In Yang's test, the maximum step size of the living room is 200, which is more than that of the proposed MGRL.

Considering these differences between the two methods, the success rate is still improved by our method, which suggests the powerful generalization of MGRL. The performance improvement can be attributed to the Markov network model used in MGRL, which provides more useful information about environment. The specific reasons for the performance improvement will be analyzed in the following experiments.

5.4. Analysis

The ablation experiments and the variable visualization of the intermediate output are conducted to study the role of the Markov network.

5.4.1. Ablation Experiments

In the ablation of the Markov network, only the visual features $\mathbf{s}_t^v$ are included in state $\mathbf{s}_t$, thus corresponding to the baseline A2C method; the comparison analysis of the proposed method with A2C is in section 5.2. In the ablation of visual features, only graph relational features $\mathbf{s}_t^g$ are used in MGRL which named the graph-only MGRL. Two experiments are conducted, one with the seen scenes in the training set and another one with new scenes in the test set.

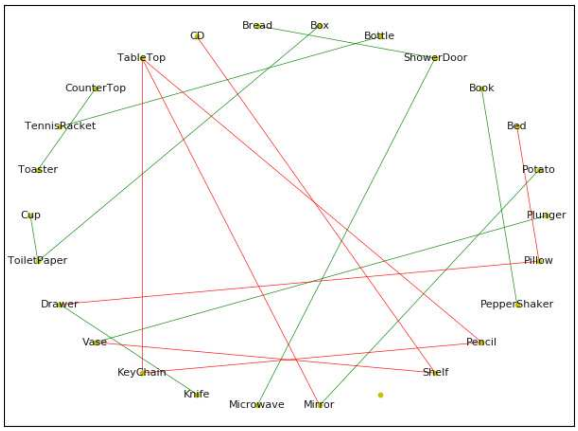

Figure 2: Changes in the knowledge graph. Part of the knowledge graph structure is shown in the picture. The red lines indicate the new relations obtained by updating the structure.

The performance of the graph-only MGRL can surpass the performance of MGRL in the case of new scenes with known objects, as shown in Table 3. In the new scenes, the visual features in MGRL contain the visual details learned in the training set which may be not suitable for new environments. This result confirms the validity of the features obtained by the Markov network for navigation. In case of seen scenes with known objects, MGRL with the visual feature performs better than the graph-only MGRL. This implies that the visual feature is effective for a familiar environment with known visual details.

Table 3: Success rates of graph-only MGRL and MGRL.

Methods(%)	Seen scenes	New scenes
MGRL	**73.2**	60.12
Graph-only MGRL	71.0	**63.12**

5.4.2. Visualization Analysis

We study the graph structure changes and the graph relational features to analyze the reason why MGRL provides good results.

The changes in the graph structure are presented in Fig.2. For the sake of presentation, part of the nodes and relations are displayed in Fig.2. In the figure, the initial relations are indicated by green lines and the updated relations are in red lines. The new relations, such as Keychain-TableTop, Pencil-TableTop, and CD-Shelf, reflect the new habits of arranging things that do not appearance in the initial graph and thus are new knowledge of the environment. Since the method learns the new knowledge, the environmental cognition is enhanced by the graph updating process.

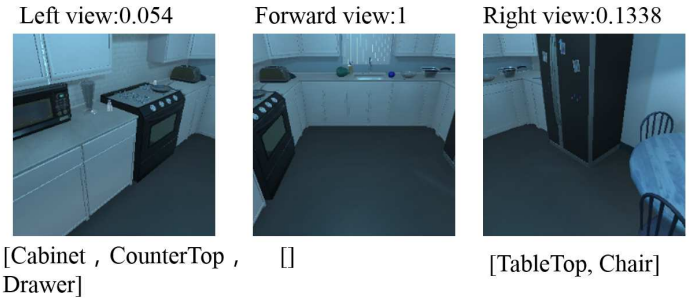

Figure 3: Predicted distance and the corresponding images. This figure shows the predicted distance obtained by MGRL in the one-step decision of navigating to the *toaster*: The three images are the observations from three views: left, front, and right. The predicted distances are up to the images, and the observed objects are under the images. In the front view, there is no object detected with the largest distance 1. The predicted distance of the left view 0.054 is less than that of the right view 0.1338 which corresponds to the actual relative distance.

Furthermore, the predicted distance and the corresponding images are presented in Fig.3. In the navigation to the toaster, from three views, there is a larger probability that the toaster is on the countertop in the left view than on the table and the chair in the right view, so the predicted distance in the left is less than that in the right. This result indicates that the predicted distance is consistent with reality; thus, Equation (3) is confirmed.

5.5. Robot Experiment

To validate the effectiveness of MGRL in physical environment, we conduct the experiments by using a Kobuki robot (shown in Fig.4 (a)) in a physical environment (shown in Fig.4 (b)). The input images are from the fixed three cameras on the top of the Robot, seen in Fig.4 (a). A successful navigation means that the robot reaches the given target within 100 steps in the physical environment. We compare the success rate of random navigation and MGRL. The trained MGRL by the AI2thor framework is directly transformed into the physical environment without other training. The results are the average success rates of 300 trials for each method, shown in Table 4. The increase of the success rate verifies the validity and generalization of the proposed MGRL.

Specially, when searching for objects such as cup, pen or cellphone, the robot always moves to the *desk* with MGRL. This is the obvious difference between MGRL and random methods, especially when the target is not visible. Compared with the random method, the success rate improvement may be in that the robot makes correct decision by relative distance inference that the target is near to the Desk. The images of one epoch that the robot navigates to *box* are shown in Fig.C.5.

5.6. Discussion

To ensure generalization and obtain global information of the environment, we develop MGRL method combined the Markov network with A2C framework

Figure 4: Robot and physical environment. There are three cameras attached to the top of the Robot. The cameras on both sides are at a 45-degree angle to the middle camera.

Table 4: Success rates of Random and MGRL methods in physical environment.

Methods	Success rate(%)
Random	45.7
MGRL	**61.3**

for indoor navigation. A GNN is applied to solve the probability inference problem in the proposed Markov network model. The knowledge graph is updated online based on interaction with the environment. Experimental results indicate that the proposed MGRL shows excellent performance in generalization.

The development of our method is significant because MGRL contains a Markov network initialized by a knowledge graph and probability inferred by the GNN. These approaches solve the navigation task and achieve deep reinforcement learning interpretability for indoor navigation.

First, the semantic map built by Markov network improves the presentation ability of the environment model in reinforcement learning. Different from the traditional map, in MGRL, the Markov network serves as a semantic map with no specific location information of objects. The map includes the position relations of the environment, thus providing a global view for the partial observation task. Although Yang *et al.* use knowledge graphs to model the environment and achieve excellent results, the comparison and analysis experiments confirm a better adaptive capacity of the Markov network in MGRL. Besides, a GNN is introduced to solve the probability inference in our method, which allows accurate prediction of distances in the experiments and the visualized results in section 5.4.2. Furthermore, based on the interaction with the environment in the training process of reinforcement learning, the observations of new relations are used to update the structure of the knowledge graph. As GNN is introduced, MGRL combines the structure learning with probability inference by reinforcement learning. Thus, the generalization of the method is greatly improved, and to the best of our knowledge, this is the first time this combination is applied

in a navigation task.

Moreover, the Markov network is effective in MGRL, which is not only validated through experiments but also proved theoretically. We can conclude that the proposed Markov network is a formulation of maximum entropy reinforcement learning with the derivation process, as discussed in Appendix B. Reinforcement learning to achieve probability inference is studied in many works such as maximum entropy reinforcement learning [42], KL-divergence control [43], and stochastic optimal control [44]. Moreover, in 2018, Levine of UC Berkeley presentes a general form of this problem [45]. Because maximum entropy reinforcement learning is an optimization of reinforcement learning with the maximum entropy principle [46], MGRL builds the Markov network to represent object position distribution, which is an optimization of reinforcement learning in navigation task.

6. Conclusion

We propose MGRL for visual navigation by introducing Markov network to model the environment. The proposed MGRL uses reinforcement learning and adopts a GNN to predict the joint probability on the Markov network. The joint probability predicts the relative distance between two objects. In addition, the structure of the graph is initialized by a knowledge graph, and the observed relations are used to update the graph structure, which helps the graph module dynamically adapt to the various environments. By conducting experiments using the AI2THOR framework, we validate the generalizability of our method in learning and extensibility to various categories of scenes.

In the future, we will consider using richer types of position relations, such as "above" and "under", to refine the probability graph model and provide better precise guidance to navigation.

Acknowledgements

The authors would like to thank Yanqiang Li for his supports in providing the Robot, and thank Prof. Jun Wang, Qichao Zhang, Junwen Chen and Shuxian Jiang for their helpful comments and discussions.

Appendix A. Experiments Implementation

The training and test targets are listed in Table A.5.

Appendix B. Theoretical analysis of the method

Theorem 1. *The objective of the proposed Markov network is to maximize entropy reinforcement learning for navigation.*

Table A.5: Targets list

Scene type	Training targets	Test targets
Kitchen	HousePlant, StoveKnob, Sink, TableTop, Potato, Bread, Tomato, Knife, Cabinet, Fridge, ButterKnife, Lettuce, Pan, Bowl, CoffeeMachine, StoveBurner, Plate	Mug, Apple, Microwave, Toaster
Living room	Television, HousePlant, Chair, TableTop, Box, Cloth, Newspaper, KeyChain, WateringCan	Painting, Statue
Bed room	Painting, HousePlant, CellPhone, LightSwitch, Candle, TableTop, Bed, Statue, Book, CreditCard, KeyChain, Bowl, Pen, Box, Pencil, Blinds, Laptop, AlarmClock	Television, Cabinet, Mirror
Bathroom	SprayBottle, Candle, LightSwitch, Sink, TowelHolder, Watch, ShowerDoor, SoapBottle, Toilet	Towel, SoapBar, Cabinet, ToiletPaper, Mirror

proof 1. *More generally, let us take each node in the Markov network as a different state, denoted as* $s^i \doteq n_i$.

In reiforcement learning, the reward can reflect as the reduction in the distance of the state from the target; therefore, the reward function can be written as

$$\begin{aligned} & r(\boldsymbol{s}_t, a_t^i) \propto 1/(distance(s_{t+1}^i, s_T) - distance(s_t, s_T)) \\ \Rightarrow & r(\boldsymbol{s}_t, a_t^i) \propto 1/distance(s_{t+1}^i, s_T). \end{aligned} \tag{B.1}$$

In the Markov network, action a_t^i *is selected according to the predict distance, with the aim to go to the closest state to the target. In our Markov network, the probability, used to select the optimal action, is inversely proportional to the distance of the target, written as*

$$p(a_t^i|\boldsymbol{s}_t) \propto 1/distance(s_{t+1}^i, s_T). \tag{B.2}$$

From equations B.1 and B.2, we get

$$p(a_t^i|\boldsymbol{s}_t) \propto r(\boldsymbol{s}_t, a_t^i). \tag{B.3}$$

Without loss of generality, the proportional relationship can be written as

$$p(a_t|\boldsymbol{s}_t, \theta) \doteq \frac{1}{z_t} exp(r_\theta(\boldsymbol{s}_t, a_t)), \tag{B.4}$$

where z_t *is a non-negative coefficient. By equation B.4 and the derivation in*

[45] , we get

$$
\begin{aligned}
p(s_1, a_t \ldots, s_T, a_T) &= \prod_{t=1}^{T} \frac{1}{z_t} p(s_1) \prod_{t=1}^{T} p(s_{t+1}|\boldsymbol{s}_t, a_t) exp(\sum_{t=1}^{T} r_\theta(\boldsymbol{s}_t, a_t)) \\
&= \frac{1}{Z} p(s_1) \prod_{t=1}^{T} p(s_{t+1}|\boldsymbol{s}_t, a_t) exp(\sum_{t=1}^{T} r_\theta(\boldsymbol{s}_t, a_t))
\end{aligned}
\tag{B.5}
$$

where $Z = \prod_{t=1}^{T} z_t$ is non-negative coefficient and can be merged into $p(s_1)$. The Markov network we built is an environment model to predict the relevant distance, expecting to minimize the modeled distribution with the real distribution. The goal can be written as $argmin_\theta D_{KL}(p_\theta(\tau)||p(\tau^*))$, where τ^* is the optimal trajectory $(s_1, a_t^* \ldots, s_T, a_T^*)$.

In the reinforcement learning the $p(a_t^i|\boldsymbol{s}_t, \theta)$ is usually written as policy function $\pi_\theta(a_t|\boldsymbol{s}_t)$.
From the equation (11) in [45] and equation B.5 ,we get

$$
-D_{KL}(p_\theta(\tau)||p(\tau^*)) = \sum_{t=1}^{T} E_{(\boldsymbol{s}_t, a_t)\sim p(a_t|\boldsymbol{s}_t, \theta)}[r(\boldsymbol{s}_t, a_t)] + E_{\boldsymbol{s}_t \sim p_\theta(\boldsymbol{s}_t)}[\mathcal{H}(\pi_\theta(a_t|\boldsymbol{s}_t))]
\tag{B.6}
$$

The objective of the proposed Markov network is equal to maximize

$$
\sum_{t=1}^{T} E_{(\boldsymbol{s}_t, a_t)\sim p_\theta(\boldsymbol{s}_t, a_t)}[r(\boldsymbol{s}_t, a_t)] + E_{\boldsymbol{s}_t \sim p_\theta(\boldsymbol{s}_t)}[\mathcal{H}(\pi_\theta(a_t|\boldsymbol{s}_t))]. \tag{B.7}
$$

This in turn is to maximize the sum of reward and entropy of the policy function, which is exactly the goal of maximum entropy reinforcement learning.

Appendix C. The images of Robot experiments

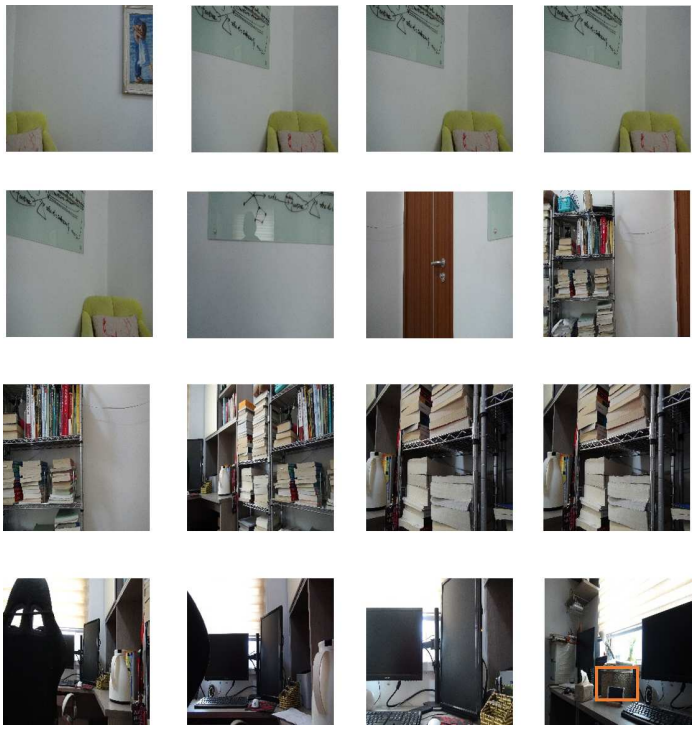

Figure C.5: Images of navigation to the *box* in physical environment. The images are from the front view camera of the robot. Each image represents one step observation in front view of the Robot.

References

[1] S. Thrun, Learning metric-topological maps for indoor mobile robot navigation, Artificial Intelligence 99 (1) (1998) 21–71.

[2] F. Bonin-Font, A. Ortiz, G. Oliver, Visual navigation for mobile robots: A survey, Journal of Intelligent and Robotic Systems 53 (3) (2008) 263–296.

[3] J. Kim, H. Jun, Vision-based location positioning using augmented reality for indoor navigation, IEEE Transactions on Consumer Electronics 54 (3) (2008) 954–962.

[4] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, Autonomous Robot Vehicles (1986) 396–404.

[5] J. Borenstein, Y. Koren, Real-time obstacle avoidance for fast mobile robots, IEEE Transactions on systems, Man, and Cybernetics 19 (5) (1989) 1179–1187.

[6] K. O. Arras, R. Y. Siegwart, Feature extraction and scene interpretation for map-based navigation and map building, Mobile Robots XII 3210 (1998) 42–53.

[7] J.-A. Meyer, D. Filliat, Map-based navigation in mobile robots, Cognitive Systems Research 4 (4) (2003) 283–317.

[8] B. Yamauchi, A. Schultz, W. Adams, Mobile robot exploration and map-building with continuous localization, in: IEEE International Conference on Robotics and Automation, 1998, pp. 3715–3720.

[9] N. Lama, B. Sen, K. Gautam, Survey: Visual navigation for mobile robot, in: International Conference on Computing and Communication, 2016, pp. 5–11.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: IEEE International Conference on Robotics and Automation, 2017, pp. 3357–3364.

[11] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, J. Davidson, Visual representations for semantic target driven navigation, in: IEEE International Conference on Robotics and Automation, 2019, pp. 8846–8852.

[12] W. Yang, X. Wang, A. Farhadi, A. Gupta, R. Mottaghi, Visual semantic navigation using scene priors, in: International Conference on Learning Representations, 2019.

[13] D. Koller, N. Friedman, Probabilistic graphical models: principles and techniques, MIT press, 2009.

[14] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, A. Farhadi, Ai2-thor: An interactive 3d environment for visual ai, in: arXiv preprint, 2017. arXiv:1712.05474.

[15] D. Zhao, Y. Chen, L. Lv, Deep reinforcement learning with visual attention for vehicle classification, IEEE Transactions on Cognitive and Developmental Systems 9 (4) (2017) 356–367.

[16] N. Zeng, H. Zhang, Y. Chen, B. Chen, Y. Liu, Path planning for intelligent robot based on switching local evolutionary pso algorithm, Assembly Automation 36 (2) (2016) 120–126.

[17] Y. Zhou, E.-J. van Kampen, Q. Chu, Hybrid hierarchical reinforcement learning for online guidance and navigation with partial observability, Neurocomputing 331 (2019) 443–457.

[18] H. Li, Q. Zhang, D. Zhao, Deep reinforcement learning-based automatic exploration for navigation in unknown environment, IEEE Transactions on Neural Networks and Learning Systems,doi:10.1109/TNNLS.2019.292786.

[19] K. Shao, D. Zhao, N. Li, Y. Zhu, Learning battles in vizdoom via deep reinforcement learning, in: IEEE Conference on Computational Intelligence and Games, 2018, pp. 1–4.

[20] K. Shao, D. Zhao, Y. Zhu, Q. Zhang, Visual navigation with actor-critic deep reinforcement learning, in: 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–6.

[21] K. Shao, Z. Yuanheng, Z. Dongbin, Starcraft micromanagement with reinforcement learning and curriculum transfer learning, IEEE Transactions on Emerging Topics in Computational Intelligence 3 (1) (2019) 73–84.

[22] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, J. Malik, Cognitive mapping and planning for visual navigation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2616–2625.

[23] D. Li, D. Zhao, Q. Zhang, Y. Zhuang, B. Wang, Graph attention memory for visual navigation, in: arXiv preprint, 2019. arXiv:1905.13315.

[24] N. Savinov, A. Dosovitskiy, V. Koltun, Semi-parametric topological memory for navigation, in: arXiv preprint, 2018. arXiv:1803.00653.

[25] D. W. Ko, C. Yi, I. H. Suh, Semantic mapping and navigation: A bayesian approach, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2013, pp. 2630–2636.

[26] S. Bataineh, A. Bahillo, L. Díez, E. Onieva, I. Bataineh, Conditional random field-based offline map matching for indoor environments, Sensors 16 (8) (2016) 1302.

[27] M. Ren, H. Guo, J. Shi, J. Meng, Indoor pedestrian navigation based on conditional random field algorithm, Micromachines 8 (11) (2017) 320.

[28] S. Wang, S. Fidler, R. Urtasun, Lost shopping! monocular localization in large indoor spaces, in: IEEE International Conference on Computer Vision, 2015, pp. 2695–2703.

[29] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: IEEE International Joint Conference on Neural Networks, 2005, pp. 729–734.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, in: IEEE Transactions on Neural Networks, 2009, pp. 61–80.

[31] Y. Lu, Y. Chen, D. Zhao, J. Chen, Graph-fcn for image semantic segmentation, in: International Symposium on Neural Networks, 2019, pp. 97–105.

[32] Y. Lu, Y. Chen, D. Zhao, B. Liu, Z. Lai, J. Chen, CNN-G: convolutional neural network combined with graph for image segmentation with theoretical analysis, IEEE Transactions on Cognitive and Developmental Systems,doi:10.1109/TCDS.2020.2998497.

[33] K. Yoon, R. Liao, Y. Xiong, L. Zhang, E. Fetaya, R. Urtasun, R. S. Zemel, X. Pitkow, Inference in probabilistic graphical models by graph neural networks, in: International Conference on Learning Representations Workshop, 2018.

[34] M. Qu, Y. Bengio, J. Tang, Gmnn: Graph markov neural networks, in: International Conference on Machine Learning, 2019, pp. 5241–5250.

[35] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, P. Zhokhov, Openai baselines, https://github.com/openai/baselines (2017).

[36] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[37] A. Popescul, L. H. Ungar, Statistical relational learning for link prediction, in: IJCAI Workshop on Learning Statistical Models from Relational Data, 2003.

[38] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. Shamma, M. Bernstein, F. F. Li, Visual genome: Connecting language and vision using crowdsourced dense image annotations, International Journal of Computer Vision 123 (2016) 32–73.

[39] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532–1543.

[40] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[41] M. M. Thao, C. M. Tien, N. D. Quang, Ai2thor-objdetect-yolov3, https://github.com/thaomm/ai2thor-objdetect-yolov3 (2018).

[42] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, Maximum entropy inverse reinforcement learning, in: National Conference on Artificial Intelligence, 2008, pp. 1433–1438.

[43] H. J. Kappen, Optimal control theory and the linear bellman equation, Neural Networks (2011) 363–387.

[44] M. Toussaint, Robot trajectory optimization using approximate inference, in: International Conference on Machine Learning, 2009, pp. 1049–1056.

[45] S. Levine, Reinforcement learning and control as probabilistic inference: Tutorial and review, in: arXiv preprint, 2018. arXiv:1805.00909.

[46] B. D. Ziebart, J. A. Bagnell, A. K. Dey, Modeling interaction via the principle of maximum causal entropy, in: International Conference on Machine Learning, 2010, pp. 1255–1262.