A Hierarchical Approach for Tractor-trailer Motion Planning Using Graph Search and Reinforcement Learning

Haitong Ma, Tianpeng Zhang, Na Li, Stefano Di Cairano, Yebin Wang

***Abstract*— This paper introduces a hierarchical motion planning strategy for autonomous tractor-trailer systems, designed for efficient long-horizon, collision-free maneuvering in complex environments. By combining high-level reference line graph search with low-level primal-dual reinforcement learning (RL)-based trajectory optimization, our approach addresses the computational challenges inherent to the motion planning of tractor-trailer dynamics. The high-level graph search decides waypoints guided by Reeds-Shepp cost, and the low-level RL connects the waypoints with dynamically feasible and collision-free trajectories. To enhance safety and accuracy, we incorporate reachability constraints and batch trajectory sampling in the RL algorithm design. Empirical results show that our method significantly reduces computation time, outperforming traditional state-lattice-based planning approaches and enabling real-time applicability.**

I. Introduction

Autonomous tractor-trailer systems have attracted strong interest from both industry and academia due to their high cargo transportation efficiency. However, their complicated dynamics pose significant challenges in motion planning, particularly, when reversing maneuvers and collision avoidance are needed. Moreover, there exists a tight budget for the computation time for the practical applications, which makes the problem more challenging.

The motion planning problem aims to solve an optimal control problem (OCP) that minimizes the cost-to-go while satisfying dynamics, inputs, states, and collision-avoidance constraints. Such a complex constrained optimization algorithm usually lacks global convergence guarantees, which requires non-trivial initialization to converge to global optimum [1]–[4], especially in cluttered environments. Local optimum is acceptable in practical applications. However, a bigger challenge in practice is the computation time. Directly solving the optimization problem is not possible in real-time. To reduce the computation time and make the problem real-time solvable, recent works can be categorized as (1) search-based motion and state-lattice-based planning with motion primitives and (2) learning-based motion planning.

Search-based motion planning abstracts the planning as a graph with nodes and edges and finds the shortest collision-free paths on the graph [5], [6]. To ensure the graph connection paths are dynamically feasible, the connection is constructed via pre-computing a series of dynamically feasible trajectories, also called *motion primitives* [6]–[8]. Then in the online phase, the algorithms conduct graph search such as A* [8]–[10] with collision check to find collision-free trajectories. Methods combining motion primitives and graph search are also called state-lattice-based algorithms. The major limitation of state-lattice-based methods is the curse of dimensionality, which means the number of pre-computed motion primitives scales exponentially with the system dimension and resolution error. Reducing the number of motion primitives will reduce the online planning success rate. As we have an additional trailer dimension in the dynamics, this method either requires a large number of motion primitives or suffers a low success rate if the number of primitives is insufficient [11]–[13].

Another solution to reduce computation time is to train a neural network as the motion planner, where the online computation is only forward passes of the neural network. Many recent works leveraged reinforcement learning (RL) to achieve end-to-end motion planning [14]–[17], where end-to-end means the inputs of the learning-based planner are maps and outputs are the planned trajectories. The learning-based planner usually maintains a waypoint graph, either in configuration space [14] or latent space [16]. Then an exploration policy is trained to expand the graph using neural network models such as transformer [15] and graph neural networks [17]. The major difficulties of learning-based planning are exploration scheme design and collision checking. [15] shows that even with the simple Dubin's car in a 2D workspace, the shape of the corresponding 3D free space in configuration space is very irregular, posing significant difficulties in exploration scheme design. Collision checking also reduces training efficiency. These difficulties make end-to-end learning-based planning not practical on the tractor-trailer system.

Contributions. In this paper, we proposed a hierarchical approach that combines high-level graph search and low-level RL-based trajectory optimization to perform long-horizon planning for tractor-trailer systems. We first search a series of intermediate reference lines via the shortest path on the reference line graph search, where graph nodes are directed straight reference line segments in the environments, and the edge exists only if the transition is feasible with one forward or reverse turn of the tractor. Then we design a primal-dual reinforcement learning (RL) algorithm to generate dynamically feasible collision-free trajectories connecting successive

H. Ma, T. Zhang, N. Li are with the Harvard School of Engineering and Applied Sciences. This work was done while H. Ma and T. Zhang were research interns and N. Li was a visiting researcher at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA. (email: {haitongma,tzhang}@g.harvard.edu, nali@seas.harvard.edu).

S. Di Cairano and Y. Wang are with MERL. (email: {dicairano,yebinwang}@ieee.org).

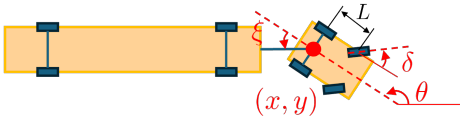


Fig. 1: Dynamics of a front-drive tractor-trailer.

reference lines. We incorporate other practical tricks, such as statewide dual variable, reachability-based safety constraints, and batch trajectory sampling with stochastic policy, to make the trajectory precise and collision-free. Compared to the state-of-the-art state-lattice-based methods, we significantly improve the planning efficiency by avoiding solving numerous motion primitives offline and online tree searches. Simulations demonstrate the effectiveness of the proposed method in terms of computation time and planning success rate.

The paper is organized as follows. Section II introduces the tractor-trailer systems and overall motion planning problem formulation. The following two sections provide the technical details about the hierarchical framework, where Section III presents the high-level graph search and Section IV shows the low-level RL-based trajectory optimization. Empirical results are demonstrated in Section V and Section VI concludes the paper.

II. Problem Setup and Preliminaries

A. Tractor-trailer System

Consider a front wheel drive *standard trailer system* [18], [19] as shown in Fig. 1, where $(x, y)^\top$ are the coordinates of the midpoint of the tractor's rear wheel axis, θ is the tractor orientation, θ_1 is the orientations of trailer, v_f is the front-wheel velocity of the tractor, δ is the steering angle of the tractor, and L is the wheelbase on the tractor. A mechanical constraint $|\delta| \leq \delta_{\max}$ limits the minimum turning radius R of a path with $\tan(\delta_{\max}) = \frac{L}{R}$. We write the tractor-trailer dynamics model $\dot{s} = f(s, u)$ in the coordinates $s = (x, y, \theta, \xi, v, \delta)^\top$ as follows:

$$\begin{aligned} \dot{x} &= \cos(\theta)v \qquad \dot{\xi} = -\frac{v\sin(\xi)}{d_1} - \frac{v\tan(\delta)}{L} \\ \dot{y} &= \sin(\theta)v \qquad \dot{v} = u_1 \\ \dot{\theta} &= \frac{v\tan(\delta)}{L} \qquad \dot{\delta} = u_2 \end{aligned} \tag{1}$$

where the control inputs are the acceleration u_1 and the steering rate u_2. The system needs to avoid a jack-knife configuration,

$$|\xi| \leq \xi_{\max}, \tag{2}$$

where $\xi_{\max}$ must be less than $\frac{\pi}{2}$.

B. Motion Planning Problem Formulation

The overall motion planning problem is stated below. Denote $\mathcal{S}_{free}$ the collision-free configuration space of system (1). Given an initial configuration $s_i \in \mathcal{S}_{free}$ and a goal configuration $s_f \in \mathcal{S}_{free}$, the motion planning algorithm aims to find a feasible trajectory which starts at s_i and ends at s_f, while satisfying (1) and a series of constraints,

$$\min_{u_0, u_1, \ldots, u_{N-1}, N} \sum_{t=1}^{N} l(s_t, u_t) \tag{3a}$$

$$\text{s.t. } s_0 = s_i, s_N = s_f \tag{3b}$$

$$s_t = f_d(s_{t-1}, u_{t-1}), s_t \in \mathcal{S}_{free} \tag{3c}$$

$$|v_t| \leq v_{\max}, \ |\delta_t| \leq \delta_{\max}, |\xi_t| \leq \frac{\pi}{2} \tag{3d}$$

$$a_{\min} \leq a_t \leq a_{\max}, \ \dot{\delta}_{\min} \leq \psi_t \leq \dot{\delta}_{\max} \tag{3e}$$

$$\forall t = 1, 2, \ldots, N$$

where l is a cost function, f_d in (3c) is the proper discretization of continuous tractor-trailer dynamics in (1), (3d) indicates the operation range of velocity v, steering angle δ, and avoidance of jack-knife configuration on the relative angle ξ, and (3e) indicates the input constraints.

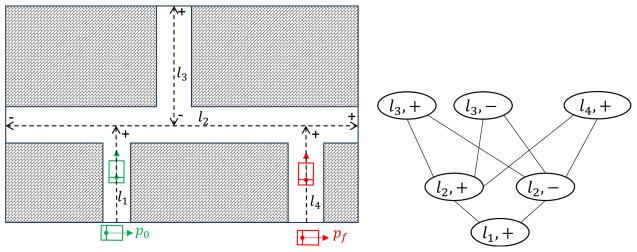
Directly solving (3) with RL is very difficult empirically for multiple reasons:

- The dynamics is non-holonomic, which results in that the tractor-trailer usually needs complex maneuvers to reach goal state. Simply defining reward as minimizing the Euclidean distance will not guide to correct solutions.
- The time step N is not fixed while there exists terminal constraints on the terminal time step.

To handle these challenges, we present the hierarchical framework, where a high-level graph search decides a set of intermediate waypoints as guidance, and a low-level RL-based trajectory generator solves the sub-tasks connecting successive waypoints.

III. Reference Line Guided Search

In this section, we conduct the *high-level graph search* to decide a series of intermediate sub-goals. We first introduce how to abstract the environments and construct the reference line graph, and then introduce our graph-search algorithm design to find the shortest reference line path and generate sub-goals.



(a) Planning environment (b) The connection between reference lines

Fig. 2: Problem setup for tractor-trailer planning with reference lines.

A. Reference Line Graph Construction

Given a road map shown in Figure 2(a), the reference lines are directed line segments $l_1, l_2, ..., l_n$ that constitute a skeleton of the roadmap, as shown in Figure 2(a). Each l_i

is endowed with two cardinal directions $+$ and $-$ denoted as $\text{node}_\text{i} = (l_i, \sigma_i)$ where $\sigma \in (+, -)$. The reference line graph, shown by Figure 2(b), is defined by the transition feasibility between reference line segments. Each node is a directed reference line segment (l_i, σ_i) and the edge exists between two nodes (l_i, σ_i) and (l_j, σ_j) if there are states $s_i \in (l_i, \sigma_i)$ and $s_j \in (l_j, \sigma_j)$, such that s_i can transition to s_j with a simple motion (forward, backward, left turn, right turn) while ignoring obstacles. For instance, there is an edge between $(l_1, +)$ and $(l_2, +)$ because the tractor can transition between the two with a 90-degree turn; meanwhile, since it is impossible to transition from $(l_1, +)$ to $(l_3, +)$ without chaining more than one simple motion together, there is not a direct edge between them.

B. Heuristic and Sub-goal Selection

We leverage A^*-based graph search, which uses a heuristic function to guide the shortest path search [9]. The cost to reach the goal is estimated as the sum of the heuristic cost to reach the goal and the actual cost from the start to that node. Then the A* algorithm maintains an open set of nodes to explore, prioritizing nodes with the lowest total estimated cost until the goal node is explored. Therefore, heuristics design is key to the graph search.

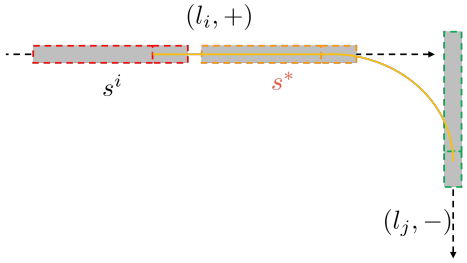


Fig. 3: Demonstration of edge cost calculation as minimization of two-stage Reeds-Shepp cost of the tractor in orange color.

We first define the edge cost function. The edge cost function is defined as the sum of two Reeds-Shepp costs [1], shown in Figure 3, which is solved the following optimization problem,

$$\text{EdgeCost}(\text{node}_i, \text{node}_j) = \min_{\substack{s^* \in (l_i, \sigma_i) \\ s^j \in (l_j, \sigma_j)}} c_{\text{RS}}(s^i, s^*) + c_{\text{RS}}(s^*, s^j) \tag{4}$$

where notation $\text{node}_i = (l_i, \sigma_i)$, and we slightly abuse the notation to let $s^* \in (l_i, \sigma_i)$ mean the tractor align with the reference line (l_i, σ_i). A tractor aligning with the reference line means the position is on the reference line and the heading angle is the same as the reference line direction. $c_{\text{RS}}(s, s')$ is the Reeds-Shepp cost transition from the tractor state in s to tractor state in s'. Note that we only consider the Reeds-Shepp cost of the tractor for simplicity. s_* is an intermediate waypoint. This two-stage transition ensures the tractor stays on the reference line for as long as possible, then deviates minimally from the reference lines.

For the heuristic function, we also use the Reeds-Shepp cost,

$$\text{Heurstic}(\text{node}_\text{i}, \text{node}_\text{goal}) = c_{\text{RS}}(s^i, s_f) \tag{5}$$

where s_i is already solved in (4) since the heuristic can be always calculated after the actual cost is calculated.

Algorithm 1: Reference Line Guided Tree Search

Required:
The reference graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{(l_1, \pm), (l_2, \pm), ..., (l_n, \pm)\}$;
Start and goal states s_0, s_f;

1 Find start and goal node $\text{node}_\text{start}, \text{node}_\text{goal}$
2 Search shortest path using A* with cost (4) and heuristic (5)
3 Recover sub-goals on the shortest path using solutions of (4)

The graph search algorithm is summarized in Algorithm 1. The computational burden of this method clearly lies priority in computing the pair-wise minimal distance points between neighboring reference lines. We only consider the Reeds-Shepp cost on 2D space, so the computation burden is not significant.

IV. Reinforcement Learning-based Trajectory Optimization

In this section, we solve the *low-level trajectory optimization* using the primal-dual RL algorithm, which connects successive sub-goals generated by the high-level graph search described in Section III. We first introduce the RL problem formulation and training setup, then we demonstrate several practical tricks to handle issues such as constraint satisfaction, solution accuracy, supervised learning initialization, etc.

A. RL Problem Formulation

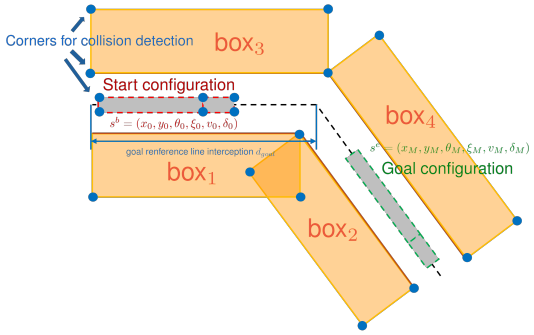


Fig. 4: Demonstration of RL-based trajectory optimization.

The reference line guided search outputs a high-level path $\{s^0, s^1, s^2, \ldots, s^K\}$ with $s^0 = s_i, s^K = s_f$, and the goal of the low-level task is to connect successive waypoint pairs $(s^0, s^1), (s^1, s^2), \ldots, (s^{K-1}, s^K)$. Figure 4 shows an example of i^{th} low-level task, the goal is connecting (s^{i-1}, s^i) within finite time steps while avoiding obstacles. We model the free space of i^{th} as two safety corridors centered around

[1]Reeds-Shepp cost means the minimal distances to turn a Reeds-Shepp car from initial state to goal state, where Reeds-Shepp car is a car that can move in both forward and reverse directions. Reeds and Shepp have proved that for arbitrary initial and goal states, one only needs to consider paths with at most 2 reversals [20].

the start reference line and the goal reference line, denoted by $\mathcal{S}^i_{free}$. We use (s^b, s^e) to denote the beginning and ending sub-goals in this section, differentiating them from the overall initial and goal states.

We consider MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, r, h, P, M, \rho^b, \rho^e \rangle$. The state space includes tractor-trailer states s_t and free space information. The inputs are acceleration $\dot{v}$ and steering rate $\dot{\delta}$ in (1). The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and collision function $h : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ design will be presented in the following section, where d is the constraint dimension. The transition dynamics P follows the vehicle transition dynamics (1), and M is the maximum number of time steps to reach the goal configuration. ρ^b, ρ^e is the initial state and terminal state distribution. Given the observation function $\boldsymbol{o}$ that maps state $s \in \mathcal{S}$ to observation space $o \in \mathcal{O}$ (construction explained later) and the policy $\pi : \mathcal{O} \to \Delta(\mathcal{A})$ to optimize, the optimization problem for RL is

$$\min_{\pi} J(\pi) := \mathbb{E}_{\substack{s_0 \sim \rho^b \\ s_t \sim P(\cdot|s_t, u_t) \\ u_t \sim \pi(\cdot|\boldsymbol{o}(s_t))}} \left[\sum_{t=1}^{M} r(s_t, u_t) \right] \tag{6a}$$

$$\text{s.t. } s_M = s^e \sim \rho^e \tag{6b}$$

$$s_t \in \mathcal{S}^i_{free}, |\xi_t| \leq \frac{\pi}{2}, \forall t = 1, \ldots, M \tag{6c}$$

where the dynamics feasibility and input constraints are implicitly embedded in dynamics P and policy π, we only need to consider the terminal state, obstacle avoidance, and jack-knife constraints explicitly.

B. Training Setup

We describe other detail training setup we need to consider **Map generation.** To learn a general RL controller that works for multiple sub-tasks, we need to generate many local maps. We first randomly sample reference line interception l_{goal} and relative angle of start and end reference line, then we randomly sample the initial and terminal states aligning with the two reference lines, and finally we sample safety corridor widths of the start and goal reference lines within a reasonable range.

Observation space. The observations include $\{s^b, s_t, s^e\} \cup \{p^j_i | j \in \{ll, ul, ur, lr\} i \in \{tractor, tailer, box_1, box_2, box_3, box_4\}\}$, where s^b is the initial states, s_t is the current states, s^e is the terminal states, and others are the free space information. p^j_i is four corner points ("ll" for lower left corner, "ul" for upper left corner, etc.) of the tractor, trailer, and four obstacles in the figure 4.

Reward and constraints design. Our primary goal is to reach goal configuration s_f with small admissible errors. Therefore, our reward function is rather simple,

$$r(s_t, u_t) = -\mathbf{1}(\|s_M - s^e\| \leq \epsilon) + 0.01\|u\|^2 \tag{7}$$

where ϵ is the admissible error. For the constraint design, every time step has safety constraints in (6b) and (6c). We reformulate it to trajectory-based reachability constraints following [21],

$$\max_{t \in \{1,2,\ldots,M\}} -d(s_t) \leq 0, \max_{t \in \{1,2,\ldots,M\}} |\xi_t| \leq \frac{\pi}{2} \tag{8}$$

where $d(s) : \mathcal{S} \to \mathbb{R}$ as the signed distance to all the collision boxes and $d(s) \geq 0$ indicates the tractor and trailer are not colliding with the four collision boxes. The calculation of distance is based on separating axis theorem since both the vehicles and obstacles are convex. For terminal constraints, we set admissible error to it and convert it to inequality constraints $\|s_M - s^e\| \leq \epsilon$. Combining these constraints, we reformulate (6) to the optimization algorithm we solve in practice,

$$\begin{aligned} &\min_{\pi} \mathbb{E}_{\substack{s_0 \sim \rho^b \\ s_t \sim P(\cdot|s_t, u_t) \\ u_t \sim \pi(\cdot|\boldsymbol{o}(s_t))}} \left[\sum_{t=1}^{M} r(s_t, u_t) \right] \\ &\text{s.t. } \|s_M - s^e\| \leq \epsilon \ \forall s^e \sim \rho^e \\ &\qquad \max_{t \in \{1,2,\ldots,M\}} -d(s_t) \leq 0, \max_{t \in \{1,2,\ldots,M\}} |\xi_t| \leq \frac{\pi}{2}, \forall s_0 \sim \rho^b \end{aligned} \tag{9}$$

C. Algorithm Design

It is natural to use primal-dual to solve constrained policy optimization (9). One issue is that every initial state has constraints, so we need to assign dual variables or multipliers for every initial state, which is similar to our previous work [22]. We denote the statewise multiplier mapping as $\lambda : \mathcal{S} \to \mathbb{R}^d_+$. The primal-dual optimization problem becomes

$$\begin{aligned} &\max_{\lambda} \min_{\pi} L(\pi, \lambda) := \\ &\mathbb{E}_{\substack{s_0 \sim \rho \\ s_t \sim P(\cdot|s_t, u_t) \\ u_t \sim \pi(\cdot|\boldsymbol{o}(s_t))}} \left[\sum_{t=1}^{M} r(s_t, u_t) \right] + \int_{s_0} \lambda^\top(s_0) h^\pi(s_0) ds_0 \end{aligned} \tag{10}$$

where $L(\pi, \lambda)$ is the Lagrangian and $h^\pi(s_0) = [\|s_M - s^e\| - \epsilon, \max_{t \in \{1,2,\ldots,M\}} -d(s_t), \max_{t \in \{1,2,\ldots,M\}} |\xi_t| - \frac{\pi}{2}]^\top$ are the constraints. In the practical algorithm, we use neural networks to parametrize policy π_ζ and dual variable λ_η with parameters ζ and η. We use uniform initial state distribution in a bounded support and approximate the integral term in (10) by sample mean $\mathbb{E}_{s_0 \sim \rho} \lambda(s_0)^\top h^\pi(s_0)$. This will cause the practical λ to scale up by a constant factor $1/\rho(s)$, which does not affect the optimal policy.

Model-based policy gradient. As we fully know the differentiable dynamics model f, reward function r, and every element in constraints h^π, we can directly calculate the model-based policy gradient $\frac{\partial L(\pi, \lambda)}{\partial \pi}$. Compared to the model-free policy gradient, the model-based policy gradient is more accurate and the training will be more stable.

Stochastic policy. We require accurate control to reach the goal state but RL policy has been shown to not be precise enough empirically. Here we show empirical tricks with stochastic policy to generate trajectories to precisely reach the goal position. We take use Gaussian random policy $u_t \sim \mathcal{N}(\pi_d(o_t), \sigma^2 I_2)$, where π_d is a deterministic mapping that maps from observations to the mean value and σ^2 is a constant variance. Then after training, we use the stochastic

policy to sample a batch of trajectories and select the one with the terminal state closest to the given goal state without causing collision.

Supervised learning initialization from state-lattice-based methods. The initialization is key to the performance of optimization problem. Therefore, we leverage the existing state-lattice-based methods, specially the improved A-search guided tree [23] to generate feasible trajectories and initialize the policy mean value with supervised learning.

The practical RL algorithm is summarized in Algorithm 2.

Algorithm 2: Primal-dual RL for trajectory optimization

Required: Distribution of successive sub-goals (ρ^b, ρ^e), map generators, policy noise σ, time steps M, reward function r, distance function d, initial policy π_{ζ_0}, initial multiplier λ_{η_0}, learning rate β_ζ, β_η.

```
# Pre-training via supervised learning from search-based methods.
```

Sample $s^b \sim \rho^b, s^e \sim \rho^e$

for *Each sampled* s_0 **do**

  Solve trajectories $\{s_0 = s^b, a_0, s_1, \ldots, s_M = s^e\}$ using [23] and save in dataset $\mathcal{D}$

for *Pre-training iteration* $k = 1, 2, \ldots, T_{pre}$ **do**

  Sample a minibatch of states and actions $\{s_i, a_i \mid, i = 1, 2, \ldots, B\}$

  Supervised learning on policy mean π_d via $\zeta_{i+1} = \zeta_i - \beta \frac{\partial J_{pre}(\pi)}{\partial \zeta}$, where $J_{pre}(\pi) = \frac{1}{B} \sum_{i=1}^{B} (\pi_d(s_i) - a_i)^2$

```
# Main Training stage.
```

for *Iteration* $k = 1, 2, \ldots, T$ **do**

  Randomly sample maps and initial states $s_0 \sim \rho$

  for *time step* $t = 0, 1, 2, \ldots, M-1$ **do**

    Rollout s_{t+1} using $a_t \sim \pi$ and dynamics (1)

    Calculate reward $r(s_t, a_t)$, distance $d(s_t)$ and relative angle ξ_t

  Calculate the constraints $h^\pi(s_0)$ from terminal state s_M and sequences of $d(s_t), \xi_t$

  Calculate the Lagrangian in (10)

  Update policy parameters ζ $\zeta_{T_{pre}+k} \leftarrow \zeta_{T_{pre}+k-1} - \beta \frac{\partial L(\pi, \lambda)}{\partial \zeta}$

  Update dual variable parameters ζ_0 $\eta_k \leftarrow \eta_{k-1} + \beta \frac{\partial L(\pi, \lambda)}{\partial \theta}$

```
# Online trajectory generation.
```

Use $\pi_{\zeta_{T_{pre}+T}}$ to rollout multiple trajectories and select the one whose terminal state are closest to goal.

V. Empirical Evaluation

In this section, we show the empirical results of our proposed hierarchical motion planning framework.

A. Experimental Setup

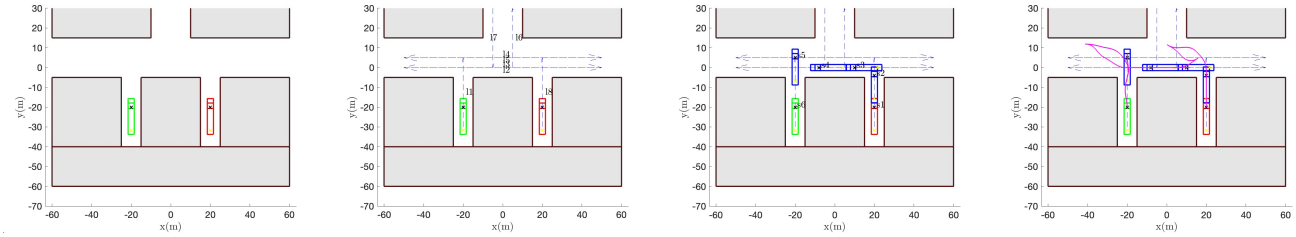
We select one representative cluttered environment shown in Figure 5(a), and generate 5 representative tasks with different initial and goal states: moving to other positions, parking, U-turn, and driving out.

Baselines and performance metrics. We compare our proposed algorithm, named **Ref-Guided RL** against two baseline motion planning algorithms. One is the plain improved A-search Guided Tree search (**plain i-AGT**) [23], where no high-level reference graph guided search is conducted. The initial state s_i and goal state s_f are directly input to the motion planner. To further verify the RL-based trajectory generalization, we have another baseline that uses the i-AGT motion planner to plan low-level tasks connecting sub-goals. The baseline is called **Ref-Guided i-AGT**. We evaluate the motion planning performance by the planning time, terminal position and angle errors and planned path length. The computation time is counted on an Apple Macbook Air with the Apple M3 processor. When solving the RL problem, we decide the admissible terminal position error in the practical RL problem formulation (9) as $0.2m$ and terminal angle error as $1^\circ \approx 0.017rad$.

B. Experimental Results

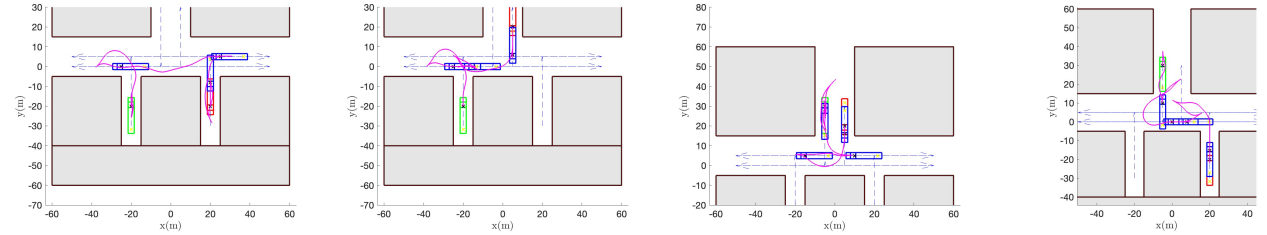
First, we visualize the proposed hierarchical motion planning pipelines in Figure 5 using our first task, moving the tractor-trailier from one garage to another one. Then we construct reference lines, shown in Figure 5(b). Then we conducted the reference line guided search, and the search results is $(l_8, +) \rightarrow (l_2, -) \rightarrow (l_1, +)$, where $+, -$ matches the direction of x, y axes since the reference line segments are all parallel with the axes. The reference line guided search also outputs the sub-goals shown in Figure 5(c), where every pair of successive sub-goals are connected with only one turn or straight line. Figure 5(d) shows the trajectory optimization results solved by the primal-dual RL algorithm. Note that the graph search only considered collision-free Reeds-Shepp cost, where the single-turn or single-straight transition might not be possible when then tailor and collision avoidance are considered.

Additionally, we considered four more cases, including moving to other positions while changing direction, parking, U-turn, and driving out and the planned trajectories of tractor are shown in Figure 6. The performance metrics of all 5 cases are compared in Table I. We can see that the proposed **Ref-Guided RL** achieves significantly lower planning time compared to other i-AGT-based baselines. The low-planning time only includes neural network forward pass and accuracy comparison in Algorithm 2. The planned trajectory length is also shorter than the Ref-guided i-AGT planner. However, there is no free lunch and the price to pay is the planning accuracy. As shown in Table I, the position error of RL are only around 0.2m and 0.017rad (1 degree), which is exactly the terminal position and angle error we set. Therefore, the proposed motion planner actually achieves target planning accuracy. For the baseline i-AGT without reference line search, for some tasks like case 3 and 5 the search is very



(a) Case 1: moving to other position. (b) Step 1: Select reference lines. (c) Step 2: Searching for sub-goals. (d) Step 3: Solving trajectories.

Fig. 5: Step-by-step visualizations of the proposed motion planning algorithm. The initial position of the trailer is colored in red and the goal position is colored in green. The tractor-trailer sub-goals are colored blue. The magenta line indicates the trajectory of the tractor.



(a) Case 2: move to other position 2 (b) Case 3: parking. (c) Case 4: U-turn. (d) Case 5: Driving out.

Fig. 6: Visualization of planned trajectories of the tractor.

TABLE I: Comparison path-planning performance between reference-guided RL planner, reference-guided i-AGT planner, and plain i-AGT planner

Case No.	Ref-Guided RL				Ref-Guided i-AGT				Plain i-AGT			
	Planning Time [s]	Terminal position error [m]	Terminal angle error [rad]	Path length [m]	Planning Time [s]	Terminal position error [m]	Terminal angle error [rad]	Path length [m]	Planning Time [s]	Terminal position error [m]	Terminal angle error [rad]	Path length [m]
1	0.315	0.241	0.014	212.36	14.441	0.000	0.000	212.37	83.264	0.002	0.003	153.38
2	0.342	0.202	0.025	221.71	20.284	0.005	0.004	321.83	146.29	0.002	0.003	146.01
3	0.228	0.193	0.023	174.67	25.678	0.009	0.002	310.96	9.985	0.004	0.002	147.84
4	0.226	0.211	0.028	189.52	27.705	0.001	0.003	310.96	N/A*	N/A	N/A	N/A
5	0.271	0.241	0.017	197.77	27.901	0.005	0.008	237.41	5.950	0.000	0.001	100.547

* Solver failed or time out. The maximum computation time is 180 seconds.

fast. However, for other task the search time is extremely long and it even fails case 4.

VI. Conclusion and Future Work

In this paper, we proposed a hierarchical motion planning framework that combines high-level graph search and low-level RL-based trajectory optimization to perform long-horizon planning for tractor-trailer systems. We first search for a series of sub-goals by constructing reference line graph and searching for the shortest path. Then we design a primal-dual RL algorithm to generate dynamically feasible collision-free trajectories connecting successive reference lines. Empirical results have shown significant improvements on the planning time with satisfactory accuracy.

There are several places to improve in this work, for example, improving heuristic and cost design in graph search, designing automatic reference line graph construction algorithm, etc.

References

[1] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[2] X. Zhang, A. Liniger, A. Sakai, and F. Borrelli, "Autonomous parking using optimization-based collision avoidance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 4327–4332.

[3] K. Bergman and D. Axehill, "Combining homotopy methods and numerical optimal control to solve motion planning problems," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 347–354.

[4] J. Leu, G. Zhang, L. Sun, and M. Tomizuka, "Efficient robot motion planning via sampling and optimization," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 4196–4202.

[5] J.-W. Choi and K. Huhtala, "Constrained global path optimization for articulated steering vehicles," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 1868–1879, 2015.

[6] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer, "Lattice-based motion planning for a general 2-trailer system," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 819–824.

[7] M. Cirillo, T. Uras, and S. Koenig, "A lattice-based approach to multi-robot motion planning for non-holonomic vehicles," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 232–239.

[8] M. Cirillo, "From videogames to autonomous trucks: A new algorithm for lattice-based motion planning," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 148–153.

[9] 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.

[10] F. Islam, V. Narayanan, and M. Likhachev, "A*-connect: Bounded suboptimal bidirectional heuristic search," in *2016 IEEE International Conference On Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2752–2758.

[11] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[12] O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo, and H. Pettersson, "A path planning and path-following control framework for a general 2-trailer with a car-like tractor," *Journal of field robotics*, vol. 36, no. 8, pp. 1345–1377, 2019.

[13] K. Bergman, O. Ljungqvist, and D. Axehill, "Improved path planning by tightly combining lattice-based path planning and optimal control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 1, pp. 57–66, 2020.

[14] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *Advances in neural information processing systems*, vol. 29, 2016.

[15] B. Chen, B. Dai, Q. Lin, G. Ye, H. Liu, and L. Song, "Learning to plan in high dimensions via neural exploration-exploitation trees," *arXiv preprint arXiv:1903.00070*, 2019.

[16] A. Deac, P.-L. Bacon, and J. Tang, "Graph neural induction of value iteration," Sep. 2020.

[17] C. Yu and S. Gao, "Reducing Collision Checking for Sampling-Based Motion Planning Using Graph Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 4274–4289.

[18] P. Rouchon, M. Fliess, J. Lévine, and P. Martin, "Flatness and motion planning: the car with n trailers," in *Proc. ECC'93, Groningen*, 1993, pp. 1518–1522.

[19] C. Altafini, A. Speranzon, and B. Wahlberg, "A feedback control scheme for reversing a truck and trailer vehicle," *IEEE Transactions on robotics and automation*, vol. 17, no. 6, pp. 915–922, 2001.

[20] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[21] D. Yu, H. Ma, S. Li, and J. Chen, "Reachability constrained reinforcement learning," in *International conference on machine learning*. PMLR, 2022, pp. 25 636–25 655.

[22] H. Ma, Y. Guan, S. E. Li, X. Zhang, S. Zheng, and J. Chen, "Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety," *arXiv preprint arXiv:2105.10682*, 2021.

[23] J. Leu, Y. Wang, M. Tomizuka, and S. Di Cairano, "Improved a-search guided tree for autonomous trailer planning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7190–7196.