MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Fast Adaptive Planning for Autonomous Tractor-Trailer Systems via Iterative LQR Steering

Zhou, Tianyu; Wang, Yebin; Umat, Akhil; Tiwari, Astha; Di Cairano, Stefano

TR2026-118 September 01, 2026

Abstract

Aiming to tackle the computation challenges arising from the stringent positioning accuracy requirement as well as the uncertainties in system dynamics, this paper presents a fast adaptive planning method for tractor-trailer systems. Building on the delayed-expansion A-search guided tree (DEAGT) that grows a tree from pre-computed motion primitives (MPs) via A*-like search, Adaptive-iAGT contributes: (i) an iterative Linear-Quadratic Regulator (iLQR) steering module that reaches the goal without a reference trajectory, reducing the planning time; (ii) a two-stage framework that resolves parameter uncertainties by adapting a nominal trajectory to true parameters via iLQR, eliminating the need to store MPs for various configurations; and (iii) analytical control scaling that lets iLQR ignore velocity/control bounds and reinstates them afterwards, increasing the success rate. Simulations and field tests validate the effectiveness of the proposed algorithm.

IEEE Conference on Automation and Science Engineering 2026

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Fast Adaptive Planning for Autonomous Tractor-Trailer Systems via Iterative LQR Steering

Tianyu Zhou, Yebin Wang, Akhil Umat, Astha Tiwari, and Stefano Di Cairano

***Abstract*— Aiming to tackle the computation challenges arising from the stringent positioning accuracy requirement as well as the uncertainties in system dynamics, this paper presents a fast adaptive planning method for tractor-trailer systems. Building on the delayed-expansion A-search guided tree (DE-AGT) that grows a tree from pre-computed motion primitives (MPs) via A*-like search, Adaptive-iAGT contributes: (i) an iterative Linear-Quadratic Regulator (iLQR) steering module that reaches the goal without a reference trajectory, reducing the planning time; (ii) a two-stage framework that resolves parameter uncertainties by adapting a nominal trajectory to true parameters via iLQR, eliminating the need to store MPs for various configurations; and (iii) analytical control scaling that lets iLQR ignore velocity/control bounds and reinstates them afterwards, increasing the success rate. Simulations and field tests validate the effectiveness of the proposed algorithm.**

I. Introduction

Autonomous tractor-trailer systems have garnered significant interest from both industry and academia due to their high cargo transportation efficiency. Their complex dynamics and model mismatches present substantial challenges in planning and control, especially during reversing maneuvers, which are inherently unstable. A notable example is docking, where maneuvering into a narrow space often requires multiple attempts, even for trained professionals. Efficient and adaptive planning is crucial to address challenges accompanied by demanding tasks such as docking and parking.

Existing kinodynamic planning methods are generally categorized into four approaches: optimization-based, sampling-based, learning-based, and state-lattice. Optimization-based methods solve optimal control problems (OCPs) by carefully formulating obstacle constraints and requiring a good initial guess. They may converge to local minima [1]–[5] and struggle with scalability [1], [6], especially when obstacle constraints are represented using mixed-integer programs. Sampling-based methods such as Rapidly-exploring Random Trees (RRT) [7]–[9] and Stable Sparse RRT (SST) [10] explore collision-free space using random samples with steering functions or simulation to ensure dynamic feasibility. Steering solves challenging two-point boundary value problems (TPBVPs) for nonlinear systems. SST avoids TPBVPs via random control sampling, but suffers from slow convergence and non-exact goal reaching.

Learning-based approaches have been explored to reduce the computational burden of kinodynamic planning. Semi-supervised learning generates feasible tractor–trailer paths without expert data [11], neural predictors improve tree expansion by estimating transition costs [12], RL-based planners replace analytic steering functions with learned policies [13], and neural feedback controllers approximate steering functions in RRT using offline trajectories [14]. These methods enhance efficiency and goal-reaching but remain limited by heavy offline computation, generalization issues, and lack of safety guarantees. State-lattice methods discretize the state space into a structured grid [15]–[17], with motion primitives (MPs) pre-computed offline by solving OCPs for dynamic feasibility. The resulting lattice graph enables efficient online search with A* [18], and hybrid variants combine RRT* with lattice sampling [19]. These methods face challenges of dimensionality and resolution completeness [20], though several tractor–trailer planners mitigate these issues using dimensionality reduction [21]–[23].

Works [24], [25] introduce an improved A-search guided tree (i-AGT) algorithm for tractor-trailer systems, extending A* by incorporating mode-based node expansion. MPs from the most promising mode are selectively applied during node expansion. DE-AGT [26] further advances this framework by leveraging a neural network-based cost-to-go estimation as well as integrating LQR-based steering to enhance goal-reaching likelihood. While these advancements improve the efficiency and scalability of tractor-trailer motion planning, several challenges remain. First, LQR steering relies on a reference trajectory and is activated only if the tree gets close to the goal state. This can lead to an excessively large search tree, reducing overall efficiency. Second, tractor-trailer systems have different model parameters. Applying nominal MPs across different systems can result in significant deviation of the trailer's state and collisions, while pre-generating and storing MPs for various model parameter configurations is memory-inefficient. Third, applicability to varying velocity and control constraints is limited, reducing adaptability in real-world scenarios.

This paper introduces the Adaptive-iAGT algorithm to address these challenges, Compared to [26], we incorporate an iterative LQR (iLQR) function to facilitate the direct connection of distant nodes with the goal, without a reference trajectory and with relaxed constraints [27]. Secondly, instead of pre-computing MPs for every possible model parameter configuration, we propose a two-stage framework

T. Zhou is with the School of Aeronautics and Astronautics, Purdue University, IN. This work was done while T. Zhou was a research intern at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA. (email: zhou1043@purdue.edu)

A. Umat and A. Tiwari are with Mitsubishi Electric Automotive America, Northville, MI. (email: {aumat,atiwari}@meaa.mea.com)

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

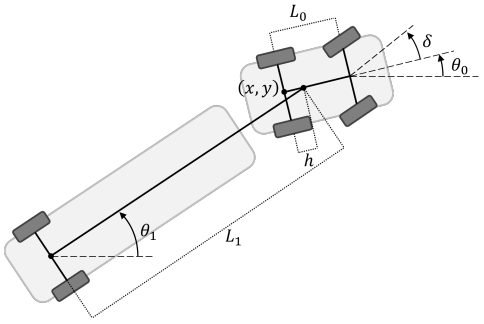


Fig. 1: Kinematics of a front wheel drive tractor with one trailer. All angles representing the orientations (θ_0, θ_1) are w.r.t. the x-axis.

to handle different model parameters: first, a path planner generates a nominal trajectory based on the nominal parameters; and second, a motion planner adapts the nominal trajectory using an iLQR-based tracking method, making the planning process more flexible and realistic. Finally, we adopt an analytical scaling method to adapt control inputs according to customized velocity and control constraints while following the path. The scaling allows us to omit these constraints when trying the iLQR, and thus significantly reduces computation time. We present simulations and field tests to demonstrate the effectiveness and performance of the Adaptive-iAGT algorithm in real-world applications.

II. Preliminaries

A. Tractor-Trailer System Modeling

Consider the front-wheel-drive tractor-trailer system with non-zero hitching offset [28], [29] shown in Fig. 1: (x, y) is the tractor rear-axle midpoint, θ_0, θ_1 are the tractor/trailer orientations, L_0, L_1 are the tractor/trailer wheelbases, h is the hitching offset, v_f is the front-wheel speed, and δ is the steering angle. With v_f and δ independently actuated, define heading-frame speed $v = \cos(\delta)v_f$ and control $\boldsymbol{u} = [a, \dot{\delta}]^\top$ with $a = \dot{v}$. Let $\theta_r = \theta_1 - \theta_0$ represent the relative angle between the tractor and trailer. The kinematic model is:

$$\begin{aligned} &\dot{x} = \cos(\theta_0)v,\ \dot{y} = \sin(\theta_0)v,\ \dot{\theta}_0 = \frac{v\tan(\delta)}{L_0}, \\ &\dot{\theta}_r = v\frac{h\tan(\delta)\cos(\theta_r) - L_0\sin(\theta_r) - L_1\tan(\delta)}{L_0 L_1}, \\ &\dot{v} = u_1,\ \dot{\delta} = u_2. \end{aligned} \tag{1}$$

The constraint to prevent a jack-knife configuration is $|\theta_r| \leq \theta_{r,\max} \leq \frac{\pi}{2}$. The tractor's heading needs to satisfy $0 \leq \theta_0 < 2\pi$. The system is subject to the following constraints:

$$|\delta| \leq \delta_{\max},\ |\theta_r| \leq \theta_{r,\max}, \tag{2a}$$

$$|a| \leq a_{\max},\ |\dot{\delta}| \leq \dot{\delta}_{\max},\ v_{\min} \leq v \leq v_{\max}, \tag{2b}$$

where $v_{\min} < 0, v_{\max}, \delta_{\max}, a_{\max}, \dot{\delta}_{\max}$ are scalars.

Notations. Denote $\boldsymbol{x} = [x, y, \theta_0, \theta_r, v, \delta]^\top \in \mathcal{X}$ and $d(\boldsymbol{x}_1, \boldsymbol{x}_2)$ as the element-wise absolute difference between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Define a neighborhood box $\mathcal{B}_\epsilon(\boldsymbol{x}) \triangleq \{\boldsymbol{x}_i | d(\boldsymbol{x}, \boldsymbol{x}_i) \leq \boldsymbol{\epsilon}, \forall \boldsymbol{x}_i \in \mathcal{X}\}$. An MP, denoted by $mp \triangleq \{(\boldsymbol{x}_0, ..., \boldsymbol{x}_N), (\boldsymbol{u}_0, ..., \boldsymbol{u}_{N-1}), \Delta t\}$, consists of states and controls at $N+1$ and N time instants with a timestep Δt. Applying the MP at state $\boldsymbol{x}$ results in a trajectory $\boldsymbol{\xi}$.

B. Baseline Planning Work

The free space in which the system state lies in, denoted as $\mathcal{X}_{\text{free}} \in \mathcal{X}$, consists of all states $\boldsymbol{x}$ where the system (1) remains collision-free within the environment. To check for collisions, we simply wrap around the tractor, trailer, and obstacles with bounding boxes (rectangles). Baseline work solves the following planning problem:

Problem 2.1 (Baseline Planning Problem): Given an initial state $\boldsymbol{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$, a goal state $\boldsymbol{x}_{\text{goal}} \in \mathcal{X}_{\text{free}}$ and system (1) with fixed constraints (2), find a collision-free trajectory $\mathcal{P}$ which starts at $\boldsymbol{x}_{\text{init}}$ and ends at $\boldsymbol{x}_{\text{goal}}$.

Existing solutions to Problem 2.1 are predominantly based on constructing a graph or tree (connecting $\boldsymbol{x}_{\text{init}}$ to $\boldsymbol{x}_{\text{goal}}$) by repetitively applying MPs on a selected node, where the node selection follows well-known A* algorithm or its variants [22], [24]–[26]. MPs are pre-generated by sampling initial-goal state pairs and solving the corresponding OCPs [25].

C. Adaptive Planning Problem

Real-world tractor-trailer system presents various characteristics which complicate the development of realistic planning solutions. This work particularly address the challenges arising from two aspects: the true values of model parameters in (1), denoted by $\Theta = [L_0, h, L_1]^\top$, might deviate from the nominal values used in MPs generation, denoted by Θ^*; and the bounds of velocity/control constraints might be different owing to loading conditions. This work tackles the following adaptive planning problem:

Problem 2.2 (Adaptive Planning Problem): Given an initial state $\boldsymbol{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$, a goal state $\boldsymbol{x}_{\text{goal}} \in \mathcal{X}_{\text{free}}$, true parameters $\Theta \in [\underline{\Theta}, \overline{\Theta}]$, system (1), and constraints (2) with the bounds of velocity and control constraints given by $(v_{\min}, v_{\max}, a_{\max}, \dot{\delta}_{\max})$, find a collision-free trajectory $\mathcal{P}$ which starts at $\boldsymbol{x}_{\text{init}}$ and ends at $\boldsymbol{x}_{\text{goal}}$.

Remark 2.3: Established planning works typically assume the exact knowledge of system model and constraints, leaving model mismatches to be handled by underlying controllers. This treatment is not realistic for navigation tasks with high positioning accuracy because the mismatches render collision or large deviations during trajectory tracking.

Remark 2.4: Solving the adaptive planning problem as a constrained optimal control problem (e.g., via CasADi/Ipopt) entails nonlinear programming with nonconvex collision-avoidance constraints, resulting in high per-instance solve times (7s to adapt one MP in our tests), which is incompatible with real-time operation. In contrast, the proposed algorithm achieves two orders of magnitude faster runtime (0.06s) while maintaining feasibility in practice.

III. Main Algorithm

This section presents Adaptive-iAGT algorithm to solve Problem 2.2. As shown in Fig. 2, Adaptive-iAGT consists of two modules: a relaxed path planner and an adaptive motion planner. Specifically, the relaxed path planner generates a nominal trajectory based on a set of nominal parameters Θ^* and constraints (2a), and adaptive motion planner tailors the nominal trajectory to true parameters Θ and constraints (2b).

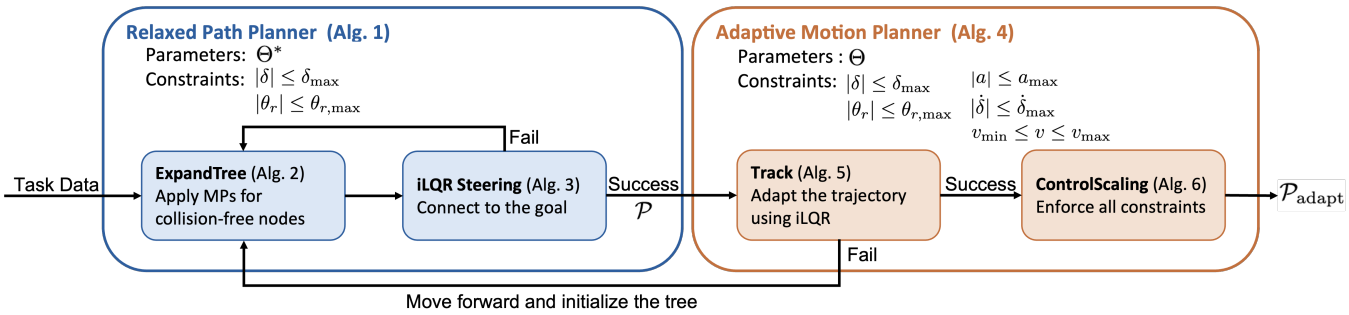


Fig. 2: Adaptive-iAGT Framework.

A. Relaxed Path Planner

Algorithm 1: RelaxedPathPlanner($\boldsymbol{x}_{\text{init}}, \boldsymbol{x}_{\text{goal}}, \Theta^*$)

$\mathcal{T} \leftarrow (\boldsymbol{x}_{\text{init}}, \emptyset)$; $\mathcal{Q} \leftarrow \{\boldsymbol{x}_{\text{init}}\}$; flag $\leftarrow$ **False**;
while CPU time $\le T_{\max}$ **and** $\neg$flag **do**
  $\boldsymbol{x}_{\text{select}} \leftarrow \mathcal{Q}.\text{pop}$;
  $(\mathcal{T}, \mathcal{Q}, \text{flag}) \leftarrow \text{ExpandTree}(\mathcal{T}, \mathcal{Q}, \boldsymbol{x}_{\text{select}}, \Theta^*)$;
$\mathcal{P} \leftarrow \text{GetTrajectory}(\mathcal{T})$;
return $\mathcal{P}$, flag;

Alg. 1 sketches the pseudo-code of relaxed path planner. It expands the tree from $\boldsymbol{x}_{\text{init}}$ toward $\boldsymbol{x}_{\text{goal}}$ until succeeds or the allocated CPU time is reached. GetTrajectory($\mathcal{T}$) retrieves planned trajectories $\mathcal{P} = \{\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, ...\}$ from the tree.

Algorithm 2: ExpandTree($\mathcal{T}, \mathcal{Q}, \boldsymbol{x}_{\text{select}}, \Theta^*$)

$\mathcal{MP}_{\text{best}} \leftarrow \text{BestMode}(\boldsymbol{x}_{\text{select}})$, flag $\leftarrow$ **False**;
foreach $mp \in \mathcal{MP}_{\text{best}}$ **do**
  $(\boldsymbol{x}_{\text{next}}, \boldsymbol{\xi}) \leftarrow \text{Expand}(\boldsymbol{x}_{\text{select}}, mp)$;
  if CollisionFree($\boldsymbol{\xi}$) **then**
    AddNode($\boldsymbol{x}_{\text{next}}, \boldsymbol{\xi}$);
    if TryiLQRSteering($\boldsymbol{x}_{\text{next}}, \boldsymbol{x}_{\text{goal}}$) **then**
      $\boldsymbol{\xi} \leftarrow \text{ApplyZeroControl}(\boldsymbol{x}_{\text{next}})$;
      $(\boldsymbol{\xi}, \boldsymbol{x}_{\text{new}}) \leftarrow$ iLQRSteering($\boldsymbol{x}_{\text{next}}, \boldsymbol{x}_{\text{goal}}, \boldsymbol{\xi}, \Theta^*$);
      if Reach($\boldsymbol{\xi}, \boldsymbol{x}_{\text{goal}}$) **then**
        AddNode($\boldsymbol{x}_{\text{next}}, \boldsymbol{\xi}$), flag $\leftarrow$ **True**;
        break;
return $\mathcal{T}, \mathcal{Q}$, flag;

Alg. 2 expands the tree by applying MPs at the selected state $\boldsymbol{x}_{\text{select}}$ and uses iLQR to establish a connection between the goal state $\boldsymbol{x}_{\text{goal}}$ and the tree. Readers are refered to [26] for the details of finding best-mode MP set. Each collision-free expansion is committed by AddNode, which adds $\boldsymbol{x}_{\text{next}}$ and $\boldsymbol{\xi}$ to $\mathcal{T}$ and pushes the $\boldsymbol{x}_{\text{next}}$ onto $\mathcal{Q}$. TryiLQRSteering determines whether iLQR is worth trying to connect $\boldsymbol{x}_{\text{next}}$ to $\boldsymbol{x}_{\text{goal}}$. It checks if a virtual tractor (length = tractor + trailer) can reach $\boldsymbol{x}_{\text{goal}}$ from $\boldsymbol{x}_{\text{next}}$ via a cusp-free CSC Reeds-Shepp path [30], whose existence implies high iLQR success likelihood. If yes, ApplyZeroControl($\boldsymbol{x}_{\text{next}}$) generates a trajectory $\boldsymbol{\xi}$ starting from state $\boldsymbol{x}_{\text{next}}$ with zero control inputs. iLQRSteering($\boldsymbol{x}_{\text{next}}, \boldsymbol{x}_{\text{goal}}, \boldsymbol{\xi}, \Theta^*$) generates a trajectory ending at state $\boldsymbol{x}_{\text{new}}$ by applying Alg. 3 from $\boldsymbol{x}_{\text{next}}$ to $\boldsymbol{x}_{\text{goal}}$, using $\boldsymbol{\xi}$ as the reference and considering nominal parameters Θ^*. Reach($\boldsymbol{\xi}, \boldsymbol{x}_{\text{goal}}$) verifies if $\boldsymbol{\xi}$ successfully reaches $\boldsymbol{x}_{\text{goal}}$ by checking three criteria: 1) $\boldsymbol{\xi}$ is collision-free, 2) $\boldsymbol{\xi}$ adheres to steering and orientation constraints in (2a), and 3) $\boldsymbol{x}_{\text{new}} \in \mathcal{B}_\epsilon(\boldsymbol{x}_{\text{goal}})$.

Algorithm 3: iLQR($\boldsymbol{x}_0, \boldsymbol{x}_N^*, \boldsymbol{\xi}, \Theta$) steering/tracking

$(\boldsymbol{x}'_0, ..., \boldsymbol{x}'_N), (\boldsymbol{u}'_0, ..., \boldsymbol{u}'_{N-1}), \Delta t \leftarrow \text{Extract}(\boldsymbol{\xi})$;
for $i = 1, ..., i_{\max}$ **do**
  $S_N, v_N \leftarrow (10)$;
  for $t = N-1, ..., 0$ **do**
    $K_t, S_t, v_t, K_{v,t}, K_{u,t} \leftarrow (9)$
  for $t = 0, ..., N-1$ **do**
    $\boldsymbol{u}_t \leftarrow (8)$, $\boldsymbol{u}'_t \leftarrow \boldsymbol{u}_t$;
    $\boldsymbol{x}_{t+1} \leftarrow (4)$, $\boldsymbol{x}'_{t+1} \leftarrow \boldsymbol{x}_{t+1}$;
$\boldsymbol{\xi} \leftarrow \{(\boldsymbol{x}_0, ..., \boldsymbol{x}_N), (\boldsymbol{u}_0, ..., \boldsymbol{u}_{N-1}), \Delta t\}$;
return $\boldsymbol{\xi}, \boldsymbol{x}_N$;

Alg. 3 generates a trajectory from $\boldsymbol{x}_0$ to $\boldsymbol{x}_N^*$ using reference $\boldsymbol{\xi}$. Details of iLQR are provided in Section III-C.

B. Adaptive Motion Planner

AdaptiveMotionPlanner($\mathcal{P}, \Theta, \Theta^*$), as shown in Alg. 4, adapts a nominal trajectory $\mathcal{P}$ to the true model parameters Θ and enforces all constraints in (2). If $\Theta = \Theta^*$, no adaptation is needed and $\mathcal{P}$ passes straight to control scaling. Otherwise, TrackAll applies Track in Alg. 5 to adapt every segment of $\mathcal{P}$ by iLQR tracking. If the resulting trajectory $\mathcal{P}$ reaches $\boldsymbol{x}_{\text{goal}}$, the algorithm proceeds to control scaling; otherwise, RelaxedPathPlanner is to be re-initiated. Starting from $\boldsymbol{x}_{\text{init}}$, Forward($\boldsymbol{x}_{\text{init}}, \Theta, i$) generates a forward trajectory $\boldsymbol{\xi}_{\text{forward}}$ with zero steering angle and reaches $\boldsymbol{x}_{\text{forward}}$, where the distance of the forward trajectory increases with i. The forward motion can facilitate planning success due to two key reasons: 1) the relative angle θ_r reduces naturally due to the stability of forward motion; 2) the adapted trajectory is likely to collide with the narrow space during turning, whereas moving the vehicle straight forward reduces this risk. A new nominal trajectory is generated for adaption. This process repeats until either the iteration reaches the max number or the adaptation succeeds. The aforementioned

Algorithm 4: AdaptiveMotionPlanner($\mathcal{P}, \Theta, \Theta^*$)

```
if Θ = Θ* then
    P_adapt ← ControlScaling(P);
    return P_adapt, True;
for i = 1, ..., MaxIteration do
    if i > 1 then
        (ξ_forward, x_forward) ← Forward(x_init, Θ, i);
        (P, success) ← RelaxedPathPlanner(x_forward, x_goal, Θ*);
        P ← ξ_forward ∪ P;
        if ¬success then
            continue;
    P ← TrackAll(P, x_init, Θ);
    if Reach(P, x_goal) then
        P_adapt ← ControlScaling(P) ;
        return P_adapt, True;
return ControlScaling(P), False;
```

adaptation only enforces hard constraints in (2a) to improve success rate and computation efficiency. As a result, the generated trajectory $\mathcal{P}$ might still violate the constraints in (2b). ControlScaling($\mathcal{P}$) adapts the control inputs for each $\boldsymbol{\xi} \in \mathcal{P}$, ensuring compliance with (2b) while maintaining the original path.

Algorithm 5: Track($\boldsymbol{\xi}, \boldsymbol{x}_{\text{start}}, \Theta$)

```
x*_N ← LastState(ξ);
(ξ_ref, x_ref) ← ApplyControl(ξ, x_start, Θ);
foreach mode ∈ {Tracking, Steering} do
    (ξ_adapt, x_new) ← iLQR_mode(x_start, x*_N, ξ_ref, Θ);
    if Reach(ξ_adapt, x*_N) then
        return ξ_adapt;
return ξ_ref;
```

Alg. 5 adapts each $\boldsymbol{\xi} \in \mathcal{P}$. LastState($\boldsymbol{\xi}$) retrieves the desired state $\boldsymbol{x}_N^*$, while ApplyControl($\boldsymbol{\xi}, \boldsymbol{x}_{\text{start}}, \Theta$) generates a trajectory $\boldsymbol{\xi}_{\text{ref}}$ by applying $\boldsymbol{\xi}$'s control sequence from $\boldsymbol{x}_{\text{start}}$, considering true parameters Θ. Due to the variation in model parameters, $\boldsymbol{\xi}_{\text{ref}}$ may not reach $\boldsymbol{x}_N^*$. iLQR is then tried in two modes: tracking and steering, differing in whether the reference trajectory is used. If both fail, the trajectory with original controls is returned, and any offset is addressed during the tracking of the next trajectory segment.

Algorithm 6: ControlScaling($\boldsymbol{\xi}$)

```
(x_0, ..., x_N), (u_0, ..., u_{N-1}), Δt ← Extract(ξ);
k ← (3); Δt̄ ← kΔt;
for t = 0, ..., N − 1 do
    ū_t ← (4);
ξ̄ ← {(x_0, ..., x_N), (ū_0, ..., ū_{N-1}), Δt̄};
return ξ̄;
```

Alg. 6 scales control inputs to enforce constraints (2b). It first determines the scaling factor k and adjusted timestep $\Delta\bar{t}$, and then computes the scaled controls sequentially. Further details are provided in Section III-D.

C. iLQR for Steering and Tracking

We discretize system (1) using the 4^{th}-order Runge-Kutta method at time-step Δt:

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_d(\boldsymbol{x}_t, \boldsymbol{u}_t). \tag{4}$$

The goal is to find a trajectory from $\boldsymbol{x}_0$ to $\boldsymbol{x}_N^*$, minimizing

$$\begin{aligned} J = &\frac{1}{2}\sum_{t=0}^{N-1}\{(\boldsymbol{x}_t - \boldsymbol{x}_t')^\top \boldsymbol{Q}(\boldsymbol{x}_t - \boldsymbol{x}_t') + \boldsymbol{u}_t^\top \boldsymbol{R}\boldsymbol{u}_t\} \\ &+ \frac{1}{2}(\boldsymbol{x}_N - \boldsymbol{x}_N^*)^\top \boldsymbol{Q}_f(\boldsymbol{x}_N - \boldsymbol{x}_N^*), \end{aligned} \tag{5}$$

where $\boldsymbol{x}_t'$, $\boldsymbol{u}_t'$ are the reference state and control, $\boldsymbol{Q}, \boldsymbol{Q}_f \succeq 0$, and $\boldsymbol{R} \succ 0$. Following standard iLQR [27], we linearize (4) around the reference and solve the resulting LQR problem. The forward pass computes, for $t \in [0, N-1]$:

$$\begin{aligned} \delta\boldsymbol{x}_t &= \boldsymbol{x}_t - \boldsymbol{x}_t', \\ \delta\boldsymbol{u}_t &= -\boldsymbol{K}_t\delta\boldsymbol{x}_t - \boldsymbol{K}_{v,t}\boldsymbol{v}_{t+1} - \boldsymbol{K}_{u,t}\boldsymbol{u}_t', \\ \boldsymbol{u}_t &= \boldsymbol{u}_t' + \delta\boldsymbol{u}_t, \end{aligned} \tag{8}$$

where gains are obtained via backward Riccati recursion:

$$\begin{aligned} \boldsymbol{K}_t &= (\boldsymbol{B}_t^\top \boldsymbol{S}_{t+1}\boldsymbol{B}_t + \boldsymbol{R})^{-1}\boldsymbol{B}_t^\top \boldsymbol{S}_{t+1}\boldsymbol{A}_t, \\ \boldsymbol{K}_{v,t} &= (\boldsymbol{B}_t^\top \boldsymbol{S}_{t+1}\boldsymbol{B}_t + \boldsymbol{R})^{-1}\boldsymbol{B}_t^\top, \\ \boldsymbol{K}_{u,t} &= (\boldsymbol{B}_t^\top \boldsymbol{S}_{t+1}\boldsymbol{B}_t + \boldsymbol{R})^{-1}\boldsymbol{R}, \\ \boldsymbol{S}_t &= \boldsymbol{A}_t^\top \boldsymbol{S}_{t+1}(\boldsymbol{A}_t - \boldsymbol{B}_t\boldsymbol{K}_t) + \boldsymbol{Q}, \\ \boldsymbol{v}_t &= (\boldsymbol{A}_t - \boldsymbol{B}_t\boldsymbol{K}_t)^\top \boldsymbol{v}_{t+1} - \boldsymbol{K}_t^\top \boldsymbol{R}\boldsymbol{u}_t', \end{aligned} \tag{9}$$

with $\boldsymbol{A}_t, \boldsymbol{B}_t$ the Jacobians of $\boldsymbol{f}_d$ at $(\boldsymbol{x}_t', \boldsymbol{u}_t')$, and boundary conditions

$$\boldsymbol{S}_N = \boldsymbol{Q}_f, \quad \boldsymbol{v}_N = \boldsymbol{Q}_f(\boldsymbol{x}_N' - \boldsymbol{x}_N^*). \tag{10}$$

The updated trajectory serves as reference for the next iteration (Alg. 3). For *steering*, $\boldsymbol{Q} = \boldsymbol{0}_{6\times 6}$ so only the terminal cost drives the system to $\boldsymbol{x}_N^*$; for *tracking*, non-zero $\boldsymbol{Q}$ prioritizes following the reference.

D. Control Scaling

We briefly describe the method for scaling the control inputs of a trajectory to satisfy constraints (2b) while preserving the original path. Let $\boldsymbol{v}, \boldsymbol{a}, \dot{\boldsymbol{\delta}}$ denote the velocity, acceleration, and steering-rate sequences of trajectory $\boldsymbol{\xi}$. A constant factor k is determined as

$$k = \max\left(\frac{\max(\boldsymbol{v})}{v_{\max}}, \frac{\min(\boldsymbol{v})}{v_{\min}}, \sqrt{\frac{\max(\boldsymbol{a})}{a_{\max}}}, \frac{\max(\dot{\boldsymbol{\delta}})}{\dot{\delta}_{\max}}, 1\right). \tag{3}$$

If no constraint in (2b) is violated, $k = 1$ and the control inputs remain unchanged. For simplicity, we omit the time instant t. At each time instant, the velocity v is adjusted by a factor k, resulting in $\bar{v} = \frac{v}{k}$. To accommodate this change, a new time system $\bar{t}$ is introduced, with the new timestep $\Delta\bar{t} = k\Delta t$. Consequently, the scaled control variables are

$$\bar{\boldsymbol{u}} = \left(\frac{a}{k^2}, \frac{\dot{\delta}}{k}\right)^\top. \tag{4}$$

IV. Simulation Validation

Simulations are conducted to evaluate the computation efficiency and flexibility of the proposed algorithm versus the baseline. Simulation environment setup is MATLAB® Simulink 2022b with Intel i9-14900K CPU. System (1) uses nominal parameters $\Theta^* = [3.0988, 0.1778, 11.7348]^\top$m, with 30° maximum steering angle and 80° maximum relative angle. The nominal velocity limits for both forward and backward maneuvers are 1 m/s, and the nominal control limits are: $a_{\max} = 1$ m/s^2 and $\dot{\delta}_{\max} = 7.5^\circ$/s. The state $\boldsymbol{x}$ is in neighborhood box $\mathcal{B}_\epsilon(x_{\text{goal}})$ if the difference between $\boldsymbol{x}$ and $\boldsymbol{x}_{\text{goal}}$ is within 0.1 m for x, y positions and 1° for orientations. The maximum CPU time allocated to the relaxed path planner is $T_{\max} = 30$s. If it's re-initiated during adaptive motion planning, $T_{\max} = 1$s.

TABLE I: Comparison between DE-AGT and Adaptive-iAGT with nominal parameters. Data are computed based on successful cases.

		DE-AGT	Adaptive-iAGT
Success rate		87.3%	100%
Planning time (s)	Average	7.1289	0.1545
	STD	7.1890	0.0676
Path length (m)	Average	79.74	70.34
	STD	8.54	13.92
# of node expanded	Average	6090	53
	STD	4468	100
Percentage of cases for different # of cusps	1 cusp	5.84%	92.4%
	2 cusps	43.87%	7.3%
	3 cusps	36.54%	0.3%
	4+ cusps	13.75%	0%

Table I compares DE-AGT and Adaptive-iAGT across 1000 random test cases with nominal model parameters. Each task begins at position (515, 200) with a positional uncertainty of ±5 meters and a fixed 90° orientation for both the tractor and trailer. Initial orientation uncertainties are excluded from this comparison, as DE-AGT is not designed to handle such variations effectively. The goal is set at position (526, 201.8) with both tractor and trailer oriented at 180°. The planning is considered successful if a feasible and collision-free trajectory to the goal is generated within the maximum allowed CPU time. The empirical results demonstrate that Adaptive-iAGT achieves a higher success rate, lower computation burden, shorter paths, and fewer node expansions compared to DE-AGT. The path of Adaptive-iAGT contains fewer cusps, with most paths having only one cusp, whereas most DE-AGT paths have two or three cusps. These benefits are empowered by 1) the employment of a more powerful and more aggressive goal-reaching strategy via iLQR steering; 2) the use of control scaling, which enables the planner to find paths more quickly by temporarily allowing constraint violations that are later corrected through the scaling process. Fig. 3a-3d demonstrate tractor-trailer docking and drive-out maneuver for both DE-AGT and Adaptive-iAGT with nominal parameters. Adaptive-iAGT provides smoother paths by having fewer cusps.

Next, by applying both DE-AGT and Adaptive-iAGT to Problem 2.2 with parameter mismatches in L_1 and backward velocity limit $v_{\min}$, we showcase the efficacy of the proposed algorithm in handling parametric uncertainties in the system model and constraints without generating new MPs. Fig. 3e-3f and Fig. 4 illustrate the results for a docking task. Specifically, Fig. 4 presents the state and control trajectories of the true system when executing the motion plans from DE-AGT and Adaptive-iAGT, respectively; Fig. 3e-3f visualize the projections of the state trajectories and the resultant sweeping volume onto the xy plane.

Monte Carlo simulation results for 3 model configurations are presented in Table II. Each configuration is evaluated through 1000 randomized tests, starting from (515, 200) with a positional uncertainty of $\pm$5m. The initial orientations of both the tractor and trailer are 90° with an angular uncertainty of $\pm 15^\circ$. A customized backward velocity limit $v_{\min} = -0.5$ m/s is applied. Results in Table II show Adaptive-iAGT achieves high success rates, computational efficiency, few cusps, and precise goal reaching. Fig. 5 illustrates the distributions of planning time, cusp count, and final errors across 10,000 tests with different parameter configurations ($0 \leq h \leq 0.30$, $10.5 \leq L_1 \leq 12.5$). Overall, Adaptive-iAGT achieves 99.93% success rate, with 95.56% completed within 1 s and 92.9% with only one cusp. The planning time includes the relaxed path planner, the adaptive motion planner, and necessary planner re-runs.

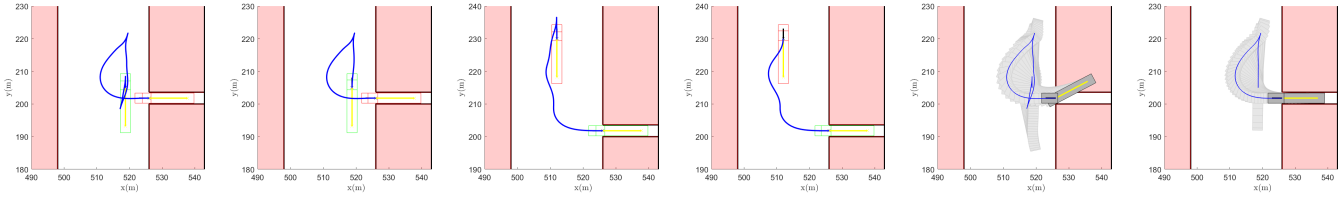
V. Field Test

Field tests of a tractor and three trailers[1] are performed to validate the applicability of Adaptive-iAGT. The trailers have different L_1 parameter values as shown in Table III. The vehicle follows Adaptive-iAGT trajectories using an MPC controller, as described in [29]. Table III summarizes the results for trailers with different axles, demonstrating smooth paths with only one cusp and precise goal-reaching with minimal errors. Final errors are measured with respect to the goal-reaching accuracy of Adaptive-iAGT, rather than the tractor–trailer's tracking accuracy in real-world, since path execution relies on an external MPC controller that introduces offsets and is beyond the scope of this work. The experiment aims to demonstrate feasibility in real-world settings, and docking is considered successful as long as constraints are satisfied, collisions are avoided, and the vehicle docks within the target region. Fig. 6 illustrates a docking test with the trajectories of the vehicle and Adaptive-iAGT. The vehicle successfully docks by following the trajectory generated by Adaptive-iAGT.

VI. Conclusion

This paper proposed Adaptive-iAGT to achieve fast adaptive planning where system parametric uncertainties

[1]Readers are referred to www.hubpilot.ai for vehicle details.



(a) DE-AGT (b) Adaptive-iAGT (c) DE-AGT (d) Adaptive-iAGT (e) DE-AGT (f) Adaptive-iAGT

Fig. 3: Comparisons of vehicle paths for different tasks. The blue solid lines represent the tractor's path. Fig. 3a-3b: docking maneuver with nominal parameters. Fig. 3c-3d: drive-out maneuver with nominal parameters. Fig. 3e-3f: docking maneuver with non-nominal parameters.

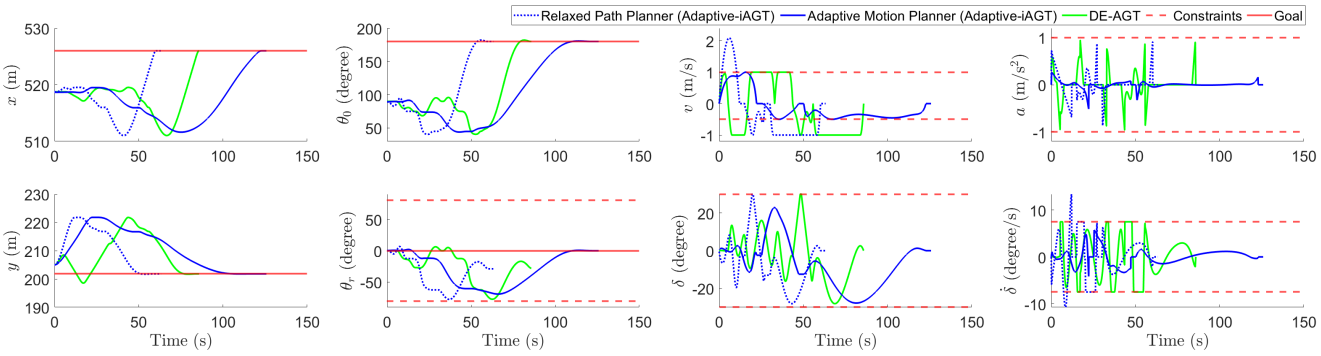


Fig. 4: Comparison of state/control trajectories of true system produced by DE-AGT, relaxed path planner, and adaptive motion planner (Adaptive-iAGT). The true system has a non-nominal parameter $L_1 = 11$ m and constraint $v_{\min} = -0.5$ m/s. Both DE-AGT and relaxed path planner generate nominal trajectory based on nominal parameters. Applying the nominal control inputs to the true system results in final state errors due to parameter discrepancies. DE-AGT fails as the resultant trajectory is not collision-free (see Fig. 3e), θ_r deviates from its goal, and the backward velocity limit is violated; so does the relaxed path planner case. Adaptive-iAGT succeeds: even though relaxed path planner generates a nominal trajectory that violates velocity/control constraints as anticipated, the trajectory corresponding to adaptive motion planner meets all criteria in Problem 2.2: collision-free (see Fig. 3f), reaching goal accurately, and constraint satification.

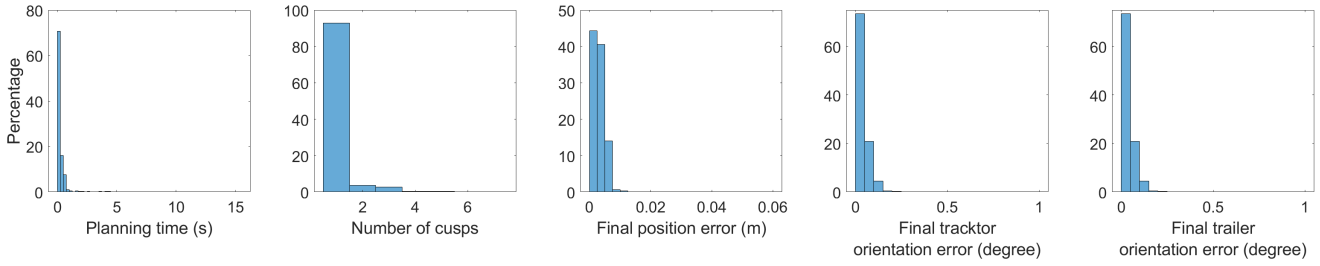


Fig. 5: Distributions of Adaptive-iAGT simulation results for 10,000 random tests.

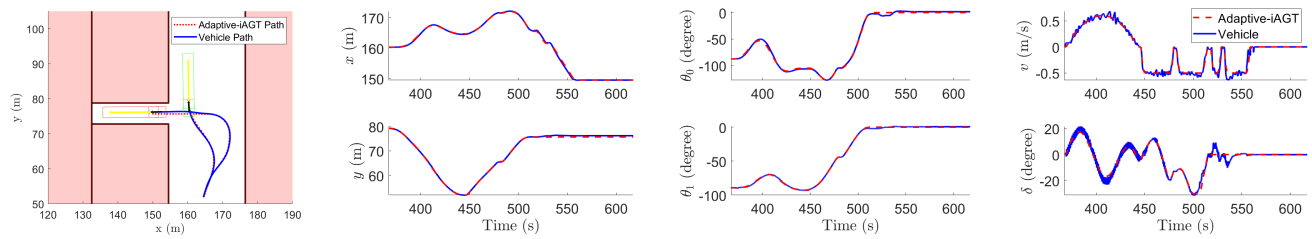


Fig. 6: Adaptive-iAGT planned trajectory and actual vehicle trajectory for field test case #1.

TABLE II: Simulation results for different tractor-trailer parameter configurations.

configuration		Success rate	Planning time (s)		# of node expanded		# of cusps				Final errors (average)
h (m)	L_1 (m)		Average	STD	Average	STD	1	2	3	4+	[x(m), y(m), $\theta_0(^\circ)$, $\theta_1(^\circ)$]
0	10.5	99.9%	0.3682	1.1729	85	319	96.1%	1.2%	1.4%	1.3%	[0.0028, 0.0033, 0.0121, 0.0496]
0.15	11.5	100%	0.4081	0.8377	142	528	94.8%	2.4%	2.5%	0.3%	[0.0014, 0.0016, 0.0068, 0.0279]
0.30	12.5	99.6%	0.5753	1.1318	169	545	89.4%	4.3%	4.8%	1.5%	[0.0012, 0.0023, 0.0084, 0.0396]

TABLE III: Field tests docking results.

Case	L_1 (m)	Adaptive-iAGT final errors [x(m), y(m), $\theta_0(^\circ)$, $\theta_1(^\circ)$]	# of Cusps
#1	11.72	[0.0003, 0.0010, 0.0005, 0.0072]	1
#2	11.87	[0.0014, 0.0010, 0.0069, 0.0757]	1
#3	12.03	[0.0008, 0.0018, 0.0114, 0.0343]	1

are taken into account. Adaptive-iAGT embraces a two-stage framework to handle model uncertainties without pre-computing/storing motion primitives for each set of parameters. Distinguishably, a relaxed path planner for nominal trajectory generation is equipped with iLQR steering, where only mechanical constraints are enforced; an adaptive motion planner adapts the nominal trajectory for a new trajectory which is dynamically feasible for system with true parameters; and an analytical scaling technique is developed to accommodate customizable velocity and control constraints. Both simulations and field tests validate the effectiveness and practical applicability of the proposed algorithm.

REFERENCES

[1] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 42–49.

[2] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "Gusto: Guaranteed sequential trajectory optimization via sequential convex programming," in *2019 International conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6741–6747.

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

[4] Z. Liang, T. Zhou, Z. Lu, and S. Mou, "Online control-informed learning," *Transactions on Machine Learning Research*, 2025.

[5] 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.

[6] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2015, pp. 109–124.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] ——, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[9] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5054–5061.

[10] Y. Li, Z. Littlefield, and K. E. Bekris, "Asymptotically optimal sampling-based kinodynamic planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 528–564, 2016.

[11] X. Zhang, J. Eck, and F. Lotz, "A path planning approach for tractor-trailer system based on semi-supervised learning," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2022, pp. 3549–3555.

[12] T. Lai, W. Zhi, T. Hermans, and F. Ramos, "Neural kinodynamic planning: Learning for kinodynamic tree expansion," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 789–11 795.

[13] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[14] D. Zheng and P. Tsiotras, "Sampling-based kinodynamic motion planning using a neural network controller," in *AIAA Scitech 2021 Forum*, 2021, p. 1754.

[15] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[16] 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.

[17] 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.

[18] 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.

[19] B. Sakcak, L. Bascetta, G. Ferretti, and M. Prandini, "Sampling-based optimal kinodynamic planning with motion primitives," *Autonomous Robots*, vol. 43, no. 7, pp. 1715–1732, 2019.

[20] 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.

[21] 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.

[22] 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.

[23] P. Töws and D. Zöbel, "Reversing general 2-trailer vehicles using a 2d steering function model and a novel mesh search algorithm," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1274–1281.

[24] Y. Wang, "Improved a-search guided tree construction for kinodynamic planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5530–5536.

[25] 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.

[26] D. Zheng, Y. Wang, S. D. Cairano, and P. Tsiotras, "Delayed expansion agt: Kinodynamic planning with application to tractor-trailer parking," *arXiv*.

[27] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *First International Conference on Informatics in Control, Automation and Robotics*, vol. 2. SciTePress, 2004, pp. 222–229.

[28] 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.

[29] S. You, M. Greiff, R. Quirynen, S. Ran, Y. Wang, K. Berntorp, R. Dai, and S. Di Cairano, "Integral action nmpc for tight maneuvers of articulated vehicles," in *2023 American Control Conference (ACC)*. IEEE, 2023, pp. 3155–3161.

[30] 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.