Unmanned Systems, Vol. 12, No. 5 (2024) 927–938

DOI: 10.1142/S2301385024500286

UNMANNED SYSTEMS
https://www.worldscientific.com/worldscinet/us

Hierarchical MPC-based Motion Planning for Automated Vehicles in Parallel Autonomy

Zijun Cheng*, Xianlin Zeng*,‡, Hao Fang*, Gang Wang*,†, Lihua Dou*

*National Key Lab of Autonomous Intelligent Unmanned Systems, Beijing Institute of Technology, Beijing 100081, P. R. China

†Beijing Institute of Technology Chongqing Innovation Center, Chongqing 401120, P. R. China

Automated vehicles with parallel autonomy show advantages over fully automated vehicles and manual driving. This paper proposes a hierarchical motion planning method that mixes inputs of human drivers and the automated driving systems for automated vehicles in scenarios such as multi-lane roads and multi-intersections with dynamic obstacles. The proposed method comprises a reference path generator in the upper level and a nonlinear model predictive controller with mixed human-vehicle control in the lower level. The path planner considers dynamic obstacles, static obstacles, and human comfort to generate a reference path composed of splines with continuous curvatures in the upper level. In the lower level, the MPC generates a trajectory by tracking the reference path and optimizing the cost function containing inputs of drivers while avoiding both dynamic and static obstacles. The simulation verifies the efficacy and the computational tractability of the proposed method.

Keywords: Hierarchical model predictive control; motion planning; automated vehicle; shared control.

US

1. Introduction

Automated vehicles have the potential to drastically reduce traffic accident rates and improve traffic efficiency [1–4]. Techniques for intelligent autonomous systems (see, e.g. [5–11]) have received lots of attention and have been applied to automated vehicles. Fully automated vehicles with no driver have a few disadvantages despise that fully autonomous driving is the ultimate goal. For one thing, fully autonomous driving may increase boredom and distractedness in driving operations [12]. For another, fully autonomous vehicles can suffer system failures and performance degradation because of complex traffic, weather, and hardware/software issues.

In the concept of shared control or parallel autonomy of automated vehicles [13–15], the driver and the automated system operate in the control loop to perform the same task. The automated system in shared control maintains basic safety when the human driver is affected by unexpected issues or biological factors. Furthermore, the shared control keeps the driving pleasure of human drivers and is adaptive to complex driving scenarios compared to fully automated vehicles. Hence, there is an increasing interest in various forms of shared control for automated vehicles [16–19].

One of the most crucial tasks for automated vehicles in parallel autonomy is motion planning [20, 21], which computes a trajectory that satisfies certain constraints and requirements from drivers and vehicles. One widely studied shared control theme is continuously combining inputs from human drivers and automated driving agents without driver–vehicle authority transitions. In this theme, an intuitive technique in [22–26], analogous to human–robot shared control, is directly using a linear combination of human drivers and driving systems system inputs as the control. Recently, the works [27, 28] have further proposed optimization-based methods, which consider human inputs in the cost function of an MPC and minimize the deviation

Received 6 August 2022; Revised 11 March 2023; Accepted 12 March 2023; Published 12 May 2023. This paper was recommended for publication in its revised form by editorial board member, Shupeng Lai.
Email Addresses: ‡*xianlin.zeng@bit.edu.cn; xianlinzeng@gmail.com*

between human input and automated system while satisfying safety constraints. However, the non-convex MPC caused by dynamic vehicle models and obstacles may be hard to solve due to high computational complexities and even infeasible optimization models.

Hierarchical model predictive control (MPC) adopts the strength of MPC for handling constraints and often owns lower computational complexities. In this framework, the upper level is a reference path planner, and the lower level is usually an MPC solver to track the reference path. Specifically, the hierarchical structure in [29–31] computes an obstacle-free path in the upper level of the controller. It generates the reference path following trajectory using the lower-level MPC solver, which considers fundamental limitations on the performance of vehicles. The hierarchical motion planning method in [31] uses input space discretization methods as the high-level planner, which chooses a collision-free lane. However, the high-level planner in [31] generates nonsmooth paths, which are difficult to track. Another hierarchical motion planning framework [32–34] adds the avoidance of boundaries in low-level model predictive contouring control (MPCC) solvers in car racing scenarios, where trajectories of racing cars need to be close to the boundary of roads to save time. Specifically, the work of [34] considers avoiding boundaries in low-level MPC solvers and using the input space discretization method in the upper level. Furthermore, a simple MPC solver in the upper level to generate reference paths is implemented in [32, 33]. In the hierarchical structure in [35], whose low-level MPC solver tackles dynamic obstacles, the upper-level planner in [35] is a finite state machine that generates decisions such as following the head car, overtaking, and changing lanes. However, hierarchical MPC methods for motion planning of vehicles in shared control are rarely reported.

In this paper, we propose a shared control algorithm based on hierarchical MPC for the motion planning of the automated vehicle in multi-lane roads and multi-intersections; see Fig. 1 for a pictorial description. Our contribution is twofold and summarized as follows:

(1) We propose a hierarchical MPC-based motion planning method for automated vehicles in parallel autonomy suitable for complex driving scenarios, including multi-lane roads, multi-intersections, and some traffic rules. Using a spline-based pruning algorithm, the upper-level planner builds a reference path and reference velocities in scenarios like multi-lane roads and multi-intersections. The low-level trajectory generator generates a safe trajectory by following the reference path while considering human inputs and constraints. This method extends that in [27] to multi-lanes cases and improves safety.
(2) We test the efficacy of the proposed method in traffic scenarios, including multi-lane roads and multi-intersections, via simulations using the MATLAB Simulink and the CARLA Simulator. The simulation shows empirically that the proposed method can generate real-time and high-quality trajectories in complex scenarios.

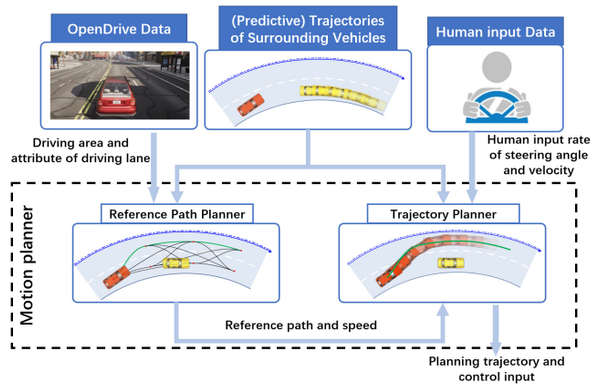


Fig. 1. The main framework of our methods.

The remainder of this paper organizes as follows. Section 2 introduces related preliminaries and formulates the problem. Section 3 presents a hierarchical MPC method for motion planning for automated vehicles with parallel autonomy. Section 4 conducts simulation results to evaluate and show the efficacy of the proposed method. Section 5 gives the conclusion.

2. Problem Description

This section presents mathematical preliminaries related to motion planning and formulates the problem.

2.1. *Definitions and motion modeling*

The symbol $\mathbb{R}$ denotes the set of real numbers; $\mathbb{R}^n$ denotes the set of n-dimensional real column vectors; $\mathbb{R}^{n\times m}$ denotes the set of n-by-m real matrices; I_n denotes the $n \times n$ identity matrix; $(\cdot)^{\mathrm{T}}$ denotes transpose. For the ease of notation, we use k to denote time instant t_k, that is, $k \triangleq t_k = t_0 + \sum_{i=1}^{k} \Delta t_i$, where t_0 is the current time and Δt_i is the ith time step of the planner.

2.1.1. *Ego vehicle axis system*

This paper denotes the vehicle that needs motion planning as the ego vehicle. The ego vehicle axis system (ISO 8855) is attached to the ego vehicle at the current time $t_0 = 0$. The X-axis is horizontal, forwards, and parallel to the ego vehicle's longitudinal plane of symmetry. The Y-axis is perpendicular to the ego vehicle's longitudinal plane of symmetry at time 0.

2.1.2. *Ego vehicle*

Consider the ego vehicle axis system at current time $t_0 = 0$. The state of the ego vehicle at time instant k in the ego vehicle axis system is a six-tuple $\mathbf{z}_k = [x_k, y_k, v_k, \phi_k, \delta_k, s_k]^{\mathrm{T}} \in \mathcal{Z} \subset \mathbb{R}^6$, where (x_k, y_k) is the position of ego vehicle's centroid in ego vehicle axis system, v_k is the longitudinal velocity, ϕ_k is the yaw or heading, δ_k is steering angle, s_k is the driving length from time 0 to time k and $s_0 = 0$. The input of the ego vehicle at time instant k is $\mathbf{u}_k = [\dot{\delta}_k, \dot{v}_k]^{\mathrm{T}} \in \mathbb{R}^2$, where $\dot{v}_k$ is the acceleration and $\dot{\delta}_k$ is the steering velocity.

The kinematic model of the ego vehicle (c.f. Fig. 2) is a single bicycle model, which has a steerable front wheel and a fixed rear wheel. The evolution of state $\mathbf{z}(t)$ is given by

$$\dot{\mathbf{z}}(t) = \begin{bmatrix} v(t)\cos\left(\phi(t) + \arctan\dfrac{l_r \tan\delta(t)}{l_r + l_f}\right) \\ v(t)\sin\left(\phi(t) + \arctan\dfrac{l_r \tan\delta(t)}{l_r + l_f}\right) \\ 0 \\ \dfrac{v(t)}{l_r + l_f}\tan\delta(t) \\ 0 \\ v(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \mathbf{u}(t), \tag{1}$$

where $t \geq 0$, $\mathbf{z}(0) \in \mathcal{Z}$, l_f (l_r) is the distance from the front (rear) wheel to the centroid.

The discrete-time dynamical model of the ego vehicle takes the form

$$\mathbf{z}_{k+1} = f(\mathbf{z}_k, \mathbf{u}_k), \tag{2}$$

where $f(\mathbf{z}_k, u_k)$ is computed using the fourth-order Runge–Kutta method to compute the following integration with sufficient accuracy:

$$f(\mathbf{z}_k, \mathbf{u}_k) = \mathbf{z}_k + \int_k^{k+\Delta t_k} \dot{\mathbf{z}}(t)\mathrm{d}t, \quad k = 0, 1, \ldots, m. \tag{3}$$

Given the state of the ego vehicle, we use $\mathcal{B}(\mathbf{z}_k) \in \mathbb{R}^2$ to denote the set of occupied area by the vehicle body at time k.

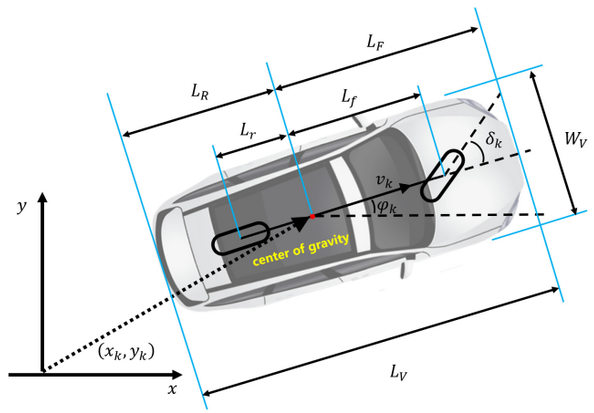


Fig. 2. Kinematic single bicycle model.

2.1.3. *Surrounding vehicles*

Consider the ego vehicle axis system at current time $t_0 = 0$. The state of a surrounding vehicle $i \in \{1, \ldots, n\}$ at time 0 is denoted as a four-tuple $\mathbf{z}_0^i = [x_0^i, y_0^i, \phi_0^i, v_0^i]^{\mathrm{T}} \in \mathcal{Z}^i \subset \mathbb{R}^4$. For simplicity, its predicted state at time instant k is denoted as $\mathbf{z}_k^i$. The predicted state of the surrounding vehicle $i \in \{1, \ldots, n\}$ from current time to time instant m is denoted as $\mathbf{z}_{0:m}^i = [\mathbf{z}_0^i, \ldots, \mathbf{z}_m^i]$. The estimated trajectories are chosen as the states at current time $t_0 = 0$ if the estimated trajectories are not available.

2.1.4. *Driving area*

We denote $\mathcal{W} \subseteq \mathbb{R}^2$ as the workspace in the ego vehicle axis system. The road information is assumed to be in the OpenDrive format. Attributes of lanes in the road include lane widths, lane IDs, speed limits, etc. The *driving area* is the drivable area of the road based on OpenDrive. Take Fig. 3 as an example. The ego vehicle is allowed a lane change in the driving area, the light blue area in Fig. 3. Boundaries of the driving area are lane markings, which do not allow a lane change. The other side of the boundaries is the lane with opposite driving direction or undrivable area such as shoulders, curbs, and parking lanes. We denote $\mathcal{O} \subseteq \mathbb{R}^2$ as the area, which does not belong to the driving area in $\mathcal{W}$.

2.1.5. *Path and route of vehicle*

Consider a smooth curve of a vehicle in the driving area. We use a 3-tuple of continuous functions to describe the path as $\mathbf{r}(s) = [x_{\mathbf{r}}(s), y_{\mathbf{r}}(s), \phi_{\mathbf{r}}(s)]$, where s is arc length of the path,

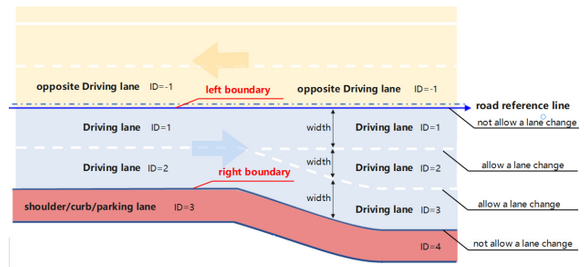


Fig. 3. The definition of driving area in OpenDrive. The driver area is light blue.

$[x_{\mathbf{r}}(s), y_{\mathbf{r}}(s)]$ is the position in the ego vehicle axis system, and $\phi_{\mathbf{r}}(s)$ is the yaw or heading.

2.2. *Problem formulation*

The objective of this paper is to propose a real-time shared control method to compute a sequence of inputs $\{\mathbf{u}_{0:m-1}\}$ for the ego vehicle such that the vehicle can drive safely and smoothly along the navigation route in the driving area with multiple roads and lanes. We separate the motion planning task into a reference path-speed generating problem and a safe trajectory planning problem with shared inputs to achieve this objective.

2.2.1. *Smooth reference path and speed generating problem*

Suppose the ego vehicle is aware of the following information:

- the navigation route of the ego vehicle;
- the lane width, speed limit, and lane ID of the driving lane in the driving area;
- current states or estimated trajectories of surrounding vehicles.

The smooth reference path and speed generating problem is to design a smooth reference path with continuous curvature $\mathbf{r}(s) = [x_{\mathbf{r}}(s), y_{\mathbf{r}}(s), \phi_{\mathbf{r}}(s)]$ and the reference speed v_{ref} without violating traffic rules and colliding surrounding vehicles. Here, $x_{\mathbf{r}}$ and $y_{\mathbf{r}}$ are the positions in the ego vehicle axis system, $\phi_{\mathbf{r}}$ is yaw or heading, and s is the arc length of the path.

Remark 2.1. The reference path and speed obtained by solving the problem can direct the vehicle to a suitable lane in the presence of dynamic obstacles without violating traffic rules. Unlike the method in [36], this paper relaxes the requirement that the generated reference path needs to own continuous curvature derivatives.

2.2.2. *Safe trajectory planning problem with shared inputs*

Suppose the reference path $\mathbf{r}(s)$, the reference speed v_{ref}, and estimated trajectories of surrounding vehicles $\mathbf{z}^i_{0:m}$ are obtained. Let $\mathbf{u}_{0:m-1}$ and $\mathbf{z}_{0:m}$ be control inputs and trajectory of the ego vehicle, respectively. We define the following cost functions:

- $J_{\text{r}}(\mathbf{z}_{0:m})$ is a cost to minimize deviations of the speed from the reference speed v_{ref} and the ego vehicle's state from the reference path $\mathbf{r}(s)$;
- $J_{\text{c}}(\mathbf{z}_{0:m}, \mathbf{u}_{0:m-1})$ is a cost to minimize energy consumption and maintain the comfort of humans;
- $J_{\text{s}}(\mathbf{z}_{0:m})$ is a cost to promote the vehicle to run faster;
- $J_{\text{h}}(\mathbf{u}^h_0, \mathbf{u}_{0:m-1})$ is a cost to minimize the deviation of control input from the current human input $\mathbf{u}^h_0$.

The *safe trajectory planning problem with shared inputs* is to find the optimal sequence of control input $\mathbf{u}_{0:m-1}$ and trajectory $\mathbf{z}_{0:m}$ by solving the following optimization problem:

$$\min_{\mathbf{u}_{0:m-1}} J_{\text{r}}(\mathbf{z}_{0:m}) + J_{\text{c}}(\mathbf{z}_{0:m}, \mathbf{u}_{0:m-1}) + J_{\text{s}}(\mathbf{z}_{0:m}) + J_{\text{h}}(\mathbf{u}^h_0, \mathbf{u}_{0:m-1}) \tag{4a}$$

$$\text{s.t.} \quad \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max}, \tag{4b}$$

$$\mathbf{z}_{\min} \le \mathbf{z}_k \le \mathbf{z}_{\max}, \tag{4c}$$

$$g_{\min} \le g(\mathbf{z}_k, \mathbf{u}_k) \le g_{\max} \tag{4d}$$

$$\mathbf{z}_{k+1} = f(\mathbf{z}_k, \mathbf{u}_k), \tag{4e}$$

$$\mathcal{B}(\mathbf{z}_k) \cap \left(\bigcup_{i=1}^{n} \mathcal{B}^i(\mathbf{z}^i_k) \right) = \emptyset, \tag{4f}$$

$$\mathcal{B}(\mathbf{z}_k) \cap \mathcal{O} = \emptyset, \quad \forall k = 0, 1, \ldots, m. \tag{4g}$$

In problem (4), there are five types of constraints.

- Constraints (4b) and (4c) are limits on the inputs of vehicles and constraints on the workspace of vehicles, given by $0 \le v_k \le v_{\max}$, $\dot{v}_{\min} \le \dot{v}_k \le \dot{v}_{\max}$, $\delta_{\min} \le \delta_k \le \delta_{\max}$, $\dot{\delta}_{\min} \le \dot{\delta}_k \le \dot{\delta}_{\max}$, $\phi_{\min} \le \phi_k \le \phi_{\max}$, $x_{\min} \le x_k \le x_{\max}$, and $y_{\min} \le y_k \le y_{\max}$.
- Constraint (4d) is the dynamic constraint of the vehicle. This paper considers the rollover constraint given by $g(\mathbf{z}_k, \mathbf{u}_k) = |v_k \dot{\phi}_k| \le g_{\max}$, $k = 0, 1, \ldots, m$.
- Constraint (4e) is the dynamical model of the ego vehicle.
- Constraint (4f) is the avoidance of collisions with surrounding vehicles.
- Constraint (4g) is the avoidance of collisions with static obstacles.

More details of problem (4) are given in Sec. 3.2.

This paper considers the scenario having multi-lane and multi-intersection roads with traffic rules, which are more general compared with [27, 33, 34]. Note that [33, 34] only apply to a racing scenario with no traffic rules, such as speed limit and allowing a lane change. Moreover, this paper designs a new hierarchical MPC-based method to achieve shared control of vehicles. Compared with other hierarchical motion planning methods [29, 31], whose low-level planners only follow the path rather than avoid obstacles, we add human input in the low-level planner while avoiding collisions with other vehicles and road boundaries.

3. Hierarchical MPC-based Method

This section proposes a hierarchical MPC-based motion planning method (see Fig. 4) by combining a spline-based reference path planner and an MPCC-based trajectory planner.

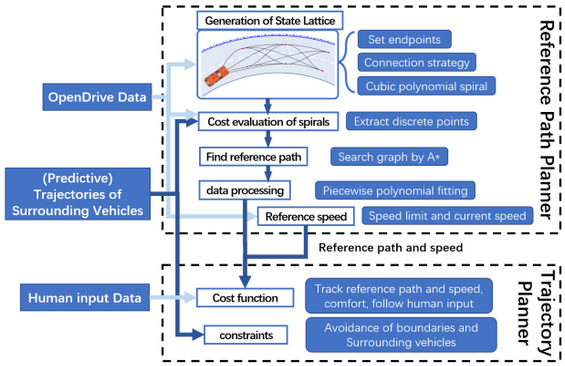


Fig. 4. The main framework of the hierarchical MPC-based motion planning method.

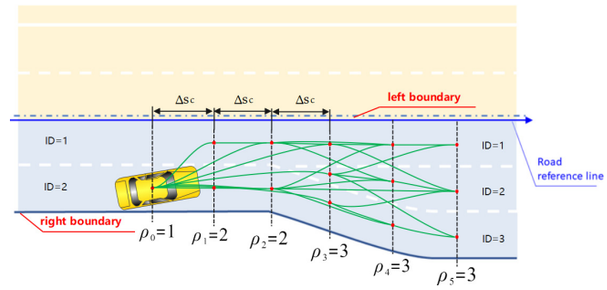


Fig. 5. Generation of state lattice. Red points are endpoints in rows. Green solid curves are spirals.

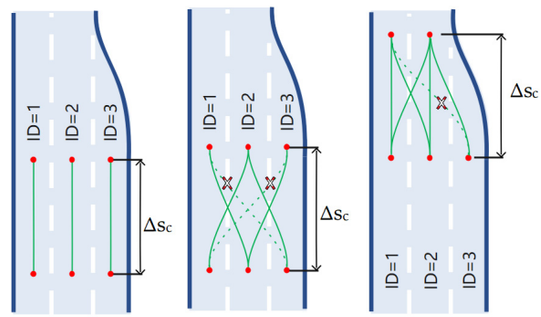


Fig. 6. Left: direct connection. Middle: indirect connection. Right: full connection.

Specifically, the high-level planner constructs a state lattice composed of connected cubic polynomial spirals using the OpenDrive Data. Then, the high-level planner generates a reference path by searching the lattice via the A* algorithm and reference speed based on rules, including the lane speed limit and current speed of the ego vehicle. The low-level planner uses a human–vehicle shared model predictive contouring control method to follow the reference path and reference speed by combining human input with the predictive trajectories or positions of surrounding vehicles while satisfying safety constraints, including the state limit of the ego vehicle, avoidance of road boundaries and surrounding vehicles.

3.1. *Reference path planner*

3.1.1. *Generation of state lattice*

We build the state lattice, which is a discrete search graph on a continuous state space, by three steps.

First, inspired by [36], we sample multiple rows of endpoints ahead of the ego vehicle. The distance between adjacent rows of endpoints along the road reference line is $\Delta s_{\mathbf{c}}$. The endpoints in each row are located in the center of each lane. The endpoints in the same lane have the same lane ID. Note that the whole endpoints are in the driving area. That means the ego vehicle is allowed a state transition between the endpoints, including the lane change, under the traffic rules restrictions. The speed limits of endpoints are the speed limits of lanes in which the endpoints locate. The number of rows is Q, which is chosen depending on the current speed v_0 of the ego vehicle. The number of endpoints in the q-th row is denoted as ρ_q, as shown in Fig. 5.

Second, we connect endpoints in different rows based on three basic connection modes (see Fig. 6). Mode 1 is *direct connection* that connects endpoints in the same lane between two adjacent rows; mode 2 is *indirect connection* that connects points in different lanes between two rows; and mode 3 is *full connection* that connects all possible combinations of points between two rows. We only allow connecting endpoints in the same lane or the adjacent lane in the whole three connection modes. Algorithm 1 presents the specific algorithm to build the state lattice by pruning unnecessary connections.

Third, we compute cubic polynomial spirals [37] that connect endpoints using Newton's method and delete spirals whose computation has computational or geometric errors. In this setup, curvatures of cubic polynomial spirals are continuous cubic polynomial functions of the arc length. The curvature of the endpoint is the same as the center of the lane in which the endpoint locates. The curvature of the endpoint located in the centroid of the ego vehicle is

$$(\tan\delta_0)/(l_f + l_r). \tag{5}$$

We consider three common errors that often occur in the computation using Newton's method: (1) circle error in which the spiral creates a circle, (2) iteration error whose the number of iterations is too large, (3) singular value error which means the Jacobian matrix can not be inverted. To improve the method's stability, spirals whose computation shows any of the three errors are deleted.

Algorithm 1. Pruning algorithm for building the state lattice

Data: $Q \in \{6, \ldots, 10\}$, $\rho_0 = 1$, and $\rho_{-1} = \rho_{-2} = 0$
for $q = 1, 2, ..., Q$ **do**
  if ρ_q *equals* ρ_{q-1} **then**
    do **direct connection** between q-th and $(q-1)$-th row.
  else
    do **full connection** between q-th and $(q-1)$-th row.
  end
  if ρ_q *equals* ρ_{q-2} **then**
    do **indirect connection** between q-th and $(q-2)$-th row.
  else
    do **full connection** between q-th and $(q-2)$-th row.
  end
  if ρ_q *equals* ρ_{q-3} **then**
    do **indirect connection** between q-th and $(q-3)$-th row.
  end
  $\rho_{q-2} \to \rho_{q-3}$
  $\rho_{q-1} \to \rho_{q-2}$
  $\rho_q \to \rho_{q-1}$
end

Remark 3.1. We extend the state lattice method in [36] to a multi-lane road. Compared with the method in [36], the proposed reference path planner uses a larger Q (the number of rows) and a smaller ρ_q (the number of endpoints in each row). This modification may increase computation complexity and redundancy, and hence, we propose the pruning Algorithm 1 to curtail redundancy.

Remark 3.2. One also needs to choose appropriate Q and $\Delta s_{\mathbf{c}}$. If Q is too large, the graph searching algorithm A*, used in Sec. 3.1.3, may be time-consuming because its computational complexity is $O(Q^2 \log Q)$. In addition, the spirals generated by too large $\Delta s_{\mathbf{c}}$ may deviate from the lanes in sharp turning areas. This paper chooses $Q \in \{6, 7, 8, 9, 10\}$ in the numerical simulation.

3.1.2. *Cost evaluation of spirals*

To evaluate its cost, following [38], we evenly extract w discrete points from each spiral. Denote the curvature at jth point as κ_j and the position of jth point in the ego vehicle axis system as p_j. Denote the length of a spiral as l^{len} and the length of a spiral from the first point to the jth as $l^{\text{len}}_{1:j}$. The lane IDs, in which endpoints of a spiral locate, are I_1 and I_2.

The cost of a spiral is composed of the following parts:

- cost of the curvature derivative $J_{\text{dcurv}} = \sum_{k=2}^{w} |\kappa_k - \kappa_{k-1}|/l^{\text{len}}$;
- cost of the curvature $J_{\text{curv}} = \sum_{k=1}^{w} |\kappa_k|/l^{\text{len}}$;
- cost of the lane change $J_{\text{lane}} = |I_1 - I_2|$;
- cost of the collision with boundaries J_{static}, which is the number of evaluation points outside the driving area, i.e. $J_{\text{bound}} = \mathbf{card}(p_{1:w} \cap \mathcal{O})$;
- cost of the collision with dynamic obstacle J_{dynamic}, which is the number of evaluation points within the circle with center at the position of surrounding vehicles, i.e. $J_{\text{dynamic}} = \sum_{i=1}^{n} \sum_{k=1}^{w} \mathbf{card}\{\|p_k - \mathbf{z}^i_{\text{pos}}\|_2 < \sqrt{6}\}$, where $\mathbf{z}^i_{\text{pos}} = [x^i_0, y^i_0]$ is the current position of the ith surrounding vehicle, and n is the number of surrounding vehicles.

The cost of a spiral is

$$J_{\text{spiral}} = \alpha_1 J_{\text{dcurv}} + \alpha_2 J_{\text{curv}} + \alpha_3 J_{\text{lane}} + \alpha_4 J_{\text{dynamic}} + \alpha_5 J_{\text{bound}}, \quad (6)$$

where $\alpha_1, \ldots, \alpha_5$ are positive weights for different costs.

3.1.3. *Reference path and piecewise polynomial fitting*

Consider the obtained state lattice by Algorithm 1 and the cost function of spirals. We use A* algorithm to generate a path by searching for a feasible combination spiral from the 0th row to Qth row with minimum cost, which is the sum of costs on spirals in the path. The reference path $K(s_{\mathbf{r}})$ curvature is represented as a piecewise continuous cubic function of the arc length of the reference path $s_{\mathbf{r}}$. The positions of points in the reference path can be computed by integrating $K(s_{\mathbf{r}})$. However, computing the position function of the path is computationally expensive. Hence, we use the piecewise polynomial fitting technique [39] to approximate the reference path in ego vehicle axis system $(X(s_{\mathbf{r}}), Y(s_{\mathbf{r}}))$, which is a continuous function of the arc length $(s_{\mathbf{r}})$ traveled in the reference path. Similarly, we can get the yaw of the reference path $\Phi(s_{\mathbf{r}})$, the distance between the left (or right) boundary and reference path $B_l(s_{\mathbf{r}})$ (or $B_r(s_{\mathbf{r}})$). We assume that $B_l(s_{\mathbf{r}}) > 0$ and $B_r(s_{\mathbf{r}}) < 0$.

Remark 3.3. We find that an eighth-order polynomial is sufficient to fit the position and angle of any geometric lines with the variable of arc length, even a nonsmooth line. Moreover, the representation is feasible and solvable in the existing nonlinear solver.

3.1.4. *Reference speed generation*

The current speed of the ego vehicle is v_0. The minimum speed limit of the endpoints which belong to the reference

path is v_{limit}. In order to ensure the stability of acceleration, the reference speed is $v_{\text{ref}} = \min\{v_0 + 6, v_{\text{limit}}\}$. In addition, the reference speed is set to zero in case the ego vehicle needs to stop. For example, the ego vehicle needs to stop at the crossing to wait for the traffic light to change.

3.2. *Trajectory planner*

3.2.1. *Cost function*

The cost function comprises four parts: the cost of tracking reference path, the cost of human comfort, the cost of tracking reference speed, and the cost of human input.

Cost of tracking reference path. Note that the computational cost of tracking a path by minimizing the minimum distance between (x_k, y_k) and the reference path is high. We instead minimize the distance between the state of the ego vehicle and the reference point $(X(s_k), Y(s_k), \Phi(s_k))$, where s_k is the traveling distance of the ego car at time k. The error between (x_k, y_k) and $(X(s_k), Y(s_k))$ is composed of the lateral error e_c and the longitudinal error e_l, as shown in Fig. 7. The objective of the tracking is to minimize $e_c(s_k)$ and $e_l(s_k)$, $k = 1, 2, \ldots, m$.

We replace s_k with $s_k - e_l(s_k)$ and get a new reference point $(X(s_k - e_l(s_k)), Y(s_k - e_l(s_k)), \Phi(s_k - e_l(s_k)))$. We denote the new lateral error as e'_c. The cost of tracking the reference path at time k is

$$J_{\text{r}}(\mathbf{z}_k) = \beta_1 |e'_c|^2, \tag{7}$$

where

$$\begin{aligned} e'_c = &-(x_k - X(s_k - e_l(s_k))) \sin \Phi(s_k - e_l(s_k)) \\ &+(y_k - Y(s_k - e_l(s_k))) \cos \Phi(s_k - e_l(s_k)) \end{aligned} \tag{8}$$

and $e_l = -(x_k - X(s_k)) \cos \Phi(s_k) - (y_k - Y(s_k)) \sin \Phi(s_k)$.

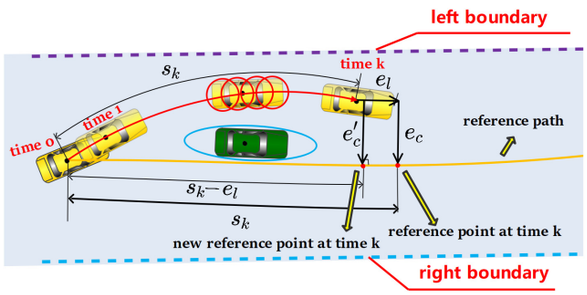


Fig. 7. The yellow car is the ego vehicle at different times, and the green car is the ith surrounding vehicle. The solid red curve denotes the generated trajectory of the ego vehicle; the yellow solid curve is the reference path; the purple and blue dashed curves are the left and right boundaries. Red solid circles are denoted by $C_j(x_\tau, y_\tau, \phi_\tau)$. The blue ellipse is denoted by $\mathcal{E}_i(\mathbf{z}^i_\tau, a^i_\tau, b^i_\tau)$.

Cost of human comfort. We evaluate human comfort using the quadratic function

$$J_{\text{c}}(\mathbf{z}_k, \mathbf{u}_k) = \beta_{21} \dot{v}_k^2 + \beta_{22} \dot{\delta}_k^2 + \beta_{23} \dot{\phi}_k^2. \tag{9}$$

Cost of tracking reference speed. The speed cost is

$$J_s(\mathbf{z}_k) = \beta_3 (v_k - v_{\text{ref}})^2. \tag{10}$$

Cost of human input. Suppose that current human inputs are $\dot{v}_0^h, \delta_0^h$. We define a cost function measuring the difference between human input and actual input of the ego vehicle. To be specific, the cost of human input is

$$J_h(\mathbf{z}_k, \mathbf{u}_0^h) = \beta_{41} (\dot{v}_k - \dot{v}_0^h)^2 + \beta_{42} (\delta_k - \delta_0^h)^2. \tag{11}$$

The ego vehicle is fully-automated when $\beta_{41} = \beta_{42} = 0$.

Total cost function. Similar to [27], we define the total cost function by multiplying different costs with coefficients

$$\begin{aligned} J_{\text{total}}(\mathbf{z}_k, \mathbf{u}_k, \mathbf{u}_0^h) \triangleq\ & (J_{\text{r}}(\mathbf{z}_k) + J_{\text{c}}(\mathbf{z}_k, \mathbf{u}_k) + J_{\text{s}}(\mathbf{z}_k)) \\ & \times (1 - e^{-\beta_5 k}) + \beta_6 e^{-\beta_5 k} J_{\text{h}}(\mathbf{z}_k, \mathbf{u}_0^h). \end{aligned} \tag{12}$$

3.2.2. *Constraints*

To make the motion planning safe, the trajectory of the ego vehicle needs to satisfy the following collision avoidance constraints.

Avoidance of boundaries. The length of the projection of the ego vehicle's contouring to the lateral direction is W', given by

$$\begin{aligned} W' = &\frac{W_v}{2} \cos(\Phi(s_k - e_l(s_k)) - \phi_k) + \max(L_F, L_R) \\ &\times \sin(|\phi_k - \Phi(s_k - e_l(s_k))|), \end{aligned} \tag{13}$$

where W_v, L_F, L_R are shown in the Fig. 2. Regard the modified lateral error $e'_c(s_k)$ in (8) as the distance between (x_k, y_k) and reference path. We have constraints

$$e'_c - B_r(s_k - e_l(s_k)) - W' \geq 0, \quad (k = 0, 1, \ldots, m), \tag{14a}$$

$$B_l(s_k - e_l(s_k)) - W' - e'_c \geq 0, \quad (k = 0, 1, \ldots, m), \tag{14b}$$

where $e'_c - B_r(s_k - e_l(s_k)) - W'$ is a safe distance between (x_k, y_k) and right boundary, $B_l(s_k - e_l(s_k)) - W' - e'_c$ is a safe distance between (x_k, y_k) and left boundary.

Remark 3.4. At the planning time k, the ego vehicle needs to find the nearest reference point to get the distance between (x_k, y_k) and the left road boundary. The MPCC algorithm in [27] uses a reference point to approximate the nearest reference point for computation efficiency. The estimated distance between (x_k, y_k) and the reference path is represented as $B_l(s_k) - W - e_c$, where $W = \frac{W_v}{2} \cos(\Phi(s_k) - \phi_k) + \max(l_f, l_r) \sin(|\phi_k - \Phi(s_k)|)$, which may be very different from the actual distance, as shown in Fig. 8. This paper proposes a new method to compute a

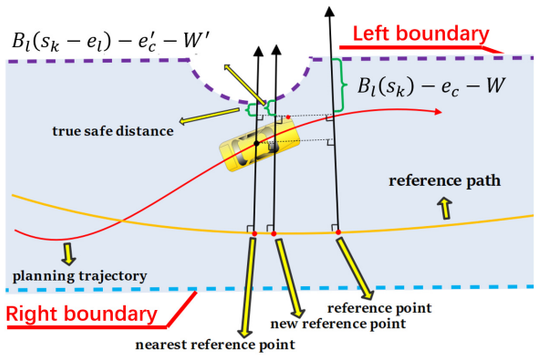


Fig. 8. Estimation of the safe distance.

reference point, whose estimation of the distance between (x_k, y_k) and the left road boundary is $B_l(s_k - e_l(s_k)) - e'_c(s_k) - W'$. This new method reduces the approximation error compared with the method in [27].

Avoidance of dynamic obstacles. We model the area occupied by the ego vehicle as the union of four circles that wrap the ego vehicle, as shown in Fig. 7. Circle $j \in \{1, 2, 3, 4\}$ at time k is denoted by $\mathcal{C}_j(x_k, y_k, \phi_k)$. Following the technique in [27], we wrap surrounding vehicles using probability ellipses, which are denoted as $\mathcal{E}_i(\mathbf{z}_k^i, a_k^i, b_k^i), (i = 1, 2, \ldots, n)$, where a_k^i and b_k^i are the major and minor axes of the ellipse of ith participant at time k. Readers may refer to Sec.IV.E of [27] for more details. For simplicity, we determine the ellipse based on velocity, as

$$a_k^i = a_{\min}^i + 3 \cdot (1 - e^{-0.05 \cdot v_k^i}) v_k^i \quad \text{(meters)}, \tag{15}$$

where $a_{\min}^i$ is the major axes of a minimum ellipse that can wrap the surrounding vehicle. The constraints of avoidance of dynamic obstacles can be represented as

$$\mathcal{C}_j(x_k, y_k, \phi_k) \cap \mathcal{E}_i(\mathbf{z}_k^i, a_k^i, b_k^i) = \emptyset, \tag{16}$$

for all $j \in \{1, 2, 3, 4\}$, $i \in \{1, 2, \ldots, n\}$, and $k \in \{1, 2, \ldots, m\}$.

3.2.3. *Overview*

By substituting (7), (14), and (16) into (4), we have the following problem:

$$\min_{\mathbf{u}_{0:m-1}} \sum_{k=0}^{m-1} (1 - e^{-\beta_5 k})[J_r(\mathbf{z}_k) + J_c(\mathbf{z}_k, \mathbf{u}_k) + J_s(\mathbf{z}_k)] + \beta_6 e^{\beta_5 k} J_h(\mathbf{u}_k, \mathbf{u}_0^h) \tag{17a}$$

$$\text{s.t.} \quad \mathbf{u}_{\min} \le \mathbf{u}_k \le \mathbf{u}_{\max} \tag{17b}$$

$$\mathbf{z}_{\min} \le \mathbf{z}_k \le \mathbf{z}_{\max} \tag{17c}$$

$$\mathbf{z}_{k+1} = f(\mathbf{z}_k, \mathbf{u}_k) \tag{17d}$$

Table 1. Main symbol.

Symbol	Description	Type
m	Maximum number of time steps	Constant positive integer
$\mathbf{z}_k$	State of ego vehicle at time k	Discrete 6-tuple
$\mathbf{z}_k^i$	Predicted state of ith surrounding vehicle at time k	Discrete 4-tuple
Q	Rows of sampling points used for generating spirals	$Q \in \{6, 7, 8, 9, 10\}$
ρ_q	Number of endpoints in qth row	Constant positive integer
w	The number of discrete points of the spiral	Constant positive integer

$$0 \le |v_k \dot{\phi}_k| \le g_{\max} \tag{17e}$$

$$B_r(s_k - e_l(s_k)) + w \le e'_c(s_k) \le B_l(s_k - e_l(s_k)) - w \tag{17f}$$

$$\mathcal{C}_j(x_k, y_k, \phi_k) \cap \mathcal{E}_i(\mathbf{z}_k^i, a_k^i, b_k^i) = \emptyset, \tag{17g}$$

where $k = 0, 1, \ldots, m$, $i = 1, 2, \ldots, n$, $j = 1, 2, 3, 4$, and other coefficients are in Table 1. Then we use FORCESpro [40] to solve problem (17).

4. Simulations

4.1. *Hyper-parameters and coefficients*

4.1.1. *Time step and horizon*

The number of time steps for the planner is $m = 30$. The first 10 time steps are 0.1 s, and the others are 0.2 s. The time horizon of the planner is 5 s.

4.1.2. *Coefficients in cost function*

Table 2 shows the coefficients in (3.1.2) and (17).

4.1.3. *The contour of the ego car and coefficients in constraints*

For simplicity, we assume that the ego and surrounding vehicles have the same shape and structure, where

Table 2. Coefficients in cost function.

Reference path planner					
Name	α_1	α_2	α_3	α_4	α_5
Value	12	5	0.5	0.4	0.4

Trajectory planner									
Name	β_1	β_{21}	β_{22}	β_{23}	β_3	β_{41}	β_{42}	β_5	β_6
Value	1	0.8	0.8	0.1	0.06	1	1	0.25	6

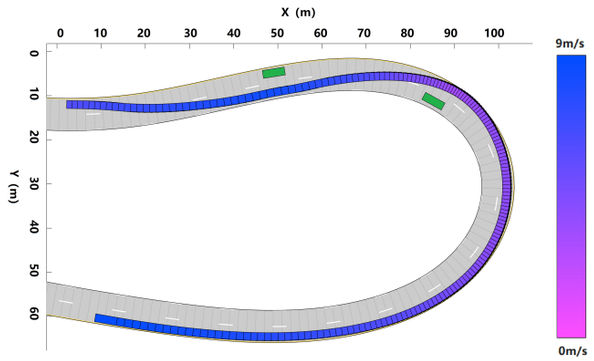


Fig. 9. Overtaking and lane changing simulation. The blue car is the ego vehicle in a gray driving area, and the green cars are stationary surrounding vehicles. This figure shows the trajectory of the ego vehicle every 0.2 s. The fading color shows the speeds of the ego vehicle, whose initial speed is 6 m/s.

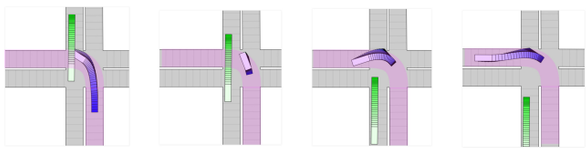
$l_f = 1.5\,\text{m}$, $l_r = 1.4\,\text{m}$, $L_v = 4.7\,\text{m}$, and $W_v = 1.8\,\text{m}$. The constraints $\mathbf{u}_{\min} \leq \mathbf{u}_k \leq \mathbf{u}_{\max}$ in Eq. (17) are $-10^\circ/s \leq \dot{\delta}_k \leq 10^\circ/s$, $-7\,\text{m/s}^2 \leq \dot{v}_k \leq 3\,\text{m/s}^2$. The constraints $\mathbf{z}_{\min} \leq \mathbf{z}_k \leq \mathbf{z}_{\max}$ in Eq. (17) are $-150\,\text{m} \leq x_k \leq 150\,\text{m}$, $-150\,\text{m} \leq y_k \leq 150\,\text{m}$, $0\,\text{m/s} \leq v_k \leq 16.67\,\text{m/s}$, $-180^\circ \leq \phi_k \leq 180^\circ$, $-36^\circ \leq \delta_k \leq 36^\circ$, $s_k \geq 0$. The constraints $0 \leq |v_k \dot{\phi}_k| \leq g_{\max}$ in Eq. (17) is $0 \leq |v_k \dot{\phi}_k| \leq 0.6\,\text{rad·m/s}$.

4.2. *Scenario I: Overtaking and lane changing*

We design the scenario to show the ability to overtake and lane change in a crooked two-lane road without human input. We assume that the vehicles always allow a lane change between the two lanes. The width of the two lanes is 3.75 m. The ego vehicle should slow down to prevent roll-over. We set stationary surrounding vehicles as obstacles in each lane ahead of the ego vehicle. The ego vehicle should choose the best lane to avoid the obstacles. The speed limit of the road is 12 m/s. As shown in Fig. 9, the ego vehicle overtakes the two stationary vehicles and slows down in a sharp turning area.

4.3. *Scenario II: Left turning*

We design the scenario to show the ability to avoid collision with dynamic obstacles when turning left. In this scenario, two straight roads meet at the intersection. Each road contains two lanes leading in the opposite direction. We assume that the vehicles are not allowed a lane change on either road. The width of each lane is 5 m. The ego vehicle turns left at the intersection. At the same time, a surrounding vehicle goes straight along the opposite driving lane at a constant speed. The ego vehicle needs to slow down or stop to wait for the other vehicle to move away. The speed limit of both roads is 12 m/s. Figure 10 shows the ego vehicle's generated trajectory and the surrounding vehicle's trajectory.

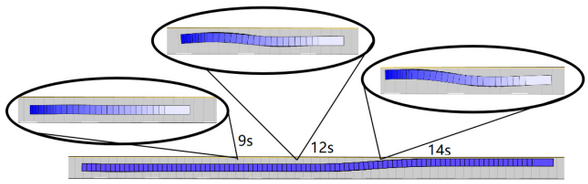


(a) (b) (c) (d)

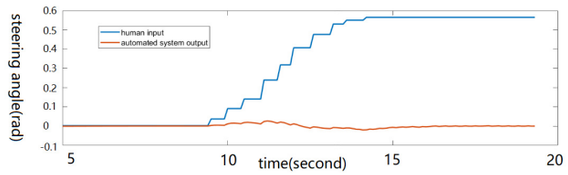
Fig. 10. Left turning simulation. The driving area is a light pink area. The blue car is the ego vehicle. The fading colors show the planning trajectories of vehicles in the next 5 s. The initial speed of the ego vehicle is 5 m/s, and the constant speed of the surrounding vehicle is 3 m/s.

4.4. *Scenario III: Shared control*

In the scenario, the ego vehicle goes along the center of the straight lane at a constant speed. The width of the lane is 5 m. The speed limit of the road is 6 m/s. The control input $\mathbf{u}_k = [\dot{\delta}_k, \dot{v}_k]$ remains zero, and the steering angle δ_k remains zero. We assume that the driver manipulates the steering wheel suddenly at a particular time, i.e. the driver imposes a dangerous step steering angle input. However, the vehicle will not collide with the boundary if the shared control system is active. We design the scenario to show the ability to avoid dangerous control input. As shown in Fig. 11(b), the driver turns the steering angle sharply to the left from 0° to 36° at the 9th s. The ego vehicle maintains the steering angle within a safe range and avoids colliding with



(a)



(b)

Fig. 11. (a) The trajectory of the ego vehicle. (b) The steering angle of the human input and automated system.

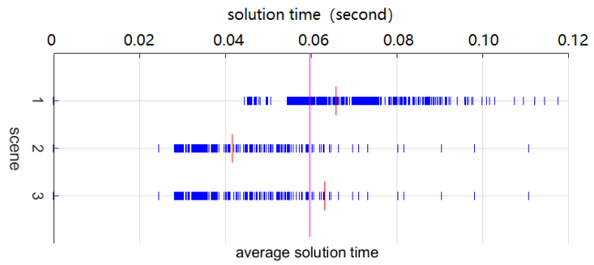


Fig. 12. Computation times for the proposed method. The solid red lines are the average solution times of the three scenarios. The solid pink line is the average solution time of the three scenarios.

the boundary though the human driver imposes a dangerous steering angle input. The ego vehicle stays in a safe area despite the driver's dangerous input, as shown in Fig. 11(a). The planning trajectories (in the black circles) show that the planner always expects the trajectory to be close to the reference path, whatever the human input and the initial state of the ego vehicle are.

4.5. *Summary*

Implement simulations using MATLAB R2021a on a desktop whose CPU is AMD-5600X (3.7 GHz). Figure 12 shows the running time. Our method computes reference paths and trajectories within 120 ms in all three scenarios. The average computation time for solving a hierarchical MPC model is 60 ms. We also test our method in CARLA-0.9.13 [41], of which the video is at https://www.bilibili.com/video/BV1CF411v7T3 or https://youtu.be/3vVXHvnWza8.

5. Conclusion

This paper proposed a hierarchical MPC-based motion planning for shared control of automated vehicles to generate a real-time trajectory in scenarios including multi-lane roads and multi-intersections. We have shown that the proposed method can generate safe trajectories in complex scenarios such as overtaking, lane changing, and turning left. Simulation tests have shown that our method can compute safe and comfortable trajectories for the future 5 s in less than 120 ms.

Future research includes the robustness of the proposed method in the presence of uncertainties in predicting other traffic participants and the numerical stability of the proposed method. Extensions of our method to more complex driving scenarios in mixed traffic flow containing fully automated vehicles and manual driving vehicles are also worth pursuing. Future research is verifying our methods on a real vehicle or a small-scale RC car.

Acknowledgment

This work was supported by the National Key R&D Program of China under Grant 2020AAA0108103, the National Natural Science Foundation of China under Grants 62133002, 62088101, 62073035, 62173034, and in part by the Chongqing Natural Science Foundation under Grant 2021ZX4100027.

References

[1] J. C. De Winter, R. Happee, M. H. Martens and N. A. Stanton, Effects of adaptive cruise control and highly automated driving on workload and situation awareness: A review of the empirical evidence, *Transp. Res. F: Traffic Psychol. Behav.* **27** (2014) 196–217.

[2] D. J. Fagnant and K. Kockelman, Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations, *Transp. Res. A: Policy Practice* **77** (2015) 167–181.

[3] J. Chen, J. Sun and G. Wang, From unmanned systems to autonomous intelligent systems, *Eng.* **12**(5) (2022) 16–19022.

[4] Y. Ma, Z. Wang, H. Yang and L. Yang, Artificial intelligence applications in the development of autonomous vehicles: A survey, *IEEE/CAA J. Autom. Sin.* **7**(2) (2020) 315–329.

[5] Y. Ding, B. Xin and J. Chen, A review of recent advances in coordination between unmanned aerial and ground vehicles, *Unmanned Syst.* **9**(2) (2021) 97–117.

[6] S. Cai, H. Ding, Y. Hu, L. Zhang, Q. Li and H. Chen, Open-source dataset of vehicle state for an electric vehicle on a low-adhesion road, *Sci. China Inf. Sci.* **65** (2022) 137201.

[7] Y. Li, X. Wang, J. Sun, G. Wang and J. Chen, Data-driven consensus control of fully distributed event-triggered multi-agent systems, *Sci. China Inf. Sci.* **66**(5) (2022) 152202, doi: 10.1007/s11432-022-3629-1.

[8] L. Guo, B. Gao, Y. Gao and H. Chen, Optimal energy management for HEVs in eco-driving applications using bi-level MPC, *IEEE Trans. Intell. Transp. Syst.* **18**(8) (2017) 2153–2162.

[9] H. Guo, C. Shen, H. Zhang, H. Chen and R. Jia, Simultaneous trajectory planning and tracking using an MPC method for cyber-physical systems: A case study of obstacle avoidance for an intelligent vehicle, *IEEE Trans. Industr. Inform.* **14**(9) (2018) 4273–4283.

[10] P. J. Durst, C. T. Goodin, C. L. Bethel, D. T. Anderson, D. W. Carruth and H. Lim, A perception-based fuzzy route planing algorithm for autonomous unmanned ground vehicles, *Unmanned Syst.* **6**(4) (2018) 251–266.

[11] X. Wang, J. Sun, F. Deng, G. Wang and J. Chen, Event-triggered consensus control of heterogeneous multi-agent systems: Model- and data-based approaches, *Sci. China Inf. Sci.* (2023) 1–13, doi: 10.1007/s11432-022-3683-y.

[12] D. Miller, A. Sun, M. Johns, H. Ive, D. Sirkin, S. Aich and W. Ju, Distraction becomes engagement in automated driving, in *Proc. Human Factors and Ergonomics Society Annual Meeting*, Vol. 59, No. 1, 2015, pp. 1676–1680, doi: 10.1177/1541931215591362.

[13] D. A. Abbink, T. Carlson, M. Mulder, J. C. De Winter, F. Aminravan, T. L. Gibo and E. R. Boer, A topology of shared control systems finding common ground in diversity, *IEEE Trans. Hum. Mach. Syst.* **48**(5) (2018) 509–525.

[14] L. Chen, X. Hu, W. Tian, H. Wang, D. Cao and F.-Y. Wang, Parallel planning: A new motion planning framework for autonomous driving, *IEEE/CAA J. Autom. Sin.* **6**(1) (2019) 236–246.

[15] F.-Y. Wang, N.-N. Zheng, D. Cao, C. M. Martinez, L. Li and T. Liu, Parallel driving in CPSS: A unified approach for transport automation and vehicle intelligence, *IEEE/CAA J. Autom. Sin.* **4**(4) (2017) 577–587.

[16] M. Marcano, S. Díaz, J. Pérez and E. Irigoyen, A review of shared control for automated vehicles: Theory and applications, *IEEE Trans. Hum. Mach. Syst.* **50**(6) (2020) 475–491.
[17] W. Wang, X. Na, D. Cao, J. Gong, J. Xi, Y. Xing and F.-Y. Wang, Decision-making in driver-automation shared control: A review and perspectives, *IEEE/CAA J. Autom. Sin.* **7**(5) (2020) 1289–1307.
[18] S. Ahlberg, A. Axelsson, P. Yu, W. S. Cortez, Y. Gao, A. Ghadirzadeh, G. Castellano, D. Kragic, G. Skantze and D. V. Dimarogonas, Co-adaptive human-robot cooperation: Summary and challenges, *Unmanned Syst.* **10**(2) (2022) 187–203.
[19] L. Zhang, H. Chen, Y. Huang, P. Wang and K. Guo, Human-centered torque vectoring control for distributed drive electric vehicle considering driving characteristics, *IEEE Trans. Veh. Technol.* **70**(8) (2021) 7386–7399.
[20] X. Lin, Y.-J. Yu, S.-Z. Chen and Y.-Y. Shi, An improved APF method for complex and dynamic obstacles avoidance, *Unmanned Syst.* **11**(2) (2023) 175–189, https://doi.org/10.1142/S2301385023410054.
[21] Q. Dong and J. Zhang, Distributed cooperative complete coverage path planning in an unknown environment based on a heuristic method, *Unmanned Syst.*, https://doi.org/10.1142/S2301385024500109.
[22] S. J. Anderson, S. B. Karumanchi, K. Iagnemma and J. M. Walker, The intelligent copilot: A constraint-based approach to shared-adaptive control of ground vehicles, *IEEE Intell. Transp. Syst. Mag.* **5**(2) (2013) 45–54.
[23] F. Borroni and M. Tanelli, A weighting approach to the shared-control of lateral vehicle dynamics, *IFAC-PapersOnLine* **51**(9) (2018) 305–310.
[24] M. Li, H. Cao, X. Song, Y. Huang, J. Wang and Z. Huang, Shared control driver assistance system based on driving intention and situation assessment, *IEEE Trans. Industr. Inform.* **14**(11) (2018) 4982–4994.
[25] C. Sentouh, A.-T. Nguyen, M. A. Benloucif and J.-C. Popieul, Driver-automation cooperation oriented approach for shared control of lane keeping assist systems, *IEEE Trans. Control Syst. Technol.* **27**(5) (2019) 1962–1978.
[26] C. Guo, C. Sentouh, J.-C. Popieul and J.-B. Hau, Predictive shared steering control for driver override in automated driving: A simulator study, *Transp. Res. F: Traffic Psychol. Behav.* **61** (2019) 326–336.
[27] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman and D. Rus, Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model, *IEEE Trans. Intell. Transp. Syst.* **19**(9) (2017) 2994–3008.
[28] S. M. Erlien, S. Fujita and J. C. Gerdes, Shared steering control using safe envelopes for obstacle avoidance and vehicle stability, *IEEE Trans. Intell. Transp. Syst.* **17**(2) (2015) 441–451.
[29] Y. Xu, H. Zheng, W. Wu and J. Wu, Robust hierarchical model predictive control for trajectory tracking with obstacle avoidance, *IFAC-PapersOnLine* **53**(2) (2020) 15 745–15 750.
[30] K. Yuan, H. Shu, Y. Huang, Y. Zhang, A. Khajepour and L. Zhang, Mixed local motion planning and tracking control framework for autonomous vehicles based on model predictive control, *IET Intell. Transp. Systems* **13**(6) (2019) 950–959.
[31] Y. Huang, H. Wang, A. Khajepour, H. Ding, K. Yuan and Y. Qin, A novel local motion planning framework for autonomous vehicles based on resistance network and model predictive control, *IEEE Trans. Veh. Technol.* **69**(1) (2019) 55–66.
[32] T. Novi, A. Liniger, R. Capitani and C. Annicchiarico, Real-time control for at-limit handling driving on a predefined path, *Veh. Syst. Dyn.* **58**(7) (2020) 1007–1036.
[33] J. L. Vázquez, M. Brühlmeier, A. Liniger, A. Rupenyan and J. Lygeros, Optimization-based hierarchical motion planning for autonomous racing," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IEEE, 2020), pp. 2397–2403.
[34] A. Liniger, A. Domahidi and M. Morari, Optimization-based autonomous racing of 1:43 scale RC cars, *Optim. Control Appl. Methods* **36**(5) (2015) 628–647.
[35] Q. Wang, T. Weiskircher and B. Ayalew, Hierarchical hybrid predictive control of an autonomous road vehicle, in *Dynamic Systems and Control Conf.* (American Society of Mechanical Engineers, 2015), No. DSCC2015-9773.
[36] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li and Q. Kong, Baidu Apollo EM motion planner, arXiv:1807.08048, 2018.
[37] M. McNaughton, C. Urmson, J. M. Dolan and J.-W. Lee, Motion planning for autonomous driving with a conformal spatiotemporal lattice, in *IEEE Int. Conf. Robotics and Automation* (IEEE, 2011), pp. 4889–4895.
[38] W. Xu, J. Wei, J. M. Dolan, H. Zhao and H. Zha, A real-time motion planner with trajectory optimization for autonomous vehicles, in *IEEE Int. Conf. Robotics and Automation* (IEEE, 2012), pp. 2061–2067.
[39] K. Ichida, F. Yoshimoto and T. Kiyono, Curve fitting by a piecewise cubic polynomial, *Computing* **16**(4) (1976) 329–338.
[40] A. Zanelli, A. Domahidi, J. Jerez and M. Morari, FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs, *Int. J. Control* **93**(1) (2020) 13–29.
[41] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, CARLA: An open urban driving simulator, in *Proc. 1st Annual Conf. Robot Learning*, 2017, pp. 1–16.

Zijun Cheng received his B.S. degree in Automation and his M.S. degree in Control Science and Engineering from the Beijing Institute of Technology, China, in 2020 and 2023, respectively. From 2023 until now, he is a Ph.D. student at the Beijing Institute of Technology. His research interests include planning and shared control of automated vehicles, management and control of vehicle platoons, and social behavior of vehicles.

Xianlin Zeng received his M.S. degree in Control Science and Engineering from the Harbin Institute of Technology, China, and his Ph.D. degree in Mechanical Engineering from the Texas Tech University, USA, in 2011 and 2015, respectively. He is currently an associate professor in the National Key Laboratory of Autonomous Intelligent Unmanned Systems, School of Automation, Beijing Institute of Technology, Beijing, China.

Xianlin Zeng is the author of over 40 technical publications and proceedings. His current research interests include planning and control of autonomous systems, and distributed optimization, and noncooperative games.

Hao Fang received the B.S. degree in automation from the Xi'an University of Technology, Xi'an, China, in 1995, and the M.S. and Ph.D. degrees in control theory and application from Xi'an Jiaotong University, Xi'an, in 1998 and 2002, respectively. He held two postdoctoral appointments with the INRIA/France research group of COPRIN and with the Laboratoire des Sciences et Matériaux pour l' Electronique et d'Automatique (UNR6602 CNRS/ Blaise Pascal University, Clermont-Ferrand, France). Since 2011, he has been a Professor with the Beijing Institute of Technology, Beijing, China.

His research interests include all-terrain mobile robots, robotic control, and parallel manipulators.

Gang Wang received a B.Eng. degree in Automatic Control in 2011, and a Ph.D. degree in Control Science and Engineering in 2018, both from the Beijing Institute of Technology, Beijing, China. He also received a Ph.D. degree in Electrical and Computer Engineering from the University of Minnesota, Minneapolis, USA, in 2018, where he stayed as a postdoctoral researcher until July 2020. Since August 2020, he has been a professor with the School of Automation at the Beijing Institute of Technology. His research interests focus on the areas of signal processing, control, and reinforcement learning with applications to autonomous systems. He was the recipient of the Best Paper Award from the Frontiers of Information Technology & Electronic Engineering in 2021, the Excellent Doctoral Dissertation Award from the Chinese Association of Automation in 2019, the Best Student Paper Award from the 2017 European Signal Processing Conference, and the Best Conference Paper at the 2019 IEEE Power & Energy Society General Meeting. He is currently on the editorial board of Signal Processing and IEEE Transactions on Signal and Information Processing over Networks.

Lihua Dou received the B.S., M.S., and Ph.D. degrees in control theory and control engineering from the Beijing Institute of Technology, Beijing, China, in 1979, 1987 and 2001, respectively. She is currently a Professor of Control Science and Engineering with the Beijing Institute of Technology.

Her current research interests include command and control, pattern recognition, and *ad hoc* networks.