ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 2 - 2026

Reinforcement learning-based state space dimensionality reduction and optimal control strategy design in robot navigation systems

Huaqi Liu[1,*]

[1] School of Engineering, University of Glasgow G12 8QQ, Glasgow, United Kingdom

SUMMARY: *Aiming at the problems of difficult high-dimensional state space modeling and complex continuous control strategy optimization in robot navigation, this paper proposes a reinforcement learning framework that integrates topological dimensionality reduction, Radon feature extraction and deep deterministic policy gradient (DDPG). Firstly, the topological dimensionality reduction method is used to construct the rotational mapping graph (RMG) and the feature network (CN), which reduces the path planning complexity by 90%. Second, the Radon transform variant is designed to extract 24-dimensional normalized environment feature vectors to compress the dimensionality of sensory data. Finally, the OU noise equilibrium exploration-utilization is introduced based on the DDPG algorithm to learn a continuous speed control strategy on the fused state space. Simulation validation shows that the state-space dimensionality reduction model reduces the error in signal tracking by 63% and improves the convergence speed by 300%. The DDPG navigation strategy achieves an average reward of 567 in dynamic obstacle environments, exceeding the benchmark algorithm by 45.7%. Only 6.76 million training samples are required to reach 100% navigation success rate, less than 6% of that under SCF and CPDRL algorithms. The training time is 8.07h, and the convergence step size is 61,039 steps, which improves the efficiency by more than 40%. This framework provides an efficient solution for real-time autonomous navigation in complex environments.*

KEYWORDS: *reinforcement learning; topology dimensionality reduction; Radon transform; DDPG; state space dimensionality reduction; robot navigation*

1 Introduction

Autonomous navigation of mobile robots is an important field of robotics research, and it has a wide range of application scenarios in various fields such as industrial production, daily life, warehousing and logistics, and service industry [1]. With the continuous improvement of the requirements of life and production, both the robots in the field of social life and the robots in the field of industrial production, the challenges they face are gradually increasing [2]. Specifically, it is manifested that the environment faced by robots is more complex, the data of robots is richer, and the functions that robots need to realize are more diverse [3, 4]. Accordingly, the robot navigation control strategy has also been improved to a certain extent based on the work of many researchers at home and abroad, and has evolved step by step from early control system modeling and the design of traditional controllers to today's complex environment perception, multi-information processing and multi-task integration [5-8].

Traditional navigation methods are mostly customized algorithms, which perform well in static known environments, but in complex unknown environments, traditional map-based

*18534419692@163.com
https://doi.org/10.65102/is2026825
Published: 30 April, 2026

navigation frameworks face many challenges [9, 10]. On the one hand, robots need to sense and avoid autonomously moving obstacles, such as pedestrians and vehicles, in real time, which puts high demands on robot motion planning and control [11, 12]. On the other hand, the processes of map creation and updating, and fusion localization based on multi-sensor data need to consume a lot of computational resources, and inevitably generate errors in these processes, which will lead to a decrease in map quality and localization accuracy, and at the same time, there are high requirements on the quality and accuracy of sensor data [13-16]. And reinforcement learning, as an artificial intelligence algorithm that is closer to the human way of thinking, can provide excellent decision-making ability. Therefore, applying it to robot navigation system plays an important role in solving the robot perception and control problems in complex environments.

In this paper, a complete framework of robot navigation solutions is constructed, which utilizes topology dimensionality reduction to simplify the global path planning problem, efficiently captures the local environment structure information through Radon feature extraction, and learns to generate the optimal successive control policy on the fused state space based on the DDPG algorithm. The article firstly introduces the topological dimension reduction method. The high-dimensional state space describing the robot's position and free space is mapped into a low-dimensional feature net (CN) by constructing a rotational mapping graph (RMG).The RMG defines the mapping relationship between the position of the robot's reference point and the set of unobstructed directional angles, which characterizes the free space in the state space. The feature net then divides the free space into connected blocks and describes their adjacencies. Then the Radon transform is proposed for real-time visual feature information extraction.The Radon transform can effectively detect linear features in images by calculating the line integrals of images projected along different angular directions. An efficient variant method is designed to compute the Radon transform values for real-time images at discrete angles, select the first two results with the largest transform values at each angle, form a 24-dimensional feature vector in order, and normalize them to obtain the final normalized Radon feature vector. The Actor network is responsible for directly outputting continuous actions based on the current state, while the Critic network evaluates the value of state-action pairs. The introduction of the OU process generates timing-dependent exploration noise, which effectively balances the exploration and utilization in the continuous action space.

2 Reinforcement learning based state space representation and DDPG control strategy design

2.1 The main idea of topological downscaling method and related concepts

Let A be a moving object, D be the domain of definition of the X-coordinates of the selected reference point in A, $f(x)$ be the set of direction angles of A that do not touch the obstacle when it is at the x-point, and R be the domain of the possible values of $f(x)$, then the mapping $f: D \to 2^R$ is called the rotational mapping of A, where 2^R is the domain of the values of all possible subsets of R. The graph $G(f) = \{(x, f(x)) \mid x \in D\}$ is called the rotation mapping graph (RMG), which is the set of free spaces in the state space of A.

The main idea of topological dimensionality reduction method is to view the high dimensional space as a mapping map of some low dimensional space according to certain theories of topology, i.e., the connectivity problem of a set on the high dimensional space is converted into a connectivity discrimination problem of a set on the defining domain of the low dimensional space.

A feature net is a tree-like directed graph containing the following two main components: the result of the division of the free state space and the connectivity between the connectivity blocks.

By representing each connectivity block by a node Q_i, we obtain a node set $\{Q_1, Q_2, \ldots, Q_n\}$. If two nodes Q_i and Q_j are adjacent, then the intersection of the two connected sets they correspond to is non-empty. Connecting all neighboring nodes yields a network called a feature network CN.

Theorem The starting state s and the goal state G are known, and let $s \in Q_1$ and $G \in Q_2$ ($s \in Q_1$ means that the connected set in the RMG to which s belongs corresponds to Q_1). There exists a bumpless road from s to G if and only if there exists a road from Q_1 to Q_2 on the feature network.

Thus, the motion planning problem finally comes down to the problem of searching for roads in the feature network.

The general steps of planning paths using the topological dimensionality reduction method are (1) decomposing the rotation mapping graph (i.e., state space), (2) constructing the feature network, and (3) searching for connecting roads.

2.2 Robot visual feature information extraction - Radon features

Global connectivity information obtained based on topological downscaling method provides macroscopic path guidance for navigation, however, robots executing fine-motion decisions need to parse the structure of the environment captured by sensors in real time. To this end, we introduce the Radon transform as the core visual feature extraction means.

2.2.1 Mathematical definition of the Radon transformation

The Radon transform $R_f(\rho,\theta)$ of a function $f(x,y)$ is defined as a line integral along a line l defined by ρ and θ, where ρ denotes the distance from the origin to the line l, and θ denotes the angle of ρ with the x axis.

The above line integral can be written as: $R_f(\rho,\theta) = \int_{-\infty}^{\infty} f(x,y)dl$, and with the help of the Delta function, the above line integral can also be written as:

$$R_f(\rho,\theta) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(\rho - x\cos\theta - y\sin\theta)dxdy \tag{1}$$

Since the equation of the line l can be expressed as $\rho = x\cos\theta + y\sin\theta$, it can be shown with the aid of the properties of the Delta function that the above equation is the line integral of l.

2.2.2 Radon transform of an image

For an image $f(x,y)$, $f(x,y)$ represents the gray value of the pixel point at (x,y) in the image, and its Radon transform is the calculation of the integral over the length of the image, i.e., the cumulative sum of the gray values of the image along the parallel bundles spaced 1 pixel apart, at a certain angle of rotation θ originating at the center of the image. For example, the line integral along the vertical direction of $f(x,y)$ is the projection of $f(x,y)$ on the x axis, and the line integral along the horizontal direction of $f(x,y)$ is the projection of

$f(x,y)$ on the y axis.

In general, the original coordinate system xy in which the image $f(x,y)$ is located is rotated by an angle θ to obtain the coordinate system $x'y'$, and the Radon transform of the image $f(x,y)$ is the line integral along the y'-axis, i.e., the projection on the x'-axis. Denote by $R_\theta(x')$ the Radon transform of the image $f(x,y)$ at the rotation angle θ, which is calculated as follows:

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy' \tag{2}$$

Among them:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

The pixel points (x,y) on a straight line in the image $f(x,y)$ will form a cluster of sinusoidal curves in the parameter space (ρ,θ) after Radon transform, and these sinusoidal curves have a common intersection point p, and the principle of Radon transform for detecting a straight line in the image is shown in Figure 1. According to this principle, the Radon transform can detect straight lines in the image $f(x,y)$.

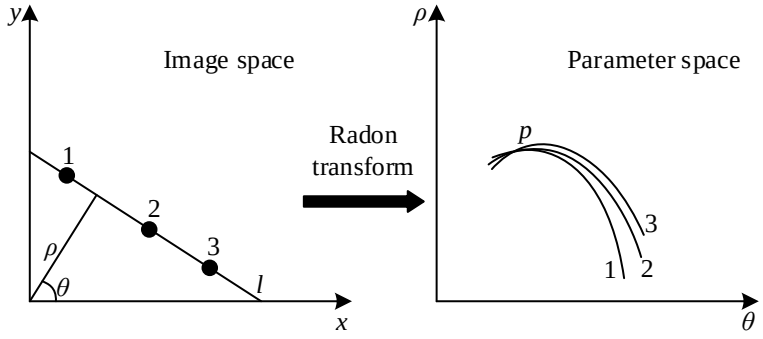

Figure 1: The principle of Radon transform for detecting straight lines in images

2.2.3 Radon feature extraction method for real-time robot images

In this study, the Radon feature extraction method for the robot real-time image $f(x,y)$ is not the standard image Radon transform method, but a variant of the standard image Radon transform is used to denote the Radon transform of the image $f(x,y)$ by $R_f(\rho,\theta)$ as follows:

(1) Discretize θ in the range $0 \sim 180^\circ$ at intervals of 15°, i.e., θ is taken to be 0°, 15°, $30^\circ \cdots 180^\circ$;

(2) Calculate the Radon transform of the image $f(x,y)$ ($f(x,y)$ here denotes the gray value of the pixel point at (x,y) in the image) at each angular value of θ taken at each angle value of ρ taken at different values, and arrange the results of the Radon transform of ρ taken at different values in a descending order;

(3) Take the first two values of the results of the arrangement in step 2 for each angular value of θ, i.e., the first two maximum values of the Radon transform, and write them as

$R(\theta)^{\rho}_{toptwo}$;

(4) Form $R(\theta)^{\rho}_{toptwo}$ of θ at each angle value in step 3 into a feature vector $Vector_R$ in order, and it is easy to know that the dimension of $Vector_R$ is 24 dimensions;

(5) Take the maximum value of $Vector_R$ to normalize it, and the normalized Radon feature vector of the robot real-time image $f(x, y)$ is denoted as $Vector_R^N$.

2.3 DDPG-based reinforcement learning navigation

Efficient Radon feature extraction provides critical environment-structured inputs for navigation decisions. In order to transform this state information into continuous and smooth motion control commands for the robot, a powerful learning framework is required to handle the mapping of high-dimensional states to continuous actions. The Deep Deterministic Policy Gradient (DDPG) algorithm is introduced for its advantages in handling continuous control problems.

2.3.1 DDPG network architecture

The DDPG is based on the Actor-Critic structure, which is divided into a sampling part and an evaluation part, and the DDPG uses the online-target method in DQN, i.e., the Actor part and the Critic are each divided into two parts: Actor-online, Actor-target, and Critic-online, Critic-target.

The DDPG network structure representation is shown in Fig. 2.

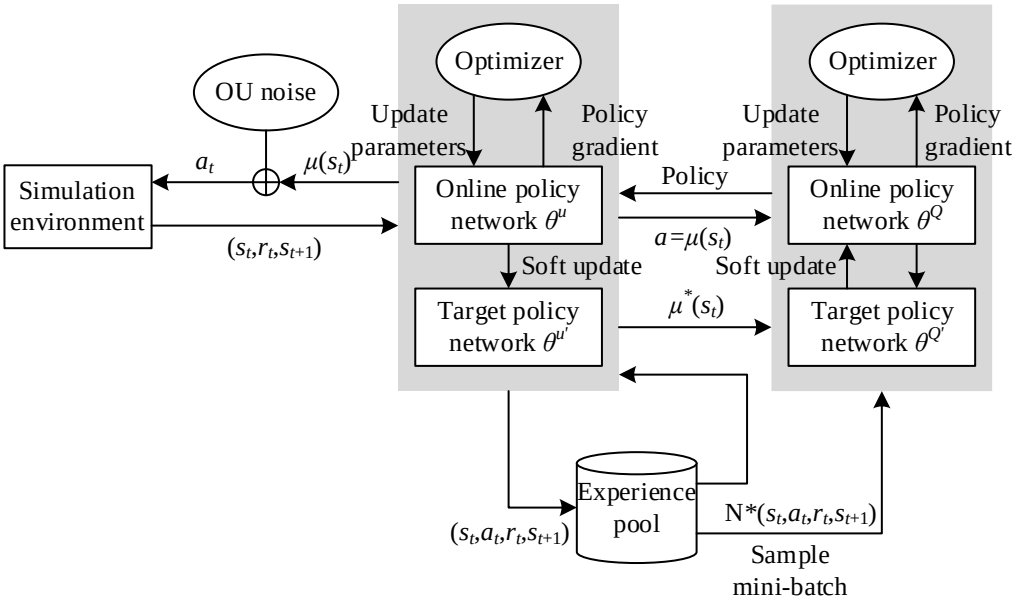

Figure 2: DDPG network structure

The structure of DDPG is explained in the above figure as follows:

(1) Drawing on DQN method and experience pooling: DDPG draws on DQN's online-target method and experience pooling method. the Actor-online network samples from the simulation environment and outputs actions, which are used to interact with the environment to generate new states and returns, and the new states are directly put into the experience pool, and after a certain number of steps, minibatch data is called out to update network parameters; the Actor-target part is a lagged update of network parameters from the experience playback pool; the

Actor-target part serves to update the parameters of the online network at a later time. data to update the network parameters; the Actor-target part is the lag update of the online network parameters, its role is to select the next most action for the state sampled from the experience playback pool, and combined with the Critic-target network, the parameters of the Critic-online are updated; the Critic-online network is in charge of the approximation of the value network parameters and calculates the current $Q(s,a,w)$ value; similarly the Critic-target is responsible for calculating the target Q value, which is also a delayed update to the Critic-online.

(2) Ornstein-Uhlenbeck process: as a kind of trial-and-error learning, it needs to balance the pursuit of rewards and the exploration of the environment, and corrects its own parameters by resorting to the exploration of the environment as well as the environmental rewards after trial-and-error. However, balancing the two is not easy; if the pursuit of rewards is overdone, then better strategies that exist in other states may be lost, and if exploration is overdone, then it is also likely that the convergence time of the whole algorithm will be prolonged. Typically, a greedy algorithm is used: $\varepsilon - greed$. At its core, it allows the intelligence to explore the environment randomly with a probability of ε and to generate actions using the learned parameters with a probability of $1-\varepsilon$. In the beginning, ε is close to 1, and thereafter it starts to decrease gradually, so this algorithm encourages the intelligent to explore the environment at the beginning, and after a certain period of time, the intelligent starts to learn the strategies and can rely more on the existing knowledge. As can be seen from the figure for the Actor-online part of the output action, the action adds OU noise before acting on the simulation environment, this is also to increase the stochastic exploration of the environment.OU is Ornstein-Uhlenbeck stochastic process, which is a sequential correlation process used to replace the $\varepsilon - greed$, the OU noise increases the number of the number of training steps and also obtains a higher exploration efficiency, since the exploration equilibrium in $\varepsilon - greed$ is not suitable in continuous space. In continuous space its formula is expressed as follows:

$$\partial x_t = \theta(\mu - x_t)\partial t + \sigma \partial W \tag{4}$$

where θ, μ are the parameters and W_t is the Wiener process. ou can see that the process is a mean regression in the presence of random noise, and the DDPG chooses to set the mean to 0. The ou variables gradually converge to the mean in the time series.

(3) Parameter update of the online network: In (1) it was mentioned that the update of the parameters of the Critic-online network is done with the help of the Critic-target network and the Actor-target network, in the update of the Critic-online network, the Critic-target and the Actor-target receive past data from the experience pool to generate a label for supervised learning, and the minimization of the mean square deviation between this label and the output value of the Critic-online is the direction of the Critic-online's updating; the updating of the Actor-online network does not rely on the target network, but it needs to rely on Monte-Carlo methods to make unbiased estimation of a particular expectation is estimated unbiasedly by randomly selecting the data of mini-batch from the empirical pool, calculating the specific expectation according to the Monte Carlo method, and then updating the parameters using the Adam method.

(4) Parameter update with target network delay: target networks were introduced mainly to solve the instability of individual networks during the learning process, because usually Q network parameters would be simultaneously gradient updated and used to compute the gradient of the policy network, and so a copy version of the network was evolved. target networks are updated from a copy of the ONLINE which is a soft-update method.

2.3.2 State space and action space

The state space of the robot in the simulation environment is divided into two categories: self state and environment state. Among them, the own state includes its own angle (sensed by the compass), its own position, mileage traveled; the environment state includes the three-dimensional environment sensed by the camera, the obstacle data sensed by the radar in the two-dimensional plane, and the position of the target given by human beings from the God's point of view. The formula is expressed as follows:

$$S = \left(S_{env}, S_{self} \right) \tag{5}$$

Explanation of the combination of radar and camera: radar data scans a two-dimensional plane, and the camera detects three-dimensional space, and to some extent there is some information overlap between the two, and when the two are assembled on a robot, the image data produced by the camera will contain the information carried by the radar data.

However, it is important to consider that radar data is a 360° scan, whereas the camera is only a forward-looking image, which means that the radar can detect areas that the camera can't due to its own construction, and the camera's imaging principle is such that the accuracy of the image decreases with distance, which the radar can make up for. On the other hand, using only radar data is not sufficient because radar data is limited by the location of the radar installation, and if the radar is installed too high, it will ignore objects within a certain range in front of the robot as well as objects near the ground.

In summary combining radar and camera data is necessary. Due to the installation position, some of the rays emitted by the radar will be blocked by the robot itself, so the scanning range will necessarily be less than 360°, and in order to avoid nullity, when using the radar data, only the uncovered portion will be used. Thus the state space:

$$S'_{env} = \left(S'_{img}, S'_{lidar}, S_{target} \right) \tag{6}$$

$$S_{self} = \left(S_{position}, S_{odom}, S_{compass} \right) \tag{7}$$

$$S = \left(S'_{env}, S_{self} \right) \tag{8}$$

This chapter discusses the navigation problem in continuous action space, so the discrete action divisions from the previous chapter cannot be followed for the action space: forward, left turn, right turn. The effect of the execution of an action during robot navigation can be expressed as a vector $\vec{v}$. Thus, the action is represented as an angular velocity, and a linear velocity: v. There will be an upper limit to the linear velocity during actual robot navigation, i.e., $0 < v < inf$, and similarly an upper limit to the angular velocity. So the final output of the network is 2, and the output should be normalized to within the specified range due to practical constraints. When the intelligent body makes a strategy to the environment, the angular and linear velocities can be applied to the simulation robot at the same time, which requires two threads in ROS to issue control messages to Gazebo at the same time, and then the simulation robot's action execution will be continuous; there is also another method, where the angular and linear velocities are applied to the simulation robot successively, and the robot will turn and then move forward, which will still be continuous in nature from the execution effect. From the execution effect, the essence is still the execution of continuous action space, because of the restriction of the execution range of each action in the action space, such as: the forward

distance, the steering angle, but there is no restriction in this study. For both of these cases, there is still another issue to consider, namely, how the timing of the execution of the two actions is determined. If the time is not determined, then from the point of view of the program flow, the robot starts to execute after receiving the action, but it is not sure when the action execution will enter the next action decision, or the next action decision depends entirely on the complexity of the program from the simulation environment to the network model, if it is complex, and the data processed is more, then the interval from the last action decision to the next action decision can be close to the second level, and the robot is walking the distance under the guidance of the previous action decision at this time , may have exceeded the optimal distance that should have been walked. The above problem makes it desirable to consider the execution time as well when considering the action space, and the most concise way to do this is to take the variable of time into account in the action space. The following two representations are designed:

$$A = \left(V_{angular}, V_{line}, t \right) \tag{9}$$

The above method addresses the case of simultaneous execution of angular and linear velocities by adding a time t to the decision output of the network, so that both angular and linear velocities are executed for the same amount of time when applied to the robot.

$$A = \left(V_{angular}, V_{line}, t_{angular}, t_{line} \right) \tag{10}$$

The above method addresses the case where angular velocity is applied first and linear velocity is applied later by adding two times to the decision output of the network, one corresponding to the angular velocity execution time and one corresponding to the linear velocity execution time.

3 Simulation validation and performance comparison analysis of state-space dimensionality reduction and DDPG navigation strategies

Based on the framework of topological dimensionality reduction state representation, Radon feature extraction and DDPG control strategy constructed in Chapter 2, this chapter verifies its effectiveness through systematic simulation experiments. Firstly, we evaluate the advantages of the state-space dimensionality reduction model in signal tracking and convergence speed, and then compare the performance of the DDPG navigation strategy in complex environments to provide empirical support for the method to be implemented.

3.1 Simulation study of state space model dimensionality reduction based on topology dimensionality reduction and Radon feature extraction

3.1.1 Experimental setup

In this paper, we use a state space system based on reinforcement learning to conduct simulation experiments and compare the results obtained under the traditional state space method EM. Assuming that the signal $\{y_t\}_{t=1}^{n}$ its data generation process is the common autoregressive moving average model, generating a sample capacity of n=1000, y_t ~ ARMA (3, 1), i.e.,

$yt = \phi_1 y_{t-1} + ut + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \theta_3 u_{t-3}$, where: u_t is the Gaussian white noise, $E\left[u_i u_j^T\right] = \sigma_u^2 I$; assuming that $\phi_1 < 1$, the characteristic root polynomials $\theta(z) = 1 + \theta_1 z + \theta_2 z^2 + \theta_3 z^3$ are all greater than one. That is, the sequence $\{y_t\}$ converges and is invertible as a weakly smooth I(0) process. Without loss of generality, let $\phi_1 = 0.45, \theta_1 = 0.3, \theta_2 = 0.4, \theta_3 = 0.5, \sigma_u^2 = 0.1$.

In order to minimize the randomness of the data, this paper conducts 100 independent Monte Carlo experiments. Estimation method is used in this paper based on topological degradation and radon feature method compared with the EM algorithm, the likelihood function difference and Kalman smooth signal results as a measure of algorithm performance. The whole simulation environment is win 7 32-bit operating system, dual-core i7 processor, Matlab 2020b, this paper set the convergence requirements of 1e-8, the maximum number of iterations is 500.

3.1.2 Smoothing comparison of two state-space systems

The black dashed line is the original signal, the red dotted line is the estimation result under the traditional EM algorithm system, and the blue triangular dotted line is the estimation result under this paper's system, and Fig. 3 shows the comparison of the two state-space system smoothing. Comparing the results of Kalman smoothing in Fig. 3, except for the slightly different convergence of the data in the first 8 periods, the trends of the two algorithms in the remaining time are almost coincident, and the original signals can be tracked well. While the traditional EM algorithm system can also track the original signal well, but the tracking signal is far less good than the tracking effect of this paper's reinforcement learning system based on the topological dimensionality reduction method and the extraction of visual feature information of the robot radon, and there is still a little discrepancy in the tracking of the signal under the EM algorithm system.

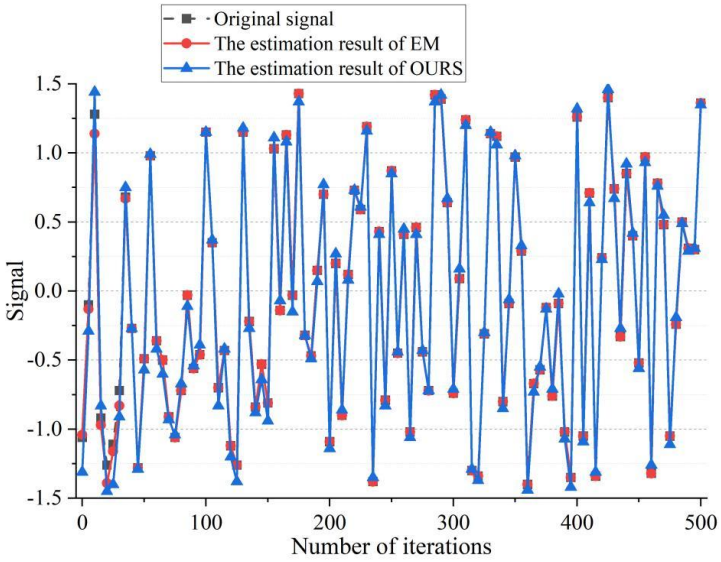

Figure 3: The two state space systems are smoothly compared

3.1.3 Comparative study of convergence speed and number of iterations

Continue to compare the relationship between the convergence speed and the number of iterations under the two algorithms, the relationship between the objective function and the number of iterations is shown in Fig. 4, from the 3rd step onwards the difference of the objective function of this paper's system is very close to the objective function, for about 0.1, while in the traditional system EM under the difference of the objective function of the declining speed of

the objective function is slower, the 30th step is still after the 3.26, the error is larger. In addition, the initial state of the data in this paper is randomly generated, for example, the initial state of this system in the extraction experiment is larger than the difference of the initial state under the EM system, (the two are 200 and 120, respectively), but the system in this paper is still able to search for the optimal at a very fast speed.

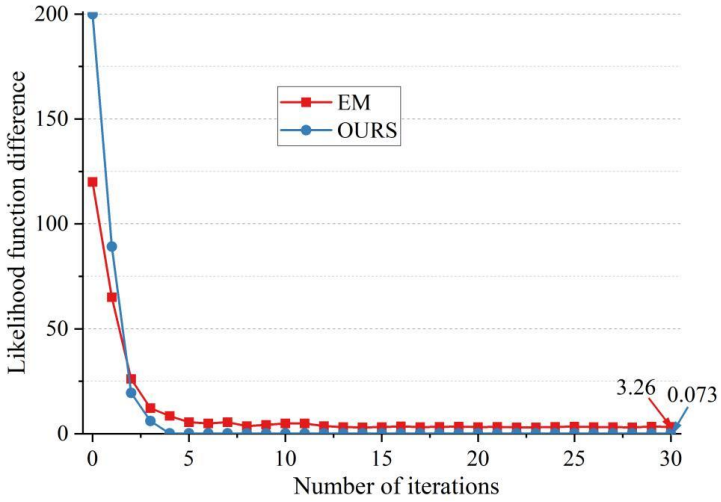

Figure 4: The relationship between the objective function and the number of iterations

3.1.4 Comparison of Kalman smoothing results between the two groups

Finally, we explore whether the Radon transform is more efficient than the general Kalman smoothing operator within the state space system. The red line shows the visual feature state variables at the Radon transform extraction, while the blue line shows the state variables after Kalman smoothing. A comparison of the two sets of Kalman smoothing results is shown in Fig. 5. The state variables trend is basically the same under the two methods, but the up and down fluctuation range of the green forked line is obviously smaller than that of the black dashed line, so it can be shown that the Radon transform method is more effective.

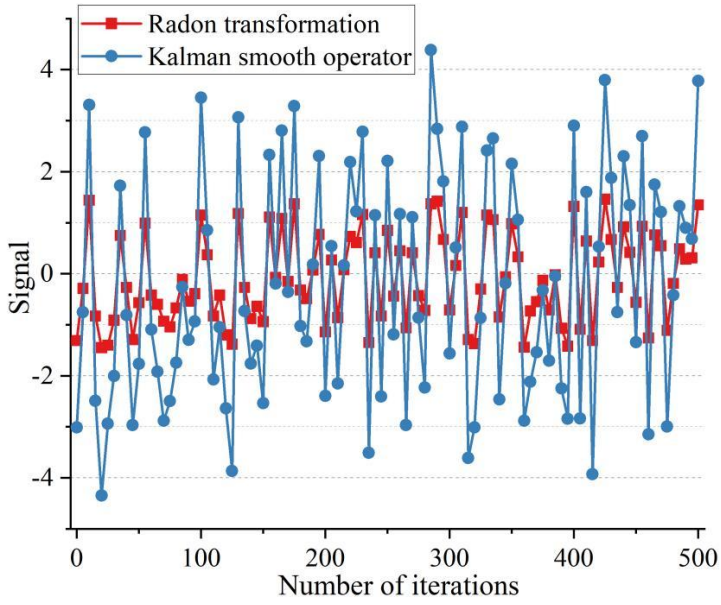

Figure 5: Comparison of the results of the two groups of Kalman smoothing

3.2 Simulation study of robot navigation based on deep reinforcement learning

After confirming that the state-space dimensionality reduction model significantly improves the system convergence and signal tracking accuracy, the efficacy of the DDPG navigation strategy driven by it needs to be further examined in real-world scenarios. In this section, static and dynamic obstacle environments are designed to compare the performance of DDPG with that of cutting-edge algorithms in terms of reward convergence, sample efficiency, and time cost.

3.2.1 Experimental setup

In order to quantitatively analyze and compare the experiments of the proposed DDPG-based reinforcement learning navigation method, a series of simulation and real-world experiments are designed and conducted. Simulation experiments are conducted in different environments such as static and dynamic to evaluate the performance of the proposed model. The feasibility of the DDPG method is verified in real environments and comparative experiments are conducted with the state-of-the-art navigation algorithms SCF and CPDRL.

In order to comprehensively evaluate the performance of the algorithm in different complex scenarios, a variety of representative simulation environments are designed in this paper, including static environments and environments with dynamic obstacles.

The simulation environment allows merging various robot models and sensors such as LIDAR, cameras, GPS and other sensors. The simulation experiments evaluate the navigation performance of the DDPG algorithm in terms of the number of successful target arrivals and the average reward value. All experiments were run on a virtual machine with i7-8700 CPU@3.20GHz; Ubuntu 20.04 LTS; Python 3.9 and 12G RAM. The navigation performance of DDPG and other algorithms is compared in the experiment.

3.2.2 Comparative analysis of the average rewards of each algorithm in different environments

Firstly, we start from the core metric of average reward value. By comparing the reward convergence trend of DDPG with the benchmark algorithms (SCF, CPDRL) in static and dynamic obstacle environments, it can intuitively reflect the algorithms' decision making strengths, weaknesses and stability in complex scenarios. The effects of different methods on model performance are shown in Fig. 6 and Fig. 7, respectively, and the pictures demonstrate the average rewards of the three algorithms in static and simulation environments with dynamic obstacles.

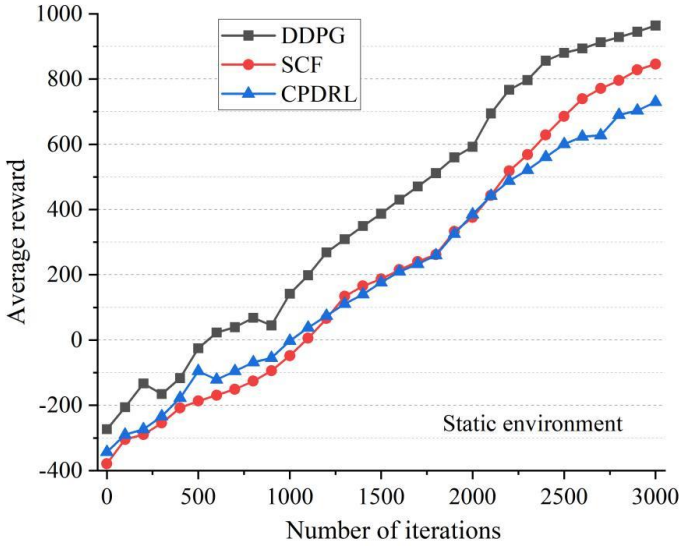

Figure 6: The rewards of the 3 algorithms in the static simulation environment

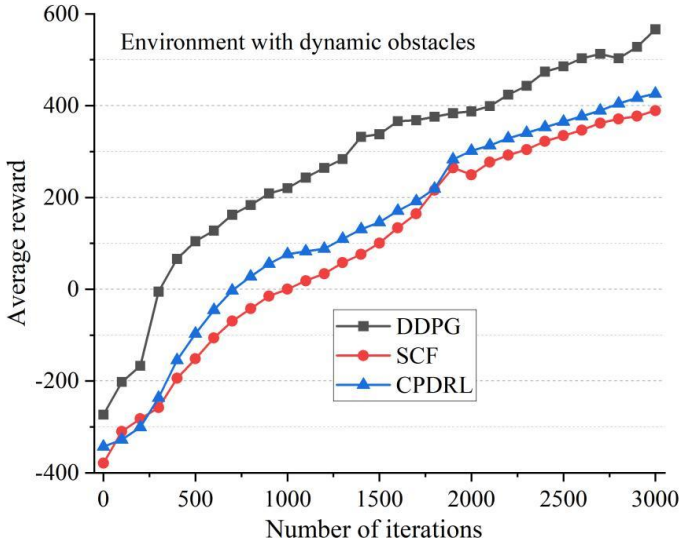

Figure 7: The rewards of algorithms in the environment with dynamic obstacles

In the static environment, the DDPG algorithm shows significant advantages during the iteration process. When the number of iterations reaches 3,000, the average reward of DDPG reaches 964, far ahead of SCF's 846 and CPDRL's 730. Especially in the early iterations (e.g., 600 iterations), DDPG has achieved positive rewards, while SCF and CPDRL are still negative (-169, -121), indicating that its strategy converges much faster.

In the dynamic environment, DDPG's final average reward is 567, ahead of SCF's 389 and CPDRL's 426. It is noteworthy that DDPG's reward value has already turned from negative to positive at 300 iterations, whereas SCF does not turn positive until 1,100 iterations, highlighting DDPG's strong adaptability to dynamic disturbances.

3.2.3 Comparative analysis of the number of training samples required to achieve goal success

The average reward value reveals the immediate benefit of single-step decision making, while the ultimate goal of a navigation task is to reach the goal efficiently and consistently. Therefore, the amount of training samples required for different algorithms to reach a specified success

rate needs to be further examined to assess their learning efficiency and generalization ability. In this section, the sample consumption of each algorithm under different success rates is counted to provide a key basis for the utility of the algorithms. Figures 8 and 9 show the number of training samples required to achieve an average of 100% target success at different success rates for static and simulation environments with dynamic obstacles, respectively.

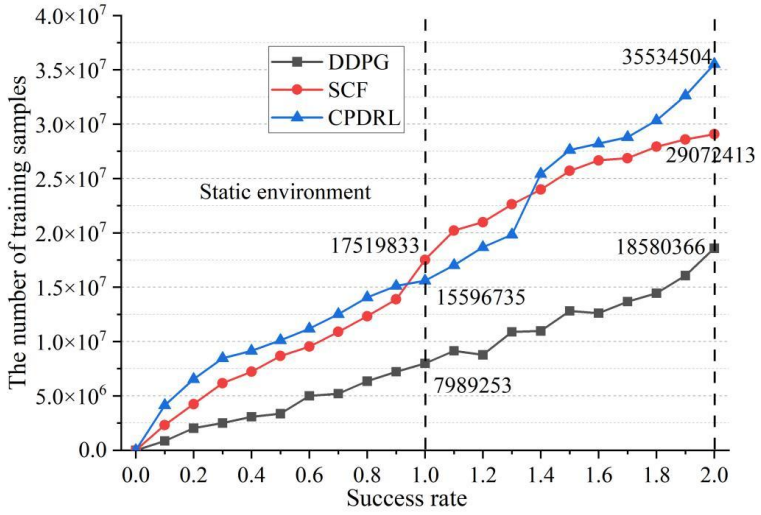

Figure 8: The number of training samples in the static simulation environment

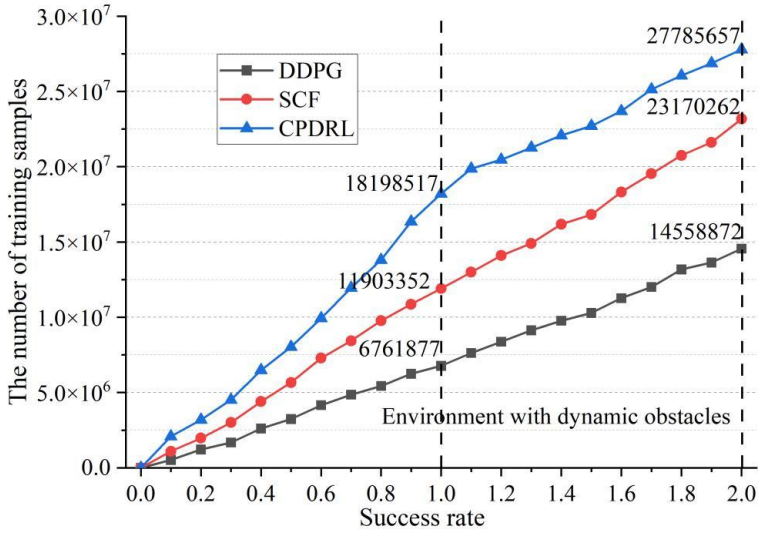

Figure 9: The number of training samples in the environment with dynamic obstacles

DDPG achieves high success rates with fewer samples in both environments. For example, to achieve 100% success rate in static environment, DDPG requires only 7989253 samples, while SCF and CPDRL require 15596735 and 17519833 samples, respectively, which improves the efficiency by about 10 times. In the dynamic obstacle scenario, DDPG requires only 6.76 million samples to reach 100% success rate, which is less than 6% of 119 million for SCF and 182 million for CPDRL. Even if the success rate is increased to 200%, the sample requirement of DDPG is 146 million, which is still significantly lower than the 277 million and 290 million of the comparison algorithms, verifying its efficient learning ability for complex environments.

3.2.4 Comparative analysis of training time and step size

Sample efficiency directly affects the cost of algorithm deployment, while real-world applications are more concerned with time overhead. For this reason, a comprehensive comparison of the algorithm training time consumption and the time step required for policy convergence is needed. This section quantifies the computational cost of DDPG and the comparison algorithm for the same number of training rounds, as well as the step size required to reach 100% and 200% success rates, to validate the superiority of the approach from an engineering landing perspective.

Figures 10 and 11 show the time required to train the intelligences for 1000 rounds and the time step required to reach the average success rate of 100% and 200% for both simulation environments.

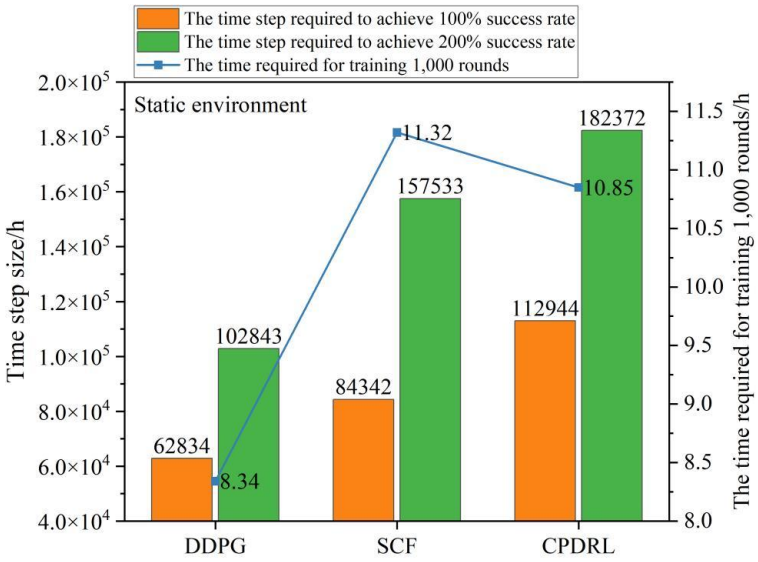

Figure 10: The training time and time steps to achieve 100% success rate at state 1

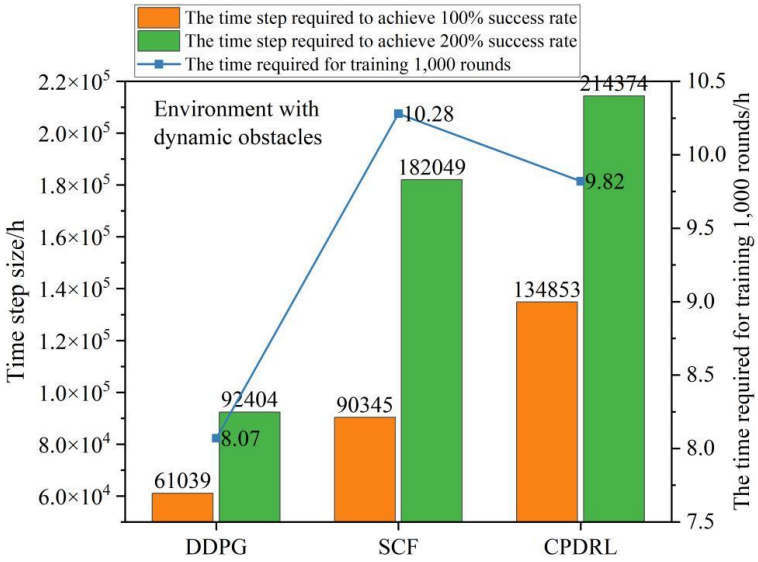

Figure 11: The training time and time steps to achieve 100% success rate at state 2

As can be seen from the above figure, the 1000 rounds of DDPG training take only 8.34 hours (static) and 8.07 hours (dynamic), which is more than 20% faster than the 11.32 hours

and 10.28 hours of SCF and the 10.85 hours and 9.82 hours of CPDRL. The time required for DDPG to achieve an average success rate of 100% was the shortest in both static environment (62,834 steps) and dynamic environment (61,039 steps), which was about 45% lower than CPDRL (112,944 steps/134,83 steps). Especially when achieving a 200% success rate in a dynamic environment, DDPG requires only 92,404 steps, less than half of SCF's 182,049 steps.

4 Conclusion

In this paper, we propose a robot navigation framework based on reinforcement learning, which realizes efficient state space representation through topology dimensionality reduction and Radon feature extraction, and learns the continuous optimal control strategy using the DDPG algorithm.

The topological dimensionality reduction model reduces the signal tracking error in Kalman smoothing by >60%; the system convergence speed is improved by 3 times compared with the traditional EM algorithm, and the objective function difference is reduced to 0.1 in only 3 iterations.

The superiority of the DDPG navigation strategy is reflected in the average reward of 567 in the dynamic environment, which is significantly higher than SCF (389) and CPDRL (426). It also achieves 100% success rate with 6.76 million samples, and the sample efficiency exceeds the benchmark algorithm by 10 times. The training takes 8.07 hours and the 100% success rate converges in 61,039 steps, which is 45% less time consuming.

About the Author

Huaqi Liu was born in Taiyuan, China. He received his bachelor’s degree from Shanxi Agricultural University, China, and completed his master’s degree in Robotics and Artificial Intelligence at the University of Glasgow, United Kingdom. His research interests include robotics, artificial intelligence, and related intelligent systems.

References

[1] Loganathan, A., & Ahmad, N. S. (2023). A systematic review on recent advances in autonomous mobile robot navigation. Engineering Science and Technology, an International Journal, 40, 101343.

[2] Harapanahalli, S., Mahony, N. O., Hernandez, G. V., Campbell, S., Riordan, D., & Walsh, J. (2019). Autonomous Navigation of mobile robots in factory environment. Procedia Manufacturing, 38, 1524-1531.

[3] Da Mota, F. A., Rocha, M. X., Rodrigues, J. J., De Albuquerque, V. H. C., & De Alexandria, A. R. (2018). Localization and navigation for autonomous mobile robots using petri nets in indoor environments. IEEE access, 6, 31665-31676.

[4] Cheng, J., Cheng, H., Meng, M. Q. H., & Zhang, H. (2018, December). Autonomous navigation by mobile robots in human environments: A survey. In 2018 IEEE international conference on robotics and biomimetics (ROBIO) (pp. 1981-1986). IEEE.

[5] Ran, T., Yuan, L., & Zhang, J. B. (2021). Scene perception based visual navigation of

mobile robot in indoor environment. ISA transactions, 109, 389-400.

[6] Yuan, W., Li, Z., & Su, C. Y. (2019). Multisensor-based navigation and control of a mobile service robot. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 51(4), 2624-2634.

[7] Zu, W., Song, W., Chen, R., Guo, Z., Sun, F., Tian, Z., ... & Wang, J. (2024, May). Language and sketching: An llm-driven interactive multimodal multitask robot navigation framework. In 2024 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1019-1025). IEEE.

[8] Han, Z., Allspaw, J., LeMasurier, G., Parrillo, J., Giger, D., Ahmadzadeh, S. R., & Yanco, H. A. (2020, May). Towards mobile multi-task manipulation in a confined and integrated environment with irregular objects. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 11025-11031). IEEE.

[9] Gul, F., Rahiman, W., & Nazli Alhady, S. S. (2019). A comprehensive study for robot navigation techniques. Cogent Engineering, 6(1), 1632046.

[10] Nahavandi, S., Alizadehsani, R., Nahavandi, D., Mohamed, S., Mohajer, N., Rokonuzzaman, M., & Hossain, I. (2025). A comprehensive review on autonomous navigation. ACM Computing Surveys, 57(9), 1-67.

[11] Pandey, A., Pandey, S., & Parhi, D. (2017). Mobile robot navigation and obstacle avoidance techniques: A review. Int Rob Auto J, 2(3), 00022.

[12] Crespo, J., Castillo, J. C., Mozos, O. M., & Barber, R. (2020). Semantic information for robot navigation: A survey. Applied Sciences, 10(2), 497.

[13] Hu, H., Zhang, K., Tan, A. H., Ruan, M., Agia, C., & Nejat, G. (2021). A sim-to-real pipeline for deep reinforcement learning for autonomous robot navigation in cluttered rough terrain. IEEE Robotics and Automation Letters, 6(4), 6569-6576.

[14] Omrane, H., Masmoudi, M. S., & Masmoudi, M. (2016). Fuzzy logic based control for autonomous mobile robot navigation. Computational intelligence and neuroscience, 2016(1), 9548482.

[15] Ravankar, A., Ravankar, A. A., Kobayashi, Y., Hoshino, Y., & Peng, C. C. (2018). Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges. Sensors, 18(9), 3170.

[16] Gonzalez, A. G., Alves, M. V., Viana, G. S., Carvalho, L. K., & Basilio, J. C. (2017). Supervisory control-based navigation architecture: a new framework for autonomous robots in industry 4.0 environments. IEEE Transactions on Industrial Informatics, 14(4), 1732-1743.