

PaaS: Planning as a Service for reactive driving in CARLA Leaderboard

Nhat Hao Truong, Huu Thien Mai, Tuan Anh Tran,
Minh Quang Tran, Duc Duy Nguyen, Ngoc Viet Phuong Pham

Abstract—End-to-end deep learning approaches have been proven to be efficient in autonomous driving and robotics. By using deep learning techniques for decision-making, those systems are often referred to as a black box, and the result is driven by data. In this paper, we propose PaaS (Planning as a Service), a vanilla module to generate local trajectory planning for autonomous driving in CARLA simulation. Our method is submitted in International CARLA Autonomous Driving Leaderboard (CADL), which is a platform to evaluate the driving proficiency of autonomous agents in realistic traffic scenarios. Our approach focuses on reactive planning in Frenet frame under complex urban street’s constraints and driver’s comfort. The planner generates a collection of feasible trajectories, leveraging heuristic cost functions with controllable driving style factor to choose the optimal-control path that satisfies safe traveling criteria. PaaS can provide sufficient solutions to handle well under challenging traffic situations in CADL. As the strict evaluation in CADL Map Track, our approach ranked 3rd out of 9 submissions regarding the measure of driving score. However, with the focus on minimizing the risk of maneuver and ensuring passenger safety, our figures corresponding to infraction penalties dominate the two leading submissions by 20 percent.

Index Terms—autonomous driving, dynamic control, trajectory planning, carla leaderboard, simulation

I. INTRODUCTION

A. Motivation

Autonomous driving is a rapidly growing field that has the potential to revolutionize transportation by providing safer, more efficient, and convenient transportation for people around the world. One of the most critical aspects of autonomous driving is motion planning, which involves determining the optimal path for a vehicle to follow to reach its destination while avoiding obstacles and adhering to traffic rules. Motion planning is a complex task that requires sophisticated algorithms, advanced sensors, and powerful computing resources. Furthermore, with the rise of autonomous vehicles, there is a growing need for robust and reliable algorithms that can safely and efficiently navigate complex driving scenarios. One of the key challenges in developing such algorithms is the lack of a standardized evaluation platform that can be used to compare the performance of different algorithms. The CARLA (Car Learning to Act) Leaderboard Challenge [24] is an open competition designed to address this challenge by providing a common platform for evaluating the performance of autonomous driving algorithms. The challenge is based on the CARLA simulator [25], an open-source software platform that enables researchers and developers to test and develop their algorithms in a safe and controlled environment. In this

paper, our main focus is on the motion planning stage in autonomous driving (see Section II). Moreover, we briefly introduce our approaches applied in CADL in Section III. Finally, the evaluation of our result on CADL challenge compared to other competitors will be shown in Section IV.

B. Related work

Motion planning is a crucial component of autonomous driving that involves generating a safe trajectory or path for the vehicle to follow in order to navigate through the environment. Many existing methods have been proposed to address this task, each with its own strengths and limitations.

Sampling-based methods randomly sample the environment and attempt to connect the samples, based on agent kinematics, to create a path, known as probabilistic sampling-based algorithms. Examples of these methods include Rapidly-exploring Random Trees (RRT) [1], [2] and Probabilistic Roadmaps (PRM) [3], [4]. Although the best path produced by these algorithms is often far from optimal, in [5], the authors proposed the Rapidly-exploring Random Graphs (RRG) algorithm to ensure the returned solution is asymptotically optimal. The authors introduced RRT*, which is based on RRG to construct a tree from an existing graph, while also improving the cost-to-come value of the neighbor vertices for sampling points [6]. Although these methods are computationally efficient and can handle unknown environments, their performance suffers from random distribution, making them insufficient to apply in well-defined environments, such as urban or highway driving. To take advantage of structured environments, lattice planners are introduced in reactive motion planning. These methods produce a finite set of trajectories that sample over the spatio-temporal evolution space, satisfying agent kinematics, dynamics, and environmental constraints, such as obstacles, lane markings, and traffic signals. In [7], [8], the authors decouple lateral and longitudinal movements to generate a dynamics profile under Frenet frame to generate a set of trajectories, defined according to the driving state of the agent. Following the generation of the trajectory set, obstacle representation is a crucial aspect of ensuring safe and efficient navigation. Obstacle representations, such as circular [9], triangular [10], and rectangular [11], are proposed to improve coverage and computational efficiency.

Optimization-based methods use optimization techniques to find the optimal path that satisfies a set of constraints, such as minimum time or energy consumption. Model Predictive Control (MPC) [11]–[13] and Differential Dynamic

Programming [16], [17] are examples of optimization-based methods. In urban driving scenarios, the automated vehicle needs to solve a joint optimization of neighboring vehicles (NV) costs over a receding horizon. To handle such task, a mixed integer quadratic programming (MIQP) formulation of the MPC is presented in [14] to handle the multi-input multi-output (MIMO) control problem with indicator variables and disjunctive constraint. Based on this idea, [15] suggests adaptive interactive mixed integer MPC (aiMPC), which captures dynamics and collision avoidance constraints to predict the trajectory of the ego and NV in MPC horizon.

AI-based methods are another category of motion planning algorithms that use machine learning, deep learning, or reinforcement learning algorithms to learn mappings between the current state of the vehicle, sensors, environments, and the optimal action to take, namely end-to-end autonomous driving. These methods can handle complex and dynamic environments, but they require large amounts of data and time to train. [18] vectorizes the sensors' information with multiple modalities, such as OpenDRIVE HD Map, Radar, LiDAR, and front camera image, into same-sized inputs. With different sensor domains, CNN-based fusion layers, or Transformer-based ones [19]–[21], are proposed to offer a chance for these sensor data to find relations with each other. [22], [23] offer deep reinforcement learning (DRL) techniques that learn state-action policies from offline replay buffer (RB) (recorded from expert data) and online exploration agent from online RB.

II. MOTION PLANNING METHOD

In this section, the reactive motion planning task is formulated in Frenét frame by the decoupling of lateral and longitudinal movements to generate dynamics profiles for the agent. Furthermore, the constraints in urban driving scenarios are explained in detail.

A. Trajectory Planning in Frenét Frame

The work in this section is based on the proposal of using Frenét Frame method in [7]. Following this paper, we generate the trajectory sets using multiple terminal conditions to follow the center lines on the road (reference lines), then the conversion from Frenét frame to Cartesian frame is performed to spatio-temporal collision checking with other surrounding agents. As the most advantage of using Frenét frame in the trajectory generation stage is that the reference lines are standardized in the same form of a straight line with the difference in road boundaries. Nevertheless, the side effects of the usage of this method are mentioned in [26]. To avoid those disadvantages, the sampling of trajectory set is executed in Frenét frame using a simplified kinematic model and driving comfort constraints, then the valid set is evaluated with environment constraints (detailed in Section II-B) in Cartesian space.

The Frenét frame is composed of the tangential and normal vector to model the reference curve in Cartesian coordinate. In motion planning using Frenét frame, the tracking problem is composed of two factors: lateral and longitudinal offset along

the temporal dimension, namely $d(t)$ and $s(t)$ respectively. The movement profiles of our agent are defined in lateral direction $D(t) = [d(t), \dot{d}(t), \ddot{d}(t), \dddot{d}(t)]$ and longitudinal one $S(t) = [s(t), \dot{s}(t), \ddot{s}(t), \dddot{s}(t)]$. We use quintic polynomials for the generation of $D(t)$ in lateral space and $S(t)$ in longitudinal space within time interval T .

The generated trajectory needs to satisfy the jerk-optimal constraints. Therefore, the total jerk is the accumulation of jerk over the planning horizon, applied in both lateral jerk polynomial $\ddot{s}(t)$ and longitudinal jerk $\ddot{d}(t)$

$$J_s = \int_0^T \ddot{s}^2(t) dt; \quad J_d = \int_0^T \ddot{d}^2(t) dt. \quad (1)$$

Furthermore, to satisfy the comfort movement, the trajectory must obey the conditions that

$$\ddot{s}(t) < J_{max}, \quad \ddot{d}(t) < J_{max}, \quad \forall t \in [0, T], \quad (2)$$

where J_{max} is the maximum comfort jerk.

In urban driving, many modes are selected and executed while traveling to ensure the continuity of movement. These modes, in descending order of aggressiveness, are merging, following, velocity-keeping in free space, and stopping. The behavior layer evaluates the consequences of mentioned modes to make decisions. In the scope of this paper, we use solely the total cost function to select the most optimal trajectory and the driving style of our agent is configurable by parameters regarding these functions.

Each mode has two cost functions related to lateral and longitudinal space in Frenét frame. The start condition is the current state of the ego agent, which is retrieved from sensors (explained in Section III-A). The longitudinal and lateral terminal condition is given on the selected mode, namely TC . The evaluation of trajectory corresponding to terminal condition is formulated by cost function C .

We define the common terminal state and cost function for longitudinal trajectory generation as

$$TC_{ij} = [s_j, \dot{s}_j, \ddot{s}_j, T_i], \quad (3)$$

$$C_s = k_j J_s + k_t T_i + k_s (s_d - s_j)^2, \quad (4)$$

with s_d is the target distance in the longitudinal frame.

In the stopping mode, the agent has to stop its movements before the longitudinal position of the traffic sign (or red light), which is s_d . Following (3), the terminal condition with the longitudinal difference Δs_j is

$$TC_{ij} = [s_j, \dot{s}_j, \ddot{s}_j, T_i] = [s_d - \Delta s_j, 0, 0, T_i]. \quad (5)$$

In following mode or cruise control, we maintain a safe distance from the preceding vehicle. The state corresponding to the preceding vehicle $S_{pv} = [s_{pv}(t), \dot{s}_{pv}(t), \ddot{s}_{pv}(t)]$. We define the predicted terminal state of the target vehicle in the prediction horizon as $\hat{S}_{pv}(T) = [\hat{s}_{pv}(T), \hat{\dot{s}}_{pv}(T), \hat{\ddot{s}}_{pv}(T)]$, which is retrieved from the Trajectory Prediction module (detailed in Section III-C). Applying into (3), the desired states of our agent are as follows

$$\begin{aligned}
s_j &= \hat{s}_{pv}(T) - [D_0 + \tau \hat{s}_{pv}(T)], \\
\dot{s}_j &= \hat{s}_{pv}(T) \pm \Delta \dot{s}_j - \tau \hat{s}_{pv}(T), \\
\ddot{s}_j &= \hat{s}_{pv}(T),
\end{aligned} \tag{6}$$

with D_0 is the constant safety distance with the other agent, followed by the gap defined by constant time-to-collision τ .

In merging mode, it requires the agent to keep an appropriate distance from both preceding and following vehicles. Using the predicted trajectories of preceding $\hat{S}_{pv}(t)$ and following $\hat{S}_{fv}(t)$ vehicles, we can define the target point in the terminal condition (3) as

$$s_j = \frac{1}{2} [\hat{s}_{pv}(T) + \hat{s}_{fv}(T)]. \tag{7}$$

Corresponding to velocity-keeping mode, we keep our agent to maintain its velocity without specific s_d . Therefore, we use quartic polynomials for the generation of $S(t)$ as shown in Fig. 1. The terminal state, without target longitudinal distance, is formulated as follows, where \dot{s}_d is the desired velocity

$$TC_{ij} = [\dot{s}_d \pm \Delta \dot{s}_j, 0, T_i], \tag{8}$$

$$C_v = k_j J_s + k_t T_i + k_s (\dot{s}_d - \dot{s}_j)^2. \tag{9}$$

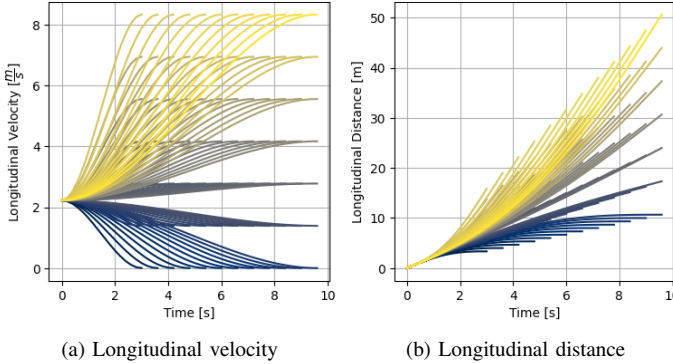


Fig. 1. The longitudinal profile based on velocity tracking is shown as (a) longitudinal velocity, (b) corresponding longitudinal distance. Each solution is color-mapped by its longitudinal target velocity \dot{s}_j . $\Delta v = 5 \text{ km/h} = 1.38 \text{ m/s}$ and $\Delta T = 0.2 \text{ s}$

Regarding lateral space, we generate the trajectory set in lateral space with multiple offset d_j (with the difference Δd) and time interval T_i in the terminal condition. The final state is when our agent successfully tracks the reference line ($d_T = 0$). The movement profiles of lateral movement are shown in Fig. 2. Similar to longitudinal space, the terminal condition is defined as

$$TC_{ij} = [d_j, 0, 0, T_i], \tag{10}$$

$$C_d = k_j J_d + k_t T_i + k_s d_j^2. \tag{11}$$

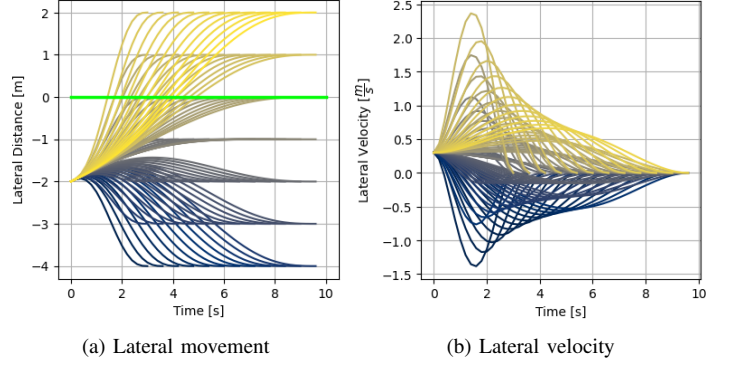


Fig. 2. Illustrations of (a) lateral movement, (b) corresponding lateral velocity. The green line is the target lateral offset with the reference line (at zero). Multiple solutions in (a) are generated with multiple lateral and time interval differences, with $\Delta d = 1 \text{ m}$ and $\Delta T = 0.2 \text{ s}$ respectively, color-mapped by its lateral offset d_j .

B. Environment Constraints

In our urban driving task, the constraints include the collision boundaries of NV and pedestrians, road barriers, traffic lights, and road signs.

First of all, in CARLA simulation, traffic lights and road signs' positions can be retrieved from the provided map. However, the state of the traffic light (green, yellow, red) can only be identified visually by the cameras (by methods explained in Section III-B). We convert the queried positions on the global frame and their states as terminal conditions in Frenét frame in the motion planning module, as described in the previous section.

Furthermore, we execute collision checking on the sampling set of trajectories on Cartesian space. Each trajectory is denoted by $\chi = [x(t), y(t), \theta(t), v(t), a(t)]$, where $[x(t), y(t)]$ denotes the position, $\theta(t)$ represents the orientation angle, $v(t)$ indicates the velocity and $a(t)$ corresponds the acceleration of trajectory points with reference to (w.r.t) our ego agent. Through collision checking, we assess a set of feasible trajectories denoted as Φ , resulting in the formation of a collision-free set known as Φ^* .

In order to enhance the computational efficiency of collision checking, we adopt a circle representation that encompasses the rectangular shape of the vehicles, including our agent. This approach involves employing an odd number of circles, denoted as n , to cover the overall shape. As illustrated in Fig. 3, there are two types of representation that are being used for collision checking: three-disk and five-disk models. While both models are able to cover the whole rectangular section, there is a trade-off between computational efficiency by using the three-disk model and precise representation by using the latter model, which has smaller circle coverage by 9.02%. Based on our experiment, we decide to use the three-disk model to have the computational advantage, with time complexity being $\mathcal{O}(n^n)$. Each disk in the model could be constructed by 2D center point and radius by $D = [x, y, R]$ in the relative coordinate of the vehicle. Therefore, we define

that

$$D_1 = [0, 0, R_{max}], D_{2,3} = \left[\pm \frac{l}{3}, 0, R_{max} \right], \quad (12)$$

where D_1, D_2, D_3 are the models of center, front, and back disks respectively. w is width, l is length of the vehicle, and $R_{max} = \left(\frac{w^2}{4} + \frac{l^2}{36} \right)$.

Within 2D space, the collision is violated when

$$\|p_{ego}^i - p_{nv}^j\| < R_{ego} + R_{nv}, i, j = 1, \dots, 3, \quad (13)$$

with p_{ego}^i is the center location of i th disk of ego vehicle and p_{nv}^j is the center location of j th disk of NV. R_{ego}, R_{nv} are disk radiuses of ego vehicle and NV respectively.

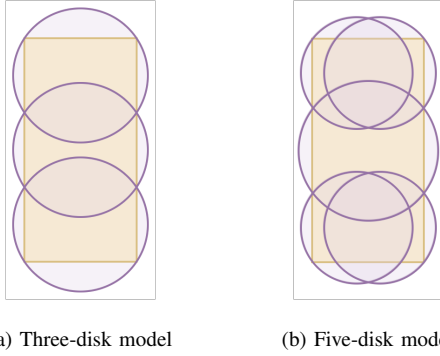


Fig. 3. Disk representation (purple) of the rectangular vehicle (orange). By using more disks, the rectangular could be represented more precisely. However, there will be computational overhead in the collision-checking process.

In real scenarios, the collision needs to be checked in three dimensions, including 2D space and additional dimension of time, within the planning time horizon t_h . At every future timestep t_k over t_h , (13) must not be satisfied to consider a trajectory collision-free. The formulation of collision violation, in this case, is defined as

$$\|p_{ego}^{i,t_k} - p_{nv}^{j,t_k}\| < R_{ego} + R_{nv}, \forall t_k \in [0, t_h], i, j = 1, \dots, 3. \quad (14)$$

Following that, by utilizing the collision-free set Φ^* , we ensure the executability of all feasible trajectories for the ego vehicle. Next, we determine the optimal solution χ^* by considering the trajectory with the lowest cost value. It is important to note that in this particular case, the term "optimal" does not refer to the concept of optimality in trajectory optimization problems. Instead, it signifies the selection of the most favorable feasible trajectory from the set of sampled trajectories. By continually replanning the optimal trajectory within a fixed time interval, the module guarantees consistency in the resulting trajectory over time.

III. CARLA LEADERBOARD APPROACH

In CADL, there are two tracks to submit the solution, namely SENSORS and MAPS tracks. In SENSORS, the set of sensors provided includes GNSS, IMU, LiDAR, RADAR,

RGB camera (limit to 4 units), and Speedometer. Similarly, the MAPS track provides the same set with an additional OpenDRIVE map. Our approach is tested and submitted on MAPS track. In this section, because of the limited scope of this paper, we briefly introduce the methods, without digging into details, that we use to handle the sensor signals.

The flow of PaaS, which includes the Motion Planning module in Section II, as well as the Localization, Perception, and Trajectory Prediction modules described in this section, is presented in Algo. 1. The initial steps involve extracting semantic information about the surrounding environment (Line 1 to 4). This extracted information is then utilized in the motion planning modules, resulting in the generation of a set of feasible and collision-free trajectories (Line 5 to 6). Finally, each trajectory is evaluated using the cost function, leading to the selection of the optimal trajectory (Line 7).

Algorithm 1 Process of PaaS in CADL

Inputs: Navigational sensor signals Ψ ; Camera images I ; Point cloud PC ; Reference path ξ

Output: χ^* //optimal trajectory

```

1:  $S_{mo} \leftarrow \text{MovingObjectDetection}(PC)$ 
2:  $Z_{ego} \leftarrow \text{TrafficSignDetection}(I)$ 
3:  $\hat{X} \leftarrow \text{Localization}(\Psi)$ 
4:  $\hat{\Pi} \leftarrow \text{TrajectoryPrediction}(S_{mo})$ 
5:  $\Phi \leftarrow \text{MotionPlanning}(\xi, \hat{X}, Z_{ego})$ 
6:  $\Phi^* \leftarrow \text{CollisionCheck}(\Phi, \hat{\Pi})$ 
7:  $\chi^* \leftarrow \min(\Phi^*)$  //select path with lowest cost

```

A. Localization

To retrieve an estimated state of ego vehicle

$$\hat{X} = [\hat{x}, \hat{y}, \hat{\theta}, \hat{v}, \hat{a}], \quad (15)$$

the method [28] relies on Kalman Filtering (KF), which uses navigational sensor signals

$$\Psi = [\lambda, \phi, \varphi, va_x, va_y, va_z, al_x, al_y, al_z, \psi], \quad (16)$$

where latitude λ , longitude ϕ , altitude φ are obtained from GNSS information, angular velocity $[va_x, va_y, va_z]$, linear acceleration $[al_x, al_y, al_z]$, yaw compass ψ are derived from IMU sensor.

B. Perception

First of all, we use the point cloud data PC from the LiDAR sensor to handle the detection of surrounding vehicles and pedestrians. The whole process is described as Moving Object Detection. We apply a pre-trained model from [29] to predict the bounding boxes of surrounding vehicles and pedestrians, called moving obstacles. The detection outputs are objects' 3D shapes $[h_{mo}, w_{mo}, l_{mo}]$, 3D positions $[x_{mo}, y_{mo}, z_{mo}]$, yaw angles ψ_{mo} , and the detection confidences. Subsequently, we apply the KF to track and predict velocity v_{mo} , acceleration a_{mo} of the moving obstacles. The state that describes a moving obstacle is

$$S_{mo} = [x_{mo}, y_{mo}, \psi_{mo}, v_{mo}, a_{mo}]. \quad (17)$$

On the other hand, the detection and recognition of traffic lights, and traffic signs are executed from the front camera's images, following the detection model in [30]. Firstly, we collect the training images from our test run in CARLA simulator. Secondly, we re-train the model using the approach of transfer learning. Eventually, with the output of the trained model, which are 2D location including coordinates and dimensions $P_{im} = [x, y, w, h]$ on image and the recognition of traffic light state, traffic sign S_{sign} , we are able to map P_{im} in image coordinate into P_{ego} , which is the relative position w.r.t ego vehicle's coordinate, using simple kinematic transformation and OpenDRIVE map. The final output state of the traffic light/sign w.r.t the ego vehicle will be

$$Z_{ego} = [P_{ego}, S_{sign}]. \quad (18)$$

Furthermore, to improve efficiency, we extend the model to detect junctions that contain traffic lights to avoid false positive detections. Illustrated in Fig. 4, our autonomous vehicle advances towards the intersection, identified by the blue bounding box and in conjunction with the detected traffic light. Incorporating this comprehensive information, the algorithm effectively executes trajectory planning while ensuring compliance with traffic regulations.

C. Trajectory Prediction

Based on the dynamic state of the moving obstacle S_{mo} and OpenDRIVE map given in CARLA simulator, we can predict its future trajectory in the planning horizon.

The original map topology in CARLA contains the tuple of pairs of waypoints located either at the beginning or end point of a road. Because each road has a different length, we create a dense graph to represent the given topology with a fixed sampling distance. An R-Tree is additionally used for graph index querying in spatial dimensions. At first, we assume that other vehicles in the simulation strictly follow the road center without any deviation. Based on the target vehicle's position, the corresponding node in the dense graph is queried using R-Tree indexing. Then, the Breadth-First Search algorithm is applied to the dense graph to extract the possible paths that have their length closed to a desired distance, which is approximated by the current vehicle dynamic states. Finally, a set of predicted trajectories $\hat{\Pi}$ is generated using the Pure Pursuit algorithm.

IV. RESULT AND DISCUSSION

A. Metrics

In the CADL challenge, a set of metrics is provided to describe the driving proficiency of an agent. The main metrics are as follows

- **Route completion** is the percentage of the route distance completed by an agent. The evaluation stops for each scenario when the violation occurs, such as collision and agent being blocked for a specific interval (180 seconds).
- **Infraction penalty** is the metric used to show how safe the maneuver is. The value will decay for each time the

agent commits a violation, based on the corresponding ratio. This metric value is in the range of $[0, 1]$.

- **Driving score** is the main metric of the CADL, which is used to rank the participants. It is the product between the route completion ratio and the infraction score. Unit is %.

In CADL, there are several types of infractions, such as Collisions with pedestrians, Collisions with vehicles, Collisions with layout, Running a red light, Route deviation, and Agent blocked. Each type of infraction has a specific penalty coefficient that impacts the overall score of the infraction penalty.

B. Evaluation / Result on CADL

When evaluating the results of our approach on CADL, we compared our method with other participants in the MAPS track of the CADL challenge [24]. The results of running test set provided by CADL are shown in Table I. Our PaaS ranks third on the leaderboard, achieving a driving score of 48.24. However, our approach has the highest infraction penalty of 0.84, indicating that our agent can navigate safely throughout the given scenarios in the CADL challenge.

By examining the details of infractions and the behavior of our ego agent in typical scenarios depicted in Fig. 5, we can conclude that our agent successfully avoids the violations encountered in CADL. Regarding infractions involving dynamic or surrounding agents, our figure for Collision with vehicles ranks first. This clearly demonstrates that our agent is capable of performing safe maneuvers. For instance, in Fig. 5a, the preceding vehicle stops at a red light, and our ego agent plans a trajectory to stop and maintain a safe distance from it. Additionally, in Fig. 5b, our agent effectively navigates through a junction with many pedestrians by incorporating spatial-temporal trajectory prediction, while maintaining good tracking performance with the reference path.

Another notable example is shown in Fig. 5c, where our ego agent confidently executes a right turn at a junction, knowing that the nearby vehicle will not obstruct its path. Similarly in Fig. 5d, the ego agent considers the predicted trajectory of a crossing pedestrian but still makes a prudent decision by maintaining a low speed for forward acceleration, ultimately avoiding collisions with moving obstacles.

In terms of adhering to traffic rules, our agent ranks second in red light infractions, which is primarily influenced by the performance of the Perception module. Moreover, we achieve perfect scores in Route deviations and Collision layout metrics, indicating a decent tracking performance with the reference path.

However, the number of pedestrian collisions in our approach is higher compared to the participants ranking first and second. This can be attributed to a few factors. Firstly, the effectiveness of LiDAR in detecting pedestrians is limited due to the small size of humans relative to the sensor's resolution. Secondly, the predicted trajectories of pedestrians do not always align with their actual behavior in the CADL environment. Furthermore, there are instances in CADL where

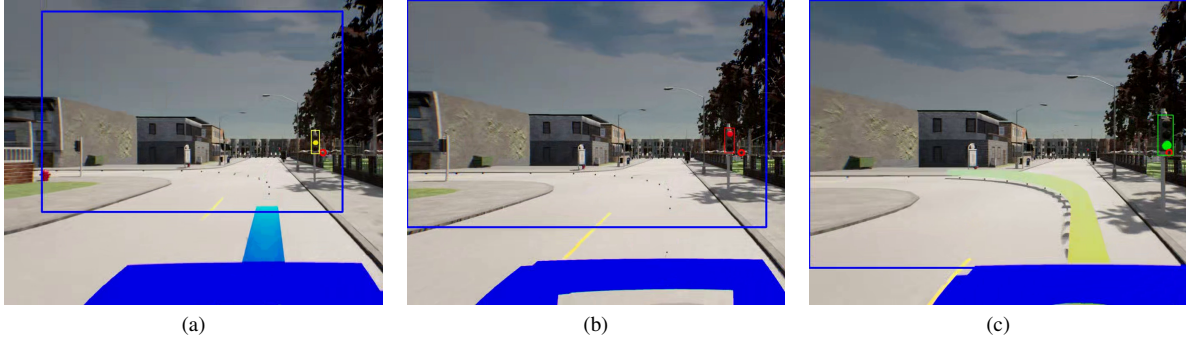


Fig. 4. Illustrations of traffic light detections captured from our vehicle's front camera. The vehicle encounters different states as it approaches the traffic light: (a) yellow - the planned trajectory is stopped before the pole; (b) red - no planning is executed; (c) green - the planned trajectory making a left turn. The approximate traffic light positions are mapped from the OpenDRIVE map, shown as a red circle in the images.

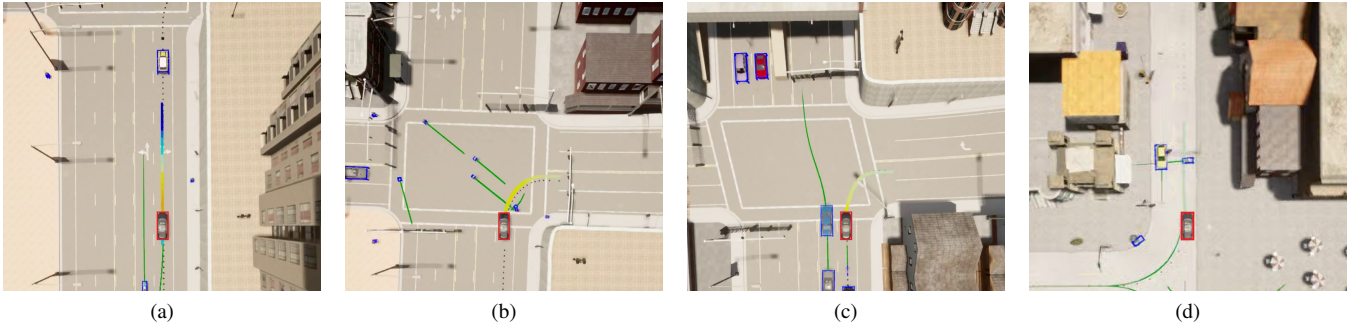


Fig. 5. Illustrations of our ego vehicle maneuvering through typical scenarios. The Localization module determines the position of our system relative to the map. The Perception module identifies moving obstacles, such as vehicles and pedestrians, within the LiDAR's field of view and represents them with blue bounding boxes. The green paths illustrate the predicted trajectories of these obstacles. The reference path is visualized as a black dotted line. Using this information, the Motion Planning module generates the optimal trajectory for our agent, which is color-coded based on velocity: blue represents stopping speed, orange indicates low speed, and red represents high speed.

pedestrians unexpectedly cross the road from the blind spots of our agent, and we were unable to react promptly to such behavior. Unfortunately, the testing scenarios only provide metrics and do not include visual images from cameras, making it more challenging to analyze the root cause.

As discussed in Section II-A, our approach relies solely on the cost function of the trajectory to determine the most optimal path. However, in certain specific scenarios, selecting the trajectory with the lowest cost does not always lead to the best overall solution. This phenomenon, known as local minima, is particularly prevalent in dense traffic situations. Consequently, our figure of Agent blocked infractions is unusually high compared to the performance of other participants.

V. CONCLUSION AND FUTURE WORKS

In this paper, we propose the trajectory planning method for autonomous driving in urban environments and our solutions to conquer the CADL challenge. The trajectory generator produces a reliable and safe maneuver over the planning horizon. However, in several complex cases, the planner is not able to compose the most optimal path due to the fact that the parameters of the cost function are fixed during different scenarios.

To solve this circumstance, adaptive cost functions could be added to the current approach, which requires an extensive review of the scenarios in the challenge. Furthermore, adding a behavior layer would fill the gap in the decision-making problem in our current approach. By applying the Partially observable Markov decision process and its alternatives, we can further increase the reasoning capability of our agent.

REFERENCES

- [1] J. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Dynamically-Stable Motion Planning for Humanoid Robots," *Autonomous Robots*, vol. 12, no. 1, pp. 105–118, Jan. 2002.
- [2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Apr. 2000, pp. 995–1001 vol.2.
- [3] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 1, pp. 166–171, Feb. 1998.
- [4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
- [5] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ -calculus specifications," in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, Dec. 2009, pp. 2222–2229.

TABLE I
TEST-SET EVALUATION RESULT ON CARLA LEADERBOARD CHALLENGE

| Rank | Name | Driving score | Route compl. | Infrac. penalty | Collision pedes. | Collision vehicle | Collision layout | Red light infrac. | Route deviations | Agent blocked |
|------|--------------------|---------------|--------------|-----------------|------------------|-------------------|------------------|-------------------|------------------|---------------|
| | | %, ↑ | %, ↑ | [0, 1], ↑ | #/km, ↓ | #/km, ↓ | #/km, ↓ | #/km, ↓ | #/km, ↓ | #/km, ↓ |
| 1 | Map TF++ | 61.17 | 81.81 | 0.70 | 0.01 | 0.99 | 0.00 | 0.08 | 0.00 | 0.55 |
| 2 | MMFN+ [18] | 59.85 | 82.81 | 0.71 | 0.01 | 0.59 | 0.00 | 0.51 | 0.00 | 0.06 |
| 3 | PaaS (ours) | 48.24 | 60.68 | 0.84 | 0.10 | 0.23 | 0.00 | 0.13 | 0.00 | 4.13 |
| 4 | GRI-based DRL [22] | 33.78 | 57.44 | 0.57 | 0.00 | 3.36 | 0.50 | 0.52 | 1.47 | 0.80 |
| 5 | MMFN [18] | 22.80 | 47.22 | 0.63 | 0.09 | 0.67 | 0.05 | 1.07 | 0.00 | 1003.88 |
| 6 | Techs4AgeCar+ | 18.75 | 75.11 | 0.28 | 1.52 | 2.37 | 1.27 | 1.22 | 0.17 | 1.28 |
| 7 | Pylot | 16.70 | 48.63 | 0.50 | 1.18 | 0.79 | 0.01 | 0.95 | 0.44 | 3.30 |
| 8 | CaRINA [27] | 15.55 | 40.63 | 0.47 | 1.06 | 3.35 | 1.79 | 0.28 | 0.34 | 7.26 |
| 9 | Techs4AgeCar | 12.63 | 61.59 | 0.33 | 2.25 | 0.63 | 0.00 | 0.96 | 0.02 | 1.34 |

↑ : Higher is better.

↓ : Lower is better.

#/km : number of infractions per kilometer.

- [6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
- [7] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Frame," presented at the Proceedings - IEEE International Conference on Robotics and Automation, Jun. 2010, pp. 987–993.
- [8] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, Mar. 2012.
- [9] H. Mouhagir, R. Talj, V. Cherfaoui, F. Aioun, and F. Guillemard, "Integrating safety distances with trajectory planning by modifying the occupancy grid for autonomous vehicle navigation," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 1114–1119.
- [10] J. Nilsson, J. Fredriksson, and E. Coelingh, "Trajectory planning with miscellaneous safety critical zones **This work was supported by FFI - Strategic Vehicle Research and Innovation.," *IFAC-PapersOnLine*, vol. 50, no. 1, Elsevier BV, pp. 9083–9088, Jul. 2017.
- [11] X. Yang and H. Li, "Model Predictive Motion Planning for Autonomous Vehicle in Mid-high Overtaking Scene," *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–5, May 2020.
- [12] H. Wang, Y. Huang, A. Khajepour, Y. Zhang, Y. Rasakhipour, and D. Cao, "Crash Mitigation in Motion Planning for Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 9, pp. 3313–3323, Sep. 2019.
- [13] S. Dixit et al., "Trajectory Planning for Autonomous High-Speed Overtaking in Structured Environments Using Robust MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2310–2323, Jun. 2020.
- [14] R. A. Dollar and A. Vahidi, "Predictively Coordinated Vehicle Acceleration and Lane Selection Using Mixed Integer Programming," *ASME 2018 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers Digital Collection, Nov. 2018.
- [15] V. Bhattacharyya and A. Vahidi, "Automated Vehicle Highway Merging: Motion Planning via Adaptive Interactive Mixed-Integer MPC," unpublished, 2022.
- [16] J. van den Berg, S. Patil, and R. Alterovitz, "Motion Planning Under Uncertainty Using Differential Dynamic Programming in Belief Space," in *Robotics Research: The 15th International Symposium ISRR*, H. I. Christensen and O. Khatib, Eds., in Springer Tracts in Advanced Robotics. Cham: Springer International Publishing, 2017, pp. 473–490.
- [17] H. Li and P. M. Wensing, "Hybrid Systems Differential Dynamic Programming for Whole-Body Motion Planning of Legged Robots," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 5448–5455, Oct. 2020.
- [18] Q. Zhang, M. Tang, R. Geng, F. Chen, R. Xin, and L. Wang, "MMFN: Multi-Modal-Fusion-Net for End-to-End Driving," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2022, pp. 8638–8643.
- [19] A. Prakash, K. Chitta, and A. Geiger, "Multi-Modal Fusion Transformer for End-to-End Autonomous Driving," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7073–7083, Jun. 2021.
- [20] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving," *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.
- [21] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-Enhanced Autonomous Driving Using Interpretable Sensor Fusion Transformer," *Conference on Robot Learning (CoRL)*, 2022.
- [22] R. Chekroun, M. Toromanoff, S. Hornauer, and F. Moutarde, "GRI: General Reinforced Imitation and its Application to Vision-Based Autonomous Driving," *Conference on Neural Information Processing Systems (NeurIPS) 2021, Machine Learning for Autonomous Driving Workshop*, Dec 2021.
- [23] D. Chen, V. Koltun, and P. Krahenbuhl, "Learning to drive from a world on rails," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, IEEE, Oct. 2021.
- [24] CARLA. Autonomous driving leaderboard. [Online]. Available: <https://leaderboard.carla.org/leaderboard>
- [25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," *arXiv*, Nov. 10, 2017.
- [26] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15729–15741, Sep. 2022.
- [27] L. A. Rosero et al., "A Software Architecture for Autonomous Vehicles: Team LRM-B Entry in the First CARLA Autonomous Driving Challenge," 2020, unpublished.
- [28] J. H. Ryu, G. Gankhuyag, and K. T. Chong, "Navigation System Heading and Position Accuracy Improvement through GPS and INS Data Fusion," *Journal of Sensors*, vol. 2016, p. e7942963, Jan. 2016.
- [29] S. Shi et al., "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.
- [30] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *Aug.*, 2021, unpublished.