

From Semantic Decisions to Feasible Trajectories: Self-Evolving LLM-Guided Optimal Control for Narrow-Space Parking

Zhengbao Yao¹, Yuanfu Luo^{1,2}, Kehan Xue¹

¹All authors are with xLean Robotics Co., Ltd.

²Corresponding author: Yuanfu Luo (e-mail: yuanfu@xlean.ai)

Abstract—Autonomous parking in nonconvex and narrow environments remains challenging. Although optimal-control methods can explicitly enforce vehicle dynamics and collision constraints, nonconvexity compromises solver robustness and can cause failures. Large language models (LLMs) exhibit strong semantic reasoning capabilities, but directly generating dense trajectories makes it difficult to guarantee physical feasibility. We introduce SE-LLM-OCP, a unified framework in which LLMs make high-level discrete maneuver decisions, while an optimal-control module enforces low-level vehicle dynamics and collision constraints. Online, the LLM proposes sparse maneuver plans, decomposing the parking task into a sequence of short-horizon trajectory-optimization problems. A low-level solver then sequentially solves optimal-control problems. If the solver fails, the LLM aggregates failure evidence from the solver and validation stages to guide replanning. Offline, SE-LLM-OCP automatically evolves a structured decision-making knowledge base from scratch, driven by accumulated online failures. We validate our proposed framework in simulation on a car-like vehicle model and on a differential-drive robot. Our experimental results show that SE-LLM-OCP enables safer autonomous parking in narrow scenarios and demonstrates transfer of the same maneuver representation to a different kinematic platform.

I. INTRODUCTION

Autonomous parking in cluttered environments is a canonical hybrid motion-planning problem. A vehicle trajectory must satisfy nonholonomic kinematics and full-footprint collision constraints, yet planner success often depends on discrete maneuver choices: the approach direction, the ordering of forward and reverse segments, and the placement of intermediate configurations. Search-based planners such as Hybrid A* (HA*) explore these choices with discrete nonholonomic motion primitives [1]. Optimization-based planners instead optimize continuous paths or trajectories under collision costs or constraints and, where modeled, vehicle kinematics or dynamics [2], [3]. The two paradigms fail in different ways. Search effort depends on the resolution and coverage of the motion primitives, whereas nonlinear trajectory optimization is sensitive to initialization and the selected homotopy class. These weaknesses become pronounced in nonconvex and narrow spaces: an unsuitable maneuver structure may lead to excessive search or local optimization failure even though a feasible trajectory exists.

Large language models offer a complementary source of high-level structure, especially when paired with visual or structured scene representations. Language-grounded robotic

systems have used foundation models to decompose instructions into admissible actions, ground skill selection in affordances, synthesize executable policy programs, and plan over scene-level representations [4]–[7]. Autonomous-driving studies have likewise explored language-model reasoning for motion planning and coordinate-based trajectory generation [8], [9]. These advances, however, do not remove the physical-grounding problem. A maneuver can be semantically plausible yet fail because of a small terminal-pose error, insufficient clearance, or an infeasible swept volume. Asking the model to generate dense waypoints or controls also assigns it a long sequence of metric decisions that are difficult to verify. The question, then, is how to give an LLM the maneuver-planning role in narrow-space parking while keeping physical feasibility with the solver: semantic decisions must be turned into physically feasible trajectories, and low-level failures must come back as guidance the LLM can act on.

This paper answers this question with SE-LLM-OCP (Self-Evolving LLM-Guided Optimal Control Planning), a hierarchical framework for narrow-space autonomous parking. In SE-LLM-OCP, an LLM serves as the maneuver-level planner: from the scene, the task, and a parking-decision knowledge base, it decides how the vehicle should maneuver—where to stage, which direction to move, and how to approach the goal—and expresses these decisions as sparse terminal poses and forward or reverse motion modes, each accompanied by a short rationale. It never generates dense trajectories or low-level controls. A deterministic optimal-control layer then turns each decision into a short-horizon optimal control problem (OCP), enforcing nonholonomic vehicle kinematics, actuation limits, and full-footprint collision avoidance. Semantic reasoning and metric optimization are thus kept in strictly separate layers, each doing what it does best.

Because LLM-proposed poses are approximate, the grounding stage does not demand that they be met exactly: each terminal pose is grounded into a small tolerance region adapted to the surrounding clearance, so minor endpoint errors are absorbed without weakening the kinematic or collision constraints. When optimization or validation nonetheless rejects a maneuver, the failure is made actionable. Solver diagnostics, validation errors, and the rationales of the failed decisions are summarized by a feedback LLM into maneuver-level guidance, and the planning LLM revises its

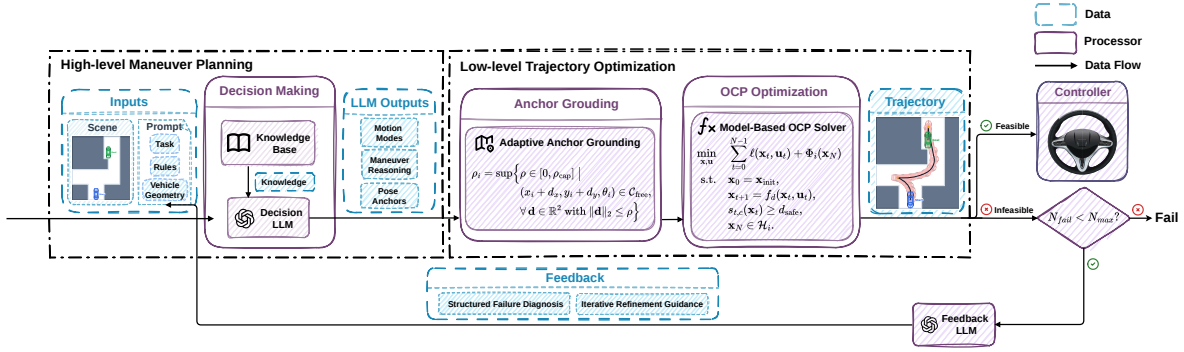


Fig. 1. Online hierarchical pipeline of SE-LLM-OCP: the decision-making LLM turns scene and task inputs plus retrieved parking knowledge into pose anchors and motion modes; these define sequential segment OCPs, and accepted trajectories are passed to the controller. Failed attempts feed a feedback LLM that revises the maneuver while the attempt budget permits.

decision. Accumulated failures additionally drive an offline loop that grows the parking-decision knowledge base from an initially empty state, with every entry generated automatically; an update is retained only when the resulting planner again produces a feasible trajectory. Throughout, the LLM parameters remain fixed: *self-evolution* proceeds through bounded, failure-conditioned revision of the maneuver online and of the decision knowledge offline.

The main contributions of this work are:

- a hierarchical framework for narrow-space autonomous parking that pairs LLM-based maneuver reasoning with deterministic optimal control, producing physically feasible trajectories under full vehicle kinematics and collision avoidance;
- a semantic-to-optimization grounding that absorbs the metric approximation inherent in LLM decisions, compiling them into tolerance-adapted terminal conditions for short-horizon OCPs without weakening kinematic or collision constraints;
- a bounded, failure-conditioned self-evolution loop that converts failed attempts into maneuver-level guidance, letting the LLM recover from infeasible decisions without unbounded replanning; and
- an offline knowledge self-evolution loop that bootstraps the parking-decision knowledge base from an initially empty state, with all entries generated automatically, and retains an update only when it yields a feasible trajectory.

We validate SE-LLM-OCP in simulation across confined parking and turnaround tasks. Under a shared Stanley-PID controller, SE-LLM-OCP attains the highest closed-loop execution success and the lowest collision rate among the compared planners, with fewer gear changes and lower path-curvature metrics than HA*+OCP; ablations isolate the contributions of the evolved knowledge base, failure feedback, and intermediate soft constraints, and a differential-drive experiment demonstrates that the same maneuver representation transfers to a different platform.

II. RELATED WORK

A. Traditional Planning Approaches

Traditional parking planners primarily employ search-based methods and trajectory optimization. Reeds–Shepp curves provide shortest paths for a car with bounded curvature and forward–reverse motion in obstacle-free space [10]. Hybrid A* incorporates vehicle kinematics into heuristic search, retaining continuous vehicle states within a discretized search space and refining the resulting path through numerical optimization [1]. These approaches provide geometric foundations and initial paths for parking trajectory generation.

Optimization-based methods formulate parking as an optimal control problem with explicit vehicle-motion and collision-avoidance constraints. Li et al. addressed time-optimal parallel parking through simultaneous dynamic optimization [11]. They later introduced safe travel corridors for parking, where a coarse path guides corridor construction and different parts of the vehicle are constrained within corresponding collision-free regions [12]. This representation replaces large-scale collision-avoidance constraints with a reduced-scale OCP whose size is independent of environmental complexity. A subsequent lightweight iterative framework repeatedly reconstructs the corridor to recover free space discarded by a fixed corridor and improve trajectory optimization among irregularly placed obstacles [13]. Corridor-based formulations therefore make continuous optimization tractable after a coarse route has been supplied. SE-LLM-OCP addresses the complementary high-level problem: an LLM proposes and revises sparse staging poses and motion modes, while the model-based OCP layer generates continuous trajectories under vehicle-kinematic and collision-avoidance constraints.

B. End-to-End Parking Approaches

End-to-end approaches learn parking behavior from scene observations and target information. Yang et al. investigated end-to-end neural parking in the CARLA simulator [14]. ParkingE2E predicts parking waypoints from surround-view images and a target parking slot using imitation learning

and an autoregressive transformer [15]. MultiPark extends learned parking to multimodal, multi-segment prediction, using parking queries that encode gear, longitudinal, and lateral behaviors together with goal- and collision-related training objectives [16]. These developments address maneuver diversity and long-horizon parking within learned planning architectures. In SE-LLM-OCP, the learned component instead proposes sparse pose anchors and motion modes. Continuous trajectories are generated by solving model-based OCPs, with vehicle constraints enforced during optimization and complete plans checked before execution.

C. Grounding and Self-Evolution in LLM-Based Planning

Grounding connects language-level reasoning to robot capabilities and physical environments. SayCan combines language-model scores with skill affordances to select executable actions [5]. AutoTAMP translates language instructions into formal task representations for task and motion planning, using corrective feedback to resolve translation errors [17]. VoxPoser constructs spatial value maps that guide model-based motion planning [18]. For narrow-space parking, the grounding interface must accommodate non-holonomic motion and vehicle-footprint collision constraints.

Feedback and persistent memory further support the revision of LLM-generated decisions. Inner Monologue incorporates environment feedback into subsequent planning [19], while Reflexion stores verbal reflections on task outcomes to guide later attempts without updating model parameters [20]. In autonomous driving, DiLu combines reasoning, reflection, and memory to retrieve relevant experiences and revise unsuccessful decisions, but its memory is seeded with manually outlined scenarios [21]. Building on these directions, SE-LLM-OCP couples online maneuver refinement with offline revision of parking knowledge through a common optimization and validation pipeline, evolving the knowledge base from an initially empty state without hand-authored entries. Online, solver diagnostics and trajectory-validation failures guide bounded replanning of pose anchors and motion modes. Offline, accumulated failures inform revisions to decision principles; a candidate knowledge update is retained only after the resulting planning process yields a trajectory that passes the same online acceptance checks. This links both decision revision and knowledge retention to the feasibility of the generated parking trajectories.

III. METHODS

Our framework SE-LLM-OCP comprises an online hierarchical pipeline (Fig. 1) and an offline knowledge self-evolution loop (Fig. 2). The online hierarchy allows LLM-based maneuver decisions to guide trajectory optimization while vehicle dynamics and collision constraints are enforced by the OCP solver. Online replanning revises the maneuver for the current task using a fixed knowledge-base version. Knowledge revision is performed separately: accumulated failures guide candidate knowledge-base updates. These updates are evaluated through the online pipeline before being used in subsequent planning tasks.

A. Online Hierarchical Pipeline

The online pipeline comprises high-level maneuver planning, low-level trajectory optimization, and feedback and replanning. The LLM specifies pose anchors and motion modes that divide the parking task into segments. These decisions are converted into terminal conditions for sequential OCPs, and the resulting trajectory is checked for feasibility before controller execution. Failed optimization or trajectory validation provides feedback for revising the maneuver within a prescribed attempt budget (three planning attempts per task).

1) *High-level Maneuver Planning*: A parking scene is defined by a bounded workspace $\mathcal{W} \subset \mathbb{R}^2$, the union of polygonal obstacle regions $\mathcal{O}$, an initial pose $\mathbf{p}_0 = (x_0, y_0, \theta_0)$, and a goal pose $\mathbf{p}_g$. The region $\mathcal{W} \setminus \mathcal{O}$ is available for vehicle motion. Let $\mathcal{F}(\mathbf{p})$ denote the oriented rectangular vehicle footprint at pose $\mathbf{p} = (x, y, \theta)$. The collision-free configuration set is

$$\mathcal{C}_{\text{free}} = \{\mathbf{p} \mid \mathcal{F}(\mathbf{p}) \subseteq \mathcal{W} \setminus \mathcal{O}\}. \quad (1)$$

The task is to reach $\mathbf{p}_g$ from $\mathbf{p}_0$ through configurations in $\mathcal{C}_{\text{free}}$ while satisfying the vehicle dynamics and actuation limits.

The decision-making LLM receives the task specification, scene image, metric geometry, start and goal poses, and vehicle dimensions. Relevant knowledge is retrieved by matching the current scene against the accumulated entries, each of which records a scene description together with explicitly allowed and forbidden operations. The retrieved entries guide the selection of staging poses and motion modes. The LLM generates an ordered maneuver plan with N segments:

$$\Pi = \{(\mathbf{p}_i, m_i, e_i)\}_{i=1}^N, \quad (2)$$

where $\mathbf{p}_i$ is the terminal-pose anchor of segment i , $m_i \in \{\text{forward}, \text{reverse}\}$ specifies its direction of travel, and e_i is a concise natural-language rationale explaining why the segment moves this way. Each rationale is carried through grounding and optimization; when a segment subsequently fails to solve or validate, its rationale is returned to the LLM together with the failure evidence, so that revisions can build on the original intent of the failed decision. Intermediate anchors specify staging poses for reorientation and goal approach, and the final anchor coincides with the goal, $\mathbf{p}_N = \mathbf{p}_g$. Vehicle parameters, objective weights, horizon lengths, and collision constraints remain system-defined.

2) *Low-level Trajectory Optimization*: The high-level maneuver plan is converted into terminal constraints and penalties for sequential OCPs via adaptive anchor grounding. Requiring every intermediate anchor to be reached exactly can overconstrain these OCPs, even when nearby staging poses would permit feasible motion. Bounded endpoint adjustments are therefore allowed, with penalties favoring the nominal anchors. The final goal remains fixed. We first describe how the anchors are grounded into terminal regions and penalties, and then formulate the segment OCPs.

Anchor grounding. To make the allowed adjustments precise, we attach to each intermediate anchor $\mathbf{p}_i$ a positional

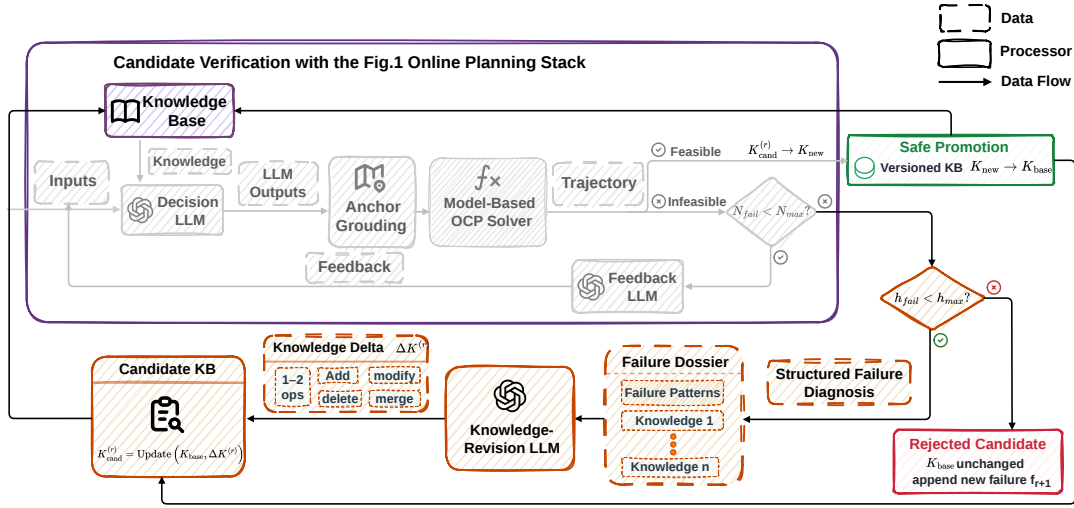


Fig. 2. Offline knowledge self-evolution loop of SE-LLM-OCP: failure dossiers drive a knowledge-revision LLM to propose knowledge deltas; each candidate version is verified through the online pipeline in Fig. 1, and candidates yielding feasible trajectories are retained, while failed verification guides further revisions within the offline budget.

tolerance r_i and a heading tolerance $\Delta\theta_{\text{tol}}$, which bound how far the endpoint of segment i may deviate from the nominal anchor in position and heading, respectively. The positional tolerance is derived from a clearance radius ρ_i , which measures the vehicle's translational freedom at the anchor. For an intermediate anchor $\mathbf{p}_i = (x_i, y_i, \theta_i)$, ρ_i is the largest radius within which the vehicle can translate in any direction without collision while its heading is held at θ_i . With a cap ρ_{cap} and planar displacement $\mathbf{d} = (d_x, d_y)$, it is defined as

$$\rho_i = \sup\{\rho \in [0, \rho_{\text{cap}}] \mid (x_i + d_x, y_i + d_y, \theta_i) \in \mathcal{C}_{\text{free}}, \forall \mathbf{d} \in \mathbb{R}^2 \text{ with } \|\mathbf{d}\|_2 \leq \rho\}, \quad i < N. \quad (3)$$

The positional tolerance r_i bounds the positional deviation of the segment endpoint from its nominal anchor and is computed as

$$r_i = \max\{r_{\text{min}}, \min(\varepsilon_{\text{task}}, k\rho_i)\}, \quad i < N. \quad (4)$$

Here, k is a scale factor and $\varepsilon_{\text{task}}$ is the task-level positional tolerance. The lower bound r_{min} prevents the tolerance from vanishing.

The terminal pose region $\mathcal{T}_i$ combines the positional tolerance r_i with the heading tolerance $\Delta\theta_{\text{tol}}$. The endpoint of segment i is required to lie in the set

$$\mathcal{T}_i = \{\mathbf{p} = (x, y, \theta) \mid \|(x, y) - (x_i, y_i)\|_2 \leq r_i, |\text{wrap}(\theta - \theta_i)| \leq \Delta\theta_{\text{tol}}\}, \quad i < N. \quad (5)$$

Here, wrap returns the principal angular difference. For the final segment, the set reduces to $\mathcal{T}_N = \{\mathbf{p}_g\}$.

The terminal region bounds the endpoint adjustment, while a terminal pose penalty J_{term}^i favors the nominal anchor. Its position and heading weights are w_{pos}^i and w_{θ}^i . The position weight is reduced as the tolerance grows, so that a generous

tolerance weakens the pull of the penalty toward the nominal anchor; the heading weight, in contrast, is held fixed at w_{θ}^{int} to keep the approach direction precise. For intermediate anchors, the position weight is computed as

$$w_{\text{pos}}(r_i) = \max\left(w_{\text{pos}}^{\text{min}}, w_{\text{pos}}^{\text{max}} \frac{r_{\text{min}}}{\max(r_i, r_{\text{min}})}\right) \quad (6)$$

where the constants are set empirically to $w_{\text{pos}}^{\text{min}} = 80$, $w_{\text{pos}}^{\text{max}} = 200$, and $w_{\theta}^{\text{int}} = 400$. For the final segment, the weights are fixed to $w_{\text{pos}}^{\text{goal}} = 1000$ and $w_{\theta}^{\text{goal}} = 2000$, likewise set empirically. The operation TerminalCost constructs J_{term}^i from the anchor and these weights.

The region itself does not certify collision freedom: ρ_i assumes a fixed heading, whereas $\mathcal{T}_i$ permits heading changes and the radius mapping can yield $r_i > \rho_i$. Collision constraints are therefore imposed independently in the OCP.

Algorithm 1 summarizes the mapping from the maneuver plan Π to the grounded plan Γ . Intermediate segments receive adaptive position tolerances and penalty weights; the final segment retains the exact goal and fixed weights. Each entry Γ_i contains the maneuver attributes, positional tolerance, terminal region, and terminal penalty needed to formulate the corresponding segment OCP.

Segment OCP formulation. Vehicle motion follows the rear-axle kinematic model with state $\mathbf{x} = [x, y, \theta, v, \delta]^\top$ and control $u = [a, \omega_\delta]^\top$, where v is signed longitudinal velocity, δ is steering angle, a is longitudinal acceleration, and ω_δ is steering rate. With wheelbase L , the dynamics are

$$\begin{aligned} \dot{x} &= v \cos \theta, & \dot{y} &= v \sin \theta, \\ \dot{\theta} &= \frac{v \tan \delta}{L}, & \dot{v} &= a, & \dot{\delta} &= \omega_\delta. \end{aligned} \quad (7)$$

For segment i , the terminal-state set H_i requires the terminal pose to lie in $\mathcal{T}_i$, and the admissible velocity interval $V(m_i)$ enforces the specified direction of travel. The terminal cost Φ_i combines J_{term}^i with a penalty that encourages stopping

Algorithm 1: Adaptive Soft-Constraint Grounding of LLM Anchors

Input: Plan Π ; $\mathcal{C}_{\text{free}}$; goal $\mathbf{p}_g$; parameters $(\varepsilon_{\text{task}}, k, r_{\text{min}}, \rho_{\text{cap}}, \Delta\theta_{\text{tol}})$

Output: Grounded plan Γ

```

1  $N \leftarrow |\Pi|$ ;  $\Gamma \leftarrow \emptyset$ ;
2 for  $i \leftarrow 1$  to  $N$  do
3    $(\mathbf{p}_i, m_i, e_i) \leftarrow \Pi[i]$ ;
4   if  $i = N$  then
5      $\mathbf{p}_i \leftarrow \mathbf{p}_g$ ;  $r_i \leftarrow 0$ ;  $\mathcal{T}_i \leftarrow \{\mathbf{p}_g\}$ ;
6      $(w_{\text{pos}}^i, w_{\theta}^i) \leftarrow (w_{\text{pos}}^{\text{goal}}, w_{\theta}^{\text{goal}})$ ;
7   else
8     Compute  $\rho_i$  using (3) and  $r_i$  using (4);
9     Compute  $w_{\text{pos}}^i$  using (6) and set  $w_{\theta}^i \leftarrow w_{\theta}^{\text{int}}$ ;
10    Compute  $\mathcal{T}_i$  using (5);
11     $J_{\text{term}}^i \leftarrow \text{TerminalCost}(\mathbf{p}_i, w_{\text{pos}}^i, w_{\theta}^i)$ ;
12     $\Gamma_i \leftarrow (\mathbf{p}_i, m_i, e_i, r_i, \mathcal{T}_i, J_{\text{term}}^i)$ ;
13 return  $\Gamma = \{\Gamma_i\}_{i=1}^N$ ;

```

at the endpoint. Given the initial state $\mathbf{x}_{\text{init}}$ and the discrete dynamics f_d from (7), each segment is optimized by solving

$$\begin{aligned}
& \min_{\mathbf{x}, u} \sum_{t=0}^{T-1} \ell(\mathbf{x}_t, u_t) + \Phi_i(\mathbf{x}_T) \\
& \text{s.t. } \mathbf{x}_0 = \mathbf{x}_{\text{init}}, \\
& \quad \mathbf{x}_{t+1} = f_d(\mathbf{x}_t, u_t), \\
& \quad s_{t,c}(\mathbf{x}_t) \geq d_{\text{safe}}, \\
& \quad \mathbf{x}_T \in H_i.
\end{aligned} \tag{8}$$

Here, T denotes the prediction horizon of the current segment. The quantity $s_{t,c}$ denotes clearance for collision constraint c at step t , and d_{safe} is the required margin. Control and steering bounds and $v_t \in V(m_i)$ are also imposed. The stage cost ℓ regularizes acceleration, steering rate, steering angle, velocity, and obstacle slack. Horizon selection and solver retry settings remain system-defined.

The first OCP starts from the initial vehicle state. Each subsequent OCP starts from the preceding segment’s optimized terminal state, including velocity and steering angle. Transferring the full state preserves continuity and allows the next segment to account for the preceding endpoint adjustment.

3) *Feedback and Replanning:* The complete maneuver is checked after all segment OCPs have been solved. Validation assesses endpoint accuracy, continuity between segments, satisfaction of the terminal regions, and footprint feasibility using a sampled vehicle sweep. Trajectories that pass all feasibility checks are sent to the controller.

When OCP optimization fails or the assembled trajectory fails validation, solver diagnostics and validation errors are passed to the feedback LLM together with the rationales of the failed segments. It summarizes potential causes at the maneuver level and provides guidance for replanning. The decision-making LLM uses this feedback, the task

inputs, and retrieved knowledge to revise the pose anchors and motion modes. The revised plan undergoes the same grounding, optimization, and validation procedure, with the vehicle model and acceptance criteria held fixed.

The online loop terminates when a feasible trajectory is found or the prescribed attempt budget is exhausted. This feedback revises the maneuver for the current task; the associated failure evidence is retained for offline knowledge revision.

B. Offline Knowledge Self-Evolution Loop

The offline loop in Fig. 2 uses accumulated planning failures to revise the decision principles retrieved during high-level planning. The knowledge base is initialized empty—that is, zero-shot: every entry is generated automatically, with no parking maneuvers, staging heuristics, or exemplars authored by hand. Prompt structure and revision operations remain system-defined; only the knowledge entries evolve through the automated revise-and-verify cycle below. A failure dossier combines observed failure patterns with relevant existing knowledge entries. The knowledge-revision LLM uses this dossier to propose a knowledge delta comprising one or two operations selected from addition, modification, deletion, and merging. Every generated entry must record the scene it applies to together with explicitly allowed and forbidden operations. Applying the delta to the retained knowledge base produces a candidate version for verification.

Each candidate knowledge-base version is evaluated through the online pipeline in Fig. 1, with knowledge retrieved from that version. The resulting maneuver plans undergo grounding, sequential OCP solution, and trajectory validation. Online feedback and replanning remain active within the prescribed attempt budget. The vehicle model, grounding rules, optimization settings, and acceptance criteria are held fixed during verification.

A candidate is retained as the new knowledge-base version only when verification produces a feasible trajectory. If verification fails, the retained version remains unchanged and the new failure evidence is appended to the dossier to guide further revision. The online attempt budget limits maneuver attempts during each candidate evaluation; a separate offline revision budget limits candidate knowledge updates. The offline loop terminates when a candidate passes verification or the revision budget is exhausted. The loop thus grows the knowledge base from an empty initialization into a task-specific body of parking expertise, retaining only revisions that demonstrably preserve feasibility.

IV. EXPERIMENTS

We evaluate SE-LLM–OCP in confined parking environments through planner comparisons, component ablations, and real-robot execution. The simulation study examines planning success, trajectory quality, and closed-loop performance; the physical experiment demonstrates execution on a differential-drive robot.

TABLE I
PLANNING QUALITY AND CLOSED-LOOP PERFORMANCE UNDER THE COMMON STANLEY-PID CONTROLLER ON 38 SCENARIOS.

Method	Planning						Closed-loop execution	
	Succ. (%) $\uparrow$	Gear shifts $\downarrow$	Mean $ \kappa $ (m^{-1}) $\downarrow$	Max $ d\kappa/ds $ (m^{-2}) $\downarrow$	Pos. err. (m) $\downarrow$	Orie. err. (deg) $\downarrow$	Succ. (%) $\uparrow$	Coll. (%) $\downarrow$
HA*	92.1	2.571	0.158	4.000	—	—	55.3	40.0
Direct OCP	0.0	—	—	—	—	—	0.0	—
HA*+OCP	76.3	2.138	0.158	3.556	0.028	0.529	63.2	17.2
SE-LLM-OCP	94.7	1.593	0.116	2.437	0.006	0.112	84.2	11.1

Planning metrics are averaged over successful plans, and terminal errors are measured on the planned trajectories; collision rates count an obstacle collision or boundary violation once per run, over successfully planned runs that enter execution. Dashes indicate values that are not reported (terminal errors for HA*) or undefined (Direct OCP, which produces no validated trajectory).

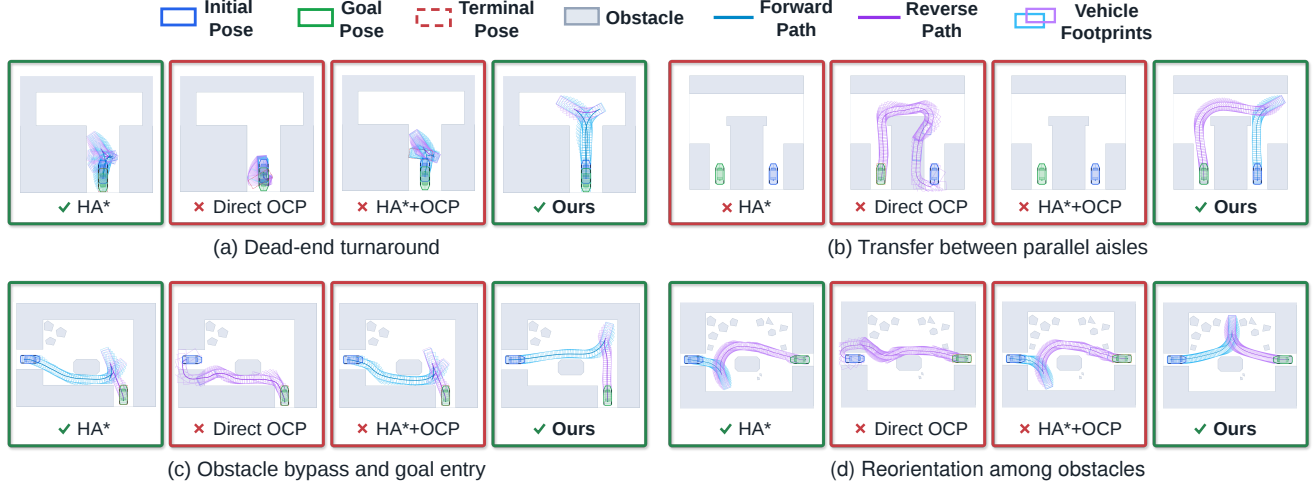


Fig. 3. Qualitative comparisons across four parking scenarios: (a) dead-end turnaround, (b) transfer between parallel aisles, (c) obstacle bypass and goal entry, and (d) reorientation among obstacles. Each group shows, from left to right, HA*, Direct OCP, HA*+OCP, and SE-LLM-OCP (Ours); green frames with check marks indicate success, red frames with crosses failure. Examples are from additional runs, separate from the aggregate evaluation in Table I.

A. Experimental Setup

Scenes and vehicle. The simulation benchmark comprises parking tasks in confined environments with nonconvex free space. Each task specifies an initial pose and a goal pose. The car-like model has a 4.50×1.85 m footprint, a 2.70 m wheelbase, and a minimum rear-axle turning radius of 5.00 m.

Controller and execution. All planners are evaluated on the same simulation tasks. Their reference trajectories are tracked using a Stanley lateral controller and a PID speed controller. Controller settings and the execution model are shared across planners, with acceleration and steering-rate limits enforced during execution.

Evaluation metrics. Planning success requires the generated reference trajectory to pass trajectory validation. For SE-LLM-OCP, this is the validation stage of the online pipeline. Closed-loop success requires successful planning followed by execution to within 0.50 m of the goal position and 10° of the goal heading, without obstacle collision or departure from the drivable region. Both success rates use all runs as the denominator. The collision rate is computed only over runs that pass planning validation and proceed to controller execution, including unsuccessful executions. A

run is counted once if an obstacle collision or a boundary violation occurs. Planning failures are excluded from this denominator.

Planning-quality metrics are averaged over successful planning runs. They comprise the number of gear shifts, mean absolute curvature $|\kappa|$ (m^{-1}), and maximum absolute spatial curvature gradient $|d\kappa/ds|$ (m^{-2}), where s denotes arc length. The curvature-gradient entry is the mean of the per-trajectory maxima. Terminal position and orientation errors measure the final pose error of the planned trajectory; they are distinct from errors after controller execution. Terminal errors are compared between SE-LLM-OCP and HA*+OCP and are not reported for HA*.

Evaluation protocol and planning budget. For the planner comparison and ablation studies, all configurations are evaluated under a consistent experimental protocol. Independent evaluation trials are distinguished from the planning attempts performed within each trial. All LLM components use gpt-5.6-sol. Unless replanning is ablated, SE-LLM-OCP performs bounded feedback-guided replanning when the current proposal fails validation. Each planning attempt undergoes grounding, optimization, and validation, while the knowledge-base version remains fixed throughout each

online trial.

B. Experiment 1: Planning and Closed-Loop Performance

We compare SE-LLM-OCP with HA* [1], Direct OCP, and HA*+OCP. HA* searches for a kinematically admissible reference, which HA*+OCP then refines through trajectory optimization. Direct OCP optimizes the maneuver without LLM-generated intermediate anchors. In SE-LLM-OCP, pose anchors and motion modes specify the maneuver structure before the segment OCPs are solved.

Table I shows that SE-LLM-OCP achieves the best overall balance between planning feasibility and closed-loop execution. It attains 94.7% planning success and 84.2% closed-loop success, while also producing trajectories with fewer gear shifts and lower curvature-related metrics than HA* and HA*+OCP. Although HA* reaches a comparable planning success rate, its substantially lower closed-loop success indicates that a geometrically valid reference does not necessarily translate into an easily executable parking maneuver. Adding OCP refinement improves the execution performance of HA*, but HA*+OCP still remains clearly below SE-LLM-OCP, suggesting that continuous trajectory refinement alone cannot fully compensate for an unfavorable maneuver structure. Direct OCP produces no validated trajectory in the evaluated nonconvex scenes.

Figure 3 explains where this advantage comes from. The key difference is that SE-LLM-OCP determines the maneuver structure before numerical optimization by jointly considering obstacle bypass, staging, and the required goal approach. This allows major heading changes to be completed in regions with more available space and leaves the confined goal region mainly for final entry. In contrast, HA* determines the maneuver progressively through search, while HA*+OCP mainly refines the resulting reference without explicitly revisiting its earlier staging and bypass decisions.

Consequently, SE-LLM-OCP supplies the segment OCPs with a more favorable maneuver structure rather than relying on optimization to recover from a poor high-level choice. The OCPs then enforce vehicle and collision constraints within this structure. This division of roles explains why SE-LLM-OCP outperforms the baselines not only in planning and closed-loop success, but also in gear-shift and curvature metrics.

C. Experiment 2: Ablation Studies

We examine the evolved knowledge base, failure feedback, and intermediate soft constraints by removing each component in turn. The scene inputs, LLM, vehicle model, controller, and remaining planning settings are held fixed. Table II reports the results.

Evolved knowledge. The knowledge base is initialized empty and populated entirely through automated offline self-evolution. The variant without evolved knowledge therefore uses an empty knowledge base. In the full method, the accumulated knowledge guides staging-pose selection, approach headings, and motion modes before OCP construction. Using the evolved knowledge increases planning success from

TABLE II
COMPONENT ABLATIONS ON 38 SCENARIOS.

Configuration	Planning	Closed-loop execution	
	Succ. (%)↑	Succ. (%)↑	Coll. (%)↓
w/o evolved knowledge	85.1	74.6	12.4
w/o failure feedback	79.8	68.4	13.2
w/o soft constraints	83.3	76.3	8.4
Full method	94.7	84.2	11.1

Success rates use all runs; collision rates count an obstacle collision or boundary violation once per run, over successfully planned runs that enter execution (including execution failures).

85.1% to 94.7% and closed-loop success from 74.6% to 84.2%. These gains show the benefit of transferring accumulated planning experience to online maneuver selection.

Failure feedback. This variant retains the same computational budget and planning-attempt limit as the full method, but subsequent proposals receive no feedback from earlier failures. The full method uses solver and validation feedback to revise pose anchors and motion modes in response to the difficulties encountered during previous attempts. With the same budget, failure feedback increases planning success from 79.8% to 94.7% and closed-loop success from 68.4% to 84.2%. The improvement shows the value of using failed attempts to guide subsequent maneuver proposals.

Intermediate soft constraints. This variant solves each segment as a separate start-to-goal problem, applying the task-terminal constraints at every segment endpoint. Treating each intermediate anchor as a task goal restricts where adjacent segments can connect, even when a nearby pose could serve the same maneuver. The full method allows intermediate poses to adjust during grounding while retaining the final goal constraints. This increases planning success from 83.3% to 94.7% and closed-loop success from 76.3% to 84.2%, supporting the use of soft constraints to accommodate approximate LLM anchors. The ablated variant has a lower collision rate among executed plans (8.4% versus 11.1%), while the full method completes more tasks overall.

D. Experiment 3: Real-Robot Experiment

We conducted a physical experiment with the differential-drive robot shown in Fig. 4(a). The platform is equipped with a 2D LiDAR. The task requires the robot to leave a narrow aisle, pass around a partition, and reach the marked goal in a confined indoor environment.

Figure 4(b) shows the corresponding simulated reference, produced by the same planning pipeline with the segment OCPs re-solved under differential-drive kinematics instead of the car-like model. A reverse segment brings the robot out of the starting aisle into the connecting area, followed by a forward segment around the partition toward the goal. The intermediate pose markers indicate the staged maneuver, and the sampled footprints show the robot configurations along the reference.

The physical execution in Fig. 4(c) follows the same reverse-then-forward maneuver structure. The robot reverses

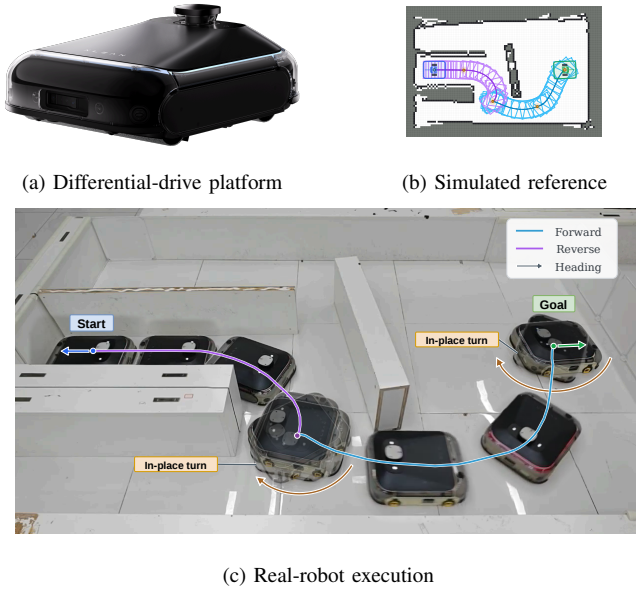


Fig. 4. Real-robot experiment and corresponding simulated reference. (a) Differential-drive robot equipped with a 2D LiDAR. (b) Simulated reference with intermediate pose markers and sampled footprints. (c) Successive robot poses overlaid to show the executed motion. Purple and blue curves indicate reverse and forward motion, respectively; the curves in (b) are the reference, those in (c) the motion sequence.

out of the aisle, reorients in the connecting area, and moves forward around the partition. In-place rotations are visible at the intermediate configuration and at the goal. The first separates departure from the confined aisle from the forward approach; the second adjusts the heading at the marked goal. The overlaid poses show the executed motion, while the colored curves illustrate its forward and reverse portions.

This experiment provides qualitative evidence that the reverse departure and forward goal approach can be carried through to physical execution. The intermediate staging configuration provides room to reorient, while in-place rotations separate translation from heading adjustment.

V. CONCLUSION

We presented SE-LLM-OCP, a parking framework that bridges LLM-based maneuver reasoning and model-based trajectory optimization. The framework uses a knowledge base evolved from scratch to guide staging poses and motion modes, grounds approximate intermediate anchors through adaptive soft constraints, and converts solver and validation failures into feedback for bounded replanning. Simulation experiments in narrow, nonconvex parking scenes demonstrated the effectiveness of this decision-optimization interface, while a differential-drive robot with a 2D LiDAR further illustrated physical execution of the planned maneuver. Future work will focus on repeated quantitative real-robot evaluation and extending the framework to additional kinematic platforms and parking environments.

REFERENCES

- [1] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," in *Proc. 1st Int. Symp. Search Tech. Artif. Intell. Robot.*, 2008.
- [2] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 489–494.
- [3] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, 2021.
- [4] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proc. 39th Int. Conf. Mach. Learn.*, ser. Proc. Mach. Learn. Res., vol. 162, 2022, pp. 9118–9147.
- [5] B. Ichter, A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Proc. 6th Conf. Robot Learn.*, ser. Proc. Mach. Learn. Res., vol. 205, 2023, pp. 287–318.
- [6] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 9493–9500.
- [7] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. S nderhauf, "SayPlan: Grounding large language models using 3d scene graphs for scalable robot task planning," in *Proc. 7th Conf. Robot Learn.*, ser. Proc. Mach. Learn. Res., vol. 229, 2023, pp. 23–72.
- [8] J. Mao, Y. Qian, J. Ye, H. Zhao, and Y. Wang, "GPT-Driver: Learning to drive with GPT," *arXiv preprint arXiv:2310.01415*, 2023.
- [9] X. Tian, J. Gu, B. Li, Y. Liu, Y. Wang, Z. Zhao, K. Zhan, P. Jia, X. Lang, and H. Zhao, "DriveVLM: The convergence of autonomous driving and large vision-language models," in *Proc. 8th Conf. Robot Learn.*, ser. Proc. Mach. Learn. Res., vol. 270. PMLR, 2025, pp. 4698–4726.
- [10] J. A. Reeds and L. A. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, 1990.
- [11] B. Li, K. Wang, and Z. Shao, "Time-optimal maneuver planning in automatic parallel parking using a simultaneous dynamic optimization approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 11, pp. 3263–3274, 2016.
- [12] B. Li, T. Acarman, X. Peng, Y. Zhang, X. Bian, and Q. Kong, "Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach," in *Proc. Eur. Control Conf.*, 2020, pp. 1993–1998.
- [13] B. Li, T. Acarman, Y. Zhang, Y. Ouyang, C. Yaman, Q. Kong, X. Zhong, and X. Peng, "Optimization-based trajectory planning for autonomous parking with irregularly placed obstacles: A lightweight iterative framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11 970–11 981, 2022.
- [14] Y. Yang, D. Chen, T. Qin, X. Mu, C. Xu, and M. Yang, "E2E Parking: Autonomous parking by the end-to-end neural network on the CARLA simulator," in *Proc. IEEE Intell. Veh. Symp.*, 2024, pp. 2375–2382.
- [15] C. Li, Z. Ji, Z. Chen, T. Qin, and M. Yang, "ParkingE2E: Camera-based end-to-end parking network, from images to planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 13 206–13 212. [Online]. Available: <https://arxiv.org/abs/2408.02061>
- [16] H. Zheng, Z. Zhou, G. Zhang, Z. Wang, K. Wang, P. Li, S. Shen, M. Yang, and T. Qin, "MultiPark: Multimodal parking transformer with next-segment prediction," *arXiv preprint arXiv:2508.11537*, 2025. [Online]. Available: <https://arxiv.org/abs/2508.11537>
- [17] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2024, pp. 6695–6702.
- [18] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "VoxPoser: Composable 3d value maps for robotic manipulation with language models," in *Proc. 7th Conf. Robot Learn.*, ser. Proc. Mach. Learn. Res., vol. 229, 2023, pp. 540–562.
- [19] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, *et al.*, "Inner monologue: Embodied reasoning through planning with language models," in *Proc. 6th Conf. Robot Learn.*, ser. Proc. Mach. Learn. Res., vol. 205, 2023, pp. 1769–1782.
- [20] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," in *Adv. Neural Inf. Process. Syst.*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html

- [21] L. Wen, D. Fu, X. Li, X. Cai, T. Ma, P. Cai, M. Dou, B. Shi, L. He, and Y. Qiao, "DiLu: A knowledge-driven approach to autonomous driving with large language models," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2024/hash/93c936b9e492def9c00782cab79dbc6d-Abstract-Conference.html