

Frenet Corridor Planner: An Optimal Local Path Planning Framework for Autonomous Driving

Faizan M. Tariq¹, Zheng-Hang Yeh¹, Avinash Singh¹, David Isele¹, Sangjae Bae¹

Abstract—Motivated by the requirements for effectiveness and efficiency, path-speed decomposition-based trajectory planning methods have widely been adopted for autonomous driving applications. While a global route can be pre-computed offline, real-time generation of adaptive local paths remains crucial. Therefore, we present the Frenet Corridor Planner (FCP), an optimization-based local path planning strategy for autonomous driving that ensures smooth and safe navigation around obstacles. Modeling the vehicles as safety-augmented bounding boxes and pedestrians as convex hulls in the Frenet space, our approach defines a drivable corridor by determining the appropriate deviation side for static obstacles. Thereafter, a modified space-domain bicycle kinematics model enables path optimization for smoothness, boundary clearance, and dynamic obstacle risk minimization. The optimized path is then passed to a speed planner to generate the final trajectory. We validate FCP through extensive simulations and real-world hardware experiments, demonstrating its efficiency and effectiveness.

I. INTRODUCTION

Path-speed decomposition approaches are widely used in autonomous vehicle trajectory planning (path and speed) due to their reliability and efficiency [1]–[3]. These methods simplify the overall trajectory planning problem by separately handling path and speed planning, resulting in computationally efficient algorithms [4]–[6]. Building on this framework, this work focuses on developing an efficient path planning strategy for autonomous driving.

Consider the scenario illustrated in Fig. 1. If a route were generated in advance from the map data, it would likely not incorporate the cars parked on the side of the road. However, deviating around the parked cars must be done in a way that minimizes disruption to the oncoming traffic. This necessitates online path generation where the path can be updated smoothly while considering the dynamic limitations of the ego vehicle.

Related Work

There is an extensive history of path planning research including learning-based approaches [7], [8], artificial potential field methods [9], graph search techniques [10], [11], sampling-based methods [12], [13], polynomial (parametric) optimization strategies [14], and non-parametric optimization methods [15]. However, much of this work does not simultaneously address kinematic feasibility, path smoothness, and computational efficiency for time-sensitive and safety-critical applications, such as autonomous driving.

¹Honda Research Institute, USA. Email: {faizan.tariq, zheng-hang-yeh, avinash.singh, disele, sbae}@honda-ri.com

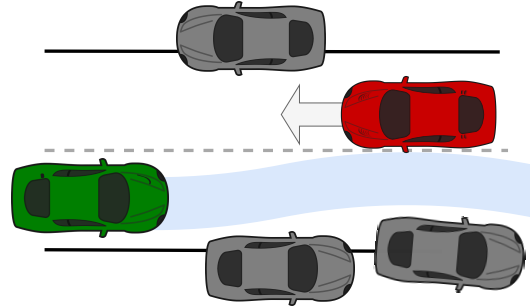


Fig. 1. **Motivational scenario.** The ego vehicle (in green) must deviate from the lane center to avoid a collision with the parked cars on the roadside while being cognizant of the oncoming traffic. Without a local path planner, the ego vehicle may remain stuck, waiting indefinitely for the parked cars to move before proceeding along its pre-determined global route.

Formal optimization methods can directly incorporate dynamic and safety constraints, but this often comes at a detriment to computational efficiency. Non-linear vehicle dynamics [16], [17], the existence of multiple routes through space-time [18], [19], and the uncertainty and interactivity of other traffic participants [20]–[24] all combine to greatly increase the complexity of the problem. This results in a highly non-convex formulation which takes significant computation time and resources to solve. Our approach follows this line of work but focuses on systematically reducing complexity by dividing the problem into stages, achieving efficient path planning at a low computational cost.

Contribution

In this work, we propose the Frenet Corridor Planner (FCP), an efficient optimization-based path planning strategy that generates smooth paths around obstacles through a multi-stage process. First, vehicles in the environment are represented as safety-augmented bounding boxes, while pedestrian clusters are modeled as convex hulls in the Frenet space. The appropriate deviation side is then determined for each static obstacle, and the drivable region (corridors) for the ego vehicle is established. Next, an optimization problem is solved to generate a path that maximizes smoothness, maintains a safe distance from corridor boundaries, and aligns with the reference path while considering the risk associated with dynamic obstacles. This is achieved using a modified space-domain bicycle kinematics model, which, to the best of our knowledge, has not been previously explored. Finally, the generated path is passed to the speed planner to produce the overall trajectory. Our approach is rigorously evaluated in the presence of perception noise through both simulations and physical hardware experiments.

II. PROBLEM SETTING

A. Frenet Coordinate System

In this work, we utilize the Frenet Coordinate System [25], where the s -axis is representative of the longitudinal displacement along a given reference (global) path (ref), while the d -axis denotes the lateral displacement orthogonal to ref. Any point $\alpha \in \mathbb{R}^2$ can be transformed from the Cartesian to the Frenet frame using a non-linear transformation as follows:

$$f_c: (\alpha_x, \alpha_y, \text{ref}) \mapsto (\alpha_s, \alpha_d). \quad (1)$$

B. System Architecture

The overall modular system architecture is adopted from our previous work [26], with this study focusing specifically on the motion planning layer. We tackle the trajectory planning problem with a decoupled scheme (Fig. 2) that allows us to tackle the path and speed planning problems independently. This decomposes the overall problem complexity to yield an efficient trajectory planning algorithm. In our previous work, we developed a robust speed planning method [3], so this work serves to fill in the gap by proposing a corresponding path planning scheme. The generated trajectory, consisting of both path and speed, is forwarded to the downstream Control module, which also utilizes a decoupled control scheme [22]. Longitudinal control is managed by a PID controller, tuned using the CARLA simulator [27], while lateral control is handled by a model predictive controller (MPC) designed with a one-time-step planning horizon based on the vehicle rotation model [28].

Remark 1: Although we have used our previously developed algorithms for the downstream speed and control modules in the validation studies (Section IV), the modular system architecture provides the flexibility to integrate any external algorithm from the literature.

C. Planning Pipeline

The Frenet Corridor Planner (FCP) in itself consists of several submodules that work together to generate a path for the downstream speed planning and control modules to follow, as depicted in Fig. 2. The Data Processor (DP) processes perception and localization data to extract obstacle-related information in the Frenet frame, which is then sent to the Decision Governor (DG). The DG determines the appropriate side to deviate each obstacle and forwards this information to the Boundary Generator (BG). The BG defines the boundaries of the drivable region (corridor) and transmits them to the Path Optimizer (PO), which generates a path using our proposed space domain kinematics model.

III. FRENET CORRIDOR PLANNER

A. Data Processor (DP)

The DP takes in the obstacles' state (position, orientation, and size) information from the external perception and localization module [26] to generate safety-augmented bounding boxes for vehicles and convex hulls for pedestrian clusters with the help of DBSCAN [29], a density-based clustering

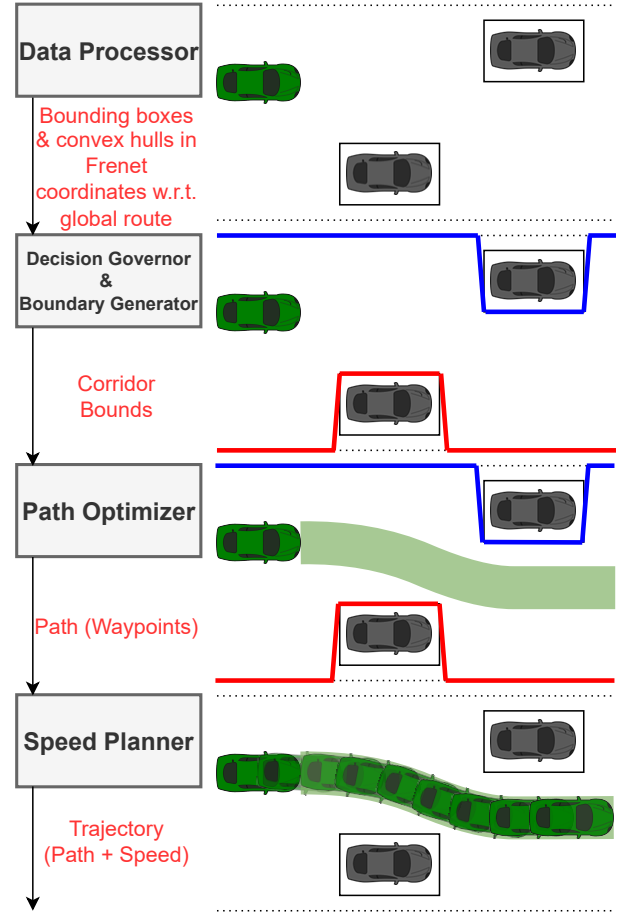


Fig. 2. **Trajectory planning pipeline.** The data flow between the various building blocks of FCP is illustrated on the left, while the output visualization from each module is shown on the right.

algorithm. Any pedestrian not associated with a cluster is treated as an independent obstacle. At the current time t , the obstacle set containing the linearly interpolated points along the edges of the bounding boxes and convex hulls in the Frenet frame is denoted by $\mathcal{O}_t$.

B. Decision Governor (DG)

In addition to having information on the locations and speeds of the obstacles present in the environment, FCP requires further information on the appropriate side of deviation for a static obstacle. Specifically, FCP needs to know whether a static obstacle should be considered in the upper or lower boundary of the corridor, and this information is provided by the DG. Therefore, DG partitions the obstacle set ($\mathcal{O}_t$) into disjoint lower ($\mathcal{O}_t^{\text{lb}}$) and upper ($\mathcal{O}_t^{\text{ub}}$) bound obstacle sets such that $\mathcal{O}_t = \mathcal{O}_t^{\text{lb}} \cup \mathcal{O}_t^{\text{ub}}$ and $\mathcal{O}_t^{\text{lb}} \cap \mathcal{O}_t^{\text{ub}} = \emptyset$.

Since this work focuses primarily on the path planning layer, we demonstrate the efficacy of FCP with a simple decision tree [30] for our DG, as shown in Fig. 3. This simple decision tree performs well even with moderate noise in the perception and localization data. However, with large noise, it may cause inconsistent obstacle classification (i.e., switching between lower and upper bounds), especially when an obstacle is located near the center of the drivable space.

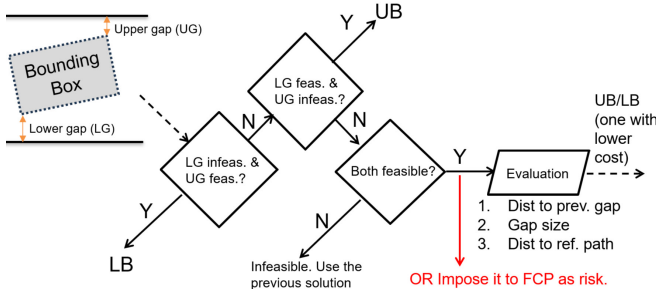


Fig. 3. **Decision tree for boundary classification of each obstacle.** The decision tree evaluates the lower and upper gaps within the drivable space. If both gaps are available, two approaches can be used: selecting the preferred gap based on cost evaluation or treating the obstacle as a risk in PO.

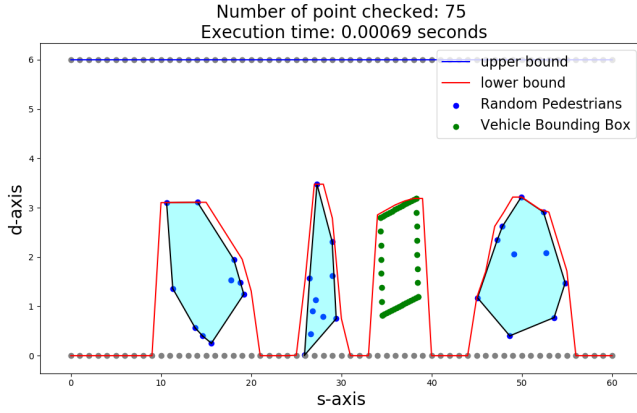


Fig. 4. **Boundary Generation.** With the pedestrians shown as blue dots, the augmented vehicle boundary depicted by green dots, and the convex hulls of pedestrian clusters given by the blue lines, Algorithm 1 generates the lower and upper bounds, shown by the red and blue lines, respectively.

Alternatively, the obstacle can be treated as a risk in the PO (see Section III-D.4) to enhance path consistency. Owing to the modularity of our approach, more advanced methods from the literature [31] can be incorporated into DG to improve the robustness of the trajectory planning pipeline.

C. Boundary Generator (BG)

The BG takes in $\mathcal{O}_t^{\text{lb}}$ and $\mathcal{O}_t^{\text{ub}}$ as well as the lower and upper road limits, l_t^{lb} and l_t^{ub} , to generate the lower and upper bounds, d_t^{lb} and d_t^{ub} , in the Frenet space for a planning horizon of N steps with a longitudinal spacing of Δs meters, resulting in a planning distance of $L = N \times \Delta s$ meters. The boundary generation algorithm, outlined in Algorithm 1, essentially identifies the “extremum” d-value for each obstacle point at the queried s-positions. Therefore, the algorithm requires only a single pass over the obstacle points resulting in a runtime complexity of $\mathcal{O}(N_o)$ where N_o corresponds to the sum of the number of boundary points for all obstacles in $\mathcal{O}_t$. The output of BG is depicted in Fig. 4.

D. Path Optimizer (PO)

A path at the current time step t is defined as $P_t = [p_t^1, \dots, p_t^N]$ where N denotes the number of waypoints and $p_t^i \in \mathbb{R}^2 \forall i \in \{1, \dots, N\}$ denotes a waypoint in space.

Algorithm 1: Boundary Generation Algorithm

Input: $\mathcal{O}_t^{\text{lb}}, \mathcal{O}_t^{\text{ub}}, l_t^{\text{lb}}, l_t^{\text{ub}}, N, \Delta s, s_0$

Output: $d_t^{\text{lb}}, d_t^{\text{ub}}$

Initialize: $d_t^{\text{lb}} \leftarrow l_t^{\text{lb}}; d_t^{\text{ub}} \leftarrow l_t^{\text{ub}}$

```

1 for  $O \in \{\mathcal{O}_t\}$  do
2   for  $p = [p_s, p_d] \in O$  do
3     ind =  $\lfloor (p_s - s_0) / \Delta s \rfloor$  // Compute  $p$  index
4     if  $O \in \mathcal{O}_t^{\text{lb}}$  then
5       if  $d_t^{\text{lb}}[\text{ind}] < p_d$  then
6          $d_t^{\text{lb}}[\text{ind}] = p_d$ 
7         if  $\text{ind} \in \{0, \dots, N-2\}$  then
8            $d_t^{\text{lb}}[\text{ind}+1] = p_d$  // Obstacle
              corners handling
9         end
10      end
11    else if  $O \in \mathcal{O}_t^{\text{ub}}$  then
12      if  $d_t^{\text{ub}}[\text{ind}] > p_d$  then
13         $d_t^{\text{ub}}[\text{ind}] = p_d$ 
14        if  $\text{ind} \in \{0, \dots, N-2\}$  then
15           $d_t^{\text{ub}}[\text{ind}+1] = p_d$  // Obstacle
              corners handling
16        end
17      end
18    end
19  end
20 end
  
```

1) *Space Domain Kinematics Model:* Since the goal of this work is to generate P_t , we transform the vehicle kinematic model from the space-time to space-only domain, allowing us to reduce the problem’s complexity. For our base model, we opt for the non-linear kinematic bicycle model, which has been a popular choice for automated vehicle trajectory planning due to its accuracy and efficiency [32].

In the Cartesian Frame, the states, X_t , and the control inputs, U_t , of the ego vehicle at time instant t are given by $X_t = [x_t \ y_t \ \psi_t \ v_t]^\top \in \mathbb{X}_t$ and $U_t = [a_t \ \delta_t]^\top \in \mathbb{U}_t$, respectively. Here, x_t, y_t, ψ_t , and v_t respectively denote the x-coordinate (m), y-coordinate (m), yaw angle with respect to the x-axis (rad), and speed (m/s), whereas $a(k)$ and $\delta(k)$ respectively denote acceleration (m/s²), and steering angle (rad). The sets $\mathbb{X}_t = \mathbb{R}^2 \times \mathbb{R}_{[0, 2\pi)} \times \mathbb{R}_{[0, V^{\text{max}}]}$ and $\mathbb{U}_t = \mathbb{R}_{[U^{\text{min}}, U^{\text{max}}]}^2$, respectively, denote the feasible states and the actuation limits. Then, the system dynamics read:

$$x_{t+1} = x_t + v_t \cos(\psi_t + \beta_t) \Delta t, \quad (2)$$

$$y_{t+1} = y_t + v_t \sin(\psi_t + \beta_t) \Delta t, \quad (3)$$

$$\psi_{t+1} = \psi_t + \frac{v_t}{\ell_r} \sin \beta_t \Delta t, \quad (4)$$

$$v_{t+1} = v_t + a_t \Delta t, \quad (5)$$

where t denotes the temporal index, Δt denotes the sampling timestep, and $\beta_t = \tan^{-1} \left(\frac{\ell_r}{\ell_f + \ell_r} \tan \delta_t \right)$ maps the steering input (δ_t) to the vehicle orientation (ψ_t). Now, to remove the time dependence, we fix the distance step as $l_t = v_t \Delta t$.

Moreover, since we explore only the spatial domain, we can disregard the time-varying speed v_t and fix the constant distance step as $l_t \equiv \Delta l$, giving the following modified kinematics model:

$$x_{k+1} = x_k + \cos(\psi_k + \beta_k) \Delta l, \quad (6)$$

$$y_{k+1} = y_k + \sin(\psi_k + \beta_k) \Delta l, \quad (7)$$

$$\psi_{k+1} = \psi_k + \frac{1}{\ell_r} \sin \beta_k \Delta l, \quad (8)$$

where k denotes the spatial index. Now, the kinematics model is given in terms of the fixed “arc length” step along the path, denoted by Δl . However, in order to have a uniform correspondence between the path and the upper/lower bounds, we need to further reformulate the kinematics model such that the independent variable is the constant “longitudinal distance” step (Δs), instead of Δl – this facilitates the direct integration of d_t^{lb} and d_t^{ub} , generated by the BG, into the PO.

Remark 2: For the optimization problem (Section III-D.3), the bounds d_t^{lb} and d_t^{ub} are defined at each “knot” of the path P_t , which necessitates the reformulation of the kinematics model in terms of the independent Δs variable.

Remark 3: With the updated set of states given by $\tilde{X}_k = [x_k \ y_k \ \psi_k]^\top$, the modified kinematics model is now governed only by the control input $\delta_k \in \mathbb{R}_{[\delta^{\min}, \delta^{\max}]}$.

Now, assuming that we are working in the Frenet frame (Section II-A), we can obtain the kinematics in terms of the “longitudinal distance” step by introducing the following non-linear transformation (projection): $\Delta s = \Delta l \cos(\psi_k + \beta_k)$. The kinematics model then reads:

$$x_{k+1} = x_k + \Delta s, \quad (9)$$

$$\begin{aligned} y_{k+1} &= y_k + \frac{\sin(\psi_k + \beta_k)}{\cos(\psi_k + \beta_k)} \Delta s \\ &= y_k + \tan(\psi_k + \beta_k) \Delta s, \end{aligned} \quad (10)$$

$$\psi_{k+1} = \psi_k + \frac{\Delta s}{\ell_r} \frac{\sin \beta_k}{\cos(\psi_k + \beta_k)}. \quad (11)$$

Note that the kinematics model is ill-posed for $\psi_k + \beta_k = \pi/2$. This corresponds to a path P_t orthogonal to ref – In the Frenet frame, each point along such a path has the same s value, while the path is not defined for any other s value, leading to a singularity.

Remark 4: The bicycle kinematic model does not translate directly from the Cartesian to the Frenet frame due to a non-linear transformation with respect to ref (1). To address this, we introduce a curvature-based model correction in Section III-D.2 to ensure conformance of the Cartesian-based kinematic model to the Frenet space.

Now, β_k is defined as:

$$\beta_k = \arctan \left(\frac{l_r}{l_f + l_r} \tan \delta_k \right), \quad (12)$$

which is non-linear. To linearize, we approximate:

$$\beta_k \approx \frac{l_r}{l_f + l_r} \delta_k. \quad (13)$$

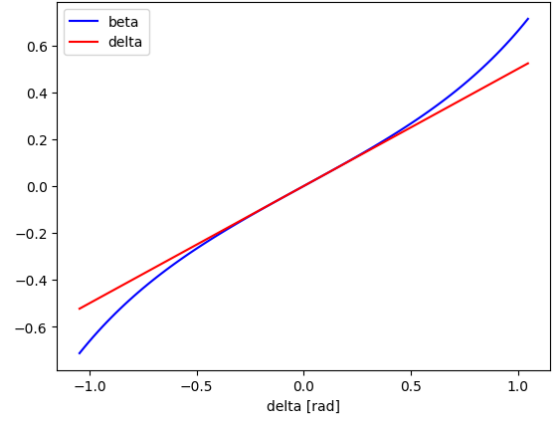


Fig. 5. **Numerical validation for $\frac{l_r}{l_f + l_r} |\delta_k|$ under-approximating $|\beta_k|$.** The linear plot $\frac{l_r}{l_f + l_r} \delta_k$ stays below the β_k curve for $d_k \in [0, \frac{\pi}{2})$ and above β_k for $d_k \in (-\frac{\pi}{2}, 0]$ showing $|\beta_k| \geq \frac{l_r}{l_f + l_r} |\delta_k| \ \forall \delta_k \in (-\frac{\pi}{2}, \frac{\pi}{2})$.

This approximation is kinematically valid (feasible) only if $\frac{l_r}{l_f + l_r} |\delta_k|$ under-approximates $|\beta_k|$ – otherwise, the approximated kinematics with the maximum steering angle $\delta_{\max}$ may not be feasible. Thus, we show that $|\beta_k| \geq \frac{l_r}{l_f + l_r} |\delta_k| \ \forall \delta_k \in (-\frac{\pi}{2}, \frac{\pi}{2})$. For brevity, subscript k is omitted in the following proof.

Proof: Let:

$$f(\delta) = \arctan(a \tan \delta) - a\delta, \quad (14)$$

where $a = \frac{l_r}{l_f + l_r} \in [0, 1]$. We want to show that $f(\delta) \geq 0 \ \forall \delta \in [0, \frac{\pi}{2})$ and $f(\delta) \leq 0 \ \forall \delta \in (-\frac{\pi}{2}, 0]$. The derivative reads:

$$f'(\delta) = \frac{a \sec^2 \delta}{1 + a^2 \tan^2 \delta} - a \quad (15)$$

$$= \frac{a}{\cos^2 \delta + a^2 \sin^2 \delta} - a \quad (16)$$

$$= a \left(\frac{1 - (\cos^2 \delta + a^2 \sin^2 \delta)}{\cos^2 \delta + a^2 \sin^2 \delta} \right). \quad (17)$$

Given $a \in [0, 1]$, the following suffices for all δ (proof is trivial):

$$\cos^2 \delta + a^2 \sin^2 \delta \leq 1. \quad (18)$$

Thus, $f'(\delta) \geq 0$, indicating f is a monotonically increasing function. When $\delta = 0$, $f(\delta) = 0$. Due to the monotonicity, for $\delta > 0$, $f(\delta) \geq 0$, i.e.,

$$f(\delta) = \arctan(a \tan \delta) - \delta \geq 0 \quad (19)$$

$$\longleftrightarrow \arctan(a \tan \delta) \geq \delta. \quad (20)$$

Similarly, one can prove for $\delta \in (-\frac{\pi}{2}, 0]$. The numerical validation is shown in Fig. 5. In practice, especially for normal highway driving where the car is not pushed to the actuation limits, the steering range is typically limited to $[-0.6, 0.6]$ rad within which the approximation holds. ■

2) Actuation limit induced by reference path's curvature:

We have thus far transformed the bicycle kinematic model into a space-only kinematic model with longitudinal space sampling. However, we are yet to account for ref 's curvature in the formulation, which is essential for converting the kinematic model from the Cartesian to the Frenet space. Neglecting this curvature can lead to kinematic infeasibility, making it difficult for the vehicle to follow the computed path. To address this, we directly impose curvature limitation on the path-planning problem in Section III-D.3 by restricting the feasible actuation set using the angle changes along ref at each step k , given as $\Delta\bar{\psi}_k$.

Based on the kinematics model derived in Section III-D.1, the change in heading angle (11) at any step k reads:

$$\Delta\psi_k = \frac{\Delta s}{\ell_r} \frac{\sin u_k}{\cos(\psi_k + u_k)}, \quad (21)$$

which is lower bounded by $\frac{\Delta s}{\ell_r} \tan u_k$ for $|\psi_k + u_k| < \frac{\pi}{2}$ where $u_k = \frac{\ell_r}{l_f + l_r} \delta_k$. We approximate $\Delta\bar{\psi}_k \approx \frac{\Delta s}{\ell_r} \tan u_k$ to over-constrain the feasible set for the steering input. Consequently, the steering required to follow ref reads:

$$\bar{u}_k = \arctan\left(\frac{\ell_r}{\Delta s} \Delta\bar{\psi}_k\right). \quad (22)$$

Thereafter, the control bounds in (30) are restricted as:

$$u_k + \bar{u}_k \in [u^{\min}, u^{\max}] \forall k, \quad (23)$$

where $u^{\min/\max} = \frac{\ell_r}{l_f + l_r} \delta^{\min/\max}$.

3) *Non-Linear Optimization Problem:* Using the reformulated Frenet space bicycle kinematics model derived in Section III-D.1 and the spatial corridor boundaries, d_t^{lb} and d_t^{ub} , obtained through the BG, the path planning optimization problem is posed as follows:

$$\begin{aligned} \min_u \quad & d^\top Q_d d + u^\top Q_u u + \lambda_{\text{curve}} \sum_k \tan^2 u_k \\ & + \lambda_{\text{risk}} \sum_k \left(d_k - \frac{d_{t_k}^{\text{lb}} + d_{t_k}^{\text{ub}}}{2} \right)^2 \end{aligned} \quad (24)$$

subject to:

$$s_{k+1} = s_k + \Delta s, \quad (25)$$

$$d_{k+1} = d_k + \tan(\phi_k + u_k) \Delta s, \quad (26)$$

$$\phi_{k+1} = \phi_k + \frac{\Delta s}{\ell_r} \frac{\sin u_k}{\cos(\phi_k + u_k)} \quad (27)$$

$$d_{t_k}^{\text{lb}} \leq d_k \leq d_{t_k}^{\text{ub}} \quad (28)$$

$$s_0 = \hat{s}_t, \quad d_0 = \hat{d}_t, \quad \phi_0 = \hat{\phi}_t \quad (29)$$

$$u_k^{\min} \leq u_k \leq u_k^{\max} \quad (30)$$

for all $k \in \{0, \dots, N-1\}$ where $d = [d_1, \dots, d_k]^\top$, $u = [u_1, \dots, u_k]^\top$, N is the number of planning steps, operator $\hat{\cdot}$ (hat) denotes the current measurement, the set $\mathbb{R}_{[u_k^{\min}, u_k^{\max}]}$ denotes actuation limits incorporating the limits induced by ref 's curvature (23), and Q and λ denote penalty weight matrix and scalar, respectively. The objective function in (24) penalizes: (i) deviation from the global route (recall that ref

corresponds to $d = 0$ in the Frenet frame), (ii) steering effort, (iii) path curvature, and (iv) distance to boundaries.

Remark 5: To ensure computational efficiency, the optimization problem is formulated with a convex objective function and no inequality constraints, making the kinematics model the only source of non-convexity. This is achieved by replacing the non-convex collision avoidance inequality constraints with the precomputed corridor bounds. While the kinematics model could be linearized, at the cost of model accuracy, to formulate a convex problem, we found this linearization unnecessary considering the strong computational performance demonstrated in Section IV-D.

4) *Dynamic Obstacle Handling:* Generating boundaries for dynamic obstacles (using Algorithm 1) may lead to recursive infeasibility as fluctuating behaviors and predictions over time may cause the upper and lower bounds to intersect. Therefore, we treat dynamic obstacles as additive risks in the optimization problem. Specifically, each predicted position over a given time horizon is included as a convex cost in the optimization cost (24). Therefore, the additive penalty reads:

$$r_{\text{dyn}} = \lambda_{\text{dyn}} \sum_{i \in \{1, \dots, N_t^{\text{dyn}}\}} \sum_k \frac{1}{(\hat{d}_k^{(i)} - d_k)^2}, \quad (31)$$

where λ_{dyn} is the penalty weight, N_t^{dyn} is the number of dynamic obstacles within the planning space at the current time t , and $\hat{d}_k^{(i)}$ is the predicted lateral (center of mass) position of a dynamic obstacle i at step k . Recall that our pipeline is based on a path-speed decomposition method (Fig. 2), so the speed planner guarantees safety in space-time with respect to dynamic obstacles.

5) *Perception Noise Handling:* In the presence of perception noise, the existing optimization problem can become infeasible, especially when the ego vehicle is located close to the corridor boundaries. To ensure problem feasibility, we introduce a bounded slack variable for the bounds in (28) as:

$$d_{t_k}^{\text{lb}} - \alpha_k \leq d_k \leq d_{t_k}^{\text{ub}} + \alpha_k, \quad (32)$$

where $\alpha_k \leq \bar{\alpha}$ with a fixed $\bar{\alpha}$. Then, we add a slack penalty term in (24) as $\lambda_\alpha \sum_k \alpha_k^2$ with $\lambda_\alpha \gg 0$.

IV. EXPERIMENTS

A. Experimental Setup

The validation studies are performed on a system running Ubuntu 22.04 LTS, equipped with an Intel® Xeon(R) Silver 4210R CPU @ 2.40GHz $\times$ 40 and an NVIDIA RTX A5000 graphics card. To assess the performance of FCP, we consider a scenario designed to replicate real-world conditions where the reference path is obstructed by stationary vehicles, requiring the ego vehicle to generate an alternative trajectory, as illustrated in Fig. 6. To maintain progress along its global route, the ego vehicle must navigate around the obstacles by temporarily entering the oncoming lane, return to the original lane to evade an oncoming vehicle and deviate to the oncoming lane again to finish the maneuver.

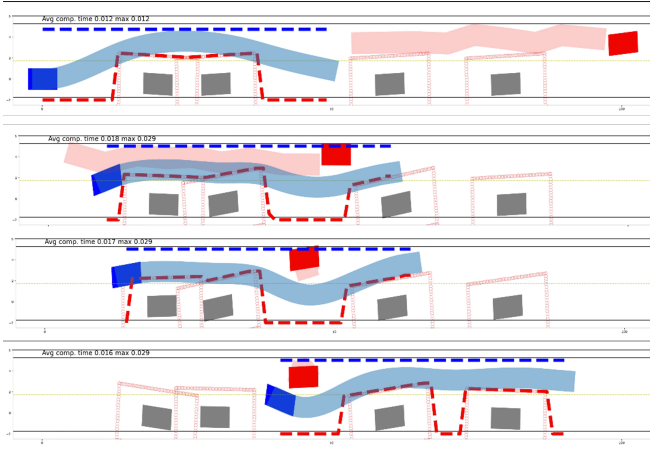


Fig. 6. **Testing Scenario for Comparative Analysis.** The ego vehicle is depicted in blue, the oncoming vehicle in red, the stationary vehicles in gray, their bounding boxes with red circles, the upper/lower bound in dashed blue/red lines, the noisy perception/prediction in light red, and the ego vehicle's planned path in light blue. The scenario progression is shown from top to bottom. Note that the lower and upper boundaries are generated w.r.t. the centroid of the ego vehicle, and the ego vehicle's path is updated w.r.t. the dynamic obstacle and perception/prediction noises.

B. Comparative Analysis

We consider A^* [10], RRT^* [12] and Bidirectional $B-RRT^*$ [33], planning algorithms as baselines to compare against FCP in the scenario depicted in Fig. 6.

The metrics chosen for this comparative analysis are: (i) Algorithm runtime (s) - M_t , (ii) Maximum change in yaw (rad) - M_{my} , (iii) Average change in yaw (rad) - M_{ay} , (iv) Average path deviation from reference path (m) - M_l , (v) Minimum distance to the closest vehicle (m) - M_{md} , and (vi) Average distance to the closest vehicle (m) - M_{ad} .

These metrics are selected to assess key characteristics of the tested algorithms and the paths they generate. Specifically, M_t measures computational efficiency, while M_l quantifies the deviation of the generated path from the reference trajectory. The safety performance of the algorithm concerning the vehicles in the environment is evaluated through M_{md} and M_{ad} . Additionally, the smoothness of the generated path is assessed using M_{my} and M_{ay} .

The results of the quantitative comparative analysis are summarized in Table I. To ensure a fair comparison, paths for the sampling-based methods, RRT^* and $B-RRT^*$, are generated 1000 times, with metric values averaged over these runs. FCP *significantly* outperforms the baseline methods in M_t , and M_{my} highlighting its superior computational efficiency and path smoothness. FCP also outperforms the baselines in M_{md} and M_{ad} , demonstrating enhanced safety performance. In terms of M_{ay} , FCP surpasses the A^* and RRT^* but performs worse than $B-RRT^*$ due to trade-offs with other critical metrics. Notably, FCP is the only algorithm that successfully passes the test scenario (without going out of bounds or colliding with any obstacles).

Qualitatively speaking, as the dynamic obstacle approaches (Fig. 6), the FCP path smoothly adapts to shift closer to the lower bound to account for the risk associated with the dynamic obstacle. The path remains robust to

TABLE I
SCENARIO-BASED COMPARATIVE ANALYSIS

Method/Metric	FCP	A^*	RRT^*	$B-RRT^*$
Scenario Passed	Y	N	N	N
Run-time	0.035	0.628	0.173	0.187
Max delta yaw	0.053	0.785	0.816	0.717
Avg delta yaw	0.016	0.051	0.003	0.002
Avg path div.	2.336	2.360	2.867	2.922
Min dist obs	3.182	2.683	2.693	2.586
Avg dis obs	34.99	29.90	29.81	30.22

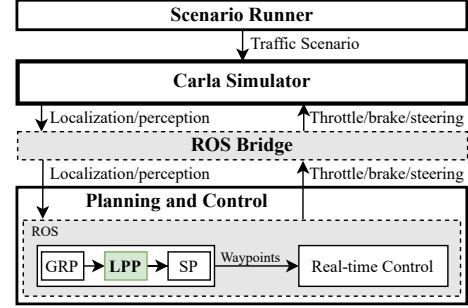


Fig. 7. **CARLA Simulation Setup.** The simulation scenario, generated by the Scenario Runner, is passed on to the CARLA Simulator, which communicates with the Planning and Control ROS nodes through the ROS bridge node at a frequency of 10 Hz. GRP denotes global route planning, LPP denotes local path planning, and SP denotes speed planning.

perception noise, maintaining consistency despite fluctuating bounds. This resilience stems from IPOPT [34] navigating within the interior of the feasible region rather than along its boundaries. Additionally, the solution remains feasible even under high noise levels (second row of Fig. 6) due to the slack variable introduced in Section III-D.5.

C. CARLA Simulations

To thoroughly evaluate the performance of FCP in a high-fidelity environment, the scenario shown in Fig. 8 is implemented in the CARLA Simulator [27], which provides a realistic urban driving environment with sensor simulation

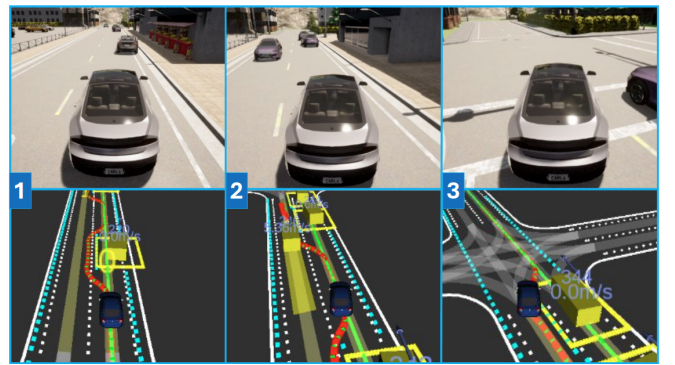


Fig. 8. **CARLA Simulations.** The numbered frames show the progression of the ego vehicle through the scenario. The Rviz windows, below the Carla Pygame windows, show various objects considered during planning and the output of FCP. The ego vehicle is depicted in blue, and the obstacles are depicted as yellow cuboids surrounded by yellow safety-augmented bounding boxes. The transparent path in front of the obstacle is a constant velocity prediction path. The reference path is shown as the green line, and the local path generated by FCP is shown in red.

TABLE II
MONTE CARLO SIMULATIONS

Model	Time	Acc. Min	Acc. Max	Jerk Min	Jerk Max	Ang. Acc. Max	Ang. Jerk Max
Average							
FCP	24.743	-2.83	1.85	-2.82	1.9	40.02	67.38
A*	28.834	-2.85	1.78	-2.93	2.48	42.31	91.1
Standard Deviation							
FCP	0.843	0.142	0.070	0.120	0.130	1.855	4.235
A*	2.081	0.221	0.066	0.124	0.200	3.348	11.685

and dynamic actors. The simulation setup within CARLA is illustrated in Fig. 8. The local path planner (LPP as mentioned in Fig. 7) employed is FCP. The other system module i.e., the speed planner (SP) is derived from [3]. The performance of FCP in this scenario within the CARLA environment is demonstrated in Fig. 8, where the ego vehicle successfully navigates around dynamic and static obstacles.

To rigorously evaluate FCP against a standard graph-based planner, namely A*, we conduct Monte Carlo simulations for the scenario depicted in Fig. 8. In these simulations, the positions and orientations of the other three vehicles are randomized within predefined ranges: the longitudinal and lateral positions vary within $10m$ and $2m$, respectively, while the vehicle headings are randomized within a 10° range. Maintaining all system modules identical, except for the local path planning component from Fig. 7, we run 50 simulation trials each and present the results across various performance metrics in Table II. Notably, none of the trials for either method resulted in a collision.

The metrics used for this Monte Carlo analysis include: (i) Completion time (s); (ii) Minimum linear acceleration (m/s^2); (iii) Maximum linear acceleration (m/s^2); (iv) Minimum linear (deceleration) jerk (m/s^3); (v) Maximum linear (acceleration) jerk (m/s^3); (vi) Maximum angular acceleration (rad/s^2); and, (vii) Maximum angular jerk (rad/s^3). The linear metrics reflect the vehicle's throttle and braking behavior, primarily influenced by the speed planner, while the angular metrics correspond to steering dynamics, which are governed by the smoothness of the generated path.

The results in Table II demonstrate that FCP outperforms A* in terms of path smoothness, which directly impacts the angular and steering-related metrics. A smoother path enhances passenger comfort and allows the ego vehicle to efficiently return to the reference trajectory, as indicated by lower completion time. Additionally, the smoother trajectories generated by FCP provide the speed planner with greater confidence to accelerate and decelerate along the deviation path, leading to increased linear acceleration values compared to A*.

Furthermore, the standard deviation values reinforce the consistency of FCP's performance relative to A*. Lower standard deviation values indicate that FCP consistently generates smooth paths across a wide range of scenarios, ensuring reliable and predictable behavior.

D. Computational Time Analysis

We assess the computational efficiency of our algorithm, implemented using IPOPT in Casadi (Python), by analyzing

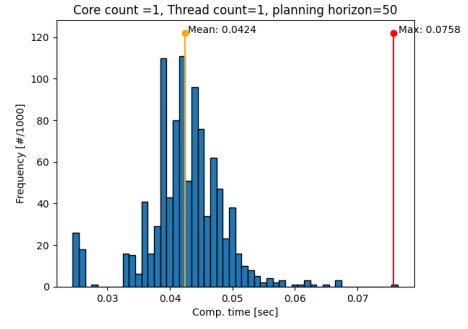


Fig. 9. **Computational Efficiency Analysis.** The test scenario in Fig. 6 is randomized, similar to the Monte-Carlo simulations in Section IV-C, and repeated 1000 times to obtain the FCP computation time distribution.

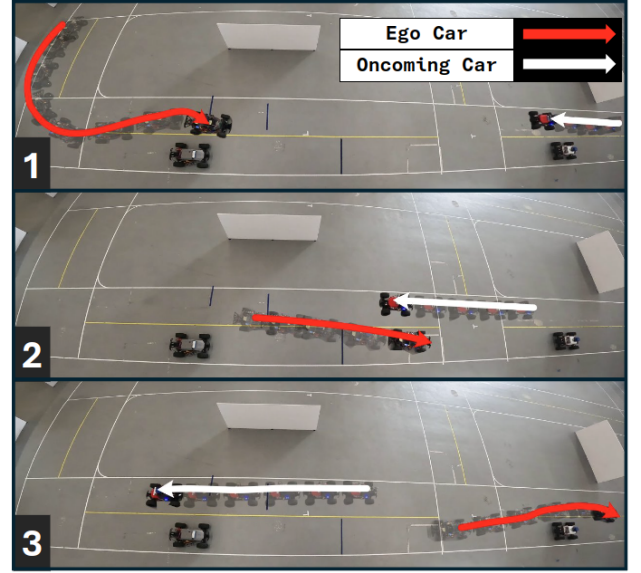


Fig. 10. **Hardware Demonstration.** A scenario involving two parked vehicles and an oncoming vehicle is demonstrated using 1/10-scale autonomous cars. The numbered frames illustrate the progression of the scenario, with each frame containing snapshots of the robots on the test track along with super-imposed trajectories depicting the paths taken by the vehicles.

the algorithm runtime distribution across 1000 randomized test scenario runs, with both core and thread counts restricted to one. The runtime distribution is illustrated in Fig. 9. Our algorithm achieves an average computation time of 0.0424 seconds and a maximum of 0.0758 seconds, demonstrating its capability for real-time operation. Furthermore, even greater computational efficiency could be achieved if implemented in a compiled language such as C/C++.

E. Hardware Demonstration

We demonstrate the performance of FCP in a scenario involving four customized 1/10-scale Multi-agent System for non-Holonomic Racing (MuSHR) [35] autonomous vehicles—comprising one ego vehicle, two stationary vehicles mimicking parked vehicles, and one moving vehicle—at our testing facility located at Honda Research Institute Inc USA, San Jose, CA. This scenario is similar to the simulation scenario shown in Fig. 8, with an added layer of difficulty for FCP to encounter a static obstacle on a curved path as soon as it turns left. The Robot Operating System (ROS)

serves as the communication framework, facilitating interaction between sensors, actuators, and computing units, and is executed within a Docker container. FCP runs inside a separate Docker container, ensuring efficient execution. The ego vehicle utilizes LiDAR based localization to estimate its state, including position, velocity, and orientation, using a predefined grid map of the track and surrounding landmarks. The planner operates at a frequency of 10Hz and runs on an Intel NUC mini PC onboard the MuSHR vehicle, which has Ubuntu 20.04 LTS as its operating system (OS). The ego vehicle's trajectory is visualized through sequentially numbered frames in Fig. 10 with the paths taken by the ego and the dynamic obstacle superimposed in each frame. The successful deployment of FCP on the physical robot, along with its smooth execution, highlights its computational efficiency and robustness in handling real-world uncertainties, perception noises and delays.

V. CONCLUSION

We propose a computationally efficient risk-aware local path planning algorithm to generate smooth deviation paths in reference to a fixed path for automated driving applications. Validation results from CARLA simulations, comparative analysis and scaled autonomous vehicle tests demonstrate the effectiveness of our approach.

REFERENCES

- [1] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *The international journal of robotics research*, vol. 5, no. 3, pp. 72–89, 1986.
- [2] T. Fraichard and C. Laugier, "Dynamic trajectory planning, path-velocity decomposition and adjacent paths," in *IJCAI*, 1993, pp. 1592–1599.
- [3] A. M. Añon, S. Bae, M. Saroya, and D. Isele, "Multi-profile quadratic programming (mpqp) for optimal gap selection and speed planning of autonomous driving," *arXiv preprint arXiv:2401.06305*, 2024.
- [4] S. Koenig and M. Likhachev, "D* lite," in *Eighteenth national conference on Artificial intelligence*, 2002, pp. 476–483.
- [5] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic a*: An anytime, replanning algorithm," in *ICAPS*, vol. 5, 2005, pp. 262–271.
- [6] D. Isele, A. M. Anon, F. M. Tariq, G. Yeh, A. Singh, and S. Bae, "Delayed-decision motion planning in the presence of multiple predictions," *arXiv preprint arXiv:2502.20636*, 2025.
- [7] Z. Huang, H. Chen, and K. Driggs-Campbell, "Neural informed rrt* with point-based network guidance for optimal sampling-based path planning," *arXiv preprint arXiv:2309.14595*, 2023.
- [8] A. Li, S. Bae, D. Isele, R. Beeson, and F. M. Tariq, "End-to-end predictive planner for autonomous driving with consistency models," *arXiv preprint arXiv:2502.08033*, 2025.
- [9] Y. K. Hwang, N. Ahuja *et al.*, "A potential field approach to path planning," *IEEE transactions on robotics and automation*, vol. 8, no. 1, pp. 23–32, 1992.
- [10] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [11] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, pp. 533–579, 2010.
- [12] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," 2011.
- [13] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.
- [14] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 2061–2067.
- [15] D. Dirckx, M. Bos, W. Decré, and J. Swevers, "Optimal and reactive control for agile drone flight in cluttered environments," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 6273–6278, 2023.
- [16] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.
- [17] F. M. Tariq, D. Isele, J. S. Baras, and S. Bae, "Rcms: Risk-aware crash mitigation system for autonomous vehicles," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2023, pp. 3950–3957.
- [18] T. Phan-Minh, E. C. Grigore, F. A. Boulton, O. Beijbom, and E. M. Wolff, "CovertNet: Multimodal behavior prediction using trajectory sets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 14 074–14 083.
- [19] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.
- [20] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and systems*, vol. 2. Ann Arbor, MI, USA, 2016, pp. 1–9.
- [21] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, "Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6064–6071.
- [22] S. Bae, D. Isele, A. Nakhaei, P. Xu, A. M. Añon, C. Choi, K. Fujimura, and S. Moura, "Lane-change in dense traffic with model predictive control and neural networks," *IEEE Transactions on Control Systems Technology*, vol. 31, no. 2, pp. 646–659, 2022.
- [23] H. Hu, D. Isele, S. Bae, and J. F. Fisac, "Active uncertainty reduction for safe and efficient interaction planning: A shielding-aware dual control approach," *The International Journal of Robotics Research*, p. 02783649231215371, 2023.
- [24] F. M. Tariq, N. Suriyarachchi, C. Mavridis, and J. S. Baras, "Autonomous vehicle overtaking in a bidirectional mixed-traffic setting," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 3132–3139.
- [25] F. Frenet, "Sur les courbes à double courbure," *Journal de mathématiques pures et appliquées*, vol. 17, pp. 437–447, 1852.
- [26] F. M. Tariq, D. Isele, J. S. Baras, and S. Bae, "Slas: Speed and lane advisory system for highway navigation," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, 2022, pp. 6979–6986.
- [27] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [28] T. Tashiro, "Vehicle steering control with mpc for target trajectory tracking of autonomous reverse parking," in *2013 IEEE international conference on control applications (cca)*. IEEE, 2013, pp. 247–251.
- [29] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [30] Y.-Y. Song and L. Ying, "Decision tree methods: applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.
- [31] O. Sharma, N. C. Sahoo, and N. B. Puhana, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Engineering applications of artificial intelligence*, vol. 101, p. 104211, 2021.
- [32] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE intelligent vehicles symposium (IV)*. IEEE, 2015, pp. 1094–1099.
- [33] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," 2013.
- [34] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.
- [35] S. S. Srinivasa *et al.*, "Mushr: A low-cost, open-source robotic racecar for education and research," *arXiv preprint arXiv:1908.08031*, 2019.