symmetry

MDPI

Article

Laser Based Navigation in Asymmetry and Complex Environment

Yuchen Zhao [1], Keying Xie [2,3], Qingfei Liu [4,*], Yawen Li [5] and Tian Wu [6]

[1] Forwardx Robotics Inc., Beijing 100096, China; zhaoyuchen@forwardx.com
[2] School of Economics, Beijing Wuzi University, Beijing 101149, China; 2019103026@bwu.edu.cn
[3] School of Fashion Communication, Beijing Institute of Fashion Technology, Beijing 100029, China
[4] Undergraduate School, National University of Defense Technology, Changsha 410073, China
[5] School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing 100876, China; warmly0716@bupt.edu.cn
[6] Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China; wutian@amss.ac.cn
* Correspondence: shenliulei12@nudt.edu.cn

Abstract: For collision-free navigation in unstructured and cluttered environments, deep reinforcement learning (DRL) has gained extensive successes for being capable of adapting to new environments without much human effort. However, due to its asymmetry, the problems related to its lack of data efficiency and robustness remain as challenges. In this paper, we present a new laser-based navigation system for mobile robots, which combines a global planner with reinforcement learning-based local trajectory re-planning. The proposed method uses Proximal Policy Optimization to learn an efficient and robust local planning policy with asynchronous data generation and training. Extensive experiments have been presented, showing that the proposed system achieves better performance than previous methods including end-to-end DRL, and it can improve the asymmetrical performance. Our analysis show that the proposed method can efficiently avoid deadlock points and achieves a higher success rate. Moreover, we show that our system can generalize to unseen environments and obstacles with only a few shots. The model enables the warehouse to realize automatic management through intelligent sorting and handling, and it is suitable for various customized application scenarios.

Keywords: deep reinforcement learning; navigation; obstacle avoidance; unmaned-vehicle

Citation: Zhao, Y.; Xie, K.; Liu, Q.; Li, Y.; Wu, T. Laser Based Navigation in Asymmetry and Complex Environment. *Symmetry* **2022**, *14*, 253. https://doi.org/10.3390/sym14020253

Academic Editor: José Carlos R. Alcantud

Received: 30 December 2021
Accepted: 24 January 2022
Published: 27 January 2022

1. Introduction

A basic skill for a mobile robot to serve human society is to navigate to a required location without collision. Navigation based on merely simple sensors such as lasers can be quite challenging, especially in cluttered environments. Traditional methods in this topic combine planning methods with artificially designed rules and models [1–3]. Those methods requires sedulous calculation, which is typically designed for specific robots and structured obstacles. For map building or mapless navigation where there is no prior knowledge of the asymmetric environment and non-structured obstacles, the decision must be made with only partial observations from the sensors, which make the problem even more challenging.

Following the eruption of deep learning-based methods which exhibits a powerful capability of representing raw sensor signals, deep reinforcement learning (DRL) gains extensive success in robot control [4]. DRL embraces the capability of pursuing any predefined targets through exploring the asymmetric environment freely. It has been widely applied to navigation in an end-to-end manner, which can generate smooth and collision-free paths toward the target to some extent [5–7]. However, several challenges remain: First, mapless end-to-end DRL navigator requires a huge amount of data for training, and generalization to new scenarios can be costly. Second, our investigation shows that end-to-end DRL is prone to being trapped in so-called deadlock points, where the cost

function lies in local minimum and the robot can hardly explore a way out. Third, widely used off-policy DRL algorithms such as deep deterministic policy gradients (DDPG) [8] may yield non-robust solutions, where noises or errors in the observation space lead to failure to reach the goal.

In map-based and map-building navigation, the planning-based method has served as a powerful tool for robustness and data efficiency, while in mapless navigation, DRL-based methods exhibited higher capability. To combine the favorable aspects of the previous works, we integrate a map-based global planner with mapless DRL controller. We show that with the global planner, the robot is less likely to be trapped in deadlock points, which yields a higher success rate of reaching the desired goal. Meanwhile, we are interested in the performance of different DRL algorithms in navigation. We show that on-policy methods such as Proximal Policy Optimization (PPO) achieve better performance in this problem. Our experiments are performed in a robot platform called Tyran. We use a 360-way laser detector as the observations and target velocity in the next time step as the actuator. We show that the proposed method produces both higher reward and a higher success rate. Moreover, we prove that the proposed framework is able to generalize to unseen environments with only a few shot of records in new environments.

The contributions of this paper is summarized as follows: We propose a navigation method combining a map-building global planner with the deep reinforcement learning method. For DRL, we apply PPO with asynchronous training. The proposed method achieves new state-of-the-art (SOTA) performance in laser-based navigation. In addition, we thoroughly test the proposed methods along with other baselines in asymmetric environments including hard modes and unseen obstacles. The results show that the proposed methods have a higher generalization capability and asymmetrical performance compared with other methods. Moreover, we proposed a novel learning scheme for training the agent to deal with a complex environment by adapting the PPO method in asynchronous fashion, and the proposed learning scheme can be seamlessly integrated with the navigation framework without modifying it to the learning task (i.e., the learning task is carried out alongside the navigation framework). Furthermore, the method made a solid proof that the deep RL agent can be gradually trained for solving the real-world navigation task just like how human learns a new task. In addition, it is capable of transferring and growing its knowledge in another novel environment without learning from scratch.

The rest of the paper is organized as follows. Section 2 reviews the literature for traditional and deep reinforcement learning-based approaches for resolving the navigation problem. In Section 3, the problem has been defined, and the proposed methodology has been explained. In Section 4, four experiments has been carried out for evaluating the proposed approach, and the results are explained and discussed. Section 5 concludes the findings and states the future works.

2. Related Work

2.1. Deep Reinforcement Learning

Reinforcement learning (RL) is able to learn from delayed feedback, which has been widely adopted in playing games [9,10] and robot control [11]. Among the most well-known DRL methods, the off-policy DDPG [8] and TD3 [12] are widely accepted for relatively higher sample efficiency, and the on-policy TRPO [13] and PPO [14] have been vastly tested regarding their capability of generating robust solutions in large action spaces. To accelerate the learning process, an asynchronous approach [15,16] can reduce the learning time up to hundreds of times by running data acquisition in parallel. In contrast to the tremendous success of model-free RL in robot control, the problem of low sample efficiency is still frequently mentioned [17]. As an alternative to model-free RL methods, model-based RL methods try to learn a neural environment model and use a planner to search for the optimal trajectory [18]. The model-based RL achieves greater success in cases where there is higher risks or costs in data acquisition.

2.2. Traditional Collision-Free Navigation

The navigation of mobile robots can be based on different sensors including Light Detection And Ranging (LiDAR) [19], laser [5,20,21], stereo camera [2,22], and RGB camera [6,7]. Our work is mainly related to laser-based navigation. In most circumstances, the robots do not have full knowledge of the asymmetric environment. Even for map-based navigation, unexpected obstacles may be encountered due to the dynamic environment or mapping errors. Traditional collision avoidance or the local planner use carefully designed policy such as boundary following [1,2] and potential fields [3]. Unfortunately, many of those methods relies on precise modeling of the surrounding environment, which even requires external sensors deployed in the asymmetric environment [23].

2.3. Reinforcement-Learning-Based Navigation

It has been noticed that RL has the capability of generalizing to unknown environments in collision avoidance in early years [24]; however, representing the state is difficult without deep neural network. Recently, attempts to apply DRL in navigation have been widely covered, either in vison-based navigation [6,7] or laser-based navigation [5,20,23], where goal-reaching (target-driven) reward [6], collision punishment [5], and even social force models [25,26] are frequently used reward settings. In previous works, DRL are applied in an end-to-end manner, where the model learns to navigate to the goal given information about the target and observation of the current frame. However, transferring the model from one environment to another one for DRL requires a large amount of training data and exploration efforts. An alternative is to use a hierarchical planning structure [27], where the global planner gives milestones toward the final target and the local planner searches a local trajectory to the next milestone. Similar settings include combining model-based RL with model-free RL in vision and language navigation [28] and combining a supervised learning global planner and RL local planner [29]. Yet, those hierarchical methods either use behavior cloning, which requires demonstration from humans, or adopts very simple action settings (such as a small discrete action set of go-straight, turn left, and turn right), which is incapable of accomplishing navigation for complex robot dynamics and environments. We summarize the related work in Appendix A Table A1.

3. Methodology

3.1. Preliminary

In reinforcement learning, we consider that an agent interacts with an asymmetric environment over a number of discrete time steps, which in general follows the Markov Decision Process (MDP). The MDP is a $\{S, A, R, P, \mu\}$ tuple, where S is a set of states, A is a set of actions, R is a set of immediate rewards, $P : S \times A \times S' \to [0, 1]$ is the transition probability (it represents the probability of transition from current state s to next state s' by taking an action a), and $\mu \to [0, 1]$ is the robot's initial state distribution. A policy $\pi : S \to A$ maps the state to the designated action, where $\pi(a|s)$ denotes the probability of choosing action over state. The localization of the robot is taken from the simulator by default, and it is calculated by the track deduction.

During the robot's learning process, we aim to choose a policy π that maximizes the performance measure, $J(\pi)$, which represents a finite horizon discounted total return, $J(\pi) = E_{\tau\sim\pi}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})]$ where $\gamma \in [0, 1]$ is the discount factor, τ is a roll-out of $[s, a, r, s']$ pairs by following a policy π (note the distribution of π may change during the roll-out). The distribution of τ depends on policy π, where $s_0 \sim \mu, a_t \sim \pi(\cdot|s_t)$,$s_{t+1} \sim P(\cdot|s_t, a_t)$.

A value function $V^\pi(s) = E_{\tau\sim\pi}[R(\tau)|s]$ denotes the accumulated discounted return $R(\tau)$ of the roll-out τ by selecting actions according to π. The advantage function is $A^\pi(s) = r + \gamma \cdot v(s') - v(s)$.

The classical policy gradient method estimates the policy gradient $\hat{g} = \hat{E}_t[\delta_\theta log\pi_\theta(a_t|s_t)\hat{A}_t]$ by sampling the environment using π in an on-policy manner and it is estimated using the stochastic gradient ascent algorithm. The loss function is $L(\theta) = E_t[log\pi_\theta(a_t|s_t)\hat{A}_t]$.

However, in general, the on-policy method learns much slower than the off-policy since it has to learn episode by episode due to the fact that it must sample and learn over the same π. Therefore, it is appealing to use the off-policy method, and PPO is currently one of the candidates:

$$L^{CPI}(\theta) = \hat{E}_t[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \cdot \hat{A}_t] = \hat{E}_t[r_t(\theta)\hat{A}_t] \tag{1}$$

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2}$$

where L^{CPI} is the conservative policy iteration. It may cause a large update step without a constraint. Therefore, L^{CLIP} is used to penalize a large gradient step when the ratio $r_t(\theta)$ is too large. The PPO method takes advantage of the policy gradient method in terms of its gradient estimator and constrains the ratio of the learned policy π and π_{old} in the interval $[1-\epsilon, 1+\epsilon]$.

3.2. Problem Setting

The aim of this paper is to provide a local planner that predicts the velocity of the robot by given the observations and the local target g_{local}:

$$v_t = \pi_\theta(x_t, g_{local}, v_{t-1}) \tag{3}$$

where x_t is the observation from the laser scanner, g_{local} is the local target calculated by the global planner, and v_{t-1} is the velocity command from the last time step. The functionality of elements of methodology is shown in Figure 1. Here, g_{local} is used to derive the relative local pose and the angle to path, which are concatenated in y_t. The final state representation is $s = [x_t, y_t, v_{t-1}]^T$.

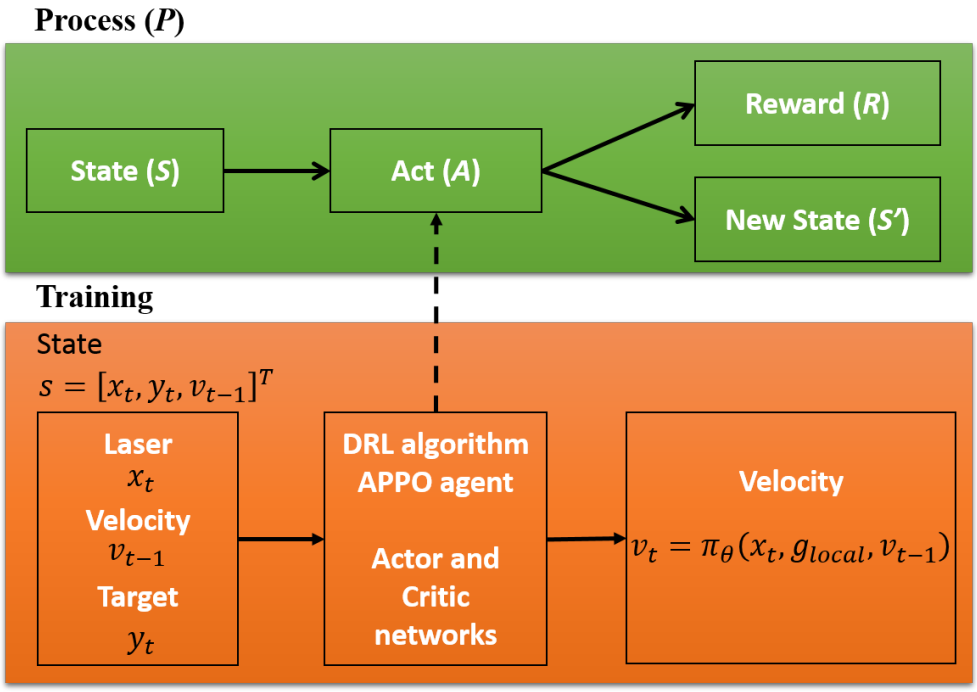

Figure 1. The functionality of elements of methodology.

The objective function is formulated as

$$\begin{aligned} J(\pi) &= r(s_T) + \int_0^T [r(s_t) + r(a_t)] \\ \pi^* &= max_{\pi \in \Pi} J(\pi) \\ s.j.x_{t+1} &= f(x_t, a_t) \end{aligned} \tag{4}$$

where the objective function J defines the terminal reward $r(s_T)$ and immediate reward $r(s_t)$ and $r(a_t)$. Since the robot interacts with the environment using its own kinematic model, therefore, the kinematic constrains are embedded in the sample roll-outs. The learning process is model free.

3.3. Algorithms

The method in this paper adapts the PPO method and modifies it into an asynchronous manner to solve the navigation task. The robot learning approach is shown in Figure 2. The approach is to have the robot learn the strategy in an asynchronous manner. The robot interacts with scenes in an asymmetric environment. Each scene that the robot learns is divided into 4 parts, each representing a specific start and goal. A dedicated worker is sent to collect a representation of the state in each puzzle. Once the robot learns the generalized strategy element π_{θ_0} in one scenario (i.e., scenario 1 in the upper left corner), it copies a set of initialization parameters to another scenario (i.e., scenario 2 in the upper right) and explores the effect of the parameters learned in scenario 1 on the success rate of the robot in scenario 2. A global PPO is created to interact with the scenes in the asymmetric environment. Each scene is divided into sub-scenes to provide a training environment for each worker. The worker is initialized with one patch of the scene, and it collects roll-outs from the environment. The environment, on the other hand, runs as a server to generate $[s, a, r]$ packets that wait for a worker to collect them. Unlike the normal environment, which purely reports the locomotion of the robot and controls its movements, the environment used here is essentially a navigation framework. It also provides a valid path from a global planner. The global planner used in this paper adapts the A* algorithm and runs a trajectory smoother afterward. It serves as a plug-in in the complete environment. By taking advantage of the global planner, the policy learned in this paper is a local planner in the navigation task. Once learned, the proposed method can be easily modified and adapted in the real environment and the navigation framework.

As mentioned in Section 3, the state observations $s = [x, y, v_{t-1}] \in R^{30}$ have three manifolds. The state $x \in R^{25}$ that related to the laser scans was originally 360-ways, and it was reduced to 25 by every 10° in a range $[60^\circ, 300^\circ]$. We did not use the full set of laser scans for 2 reasons: (i) the robot was not allowed to move backward; (ii) we did not want to deliberately increase the complexity of the learning task. The local goal g_{local} was computed by tracing from the last node of the global path toward the start node and stopped at the edge of the a pre-defined window size. The concept is illustrated in the top left corner of Figure 2 (a zoom-in view of one particular task). Once g_{local} was determined, it was used as a local target to attract the agent to consider approaching it. The first two dimensions of $y_t \in R^3$ were calculated by transforming the target in the robot coordinate and expressed in polar space (distance and angle). The third dimension was the angle to this path. $v_{t-1} \in R^2$ was the linear and angular velocity from the last time step.

The design of reward function is critical in the reinforcement learning method. Although a sparse rewarding scheme may also work, it is desirable to give the agent a prompt reward to guide the learning process. The reward functions are defined as follows:

$$r(s_t, a_t) = \begin{cases} r_{arrive} & if\ d_t < c_d \\ r_{collision} & if\ min_{x_t} < c_0 \\ w_1(d_{t-1}^{target} - d_t^{target}) \\ w_2(d_{t-1}^{path} - d_t^{path}) \end{cases}. \quad (5)$$

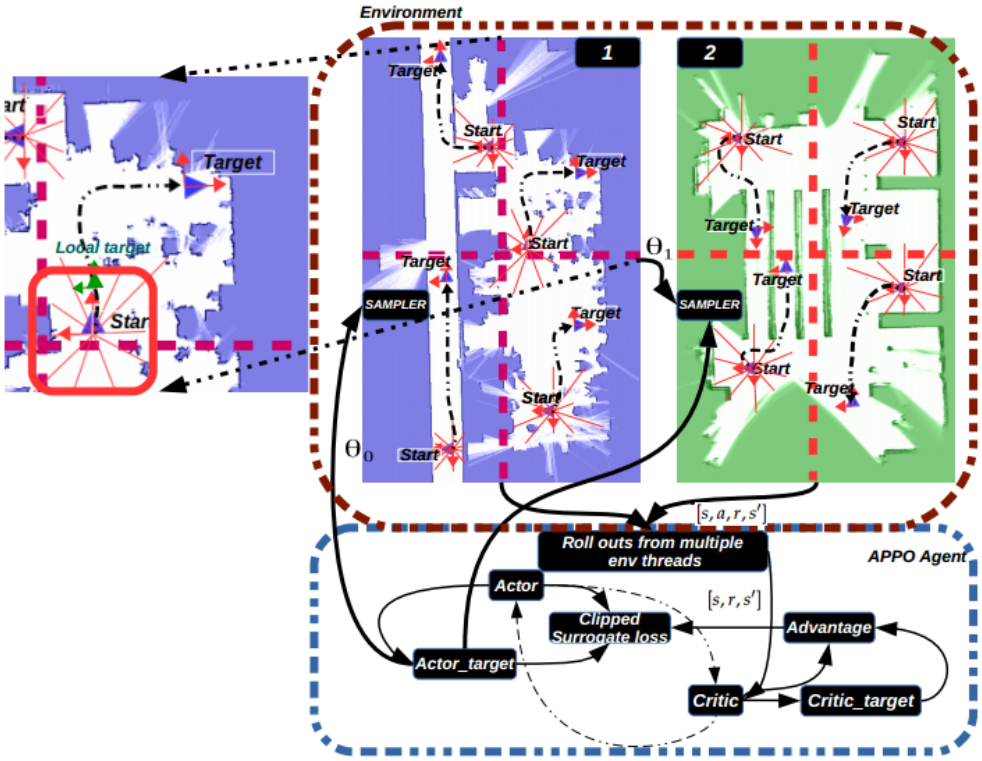

Figure 2. The proposed method learns the policy in an asynchronous manner. The robot interacts with the scene in the asymmetric environment. Each scene is divided into four segments where each puzzle represents one specific start and target situations. A dedicated worker is dispatched to collect the state representation in each puzzle. Once the robot has learned a generalized policy π_{θ_0} in one scene (i.e., scene 1 in top left), it copies to a set of initialized parameters in another scene (i.e., scene 2 in top right).

If the robot has arrived at the target within a distance threshold check $d_t < c_d$, a positive reward r_{arrive} is given, but if the robot has collided with the the obstacle through a minimum range check $min_{x_t} < c_0$, a negative reward $r_{collision} = -10$ is given. These conditions stop the episode and restart a new one. Otherwise, the rewards are the distance difference between the last target d_{t-1}^{target} in the robot frame and the current target d_t^{target} in the robot frame and the angle difference between the last angle to path d_{t-1}^{path} and the current angle to path d_t^{path}. These encourage the robot to follow the global plan but in a local manner.

The detailed implementation of the methodology is summarized in Algorithm 1. In Algorithm 1, we introduce the main structure of our asynchronous PPO.

Algorithm 1. Asynchronous PPO Pseudo-Code.

Set number of workers N
Set minimum batch size I_{update}
Initialize global buffer $v_{buffer}, a_{buffer}, s_{buffer}$
For n in N **then**
  Initialize each worker using **Worker** (I_{update}) in Algorithm 2
end for
Train network using **Train method** $\left(v_{buffer}, a_{buffer}, s_{buffer}\right)$ in Algorithm 3 when notified

In Algorithm 2, we initialized the data for each worker to update the robot data for velocity, reward, act, and state according to the actual situation. In Algorithm 3, we

combined the robot with velocity, act, and state, and train critic network weights and actor network weights. The main thread created N workers to collect roll-outs from different scenes. Each worker was initialized by using the method in Algorithm 2. Then, the training method was created by using Algorithm 3.

Algorithm 2. Asynchronous PPO Pseudo-Code for Each Worker Thread

Worker$\left(I_{update}\right)$
Set global step counter $t = 0$
Set global PPO network $\boldsymbol{PPO_{global}}$
Set global buffer queue **QUEUE**
Initialize training environment **ENV**
Initialize local buffer $s_{buffer} = [], a_{buffer} = [], r_{buffer} = [], \hat{v}_{buffer} = []$
Restart a new game and get Initial state s_0
repeat:
  Take action by sampling the target distribution $\pi_{\hat{\theta}_a}$
  Receive $\tau = [s, a, r, s']$ from **ENV** and append them to buffer
  If done or $mod\left(t, I_{update}\right) == 0$ **then:**

$$\hat{v}(s') = \begin{cases} 0 & for\ terminal\ s' \\ PPO_{global} \cdot v_{\hat{\theta}_c}(s') & for\ noneterminal\ s' \end{cases}$$

    For r in $r_{buffer}[:-1]$ **then:**
      Compute critic $\hat{v}(s') = r + \gamma\hat{v}(s')$
      After each batch during training, then $\hat{v}_{buffer}.append(\hat{v}(s'))$
    End For
    Reverse v_{buffer}
    Push $s_{buffer}, a_{buffer}, v_{buffer}$ into ***Queue***
    Reset $s_{buffer} = [], a_{buffer} = [], r_{buffer} = [], \hat{v}_{buffer} = []$
  end if
Notify **main thread** for update network

Algorithm 3. Pseudo-Code for **Train Method** in PPO_{global}

Initialize critic network weights $\theta_c \leftarrow U\left(\frac{-\sqrt{6}}{n_i+n_{i+1}}, \frac{\sqrt{6}}{n_i+n_{i+1}}\right)$
Initialize actor network weights $\theta_a \leftarrow U\left(\frac{-\sqrt{6}}{n_i+n_{i+1}}, \frac{\sqrt{6}}{n_i+n_{i+1}}\right)$
Initialize target critic network weights $\hat{\theta}_c \leftarrow \theta_c$
Initialize target actor network weights $\hat{\theta}_a \leftarrow \theta_a$
Set learning rate for actor and critic lr_a and lr_c
Set decay rate for critic network $decay$
Train method $(v_{buffer}, a_{buffer}, s_{buffer})$:
  Critic update:
    $\omega_{decay} = L2_{norm}(\theta_c)$
    $\theta_c \leftarrow \theta_c + \frac{\partial((v_{target} - v_{\theta_c}(s))^2 + \omega_{decay})}{\partial\theta_c}$ where $v_{target} - v_{\theta_c}(s)$ is **advantage**
    $\hat{\theta}_c = decay \times \hat{\theta}_c + (1 - decay) \times \theta_c$
  Actor update:
    $ratio = \frac{\pi_{\theta_a}(a_{buffer})}{\hat{\pi}_{\hat{\theta}_a}}$
    $surrogate\ loss = ratio \times \boldsymbol{advantage}$
    $clipped\ surrogate\ loss = min(surrogateloss, clip(ratio, 1-\epsilon, 1+\epsilon) \times \boldsymbol{advantage})$
    $\theta_a \leftarrow \theta_a + \frac{\partial(clipped\ surrogate\ loss)}{\partial\theta_a}$

In Algorithm 2, the worker was initialized with (i) a PPO_{global}, which was used to predict the value $v_\theta(s')$ (we use the target critic network to calculate $v_{\hat{\theta}_c}(s')$ to enhance the

training stability); (ii) an environment that was carefully designed by the user; (iii) a queue that was used to collect the sample roll-outs from each worker. In Algorithm 3, the actor and critic networks were firstly initialized with a uniform distribution. The detailed structure of the network is shown in Figure 3.

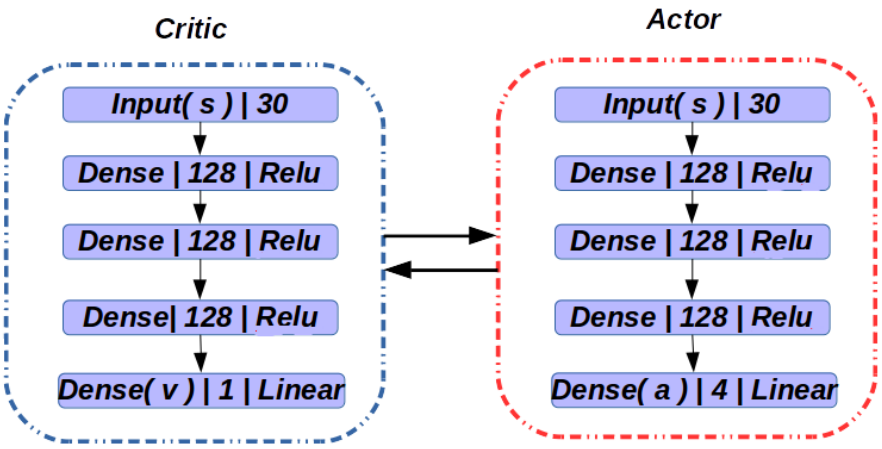

Figure 3. Network structure. On the left is the critic network that predicts a scalar value from state input variables s. On the right is the actor network that generates a stochastic policy from the same state inputs. The complete network was learned in an actor–critic fashion. The dimension of each layer and the type of activation functions are listed in the box.

According to Algorithm 3, the critic network will firstly iterate and then iterate the actor network until the convergence of the algorithm is completed. The critic network implemented 3 dense layers with 128 hidden units in each layer. The actor network used a similar structure but gave 4 dimensions of outputs $[v_\mu, v_\sigma, \omega_\mu, \omega_\sigma]$. Then, these outputs were plugged into a joint normal distribution to enable the robot to make decisions in a stochastic manner.

4. Experiments

4.1. Experiment Settings

The training process of the model developed in this paper was implemented alongside a navigation framework developed by **EwayOS** (http://www.ewaybot.cn/ewayos.html (accessed on 3 December 2019)). As explained, we did not train our agent with the environment in a synchronous way where the training step and the roll-out step was in a single thread. Alternatively, we treat the whole navigation framework as an environment so that we can not only take the observations from the robot but also take the result from the global planner. In this paper, the laser message and the robot positions and speed information were generated in the virtual environment. However, the map information is built with real laser scanners on a robot platform named **Tyran** (http://www.ewaybot.cn/tyran.html (accessed on 3 December 2019)).

To better evaluate our proposed method, we have developed 4 experiments, and the processing of the experiment setup has shown in Figure 4:

- A benchmark of learning methods that uses a deep RL agent in laser-based navigation. The scene was a typical office area. The start and target pose were designed to represent 4 typical situations that a robot may encounter during the navigation task. Each situation was randomly initialized and taken care of by a dedicated worker. For all methods, the learning rate for the critic network was 2×10^{-4} and that for the actor network was 1×10^{-4}. The hyperparameters for the reward functions are $w_1 = 40$ and $w_2 = -20$. The model was trained by using an Adam optimizer on a single Nvidia GeForce GTX 1060 GPU for 1000 episodes. We repeated 5 times for each method with different random seeds to show the stability and repeatability of the training process.

- A decomposition of the decision process in harder tasks. We initialized the new Asynchronous PPO (APPO) model with the parameters trained from the first experiment and continuously learned a harder version of the tasks. The start poses were deliberately chosen (e.g., randomized with the positions close to the corridor and angle opposed to the target) to increase the complexity of the task. In the test stage, each task was run 100 times.
- An evaluation of the model in unseen environments. In this experiment, we modified the scene with some unseen obstacles and tested the success rate (repeated over 100 times) of the model learned from experiment 2.
- An evaluation of the model in a new scene. This experiment initialized a new model with the parameters learned from experiment 2 and continuously improved it in scene 2 (a warehouse) with 4 designated tasks, which have been plotted in Figure 2.

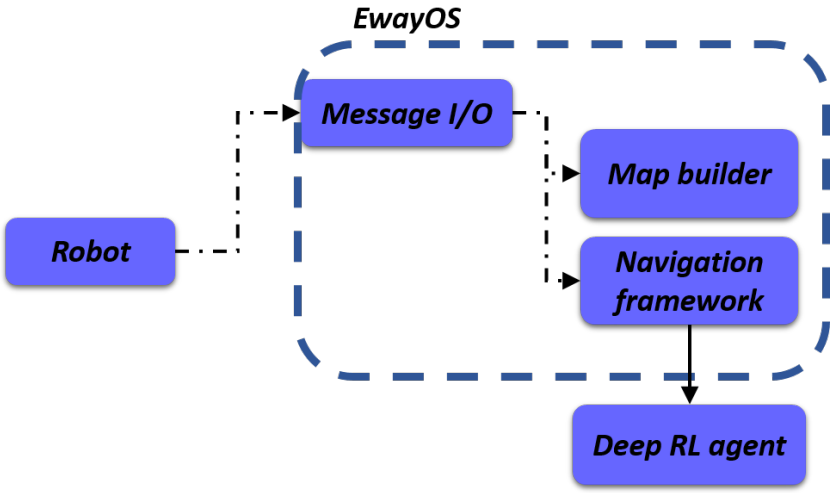

Figure 4. Experiment setup. The robot used for building the map was Tyran. Messages were processed by the map builder element and interacted with the navigation framework. The navigation framework provided the available state information for the deep RL agent.

4.2. Results

We evaluated the result by using the proposed experiments. The map was built by using real laser scans and odometry information installed on a Tyran platform. The mapping technique used was cartographer, which is embedded in **EwayOS** as well. To exclude the influence of the localization error, the navigation task was conducted in the virtual environment. We first evaluated our model against the baseline methods. Then, we showed that our model can grow even stronger with some more training efforts by evaluations against harder tasks and unseen obstacles. Finally, we showed that the model could be generalized to another scene without learning from scratch.

4.2.1. The Baseline Algorithm

The baseline algorithm used in this paper was an end-to-end method. It required no map information in the navigation task. The target in the reward function for encouraging approaching behavior was fixed by the global goal rather than a local goal, and the state vector did not contain angle to path information. In the end-to-end method, it compared the performance with a **Move-base** algorithm, which was used as a baseline. The performance index was the max control frequency, planning time, and planning distance. A better result was achieved in the end-to-end method. Therefore, we did not compare our method with the **Move-base** here. In this paper, we will show that the end-to-end method may not be able to learn a proper behavior in the scenes where the navigation tasks varied in different levels of complexity. To make a fair comparison, we extend its method by using the same state that was proposed in this paper and trained the model in an asynchronous fashion with the same four workers. Both of the baseline algorithms used the *DDPG* framework

for training a deep RL agent. Specifically, the replay buffer size was $1e4$, the minimum number of roll-out in buffer for start training was 2000, the mini-batch size was 128, the exploration factor for ϵ-greedy policy was initialized with 1.5 and gradually decays every 5 training epochs by 0.9998, the discount factor $\gamma = 0.99$.

4.2.2. Evaluation with the Baseline Algorithm

To evaluate the method, as illustrated in experiment 1, we firstly created four different navigation tasks with similar total distance. As shown in Figure 5, each task had a different start and end pose with a random heading derived from a uniform distribution $U[-3.14, 3.14]$. Note that the start and end positions were not fixed, it varied in that area in a tolerance of 0.5 m, and the start and end positions were randomly swapped as well.

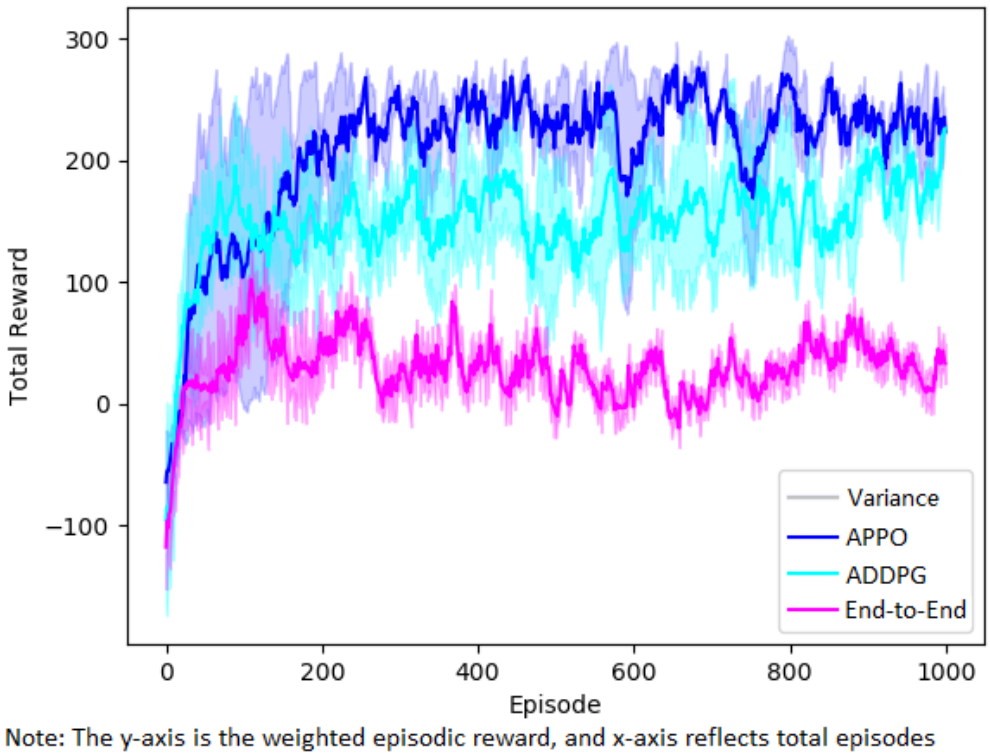

Figure 5. Evaluation result with baseline algorithms. The baselines used for comparison with the APPO method were end-to-end and ADDPG methods. The horizontal axis was the episode, and the vertical axis was the moving averaged total reward.

The performance is shown in Figure 5. The mean and standard deviation of each method at each training step were plotted across five benchmarks with different random seeds. The total reward achieved by the end-to-end method was around 50, whereas its asynchronous counterpart achieved 150. Our method had achieved around 250, which means that the proposed method was performing the best in the training process. By looking at the plotted curve, both the ADDPG and APPO methods converged in about 200 episodes. The end-to-end method converged in about 250 episodes. Among the first two methods, the convergence of the APPO was a little slower than the Asynchronous DDPG (ADDPG) method, but it reached the highest total reward. After the training stage, to make a fair comparison, we collected the three models and tested them in a similar environment with the set of start and end poses fixed. Specifically, task 1 was an elevator task; task 2 was a long corridor task; task 3 was a door entrance task; and task 4 was an office area task. As shown in Table 1, even though the end-to-end method converged in the learning process, it had a low success rate in the most cases. The ADDPG method and the APPO method can both work in the test; however, APPO achieved a better success rate in all four tasks. The best performance (98%) was achieved in task 2 and task 4.

4.2.3. Evaluation with Hard Mode

In Section 4.2.2, we compared the proposed method against various baseline methods. We randomly set the initial start pose to make sure the model was robust. The results in Table 1 indicated that the agent still had difficulties in some tasks, which may be hard to generalize in the previous setup. In this section, we upgraded the level of task complexity by changing the initial pose. The protocol had been defined in experiment 2. For instance, in the door entrance task (task 3), the heading angle was initialized by uniform distribution $U[-10^\circ, 10^\circ]$. This led to a result where the angle difference in between the agent's heading vector and the vector from the agent to the target was almost 180°.

Table 1. Evaluation results. The table summarizes the success rate of the benchmark methods in four different tasks. Each task was initialized randomly according to the distributions explained in the experiment setup.

Method	Task1	Task2	Task3	Task4
End-to-End	62%	65%	20%	60%
ADDPG	90%	93%	89%	95%
APPO	95%	98%	93%	98%

In Figure 6, a decompositionof the agent behavior using the refined model was plotted to illustrate the decision process in the critical stage of the individual task. As an example, in task 1, the agent was initialized with a pose that was almost heading toward the close end of a corridor.

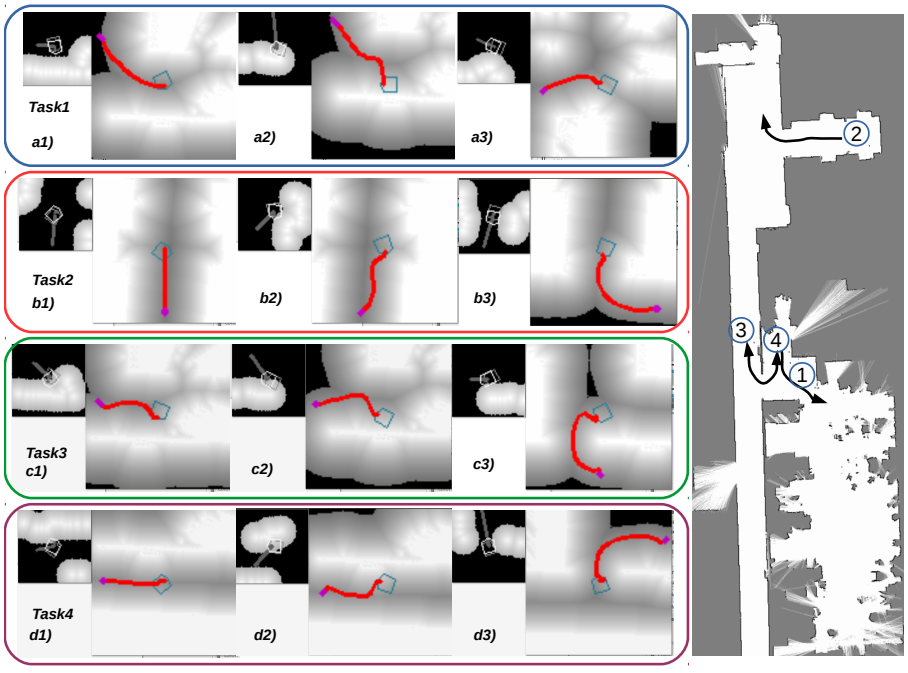

Figure 6. Decomposition of the agent behavior in hard mode. We use a map with size 100×100 for calculating the path (solid red line) to the local target. We use a map with size 60×60 to monitor the local behavior of the agent. The gray solid line is used to calculate the angle to path. The white solid footprint in the window is the next state by making the current move. Most of the time, it is small, as the agent makes decisions every 200 ms with a maximum of 0.5 m/s. We selected three typical stages for each task, and its number is shown on the right. These are listed as (**a1**–**d3**).

Its first move was spinning right, as shown in Figure 6(a1), and the local target was attracting it to go left. Then, its second move was keeping spinning right with a small step as the agent was almost hitting the side wall (Figure 6(a2)). This time, the local goal

was dragging the agent to go up. Later, the agent came to a narrow path, and it was heading straight (Figure 6(a3)). This was because it was in the middle of the obstacle. Similar decisions can be found in other tasks as well. For instance, in task 4, the agent was initialized right in the middle of a corridor. It firstly turned a circle to the right immediately (Figure 6(d1,d2)) and then went past the door entrance, as shown in Figure 6(d3). In Table 2, we summarized the success rate in these four tasks. For all tasks, they have reached more than a 98% success rate, which was a solid improvement compared with the elevator task and the door entrance task in Section 4.2.2.

Table 2. Evaluation results for hard mode.

Task1	Task2	Task3	Task4
98%	99%	98%	99%

4.2.4. Evaluation with Unseen Obstacles

In this section, we inherited the model learned from Section 4.2.3 and evaluated it against unseen obstacles (as shown in Figure 7). The protocol had been defined in experiment Section 4.1. We have plotted snapshots for the decisions made by the agent. In the first row, the agent was firstly trying to avoid an obstacle on its right, and the local target had changed due to the added new obstacle. Then, the agent moved with a small step when the path was becoming even more narrow (see Figure 6(a3)). Later, it moved normally after passing the narrow path. In the second row, the agent firstly moved slowly to avoid the wall on its left; then, it moved straight in between two newly added obstacles; later, it moved further away and got around a obstacle on its right. In Table 3, we have evaluated the same task from Section 4.2.3. The success rate went down for all tasks. In the best case, the success rate dropped from 99% to 94%. In the worst case, the success rate for door entrance task 2 dropped from 99% to 89%. However, it accomplished the tasks in most cases.

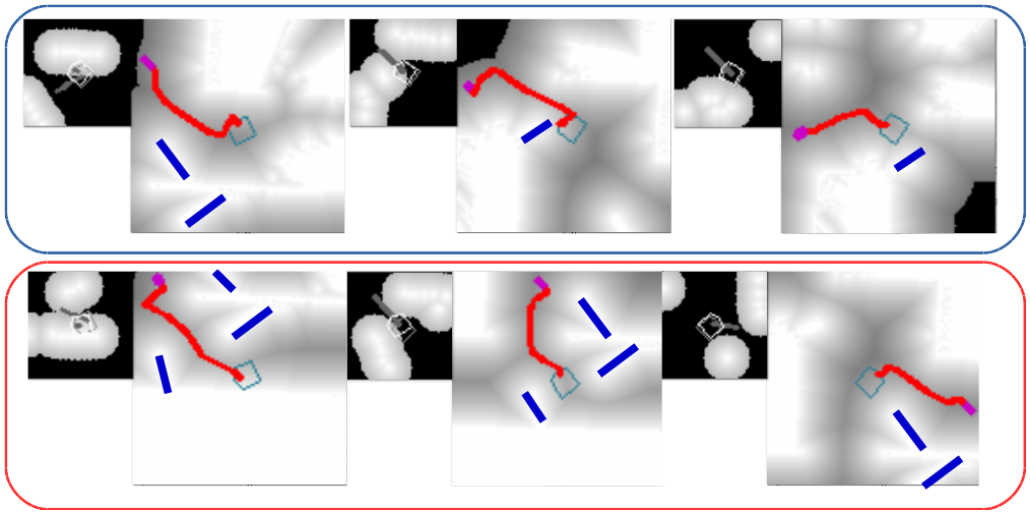

Figure 7. Snapshots of the agent decisions for unseen obstacles. The blue sticks are the synthetic obstacles that have not been seen before.

Table 3. Evaluation results for unseen obstacles.

Task 1	Task 2	Task 3	Task 4
93%	94%	92%	89%

4.2.5. Evaluation with the New Scene

In this section, we initialized a new model with the parameter trained in Section 4.2.3 and applied it to a new scene. The scene was again split into four parts, and each one represented a specific task. As shown in Figure 8, the training process of the model $\pi_{\theta_{adapted}}$ which adapted the old parameters was compared with the model that learned from scratch $\pi_{\theta_{scratch}}$. Although, in both cases, they converged to a total reward around 200, and the adapted model learned two times quicker. This result can be found by looking at the curve slope in the early stage of training. The adapted model reached the plateau in less than 50 episodes, whereas its competitor reached the same in 100 episodes. Table 4 summarizes the success rate for the four sub-tasks in the new scene. In both cases, they have completed the task with a high success rate (at minimum 95%).

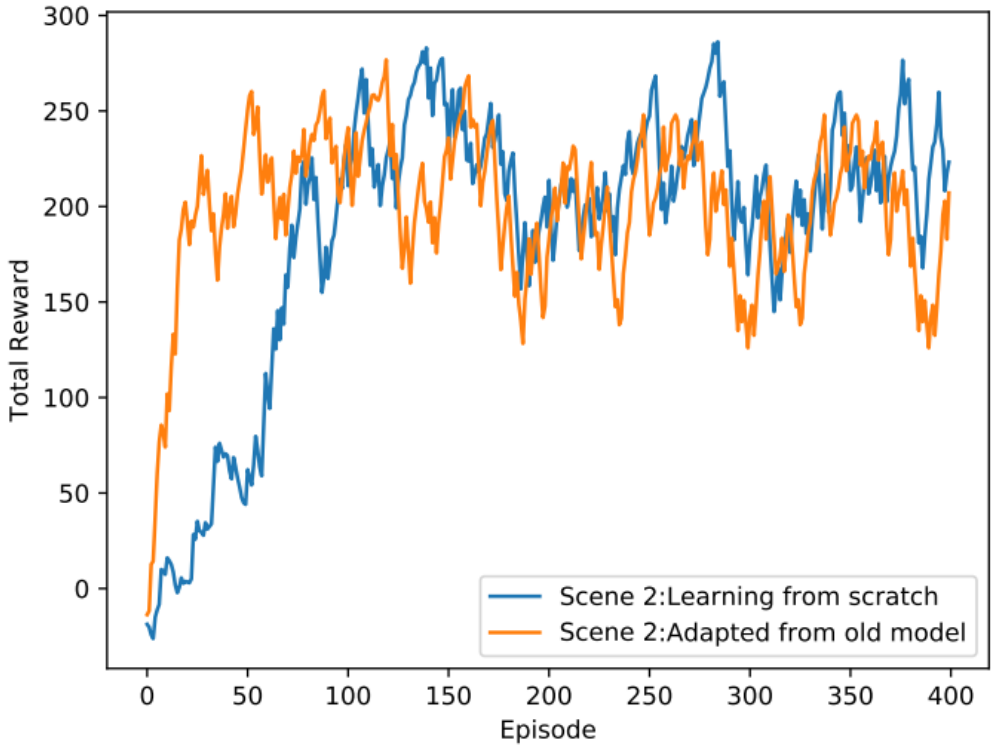

Figure 8. Evaluation result with the new scene.The learning process of scene 2 by using the old parameters derived from scene 1. It was compared with the result, which was learned from scratch.

Table 4. Evaluation results for scene 2. The table summarized the success rate of the the navigation task by using the new model $\pi_{\theta_{scratch}}$ and the adapted one $\pi_{\theta_{adapted}}$.

Parameter	Task1	Task2	Task3	Task4
$\pi_{\theta_{scratch}}$	97%	95%	96%	95%
$\pi_{\theta_{adapted}}$	98%	96%	97%	96%

4.3. Discussion

The results in Section 4.2.2 showed that the proposed APPO method outperformed the baseline algorithms in both the training stage and testing stage. The end-to-end method cannot reach a satisfactory performance. This indicated that the agent could not learn a generalized model in an end-to-end fashion, or at least it was not easy to train. As shown in the training process, it converged in the training process but it could not fulfill the task. For instance, in the door entrance task, it used a global target instead of a local target; therefore, when the agent tried to go across the door, it always turned early and hit the wall.

This indicated that the model may be trapped in the local minimum of the optimization problem and could not escape (i.e., it could not learn a generalized obstacle avoidance behavior in this situation, since the reward was contradicting against the right movements). This was a common scenario in every optimization task; however, the solutions were mostly related to fine tuning the learning parameters such as learning rate or batch size. This made the method implausible in the real world. However, the proposed method did not have this issue; it converged with a higher total reward than the ADDPG method and it reached at least a 93% success rate in the testing tasks. We did not implement any recovery behavior in the experiment, which means if it hit, it restarted. However, the local planner is always used with a recovery planner, since the real-world scenario is unpredictable. We have noticed that most of the fail times of our method were mostly related to a start heading that was completely opposite to the target in a narrow corridor. This means that the agent had to learn to rotate at the beginning. However, such behavior sometimes was hard coded at the beginning of the local planner. We wanted the agent to learn such a behavior (as well as the recovery behavior) from the objective function rather than trick it. If we use them, we can achieve an even better success rate.

The results in Section 4.2.3 indicated that the model performance could be improved by increasing the task complexity. This was because the harder version of the task gave more perturbation on the action spaces so that the agent could be able to learn a better model from these new experiences. By closely looking at the decomposition behavior of the agent, it had evolved the capability of escaping from a difficult start pose and finished the task normally. It also had the capability of moving through a narrow corridor or a door entrance. These results indicated that the deep RL agent had gained some useful skills of solving the navigation task.

Moreover, the results from Section 4.2.4 showed that our agent was able to deal with unseen obstacles by using the old model. This was mainly because when the new obstacle was detected, the local target changed. This leads to a change of decision from the deep RL agent. It uses local information to resolve a decision path to get around the new obstacle. However, the degradation of the performance does happen. This is because the model still lacks generalization capability for those new states. This can be potentially resolved by adding more state disturbance during the training stage.

Furthermore, the results from Section 4.2.5 showed the capability of the model to continuously improve in an unseen map. The results indicated that the new scene can adapt the model parameters learned from the old scene without learning from scratch. This is because the state we designed in general worked for different scenarios. In another word, the state was mostly related to the local information rather than the global one. This made it focus on solving the obstacle avoidance problem locally and left the global plan as it is. Of course, the performance of the global planner had influenced our method, but the result showed that our combinations could work robustly. A better global planner, such as the hybrid A* algorithm, can be a good candidate.

In this paper, we guarantee the data diversity from the mechanism of the training procedure, and we use the off-policy method for training. Compared to the on-policy method, the off-policy method improves the data efficiency, and the network can be updated without waiting for the end of each episode. At the same time, we adopt the asynchronous training method, which also ensures the data's diversity and provides the robustness of the algorithm.

Although the proposed method made a contribution to a robust and promising local planner in the complete navigation framework, there are still limitations. This paper features dynamic path planning; we studied obstacle avoidance and unseen obstacles, but we did not test dynamic obstacles. In order to get these results, we select kinematic simulation, but the dynamic model has not be included. Furthermore, we have not combined all the three traditional modules of mapping, positioning, and navigation into one learning model. In addition, we cannot use the transfer learning and meta-learning to achieve the zero-shot or few-shot. There is still some work to be done in the future: A dedicate training

scheme is needed to help the deep RL learner better deal with the rotation behavior. This is especially helpful when the agent needs to escape from a narrow path, but its heading is completely opposite to the target. Two real scenes were used in this paper; however, it can be tested in more environments to evaluate the learning process, which might go wrong. It can be interesting to study whether the agent can recover from it or how to avoid it. Furthermore, the proposed method can also be trained as a standalone recovery plug-in in the local planner.

5. Conclusions

In this study, we proposed a novel learning scheme for training the agent to deal with a complex environment by adapting the PPO method. The results show that with the global planner, the robot is less likely to be trapped in deadlock points, which yields a higher success rate (about 96%) of reaching the desired goal and asymmetrical performance. The robot success rate is about 35% higher than the end-to-end method, and it is about 5% higher than the ADDPG method. Moreover, the success rate of the robot adapted to the old parameters was just 1% lower than with a robot learning from scratch. Therefore, we think that the model could be generalized to unseen environments without learning from scratch. There are several directions for extending the present study.

One future work will focus on training the proposed method as a standalone recovery plug-in in the local planner, and it also should be tested in more environments to evaluate the learning process. The model will enable the warehouse to realize automatic management through intelligent sorting and handling, and it will be suitable for various customized application scenarios.

This study also has certain application prospects: warehouse management, inventory management, and fashion retail management are important application scenarios. Application of the laser-based navigation can reduce physical demands, offer speed and strength, and take over aspects of warehouse work that are tedious or dangerous.

Author Contributions: Conceptualization, Y.Z., Q.L. and T.W.; methodology, Y.Z.; software, Y.Z.; formal analysis, Y.Z. and K.X.; writing—original draft preparation, Y.Z. and K.X.; writing—review and editing, Y.L. and Q.L.; visualization, Y.Z.; supervision, Q.L. and T.W.; funding acquisition, Y.L. and Q.L. All authors have read and agreed to the published version of the manuscript.

Funding: This project was supported by the National Natural Science Foundation of China (61902037, 11902350), the Fundamental Research Funds for the Central Universities (500419804), the Key Deployment Projects of Chinese Academy of Sciences (KJZD-EW-G20-03), Beijing Intelligent Logistics System Collaborative Innovation Center Open Project (BILSCIC-2018KF-09), and the National Center for Mathematics and Interdisciplinary Sciences, CAS.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: For helpful comments and discussions, we thank Liulei Shen, JiZhou Huang, Fan Wang, and Ziheng Niu.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Summarizes of the related work.

Author	Contribution	Limitation
Zhu et al. [6]	Navigation of mobile robots based on RGB-camera sensors	Do not consider the dynamics in the real-world environments [t]
Wang et al. [28]	Propose a planned-ahead hybrid reinforcement learning model to bridge the gap between synthetic studies and real-world practices	Incapable of accomplish navigation for complex robot dynamics and environments
Zhou et al. [29]	Present a goal-directed robot navigation system that integrates global planning based on goal-directed end-to-end learning and local planning based on reinforcement learning	Learning algorithms for continuous action spaces is not considered
Misra et al. [30]	Use reinforcement learning in a contextual bandit setting to train a neural network agent	Do not consider the dynamics in the real-world environments
Zeng et al. [31]	Propose an improved A3C (IA3C) algorithm to learn the control policies of the robots' local motion	Complex behaviors of moving obstacles is not considered
Chen et al. [32]	Propose a map-based deep reinforcement learning approach for multi-robot collision avoidance in a distributed and communication-free environment	More dynamic and crowding environment is not considered
Doukhi and Lee [33]	Present a novel approach for enabling a micro aerial vehicle system equipped with a laser range finder to autonomously navigate among obstacles and achieve a user-specified goal location in a GPS-denied environment, without the need for mapping or path planning	The problem of dynamic obstacle avoidance in 3D space is not considered
Han and Kim [34]	Propose a lane detection algorithm using a laser range finder for the autonomous navigation of a mobile robot	Different environmental conditions is not considered
Elfakharany and Ismail [35]	Present a novel deep reinforcement learning based method that is used to perform multi-robot task allocation and navigation in an end-to-end fashion, without the need to construct a map of the environment	More complex environments is not considered [b]

References

1. Ramirez, G.; Zeghloul, S. A new local path planner for nonholonomic mobile robot navigation in cluttered environments. In Proceedings of the 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation, Symposia Proceedings (Cat. No. 00CH37065), San Francisco, CA, USA, 24–28 April 2000; Volume 3, pp. 2058–2063.
2. Oleynikova, H.; Honegger, D.; Pollefeys, M. Reactive avoidance using embedded stereo vision for mav flight. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 50–56.
3. Sanchez-Lopez, J.L.; Wang, M.; Olivares-Mendez, M.A.; Molina, M.; Voos, H. A real-time 3d path planning solution for collisionfree navigation of multirotor aerial robots in dynamic environments. *J. Intell. Robot. Syst.* **2019**, *93*, 33–53. [CrossRef]
4. Levine, S.; Finn, C.; Darrell, T.; Abbeel, P. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.* **2016**, *17*, 1–39.

5. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36.
6. Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J.J.; Gupta, A.; Fei-Fei, L.; Farhadi, A. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on robotics and Automation (ICRA), Sands Expo and Convention Centre, Marina Bay Sands, Singapore, 29 May–3 June 2017; pp. 3357–3364.
7. Wang, F.; Zhou, B.; Chen, K.; Fan, T.; Zhang, X.; Li, J.; Tian, H.; Pan, J. Intervention aided reinforcement learning for safe and practical policy optimization in navigation. In Proceedings of the 2nd Annual Conference on Robot Learning, CoRL 2018, Zurich, Switzerland, 29–31 October 2018; pp. 410–421.
8. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the 4th International Conference on Learning Representations, Conference Track Proceedings, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016.
9. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.A.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
10. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M. et al. Mastering the game of go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
11. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.* **2013**, *32*, 1238–1274. [CrossRef]
12. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmassan, Stockholm, Sweden, 10–15 July 2018; pp. 1582–1591. Available online: http://proceedings.mlr.press/v80/fujimoto18a.html (accessed on 18 July 2021).
13. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.I.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015; pp. 1889–1897.
14. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
15. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the 33th International Conference on Machine Learning (ICML), New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
16. Gu, S.; Holly, E.; Lillicrap, T.P.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, 29 May–3 June 2017; pp. 3389–3396.
17. Irpan, A. Deep Reinforcement Learning Doesn't Work Yet. 2018. Available online: https://www.alexirpan.com/2018/02/14/rl-hard.html (accessed on 18 October 2021).
18. Deisenroth, M.; Rasmussen, C.E. Pilco: A model-based and data-efficient approach to policy search. In Proceedings of the 28th International Conference on Machine Learning (ICML), Bellevue, WA, USA, 28 June–2 July 2011; pp. 65–472.
19. Malavazi, F.B.; Guyonneau, R.; Fasquel, J.-B.; Lagrange, S.; Mercier, F. Lidar-only based navigation algorithm for an autonomous agricultural robot. *Comput. Electron. Agric.* **2018**, *154*, 71–79. [CrossRef]
20. Sampedro, C.; Bavle, H.; Rodriguez-Ramos, A.; de la Puente, P.; Campoy, P. Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, Spain, 1–5 October 2018; pp. 1024–1031.
21. Qin, H.; Bi, Y.; Feng, L.; Zhang, Y.; Chen, B.M. A 3d rotating laser-based navigation solution for micro aerial vehicles in dynamic environments. *Unmanned Syst.* **2018**, *6*, 297–305. [CrossRef]
22. Perez-Higueras, N.; Ramon-Vigo, R.; Caballero, F.; Merino, L. Robot local navigation with learned social cost functions. In Proceedings of the 2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Vienna, Austria, 1–3 September 2014; Volume 2, pp. 618–625.
23. Jeni, L.A.; Istenes, Z.; Szemes, P.; Hashimoto, H. Robot navigation framework based on reinforcement learning for intelligent space. In Proceedings of the 2008 Conference on Human System Interactions, Krakow, Poland, 25–27 May 2008; pp. 761–766.
24. Macek, K.; PetroviC, I.; Peric, N. A reinforcement learning approach to obstacle avoidance of mobile robots. In Proceedings of the 7th International Workshop on Advanced Motion Control. Proceedings (Cat. No. 02TH8623) . Maribor, Slovenia, 3–5 July 2002; pp. 462–466.
25. Kim, B.; Pineau, J. Socially adaptive path planning in human environments using inverse reinforcement learning. *Int. J. Soc. Rob.* **2016**, *8*, 51–66. [CrossRef]
26. Gil, O.; Sanfeliu, A. Effects of a social force model reward in robot navigation based on deep reinforcement learning. In Proceedings of the Robot 2019: Fourth Iberian Robotics Conference, Porto, Portugal, 20–22 November 2019; pp. 213–224.
27. Gao, W.; Hsu, D.; Lee, W.S.; Shen, S.; Subramanian, K. Intention-net: Integrating planning and deep learning for goaldirected autonomous navigation. In Proceedings of the 1st Annual Conference on Robot Learning, CoRL 2017, Mountain View, CA, USA, 13–15 November 2017; pp. 185–194.
28. Wang, X.; Xiong, W.; Wang, H.; Wang, W.Y. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In Proceedings of the ECCV 2018 European Conference on Computer Vision, Munich, Germany, 8–14 September 2018; pp. 38–55.

29. Zhou, X.; Gao, Y.; Guan, L. Towards goal-directed navigation through combining learning based global and local planners. *Sensors* **2019**, *19*, 176. [CrossRef] [PubMed]
30. Misra, D.; Langford, J.; Artzi, Y. Mapping instructions and visual observations to actions with reinforcement learning. *arXiv* **2017**, arXiv:1704.08795.
31. Zeng, J.; Qin, L.; Hu, Y.; Yin, Q.; Hu, C. Integrating a path planner and an adaptive motion controller for navigation in dynamic environments. *Appl. Sci.* **2019**, *7*, 1384. [CrossRef]
32. Chen, G.; Yao, S.; Ma, J.; Pan, L. Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning. *Sensors* **2020**, *20*, 4836. [CrossRef] [PubMed]
33. Doukhi, O.; Lee, D.J. Deep reinforcement learning for end-to-end local motion planning of autonomous aerial robots in unknown outdoor environments: Real-time flight experiments. *Sensors* **2021**, *21*, 2534. [CrossRef] [PubMed]
34. Han, J.H.; Kim, H.W. Lane detection algorithm using lrf for autonomous navigation of mobile robot. *Appl. Sci.* **2021**, *11*, 6229. [CrossRef]
35. Elfakharany, A.; Ismail, Z.H. End-to-end deep reinforcement learning for decentralized task allocation and navigation for a multi-robot system. *Appl. Sci.* **2021**, *11*, 2895. [CrossRef]