TriPlanner: A Tri-Stage Planner for Safe Vehicle Navigation in Complex Environments

Zhaolun Li[1], Jinya Su[2], *IEEE Senior Member*, Wang Pan[1]

***Abstract*—In complex urban environments, ensuring both safety and feasibility in autonomous vehicle trajectory planning remains a critical challenge. This paper presents TriPlanner, a tightly coupled three-stage framework for robust and real-time safe trajectory generation. The first stage employs a lattice-based planner enhanced with the Separating Axis Theorem (SAT), enabling precise and computationally efficient collision filtering among candidate trajectories. The second stage constructs orientation-aware convex safety corridors via an iterative maximum-inscribed-ellipse expansion. This process utilizes a novel analytical fast Small-Dimensional Minimum-Norm (SDMN) solver and affine normalization to rapidly generate separating hyperplanes, significantly reducing computational overhead compared to conventional quadratic programming methods. The third stage formulates a nonlinear trajectory optimization problem that integrates vehicle dynamics, control limits, and polygon-based spatial convex constraints to generate smooth, safe, and dynamically feasible trajectories. Quantitatively, the framework achieves a 29.37% reduction in mean absolute curvature and enlarges the average safe corridor area by up to 159%, highlighting its ability to generate smoother, safer, and more efficient trajectories in dense obstacle scenarios.**

***Index Terms*—Autonomous vehicles, trajectory planning, lattice planner, convex safety corridor, nonlinear optimization**

I. Introduction

Autonomous navigation in complex urban environments is vital for safe autonomous driving [1] and mobile robotics applications, requiring planning systems to generate collision-free and dynamically feasible trajectories with high computational efficiency. However, the diversity of obstacle distributions and the presence of narrow feasible corridors often leads to non-convex and discontinuous solution spaces. These factors cause conventional one-shot optimization approaches to suffer from poor convergence and strong sensitivity to initialization, thereby limiting their applicability in safety-critical navigation under cluttered environments and real-time constraints [2]–[4].

To address these challenges, multi-stage planning has emerged as an effective strategy that decomposes the complex trajectory optimization problem into modular components: global search, local feasibility enforcement, and trajectory refinement. This paradigm leverages the strengths of sampling-based methods for coarse trajectory generation, corridor-based formulations for local obstacle separation, and nonlinear optimization for trajectory smoothness and feasibility guarantees [5]. Key related components are reviewed below.

Sampling approach: Sampling-based motion planning has a long history as the foundation of global search. The A* algorithm [6] pioneered the use of admissible heuristics to compute minimum-cost trajectories in discrete graphs, while Rapidly-exploring Random Trees (RRT) [7] extended planning capabilities to high-dimensional systems with dynamic constraints. Later variants such as RRT-Connect [8] and Field D* [9] improved search efficiency and enabled real-time replanning in partially known environments. For driving applications, lattice-based planners incorporate vehicle kinematics to generate dynamically feasible coarse trajectories. For example, [10] demonstrated that lattice bundles, libraries of precomputed trajectory segments, can be effectively used in real autonomous vehicles to plan safe and feasible trajectories in complex urban environments.

Learning approach: While search-based motion planners offer interpretable and efficient coarse trajectory generation, they alone may not guarantee trajectory optimality or sufficient safety margins in cluttered environments. In parallel, data-driven approaches have gained attention as end-to-end alternatives. Early imitation learning systems, such as ALVINN [11]

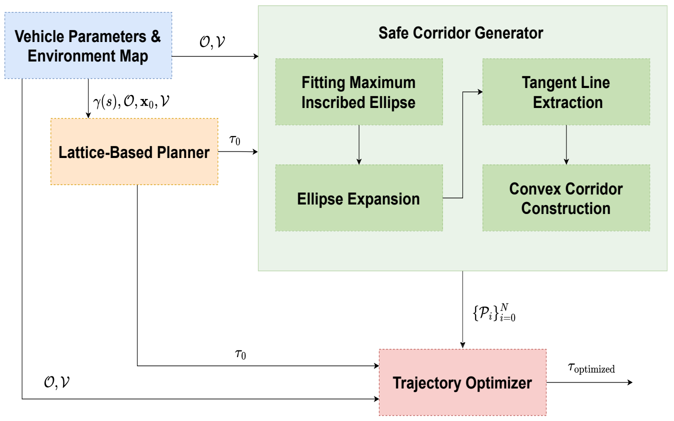
Vehicle Parameters & Environment Map
$\mathcal{O}, \mathcal{V}$
$\gamma(s), \mathcal{O}, \mathbf{x}_0, \mathcal{V}$
Lattice-Based Planner
τ_0
Safe Corridor Generator
Fitting Maximum Inscribed Ellipse
Ellipse Expansion
Tangent Line Extraction
Convex Corridor Construction
$\{\mathcal{P}_i\}_{i=0}^N$
τ_0
$\mathcal{O}, \mathcal{V}$
Trajectory Optimizer
$\tau_{\text{optimized}}$

Fig. 1. TriPlanner pipeline combining coarse trajectory generation, safety corridor construction, and trajectory optimization. Lattice-based planner first generates coarse/initial trajectory, then safe convex corridor at each step is constructed, followed by trajectory optimization with dynamic constraints in a receding horizon manner.

*This work was supported by the Taihu Lake Innovation Fund for the School of Future Technology of Southeast University and Xiaomi Young Talents Program. Corresponding author: Prof. Jinya Su.

[1]Zhaolun Li and Wang Pan are with Product Planning and Automotive New Technology Institute, BYD Auto Co., Ltd., Shenzhen, Guangdong 518118, China. lizhaolun10@gmail.com, Pan.wang3@byd.com

[2]Jinya Su is with School of Automation, Key Laboratory of Measurement and Control of CSE, Ministry of Education, and Institute of Intelligent Unmanned Systems, Southeast University, Nanjing 210096, China. sucas@seu.edu.cn

A video demonstration of this work is available at https://drive.google.com/drive/folders/1QjFvlZgcHpAzR_rjw2OBgk90D16gpkls?usp=drive_link.

and the work of [12], learned to map sensor data directly to control commands by mimicking expert demonstrations. Subsequent research addressed distributional shift and data limitations with methods such as DAgger [13] and ChauffeurNet [14], while recent architectures like TransFuser [15] integrated multimodal perception for improved robustness. Similarly, reinforcement learning has been explored to train driving policies through trial-and-error interaction [16]–[19]. However, both imitation and reinforcement learning face important limitations, including lack of guaranteed dynamic feasibility, difficulty in explicitly enforcing safety constraints, and dependence on either large-scale expert data or extensive simulation. These challenges motivate the adoption of hybrid modular pipelines that retain the interpretability and constraint compliance of model-based planning while providing opportunities to integrate learning-based components.

Modular approach: Compared to monolithic or black-box approaches, multi-stage modular frameworks offer several practical advantages. They promote modularity and flexibility, as each stage can be adapted or replaced to fit different planning scenarios, such as integrating hybrid A* in parking environments or lattice planners for structured roads, thereby reusing specialized methods while maintaining a unified pipeline [20]. Moreover, they enhance interpretability, since explicit representations of search, constraint modeling, and optimization make the overall process more explainable and easier to verify in safety-critical applications. Finally, their incremental refinement capability enables decoupled modules to naturally support receding-horizon replanning and rapidly adapt to dynamic changes in the environment [21].

In this work, therefore, we adopt a systematic and modular planning architecture that balances interpretability, safety, and real-time tractability. The method first employs lattice planning to generate a coarse but feasible trajectory. This is followed by the construction of convex safety corridors that explicitly encode spatio-temporal constraints. Finally, a nonlinear optimization stage refines the trajectory within these corridors while satisfying vehicle dynamics and control limits in a receding-horizon manner.

Corridor construction: Regarding safety corridor construction, prior methods have explored different trade-offs between shape fidelity and computational efficiency. For example, Bubble Planner [22] grows overlapping isotropic spheres along trajectories, but this limits adaptation to vehicle geometry. Faster [5] and Safe Flight Corridors [23] build polyhedral corridors by inflating fixed directions or projecting onto voxel maps, often relying on heuristics and ignoring vehicle orientation. The method in [24] inflates ellipsoids and generates separating hyperplanes to construct convex corridors, but it performs only a single iteration and assumes axis-aligned growth. [25] further adapts this idea for autonomous parking by incorporating vehicle orientation and accelerating closest-point search using the GJK algorithm.

Inspired by these advancements, we construct orientation-aware convex corridors by iteratively expanding maximum inscribed ellipses at each trajectory point. By applying affine normalization and performing tangent extraction, either geometrically or via convex optimization, we obtain tight, geometry-conforming polygons that adapt to both environmental clutter and vehicle dynamics. This refined corridor representation forms the foundation for the downstream nonlinear trajectory optimization, which benefits from precise constraint encoding and improved feasible-region coverage.

In summary, the proposed TriPlanner framework integrates lattice-based sampling, iterative convex corridor generation, and nonlinear refinement in a modular fashion, providing a robust and interpretable solution for autonomous trajectory planning in complex environments. A schematic overview of the complete procedure is presented in Fig. 1, and the main contributions are summarized as follows:

(1) **Tri-stage planning framework**: We develop a flexible and interpretable "TriPlanner" that integrates lattice-based sampling, convex safety corridor, and nonlinear trajectory optimization, enabling robust trajectory generation in dense and cluttered driving environments.
(2) **Iterative convex safety corridor**: We propose an approach that leverages maximum inscribed ellipses, affine normalization, and tangent extraction to build orientation-aware convex corridors that tightly approximate free space while fully accounting for vehicle geometry and dynamics.
(3) **Receding-horizon trajectory optimization**: We formulate a constrained optimization problem that refines trajectories within the constructed corridors, explicitly incorporating vehicle kinematics, control bounds, and spatial constraints to ensure smoothness, feasibility, and reliable obstacle avoidance.

II. Problem and Preliminaries

A. Problem Formulation

To address the trajectory planning challenges in complex environments, we propose a hierarchical framework named TriPlanner, comprised of three distinct stages: coarse trajectory initialization, safety corridor generation, and optimization-based refinement.

Given a reference path $\gamma(s)$ based on the center of the lane and a set of static and dynamic obstacles $\mathcal{O}$, the first stage (Lattice-based Coarse Trajectory Generation) samples polynomial trajectories in the Frenet frame. These candidates are transformed into Cartesian coordinates and filtered using efficient SAT(Separating Axis Theorem)-based collision checking to yield a coarse, collision-free reference trajectory τ^*.

In the second stage (Convex Safe Corridor Construction), based on the generated τ^*, a convex safety corridor $\mathcal{C}$ is constructed at each trajectory point. This corridor is formed by iteratively expanding maximum inscribed ellipses and generating separating hyperplanes, resulting in a sequence of orientation-aware convex polygons that effectively represent the collision-free space.

Finally, the third stage (Trajectory Optimization) models the generation process as a constrained nonlinear optimization problem. Using τ^* as the initial guess (warm start), the objective is to compute the optimal trajectory τ^{Opt} that minimizes a comprehensive cost function J. This optimization is subject to the vehicle's kinematic constraints and strictly confined within the generated convex safety corridor $\mathcal{C}$, ensuring the resulting trajectory is smooth, dynamically feasible, and collision-free.

B. Preliminaries

1) Reference Trajectory and Frenet Frame: Let the initial reference trajectory be defined as a smooth curve $\gamma(s) = [x_r(s), y_r(s)]$ parameterized by arc length s. The Frenet frame describes vehicle motion relative to this trajectory: s denotes longitudinal progression along $\gamma(s)$, and l denotes the lateral offset perpendicular to $\gamma(s)$. The vehicle state in this frame is expressed as $\mathbf{x} = [s, l, \dot{s}, \dot{l}, \ddot{s}, \ddot{l}]$.

2) Trajectory Parameterization and Conversion: Candidate trajectories are constructed by sampling polynomial profiles: a quartic polynomial $s(t)$ for longitudinal motion, satisfying specified initial and terminal conditions, and a quintic polynomial $l(t)$ for lateral motion, satisfying boundary conditions in position, velocity, and acceleration. Each $(s(t), l(t))$ pair is transformed into Cartesian coordinates,

$$\begin{aligned} x(t) &= x_r(s(t)) - l(t)\sin(\theta_r(s(t))), \\ y(t) &= y_r(s(t)) + l(t)\cos(\theta_r(s(t))), \end{aligned} \tag{1}$$

where $\theta_r(s)$ denotes heading angle of the reference trajectory.

3) Vehicle Geometry: The vehicle is modeled as a rectangle centered at the rear axle,

$$\mathbf{p}_{\text{local}} = \begin{bmatrix} L_f & W/2 \\ L_f & -W/2 \\ -L_r & -W/2 \\ -L_r & W/2 \end{bmatrix}, \tag{2}$$

which is transformed at each time step to the global frame as

$$\mathbf{p}_{\text{global}} = R(\theta(t))\mathbf{p}_{\text{local}} + \left[x(t), y(t)\right]^T, \tag{3}$$

with $R(\theta)$ denoting the 2D rotation matrix.

4) Collision Checking via Separating Axis Theorem: To ensure collision-free motion, the Separating Axis Theorem (SAT) [26] is applied between the convex vehicle polygon and obstacle polygons.

Lemma 1 (Separating Axis Theorem): Two convex polygons do not intersect if and only if there exists at least one axis, orthogonal to an edge of either polygon, along which their projections are disjoint. Formally, given convex polygons A and B, they are guaranteed to be non-overlapping if there exists a unit vector $\mathbf{n}$ such that

$$\max_{a\in A}(\mathbf{n}^\top a) < \min_{b\in B}(\mathbf{n}^\top b) \quad \text{or} \quad \max_{b\in B}(\mathbf{n}^\top b) < \min_{a\in A}(\mathbf{n}^\top a).$$

This efficient test certifies the geometric safety of each candidate trajectory at discrete time steps.

III. Lattice-Based Coarse Trajectory Generation

In complex driving scenarios, generating a reliable initial trajectory is crucial for downstream safety-aware optimization. Building upon the preliminaries, the first stage employs a lattice-based planner that produces dynamically feasible and collision-free coarse trajectories. The overall procedure is summarized in Algorithm 1.

Algorithm 1: Coarse Trajectory Generation via Lattice Planner with SAT Collision Checking

```
Input: Reference trajectory $\gamma(s)$, obstacle set $\mathcal{O}$,
       vehicle initial state $\mathbf{x}_0$, vehicle parameters $\mathcal{V}$
Output: Optimal feasible coarse trajectory $\tau^*$
1  Initialize empty candidate set $\mathcal{T} \leftarrow \emptyset$;
2  foreach terminal velocity $v_T \in [0.2v_{target}, 1.2v_{target}]$ do
3      foreach terminal time $T \in [4.5, 5.5]s$ do
4          Generate longitudinal polynomial $s(t)$
             satisfying boundary conditions;
5          foreach lateral offset $l_T \in [-W_{max}, W_{max}]$ do
6              Generate lateral polynomial $l(t)$ satisfying
                 boundary conditions;
7              Convert $(s(t), l(t))$ to global trajectory $\tau(t)$
                 using the reference trajectory;
8              if SAT_Collides$(\tau(t), \mathcal{O}, \mathcal{V})$ = false then
9                  if $\tau(t)$ satisfies dynamic constraints
                     then
10                     Compute cost $J(\tau)$;
11                     Append $(\tau, J(\tau))$ to $\mathcal{T}$;
12 if $\mathcal{T} = \emptyset$ then
13     return failure: no feasible
         trajectory found;
14 else
15     return $\tau^* = \arg\min_{\tau\in\mathcal{T}} J(\tau)$;
```

A. Sampling Strategy

Candidate trajectories are generated by systematically sampling combinations of terminal velocity v_T, terminal time T, and lateral offset l_T. For each sampled combination:

(1) Compute the longitudinal and lateral polynomials;
(2) Convert the trajectory to Cartesian space;
(3) Perform SAT-based collision checking;
(4) Enforce dynamic feasibility constraints,

$$|v(t)| \le v_{\max}, \quad |a(t)| \le a_{\max}, \quad |\kappa(t)| \le \kappa_{\max}. \tag{4}$$

Remark 1: Unlike traditional lattice-planning approaches where collision avoidance is incorporated merely as a penalty term, potentially admitting collision-prone trajectories when weights are poorly tuned, the proposed method treats collision checking as a strict feasibility condition. Furthermore, using SAT for polygon-level collision detection significantly improves computational efficiency compared to edge-intersection methods.

B. Cost and Output

Each feasible trajectory is evaluated using the cost

$$\begin{aligned} J_i = K_j \sum_t (j_l(t) + j_s(t)) + K_T T \\ + K_v\,|v_T - v_{\text{target}}| + K_l\,|l_T|, \end{aligned} \tag{5}$$

where j_s and j_l denote longitudinal and lateral jerk, and the weights K_j, K_T, K_v, K_l balance smoothness, time, terminal velocity, and lane alignment.

The trajectory with the minimum cost is selected as the initial coarse reference τ_0, which serves as the foundation for subsequent convex corridor construction and nonlinear trajectory optimization.

IV. Convex Safety Corridor

Due to its discretized structure and limited sampling resolution, the coarse trajectory τ_0 in Section III may not fully exploit the available free space or provide the smoothness required by the control modules. Therefore, we construct a sequence of convex safety corridors along the horizon of τ_0. At each trajectory point, a local convex polygon is generated by considering the vehicle's bounding box together with the geometry of nearby obstacles. These time-indexed convex regions serve as spatial constraints for the subsequent nonlinear trajectory optimization, effectively capturing environmental boundaries in a compact and tractable form. By embedding τ_0 within this sequence of convex corridors, the final trajectory refinement inherits hard safety guarantees while retaining sufficient flexibility for dynamic feasibility and smoothness. The pseudo-code for convex safety corridor generation is summarized in Algorithm 2.

The corridor construction involves geometric projection, tangent line computation, and constraint pruning, which are elaborated as below. Let $\mathbf{x}_i = [x_i, y_i, \theta_i]$ denote the vehicle's pose at trajectory index i, and let $\mathcal{O}_i$ denote the set of nearby obstacle boundaries. The goal is to generate a convex polygon $\mathcal{P}_i$ satisfying the following conditions:

(1) $\mathcal{P}_i$ contains the vehicle at pose $\mathbf{x}_i$;
(2) $\mathcal{P}_i$ excludes all obstacles in $\mathcal{O}_i$;
(3) $\mathcal{P}_i$ is maximally inflated for trajectory optimization.

A. Elliptical Initialization

We begin by fitting a maximum inscribed ellipse $\mathcal{E}_i$ within the vehicle's bounding rectangle, centered at the vehicle pose $\mathbf{x}_i = (c_x, c_y, \theta_i)$. The ellipse is parameterized by

$$\mathcal{E}_i = \left\{ \begin{bmatrix} x \\ y \end{bmatrix} \;\middle|\; (\mathbf{z}-\mathbf{c})^\top A_i (\mathbf{z}-\mathbf{c}) \le 1 \right\} \tag{6}$$

where $\mathbf{z} = [x \;\; y]^\top$ is any point in space, $\mathbf{c} = [c_x \;\; c_y]^\top$ is the center of the ellipse (vehicle position), and A_i is a symmetric positive definite matrix defined by

$$\begin{aligned} A_i &= R(\theta_i) \begin{bmatrix} \frac{1}{a^2} & 0 \\ 0 & \frac{1}{b^2} \end{bmatrix} R(\theta_i)^\top, \\ R(\theta_i) &= \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}. \end{aligned} \tag{7}$$

with a and b the semi-major and semi-minor axes of the ellipse, aligned with the vehicle's orientation.

Algorithm 2: Convex Safe Corridor Generation

```
  Input: Coarse trajectory $\tau_0 = \{\mathbf{x}_i\}_{i=0}^N$, obstacle set $\mathcal{O}$,
         vehicle parameters $\mathcal{V}$
  Output: Sequence of convex polygons $\{\mathcal{P}_i\}_{i=0}^N$
          forming the safety corridor
1 Initialize vehicle parameters and corridor settings (e.g.,
    max iterations $K$, area threshold $\epsilon$);
2 foreach trajectory point $\mathbf{x}_i = (c_x, c_y, \theta_i) \in \tau_0$ do
3     Obtain nearby obstacles $\mathcal{O}_i$;
4     Construct initial ellipse $\mathcal{E}_i$ inside vehicle footprint
        at pose $\mathbf{x}_i$;
5     repeat
6         Compute affine transform $T_i = A_i^{1/2}$ and apply
            it:
              • Map ellipse $\mathcal{E}_i$ to unit circle $\hat{\mathcal{E}}_i$
              • Transform obstacle points $\mathcal{O}_i$ to
                normalized space $\hat{\mathcal{O}}_i$
          Initialize empty tangent set $\mathcal{L}_i \leftarrow \emptyset$;
          foreach obstacle polygon in $\hat{\mathcal{O}}_i$ do
              Attempt to generate geometric tangent to
                $\hat{\mathcal{E}}_i$;
              if geometric tangent fails validity check
              then
                  Solve QP to compute separating
                    hyperplane $\boldsymbol{\beta}$;
                  Compute tangent point on unit circle:
                    $(x_t, y_t) = \boldsymbol{\beta}/\|\boldsymbol{\beta}\|^2$;
              Transform valid tangent back to global
                frame, append to $\mathcal{L}_i$;
          Filter redundant tangents in $\mathcal{L}_i$ by polar angle
            sweep;
          Construct convex polygon $\mathcal{P}_i$ from supporting
            lines in $\mathcal{L}_i$;
          Solve convex optimization to fit maximal
            inscribed ellipse $\mathcal{E}_i$ within $\mathcal{P}_i$;
7     until relative area increase below threshold $\epsilon$ or
        max iterations reached;
8     Store $\mathcal{P}_i$ as local convex corridor at time $i$;
9 return $\{\mathcal{P}_i\}_{i=0}^N$
```

B. Affine Transformation

To enable efficient geometric filtering, we map the ellipse $\mathcal{E}_i$ to a unit circle centered at the origin via an affine transformation. Specifically, we define

$$\hat{\mathbf{z}} = T_i(\mathbf{z}-\mathbf{c}), \quad \text{with} \quad T_i = A_i^{1/2}, \tag{8}$$

where T_i is the principal square root of A_i, satisfying $T_i^\top T_i = A_i$. Under this transformation, the ellipse becomes

$$\hat{\mathcal{E}}_i = \left\{ \hat{\mathbf{z}} \in \mathbb{R}^2 \;\middle|\; \|\hat{\mathbf{z}}\|_2^2 \le 1 \right\}, \tag{9}$$

that means the unit circle centered at the origin.

This same transformation is applied to the surrounding obstacle boundary points $\mathcal{O}_i$ to obtain a normalized obstacle set $\hat{\mathcal{O}}_i$. This normalization allows all geometric computations (e.g., tangent generation, convex hull filtering) to be performed in a standard unit space.

C. Geometric Tangent Construction

Following affine normalization, the surrounding obstacles are represented as convex polygons in the transformed space. We initially employ a geometric approach to construct supporting tangents between the unit circle and each obstacle. For each edge $\overline{A_i A_{i+1}}$ of the obstacle polygon in the normalized frame, we compute the orthogonal projection of the unit circle center (i.e., the origin) onto the line segment. Among all candidate projections, the one with the smallest norm is selected as the tentative contact point (x_t, y_t). Using this point, we define a tentative tangent line of the form,

$$ax + by + c = 0 \tag{10}$$

where the coefficients are defined as

$$a = y_t, \quad b = -x_t, \quad c = -x_t^2 - y_t^2 \tag{11}$$

This line is tangent to the unit circle at (x_t, y_t), and points outward from the circle. We then use this line as a separating hyperplane between the transformed vehicle points $\mathcal{V}_{\text{veh}}$ and obstacle points $\mathcal{O}_i$.

1) Geometric Tangent Validity: The tangent is considered geometrically valid if:

(1) It touches the unit circle externally;
(2) It intersects exactly one point or edge of the convex polygon obstacle;
(3) The tangent vector points outward from the origin.

However, in practice, not all such geometrically constructed tangents are suitable for building convex safety corridors due to the reason in Remark 2. And this problem arises in the iterative corridor construction process: as the inscribed ellipse rotates or scales during optimization, earlier geometric tangents may no longer separate the full set of vehicle vertices from the obstacles in the original planning space.

Remark 2: Even though a line may be tangent to both the unit circle and the obstacle in the normalized space, it may **fail to separate the full vehicle body (i.e., its bounding rectangle)** from the obstacle after inverse affine transformation.

2) Validity Check: To ensure robustness, each geometrically constructed tangent is subjected to a strict separation check. If any point violates the constraint, the geometric tangent is deemed invalid and discarded.

- All transformed vehicle boundary points $\mathbf{v}_j \in \mathcal{V}_{\text{veh}}$ must satisfy $ax_j + by_j + c < 0$;
- All transformed obstacle points $\mathbf{o}_k \in \mathcal{O}_i$ must satisfy $ax_k + by_k + c \geq 0$.

D. Real Time Tangent Generation

We may formulate the tangent search as a nonlinear optimization problem by maximizing the distance from the origin to the tangent point,

$$\max_{(x_t, y_t)} \quad x_t^2 + y_t^2 \tag{12}$$

subject to the same linear separation constraints as above. However, since the tangent parameters (a, b, c) depend nonlinearly on (x_t, y_t) via

$$a = y_t, \quad b = -x_t, \quad c = -x_t^2 - y_t^2, \tag{13}$$

this formulation leads to a non-convex optimization problem.

Although solvers such as IPOPT are capable of handling the nonlinear formulation, the computational time makes it impractical for real-time trajectory planning. To address this, we reformulate the problem in terms of the normal vector of the separating hyperplane. Let the tangent direction be $\boldsymbol{\alpha} = [x_t, y_t]^\top$, and its polar vector be denoted by $\boldsymbol{\beta}$. To facilitate reformulation, we re-parameterize the separating hyperplane using its normal vector $\boldsymbol{\beta} \in \mathbb{R}^2$. The corresponding unit direction vector $\boldsymbol{\alpha}$ is then obtained by normalizing $\boldsymbol{\beta}$:

$$\boldsymbol{\alpha} = \boldsymbol{\beta} / \|\boldsymbol{\beta}\|^2. \tag{14}$$

where $\boldsymbol{\alpha}$ points from the origin (center of the unit circle) toward the point of tangency, while $\boldsymbol{\beta}$ can be interpreted as the scaled normal vector defining the supporting hyperplane.

Substituting this into the original separation constraints yields a Quadratic Programming (QP) formulation in terms of $\boldsymbol{\beta} \in \mathbb{R}^2$. The separating function is defined as

$$f(\mathbf{z}) = \boldsymbol{\beta}^\top \mathbf{z} - 1, \tag{15}$$

$$\begin{cases} \boldsymbol{\beta}^\top \mathbf{v}_j & \leq 1, \quad \forall \mathbf{v}_j \in \mathcal{V}_{\text{veh}}, \\ \boldsymbol{\beta}^\top \mathbf{o}_k & \geq 1, \quad \forall \mathbf{o}_k \in \mathcal{O}_i. \end{cases} \tag{16}$$

To maximize the margin between the two sets, we minimize the squared norm of the normal vector, resulting in the following convex QP,

$$\begin{aligned} \min_{\boldsymbol{\beta}} \quad & \|\boldsymbol{\beta}\|^2 \\ \text{s.t.} \quad & \boldsymbol{\beta}^\top \mathbf{v}_j \leq 1, \quad \forall j \\ & \boldsymbol{\beta}^\top \mathbf{o}_k \geq 1, \quad \forall k \end{aligned} \tag{17}$$

This yields a small-dimensional minimum-norm problem: the solution $\boldsymbol{\beta}$ defines the separating hyperplane, the corresponding tangent point on the hyperplane is recovered as

$$(x_t, y_t) = \boldsymbol{\beta} / \|\boldsymbol{\beta}\|^2. \tag{18}$$

E. Solution via SDMN

Instead of solving (17) using generic quadratic programming solvers, which may be computationally costly for real-time planning, we adopt the Small-Dimensional Minimum-Norm (SDMN) method [27]. The key observation is that problem (17) is exactly in the form of:

$$\min_{z \in \mathbb{R}^2} |z|^2, \quad \text{s.t. } a_i^\top z \leq b_i, \ \forall i, \tag{19}$$

with $z \equiv \beta$.

Rather than adopting the recursive procedure of the original SDMN algorithm, we directly transform the 2D minimum-norm problem (19) into a one-dimensional constrained problem shown in Algorithm 3. When a constraint $a^\top z \leq b$ is violated, the feasible region must lie on its boundary,

$$a^\top z = b, \tag{20}$$

and the current solution is projected onto this line by

$$v = \frac{b}{|a|^2} a. \tag{21}$$

To reduce the problem to one dimension, we construct a household reflection that maps the normal vector a to a coordinate axis. Let

$$u = v + \operatorname{sgn}(v_j)|v|e_j, \quad H = I - \frac{2uu^\top}{u^\top u}, \tag{22}$$

where $j = \arg\max_k |v_k|$ ensures numerical stability and e_j denotes the j-th standard basis vector in $\mathbb{R}^n$. The orthogonal basis M is taken from the column of $H^\top$ orthogonal to a, which yields the reduced representation

$$z = My' + v. \tag{23}$$

Substituting this form into active constraints $\mathcal{I}$ produces a family of one-dimensional inequalities,

$$a'_k y' \le b'_k, \quad \forall k \in \mathcal{I}, \tag{24}$$

where $a'_k = M^\top a_k$ and $b'_k = b_k - a_k^\top v$. The 2D minimum norm problem (19) is thus reduced to

$$\min_{y' \in \mathbb{R}} (y')^2, \quad \text{s.t. } a'_k y' \le b'_k, \ \forall k. \tag{25}$$

The problem (25) admits a closed-form solution. Each one-dimensional inequality $a'_k y' \le b'_k$ defines an upper or a lower bound on y':

$$y' \le \frac{b'_k}{a'_k}, \quad \text{if } a'_k > 0, \qquad y' \ge \frac{b'_k}{a'_k}, \quad \text{if } a'_k < 0. \tag{26}$$

By scanning all constraints, we obtain the feasible interval.

$$y_{\min} = \max_{a'_k < 0} \frac{b'_k}{a'_k}, \qquad y_{\max} = \min_{a'_k > 0} \frac{b'_k}{a'_k}. \tag{27}$$

If $y_{\min} > y_{\max}$, the problem is infeasible. Otherwise, the optimal solution is given by

$$y'^\star = \begin{cases} 0, & y_{\min} \le 0 \le y_{\max}, \\ \arg \min\limits_{y \in \{y_{\min}, y_{\max}\}} |y|, & \text{otherwise.} \end{cases} \tag{28}$$

Finally, the 2D solution is reconstructed as

$$z^\star = My'^\star + v. \tag{29}$$

This direct dimension reduction approach avoids the recursive structure of the original SDMN algorithm. After projecting the 2D feasible set onto the boundary of the active constraint, the problem reduces to a one-dimensional constrained search. The resulting sub-problem is equivalent to finding the closest feasible point to the origin on a line segment, which admits a closed-form solution via interval checking. In this way, the method achieves linear complexity in the number of constraints and provides both efficiency and numerical robustness for real-time tangent generation.

In practice, the proposed SDMN-based tangent generation exhibits significant computational advantages over general-purpose quadratic programming solvers. On average, computing a tangent using SDMN requires only **0.015** ms, whereas solving the same problem with the nonlinear solver IPOPT takes approximately **30** ms. This improvement of more than two orders of magnitude in runtime makes the SDMN approach highly suitable for real-time trajectory planning applications.

Algorithm 3: Modified SDMN Tangent Generation (2D→1D Reduction)

```
Input: Constraint set $\mathcal{H} = \{(a_i, b_i)\}_{i=1}^m$
Output: Minimum-norm solution $z^\star$
1  Initialize $z \leftarrow \mathbf{0}$, $\mathcal{I} \leftarrow \emptyset$;
2  Randomly permute constraints in $\mathcal{H}$;
3  foreach $(a, b) \in \mathcal{H}$ do
4      if $a^\top z > b$ then
5          Project onto boundary $a^\top z = b$ and compute offset $v$;
6          Apply Householder reflection to obtain reduced basis $M$;
7          Form 1D constraints from $\mathcal{I}$;
8          Solve the 1D minimum-norm problem by interval checking to get $y'^\star$;
9          Reconstruct $z \leftarrow My'^\star + v$;
10     Update $\mathcal{I}$ with $(a, b)$;
11 return $z^\star = z$;
```

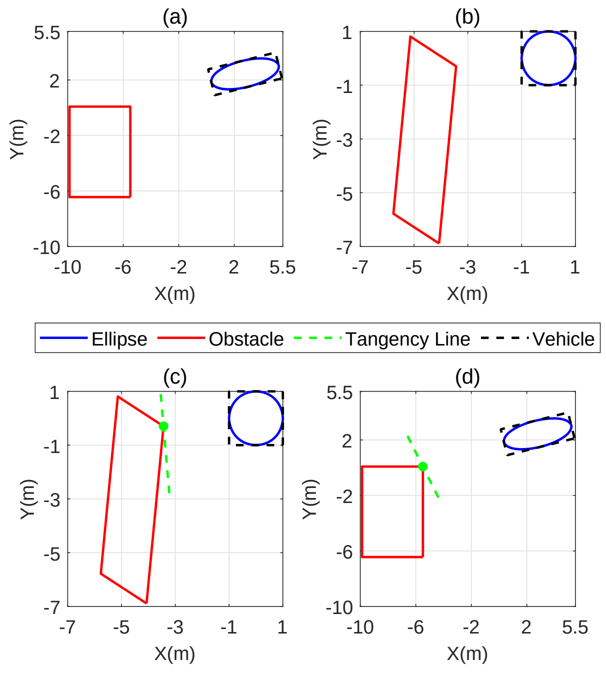

Fig. 2. Illustration of the first iteration of convex safety corridor generation: (a) Ellipse fitting in vehicle coordinates, (b) Affine normalization to unit circle, (c) Tangent line computation in normalized space, (d) Inverse transformation back to global coordinates.

Fig. 2 illustrates the first iteration of the convex safety corridor generation process, showing the sequence from ellipse initialization to tangent line computation. In particular, Fig. 2-(a) shows the initialization step, where a maximum inscribed ellipse is fitted within the vehicle's bounding rectangle. In Fig. 2-(b), the ellipse is transformed into a unit circle centered at the origin through affine normalization, with both the vehicle rectangle and surrounding obstacles mapped into the same normalized coordinate system. Fig. 2-(c) depicts the process of identifying the tangent: the closest point on the transformed obstacle to the origin is located, and a tangent line at this contact point is computed in the normalized space. Finally, Fig. 2-(d) illustrates how the tangent point and its corresponding tangent line are transformed back into the original coordinate frame, preparing them to define the convex safe corridor for the current iteration.

F. Corridor Construction

Once a valid tangent has been identified in the normalized space, it is transformed back to the global coordinate frame using the inverse of the affine mapping applied during normalization. Specifically, the tangent point on the unit circle is mapped as

$$\begin{bmatrix} x \\ y \end{bmatrix}_{\text{global}} = R(\theta_i) \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{30}$$

This yields the global coordinates of the supporting point and corresponding hyperplane, which are then used to define the half-space constraints forming the convex region around trajectory point i.

To ensure that the convex corridor remains compact and computationally efficient, all generated tangents are sorted by their polar angles and filtered using a sweep-line procedure. This step removes redundant constraints and retains only the minimal set required to form a closed convex polygon. The resulting set of supporting hyperplanes defines a convex region $\mathcal{P}_i$ at time step i.

Next, given the polygon $\mathcal{P}_i$ defined by a set of linear inequalities $A_i x \leq b_i$, we compute the largest ellipse that can be inscribed within it. This is formulated as the following convex optimization problem,

$$\begin{aligned} \max_{L,\mathbf{c}} \quad & \log\det(LL^\top) \\ \text{s.t.} \quad & \|A_i L\| + A_i \mathbf{c} \leq b_i \end{aligned} \tag{31}$$

where L is the Cholesky factor of the ellipse shape matrix, and $\mathbf{c}$ is the center of the ellipse.

This convex optimization problem of finding the maximum volume inscribed ellipsoid (MVIE) is reformulated as a series of subproblems to reduce the calculation time for polygons with N edges in Algorithm 4, where $N \in \{3, 4, 5\}$ corresponds to triangles, quadrilaterals, and pentagons. The solution for polygons with $N \geq 6$ can then be derived by considering the results from these simpler subproblems, ensuring an efficient way to compute the maximal inscribed ellipse for more complex polygons [28]. The polygon is represented by normalized half-space constraints $a_i^\top x + c_i \leq b_i$, where $a_i \in \mathbb{R}^2$ is the outward unit normal of the i-th supporting line and $b_i, c_i \in \mathbb{R}$ are scalar offsets. During enumeration, every subset S with $|S| \in \{3, 4, 5\}$ corresponds to a candidate polygon (triangle, quadrilateral, or pentagon). If the selected lines form a closed convex region, an analytic MVIE solver is invoked: (i) `SolveTriangle` returns the Steiner inellipse.

Algorithm 4: Fast Maximum-Inscribed Ellipse via MVIE Subset Enumeration

```
Input: Convex polygon represented by half-spaces
       $\mathcal{H} = \{(a_i, b_i, c_i)\}_{i=1}^N$, where $a_i^\top x + c_i \leq b_i$
Output: Largest-area inscribed ellipse $E^\star = (c^\star, X^\star)$
1  Normalize all constraints:;
2  foreach $(a_i, b_i, c_i) \in \mathcal{H}$ do
3      $(a_i, b_i, c_i) \leftarrow (a_i, b_i, c_i)/\|a_i\|$;
4  $A_{\text{best}} \leftarrow 0$, $E^\star \leftarrow \varnothing$;
5  Generate index set $\mathcal{S}$ of all subsets of $\{1, \ldots, N\}$ such
     that $|\mathcal{S}| \in \{3, 4, 5\}$;
6  foreach subset $S \subset \mathcal{S}$ do
7      if $S$ does not form a closed convex polygon then
8          continue;
9      if $|S| = 3$ then
10         $E \leftarrow$ SolveTriangle($S$);
11     else
12         if $|S| = 4$ then
13             $E \leftarrow$ SolveQuadrilateral($S$);
14         else
15             $E \leftarrow$ SolvePentagon($S$);
16     if $E$ is invalid then
17         continue;
18     if $E$ violates any constraint in $\mathcal{H}$ then
19         continue;
20     Compute ellipse area $A \leftarrow \pi/\sqrt{\det(X_E)}$;
21     if $A > A_{best}$ then
22         $A_{\text{best}} \leftarrow A$;
23         $E^\star \leftarrow E$;
24 return $E^\star$;
```

Given the three tangency lines of the triangle, the intersection points of these lines are calculated as the triangle's vertices. Let the 2-D Cartesian coordinates of the vertices of the triangle be denoted as ($v^* = (v_0, v_1, v_2)^\top$). The center v^o and the two conjugate diameters ($f_1, f_2 \in \mathbb{R}^2$) of the Steiner inellipse can be written as:

$$\begin{aligned} v^o &= \frac{1}{3}(v^i + v^{ii} + v^{iii}), \\ f_1 &= \frac{1}{2}(v^o - v^{iii}), \\ f_2 &= \frac{1}{2\sqrt{3}}(v^i - v^{ii}) \end{aligned} \tag{32}$$

These equations represent the Steiner inellipse that is tangent to the sides of the triangle at their midpoints. (ii) `SolveQuadrilateral` returns the Maximal-Area Inscribed Ellipse of a convex quadrilateral using an affine-invariant formulation [29]. Given the four tangency lines, their intersections form the quadrilateral vertices, denoted as $V = \{v^i, v^{ii}, v^{iii}, v^{iv}\}$.

By normalizing the first three vertices to a canonical frame, the fourth vertex is mapped to (s, t). The normalized center $c^\star = (h^\star, k^\star)$ is derived from the roots of a characteristic quadratic equation $Ah^2 + Bh + C = 0$, and the final ellipse parameters in the original frame are recovered as:

$$\begin{aligned} c_{\text{orig}} &= \mathcal{A}^{-1}(c^\star - t_{\text{trans}}), \\ M_{\text{orig}} &= \mathcal{A}^{-1}M(\mathcal{A}^{-1})^\top, \\ \mathcal{E} &= \{x \in \mathbb{R}^2 \mid (x - c_{\text{orig}})^\top M_{\text{orig}}^{-1}(x - c_{\text{orig}}) \le 1\} \end{aligned} \tag{33}$$

where $\mathcal{A}$ represents the affine transformation matrix, M is the shape matrix obtained in the normalized frame, and equation $\mathcal{E}$ defines the unique ellipse of maximal area contained within the quadrilateral. (iii) `SolvePentagon` returns the unique inscribed ellipse tangent to five edges using a dual-space formulation. Given the five tangency lines represented as homogeneous vectors $l_i \in \mathbb{R}^3$ for $i = 1, \dots, 5$, the coefficients of the dual conic matrix M_L are determined by the linear constraints imposed by the tangency condition $l_i^\top M_L l_i = 0$. The primal conic matrix M_P and the resulting geometric region $\mathcal{E}$ are derived as:

$$\begin{aligned} \text{vec}(M_L) &\in \ker(\mathbf{A}_{5\times 6}), \\ M_P &= M_L^{-1}, \\ \mathcal{E} &= \{\tilde{x} \in \mathbb{P}^2 \mid \tilde{x}^\top M_P \tilde{x} \le 0\} \end{aligned} \tag{34}$$

These equations represent a direct linear solution in the dual space, bypassing iterative optimization by exploiting the exact degrees of freedom of the pentagon. Among all analytic candidates, the ellipse with the largest area that remains inside all original constraints is returned as the maximal inscribed ellipse of the polygon. Compared with conventional convex optimization methods for computing the maximum inscribed ellipse, the proposed analytic MVIE solver achieves significantly higher efficiency. By exploiting the closed-form structure of 3–5 supporting edges and evaluating only feasible tangent subsets, each ellipse is obtained within 0.3ms on a standard CPU, enabling real-time construction of convex safety corridors. This substantial speed advantage makes the approach well suited for on-vehicle planning pipelines.

This procedure is applied iteratively. At each iteration, the updated ellipse defines the starting point for defining updated supporting hyperplanes. The process continues until the relative improvement in area between successive ellipses falls below a predefined threshold,

$$\frac{a_{\text{curr}}b_{\text{curr}} - a_{\text{prev}}b_{\text{prev}}}{a_{\text{prev}}b_{\text{prev}}} < \epsilon \tag{35}$$

where a_{curr} and b_{curr} denote the semi-major and semi-minor axes of the ellipse at the current iteration, and $a_{\text{prev}}, b_{\text{prev}}$ are those from the previous iteration. The product ab is proportional to the area of the ellipse, and ϵ is a user-defined convergence threshold (typically on the order of 10^{-3}) that determines when to stop the iteration.

After convergence, the final convex region $\mathcal{P}_i$ is retained. By repeating this process at every trajectory point along τ_0, we obtain a sequence of convex regions $\{\mathcal{P}_i\}_{i=0}^N$, which together define the spatio-temporal safety corridor. This corridor provides a compact constraint that accurately captures local geometry for the subsequent nonlinear trajectory optimization.

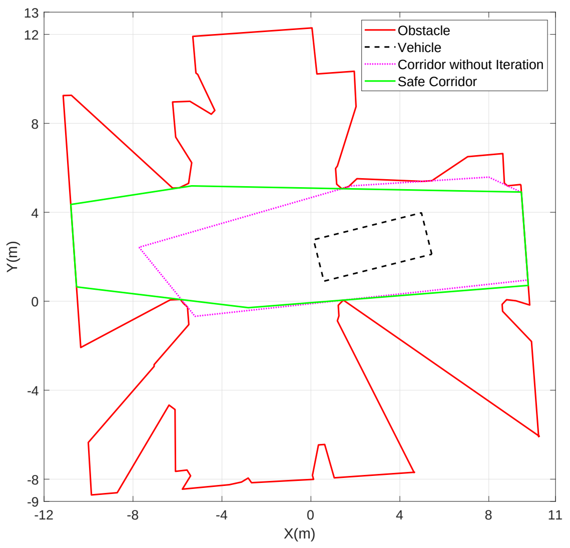

Fig. 3. Iterative expansion on convex corridor in a complex obstacle scene.

Fig. 3 illustrates the impact of iterative expansion on the convex safety corridors generated within a complex obstacle environment. The red lines denote the boundaries of free space, representing the drivable area unobstructed by obstacles. In the current time step, the vehicle outer boundary is indicated by a black dashed rectangle. The magenta dashed polygon corresponds to the convex safety corridor generated without any iterative enlargement, whereas the solid green polygon depicts the safety corridor obtained after applying up to ten iterations of maximal inscribed ellipse expansion. It is evident that the iteratively refined corridor more fully occupies the available free space, providing a larger, less conservative region for subsequent trajectory optimization.

V. Trajectory Optimization within Convex Corridor Constraints

Given the initial coarse collision-free trajectory $\tau_0 = \{\mathbf{x}_0, \dots, \mathbf{x}_N\}$ produced by the lattice-based planner in Section III, and the corresponding sequence of convex safety corridors $\{\mathcal{P}_i\}_{i=0}^N$, we formulate a nonlinear trajectory optimization problem to refine this trajectory for improved smoothness, feasibility, and safety.

Each vehicle state $\mathbf{x}_k = [x_k, y_k, \theta_k, v_k, \delta_k]^\top$ encodes position, heading angle, velocity, and front wheel steering angle, while the control input $\mathbf{u}_k = [a_k, \omega_k]^\top$ includes longitudinal acceleration and steering rate. The vehicle dynamics are modeled using a discretized kinematic bicycle model,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \Delta t \cdot f(\mathbf{x}_k, \mathbf{u}_k), \tag{36}$$

$$f(\mathbf{x}, \mathbf{u}) = \left[v\cos\theta, v\sin\theta, \tfrac{v}{L}\tan\delta, a, \omega\right]^T, \tag{37}$$

and L is the wheelbase of the vehicle.

The objective function, composed of three main components, is constructed to promote smoothness, dynamic feasibility, and accurate trajectory tracking. The first term penalizes control effort by minimizing the squared longitudinal acceleration a_k^2 and steering rate effort, modeled as $\omega_k^2 v_k^2$, weighted by w_1 and w_2, respectively. The second term encourages velocity tracking by minimizing the deviation from a desired target speed v_{target}, scaled by a weight w_3. Together, these components form the running cost,

$$J = \sum_{k=0}^{N-1} \left(w_1 a_k^2 + w_2 \omega_k^2 v_k^2\right) + \sum_{k=0}^{N} w_3 (v_k - v_{\text{target}})^2 + J_{\text{terminal}}.$$

The terminal cost J_{terminal} ensures that the final state of the optimized trajectory closely matches the reference goal pose obtained from the lattice planner. It penalizes the squared distance and heading angle error between the terminal state (x_N, y_N, θ_N) and the last reference pose (x_f, y_f, θ_f):

$$J_{\text{terminal}} = w_t \left[(x_N - x_f)^2 + (y_N - y_f)^2 + (\theta_N - \theta_f)^2\right],$$

where w_t is a user-defined weight for the terminal alignment.

To ensure safety, we enforce that the vehicle's bounding rectangle at each time step lies entirely within the convex polygon $\mathcal{P}_k$. Specifically, each of the rectangle's four corner points must satisfy all half-space constraints defining $\mathcal{P}_k$:

$$a_i x_j + b_i y_j + c_i \geq 0, \quad \forall j \in \{1,2,3,4\}, \quad \forall i. \tag{38}$$

Additionally, the optimization respects the physical limitations of the vehicle,

$$\delta_k \in [-\delta_{\max}, \delta_{\max}], , a_k \in [-a_{\max}, a_{\max}], , v_k \in [0, v_{\max}].$$

The trajectory is required to begin at specified poses $\mathbf{x}_0 = \mathbf{x}_{\text{start}}$, which is inherited directly from the initial point of the lattice trajectory τ_0.

This optimization problem is implemented in CasADi [30] with IPOPT [31] as solver. The lattice trajectory τ_0 provides a natural and feasible initial guess to warm-start the solver, promoting convergence to a smooth and dynamically feasible trajectory that strictly respects spatial safety constraints.

VI. Comparative Study

This section presents comparative simulation results against the SOTA methods in this topic [24], [32], demonstrating the complete TriPlanner pipeline including (1) lattice-based initial collision-free trajectory generation, (2) convex safety corridors around the initial trajectory, and (3) nonlinear trajectory optimization within the corridors to improve smoothness and feasibility.

A. Coarse Trajectory Generation

The coarse trajectory generation results from the lattice planner are shown in Fig. 4, illustrating how each stage progressively refines the trajectory while maintaining safety in a cluttered driving environment. In Fig. 4, obstacles and parked vehicles surrounding the driving corridor are depicted as red polygons. The blue dashed line represents the road centerline, which serves as the reference trajectory for the lattice planner, while the green solid line shows the collision-free trajectory computed by the planner. The black dashed rectangles illustrate the vehicle's outline along this trajectory, demonstrating that the lattice-based planner successfully avoids all obstacles. This trajectory thus provides a *feasible and safe initial guess* for subsequent corridor construction and optimization.

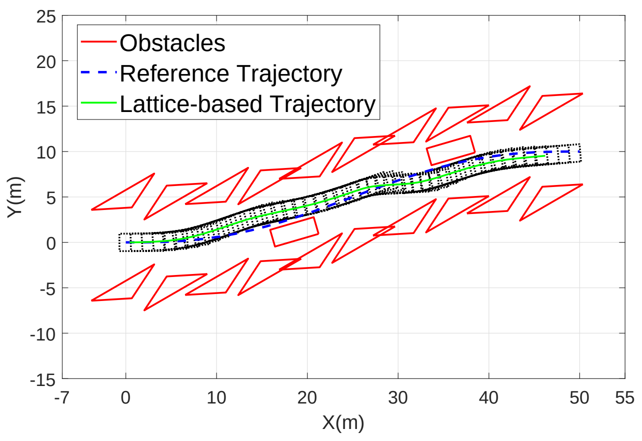

Fig. 4. Collision-free trajectory generated by Lattice planner.

B. Safe Corridor Generation

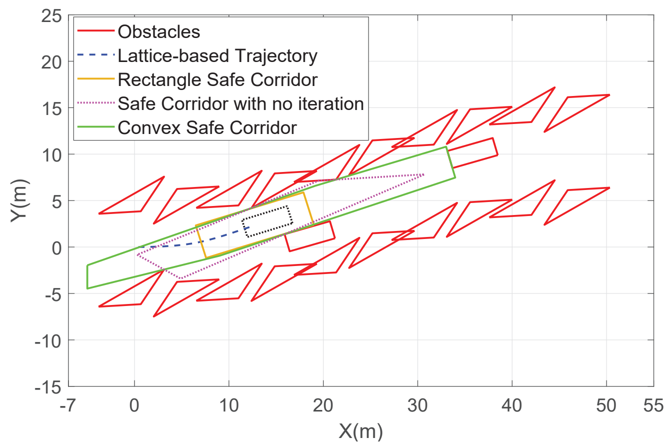

Fig. 5. Safety corridor comparison near trajectory start point.

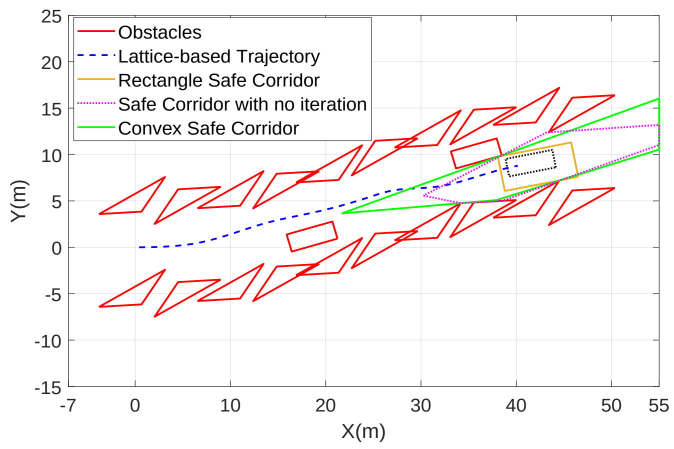

Fig. 6. Safety corridor comparison near trajectory endpoint.

Figs. 5 and 6 compare different methods for generating safety corridors along the same reference trajectory. In both figures, the red polygons represent obstacles, and the blue dashed line shows the lattice-based reference trajectory. The yellow rectangles correspond to the safety corridors generated

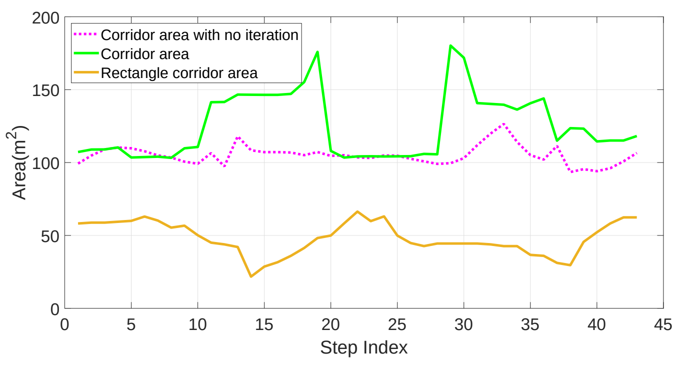

Fig. 7. Safety corridor area comparison between different methods.

using the method in [32], which uniformly expands an axis-aligned rectangle around the vehicle until it collides with obstacles. Due to the rectangular shape restriction, this approach results in corridors that cover substantially less free space. The magenta dashed polygons depict corridors constructed using the approach in [24], which fits an inscribed ellipse and extracts separating tangents to build a convex polygon. However, this method performs only a single iteration of ellipse fitting and tangent extraction, thereby limiting the final corridor size.

In contrast, the solid green polygons show the safety corridors generated by the proposed method. By performing multiple iterations—alternating between fitting the maximum inscribed ellipse and reconstructing convex hulls of separating tangents—our approach incrementally enlarges the corridor while tightly conforming to the free-space geometry. As shown in Fig. 7, this iterative refinement consistently produces corridors that are larger than those generated by [24] in most cases and significantly larger than the rectangle-based corridors of [32]. Quantitatively, the rectangle expansion method in [32] yields an average safe corridor area of $48.26\ \mathrm{m}^2$, whereas the method in [24] achieves $105.14\ \mathrm{m}^2$. In comparison, our proposed method attains an average area of $125.13\ \mathrm{m}^2$, representing a 159% increase over the rectangle-based approach and a 19% improvement over the single-iteration convex polygon approach.

These results demonstrate that the proposed iterative corridor construction method more effectively utilizes the available free space, leading to larger and less conservative safety corridors for downstream trajectory optimization.

C. Nonlinear Trajectory Optimization

Fig. 8 shows the final optimization results. The red polygons again indicate obstacles. The blue dashed line depicts the lattice-based reference trajectory, while the black solid line presents the refined trajectory obtained by solving the nonlinear optimization problem within the constructed safety corridors. The black dashed rectangles illustrate the vehicle's successive positions along the optimized trajectory. Compared to the initial reference trajectory, the optimized trajectory maintains safe separation from obstacles while exhibiting improved smoothness and dynamic feasibility.

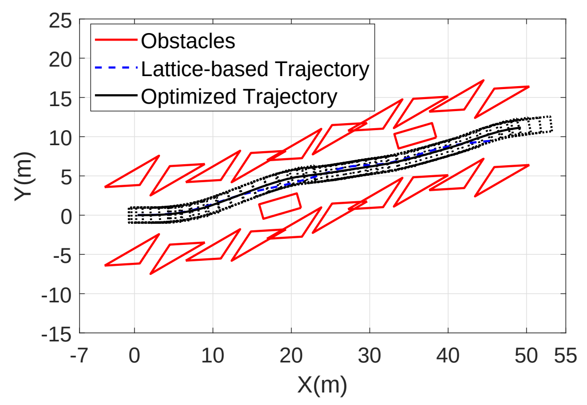

Fig. 8. Optimized trajectory vs Lattice-based reference trajectory

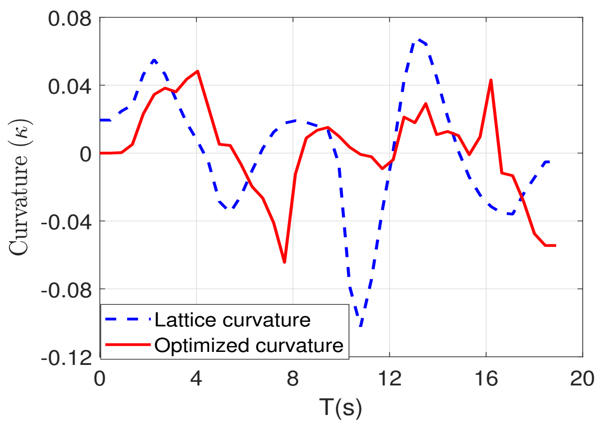

Fig. 9. Curvature comparison between optimized and Lattice-based reference

Finally, Fig. 9 compares the curvature profiles of the lattice planner trajectory and the optimized trajectory. The blue dashed curve indicates the curvature of the reference trajectory, which displays larger-amplitude oscillations. In contrast, the red curve represents the optimized curvature, which is consistently smoother with smaller variations. Quantitatively, the mean absolute curvature is computed as

$$\bar{\kappa} = \frac{1}{N} \sum_{i=1}^{N} |\kappa_i| \,, \tag{39}$$

where κ_i denotes the curvature at each waypoint and N is the total number of points. According to this metric, the optimized trajectory reduces the mean absolute curvature from 0.0286 to 0.0202, achieving an improvement of 29.37%. This reduction demonstrates that the proposed approach produces smoother and more comfortable trajectories.

VII. Conclusions

This paper presented a tri-stage framework for autonomous vehicle trajectory planning in cluttered environments. The method integrates a lattice-based coarse planner, an iterative convex safety corridor generator, and a nonlinear optimization stage into a coherent pipeline that balances safety, feasibility, and efficiency. The lattice planner provides a collision-free reference trajectory through SAT-based filtering, while the corridor construction refines local free space via unit-circle projections and accelerates tangent generation using an analytical Small-Dimensional Minimum-Norm (SDMN) solver. This enables fast and robust construction of tight convex corridors well adapted to vehicle geometry and complex obstacle layouts. The final optimization stage exploits these corridors to produce smooth, dynamically feasible, and strictly collision-free trajectories. Comparative simulations confirm that the proposed approach consistently outperforms conventional planning methods, yielding trajectories with reduced curvature oscillations, improved spatial clearance, and robust satisfaction of safety constraints in dense obstacle scenarios. Future work will extend this framework to dynamic obstacle environments, incorporate probabilistic safety margins, and further enhance computational efficiency to enable real-time deployment on embedded platforms.

References

[1] D. Yi, H. Fang, Y. Hua, J. Su, M. Quddus, and J. Han, "Improving synthetic to realistic semantic segmentation with parallel generative ensembles for autonomous urban driving," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 4, pp. 1496–1506, 2021.

[2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *ann arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[3] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—a local, continuous method," in *2014 IEEE intelligent vehicles symposium proceedings*, pp. 450–457, IEEE, 2014.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[5] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 1934–1940, IEEE, 2019.

[6] 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.

[7] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[8] 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)*, vol. 2, pp. 995–1001, IEEE, 2000.

[9] D. Ferguson and A. Stentz, "Using interpolation to improve path planning: The field d* algorithm," *Journal of Field Robotics*, vol. 23, no. 2, pp. 79–101, 2006.

[10] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[11] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[13] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, JMLR Workshop and Conference Proceedings, 2011.

[14] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[15] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE transactions on pattern analysis and machine intelligence*, vol. 45, no. 11, pp. 12878–12895, 2022.

[16] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[17] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 international conference on robotics and automation (ICRA)*, pp. 8248–8254, IEEE, 2019.

[18] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *arXiv preprint arXiv:1704.02532*, 2017.

[19] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-end model-free reinforcement learning for urban driving using implicit affordances," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7153–7162, 2020.

[20] S. L. Herbert, M. Chen, S. Han, S. Bansal, J. F. Fisac, and C. J. Tomlin, "Fastrack: A modular framework for fast and guaranteed safe motion planning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 1517–1522, IEEE, 2017.

[21] S. Veer, K. Leung, R. Cosner, Y. Chen, P. Karkus, and M. Pavone, "Receding horizon planning with rule hierarchies for autonomous vehicles," *arXiv preprint arXiv:2212.03323*, 2022.

[22] Y. Ren, F. Zhu, W. Liu, Z. Wang, Y. Lin, F. Gao, and F. Zhang, "Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6332–6339, IEEE, 2022.

[23] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 1476–1483, IEEE, 2016.

[24] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "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.

[25] H. Yu and R. Li, "Segmented trajectory optimization for autonomous parking in unstructured environments," *arXiv preprint arXiv:2504.05041*, 2025.

[26] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 171–180, 1996.

[27] Q. Wang, Z. Wang, M. Wang, J. Ji, Z. Han, T. Wu, R. Jin, Y. Gao, C. Xu, and F. Gao, "Fast iterative region inflation for computing large 2-d/3-d convex regions of obstacle-free space," *IEEE Transactions on Robotics*, vol. 41, pp. 3223–3243, 2025.

[28] M. Agarwal, J. Clifford, and M. Lachance, "Duality and inscribed ellipses," *Computational Methods and Function Theory*, vol. 15, no. 4, pp. 635–644, 2015.

[29] A. Horwitz, "Ellipses of maximal area and of minimal eccentricity inscribed in a convex quadrilateral," *Australian Journal of Mathematical Analysis and Applications*, vol. 2, no. 1, p. 12, 2005.

[30] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[31] 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.

[32] X. Jin, Y. Tao, and N. V. Opinat Ikiela, "Trajectory planning design for parallel parking of autonomous ground vehicles with improved safe travel corridor," *Symmetry*, vol. 16, no. 9, p. 1129, 2024.