2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Flexible Trajectory Planning for Autonomous Vehicles via Environmental Assessment in Extreme Scenarios

Xiang Li[1], Ke Lin[1], Xiaoqing Yang[1], Kejian Yan[1], Yanjie Li[1] and Yunjiang Lou[1]

***Abstract*— Trajectory planning is a core task in autonomous driving. However, in diverse extreme scenarios characterized by unstructured obstacles, there is a lack of solutions that provide efficient computation, safety, and scene generalization capabilities. To address this issue, we propose a two-stage spatio-temporal joint trajectory planning method based on environmental assessment. In the first stage, we introduce the EAHybrid A* algorithm, which generates high-quality initial trajectories by evaluating environmental complexity, thereby significantly improving computational efficiency. The second stage formulates the trajectory planning problem as an optimal control problem, utilizing environmental assessment for joint spatio-temporal optimization, ensuring kinematic feasibility and obstacle avoidance. Experiments demonstrate that our method achieves higher success rates and planning speeds in extreme scenarios compared to state-of-the-art planning methods. Moreover, we have deployed and validated this approach in the CARLA simulator and real vehicles, proving its effectiveness and robustness in handling extreme environments.**

I. Introduction

Recent advancements in autonomous driving have facilitated the development of autonomous vehicles, which assist human drivers and enhance travel convenience. Core modules include perception, prediction, planning, and control, with planning playing a crucial role [1]. Unlike waypoint or path planning, which only considers spatial information, trajectory planning generates smooth and feasible trajectories by assigning velocities and poses over time, enabling safe and efficient navigation in constrained environments [2]. This capability is especially critical in unstructured extreme scenarios (see Section III-A). However, trajectory planning in such scenarios remains challenging. Algorithms must generalize well across diverse environments (e.g., parking lots and mining sites), accurately model nonlinear kinematics, and ensure obstacle avoidance. Additionally, balancing safety, quality, and real-time efficiency in spatio-temporal planning is difficult [3]. Existing methods often struggle to provide robust and adaptable solutions in these complex situations.

In this paper, we propose an efficient trajectory planning algorithm with strong scene generalization capabilities for extreme scenarios, as shown in Fig. 1. Our two-stage spatio-temporal joint trajectory planning approach incorporates environmental assessment, as depicted in Fig. 2. In the first stage, we enhance the Hybrid A* algorithm [4] to quickly generate rough initial trajectories by calculating environmental complexity. The second stage formulates an optimal control problem (OCP) to optimize the trajectory, using safety corridors for complex obstacle avoidance, environmental assessment to compute environmental complexity and optimize sampling time. This method improves environmental assessment and softens kinematic constraints, ensuring safety, feasibility, and generalization in extreme scenarios. The main contributions and innovations of this paper are as follows:

1) We propose an improved Hybrid A* algorithm that incorporates environmental assessment to guide the search process, enabling more efficient coarse trajectory generation with reduced computational cost.
2) We constructed an OCP to optimize the coarse trajectory for spatio-temporal joint planning with safety corridors and environmental assessment.
3) Our algorithm is validated in both the CARLA simulator and real-world autonomous driving scenarios, demonstrating its effectiveness and robustness, and supporting data collection for downstream tasks.

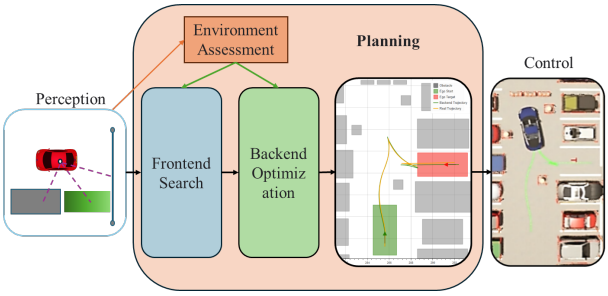

Fig. 1. Illustration of the overall structure. We adopt the two-stage trajectory planning method based on environmental assessment to achieve efficient and safe trajectory planning in diverse extreme scenarios. Related video demonstrations are provided at https://youtu.be/hLDqkyEGJfs.

II. Related Work

Trajectory planning focuses on strict spatial obstacle avoidance and temporal optimization. Existing methods for complex scenarios include sampling-based and geometry-based trajectory generation combined with trajectory optimization [3]. Additionally, advancements in neural networks have led to the application of learning-based approaches [5].

1) Geometry-Based Trajectory Generation: This approach relies on geometric modeling of the environment, using special curves with limited curvature to generate trajectories. A common example is the Reeds-Shepp (RS) curves,

This work was supported by Shenzhen Science and Technology Plan Project ZDCY20250901095402003, KJZD20230923114222045, SYSPG20241211173609005 and the Open Fund of Innovation Center for Control Actuators. (*Corresponding author: Yanjie Li, autolyj@hit.edu.cn.*)

[1]The authors are with the Shenzhen Key Lab for Advanced Motion Control and Modern Automation Equipments, School of Intelligence Science and Engineering, Harbin Institute of Technology, Shenzhen 518055, China.

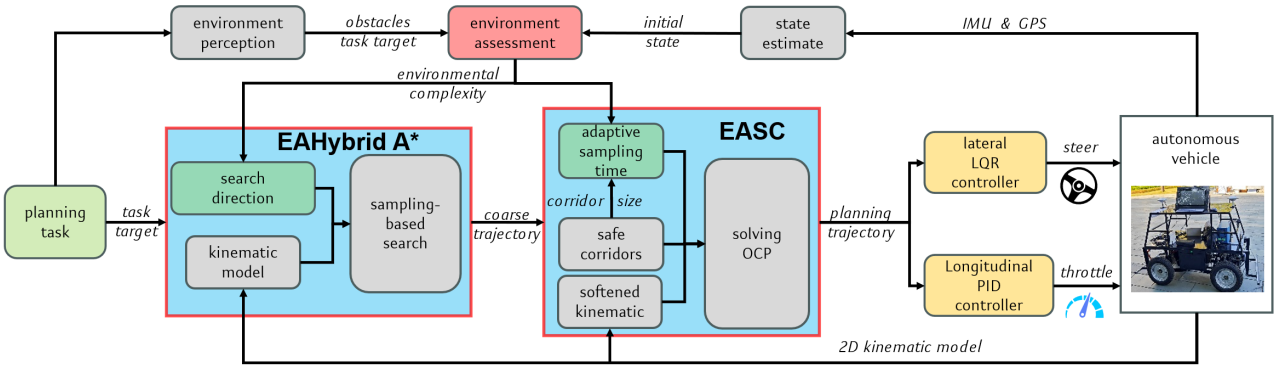

Fig. 2. Illustration of the pipeline of the trajectory planning method based on environmental assessment. The frontend search employs the EAHybrid A* algorithm with environmental assessment to generate a coarse trajectory, the backend optimization EASC refines this trajectory, incorporating temporal dynamics and leveraging safety corridors along with environmental complexity. Then the trajectory is sent to the controller for execution.

which represent trajectories with arcs and line segments, providing quick generation [6]. However, it struggles with abrupt curvature changes, which have led to the use of techniques like Bezier curves and B-splines for smoothing [7], [8]. Due to its dependence on geometric modeling, this approach performs poorly in unstructured extreme scenarios and has limited generalization capabilities.

2) Sampling-Based Trajectory Generation: This method samples the state or control space to find trajectories from the start point to the target. Key examples include the Rapidly-exploring Random Tree (RRT) [9], enhanced with guidance and kinematic constraints for better efficiency [10]. Graph search-based planners discretize the space into a graph structure. In autonomous driving, methods like Hybrid A* [4] leverage nonlinear vehicle kinematics, while RHybrid A* [11] employs bidirectional search for faster search in high-dimensional spaces. These planners provide high planning quality. While sampling-based methods handle local minima in non-convex spaces better than geometry-based methods, they still necessitate additional trajectory optimization.

3) Trajectory Optimization: In structured environments, trajectory optimization is often performed in the frenet coordinate system, which aligns with human driving behavior [12], [13]. However, this approach relies on global planning. Therefore, in diverse scenarios, trajectory planning is typically done in Cartesian coordinates to accommodate non-complete kinematics. Initially, this method optimized only spatial dimensions. The Convex Elastic Band Smoothing (CES) algorithm [14], [15] transforms trajectory optimization into a convex problem with quadratic objectives (QSQP). To integrate trajectory space with kinematics, planning is modeled as an Optimal Control Problem (OCP) and converted into a nonlinear programming (NLP) form, necessitating complex obstacle avoidance and non-complete kinematic constraints. The Optimization-Based Collision Avoidance (OBCA) algorithm [16], [17] uses dual variables to redefine distance, and the triangle-area-based criterion (TAC) uses area-based methods to describe the positional relationship between a point and polygons [18], integrating with Model Predictive Control (MPC). However, as the number of obstacles increases, both methods face significant challenges, leading to higher problem dimensions and solution difficulties. A lightweight iterative optimization framework [19] has been proposed to accelerate solutions, this can compromise feasible solutions in extreme environments. Differential flatness methods parameterize trajectories as polynomial curves, achieving robust obstacle avoidance [3], [20], [21], yet they struggle with non-convex obstacles in narrow spaces.

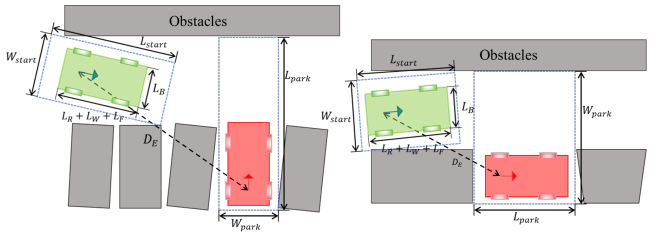

Fig. 3. Illustration of extreme scenario classification parameters in two typical scenarios.

4) Learning-Based Trajectory Planning: Learning-based methods include primarily imitation and reinforcement learning. The former mimics expert data [22], but struggles to generalize beyond the training set and cannot surpass the performance of its demonstrations. Reinforcement learning learns policies through environment interaction and can handle complex scenarios. HOPE [23] combines RL with traditional planners, achieving strong performance. However, as autoregressive methods, learning-based planners may suffer from short-sightedness and local infeasibility in unstructured environments.

III. Frontend: Sampling-Based Search

In this section, we define extreme scenarios and detail the generation of an initial spatial trajectory (comprising x, y, and θ states) using the Hybrid A* algorithm integrated with Environmental Assessment to enhance the search efficiency.

A. Extreme Scenarios Definition

We define extreme scenarios based on scene classification methods from ISO 20900 [24] and HOPE [23]. We comprehensively consider both the available maneuvering space

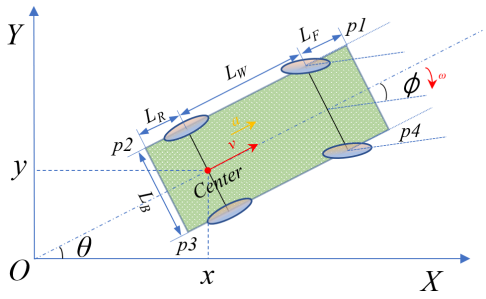

Fig. 4. Schematic of the vehicle's kinematic model, assuming a rigid suspension, rear-wheel drive, and front-wheel steering.

and the expected planning trajectory length. By expanding the states of these points, we derive maximum bounding boxes encompassing the start and end points. Accounting for the vehicle's degrees of freedom in different directions, we measure the lengths and widths of these bounding boxes as $W_{start}, L_{start}, W_{end}, L_{end}$. Additionally, we incorporate scenarios with longer task distances by including the Euclidean distance D_E from the start point to the end point. Furthermore, we define scenarios satisfying the condition in (1) as extreme scenarios, where L_R, L_W, L_F, and L_B represent geometric parameters of the vehicle, depicted in Fig. 4.

$$\begin{aligned} &L_{end}, L_{start} < 1.15 \cdot (L_R + L_W + L_F) \quad \text{or} \\ &W_{start}, W_{end} < 0.4 + L_B \quad \text{or} \quad D_E > 15m \end{aligned} \tag{1}$$

B. Sampling-based Search

To expedite the solving process, we opt for a classic 2D bicycle kinematic model. The relevant geometric representation of the vehicle's kinematic model is illustrated in Fig. 4. We use a simplified vehicle state equation as:

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{v}(t) \\ \dot{\phi}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\theta(t)) \\ v(t)\sin(\theta(t)) \\ a(t) \\ \omega(t) \\ \frac{v(t)\tan(\phi(t))}{L_W} \end{bmatrix}, t \in [0, T] \tag{2}$$

$\mathbf{x}(t)$ denotes the vehicle state at time t, $(x(t), y(t))$ is the midpoint coordinate of the rear axle of the vehicle, $v(t)$ and $a(t)$ are the vehicle speed and acceleration, $\phi(t)$ and $\omega(t)$ are the steering angle and angular velocity, $\theta(t)$ is the vehicle's heading angle, and L_W is the wheelbase. The state and control variables are defined as $\mathbf{x}(t) = [x(t), y(t), v(t), \phi(t), \theta(t)]$ and $\mathbf{u}(t) = [a(t), \omega(t)]$. Accordingly, the kinematic model in (2) can be compactly written as $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$.

C. Environmental Assessment

The traditional Hybrid A* algorithm consists of a heuristic guided sampling-based search, followed by a final connection using RS curves. While RS curves can quickly connect two vehicle states, they ignore obstacles and often fail in narrow or irregular environments. In the sampling-based search, we adopt an A* heuristic that omits the heading angle θ, which in turn causes the planner to ignore vehicle kinematics. In constrained regions, the resulting frequent steering reversals further degrade guidance quality. To improve planning efficiency, we prefer sampling-based search in narrow spaces, while leveraging RS curves for quick connections in simpler regions. In the frontend search, we evaluate the environmental complexity near the start and goal states to adaptively choose between sampling-based search and direct RS curve connection, as shown in (3).

Algorithm 1 EAHybrid A*

Input: Vehicle state $\mathbf{x}$, obstacles O.
Procedure:
1: Initialize $C_{start}, C_{end} = 0$;
2: **for** each direction $i \in$ {h, b, l, r}; **do**
3: **repeat**
4: Inflate the vehicle in direction i according to the search resolution and record the distance D_i;
5: **until** collision with obstacles O.
6: **end for**
7: Calculate the task environmental complexity C_{start} and C_{end} using (3).
8: **if** $C_{\text{start}} > C_{\text{end}}$ **then**
9: Use forward Hybrid A* from $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{end}}$.
10: **else**
11: Reverse vehicle from $t = T$ to 0 based on (2)
12: Use reverse Hybrid A* from $\mathbf{x}_{\text{end}}$ to $\mathbf{x}_{\text{start}}$.
13: **end if**
Output: **Coarse trajectory** from $\mathbf{x}_{\text{start}}$ to $\mathbf{x}_{\text{end}}$.

$$\begin{aligned} C &= \sum_{i \in \{\text{h, b, l, r}\}} w_i \cdot \left(B_i \cdot M + (1 - B_i) \cdot \frac{1}{D_i^{\alpha}} \right) \\ B_i &= \begin{cases} 1, & \text{if } D_i < D_{narrow} \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{3}$$

To compute the environmental complexity C, we expand the vehicle in four directions {h, b, l, r} and record the distance D_i to the nearest obstacle in each four directions. Directions with $D_i < D_{\text{narrow}}$ are marked as blocked ($B_i = 1$), while others retain their distances. The weights w_i reflect the maneuver difficulty: forward and backward motions are easier, so lateral directions are assigned higher weights. A large constant M amplifies the influence of blocked directions, and a nonlinear factor $\alpha \in [0.5, 1]$ emphasizes nearby obstacles. Based on this, we compute the coarse environmental complexity around the start and goal points, denoted as C_{start} and C_{end}. The coarse complexity is used to select the search directions in the proposed Environmental Assessment Hybrid A* (EAHybrid A*), as shown in Alg. 1.

IV. Backend: Numerical Optimization

This section refines the initial coarse spatial trajectory (Section III) into a spatio-temporal trajectory containing $x(t), y(t), \theta(t), v(t), \phi(t), a(t), \omega(t)$. We solve the OCP with the core idea of environmental assessment, which allows us to quickly obtain high-quality optimized trajectories.

A. Formulation of Kinematic Constraints

In the trajectory planning for autonomous driving, we can usually construct the following numerical optimal control problem framework [17].

$$\min_{\mathbf{x}(t),\mathbf{u}(t)} \; J(\mathbf{x}(t),\mathbf{u}(t)), \tag{4a}$$

$$\text{s.t.,}\ \mathbf{x}(0)=\mathbf{x}_{\text{init}}\ ,\quad \mathbf{u}(0)=\mathbf{u}_{\text{init}}\ ,$$
$$\mathbf{x}(T)=\mathbf{x}_{\text{target}}\ ,\quad \mathbf{u}(T)=\mathbf{u}_{\text{target}}\ ; \tag{4b}$$
$$\dot{\mathbf{x}}(t)=f(\mathbf{x}(t),\mathbf{u}(t)),$$
$$\underline{\mathbf{x}}\le\mathbf{x}(t)\le\overline{\mathbf{x}},\quad \underline{\mathbf{u}}\le\mathbf{u}(t)\le\overline{\mathbf{u}}; \tag{4c}$$
$$\mathbb{E}(\mathbf{x}(t))\cap\mathbb{O}=\emptyset,\quad t\in[0,T]. \tag{4d}$$

The trajectory planning problem is formulated as an Optimal Control Problem (OCP), where $\mathbf{x}(t)$ and $\mathbf{u}(t)$ denote the state and control variables of the vehicle. The OCP comprises four components: 1) the objective function J (4a); 2) boundary constraints on $\mathbf{x}(t)$ and $\mathbf{u}(t)$ at the start and goal (4b); 3) kinematic constraints (4c); 4) obstacle avoidance constraints (4d), where $\mathbb{E}(\mathbf{x}(t))$ is the vehicle geometry and $\mathbb{O}$ denotes the obstacle set. Details of each component are introduced below.

To facilitate the solution of the OCP and to construct the obstacle avoidance constraints, it is essential to discretize the continuous variables $\mathbf{x}(t)$ and $\mathbf{u}(t)$ into $\mathbf{x}(k)$ and $\mathbf{u}(k)$. We utilize cubic Hermite interpolation to convert the continuous form into a discrete form, as shown in (5).

$$\begin{aligned}
K_1 &= f(\mathbf{x}(k),\mathbf{u}(k))\\
K_2 &= f(\mathbf{x}(k+1),\mathbf{u}(k+1))\\
K_3 &= \frac{1}{2}(\mathbf{x}(k)+\mathbf{x}(k+1))+\frac{t(k)}{8}(K_1-K_2)\\
\mathbf{x}(k+1) &= \mathbf{x}(k)+\frac{t(k)}{6}(K_1+K_2)+\frac{2t(k)}{3}\\
&\quad\cdot f\left(K_3,\frac{1}{2}(\mathbf{u}(k)+\mathbf{u}(k+1))\right)
\end{aligned} \tag{5}$$

At this point, we could express the basic form of the optimization objective as J_0, which describes the comfort and energy efficiency of the trajectory through $\mathbf{u}(k)$ mentioned in [25], as shown in (6).

$$J_0=\sum_{k=1}^{N-1}(||\mathbf{u}(k+1)-\mathbf{u}(k)||^2+||\mathbf{u}(k)||^2) \tag{6}$$

B. Safe Corridor and Soft Constraints

In traditional OCP formulations, obstacle avoidance is modeled as a geometric constraint between the ego vehicle and obstacles, which is highly non-convex and difficult to handle directly in optimization. Inspired by drone navigation [26], we segment the vehicle's rectangular body using three circles to construct safety corridors, balancing geometric accuracy and computational efficiency, as shown in Fig. 5.

By simplifying the vehicle's geometric shape into circles, we are able to construct the safety corridors as the simplest enclosing rectangles. To simplify the calculations, we expand the enclosing rectangles of the circles covering the vehicle

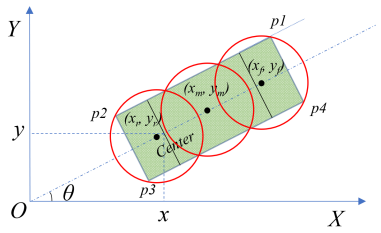

Fig. 5. Using 3 circles to cover the whole vehicle.

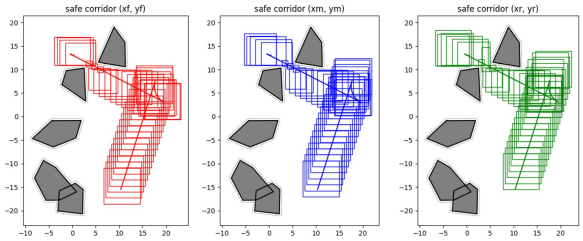

Fig. 6. The gray irregular shape represents expanded obstacles, while the red, blue, and green indicate discrete safety corridorsz along the initial trajectory.

along the initial trajectory until a collision occurs. This results in three discrete safety corridors, as shown in Fig. 6. Consequently, we only need to impose bounds on the coordinates of the centers of the three circles $(x^j(k),y^j(k)), j\in\{\text{f, m, r}\}$. This method transforms the constraints to the simplest convex constraints for three circles, as shown in (7).

$$\begin{bmatrix} x^j_{\min}(k)\\ y^j_{\min}(k)\end{bmatrix}\le\begin{bmatrix} x^j(k)\\ y^j(k)\end{bmatrix}\le\begin{bmatrix} x^j_{\max}(k)\\ y^j_{\max}(k)\end{bmatrix} \tag{7}$$

We also incorporate the constraints into the optimization objective by assigning them a significantly high weight ω_{dyn} (e.g., 10^8) to minimize them, thereby softening the hard constraints. As a result, the kinematic constraints are transformed into (8), which consists of two components.

$$\begin{aligned}
J_{dyn} &= \sum_{k=1}^{N-1}(\mathbf{x}(k+1)-\mathbf{x}(k)-\frac{t(k)}{6}(K_1+K_2)\\
&\quad-\frac{2t(k)}{3}\cdot f\left(K_3,\frac{1}{2}(\mathbf{u}(k)+\mathbf{u}(k+1))\right))^2\\
&\quad+\sum_{k=1}^{N}\sum_{j}^{f,m,r}((x^j(k)-\tilde{x}^j(k))^2+(y^j(k)-\tilde{y}^j(k))^2)
\end{aligned} \tag{8}$$

The first is the original kinematic constraint derived from (5), the second is the augmented kinematic constraint for the safety corridors, considering the rigid-body characteristics and the coordinates of the circle centers. Here, $(x^j(k),y^j(k))$ represents the optimization variable for the center coordinates of the circles, while $(\tilde{x}^j(k),\tilde{y}^j(k))$ represents the true center coordinates calculated using vehicle's state $(x(k),y(k),\theta(k))$ and geometric parameters in Fig. 4.

C. Adaptive Sampling Time via Environmental Assessment

Although safety corridors and soft constraints improve the OCP, its performance still depends on the initial trajectory. In

initial trajectory generation, uniform node sampling leads to constant-speed motion, which is often unsuitable for turning and reversing maneuvers. To address this, we introduce variable sampling time $t(k)$, which affects vehicle speed during discretization. Larger $t(k)$ yields slower motion for narrow spaces, while smaller $t(k)$ allows faster motion in open areas. By evaluating environment complexity, we adaptively adjust $t(k)$ to improve optimization feasibility and convergence. Thanks to the corridors in (7), a more detailed evaluation of environmental complexity around the initial trajectory can be performed before OCP solving. The environmental complexity $S(k)$ at each trajectory point is then calculated as shown in (9).

$$\begin{aligned} A^j(k) &= \left\| x^j_{\max}(k) - x^j_{\min}(k) \right\|_1 \cdot \left\| y^j_{\max}(k) - y^j_{\min}(k) \right\|_1 \\ \{A^{(1)}(k), A^{(2)}(k)\} &= \text{TwoSmallest}_{j\in\{f,m,r\}} \left(A^j(k)\right) \\ S(k) &= w_1 \cdot A^{(1)}(k) + w_2 \cdot A^{(2)}(k) \end{aligned} \tag{9}$$

Here, $A^j(k)$ represents the area of the j-th safety corridor at the k-th trajectory point. Considering the kinematics of the vehicle as rigid body kinematics, some of the more lenient corridors among the multiple safety corridors are still constrained by the feasible regions of the two smallest corridors. Therefore, $S(k)$ takes the weighted sum of the two smallest $A^j(k)$ values selected from $j \in \{f, m, r\}$, denoted as $A^{(1)}(k)$ and $A^{(2)}(k)$.

Based on the environmental complexity $S(k)$ at each trajectory point and a predefined threshold τ_t, the sampling time $t(k)$ is classified as either $t_n(k)$ or $t_w(k)$, corresponding to narrow and wide environments, shown in Fig. 7. Then we introduce these two sampling times into the optimization J_t in (10). Additionally, to ensure that the sampling time strictly meets the basic requirements, we introduce constraints as shown in (11).

$$J_t = \left\| \begin{bmatrix} t_n(k) - t_{n\max} \\ t_w(k) - t_{w\min} \end{bmatrix} \right\|_2^2 \tag{10}$$

$$\begin{bmatrix} t_{n\min} \\ t_{w\min} \end{bmatrix} \leq \begin{bmatrix} t_n(k) \\ t_w(k) \end{bmatrix} \leq \begin{bmatrix} t_{n\max} \\ t_{w\max} \end{bmatrix} \tag{11}$$

Here, $[t_{n\min}, t_{w\min}]$ and $[t_{n\max}, t_{w\max}]$ denote the corresponding bounds, which are determined according to the discrete spacing of the coarse trajectory and the vehicle's velocity limits.

D. Final OCP Formulation

Finally, we arrive at the OCP as follows (12). The objective (12a) comprising three terms: the comfort and energy efficiency term (6), soft kinematic constraints (8), and sampling time optimization (10). $\omega_0, \omega_{dyn}, \omega_t$ denote the corresponding objective weights. The constraints in the optimization problem can be grouped into four main categories. (12b) imposes box constraints on the state and control variables, (12c) enforces boundary conditions at the start and goal states, the sampling times are bounded as shown in (12d) and detailed in (11), obstacle avoidance is handled via the safety corridor variables $\mathbb{C} = [x^j(k), y^j(k)]_{j\in f,m,r}$ as show

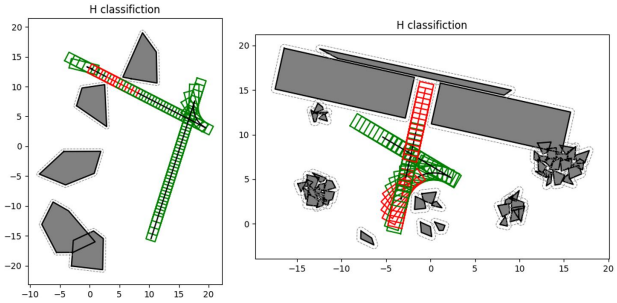

Fig. 7. The green area indicates a wide environment, while the red area represents a narrow environment. The gray regions represent the expanded obstacles.

Algorithm 2 Trajectory Optimization(EASC).

Input: Trajectory to be optimized $\mathbb{P}$

Procedure:

1: **for** each point **pt** $\in \mathbb{P}$ **do**
2: Generate the safety corridor $\mathbb{C}$ for **pt**;
3: Compute $S(k)$ for **pt** by (9);
4: **end for**
5: Segment $t_n(k)$ and $t_w(k)$ based on $S(k)$ by (10)and (11);
6: Soften the kinematic constraints by (8);
7: Construct and solve the OCP by (12) to obtain the final trajectory $\mathbb{P}_{FIN}$;

Output $\mathbb{P}_{FIN}$

in (12e) and detailed in (7). All constraints are simplified to box-type inequalities, which facilitate efficient solving.

$$\min_{\mathbf{x}(k),\mathbf{u}(k),t(k)} \omega_0 \cdot J_0 + \omega_{dyn} \cdot J_{dyn} + \omega_t \cdot J_t \tag{12a}$$

$$\text{s.t.,} \underline{\mathbf{x}} \leq \mathbf{x}(k) \leq \overline{\mathbf{x}}, \quad \underline{\mathbf{u}} \leq \mathbf{u}(k) \leq \overline{\mathbf{u}}, \tag{12b}$$

$$\mathbf{x}(0) = \mathbf{x}_{\text{init}}, \quad \mathbf{u}(0) = \mathbf{u}_{\text{init}},$$

$$\mathbf{x}(N) = \mathbf{x}_{\text{target}}, \quad \mathbf{u}(N) = \mathbf{u}_{\text{target}}, \tag{12c}$$

$$\underline{\mathbf{t}} \leq \mathbf{t}(k) \leq \overline{\mathbf{t}} \tag{12d}$$

$$\underline{\mathbf{C}} \leq \mathbb{C} \leq \overline{\mathbf{C}}, \tag{12e}$$

$$\text{for} \quad k = 0, 1, ..., N.$$

By combining the context from the previous sections, we can outline our optimization framework as shown in Alg. 2.

V. Experiment Analysis

In this section, we present experimental results that validate our planner, both in simulated environments and real-world applications on autonomous vehicles.

A. Simulations

1) Setup and Results: The experiments in the simulation environment were conducted on a system equipped with an AMD Ryzen 7500F CPU and a GeForce RTX 3090 GPU. In this environment, we employed the IPOPT solver from the CasADi framework to solve the OCP [27]. We present the key parameters used in our method in Tab. I.

Considering the complexity and variability of extreme scenarios, we categorize them into two major types: open

TABLE I

KEY PARAMETERS SETTINGS

Symbol	Description	Value
L_B	Vehicle width	2.0 m
L_R	Rear hang distance	1.0 m
L_W	Wheelbase	2.5 m
L_F	Front hang distance	1.0 m
Δ_{xy}	x-y resolution	0.3 m
Δ_θ	Angle resolution	0.05 rad
D_{narrow}	Narrow space threshold	0.35 m
$w_{h,b,l,r}$	Weight for derections	$[1, 10, 30, 30]$
v_{bounds}	Velocity range	$[-5, 6]$ m/s
a_{bounds}	Acceleration range	$[-1, 1]$ m/s^2
ϕ_{bounds}	Steering angle range	$[-0.64, 0.64]$ rad
ω_{bounds}	Steering rate range	$[-0.64, 0.64]$ rad/s
N	Number of OPT points	200
h_{bounds}	Time interval range	$[0.05, 0.3]$ s
w_0	Smoothness weight in (6)	100
w_{dyn}	Dynamic weight in (8)	10^8
w_H	Sampling weight in (11)	5000

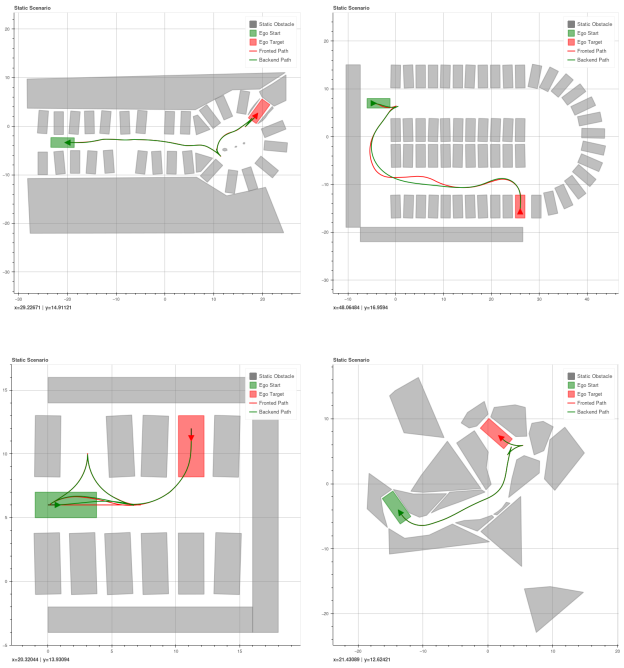

Fig. 8. Final planning results for some extreme scenarios. The green curve represents the final trajectory obtained from the backend optimization.

space and urban space. Open space primarily focuses on dealing with irregular static obstacles, while urban space mainly involves interactions with other traffic participants. To validate the performance of our algorithms in extreme environments, we carefully designed 22 open space and 11 urban space scenarios, in conjunction with the TPCAP competition [28]. Additionally, to demonstrate the universality and robustness of our algorithms, we automatically generated 124 extreme scenarios filled with irregular obstacles for testing.

The final planning results for some extreme scenarios are shown in Fig. 8. It is evident that our algorithm performs effectively in extreme scenarios.

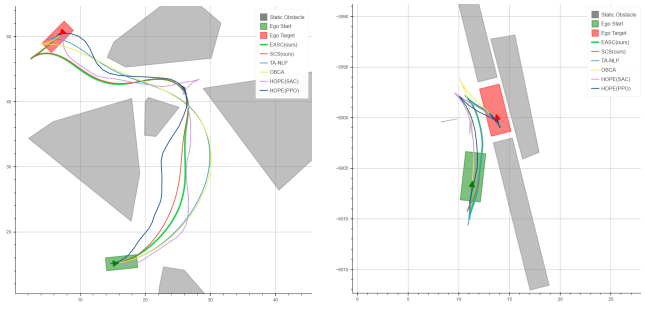

Fig. 9. Trajectory comparasion visualized in the typical extreme scenarios.

2) Frontend Methods: We first conduct frontend search algorithm validation. In extreme scenarios, most coarse trajectory generation methods struggle to achieve success. Therefore, we selected the Hybrid A* [4] and its derivative RHybrid A* [11], the latest reinforcement learning-based algorithm HOPE [23], and our proposed EAHybrid A* in both urban space and open space scenarios. Performance evaluation uses success rate (SR), average planning time (APT), maximum planning time (Max. PT), and minimum planning time (Min. PT). As shown in Table III, our frontend planner is capable of generating trajectories more efficiently while ensuring their validity. Furthermore, the environmental complexity computation in our method causes suboptimal Min. PT, but this computational overhead remains under 1 ms and is acceptable within the total cost.

3) Backend Methods: We present the experimental results of our backend methods. The SCS utilizes safety corridors and softened constraints, while EASC further incorporates environmental assessment to improve adaptability in complex environments. We compare our approach with four representative methods tailored for unstructured scenarios: OBCA [17], which models obstacles as hyperplanes; TAC [18], which applies a triangle-area-based collision criterion; and HOPE [23], which uses hybrid policy with PPO and SAC reinforcement learning. For fair comparison, all standard settings—including convergence criteria, dynamic limits, and vehicle dimensions—were kept consistent. All methods used the same initial trajectory from EAHybrid A* as a warm start, while learning-based methods optimized trajectories in an autoregressive fashion. To visualize performance, we selected two typical extreme scenarios where all methods succeeded, as shown in Fig. 9.

We conducted extensive quantitative comparisons using metrics including success rate (SR), average planning time (APT), 5% and 95% percentile planning time (PPT), average planning length (APL), average change of direction (ACD), average reverse rate (ARR), average planning acceleration (APA), and average planning jerk (APJ). As shown in Tab. II, our SCS and EASC achieved a 100% success rate and significantly lower planning times in 124 extreme scenarios, outperforming existing methods [17], [18], [23] in both efficiency and trajectory quality. While the acceleration may not be optimal, research [3] indicates that jerk-related metrics

TABLE II

EXTENDED COMPARISON OF BACKEND ALGORITHMS IN 124 RANDOMLY EXTREME SCENARIOS

Alg	SR(%)↑	APT(s)↓	PPT(s)		APL(m)↓	ACD↓	ARR(%)↓	APA(m/s^2)		APJ(m/s^3)	
			5%↓	95%↓				max↓	min↑	max↓	min↑
HOPE(PPO) [23]	74.194	2.248	0.660	5.395	45.572	5.888	50.938	0.732	-0.753	24.106	-24.415
HOPE(SAC) [23]	90.323	2.326	0.648	4.548	42.317	3.768	52.249	0.539	-0.579	14.579	-16.558
OBCA [17]	91.946	14.793	1.886	48.046	49.425	0.584	45.723	0.494	0.471	2.091	-2.274
TAC [18]	94.358	3.835	1.552	9.325	41.194	**0.271**	57.740	**0.340**	**-0.365**	2.966	-2.047
SCS(ours)	**100.000**	1.644	**0.803**	3.055	41.029	0.476	42.876	0.627	-0.517	1.782	-1.922
EASC(ours)	**100.000**	**1.418**	1.017	**1.973**	**38.877**	0.435	**42.473**	0.758	-0.697	**1.658**	**-1.646**

TABLE III

FRONTEND ALGORITHMS IN URBAN AND OPEN SPACES

Urban Space				
Alg	**SR(%)↑**	**APT(ms)↓**	**Min. PT(ms)↓**	**Max. PT(ms)↓**
HOPE [23]	54.545	3797.250	118.276	9211.585
Hybrid A* [4]	83.332	3474.324	11.992	20736.616
RHybrid A* [11]	91.674	2667.490	**3.683**	6581.065
EAHybrid A*(ours)	**100.000**	**1092.476**	3.713	**4580.558**
Open Space				
Alg	**SR(%)↑**	**APT(ms)↓**	**Min. PT(ms)↓**	**Max. PT(ms)↓**
HOPE [23]	68.182	2230.013	65.553	7435.246
Hybrid A* [4]	90.913	1097.434	**2.413**	23661.487
RHybrid A* [11]	**100.000**	421.264	3.501	3260.590
EAHybrid A*(ours)	**100.000**	**248.543**	3.226	**3254.716**

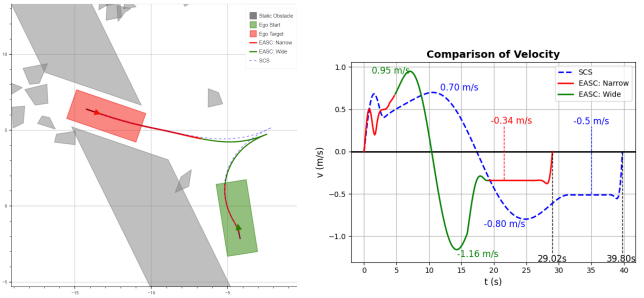

Fig. 10. Comparison of optimization results between SCS and EASC, including optimized trajectories and corresponding velocity profiles with annotated key speeds and maneuver durations.

are critical for influencing control effort and human comfort. Furthermore, our ablation study shows that incorporating environmental assessment in EASC improves both planning speed and comfort over SCS.

Since our method is a spatio-temporal joint trajectory planning approach, we simultaneously focus on the state variables that lie beneath the visualized trajectories. A comparison between the EASC and SCS methods is presented in Fig. 10. As a spatio-temporal planner, our method accounts for both trajectories and their underlying state variables. Fig. 10 compares EASC and SCS. By treating the sampling time $t(k)$ as an optimization variable, EASC achieves higher speeds in wide areas for faster task completion, and lower speeds in narrow spaces to improve trackability. Overall, the dynamic state data show that EASC better exploits the vehicle's maneuverability than SCS, enabling more efficient trajectory tracking.

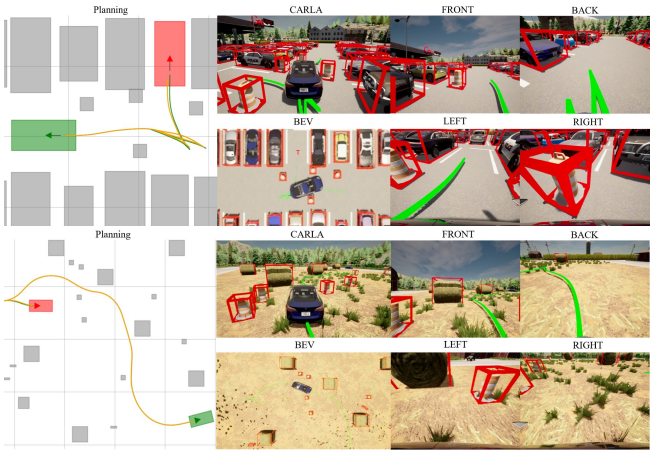

Fig. 11. Our approach, combining longitudinal and lateral decoupled control, demonstrates its performance in different scenarios.

4) Simulations in CARLA: Before real-world deployment, we tested our method in extreme CARLA scenarios [29] with realistic vehicle dynamics. A decoupled LQR and PID controller was used for lateral and longitudinal tracking, respectively, ensuring accurate trajectory following. Following this, we also collected rich sensor data (RGB and depth cameras, lidar, vehicle states, bird's-eye views) to support future end-to-end tasks, as shown in Fig. 11.

B. Real World Experiment

We deployed our algorithm on a real vehicle robot in a $30m \times 20m$ outdoor unstructured environment. To ensure safety, motion constraints were applied (velocity: $[-1, 1]m/s$, acceleration: $[-0.5, 0.5]m/s^2$, angular velocity: $[-0.27, 0.27]rad/s$), and the vehicle model was expanded by $0.2m$ to account for GPS and IMU inaccuracies. Localization was achieved using GPS and IMU with a 16-line lidar for environment perception. The objective was for the vehicle to maneuver from its initial state while avoiding obstacles and reach a predefined target position of $[x, y, \theta]$. During the trajectory planning process, to account for control errors and the inaccuracies associated with the global positioning provided by the combination of GPS and IMU, we expanded the vehicle model by 0.2 m in all four directions to ensure safety. Building upon this foundation, we implement our algorithm for rolling closed-loop trajectory planning. The planning trajectory was tracked in a closed-loop at 100 Hz

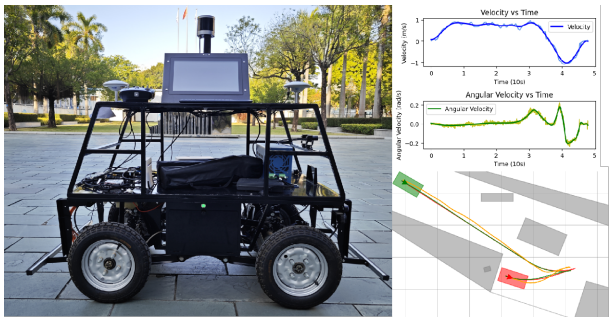

Fig. 12. The autonomous vehicle used in our real world experiments.

using LQR for lateral and PID for longitudinal control. The task is completed when the vehicle reaches the target slot. Fig. 12 demonstrates that the trajectory planned by our algorithm is feasible for control and tracking.

VI. Conclusion

In this work, we propose a spatio-temporal joint trajectory planner for autonomous driving in extreme and unstructured scenarios. The two-stage framework effectively integrates EAHybrid A* for frontend coarse trajectory searching with EASC for backend trajectory optimization. By combining environmental assessment, safety corridors, and spatio-temporal optimization, our method demonstrates significant performance improvements over existing state-of-the-art methods in extensive simulations and real-world experiments. However, the current implementation relies on high perception accuracy and has been validated with precise obstacle detection. Future work should focus on enhancing the planner's performance in under-sensed or partially observable environments and improving its robustness to uncertainties in perception.

References

[1] T. Siyu, H. Xuemin, D. Peng, *et al.*, "Motion Planning for Autonomous Driving: The State of the Art and Future Perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 3692–3711, 2023.

[2] X. Ren, L. Hongji, C. Yingbing, *et al.*, "A Generic Trajectory Planning Method for Constrained All-Wheel-Steering Robots," *arXiv preprint arXiv:2404.09677*, 2024.

[3] Z. Han, Y. Wu, T. Li, *et al.*, "An Efficient Spatial-Temporal Trajectory Planner for Autonomous Vehicles in Unstructured Environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1797–1814, 2024.

[4] S. Sedighi, D. V. Nguyen, and K. D. Kuhnert, "Guided Hybrid A-star Path Planning Algorithm for Valet Parking Applications," in *International Conference on Control, Automation and Robotics (ICCAR)*, 2019, pp. 570–575.

[5] Z. Yuan, Z. Wang, X. Li, *et al.*, "Hierarchical Trajectory Planning for Narrow-Space Automated Parking with Deep Reinforcement Learning: A Federated Learning Scheme," *Sensors*, vol. 23, no. 8, p. 4087, 2023.

[6] J. Reeds and L. Shepp, "Optimal Paths for A Car that Goes Both Forwards and Backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[7] T. Berglund, A. Brodnik, H. Jonsson, *et al.*, "Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 167–172, 2010.

[8] H. Vorobieva, N. Minoiu-Enache, S. Glaser, *et al.*, "Geometric Continuous-curvature Path Planning for Automatic Parallel Parking," in *IEEE international conference on networking, sensing and control (ICNSC)*. IEEE, 2013, pp. 418–423.

[9] S. LaValle, "Rapidly-exploring Random Trees: A New Tool for Path Planning," *Research Report 9811*, 1998.

[10] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, no. 1, 2013, pp. 5054–5061.

[11] P. He, Z. Xu, X. Long, *et al.*, "Path Planning of Mobile Robot Based on Improved A-Star Bidirectional Search Algorithm," in *International Conference on Unmanned Systems (ICUS)*, 2023, pp. 1517–1522.

[12] J. Chen, C. Liu, and M. Tomizuka, "FOAD: Fast Optimization-based Autonomous Driving Motion Planner," in *Annual American Control Conference (ACC)*, 2018, pp. 4725–4732.

[13] C. Liu, C.-Y. Lin, and M. Tomizuka, "The Convex Feasible Set Algorithm for Real Time Optimization in Motion Planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[14] Z. Zhijie, E. Schmerling, and M. Pavone, "A Convex Optimization Approach to Smooth Trajectories for Motion Planning with Car-like Robots," in *IEEE Conference on Decision and Control (CDC)*, vol. 1, no. 1, 2015, pp. 835–842.

[15] J. Zhou, R. He, Y. Wang, *et al.*, "Autonomous Driving Trajectory Optimization With Dual-Loop Iterative Anchoring Path Smoothing and Piecewise-Jerk Speed Optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 439–446, 2021.

[16] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-Based Collision Avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[17] R. He, J. Zhou, S. Jiang, Y. Wang, *et al.*, "TDR-OBCA: A Reliable Planner for Autonomous Driving in Free-Space Environment," in *American Control Conference (ACC)*, 2021, pp. 2927–2934.

[18] B. Li, T. Acarman, Y. Zhang, *et al.*, "Tractor-Trailer Vehicle Trajectory Planning in Narrow Environments With a Progressively Constrained Optimal Control Approach," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 3, pp. 414–425, 2020.

[19] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, *et al.*, "Optimization-Based Trajectory Planning for Autonomous Parking With Irregularly Placed Obstacles: A Lightweight Iterative Framework," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 11 970–11 981, 2022.

[20] J. Arrizabalaga, Z. Manchester, and M. Ryll, "Differentiable Collision-Free Parametric Corridors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 1839–1846.

[21] A. Jaitly, J. Arrizabalaga, and G. Li, "Trajectory Planning Using Safe Ellipsoidal Corridors as Projections of Orthogonal Trust Regions," 2025. [Online]. Available: https://arxiv.org/abs/2509.19734

[22] R. Chai, D. Liu, T. Liu, *et al.*, "Deep Learning-Based Trajectory Planning and Control for Autonomous Ground Vehicle Parking Maneuver," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 3, pp. 1633–1647, 2023.

[23] M. Jiang, Y. Li, S. Zhang, *et al.*, "HOPE: A Reinforcement Learning-Based Hybrid Policy Path Planner for Diverse Parking Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 5, pp. 6130–6141, 2025.

[24] *Intelligent Transport Systems—Partially-Automated Parking Systems (PAPS)—Performance Requirements and Test Procedures*, International Organization for Standardization Standard ISO 20 900, 2023.

[25] C. Ko, S. Han, M. Choi, *et al.*, "Integrated Path Planning and Tracking Control of Autonomous Vehicle for Collision Avoidance based on Model Predictive Control and Potential Field," in *International Conference on Control, Automation and Systems (ICCAS)*, 2020, pp. 956–961.

[26] S. Liu, M. Watterson, K. Mohta, *et al.*, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[27] J. A. Andersson, J. Gillis, G. Horn, *et al.*, "CasADi: A Software Framework for Nonlinear Optimization and Optimal Control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[28] B. Li, L. Fan, Y. Ouyang, *et al.*, "Online Competition of Trajectory Planning for Automated Parking: Benchmarks, Achievements, Learned Lessons, and Future Perspectives," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 16–21, 2023.

[29] D. Alexey, R. German, C. Felipe, *et al.*, "CARLA: An Open Urban Driving Simulator," in *Annual Conference on Robot Learning (ACRL)*, 2017, pp. 1–16.