Hierarchical Motion Planning for Autonomous Vehicles in Unstructured Dynamic Environments

Yao Qi[1], Binbing He[1*], Yang Tai[2], Rendong Wang[1], Le Wang[1] and Youchun Xu[1]

***Abstract*— This paper presents a hierarchical motion planner for generating smooth and feasible trajectories for autonomous vehicles in unstructured environments with static and moving obstacles. The framework enables real-time computation by progressively shrinking the solution space. First, a graph searcher based on combined heuristic and partial motion planning is proposed for finding coarse trajectories in spatiotemporal space. To enable fast online planning, a time interval-based algorithm that considers obstacle prediction trajectories is proposed, which uses line segment intersection detection to check for collisions. Second, to practically smooth the coarse trajectory, a continuous optimizer is implemented in three layers, corresponding to the whole path, the near-future path and the speed profile. We use discrete points to represent the far-future path and parametric curves to represent the near-future path and the whole speed profile. The approach is validated in both simulations and real-world off-road environments based on representative scenarios, including the "wait and go" scenario. The experimental results show that the proposed method improves the success rate and travel efficiency while actively avoiding static and moving obstacles.**

I. INTRODUCTION

The problem of motion planning has attracted much interest regarding various robotic systems in recent decades. In constrained dynamic environments, motion planning methods for autonomous vehicles must consider both nonholonomic and spatiotemporal constraints and thus present more challenges than methods for omnidirectional platforms.

The core objective of motion planning is to maximize the safety, efficiency and comfort of robot navigation. Several works have provided comprehensive overviews of current motion planning approaches in the vehicle automation domain [1]–[3]. However, most of these approaches have been demonstrated in on-road environments or for microrobots. Motion planning in off-road dynamic environments faces additional challenges. First, methods without consideration of the time dimension incur great losses in the optimality of their results, but the solution time cost sharply increases with the addition of the time dimension. Moreover, a behavioral layer using finite state machines offers poor performance for moving obstacles in off-road environments, which have few, if any, clear road boundaries. In addition, the uncertainty in the perception, prediction and control results necessitates a high frequency of long-horizon replanning to ensure safety. Finally, the presence of irregular static obstacles gives rise to a nonconvex environment, and the nonholonomic constraints reduce the feasible state space. The above challenges make the problem even more intractable, and thus, optimality and completeness cannot be ensured. Therefore, focusing on finding a near-optimal solution and executing fast replanning to approximate the global solution is a practical approach.

We focus on addressing these problems for autonomous driving in unstructured dynamic environments in a more practical way. We first detail the mechanism of a spatiotemporal searcher, which uses partial motion planning (PMP) to obtain a time-bounded solution with a final state that is free of inevitable collision states (ICS) [4]. Spatiotemporal motion primitives are generated by discretizing the control space to ensure the feasibility of the search results. To efficiently guide the search tree to the goal state with the correct position, orientation and speed, a heuristic function combining the time cost and the length of the Dubins curve is proposed. Collision checking is a time-consuming process; thus, we use line segment intersection detection to fastly check for collisions with obstacle prediction trajectories. We also propose a three-layer optimizer to smooth the path and speed profile once feasible results have been searched. Considering that motion planning is a cyclical process, we constrain the path curvature for only the near-future path. All discrete path points are smoothed subject to the constraints of the dynamic environment in the first layer. The second layer transforms the near-future path to a cubic spiral curve via uniform sampling and candidate path valuation. The third layer optimizes the speed profile with a quintic spline curve. We compare our method with representative methods in simulations. Results show that our method improves the success rate and travel efficiency. What's more, we conduct experiments in off-road environments with moving cars. Results demonstrate the capabilities of our method in real-time planning and overcoming perception noises.

The contributions of this paper are as follows:

1) An online searcher is proposed to obtain the coarse trajectory in constrained dynamic environments based on PMP. The searcher can efficiently explore spatiotemporal space based on fast collision checking and combined heuristic.

2) A multilayer optimizer is designed to smooth the coarse trajectory in real time. By optimizing the near-future path and the far-future path differently, the optimizer reduces the time complexity of satisfying the path curvature constraints.

3) Based on the above two algorithms, we implement a

*The corresponding author of this paper.

This work was supported by National Key Research and Development Program of China 2016YFB0100903.

[1]The authors are with the Institute of Military Transportation, Army Military Transportation University, Tianjin, China. yaoqi_cn@outlook.com, frozen_ice@126.com, wrd1992@163.com, m15155226113@163.com, xu56419@126.com.

[2]Yang Tai is with the Tianjin Navigation Instruments Research Institute, Tianjin, China. 13821100288@163.com.

motion planner with high success rates and travel efficiency in simulations and real-world experiments.

II. Related Work

Motion planning has been divided into path planning and speed planning in [11], [12]. A path is a sequence of configuration vectors that consist of position and orientation, and a speed profile consists of velocity and time. Path planning techniques can be classified into four groups according to their implementation [5]: graph search-based planners, random sampling-based planners, curves-interpolating planners and numerical optimization approaches. These include the hybrid-state A* family [6], the Rapidly Exploring Random Tree (RRT) family [7], parametric curves [8] and multiobjective optimization [9]. Speed planning is a similar problem to path planning [10]–[12].

We classify motion planning techniques for autonomous vehicles in dynamic environments into four categories in accordance with their strategies for dealing with moving obstacles. The first strategy is to treat a moving object or its short-term prediction trajectory into a static obstacle to avoid collisions in the near future [13]. However, this strategy performs poorly in most "wait and go" cases [14]. The second strategy is to perform decoupled path-speed planning, which entails ignoring moving obstacles during path planning but considering them during speed planning [15], [28]. This strategy is always combined with behavioral decisions, such as vehicle overtaking or vehicle following. These methods are popular for on-road autonomous driving but are poor in roadless dynamic environments [14], [15]. In unstructured environments, the above two strategies suffer from suboptimality and incompleteness.

The third strategy, which underlies the class of methods known as velocity obstacle methods, entails selecting avoidance maneuvers to avoid static and moving obstacles in velocity space. This method was first introduced in [16]. [17] combined Voronoi diagrams and reciprocal velocity obstacles for decentralized multiagents. The main challenges of these methods are avoiding local minima and satisfying the nonholonomic constraints when applied to autonomous vehicles [18]. The fourth strategy is planning in spatiotemporal space, which mainly consists of graph search-based and random sampling-based methods. [19] used Hierarchical Cooperative A* to search for paths in XYT space, and [14] defined safe intervals in order to simplify the time dimension into discrete states to reduce the time cost. [20] solved multiobjective path planning problems with dynamic obstacles based on [14]. [21] explored a state-time graph built on motion primitives and proposed a fast collision checking algorithm to ensure real-time performance, but only the circular robots with low-speed ($\leq$ 1.5 m/s) were considered. [22] formulated the planning problem in the triangulation space for fast searching. [21], [22] ignored irregular static obstacles and the smoothness of the results. [23], [24] generated trajectories in the sparse search space, which resulted in poor adaptability in narrow scenarios. [4] combined PMP and RRT to find time-bounded planning solutions in simulations. In addition, [25] proposed the model predictive contouring control (MPCC) for a mobile robot in dynamic indoor environments.

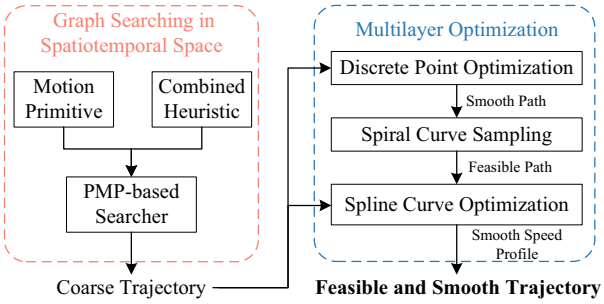

Fig. 1. An overview of the proposed hierarchical framework.

Our algorithm builds on the existing work discussed above and consists of two main phases, as illustrated in Fig. 1. In the first phase, a heuristic searcher is applied in spatiotemporal state space to obtain a feasible solution in dynamic environments using motion primitives (Sect. III). We propose several techniques to ensure the real-time performance of the searcher. In the second phase, we propose a multilayer optimizer to progressively improve the smoothness of the solution and maintain the solution's stability between successive frames (Sect. IV).

III. Graph Searching in Spatiotemporal Space

A. Autonomous Vehicle Motion Primitives

We use the Cartesian coordinate system rather than the Frenet coordinate system to adapt to complex environments [26]. Spatiotemporal space Ω is used as the search space:

$$\Omega = \left\{ \mathbf{q} = \begin{bmatrix} t & x & y \end{bmatrix}^T \mid t \in [0, N_t], x \in \mathbb{R}^+, y \in \mathbb{R}^+ \right\} \quad (1)$$

where t is time, the t of the initial state is taken as the origin in the time dimension, the maximum time is N_t, and (x, y) are Cartesian coordinates.

Motion primitives for an autonomous vehicle are constructed in Ω using the bicycle kinematic model. $\mathbf{u} = (a, \phi)\,(a \in [a_{\min}, a_{\max}], \phi \in [-\phi_{\max}, \phi_{\max}])$ is defined as the longitudinal and lateral control input of the vehicle. The feasible a is limited by the maximum acceleration $a_{\max}$ and the minimum acceleration $a_{\min}$, while the lateral control input $|\phi|$ should be smaller than $\phi_{\max}$. Each ϕ corresponds to a radius r defined by $\tan(\phi)/l$, where l is the wheel base of the car. $\mathbf{x} = (x, y, \theta, v) \in \mathcal{X}$ defines the vehicle state space, which includes position (x, y), heading θ and speed v. We obtain the vector difference between consecutive states when $\phi \neq 0$ as follows:

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \\ \Delta v \end{bmatrix} = \begin{bmatrix} r\cos\theta\sin\Delta\theta + r\sin\theta(1-\cos\Delta\theta) \\ r\sin\theta\sin\Delta\theta - r\cos\theta(1-\cos\Delta\theta) \\ d/r \\ a\Delta t \end{bmatrix} \quad (2)$$

where d is the travel distance between consecutive nodes and Δt is their time interval. The turn radius is infinite when $\phi = 0$, meaning that the vehicle travels in a straight

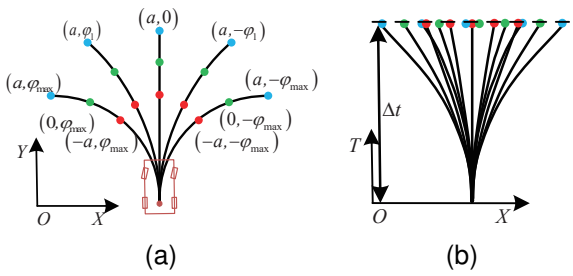

Fig. 2. Motion primitives based on the car model. (a) is plotted in XY space, while (b) is plotted in XT space.

line; therefore, we instead obtain the position difference as follows:

$$\Delta x = d\cos\theta, \Delta y = d\sin\theta \tag{3}$$

Spatiotemporal motion primitives are generated as shown in Fig. 2 by discretizing the control input $\mathbf{u}$. Single-step primitive generation for node ε_i is performed at a fixed time interval Δt_0, where Δt_0 is limited by an upper boundary $d_{\max}$ and a lower boundary $d_{\min}$. For example, we compute d as $d = v_i\Delta t_0 + a{\Delta t_0}^2/2$; if $d > d_{\max}$, then Δt is recalculated as $\left(\sqrt{2ad_{\max}+v_i^2}-v_i\right)/a$, where v_i denotes the speed at ε_i.

B. PMP-based Online Searcher

Considering the complexity of motion planning with nonholonomic constraints in an unstructured dynamic environment [27], we use a PMP-based searcher inspired by [4] to obtain a partial solution online.

In Alg. 1, we use a hash table, $\mathcal{M}$, and a priority list, $\mathcal{Q}$, to manage the trajectory search process. Each node ε_i has a key defined as (x, y, θ, t), and each node in the same spatiotemporal grid cell has the same key. $\mathcal{M}$ stores the node connection graph, and $\mathcal{Q}$ sorts the key by $f(f = g + h)$ in ascending order. In Lines 5-6, we select the top key k_{top} from $\mathcal{Q}$, which has the minimum f, and find the corresponding node in $\mathcal{M}$ using k_{top}. Then, we check whether the goal state $\mathbf{x}_{\text{g}}$ is achieved or analytic expansion is valid, which is defined by the safety of the analytic trajectory (Lines 7-8). The analytic trajectory consists of a Dubins curve to the goal position and a trapezoidal speed profile to the goal speed. To ensure that we can solve for a feasible trajectory within the planning horizon, an ICS-free node $\mathcal{X}_{\text{ICS-free}}$ and its parents will be returned if the timeout occurs (Lines 9-10). In Line 12, we generate the child nodes and update their cost and heuristic values. $\varepsilon_{\text{valids}}$ stores the valid nodes that are found to be safe in collision checking for static and moving obstacles. We add new nodes and replace old nodes in $\mathcal{Q}$ and $\mathcal{M}$ in accordance with A*.

C. Heuristics

To reduce the time cost of searching after the addition of the time dimension, we combine two heuristics as described below to guide the search tree. These heuristics are also applicable to other search methods.

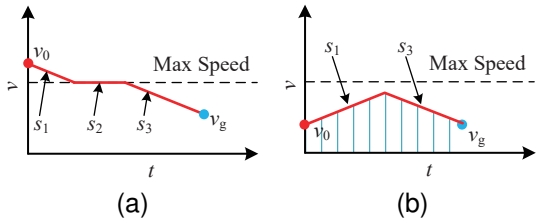

Fig. 3. Examples of trapezoidal speed planning. (a) shows a case of slowing down, while (a) shows a case of speeding up.

Algorithm 1 OnlineSearching($O, \mathbf{x}_0, \mathbf{x}_{\text{g}}$)

1: $\varepsilon_0 \leftarrow$ GENERATENODE($\mathbf{x}_0$)
2: $\mathcal{Q} \leftarrow \emptyset$, $\mathcal{Q}$.INSERT($\varepsilon_0.key, \varepsilon_0.f$)
3: $\mathcal{M} \leftarrow \emptyset$, $\mathcal{M}$.INSERT($\varepsilon_0.key, \varepsilon_0$)
4: **while** $\mathcal{Q} \neq \emptyset$ **do**
5: $k_{\text{top}} \leftarrow \mathcal{Q}$.POP().FIRST()
6: $\varepsilon_i \leftarrow \mathcal{M}$.QUERY($k_{\text{top}}$), $\varepsilon_i \leftarrow$ CLOSE()
7: **if** $\varepsilon_i = \mathbf{x}_{\text{g}} \vee$ ANALYTICEXPANSION($\varepsilon_i, \mathbf{x}_{\text{g}}$) **then**
8: **return** TRACERESULTS(ε_i)
9: **else if** TIMEHORIZONREACHED() $\wedge\, \varepsilon_i \notin \mathcal{X}_{\text{ICS}}$ **then**
10: **return** TRACERESULTS(ε_i)
11: **end if**
12: $\varepsilon_{\text{valids}} \leftarrow$ GENERATEVALIDPRIMITIVES(ε_i,O)
13: UPDATELIST($\varepsilon_{\text{valids}}, \mathcal{Q}, \mathcal{M}$)
14: **end while**
15: **return** FAILURE()

The first heuristic, which is inspired by [6], ignores obstacles and considers the nonholonomic constraint for cars. To reduce the calculation cost in the online search process, we compute the path length of the Dubins curve from $(0, 0, 0)$ to the discretized state (x, y, θ) offline. This heuristic guides the search tree to the goal state with the correct heading.

The second heuristic is applied in the time dimension and depends on the first heuristic. We compute the time cost from the current state $(v_0, 0)$ to the goal state $(v_{\text{g}}, s_{\text{g}})$ using a trapezoidal speed profile that satisfies the maximum speed constraint as shown in Fig. 3, where s_{g} is the length of the Dubins curve in the first heuristic. The second heuristic t_{A} is calculated as expressed in (4), where s_1, s_2, and s_3 are shown in Fig. 3 and a_{c} is a comfortable acceleration. The minimum time cost t_{A} is clearly an admissible heuristic. The effect of this second heuristic is that it assigns high costs to search nodes that approach the goal at an undesirable speed.

$$t_{\text{A}}=\frac{-v_0+\sqrt{{v_0}^2+2a_{\text{c}}s_1}}{a_{\text{c}}}+\frac{s_2}{v_{\max}}+\frac{v_{\max}-\sqrt{{v_{\max}}^2-2a_{\text{c}}s_3}}{a_{\text{c}}} \tag{4}$$

Given the above two heuristics, we combine them by means of normalization and weighting. The combined heuristic leads to a substantial improvement in the number of nodes.

D. Collision Checking

The obstacles, O, are divided into static obstacles, O_{S}, and moving obstacles, O_{D}, in accordance with their speed.

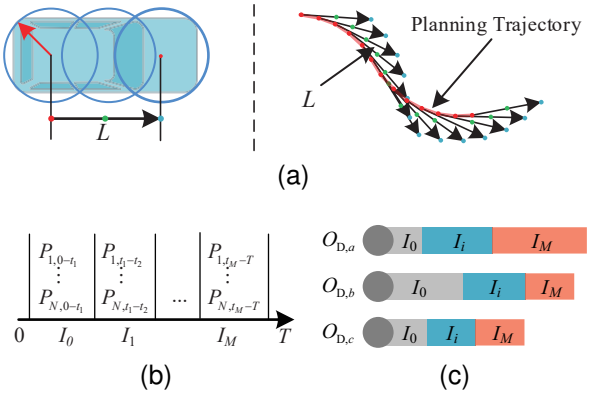

Fig. 4. Time intervals of the prediction trajectory. (a) shows the collision checking model. (b) illustrates time intervals, and (c) presents line segments corresponding to moving obstacles a, b, c divided into time intervals.

Suppose that each O_{D} includes one or more prediction trajectories P_i. Whether the irregular O_{S} such as stones, mounds and bushes, are converted into circles or polygons, there will be some transformation errors and high computational costs. To ignore the shape of the irregular O_{S}, we use a dilated configuration space to achieve efficient static obstacle collision checking. We transform the vehicle into a line segment L that connects by the centers of multiple fitted circles enclosing the vehicle, as shown in Fig. 4(a). After dilating the grid map with the radius of the fitted circles, we check the safety of the nodes based on L.

The O_{D} such as moving cars, can be represented by convex envelopes. A typical collision checking method is to discretize the prediction trajectory to several points, but this method cannot ensure safety between points. Hence, the reliability of this method depends on the discretization interval. To achieve redundant and fast collision checking, we propose considering the intersection of line segments. As shown in Fig. 4(b), we transform each prediction trajectory into several line segments rather than points based on a fixed time interval. Fig. 4(c) shows that obstacles a, b and c have different segment lengths in the same time interval due to the different speed trends of prediction trajectories. In practice, we choose the line segments of prediction trajectories based on the time of ε_k and set the state $\varepsilon_k \cap P_{i,t_k-t_{k+1}} = \varnothing (i = 1, \cdots, N)$ as safe.

IV. Multilayer Optimization

Although the heading of the coarse trajectory found by the above algorithm is continuous, the trajectory typically cannot be tracked with continuous steering and acceleration due to jagged curvature and jerk. Therefore, we postprocess the trajectory by applying the following three-layer optimization.

A. Discrete Point Optimization

Given a sequence of vertices $\mathbf{x}_i = (x_i, y_i)$, $i \in [1, N]$, which is uniformly interpolated from the search results. The objective function is defined as $\sum_{i=2}^{N-1} \mathbf{h}_i^2$, where $\Delta \mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{i-1}$, and $\mathbf{h}_i = \Delta \mathbf{x}_{i+1} - \Delta \mathbf{x}_i$. The constraint function is $\mathbf{x}_i - \mathbf{x}_{\mathrm{o}i} \in \Xi_i$, and the Ξ_i of $\mathbf{x}_{\mathrm{o}i}$ can be obtained by expanding the minimum box, as shown in Fig. 5(a). Based on the time t_i of $\mathbf{x}_{\mathrm{o}i}$, the states of moving obstacles are calculated from the prediction trajectories of O_{D}, which are merged with O_{S} to compose the obstacle set O_i at time t_i. There are no obstacles within Ξ_i when $\Xi_i \cap O_i = \varnothing$.

The objective function is quadratic and the constraint is linear. Thus, we efficiently solve the smoothed trajectory via quadratic programming as shown in Fig. 5(b). Furthermore, the above solver does not attempt to ensure that the position and orientation of the end state remain unchanged during continuous driving. And anchor points can be set with an infinitesimal Ξ_i when needed.

B. Parametric Curve-Based Optimization

Note that motion planning is a cyclical process. The long-distance trajectory is used to maintain global optimality, the robot executes only the part of the trajectory nearest itself (approximately the first 100 ms of the trajectory) before another round of motion planning is performed. The smoothed coarse path obtained as described above has continuous curvature and achieves collision avoidance for all obstacles. However, considering the efficiency of discrete point optimization, the curvature of the smoothed coarse path is not strictly constrained. Inspired by a sampling-based strategy with a reference path [28], our smoothed coarse path is treated as a reference path of high confidence, and then we generate a spatial path to replace the near-future path.

To improve the resolution of the near-future path, we use a cubic polynomial of the cumulative distance s to express the curvature ρ, which is defined as $\rho(s) = \rho_0 + k_1 s + k_2 s^2 + k_3 s^3$. The vehicle state can be obtained by (5), where s_{g} is the length of the curve and $(x_0, y_0, \theta_0, \rho_0)$ is the initial state.

$$\begin{cases} \theta(s) = \theta_0 + \int_0^{s_{\mathrm{g}}} \rho(s) ds \\ x(s) = x_0 + \int_0^{s_{\mathrm{g}}} \cos\theta(s) ds \\ y(s) = y_0 + \int_0^{s_{\mathrm{g}}} \sin\theta(s) ds \end{cases} \tag{5}$$

The unknown parameters $(k_1, k_2, k_3, s_{\mathrm{g}})$ can be solved for given an initial state $(x_0, y_0, \theta_0, \rho_0)$ and a goal state $(x_{\mathrm{g}}, y_{\mathrm{g}}, \theta_{\mathrm{g}}, \rho_{\mathrm{g}})$ [29]. However, extreme values of ρ cannot be constrained in this way. To achieve this, we transform optimization parameters into $(\rho_1, \rho_2, \rho_3, s_{\mathrm{g}})$ and recast the optimization problem as follows:

$$\begin{aligned} &\min f(\rho_1, \rho_2, \rho_3, s_{\mathrm{g}}) \\ &\text{s.t.} \begin{cases} \rho_1 \le |\rho_{\max}|, \rho_2 \le |\rho_{\max}|, \rho_3 \le |\rho_{\max}| \\ s_{\mathrm{g}} \ge s_{\mathrm{E}} \end{cases} \end{aligned} \tag{6}$$

where $s_{\mathrm{E}} = \sqrt{(x_0 - x_{\mathrm{g}})^2 + (y_0 - y_{\mathrm{g}})^2}$; ρ_1, ρ_2 and ρ_3 are uniform points on the curve $\rho(s)$; and $\rho_{\max}$ is the maximal curvature of the vehicle.

As shown in Fig. 5(b), path candidates are generated in two layers within a time horizon $\mathcal{T}$. The first sampling layer consists of n longitudinal sampling sublayers, each of which contains m lateral sampling points. The second sampling layer, which is used to connect sampling points in the first

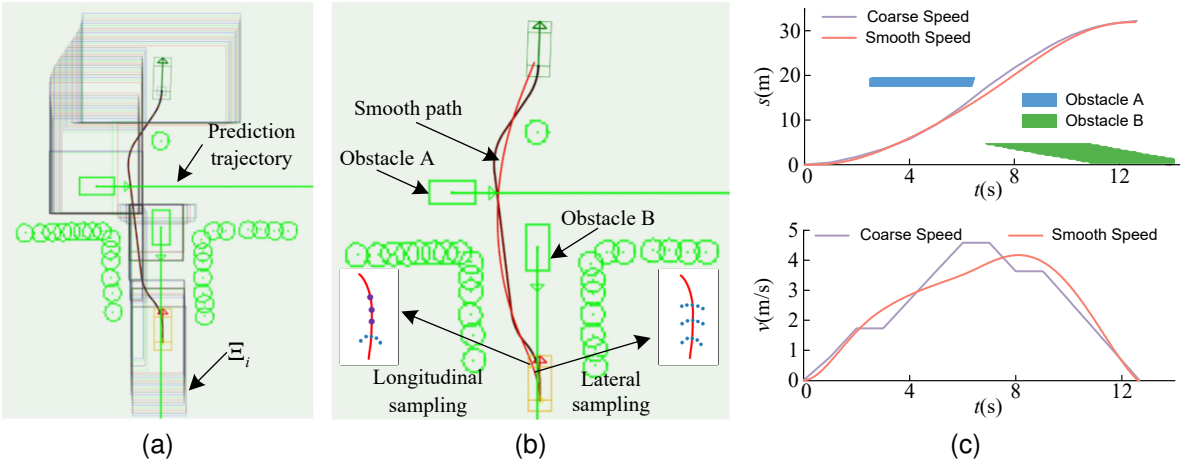

Fig. 5. An example of optimization results. The green arrows indicate the directions of moving obstacles, and static obstacles are represented by circles without arrows. (a) shows all of the Ξ values in the dynamic environment. (b) shows the solution obtained by the searcher and the corresponding smoothed result. (c) shows the coarse speed profile and the spline-based speed profile.

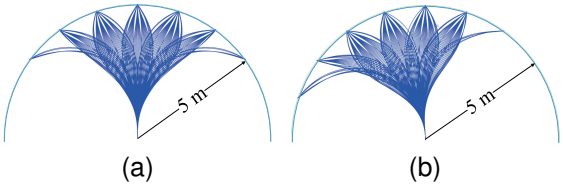

Fig. 6. Offline spiral curve groups.

layer to the smoothed coarse result, consists of $p \times 1$ sampling sublayers. Then, we check for collisions and evaluate candidates (the total number is $m \times n + (m-1) \times n \times p$), and the lowest cost path is selected and regenerated.

To ensure high efficiency of collision checking and curve generation, we perform the corresponding computations offline for several arcs of a fixed radius, which is the same as the sampling radius. We store the collision states and curves based on the initial curvature, an approach that is inspired by [30]. For example, Fig. 6 shows two groups of paths on an arc of radius 5 m; here, $\rho_{\max} = 0.2\ \mathrm{m}^{-1}$, $\rho_{\mathrm{g}} \in [-0.2, 0.2]$, and the curvature interval in ρ_{g} is $0.05\ \mathrm{m}^{-1}$.

C. Spline-Based Optimization

The coarse speed profile in the search results has a jagged jerk. Therefore, we smooth this profile using a quintic spline curve, which consists of several polynomial curves, as expressed in (7).

$$s(t) = k_0 + k_1 t + k_2 t^2 + k_3 t^3 + k_4 t^4 + k_5 t^5 \qquad (7)$$

We map the coarse speed profile to the ST coordinate system. Thus, the stretch at each discretized time point is linearly limited by the moving obstacle envelopes and the safe distance in ST coordinates. Each point's speed limit is also computed from the maximal lateral acceleration a_{MaxLat} as $v_{\mathrm{limit}}^2 |\rho| \leq a_{\mathrm{MaxLat}}$, and the acceleration is limited as $a \in [a_{\min}, a_{\max}]$. Finally, we solve the objective function in (8) to obtain a smooth speed profile. Where $\omega\{1,2,3\}$ are the weight factors of the speed, acceleration and jerk terms, respectively, and ω_4 is the weight of difference between the speed and the expected speed v_{E}. As shown in Fig. 5(c), the coarse speed profile and relevant moving obstacles are mapped into ST coordinates, and the trapezoidal speed profile is smoothed with a quintic spline curve.

$$\sum_{i=1}^{K} \int_{t_i}^{t_i} \left(\omega_1 \left(s_i'\right)^2 + \omega_2 \left(s_i''\right)^2 + \omega_3 \left(s_i''\right)^2 + \omega_4 \left(s' - v_{\mathrm{E}}\right)^2 \right) dt \qquad (8)$$

V. Experiments

A. Implementation Details

We present experiments conducted both through simulation and in the real-world. In the simulation experiments, the maximal speed is set to 6 $\mathrm{m{\cdot}s^{-1}}$, the vehicle shape is fitted with 3 circles, the grid size used in collision checking for static obstacles is 0.2 m, the search time is limited to within 90 ms, the grid size and orientation resolution in the offline computation of the Dubins curve are 1 m and 10°, and the value of a_c in the heuristics is 1.0 $\mathrm{m{\cdot}s^{-2}}$. The total number of path candidates is 51, and we can also sample more densely to improve the probability of the existence of a safe near-future spatial path. For instance, the first sampling layer on the smoothed coarse path meets the safety requirement and has the lowest cost in most cases. All methods run on a 3.2 GHz i7-8700 computer using a single CPU, and the motion planning system performs replanning at 10 Hz. The replanning initial state is a state deduced from the last trajectory based on the planning period.

B. Simulation and Analysis

We compare our first phase search-based method with two representative methods, namely, the velocity obstacles (VO) and hybrid-state A*. To enable the application of the hybrid-state A* in dynamic environments, a combined hybrid-state A* & wait and go (HAWG) method is designed by means of a decoupled path-speed planning strategy. The "wait and go" refers to the case in which the autonomous vehicle needs to

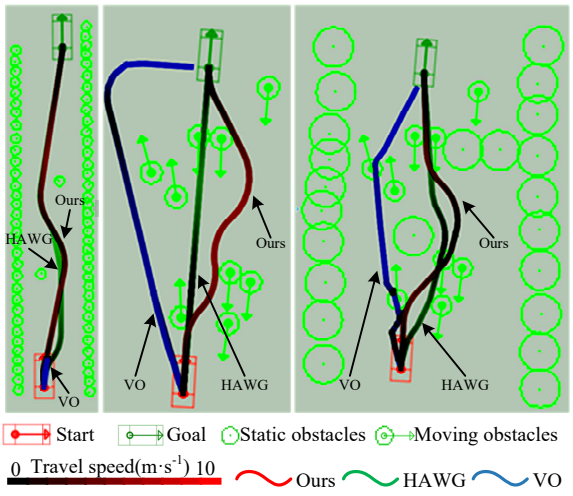

Fig. 7. Experimental scenarios and tracks traveled.

stop and wait for dynamic obstacles to pass before passing through the narrow driving space, as shown in Fig. 8(c). The HAWG uses hybrid A* to find a collision avoiding path for static obstacles and avoids moving obstacles by speed planning, which means that the robots passively slow down.

To adapt the VO method, all static and moving obstacles are set as circles in the simulations. For validation, three scenarios are created: only static obstacles (scenario 1), only moving obstacles (scenario 2) and mixed moving and static obstacles (scenario 3). Scenario 2 is from the pedestrian motion dataset ETH [31], and static obstacles are added to scenario 2 to construct scenario 3. The distance between the starting point and the goal point is approximately 35 m. The conditions for the success of an experiment are that there is no collision with any obstacles and the goal state is reached within 60 s.

The global travel tracks obtained by the three methods are shown in Fig. 7. The VO cannot guide the robot to reach the destination in scenario 1, in which there are 72 static obstacles. The proposed method and the HAWG reach the goal state in all scenarios. The trajectory of the VO does not satisfy the kinematic constraint as shown in Fig. 8(a). The HAWG does not have the ability to actively avoid moving obstacles but instead attempts to achieve passive avoidance; the resulting collisions are shown in Fig. 8(b). In contrast, Our method achieves active avoidance of moving obstacles in spatiotemporal space, as shown in Fig. 8(c). The vehicle stops at the eighth second in scenario 3, as shown in Fig. 8(d), which shows that our method can support "wait and go" decision-making by the motion planning system.

To quantify the influence of the environmental density and perception range on each method, we present three experiments conducted in a 50 m-by-50 m simulation environment with different parameters. The default parameters are as follows: 10 static obstacles and 10 moving obstacles, and a perception range of 20 m. The positions and directions of the obstacles are randomly initialized within the range of the environment, the speed is randomly generated between 1.0

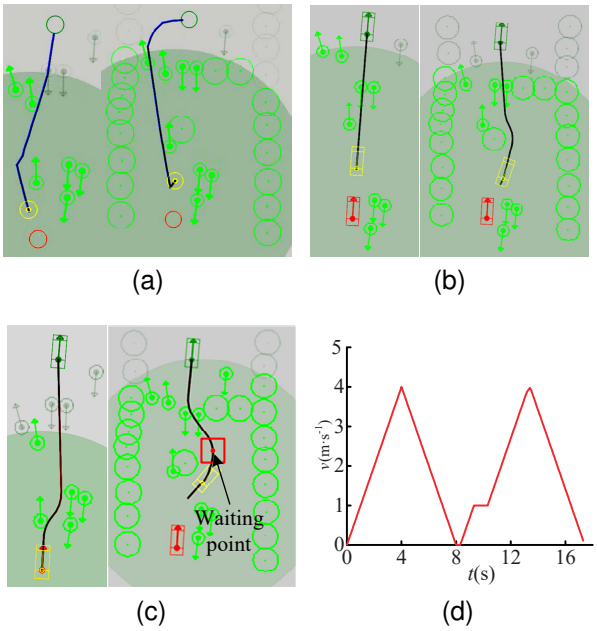

Fig. 8. Search results of all methods in scenarios 2 and 3. (a), (b) and (c) are the search results of VO, HAWG and Our method, respectively, and (d) is the travel speed result of our method, the robot stops and waits for the moving obstacles at "waiting point" indicated in (c).

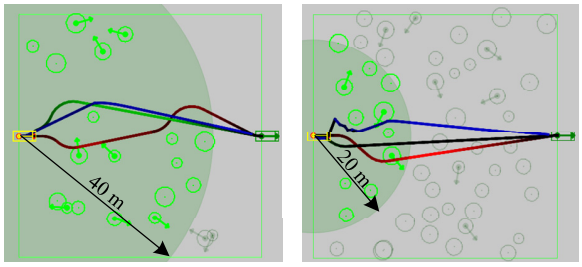

Fig. 9. Random scenarios.

and 2.0 m·s^{-1}, and the radius of each obstacle is between 1.0 and 2.0 m. Fig. 9 shows two randomly generated scenarios with the perception range set to 40 m and the number of static obstacles set to 40. In each group of experiments, only one parameter is varied and 50 experiments are conducted. The experimental results are shown in Fig. 10. The total travel time under the VO method is not considered because this method does not consider kinematic constraints.

We can see that an increase in the number of static obstacles has the greatest impact on the success rate of the VO from Fig. 10(a). When there are 40 static obstacles, the travel success rate drops to 38 %. For the HAWG, the standard deviation of the travel time is largest with 40 static obstacles because it cannot actively avoid moving obstacles. The success rate of our method is less affected by the density of static obstacles. The proposed method can be regarded as a variant of hybrid A* and thus has almost the same time cost as HAWG in sparse static environments.

All the planners exhibit worse performance as the number of moving obstacles increases, as shown in Fig. 10(b). The reason for this is that the feasible region in the planning

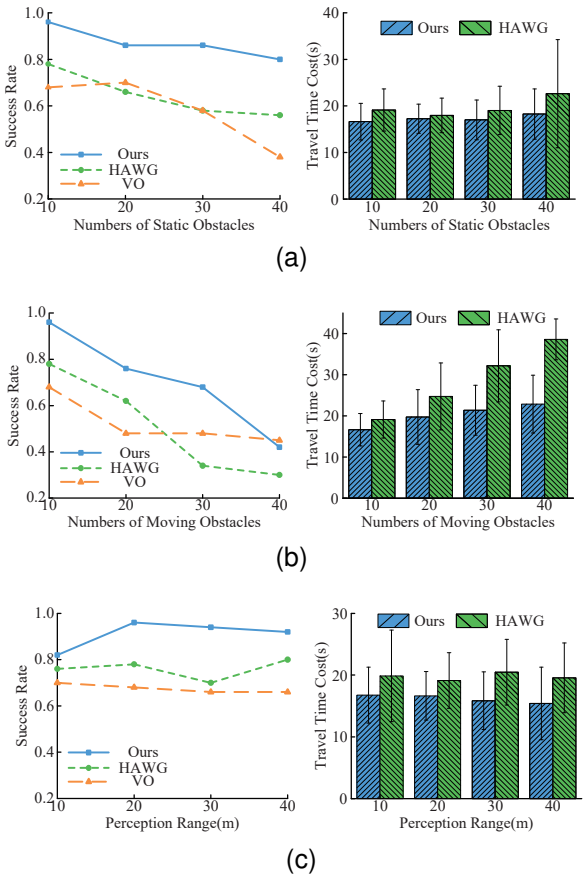

Fig. 10. Experimental results on the effect of observability. (a), (b) and (c) show the results obtained with different numbers of static obstacles, different numbers of moving obstacles and different perception ranges.

space is occupied by moving obstacles and their prediction trajectories. The classical VO has the highest success rate when there are 40 moving obstacles due to its neglect of kinematic constraints. The success rate of our method is higher than those of the other methods before the number of obstacles reaches 40, and its increase in travel time is much less than that of the HAWG.

Fig. 10(c) shows that all methods are less affected by the perception range. Because each method has high real-time performance, they do not easily fall into local optima in a sparse environment. Nevertheless, Our method has the highest success rate and the lowest travel time.

The average success rates shown in Table I indicate that the VO has poor adaptability to static scenes, with the lowest average success rate among all methods (58.5%), the HAWG always has the lowest success rate in the dynamic scenario. Meanwhile, the success rates of our method in various scenarios are higher than those of the other methods. Its average success rate in all environments is 82.8%, which is 19% and 23.4% higher than those of the HAWG and VO methods, respectively. The results for the average travel time cost show that compared with the HAWG, our method reduces the average time cost by 30% in experiments with different numbers of moving obstacles and by 21% in all scenarios.

TABLE I

AVERAGE STATISTICS IN THE EXPERIMENTS

Scenario	Success Rate (%)			Travel Time (s)	
	VO	HAWG	Ours	HAWG	Ours
Different Numbers of Static Obstacles	58.5	64.5	87.0	19.7	17.3
Different Numbers of Moving Obstacles	52.3	51.0	70.5	28.7	20.2
Different Perception Ranges	67.5	76.0	91.0	19.8	16.2
Average	59.4	63.8	82.8	22.7	17.8

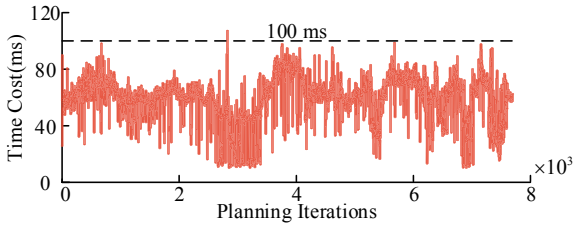

Fig. 11. Time cost in the real-world experiments.

C. Real-World Tests

To further validate the proposed method, we conduct extensive real-world tests in off-road environments. In all tests, we set the maximum speed to 8 $m{\cdot}s^{-1}$. In the real-world tests, we implement the motion planning system based on a reference path from a topological map. To play the important role of the reference path, we construct a virtual boundary along the reference path, which is expanded in the presence of obstacles on the reference path and shrunk in the absence of obstacles. We also present a new search termination condition in addition to those in Alg. 1, which stops the search process when any of the search nodes reaches the threshold distance.

The overall time cost of the graph search and multilayer optimization modules in this experiment is statistically analyzed, as shown in Fig. 11. We conduct this experiment over a duration of approximately 780 s without any human intervention, and the total travel distance is 3170 m. In the real-world off-road experiment, the average time in the environment dominated by static obstacles is 59.47 ms. In practice, this can meet real-time requirements. Note that in the dynamic scenarios as shown at the bottom of Fig. 12, the moving cars are the same model as the autonomous vehicle, but there are some size errors of moving cars in the perception data. Our method efficiently overcomes such noises in the perception and guides the autonomous vehicle to drive safely.

VI. CONCLUSION

In this work, we propose a practical hierarchical framework for autonomous vehicle motion planning in unstructured dynamic environments. The motion planner consists of two phases. In the first phase, an online graph searcher is applied to find a coarse trajectory in spatiotemporal space. We propose time heuristics and a fast collision checking algorithm to accelerate the search process. The second phase is multilayer continuous optimization, in which approaches

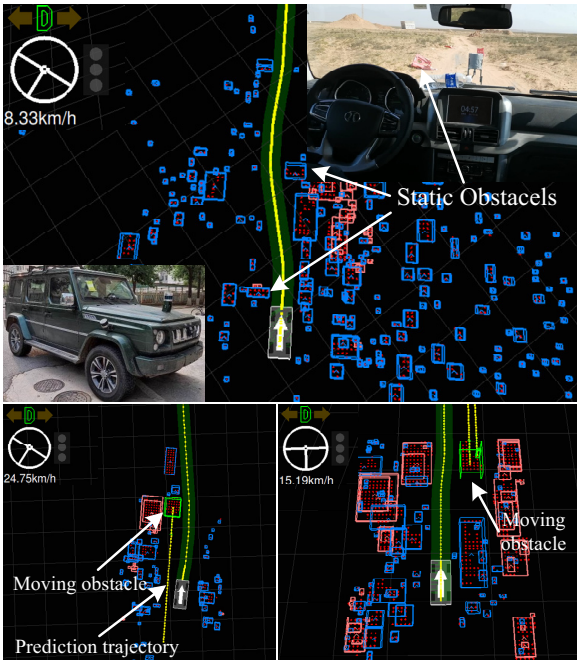

Fig. 12. Experiments in off-road environments: a scenario with irregular static obstacles (top) and two scenarios with moving cars travelling in the opposite direction and in the same direction (bottom).

are used to smooth the near-future and far-future paths based on practical considerations. Both simulations and real-world tests show the competence of our method.

One limitation of our method is the lack of uncertainty in prediction. However, uncertainty always exists and generally influences the driving trajectory, especially in unstructured and semi-structured environments. In the future, we plan to consider such issues of uncertainty arising from prediction.

REFERENCES

[1] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Eng. Appl. Artif. Intell.*, vol. 101, pp. 104211, May. 2021.

[2] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Rob. Auton. Syst.*, vol. 100, pp. 171–185, Feb. 2018.

[3] B. Paden, M. Čáp, S. Z. Yong, D. Yershov and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[4] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, AB, Canada, 2005, pp. 2210–2215.

[5] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2015.

[6] D. Dolgov, S. Thrun, M. Montemerlo and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, Apr. 2010.

[7] J. D. Gammell and M. P. Strub. "Asymptotically optimal sampling-based motion planning methods," *Annu. Rev. Control. Robot. Autom. Syst.*, vol. 4, pp. 295–318, May. 2021.

[8] D. González, J. Pérez, R. Lattarulo, V. Milanés and F. Nashashibi, "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," in *Proc. IEEE Int. Conf. Intell. Transport. Syst.*, Qingdao, China, 2014, pp. 1430–1435.

[9] T. Schoels, L. Palmieri, K. O. Arras and M. Diehl, "An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *Proc. IEEE Int. Conf. Robot. Autom.*, Paris, France, 2020, pp. 3574–3580.

[10] C. Liu, W. Zhan and M. Tomizuka, "Speed profile planning in dynamic environments via temporal optimization," in *Proc. IEEE Intell. Veh. Symp.*, Los Angeles, CA, USA, 2017, pp. 154–159.

[11] A. Artuñedo, J. Villagra and J. Godoy, "Jerk-limited time-optimal speed planning for arbitrary paths," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8194–8208, Jul. 2022.

[12] J. Zhou, R. He, Y. Wang, and S. Jiang, "Autonomous driving trajectory optimization with Dual-Loop iterative anchoring path smoothing and Piecewise-Jerk speed optimization," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 439–446, Apr. 2021.

[13] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res.*, vol. 28, no. 8, pp. 933–945, June. 2009.

[14] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 5628–5635.

[15] X. Li, Z. Sun, Z. He and D. Liu, "A practical trajectory planning framework for autonomous ground vehicles driving in urban environments," in *Proc. IEEE Intell. Veh. Symp.*, Seoul, Korea (South), 2015, pp. 1160–1166.

[16] P. Fiorini and Z Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, July. 1998.

[17] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Prague, Czech Republic, 2021, pp. 8097–8104.

[18] E. G. Szádeczky-Kardoss and Z. Gyenes, "Velocity obstacles for car-like mobile robots: Determination of colliding velocity and curvature pairs," *Adv. Sci. Technol. Eng. Syst. J.*, vol. 3, no. 1, pp. 225–233, Jan. 2018.

[19] D. Silver, "Collaborative pathfinding," in *Proc. AIIDE.*, 2005, pp. 23–28.

[20] Z. Ren, S. Rathinam, M. Likhachev and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8154–8161, July. 2022.

[21] J. Lin, T. Zhou, D. Zhu, J. Liu and M. Q. H. Meng, "Search-based online trajectory planning for car-like robots in highly dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, Xi'an, China, 2021, pp. 8151–8157.

[22] C. Cao, P. Trautman and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Montreal, QC, Canada, 2019, pp. 5551–5557.

[23] Z. Ajanovic, B. Lacevic, B. Shyrokau, B. M. Stolz and M. Horn, "Search-based optimal motion planning for automated driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Madrid, Spain, 2018, pp. 4523–4530.

[24] B. Brito, B. Floor, L. Ferranti and J. A. Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4459–4466, Oct. 2019.

[25] W. Ding, L. Zhang, J. Chen and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robot. Automat. Lett.*, vol. 4, no. 3, pp. 2997–3004, Jun. 2019.

[26] M. Werling, J. Ziegler, S. Kammel and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," in *Proc. IEEE Int. Conf. Robot. Autom.*, Anchorage, AK, USA, 2010, pp. 987–993.

[27] H. L. Chiang, B. HomChaudhuri, L. Smith, and L. Tapia, "Safety, challenges, and performance of motion planners in dynamic environments," in *Robot. Res.*, N. Amato, G. Hager, S. Thomas, and M. Torres-Torriti, Eds. Springer: Cham, 2020, pp. 793–808.

[28] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *Proc. IEEE Intell. Veh. Symp.*, Dearborn, MI, USA, 2014, pp. 376–381.

[29] T. M. Howard and A. Kelly, "Optimal rough terrain trajectory generation for wheeled mobile robots," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 141–166, Feb. 2007.

[30] J. Zhang, C. Hu, R. G. Chadha and S. Singh, "Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation," *J. Field Robot.*, vol. 37, no. 8, pp. 1300–1313, Dec. 2020.

[31] A. Lerner, Y. Chrysanthou and D. Lischinski, "Crowds by example," *Comput. Graph. Forum.*, vol. 26, no. 3, pp. 655–664, Sep. 2007.