Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Recursive Neural Network Optimization Strategies for Multi-Target Detection and Path Planning in Autonomous Systems for Multi-Intelligent Collaboration

Yuxuan Li[1,✉]

[1] *School of Information, Shanxi University of Finance and Economics, Taiyuan, Shanxi, 030006, China*

ABSTRACT

The article applies recurrent neural networks to multi-intelligent body collaborative autonomous systems and uses optimized RNN algorithms for multi-objective detection and path planning of intelligent bodies. The multi-intelligent body multi-target detection and path planning model optimized based on recurrent neural network is constructed to realize multi-target detection and tracking of intelligent bodies and multi-intelligent collaborative path planning. Simulation experiments are designed with a mobile robot as the research object to analyze the trajectory tracking and path planning effects of the multi-target detection and path planning model in this paper. The error between the actual trajectory and the reference position of the robot trajectory tracking is continuously reduced, and reaches complete coincidence at the 127th reference tracking point. The actual speed and acceleration errors of the robot are infinitely close to 0. The accuracy of this paper's algorithm in multi-objective path planning is 100%, the average arrival time is 20.02s, and the probability of collision is 0%, which is much better than other algorithms. The algorithm in this paper has the highest path smoothing validity for planning in three environments. In the 30 × 83 warehouse map, the total path length of this paper's algorithm is shortened by 13.00% and 10.77%, and the total path cost is shortened by 9.71% and 11.52% compared with the Wd-SIPP algorithm for the number of collaborative robots in a single group of three and five, respectively. In 100*100 storage map, the total path length is shortened by 10.32% and 11.67%, and the total path cost is shortened by 7.34% and 12.09%, respectively.

Keywords: recurrent neural network, target detection, target tracking, path planning, multi-intelligence body collaboration

1. Introduction

With the rapid development of 5G technology and the continuous improvement of the computing

✉ Corresponding author.
E-mail address: 18335548302@139.com (Y. Li).
Received 01 February 2024; Revised 20 April 2024; Accepted 10 November 2024; Published Online 15 April 2025.
DOI: 10.61091/jcmcc127a-507

power of computing devices represented by GPUs, the importance of edge computing has become increasingly prominent. On the one hand, target detection, recognition and tracking algorithms based on single-view image encounter a bottleneck in performance enhancement, and big data-driven deep learning algorithms are difficult to solve the problem of drastic deformation and anti-obscuration of targets in complex environments, and it is necessary to introduce the information of heterogeneous viewpoints to enhance the algorithm performance and robustness [1-2]. On the other hand, the traditional cloud computing model presents drawbacks such as communication bandwidth bottleneck and security [3]. The study of target detection, recognition and tracking methods oriented to multi-intelligence body collaboration can be a good solution to the problem of effective target recognition and accurate positioning under strong anti-jamming conditions.

The introduction of multi-view multi-sensor information can bring richer data and realize the complementary advantages of different kinds of sensor information. The collaborative information processing system using edge computing architecture is not only in line with the development of today's technology trend, but also helps to break through the development bottleneck encountered by deep learning. In addition, the essence of collaboration is the aggregation of multi-source information and global decision-making process, for different application scenarios, the appropriate architecture should be used in order to fully explore the hidden information in massive data [4-5]. A good collaborative system should be a modular information processing system that combines simplicity, robustness, and stability [6]. Among them, detection planning is a prerequisite for autonomous system detection, which mainly includes two parts: target detection and path planning, both of which are proved to be nondeterministic problems of polynomial complexity, where the planned paths are difficult to be optimized, thus affecting the efficiency of detection.

Literature [7] shows that Multi-Intelligent Physical Collaborative Planning (MAP) is a technique that combines the fields of AI planning and multi-intelligent systems, and compared to the traditional planning process for single-task goals, MAP can accomplish the action process of group goals through the cooperation of multiple intelligences. Literature [8] constructed two collaborative path planning frameworks for multi-intelligent body systems, the compromised view model and the nearest neighbor search model, both models can not only control multiple autonomous systems to complete multiple tasks at different locations, but also dynamically adapt to changing environments, and by comparing the length of the planning time and the path length of the two models, it is found that the nearest neighbor search model has a quicker path search speed, but its generated paths are longer . Literature [9] proposes a reinforcement learning based multi-intelligent spatial planner (MSP), which recognizes spatial relationships and intra-intelligent interactions through hierarchical attention and combines with multi-intelligent augmentation techniques to achieve aligned spatial representations and more accurate planning, and experiments show that the proposed scheme outperforms the classical planning-based baseline in 3D simulations. Literature [10] designed an information-based multi-objective intelligent body control algorithm in conjunction with Markov Decision Process (POMDP), in order to compute the optimal path planning strategy, firstly, the proposed multi-objective value function is proved to be a monotone submodular ensemble function, and secondly, the greedy search algorithm is utilized to successfully decouple the low-cost suboptimal solutions. Literature [11] explored the multi-intelligence body path planning problem incorporating cooperative behaviors, in order to complete the collaborative task and avoid interactive collisions, the cooperative conflict-based search (Co-CBS) algorithm is proposed, which decouples the cooperative planning from the path planning and ensures that the generated paths are optimal paths. Literature [12] studied the multi-target search problem in uncertain environments, using a multi-intelligent target cooperative search task planning model and a reinforcement learning algorithm for action preference selection strategy, the proposed target search algorithm significantly improves the success rate in multi-target search tasks, search efficiency, and generates a more concise movement trajectory. Literature [13] addresses the problem that classical learning algorithms are not robust to control decisions in uncertain state-

space models, and establishes a neural network-based state estimator to obtain higher tracking capability and lower localization error for moving targets by explicitly giving the target state and estimating the uncertainty and feeding it into the multi-intelligent body reinforcement learning model. Literature [14] used recurrent neural network to predict the target information among distributed agents, optimized the updating rule and cooperative model of target probability graph, and effectively solved the problems of communication interruption and control failure of multi-intelligence collaborative search learning algorithms in complex environments. Literature [15] introduces Long Short-Term Memory Neural Network (LSTM) to Multi-Intelligent Reinforcement Learning (MARL) algorithm to improve the UAV's ability to describe complex systems and predict dynamic environments in the task of detecting and tracking targets.

In this paper, through the optimization processing of traditional recurrent neural network algorithm, multi-intelligent body multi-target detection, multi-target tracking and path planning based on RNN optimization are realized, and multi-target detection and path planning model based on RNN optimization is constructed. Design related simulation experiments, choose mobile robot as the research object, and deeply explore the trajectory tracking effect of this paper's multi-target detection and path planning algorithm based on RNN optimization. Compare this paper's model with other algorithm models, and test the effectiveness of this paper's algorithm model by comparing the path length, smoothness, path cost and other indexes of each algorithm on multi-objective path planning and multi-intelligent body collaborative path planning.

2. Multi-objective detection and path planning based on RNN optimization

2.1. Multi-objective detection algorithm based on RNN optimization

2.1.1. Recursive neural network structure. To utilize the implied sequence information, spatially supervised recurrent neural network based multi-target detection is used in this chapter. Recurrent neural network (RNN) is a general term for two types of artificial neural networks: temporal recurrent neural networks and structural recurrent neural networks [16-17]. It is a type of neural network that models sequential data, i.e., the current output of a sequence is also correlated with the output result of the previous time step. The specific manifestation is that the network memorizes the previous information and applies it to the calculation of the current output, the nodes between the hidden layers are also related to each other, and the input of the hidden layer includes not only the output of the input layer but also the output of the hidden layer in the previous moment.

2.1.2. Recursive Neural Network Optimization Algorithm BPTT. In this section of the model, the inputs are the feature vectors corresponding to a single frame of the image, while the outputs are the location information and category probabilities of the corresponding target. The input matrix W maps the input vectors to the hidden layer space, transforming the inputs into a form that is understandable and processable within the model. U on the other hand is the mapping of the hidden layer to itself, which defines how the model incorporates contextual information and utilizes previous history information. W , U combines the current input vectors as well as previous historical information to form the state of knowledge at the current moment. V is the mapping from the implicit layer to the output, and $z = Vh$ is the output score of each component after the mapping, and the output probability after the softmax layer.

In this section back propagation over time algorithm (BPTT) parameters:

$$\begin{cases} s_t = Uh_{t-1} + Wx_t \\ h_t = \tanh\left(s_t\right) \\ z_t = Vh_t \\ \hat{y}_t = soft\max\left(z_t\right) \\ E_t = -y_t^T \log\left(\hat{y}_t\right) \\ E = \sum_t^T E_t \end{cases} \tag{1}$$

Calculate the partial derivative of the loss function at moment t with respect to the weights from the hidden layer to the output layer $\frac{\partial E_t}{\partial V}$:

$$\frac{\partial E_t}{\partial V} = \left(\hat{y}_t - y_t\right) \otimes h_t \tag{2}$$

$$\frac{\partial E}{\partial V} = \sum_{k=0}^{t} \left(\hat{y}_t - y_t\right) \otimes h_k \tag{3}$$

Calculate the partial derivative of the loss function at moment t with respect to the weights of the hidden layer from the previous moment to the current moment $\frac{\partial E_t}{\partial U}$:

$$\Delta_t = \left(V^T\left(\hat{y}_t - y_t\right)\right) \square \left(1 - h_t \square h_t\right) \tag{4}$$

$$\Delta_k = \left(U^T \delta_{k+1}\right) \square \left(1 - h_k \square h_k\right) \tag{5}$$

$$\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \delta_k \otimes h_{k-1} \tag{6}$$

$$\frac{\partial E}{\partial U} = \sum_{k=0}^{T} \sum_{k=0}^{t} \delta_k \otimes h_{k-1} \tag{7}$$

Calculate the loss function at moment t with respect to the weights from the input layer to the hidden layer $\frac{\partial E_t}{\partial W}$:

$$\frac{\partial E_t}{\partial U} = \sum_{k=0}^{t} \delta_k \otimes x_k \tag{8}$$

Combine the above E_t on the partial derivatives of each parameter and update each parameter using gradient descent:

$$V = V - \lambda \sum_{k=0}^{T} \left(\hat{y}_t - y_t\right) \otimes h_t \tag{9}$$

$$U = U - \lambda \sum_{k=0}^{T} \sum_{k=0}^{t} \delta_k \otimes h_{k-1} \tag{10}$$

$$W = W - \lambda \sum_{k=0}^{T} \sum_{k=0}^{t} \delta_k \otimes x_k \tag{11}$$

Among them, $\delta_t = \left(V^T\left(\hat{y}_t - y_t\right)\right) \square \left(1 - h_t \square h_t\right)$.

After many rounds of backpropagation, the internal parameters of the LSTM are continuously

adjusted as the loss decreases, and finally the network is able to fit the distribution of the sequence data better.

2.2. Multi-feature based RNN multi-target tracking algorithm

2.2.1. Multi-feature based RNN online tracking framework. In this section, a multi-feature based RNN online multi-target tracking algorithm is proposed. The overall framework of the tracking model is shown in Fig. 1.

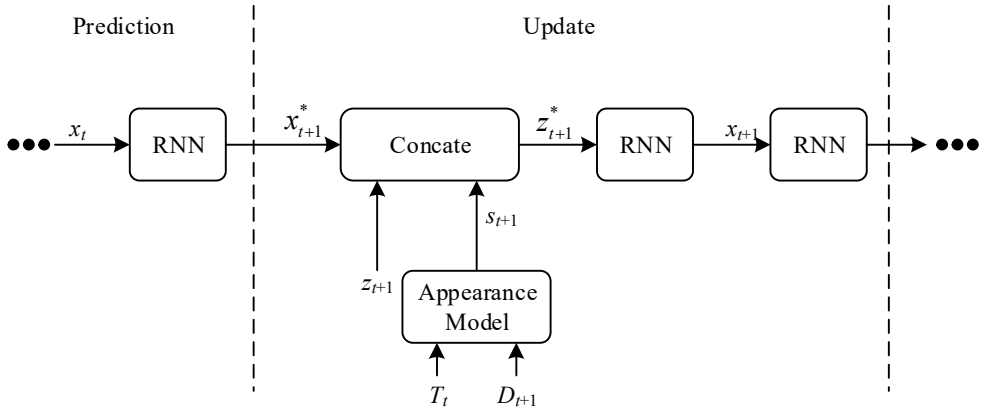


Fig. 1. Multi-feature RNN online tracking framework

2.2.2. Multi-target tracking model. In this section, the location information of the target is first used to predict the possible states of the target by RNN, after that, the appearance features of the target are extracted by the dual VGG model and the appearance similarity between the target and the detection response is calculated, and then combined with the metric of the location information to construct the cost matrix, and then the Hungarian algorithm is used to solve the matching relationship between the target and the detection response, so as to update the predicted value of the network to get the final state of the target [18]. The network structure of the multi-feature based RNN online multi-target tracking algorithm is shown in Figure 2.

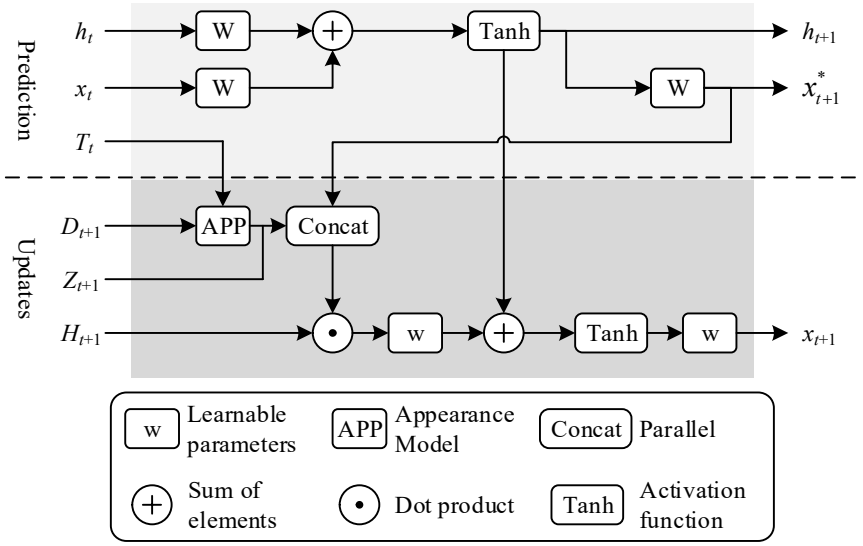


Fig. 2. A multi-feature RNN multi-target tracking network structure

There are six inputs to the network, which are the hidden layer state of the RNN $h_1 \in \square^n$, the bounding box $x_t \in \square^{ND}$ and image T_t of the target at moment t, the bounding box $z_{t+1} \in \square^{M+D}$ and

image D_{t+1} of the detection response at moment $t+1$, and the correlation matrix H_{t+1}. n denotes the size of the hidden layer, N denotes the maximum number of targets per frame, M denotes the maximum number of detection responses per frame, and D denotes the dimension of state vector dimension of the state vector. In this chapter, the Hungarian algorithm is used to solve the data association problem, H denotes the Hungarian association matrix of size $N\times(M+1)$, which is used to represent the matching relationship between the predicted state and the detected response of the network between two neighboring frames.

The multi-feature based RNN online tracking model is divided into two parts: prediction and update:

1) Prediction: the RNN network is utilized to learn the dynamic model of the target and predict the state of the target at the next moment. Knowing the position of the target x_t and the state of the hidden layer h_i at moment t, the hidden layer state h_{t+1} and the predicted target state x_{i+1}^* of the network at the next moment can be calculated by the RNN network.

2) Update: The target image T_t and the predicted target position x_{t+1}^* at time t are known, as well as the latest observation z_{i+1} and the image of the detection response D_{i+1} at time $t+1$. In the update phase, the first step is to solve the data correlation problem, and we need to determine the matching relationship between the predicted target state at time t and the detection response at time $t+1$, so as to update the predicted values accordingly. The judgment of the matching relationship is to describe the target as comprehensively as possible through the feature expression, and then select the detection response that most closely matches the target features. In this paper, we choose the Hungarian algorithm to solve the data association problem, construct the cost matrix C, and convert the association problem into solving the objective function: how to allocate the target and the detection response so as to minimize the association cost.

Assuming N targets at moment t and M detected responses at moment $t+1$, the trained dual VGG appearance model is used to compute the appearance similarity $\mathrm{S}=\left\{s_{ij}\in[0,1]\,|\,i=1,\ldots,N,j=1,\ldots,M\right\}$ between the i th target at moment t and the j th detected response at moment $t+1$, and the Euclidean distance $d_{ij}=\left\|x_{i+1}^{r}-z_{i+1}^{j}\right\|_2$ between the network prediction x_{i+1}^* of the i th target and the j th detected response z_{t+1} is also computed, $d\in\square^{N\times N}$. A cost matrix C is constructed with a size of $N\times(M+1)$:

$$C=\begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1m} & \sigma \\ c_{21} & c_{22} & \ldots & c_{2m} & \sigma \\ \ldots & \ldots & \ldots & \ldots & \sigma \\ c_{n1} & c_{n2} & \ldots & c_{nm} & \sigma \end{bmatrix} \tag{12}$$

where σ is a human-set cost threshold, and if any $c_{ij}>\sigma$ within row i, the i th target is judged to be missed. In the cost matrix C:

$$c_{ij}=\mathrm{f}\left(d_{ij}\right)-\ln\left(s_{ij}\right) \tag{13}$$

Among them, the larger the value of $s_{ij}\in[0,1]$, s_{ij} indicates that the higher the appearance similarity between target i and detection response j, the smaller the value of $d_{ij}\in\square^{N\times M}$, d_{in} indicates that the location of the predicted target i and the spatial location of the detection response j is more similar to each other, and the greater the possibility of belonging to the same target. $f(\cdot)$ for the mapping function, the text selects the most commonly used sigmoid function to map the values in the real number domain to between $[0,1]$, defined as follows:

$$f(x)=\frac{1}{1+e^{-x}} \tag{14}$$

For Eq. (13), the higher the appearance similarity s_{ij} between the target and the detection response, and the smaller the position distance d_{ij}, the smaller the association cost c_{ij}, and the more likely to belong to the same target. With the cost matrix C, the solution matrix H is obtained by the Hungarian algorithm with sizes $N \times (M+1)$, $h_{ij} \in \{0,1\}$. If $h_{ij}=1$ and $i \le N$ and $j \le M$ then it means that the j th detection response is assigned to the i th target. If $h_{ij}=1$ and $j=M+1$ then it means that the target misses detection, otherwise $h_{ij}=0$.

The loss function of the multi-feature based RNN network is defined as follows:

$$L\left(\tilde{x}, x^{*}, x\right)=\frac{\lambda}{ND}\sum\left\|x^{*}-\tilde{x}\right\|^{2}+\frac{\kappa}{ND}\sum\left\|x-\tilde{x}\right\|^{2} \tag{15}$$

The error between the network's predicted and updated values and the true value is measured by the l_2-parameter, and the model is optimized by minimizing the loss function.

2.3. Improved RNN-based multi-intelligent body path planning

2.3.1. Coding scheme. A real number coding scheme is used, where each connection weight is directly represented by a real number and the entire neural network weight distribution is represented by a set of real numbers. Each gene of a chromosome is represented by a real number in the range [-2, 2], and the coding length of an individual is 40, which represents each of the 40 connection weights of the neural network. When connecting the strings corresponding to each weight together, a preferred order of connection is to place the strings corresponding to the connection weights connected to the same hidden node together. This is because the hidden nodes play the role of feature extraction in the neural network and they have stronger connections to each other. This connection is not well represented in the algorithm if the strings corresponding to connection rights connected to the same hidden node are separated.

2.3.2. Adaptation assessment. In intelligent body experiments, it is common to determine the performance of an intelligent body using a fitness function for a specific behavior, which narrows the space searched during the evolution of the intelligent body and allows the intelligent body controller to converge faster. Here we utilize an adaptive function that contains as few variables and constraints as possible as the evaluation function:

$$F(p)=F_{m1}(p)\oplus F_{m2}(p) \tag{16}$$

$$F_{m1}(p)=D-d \tag{17}$$

$$F_{m2}(p)=D+\frac{C}{Rn} \tag{18}$$

All individuals in the population of intelligences in each generation run for a number of steps from the starting point, and stop running if the intelligences reach the target point or collide with an obstacle or exceed the maximum number of running steps. As shown in Equation (16), if the intelligent body does not reach the target point, the value of the fitness function will be calculated from $F_{m1}(p)$, whose value is obtained from Equation (17), D is the maximum convergence reward that the intelligent body may receive (the reward it receives when it reaches the target point), and d is the distance between the final position of the intelligent body and the target point. If the smart body reaches the target point, the value of the fitness function will be calculated by $F_{m2}(P)$, whose value is obtained by equation (18), D is the maximum convergence reward, Rn is the number of steps run by the smart body to reach the target point, C is a constant, and C/Rn is the reward according to the number of steps run, and the lesser the number of steps run the greater the reward. In this way the intelligence that reaches the goal with the shortest path gets the largest reward value, and all the

intelligences that reach the goal point have a fitness value greater than the fitness value of the intelligences that do not reach the goal point.

The individual fitness in each generation of the population is stretched using the simulated annealing idea, so that the probability of offspring produced by individuals with similar fitness is close at the early stage of evolution, and the difference in fitness of individuals with similar fitness is enlarged at the late stage of evolution to make the advantage of the excellent individuals more obvious, so as to avoid the problems of premature maturity of the algorithm and the slower rate of evolution at the late stage of evolution. The fitness stretching method is shown in Equation (19):

$$f_i = \frac{\exp\left(f_i / T\right)}{\sum_{i}^{M} \exp\left(f_i / T\right)} \tag{19}$$

In Eq. $T = T_0\left(0.99^{g-1}\right)$, f_i is the fitness of the i rd individual, M is the population size, g is the number of genetic generations, T is the temperature parameter, and T_0 is the initial temperature.

2.3.3. Variational operators. The performance of a neural network is jointly determined by multiple weight parameters, and a small amount of weight variation may make the change in fitness insignificant, thus preventing useful evolutionary information from being transmitted to the next generation of the population. Therefore, in order to ensure that the mutation operation of the evolutionary algorithm has a certain magnitude and can be evenly distributed over the entire network structure, we use a combination of Gaussian and Cauchy mutation to perform the mutation operation on the connection weights, with the Gaussian mutation being able to produce a larger magnitude of mutation near the origin, and the Cauchy mutation being able to ensure a certain magnitude of mutation in the two flanking parts. Thus, it can accelerate the convergence speed of the evolutionary algorithm, and can jump out of the local minimal region as soon as possible. The connection right mutation operation is shown in Equation (20):

$$\begin{aligned} \sigma_i' &= \sigma_i^* \exp\left(c_1 N(0,1) + c_2 C(0,1)\right) \\ w_i' &= w_i + \sigma_t'^* C(0,1) \end{aligned} \tag{20}$$

where σ_i is the step of variation of the weights variable (decision variable), $N(0,1)$ is a standard normally distributed random variable, and $C(0,1)$ is a Cauchy random variable of $t=1$. c_1 and c_2 are selection factors, and in the process of mutating the decision variable, a Gaussian distribution plays a role near the origin and a Cauchy distribution plays a role in the two flanking parts. The decision variable and the random numbers generated by the Cauchy distribution are then used to adjust the target variable.

2.3.4. Crossing probabilities and variance probabilities. The crossover probability P_c and the mutation probability P_m are two very critical parameters of the evolutionary algorithm that directly affect the behavior and performance. The larger the values of P_c and P_m, the faster the search speed. However, taking too large a value tends to make the structure of the highly adapted individual quickly destroyed, turning the evolutionary algorithm into a random search algorithm. Therefore, ideally, the values of P_c and P_m should be able to be adjusted adaptively according to the diversity index of the population and the fitness value of individuals. The improved formulas of P_c and P_m are shown in equations (21) and (22):

$$P_c = \begin{cases} P_{c1} \dfrac{\left(P_{c1} - P_{c2}\right)\left(f' - f_{avg}\right)}{f_{\max} - f_{avg}} \dfrac{1}{1.0 + \exp\left(-k_c \Omega\right)} & f' \geq f_{avg} \\ P_{c1} & f' < f_{avg} \end{cases} \tag{21}$$

$$P_m = \begin{cases} P_{m1} - \dfrac{(P_{m1} - P_{m2})(f - f_{avg})}{f_{\max} - f_{avg}} \dfrac{1}{1.0 + \exp(-k_m \Omega)} & f \geq f_{avg} \\ P_{m1} & f < f_{avg} \end{cases} \tag{22}$$

Parameter f' in equations (21) and (22) is the fitness value of the larger of the two individuals to be crossed, f is the fitness value of the individual to be mutated, $f_{\max}$ is the largest fitness value in the population, and f_{avg} is the average fitness value of the population in each generation. Ω is the index of population diversity, P_{c1}, P_{c2}, P_{m1}, P_{m2}, k_c, k_m are constants, which can be determined according to the actual situation. From equations (21) and (22), it can be seen that in the population of the same generation, the greater the fitness, the smaller the probability of crossover and the probability of variation of individuals. In different populations, the crossover probability of individuals in populations with larger diversity indexes is also smaller, with the minimum possible values of P_{c2} and P_{m2}, respectively, while the crossover probability and mutation probability of individuals with smaller adaptations are always taken as larger values of P_{cl} and P_{m1}, respectively, which ensures that the excellent individuals at the early stage of evolution can also crossover and mutate according to a certain probability to avoid the phenomenon of precocious maturation. In the late stage of evolution, due to the decrease of population diversity index, the P_c and P_m increase, which accelerates the convergence speed of the algorithm.

3. Simulation experiment analysis

3.1. Trajectory tracking effect analysis

The experiments use the multi-objective detection and path planning model based on RNN optimization in this paper in robot path planning, and in order to verify the effectiveness of the recurrent neural network proposed in this paper in solving the trajectory tracking problem of the intelligent body, this paper carries out the numerical simulation verification of the diagonal trajectory tracking by Matlab.

In the experiment, the trajectory equations of the diagonal line are as follows:

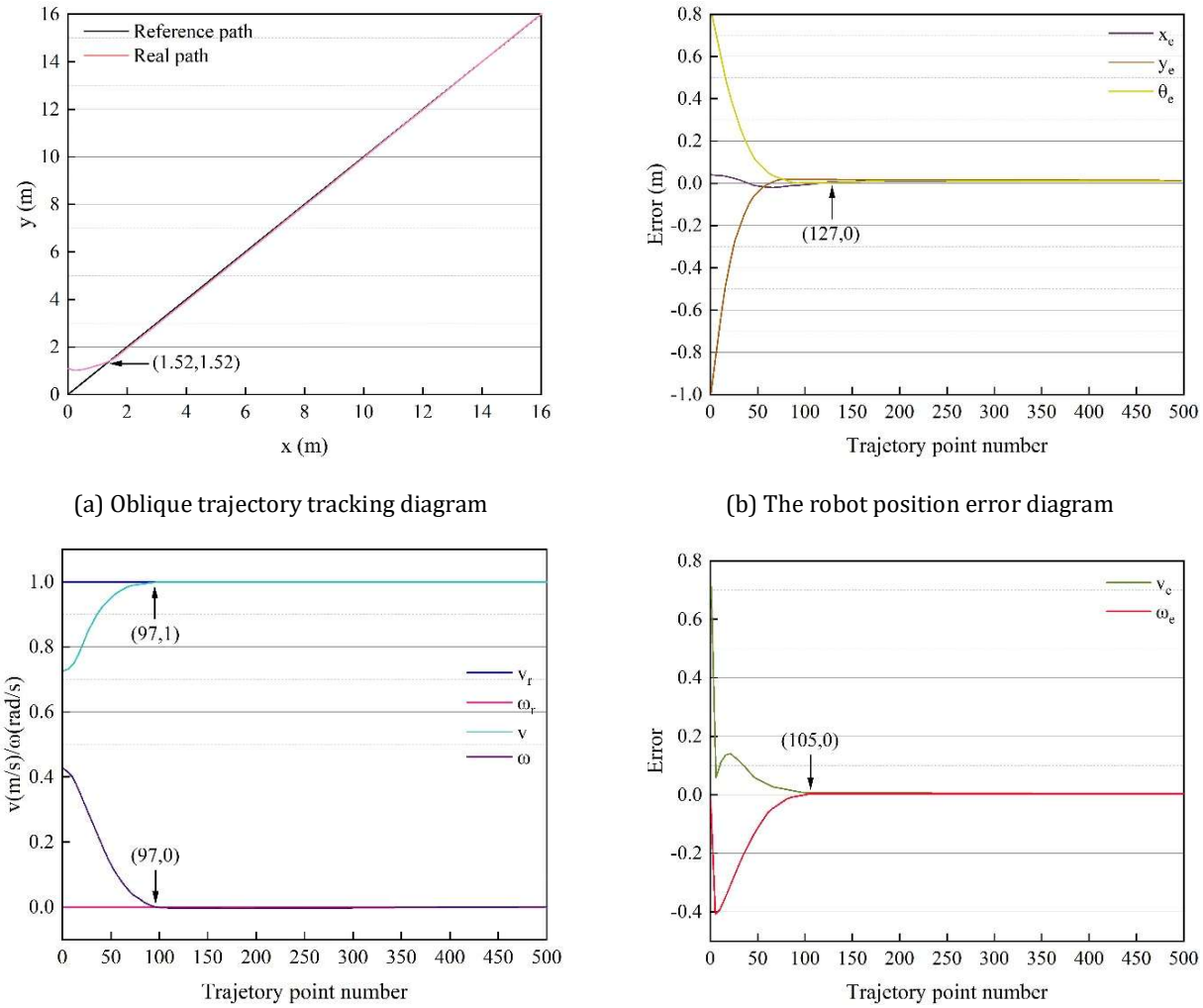
$$\begin{cases} x_r = t \\ y_r = t \end{cases} \tag{23}$$

Its reference angle θ_r=π/4, reference linear velocity v_r=1m/s, and reference angular velocity ω_r=0rad/s. The control and actual domains of the model predictive control are N_c=50and N_p=49, respectively, the sampling time T=0.05, the associated matrix systems are $Q = I_{5N_c \times 5N_c}$ and $R = 0.01 I_{2N_c \times 2N_c}$, respectively, the exponential variable parameter $g(t)$=t^1+1, and the initial bit position of the mobile robot is x_0=[0, 1, 0], and the number of reference trajectory points is 500. The specific diagonal trajectory tracking results are shown in 3.

Figure 3(a) represents the variation of the actual motion curve (pink line) and the reference trajectory curve (black line) of the robot. Fig. 3(b) represents the error variation curve of the robot's actual position with respect to the reference position, the purple line indicates the error x_e in the x-axis, the brown line indicates the error y_e in the y-axis, and the yellow line indicates the angular error θ_e. Fig. 3(c) represents the variations of the mobile robot's actual with respect to the reference velocity and angular velocity, the blue solid line and the pink solid line refer to the reference velocity v_r and the reference angular velocity ω_r, respectively, and the cyan and purple lines refer to the actual output velocity v and actual output angular velocity ω. Fig. 3(d) represents the variation of the error of the mobile robot's actual versus reference velocity and angular velocity, with the green line indicating the velocity error v_e and the red line indicating the angular velocity error ω_e.

From Figs. 3(a) and 3(b), it can be seen that due to the inconsistency between the starting point of

the robot and the starting point of the reference curve, after the tracking starts, the actual position of the robot starts to approach the reference position gradually, and the position error is gradually reduced from the maximum at the beginning, and when it comes to the 100th reference point, the error between the actual trajectory and the reference position is already lower than 10^{-1}m, and after the 127th reference tracking point, the actual After the 127th reference tracking point, the actual trajectory and the reference trajectory basically coincide completely, realizing that the robot's actual trajectory based on the RNN trajectory tracking algorithm tracks the reference trajectory quickly and maintains the high-precision diagonal trajectory tracking with 0 positional error in the subsequent tracking. From Figures 3(c) and 3(d), it can be seen that the robot's actual velocity and acceleration outputs are adjusted to the robot at the beginning for the large positional error to approximate the reference trajectory, and when the actual linear and angular velocities are also guaranteed to be the same as those of the reference after the trajectories are overlapped, with an error infinitely close to 0, realizing the high-precision tracking of the slanting trajectory of the mobile robot.



(a) Oblique trajectory tracking diagram

(b) The robot position error diagram

(c) Robot v_r and ω_r change diagram

(d) Robot v_e and ω_e diagram

Fig. 3. Robot oblique trajectory tracking experiment

3.2. Analysis of the effects of route planning

3.2.1. Experimental performance comparison. This paper selects 3 indicators to analyze the performance of the improved RNN multi-agent path planning algorithm and 5 common path planning algorithms (A3C, LSTM+A3C, R-A3C, artificial potential field method APF, DQN), 100 times accuracy

represents the end of the training of these 6 algorithms, 100 obstacle avoidance attempts, the number of times to reach the destination is counted, if the destination is not reached for more than 1 minute, it is considered that the destination cannot be reached, because it takes too long. The average time to reach the destination represents, out of these 100 obstacle avoidance attempts, counting the number of times in which the destination was reached and how long it took to reach the destination, and finally calculating the average time to reach the destination each time. The collision probability represents counting the number of times a collision with an obstacle occurs out of the 100 obstacle avoidance attempts. The results of the experimental data for algorithm performance comparison are shown in Table 1.

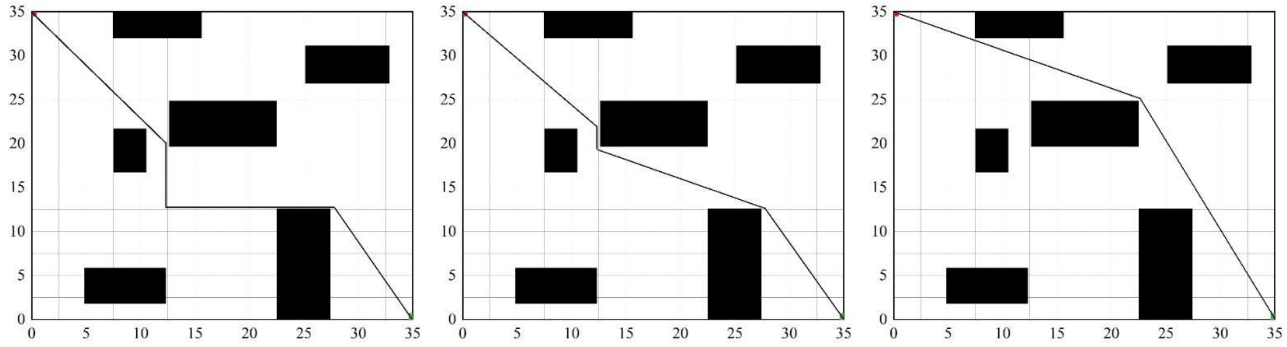
As can be seen in Table 1, firstly DQN has the worst performance among the six algorithms, with an accuracy rate of only 45%, mainly due to the limitations of the DQN algorithm, which is unable to deal with a large number of dynamic obstacles. Secondly A3C has a more inadequate performance with an accuracy rate of 87%, mainly due to the fact that when faced with a large number of obstacles, the A3C algorithm is unable to utilize the information of the surrounding obstacles, which makes each time to find the destination spends a long time, and again 10 times is greater than 1 minute which leads to the statistic as a failure. Then the traditional artificial potential field method in the case of a particularly large number of obstacles, it will be in order to arrive at the destination directly and obstacles collision, because the artificial potential field method is each step to choose the direction of the smallest gradient to move forward, it can not guarantee that the direction of the smallest gradient there is no obstacle. Finally the better performance is to use LSTM to deal with the surrounding obstacles, and then after the A3C algorithm, LSTM can deal with the surrounding obstacle information, so that the A3C algorithm to use the information of the surrounding obstacles, so that the algorithm can choose the optimal path faster. The performance of the optimized RNN algorithm proposed in this paper has the best effect in these five algorithms, with 100% accuracy, the shortest average arrival time (20.02s) and 0% probability of fitting, mainly because RNN can be based on the obstacles away from the unmanned vehicle's distance weights, so that the last vector obtained through RNN contains more obstacles away from the robot than the vector processed by LSTM, making the path planning algorithm can avoid obstacles more effectively.

Table 1. Algorithm performance comparison results

Algorithms	100-time accuracy	Arrival average time	Collision probability
A3C	87%	58.62s	9%
LSTM+A3C	98%	27.59s	3%
R-A3C	97%	25.75s	4%
APF	94%	44.62s	11%
DQN	45%	59.48s	43%
Ours	100%	20.02s	0%

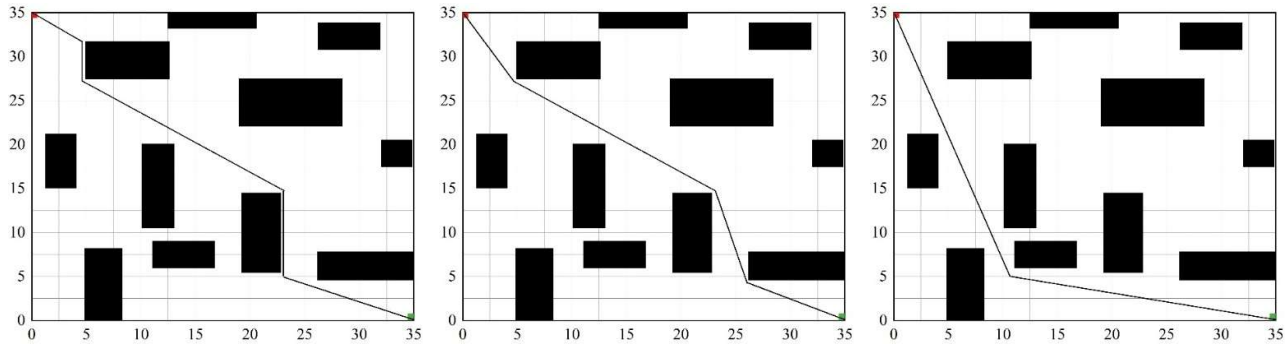
3.2.2. Comparison of experimental results:

1) Multi-objective path planning results. In order to make the experimental results more convincing, this paper adopts three kinds of raster maps with the size of 30 × 30 but with different obstacle densities to conduct simulation experiments. On the raster maps with different degrees of complexity, the simulation comparison experiments of traditional JPS algorithm, traditional RNN algorithm and this paper's multi-objective optimized RNN algorithm are carried out, such as the three algorithm comparisons under the scenes of maps 1~3 as shown in Fig. 4~Fig. 6.



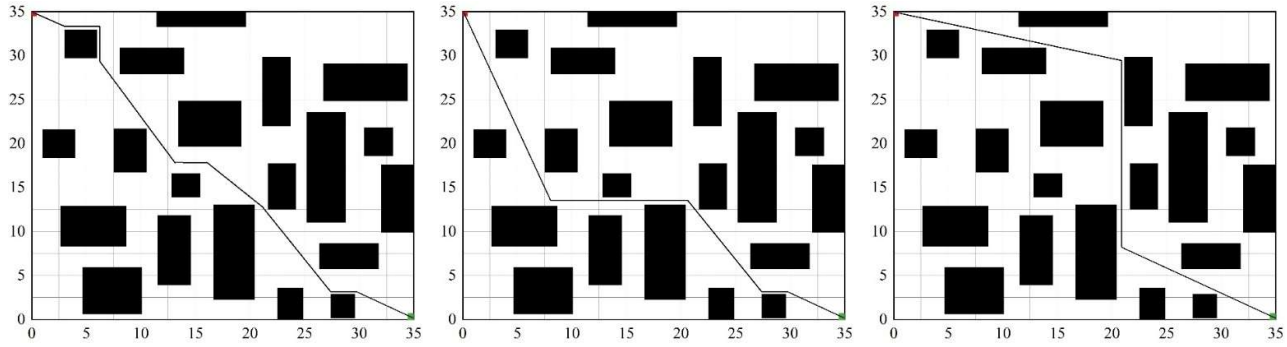
(a) JPS algorithm (b) RNN algorithm (c) Multi-objective optimized RNN algorithm

Fig. 4. Comparison of four algorithms in map 1 environment



(a) JPS algorithm (b) RNN algorithm (c) Multi-objective optimized RNN algorithm

Fig. 5. Comparison of four algorithms in map 2 environment



(a) JPS algorithm (b) RNN algorithm (c) Multi-objective optimized RNN algorithm

Fig. 6. Comparison of four algorithms in map 3 environment

The various simulation data of the three different algorithms in different grid environments are shown in Table 2. From the simulation results and experimental data, it can be seen that the traditional JPS algorithm and the traditional RNN algorithm plan paths with many inflection points, not smooth enough, and lack of security. The paths planned by the multi-objective optimized RNN algorithm in this paper are significantly better than the first two algorithms in terms of comprehensive performance, indicating that the multi-objective optimized RNN algorithm has stronger global optimization capability. The multi-objective optimized RNN algorithm is optimized for the RNN algorithm, which can significantly reduce the path length and further improve the path smoothness by

introducing the inverse jump-point elimination method. In this paper, the multi-objective optimized RNN algorithm plans the multi-objective path smoothness in all three environments with the largest value among the three algorithms (96.46, 96.84, 94.37), which indicates that in simple environments, the multi-objective optimized RNN algorithm has an obvious optimization effect. Meanwhile, in complex environments, the multi-objective optimized RNN algorithm still has a good comprehensive performance optimization effect.

Table 2. Comparison of experimental results in different environments

Map number	Path length			Smoothness		
	JPS	RNN	Ours	JPS	RNN	Ours
1	54.52	50.15	51.23	68.97	77.48	96.46
2	55.84	53.06	50.03	67.25	80.46	96.84
3	55.87	54.89	56.48	66.34	82.59	94.37

2) Multi-intelligent body path planning results. The Wd-SIPP algorithm for multi-intelligent body path planning is compared with the RNN algorithm after multi-objective optimization in this paper for robot collaborative path planning, and the test comparison results of the two algorithms in the 30 × 83 storage maps and the 100 × 100 storage maps are shown in Table 3 and Table 4, respectively. It can be seen that the path scheme obtained by the multi-objective optimized RNN algorithm in this paper has a greater advantage over the Wd-SIPP algorithm in terms of path length and total path cost. In the 30 × 83 warehousing map, the total path length can be shortened by 13.00% and 10.77% for a single group of 3 and 5 collaborative robots, respectively. And the total cost of the path can be shortened by 9.71% and 11.52%, respectively. In a 100 × 100 warehouse map, the total path length can be shortened by 10.32% and 11.67% for the cases of 3 and 5 collaborative robots in a single group, respectively. And the total cost of the path can be shortened by 7.34% and 12.09%, respectively. The test results in two different sizes of maps show that the path scheme obtained through this paper's algorithm can obtain a path scheme with smaller total path length and total path cost compared to the Wd-SIPP algorithm. This suggests that the solution obtained through the method in this paper can reduce the energy consumption of robot movement, and thus higher system benefits can be obtained. However, the maximum completion time is larger than that of the Wd-SIPP algorithm because the collaboration for the stringing process requires the tractor robot to wait for the arrival of all the acquisition machines in the stringing area for stringing, and in addition, solving for the stringing area that is favorable for robots' stringing also requires a certain amount of time for searching on the map, and thus the solution time is also longer than that of the Wd-SIPP algorithm. Overall, the algorithm in this paper saves the energy expenditure of the system by sacrificing some solution time and maximum completion time to obtain a solution with lower energy consumption.

Table 3. Comparison of path solution quality obtained by algorithms on 30×83 warehouse map

Collaborative robots number	N_{group}	Makespan/step		Flowtime/step		Length/step		Runtime/s	
		Wd-SIPP	Ours	Wd-SIPP	Ours	Wd-SIPP	Ours	Wd-SIPP	Ours
3	3	122.2	140.8	808.4	732.5	635.2	550.2	0.28	1.32
	6	150.4	156.9	1762	1586.4	1372.6	1203.4	0.83	4.72
	9	145.6	165.5	2548.3	2301.5	1986.4	1701.7	1.51	7.63
	12	144.2	168.7	3352.3	3035.4	2620.4	2273.5	2.16	10.85
	15	153.8	186.2	4496.7	4052.9	3432.5	3012.6	3.89	24.11
5	2	130.4	156.2	920.6	836.5	752	659.5	0.35	2.13
	4	144.5	172.3	1887.5	1715.6	1542	1372.4	0.88	5.14

	6	145.6	170.3	2841.6	2321.5	2263	1992.3	1.67	8.52
	8	162.3	191.6	3897.8	3506.8	3120.5	2799.3	2.98	26.32
	10	154.8	193.4	4993.2	4485.4	3935.8	3538.9	4.55	21.02

Table 4. Comparison of path solution quality obtained by algorithms on 100×100 warehouse map

Collaborative robots number	N_{group}	Makespan/step		Flowtime/step		Length/step		Runtime/s	
		Wd-SIPP	Ours	Wd-SIPP	Ours	Wd-SIPP	Ours	Wd-SIPP	Ours
3	3	198.8	220.6	1268.5	1192.8	978.6	887.3	0.97	2.73
	6	224.8	230.9	2611.6	2367	1988	1758.3	2.01	7.81
	9	229.6	250.4	3824.4	3532.8	3002.5	2714.6	3.25	16.14
	12	246.3	260.4	5443.6	4961.1	4232.2	3757.5	5.5	26.11
	15	246.7	276.1	6666.6	6307.2	5207.6	4701.2	6.04	52.4
5	2	201.2	230.7	1452.5	1317.7	1203.3	1074.1	1.25	6.08
	4	214.2	235	2885.9	2474.4	2336.4	2043.8	3.24	12.23
	6	235.5	259.3	4581.4	4093.9	3718.5	3350.9	6.46	21.32
	8	236.9	266.7	6226.3	5466.7	5058.4	4415.2	11.14	28.07
	10	250.4	281.5	8123.2	7102.5	6400.9	5687	16.56	40.4

4. Conclusion

In this paper, target detection, target tracking and path planning of multiple intelligences are realized by optimizing recurrent neural network algorithm, and the multi-target detection and path planning model based on RNN optimization is constructed. Through simulation experiments, the trajectory tracking effect and path planning effect of the algorithm model in this paper are verified.

1) The error between the actual trajectory of robot trajectory tracking and the reference position is gradually reduced, and after the 127th reference tracking point, it is basically completely overlapped. The actual speed and acceleration errors of the robot are infinitely close to 0. The algorithm in this paper realizes the high-precision tracking of the oblique trajectory of the mobile robot.

2) In the multi-objective planning simulation experiment, the accuracy of this paper's algorithm is 100%, the average arrival time is 20.02s, and the collision probability is 0%, which is the best performance among all algorithms. The multi-objective optimized RNN algorithm in this paper has the best planning path smoothing effect and the highest security in the three environments.

3) In 30 × 83 and 100 × 100 storage maps, the total path length can be shortened by 13.00% and 10.77%, 10.32% and 11.67% for the number of 3 and 5 collaborative robots in a single group, respectively. On the total cost of the path, it can be shortened by 9.71% and 11.52%, 7.34% and 12.09%, respectively.

References

[1] Dalmasso, M., Garrell, A., Domínguez, J. E., Jiménez, P., & Sanfeliu, A. (2021, May). Human-robot collaborative multi-agent path planning using monte carlo tree search and social reward sources. In 2021 IEEE international conference on robotics and automation (ICRA) (pp. 10133-10138). IEEE.

[2] Zhang, T., Zhu, K., Zheng, S., Niyato, D., & Luong, N. C. (2022). Trajectory design and power control for joint radar and communication enabled multi-UAV cooperative detection systems. IEEE Transactions on Communications, 71(1), 158-172.

[3] Wang, J., Hong, Y., Wang, J., Xu, J., Tang, Y., Han, Q. L., & Kurths, J. (2022). Cooperative and competitive multi-agent systems: From optimization to games. IEEE/CAA Journal of Automatica Sinica, 9(5), 763-783.

[4] Xiao, Z., Li, P., Liu, C., Gao, H., & Wang, X. (2024). MACNS: A generic graph neural network integrated deep

reinforcement learning based multi-agent collaborative navigation system for dynamic trajectory planning. Information Fusion, 105, 102250.

[5] Wei, X., Cui, W., Huang, X., Yang, L., Tao, Z., & Wang, B. (2023). Graph MADDPG with RNN for multiagent cooperative environment. Frontiers in Neurorobotics, 17, 1185169.

[6] Liu, W., Teng, F., Fang, X., Liang, Y., & Zhang, S. (2023). An rnn-based performance identification model for multi-agent containment control systems. Mathematics, 11(12), 2760.

[7] Torreno, A., Onaindia, E., Komenda, A., & Štolba, M. (2017). Cooperative multi-agent planning: A survey. ACM Computing Surveys (CSUR), 50(6), 1-32.

[8] Biswas, S., Anavatti, S. G., & Garratt, M. A. (2019). A time-efficient co-operative path planning model combined with task assignment for multi-agent systems. Robotics, 8(2), 35.

[9] Yu, C., Yang, X., Gao, J., Yang, H., Wang, Y., & Wu, Y. (2022, October). Learning efficient multi-agent cooperative visual exploration. In European Conference on Computer Vision (pp. 497-515). Cham: Springer Nature Switzerland.

[10] Van Nguyen, H., Vo, B. N., Vo, B. T., Rezatofighi, H., & Ranasinghe, D. C. (2024). Multi-objective multi-agent planning for discovering and tracking multiple mobile objects. IEEE Transactions on Signal Processing.

[11] Greshler, N., Gordon, O., Salzman, O., & Shimkin, N. (2021, November). Cooperative multi-agent path finding: Beyond path planning and collision avoidance. In 2021 International symposium on multi-robot and multi-agent systems (MRS) (pp. 20-28). IEEE.

[12] Wang, X., & Fang, X. (2023). A multi-agent reinforcement learning algorithm with the action preference selection strategy for massive target cooperative search mission planning. Expert Systems with Applications, 231, 120643.

[13] Zhou, T., Shi, T., Gao, H., & Rao, W. (2024). Learning to Optimize State Estimation in Multi-agent Reinforcement Learning-based Collaborative Detection. IEEE Transactions on Mobile Computing.

[14] Zhang, R., Wang, J., Ge, J., & Huang, Q. (2024). Multi-agent Cooperative Search Learning with Intermittent Communication. IEEE Intelligent Systems.

[15] Wei, X., Yang, L., Cao, G., Lu, T., & Wang, B. (2020). Recurrent MADDPG for object detection and assignment in combat tasks. Ieee Access, 8, 163334-163343.

[16] Saurabh Balkrishna Tandale,Prashant Sharma,Vasileios Polydoras & Marcus Stoffel. (2024). Recurrent neural networks as a physics-based self-learning solver to satisfy plane stress viscoplasticity undergoing isotropic damage. Mechanics Research Communications104347-104347.

[17] Tomohiro Shiraishi,Daiki Miwa,Vo Nguyen Le Duy & Ichiro Takeuchi. (2024). Selective Inference for Change Point Detection by Recurrent Neural Network. Neural computation31-33.

[18] Juan Pablo Vásconez,Elias Schotborgh,Ingrid Nicole Vásconez,Viviana Moya,Andrea Pilco,Oswaldo Menéndez... & Leonardo Guevara. (2024). Smart Delivery Assignment through Machine Learning and the Hungarian Algorithm. Smart Cities(3),1109-.