Autonomous Driving Decision-making Using Vision-based Reinforcement Learning in Multi-lane Highways

Pei-En Kao*, Chuan-Wei Cho*, Jen-Chu Liu†, and Meng-Shiuan Pan*
*Department of Electronic Engineering, National Taipei University of Technology, Taipei, Taiwan
Email: {t112368007, t109369012, mspan}@ntut.edu.tw
†Department of Information Management, Yuanpei University of Medical Technology, Hsin-Chu, Taiwan
Email: jcliu@mail.ypu.edu.tw

***Abstract*—Recently, autonomous driving systems have been discussed to provide safer and more efficient driving experiences. In this work, we design an autonomous driving decision-making method for vehicles that relies on camera sensors on vehicles. The proposed method leverages YOLO object detection to derive relative information from the environment and then utilizes a deep Q-network (DQN) reinforcement learning model to learn from this environmental information for making efficient autonomous driving decisions. This proposed method relies only on multiple cameras and vehicle speed data to determine appropriate speed and make lane-changing decisions. We implement training and experiments on the CARLA platform. Our experimental results demonstrate that the proposed method can facilitate autonomous vehicle control while maintaining safety.**

***Index Terms*—autonomous driving, reinforcement learning, decision-making, deep Q-learning network.**

I. Introduction

Autonomous driving is widely regarded as a transformative technology for future transportation systems. An autonomous driving system can perceive obstacles or risks continuously without lapses in attention based on sensors onboard vehicles (e.g., LiDAR, stereo cameras, and ultrasonic radar). In recent years, major automobile manufacturers have actively accelerated the development of autonomous vehicles, most of which rely on a combination of high-end sensors and powerful computing platforms to ensure safety and reliability. However, the widespread adoption of autonomous driving that are not only accurate and robust, but also cost-efficient and lightweight. Thus, developing a safe and efficient system for vehicles equipped only with common sensors is a critical challenge.

Many economical vehicles are now equipped with multiple monocular cameras. These cameras offer a promising alternative that can provide essential visual data at relatively low cost. However, the development of effective vehicle perception using limited camera sensor inputs is constrained by two main factors: (1) the sparsity and lower fidelity of environmental data compared to high-end sensor suites, and (2) the restricted computing capacity of affordable embedded GPUs.

To overcome these limitations, we propose a lightweight autonomous driving system, which balances computational efficiency and perception accuracy. To achieve our goal, we propose a reinforcement learning (RL) based method, which relies on camera sensors available on vehicles. In more detail, the approach integrates the you-only-look-once (YOLO) system [1] and a deep Q network (DQN) model to analyze the environment, and then makes vehicle control decisions (such as acceleration, deceleration, or lane change). We train our model and conduct experiments on the CARLA simulator [2]. The experiment results validate that the proposed approach can effectively make control decisions to enhance the average speed with a low collision rate. The key contributions of this work are twofold.

1) We propose an RL-based method with a complete data extraction pipeline that integrates YOLO and DQN for efficient autonomous driving.
2) We designed a method to convert images into relative information, thereby reducing both the state space and the required computational overhead.
3) The proposed method relies solely on raw data collected by cameras, learning to infer essential parameters for effective training and execution.

In this work, we also provide an analysis on failure cases and discuss potential directions for future improvement.

The remainder of this paper is organized as follows. Section II introduces concepts of YOLO and DQN and presents some related works. Section III describes the proposed system flow and the details of the designed model. Section IV presents our experiments and the training process. Finally, Section V concludes the paper.

II. Preliminaries

A. YOLO and DQN

YOLO is an object detection technology based on supervised learning that is known for its balance of accuracy and speed [3]. This technology divides an input image into multiple grids and utilizes convolutional neural networks to extract features and evaluate the likelihood of each grid containing

an object of a certain class. Grids with a high likelihood for a specific class are combined to form a bounding box for the object, and the model then outputs the coordinates and size of this box. For each image frame, YOLO can detect multiple objects and output their classification labels along with the corresponding bounding box *coordinates*.

DQN is an RL technique derived from Q-learning. The key component of Q-learning is the Q-function, denoted as $Q(s, a)$. It takes a *state* s and an *action* a as inputs. An agent (e.g., the vehicle) uses sensors to perceive its environment as a numerical state and then executes an action to alter that environment. The Q-function evaluates the value (Q-value) of each action for a given state $s \in \mathcal{S}$. The agent selects the action a with the highest Q-value in the current state s:

$$\arg\max_{a \in \mathcal{A}} Q(s, a)$$

After the agent performs the action a, the environmental state transitions to s'. This transition results in a reward, which is used to update the Q-values. Conventional Q-learning algorithms use a Q-table to represent the Q-function, which records the Q-value for each state-action pair. DQN replaces this table with a Deep Neural Network (DNN) that approximates the Q-function, enabling it to solve problems with continuous and high-dimensional state spaces.

B. Related Works

In the literature, autonomous driving systems can be broadly categorized into rule-based and RL-based approaches. Rule-based schemes provide trackable expressions for autonomous driving systems [4], [5], [6], [7], [8]. References [4] and [5] investigate autonomous driving based on model predictive control (MPC) for highway scenarios. The work [4] proposes an approach to determine a lane change plan for maintaining a target velocity. Reference [5] designs an interaction-aware MPC that considers benefits for the overall traffic flow to make lane change decisions. However, these works [4], [5] are only designed for specific scenarios. The research [6] proposes an integrated behavior planning and motion control scheme to support various urban driving scenarios and operations. The work [7] utilizes probabilistic trajectory predictions of surrounding vehicles within a 3D grid to construct a spatio-temporal probability map. Based on this map, the authors propose a spatio-temporal trajectory search method to enable risk-aware continuous decision-making. Reference [8] proposes an autonomous driving framework designed to enhance on-road safety and rule adherence. This framework integrates decision-making and motion control into a single optimal control problem that considers dynamic interactions with surrounding vehicles, pedestrians, road lanes, and traffic signals. However, these works [6], [7], [8] cannot directly extract essential data from camera images.

Reinforcement learning is a promising technology for implementing autonomous driving systems in diverse scenarios without dedicated feature design [9], [10], [11], [12], [13]. The work [9] utilizes an Actor-Critic approach and a DQN model to improve the safety of autonomous lane change maneuvers in static environments. The work [10] proposes a deep reinforcement learning decision-making method based on driving risk fields to address continuous lane-changing and overtaking decision-making problems. The work [11] proposes a Dynamic Option Policy enabled Hierarchical Deep Reinforcement Learning (DOP-HDRL) approach that can break down overtaking maneuvers into several sub-maneuvers and uses a single policy to train and perform them. However, these schemes [9], [10], [11] rely on pre-processed data. On the other hand, some schemes focus on implementing autonomous driving with vision-based systems [13], [14]. The work [14] designs an actor-critic network with an auxiliary network that can leverage real-time measurement information to understand the environment without guidance from demonstrators. The work [13] utilizes YOLO and a DQN to implement car-following control and energy management for a hybrid electric vehicle. However, these works [14], [13] do not consider the task of overtaking.

III. The Proposed Method

Fig. 1 indicates the flow of the proposed method, which operates on a single vehicle, say v. The proposed system consists of an *environment construction module* and a *DQN module*. The vehicle v continuously collects data from its onboard cameras and speedometer. The images captured by the cameras serve as inputs to the environment construction module, which is composed of a *YOLO model* and a *relative relationship inference layer*. After processing inputs, this module extracts the relationships of surrounding objects and structures this information as outputs. In the system, the DQN module has two input sources. The first is from the environment construction module, and the second is the speed measured by the speedometer. Based on these two inputs, the DQN module perceives the environment, learns effective behaviors, and determines vehicle control actions. In the following, we describe these two modules in more detail.

A. Environment Construction Module

In this work, we assume that the vehicle v has three cameras on its front, left, and right sides. In the system, the vehicle v continuously captures images from its cameras to train the YOLO models. Considering that neighboring vehicles captured by the side cameras may exhibit barrel distortion, the vehicle v prepares two YOLO models: one for the front camera and another for the side cameras. In this module, the relative relationship inferring layer infers (i) the relative distance and (ii) position of neighboring vehicles. First, to infer relative distance to surrounding vehicles, given the camera's focal length f, the vehicle v's width W_{actual}, and the pixel width of a neighboring vehicle's bounding box (from the YOLO model) $W_{\text{pixel}}(n)$, the distance $D_{\text{est}}(n)$ between the vehicle v and a neighboring vehicle n can be estimated as:

$$D_{\text{est}}(n) = \frac{f \times W_{\text{actual}}}{W_{\text{pixel}}(n)}$$

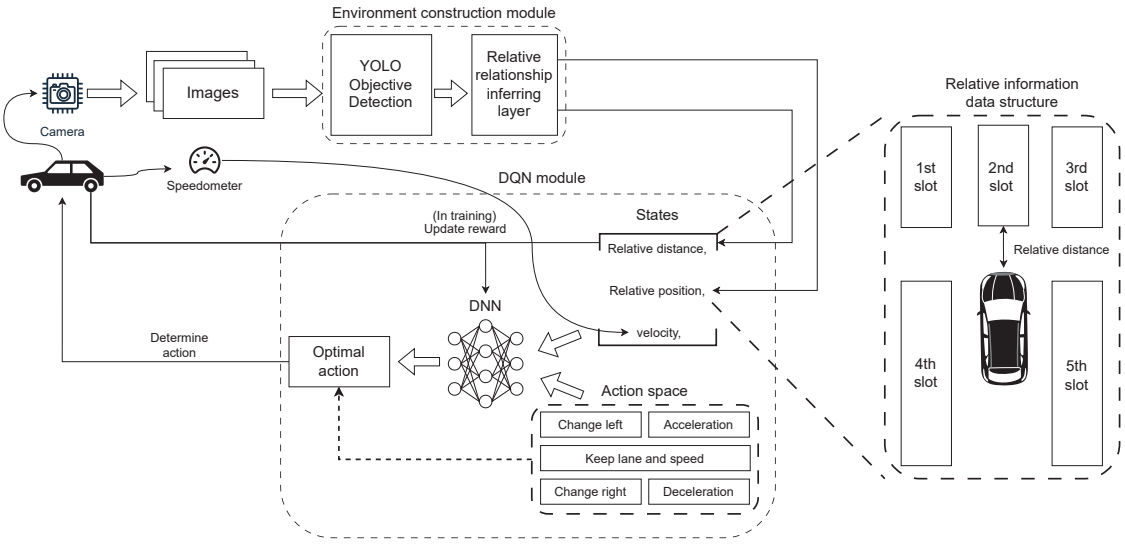

Fig. 1. The proposed system flow.

The actual width of a neighboring vehicle is assigned based on its YOLO classification, which itself is trained on a dataset annotated with the physical widths of different vehicle types.

Second, to infer position, the vehicle v distinguishes positions of surrounding vehicles in different lanes. More specifically, the vehicle v labels lanes sequentially from left to right with an ascending index starting from 1. Then, using the coordinates provided by YOLO and the cameras' mounting locations, the vehicle v can determine the lane of each detected vehicle and map neighboring vehicles into one of five relative position slots. These five slots correspond to the front-left, front-center, front-right, left-rear, and right-rear positions. The final data structure is illustrated as the *relative information data structure* in Fig. 1. In this work, the vehicle v will identify all neighbors. However, vehicle v will further record five vehicles, denoted as $\hat{n}_{s1}$, $\hat{n}_{s2}$, .., $\hat{n}_{s5}$, in those five relative position slots that are closest to v.

B. DQN Module

In the proposed scheme, the DQN module receives the current state of the environment from the environment construction module and the speedometer. Specifically, we define the state s includes five elements:

- I_{lane}: the lane index where the vehicle is located.
- V: the vehicle v's speed.
- $[D_{\text{est}}(\hat{n}_{s1}), D_{\text{est}}(\hat{n}_{s2}), .., D_{\text{est}}(\hat{n}_{s5})]$: the relative distance to the neighbor vehicle $\hat{n}_i$ in the relative position slot i.
- $[S_{\text{blind}}(\hat{n}_{s1}), S_{\text{blind}}(\hat{n}_{s2}), .., S_{\text{blind}}(\hat{n}_{s5})]$: five flags to indicate whether the neighbor vehicle $\hat{n}_i$ may be hidden in a blind spot.
- $[D_{\text{hc}}(\hat{n}_{s1}), D_{\text{hc}}(\hat{n}_{s2}), .., D_{\text{hc}}(\hat{n}_{s5})]$: the coordinates of bounding boxes of the neighbor vehicle $\hat{n}_i$.

Note that $S_{\text{blind}}(\hat{n}_{s1}), \ldots, S_{\text{blind}}(\hat{n}_{s3})$ are always set to 0. However, the flags $S_{\text{blind}}(\hat{n}_{s4})$ and $S_{\text{blind}}(\hat{n}_{s5})$, corresponding to the closest vehicles in relative position slots 4 and 5, may take values of either 0 or 1. Specifically, these flags indicate whether the closest vehicles in the left-rear and right-rear positions fall within the blind spot regions of vehicle v. Each flag is determined by comparing the real-time bounding boxes generated by YOLO with a set of predefined spatial boundaries, which are calibrated in advance according to the fixed field of view of each camera. This mechanism enables rapid identification of whether an adjacent lane is occupied, even in cases where vehicles are not visually detected due to blind spot occlusion. If a blind spot is considered occupied, the corresponding flag is set to 1. Also note that for vehicles located entirely within blind spot regions, their bounding boxes cannot be reliably detected.

Based on the current state, the DQN module selects an action $a \in \mathcal{A}$ that it estimates to be optimal. In our design, the action space $\mathcal{A}$ includes five actions: 1) acceleration (ACC), 2) deceleration (DEC), 3) change to the left lane (LFT), 4) change to the right lane (RIT), 5) and keep current lane and speed (KEP). In practice, the ACC and DEC actions are performed while maintaining the current lane.

After the action is decided, the vehicle v performs the control command. Then, the environment will transition to a new state s' and return a reward r. In this work, the vehicle v's speed cannot exceed the speed limit V_{max} of the highway. But, a positive reward can be higher if the speed of vehicle is higher. Given a speed reward constant K_{speed}, we define the speed reward as

$$R_{\text{eff}} = K_{\text{speed}}^{\left(\frac{V}{V_{\text{max}}}\right)} - 1 \tag{1}$$

On the other hand, to ensure safety, a negative reward is designed for risky behaviors. First, to maintain a safe following distance D_{safe}, we define a penalty P_{cls} when the distance to the front vehicle (i.e., the vehicle $\hat{n}_{s2}$) is shorter than D_{safe}.

TABLE I
DQN HYPERPARAMETER CONFIGURATION.

Parameters	Values
Hyperparameters	
Discount Factor (γ)	0.99
Optimizer	Adam
Learning Rate (α)	0.0001
Target Update Frequency	2000
Initial Epsilon ($\varepsilon_{initial}$)	1.0
Final Epsilon (ε_{final})	0.1
Epsilon Decay Period	20000
DQN Model	
Input layer	128
Hidden layer	128×2
Output layer	128

The definition of P_{cls} is expressed as:

$$P_{cls} = K_{\text{penalty}} \times \frac{\exp(b \times D_{\text{est}}(\hat{n}_{s2})) - \exp(b \times D_{\text{safe}})}{\exp(b \times D_{\text{safe}}) - 1 + 0.001} \quad (2)$$

In Eq. (2), K_{penalty} is a constant value, and b is the distance reduction rate. This distance reduction rate b is derived from the previous relative distance $D'_{\text{est}}(\hat{n}_{s2})$ and the current relative distance $D_{\text{est}}(\hat{n}_{s2})$:

$$b = \frac{D'_{\text{est}}(\hat{n}_{s2}) - D_{\text{est}}(\hat{n}_{s2})}{D'_{\text{est}}(\hat{n}_{s2})} \times 0.1 \quad (3)$$

Moreover, to encourage deceleration in this scenario, the reward is defined by:

$$R_{\text{sp}} = \begin{cases} R_{\text{eff}}, & D_{\text{est}}(\hat{n}_{s2}) \geq D_{\text{safe}}, \\ -P_{cls}, & D_{\text{est}}(\hat{n}_{s2}) < D_{\text{safe}} \end{cases} \quad (4)$$

Moreover, when the vehicle reaches the destination, the system will obtain a completion reward R_{finish}. If the vehicle suffers a collision, there will be a collision penalty P_{cll}. Consequently, the reward function is defined as

$$R_{\text{total}} = \begin{cases} R_{\text{sp}} + R_{\text{finish}}, & \text{if destination is reached} \\ -P_{\text{cll}}, & \text{if a collision occurs} \\ R_{\text{sp}}, & \text{otherwise} \end{cases} \quad (5)$$

This reward function balances the objectives of efficiency and safety, guiding the DQN agent to make reasonable lane-changing and speed adjustments. Note that the reward mechanism is only active during training phase. When operation, the vehicle v relies on the trained model to determine its actions.

IV. EXPERIMENTS

We adopt the open-source CARLA simulator [2] as our training and evaluation platform. The CARLA environment features surrounding vehicles distributed across different lanes, at varying distances, and traveling at different speeds. To evaluate the performance of the proposed DQN-based decision-making model, we design and implement four baseline strategies for comparison:

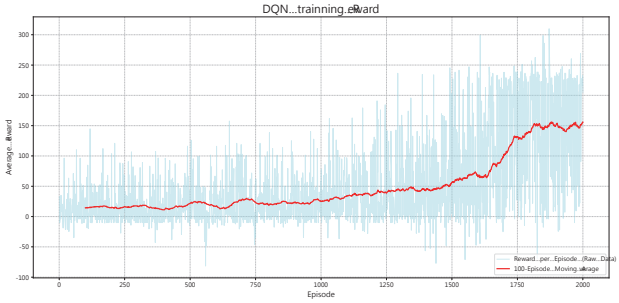

Fig. 2. Total reward curve during the DQN model's training process.

- **Pure Lane-Following**: In this baseline strategy, the vehicle strictly follows its current lane without performing any lateral lane changes.
- **Rule-Based Lane Changing**: This strategy enables the vehicle to perform basic lane changes. It will change to the right lane only if the left lane is unavailable and the right lane is viable. If neither adjacent lane offers safe conditions, the vehicle remains in its current lane and follows the leading vehicle.
- **CARLA Autopilot**: This strategy utilizes the built-in autopilot feature provided by the CARLA simulator, whose behavior is pre-designed and calibrated by the CARLA development team.
- **Literature-Based**: To situate our research within the current academic context, we referenced, adapted, and re-implemented the integrated decision-making and control method proposed in [6].

A. Model Training

To train our model, we generate a 400-meter-long, one-way, three-lane highway environment in CARLA. At the beginning of each training episode, the vehicle's initial speed is set to 80 km/h, and its initial lane is randomly selected from the left, middle, or right lane. This ensures that the agent begins learning from varied lateral positions and avoids developing a bias toward any specific lane. For the surrounding vehicles, their speeds are randomly selected from a predefined set of values: 60, 65, or 70 km/h, and maintained consistently throughout the episode. To create overtaking opportunities, these vehicles are randomly generated ahead of the vehicle and distributed across different lanes. The number of surrounding vehicles is randomly set between one and three to simulate varying traffic densities. In this configuration, the vehicle travels faster than the surrounding vehicles, allowing them to be regarded as slow-moving obstacles that reduce travel efficiency. This setup provides the necessary conditions for the vehicle to practice overtaking decisions in order to gain greater reward. The hyperparameters of the training process and the configuration of the DQN model are presented in Table I. The learning progression of our DQN model is illustrated in Fig. 2, which demonstrates that the total reward gradually converges to a relatively high level during training.

TABLE II
OVERALL PERFORMANCE COMPARISON OF ALL STRATEGIES UNDER DIFFERENT TRAFFIC DENSITIES.

Strategy	One-Vehicle		Two-Vehicle	
	Collision Rate	Avg. Speed (km/h)	Collision Rate	Avg. Speed (km/h)
Pure Lane-Following	0%	91.92	0%	83.80
Rule-Based Lane Changing	0%	99.56	0%	88.46
CARLA Autopilot	0%	104.01	0%	87.79
Literature-Based	17.6%	109.38	34.78%	108.63
Proposed DQN Model	0%	95.95	0.89%	90.62
Strategy	Three-Vehicle		Overall Average	
	Collision Rate	Avg. Speed (km/h)	Collision Rate	Avg. Speed (km/h)
Pure Lane-Following	0%	75.49	0%	83.34
Rule-Based Lane Changing	1.82%	80.65	0.61%	89.51
CARLA Autopilot	0%	75.31	0%	89.15
Literature-Based	64.39%	107.24	38.9%	108.7
Proposed DQN Model	9.8%	85.14	3.32%	90.98

B. Results

Table II presents the experimental results. These results demonstrate that the proposed scheme improves traveling efficiency while sustaining a lower collision rate. As indicated by the overall data, a common trade-off exists between efficiency and safety across different autonomous driving decision strategies. The pure lane-following strategy and the CARLA autopilot mode ensure driving safety with a 0% collision rate, but at the cost of significantly reduced traffic efficiency as density increases. The literature-based optimization strategy represents the opposite extreme. It prioritizes high-speed travel, resulting in excellent efficiency, but also leads to an overall collision rate of 38.9%.

When the number of surrounding vehicles increase from one to three, the average speed achieved by the CARLA autopilot method decreased by approximately 27.6%, whereas our proposed approach experienced a reduction of only about 11.3%. In both two- and three-vehicle scenarios, the average speed of our proposed approach remained higher than those of other safety-focused baseline strategies. This indicates that, compared to rule-based or conservative logic, the proposed scheme can learn a more flexible decision-making policy, dynamically assess the environment, and identify and utilize gaps to maintain traffic efficiency. However, this improvement in efficiency comes at the cost of a slight reduction in safety. Our proposed model exhibits an overall collision rate of 3.32%, with the rate rising to 9.8% in the three-vehicle scenario, revealing certain instabilities in its decision-making under high-risk conditions. To further enhance safety, in the following article, we analyze the failure cases of our approach to identify areas for future improvement.

C. Analysis

We conducted an in-depth analysis of the collision incidents. Our examination revealed several key contributing factors. The most significant issue was the model's difficulty in accurately predicting the states of vehicles approaching from the left-rear and right-rear. This single factor was responsible for approximately 73% of all collisions. The second major factor stemmed from the design of our reward function, which accounted for about 20% of the failures. The current design prioritizes efficiency, sometimes neglecting critical safety considerations. For example, in extreme scenarios with three vehicles driving abreast, the agent would still attempt a lane change to seek a higher reward. This behavior often led to a collision when no safe gap was available. Finally, perception errors from the YOLO system contributed to the remaining 6% of collisions. On rare occasions, the system failed to detect a nearby vehicle, resulting in an unsafe maneuver. Based on our analysis, future improvements should focus on enhancing rear-side vehicle prediction, increasing the safety weight in the reward function, and improving perception reliability.

V. CONCLUSION

In this work, we developed an RL-based scheme for autonomous drive decision-making that includes a comprehensive data extraction process. The proposed method integrates YOLO v11 and a DQN model. We utilize the output of the YOLO model to construct the relative position and distance between the vehicle and surrounding vehicles. Based on this relative information and the vehicle's velocity, the DQN model determines speed control and lane-changing actions to enhance traveling efficiency and avoid potential risks. The experimental results show that the proposed approach can be effectively implemented on multi-lane highways built with the CARLA simulator, and they reveal the trade-off between efficiency and safety for different driving strategies. Consequently, we analyzed the reasons for the failure cases of our proposed approach. For future work, we can design a more lightweight model that fuses vision-based object detection and low-level control to reduce computational complexity for realistic, high-speed scenarios. Meanwhile, a novel mixed-model approach could be explored to address the safety issue.

REFERENCES

[1] R. Khanam and M. Hussain, "YOLOv11: An overview of the key architectural enhancements," 2024. [Online]. Available: https://arxiv.org/abs/2410.17725

[2] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. of Conference on Robot Learning*, 2017.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] H. Bey, F. Dierkes, S. Bayerl, A. Lange, D. Faßender, and J. Thielecke, "Optimization-based tactical behavior planning for autonomous freeway driving in favor of the traffic flow," in *Proc. of IEEE Intelligent Vehicles Symposium (IV)*, 2019.

[5] X. Zhang, S. Zeinali, H. Wen, and G. Schildbach, "MOBIL-based traffic prediction and interaction-aware model predictive control for autonomous highway driving," *Control Engineering Practice*, vol. 164, p. 106434, 2025.

[6] H. Liu, K. Chen, Y. Li, Z. Huang, J. Duan, and J. Ma, "Integrated behavior planning and motion control for autonomous vehicles with traffic rules compliance," in *Proc. of IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2023.

[7] D. Li, S. Cheng, S. Yang, W. Huang, and W. Song, "Multi-step continuous decision making and planning in uncertain dynamic scenarios through parallel spatio-temporal trajectory searching," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8282–8289, 2024.

[8] H. Liu, K. Chen, Y. Li, Z. Huang, M. Liu, and J. Ma, "UDMC: Unified decision-making and control framework for urban autonomous driving with motion prediction of traffic participants," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 5, pp. 5856–5871, 2025.

[9] B. Safae, B. Jaouad, and N. N. S, "Deep reinforcement learning actor-critic algorithm based autonomous lane change decisions in static environments," in *Proc. of International Conference on Intelligent Computing in Data Sciences (ICDS)*, 2024.

[10] S. Wu, D. Tian, X. Duan, J. Zhou, D. Zhao, and D. Cao, "Continuous decision-making in lane changing and overtaking maneuvers for unmanned vehicles: A risk-aware reinforcement learning approach with task decomposition," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 4, pp. 4657–4674, 2024.

[11] S. Singh Lodhi, N. Kumar, and P. Kumar Pandey, "Dynamic option policy enabled hierarchical deep reinforcement learning model for autonomous overtaking maneuver," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 4, pp. 5018–5029, 2025.

[12] J. Chen, S. E. Li, and M. Tomizuka, "Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5068–5078, 2022.

[13] X. Tang, J. Chen, K. Yang, M. Toyoda, T. Liu, and X. Hu, "Visual detection and deep reinforcement learning-based car following and energy management for hybrid electric vehicles," *IEEE Transactions on Transportation Electrification*, vol. 8, no. 2, 2022.

[14] S. Chen, M. Wang, W. Song, Y. Yang, Y. Li, and M. Fu, "Stabilization approaches for reinforcement learning-based end-to-end autonomous driving," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 4740–4750, 2020.