

Article

Deep Reinforcement Learning for Autonomous Driving with an Auxiliary Actor Discriminator

Qiming Gao ¹, Fangle Chang ^{1,2,*} , Jiahong Yang ^{1,3}, Yu Tao ¹, Longhua Ma ^{1,4} and Hongye Su ¹

¹ Ningbo Innovation Center, Zhejiang University, Ningbo 315100, China; 22032106@zju.edu.cn (Q.G.); 22360417@zju.edu.cn (J.Y.); taoyu@zju.edu.cn (Y.T.); lhma_zju@zju.edu.cn (L.M.); hysu69@zju.edu.cn (H.S.)
² State Key Laboratory of Fluid Power and Mechatronic Systems, Zhejiang University, Hangzhou 310027, China
³ Polytechnic Institute, Zhejiang University, Hangzhou 310013, China
⁴ Institute of Intelligent Automation, NingboTech University, Ningbo 315100, China
* Correspondence: changfl415@gmail.com

Abstract: In the research of robot systems, path planning and obstacle avoidance are important research directions, especially in unknown dynamic environments where flexibility and rapid decision makings are required. In this paper, a state attention network (SAN) was developed to extract features to represent the interaction between an intelligent robot and its obstacles. An auxiliary actor discriminator (AAD) was developed to calculate the probability of a collision. Goal-directed and gap-based navigation strategies were proposed to guide robotic exploration. The proposed policy was trained through simulated scenarios and updated by the Soft Actor-Critic (SAC) algorithm. The robot executed the action depending on the AAD output. Heuristic knowledge (HK) was developed to prevent blind exploration of the robot. Compared to other methods, adopting our approach in robot systems can help robots converge towards an optimal action strategy. Furthermore, it enables them to explore paths in unknown environments with fewer moving steps (showing a decrease of 33.9%) and achieve higher average rewards (showing an increase of 29.15%).

Keywords: autonomous driving; reinforcement learning (RL); auxiliary actor discriminator (AAD); heuristic knowledge (HK)



Citation: Gao, Q.; Chang, F.; Yang, J.; Tao, Y.; Ma, L.; Su, H. Deep Reinforcement Learning for Autonomous Driving with an Auxiliary Actor Discriminator. *Sensors* **2024**, *24*, 700. <https://doi.org/10.3390/s24020700>

Academic Editors: Wei Li and Honglu Jiang

Received: 12 December 2023
Revised: 13 January 2024
Accepted: 17 January 2024
Published: 22 January 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In recent years, driverless technology has caused much attention with the development of artificial intelligence and information perception [1]. Avoiding obstacles efficiently is important for intelligent robots to explore an accurate route in unknown environments, which include sweeping robots, mining robots, and rescue robots [2]. Commonly used obstacle avoidance methods include the gap-based algorithm [3], artificial potential field algorithm [4], velocity obstacle algorithm, and neural network algorithms [5]. The gap-based algorithms are based on classical reactive navigation and show good performance in areas with dense obstacles by sensing real-time environment information. One problem is that they may cause unreasonable deviation toward free areas, increasing the total distance and time to execute the mission. The artificial potential field algorithm is commonly developed for dynamic obstacle avoidance, and always has a local minimum. It calculates the resultant virtual force to facilitate the real-time control of the intelligent robot's control layer. The local minimum causes the robot to fall into a local oscillation and fail to find a global optimal solution, and many studies solve the problem by performing a random walk mechanism [6] and by using a navigation function [7]. Bounini et al. solved the problem by adding some extra repulsive potential, inspired by pouring a liquid with high pressure [4].

Reciprocal velocity obstacles (RVO) and its subsequent products aim to formulate a potential collision area for a moving obstacle using the relative velocity and position [8]. In these methods, a velocity outside this area is chosen for the robot to complete the collision avoidance task, and a distributed real-time multiple vehicle collision avoidance (MVCA)

algorithm is proposed by extending the reciprocal n-body collision avoidance method [9]. However, it requires a perfect sensing situation in the approaches [10]. Fuzzy algorithms do not require an exact mathematical model and perform well to overcome local extreme value problems. One needs to define all the regular behaviors for a fuzzy algorithm [11]. The application of artificial intelligence in obstacle avoidance has received a lot of attention in recent years. Jiang et al. [2] proposed the Deep Q-learning (DQL) algorithm to achieve obstacle avoidance in unknown environments for navigation. Zhang et al. [12] proposed a novel adaptive obstacle avoidance algorithm for USVs, based on the Sarsa on-policy Reinforcement Learning (RL) algorithm. Generally, neural network algorithms need to be trained with a big number of obstacle avoidance data.

Intelligent robots need detailed environment information to autonomously plan a path. In path planning, one common method is using global environmental data, and the other one is only using local environmental information [7]. Using global environmental data, predefined maps are usually constructed to describe the geometric information of the environment and are constantly updated during the navigation process, which requires a lot of memory and computational resources [13]. The whole process needs a model of the entire gridded/topology map and includes search-based algorithms (Dijkstra [14], A* [15], D* [16]) and sampling-based algorithms (RRT [17], RPM [18]). The artificial potential field method uses local environmental information obtained by airborne sensors to treat the robot environment as a potential field, in which the target point generates gravity to attract the robot and obstacles generate repulsion to repel the robot [19]. Velocity obstacle (VO)-based approaches [20,21] are widely used to predict collision regions and determine the robot's velocity in real time. They incorporate the sensor information (e.g., laser scanners and cameras) within the control loop [22]. Gap-based methods have been proposed to determine subsequent actions in navigation tasks. Most reactive methods face the problems of falling into local minima and tending to oscillate in a narrow passage [3]. One can build a global environment map online, but it takes a lot of time to implement.

With the rapid development of deep learning, the capabilities of feature extraction and function approximation have become more powerful. Combined with neural networks, high-dimensional state space information can be obtained [23]. Deep learning-based methods use deep neural networks to extract reasonable navigation behavior patterns from large amounts of labeled expert data. However, collecting labeled samples for navigation in unknown environments is time-consuming and energy-consuming, which hinders the widespread application of deep learning-based methods to solve the proposed problem [7]. DRL learns from labeled data, but its experience is generated from interactions between the agent and the environment, and thus it trains the neural networks with manually designed rewards [24,25]. The DRL approach transforms extensive training experience into the ability to think multiple steps ahead for more proactive movement decisions [21].

The deep reinforcement learning (DRL)-based collision avoidance approach can learn from a large amount of training experience, which is advantageous, and it can perform well in complex scenarios with high efficiency and robustness. Mnih et al. [26] applied a DRL algorithm to perform better than human players in a video game, in which the algorithm combined deep learning (DL) with reinforcement learning (RL) to solve complex decision-making problems. Based on the strong feature presentation ability of Q-learning (QL) [27] and convolutional neural network (CNN), deep Q-network (DQN) has shown its tremendous potential in robot control and decision making. Haarnoja et al. [28] developed the Soft Actor-Critic (SAC) algorithm to deal with inefficient exploration in continuous action settings. This model has been used with great success in control tasks such as dexterous hand manipulation [29] and DeepMind Control Suite [30]. In the field of robot control, the DRL methods in continuous action spaces can establish concise mapping from image inputs to the control policy [31]. Researchers have been applying DRL to navigate the intelligent agents in an unknown environment. Zeng et al. [7] proposed a novel DRL algorithm for continuously controllable navigation of non-holonomic robots in unknown dynamic environments with moving obstacles. A RL framework in decentralized

collision avoidance was proposed in [32], in which each agent can independently make its decision without communicating with others. Zhelo et al. [33] proposed a curiosity-driven exploration strategy and discussed the robot's ability to explore in complex and unknown environments.

The state representation is critical for the performance of DRL, especially for real-time decision-making tasks of navigation in complex environments. Many previous works [7,34] have targeted RL models with vector-based state representations. Choi et al. [32] used the LiDAR data, the forward velocity (v) and the rotational velocity (ω) of the robot, and the relative distances of (x) and (y) from the robot to the target position to describe the robot's state in the environment. Sparse LiDAR data is usually insufficient to describe the environment in detail, and in the case of any surrounding vehicles, it is necessary to find the interacting areas that have a more significant impact on the autonomous robot's decision to make decision behavior safe and effective. Attentional mechanisms [35] have achieved great success in different areas, and self-attention is one of the most used approaches. The self-attention mechanism [36,37] uses a self-supervised approach to calculate the response at a certain position in the sequence. The attention mechanism can discover the interdependencies between a variable number of inputs and is suitable for autonomous driving decision making problems.

In this article, we proposed a state attention network (SAN) to extract features to represent the interaction state of an intelligent robot with its environment using a self-attentive mechanism. Based on the Soft Actor-Critic (SAC) algorithm, an auxiliary actor discriminator (AAD) was designed to evaluate for collisions before executing the action and guide the agent to explore the environment safely and improve exploration efficiency. Goal-directed and gap-based navigation strategies were proposed to guide robotic exploration and help the network converge faster.

The structure of this paper is organized as follows: Section 2 introduces the framework of our approach and presents a method to train the neural network with AAD, goal-directed, and gap-based navigation strategies; Section 3 shows the simulation experiment and experimental result; and finally, Section 4 gives some conclusions about this paper.

2. Materials and Methods

2.1. Experiment Design and Data Collection

The virtual training environments were simulated by Gazebo, in which two indoor environments were constructed to demonstrate the effect of the environment on the model (Figure 1). A Turtlebot3 was applied as the robot platform to test the adaptability of the model in different environments. In our model, the control frequency is 5 Hz, and the moving steps of the robot are up to a maximum of 5000 in one episode. In every episode, the target position is randomly initialized throughout the area and is guaranteed not to collide with other obstacles.

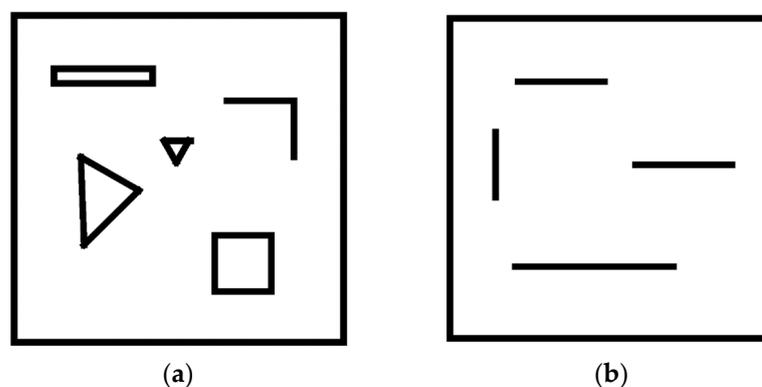


Figure 1. Two indoor environments ($8 \times 8 \text{ m}^2$) were created, in which solid black lines were walls and hollow geometric figures were obstacles. (a) Environment I. (b) Environment II.

The hyper-parameters of our method are described as follows: The learning rates for the critic and actor network are both 0.0001, the discount factor is 0.98, the goal tolerance distance is 0.15, and the replay buffer size is 1×10^6 . We trained the model with an Adam optimizer on a single Nvidia GeForce GTX 3070 GPU (i7-11700, RAM 16 G) for 1×10^8 training steps which took almost 20 h.

The robot was tested to move from the given start point to the given end point without collision in an unknown environment. It sensed its own state and local environment state (S) through on-board sensors (Equation (1)). The inertial measurement unit calculates its own posture with the Euler angle description $[\theta_r, \theta_p, \theta_y]$, velocity information $[v_x, v_y, \omega_z]$, GPS estimates of the robot's position $[p_x, p_y]$, and LiDAR scans of the surrounding obstacles.

$$S = [s_t^{lidar}, s_t^{goal}, s_t^{robot}] \quad (1)$$

$$s_t^{goal} = [s_{goal}, \theta_{goal}] \quad (2)$$

$$s_t^{robot} = [p_x, p_y, v_x, v_y, \theta_y] \quad (3)$$

where s_t^{lidar} are the LiDAR data which indicate the relationship of the obstacles and the robot by measuring the distance between the recognizing objects, s_t^{goal} are the relative distance and heading angle from the robot to the target's position, and s_t^{robot} describe the state of the robot itself.

The action space is the robot's linear velocity increment in the x-plane (Δv_x) and the angular velocity increment in the z-coordinate axis ($\Delta \omega_z$) (Equation (4)). The control velocity at the next time step (v_{t+1}) should be the current velocity (v_t) plus the robot's action space with a constant μ (Equation (5)). The reward function (r) of RL model was set as Equation (6).

$$a = [\Delta v_x, \Delta \omega_z] \quad (4)$$

$$v_{t+1} = v_t + \mu \cdot a, \quad v_t \in [v_{min}, v_{max}] \quad (5)$$

$$r = \begin{cases} 500.0 & \text{if } d_{current} < 0.2, \quad d_{current} = \text{dist}(\text{robot}, \text{goal}) \\ -500.0 & \text{if } d_{obs} < 0.2 \\ -\lambda \cdot \frac{d_{current}}{d_{init}} & \text{other case} \end{cases} \quad (6)$$

where $d_{current}$ indicates the distance of the current robot from the target point, d_{init} is the initial distance, and λ is a factor that regulates the scale of the reward value.

2.2. Model Development

A modified SAC model was developed in this work. The SAC algorithm was developed to reduce inefficient sample sizes in continuous action settings [29], and it attempts to find a policy that maximizes the entropy objective (Equation (7)).

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \sum_{t=0}^T E_{(s_t, a_t) \sim \tau_\pi} [\gamma^t (r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot | s_t)))] \quad (7)$$

$$r : S \times A \rightarrow R, \quad \gamma \in [0, 1], \quad s_t \in S, \quad a_t \in A$$

$$\mathcal{H}(\pi(\cdot | s_t)) = -\log \pi(\cdot | s_t) \quad (8)$$

where π is a policy; π^* is the optimal policy; T is the number of timesteps (t); r is the reward function; γ is the discount rate falling; s_t is the state at timestep t ; a_t is the action at timestep t ; τ_π is the distribution of trajectories induced by policy π ; α is the temperature parameter which is used to determine the relative importance of the entropy term versus the reward; and $\mathcal{H}(\pi(\cdot | s_t))$ is the entropy of the policy π at state s_t and was calculated in Equation (8).

The soft state value function (V) was applied to maximize the objective within the maximum entropy framework (Equation (9)). The soft q-function can be obtained by starting from a randomly initialized function $Q(s_t, a_t)$ and repeatedly applying the modified

Bellman backup operator (T^π) (Equation (10)). In the continuous state space, the soft q-function $Q_\theta(s_t, a_t)$ was parameterized using a neural network with parameter θ . The soft q-function was trained to minimize the soft Bellman residual (Equation (11)).

$$V(s_t) := E_{a_t \sim \pi} [Q(s_t, a_t) - \alpha \log(\pi(a_t | s_t))] \quad (9)$$

$$\begin{aligned} T^\pi Q(s_t, a_t) &:= r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_t, a_t)} [V(s_{t+1})] \\ Q : S \times A &\rightarrow R, \quad p : S \times A \rightarrow S \end{aligned} \quad (10)$$

$$J_Q(\theta) = E_{(s_t, a_t) \sim D} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \left(r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_t, a_t)} [V_\theta(s_{t+1})] \right) \right)^2 \right] \quad (11)$$

where p gives the distribution over the next state when the current state and action have been given; T^π is the modified Bellman backup operator; θ is the parameter of a neural network; D is the replay buffer of past experiences; and $V_\theta(s_{t+1})$ is an estimate of a target network of Q .

The soft q-function can guide the policy improvement step by updating the policy in a direction to maximize the obtained rewards. In the continuous state setting, the policy $\pi_\phi(a_t | s_t)$ was parameterized using a neural network with parameter ϕ , and output a mean and a covariance to define a Gaussian policy. The policy parameters (π_{new}) were updated by minimizing the expected KL divergence (D_{KL}) using π_{old} (Equation (12)). Usually, the partition function $Z^{\pi_{\text{old}}}(s_t)$ can be ignored since it does not impact the gradient.

$$\pi_{\text{new}} = \underset{\pi \in \Pi}{\operatorname{argmin}} D_{KL} \left(\pi(\cdot | s_t) \parallel \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{\text{old}}}(s_t, \cdot)\right)}{Z^{\pi_{\text{old}}}(s_t)} \right) \quad (12)$$

The output distribution of this strategy means that errors cannot be backpropagated in the normal way. To solve this, we used the reparameterization trick (Equation (13)) and obtained the new policy objective (Equation (15)).

$$J_\pi(\phi) = E_{s_t \sim D} \left[E_{a_t \sim \pi_\phi} [\alpha \log(\pi_\phi(a_t | s_t)) - Q_\theta(s_t, a_t)] \right] \quad (13)$$

$$a_t = f_\phi(\epsilon_t; s_t), \quad \epsilon_t \in N(0, 1) \quad (14)$$

$$J_\pi(\phi) = E_{s_t \sim D, \epsilon_t \sim N} [\alpha \log(\pi_\phi(f_\phi(\epsilon_t; s_t) | s_t)) - Q_\theta(s_t, f_\phi(\epsilon_t; s_t))] \quad (15)$$

where π_ϕ is now defined implicitly in terms of f_ϕ . Policy evaluation and policy improvement will converge to optimal policies. In continuous state space, the SAC algorithm has advantages in continuous state space. Therefore, the SAC algorithm was used as the base model in this paper.

The modified SAC model included a SAN for state learning, and an auxiliary action discriminator for collision probability calculation (Figure 2). It introduced prior knowledge to propose goal-based and gap-based navigation strategies to guide the learning strategies. Raw data from the LiDAR sensor and inertial measurement unit were aligned using a SAN based on an attention mechanism to a uniform state (s_t). It is hard to prepare enough data to train neural networks when robots are exploring unknown environments. A replay memory D was applied to store experience (s, a, s', r) to train the neural network by randomly sampling data. During the training phase, the robot moved randomly in unknown environments and generated a lot of collision data, which can make the model fall into local best. An AAD based on the RVO algorithm was developed to calculate the probability of a collision for the current action and reduce the number of collisions. With the help of prior knowledge, neural networks can quickly converge to an optimal action strategy.

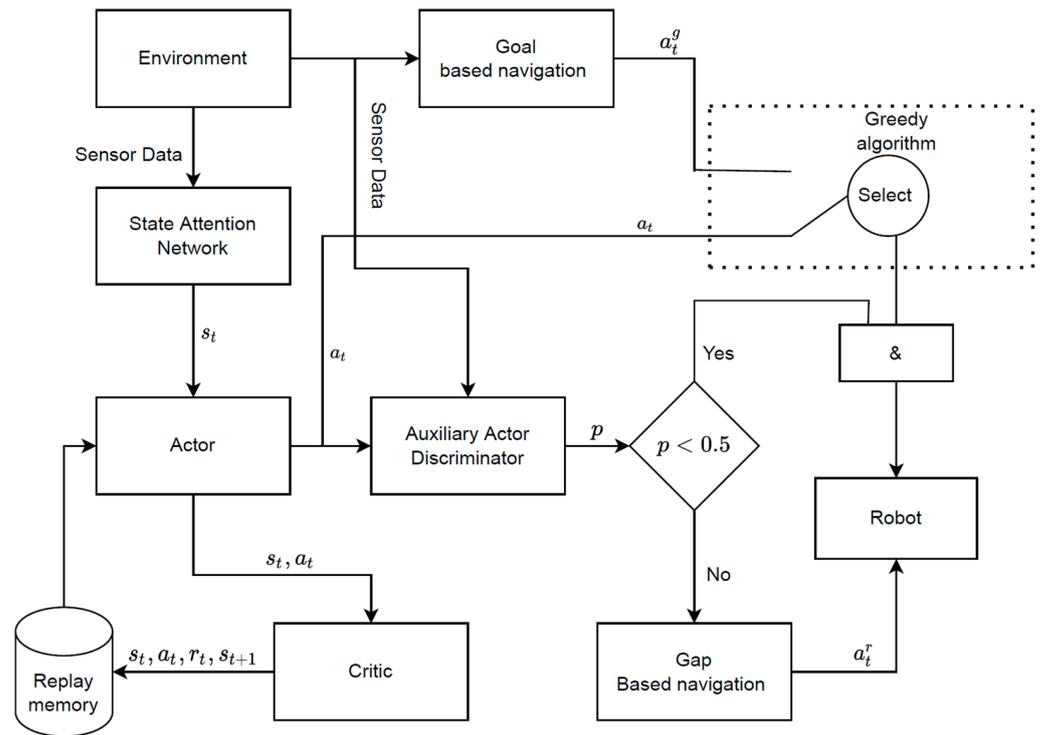


Figure 2. The modified SCA model. The actor was a neural network that can learn a navigation strategy from current state and make an action in real time, and the critic was a q-value function fitted using a neural network to evaluate state–action pairs. & means and.

2.3. State Attention Network (SAN)

The state representation is critical in an autonomous driving task using DRL with multi-sensor data. For path planning tasks, the state space contains information related to the robot’s own state (e.g., speed, position, orientation angle) and the environment (e.g., obstacles). The input of the SAN includes the vector-based states and the image-based states. In this work, the vector-based states included the robot position, velocity, and distance to the target point. The orientation angle and the image-based states were represented by the LiDAR measurements using a signed distance field algorithm (Figure 3a).

The spatial attention module was proposed by CBAM to extract the weights of the spatial information, in which a CNN encoder was developed to extract image deep-level features. Two feature maps were obtained from maximum pooling and average pooling, and a deep-level fusion map was created through a convolution operation (Figure 3b). The segmented attention (seg-attention) module was developed to study correlations between different regions of the fusion map, which consisted of input segmentation, feature extraction, and fusion feature (Figure 3c). The input segmentation module segmented the fusion map into multiple partial states (small image patches in the grid). The feature extraction module extracted key (Equation (16)) and value (Equation (17)) features from each partial state to determine where the model should attend to encode information for path planning tasks.

$$K_i = L_K * s_i \quad (16)$$

$$V_i = f_c(L_V * s_i) \quad (17)$$

where s_i is i -th partial state, L_K and L_V are the weight matrixes, K_i is the key feature for i -th partial state, V_i is the value feature for i -th partial state, and f_c is the ReLU function.

The attention layers (neural network θ) determined important partial states based on key features (K) to obtain the attention weight vector (A) (Equation (18)). They were randomly initialized in the beginning of training. Value features were weighted by A

(Equation (19)). After processing the image-based state, the robot's own related state was encoded by the MLP and connected f to form a unified state space (Figure 3c).

$$A = \text{softmax}(\theta * K) \quad (18)$$

$$f = \sum_i V_i * A_i \quad (19)$$

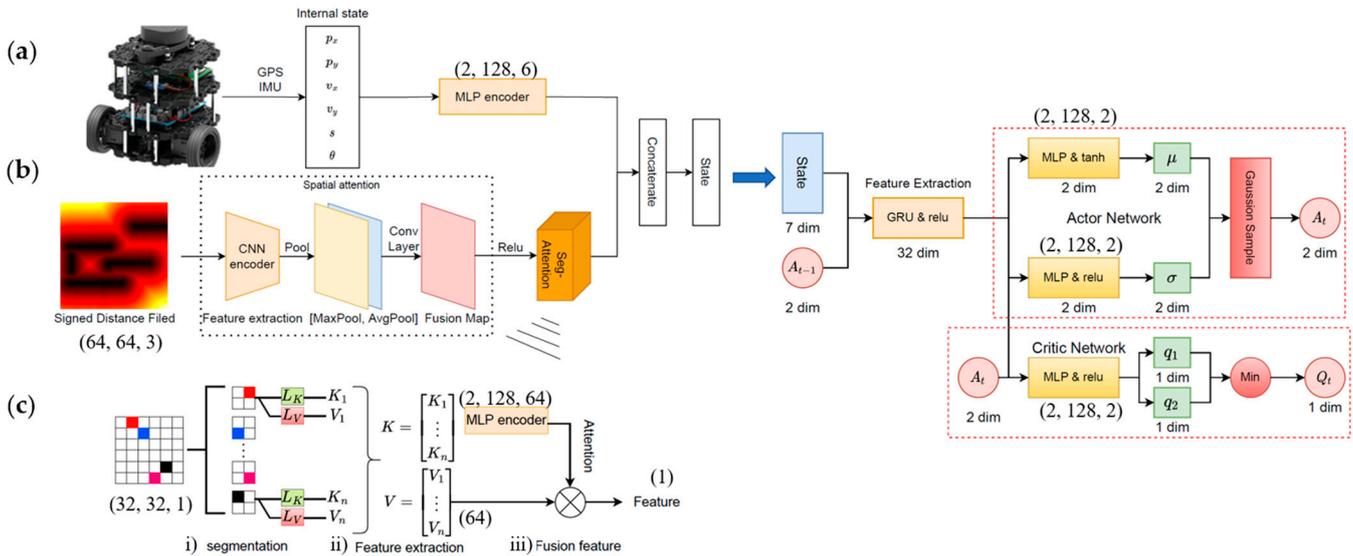


Figure 3. The proposed SAN's structure: (a) robot's information and (b) local environment information were analyzed to advance the model's perceptual ability, in which (c) the seg-attention module structure extracted key information using a self-attention mechanism. The hidden layers, neuron number per layer, and output dimension of the MLP encoder are shown next to the network. The image size of the signed distance field is (64, 64, 3) and its output is (32, 32, 1). The final feature scale is (1) after the seg-attention module.

2.4. Auxiliary Actor Discriminator (AAD)

The velocity obstacle (VO) was developed for mobile robots to deal with dynamic obstacles in their local path planning. Robots A and B are obstacles to each other and both have independent target points. They both need to perform obstacle avoidance and plan a path to reach the target point (Figure 4a). The VO of A to B contains all possible collide velocities, and the VO area of the robot A generated by the robot B ($VO_{A|B}$) is a cone area with an apex at v_B (Figure 4b) and can be calculated using Equations (20) and (21).

$$A \oplus B = \{a + b | a \in A, b \in B\}, -A = \{-a | a \in A\} \quad (20)$$

$$VO_{A|B} = \{v | \lambda(p_A, v_A - v_B) \cap B \oplus -A \neq \emptyset\} \quad (21)$$

where $A \oplus B$ is the Minkowski sum of two points sets in A and B , $-A$ denotes the robot A reflects in its references point, and $\lambda(p, v)$ denotes the ray with a starting point of p and in the direction of v . In order to avoid collision with obstacles, robot A should choose a speed outside the VO area ($v_A \notin VO_{A|B}$). The reciprocal velocity obstacle (RVO) was calculated to reduce oscillations of the VO method (Equation (16)). The RVO was translated geometrically from $VO_{A|B}$ with a vector $(v_A - v_B)/2$, and an apex at $(v_A + v_B)/2$ (Figure 4c).

$$RVO_{A|B} = \left\{ v \mid 2v - v_A \in VO_{A|B} \right\} \quad (22)$$

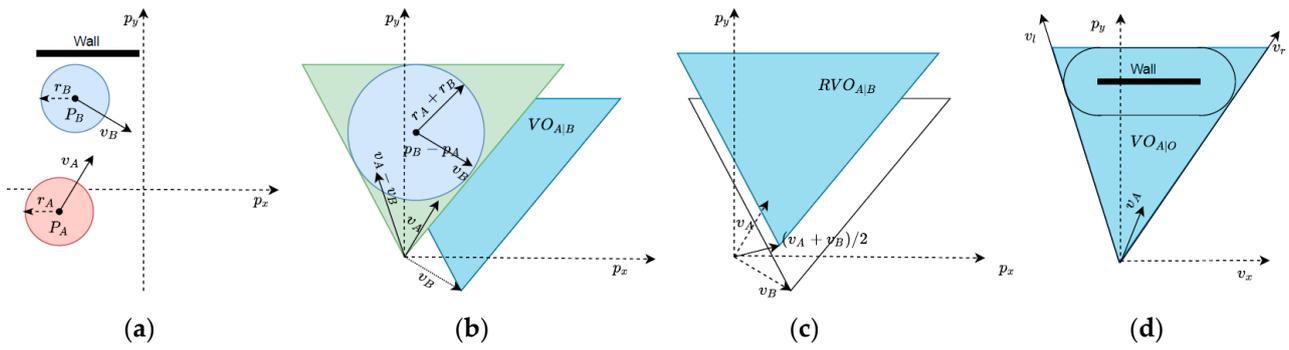


Figure 4. VO and RVO in a workspace configuration, where A and B are two moving robots in the 2D workspace (p_x, p_y) , r_A and r_B mean the radius that described the current state of robot A and B, p_A and p_B mean the quality hearts' positions, v_A and v_B mean the current velocity of A and B, $VO_{A|B}$ means the VO area of the robot A generated by the robot B, and $RVO_{A|B}$ means the set of speeds that robot A can choose to be safe. (a) Workspace, (b) VO, (c) RVO, and (d) static line obstacle.

An AAD was developed based on the RVO to calculate the probability of collision for the next action. The collision probability function (P_t) at time t represented the quality of the selected velocity v_t judged by the joint RVO area (Equation (23)).

$$P_t = \begin{cases} 0 & \text{if } v_t \notin RVO_{A|O} \\ (\zeta + 1)^{-1} & \text{if } v_t \in RVO_{A|O} \end{cases} \quad (23)$$

$$\begin{cases} v_t \notin RVO_{A|O} & v_t \times v_l < 0 \vee v_t \times v_r > 0 \\ v_t \in RVO_{A|O} & v_t \times v_l \geq 0 \wedge v_t \times v_r \leq 0 \end{cases}$$

where ζ is the estimated shortest time for the robot to collide with an obstacle at the current speed, and \times is cross operate. After the action collision probability was calculated by the auxiliary action discriminator, the ε -greedy algorithm was applied to perform action selection (goal orientation or actor output) if no collision will occur. Otherwise, reactive navigation based on gaps was used to avoid obstacles and move towards the target point.

2.5. Prior Knowledge

It is important to have many valid data in the replay buffer to train the neural network effectively in learning strategies for navigation. In the traditional actor-critic algorithm, the robot randomly selected actions to explore, and the replay buffer stored many collision data, which were randomly selected to train the neural network. This can help the robot to learn collision avoidance strategies but may cause the robot to get stuck in a local solution, where the robot oscillates in a safe region and fails to achieve the effect of navigation. We used a priori knowledge and goal-directed/gap-based strategies (Algorithm 1) to guide the robot and reduce the network training time.

The goal-directed knowledge is based on the angle (yaw_{err}) between the robot's direction and the end point. The angle (yaw_{err}) is defined as the rotating angle from 0 to 360 degrees at which the robot rotates counterclockwise until it points to the end point. When the robot acquired the environment state data, the actor network gave the current action a_t . The probability of collision after executing this action was calculated by an auxiliary action discriminator, and an ε -greedy algorithm was used to select the final action to be executed between the goal-based a_t^g (Equation (24)) and a_t when the probability was less than a threshold η . With probability ε , the robot selected the action a_t^g that would bring it closer to the end point according to the goal-directed knowledge.

$$a_t^g = [0.05, \omega] \quad (24)$$

$$\omega = \begin{cases} yaw_{err}/180 & \text{if } yaw_{err} < 180 \\ 1 - yaw_{err}/180 & \text{othercase} \end{cases} \quad (25)$$

To determine subsequent actions, the gap-based strategy analyzed the form of high-level descriptions of the environment on the basis of sensor information. The boundary gap model first extracted the gaps by analyzing the radar data to filter out the gaps that satisfied the passage conditions. The movement direction was selected according to the target point and the robot could be smoothly controlled through the gap.

Algorithm 1: Our proposed algorithm

- 1: Initialize replay buff D , actor network π_θ , old actor network parameters $\bar{\theta} \leftarrow \theta$, critic network V_{ψ_1} and target critic network V_{ψ_2} , actor learning rate lr_a , critic learning rate lr_c , update frequency K , Collision threshold th , Temperature parameter α , target critic network update weight τ
 - 2: **for** episode = 1, N **do**
 - 3: Initialize environment and set robot to the start point
 - 4: **for** steps = 1, T **do**
 - 5: Get Lidar data l_t , robot data r_t
 - 6: Get uniform state s_t using SAN module
 - 7: Get current action a_t by Actor Network
 - 8: define final action a_t^*
 - 9: Compute collision probability p_t by AAD module.
 - 10: **if** $p_t < \eta$ **then**
 - 11: With probability ε select an action $a_t^* = a_t^g$ through goal-directed knowledge
 - 12: With probability $1 - \varepsilon$ select an action $a_t^* = a_t$
 - 13: **else**
 - 14: Get an action $a_t^* = a_t^g$ through Gap-based knowledge
 - 15: **end if**
 - 16: Execute action a_t^* , get reward r_t , and new state s_t
 - 17: Store (s_t, a_t^*, r_t, s_t) in D
 - 18: **if** steps% $K == 0$ **then**
 - 19: Randomly sample a minibatch $(s_j, a_j, r_j, s_j) \in D$
 - 20: Compute the target Q value:

$$y_{target} = r_j + \gamma V_{\psi_2}(s_j, a_j) - \alpha \log(\pi_\theta(a_j | s_j))$$

$$a_j \sim \pi_\theta(s_j)$$
 - 21: Update critic network parameters ψ_1 by minimizing the following loss function

$$L_V^{loss}(\psi_1) = \frac{1}{2} \sum_{(s_j, a_j, r_j, s_j) \in D} (y_{target} - V_{\psi_1}(s_j, s_j))^2$$
 - 22: Update actor network parameters θ by minimizing the following loss function

$$L_\pi^{loss}(\theta) = \sum_{(s_j, a_j, r_j, s_j) \in D} (\alpha \log(\pi_\theta(f_\phi(\epsilon_j; s_j) | s_j)) - V_{\psi_1}(s_j, f_\phi(\epsilon_j; s_j))), \epsilon_j \sim N(0, 1)$$
 - 23: Update temperature parameter α by minimizing the following loss function

$$L_\alpha^{loss}(\alpha) = -\alpha \log \pi_\theta(a_j | s_j) + H_0, H_0 = -\dim(A)$$
 - 24: Update target critic network parameters

$$\psi_2 = \tau \psi_1 + (1 - \tau)$$
 - 25: Update old actor network parameters

$$\bar{\theta} \leftarrow \theta$$
 - 26: **end if**
 - 27: **end for**
 - 28: **end for**
-

3. Results

3.1. Radar Data Representation

Radar data was graphically represented to apply high-dimensional LiDAR data to RL model. The distance obstacle map s_t^{map} was calculated using the signed distance field (SDF) algorithm (Figure 5). We use the s_t^{map} as the description of the robot's local environment.

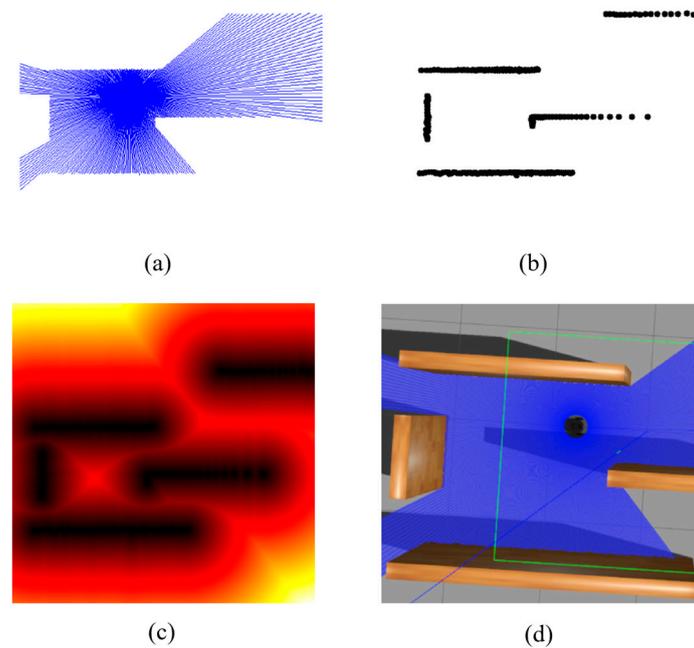


Figure 5. Graphical representation of radar data in process: (a) Represent the radar data in images, (b) extract the obstacle boundary, (c) calculate the distance obstacle map s_t^{map} using the signed distance field algorithm, and (d) show in Gazebo environment. (a) Current LiDAR data, (b) Extract obstacles, (c) Signed Distance Field, and (d) Gazebo environment.

The robot will keep moving closer to the target point while avoiding obstacles to get a higher reward. The reward function of Environment II is shown in Figure 6b.

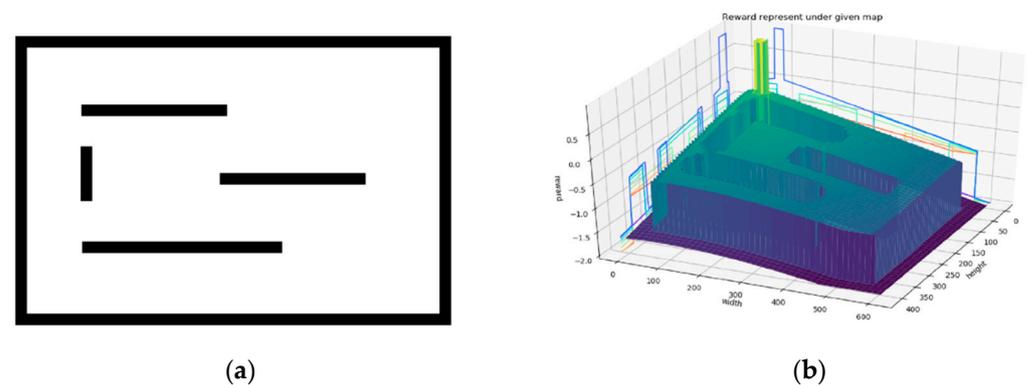


Figure 6. Reward function in Environment II. The start point is $[3.0, -3.0]^T$, the end point is $[-3.0, 3.0]^T$, $\lambda = 100$, and the reward value falls in $[-2, 1]$. (a) map, (b) reward.

3.2. Model Performance

The training phase and generalization of SAC, SAN+SAC, and SAN+SAC+AAD models were compared, and the parameters used in training were summarized in Table 1. In Environment I, the models were trained until the average reward of the smart car was stable, and the robot trajectories are shown in Figure 7. The figure shows that those three methods can guide the robot to the end point without collision. In general, the robot can

move in the best trajectory in the SAN+SAC+AAD model. The robot can travel in a shorter path when it has learned heuristic knowledge.

Table 1. The values of training parameters.

Parameter	Value
Learning rate l_{ra}	0.001
Update frequency K	1×10^4
Replay memory	1×10^6
Collision threshold th	0.6
Temperature parameter α, τ	0.5
Mini-batch size	1024
Discount factor γ	0.99
Action selection factor ϵ	0.5

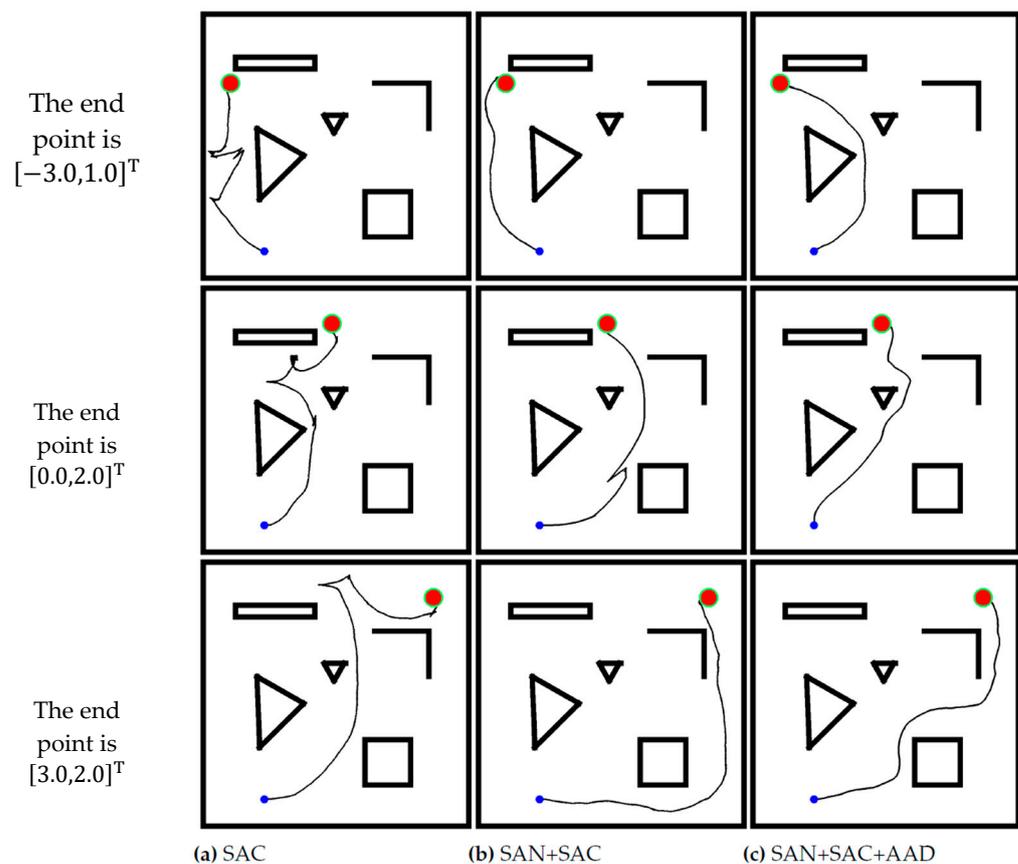


Figure 7. Nine trajectories that the robot obtained by using (a) SAC, (b) SAN+SAC, and (c) SAN+SAC+AAD models for three different end points in Environment I, in which the blue dot denotes the start point, the red circle in the upper right corner is the end point, and the black areas are four walls and obstacles.

The average reward of each model was calculated in an epoch (Figure 8). The average reward value fluctuates greatly in the early stage of training when we encourage robots to explore the map and expand the footprint range of the robot. SAC and SAN+SAC obtain similar average reward values in the convergence stage. Adding AAD, the number of collision obstacles reduced, and the highest average reward value was obtained in the early stage of training.

The map was changed to Environment II to verify the model universality in different scenarios. The trajectories of the robot are shown in Figure 9. The obstacles in Environment II are relatively long. The robot chooses to bypass obstacles instead of looking for different

topologies in SAC model. Adding SAN, the robot tries to find a better topological path. In the middle row, the target point $[0, -1]^T$ is close to the starting point $[-2, -3]^T$. The robot may keep away from the target point because of the obstacle direction. The robot needs to have a long-term vision and not be trapped in local minima. The robot oscillates constantly in the SAC model, but it has a better understanding of the environment in the SAN+SAC and SAN+SAC+AAD models.

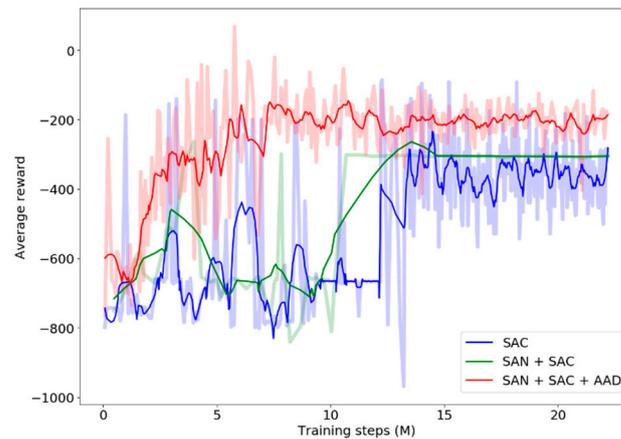


Figure 8. The average rewards of SAC, SAN+SAC, and SAN+SAC+AAD models in Environment I. Light lines mean the average rewards, and dark lines mean the values after smoothing.

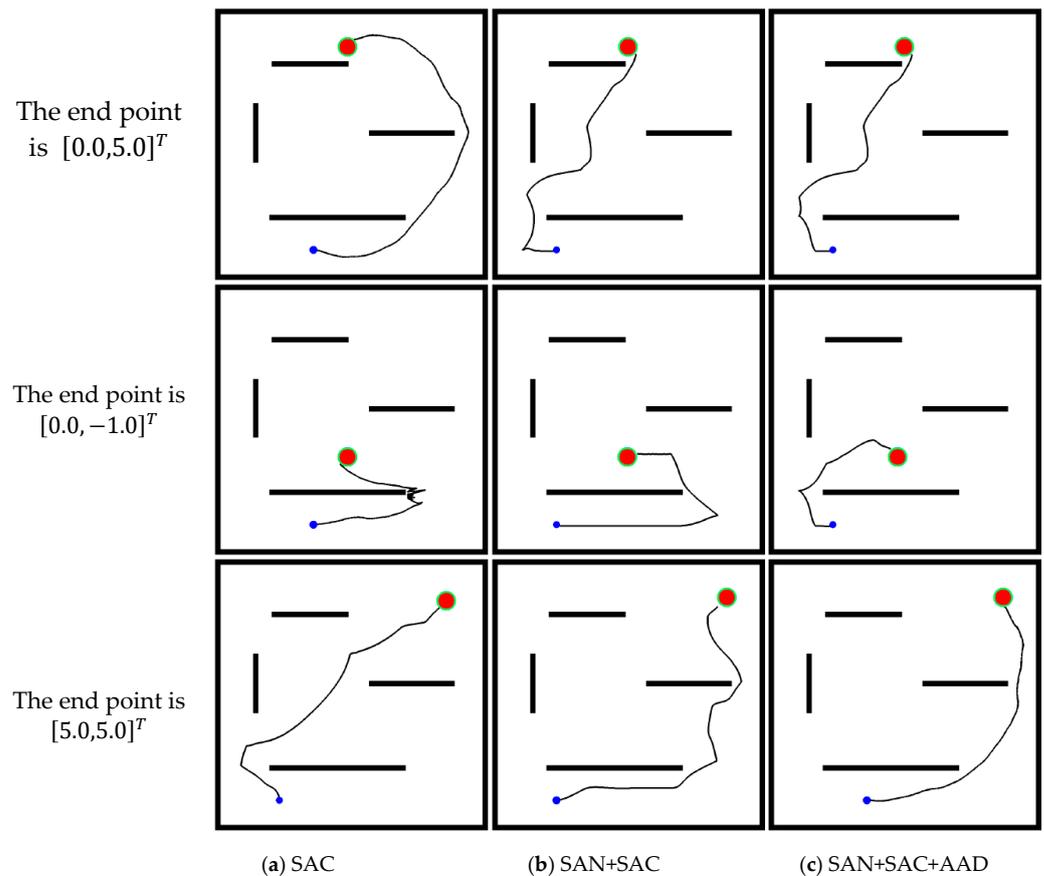


Figure 9. Nine trajectories that the robot obtained by using (a) SAC, (b) SAN+SAC, and (c) SAN+SAC+AAD models for three different end points in Environment II, in which the blue dot denotes the start point, the red circle in the upper right corner is the end point, and the black areas are four walls and obstacles.

The moving step count and average reward were calculated in Environment II (Table 2). The SAN module can enhance the model understanding of the environment and help the robots to find different topological paths. Adding SAN, the robot can reach the end point without hitting obstacles, and changing the obstacles and the environment cannot affect the robot's performance (Figure 10). It shows the multi-heads' attention can filter irrelevant areas, focus on obstacle walls, and find safe paths. The AAD module aims to find the shorter path to arrive at the target point. Adding AAD, the moving steps dropped and the average reward increased quickly.

Table 2. Moving step count and average training steps in Environment II.

	Model	Target Point 1	Target Point 2	Target Point 3
Moving step count	SAC	1805	2049	2224
	SAN+SAC	1708	1921	2085
	SAN+SAC+ADD	1573	1351	1558
Average reward	SAC	−415.26	−524.36	−583.46
	SAN+SAC	−365.28	−460.87	−468.23
	SAN+SAC+ADD	−294.57	−136.86	−281.39

The numbers in bold mean the best model performance.

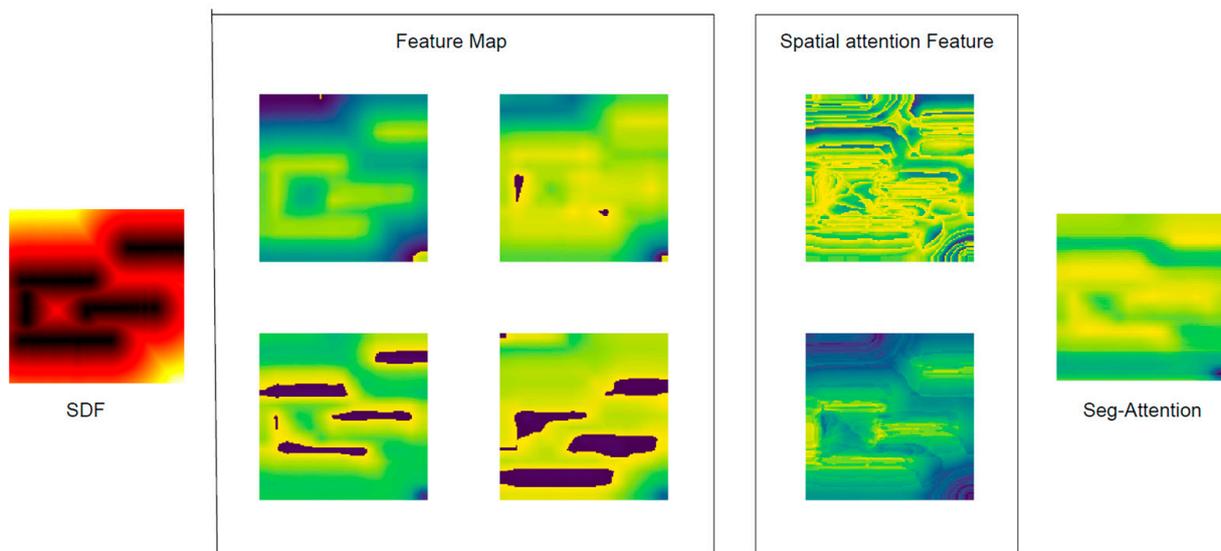


Figure 10. The visual feature map bypasses the SAN module, the SDF picture was used as the input of the SAN module, the feature maps were extracted using CNN, the feature maps were fused using spatial attention, and the seg-attention network was used to correlate the characteristics of different regions in the fused feature map, in which the blue areas mean the safe path and the yellow areas mean the obstacle wall.

The moving step count and the average training steps of the robot when it reaches the end point in Environment I and Environment II are listed in Table 3. The SAN module greatly reduces the average moving step counts and training steps, because SDF can provide more abundant environmental information for the model, and with CNN's powerful reasoning ability, it can help the model to reach the target point faster. Compared to SAC, our method with SAN+SAC+AAD reduced training steps by 42.78% and 40.88% on each map.

Table 3. Moving step count and average training steps.

	Model	Environment I	Environment II
Moving steps	SAC	3190	2594
	SAN+SAC	2714	1793
	SAN+SAC+AAD	2106	1394
Average train steps (M)	SAC	15.24	9.27
	SAN+SAC	12.68	8.26
	SAN+SAC+AAD	8.72	5.48

The numbers in bold mean the best model performance.

3.3. Real Scenario Validation

In a real scenario, the RL algorithm collects robot data as the input to the model during the navigation process to decide the actions, including the robot velocity increments and the angular velocity increments. Commands are executed when the navigation task is completed or a collision occurs (Figure 11). On the outside, the mapping effect is poor because of the big LiDAR measurement error and convex ground. The robot cannot actively explore because of the cumulative error.

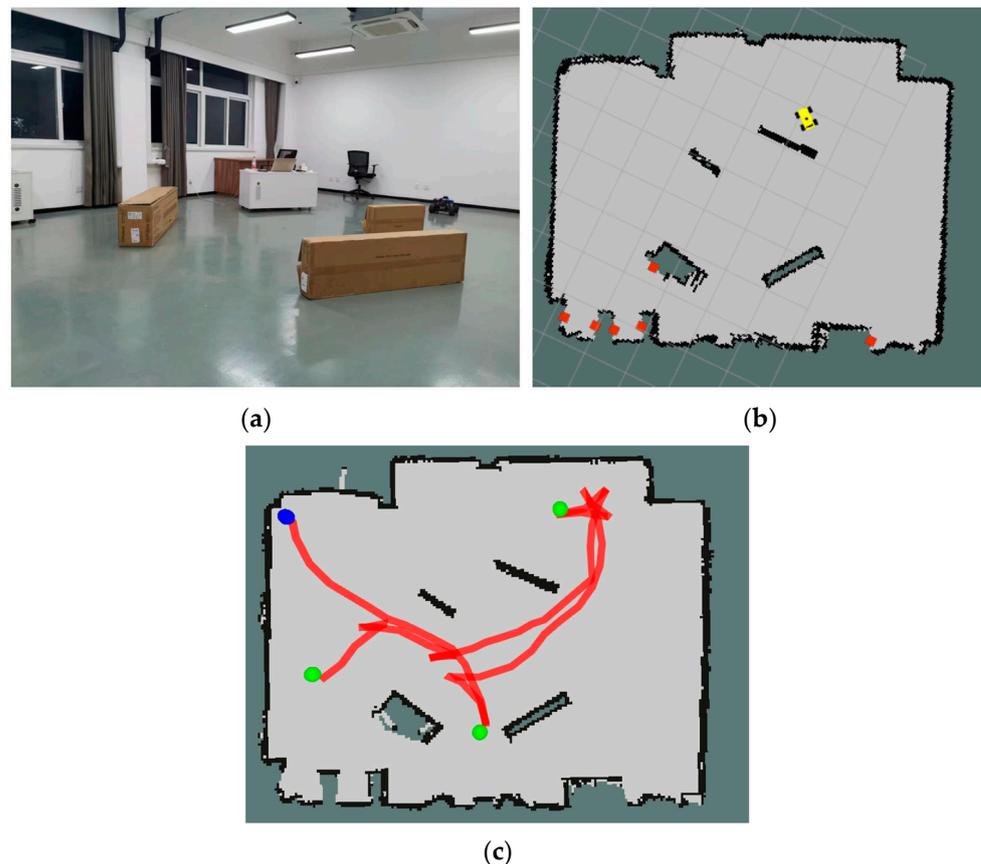


Figure 11. (a) Real scenario, (b) a map created using the Gmapping algorithm, (c) robot trajectory, in which the blue point is the starting point, and the green point is the target point.

Using the SAN module, the robot collected the LiDAR data and extracted the obstacle information to obtain SDF picture (Figure 12). The feature map focusses on safe paths and obstacles, and it makes good use of the distance information provided by the SDF map (Figure 13).

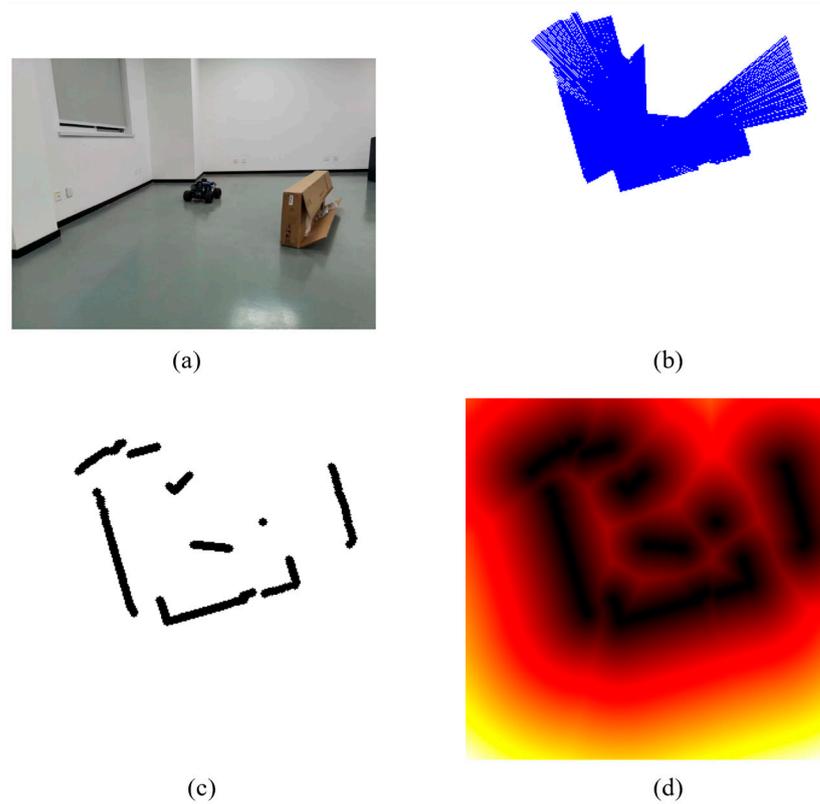


Figure 12. Visualization of LiDAR data using the SAN module in a real scenario. (a) Real Scene, (b) LiDAR data, (c) Obstacle Information, and (d) Signed Distance Field.

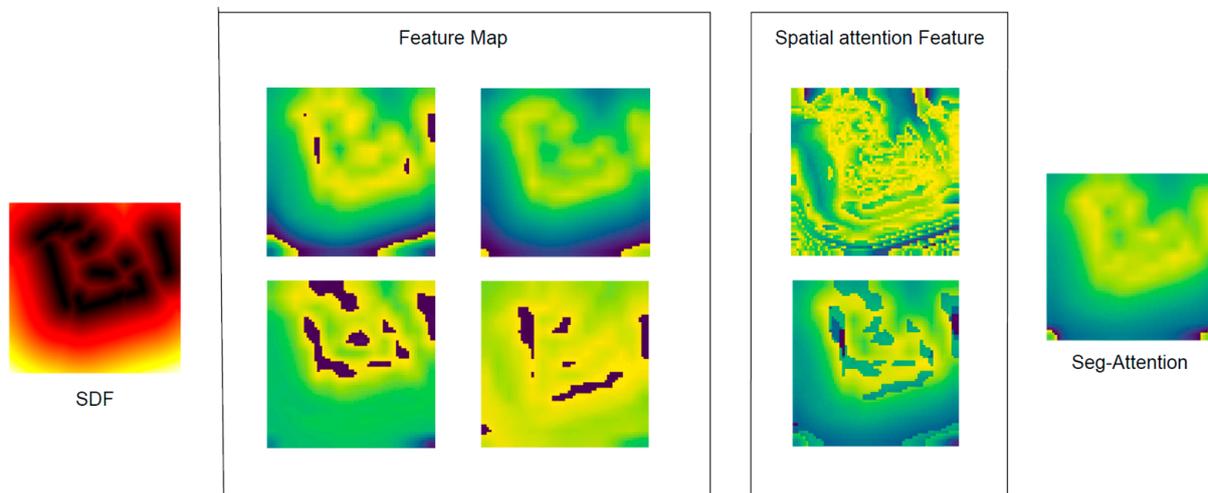


Figure 13. The visual feature map bypasses the SAN module in a real scenario, in which the blue areas mean the safe path and the yellow areas mean the obstacle wall.

4. Conclusions

In this article, we combine SAC with a SAN module and an AAD module (SAN+SAC+ADD) for intelligent robots' path planning and obstacle avoidance. The method has been tested in different environments and the results show that the robot is able to converge to the optimal policy faster and reach the end point in fewer steps than other methods. Experiments also show that our model has better adaptability in unknown environments. In this paper the obstacles are static. In the future, we will work on dynamic obstacle avoidance.

Author Contributions: Conceptualization, F.C.; Methodology, Q.G. and F.C.; Resources, F.C.; Data curation, Q.G.; Writing—original draft, Q.G.; Writing—review & editing, F.C., J.Y. and Y.T.; Supervision, F.C.; Project administration, F.C.; Funding acquisition, F.C., L.M. and H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially funded by Ningbo K&D Project under Grant Number 2023Z116, and Open Foundation of the State Key Laboratory of Fluid Power and Mechatronic Systems.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article. The code that supports the findings of this study is available from the corresponding author upon reasonable request.

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

References

1. Peng, Z.; Wang, D.; Zhang, H.; Sun, G. Distributed neural network control for adaptive synchronization of uncertain dynamical multiagent systems. *IEEE Trans. Neural Netw. Learn. Syst.* **2013**, *25*, 1508–1519. [[CrossRef](#)] [[PubMed](#)]
2. Jiang, L.; Huang, H.; Ding, Z. Path planning for intelligent robots based on deep q-learning with experience replay and heuristic knowledge. *IEEE/CAA J. Autom. Sin.* **2019**, *7*, 1179–1189. [[CrossRef](#)]
3. Gao, Z.; Qin, J.; Wang, S.; Wang, Y. Boundary Gap Based Reactive Navigation in Unknown Environments. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 468–477. [[CrossRef](#)]
4. Bounini, F.; Gingras, D.; Pollart, H.; Gruyer, D. Modified artificial potential field method for online path planning applications. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–17 June 2017; pp. 180–185.
5. Cao, X.; Ren, L.; Sun, C. Research on Obstacle Detection and Avoidance of Autonomous Underwater Vehicle Based on Forward-Looking Sonar. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *34*, 9198–9208. [[CrossRef](#)] [[PubMed](#)]
6. Barraquand, J.; Latombe, J.C. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.* **1991**, *10*, 628–649. [[CrossRef](#)]
7. Zeng, J.; Ju, R.; Qin, L.; Hu, Y.; Hu, C. Navigation in unknown dynamic environments based on deep reinforcement learning. *Sensors* **2019**, *19*, 3837. [[CrossRef](#)]
8. Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal n-body collision avoidance. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–19.
9. Zu, C.; Yang, C.; Wang, J.; Gao, W.; Wang, F.Y. Simulation and field testing of multiple vehicles collision avoidance algorithms. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1045–1063. [[CrossRef](#)]
10. Jin, J.; Kim, Y.G.; Wee, S.G.; Gans, N. Decentralized cooperative mean approach to collision avoidance for nonholonomic mobile robots. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Washington, DC, USA, 26–30 May 2015; pp. 35–41.
11. Boubertakh, H.; Tadjine, M.; Glorennec, P.Y. A new mobile robot navigation method using fuzzy logic and a modified Q-learning algorithm. *J. Intell. Fuzzy Syst.* **2010**, *21*, 113–119. [[CrossRef](#)]
12. Zhang, R.; Tang, P.; Su, Y.; Li, X.; Yang, G.; Shi, C. An adaptive obstacle avoidance algorithm for unmanned surface vehicle in complicated marine environments. *IEEE/CAA J. Autom. Sin.* **2014**, *1*, 385–396.
13. Miao, X.; Lee, J.; Kang, B.Y. Scalable coverage path planning for cleaning robots using rectangular map decomposition on large environments. *IEEE Access* **2018**, *6*, 38200–38215. [[CrossRef](#)]
14. Barbehenn, M. A note on the complexity of Dijkstra’s algorithm for graphs with weighted vertices. *IEEE Trans. Comput.* **1998**, *47*, 263. [[CrossRef](#)]
15. Valtorta, M. A result on the computational complexity of heuristic estimates for the A* algorithm. *Inf. Sci.* **1984**, *34*, 47–59. [[CrossRef](#)]
16. Stentz, A. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*; Springer: Boston, MA, USA, 1997; pp. 203–220.
17. LaValle, S.M.; Kuffner, J.J.; Donald, B.R. Rapidly-exploring random trees: Progress and prospects. *Algorithmic Comput. Robot. New Dir.* **2001**, *5*, 293–308.
18. Kavraki, L.E.; Svestka, P.; Latombe, J.C.; Overmars, M.H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. Autom.* **1996**, *12*, 566–580. [[CrossRef](#)]
19. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*; Springer: New York, NY, USA, 1986; pp. 396–404.
20. Alonso-Mora, J.; Breitenmoser, A.; Rufli, M.; Beardsley, P. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 203–216.

21. Han, R.; Chen, S.; Hao, Q. A Distributed Range-Only Collision Avoidance Approach for Low-cost Large-scale Multi-Robot Systems. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 8020–8026.
22. Ataka, A.; Lam, H.K.; Althofer, K. Reactive magnetic-field-inspired navigation for non-holonomic mobile robots in unknown environments. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 6983–6988.
23. Tai, L.; Liu, M. Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not. *arXiv* **2016**, arXiv:1612.07139.
24. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6382–6393.
25. Everett, M.; Chen, Y.F.; How, J.P. Motion planning among dynamic, decision-making agents with 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. 3052–3059.
26. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)]
27. Wiering, M.A.; Van Otterlo, M. Reinforcement learning. *Adapt. Learn. Optim.* **2012**, *12*, 729.
28. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Kumar, V.; Zhu, H.; Gupta, A.; Abbeel, P.; et al. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905, 2018.
29. Christodoulou, P. Soft actor-critic for discrete action settings. *arXiv* **2019**, arXiv:1910.07207.
30. Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. *arXiv* **2019**, arXiv:1910.01741. [[CrossRef](#)]
31. Kiumarsi, B.; Vamvoudakis, K.G.; Modares, H.; Lewis, F.L. Optimal and autonomous control using reinforcement learning: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *29*, 2042–2062. [[CrossRef](#)] [[PubMed](#)]
32. Choi, J.; Lee, G.; Lee, C. Reinforcement learning-based dynamic obstacle avoidance and integration of path planning. *Intell. Serv. Robot.* **2021**, *14*, 663–677. [[CrossRef](#)] [[PubMed](#)]
33. Zhelo, O.; Zhang, J.; Tai, L.; Liu, M.; Burgard, W. Curiosity-driven exploration for mapless navigation with deep reinforcement learning. *arXiv* **2018**, arXiv:1804.00456, 2018.
34. Wang, C.; Wang, J.; Zhang, X.; Zhang, X. Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning. In Proceedings of the 2017 IEEE Global Conference on Signal and Information Processing (Glob-SIP), Montreal, QC, Canada, 14–16 November 2017; pp. 858–862.
35. Yang, X. An overview of the attention mechanisms in computer vision. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2020; Volume 1693, p. 012173.
36. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
37. Li, X.; Liu, Y.; Wang, K.; Wang, F.-Y. A recurrent attention and interaction model for pedestrian trajectory prediction. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 1361–1370. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.