End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile Robots

Haobin Shi, Lin Shi, Meng Xu, and Kao-Shing Hwang, *Senior Member, IEEE*

***Abstract*—In this article, we develop a navigation strategy based on deep reinforcement learning (DRL) for mobile robots. Because of the large difference between simulation and reality, most of the trained DRL models cannot be directly migrated into real robots. Moreover, how to explore in a sparsely rewarded environment is also a long-standing problem of DRL. This article proposes an end-to-end navigation planner that translates sparse laser ranging results into movement actions. Using this highly abstract data as input, agents trained by simulation can be extended to the real scene for practical application. For map-less navigation across obstacles and traps, it is difficult to reach the target via random exploration. Curiosity is used to encourage agents to explore the state of an environment that has not been visited and as an additional reward for exploring behavior. The agent relies on the self-supervised model to predict the next state, based on the current state and the executed action. The prediction error is used as a measure of curiosity. The experimental results demonstrate that without any manual design features and previous demonstrations, the proposed method accomplishes map-less navigation in complex environments. Through a reward signal that is enhanced by intrinsic motivation, the agent explores more efficiently, and the learned strategy is more reliable.**

***Index Terms*—Curiosity-driven, deep reinforcement learning (DRL), mobile robots, navigation.**

I. Introduction

An AUTONOMOUS navigation system for mobile robots allows the robot to find a suitable path from the current position to the target position without colliding with obstacles. The classical approach to solve this problem uses a combination of algorithms, including simultaneous localization and mapping (SLAM) [1], path planning, and motion control [2], [3]. Most navigation solutions use highly accurate global maps [4], [5], which are constructed by passive programs with special exploration strategies or manually controlled robots for collecting environmental data. Therefore, expensive sensors and manual inspections are required to ensure that the derived obstacle map meets the requirements for path planning and positioning.

Recent advances in the deep learning models mean that learning-based approaches have become more popular. Deep reinforcement learning (DRL) continuously interacts with the environment to learn a mapping from the state to the action and relies on feature extraction of deep learning to address the problem of dimensionality disaster in complex scenes. DRL has achieved great success in many tasks, such as video games [6], simulation control agents [7], and robots [8], [9]. For the navigation task, several DRL studies have been proposed: using the feedforward architecture to learn the depth follow-up representation [10], using the two-stream Q-network for the navigation tasks in dynamic environments [11], using the auxiliary tasks to drive DRL training [12], [13], and using the asynchronous deep deterministic policy gradient algorithm for map-less navigation [14] and target-driven navigation [15]. In these works, the navigation strategy is learned directly from the original sensor input.

DRL algorithms create difficulties when they are used for the real scenes. Models based on reinforcement learning (RL) often need to constantly interact with the environment to achieve convergence. Unfortunately, training directly in the real world requires much time and robots can be damaged [16], so this method can often be costly. The model trained in the simulation is not always applicable to a real scene because of the large difference between the virtual world and reality [17], [18]. A related topic to bridge the reality gap is domain adaptation, which allows a machine learning model trained with samples from a source domain to generalize to a target domain [19], [20]. Some of these studies are based on image-conditioned generative adversarial networks [21]. Moreover, the way in which agents explore an environment of the sparse rewards is also a long-standing problem for DRL, particularly for the map-less navigation tasks in complex scenes. The mobile robot must avoid obstacles and do not trap into corners, but the rewards are usually sparsely distributed in the environment and there may be only one target location. Accidentally reaching the target state by a random method such as ε-greedy is likely to be futile

Manuscript received May 1, 2019; revised June 20, 2019, July 11, 2019, and August 8, 2019; accepted August 11, 2019. Date of publication August 20, 2019; date of current version January 17, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61976178, in part by the Shaanxi Province Key Research and Development Program of China under Grant 2018GY-187, and in part by the Seed Foundation of Innovation and Creation for Graduate Students in Northwestern Polytechnical University under Grant ZZ2018169. Paper no. TII-19-1706. *(Corresponding author: Kao-Shing Hwang.)*

H. Shi and M. Xu are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: shihaobin@nwpu.edu.cn; menghsu@nwpu.edu.cn).

L. Shi is with the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: slsl@nwpu.edu.cn).

K.-S. Hwang is with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung 80424, Taiwan, and also with the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Kaohsiung 80708, Taiwan (e-mail: hwang@ccu.edu.tw).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2936167

for difficult tasks. Therefore, we suspect that in more complex environments, these methods [10]–[15] of exploration are not sufficient to efficiently learn navigation strategies.

In order to reduce the agent's observable difference between the simulation environment and the real scene, this article proposes an end-to-end DRL navigation planner that uses sparse laser ranging results as an input. The visual sensory information tends to have ambient noises, such as the frequent changes of lighting, illumination, color contrast, and so on. Therefore, sparse laser ranging is used as a sensing input so that the DRL model can learn more accurately the mapping between the distance of the obstacle and the movement command. In contrast to a visual-based approach, this precise data allow agents trained under the simulation to migrate directly to the real scene without adjusting the training model or special processing of the input image. A learning-based agent does not require highly accurate sensors to construct a map of the environment to complete a navigational task, so the navigation process is simpler, and the cost is reduced. This is an advantage over the SLAM-based navigation.

Complex navigation tasks are often not completed because there are few external rewards. Recent studies have proposed various methods to obtain an intrinsic reward, including Bayesian surprise [22], information gain [23], measures by novelty of observed states [24], and measures by the agent's learning progress in predicting action outcomes [25], [26]. An exploration strategy that uses curiosity as an intrinsic reward effectively addresses the problem [27], [28]. In an unfamiliar environment for which there is no global map information, curiosity drives agents to explore the states that have not been visited, even if there are no clear rewards or goals. The future state is predicted based on the current state and action. The prediction error is used as the criterion to determine the familiarity of the current state and as the agent's internal reward in an environment where there are few external rewards. To avoid the complexity of modeling original observations, the prediction process uses a feature space that is encoded by the environmental state. This strategy allows agents to quickly become familiar with the environment, to learn new strategies to accomplish tasks (such as avoiding obstacles of different shapes), and to cover as many maps as possible in a maze-like environment in order to reach the target.

The main contributions of this article are as follows:

1) An end-to-end navigation method that uses DRL is proposed. This method addresses the problem of the sparse rewards by introducing curiosity and allows agents to learn more efficiently in complex environments.
2) The method is a low-cost solution for the navigational tasks because there is no need for highly accurate sensors to build maps.
3) Sparse laser ranging results are used as the input to reduce the difference between the real world and the physical simulator. The model trained under the simulation is applicable to real mobile robots.

The remainder of this article is organized as follows. Section II describes the background for the proposed method, in terms of the asynchronous advantage actor-critic (A3C) algorithm and its improved version, Nav A3C. Section III describes the curiosity-driven exploration strategy and the navigation method that uses DRL. The simulated and real experiments are described in Section IV. Finally, Section V concludes this article.

II. Background

A. A3C Algorithm

In addition to value-based methods, there is a large class of policy-based methods for traditional RL. For a task that involves RL, a policy must be learned. Value-based methods use an indirect method, whereby a policy is devised by learning a value function or an action-value function. A policy-based method directly models and learns a policy. It is also known as a policy optimization model.

These two methods have individual advantages and disadvantages. The Actor-Critic (AC) algorithm [29] combines the advantages of both. The AC algorithm combines policy-based and value-based methods. The AC algorithm updates the weights of the neural network using two loss functions. The loss function for the policy is given as

$$f_\pi(\theta) = \log\pi(a_t|s_t;\theta)\left(R_t - V(s_t;\theta_v)\right) \tag{1}$$

$$R_t = \sum_{i=0}^{k-1}\gamma^i r_{t+i} + \gamma^k V(s_{t+k};\theta_v). \tag{2}$$

In (2), the time step is t. The operation continues until the final environmental state is reached and the reward is accumulated by the agent. The reward is gradually discounted by the factor γ over several time steps. In addition, the value of k is determined by $t_{\max}$.

The loss function for the value function is given as

$$f_v(\theta) = \left(R_t - V(s_t;\theta_v)\right)^2. \tag{3}$$

The A3C algorithm [30] based on the AC algorithm updates the weights of the neural network by a multiagent with an asynchronous gradient decent. In the A3C algorithm, each agent interacts with an independent environment. Therefore, the environmental state observed by each agent, the actions taken, and the rewards obtained are different from those of other parallel agents, thereby reducing the relevance of the training set.

Similarly to other nonsynchronous methods, the weights of the neural network are stored in a central parameter server that is shared by each agent, as shown in Fig. 1. Each agent, after every t actions or reaching the environmental state of the terminal, uses the loss function of the strategy, as shown in (1), and the loss function of the value function, as shown in (3). Each weight of the network calculates the gradient and sends it to the server. The server updates the weight stored in the server by collecting the gradient of each agent and using the rmsProp algorithm as shown in (4), where all operations are performed elementwise

$$\begin{aligned} g &= \alpha g + (1-\alpha)\nabla\theta^2 \\ \theta &\leftarrow \theta - \eta\nabla\theta/\sqrt{g+\varepsilon} \end{aligned} \tag{4}$$

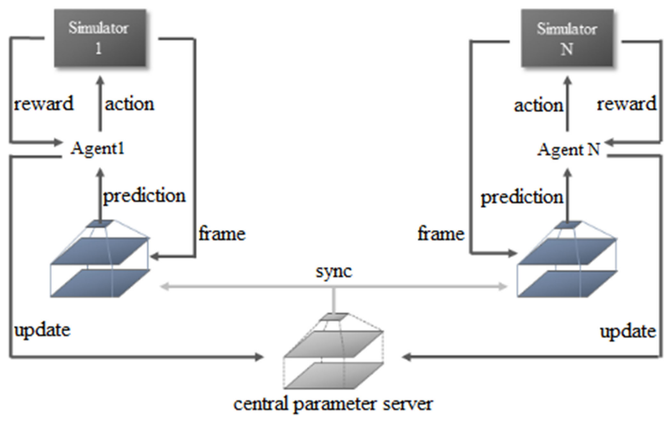


Fig. 1. Architecture for the A3C algorithm.

where g is the moving average of the elementwise squared gradients, $0 \leq \alpha \leq 1$ is a hyperparameter, η is the learning rate, and ε is the constant added to maintain numerical stability. After each update, the server transmits the updated weights to each agent to ensure that each agent has the same policy. The A3C algorithm is shown in Fig. 1.

B. Nav A3C Algorithm

The method described in this article is realized on the Nav A3C algorithm [12]. The Nav A3C algorithm is based on the A3C algorithm and adjusts the network architecture. Two layers of long short-term memory (LSTM) [31] are added between the convolutional neural network (CNN) and the output layer to distinguish an ambiguous environmental state.

The observation information for the Nav A3C algorithm includes the action that is taken by the stochastic strategy in the previous time step, the remuneration for the action that is taken in the previous time step, and the speed of the agent. In terms of the Nav A3C algorithm, the agent's observations include the current state, past actions, and past rewards.

The Nav A3C algorithm receives the strategy for the action that is taken in the previous time step (as well as the agent's speed information) from the second layer of recursion and the reward for the previous time step from the first layer of recursion. The Nav A3C algorithm assumes that the first layer of recursion establishes a correlation between the reward that is received by the agent and the environmental observations. This correlation allows the strategies and value functions for the second level of recursion to be approximated.

III. Method

A. Problem Statement

This article is to provide navigation strategies for mobile robots in a map-less environment. We developed a motion planning model based on DRL. This end-to-end model transfers directly the inputs of the sensor observations and the relative position of the target to the commands of robot's movement.

Two problems inherent in DRL and the same challenges exist for the navigation tasks. Learning policies to achieve target tasks by maximizing the reward that is provided by the environment is the basis of the RL algorithm, but in many cases these external rewards are extremely sparse, so it is difficult to construct an appropriate reward function. The agent receives an enhancement that updates its policy only if the agent successfully reaches a prespecified target state. However, during complex tasks, it is difficult for an agent to achieve a target state by accidental means. On the other hand, the large difference between virtual and real often causes the model trained under the simulation to be unsuitable for real robots.

In order for the model to cope with a more complex environment and to be applicable in the real world, this article uses intrinsic curiosity to allow agents to explore the environment more effectively and as an additional reward in an environment where the rewards are sparse. Sparse laser ranging results are also used as the highly abstracted inputs, so agent strategies that are learned by the simulation are effective in the real world.

B. Curiosity-Driven Exploration

The rewards in the real world are sparse, so most RL algorithms are not applicable to some complex tasks. The navigational tasks are difficult in a labyrinthine environment or one in which there are many obstacles. One solution is to allow the agents to create the appropriate rewards themselves, so the rewards are more abundant, which increases the efficiency of learning. Specifically, similarly to animal curiosity, observing something new or exploring an unfamiliar environment is rewarded. For agents that use RL, curiosity can serve as an intrinsic reward signal that enables them to explore their environment more efficiently and to avoid becoming trapped by a lack of feedback. This navigational strategy allows agents to continually explore unfamiliar environments and acquire new knowledge to accomplish tasks (such as avoiding obstacles of different shapes) and allows agents to cover more unknown maps in complex environments to find hidden path to reach targets.

The idea of using curiosity as an intrinsic reward is to predict the future state s_{t+1} based on the current state s_t and action a_t, and then calculate the error from the actual state. By providing the agent with a predictive error of the state, as a criterion for judging the current state familiarity, it also acts as an agent's internal reward in the sparse reward environment, thereby encouraging the agent to explore the "novel" state.

However, it is very difficult to directly predict the original observations. When an image is used as a state space, an agent cannot construct an appropriate correct model to predict the pixels. Ideally, the model predicts only those changes in the environment that may be caused by the agent's behavior or affect the agent, while ignoring the remaining changes. A common mechanism must be used to represent original observations with environmental features. The original states $(s_t,\ s_{t+1})$ are encoded into the corresponding features $(\phi(s_t),\ \phi(s_{t+1}))$. The inverse model is trained to predict a_t using state features $(\phi(s_t),\ \phi(s_{t+1}))$. Then, a_t and $\phi(s_t)$ are passed to the forward model, which is used to predict the feature representation

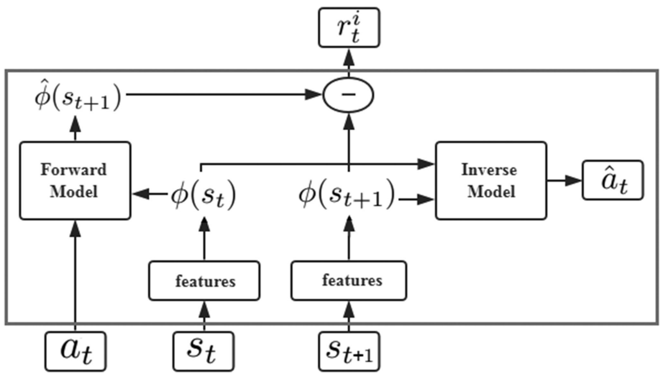


Fig. 2. Architecture for the ICM.

$\hat{\phi}(s_{t+1})$ for the state (s_{t+1}) and compare it with the real feature $(\phi(s_t))$. The prediction error is used as an intrinsic curiosity reward to encourage explorative behavior in agents. This model is called an intrinsic curiosity module (ICM), as shown in Fig. 2.

For the inverse model, the goal is to learn the function g as follows:

$$\hat{a}_t = g\left(s_t, s_{t+1}; \theta_I\right) \tag{5}$$

where $\hat{a}_t$ is the predicted value of the agent's action (a_t) and θ_I is the neural network parameter that is trained to be optimized as follows:

$$\min_{\theta_I} L_I(\hat{a}_t, a_t) \tag{6}$$

where L_I is the loss function that evaluates the error between the estimate $\hat{a}_t$ and the real action a_t. Since the action space is discrete, the output of g is a soft-max distribution across all possible actions. Cross entropy is mainly used to measure the discrepancy between two probability distributions. Therefore, the loss function L_I is set in the form of cross entropy as follows:

$$L_I\left(a_t, \hat{a}_t\right) = \sum_{i=1}^{n} -P(a_{t_i})\ln q(\hat{a}_{t_i}) \tag{7}$$

where n is the size of the action space, $P(a_{t_i})$ is whether the agent chooses the ith action in the actual situation, and $q(\hat{a}_{t_i})$ is the probability of the agent choosing the ith action in the prediction result.

As there is no incentive for this inverse model to encode any environmental states that are not influenced by the agent's actions, our model will be robust to uncontrollable noise signals.

Another neural network is trained for the forward model, which uses $\phi(s_t)$ and a_t as inputs to predict the feature encoding at the next time step $t+1$

$$\hat{\phi}(s_{t+1}) = f\left(\phi(s_t), a_t, \theta_F\right) \tag{8}$$

where $\hat{\phi}(s_{t+1})$ is the predicted value of the state features $(\phi(s_{t+1}))$ and θ_F is the neural network parameter that is trained to be optimized. The training process is performed by minimizing the loss function L_F as follows:

$$L_F\left(\phi(s_t), \hat{\phi}(s_{t+1})\right) = \frac{1}{2}\left\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\right\|_2^2. \tag{9}$$

The intrinsic reward signal r^i_t measured by the prediction error of the feature encoding is calculated as

$$r^i_t = \frac{\eta}{2}\left\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\right\|_2^2 \tag{10}$$

where $\eta > 0$ is the scale factor used to adjust the intrinsic reward.

The agent is provided with an appropriate intrinsic reward signal by optimizing the inverse and forward dynamic losses in (6) and (9). The goal of the inverse model is to generate a feature space that only encodes the original observations that affects the action of agents and the forward model uses this feature space to encode the original observations for a feasible state prediction.

C. Navigation Strategy Based on DRL

A lack of key information and sparse rewards mean that the navigational tasks in a labyrinthine environment using algorithms such as A3C are not successful enough. Mirowski *et al.* [12] proposed a DRL navigational strategy for complex maze environments using an improved version of A3C and auxiliary tasks, which significantly improves the performance. However, these results are for three-dimensional (3-D) games and use auxiliary tasks such as depth prediction and changes in pixel values between frames. These methods work well in a gaming environment but do not accommodate the real world, where depth measurements and color values are highly susceptible to noise. The difference between the simulation and reality also means that these trained models are not applicable to real environments. In order to more efficiently complete a navigational task in a complex structure, this article proposes a new DRL model. The prediction error in the feature space provides the agent with an intrinsic reward signal instead of predicting depth as an auxiliary task. The model architecture is shown in Fig. 3.

Environments usually contain dynamic elements such as random target locations in the same training environment, requiring agents to use memory at different time scales. Moreover, in order to support more powerful navigation capabilities, it is necessary to expand the scope of observation of agents. The ICM A3C adjusts the network structure by adding two layers of LSTM after the convolutional encoder, so the agent can remember different environmental observations. The previous rewards and action information are also used as an input to provide a more complete account of the conditions. The first layer of LSTM accepts the previous rewards and observations. The aim is to establish an association between the original observations and rewards and transmit this to the next level of LSTM. The previous action is passed directly to the LSTM of the second layer.

The network structure of the model is as follows. The first convolutional layer has 16 filters, each with a kernel size 8×8 and a stride of 4×4. Then, the second convolutional layer has 32 filters, each with a kernel size 4×4 and a stride of 2×2. The model also adds two layers of LSTM with 64 and 256 hidden neural units, respectively. Finally, connected to two separate output layers including softmax layer and linear layer, respectively, generating policy π and value function V. The ICM includes an inverse model as well as a forward model. The

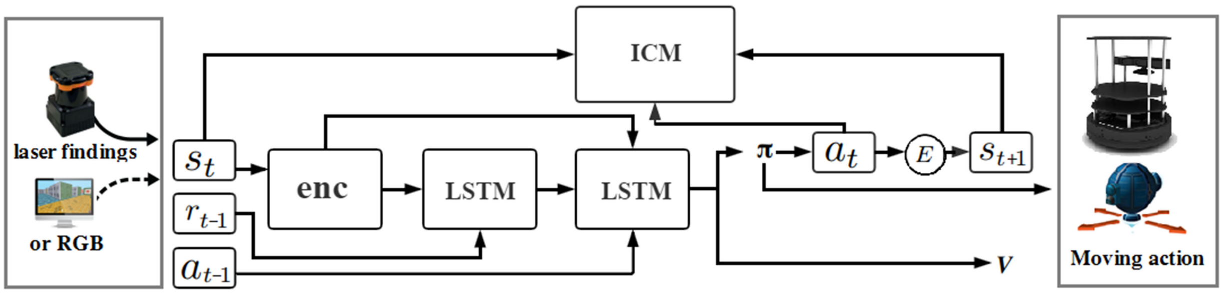


Fig. 3. Architecture for the ICM A3C model.

inverse model consists of four convolutional layers, each with 32 filters. The kernel size of the filter is 3×3 and the stride is 2×2. The environmental state is encoded as a feature representation $\phi(s_t)$. Those features are then concatenated and put through a fully connected layer with 256 units, and finally the predictive action $\hat{a}_t$ is output by the softmax layer. The forward model inputs the real features $\phi(s_t)$ and the action a_t into the fully connected layer of 256 units, and then outputs the predicted features $\hat{\phi}(s_{t+1})$. When the original observations of different experiments change, we will adjust the network structure (such as using laser ranging as input), but the overall architecture is the same as Fig. 3.

The reward function is divided into two components; the agent interacting with the environment obtains the extrinsic reward r_t^e and the intrinsic reward r_t^i, as shown in (9). In Experiment 1, r_t^e is set by the public experimental platform and will be introduced in the following section. The r_t^e of other experiments is calculated as follows:

$$r_t^e = \begin{cases} r_{\text{goal}}, & d_t < d_x \\ r_{\text{collision}}, & d_o < d_y \\ d_{t-1} - d_t \end{cases} \tag{11}$$

where r_{goal} is the positive reward obtained by the robot successfully reaching the target position, and $r_{\text{collision}}$ is the negative reward generated when the robot collides with the obstacle during the exploration. d_t is the distance between the robot and the target at time *t*, and d_o is the distance from the obstacle. d_x and d_y are used to check if the target is reached or a collision occurs. In order for the robot to learn to approach the target, we use the distance change $(d_{t-1} - d_t)$ at different time steps as a reward for other situations.

The policy $\pi(s_t; \theta_p)$ is represented by a deep neural network with the parameter θ_p. The agent relies on the $a_t \sim \pi(s_t; \theta_p)$ to select the actions in the state. θ_p is trained to optimize the policy to maximize the sum $r_t = r_t^e + r_t^i$ of the intrinsic reward and the extrinsic reward

$$\max_{\theta_p} E_{\pi(s_t;\theta_p)} \left[\sum\nolimits_t r_t \right]. \tag{12}$$

Finally, the combination of (6), (9), and (12) is the problem that needs to be optimized overall

$$\min_{\theta_p,\theta_I,\theta_F} \left[-\lambda E_{\pi(s_t;\theta_p)} \left[\sum\nolimits_t r_t \right] + (1-\beta) L_I + \beta L_F \right] \tag{13}$$

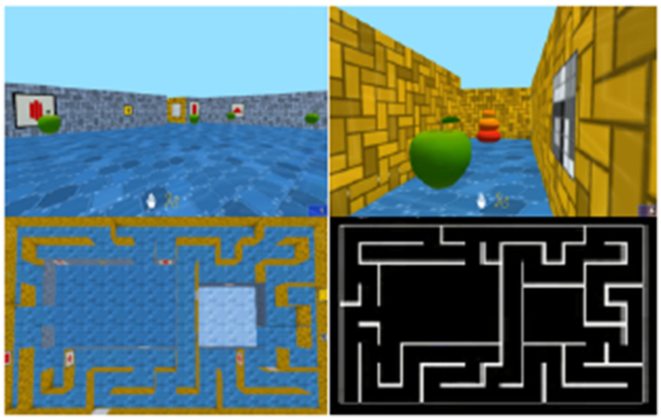


Fig. 4. First perspective and top view of a complex maze.

where $\lambda > 0$ is used to adjust the weight of the policy gradient loss for the curiosity reward, and $0 \le \beta \le 1$ is used to adjust the weight between the forward model loss and the inverse model loss.

IV. Experiments and Results

A. Simulation Experiment—First-Person 3-D Maze

1) Design of the 3-D Maze Experiment: In order to verify the effectiveness of the proposed algorithm, a first-person 3-D maze experiment called labyrinth was performed. The experimental environment is shown in Fig. 4. In order to ensure that the experimental environment is not unduly favorable to the proposed algorithm, the public experimental platform—DeepMind Lab [32] was used to test the performance of the algorithm.

DeepMind Lab is used to study how automated agents learn to perform complex tasks in large, partially visual, and visually diverse environments. It is specifically designed to study general artificial intelligence and machine learning systems. In this experiment, the agent's action space includes eight actions: the agent can rotate in small increments (left or right), accelerate in four directions horizontally, or induce rotational acceleration while moving (left or right). At the beginning of each episode, the agent randomly appears somewhere in the maze and using the 84×84 red green blue (RGB) observation as the input of the neural network explores the target position in the labyrinth environment. A few green apples are dispersed in the map as

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Value
$t_{\max}$	50
Discount	0.95
Learning Rate	0.0003
RMSProp Decay	0.99
RMSProp Epsilon	0.1
β	0.2
λ	0.1

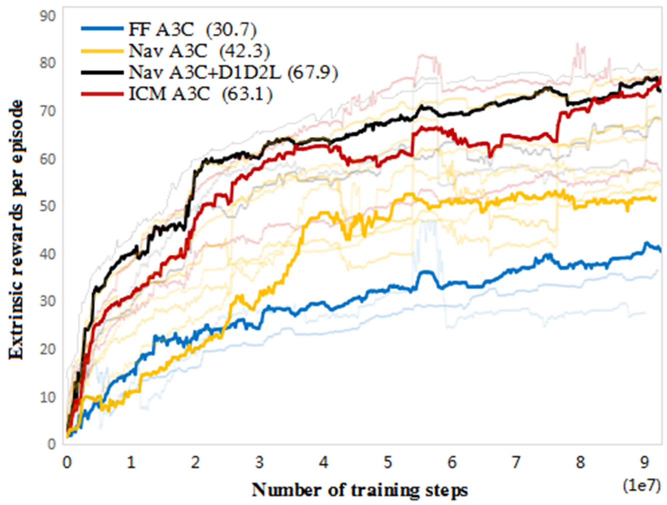


Fig. 5. Learning curve for different algorithms.

the extra sparse rewards. Apples are worth one point and goals are worth ten points. The labyrinth is 9 × 15 and episodes last for 10 800 time steps to ensure that the agent has ample time to find the goal.

The DRL model, which uses curiosity as an intrinsic reward (see Section III), is evaluated by training in the experimental environment. The algorithms to be compared are the feedforward model (FFA3C), the stacked LSTM version with speed, the previous rewards and actions as an input (Nav A3C), and an improved version that includes depth prediction and loop closure prediction (NavA3C + D1D2L) [12]. In order to eliminate the contingency of training, eight experiments were performed using each algorithm with random sampling hyperparameters and the mean scores for the three best-performing agents were plotted. The experimental parameters are given in Table I, where β and λ are the parameters in (13). All the agents were trained asynchronously with eight workers.

2) Results and Analysis: Fig. 5 shows the learning curves for the different algorithms. The horizontal axis shows the number of episodes and the vertical axis shows the rewards that the agent accumulates during each episode. In order to fairly evaluate the performance of different algorithms, all the rewards in the learning curves are the extrinsic rewards. The intrinsic rewards are only used to improve the agent's exploratory ability, not included in the experimental statistics. The lowest average score for FF A3C is 30.7. Nav A3Cs has an average score of 42.3. Nav A3C + D1D2L and the proposed ICM A3C obviously perform better with average scores of 67.9 and 63.1, respectively. Only when the navigation task is successfully completed is a score obtained, so the score reflects the performance of the algorithm.

The results show that it is very difficult for a standard A3C to achieve good results in the navigational tasks in complex environments and environments in which the rewards are sparse.

For navigation, an agent requires the current image information so Nav A3C adjusts the network structure through LSTM and uses the last action information and reward as an additional input. This gives more complete information and the agent has a memory, so the training has a better effect. However, Nav A3C does not address the problem of the sparse rewards. Nav A3C + D1D2L maximize the cumulative reward, supplemented by multiple auxiliary tasks including depth prediction and loop closure prediction. During the network training process, both the cumulative reward maximization and the auxiliary task's loss function are reduced. This method further improves data efficiency and increases performance. Agents explore more effectively in a sparsely rewarded environment by adding the intrinsic rewards and using state prediction errors as the reward signals. The experimental results show that the proposed algorithm performs similarly to Nav A3C + D1D2L and better than standard A3C and Nav A3C, but the proposed method is more versatile. Since Nav A3C + D1D2L must use image depth information, it is limited in many applications, such as the next navigation experiment that uses laser ranging as an input. Moreover, agent trained in the simulation environment with image as an input can hardly migrate directly to the real world.

B. Simulation Experiment—Gazebo

1) Design of the Simulation Experiment: A large number of training sessions and repeated interactions with the environment are time consuming and can cause damage to real robots, just as there is a risk of collisions with obstacles in the navigation tasks, so most DRL training is implemented in the virtual environments. However, the large difference between the virtual environment and reality means that the training model is not applicable to a real environment, especially if tasks are vision based. Therefore, this article uses sparse laser ranging results as an input to reduce the agent's observable difference between the virtual world and reality.

We assume that discretization of the action space can make agents more robust to disturbance in scene and noise of robot control system, while continuous space is more sensitive. In this case, the discrete action is more suitable for wheeled mobile robots because some random errors are inevitable, such as slipping or sliding in operations. In this experiment, the agent's action space includes five actions: moving forward ($v = 0.3$ m/s, $w = 0$ rad/s), turning left or right ($v = 0.2$ m/s, $w = \pm 0.3$ rad/s), or turning left or right substantially ($v = 0.1$ m/s, $w = \pm 0.8$ rad/s).

The virtual environment in *gazebo* is used to simulate real robotic navigation experiments. Two indoor environments were constructed to compare the performance of different algorithms for the navigation tasks, as shown in Fig. 6.

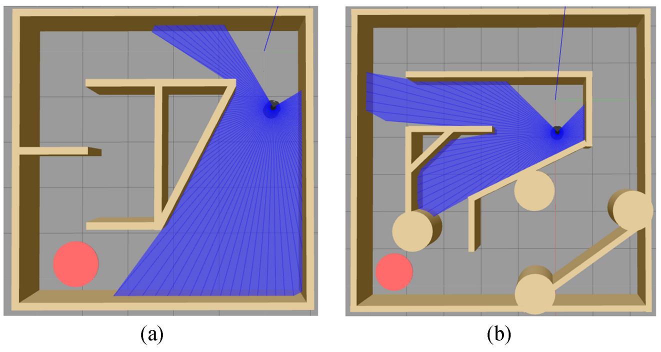


Fig. 6. Experimental environment. (a) Simulation *Room-1.* (b) Simulation *Room-2.*

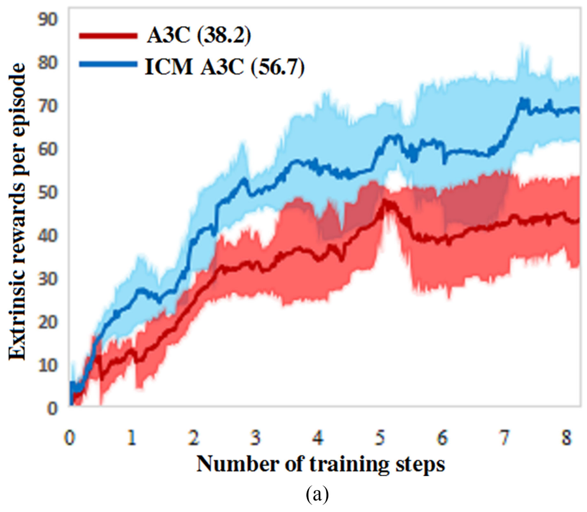
A Turtlebot with a LIDAR sensor is the agent for the navigation experiment. Experimental environment 1 has a sloping wall and turns of different angles. Experimental environment 2 is a more complicated labyrinth with different wall shapes and some narrow tracks. The Turtlebot uses sparse 70-D laser ranging results and its position relative to the target to complete the navigational tasks without collisions. During each episode, the target location is randomly initialized throughout the region and does not coincide with the positions of obstacles. The experimental parameters are the same as those listed in Table I.

2) Results and Analysis: The experimental results for the two environments are plotted in Fig. 7. We performed eight experiments using random sampling hyperparameters and selected the best three times for plot comparison. The blue line represents the ICM A3C algorithm, the red line represents the normal A3C algorithm, and the darker line represents the average value. The shaded area represents the average $\pm$ the standard error. No adjustments are made to the random seed.

In *Room-1*, the A3C algorithm achieves an average reward of 38.2 and the ICM A3C achieves an average reward of 56.7. *Room-2* is more complex and challenging, and highlights the advantages of the proposed algorithm. The average reward for the A3C algorithm is 23.4 and for the ICM A3C is 51.3.

Room-1 has a relatively small area. The only obstacles are only walls at different angles so the agent must only learn to turn and avoid collisions. The navigational task for the agent is not complicated in this environment. In a simple environment, the proposed ICM A3C algorithm explores the environment more effectively than the standard A3C algorithm. Even in the later stages of training, the standard A3C does not successfully complete the navigational task for some difficult target positions that were randomly generated, such as when the target is distant or hidden in a dead corner close to the wall.

Room-2 is larger and more complex. Cylindrical objects are placed around the wall to give a more varied range of obstacles and restrict the range of access to key locations. In a more challenging environment, an agent must learn more mature navigational strategies. The results of Experiment 2 show that both the ICM A3C algorithm and the normal A3C algorithm require a longer time to explore and learn the navigational strategy than is required for Environment 1. When the external



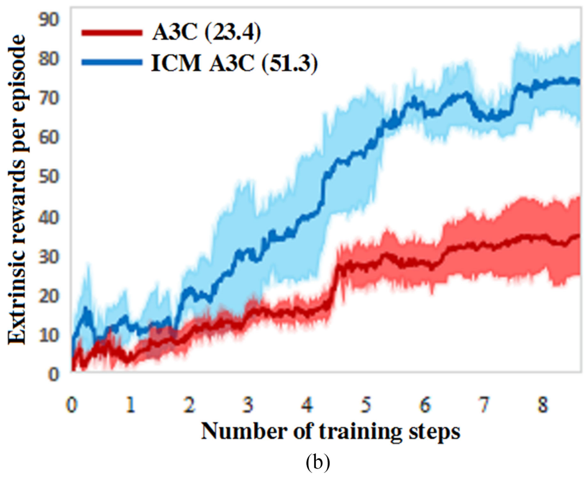
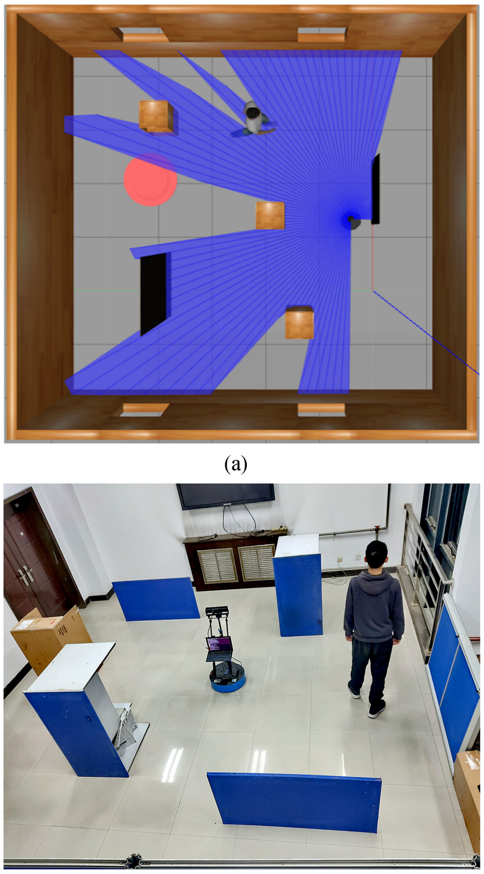


Fig. 7. Learning curves for different algorithms. (a) Learning curve in Simulation *Room-1.* (b) Learning curve in Simulation *Room-2.*

rewards are sparse, as in the case of locally observable navigational tasks, the agent finds it difficult to reach the target and obtain a reward without colliding with obstacles so the intrinsic reward is crucial. The use of curiosity as an intrinsic reward encourages agents to explore a new state so they continually seek new knowledge and data efficiency is improved during the exploration process. Exploring a new state also prevents the agent from becoming trapped in a fixed location, such as a dead space in an experimental environment, so it is still possible to reach the target and complete the navigational task in a complex environment.

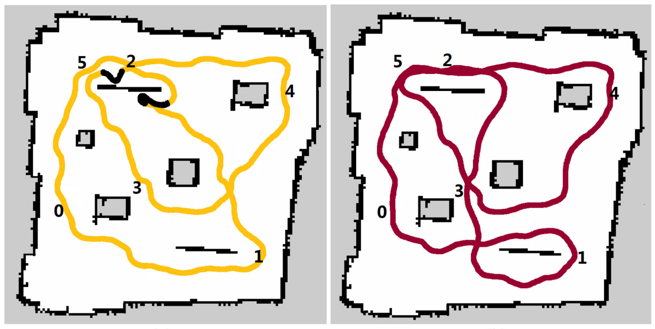
C. Real-World Experiment

1) Design of the Real-World Experiment: In order to test the effect of the algorithm for the real scenes and to determine whether this method has a practical value, a more realistic environment for pretraining was constructed in gazebo, which includes walls, humans, desks, and partitions that render the environmental more complex. In practice, an experimental site with a similar size and structure to the simulation environment is used, as shown in Fig. 8.


(a)

(b)

Fig. 8. Experimental environments. (a) Simulation environment. (b) Real environment.



(a) (b)

Fig. 9. Path trajectory for different algorithms. (a) A3C model. (b) ICM A3C model.

An agent that was trained for about 80 000 episodes in the simulation environment was assigned the task of navigating in the real scene. A wheeled robot with a laser ranging sensor was used for the experiment. Six sequential target points were established as tasks for the test. The robots must visit these subgoals in order to avoid obstacles. The A3C model and the ICM A3C model were tested. The results for the navigational path are shown in Fig. 9. The time that each algorithm requires for the navigational task and the cumulative distance that is traveled when all tasks are completed is shown in Fig. 10.

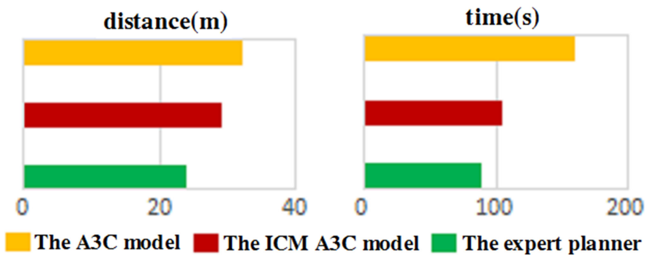


Fig. 10. Time required for the navigational task and the distance traveled for different algorithms.

2) Results and Analysis: Fig. 9 shows that the robot that uses the ICM A3C model successfully avoids obstacles and reaches the target positions in turn and the path trajectory is smoother. It is worth noting that the standard A3C model fails to complete the task three times because it collides with obstacles. In order to prevent damage to the robot in this experiment, there are no real collisions, but when the robot is very close to an obstacle and there is no steering action, the experiment is manually terminated. Manual intervention is marked with black in the path trajectory.

The experimental results show that the ICM A3C model executes the navigational task more stably. The A3C performs poorly because it does not handle data efficiently, so it cannot learn sufficient navigational strategies after the same number of training iterations under the simulation environment. Agents that use the standard A3C algorithm also occasionally collide during navigation. This often occurs during training in the experimental environment when the obstacle is in the middle of the path between the agent and the target. Standard exploration strategies may lack curiosity-driven intrinsic rewards so agents often only learn to approach the target to obtain the external rewards. However, when the target location and the agent are very close, but separated by a wall or obstacles, the agent eventually falls into a partial deadlock from which it is difficult to find the target using standard random exploration strategies. The intrinsic curiosity reward in the ICM A3C model allows agents to explore new areas after multiple failures in familiar locations, in order to discover alternative paths to the target. It is seen that the proposed algorithm performs better in the navigational tasks than the other tested algorithms under the same training conditions.

However, agents using DRL still have some problems. A lack of global map information means that the planning trajectory for the end-to-end method is more circuitous so a greater distance is traveled. This is a subject for future study.

For applications where accuracy and safety are critical, such as driverless operation or an environment with many obstacles and hidden paths, SLAM technology must be used with traditional path planning algorithms. This article does not seek to replace traditional solutions. It proposes a low-cost solution for relatively simple environments for which changes in map information are small and security requirements are low. The learning planner does not require a highly accurate map so laser radar can be replaced with a low-cost distance sensor. DRL is very successful in many fields because it offers strong learning potential, end-to-end direct control, and its ability to operate without manual sample data. This is a meaningful initial attempt to use the technique for a navigational task for mobile robots, but there is still much room for improvement.

V. Conclusion

In this article, we proposed a map-less navigational strategy for complex indoor scenes. Based on a conventional A3C algorithm, an ICM A3C model was proposed. The ICM A3C model uses intrinsic curiosity to address the problem of the sparse rewards. By adding two layers of LSTM into the network structure, the model achieves a memory function. The previous action and the previous reward were used as an input to provide more complete information. Using sparse laser ranging as an input to reduce the difference between the simulation environment and reality, the trained model was applicable to the robots in a real environment. The experiments demonstrated that the proposed method was more efficient and stable than a classical DRL algorithm. Neither does the end-to-end model require highly accurate obstacle maps and manual design, so it was a low-cost solution for the navigational tasks. Future study will pertain to path optimization and long-distance navigation.

References

[1] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[2] H. Shi, G. Sun, Y. Wang, and K. S. Hwang, "Adaptive image-based visual servoing with temporary loss of the visual signal," *IEEE Trans. Ind. Inform.*, vol. 15, no. 4, pp. 1956–1965, Aug. 2018.

[3] C. Linegar, W. Churchill, and P. Newman, "Made to measure: Bespoke landmarks for 24-hour, all-weather localisation with a camera," in *Proc. IEEE Int. Conf. Robot. Autom.*, Stockholm, Sweden, 2016, pp. 787–794.

[4] R. Sim and J. J. Little, "Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, 2006, pp. 2082–2089.

[5] M. Tomono, "3-D object map building using dense object models with sift-based recognition features," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, 2006, pp. 1885–1890.

[6] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[7] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[8] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep Q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 2829–2838.

[9] H. Shi, X. Li, K. S. Hwang, W. Pan, and G. Xu, "Decoupled visual servoing with fuzzy Q-learning," *IEEE Trans. Ind. Inform.*, vol. 14, no. 1, pp. 241–252, Jan. 2018.

[10] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, 2017, pp. 2371–2378.

[11] W. Yuanda, H. Haibo, and S. Changyin, "Learning to navigate through complex dynamic environment with modular deep reinforcement learning," *IEEE Trans. Games*, vol. 10, no. 4, pp. 400–412, Jun. 2018.

[12] P. Mirowski *et al.*, "Learning to navigate in complex environments," 2016, *arXiv:1611.03673*.

[13] M. Jaderberg *et al.*, "Reinforcement learning with unsupervised auxiliary tasks," 2016, *arXiv:1611.05397*.

[14] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 31–36.

[15] Y. Zhu *et al.*, "Target driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3357–3364.

[16] D. Kalashnikov *et al.*, "QT-Opt: Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. Conf. Robot Learn.*, 2018, pp. 1457–1464.

[17] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.

[18] F. Sadeghi and S. Levine, "(Cad) 2 RL: Real single-image flight without a single real image," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 34–44.

[19] A. Shrivastava *et al.*, "Learning from simulated and unsupervised images through adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 2242–2251.

[20] K. Bousmalis *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Brisbane, Qld., Australia, 2018, pp. 4243–4250.

[21] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[22] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation(1990–2010)," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 3, pp. 230–247, Sep. 2010.

[23] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. D. Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2016, pp. 6591–6598.

[24] T. Hester and P. Stone, "Intrinsically motivated model learning for developing curious robots," *Artif. Intell.*, vol. 247, pp. 170–186, Jun. 2017.

[25] M. B. Hafez, C. Weber, and S. Wermter, "Curiosity-driven exploration enhances motor skills of continuous actor-critic learner," in *Proc. IEEE Int. Conf. Develop. Learn. Epigenetic Robot.*, Lisbon, Portugal, 2017, pp. 39–46.

[26] M. Luciw, V. Graziano, M. Ring, and J. Schmidhuber, "Artificial curiosity with planning for autonomous perceptual and cognitive development," in *Proc. IEEE Int. Conf. Develop. Learn. Epigenetic Robot.*, Frankfurt, Germany, 2011, pp. 1–8.

[27] S. Still and D. Precup, "An information-theoretic approach to curiosity-driven reinforcement learning," *Theory Biosci.*, vol. 131, no. 3, pp. 139–148, Jul. 2012.

[28] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 2778–2787.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

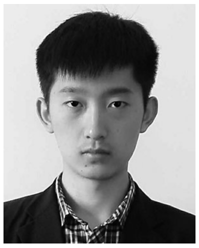
[30] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[31] B. Bakker, "Reinforcement learning with long short-term memory," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2001, pp. 1475–1482.

[32] C. Beattie *et al.*, "DeepMind lab," 2016, *arXiv:1612.03801*.

Haobin Shi received the Ph.D. degree in computer science and technology from the School of Computer Science, Northwestern Polytechnical University, Xi'an, China, in 2008.

He is an Associate Professor of Computer Science and Technology with the School of Computer Science, Northwestern Polytechnical University, and a Visiting Scholar with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan. He is the Director of the Chinese Association for Artificial Intelligence. His research interests include intelligent robots, decision support systems, artificial intelligence, multiagent systems, and machine learning.



Lin Shi received the B.S. degree in mechanical engineering from the School of Construction Machinery, Chang'an University, Xi'an, China, in 2016. He is currently working toward the master's degree in software engineering with the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China.

His research interests include mobile robot, intelligent control, and machine learning.

Meng Xu received the B.S. degree in software engineering from the School of Software and Microelectronics, Northwestern Polytechnical University, Xi'an, China, in 2017. He is currently working toward the master's degree in computer engineering with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China.

His research interests include mobile robot, intelligent control, and machine learning.

Kao-Shing Hwang (M'93–SM'09) received the M.M.E. and Ph.D. degrees in electrical and computer engineering from Northwestern University, Evanston, IL, USA, in 1989 and 1993, respectively.

He is a Professor with the Department of Electrical Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, and an Adjunct Professor with the Department of Healthcare Administration and Medical Informatic, Kaohsiung Medical University, Kaohsiung, Taiwan. He was with National Chung Cheng University, Taiwan, from 1993 to 2011. He was the Deputy Director of Computer Center during 1998 to 1999, the Chairman of the Department of Electrical Engineering during 2003 to 2006, and the Director of the Opti-Mechatronics Institute of the university during 2010 to 2011. His research interest includes methodologies and analysis for various intelligent robot systems, machine learning, embedded system design, and application specific integrated circuit (ASIC) design for robotic applications.

Dr. Hwang is a Fellow of the Institution of Engineering and Technology.