Proceedings of the 3rd International Conference on Mechatronics and Smart Systems
DOI: 10.54254/2755-2721/125/2025.20196

A Vehicle Adaptive Cruise Control Method Based on Deep Reinforcement Learning Algorithm

Lingzhi Kong[1,a,*]

[1]*College of Mechanical and Electronic Engineering, Northwest A&F University, Yangling, 712100, China*
a. 1392484659@qq.com
**corresponding author*

Abstract: Adaptive cruise control (ACC) is an upgrade to the traditional cruise control system in vehicles. This paper presents an adaptive cruise control method based on the Deep Deterministic Policy Gradient (DDPG) algorithm. The Actor network computes actions based on the current state, adding noise to enhance exploration. The Critic network computes the Q-value of the current state-action pair. The Actor target network and Critic target network compute the Q-value of the next state-action pair. The gradient descent method is used to minimize the loss function of the Critic network, which includes the error between the real Q-value and the target Q-value. By integrating an Actor-Critic network, the system demonstrates improved adaptability and efficiency over traditional ACC methods, as shown through simulated experiments.

Keywords: Intelligent Vehicles, Deep Reinforcement Learning, Adaptive Cruise Control

1. 1. Introduction

With the development of society and the widespread use of automobiles, people's demands for driving have become increasingly high. The advancement of computer and network-related technologies has significantly impacted the automotive industry. Traditional automobiles are gradually evolving towards being intelligent, electric, and networked. People can now do more in their cars, making them a convenient aspect of life, gradually turning cars into "second homes." However, as the automotive industry progresses, there are higher expectations for vehicle safety, comfort, and other performance aspects, which have also introduced numerous challenges, such as energy choices, traffic safety, and environmental pollution. Vehicle intelligence has, to some extent, addressed these issues. Among them, the Adaptive Cruise Control system (ACC) is an advanced driving assistance technology that lightens the burden on drivers and enhances driving safety. Its primary functions include cruise control and maintaining a safe distance between vehicles. The presence of ACC improves driving safety, increases comfort, promotes traffic fluidity, and is of great significance to the general public's travel needs.

Author Milanés V and Shladover S E described the development of ACC and CACC control system models based on real experimental data, deriving the models from actual vehicle response measurements and applying them to simulate simple multi-vehicle following scenarios [1]. Author Zhang J and Zhong H modeled lane markings using the Catmull-Rom curve and incorporated a spatial attention module to leverage the near-vertical distribution of lane lines, offering both accuracy and

DOI: 10.54254/2755-2721/125/2025.20196

real-time performance, effectively addressing the problem of lane detection [2]. Author Zhang Y, Lin Y, and others proposed a hierarchical ACC considering collision avoidance, designing a TTC-based switching mechanism to select upper-level planning modes to avoid collisions while maintaining tracking ability, ensuring vehicle stability [3]. Authors Li, Ye, and Zhibin Li evaluated the impact of ACC parameter settings on rear-end collisions on highways, with results showing that the safety impact of ACC is largely influenced by parameters, and that combining ACC with variable speed limits can greatly enhance safety [4]. An adversarial pedestrian detection model based on virtual fisheye image training is proposed by Zhang J and Dou J, transforming features extracted from the backbone network into hard examples through introducing a spatial transformation network, using adversarial learning during training to enable the network to handle highly deformed objects, significantly improving the accuracy of image detection [5]. Authors Dey KC, Yan L, and others reviewed CACC systems in terms of communication, driver characteristics, and control, discussing the issues faced by current CACC control modules when approaching ideal driving conditions [6]. Authors Öncü S, et al. considered a cooperative adaptive cruise control system to regulate vehicle spacing in platoons, enhancing traffic flow stability and throughput [7]. Authors TIANBO LIU and JINDONG ZHANG proposed an adaptive traffic flow prediction model AD-GNN based on a spatio-temporal graph neural network, addressing the task of traffic flow prediction [8]. Authors Milanés V, Shladover SE, and others introduced the design, development, implementation, and testing of a CACC system, providing detailed experimental results to verify the controller's performance and its improvements over commercial ACC systems [9]. Authors JIN, JINGYI, ZHANG, JINDONG, and ZHANG proposed an online 3D MOT framework based on detection and tracking, effectively improving tracking accuracy [10]. Authors Y Zhao, X Ma et al. constructs a novel OAM generator, which successfully reduces channel correlation in LoS scenarios and significantly enhances channel capacity. Furthermore, this article emphasizes the electromagnetic characteristics in the physical domain that have been previously overlooked, which can propel the development of wireless communication by reconstructing the electromagnetic wavefront[11]. Authors C. Zhang and Y. Zhao introduces the concept of non-degenerate index mapping in orbital angular momentum (OAM) during partial phase plane reception, proposing the establishment of an OAM-based index mapping transmission scheme. Additionally, the author offers supplementary index channels to convey information through variations in OAM modes[12].

Based on DRL, the paper applies the Deep Deterministic Policy Gradient (DDPG) framework to achieve adaptive cruise control under DRL. Compared to human drivers, this algorithm demonstrates higher adaptability and stability.

2. Relevant Knowledge

Reinforcement learning (RL) is made up of two components:: the agent and the environment. The key question is how the agent can maximize the rewards it obtains in a complex and uncertain environment. During the RL process, the agent interacts continuously with the environment. After the agent acquires a certain state in the environment, it outputs an action based on that state, which is also referred to as a decision. This action is then executed within the environment, which provides the next state and the reward resulting from the action taken by the agent. Figure 1 illustrates the flowchart of the RL process.

DOI: 10.54254/2755-2721/125/2025.20196

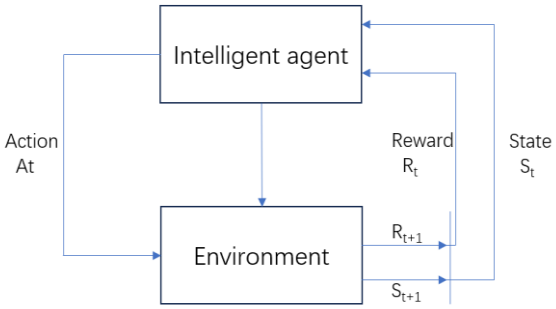

Figure 1: Reinforcement Learning Flowchart

Q-learning is a RL algorithm based on a value function with the objective of maximizing cumulative rewards through the acquisition of an optimal strategy. It employs a Q-table as a storage mechanism for the Q-values corresponding to each action in every state, where these Q-values represent the estimated future rewards anticipated from performing that action in. The update rule for the Q-value can be expressed as:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right) \tag{1}$$

Where s denotes the current state, a signifies the current action, r represents the immediate reward, s' indicates the subsequent state, a' is the next action, α stands for the learning rate, and γ is the discount factor. At each time step, the agent chooses an action based on the Q-table, carries it out, observes the resulting feedback from the environment, and subsequently updates the Q-table accordingly. Through continuous iteration, the Q-table gradually converges to the optimal values. Q-learning is simple and easy to implement, making it suitable for scenarios with small state and action spaces. The A3C (Asynchronous Advantage Actor-Critic) algorithm combines the actor-critic algorithm with the concept of asynchronous updates. It trains multiple concurrent environments, each with its own actor and critic. The process is as follows:

Step 1: Initialize multiple concurrent environments.

Step 2: At each time step, the actor in each environment selects an action based on the current state.

Step 3: Execute the action, observe the environment's feedback, and calculate the reward.

Step 4: The critic calculates the value function based on the current state and action to evaluate the action chosen by the actor.

Step 5: Use optimization algorithms such as gradient ascent to update the parameters of both the actor and the critic.

The approach enables fast training and is suitable for complex environments and multi-task learning. The A3C architecture diagram is shown in Figure 2.

DOI: 10.54254/2755-2721/125/2025.20196

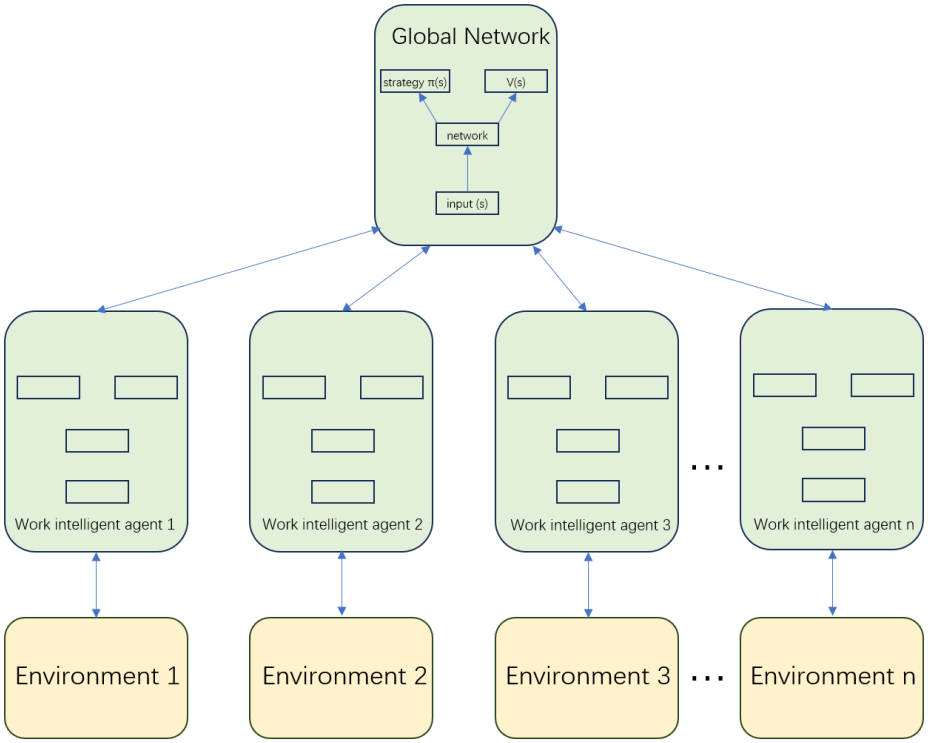

Figure 2: A3C Architecture Diagram

3. ACC Based on Deep Reinforcement Learning

The DRL represents a technological fusion of deep learning (DL) and RL, empowering intelligent systems to acquire optimal behavioral strategies through interactions with their environment. It utilizes deep neural networks to approximate the value function or policy in reinforcement learning, effectively handling high-dimensional data and complex decision-making problems. The DQN (Deep Q-Network) algorithm is developed from the Q-learning algorithm, with key differences being: first, the DQN algorithm replaces the Q-table in Q-learning with a neural network that approximates the action value function; second, the DQN algorithm introduces target networks and experience replay units, which enhance the stability of the algorithm. Target Network: The target network furnishes stable Q-value estimates for the computation of the loss function. Its parameters are not updated frequently during training but are synchronized periodically to avoid instability in the training process. Experience Replay Unit: This unit stores the agent's historical experiences, including state, action, reward, and next state, and samples from it randomly to mitigate the temporal correlation of data, thereby enhancing training efficiency and stability. The training process of DQN is illustrated in Figure 3.

DOI: 10.54254/2755-2721/125/2025.20196

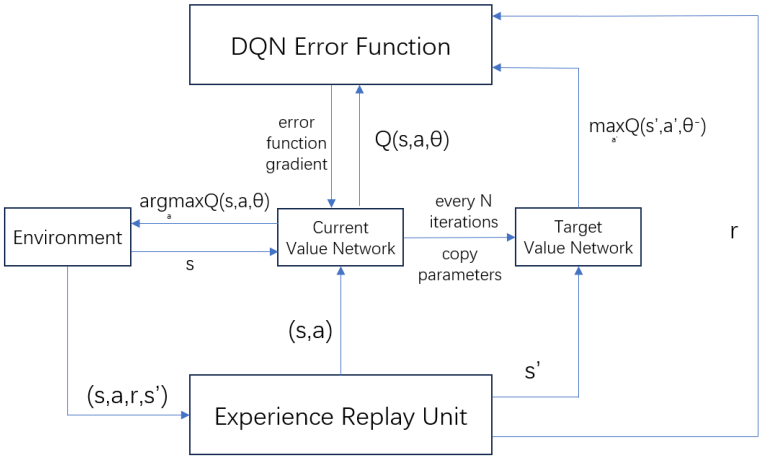

Figure 3: DQN Training Process

The DDPG algorithm is specifically tailored for the RL in continuous action spaces, developed by the DeepMind team based on DQN. Its core principle is to introduce deterministic policy gradients, allowing the algorithm to achieve efficient learning in continuous action spaces. The Actor network computes actions to maximize cumulative rewards, while the Critic network evaluates these actions. Through iterative learning, the system refines its decision-making capabilities, achieving efficient ACC. The implementation steps of the DDPG algorithm are as follows:

Step 1: Begin training when the number of data points in the experience replay pool exceeds a certain threshold.

Step 2: Randomly sample a batch of data from the experience replay pool.

Step 3: Use the Actor network to calculate the action for the current state, adding noise to increase exploration.

Step 4: Use the Critic network to compute the Q-value for the current state-action pair.

Step 5: Use the Actor target network and Critic target network to calculate the Q-value for the next state-action pair.

Step 6: Calculate the target Q-value according to the Bellman equation:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{S_{t+1}}\left[R_t + \gamma \cdot V_\pi(S_{t+1}) \mid S_t = s_t, A_t = a_t\right]$$

Step 7: Utilize gradient descent to minimize the loss function of the Critic network, which encompasses the discrepancy between the actual Q-value and the target Q-value.

Step 8: Use gradient ascent to maximize the cumulative expected return of the Actor network, i.e., maximize the Q-value output by the Critic network.

Step 9: Update the parameters associated with the target networks using a soft update mechanism.

DOI: 10.54254/2755-2721/125/2025.20196

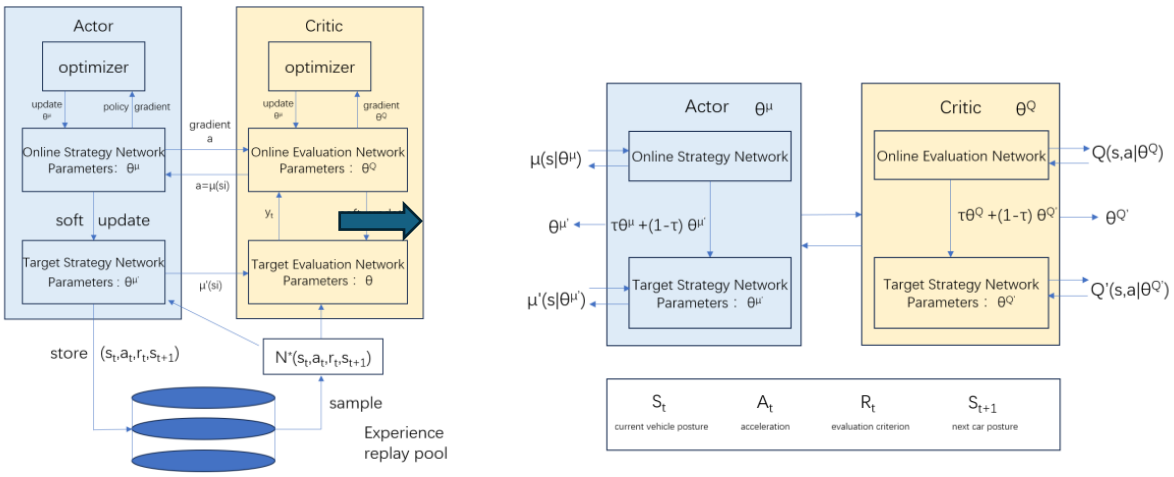

Figure 4: Basic Structure of the DDPG Algorithm

The combination of DRL with ACC utilizes the three main parameters of adaptive cruise control as inputs, which are then trained. Suitable activation functions for steering and braking are subsequently selected. The DDPG algorithm enhances the efficiency of DRL. The update formula for the policy network is given in equation (2), where the first half updates the parameters of the Critic neural network, and the second half modifies the parameters of the Actor network, increasing the probability of selecting that action.

$$\nabla_{\theta\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a\middle|\theta^Q\right)_{s=si,a=\mu(si)} \nabla_{\theta\mu}\,\mu\left(s\middle|\theta^\mu\right)|s_i \tag{2}$$

The update formula for the Q-value network is presented in equation (3), while equation (4) defines its loss function. The goal of the policy network is to modify the Actor network parameters to obtain a larger Q-value.

$$y_i = r_i + \gamma Q'\ (s_i+1, \mu'\ (s_i+1|\theta^{\mu\prime}\ )|\theta^Q \tag{3}$$

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{4}$$

The advantage of using deterministic policy gradients in the adaptive cruise control algorithm for intelligent vehicles is that it eliminates the need to debug a large number of parameters between inputs and outputs. It only requires the vehicle data from the perception layer and sensor data related to the reward function as inputs, with the outputs of DRL corresponding to the vehicle's low-level control of throttle, braking, and steering.

4. Experiment

The hardware environment used in this study includes an AMD Ryzen 7 7840H CPU with Radeon 780M Graphics at 3.80 GHz, 16.0 GB of RAM, and 1TB of disk space. The software environment consists of IDLE (Python 3.11 64-bit) running on Windows 11. The experimental results of the vehicle adaptive cruise control method based on DRL algorithms are shown in Figures 5, 6, and 7.

DOI: 10.54254/2755-2721/125/2025.20196

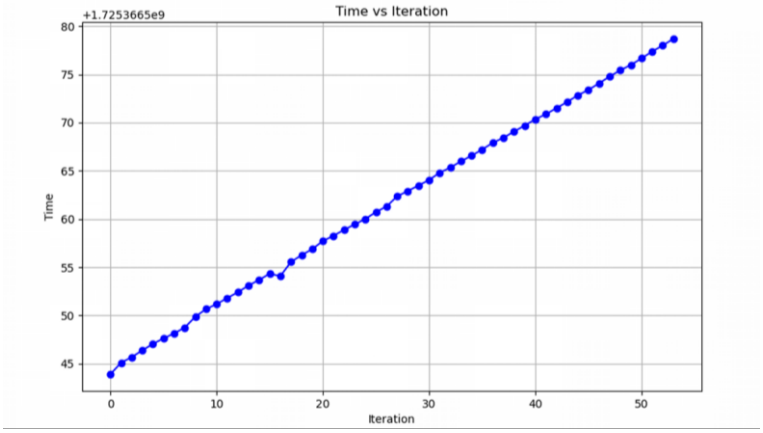

Figure 5: Program Running Time: Time required for each iteration of the DDPG-based adaptive cruise control algorithm.

The execution time of the algorithm is depicted in Figure 5. The rewards obtained during the execution action cycles are displayed in Figure 6, while the duration of each execution action cycle is presented in Figure 7.

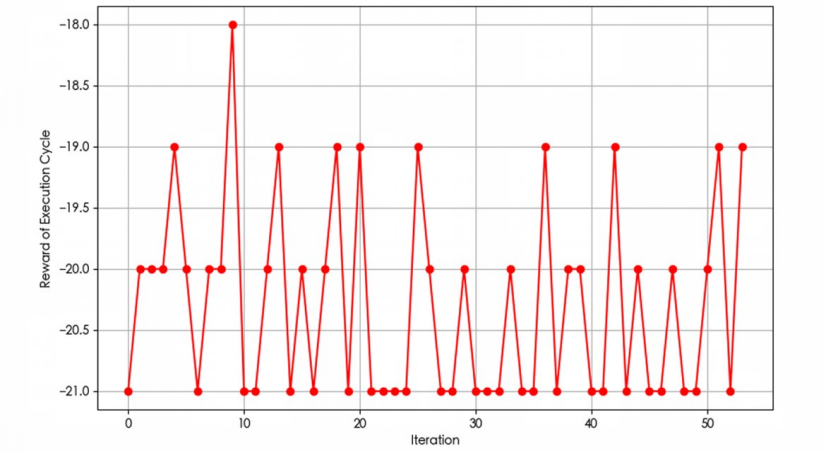

Figure 6: Execution Action Cycle Rewards: The reward given to each execution cycle of the adaptive cruise control algorithm based on DDPG.

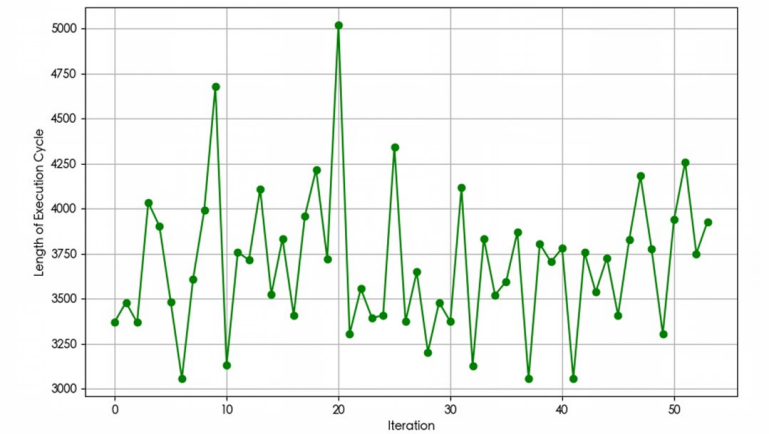

Figure 7: Execution Action Cycle Length: The adaptive cruise control algorithm based on DDPG corresponds to the length of each execution action cycle

DOI: 10.54254/2755-2721/125/2025.20196

5. Conclusion

This paper utilizes DRL to implement an adaptive cruise control algorithm for vehicles. The core principle lies in the introduction of deterministic policy gradients, allowing the algorithm to achieve efficient learning in continuous action spaces. It combines the DL technology with the Actor-Critic network, inheriting the advantages of the target network and experience replay from the DQN algorithm. Through the DDPG algorithm, the following outcomes were achieved: reduced vehicle spacing under safe conditions, improved traffic safety, reduced traffic energy consumption, rapid and accurate responses to sudden following situations, enhanced decision-making efficiency, and elevated levels of vehicle intelligence, meeting the expected results. This fully demonstrates the perceptual classification abilities of the DL and the efficient decision-making capabilities of the RL. Future research will focus on the practical application of this algorithm.

References

[1] *Milanés, V., & Shladover, S. E. (2014). Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data. Transportation Research Part C: Emerging Technologies, 48, 285-300.*

[2] *Zhang, J., & Zhong, H. (2023). Curve-based lane estimation model with lightweight attention mechanism. Signal, Image and Video Processing, 17(5), 2637-2643.*

[3] *Zhang, Y., Lin, Y., Qin, Y., Dong, M., Gao, L., & Hashemi, E. (2023). A new adaptive cruise control considering crash avoidance for intelligent vehicles. IEEE Transactions on Industrial Electronics, 71(1), 688-696.*

[4] *Li, Y., Ye, Z., & Zhibin, L. (2017). Evaluating the safety impact of adaptive cruise control in traffic oscillations on freeways. Accident Analysis & Prevention, 104, 137-145.*

[5] *Zhang, J., & Dou, J. (2024). An adversarial pedestrian detection model based on virtual fisheye image training. Signal, Image and Video Processing, 18(4), 3527-3535.*

[6] *Dey, K. C., Yan, L., Wang, X., Wang, Y., Shen, H., Chowdhury, M., Yu, L., Qiu, C., & Soundararaj, V. (2015). A review of communication, driver characteristics, and control aspects of cooperative adaptive cruise control. IEEE Transactions on Intelligent Transportation Systems, 17(2), 491-509.*

[7] *Öncü, S., Ploeg, J., Van de Wouw, N., & Nijmeijer, H. (2014). Cooperative adaptive cruise control: Network-aware analysis of string stability. IEEE Transactions on Intelligent Transportation Systems, 15(4), 1527-1537.*

[8] *Liu, T., & Zhang, J. (2023). An adaptive traffic flow prediction model based on spatiotemporal graph neural network. The Journal of Supercomputing, 79(14), 15245-15269.*

[9] *Milanés, V., Shladover, S. E., Spring, J., Nowakowski, C., Kawazoe, H., & Nakamura, M. (2013). Cooperative adaptive cruise control in real traffic situations. IEEE Transactions on Intelligent Transportation Systems, 15(1), 296-305.*

[10] *Jin, J., Zhang, J., & Zhang, K. (2024). 3D multi-object tracking with boosting data association and improved trajectory management mechanism. Signal Processing, 218, 109367.*

[11] *Y. Zhao et al. Near-Orthogonal Overlay Communications in LoS Channel Enabled by Novel OAM Beams Without Central Energy Voids: An Experimental Study. IEEE Internet of Things Journal, 2024, doi: 10.1109/JIOT.2024.3449975.*

[12] *C. Zhang and Y. Zhao. Orbital Angular Momentum Nondegenerate Index Mapping for Long Distance Transmission. IEEE Transactions on Wireless Communications, 2019, 18(11), 5027-5036.*