BND*-DDQN: Learn to Steer Autonomously through Deep Reinforcement Learning

Keyu Wu, Han Wang, Mahdi Abolfazli Esfahani, Shenghai Yuan

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

BND*-DDQN: Learn to Steer Autonomously through Deep Reinforcement Learning

Keyu Wu, Han Wang*, Mahdi Abolfazli Esfahani and Shenghai Yuan

***Abstract*—It is vital for mobile robots to achieve safe autonomous steering in various changing environments. In this paper, a novel end-to-end network architecture is proposed for mobile robots to learn steering autonomously through deep reinforcement learning. Specifically, two sets of feature representations are firstly extracted from the depth inputs through two different input streams. The acquired features are then merged together to derive both linear and angular actions simultaneously. Moreover, a new action selection strategy is also introduced to achieve motion filtering by taking the consistency in angular velocity into account. Besides, in addition to the extrinsic rewards, the intrinsic bonuses are also adopted during training to improve the exploration capability. Furthermore, it is worth noting the proposed model is readily transferable from the simple virtual training environment to much more complicated real-world scenarios so that no further fine-tuning is required for real deployment. Compared to the existing methods, the proposed method demonstrates significant superiority in terms of average reward, convergence speed, success rate, and generalization capability. In addition, it exhibits outstanding performance in various cluttered real-world environments containing both static and dynamic obstacles. A video of our experiments can be found at https://youtu.be/19jrQGG1oCU.**

***Index Terms*—Deep reinforcement learning; depth image; difference image; autonomous steering; intrinsic reward**

I. Introduction

AUTONOMOUS robots have been extensively studied throughout the history of artificial intelligence and they can be used in a wide range of applications, such as exploration, inspection, surveillance, information collection, and so on [1]. A great many of methods have thereby been developed to enable mobile robots to steer safely in various unknown cluttered environments autonomously [2], [3]. Nevertheless, conventional methods typically build on a variety of assumptions which may not hold in practical applications [4], [5] and are also likely to be associated with intensive computational cost [6], [7]. In addition, conventional methods generally include a quantity of manually designed parameters [8] instead of learning automatically from previous experiences [9]. As a result, it is challenging to generalize these approaches well to unseen circumstances due to their limited adaptability.

In recent years, approaches developed based on various deep learning techniques have been introduced to address the bottlenecks of conventional methods and explore solutions to the aforementioned issues [10]–[13]. As an important category, deep reinforcement learning (DRL) based methods have gained rapidly growing popularity since they demonstrate promising performance while impose no requirement on labeled data. So far, many impressive DRL-based methods have validated that control commands can be derived efficiently from raw sensor inputs [14]–[17]. For instance, both networks proposed in [18] and [19] acquired control commands from 2D laser inputs with the former used an asynchronous DRL model while the latter applied an external controller assisted DRL method. These laser range sensor based methods take advantage of the negligible difference between the synthetic and actual scenarios, and hence present high transferability to real deployments. This is important because training DRL models directly in real-world environments is typically impractical due to the extremely heavy workload on interaction and the consequent excessive consumption of time.

However, the input observations of these laser-based DRL approaches are provided by 2D laser range finders and contain quite limited sensing information which may fail to be descriptive enough of the three-dimensional scenarios. In contrast, visual sensors are able to provide more sensing information of the surroundings. In addition, they are generally more affordable as well. Therefore, Zhu et al. proposed a DRL architecture which mapped RGB images to control policies directly [20]. Instead of using raw RGB images, segmented ones were adopted in [21] to control humanoid robots. Nevertheless, although RGB images are more informative, they suffer from the significant deviations between the simulated and real-world domains. As a result, DRL models with RGB inputs are deficient in generalization capability and often require further fine-tunings.

In order to relieve the burden of real-world deployment, depth images can be utilized rather than RGB images since they exhibit much better visual fidelity in virtual environments because of their textureless nature [22]. Therefore, both Tai et al. [23] and Zhang et al. [24] chose raw depth maps as the network inputs. Instead of acquiring depth inputs directly, Xie et al. used depth maps converted from RGB images as the inputs [8]. In their work, the depth information was firstly estimated from RGB images through the FCRN introduced in [25]. And the predicted depth maps were then fed into the network as inputs for action determination. In most of these works, Deep Q-Network (DQN) [26] and its well-known variants, were utilized to train the networks. And in [27], a noisy branching architecture was embedded in the double DQN (DDQN) [28] to derive control commands from raw depth input with improved performance.

Extended from [27], in this paper, a novel deep neural network architecture is proposed to substantially improve the

* Corresponding Author
The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, e-mail: (wukeyu@ntu.edu.sg; HW@ntu.edu.sg).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

effectiveness of the DRL model for autonomous steering. Distinctive from the previous depth-based methods which feed merely the raw depth images into the network, we additionally calculate the difference images between successive depth maps and pass them as an implicit set of inputs. In this way, the performance of our model can be improved through extracting spatio-temporal feature representations. Generally, spatio-temporal features are important for motion-related applications, and thus many studies have been conducted in this field [29], [30]. As an important category, a variety of Convolutional Neural Network (CNN) based architectures have been explored. For instance, two-stream 2D CNNs, 3D CNNs, 3D Residual CNNs and two-stream inflated 3D CNNs were proposed in [31], [32], [33] and [34] to learn spatiotemporal features from video sequences. Besides, in [35] and [36], 3D CNNs were separated to spatial and temporal convolutions to improve the computational efficiency. Inspired by works where depth motion maps were adopted to improve the extraction of spatio-temporal features for human action recognition [37], [38], we extract the spatio-temporal features from the depth maps and their difference images in this paper, which is simple, computationally efficient and proved to be highly effective in our experiments. Specifically, after feeding the raw depth images into the network, the difference images between successive depth maps are calculated and processed in the network as an implicit set of inputs. The two sets of images are then regarded as the inputs of two separate streams of the network respectively. For each stream, CNNs are implemented to extract features from the input images. And the generated feature representations are subsequently combined and mapped to two vectors of Q-values using separate branches of the succeeding noisy fully connected layers. In this manner, both linear and angular control commands are yielded simultaneously according to the estimated Q-values.

In addition, a β-consistency action selection strategy is introduced in this paper so that the consistency in angular control command is also taken into account during action selection. Therefore, an angular velocity is determined in consideration of both the estimated Q-values and the previously executed velocity command. In this manner, the stability of the developed system is improved essentially through motion filtering while the agent is also capable to drive itself out of dead ends through executing consistent actions. In the meanwhile, the training framework of our network is also modified. Besides extrinsic rewards, intrinsic rewards are adopted as well during training to improve the exploration capability, and the intrinsic rewards are calculated as the random network distillation (RND) exploration bonuses [39]. Different from other intrinsic rewards, the RND bonus overcomes the problem that the prediction errors can be caused by stochastic transitions instead of experience novelty. And generally, it grants higher rewards to more dissimilar states so as to motivate efficient exploration.

Therefore, the main contributions of our work can be summarized as follows. Firstly, a new end-to-end network architecture is proposed to improve the quality of feature representations through feeding the temporal changes into an additional input stream of the network. Secondly, a novel action selection scheme is also introduced and acts as a motion filter. By considering the consistency in angular velocity command, the proposed β-consistency strategy is capable to increase the stability of the system, and in the meantime, improve the model's capability to cope with dead ends. Moreover, in addition to extrinsic reward, the RND-based intrinsic bonus is adopted as well to augment the reward signal and improve the exploration ability. Last but not least, although the proposed model is trained in a simple virtual environment, it is directly deployable in complex cluttered environments without any fine-tuning. Hence, the trained model is readily generalizable to various real-world environments containing both static and dynamic obstacles. Experiments have been carried out in various virtual and real-world scenarios to demonstrate the superiority of our model.

The rest of this paper is organized as follows. Section II introduces the related work. In Section III, the proposed deep neural network architecture and the DRL-based autonomous steering algorithm are described in detail. Section IV presents the experimental results and discussions. Finally, conclusions are given in Section V.

II. Related Work

Proposed by Mnih et al. [26], [40], DQN has already been applied in a broad range of fields, such as video games [26], object localization [41], robotic manipulation [42], and autonomous steering [43]. The remarkable success of DQN mainly attributes to two techniques which increase the stability of the training and addresses the issue of convergence. Firstly, the experience replay technique significantly improves the data efficiency and removes the temporal correlation among data. Moreover, a separate target network is also adopted to stabilize bootstrap updates. In [23], a DQN model was used to generate control commands for a mobile robot based on input depth maps to achieve collision avoidance in unknown environments.

Built upon these two techniques, many variants of DQN have been introduced. As one of the most representative extensions, the double DQN (DDQN) decoupled action selection from evaluation to solve the over-estimation problem by correcting biases of Q-value functions [28]. Specifically, DDQN utilized the behavior network to choose the action while used the target network to calculate the corresponding action value. Rather than computing Q-values directly, the dueling network architecture introduced in [44] decomposed Q-values into state values and advantage functions. As a result, it calculated the state values and the advantages separately to generalize the learning across actions. In [8], the dueling architecture based DDQN was applied to map the estimated depth images to control commands for autonomous steering.

Rather than adopted the commonly used ϵ-greedy strategy to achieve exploration, Fortunato et al. introduced the NoisyNet version of DQN which enhanced the efficiency of exploration through adding noise to parameters of the network [45]. In spite of the success of DQN and its variants mentioned above, the competence of these algorithms is restricted when coping with high-dimensional action spaces [46]. And as a consequence, merely quite limited number of discrete control

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

outputs are allowed typically. In order to address this limitation and improve the performance of autonomous steering, a new variant of DQN was introduced in [27] which derived linear and angular velocities simultaneously from the raw depth inputs via incorporating a branching noisy architecture in the dueling DDQN. It is worth noting that in all the previous depth-based algorithms, the feature representations were directly extracted from the depth maps through CNNs. Nevertheless, the approach proposed in this paper extract feature representations simultaneously from both the raw depth maps and the difference images between successive depth maps, which distinguishes it from all the existing works. As a second set of inputs, the difference images are processed by an additional stream of CNNs so that critical features can be extracted explicitly from the encoded temporal variations.

In addition, the proposed method also employs intrinsic rewards along with the extrinsic rewards. Intrinsic motivation is responsible of spontaneous curiosity and thereby can facilitate the exploration. For instance, an intrinsically motivated active learning algorithm was proposed in [47] to improve the exploration in sensorimotor spaces, and goal-directed exploration was introduced for motor control based on intrinsic motivation in [48]. Typically, intrinsic rewards are used to quantify the intrinsic motivation [49] and the rationale is to predict the consequences resulted from taking an action [50], [51]. Firstly, visitation counts of states are measures that can be used as intrinsic rewards for effective exploration [52], [53]. Besides, intrinsic rewards can be modeled based on comparisons between the current observation and the past episodic memories [54]. Moreover, the differences between the actual and predicted consequences can also be regarded as a measure of surprise [55], [56]. Generally, the latter dynamic-based rewards are straightforward to scale and parallelize [57]. And in these exploration methods, since the intrinsic rewards were induced by prediction errors, the agents were motivated to explore regions that were difficult to predict [58], [59]. In the meantime, the dynamics models were updated to improve the predictions in these regions. However, an important limitation of these methods is that the prediction errors can also be induced by stochastic transitions in addition to experience novelty [58]. By combining intrinsically motivated learning with imitation learning, the algorithm introduced in [60] was able to improve performance in stochastic environments. More recently, Burda et al. [39] proposed the random network distillation bonus to obviate undesirable stochasticity and achieved remarkable success in playing various Atari games which involved challenging exploration. Although the intrinsic rewards were typically employed in tasks with sparse extrinsic reward signals, they can also be used jointly with dense extrinsic rewards for better exploration of the environments. In [61], the reward signals were augmented by intrinsic motivation to learn laser-based navigation policies for mobile robots. And it was also demonstrated in [62] that the intrinsic motivation can be combined with dense extrinsic reward to improve the learning of reaching and grasping skills. In this paper, we adopt the random network distillation bonus as the intrinsic reward and incorporate it into our training framework for performance improvement.

III. Method

A. Problem Statement

The goal of our work is to enable mobile robots to learn autonomous steering effectively via DRL. And the developed model is required to yield effective control commands from raw depth images with high performance. In addition, the acquired depth-based model is also expected to have great transferability capability so that it can be deployed to practical applications without any fine-tuning.

In this paper, the autonomous steering problem is considered as a Markov Decision Process (MDP) defined using a tuple $M = (S, A, R, P, \gamma)$, where S, A, R, P and $\gamma \in [0, 1]$ represent the state space, action space, immediate reward, transition function, and discount factor, respectively [63]. Therefore, at time step t, the robot executes an action $a_t \in A$ based on the input state $s_t \in S$. It then receives a reward r_t and transits to the next state s_{t+1}. In an MDP, the mapping from a state s to an action a is specified by a policy $\pi(a|s)$. And given a policy π, the Q-value is defined as the expectation of discounted cumulative rewards for taking action a in state s, which can be formulated as follows:

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|s_0 = s, a_0 = a]. \quad (1)$$

The objective of the MDP is then to find a policy which maximizes this expected discounted accumulative reward. The Q-learning algorithm can be used to solve this problem by approximating the optimal Q-value iteratively using the following Bellman equation:

$$Q^{*}(s_t, a_t) = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}). \quad (2)$$

In our work, a state is composed of a series of four depth maps which are all resized to 80×100. Besides, rather than outputting simple commands, such as 'forward', 'right', and 'left', the action space of our model comprises a set of velocities to avoid the issue of coarse action discretization. And in the meanwhile, both angular and linear velocities can be executed separately at the same time in accordance with the input state. With the aforementioned design of state and action space, the proposed network is aimed to learn a more effective policy which leads to higher rewards, increased success rates, as well as improved generalization capability.

B. Network Architecture

To achieve the objectives mentioned above, a novel deep reinforcement learning model named advanced branching noisy dueling DDQN (BND*-DDQN) is proposed. It is essentially an extension of the BND-DDQN model [27] and its network architecture is illustrated in Fig. 1. Distinct from previous works, the feature representations in the proposed network are not only extracted from the raw depth images, but also from a second set of implicit inputs which are the difference images between consecutive frames. The difference image generation unit is embedded at the beginning of our network. And in addition to performing subtraction operation between successive depth images, it is also critical for the generation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

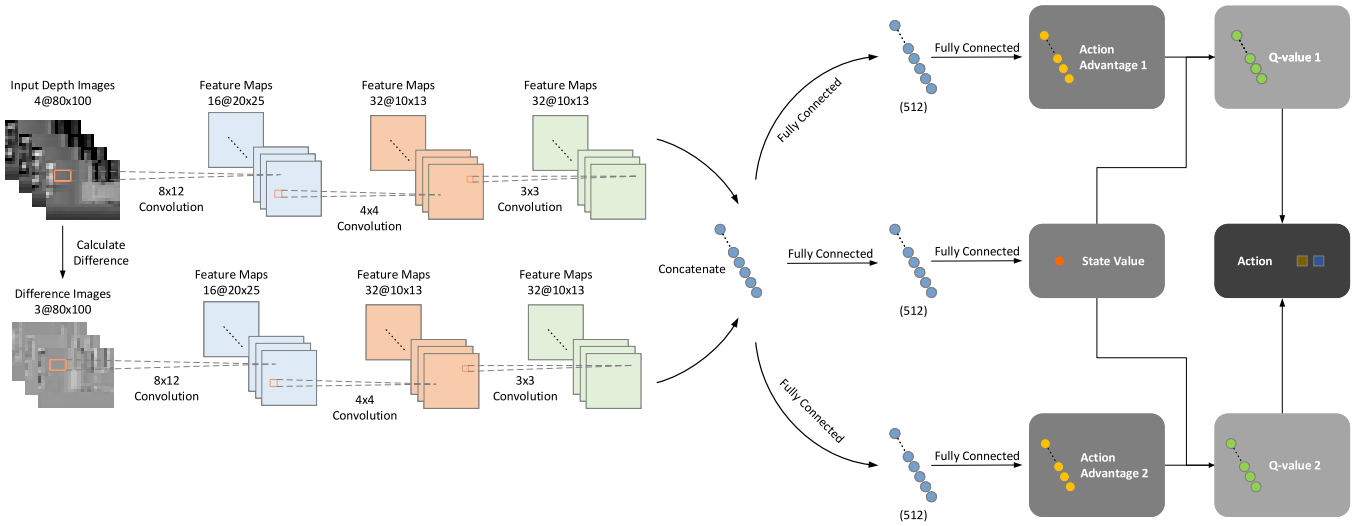


Fig. 1: Network architecture of the BND*-DDQN model. Input of the network is a sequence of depth images obtained at four successive steps. Based on the input depth maps, three difference images between consecutive frames are generated accordingly. The feature representations are then extracted from these two sets of inputs separately through different CNN streams. Subsequently, the network splits into three branches to map the extracted features to a state value and two advantage functions, respectively. The vectors of Q-values are then calculated through the aggregation layers. And based on the estimated Q-values, the linear and angular control commands are determined simultaneously.

unit to implement instance normalization so as to enhance the quality of the difference images and thereby improve the training performance. In this way, the temporal changes and dynamic processes can be better expressed by the generated difference images, whereby more crucial information can be contributed from these temporal variations via extraction of feature representations.

Therefore, at each time step, a stack of four sequential depth images of size 80×100 are concatenated together as the first set of inputs of the network. And then, a sequence of three difference images are calculated according to the input depth maps using the difference image generation unit. As depicted in Fig. 1, feature representations are extracted from both the explicit depth map inputs and the implicit difference image inputs separately through two streams of CNNs. And for each stream, three convolutional layers are adopted to extract features. Specifically, the first CNN layer maps the input images to 16 feature maps of size 20×25 using filters with size 8×12 and stride four. Then, another CNN layer filters the obtained feature maps using 32 4×4 kernels applied with stride two. Next, the final CNN layer subsequently generates 32 feature maps through 3×3 filters with stride one. The yielded feature maps are then flattened to form a vector of features. Therefore, one feature vector is generated from the raw depth images via the first stream of CNNs while another feature vector is produced from the difference images through the second stream of CNNs. Subsequently, the two feature representations are concatenated together and fed into the succeeding fully connected layers.

After feature extraction, our network is split into three branches with one of them being used to predict the state value. Meanwhile, the other branches are aimed to estimate the two advantage functions corresponding to angular and linear control commands, respectively. Generally, each of these three branches consists of two dense layers. In the first branch, the state value is estimated by passing the feature representations to two fully connected layers with 512 units and 1 unit, respectively. And in the second branch, the action advantages corresponding to various linear velocities are predicted through two fully connected layers with 512 and N neurons, respectively, where N denotes the number of discretized linear velocity commands. Similarly, the feature representations are also mapped to the advantage functions associated to various angular velocities through two dense layers which contain 512 and N hidden units, respectively.

In most of the previous works, the exploration is achieved by adopting the ϵ-greedy strategy. However, the exploration introduced in this way can be insufficient and is also lack of consistency due to the sudden switches. Hence, in the proposed model, the exploration is achieved via adding Gaussian noises to network parameters. Therefore, instead of employing conventional dense layers, we adopt the noisy ones [45] in our model to improve the efficiency as well as consistency of exploration. A linear layer can typically be written as $y = \omega x + b$, where x and y are the inputs and outputs, respectively, while ω and b denote the weight matrix and biases, respectively. In noisy layers, uncertainty can be induced by perturbing the weights and biases, and the outputs can be expressed as:

$$y = (\mu_\omega + \sigma_\omega \odot \epsilon_\omega)x + \mu_b + \sigma_b \odot \epsilon_b, \tag{3}$$

where $\odot$ represents element-wise multiplication, μ_ω, σ_ω, μ_b, and σ_b are the network parameters, whereas ϵ_ω and ϵ_b denote the random noises.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

Despite of the improvement in exploration capability, the addition of noise in this manner results in a large number of additional noise variables. To alleviate the computational burden, factorized Gaussian noises are applied instead of the above independent ones. Therefore, the elements of weight matrix ω can be rewritten as:

$$\omega_{i,j} = \mu^{\omega}_{i,j} + \sigma^{\omega}_{i,j} f(\epsilon_i) f(\epsilon_j), \tag{4}$$

and the biases b can be restated accordingly as:

$$b_j = \mu^b_j + \sigma^b_j f(\epsilon_j), \tag{5}$$

where ϵ_i and ϵ_j are random values from normal distributions and f is a function defined as $f(\epsilon) = \text{sgn}(\epsilon)\sqrt{|\epsilon|}$. In Equations (4) and (5), the initial values of μ^{ω} and μ^b are randomly generated from the uniform distribution in the range $[-\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}})$, where N is the number of hidden neurons in the input layer. Meanwhile, the initial values of σ^{ω} and σ^b are set to $\frac{0.4}{\sqrt{N}}$.

In dueling DDQN, the network estimates the state value function $V(s)$ and the advantage function $A(s, a)$ separately through two different streams. The Q-value is then predicted by combining these two streams through an aggregation layer. In general, the Q-value can be specified as:

$$\begin{aligned} Q(s_t, a_t; \theta, \theta_V, \theta_A) =& V(s_t; \theta, \theta_V) + A(s_t, a_t; \theta, \theta_A) \\ & - \frac{1}{N} \sum_a A(s_t, a; \theta, \theta_A), \end{aligned} \tag{6}$$

where θ, θ_V, θ_A represent the parameters of the common layers, the value stream and the advantage stream, respectively, while N signifies the total number of actions. On account of the dueling architecture, both the action-independent state values and action-dependent advantage functions can be calculated straightforwardly, which improves the reliability for Q-value estimation.

In our model, the dueling architecture is also adopted. Nevertheless, instead of outputting one Q-value vector which mixes up linear and angular actions, the proposed network yields two Q-value vectors which associated with angular and linear velocities, respectively. As introduced above, the state-value function $V(s)$ and the advantage functions A_1 and A_2 are predicted using three separate branches. And each of the branches is consisted of different noisy fully connected layers. The state value is thereafter combined with each advantage function through an aggregation layer to estimate the Q-value functions Q_1 and Q_2 respectively. Therefore, each of the Q-values can be expressed as:

$$\begin{aligned} Q_i(s, a_i) =& V(s) + A_i(s, a_i) \\ & - \frac{1}{N_i} \sum_{a'_i} A_i(s, a'_i), \quad i = 1, 2 \end{aligned} \tag{7}$$

where N_i is the size of the i-th dimension of the 2D action space.

C. Training Framework

The training framework for the proposed deep neural network is demonstrated in Fig. 2. During training, the supervision signals are contributed from both the the extrinsic and intrinsic rewards. Hence, the total reward signal r is defined as the weighted sum of these two types of rewards, and hence can be expressed as:

$$r = \lambda_e r_e + \lambda_i r_i. \tag{8}$$

where λ_e and λ_i are the scaling coefficients. Specifically, an instructive extrinsic reward is devised as shown in Equation (9). In general, our agent is expected to move forward rapidly and only turns when it is necessary to avoid obstacles. Therefore, to simplify and speed up the training process, the desired behaviors are articulated in the extrinsic reward function.

$$r(s_t, a_t) = \begin{cases} -10 & \text{if in collision} \\ \lambda_1 v^2 cos(\lambda_2 v\omega) - \lambda_3 & \text{otherwise} \end{cases} \tag{9}$$

In Equation (9), v denotes the linear velocity while ω represents the angular velocity. Besides, λ_1, λ_2 and λ_3 are three constant coefficients. In specific, the constants λ_1 and λ_3 are the scaling factor and bias of the reward function, respectively. And these two factors can be tuned to prescribe the ranges of the extrinsic reward signals so that an appropriate distribution of rewards is defined with the highest reward approximately equaling to one. In the meantime, the second coefficient λ_2 is designed to regulate the influence of linear control commands at different angular velocities. In general, if a small angular velocity is executed, the agent is expected to maneuver forward as speedily as possible. In these cases, the difference in linear velocity can lead to substantial variance in extrinsic reward. In addition, since safety is given the first priority, the largest penalty is resulted from collision. Therefore, larger angular velocities are expected to be chosen when it is essential for the agent to avoid collision via changing its heading orientation. And under these circumstances, the influence of linear control command to extrinsic reward is lessened. In consideration of the safety issue, the increase in linear velocity is even possible to cause a decrease in extrinsic reward when a large angular velocity need to be executed. Generally, the effect of linear and angular velocities in various situations can be customized by tuning λ_2.

Besides extrinsic supervision, the random network distillation (RND) bonuses are regarded as intrinsic rewards during training to achieve better exploration. The RND bonus is employed because it is efficient to compute, simple to implement, and can be used to deal with image-based inputs effectively. Moreover, it removes the interference caused by undesirable stochastic transitions, which outperforms the other prediction error based exploration bonuses. As depicted in Fig. 2, two neural networks are involved in calculating the RND bonuses. The first one is the fixed target network which is initialized randomly and the other one is the prediction network which need to be trained on collected data. In our case, both target network and prediction network map the depth inputs at time step $t+1$ to embeddings through multiple CNN layers followed by fully connected layers. The parameters of the prediction network θ_P are updated by minimizing the following mean squared error:

$$L(\theta_P) = ||\hat{f}(s_{t+1}; \theta_P) - f(s_{t+1})||^2, \tag{10}$$

where s_{t+1} represents the next state while f and $\hat{f}$ indicate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

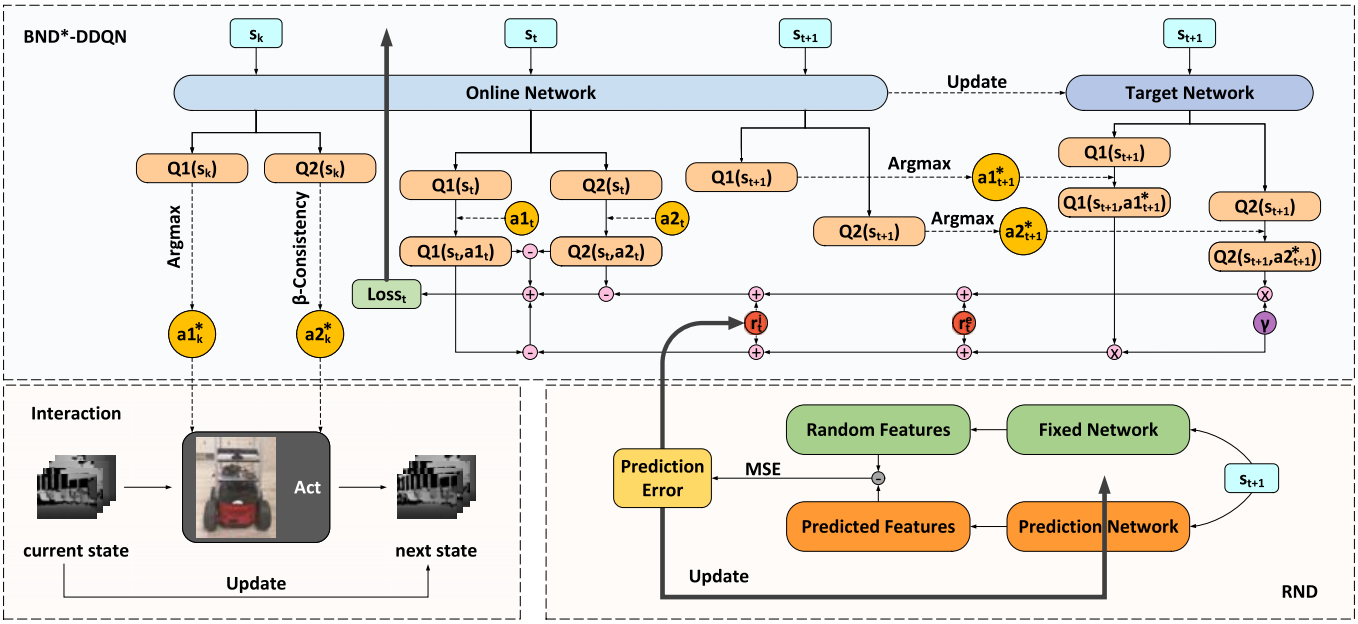


Fig. 2: Training framework of the BND*-DDQN model. At each iteration, the current state is fed into the online network to calculate the Q-values of the executed actions. If intrinsic rewards are adopted, the next state is passed into the RND network to calculated the exploration bonus. In the meantime, the next state is also passed into the online and target networks to determine the target Q-values along with the reward signal and the discounted factor. The online network is then updated based on the loss function which is defined as the weighted sum of three terms, including the differences between the estimated Q-values and their target values, and the difference between the two estimated Q-values.

the mapping functions of the target and prediction networks, respectively. And in the meanwhile, the prediction errors are normalized and thereafter regarded as the intrinsic rewards to motivate the exploration of novel states. With the designed extrinsic and intrinsic reward signals, the goal of training is to learn a policy which maximizes the discounted cumulative reward received by the agent.

Throughout the training, the online network is responsible for action selections based on the input states, and the target network is used to determine the target Q-values. At every time step, the agent executes the predicted control commands, transits to the next state and receives an immediate reward. The current state is then updated to determine the actions in the next step. The experience replay buffer used to store the interactions are updated continuously and its maximum capacity is set to 20,000. It is worth noting that the exploration is achieved by adding Gaussian noises to the network instead of using the the ϵ-greedy strategy. Hence, the velocity commands can be greedily selected according to the Q-values predicted by the online network. Furthermore, instead of the deterministic policy, the β-consistency action selection strategy introduced in this paper can be adopted to improve the performance of the learned policy. The motivation of the β-consistency strategy is to mitigate the left-right swing behavior of the agent while also enable the agent to steer itself clear of dead ends. In essence, the β-consistency strategy acts as an additional motion filter by taking account of the consistency in angular velocity command. Therefore, while the linear control commands are still greedily chosen, an angular velocity is selected in consideration of both the estimated Q-values and the command executed in the previous step. And the selection scheme for angular control commands can be described as:

$$a2_k^* = \begin{cases} a2_{k-1}^* & \text{if } \frac{\exp(Q2^*(s_k,a2_i;\theta)) - \exp(Q2(s_k,a2_{k-1}^*;\theta))}{\sum_{i=1}^{N}\exp(Q2(s_k,a2_i;\theta))} < \beta, \\ \underset{a2_i}{\text{argmax}} Q2(s_k,a2_i;\theta) & \text{otherwise,} \end{cases} \tag{11}$$

where β is the threshold and $a2_{k-1}^*$ denotes the previously executed action.

As described in the pseudo-code in Algorithm 1, the parameters of the online network are initialized randomly in the beginning of training, whereas the parameters of the target network are duplicated from θ. And then, at each training step, we sample a batch of transitions from the buffer to update the parameters of the online network. Specifically, the online network estimates the two Q-value vectors, $Q1(s_t;\theta)$ and $Q2(s_t;\theta)$, based on the input current state s_t. And the two vectors of Q-values are in correspondence with the discretized angular and linear velocities, respectively. The actions to be executed, $a1_t$ and $a2_t$, are then determined by following either the deterministic or the β-consistency strategy. And their Q-values, $Q1(s_t,a1_t;\theta)$ and $Q2(s_t,a2_t;\theta)$, are also extracted accordingly. Complying with DDQN, the next state s_{t+1} is, in the meantime, passed into the online network to determine the best actions at time step $t+1$. Besides, it is also fed into the target network to calculate the Q-values of the selected actions. If the intrinsic reward is adopted, the next state s_{t+1} is also used to calculate the exploration bonus. Next, in consideration of the Q-values at step $t+1$, the current reward r_t, and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

Algorithm 1: BND*-DDQN

for t = *1 to* T **do**
  Select $a1_k^* = \underset{a1_i}{\text{argmax}} Q1(s_k, a1_i; \theta)$
  if *use* β*-consistency* **AND** $\frac{\exp(Q2^*(s_k, a2_i;\theta)) - \exp(Q2(s_k, a2_{k-1}^*;\theta))}{\sum_{i=1}^{N} \exp(Q2(s_k, a2_i;\theta))} < \beta$ **then**
    Select $a2_k^* = a2_{k-1}^*$
  else
    Select $a2_k^* = \underset{a2_i}{\text{argmax}} Q2(s_k, a2_i; \theta)$
  end
  Execute $a1_k^*$ and $a2_k^*$ simultaneously, transit to the next state s_{k+1}, and receive reward r_k
  Store transition $(s_k, a1^*, a2^*, r_k, s_{k+1})$ in replay buffer D
  Sample noise variables $\epsilon_i, \epsilon_j, \epsilon_i^-, \epsilon_j^- \sim N(0, 1)$
  Sample a batch of N_B transitions $(s_t, a1_t, a2_t, r_t, s_{t+1})$ from D
  if *episode terminates at t+1* **then**
    Set $y_1 = r_t$ and $y_2 = r_t$
  else
    Set $y_1 = r_t + \gamma Q1(s_{t+1}, \underset{a1_{t+1}}{\text{argmax}} Q1(s_{t+1}, a1_{t+1}, \epsilon; \theta), \epsilon^-; \theta^-)$
    Set $y_2 = r_t + \gamma Q2(s_{t+1}, \underset{a2_{t+1}}{\text{argmax}} Q2(s_{t+1}, a2_{t+1}, \epsilon; \theta), \epsilon^-; \theta^-)$
  end
  Compute loss
  $L(\theta) = \mathbb{E}[\alpha_1(y_1 - Q1(s_t, a1_t, \epsilon; \theta))^2 + \alpha_2(y_2 - Q2(s_t, a2_t, \epsilon; \theta))^2 + \alpha_3(Q1(s_t, a1_t, \epsilon; \theta) - Q2(s_t, a2_t, \epsilon; \theta))^2]$
  Update parameters of the online network θ
  if t *mod* N_T = *0* **then**
    Update parameters of the target network $\theta^- \leftarrow \theta$
  end
end

discount factor γ, the target Q-values at time step t can be calculated using:

$$y_i = r_t + \gamma Qi(s_{t+1}, \underset{ai_{t+1}}{\text{argmax}} Qi(s_{t+1}, ai_{t+1}, \epsilon; \theta), \epsilon^-; \theta^-), \tag{12}$$

where i indicates the dimension of the action space. Besides, ϵ and ϵ^- denote the noise variables of the online and target network, respectively, while θ and θ^- represent the parameters of the online and target network, respectively.

According to the target Q-values and their current values estimated by the online network, our loss function is then defined as:

$$\begin{aligned} L(\theta) = \mathbb{E}[&\alpha_1(y_1 - Q1(s_t, a1_t, \epsilon; \theta))^2 \\ &+ \alpha_2(y_2 - Q2(s_t, a2_t, \epsilon; \theta))^2 \\ &+ \alpha_3(Q1(s_t, a1_t, \epsilon; \theta) - Q2(s_t, a2_t, \epsilon; \theta))^2], \end{aligned} \tag{13}$$

where α_1, α_2, and α_3 are three scaling factors. The first two terms in Equation (13) are aimed at minimizing the differences between the predicted Q-values and their corresponding target values. And because the angular and linear velocities are performed concurrently during the interaction, the last term is designed to minimize the difference between the Q-value estimates corresponding to the two commands. The weights and biases of the online network are updated constantly through backpropagation and the Adam optimizer is used with a learning rate of 1e-5. During the backpropagation, the gradients with regard to the CNN layers are divided by 2 in consideration of the branching architecture. On the contrary, the target network is not trainable and its parameters are synchronized with those of the online network every 1000 steps.

IV. Experiments

A. Experiments in Virtual Environments

A 10×10 virtual training environment with various obstacles is created in Gazebo [64] as demonstrated in Fig. 3 and a simulated Pioneer 3-AT robot is used for interaction. The training is performed using Tensorflow [65] on a desktop with Intel i7-7700 CPU, 32GB RAM and NVIDIA GeForce GTX 1080 Ti GPU, and the average training time for each iteration is approximately 0.22 second. It is noteworthy that even if trained in a simple virtual environment, the proposed model can generalize well to various complex virtual as well as real-world environments without any fine-tuning. During training, the raw depth images are captured using the depth sensor of a simulated Kinect. As introduced in Section III, after the input depth maps are fed into the network, the difference images are generated accordingly and both sets of images are then used to extract the crucial features. Subsequently, the control commands are derived based on the feature representations. During the experiments, the velocities are published as a ROS twist message, and in the meantime, the ROS odometry message is subscribed to calculate the extrinsic reward signal.

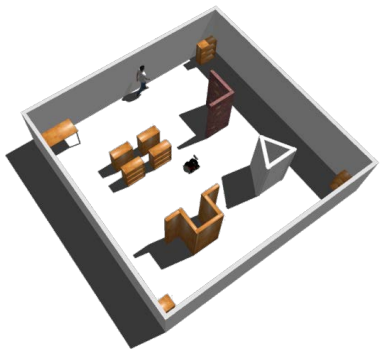


Fig. 3: The virtual training environment.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

In addition to the proposed model, a series of baseline models, including DQN, DDQN, Dueling DDQN, Noisy Dueling DDQN, and BND-DDQN, are also trained in the same environment for comparison. To extract features from depth inputs, these baseline models adopt the same CNN architecture as our depth image stream illustrated in Fig. 1. Besides, all models are trained from scratch with the batch size and discount factor being set to 64 and 0.99, respectively. And for models which adopt the ϵ-greedy strategy to achieve exploration, the value of ϵ is initially set to 0.1 and reduced constantly to 0.0001 within the first 200,000 steps. In all experiments, seven discrete linear velocities are considered, including 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, and 0.7 m/s. Meanwhile, the angular velocity is also chosen from seven values, including $-\pi/4$, $-\pi/6$, $-\pi/12$, 0, $\pi/12$, $\pi/6$, and $\pi/4$. Since there is no branching architecture embedded in the first four baseline models for Q-value calculation, these four models can only select either an angular or a linear velocity at each step. However, for the BND-DDQN and the proposed models, both velocities are picked and executed at every time step. Besides, the weighting factors, α_1, α_2 and α_3, of the loss function defined in Equation (13) are set to 0.4, 0.4 and 0.2, respectively. During training, all baseline models are supervised only by the extrinsic rewards introduced in Equation (9), where λ_1, λ_2 and λ_3, are set to 2, 2, 0.1, respectively. To demonstrate the superiority of the proposed network architecture, the training of our BND*-DDQN is firstly carried out without implementing the β-consistency strategy while driven only by the extrinsic rewards as well. And then the β-consistency strategy is adopted additionally and the corresponding model is indicated as BND*-DDQN w/ β-Consistency. During training, the threshold β in Equation (11) is increased from the initial value of 0.001 to the final value of 0.05 within 2e5 training iterations. On this basis, the network, denoted as BND*-DDQN-RND w/ β-Consistency, is trained by employing the RND exploration bonus together with the extrinsic reward. The scaling coefficients, λ_e and λ_i, in Equation (8) are set to 1 and 0.1, respectively. Since the extrinsic reward is adopted to yield the optimal policy for the autonomous steering task while the intrinsic reward is to improve the exploration capability during the training, a larger scaling coefficient is assigned to the extrinsic rewards because it is associated to our primary objective more closely.

For each model, the maximum training iterations is set to 5e5, and the maximum length of an episode is set to 500 so that an episode terminates without any penalty after 500 steps. At the beginning of each episode, the agent starts from the center of the training environment with a random orientation. And throughout the training, each model is evaluated every 5000 iterations. The evaluation is performed by calculating the average episodic reward of 5 episodes. And the episodic reward refers to the sum of the instantaneous extrinsic rewards received within an episode. To achieve better statistical significance, each model is trained from scratch three times using different random seeds and the means of the average episodic rewards obtained by the eight models are illustrated in Fig. 4. It can be observed that, at the end of training, the DQN model results in the lowest average episodic reward while the DDQN and Dueling DDQN models achieve better and similar performance. Compared to these three models, the Noisy Dueling DDQN model leads to a higher episodic reward through enhancing the exploration. However, it is less competitive than the BND-DDQN model which results in improved average episodic reward as well as convergence rate. Compared to these baseline models, the proposed BND*-DDQN model exhibits the most prominent performance. It leads to notably increased average episodic rewards throughout the training with a convergence rate similar to that of the BND-DDQN model. And it is worth noting that if the β-consistency action selection strategy is employed additionally, the proposed model is able to yield even higher average episodic rewards and the stability of the training process is also improved at the same time. Moreover, the introduction of the RND-based intrinsic reward enables our model to reach the convergence more rapidly, and hence further improves the training efficiency.

Besides average episodic reward, a comparison of success rate is also performed. The success rate is defined as the proportion of successful episodes and an episode is identified to be successful if it terminates without any punishment. To examine the generalization capability, the trained models are evaluated in three different types of virtual environments shown in Fig. 5. The first Willow Garage world is an office-like environment which contains a number of narrow corridors and cramped rooms with small entrances. The second world is a constrained indoor scenario containing a number of unseen furnitures [66] and the third world provides an outdoor environment. Generally, the situations encountered in these virtual environments are much more complex compared with those involved in the training environment. And it is noteworthy that the models trained in the simple scenario are applied in these virtual environments directly without any fine-tuning. During the evaluation, an episode's maximum length is set to 300. In each environment, a model's success rate is computed based on 50 episodes and the starting poses are specified in Fig. 5 where the red dots indicate the starting locations and the yellow arrows denote the starting orientations. In the first and third environments, the robot starts with each pose 10 times. And in the second environment, the robot starts with the orientations denoted by the left and right arrows 20 times, respectively, and starts with the orientation indicated by the down arrow 10 times.

The comparison results of success rate are described in Table I. As expected, the proposed model outperforms the existing methods consistently. In the first scenario, the success rates achieved by the DQN and DDQN models both plunge below 25%. With more sophisticated network architectures, the Dueling DDQN, Noisy Dueling DDQN, and BND-DDQN models leads to increased success rates. However, the highest success rate attained by the baseline methods is merely 40%. On the contrary, the proposed BND*-DDQN model yields a success rate of 84%. Moreover, the BND*-DDQN w/ β-Consistency and the BND*-DDQN-RND w/ β-Consistency models further raise the success rate up to 96% and 98%, respectively. Similarly, the proposed model also achieves noticeably increased success rate in the second environment. Although the BND-DDQN and the BND*-DDQN models

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

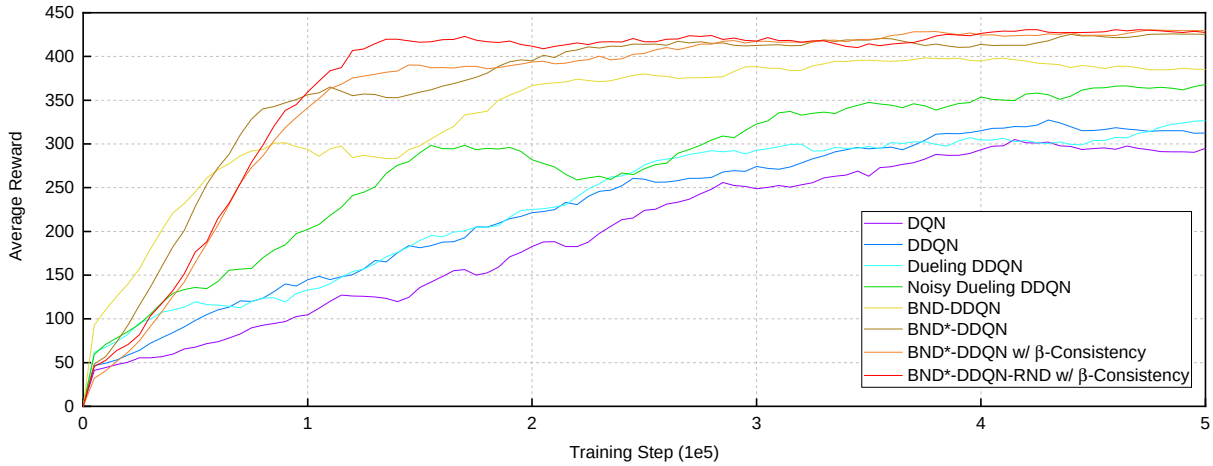
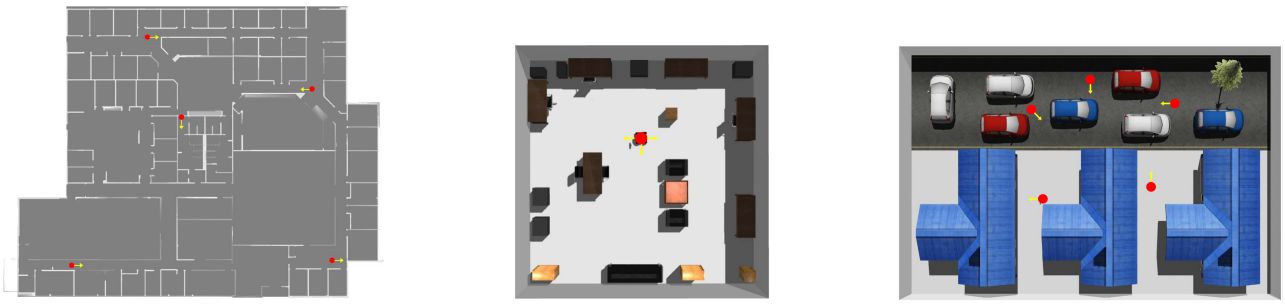


Fig. 4: Average episodic rewards obtained by different models.



(a) (b) (c)

Fig. 5: The virtual environments used for evaluation. The red dots indicate the starting locations during the experiments and the yellow arrows illustrate the corresponding starting orientations. (a) The office-like Willow Garage environment. (b) An indoor environment with a number of unseen furnitures. (c) An outdoor environment.

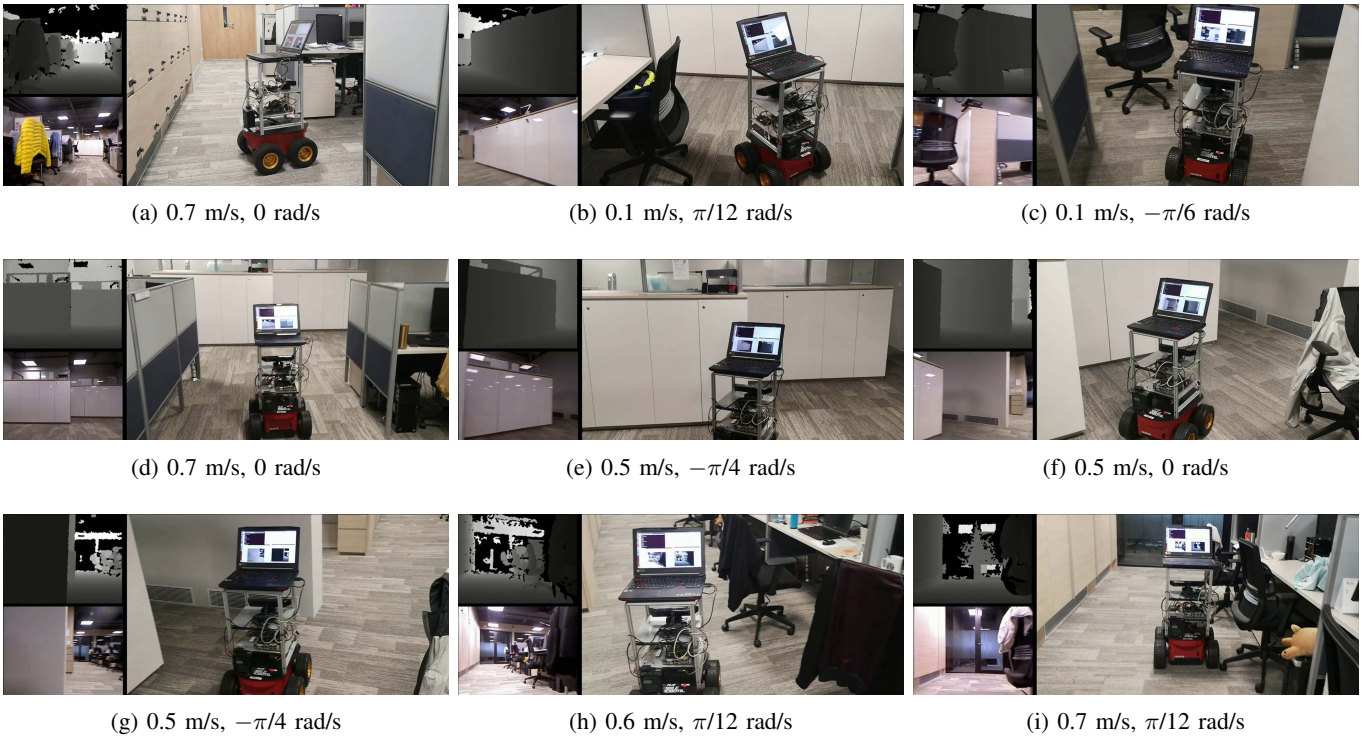
achieve the same success rates in the third environment, the proposed BND*-DDQN w/ β-Consistency and the BND*-DDQN-RND w/ β-Consistency models retain their superiority. And it is worth noting that the success rate of our model is higher than 90% in all scenarios if the β-consistency action selection strategy is adopted. Therefore, the proposed approach exhibits much superior generalization capability compared with the existing methods.

In general, the competence of our model is firstly attributed to the improvement in feature abstraction. By extracting additional features from the generated difference images separately, the temporal changes are taken into account explicitly to provide critical information for action determination. From the experimental results, it is also noticed that the performance improvement contributed from the addition of the extra difference image stream is more noticeable in more constrained environments. Specifically, in the less constrained outdoor environment, no obvious improvement in success rate is observed by adopting the BND*-DDQN model. However, the BND*-DDQN model leads to a much higher success rate in the constrained indoor environment and the superiority becomes even more significant in the most constrained Willow Garage world. Typically, in constrained environments, the difference images are more informative and thereby more likely to improve performance. Besides, the outdoor environment also contains more unfamiliar patterns, which imposes more difficulties in adaptability. In addition to feature extraction, the response capability of the BND*-DDQN model is also

TABLE I: Comparison of success rate between the proposed model and the existing methods.

Env	DQN	DDQN	Dueling DDQN	Noisy Dueling DDQN	BND-DDQN	BND*-DDQN	BND*-DDQN w/ β-Consistency	BND*-DDQN-RND w/ β-Consistency
1	20%	24%	32%	34%	40%	84%	96%	**98%**
2	30%	34%	30%	44%	70%	82%	**94%**	90%
3	56%	62%	66%	60%	70%	70%	92%	**94%**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems



(a) 0.7 m/s, 0 rad/s (b) 0.1 m/s, $\pi/12$ rad/s (c) 0.1 m/s, $-\pi/6$ rad/s

(d) 0.7 m/s, 0 rad/s (e) 0.5 m/s, $-\pi/4$ rad/s (f) 0.5 m/s, 0 rad/s

(g) 0.5 m/s, $-\pi/4$ rad/s (h) 0.6 m/s, $\pi/12$ rad/s (i) 0.7 m/s, $\pi/12$ rad/s

Fig. 6: Intermediate steps of the first real-world experiment. In this experiment, the BND*-DDQN model is deployed in an office environment. For each step, the depth image is demonstrated on the top left corner, the first-person view is shown on the bottom left corner, and the corresponding third-person view is displayed on the right. Besides, the derived linear and angular velocity commands are indicated in the sub-caption.

outstanding because angular and linear velocity commands are selected and executed concurrently. Moreover, the β-consistency strategy substantially improves the performance as well through increasing the stability and motion consistency of the agent. Last but not least, the training efficiency can also be improved by introducing the RND-based intrinsic rewards. Although the proposed model outperforms the existing approaches in terms of average episodic reward, convergence speed, success rate, as well as generalization capability, it still has room to improve. For instance, although the difference images are generated as an additional set of inputs, the input information can still be insufficient to describe the situations since only four successive time steps are considered. This issue can be alleviated by integrating more past memories into the network in an efficient way. Besides, the current sensor can be replaced by one with larger field of view so that more sensing information can be exploited for decision making.

Last but not least, since our BND*-DDQN requires only 0.0068s on average to map the depth images to a pair of velocity commands, it is quite competent for real-time autonomous steering tasks. Besides the aforementioned evaluations, various real-world experiments are also conducted and presented in the following to further demonstrate the superiority and generalization capability of the proposed BND*-DDQN approach.

B. Experiments in Real-World Environments

In all the real-world experiments, a Pioneer 3-AT mobile robot is used to execute the steering commands derived by our model and a Kinect is used to capture the depth images. The proposed model is evaluated in three different types of real-world scenarios, including a cluttered office environment, a constrained workshop environment, and a changing outdoor environment. Similar to the previous experiments, after training in the simple virtual environment, our model is applied in these real-world scenarios directly with no fine-tuning and the β-consistency strategy is adopted while determining the control commands based on the predicted Q-values. For all the following experiments, the intermediate steps are demonstrated, and for each step, the raw depth image is shown on the top left corner, and the first-person view is presented on the bottom left corner. Besides, a third-person view is also presented on the right to describe the situation in a more intuitive way. Moreover, the linear and angular velocity commands determined at each step are indicated in the sub-caption correspondingly.

First, our model is deployed in a cluttered office environment. As shown in Fig. 6(a), the mobile robot is controlled to move forward with the maximum linear velocity at the beginning. And when a cabinet is detected, the minimum linear velocity along with an angular velocity of $\pi/12$ rad/s is selected for the mobile robot to change its orientation in the constrained space without any collision. Similarly, the robot is commanded to slow down again when the second turning point is reached, however, a larger negative angular velocity is executed in this case to make a right turn safely. After the second turn, the robot starts to receive the maximum

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems

(a) 0.7 m/s, 0 rad/s (b) 0.6 m/s, $-\pi/4$ rad/s (c) 0.7 m/s, $\pi/12$ rad/s

(d) 0.2 m/s, $\pi/6$ rad/s (e) 0.2 m/s, $\pi/4$ rad/s (f) 0.7 m/s, $\pi/12$ rad/s

Fig. 7: Intermediate steps of the second real-world experiment. In this experiment, the BND*-DDQN model is evaluated in a workshop environment. For each step, the depth image is demonstrated on the top left corner, the first-person view is shown on the bottom left corner, and the corresponding third-person view is displayed on the right. Besides, the derived linear and angular velocity commands are indicated in the sub-caption.

linear velocity again in order to pass through the corridor rapidly. As demonstrated in Fig. 6(e)-6(g), at the third turning point, a decreased linear velocity and the maximum negative angular velocity are chosen at first to avoid bumping into a cabinet. And in a later step, a linear velocity of 0.5 m/s and a zero angular velocity are derived so that the robot is able to pass through between the cabinet and the chair. The robot is subsequently controlled to turn right with the maximum angular velocity in order to continue the maneuver without colliding with the wall. Then, the robot travels forward until it is necessary to make a left turn to avoid a chair, as shown in Fig. 6(h). And near the end of the first experiment, the robot is commanded to make a left turn again with the minimum angular velocity to avoid another chair. Therefore, the mobile robot succeeds in maneuvering in the cluttered office environment autonomously and safely by following the commands derived by the proposed model.

In addition, our BND*-DDQN model is evaluated in a workshop environment as well. And the purpose of the second experiment is to demonstrate the capability of the proposed model in dealing with dynamic obstacles and dead ends. Firstly, an example of avoiding a walking human is illustrated in Fig. 7(a)-7(c). At the beginning, the robot is traveling forward with the maximum linear velocity when the walker begins to appear in its field of view. And as the walker keeps moving ahead, the commanded linear velocity is reduced to 0.6 m/s. Meanwhile, the orientation of the robot is adjusted by executing the maximum negative angular velocity so as to avoid colliding with the walker. Subsequently, the robot is controlled to pass through between the two chairs with a linear velocity of 0.7 m/s. In the second example as depicted in Fig. 7(d)-7(f), the robot drives itself out of a dead end. At the beginning of the second scenario, the robot is running into a dead end. Therefore, as shown in Fig. 7(d), a smaller linear velocity of 0.2 m/s and a larger angular velocity of $\pi/6$ rad/s are chosen to change the steering orientation of the robot while preventing it from bumping into the wall. And at a later step in the more imperative situation, the robot is commanded to maintain a small linear velocity while continuing to turn left with the maximum angular velocity. After steering clear of the danger, the maximum linear velocity is yielded and sent to the robot again while the angular velocity is reduced accordingly. In this example, the mobile robot steers clear of the dead end by turning left consistently without any left-right swing, which exemplifies the importance of the β-consistency strategy.

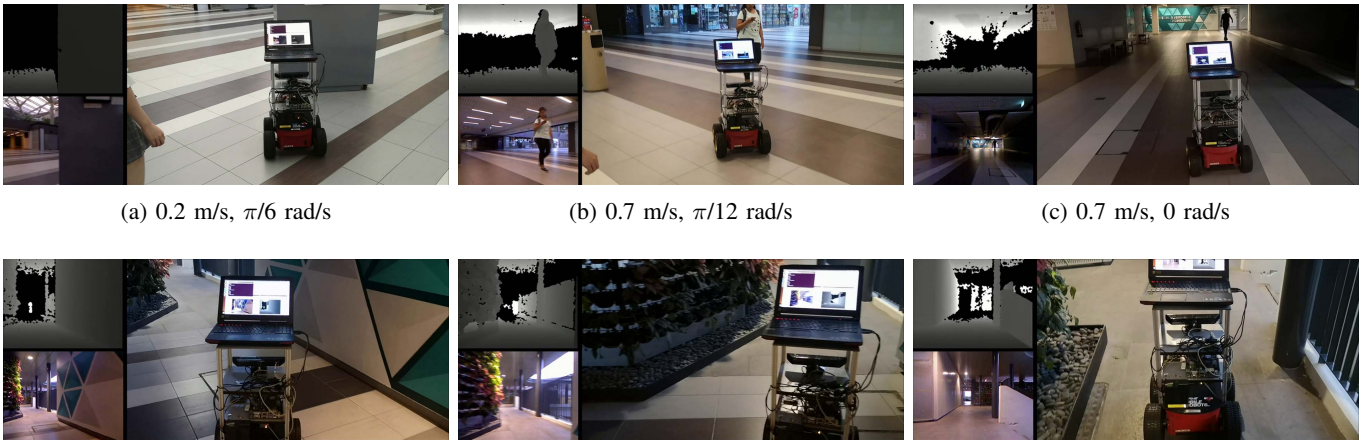
Furthermore, the proposed BND*-DDQN model is also evaluated in a changing outdoor environment as shown in Fig. 8. While steering in the outdoor environment, the robot is capable of responding rapidly to both static and dynamic obstacles. For instance, as demonstrated in Fig. 8(d), a wall is detected near the end of the experiment. Therefore, the minimum positive angular velocity is commanded to the mobile robot at first to avoid collision with the wall. However, as shown in Fig. 8(e), some plants are perceived after the turning, and hence a negative angular velocity is subsequently selected by our BND*-DDQN model to change the robot's steering direction quickly. In this way, the robot is able to pass through between the wall and the plants. And it is then commanded to move forward with the maximum linear velocity again to pass through a narrow passage rapidly.

The experimental results demonstrate the effectiveness, efficiency, as well as high transferability of the proposed BND*-DDQN model. In comparison to existing methods, our approach exhibits significant improvement in terms of average reward, success rate, and convergence speed. Moreover, even though our model is trained in a simple virtual environment, it is readily transferable and can be directly deployed in various complex virtual and real-world scenarios with no fine-tuning.

V. Conclusion

Conventional methods are typically built upon various assumptions and require manual tuning of a number of pa-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCDS.2019.2928820, IEEE Transactions on Cognitive and Developmental Systems



(a) 0.2 m/s, π/6 rad/s (b) 0.7 m/s, π/12 rad/s (c) 0.7 m/s, 0 rad/s

(d) 0.7 m/s, π/12 rad/s (e) 0.7 m/s, $-\pi$/4 rad/s (f) 0.7 m/s, 0 rad/s

Fig. 8: Intermediate steps of the third real-world experiment. In this experiment, the BND*-DDQN model is deployed in a changing outdoor environment. For each step, the depth image is demonstrated on the top left corner, the first-person view is shown on the bottom left corner, and the corresponding third-person view is displayed on the right. Besides, the derived linear and angular velocity commands are indicated in the sub-caption.

rameters. Therefore, it is challenging for them to derive general control policies which can adapt to various unseen scenarios. In this paper, an end-to-end model named BND*-DDQN is proposed for mobile robots to learn depth-based autonomous steering effectively via deep reinforcement learning. Our BND*-DDQN model derives the control commands from depth images directly through a novel network architecture.

In specific, in addition to the input depth maps, difference images between successive frames are generated within the network as well and regarded as an implicit set of inputs. The feature representations are then extracted from both the depth and the difference images through two separate CNN streams. In this way, the quality of the extracted feature representations can be improved through considering the temporal variations explicitly. The extracted features are then combined and mapped to two vectors of Q-values through different branches of noisy layers so that both angular and linear velocities can be determined simultaneously. Moreover, a new action selection scheme named β-consistency strategy is also introduced, which takes the consistency in angular velocity into account during action selection. In essence, the β-consistency strategy works as a motion filter to increase the stability of the system and improve the capability of coping with dead ends. Besides, in addition to extrinsic rewards, we also adopt the RND bonuses as intrinsic rewards during training to improve the exploration capability. Lastly but not least, it is noteworthy that although our BND*-DDQN model is trained in a simple virtual environment, it is readily transferable and can be applied to a variety of complex real-world environments directly without any fine-tuning. Our next challenge will be to develop a new model for goal-directed autonomous navigation tasks.

ACKNOWLEDGEMENT

This work is supported by the ST Engineering-NTU Corporate Lab.

REFERENCES

[1] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol. 247, pp. 10–44, 2017.

[2] A. Pandey, S. Pandey, and D. Parhi, "Mobile robot navigation and obstacle avoidance techniques: A review," *Int Rob Auto J*, vol. 2, no. 3, p. 00022, 2017.

[3] H. Isakhani, N. Aouf, O. Kechagias-Stamatis, and J. F. Whidborne, "A furcated visual collision avoidance system for an autonomous micro robot," *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[4] A. Al-Kaff, F. García, D. Martín, A. De La Escalera, and J. M. Armingol, "Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs," *Sensors*, vol. 17, no. 5, p. 1061, 2017.

[5] W. Zhang, S. Wei, Y. Teng, J. Zhang, X. Wang, and Z. Yan, "Dynamic obstacle avoidance for unmanned underwater vehicles based on an improved velocity obstacle method," *Sensors*, vol. 17, no. 12, p. 2742, 2017.

[6] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[7] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *arXiv preprint arXiv:1711.03449*, 2017.

[8] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," *arXiv preprint arXiv:1706.09829*, 2017.

[9] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," *arXiv preprint arXiv:1709.10489*, 2017.

[10] J. Wang, R. Yan, H. Tang, and F. Sun, "Automatic object searching and behavior learning for mobile robots in unstructured environment by deep belief networks," *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[11] S. Stevšić, T. Nägeli, J. Alonso-Mora, and O. Hilliges, "Sample efficient learning of path following and obstacle avoidance behavior for quadrotors," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3852–3859, 2018.

[12] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[13] K. Wu, M. A. Esfahani, S. Yuan, and H. Wang, "Tdpp-net: Achieving three-dimensional path planning via a deep neural network architecture," *Neurocomputing*, vol. 357, pp. 151–162, 2019.

[14] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[15] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," *arXiv preprint arXiv:1709.10082*, 2017.

[16] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "One-shot reinforcement learning for robot navigation with interactive replay," *arXiv preprint arXiv:1711.10137*, 2017.

[17] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for map-less navigation by leveraging prior demonstrations," *arXiv preprint arXiv:1805.07095*, 2018.

[18] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 31–36.

[19] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni, "Learning with training wheels: Speeding up training with a simple controller for deep reinforcement learning." Institute of Electrical and Electronics Engineers, 2018.

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

[21] K. Lobos-Tsunekawa, F. Leiva, and J. Ruiz-del Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247–3254, 2018.

[22] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," *arXiv preprint arXiv:1710.02543*, 2017.

[23] L. Tai and M. Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots," *arXiv preprint arXiv:1610.01733*, 2016.

[24] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*. IEEE, 2017, pp. 2371–2378.

[25] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper depth prediction with fully convolutional residual networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[27] K. Wu, M. Abolfazli Esfahani, S. Yuan, and H. Wang, "Learn to steer through deep reinforcement learning," *Sensors*, vol. 18, no. 11, p. 3650, 2018.

[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, vol. 2. Phoenix, AZ, 2016, p. 5.

[29] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[30] C. Feichtenhofer, A. Pinz, and R. Wildes, "Spatiotemporal residual networks for video action recognition," in *Advances in neural information processing systems*, 2016, pp. 3468–3476.

[31] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[32] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[33] D. Tran, J. Ray, Z. Shou, S.-F. Chang, and M. Paluri, "Convnet architecture search for spatiotemporal feature learning," *arXiv preprint arXiv:1708.05038*, 2017.

[34] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6299–6308.

[35] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 6450–6459.

[36] S. Xie, C. Sun, J. Huang, Z. Tu, and K. Murphy, "Rethinking spatiotemporal feature learning: Speed-accuracy trade-offs in video classification," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 305–321.

[37] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. O. Ogunbona, "Action recognition from depth maps using deep convolutional neural networks," *IEEE transactions on human-machine systems*, vol. 46, no. 4, pp. 498–509, 2015.

[38] Y. Zhu and S. Newsam, "Depth2action: Exploring embedded depth for large-scale action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 668–684.

[39] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[41] Y. Wang, L. Zhang, L. Wang, and Z. Wang, "Multi-task learning for object localization with deep reinforcement learning," *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[42] A. Gudimella, R. Story, M. Shaker, R. Kong, M. Brown, V. Shnayder, and M. Campos, "Deep reinforcement learning for dexterous manipulation with concept networks," *arXiv preprint arXiv:1709.06977*, 2017.

[43] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[44] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[45] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[46] A. Tavakoli, F. Pardo, and P. Kormushev, "Action branching architectures for deep reinforcement learning," *arXiv preprint arXiv:1711.08946*, 2017.

[47] A. Baranes and P.-Y. Oudeyer, "Robust intrinsically motivated exploration and active learning," in *2009 IEEE 8th International Conference on Development and Learning*. IEEE, 2009, pp. 1–6.

[48] S. M. Nguyen and P.-Y. Oudeyer, "Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner," *Paladyn, Journal of Behavioral Robotics*, vol. 3, no. 3, pp. 136–146, 2012.

[49] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 3, pp. 230–247, 2010.

[50] G. Baldassarre and M. Mirolli, "Intrinsically motivated learning systems: an overview," in *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013, pp. 1–14.

[51] J. Gottlieb and P.-Y. Oudeyer, "Towards a neuroscience of active sampling and curiosity," *Nature Reviews Neuroscience*, p. 1, 2018.

[52] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, 2016, pp. 1471–1479.

[53] G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos, "Count-based exploration with neural density models," *arXiv preprint arXiv:1703.01310*, 2017.

[54] N. Savinov, A. Raichuk, R. Marinier, D. Vincent, M. Pollefeys, T. Lillicrap, and S. Gelly, "Episodic curiosity through reachability," in *International Conference on Learning Representations (ICLR)*, 2019.

[55] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)*, vol. 2017, 2017.

[56] G. Brunner, M. Fritsche, O. Richter, and R. Wattenhofer, "Using state predictions for value regularization in curiosity driven deep reinforcement learning," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2018, pp. 25–29.

[57] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," *arXiv preprint arXiv:1808.04355*, 2018.

[58] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic motivation systems for autonomous mental development," *IEEE transactions on evolutionary computation*, vol. 11, no. 2, pp. 265–286, 2007.

[59] J. Achiam and S. Sastry, "Surprise-based intrinsic motivation for deep reinforcement learning," *arXiv preprint arXiv:1703.01732*, 2017.

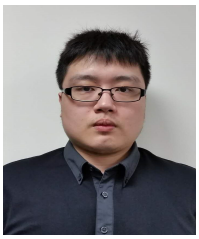
[60] S. M. Nguyen and P.-Y. Oudeyer, "Socially guided intrinsic motivation for robot learning of motor skills," *Autonomous Robots*, vol. 36, no. 3, pp. 273–294, 2014.

[61] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," *arXiv preprint arXiv:1804.00456*, 2018.
[62] M. B. Hafez, C. Weber, M. Kerzel, and S. Wermter, "Deep intrinsically motivated continuous actor-critic for efficient robotic visuomotor skill learning," *Paladyn, Journal of Behavioral Robotics*, vol. 10, no. 1, pp. 14–29, 2019.
[63] M. Zimmer and S. Doncieux, "Bootstrapping q-learning for robotics from neuro-evolution results," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 1, pp. 102–119, 2018.
[64] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator." in *IROS*, vol. 4. Citeseer, 2004, pp. 2149–2154.
[65] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: a system for large-scale machine learning." in *12th Symposium on Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.
[66] A. Rasouli and J. K. Tsotsos, "The effect of color space selection on detectability and discriminability of colored objects," *arXiv preprint arXiv:1702.05421*, 2017.

Keyu Wu received her B.Eng degree in Bioengineering from National University of Singapore, Singapore, in 2013. She is currently pursuing the Ph.D. degree with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Her research interests include deep reinforcement learning, imitation learning, visual navigation, path planning, trajectory generation, and autonomous robots.

Han Wang is currently an Associate Professor with the School of Electrical and Electronics Engineering, Nanyang Technological University. He received his Bachelor degree in Computer Science from Northeast Heavy Machinery Institute (China), and Ph.D. degree from the University of Leeds (UK), respectively. His research interests include computer vision, artificial intelligence, and robotics. He has published over 120 top quality international conference and journal papers. Dr. Wang is a senior member of IEEE.

Mahdi Abolfazli Esfahani received his B.Eng and M.Eng in Computer Engineering, and Artificial Intelligence and Robotics from Ferdowsi University of Mashhad, Iran. He was the leader of Nexus RoboCup soccer simulation team and achieved more than 20 national and international awards. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His research interests include deep learning, visual inertial odometry, and SLAM.



Shenghai Yuan received his B.Eng in Electrical and Electronics Engineering from Nanyang Technological University, Singapore, in 2013. He is currently pursuing the Ph.D. degree with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore. His research interests include visual odometry, stereo vision, object detection, and autonomous robotics system.