

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/340516864>

Enable faster and smoother spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment

Article in *Proceedings of the Institution of Mechanical Engineers Part D Journal of Automobile Engineering* · April 2020

DOI: 10.1177/0954407020906627

CITATIONS

5

READS

956

7 authors, including:



Shengbo Eben Li
Tsinghua University

255 PUBLICATIONS 8,167 CITATIONS

[SEE PROFILE](#)



Jianyu Chen
University of California, Berkeley

36 PUBLICATIONS 450 CITATIONS

[SEE PROFILE](#)



Yang Guan
Tsinghua University

28 PUBLICATIONS 125 CITATIONS

[SEE PROFILE](#)



Masayoshi Tomizuka
University of California, Berkeley

1,032 PUBLICATIONS 27,542 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:




Real time optimization for motion planning [View project](#)



robust iterative learning control [View project](#)

Enable Faster and Smoother Spatio-temporal Trajectory Planning for Autonomous Vehicles in Constrained Dynamic Environment

Proc IMechE Part D:
J Automobile Engineering
XX(X):1–10
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/


Long Xin¹, Yiting Kong¹, Shengbo Eben Li¹, Jianyu Chen², Yang Guan¹, Masayoshi Tomizuka², Bo Cheng¹

Abstract

Trajectory planning is of vital importance to decision making for autonomous vehicles. Currently, there are three popular classes of cost-based trajectory planning methods: sampling-based, graph-search-based, and optimization-based. However, each of them has its own shortcomings, e.g., high computational expense for sampling-based methods, low resolution for graph-search-based methods, and lack of global awareness for optimization-based methods. It leads to one of the challenges for trajectory planning for autonomous vehicles which is improving planning efficiency while guaranteeing model feasibility. Therefore, this paper proposes a hybrid planning framework composed of two modules, which preserves the strength of both graph-search-based methods and optimization-based methods, thus enabling faster and smoother spatio-temporal trajectory planning in constrained dynamic environment. The proposed method first constructs spatio-temporal driving space based on directed acyclic graph (DAG) and efficiently searches a spatio-temporal trajectory using the improved A* algorithm. Then taking the search result as reference, locally convex feasible driving area is designed and model predictive control (MPC) is applied to further optimize the trajectory with a comprehensive consideration of vehicle kinematics and moving obstacles. Results simulated in four different scenarios all demonstrated feasible trajectories without emergency stop or abrupt steering change, which is kinematic-smooth to follow. Moreover, the average planning time was 31ms, which only took 59.05%, 18.87%, and 0.69%, respectively of that consumed by other state-of-the-art trajectory planning methods, namely MIDP, SAIO, and GADP.

Keywords

Spatio-temporal trajectory planning, directed acyclic graph, convex feasible set, autonomous driving, dynamic environment

1. Introduction

Trajectory planning is of great importance in decision making for autonomous driving. It generates the collision-free trajectory towards the destination with the measurements from perception and localization [1]. Then the controller uses a sequence of motion-level commands to guide the autonomous vehicle along the trajectory. Planning efficiency and trajectory feasibility have a great influence on tracking performance, such as safety, comfort, and economy [2] [3]. If the trajectory is spatio-temporal rather than purely geometrical, those objectives can be better guaranteed in constrained dynamic environment [4] [5]. Moreover, the conflict-free cooperative control for multiple connected vehicles also benefits from the fast generation of spatio-temporal trajectory [6]. Current methods for spatio-temporal trajectory planning are broadly classified into three main categories: the sampling-based methods, the graph-search-based methods and the optimization-based methods.

Sampling-based methods firstly connect the points sampled randomly or deterministically and select the desired trajectory among a series of candidates by collision check and performance evaluation. The spatio-temporal trajectory is generated with samples drawn in action space [7]. LaValle

proposed a random sampling-based method called Rapidly-exploring Random Tree (RRT), which was characterized by high planning efficiency but poor trajectory feasibility. [8]. Kuwata *et al.* used a closed-loop system to improve the stability of standard RRT, where planning feasibility is guaranteed by simulations with the low-level controller in advance [9]. The deterministic sampling-based method proposed by Macfarlane *et al.* sampled in a predetermined acceleration range, resulting in jerk-bounded trajectories [10]. Gu *et al.* further improved the smoothness of the spatio-temporal trajectory by branching over the polynomial speed profiles sampled deterministically in a short-term [11]. However, in the case of small sample size, the sampling-based methods can generate trajectories far from

¹ State Key Lab of Automotive Safety and Energy, School of Vehicle and Mobility, Tsinghua University, Beijing 100084, China (e-mail: {xin113, kyt18, guany17}@mails.tsinghua.edu.cn, {lishbo, chengbo}@tsinghua.edu.cn)

² Department of Mechanical Engineering, University of California, Berkeley, CA 94720, USA (e-mail: {jianyuchen, tomizuka}@berkeley.edu)
L. Xin and YT. Kong contributed equally to this work.

Corresponding author:
Shengbo Eben Li

the global optimum. Otherwise, the planning is intractable time-consuming due to the curse of dimensionality. Besides, the collision check does not actually consider the dynamic obstacles parameterized by time, probably leading to poor feasibility of the spatio-temporal trajectory.

Graph-search-based methods search the desired trajectory under some performance requirements on a discretized map. Originally, those search methods can only deal with static obstacles. But currently, more and more researches have focused on the generation of the spatio-temporal trajectory with a consideration of dynamic obstacles in the future. State lattice proposed by Pivtoraiko is the most widely used discrete graphs composed of kinematically-feasible motion primitives and time information [12]. The spatio-temporal trajectory can be obtained by using classical graph-search-based methods in such a state lattice [13]. The Dijkstra algorithm is widely used for finding the shortest trajectory [14]. A*, an extension of the Dijkstra algorithm, is a fast heuristic search algorithm and the basis of many implementations, such as the Junior (hybrid A*) [15] and Boss (AD*) [16] autonomous vehicles. To search more efficiently in a changing environment, D* [17] and D* Lite [18] are proposed to replan the trajectory using previous information. Although the graph-search-based methods generate the trajectory near the global optimum, its high order derivatives are still discontinuous. However, the creation of state lattice takes a lot of time, leading to low planning efficiency [19].

Optimization-based methods are concerned with the mathematical problem-solving to get the minimal cost or objective function with multiple constraints. The trajectory is iteratively refined until the termination or the convergence conditions are met [20]. Shibata *et al.* employed the potential field approach to generate the obstacle-avoidance trajectory based on velocity vectors [21]. They determined the optimal parameter values for the velocity potential function by a numerical optimization method called quasi-Newton. Keller *et al.* represented the spatio-temporal trajectory by a Timed Elastic Band (TEB) function and optimized it in constrained action space [22]. They formulated the problem as a nonlinear least-squares problem and solved it by the Levenberg-Marquardt algorithm, a numerical optimization method. Gao *et al.* applied a local optimization-based method called MPC to smooth a coarse predefined trajectory [23]. They minimized the cost function by Sequential Quadratic Programming (SQP), where the problem was divided into a sequence of QP subproblems with linearized constraints. However, the optimization-based methods are often time-consuming, especially with non-convex environmental constraints.

Overall, current researches on trajectory planning provide insightful progress but still have limitations. Sampling-based methods can alleviate local optimality problem by exploring the whole space and finding a global optimum. However, it suffers from curse of dimensionality and high computational expense, leading to resolution suboptimality. Graph-search-based methods can effectively generate the trajectory near the global optimum, but it is usually not curvature continuous and doesn't satisfy model feasibility for trajectory execution. Optimization-based methods can guarantee feasibility with model information and are computationally acceptable by

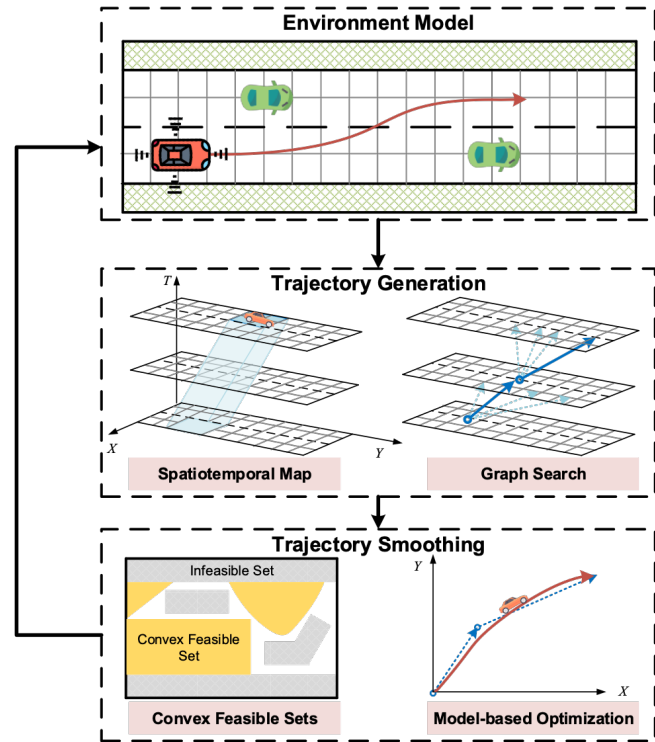


Figure 1. Framework of the proposed planning method

finding local optimum through a few iterations. However, the predetermined reference trajectory is required and needs properly design and initialization.

Therefore, this paper proposes a novel hybrid planning framework composed of two modules, which preserves strength of both graph-search-based methods for avoiding local optimality and optimization-based methods for local refinement, thus enabling faster and smoother spatio-temporal trajectory planning in constrained dynamic environment. The proposed method first constructs spatio-temporal driving space based on directed acyclic graph (DAG) and efficiently searches a spatio-temporal trajectory using the improved A* algorithm. Then taking the search result as reference, locally convex feasible driving area is designed and model predictive control (MPC) is applied to further optimize the trajectory with a comprehensive consideration of vehicle kinematics and moving obstacles.

The rest of the paper is organized as follows: Section II shows the framework of the proposed planning method. Section III introduces the reference trajectory generation. Section IV introduces the trajectory smoothing. Section V presents the simulation setup in detail. In section VI presents the results for overtaking and yielding. Section VII concludes this paper with a summary of the contributions and perspectives.

II. Proposed Method

To generate the smooth spatio-temporal trajectory efficiently in constrained dynamic environment, a hybrid planning method is proposed in this paper. As shown in Fig. 1, its framework consists of two procedures: reference trajectory generation with a search-based method and local refinement and smoothing with an optimization-based method.

- (i) to generate a reference trajectory with a search-based method. The driving map is represented by a 3D spatio-temporal map instead of the general grid map, which is constructed by DAG. The position of dynamic obstacles in the future can be illustrated in such a driving map. Then, the improved A* algorithm is used to search in action space to efficiently get the best collision-free trajectory depending on the designed performance requirements.
- (ii) to smooth the reference trajectory with an optimization-based method. The reference trajectory is not smooth enough due to the discrete action space and state space. Therefore, MPC is thus adopted for further optimization. However, the presence of dynamic obstacles generally brings non-convex feasible driving areas for the ego vehicle. It increases the computational difficulty and influences the whole planning efficiency. Then the method for reconstruction of convex feasible sets is presented previously to ensure a fast optimization. A kinematic model is introduced to make sure the high order derivatives of trajectory are continuous and accordant with the vehicle kinematics. Overall, the optimized trajectory is smooth enough, leading to good tracking performance and reasonable control cost.

III. Reference Trajectory Generation

This section presents the generation of the reference trajectory, which contains the spatio-temporal information towards the destination in dynamic environment. The driving map represented by a 3D spatiotemporal map is firstly established with the help of DAG. Then, the improved A* algorithm is used to search in action space to efficiently get the best collision-free trajectory on the driving map.

Spatio-temporal Map Construction

The spatio-temporal trajectory provides the ego vehicle not only the position but also the kinematic information, leading to the good tracking performance of the vehicle controller. To generate a reasonable trajectory in the presence of moving obstacles, their positions are necessary to be parameterized by time. However, graph-search-based algorithms generally do not take into account their positions in the future due to the lack of temporal information. Therefore, a timeline is added into the grid map, then the 3D spatiotemporal map is established with the help of DAG.

DAG consists of the points and the edges directed from one point to another along which it is impossible for any point to loop back to itself [19] [24]. A sequence of points in such topological ordering represents the change of vehicle states over time, namely the spatio-temporal trajectory. Then the grid map with directed acyclic edges connecting the neighboring points is exactly a DAG.

Firstly, the 2D spatial grid Map is established by dividing the environment space into finitely many squares with the same size. As shown in Fig.2, the ego vehicle is considered as a mass point occupying the grid where its geometric center is located. The grids occupied by other vehicles take the vehicle size into account so that the lateral and longitudinal distances from the ego vehicle are large enough to avoid collisions.

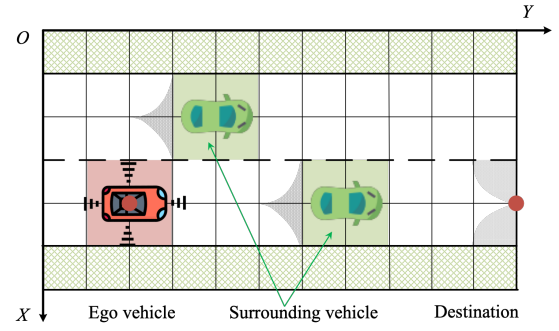


Figure 2. 2D spatial grid Map for a two-lane road

To improve search efficiency, there are some preliminaries concerned with the elimination of useless squares. It is impossible for the ego vehicle to go through those areas without collision even with the maximum steering angle. For example, the circular areas following the neighboring vehicles and those before the destination, which are depicted as grey-shaded parts. The radius of the areas, namely the minimum turning radius for the ego vehicle, is tangent to the direction of vehicle velocity.

The 2D spatial grid map is extended along the added timeline, resulting in a 3D spatio-temporal map. Its multiple map layers are parallel to each other as shown in Fig.3. Several directed edges connect the two states in neighboring map layers at different time steps. The edge is in a single direction as time goes forward, which meets the requirements of DAG. In Fig.3, a sequence of points connected by several solid blue lines illustrate the spatio-temporal trajectory discretized with a time step size Δt . Such a trajectory contains both the position and time information. Specifically, it can represent a stop at a specific time, while it is impossible for general grid maps.

In the 3D spatio-temporal map, the positions for the dynamic obstacle at different time step are depicted as the light green part in Fig.3. The future position of surrounding vehicles is assumed to be given by trajectory prediction, which is another vital module in decision-making system [25]. Then, the proposed planning method can take the dynamic unfeasible region in the future into account, while

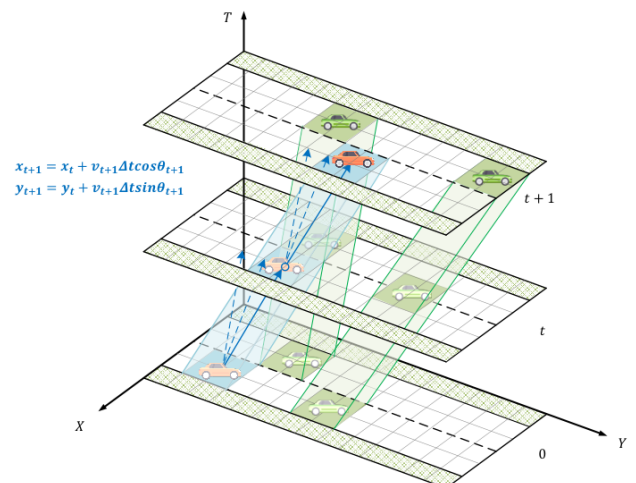


Figure 3. 3D spatio-temporal Map constructed by DAG

the classical graph-search-based methods based on the grid map only consider the static obstacles in each search step. The reference trajectory without collisions is thus feasible in a longer time horizon. Since the good suitability and scalability of the 3D spatio-temporal map established offline, only the unfeasible squares are specified from scenarios to scenarios. The unfeasible squares can be eliminated quickly so that the increase of obstacle numbers has little impact on overall processing time.

Moreover, the established 3D spatio-temporal map contains the information not only about the vehicle states s but also about the actions A , which are depicted as the blue directed lines. The slope of its projection in plan XOT denotes the lateral velocity $v_x = \frac{\Delta l_x}{\Delta t}$, while the slope of its projection in plan YOT denotes the longitudinal velocity $v_y = \frac{\Delta l_y}{\Delta t}$. The angle between its projection on the plan XOY and the axis Y approximates the yaw angle. Those actions are calculated depending on the determined neighboring states. In turn, the possible states can be expanded from the current state if the action space is determined (the blue solid line and the dotted line in Fig.3). Note that it is possible to adopt different curves to connect the two states at adjacent map layers. Since the time step is assumed to be short enough, the velocity is assumed constant. Then, the straight lines are used in this paper and the spatio-trajectory of the ego vehicle is depicted as the light blue part.

Trajectory Search Algorithm

One of the graph-search-based methods called A* algorithm is introduced to find the best reference path because of its high efficiency. It usually deals with situations with fixed obstacles [26]. The basic A* algorithm used for a grid configuration space is restricted to the spatial trajectory. Then, in this paper, the A* algorithm is improved to be applicable on the established 3D spatio-temporal map.

In this paper, the vehicle states are the collection of the lateral and longitudinal position x, y , velocity v and yaw angle θ , which is denoted as $s_t = \{x_t, y_t, v_t, \theta_t\}$. The action space includes possible acceleration and yaw angle, which is denoted as $\mathcal{A} = \{A, \Theta | A = \cup_i a_i, \Theta = \cup_i \theta_i\}$. The expansion for ego vehicle states is formulated in the equation (1).

$$\begin{aligned} \theta_{t+1} &= \theta_t + \theta_i \\ v_{t+1} &= v_t + a_i \times \Delta t \\ x_{t+1} &= x_t + v_{t+1} \times \Delta t \times \cos \theta_{t+1} \\ y_{t+1} &= y_t + v_{t+1} \times \Delta t \times \sin \theta_{t+1} \end{aligned} \quad (1)$$

During the search process, the closed set at each time step stores not only the nodes that have been selected to the desired trajectory but also the nodes that are currently unreachable for the ego vehicle. And the open set stores the candidate nodes waiting to be examined.

The A* algorithm evaluates each point in the open set by an estimated path cost function, which is defined in equation (2):

$$Cost(s_t) = J(s_t) + H(s_t) \quad (2)$$

where $J(s_t)$ is the accumulated cost to move from the initial state to the state s_t , $H(s_t)$ is a heuristic function

Algorithm 1: Improved A* Algorithm for Spatio-temporal Trajectory Search

Input: A spatio-temporal DAG map

Output: The optimal spatio-temporal trajectory from initial position to destination

```

1 Initialize the trajectory set with the initial point;
2 Initialize the open set with the initial point;
3 Construct the closedi set at different time step according
  to the dynamic obstacles;
4 while the open set is not empty do
5   Estimate the Cost of each node in the open set with
     (2);
6   Take the node current with the lowest Cost as the
     vehicle states  $s_t$ ;
7   Add the states  $s_t$  to the trajectory set;
8   if The selected node current is the destination then
9     The trajectory is found successfully;
10  else
11    Remove the current node from the open set;
12    Generate the neighbouring nodes in the next
       map layer with (1);
13    for each neighbouring node do
14      if it is in the closedt+1 then
15        ignore it
16      else
17        if it is not in the open set then
18          Add it to the open set
19        end
20      end
21    end
22  end
23   $t=t+1$ ;
24 end
25 return

```

that estimates the cost to move from s_t to destination. The A* algorithm introduces heuristic information into the path cost function to provide the expansion of vehicle states with direction, reducing the total number of nodes to expand.

At each iteration, the node with the lowest cost of the open set is selected to expand. Then, it is removed from the open set and added into the closed set at the next step. The reference trajectory is generated by repeating selections until the destination point is selected. The search space in this paper is the action space \mathcal{A} rather than the state space s , which reduces the search area a lot. Then, as formulated in equation (1), the node is expanded along the discretized yaw angle in action space Θ , and the next position is determined depending on the velocity calculated from the acceleration in action space A . The detailed implementation of the A* algorithm is described in the following Algorithm 1.

IV. Trajectory Smoothing

A smooth spatio-temporal trajectory leads to small tracking errors and high riding comfort for autonomous vehicles [27]. As stated in the previous section, the improved A* algorithm efficiently finds the best trajectory under some performance requirements. However, both the state and the

action space are highly discrete, which makes the spatio-temporal trajectory not smooth enough for the ego vehicle to follow. Therefore, MPC based on the vehicle kinematic model is thus adopted for further optimization to make sure its high order derivatives are continuous and accordant with the vehicle kinematics. The challenge of the application in real-time is how to deal with dynamic constraints as moving obstacles in mixed traffic flow generally bring non-convex feasible driving areas for the ego vehicle. In this paper, our previous research is introduced to reconstruct the convex feasible sets to accelerate solving the optimization problem. The details can be found in [28].

Convex Feasible Set Design

The whole feasible sets Γ for the ego vehicle is the intersection of supersets Γ_i , which is formulated as:

$$\Gamma := \bigcap_i^N \Gamma_i \quad (3)$$

where Γ_i is the feasible set depending on the constraint of each surrounding vehicle. The augmented vehicle state on the basis of the search results is denoted as the reference state \mathbf{X}^r . A convex feasible superset belongs to the feasible set according to each constraint $\mathcal{F}_i(\mathbf{X}^r) \in \Gamma_i$. Then, the whole convex feasible set for a given reference state is calculated by:

$$\mathcal{F}(\mathbf{X}^r) := \bigcap_i \mathcal{F}_i(\mathbf{X}^r) \quad (4)$$

where each convex feasible superset around the reference trajectory \mathbf{X}^r can be constructed according to the following rules inspired from the previous work [29]:

(i) If Γ_i is convex:

$$\mathcal{F}_i(\mathbf{X}^r) = \Gamma_i \quad (5)$$

(ii) If the complement of Γ_i is convex, its signed distance function ϕ for obstacles can be designed to be convex:

$$\phi_i(\mathbf{X}) \geq \phi_i(\mathbf{X}^r) + \Delta\phi_i(\mathbf{X}^r) \times (\mathbf{X} - \mathbf{X}^r) \quad (6)$$

Then the convex feasible superset is defined as:

$$\mathcal{F}_i(\mathbf{X}^r) := \{\mathbf{X} | \phi_i(\mathbf{X}^r) + \nabla\phi_i(\mathbf{X}^r) \times (\mathbf{X} - \mathbf{X}^r) \geq 0\} \quad (7)$$

(iii) If neither Γ_i nor its complement $R \setminus \Gamma_i$ is convex, function ϕ_i is neither convex nor concave. An auxiliary function is adopted as:

$$\tilde{\phi}_i(\mathbf{X}) := \phi_i(\mathbf{X}) + \frac{1}{2}(\mathbf{X} - \mathbf{X}^r)^T H_i(\mathbf{X} - \mathbf{X}^r) \quad (8)$$

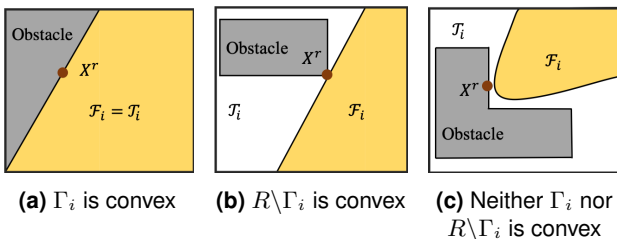


Figure 4. Three types of convex feasible sets

where $\tilde{\phi}_i(\mathbf{X})$ is convex so that:

$$\begin{aligned} \tilde{\phi}_i(\mathbf{X}) &\geq \tilde{\phi}_i(\mathbf{X}^r) + \nabla\tilde{\phi}_i(\mathbf{X}^r) \times (\mathbf{X} - \mathbf{X}^r) \\ &= \phi_i(\mathbf{X}^r) + \nabla\phi_i(\mathbf{X}^r) \times (\mathbf{X} - \mathbf{X}^r) \end{aligned} \quad (9)$$

Then the convex feasible superset is defined as:

$$\begin{aligned} \mathcal{F}_i(\mathbf{X}^r) &:= \{\mathbf{X} | \phi_i(\mathbf{X}^r) + \nabla\phi_i(\mathbf{X}^r) \times (\mathbf{X} - \mathbf{X}^r) \\ &\geq \frac{1}{2}(\mathbf{X} - \mathbf{X}^r)^T H_i(\mathbf{X} - \mathbf{X}^r)\} \end{aligned} \quad (10)$$

As shown in Fig.4, all the constraints of dynamic obstacles to reference trajectory can be reconstructed as the above-mentioned three types of convex feasible sets, which guarantees high efficiency in the following optimization step.

Trajectory Optimization Method

The reference trajectory is optimized by MPC, and the problem is formulated as:

$$\begin{aligned} \min_{\mathbf{U}_n} \quad & \sum_{n=1}^T \omega_1 \times \|\mathbf{X}_n - \mathbf{X}_n^r\|_Q^2 + \omega_2 \times \|\mathbf{U}_n\|_R^2 \\ \text{s.t.} \quad & \mathbf{X}_0 = \mathbf{X}^{\text{start}}, \mathbf{X}_N = \mathbf{X}^{\text{target}} \\ & \mathbf{X}_n = \mathbf{X}_{n-1} + F(\mathbf{X}_{n-1}, \mathbf{U}_{n-1}) \times \Delta t \\ & \mathbf{U}_{\min} \leq \mathbf{U}_n \leq \mathbf{U}_{\max} \\ & \phi(\mathbf{X}_n) \geq 0, \forall n = 1, \dots, N-1 \end{aligned} \quad (11)$$

where, T denotes the prediction horizon, $\mathbf{X}^{\text{start}}$ denotes the initial position, $\mathbf{X}^{\text{target}}$ denotes the destination, \mathbf{X}_n^r denotes the reference state of the spatio-temporal trajectory, \mathbf{U}_n denotes the control input with limits of $\mathbf{U}_{\min}, \mathbf{U}_{\max}$, Q, R are the weighting matrices for each component of states and control inputs, ω_1, ω_2 are the weights on two different objectives, $F(\cdot)$ is the vehicle state-space model and $\phi(\cdot)$ denotes the state constraints for each state.

Expand the nonlinear function $F(\mathbf{X}_n, \mathbf{U}_n)$ as a Taylor series around the reference point:

$$\begin{aligned} F(\mathbf{X}_n, \mathbf{U}_n) &:= F(\mathbf{X}_n^r, \mathbf{U}_n) + \\ &\quad \text{Jac}_F(\mathbf{X}_n, \mathbf{U}_n) \times (\mathbf{X}_n - \mathbf{X}_n^r) \end{aligned} \quad (12)$$

where, $\text{Jac}_F(\cdot)$ is the Jacobian matrix of $F(\cdot)$. Then, the state-space model is linearized as:

$$\begin{aligned} \mathbf{X}_n &= \mathbf{X}_{n-1} + F(\mathbf{X}_{n-1}, \mathbf{U}) \times \Delta t \\ &\triangleq \mathbf{A}_n \mathbf{X}_{n-1} + \mathbf{B}_n \mathbf{U}_{n-1} + \mathbf{C}_n \end{aligned} \quad (13)$$

where $\mathbf{A}_n, \mathbf{B}_n, \mathbf{C}_n$ are the time-invariant matrices.

The problem solving is time consuming if the dynamic constraints ϕ are concave. With the convex feasible set construction method introduced previously, the problem (11) is reformulated as:

$$\begin{aligned} \min_{\mathbf{U}_n} \quad & \sum_{n=1}^T \omega_1 \times \|\mathbf{X}_n - \mathbf{X}_n^r\|_Q^2 + \omega_2 \times \|\mathbf{U}_n\|_R^2 \\ \text{s.t.} \quad & \mathbf{X}_0 = \mathbf{X}^{\text{start}}, \mathbf{X}_N = \mathbf{X}^{\text{target}} \\ & \mathbf{X}_n = \mathbf{X}_{n-1} + \mathbf{A}_n \mathbf{X}_{n-1} + \mathbf{B}_n \mathbf{U}_{n-1} + \mathbf{C}_n \\ & \mathbf{U}_{\min} \leq \mathbf{U}_n \leq \mathbf{U}_{\max} \\ & \mathbf{X}_n \in \mathcal{F}(\mathbf{X}_n^r), \forall n = 1, \dots, N-1 \end{aligned} \quad (14)$$

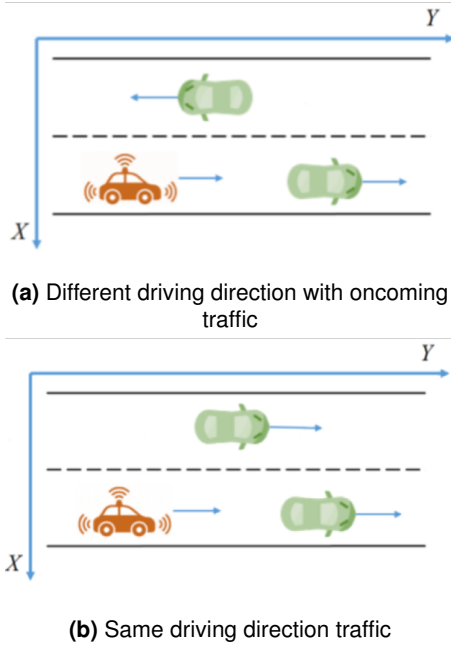


Figure 5. Illustration of simulation scenarios

V. Simulation Setup

Scenario

The proposed trajectory planning method is demonstrated in different scenarios based on two-lane roads with a lane width of 4m and a length of 100m. A dotted line separates the two lanes, which allows vehicles to cross. The ego vehicle is driving on the downside lane while the other two vehicles are in different lanes, respectively. The surrounding vehicle on the upside land drives in the same or opposite direction depending on specific situations. The map is homogeneously discretized by regular squares with a size of $0.2\text{m} \times 0.2\text{m}$, leading to the facility of meshing and the efficiency of computation.

Vehicle Model

As for the implementation of MPC, the time-invariant matrices A_n, B_n, C_n are determined by the vehicle model. A kinematic bicycle model is adopted in this paper, which is formulated as:

$$\begin{aligned}\dot{x} &= v \times \cos \beta + \dot{\psi} \\ \dot{y} &= v \times \sin \beta + \dot{\psi} \\ \dot{\psi} &= v \times \frac{\sin \beta}{l_f} \\ v \cos \beta &= v_f \times \cos \delta_f\end{aligned}\quad (15)$$

where, \dot{x} denotes the longitudinal velocity of the car in the geometric center, \dot{y} denotes its lateral velocity, v denotes its linear velocity, v_f denotes the linear velocity of the front wheel, β denotes the slip angle, ψ denotes the yaw angle of the vehicle, δ_f denotes the steering angle of the front wheel, and l denotes the wheelbase. The vehicle states \mathbf{X} is defined as a collection of $(x, y, \beta, \psi, \delta_f, v)^T$, and the control input \mathbf{U} is defined as the steering angle rate and the acceleration of the front wheel $(\dot{\delta}_f, \dot{v}_f)$.

Cost Function

The cost function $Cost(s_t)$ of the improved A* algorithm includes mainly two parts, the accumulated cost $J(s_t)$ and the heuristic function $H(s_t)$. In this paper, the accumulated cost $J(s_t, a_t)$ to move from the initial state to current state s_t consists of four indexes, namely the efficiency E , the safety S , the comfort C and the reasonableness D :

$$J(s_t) = w_E E(s_t) + w_S S(s_t) + w_C C(s_t) + w_D D(s_t) \quad (16)$$

where w_E, w_S, w_C, w_D are the corresponding weights on the four indexes.

The efficiency E evaluates the search speed of the 3D spatio-temporal map, which is defined as the displacement in a step:

$$E(s_t) = \{(x_t - x_{t+1})^2 + (y_t - y_{t+1})^2\}^{\frac{1}{2}} \quad (17)$$

The safety index S is designed as the change of the APF value:

$$S(s_t) = APF_{Node_t} - APF_{Node_{t-1}} \quad (18)$$

where the higher the APF value is, the safer the ego vehicle will be [30]. It is calculated depending on the distance that the ego vehicle keeps from the obstacles and their speed difference, as defined in:

$$\begin{aligned}AFP &= \frac{c_1}{(\frac{\Delta d}{\Delta v + c_2} - c_3)^{c_4}} \\ \Delta d &= y_{\text{front}} - y_{\text{ego}} \\ \Delta v &= v_y - v_y^{\text{front}}\end{aligned}\quad (19)$$

where, c_1, c_2, c_3, c_4 are parameters, Δd denotes the difference in longitudinal distance between ego car and the front vehicle, and Δv denotes their difference in longitudinal velocity.

The comfort C is defined as the deviation of longitudinal velocity from the desired value:

$$C(s_t) = |v_t - v_{\text{desired}}| \quad (20)$$

The reasonableness D is defined as the deviation from the center-line of the road:

$$D(s_t) = \{(x_t - x_{\text{CenterLine}})^2 + (y_t - y_{\text{CenterLine}})^2\}^{\frac{1}{2}} \quad (21)$$

The heuristic function $H(s_t)$ estimates the distance from the current point to destination, such as the Manhattan distance, the diagonal distance, and the Euclidean distance. The search criterion focuses on the cumulative direction to the target rather than the motion direction at each step. In other words, the vehicle is allowed to move in any direction. Therefore, the heuristic function is defined as the Euclidean distance:

$$H(s_t) = \{(x_t - x_{\text{target}})^2 + (y_t - y_{\text{target}})^2\}^{\frac{1}{2}} \quad (22)$$

VI. Results

The proposed method is demonstrated in four scenarios where the ego vehicle overtakes with a slow/fast oncoming vehicle, and yields to a cut-in vehicle/a walking pedestrian, respectively. Those simulations are executed in Matlab

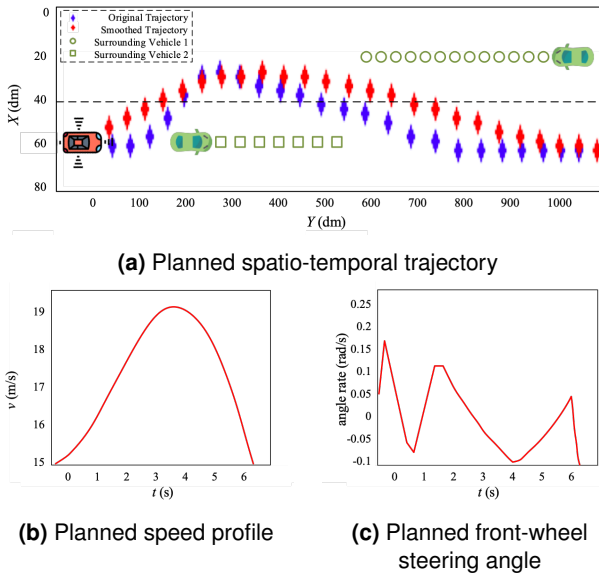


Figure 6. Simulation results for scenario I where the ego vehicle is overtaking with a slow oncoming vehicle

2016a, on a laptop computer equipped with an i7 2.2GHz core, an 8GB DDR3 memory, and a Win10 operating system.

The first scenario is overtaking on slow oncoming traffic as shown in Fig.6. The ego vehicle starts from a fixed position (0m, 6m) at an initial speed of 15m/s, and ends at the destination (100m, 6m). The front vehicle starts from (20m, 6m) at a constant speed of 5 m/s. The oncoming vehicle driving towards the opposite direction is in the adjacent lane at the constant speed of 10m/s from the initial position (100m, 2m). The ego vehicle considers not only the presence of the front vehicle but also the prediction on the oncoming vehicle. Taking the safety index as a priority, the ego vehicle speeds up to the speed limit 20m/s to overtake the front vehicle. Then the ego vehicle returns to the downside lane as soon as possible in avoidance of collision with the oncoming vehicle. Other indexes become prior to the safety

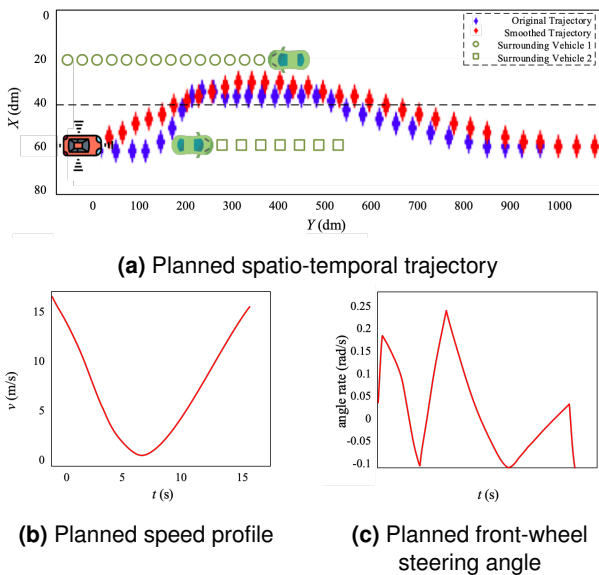


Figure 7. Simulation results for scenario II where the ego vehicle is overtaking with a fast oncoming vehicle

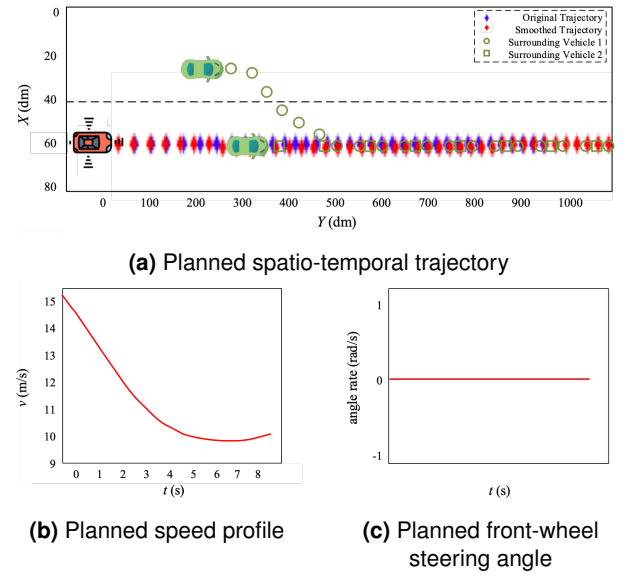


Figure 8. Simulation results for scenario III where the ego vehicle yields to a sudden cut-in vehicle

one in the cost function when the self vehicle returns to the downside lane, resulting in the speed reduction to 15m/s. The optimized trajectory turns to be less aggressive. The front-wheel steering angle and the velocity are more feasible for the controller.

The second scenario is overtaking on fast oncoming traffic as shown in Fig.7. The ego vehicle starts from a fixed position (0m, 6m), and ends at the destination (100m, 6m). It derives at an initial speed of 15m/s with a speed limit of 15 m/s. The front vehicle starts from (20m, 6m) at a constant speed of 5 m/s. The oncoming vehicle driving towards the opposite direction is in the adjacent lane at the constant speed of 15m/s from the initial position (45m, 2m). The search result shows that the ego vehicle yields to the oncoming vehicle and slows down to wait for it to pass by, and then speeds up to finish the overtaking maneuver. The trajectory after optimization becomes smooth to follow. The front-wheel steering angle and the velocity are more feasible for the controller.

The third scenario is yielding to a cut-in vehicle as shown in Fig.8. The ego vehicle starts from a fixed position (0m, 6m), and ends at the destination (100m, 6m). It derives at an initial speed of 15m/s with a speed limit of 20 m/s. The front vehicle starts from (30m, 6m) at the constant speed of 15 m/s, keeping a large initial distance from the ego vehicle. The surrounding vehicle in the neighboring lane takes a cut-in maneuver in front of the ego vehicle. It starts from the initial position (20m,2m) at the constant speed of 10m/s. When facing such a sudden cut-in, there will be considerable uncertainty and danger if the ego vehicle speeds up to follow the front vehicle or execute a lane change. Taking it into account, the ego vehicle yields to the cut-in vehicle and slows down to follow it until the end. To avoid an emergency stop, the optimization method is applied to smooth the reference trajectory. Notice that the steering angle for the front wheel keeps at 0 rad because there is no lateral displacement.

The last scenario is yielding courteously to a pedestrian as shown in Fig.9. The ego vehicle starts from a fixed position (0m, 6m), and ends at the destination (100m, 6m). It drives

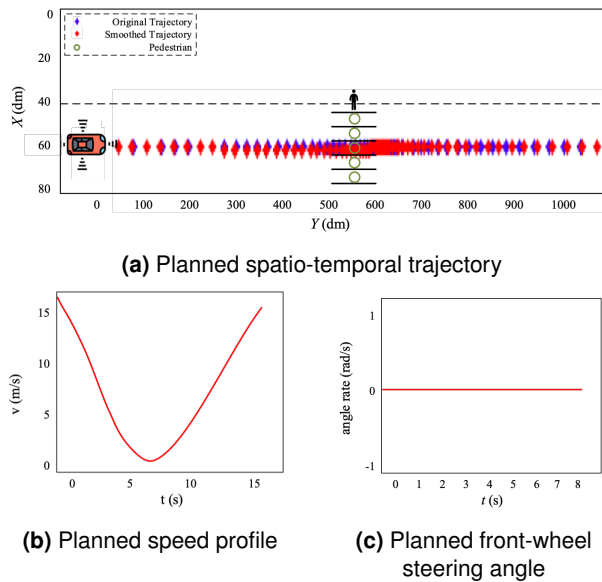


Figure 9. Simulation results for scenario IV where the ego vehicle yields to a pedestrian

along the road at an initial speed of 15m/s with the limit of 20m/s. A pedestrian is crossing the road from an initial position of (55m,6m) at a speed of 1m/s, which lasts from 1s to 7s. In theory, the ego vehicle could have taken a lane change to overtake the pedestrian. But it stops to wait for the pedestrian to pass by and then continues driving towards the end. It is because the safety index is assigned a huge weight in the cost function. The optimization method is applied to smooth the reference trajectory, leading to a smooth speed reduction rather than an emergency stop.

Furthermore, the proposed planning method is also evaluated in terms of computational cost. Three other methods are introduced for comparison. The first method, maximum interaction defensive policy (MIDP), combines the sampling-based and optimization-based methods. It generates the spatio-temporal trajectory by connecting the current vehicle states and the next states via the optimal speed profile. The optima speed profile is selected according to an optimization problem among a set of speed profile sampled by cubic polynomials [31]. Another method named as SAIO is based on sampling-based methods with post-optimization, where the trajectory is connected by speed profile sampled by quartic polynomials and the optimization is iterative [32]. The last method named as GADP is a combination of graph-search-based and optimization-based methods. It selects spatio-temporal trajectory on the state lattices by using the dynamic programming (DP) algorithms [19].

Time (ms) \ Method				
Scenario	Our Method	MIDP	SAIO	GADP
aggressive overtaking	28	51	160	4519
courtesy yielding	31	52	165	4524
emergency cut-in	30	52	162	4520
courtesy stop	35	55	170	4527

Table 1. Comparison of planning time

Table.1 shows the total computation time consumed by different planning methods. It can be seen from the table that the proposed method only took 31ms on average, which is only 59.05%, 18.87%, 0.69% of that consumed by MIDP, SAIO, and GADP, respectively.

VII. Conclusion

This is paper proposed a hybrid framework for fast and smooth spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment. First, a spatio-temporal driving map is constructed using DAG, which can illustrate both search space and future positions of dynamic obstacles. Based on the driving map, a spatio-temporal trajectory is efficiently searched via action space with the help of the improved A* algorithm. Then, taking the search result as reference trajectory, convex feasible driving area for autonomous vehicles considering surrounding vehicles is designed and MPC is applied for model feasibility. The proposed framework takes a comprehensive consideration of key advantages of both graph-search-based and optimization-based trajectory planning methods while reducing their limitations. Simulation results show that it only takes about 31ms on average for planning, while the planned trajectory, velocity and steering angle curves are reasonable and continuous. Specifically, there is neither an emergency stop nor an abrupt steering change in all simulated scenarios. Compared to the state-of-the-art methods, the proposed method has the following advantages: (1) be able to generate spatio-temporal trajectory rather than just geometrical path; (2) be able to generate trajectory which is kinematically smooth to follow; (3) enjoy a much higher efficiency while guaranteeing optimum.

This study provides a promising approach for fast and smooth trajectory planning for autonomous vehicles in constrained dynamic environment. The preliminary research opens various perspectives for future research: (1) to enhance the proposed method in order to deal with multiple stochastic moving obstacles; (2) to consider the interaction between the ego vehicle and surrounding vehicles; (3) to generalize the proposed method in different driving scenarios, e.g. intersections and unstructured roads.

Acknowledgements

This study was supported by National Key R&D Program of China with 2016YFB0100906, NSF China with 51575293, U1664263 and 51622504.

References

- [1] Liu C, Li SE and Hedrick JK. Measurement dissemination-based distributed bayesian filter using the latest-in-and-full-out exchange protocol for networked unmanned vehicles. *IEEE Transactions on Industrial Electronics* 2017; 64(11): 8756–8766.
- [2] Ni J, Hu J and Xiang C. Robust path following control at driving/handling limits of an autonomous electric racecar. *IEEE Transactions on Vehicular Technology* 2019; 68(6): 5518–5526.

- [3] Li SE, Guo Q, Xu S et al. Performance enhanced predictive control for adaptive cruise control system considering road elevation information. *IEEE Transactions on Intelligent Vehicles* 2017; 2(3): 150–160.
- [4] Li SE, Gao F, Li K et al. Robust longitudinal control of multi-vehicle systems a distributed h-infinity method. *IEEE Transactions on Intelligent Transportation Systems* 2017; 19(9): 2779–2788.
- [5] Li SE, Li G, Yu J et al. Kalman filter-based tracking of moving objects using linear ultrasonic sensor array for road vehicles. *Mechanical Systems and Signal Processing* 2018; 98: 173–189.
- [6] Xu B, Li SE, Bian Y et al. Distributed conflict-free cooperation for multiple connected vehicles at unsignalized intersections. *Transportation Research Part C: Emerging Technologies* 2018; 93: 322–334.
- [7] Frazzoli E, Dahleh MA and Feron E. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics* 2002; 25(1): 116–129.
- [8] Lavalley SM. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.
- [9] Kuwata Y, Teo J, Fiore G et al. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology* 2009; 17(5): 1105–1118.
- [10] Macfarlane S and Croft EA. Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation* 2003; 19(1): 42–52.
- [11] Gu T, Dolan JM and Lee JW. Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Daejeon, South Korea: IEEE, pp. 5474–5480.
- [12] Pivtoraiko M, Knepper RA and Kelly A. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics* 2009; 26(3): 308–333.
- [13] Kushleyev A and Likhachev M. Time-bounded lattice for efficient planning in dynamic environments. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan: IEEE, pp. 1662–1668.
- [14] Bohren J, Foote T, Keller J et al. Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics* 2008; 25(9): 598–614.
- [15] Montemerlo M, Becker J, Bhat S et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics* 2008; 25(9): 569–597.
- [16] Ferguson D, Howard TM and Likhachev M. Motion planning in urban environments. *Journal of Field Robotics* 2008; 25(11-12): 939–960.
- [17] Ruffli M and Siegwart R. On the application of the d* search algorithm to time-based planning on lattice graphs. In *Proceedings of The 4th European Conference on Mobile Robotics*. Eidgenössische Technische Hochschule Zürich, pp. 105–110.
- [18] Koenig S and Likhachev M. Fast replanning for navigation in unknown terrain. *IEEE Transactions on Robotics* 2005; 21(3): 354–363.
- [19] Ziegler J and Stiller C. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. St. Louis, MO, USA: IEEE, pp. 1879–1884.
- [20] Gu T. *Improved trajectory planning for on-road self-driving vehicles via combined graph search, optimization & topology analysis*. PhD Thesis, Carnegie Mellon University, 2017.
- [21] Shibata N, Sugiyama S and Wada T. Collision avoidance control with steering using velocity potential field. In *Proceedings of the 2014 IEEE Intelligent Vehicles Symposium*. Dearborn, MI, USA: IEEE, pp. 438–443.
- [22] Keller M, Hoffmann F, Hass C et al. Planning of optimal collision avoidance trajectories with timed elastic bands. *IFAC Proceedings Volumes* 2014; 47(3): 9822–9827.
- [23] Gao Y, Lin T, Borrelli F et al. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *Proceedings of the ASME 2010 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers, pp. 265–272.
- [24] Karampiperis P and Sampson D. Adaptive instructional planning using ontologies. In *Proceedings of the 2004 IEEE International Conference on Advanced Learning Technologies*. Joensuu, Finland: IEEE, pp. 126–130.
- [25] Xin L, Wang P, Chan CY et al. Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks. In *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems*. Maui, HI, USA: IEEE, pp. 1441–1446.
- [26] Duchoň F, Babinec A, Kajan M et al. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering* 2014; 96: 59–69.
- [27] Ni J, Hu J and Xiang C. Envelope control for four-wheel independently actuated autonomous ground vehicle through afs/dyc integrated control. *IEEE Transactions on Vehicular Technology* 2017; 66(11): 9712–9726.

-
- [28] Liu C, Lin CY and Tomizuka M. The convex feasible set algorithm for real time optimization in motion planning. *SIAM Journal on Control and Optimization* 2018; 56(4): 2712–2733.
- [29] Liu C, Lin CY, Wang Y et al. Convex feasible set algorithm for constrained trajectory smoothing. In *Proceedings of the 2017 American Control Conference*. Seattle, WA, USA: IEEE, pp. 4177–4182.
- [30] Bing H, Gang L, Jiang G et al. A route planning method based on improved artificial potential field algorithm. In *Proceedings of the 2011 IEEE 3rd International Conference on Communication Software and Networks*. Xi'an, China: IEEE, pp. 550–554.
- [31] Chen J, Tang C, Xin L et al. Continuous decision making for on-road autonomous driving under uncertain and interactive environments. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, volume 68. Changshu, China: IEEE, pp. 1651–1658.
- [32] Xu W, Wei J, Dolan JM et al. A real-time motion planner with trajectory optimization for autonomous vehicles. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*. Saint Paul, MN, USA: IEEE, pp. 2061–2067.