International Journal for Housing Science and Its Applications
Publish August 4, 2025. Volume 46, Issue 3 Pages 3428-3442

https://doi.org/10.70517/ijhsa463289

Optimization of Multi-Target Detection and Path Planning for Multi-Intelligent Body Collaborative Autonomous Systems Combined with Deep Q Learning

Yuxuan Li[1,*]
[1] School of Information, Shanxi University of Finance and Economics, Taiyuan, Shanxi, 030006, China
Corresponding authors: (e-mail: 18335548302@139.com).

Abstract Deep reinforcement learning to help multi-intelligence to achieve multi-target detection of the environment and optimally plan paths. To this end, the article proposes an improved YOLOv7 method for image multi-target detection. Null convolution with mean pooling is introduced in the SPPCSPC module to shallow information in the image, and a lightweight SimAM attention mechanism is introduced in the head network to focus on the region of interest. And the hybrid edge regression loss function is proposed by combining NWDloss loss with CIOU loss. Meanwhile, the article further utilizes deep Q-learning for path planning on the basis of multi-target detection algorithm, and proposes two mechanisms to improve the DQN algorithm based on adaptive exploration strategy and based on changing the objective function for the two problems of DQN algorithm in path planning. In terms of datasets, compared with the YOLOv7 algorithm, the proposed algorithm improves the AP@0.5 of all detection categories by 3.2 percentage points, and is more than 3.1 percentage points higher than the AP@0.5 of the YOLOv7 algorithm, effectively realizing multi-target detection. The results of path planning simulation experiments show that the algorithm in this paper is able to plan good AGV paths, and the stability and convergence speed of the algorithm have been improved.

Index Terms deep Q-learning, YOLOv7, target detection, path planning, multiple intelligences

I. Introduction

In multi-robot autonomous systems such as smart warehousing and robotic sorting centers, it is crucial to plan efficient and collision-free paths for the robots, and Multi-Agent Path Planning (MAPF) generates paths for a group of intelligences from a start position to a goal position [1]. Current MAPF methods, can be categorized into centralized and distributed. Centralized planning methods have a central controller with global information to plan paths for all intelligences [2]. However, as the number of intelligences grows, the system needs to consume more and more computational resources and time, and the team size is difficult to scale [3], [4].

With the rapid development of 5G technology and the increasing arithmetic power of computing devices represented by GPUs, the importance of distributed computing has become increasingly prominent [5]. 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 heterogeneous viewpoint information to enhance the algorithm performance and robustness [6]-[9]. On the other hand, the traditional cloud computing model presents drawbacks such as communication bandwidth bottleneck and security [10], [11]. The study of multi-intelligence collaborative target detection, recognition and tracking methods based on intelligent distributed computing can be a good solution to the problem of effective target detection and accurate planning under strong anti-jamming conditions [12], [13].

In the article, firstly, an improved image multi-target detection method for YOLOv7 is proposed to address the problems such as a large number of targets in an image, similar appearance, and easy to miss detection and misdetection when the background and the target are similar in color. Null convolution and mean pooling are introduced in the SPPCSPC module to expand the sensory field while preventing small targets from drowning in the background, while a lightweight SimAM attention mechanism is introduced in the head network to focus on the region of interest. And a hybrid edge regression loss function combining NWDloss loss and CIOU loss, which is more suitable for small target detection, is chosen to improve the accuracy of target detection at different scales in images. Then, a path planning model based on DQN algorithm is designed to address the problem of exploration and utilization in the path-finding process of the original DQN algorithm, and a mapping relationship between

feedback rewards and strategy parameters is designed to allow the intelligent body to reasonably utilize the existing experience or explore the new environment. Then, a change function is designed to change the value of the objective function to make the optimal Q value more easily recognized by the intelligent body. Finally, the effectiveness of the method in this paper is verified by simulation comparison experiments.

II. Improvement of image multi-target detection in YOLOv7

Considering the demand of real-time and accuracy of multi-intelligence, this paper takes YOLOv7 network as the base model for improvement. Aiming at the problems such as low contrast between target and background, and the target is easily submerged in the background information, the detector's attention to the infrared target is increased by introducing null convolution and mean pooling, and enhancing the sensory field. For the problems such as complex background and dense targets leading to false detection and missed detection, the lightweight 3D attention mechanism SimAM is chosen to enhance the region of interest and direct the network to focus on more important locations to improve the accuracy of target detection. Since the CIOU loss function of the original network does not reduce the sensitivity to the positional deviation of tiny objects. Therefore, this paper adopts the NWDloss loss function, which mainly measures the difference between tiny objects by introducing the Wasserstein distance in the original border regression loss.Taking into account the objects with different scales in the image, this paper proposes a hybrid border loss based on CIoUloss and NWDloss to improve the algorithm's ability to detect multi-targets in infrared images.

II. A.Spatial Pyramid Pooling Layer Combining Null Convolution and Mean Pooling

The SPPCSPC module in the original YOLOv7 network structure is composed of three different scales of maximum pooling in the spatial pyramid pooling layer 5×5, 9×9, and 13×13. Through the pooling operation of these three different scales, the spatial pyramid pooling layer can simultaneously extract the multi-scale feature information of the image, including the low-level features, the local high-level features, and the global features, so as to enhance the network's recognition of image capability of the network to recognize the image. At the same time, since the spatial pyramid pooling layer can eliminate the limitation on the size of the input image, it can adapt to image inputs of different sizes and improve the generalization ability of the network. For problems such as large size difference, this paper can effectively increase the sensory field and keep the output feature size unchanged by introducing the cavity convolution, so as to better capture the global information and local detail information in the image [14]. Therefore, in this paper, hollow convolution is used to replace the convolution operation of the SPPCSPC block, and the step size of the convolution module in DCBS2 in the module is set to 1 and the dilation rate is set to 2. The step size of the convolution module in DCBS3 is set to 1 and the dilation rate is set to 4, which can increase the receptive field by using different dilation rate of the hollow convolution operation on different convolutional layers, and improve the accuracy of target detection. In addition, the maximum pooling layer in the SPPCSPC module achieves the downscaling compression of the feature map by taking the maximum value of the feature points in the neighborhood, which well preserves the texture features but ignores the shallow background information. Aiming at the problem of low image contrast and small targets easily submerged in the background, this paper introduces mean pooling to enhance the infrared small target detection capability. Mean pooling averages the feature points in the domain, which retains the background information but blurs the image. Therefore, in this paper, by connecting two different scales of mean pooling modules in series on the maximum pooling branch, while retaining the advantages of the two pooling and make up for each other's shortcomings, so as to extract the image feature information more effectively, and improve the performance and accuracy of infrared multi-target detection. The improved module is named SPPCSPC-DAVG.The structure of SPPCSPC-DAVG is shown in Fig. 1.

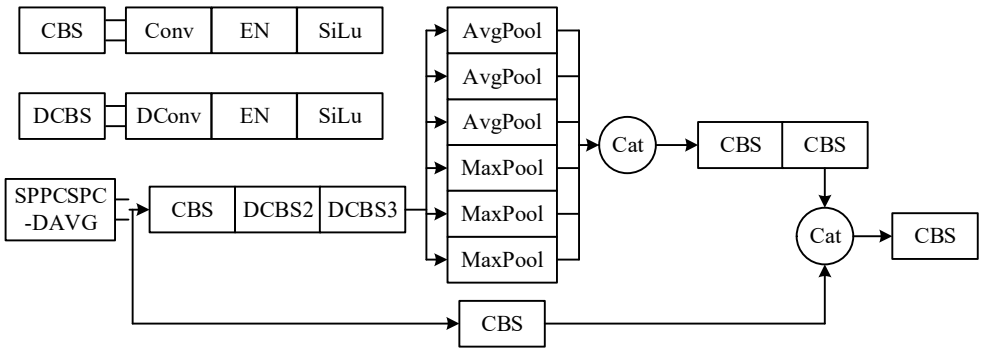

Figure 1: SPPCSPC-DAVG module

II. B.Embedding SimAM Attention Mechanisms

In order to better adapt to the scene, this paper introduces a lightweight 3D attention mechanism SimAM, the SimAM 3D attention mechanism is shown in Fig. 2.The SimAM attention mechanism focuses the attention on more important and target-related locations by learning the similarity between pixels in the feature map and focuses on targets of different sizes. Unlike previous channel or spatial attention mechanisms such as SE, ECN, CBAM, etc., the SimAM attention mechanism efficiently generates real 3D weights for each neuron without adding parameters to the original network. Instead of simply parallelizing or serializing channel and spatial attention, such weights infer 3D attention weights for feature mappings in the feature layer by considering spatial and channel dimensions.

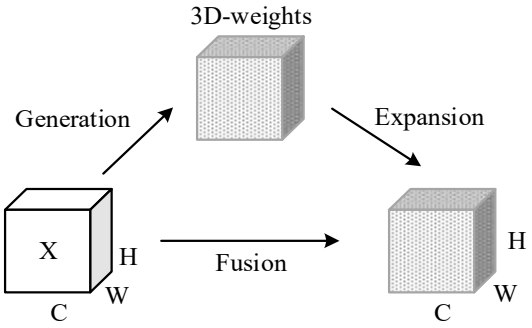

Figure 2: Three-dimensional attention mechanism of SimAM

In this paper, we add a lightweight 3D attention mechanism, SimAM, to the MPlayer, the downsampling layer of the original YOLOv7 complex.The improved module is named MPlayer-SAM.In this way, we can improve the real-time performance of the detector and reduce the memory footprint of the model.

The SimAM attention mechanism is based on neuroscience theory and determines the importance of each neuron by measuring the linear divisibility between a target neuron and other neurons. The energy function of each neuron is defined as:

$$e_t(w_t,b_t,y,x_i)=(y_t-\hat{t})^2+\frac{1}{M-1}\sum_{i=1}^{M-1}(y_0-x_i)^2 \tag{1}$$

where $\hat{t}=w_t t+b_t, x_i=w_t x_i+b_t$, and t and x_i are the input feature tensor $X\in\mathbb{R}^{C\times H\times W}$ the target neuron and other neurons for each channel. i is the neuron index of a particular channel and $M=H\times W$ is the total number of neurons on that channel. w_t and b_t are the weights and biases of the transformation. All values in Eq. are scalars. By minimizing this equation, the linear separability between the target neuron t and all other neurons x_i in the same channel is found.

For simplicity, the energy function after using binary labels for Eq. and adding a regular term can be transformed into:

$$\begin{aligned}e_t(w_t,b_t,y,x_i)=&\frac{1}{M-1}\sum_{i=1}^{M-1}(-1-(w_t x_i+b_t))^2\\&+(1-(w_t t+b_t))^2+\lambda w_t^2\end{aligned} \tag{2}$$

Since Eq. has a fast closed form solution with respect to w_t and b_t, the

The analytic solution can be found as:

$$w_t=-\frac{2(t-\mu_t)}{(t-\mu_t)^2+2\sigma_t^2+2\lambda} \tag{3}$$

$$b_t=-\frac{1}{2}(t+\mu_t)w_t \tag{4}$$

where: $\mu_t = \frac{1}{M-1}\sum_{i-1}^{M-1} x_i$ and $\sigma_t^2 = \frac{1}{M-1}\sum_{i}^{M-1}(x_i - \mu_t)^2$ represent the mean and variance, respectively, which are calculated based on all neurons in the channel except neuron t. The minimum energy can be found by bringing Eqs. w_t and b_t into Eq:

$$e_t^* = \frac{4(\sigma^2+\lambda)}{(t-\mu)^2+2\sigma^2+2\lambda} \tag{5}$$

Eq. shows that the lower the energy e_1^*, the more different the neuron t is from the surrounding neurons and the more important it is for visual processing. Therefore, the importance of each neuron can be obtained by e_t^*. The sigmoid function is then used to refine the features:

$$X = sigmoid(\frac{1}{E})\square\ X \tag{6}$$

where E categorizes all e_t^* of the channel and spatial dimensions. Adding a sigmoid both limits excessively large values in E and does not affect the relative importance of each neuron.

II. C.Hybrid loss function

In the target detection task, the purpose of border regression is to identify the target and to localize its position accurately. Therefore, choosing an appropriate loss function for edge regression is crucial to improve the accuracy of target localization. The original YOLOv7 model uses an IoU-based CIoU loss function for the border regression loss function. Although the CIoU loss function is more adaptive and accurate for rotating objects than the IoU loss function, it is very sensitive to the positional deviation of objects with small sizes, and its use in the anchor-based detector greatly reduces the detection performance, and is not suitable for the localization of small targets, such as joints, in the image. Smaller target localization [15]. Therefore, in this paper, we introduce the NWDloss loss function which is more suitable for small target detection. The NWDloss loss function is based on the Normalized Wasserstein Distance Loss Function, which can measure two non-overlapping distributions as compared to the IoU, and it is insensitive to the size when detecting small objects and can measure the similarity of the borders that do not overlap or contain each other, and is therefore particularly suitable for accurate localization of small targets.

Specifically, for two two-dimensional Gaussian distributions $\mu_1 = \mathrm{N}(m_1, \sum_1)$ and $\mu_2 = \mathrm{N}(m_2, \Sigma_2)$, the second-order Wasserstein distance between μ_1 and μ_2 is defined as:

$$W_2^2(\mu_1,\mu_2) = \| m_1 - m_2 \|_2^2 + \| \Sigma_1^{1/2} - \Sigma_2^{1/2} \|_F^2 \tag{7}$$

where $\|\cdot\|F^2$ is the Frobenius paradigm, and the Gaussian distributions N_a and N_b can be modeled according to the bounding box as $A = (cx_a, cy_a, w_a, h_a)$ and $B = (cx_b, cy_b, w_b, h_b)$, Eq. (7) can be further simplified as:

$$W_2^2(\mathrm{N}_a,\mathrm{N}_b) = \left\| \left(\left[cx_a, cy_a, \frac{w_a}{2}, \frac{h_a}{2} \right]^T \left[cx_\delta, cy_\delta, \frac{w_\delta}{2}, \frac{h_\delta}{2} \right] \right) \right\|_2^2 \tag{8}$$

However, $W_2^2(\mathrm{N}_a, \mathrm{N}_b)$ is a distance measure and cannot be used directly as a similarity measure. Therefore it needs to be normalized using its exponential form to obtain the Normalized Wasserstein Distance (NWD):

$$NWD(\mathrm{N}_a,\mathrm{N}_b) = \exp\left(-\frac{\sqrt{W_2^2(\mathrm{N}_a,\mathrm{N}_b)}}{C} \right) \tag{9}$$

where: C is set empirically as the average size of the dataset name. The final NWDloss loss function is expressed as:

$$\mathrm{L}_{NWD} = 1 - NWD(\mathrm{N}_p, \mathrm{N}_g) \tag{10}$$

Finally, integrating different scales of objects in infrared images of power equipment, this paper proposes a hybrid edge loss based on CIoU and Wasserstein distance:

$$L_{loss} = \alpha L_{NWD} + (1-\alpha) L_{CIOU} \tag{11}$$

where: L_{CIOU} is the original CIoU-based border loss. α is an adjustable hyperparameter.

III. Design of path planning model based on DQN algorithm

III. A. Deep Q learning network

DQN is the representation of Q-value by neural network, traditional Q learning is equivalent to checking the table, which contains the action value function, when the intelligent body is in a certain state, in order to make the executed action have the maximum action value function, the corresponding action is obtained by checking the table, and the table is updated after each execution of the action, and so on iteratively, so that the action value function contained in the table is close to the true value of the action. This method is effective when the state and action space is small, but when the input space increases, the table also increases, which results in dimensionality explosion, and one solution is to represent the Q-value as a nonlinear function.

Several contributions of DQN networks include:

(1) The stability of training is greatly improved by using deep learning networks (CNNs in the case of DQNs), empirical playback, and target networks.

(2) End-to-end learning is achieved, with inputs as image pixels and outputs directly as action values.

(3) The same algorithm, network architecture, and hyperparameters can perform very well across different tasks.

DQN uses a CNN network to approximate the optimal action value function, which can be expressed as:

$$Q^*(s,a) = \max_{\pi} E\left[\sum_{0}^{\infty} \gamma^i r_{t+1} \mid s_t = s, a_t = a, \pi\right] \tag{12}$$

When combining away strategies, function approximation, and bootstrapping, the training process is prone to instability as well as fragmentation.

DQN uses both empirical playback and goal network to cope with the above problems. In empirical playback, the observation sequence (s_t, a_t, r_t, s_{t+1}) is stored in the empirical pool, and in order to eliminate the correlation between samples as well as to smooth the distribution of samples, samples are randomly drawn from the sample pool during the training process. Secondly, the DQN still uses an objective network to separate the parameters of the Q network and periodically updates the parameter values of the objective network so as to eliminate the correlation between the action-value functions Q and $r + \gamma \max_{a'} Q(s', a')$. The following equation is the loss function used to update the parameters of the Q function in step i.

$$Loss = (r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a, \theta_i))^2 \tag{13}$$

where θ_i denotes the weights of the Q-network at the i th step, θ_i^- denotes the weights of the target network at the i th step, and θ_i^- is updated periodically [16].

III. B. DQN algorithm

III. B. 1) Neural network fitting Q-tables

The DQN algorithm uses a convolutional neural network nonlinear approximation of the value function in lieu of a Q-value table, where the weight parameter of each network layer is represented by a conforming \omega in the neural network. The carrier of the state-action value function in the DQN algorithm is the neural network, through which the state of the environment is read and computed, and then the state-action value function is obtained.The O-value table is replaced by approximating the Q-network with the parameter ω, as shown in Eq. (14).

$$Q(s, a, \omega) \approx Q(s, a) \tag{14}$$

The output $Q(s, a, \omega)$ function value of the neural network is used to approximate and replace the Q value, and the network model for replacing the Q value is shown in Figure 3. Where the input state is denoted as s and the value function of the action a_i in state s is denoted as $Q(s, a_i, \omega)$.

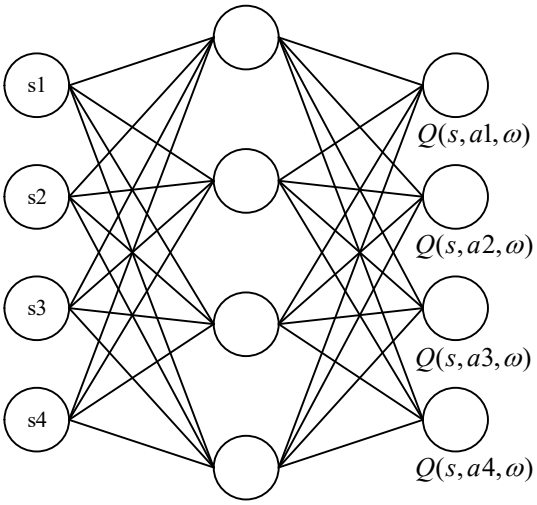

Figure 3: Network model replacing the Q value

III. B. 2) Experience playback mechanisms

The nonlinear approximation of the value function by the neural network in the DQN algorithm again causes the problem of instability or non-convergence of the algorithm, in order to solve this problem, so an experience playback mechanism is added to the deep Q complex. The structure of the experience playback mechanism is shown in Figure 4.

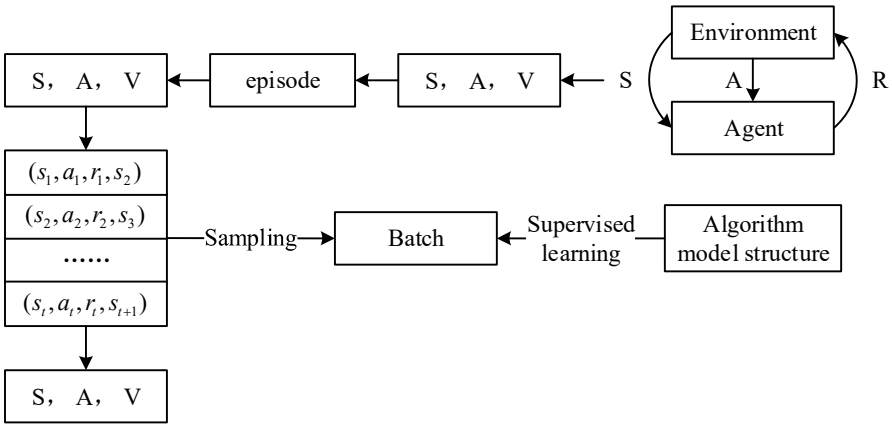

Figure 4: Structure of the experience replay mechanism

The generated empirical data $e=(s,a,r,s')$ is stored into the experience pool during the training process, and the intelligent body can use the information of these data again, which can make the data satisfy the independent homogeneous distribution. Any two data tuples e1 and e2 in the experience pool are weakly correlated with each other, and the Q network learns by sampling from the experience pool in a stochastic manner, which breaks the correlation between the data and ensures that the system can converge stably. Because neural network gradient descent does not try different directions, with this experience pool, data can be randomly sampled from it, and then the gradient is solved based on the sampled data, which can effectively avoid the problem of gradient descent in a single direction.

III. B. 3) Target networks

An objective network is used in DQN, which has an objective function of $Q(s,a,\omega^-)$, where ω^- is the objective network parameter. In order for the DQN algorithm to have a better performance, it is important to choose an appropriate update interval N for the target network. In the DQN algorithm the value function Q is a set of vectors, with \omega denoting the network weights of the neural network, and $Q(s,a,\omega)$ denoting the output of the current network, and the output of the target network is denoted by $Q^-(s_{t+1},a_{t+1},\omega^-)$. Where the current network parameters are denoted as ω and the target network parameters are denoted as ω^-. The computational formula of the

objective value function in the DQN algorithm is shown in Equation (15), where y_t is the objective label, which denotes the approximate optimization objective of the value function.

$$y_t = \begin{cases} R_t \\ R_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \omega) \end{cases} \tag{15}$$

The state value function iteration is utilized to compute the Q-value function, and then the Q-value function is used to update the weight parameter of the current network, and then after a certain number of steps the target network will copy the weight parameter away, and in this way the target network update is achieved by continuous copy update iteration [17]. Therefore the target network Q-value function is improved as shown in equation (16).

$$y_t = \begin{cases} R_t \\ R_t + \gamma \max_{a_{t+1}} Q^-(s_{t+1}, a_{t+1}, \omega^-) \end{cases} \tag{16}$$

By iterating this process over and over again, the neural network adjusts the weights until the value of the loss function is minimized. The backpropagation algorithm is an effective solution to this problem. It updates the value of each weight by calculating the contribution of each weight to the loss function. The computational formula for the loss function is shown in equation (17).

$$L(\omega) = E_{(s,a,r,s')}\left[(R_t + \gamma \max_{a_{t+1}} Q^-(s_{t+1}, a_{t+1}, \omega^-) - Q(s,a,\omega))^2\right] \tag{17}$$

The loss function is then polarized with respect to the network parameters \omega, calculated as shown in equation (18).

$$\frac{\partial L(\omega)}{\partial \omega} = E_{(s,a,r,s')}\left[(R_t + \gamma \max_{a_{t+1}} Q^-(s_{t+1}, a_{t+1}, \omega^-) - Q(s,a,\omega))\nabla_\theta Q(s,a,\omega)\right] \tag{18}$$

III. B. 4) Flow of the DQN algorithm

DQN algorithm is used to extract the feature vector data of the image by convolutional neural network and use reinforcement learning algorithm to optimize the control strategy, which combines the perceptual decision making power of both deep learning and reinforcement learning methods.The basic framework of the DQN algorithm model is shown in Fig. 5.

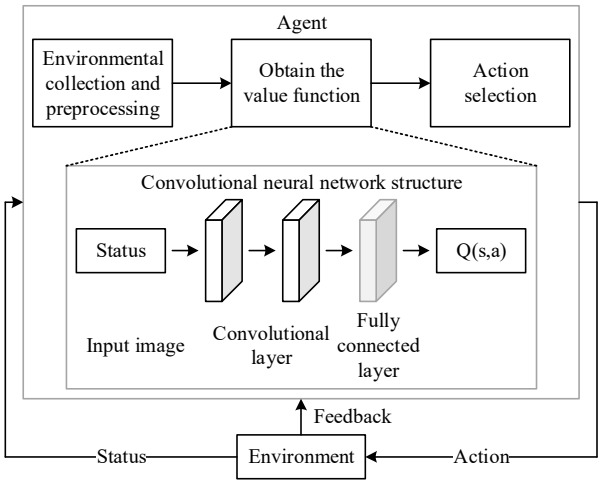

Figure 5: Basic Framework of the DQN algorithm

Randomly select actions under state s with ε greedy strategy or select the optimal action according to the model $\max Q(s,a,\omega)$. A new state s' is obtained after executing the action a and the reward value r, and this set of (s,a,r,s') quaternion sequences is then stored in the experience pool. in order to subsequently sample a small batch of (s,a,r,s') sequences from it for training and calculating the Q-value. Finally, the solution is based on stochastic gradient descent method and the target network copy parameters are updated after every N iterations.

The following is the algorithmic flow of path planning based on DQN algorithm.

(1) Initialize the parameter empirical pool D, the current network parameters ω , the target network parameters ω^{-}

(2) Set the maximum number of steps N, initialize $i=1$, and when in the round $i<N$, perform the following steps:

a) Randomly initialize the environment state to get the state s

b) Select the randomized action or the optimal action for the current state Q value according to the ε-greedy greedy policy. i.e:

$$a_t = arg\max_a Q(s_t, a) \tag{19}$$

c) After executing the action a_t , get the reward R_t , reach the new state s_{t+1} , and deposit the obtained quaternion data into the experience pool D.

d) Experience playback, a batch of sample data (s_i, a_i, r_i, s_{i+1}) is randomly sampled from the experience pool for training, and the network function values are computed to update the Q-value of each state:

$$Q(s,a) = R_t + \gamma \max_{a_{t+1}} Q^{-}(s_{t+1}, a_{t+1}, \omega^{-}) \tag{20}$$

e) Do stochastic gradient descent on the Q-network to compute the weights with a loss function:

$$L(\omega) = E_{(s,a,r,s')}\left[(R_t + \gamma \max_{a_{t+1}} Q^{-}(s_{t+1}, a_{t+1}, \omega^{-}) - Q(s,a,\omega))^2\right] \tag{21}$$

f) Network parameter omega update, updating the current state to the new state, $s_t \leftarrow s_{t+1}$

(3) End of round, $i+=1$, update the weights of the target network after a certain number of rounds $\bar{\omega} \leftarrow \omega$

III. C. Design of Adaptive Exploration Strategy Mechanisms

The DQN algorithm uses the ε-greedy greedy strategy to select the action, which is a small probability random selection, large probability selection of the optimal rule, that is, with a probability of ε random selection, $1-\varepsilon$ probability of selecting the optimal action, then the mathematical expression for the ε-greedy greedy strategy to select the action, as shown in Equation (22).

$$A(s)arrow\begin{cases} \arg\max_a Q(s,a), 1-\varepsilon \\ rand(A), \varepsilon \end{cases} \tag{22}$$

Among the parameter settings for the greedy strategy, the parameter ε is generally set to a fixed value. Because in reinforcement learning through the reward value to respond to the results of the interaction between the intelligent body and the environment, so you can establish a mapping relationship between the reward value and the parameters of the ε-greedy strategy, through which the connection to adjust the strength of the exploration, so as to make the exploration and utilization to achieve an optimal equilibrium, the mathematical expression of the mapping relationship, shown in Eq:

$$\varepsilon_t = \begin{cases} \varepsilon_{t-1} K^{|\Delta R|}, \Delta R \le 0 \\ \varepsilon_{t-1} - (1-\varepsilon_{t-1}) K^{|\Delta R|}, \Delta R > 0 \end{cases} \tag{23}$$

$$\Delta R = R_t - R_{t-1} \tag{24}$$

where ΔR denotes the difference between the reward value at round t and round t-1, K is a constant, and $0<K<1$. When $\Delta R>0$, i.e. $R_t > R_{t-1}$, the increase in the reward value of the feedback indicates that the execution of the action is closer to the target point, and it should be utilized more often, and at this time, according to the mapping relationship between the reward value and the strategy parameter we get, $\varepsilon_t = \varepsilon_{t-1} - (1-\varepsilon_{t-1}) K^{\Delta R}$, after calculating through the function the value of ε decreases, so $1-\varepsilon$ increases, that is to say, the probability of choosing the optimal action increases. Conversely, when $\Delta R \le 0$, that is, $R_t \le R_{t-1}$, the feedback reward value decreases indicates that the action is "useless", and you should continue to explore the new environment to find useful information, at this time, $\varepsilon_t = \varepsilon_{t-1} K^{\Delta R}$, that is, the value of ε increases, Increases the probability of random action selection.

III. D. Design of the mechanism for changing the objective function

The DQN algorithm combines the advantages of both deep learning and reinforcement learning algorithms, and at the same time the overestimation problem arises. The following is the process of arithmetic proof, after executing

the action a under the state s, the output real value is (s,a), and the estimated value is $Q^{-}(s,a)$, and the estimation error is denoted by Y_s^a, and the expression is shown in Eq. (25) if it is uniformly distributed on the interval $[-\beta,\ \beta\ ]$.

$$Q^{-}(s,a)=Q(s,a)+Y_s^a \tag{25}$$

The error between the estimated value and the true value of the intelligent body in state s is represented by B_s, and the calculation process is shown in equation (26).

$$\begin{aligned} B_s &= (R_s^a+\gamma\max_{a'}Q^{-}(s',a'))-(R_s^a+\gamma\max_{a'}Q(s',a')) \\ &= \gamma(\max_{a'}Q^{-}(s',a')-\max_{a'}Q(s',a')) \\ &= \gamma(\max_{a'}(Q(s',a')+Y_{s'}^{a'})-\max_{a'}Q(s',a')) \\ &= \gamma(\max_{a'}Y_{s'}^{a'}) \end{aligned} \tag{26}$$

Since Y_s^a, is uniformly distributed on the interval $[-\beta,\beta]$, its expectation $E[Y_s^a]=0$, so $P(\max_{at+1}P_s^a>0)>P(\max_{a_{t+1}}P_s^a<0)$. When $E[Y_s^a]=0$, it usually has $E[B_s]>0$ for all actions a, i.e., $Q^{-}(s,a)>Q(s,a)$. If this estimate is larger than its true value and also uses the maximization operation, it is more prone to the overestimation problem. Suppose the intelligent body can only choose actions al and a2 in state s and $Q(s,a1)>Q(s,a2)$, because the network itself carries a uniform error of β magnitude, so there may be a situation where $Q(s,a1)-\beta<Q(s,a2)+\beta$, and then the intelligent body, through the maximization operation, may not choose the optimal action, resulting in the intelligent body not learning the optimal path.

Aiming at the over-estimation problem of the DQN algorithm, a change function is designed to improve its objective value, increasing the optimal value and decreasing the suboptimal value with the corresponding relationship, so as to realize the purpose of increasing the gap between the optimal value and the suboptimal value, and make it easier for the intelligent body to recognize the optimal value and select it. The function model is defined as the change function B.

The original objective value function of the DQN algorithm:

$$y_t=R_t+\gamma\max_{a_{t+1}}Q^{-}(s_{t+1},a_{t+1},\omega^{-}) \tag{27}$$

The change function model is shown in equation (28), where b is the change function parameter, $0<b<1$.

$$B(s_t,a_t,r_t,s_{t+1})=\begin{cases}1, Q(s_t,a_t,r_t,s_{t+1})=\max_a Q(s_t,a_t,r_t,s_{t+1}) \\ b^{Q(s_t,a_t,r_t,s_{t+1})}, \text{other}\end{cases} \tag{28}$$

The objective value $y_t=V^{*}(s)*B(s,a)$ after adding the change function.

Where the state value function $V^{*}(s)$ is as follows in equation (29), where s_T is the final state of training.

$$V^{*}(s)=\begin{cases}R_t, s_{t+1}=s_T \\ R_t+\gamma\max_{a_{t+1}}Q^{'}(s_{t+1},a_{t+1},\omega^{-}), \text{other}\end{cases} \tag{29}$$

IV. Simulation experiments and analysis of results

IV. A. Target Detection Experiment

IV. A. 1) Experimental environment setup

This experiment uses PyTorch2.3.0 deep learning framework to build the network. The experimental platform is Pycharm, Python version is Python3.9. The operating system is Window11. The experimental training process are all using the same hyperparameters, the experiments are using cosine annealing strategy and SGD optimizer.

IV. A. 2) Data set selection and pre-processing

In order to deeply simulate the complex traffic intersection scene in the automatic driving scenario and make this paper's algorithm more suitable for the needs of intelligent transportation system, the classifier performance analysis is carried out for the traffic intersections in the actual monitoring video, and the data in this paper is extracted from the monitoring video of the traffic intersections in a certain region, and a total of 2532 images are collected. Through

the data augmentation technique, the data set is expanded to 6455 images. The data set is divided into training set, validation set and test set in the ratio of 12:1:1. According to the practical application scenarios, a total of 9 categories of people, car, bicycle, bus, green light, red light, tricycle, truck and vehicles are selected for this experiment and the dataset is converted to YOLO format for training and evaluation of the algorithm.

IV. A. 3) Experimental results

(1) Evaluation Metrics

The evaluation metrics of the algorithm selected for the experiments in this paper include precision rate P, recall rate R, mean average precision (mAP), and FPS denotes frame rate. Where: TP denotes the correct detection frame, FP denotes the false detection frame, AP denotes the detection precision of a target, mAP denotes the average value of AP of all categories, N denotes the number of detection categories, and Frames denotes the number of frames.

(2) Improved algorithm detection results

Table 1 shows the accuracy of the detection target. The results show that compared with the YOLOv7 algorithm, the proposed algorithm AP@0.5 and improves by 3.2 percentage points in all detection categories, and the detection speed is slightly reduced, but it still satisfies the real-time performance. Compared with YOLOv7, the AP@0.5 of the proposed algorithm is improved by more than 3.1 percentage points compared with the YOLOv7 algorithm when dealing with tasks with multiple scale targets (such as car(0) and people(1)), which shows its advantages in dealing with object detection tasks with a large scale span. Compared with YOLOv7, the AP@0.5 of the proposed algorithm is improved by more than 4.8 percentage points compared with the YOLOv7 algorithm when dealing with small targets (such as light(4-5)), indicating that the proposed algorithm has excellent performance in small target detection.

Table 1: Test target accuracy contrast

Algorithm		YOLOv7	Ours
mAP@0.5/%		80.1	83.3
Prcision		85	88.1
Recall		75.1	76.5
Frame rate/(frame·s^{-1})		69.7	64.9
AP@0.5/%	Categories 0	80.6	83.7
	Categories 1	87.1	95.1
	Categories 2	76	79.2
	Categories 3	89.4	93.9
	Categories 4	68.2	73
	Categories 5	83.1	88.7
	Categories 6	67	69.1
	Categories 7	80.7	82.7
	Categories 8	86.7	87.5

(3) Image enhancement network experiments

In order to verify the effectiveness of SPPCSPC, this paper uses some mainstream image enhancement networks to conduct comparative experiments on YOLOv7 on self-made datasets, and the comparative experiments of image enhancement networks are shown in Table 2. When SPPCSPC was added, the algorithm mAP@0.5 reached 81.7%. Compared with CPAEnhancer, although the frame rate is reduced by 0.5 frame/s, the mAP@0.5 is increased by 1.1 percentage points. Compared with the baseline, although the frame rate is reduced by 7.6 frame/s, it still meets the real-time requirements, and the mAP@0.5 is 1.9 percentage points higher, which proves the effectiveness of the method.

Table 2: Image enhancement network comparison experiment

Image enhancement network	mAP@0.5/%	Frame rate/(frame·s^{-1})
YOLOv7(baseline)	79.8	70.5
+Retinexformer	80.9	60.5
+ CPA-Enhancer	80.6	63.4
+SPPCSPC	81.7	62.9

(4) Characteristic pyramid experiment

In order to verify the effectiveness of the SimAM attention mechanism, this paper uses the SimAM attention mechanism and some mainstream feature pyramids to conduct comparative experiments on the YOLOv7 algorithm in the self-made dataset, and the feature pyramid comparison experiments are shown in Table 3. The frame rate of BiFPN is still higher than that of PANet. However, the improvement in accuracy is not significant. Compared with PANet, the frame rate of SimAM attention mechanism is increased by 4.5 frame/s, and the mAP@0.5 is also increased by 2.5 percentage points to 82.9%, which proves the effectiveness of the method.

Table 3: The characteristics pyramid comparison experiment

Characteristic pyramid	mAP@0.5/%	Frame rate/(frame·s^{-1})
PANet(baseline)	80.4	71.1
+BiFPN	80.5	78.7
+ HS-FPN	77.6	88.9
+SimAM	82.9	75.6

(5) Loss letter experiment

In order to verify the effectiveness of NWDloss, this paper uses NWDloss and some mainstream loss functions to conduct comparative experiments on the YOLOv7 method in the self-made dataset, and the comparative experiments of the loss function are shown in Table 4. The experimental results show that when NWDloss is used as the loss function, although the detection speed on the self-made dataset is slightly reduced, the mAP@0.5 is increased by 2.6 percentage points compared with the CIoU, which proves the effectiveness of the proposed method.

Table 4: Loss function comparison experiment

Loss function	mAP@0.5/%	Frame rate/(frame·s^{-1})
Ciou(baseline)	80.1	67.2
SIoU	77.9	76.7
MPDIoU	79.6	68.3
NWDloss	82.7	68.5

(6) Loss function moderating factor selection experiment

In order to verify the influence of different regulators on the detection results and select the optimal moderators, different regulators were selected for experimental comparison. Table 5 shows the comparative experiments of the moderating factor of the loss function. With the change of the moderator, the mAP@0.5 also changes, and when the ratio is 1.0, the mAP@0.5 is higher than when the ratio is other values. When the ratio is 1.0 and the u is 0.92, the maximum value of mAP@0.5 reaches 80.7%.

Table 5: Comparison of loss function adjustment factor

Regulating factor	mAP@0.5/%
CIoU(baseline)	79
ratio=0.7 d =0 u =0.95	81.6
ratio=1.0 d =0 u =0.95	82.1
ratio=1.3 d =0 u =0.95	80.7
ratio=1.0 d =0 u =0.98	80.7
ratio=1.0 d =0 u =0.92	81

(7) Comparative experiments

In order to verify the detection performance of the proposed algorithm, the proposed algorithm is compared with other object detection algorithms Faster-RCNN, SSD, YOLOv5s, YOLOv7, YOLOv8s, YOLOv7, Dynamic R-CNN, Cascade RCNN, Deformable DETR and RTMDet, and the comparison experiments are shown in Table 6. It can be seen from the table that in terms of detection accuracy, the mAP@0.5 of the proposed algorithm in this paper reaches 83.9%, which is 21.6, 18.9, 18.2, 14.1, 11.6, 4, 32.3, and higher than that of Faster-RCNN, SSD, YOLOv5s, YOLOv7, YOLOv8s, YOLOv7, DynamicR-CNN, Cascade RCNN, Deformable DETR, and RTMDet, respectively. 26.8, 9.5, 4 percentage points. In terms of real-time detection performance, except for the original YOLOv7 algorithm,

the frame rate of the proposed algorithm is significantly better than that of other algorithms. In summary, the proposed algorithm shows superior detection performance in the field of object detection.

Table 6: Comparison experiment

Algorithm	Frame rate/(frame·s^{-1})	mAP@0.5/%
Faster R-CNN	18.3	62.3
SSD	39	65
YOLOv5s	60.4	65.7
YOLOv7	32.5	69.8
YOLOv8s	62.2	72.3
YOLOv7	70.1	79.9
Dynamic RCNN	31.2	51.6
Cascade RCNN	27.4	57.1
Deformable DETR	22.8	74.4
RTMDet	17.9	79.9
Ours	64.6	83.9

(8) Generalizability comparison experiment

To verify the algorithm's generalization ability, this paper uses the dynamic driving dataset. The dataset covers diverse scenarios in several large cities, including different weather conditions, time periods and locations. The dataset contains 5,000 training images, 5,000 validation images and 10,000 test images, covering a total of six major human-vehicle scene categories: Pedestrian, Cyclist, Car, Truck, Tram and Tricycle. batch size is 8 and the number of training times is 50. The generalization comparison experiment is shown in Table 7. As can be seen from the table, in terms of detection accuracy, the mAP@0.5 of the proposed algorithm in this paper reaches 56.9%, which is significantly improved compared with other algorithms. In terms of real-time detection performance, the frame rate of the proposed algorithm reaches 57.4frame/s, which is significantly better than other algorithms except the YOLOv7 algorithm. In summary, the algorithm proposed in this paper is generalized in the field of object detection.

Table 7: Generalization comparison experiment

model	Frame rate/(frame·s^{-1})	mAP@0.5/%
YOLOv7	63.3	53.6
Dynamic RCNN	28.3	29
Cascade RCNN	25.4	30.9
Deformable DETR	19.8	39.2
RTMDet	16.6	43.8
Ours	57.4	56.9

IV. B. Path Planning Simulation Experiment

IV. B. 1) Simulation environment construction

In order to verify the superiority of the improved deep Q-learning algorithm compared to the traditional algorithm, this paper designs a simulated 2D map environment for AGV path planning under discrete manufacturing system using raster method based on python language. The upper left corner of the raster map environment is set as the origin of the coordinates, the horizontal direction is set as the x-axis, the vertical direction is set as the y-axis, the non-shadowed area is the movable area, and the shadowed area is the obstacle or congestion module. The network structure of the improved deep Q learning algorithm proposed in this paper is built and trained through Tensorflow, and the GPU hardware environment used for training is NVIDIA RTX A4000 with 16GB of video card memory.

IV. B. 2) Analysis of example results

Comparing the improved deep Q learning algorithm proposed in this paper with the traditional deep Q learning algorithm under the configuration of the hyperparameters in Table 2, the AGV path planned by the improved DQN is shown in Fig. 6. The AGV path planned by the traditional DQN is shown in Fig. 7. The two algorithms are used to conduct 5000 rounds of AGV path planning experiments in the same simulation environment, and from the comparison of Fig. 6 and Fig. 7, it can be seen that the improved algorithm plans a better smooth path, which is more efficient and energy-saving for the use of AGVs in the manufacturing system, and the paths planned by the traditional DQN algorithm are always “too careful” in the vicinity of obstacles and always turn or change early. The

traditional DQN algorithm is always “too careful” to turn or change lanes in advance when an obstacle is nearby, resulting in too many corners in the path, which is a manifestation of the traditional DQN algorithm's over-estimation problem: after encountering an obstacle and being penalized during training, once there is a move that can be avoided, it is easy to overestimate its Q value and will not explore other effective paths.

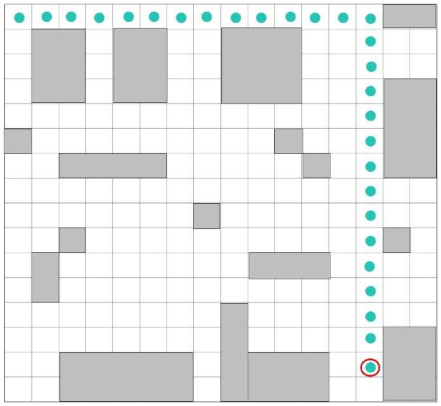

Figure 6: Improve the AGV path planned by DQN

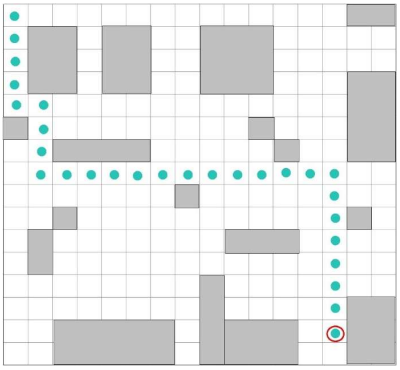

Figure 7: The AGV path planned by the traditional DQN

The reward value of the algorithm is generally used to measure the performance of the reinforcement learning algorithm, because the core idea of the reinforcement learning algorithm is to train behavioral strategies by the AGV intelligences continuously obtaining rewards or punishments from the environment. The reward value of the improved DQN is shown in Fig. 8. The reward value of traditional DQN is shown in Fig. 9. From Fig. 8 and Fig. 9, it can be seen that the cumulative rewards obtained by the improved algorithm are higher and the improved DQN algorithm can obtain the maximum cumulative rewards from the environment 100 when the interaction reaches about 1500 rounds, while the traditional DQN starts to obtain the maximum rewards only at about 4000 rounds.

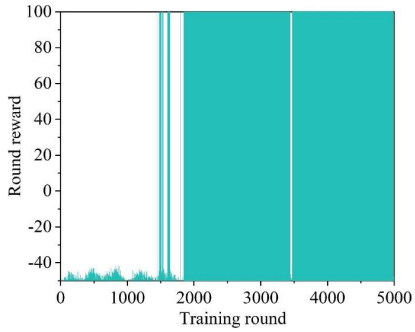

Figure 8: Improve the bonus value of DQN

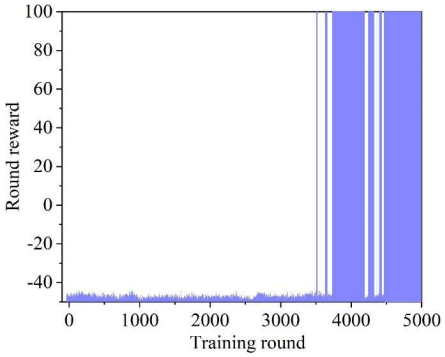

Figure 9: Traditional DQN rewards

As with other algorithms that use neural network backpropagation, the smaller the cost of the algorithm, the closer the network output is to the target value, and the smaller the magnitude of the oscillation of the cost curve, the more stable the performance is. The cost convergence graph of the traditional DQN is shown in Figure 10. The cost convergence graph of the modified DQN is shown in Fig. 11. From Fig. 10 and Fig. 11, it can be seen that compared with the traditional DQN, the generation value of the improved DQN method is significantly reduced, and the magnitude of the oscillation is very small. On the other hand, since the improved algorithm no longer samples the training data randomly, but gets the data that are more needed to be trained to by prioritizing the weight sampling, it can be seen from the change of the number of steps required for training shown in Fig. that the average number of steps of the improved algorithm is reduced by about 50%, i.e., the convergence speed of the improved algorithm is improved.

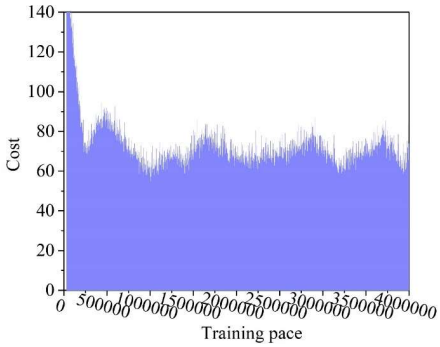

Figure 10: Traditional DQN price convergence diagram

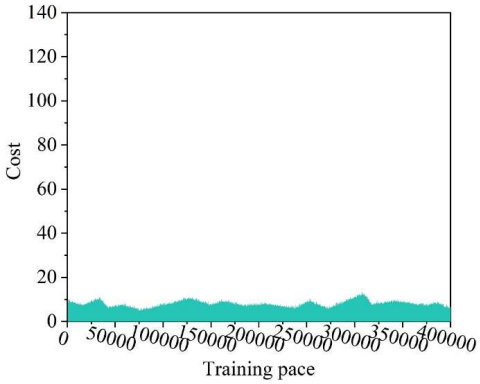

Figure 11: The cost convergence of DQN after the change

V. Conclusion

Designing an accurate target detection algorithm and path planning algorithm is important for the intelligent application of multiple intelligences in multiple scenarios. The article draws the following conclusions:

The proposed algorithm obtains satisfactory detection results in 9 categories of datasets. Experimental results show that the AP@0.5 of the proposed algorithm is improved by more than 3.1 percentage points compared with the YOLOv7 algorithm, and the AP@0.5 of the proposed algorithm is increased by more than 4.8 percentage points compared with the YOLOv7 algorithm when dealing with small targets.

Path planning experimental simulation results, this paper's algorithm is able to plan a good AGV path in the map, and the path smoothness is improved to a certain extent compared with the traditional DQN algorithm, at the same time, the improved algorithm in the reward value, the stability of the algorithm and the speed of convergence than the traditional DQN algorithm, no longer prone to fall into over-estimation problems.

References

[1] Liu, M., Ma, H., Li, J., & Koenig, S. (2019, January). Task and path planning for multi-agent pickup and delivery. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS).

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

[3] 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.

[4] 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.

[5] Dao, M. Q., Berrio, J. S., Frémont, V., Shan, M., Héry, E., & Worrall, S. (2024). Practical collaborative perception: A framework for asynchronous and multi-agent 3D object detection. IEEE Transactions on Intelligent Transportation Systems.

[6] Jiang, M., Hai, T., Pan, Z., Wang, H., Jia, Y., & Deng, C. (2019). Multi-agent deep reinforcement learning for multi-object tracker. IEEE Access, 7, 32400-32407.

[7] Gu, K., Wang, Y., & Shen, Y. (2020). Cooperative detection by multi-agent networks in the presence of position uncertainty. IEEE Transactions on Signal Processing, 68, 5411-5426.

[8] 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.

[9] Fang, Z., Zhao, J., Yang, M., Lu, Z., Zhou, W., & Li, H. (2024). Coordinate-aligned multi-camera collaboration for active multi-object tracking. Multimedia Systems, 30(4), 221.

[10] Kholladi, N. H., Bechkoum, K., & Hamoud, M. (2024). A proposed cooperative approach for edge detection based multi-agent system (MAS) using heterogeneous RGB images. Studies in Engineering and Exact Sciences, 5(3), e12993-e12993.

[11] Yang, D., Yang, K., Wang, Y., Liu, J., Xu, Z., Yin, R., ... & Zhang, L. (2023). How2comm: Communication-efficient and collaboration-pragmatic multi-agent perception. Advances in Neural Information Processing Systems, 36, 25151-25164.

[12] Sun, Z., Yu, Z., Guo, B., Yang, B., Zhang, Y., & Ng, D. W. K. (2024). Integrated sensing and communication for effective multi-agent cooperation systems. IEEE Communications Magazine, 62(9), 68-73.

[13] Zhu, Z., Du, Q., Wang, Z., & Li, G. (2022). A survey of multi-agent cross domain cooperative perception. Electronics, 11(7), 1091.

[14] Zhiqiang Wu,Jiaohua Qin,Xuyu Xiang & Yun Tan. (2025). YOLO-CBF: Optimized YOLOv7 Algorithm for Helmet Detection in Road Environments. Electronics,14(7),1413-1413.

[15] Ruijie Peng,Chuanlin Liao,Weijun Pan,Xiaolin Gou,Jianwei Zhang & Yi Lin. (2025). Improved YOLOv7 for small object detection in airports: Task-oriented feature learning with Gaussian Wasserstein loss and attention mechanisms. Neurocomputing,634,129844-129844.

[16] Sherko Salehpour,Aref Eskandari,Amir Nedaei & Mohammadreza Aghaei. (2025). Two-stage deep Q-network reinforcement learning based ultra-efficient fault diagnosis and severity assessment scheme for photovoltaic protection. Energy and AI,20,100512-100512.

[17] Shanshan He,Huixi Li,Yuelei Ji,Xingjian Chen,Feifei Zou & Xia Zhao. (2025). Energy-aware mobile edge device deployment in health data detection scenario: a DQN optimized firefly algorithm. Computing,107(5),115-115.