Multi-Stage Path Planning Strategy for Intelligent Cleaning Robot

Xingxing Cheng[a], Xianfeng Ding[b], Chia E. Tungom[c] and Ji Yuan[d]
Onewo Space-Tech Service Co., Ltd., Shenzhen, China

Keywords: Garbage Clearance, TPP, Yolov5, MDDPG.

Abstract: The clearance of public garbage is a challenging case in artificial intelligence implementation. The difficulty is how to recognize the public garbage and make path planning. In comparison with Traditional Path Planning (TPP) architecture, we utilized a method of Multi-stage Path Planning for garbage clearance to improve the accuracy and speed of TPP architecture. Within this paper, the original public video frames are taken as input and the garbage is separated into several classes by Yolov5. Its location is estimated by the location of the camera. Taking the garbage class and location as input, the path planning is calculated by an improved Multi-stage Deep Deterministic Policy Gradient (MDDPG). Our novel architecture was trained and tested using videos from a real community place and achieved ideal effects.

1 INTRODUCTION

With the fast growth of the application of the artificial intelligence technology in last decade, mobile robots have become smarter. As known that, path planning is an important topic in the mobile robots' research field. The goal of path planning is both to perfect keep away from obstacles and to fast plan the best moving path to the target location. According to the literature review, there are two main algorithms for the path planning of mobile robots, namely the traditional and reinforcement-learning-based path planning algorithms.

As the traditional algorithm, the A* algorithm is a correct path-planning algorithm based on a heuristic function. The results show that the A* algorithm has a good deceleration effect in processing time, and its running speed is fast (Guruji et al., 2016; Shukla et al., 2008; Kala et al., 2009; Weiteng et al., 2013). Nevertheless, there still exist some problems during the application, such as the length of the path, the risk of colliding with obstacles, and traffic isolation rules, etc. Ant Colony Optimization (ACO) is defined as a kind of constructive meta-heuristic algorithm for suitable path planning (Mirjalili et al., 2020; Dorigo et al., 1991; Brand et al., 2010; Liu et al., 2017).Notice that, local optimization, slow convergence, and low search efficiency are frequent problems caused by ACO.

The Artificial Potential Field (APF) method provides a simple and effective strategy for robot navigation. However, there is a local minimum problem by applying this method (Khatib, 1985; Sun et al., 2019). Genetic algorithm has been widely used in path planning of mobile robots and has achieved good results (Zhang et al., 2016; Karami and Hasanzadeh, 2015). Notice that, the genetic algorithm has some drawbacks in obstacle environment, i.e. path infeasibility. Traditional robot path planning and control methods, they cannot deal with emergency circumstances, extract information from environmental perception and analysis, and make decisions about appropriate action.

Deep reinforcement learning (DRL) applied in the path planning can be separated into two main subtype algorithms: value function algorithm (Liu et al., 2018) and strategy gradient algorithm (Liu et al., 2019). The most important four value function algorithms for robot path planning are DQN, IDDDQN, Dueling DQN, and D3QN. For instance, Tai et al. proposed a DQN algorithm to implement robot path planning in a virtual environment (Tai and Liu, 2016). There are several drawbacks of overestimation and poor stability of the algorithm. Sichkar et al. proposed the IDDDQN algorithm to improve the network's convergence speed, and results are shown that, in an unknown complex environment, this algorithm can obtain an optimal path better (Sichkar, 2019). In the paper from Wen et al., a full convolutional resid-

[a] https://orcid.org/0000-0003-3608-1984
[b] https://orcid.org/0000-0002-2990-4391
[c] https://orcid.org/0000-0003-3708-4044
[d] https://orcid.org/0000-0002-4369-647X

Cheng, X., Ding, X., Tungom, C. and Yuan, J.
Multi-stage Path Planning Strategy for Intelligent Cleaning Robot.
DOI: 10.5220/0012046400003612
In *Proceedings of the 3rd International Symposium on Automation, Information and Computing (ISAIC 2022)*, pages 757-765
ISBN: 978-989-758-622-4; ISSN: 2975-9463

ual network and the dueling DQN algorithm are applied for obstacle detection and path planning, respectively ((Wen et al., 2020)). This algorithm can help robots to recognize and keep away from static obstacles in the complex environment. Wen et al. (Wen et al., 2020) proposed a dual-depth q network obstacle avoidance algorithm (D3QN), which can be trained in the virtual environment, and directly applied in the complex unknown environment (Xie et al., 2017). The DRL algorithm based on the value function can execute continuous action decisions unless the action strategies are independent. Then, the strategy gradient method is adopted for DRL. In robot path planning, the strategy gradient algorithms mainly include TRPO, PPO, and DDPG. Lillicrap et al. proposed a DDPG algorithm which apply the DQN estimation function based on the DPG algorithm. It can be utilized for continuous state and action space, and straightforwardly improving the move stability (Lillicrap et al., 2015). Since TRPO algorithm have some drawbacks, such as the strategy and environment are too large and it can produce large errors easily. To overcome aforedmentioned problems, Schulman et al. proposed a neighborhood strategy optimization algorithm based on the TRPO algorithm (Schulman et al., 2017). The robot path planning algorithm which utilized a random gradient replace the common policy gradient to optimize the transformation of the objective function with sample data interacting with the environment, which has fine information robustness and efficiency.

According to previous research, DRL algorithms outperform the traditional algorithm in solving the mobile robot path planning problem. In the discrete action strategy scenario, the value function-based DRL algorithm can make continuous action decisions. However, the stability and convergence speed via online learning is an urgent issue which is worth to be investigating. Since it is difficult to solve a sophistical multi-stage design problem with a single algorithm. Multi-functional floor cleaning robot has gained popularity in our daily life (Milinda and Madhusanka, 2017; Milinda and Madhusanka,). Moreover, there are few types of researches on the planning and cleaning of the garbage sorting path planning. This paper is devoted to the research of the multi-step path planning for a cleaning robot. The main contributions of this paper can be summarized as follows:

- 1. Proposed MDDPG algorithm to speed up model convergence:

  1.1 Adopt multi-strategy network, centralized value network, value network receives data generated by multiple strategy networks at the same time, and improves the efficiency of value network estimation;

  1.2 Divide the experience playback pool by priority to speed up the convergence of the model;

  1.3 Design the reward function for the intermediate starting point to improve the convergence speed and degree of the reinforcement learning algorithm;
- 2. The classification of garbage is defined, and the garbage classification model is trained based on an improved YOLOv5;
- 3. We build a multi-stage garbage path planning model to improve the generalization of garbage path planning problems.

The rest of this paper is organized as follows. The DDPG is given in Section 2. Our proposed method is in Section 3. The experimental results and analysis are present in Section 4, and conclusions and future work can be found in Section 5.

2 DEEP DETERMINISTIC POLICY GRADIENT(DDPG)

Traditional reinforcement learning algorithms use tables to record value functions, and once dealing problems in cases with high states or action space exploded, it will cause a dimensionality disaster. On the other hand, deep reinforcement learning parameterizes the value function or policy function and makes full usage of the representation ability of the neural network to fit the value function or policy function. Therefore, scholars combine deep learning to propose deep reinforcement learning. This improvement enables deep reinforcement learning having a good performance in cases with high-dimensional and continuous-state spaces.

2.1 AC Network

The Actor-Critic framework(AC network) utilizes a neural network to approximate the value function and the policy function at the same time, its schematic diagram is shown in Figure 1. The AC network contains two neural networks: actor-network and the critic network. The actor-network is responsible for fitting the current policy function and outputting corresponding actions according to the state of the input, while the critic network is responsible for estimating the value function, according to the state or state of the input. The action pair outputs the corresponding state or action value. Actor and critic neural networks respectively parameterize the policy function and the value

function, and the value function can choose the state value function or the action value function, as shown in Eq.1. The parameterization of the strategy function is shown in Eq.2:

$$\phi_\pi(s) = \phi(s,w) Q_\pi(s,a) = Q(s,a,w) \quad (1)$$

$$\pi(s,a) = P[a|s,\theta] \quad (2)$$

The ultimate training goal of the AC network is to obtain an optimal policy, in other words, an actor-network that can represent the optimal policy. The training method of the network adopts gradient ascent, and the network parameters are updated according to the objective function of the actor-network. The objective function, parameter update, and gradient of the objective function of the actor-network are shown in Eq.3:

$$\begin{aligned} V(\theta) &= \sum_s \phi_{\pi_\theta}(s) d^{\pi_\theta}(s) \\ \theta &= \theta + B\Pi_\theta V(\theta) \\ \Pi_\theta V(\theta) &= \Pi_\theta log\pi_\theta(s,a)\cdot \end{aligned} \quad (3)$$

In Eq.3, the score function $\Pi_\theta log\pi_\theta(s,a)$ is widely applied in the field of machine learning and is easy to calculate. Hence, actor-networks can be trained as long as the action-value function can be accurate estimated.

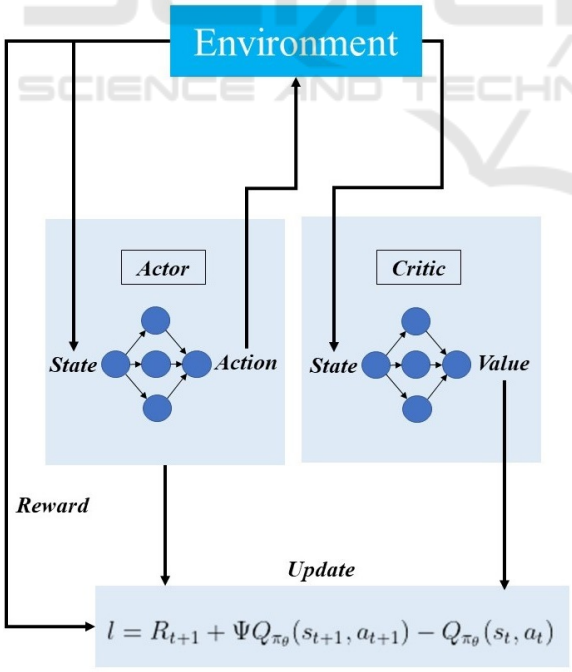

Figure 1: AC network schematic diagram.

In the actor-critic framework, the critic network is responsible for estimating the value function to assist the actor-network in training. During actor-critic training, actor and critic networks are trained simultaneously, and the critic network is updated by calculating TD error:

$$l = R_{t+1} + \Psi Q_{\pi_\theta}(s_{t+1}, a_{t+1}) - Q_{\pi_\theta}(s_t, a_t) \quad (4)$$

$$dw = \wp \frac{\pounds l^2}{w} \quad (5)$$

2.2 DDPG

DDPG methods include behavior criticism, experience replaying, objective network, and deterministic strategy gradient theorem. It proves the existence of deterministic strategies $\mu(\omega) : S \to A$, and generates the exact actions of the agent based on the given state and not based on the probability distribution of all actions. In this method, the performance target is defined as follows:

$$\begin{aligned} J(\pi_\omega) &= \int_S \rho^\pi(s) \int_A \pi_\omega(s,a) r(s,a) da ds \\ &= E_{s\sim\rho^\pi, a\sim\pi_\omega}[r(s,a)], \end{aligned} \quad (6)$$

where $\rho^\pi(s)$ denotes the state distribution. The objective with deterministic policy is:

$$\begin{aligned} J(\mu_\omega) &= \int_S \rho^\mu(s) r(s, \mu_\omega(s)) ds \\ &= E_{s\sim\rho^\mu}[r(s, \mu_\omega(s))], \end{aligned} \quad (7)$$

The state-action value or state-action critic network $Q(s_t, a_t, \theta)$ and the actor-network $\mu(s_t, \omega)$, in which the θ and ω as a parameter of neural network) approximates the state action value function and action function in this method, respectively. Once the network is updated, the training experience comes from experience playback. It is commonly a buffer that stores a tuple of four elements (s_t, a_t, r_t, s_{t+1}), and provides a batch of updates for the network of actors and film critics. As the buffer is full, the updated experience will replace the oldest one, hence only a limited number of the latest experience will remain. In addition, by providing a constant target in the training process, the target network updates the critic network. The target network is usually constructed as a copy of the critic network. The target network, denoted as Q^{tar}, is applied as a replacement of $\acute{Q}(s_{t+1}, a_t)$ in Eq.8:

$$L_{tar}(\theta) = (r(s_t, a_t) + \gamma Q^{tar}(s_{t+1}, a_{t+1}, \theta^-) - Q(s_t, a_t, \theta)) \quad (8)$$

where the θ^- is the parameter of the previous iteration. Experience replay and target network are vital for training stable DDPG methods and creating the deep neural network possible.

3 OUR PROPOSED METHOD

Our proposed method is shown in Fig.. It can be seen that the YOLOv5-MDDPG schematic diagram mainly included a garbage recognition and a path planning module.

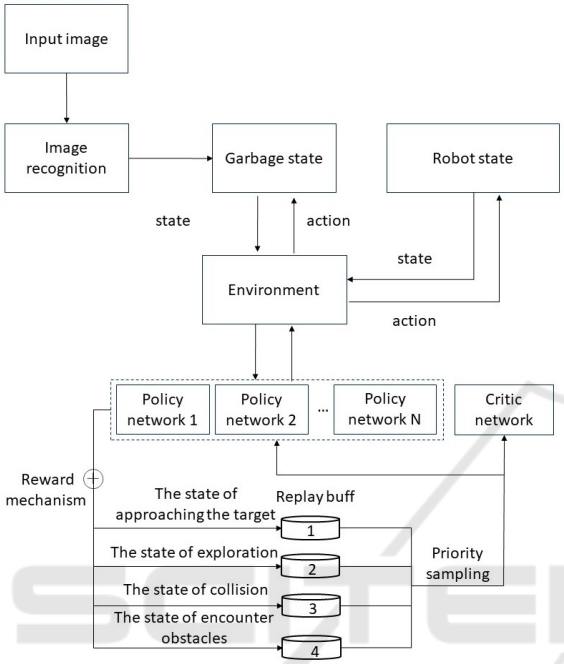

Figure 2: YOLOV5-MDDPG schematic diagram.

3.1 YOLOV5

YOLOv5 (Zhao et al., 2022) is one of the prevalent target detection models with high accuracy. Within this framework, as a regression problem, object detection trained end-to-end, it enables training quickly and achieved competitive performance. For YOLOv5, the main contribution is to port YOLO from the dark web neural network framework(Chen et al., 2019) to PyTorch. This allows 16-bit floating-point calculations to be used insteading of 32-bit ones, further, greatly reducing the reasoning time. YOLO consist of three main parts. The body mesh produces image features of different resolutions, the neck mixes different features and combines it, and the head uses mixed features to forecast classes and boxes. Its loss function includes a combination of GIoU, object loss, and class loss. In particular, the framework of YOLOv5 adopts a bottleneck based on a phase-to-phase local network (CSPNet)(Alkhamaiseh et al., 2021). CSPNet aims to reduce the number of gradient computation iterations found in densenets-based networks(Tian et al., 2019). This reduces training time and allows smaller models to have less impacts on performance. Detection is considered as a regression problem in the YOLO models. Each anchor box has x and y coordinates, width, and height. It forecasts how much each feature should modify. If it is approaching the anchor box, the box contains the target object. This implementation uses k-means and a genetic learning algorithm to first learn the anchor boxes from the distribution of boxes in the training dataset. This is useful in this case because the appearance of tear gas canisters is limited. By using an anchor frame that better fits the size of the tear gas canister, the model learns faster to forecast the correct containment area of the tear gas canister. In addition to using CSPNet and learning anchor boxes, YOLOv5 utilizes specific data augmentation techniques which are useful for applications. The most notable data extension is the mosaic data extension, which was the first used in YOLOv4 (Bochkovskiy et al., 2020). This enhancement technique combines four different images side by side at different scales into one image. Like random throttling, this helps the network handle congestion better. This also makes it easier to use, as many tear gas canisters are blocked usually. Moreover, this data addition method merges objects of various classes into a single image. It is useful for real-life images that may contain different types of tear gas canisters. Lastly, by combining various images, the image background becomes more manifold. This could be considered a type of domain randomization, and previous research has shown it to be very useful in this application(Tobin et al., 2017).

3.2 MDDPG

3.2.1 MDP

After obtaining the state information such as the category and location of garbage, it is used as the input of the subsequent model. Established a cleaning robot path planning model based on MDDPG, and describe the cleaning robot path planning problem as a Markov decision process (MDP), that is, a sequence decision problem. At the same time, the basic MDP problems such as state, action, and reward in the process of cleaning robot path planning are defined. The state space of the MDDPG algorithm(S_t) includes: lidar data (SA), the current control command of the cleaning robot (SC), the control command of the mobile robot at the last moment, the orientation of the target point (SD) and the distance (SE); the current control command of the cleaning robot refers to the angular velocity (SF) and linear velocity (SP) of the cleaning

robot, as shown in the Eq.9. The action space of the MDDPG algorithm(a_t) includes: the angular velocity of the cleaning robot body coordinate system rotating around the Z axis(SF_z) and the linear velocity along the X axis(SP_x), as shown in the Eq.10.

$$S_t = SA + SC + SD + SE + SF + SP \quad (9)$$

$$a_t = SF_z + SP_x \quad (10)$$

3.2.2 AC Network

The MDDPG algorithm utilizes a multi-actor-network. Generate data by combined N actor-networks, which improves the stability of the data, reduces the uncertainty of data, and is conducive to speeding up the convergence speed of the model and improving its convergence depth. Each actor-network is divided into the current network fatherly(responsible for the iterative update of the actor-network parameters) and the critic network (responsible for selecting the optimal next action based on the experience of replaying the next state sampled in the replay buffer), where the current network loss function($H(\omega)$) is shown in the Eq.11:

$$H(\omega) = \frac{1}{k} \bullet \frac{1}{N} \sum_{i=1}^{N} \sum_{l=1}^{m} (y - Q(\varphi(s_{li}), a_{li}, \omega))^2 \quad (11)$$

where k,y are constants, N is the number of the actor-network, Q is the target value, $\varphi(s)$ is the function of the state value, a is action, ω is the trainable parameter after parameterization of the actor-network function.

3.2.3 Reward design

The reward function of the MDDPG algorithm is designed as follows whether the cleaning robot reaches the garbage, whether the distance between the cleaning robot and the target point changes, and whether it collides and encounters obstacles.

The reward function designed considered whether the cleaning robot reaches the garbage is shown in Eq.(12) :

$$G_{arr} = \begin{cases} g_{arr} & \text{if } d_t \le d_1 \\ 0 & \text{if } d_t > d_1 \end{cases} \quad (12)$$

The reward function designed considered the distance between the cleaning robot and the target point changes is shown in Eq.13 :

$$G_{dis} = \begin{cases} g_{dis} & \text{if } d_t - d_{t-1} < 0 \\ -g_{dis} & \text{if } d_t - d_{t-1} \le 0 \end{cases} \quad (13)$$

The reward function designed considered whether it collides is shown in Eq.14 :

$$G_{col} = \begin{cases} g_{col} & \text{if } d_i \le d_2 \\ 0 & \text{if } d_i > d_2 \end{cases} \quad (14)$$

The reward function designed considered whether it encounters obstacles is shown in Eq.15 and Eq.16:

$$\Gamma_\sigma J = \frac{1}{M} \sum_{i=1}^{M} \Gamma_\sigma log \pi_\sigma(\varepsilon^i)[G(\varepsilon^i, r_0^i) - G(\pi_i, r_0^i)] \quad (15)$$

$$r_0 : G_{min} < G(\pi_i, r_0^i) < G_{max} \quad (16)$$

The total reward function is shown in Eq. (17):

$$G = G_{arr} + G_{dis} + G_{col} + \Gamma_\sigma J \quad (17)$$

3.2.4 Replay Buffer

DDPG utilizes a replay buffer to eliminate strong correlations between input experiences. Here, experience refers to a quadruple (s_t, a_t, r_t, s_{t+1}). Meanwhile, DDPG applies the target network method to stabilize the training process. As a fundamental part of the DDPG algorithm, the replay buffer greatly affects the training speed and finally effect of the network.

We have improved the replay buffer of the MDDPG algorithm, and divided the replay buffer by priority. During the training process of the cleaning robot, the training data obtained is put into four replay buffers with different priorities. Through prior knowledge, unnecessary search time is reduced, and the convergence speed of the model is improved.

(1) When the cleaning robot reaches the garbage location (i.e. target point), it puts the acquired training data into the replay buffer 1 with the highest priority;

(2) When the cleaning robot is in the exploration stage, put the acquired training data into the replay buffer 2 with the second highest priority;

(3) When the cleaning robot collides, put the acquired training data into the replay buffer 3 with lower priority;

(4) When the cleaning robot encounters an impassable obstacle (such as a ditch, a river, etc.), the obtained training data is put into the replay buffer 4 with the lowest priority.

The training data includes status information, current time and previous action instruction, and reward value data at a moment. The state information includes Lidar data, and orientation and distance information of the target point.

4 EXPERIMENTAL RESULTS AND ANALYSES

In this paper. we proposed a multi-stage garbage recognition and cleaning, and the results of public garbage recognition and path planning are shown in Fig.3. There are two parts in our method that included improved YOLOv5 for garbage recognition and path planning based on MDDPG.

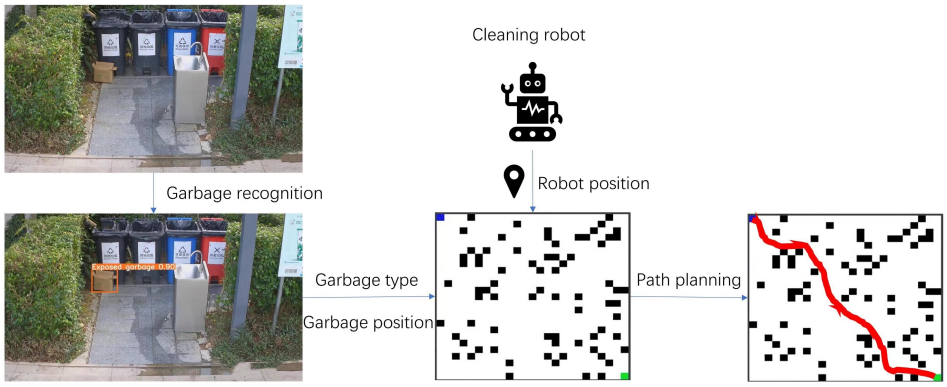

Figure 3: Results of public garbage recognition and path planning.

4.1 Imporoved YOLOv5 for Garbage Recognition

Our improved YOLOv5 model was trained by the Onewo daily images dataset, and the training dataset and the validation dataset are 52,224 and 13,056 respectively. Fig.4 shows the recognition result of 600 rounds of training based on our improved YOLOv5 model. It can be seen from the figure that the model can identify and classify the garbage accurately in different scenarios.

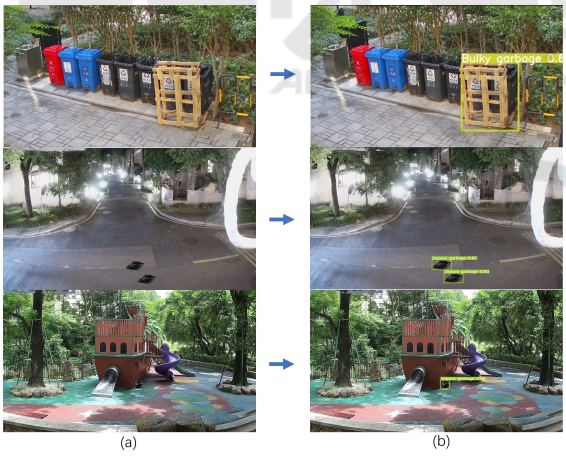

Figure 4: Part of garbage recognition by improved YOLOv5. (a) Original image; (b) Detect image.

Table.1 shows the evaluation information about the experimental data. As can be seen from the table, the validation dataset is 13,056. For class all, the Labels is 38,122, the Precision is 0.929, the Recall is 0.913, the mAP@.5 is 0.922, the mAP@.5:.95 is 0.835; For class other, the Labels is 8,022, the Precision is 0.91, the Recall is 0.906, the mAP@.5 is 0.89, the mAP@.5:.95 is 0.81; For class Bulky garbage, the Labels is 8,843, the Precision is 0.947, the Recall is 0.946, the mAP@.5 is 0.933, the mAP@.5:.95 is 0.834; For class Decoration garbage, the Labels is 8217, the Precision is 0.913, the Recall is 0.94, the mAP@.5 is 0.966, the mAP@.5:.95 is 0.84; For class Green garbage, the Labels is 4114, the Precision is 0.898, the Recall is 0.837, the mAP@.5 is 0.881, the mAP@.5:.95 is 0.803; For class Packed garbage, the Labels is 8,926, the Precision is 0.975, the Recall is 0.938, the mAP@.5 is 0.942, the mAP@.5:.95 is 0.886. By data analysis the experimental results, it can be observed that the recognition accuracy of four class garbage is well, for mAP@.5, Bulky garbage, Decoration garbage, and Packed garbage are all greater than 0.9. Thereby, it has better actual garbage recognition effect. However, in the training dataset, due to the small number of samples, the mAP@.5 of Green garbage is less than 0.9, and it can be optimized by increasing the number of samples.

In this section, as can be seen in figure 5, we analyze the training iterations and the step size of the model by comparing the traditional DDPG algorithm and our proposed MDDPG algorithm.

In this figure, the abscissa represents the iterations of training and the ordinate is the amount of each iteration steps required from the cleaning robot position to the trash position. The green dashed line represents the iterative trend results of the DDPG algorithm and the red solid line is the proposed algorithm iterative trend results. This figure obviously shows the training effect and convergence speed between these two algorithms. It can be seen that, around the 190th round, the the proposed MDPDG algorithm's number of iteration steps starts to decrease, showing a convergence trend gradually, while the DDPG algorithm starts to show a convergence trend about the 470th iteration. Meanwhile, the DDPG algorithm's iteration steps stay at the level of about 220 steps and do not converge to the minimum steps number, while our

Table 1: The evaluation of our trained improved YOLOv5 model.

Class	Images	Labels	P	R	mAP@.5	mAP@.5:.95
all	13056	38122	0.929	0.913	0.922	0.835
other	13056	8022	0.91	0.906	0.89	0.81
Bulky garbage	13056	8843	0.947	0.946	0.933	0.834
Decoration garbage	13056	8217	0.913	0.94	0.966	0.84
Green garbage	13056	4114	0.898	0.837	0.881	0.803
Packed garbage	13056	8926	0.975	0.938	0.942	0.886

proposed MDDPG algorithm converges at about 110 steps. Obviously, the path computed by the DDPG algorithm has a larger number of steps. In the following training process, our proposed MPDDG algorithm has a lower fluctuation frequency and superior stability than the DDPG algorithm. Besides, as can be seen from experimental results, our proposed MDDPG algorithm has both fast iteration speed and strong decision-making ability, and can quickly reach the debris location with fewer steps.

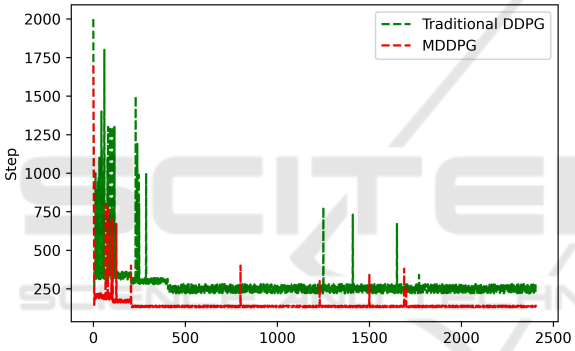

Figure 5: The convergence trend comparison results of step iteration.

Figure 6a shows the result of path planning after 2,500 training rounds by the traditional Q-learning algorithm. Intuitively, due to the size limitation of the Q table, there are many redundant corners in this path and the algorithm training takes a long time. Figure 6b shows the path planning result of 2,500 training rounds by the traditional DDPG algorithm. Intuitively, the planned path is a curve. Since the DDPG algorithm achieves a continuous action value that bringing the robot's action changes in real-time. Since the beach robot needs as few actions as possible during navigation, the path achieved by our proposed method cannot meet the real navigation needs. Figure 6c shows the result of A* algorithm path planning. As shown, the closer the planning path is to the obstacle, the more corners the path has, which increases the risk of the robot moving. Compare the aforementioned algorithm with our proposed MDDPG algorithm (figure 6d). Our proposed MDDPG path planned is smoother,

Table 2: The comparison of experimental data.

Method Path	Length/m	Time/s
Q-learning algorithm	170.238	0.7556
A* algorithm	144.626	1.2365
DDPG algorithm	177.251	0.8011
MDDPG algorithm	165.342	0.5768

maintains a safe distance from obstacles, and is more linear with the robot's real navigation specifications. In addition, based on the experimental data, different performances of algorithms are compared in terms of trip length, trip planning time, and a number of turns. The comparison result is shown in table 2, the length of planning path via algorithm in our research is 165.342m, which takes 0.5768s. The Q-learning algorithm plans path length is 170.238m and takes 0.7556s. The DDPG algorithm planned path length is 177.251m, which takes 0.8011s. The length of the route planned by algorithm A* is 144.626m, which takes 1.2365s. By data analysis of the experimental results, our proposed algorithm meets the actual navigation specifications well, the method has a shorter path planning ability and a good performance in implementation.

5 CONCLUSIONS AND FUTURE WORK

We present the multi-stage framework for public garbage recognition and path planning that accurately recognizes the classes of garbage and location, and subsequently outputs a fast worksheet for path planning. The experimental results show that, for the recognition task, and the best mPrecision is 0.929, the best mRecall is 0.913, the best mAP@.5 is 0.922, the best mAP@.5:.95 is 0.835. For path planning, the best path length planned by this paper's proposed algorithm is 165.342m and the best taking is 0.5768s.

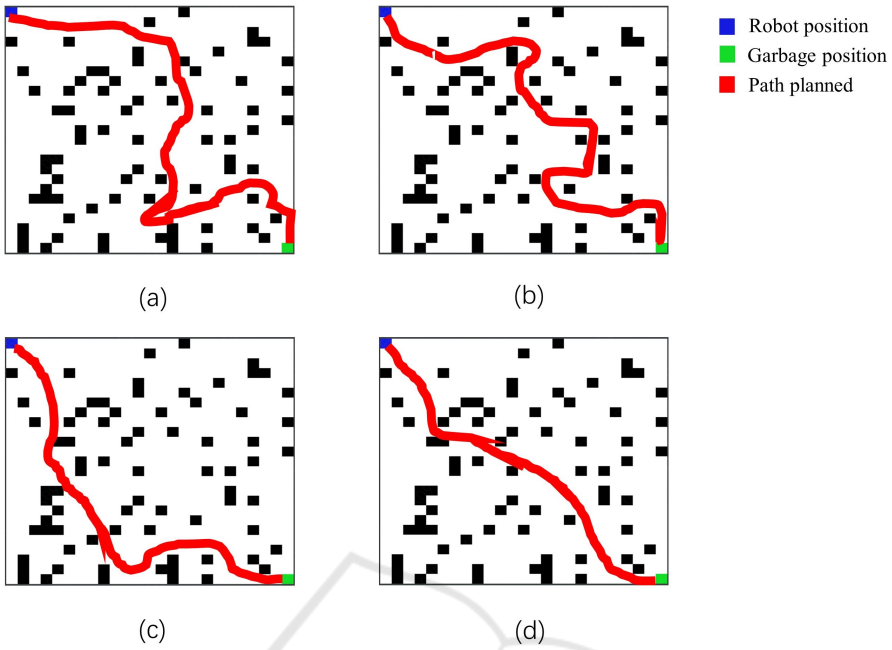

Figure 6: Different algorithm's path planning results. (a) Q-learning algorithm path planning; (b)A* algorithm path planning; (c) Traditional DDPG algorithm path planning; (d) MDDPG algorithm path planning.

REFERENCES

Alkhamaiseh, K. N., Grantner, J. L., Shebrain, S., and Abdel-Qader, I. (2021). Towards automated performance assessment for laparoscopic box trainer using cross-stage partial network. In *2021 Digital Image Computing: Techniques and Applications (DICTA)*, pages 01–07. IEEE.

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.

Brand, M., Masuda, M., Wehner, N., and Yu, X.-H. (2010). Ant colony optimization algorithm for robot path planning. In *2010 international conference on computer design and applications*, volume 3, pages V3–436. IEEE.

Chen, R.-C. et al. (2019). Automatic license plate recognition via sliding-window darknet-yolo deep learning. *Image and Vision Computing*, 87:47–56.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991). The ant system: An autocatalytic optimizing process.

Guruji, A. K., Agarwal, H., and Parsediya, D. (2016). Time-efficient a* algorithm for robot path planning. *Procedia Technology*, 23:144–149.

Kala, R., Shukla, A., Tiwari, R., Rungta, S., and Janghel, R. R. (2009). Mobile robot navigation control in moving obstacle environment using genetic algorithm, artificial neural networks and a* algorithm. In *2009 WRI World Congress on computer science and information engineering*, volume 4, pages 705–713. IEEE.

Karami, A. H. and Hasanzadeh, M. (2015). An adaptive genetic algorithm for robot motion planning in 2d complex environments. *Computers & Electrical Engineering*, 43:317–329.

Khatib, O. (1985). Real-time obstacle avoidance system for manipulators and mobile robots. In *Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA*, pages 25–28.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, J., Gao, F., and Luo, X. (2019). Survey of deep reinforcement learning based on value function and policy gradient. *Chinese Journal of Computers*, 42(6):1406–1438.

Liu, J., Yang, J., Liu, H., Tian, X., and Gao, M. (2017). An improved ant colony algorithm for robot path planning. *Soft computing*, 21(19):5829–5839.

Liu, Q., Zhai, J. W., Zhang, Z.-Z., Zhong, S., Zhou, Q., Zhang, P., and Xu, J. (2018). A survey on deep reinforcement learning. *Chinese Journal of Computers*, 41(1):1–27.

Milinda, H. and Madhusanka, B. Multi-functional floor cleaning robot for domestic environment.

Milinda, H. and Madhusanka, B. (2017). Mud and dirt separation method for floor cleaning robot. In *2017 Moratuwa Engineering Research Conference (MERCon)*, pages 316–320. IEEE.

Mirjalili, S., Song Dong, J., and Lewis, A. (2020). Ant colony optimizer: theory, literature review, and application in auv path planning. *Nature-inspired optimizers*, pages 7–21.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Shukla, A., Tiwari, R., and Kala, R. (2008). Mobile robot navigation control in moving obstacle environment using a* algorithm. *Intelligent Systems Engineering Systems through Artificial Neural Networks*, 18:113–120.

Sichkar, V. N. (2019). Reinforcement learning algorithms in global path planning for mobile robot. In *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*, pages 1–5. IEEE.

Sun, J., Liu, G., Tian, G., and Zhang, J. (2019). Smart obstacle avoidance using a danger index for a dynamic environment. *Applied Sciences*, 9(8):1589.

Tai, L. and Liu, M. (2016). A robot exploration strategy based on q-learning network. In *2016 ieee international conference on real-time computing and robotics (rcar)*, pages 57–62. IEEE.

Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., and Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Computers and electronics in agriculture*, 157:417–426.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE.

Weiteng, Z., Baoming, H., Dewei, L., and Bin, Z. (2013). Improved reversely a star path search algorithm based on the comparison in valuation of shared neighbor nodes. In *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pages 161–164. IEEE.

Wen, S., Zhao, Y., Yuan, X., Wang, Z., Zhang, D., and Manfredi, L. (2020). Path planning for active slam based on deep reinforcement learning under unknown environments. *Intelligent Service Robotics*, 13(2):263–272.

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

Zhang, X., Zhao, Y., Deng, N., and Guo, K. (2016). Dynamic path planning algorithm for a mobile robot based on visible space and an improved genetic algorithm. *International Journal of Advanced Robotic Systems*, 13(3):91.

Zhao, Y., Shi, Y., and Wang, Z. (2022). The improved yolov5 algorithm and its application in small target detection. In *International Conference on Intelligent Robotics and Applications*, pages 679–688. Springer.