IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 3, 2018, accepted June 5, 2018, date of publication June 8, 2018, date of current version July 6, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2845448

Hybrid Trajectory Planning for Autonomous Driving in Highly Constrained Environments

YU ZHANG[1], HUIYAN CHEN[1], STEVEN L. WASLANDER[2], (Member, IEEE), JIANWEI GONG[1], (Member, IEEE), GUANGMING XIONG[1], TIAN YANG[1], AND KAI LIU[1]

[1]School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China
[2]Institute for Aerospace Studies, University of Toronto, Toronto, ON M3H 5T6, Canada

Corresponding author: Jianwei Gong (gongjianwei@bit.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 91420203 and Grant 61703041.

ABSTRACT In this paper, we introduce a novel and efficient hybrid trajectory planning method for autonomous driving in highly constrained environments. The contributions of this paper are fourfold. First, we present a trajectory planning framework that is able to handle geometry constraints, nonholonomic constraints, and dynamics constraints of cars in a humanlike and layered fashion and generate curvature-continuous, kinodynamically feasible, smooth, and collision-free trajectories in real time. Second, we present a derivative-free global path modification algorithm to extract high-order state information in free space for state sampling. Third, we extend the regular state-space sampling method widely used in on-road autonomous driving systems to a multi-phase deterministic state-space sampling method that is able to approximate complex maneuvers. Fourth, we improve collision checking accuracy and efficiency by using a different car footprint approximation strategy and a two-phase collision checking routine. A range of challenging simulation experiments show that the proposed method returns high-quality trajectories in real time and outperforms existing planners, such as hybrid A* and conjugate-gradient descent path smoother in terms of path quality, efficiency, and computation resources used.

INDEX TERMS Trajectory planning, motion planning, autonomous driving, obstacle avoidance, kinodynamic constraints, collision checking.

I. INTRODUCTION

During the past few decades, autonomous driving techniques have attracted a great deal of attention in both academia and industry due to their promising potential to prevent collisions due to human error, reduce traffic congestion and emissions, and provide mobility to all, including people who are unable to drive themselves [1]–[3]. As a key module to the autonomous driving system, trajectory planning plays an important role in guaranteeing the ride safety and comfort by generating a smooth, dynamically-feasible, collision-free trajectory towards a destination, while taking into account obstacles around the vehicle, vehicle dynamics, traffic rules and other task specific constraints.

In general, a trajectory planning problem refers to finding an optimal path with time stamped positions, orientations, and velocities from the current configuration to the goal configuration by minimizing certain objectives subject to geometry constraints (feasible paths must lie in the free space), task constraints (requirements to visit certain intermediate targets), and nonholonomic constraints (vehicle kinematics and dynamics). This kind of problem is known to be PSPACE-hard, which means there is no polynomial-time algorithm able to solve all instances of the problem [4], [5]. The compounding effect of geometry, task and and nonholonomic constraints (differential constraints) increases the difficulty of developing practical trajectory planning algorithms that directly solve the complete trajectory planning problem [6], [7]. Instead, many different methods have been developed to address the trajectory planning problem by approximating the solution or solving sub-problems of the original one.

A. RELATED WORK

The methodologies used by the motion planning community for autonomous driving can be divided into three categories: **sampling-based methods**, **search-based methods** and **optimization-based methods**. They all have their own strengths and weaknesses.

Sampling-based methods can be further categorized into random sampling-based methods and deterministic sampling-based methods. Random sampling-based methods include probabilistic roadmaps [8]–[10] and rapidly-exploring random trees [11]–[15]. The latter family has been widely adopted for autonomous driving applications. Kuwata *et al*. [12] develop a closed-loop RRT method by combining physical, logical bias RRT with a low-level controller to forward simulate the path. Ryu *et al*. [13] exploit a more efficient RRT method with a different biased sampling strategy to generate the random tree and then filter out the final path with a low pass filter over the tangential vector and curvatures of the path. Although various sampling strategies and corrections are applied to improve the performance of the RRT method, the solution generated remains sub-optimal. Karaman *et al*. [10], [16] develop the RRT* algorithm that is able to converge to the optimal solution asymptotically. Jeon *et al*. [14] incorporate a fast local steering algorithm based on a half-car dynamical model and the RRT* algorithm to generate dynamically feasible trajectory for high-speed autonomous driving. The random sampling-based methods are in general good at exploring the reachability of the free space using control space paths and rapid collision checking in complex environments. The tree structure generated by sampling strategies also approximates the connectivity of the free space without having access to the explicit geometry model of the free space [5]. Some random sampling-based methods such as PRM and RRT are known to be probabilistically complete, ensuring a path will eventually be found if one exists [15]. Due to the repeated selection of the random intermediate states or control inputs, however, resulting paths tend to be jerky, redundant and not curvature continuous, which is not acceptable for autonomous driving applications, especially at high speed. The run-time is also unlimited and unpredictable. For a more comprehensive review of random sampling-based methods, we refer readers to [17].

Deterministic sampling-based methods are also referred to as local search methods [1], which mainly include control state sampling methods and state space sampling methods. The main approach is to generate several shifted curves from the car's current configuration to sample the space near the car and arrive at a set of goal locations, and then to select the best path according to factors such as whether the path is collision free and the distance to the reference path or the center line of lanes etc. Control space sampling methods simply run discrete control inputs for a certain duration to generate the shifted curves [18], [19], while state space sampling methods instead get goal states from a reference path or a center line of the road within a certain horizon, then solve a two-point boundary value problems with predefined curve models connecting the initial state of the car and the goal state set [20]–[25]. The curves that are used to sample the space can be arcs (generated from control space [18], [19]), spirals [20], [21], splines [22], [26], polynomials [23]–[25]. All the curve models except arcs required goal states such as position, orientation, and curvature to be defined from a reference path in order to construct candidate paths. The goal state plays an import role in affecting the shape of the seeds paths. Compared to random sampling-based methods, deterministic sampling-based methods reduce the solution space dramatically by using preferred actions (control space sampling) or taking advantages of the structure of roads (state space sampling), which makes behaviors of the planner predictable and also avoids unreasonable sampling seeds. The deterministic sampling-based methods can be thought of as a class of planners whose sampling depth is one, regardless of their look-ahead horizon, as seen in Figure 1a. Although these methods can generate smooth trajectories efficiently, their ability to capturing the topology of free space is restricted due to the limited sampling depth. The reliance on a goal state generated from the reference path also discards key information such as orientation and curvature in terms of obstacle avoidance with the presence of obstacles, as shown in Figure 2. The state sampling method work reasonably well without obstacles or with sparse distributed obstacles along the road, but is fundamentally limited in the complexity of the environments with dense obstacles that can be handled effectively [25], [27]. The presence of dense obstacles break the original geometry property of the road exploited by deterministic sampling methods. Following the road heading or curvature information underestimates the steering efforts required to avoid consecutive obstacles and misleads the motion planning process. In practice, a car may run into situations in which it cannot make any progress along the road, despite a feasible trajectory existing, due to the restricted search imposed by these methods.

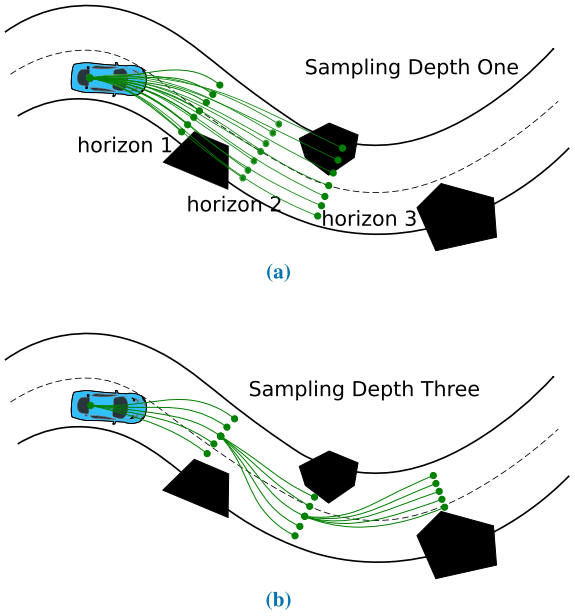


FIGURE 1. (a) The demonstrated sampling depth is one. With the different sampling horizons and many sampling paths, the sampling depth is still one. (b) The demonstrated sampling depth is three.

Search-based methods, as distinct from sampling-based methods, refer to methods which employ graph-based search

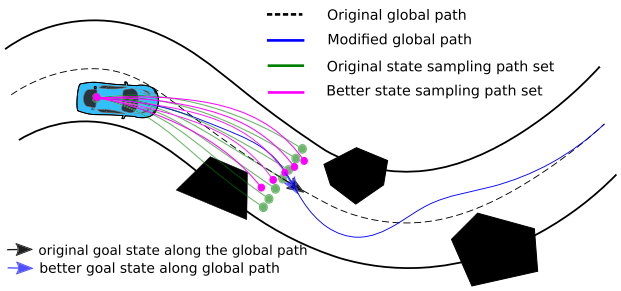


FIGURE 2. The original goal state underestimates the steering efforts and also provides an inperfect heading information, which may fail the navigation.

approaches such as A* [28]–[30], state lattice [31]–[33], Dijkstra [34], hybrid A* [35] and their variants. Some search-based methods provide a theoretical guarantee of resolution completeness. This class of methods usually constructs a directed graph to sample the configuration space uniformly and then searches the graph with different search strategies. Instead of randomly exploring the free space of the environment at run-time, search-based methods discretize the configuration space with fixed number of motion primitives and build the search graph in advance. The graph that is built has a better coverage of the whole free space with less configuration states compared to RRTs at the early stages of planning, which must construct their search tree from scratch. This distinction ensures that search-based methods have a better global view of the geometry model of free space before searching for a path. The search strategies only focus on finding a globally optimal path according to given objectives. With a well designed search space and admissible heuristics, searched-based methods are able to find a globally optimal or sub-optimal path in real-time for some autonomous driving scenarios [28], [35]. Hybrid A* and AD* have shown strong performance in path planning for autonomous driving in parking lots scenarios [21], [28], [35]. However, as the pre-constructed graph is a discrete representation of the configuration space, the solution space will be dramatically reduced if the search space is not well designed. This may lead to highly sub-optimal solutions or even no solution while one exists. When it comes to high dimensional problems, graph search-based methods inevitably fall prey to the *curse of dimensionality*. In addition, the resulting paths are not curvature continuous, which is not feasible for high speed autonomous driving. A post smoothing process can be applied to the generated path to make it feasible for on-road driving scenarios, but the collision-free property of the path generated by searching does not remain after smoothing [35]. More adjustments are then required to make it safe to follow. In summary, search-based methods are good at solving path planning problems with geometric constraints but poor at dealing with kinodynamic constraints of the system in general.

Finally, **optimization-based methods**, also called variational methods, solve two point boundary value problems using nonlinear optimization techniques to address the trajectory planning problem [25], [35]–[39]. These methods usually represent the path of a vehicle with certain parametrized curve models, such as splines [40], [41], spirals [37], [42], polynomials [43], [44] or piecewise lines [38], [45], and then optimize for the prescribed objectives over a finite dimensional parameter vector space to get smooth paths or trajectories. They are widely used in autonomous driving systems due to their fast convergence rate to local optimum and high quality solutions. However, most approaches are non-convex, and are challenged to find the global optimum unless an initial guess that is sufficiently close to optimal is available. The complexity of environments depends on the geometry shape of the free space, and not just on the number of obstacles. To the best of our knowledge, there is no optimization-based method that incorporates the explicit geometry model of the free space into the trajectory planning problem formulation. Further, the curve model chosen in problem formation for trajectory planning also limits the ability of handling obstacles. For example, the quadratic polynomial curve model can only represent a single swerve motion. Therefore, the trajectory optimization based on a quadratic polynomial parametrization can only deal with one obstacle within its planning horizon. A piecewise continuous curve model is able to resolve this limitation, but it leads to computation of optimization. This requires solving multiple coupled optimization problems at the same time. The intermediate states need to be chosen smartly as well. To sum up, optimization-based trajectory planning methods are good at providing high quality solutions with embedded high order curves or dense waypoint curves but poor at handling obstacles. Some of the methods can incorporate nonholonomic constraints in the optimization, but including obstacle avoidance within the optimization is still problematic.

B. CONTRIBUTIONS

In order to solve the motion planning problem in real time for autonomous driving, we propose a hybrid trajectory planning algorithm by combining the strengths of different methods. We assume the drivable region and the global path, which may be or may not be collision-free, are provided. As an optimization problem with both nonholonomic and geometry constraints is PSPACE-hard [4], [5], the trajectory planning problem is decomposed into spatial path planning and speed planning sub-problems. The spatial path planning problem is further decomposed to a global path modification layer within the given drivable region and a multi-phase state space sampling planner layer. The speed planning sub-problem is solved by an optimization-based speed planning layer along the fixed path. In this way, the combinatorial constraints of the motion planning problems are separated, which is convenient to address different constraints by taking advantages of different methods. Besides, the decomposition makes the planner be able to limit the search space by leveraging the constraint information while still maintaining richness of

the feasible solution space in different layers. Our contributions are:

- We propose an efficient layered trajectory planning framework. This framework handles each type of constraint using distinct methods. We first handle the geometry constraints using a search-based global path modification layer, which identifies the shortest path through the environment without regard for vehicle motion constraints. The second layer, a multi-stage state sampling method, samples in the neighbourhood around the shortest path to generate a kinematically-feasible path, which resolves the nonholonomic constraints of the vehicle. In this way, the framework reduces the infeasible state space region from the potential sampling space significantly. More precisely, it prevents the state space sampling method from spending computation resources on generating high-cost kinematically-feasible paths to sample the infeasible region that is limited by constraints, which improves the sampling efficiency. Once the best path is found in the sampling stage, the dynamical constraints of the vehicle (lateral accelerations and longitudinal accelerations and decelerations) are imposed to define a speed profile. This approach differs from the hybrid A* and RRT* families of methods, which first impose nonholonomic constraints to generate kinematically-feasible motion primitives, then incrementally construct a sampling graph using motion primitives to resolve the geometry constraint. Their strategy leads to swerving paths due to the discretized or randomized motion primitives. Our method results in more human-like solutions, by finding the best coarse path through the environment first, and then refining the path to meet kinodynamic constraints.
- We introduce a general global path modification algorithm with a derivative-free smoothing process to extract high order state information for the state space sampling. A big difference from the origin space exploration method used in the heuristic search path planning framework [46] is that we provide a derivative free path smoothing algorithm based on the curve energy function and fit a B-spline to get the precise orientation and curvature information of the path. Chen [46] uses the resulting circle path to limit the search region for the hybrid A*, which only leverages the position information of the circle path. We instead exploit the position, orientation and curvature information of the smoothed circle path, leading to more drivable paths. Gu *et al.* also propose global path deformation methods (dynamic programming method [47], elastic band method [48]). However, these methods require precise road geometry information to construct the search graph. Our method only require traversable region information, which can be easily extended to free space planning scenarios.
- We extend the regular state space sampling method to a multi-phase deterministic state space sampling method to handle navigation problems in complex and crowded environment. Different from the regular state space sampling methods [21], [25], [49] that use a single polynomial function to represent naive maneuvers with only one swerve, our method divides the path planning problem with nonholonomic constraints to several consecutive stages and uses curvature-continuous piecewise cubic spirals to represent complex maneuvers (consecutive s shape paths) subject to nonholonomic constraints of cars. In this way, our planner is not only able to get rid of high order polynomials that leads to heavy computations but also provides a long-term path with a continuous curvature profile aligned with the geometry of the modified global path.
- We developed a more efficient and accurate collision checking method by using a different footprint approximation strategy and a two-phase collision checking routine.

This paper is organized as follows. The methodology is presented in Section II, which includes the following subsections: the global path modification, multiple stage state sampling, optimization-based speed planning, hierarchical collision checking. Section III explains the experiments setup and evaluation of the proposed algorithm in details. Section IV summarizes the contributions and discusses future work.

II. METHODOLOGY

The overall structure of the hybrid trajectory planning framework is shown in Fig. 3. We first create a traversable region around the global path and set the intersection of the traversable region and on-board free-space perception results as the search space of the global path modification layer. This pre-processing step restricts the region of interest and filters irrelevant obstacles from the motion planning problem. Next, the global path modification layer employs a novel space exploration method to identify a collision-free and smooth path inside the aforementioned search space. A multiple stage state sampling layer is then applied to sample along the new reference path and identifies a collision-free, kinematically-feasible, curvature-continuous desired path. Finally, an optimization-based time-optimal speed planning layer is employed to construct a speed profile along the path, taking into account the dynamic constraints of the vehicle.

A. GLOBAL PATH MODIFICATION

The goal of the global path modification layer is to get a collision-free global path within the region limited by the original global path and the traversable region from the perception module. In realistic on-road driving scenarios, the original global path provided by a high-level route planning module will not be optimal, and may not even be feasible due to the presence of obstacles such as parked cars along the roads and reconstruction cones which appear temporarily. In a highly curved lane with tight turning radius without obstacles, following the center line of the lane results in more control

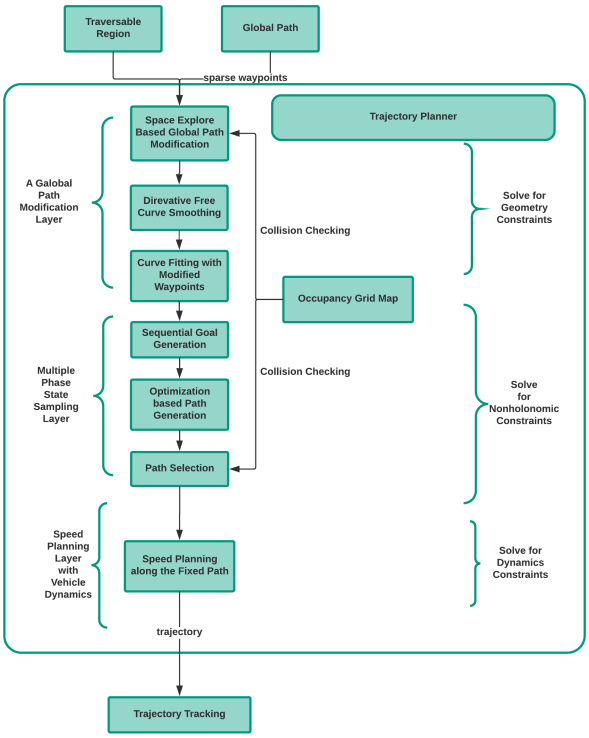


FIGURE 3. Hybrid trajectory planning framework.

efforts and larger lateral accelerations for the car than are necessary [25]. Improvements to the path cost can be made by short-cutting the corner within the road boundaries in this case.

In order to solve these problems, we propose a global path modification method to provide correct guidance information for obstacle avoidance of the multiple phase state sampling layer. The core idea is to reconstruct the topology of the on-road free space and generate a new smooth global path embedded with the geometric information of the free space within the boundary defined by the original global path, shown in Fig. 4. The resulting path in this step does not necessarily have to consider the nonholonomic constraints, but needs to be as continuous and short as possible. We adapt existing space exploration methods to modify the non-collision-free global path according to the on-board perception information. After that, a derivative-free iterative path smoothing algorithm is developed to reduce the slackness of the path generated from the search. Finally, a B-spline curve fitting algorithm is applied to generate the high order geometry information of the modified global path.

For completeness, we recall the original space explore algorithm from [46]. Given the current start position of the car and the desired goal state on the global path, the space exploration algorithm is to find a sequential overlapped circle path in workspace using a heuristic A* graph search algorithm. The centers of the circles are waypoints of the path. The circle node used to construct the search graph is a tuple $n_i = (p_i, r_i, g_i, h_i, f_i)$, where p_i is the position (x_i, y_i)

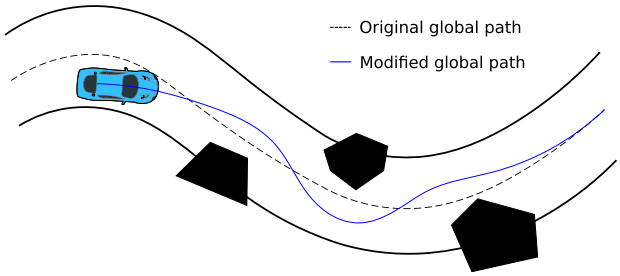


FIGURE 4. Due to the presence of obstacles, the geometry of the roads is changed. The modified path captures the geometry properties of the free space, which provides more high-order state information for state sampling.

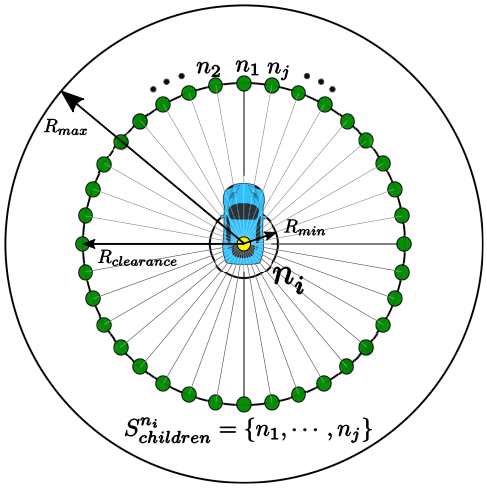


FIGURE 5. The node expanding pattern. The n_i is the parent node. $R_{clearance}$ is the radius acquired from the grid map according to current situation and radius limits $[R_{min}, R_{max}]$.

of the circle center(vertex of the graph), r_i is the clearance radius of the circle, g_i is the actual travel distance cost (the length of edges between nodes) from the start node to the current node and h_i is the Euclidean distance heuristic from the current node to the goal node. The summation of g_i and h_i is f_i, which is the total cost, and is a lower bound on the cost to travel through n_i from start to goal. The exploration begins with the start node n_{start} pushed to the open set S_{open}. The open set S_{open} stores the nodes to be explored and sorts the nodes according to the f_i in descending order. The closed set S_{closed} maintains the nodes that have been explored. At each iteration, the $\texttt{PopTop}(S_{open})$ operation will pick the node n_i with the minimum f_i value to expand. The $\texttt{ExpandNode}(n_i, j, R_{min}, R_{max})$ operation generates the children node set $S_{children}$ for n_i, with j elements evenly distributed in a circle around n_i with radius

$$r_i = \begin{cases} R_{clearance}, & if\ R_{clearance} \in [R_{min}, R_{max}] \\ R_{min}, & if\ R_{clearance} < R_{min} \\ R_{max}, & if\ R_{clearance} > R_{max}, \end{cases}$$

as shown in Fig. 5, where the $R_{clearance}$ is the max distance from n_i to the closest obstacle in clearance map.[1] The R_{min}

[1]The clearance map is the same with the one explained in section II-D

Algorithm 1 The Space Exploration Algorithm

Data: n_{start} The start node.
n_{goal} The goal node.
Result: $\{n_{start}, \cdots, n_{goal}\}$ or $\emptyset$

$S_{open} \leftarrow \{n_{start}\}$;
$S_{closed} \leftarrow \emptyset$;
$f_{goal} = \infty$;
while $S_{open} \neq \emptyset$ **do**
  `Sort` (S_{open}) ;
  $n_i \leftarrow$ `PopTop` (S_{open}) ;
  if $f_{goal} < f_i$ **then**
    break;
  else if $n_i \notin S_{closed}$ **then**
    $S_{children} = \{n_1, \cdots, n_j\} \leftarrow$ `ExpandNode` $(n_i, j, R_{min}, R_{max})$;
    $S_{open} \leftarrow S_{children} \bigcup S_{open}$;
    if `Overlap` (n_i, n_{goal}) **then**
      $f_{goal} = min(f_i, f_{goal})$;
    $S_{closed} \leftarrow \{n_i\} \bigcup S_{closed}$;
  else
    continue;
if $f_{goal} < \infty$ **then**
  return *success*
else
  return *failure*;

and R_{max} are the lower and upper bound for the radius of expanding circle nodes, respectively. Then, the children nodes of n_i in $S_{children}$ are added to S_{open} and the expanded node n_i is moved from S_{open} to S_{closed}. The total cost, f_{goal}, is initialized to infinity, and updated with a new value and associated path whenever the expanded node, n_i, overlaps with the goal node n_{goal} and a smaller cost to goal is found. The `Overlap`(n_i, n_j) is defined as the **Algorithm** 2. The process continues until all the nodes in the S_{open} are explored or have a higher total cost than f_{goal}.

Algorithm 2 `Overlap`(n_i, n_j)

Data: n_i The circle node i.
n_j The circle node j.
Result: *true* if overlapped, *false* otherwise
Operation: `Distance` Calculate the Euclidean distance of two points in plane.

if $\left(\texttt{Distance}\,(p_i, p_j) - \texttt{Max}\,(r_i, r_j)\right) \leq 0.5 * \texttt{Min}\,(r_i, r_j)$ **then**
  return *true*
else
  return *false*

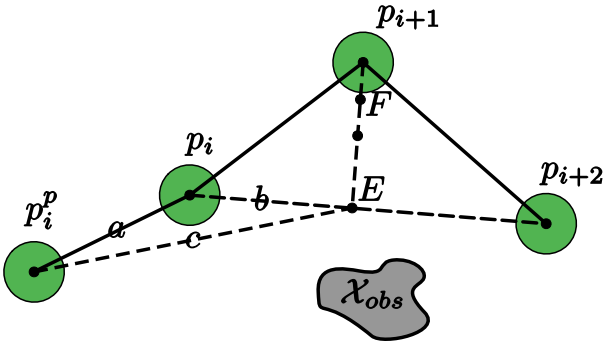


FIGURE 6. The update direction of p_{i+1}. The symbol $\mathcal{X}_{obs}$ represents an obstacle.

The output of this process is a discrete curve connecting the start and goal nodes through the free space in the environment. More precisely, a *discrete curve* is an ordered tuple $(p_0, p_1, \cdots, p_{n-1}) \in \mathbb{R}^{2n}$ of vertices $p_i \in \mathbb{R}^2$, with $p_{i+1} \neq p_i$ for all i. The consecutive vertices are connected by straight lines that are called edges.

Although the space exploration layer is able to extract the free space knowledge to approximate the optimal corridors to reach the temporary goal in 2D workspace in terms of the path length, the resulting path (red curve) generated by this algorithm is only G^0 continuous[2] as shown in Fig. 7. It is inadmissible for state sampling along the path due to the discontinuous heading profile. In order to smooth the path and extract high order state information for the state sampling module, an iterative path smooth algorithm is applied to refine the path. The core idea of path smoothing is reducing the slackness of the path (shortening the length) while increasing the smoothness. We define an energy function for a discrete curve to measure the progress of smoothing:

$$J_d = \sum_{i=2}^{N} \frac{(\kappa_{p_{i-1}}^2 + \kappa_{p_i}^2)\Delta s_{p_i}}{2} \tag{1}$$

where κ_{p_i} is the discrete curvature at point p_i, and Δs_{p_i} is the distance between point p_i and p_{i-1}.

This is an approximation of the continuous version of the bending energy representation

$$J = \int_0^s \kappa^2 \, ds$$

The objective is used to define the stop criteria for the iterative smoothing algorithm. The update rule is defined as the **Algorithm** 3: Given the adjacent three points p_i, p_{i+1}, p_{i+2} and the center point p_i^p of the parent node of p_i on the path, try to push the p_{i+1} to point E while still keep the clearance greater than R_{min} and $\kappa_{p_i} \leq \kappa_{max}$. **Algorithm** 3 calculates the discrete curvature according to the SSS Theorem [50] (line 3-5). The initial position of E is at the intersection point of $\overrightarrow{p_i p_{i+2} p_i p_{i+1}}$ with the perpendicular line going through the p_{i+1} shown in Fig. 6.

[2] G represents the geometry continuity. G^0 means a curve is joined.

Algorithm 3 GetNewCircleCenter (p_i, p_{i+1}, p_{i+2})

Data: p_i,p_{i+1},p_{i+2} The adjacent three points of nodes.
Result: p_{i+1}^{new} The new position of p_{i+1}
Parameter: R_{min} The lower boundary for the radius of a circle node.
κ_{max} The max curvature defined by users according to platforms.
p_i^p The center of the parent node of p_i.
resoution The resolution of the grid map.
Operation: `Clearance` Acquire the clearance value according to the position in the map.
`Distance` Calculate the Euclidean distance of two points in plane.

$E \leftarrow (\overrightarrow{p_i p_{i+2}} \perp \overrightarrow{p_{i+1}E}) \wedge (E \in \overrightarrow{p_i p_{i+2}})$;
do
  $a = \overrightarrow{p_i^p p_i}, b = \overrightarrow{p_i E}, c = \overrightarrow{p_i^p E}$;
  $s = \frac{a+b+c}{2}, K = \sqrt{s(s-a)(s-b)(s-c)}$;
  $\kappa_{p_i} = \frac{4K}{abc}$;
  if (`Clearance` $(E) \geq R_{min}) \wedge (\kappa_{p_i} \leq \kappa_{max})$ **then**
    $p_{i+1}^{new} \leftarrow E$;
    return p_{i+1}^{new} ;
  else
    $E \leftarrow \frac{E+p_{i+1}}{2}$;
while `Distance` $(E,p_{i+1}) >$ *resolution*;
return p_{i+1} ;

Algorithm 4 PathSmoothing($\{p_i\}$, R_{min},R_{max})

Data: $\{p_i\}$ The discrete curve consists of positions p_i of the circle nodes n_i.
Result: *refined_path* The smoothed discrete curve.
Parameter: *threshold* The optimization stop threshold defined by users.
Operation: `Clearance` Acquire the clearance value according to the position in the map.

do
  refined_path $\leftarrow \{p_i\}$;
  $J_{prev} = J_d(refined\_path)$;
  foreach $n_i \in \{p_i\} \setminus \{p_{start}, p_{goal}\}$ **do**
    $p =$ `GetNewCircleCenter` (p_{i-1}, p_i, p_{i+1}) ;
    $n_i \leftarrow (p,$ `Clearance` $(p))$;
  new_refined_path $\leftarrow \{p_i\}$;
  $J_{new} = J_d(new\_refined\_path)$;
  $\Delta J = \frac{\|J_{new} - J_{prev}\|}{J_{new}}$;
  if $J_{new} < J_{pre} \wedge \Delta J <$ *threshold* **then**
    refined_path = *new_refined_path*;
    Return *refined_path* ;
while $J_{new} < J_{pre}$;
Return *refined_path* ;

As the update rule and objective function are defined, the path smoothing algorithm performs as in Algorithm 4. Every iteration the algorithm will work to decrease the slackness of the path by performing a binary search for a feasible updated circle center, subject to the triangle inequality rule, curvature upper boundary and the clearance constraint. If the distance between the new circle center (e.g. F) and original circle center (e.g. p_{i+1}) falls below the resolution of the grid map, it means there is no improvement available. The update will skip the circle center to optimize others. The blue piecewise linear curve shows the optimized path of original result of the space explore algorithm (red curve) in Fig. 7. It is obvious that there is a great improvement after smoothing. But the optimized path are still piecewise linear and sparse, which only provides rough heading and curvature information. The orientation and curvature profiles calculated based on the optimized path are discontinuous and imprecise. In addition, the geometry information of intervals of waypoints is missing due to the linear approximation of a path. But as we mentioned before, the high order state information such as orientation and curvature have great impacts on the shape of paths. Smooth orientation and curvature profile are preferred to prevent sudden changes of paths during state sampling. It's well known the B-spline has C^2 continuity for the entire curve. By fitting a B-spline over the optimized path, we manage a smooth transition between waypoints and reconstruct the geometry information of intervals between waypoints as well. Although B-spline doesn't pass through the data points, it does stay close to them. Our algorithm generates waypoints based on clearance, which provides enough support waypoints for curve fitting. Thus, the fit B-spline still maintains a good approximation of the geometry of the circle path. The green curve in Fig. 7 shows the smoothed path after the curve fitting process.

B. MULTIPLE-PHASE DETERMINISTIC STATE SAMPLING

This section is interested in solving a path planning problem of the following class: Given an initial state of a vehicle and a reference path, plan a smooth collision-free path aligned with the reference path while respecting nonholonomic constraints of vehicles.

Once a modified smooth global reference path is generated, a multiple stage deterministic state sampling method is used to generate collision-free, kinematically-feasible paths for cars by sampling along the modified global path. Different from the regular state sampling methods [20], [21], [25] that treat state sampling as a single stage sampling problem, our method divides state sampling into several consecutive stages, as shown in Fig. 8. In every stage, state sampling is employed to generate a finite set of seeding paths with different lateral offsets shifting from the reference path, which results in solving several two-point boundary value problems (TPBVPs). The closest collision-free path to the reference path at every stage is selected as the start state of the next stage. By doing so, a greedy search strategy is employed.

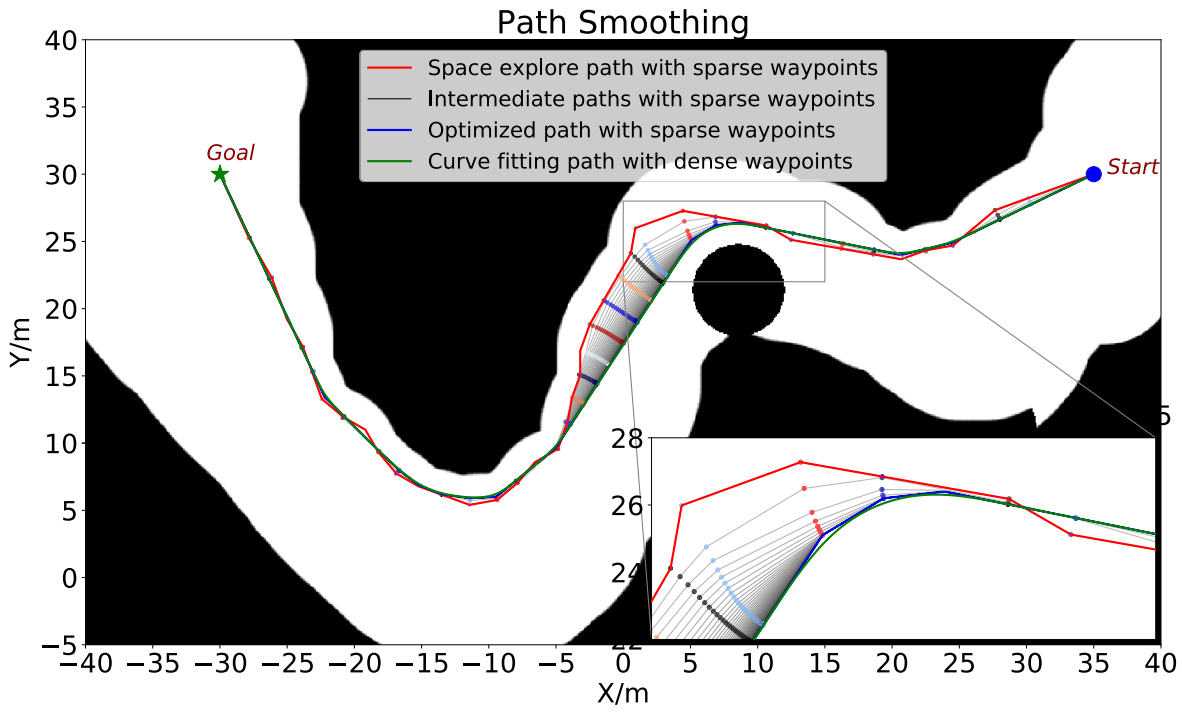


FIGURE 7. The path smoothing results in unstructured environment. The black region represents obstacles and white region is free space. The red curve represents the circle path generated by the space explore algorithm. The colorred dots show the centers of circle nodes. The grey curves are the intermediate path smoothing results. The blue curve is the final circle path generated by the smoothing algorithm. The green curve is the curve fitting result with dense way points.

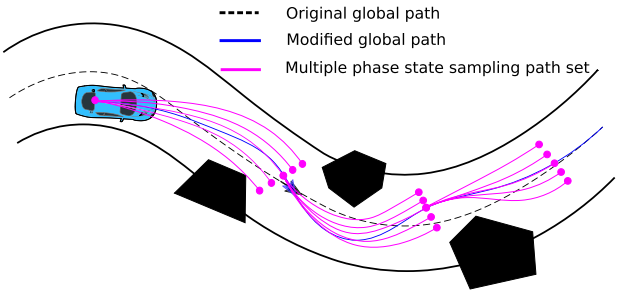


FIGURE 8. Mutiple phase state sampling.

The rational behind this will be explained in II-B.2.b part (Path Selection in Single Phase).

1) REFERENCE PATH SEGMENTING

Unlike the simple, single swerve, reference paths aligned with road geometry used in [20], [21], and [25], our smooth reference path may include several swerves with different turning directions when navigating through environments with dense obstacles. More waypoints or higher order polynomials are needed to approximate the exact shape of a path, which could lead to heavy computational requirements for our system. Instead, we divide the complex reference path into several segments sampled with simple motion primitives and use a cubic spiral representing each path segment to reduce the dimensions of the parameter space while still maintaining strong expressiveness over the set of possible paths. The reference path can be segmented based on either arc-length (distance along a curve) of the motion primitives or the curvature profile of the reference path. In our implementation, we use the former, which is set to $l_{phase} = 6.0$ m. In this way, a complex motion can be approximated by piecewise cubic spirals.

2) SINGLE PHASE STATE SPACE SAMPLING

In single phase state space sampling, the base goal state is acquired from the reference path segmenting module, and a set of goal states are generated with different lateral offsets from the base goal state at every stage. Different from the goal states lateral offsetting strategy used in [20], [21], [25], and [51], we exploit an adaptive offset sampling to determine feasible goal states to generate a kinematic-feasible path. The lazy sampling strategy used in [20], [21], [25], and [51] is shown in Fig. 9, which keeps a large fixed number of sampling points with equidistant offsets to cover the road region and generates computationally-expensive paths without checking the feasibility of the sampling points. When the dimension of the obstacle increases, more samples are needed to generate collision-free path candidates, and the number of samples required is challenging to determine for different scenarios. The calculation time increases as well in this case. In contrast, thanks to the guidance information provided by the global path modification layer, our method only needs to maintain a small set of sampling points to

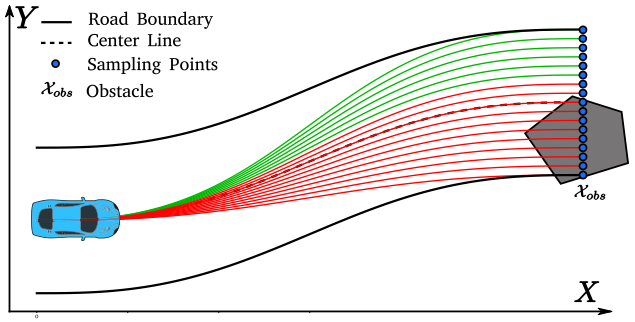


FIGURE 9. Lazy sampling strategy.

nudge around the modified global path and get collision-free and kinematic-feasible path candidates (see Fig. 10), since the modified global path already lies in free space. The total sample count of our method for one stage will not be affected by the dimension of obstacle as the locations of samples depend on the modified global path. Further, we check every sampling point for collision using the footprint of the car and the infeasible sample points are filtered out of the total sample set. Only collision-free goal states (effective sampling points in Fig. 10) that are picked adaptively according to environments, based on the car footprint, are passed to the path generation module. In this way, the computationally-expensive task of path generation is avoided when a goal state cannot be reached.

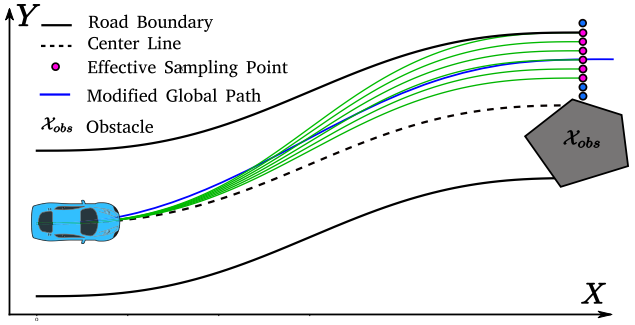


FIGURE 10. Adaptive sampling strategy.

As the current vehicle state and the goal state are known, the path generation can be formulated as a classical Two-Point-Boundary-Value problem (TPBVP) then solved using nonlinear programming techniques. Before converting a TPBVP problem to a constrained nonlinear programming problem, a parameter representation of the path needs to be determined to formulate the problem. Several parameterized path representations have been proposed, including B-spline [40], [41], Bezier Curve [52], [53], piecewise linear curve [38], clothoid [54]–[56], polynomial curve [43], [44], and polynomial spiral (greater than second order) [37], [42]. The B-spline, Bezier curve, piecewise linear curve and polynomial curve do not satisfy kinematic constraints of the car model according to their curve definitions and independent parametrization for x and y in Cartesian coordinates.

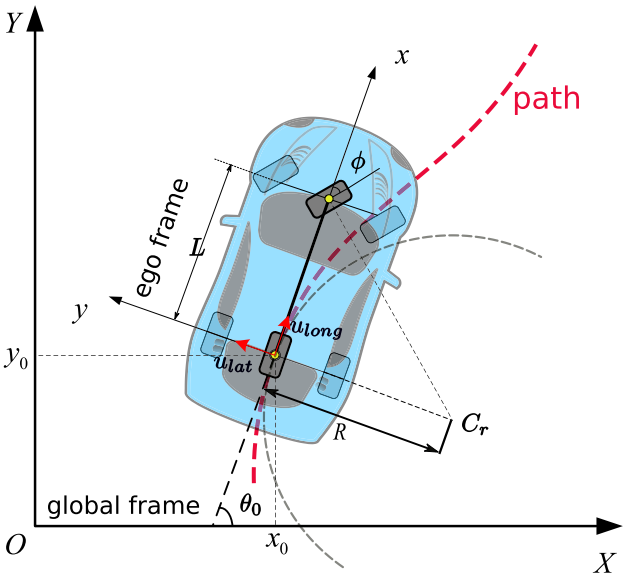


FIGURE 11. A single track vehicle model

The polynomial curve is not able to represent a simple curve like an arc or a line and may dramatically change with small perturbations. The clothoid is known as a first order polynomial spiral and widely used in both road design [57] and motion planning. However, clothoids only have three parameters and therefore cannot satisfy tangent conditions and curvature constraints at both boundary points [37] simultaneously. A cubic polynomial spiral such as (2) is sufficiently expressive to satisfy positional constraints, tangency conditions, and curvature constraints while still maintaining the acceptable low order in parametrization. A curve is said to be parameterized by arc-length if its speed is always unit speed [58]. It is well known that every regular curve has an arc-length parametrization [59]. By representing the control input function as a cubic spiral (see (2)) as in [37], an arc-length parametrized curve for the path of a vehicle can be found directly with the vehicle model.

$$\kappa(s) = a + bs + cs^2 + ds^3 \tag{2}$$

A motion model for a vehicle can be formulated as a set of nonlinear equations in terms of arc-length based on the single track vehicle model [60],

$$\begin{aligned} x(s) &= x_0 + \int_0^{s_f} \cos(\theta(s))ds \\ y(s) &= y_0 + \int_0^{s_f} \sin(\theta(s))ds \\ \theta(s) &= \theta_0 + \int_0^{s_f} \kappa(s)ds \end{aligned} \tag{3}$$

where $x(s)$, $y(s)$ represent the position of the vehicle, $\theta(s)$ is the orientation of the vehicle, $\kappa(s)$ is the curvature of a path. The $\kappa(s)$ can be mapped to a steering angle $\phi(s)$ of the vehicle using $\phi(s) = \arctan(L \cdot \kappa(s))$, where L is the wheelbase. Equation (3) can be quickly derived from a typical motion

model of a vehicle by using $s = Vt$ and $ds = Vdt$ in the time domain, where V is the longitudinal velocity of the vehicle. The motion model (3) defines a vehicle configuration at s by $q(s) = \big(x(s), y(s), \theta(s)\big)$. Since any configuration can be transformed to the origin of a local frame by a homogeneous transformation in $SE(2)$,[3] the initial configuration $q_0 = (x_0, y_0, \theta_0)$ is set to $(0, 0, 0)$.

Such a parametrization that combines (2) and (3) enables our algorithm to optimize the curve depending on the geometric shape of the path embedded with the motion model of the vehicle. In practice, however, it is very difficult to find an analytical expression for the arc-length parameterized curve, due to the fact that a length equation is challenging to integrate analytically. Instead, we can approximate the integral numerically by using the Trapezoidal rule or Simpson's rule [62].

Based on the prior discussion, the arc-length parameterized representation for the path in Cartesian coordinates becomes:

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{p}) &= (x(\boldsymbol{p}), y(\boldsymbol{p})) \\ s.t.\ x(\boldsymbol{p}) &= \int_0^s \cos\left(\int_0^s \kappa(\boldsymbol{p})ds\right) ds \\ y(\boldsymbol{p}) &= \int_0^s sin\left(\int_0^s \kappa(\boldsymbol{p})ds\right) ds \\ \kappa(\boldsymbol{p}) &= a + bs + cs^2 + ds^3 \quad \forall s \in [0, s_f], \end{aligned} \tag{4}$$

where a is the known initial curvature, while b, c, d, s_f are the unknown optimization parameters and s_f is also the final arc-length at the goal state. Let $\boldsymbol{p_c} = (b, c, d, s_f)^T$ be the coefficient parameter vector. The workspace path, $\boldsymbol{r}(s)$ defined with this parameterization, is at least C^2 and at most C^5 continuous.

Instead of using $\boldsymbol{p_c}$ directly in the optimization, we re-parametrize the path by using the curve knot spacing technique that is widely used in curve fitting [63], which is suggested in [42]. Apply a transformation T to $\boldsymbol{p_c}$ and a new parameter vector called a knot parameter vector $\boldsymbol{p}$ is obtained:

$$\boldsymbol{p} = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \end{pmatrix} = \begin{pmatrix} \kappa(\frac{s_f}{3}) \\ \kappa(\frac{2s_f}{3}) \\ \kappa(s_f) \\ s_f \end{pmatrix} = T \cdot \boldsymbol{p_c}, \tag{5}$$

where,

$$T = \begin{pmatrix} \frac{s_f}{3} & (\frac{s_f}{3})^2 & (\frac{s_f}{3})^3 & 0 \\ \frac{2s_f}{3} & (\frac{2s_f}{3})^2 & (\frac{2s_f}{3})^3 & 0 \\ s_f & (s_f)^2 & (s_f)^3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

The choice of parameter vectors $\boldsymbol{p}$ or $\boldsymbol{p_c}$ make no difference in the parametrization for the workspace path, as they are mathematically equivalent. However, this choice of parametrization leads to fast convergence, in our experience and as mentioned in [42].

As $k(s_f)$ is the known curvature at the goal state, the parameter vector is further reduced to $\boldsymbol{p_r} = (p_1^r, p_2^r, p_3^r) = (m_1, m_2, m_4)$.

In addition, this re-parameterization makes the algorithm optimize the parameters that are not merely abstract mathematical symbols, but objects having physical interpretations. In the coefficient parameter space, it is impossible to give a lower and upper bound for parameter values for our problem. But this is trivial in the knot parameter space. A lower boundary, κ_L, and an upper boundary, κ_U, on p_1, p_2 can be provided according to the limitations of steering angle actuation of the vehicle. An informed initial guess of the knot parameters can also be assigned easily.

a: PATH GENERATION PROBLEM FORMULATION

Given the current vehicle state $q_{init} = (x_{init}, y_{init}, \theta_{init}, \kappa_{init})$ and the goal state $q_{end} = (x_{end}, y_{end}, \theta_{end}, \kappa_{end})$, a path generation problem is formulated as

$$\begin{aligned} &\textbf{minimize } J = J_{smoothness}(\boldsymbol{p}_r) = \int_0^{s_f} \|\boldsymbol{\kappa}(\boldsymbol{p_r})\|^2 ds \\ &\textbf{s.t. } q(r(\boldsymbol{p}_r)) = q_{init} \quad \text{when } s = 0 \\ &\quad q(r(\boldsymbol{p}_r)) = q_{end} \quad \text{when } s = s_f \\ &\quad 0 \le s_f \le s_{max}, \end{aligned} \tag{7}$$

where $\boldsymbol{p}_r = (p_1, p_2, p_3)^T$, s_{min}, s_{max} are the lower and upper boundary for the arc-length of the curve assigned by users respectively, κ_L and κ_U are the lower boundary and upper boundary for the curvature of the path respectively. The path smoothness term $J_{smoothness}$ is a line integral of the square of curvature of the path with respect to arc-length:

$$J_{smoothness} = \int_0^{s_f} \|\boldsymbol{\kappa}(\boldsymbol{p_r})\|^2 ds \tag{8}$$

where $\|\boldsymbol{k}(\boldsymbol{p_r})\|$ is the curvature scalar of the path. A curve connecting two configurations while minimizing $J_{smoothness}$ is the so-called Least Energy Curve, which has the least bending energy [64]. By solving a nonlinear programming problem like (7), our method generates a smooth path that distributes the steering efforts evenly along the path while satisfying the boundary constraints. The initial guess for $\boldsymbol{p}_r$ and s_{min} can be easily obtained by solving a Dubin path planning problem [65].

b: PATH SELECTION IN SINGLE PHASE

In the path selection stage, an efficient collision checking algorithm is employed to filter out paths in collision, which will be further described in the II-D section. Instead of considering multiple objectives to choose the best path among these collision-free paths, we use a single objective, the distance to the reference path, to select the best path. We argue that multiple objectives may conflict with each other, which can lead to hesitation and rapid variation in the selected path. Further, a single weighting coefficient matrix of the

[3] Special Euclidean group [61].

objectives may not be suitable for all scenarios encountered, and may require repeated tuning and careful selection of coefficients to achieve the desired behaviour throughout the set of operating conditions. It is non-trivial to find perfect coefficient matrix for every situation. Since the global path modification module has already refined the original global path and provided rational geometry information to guide the state sampling, the goal of the single phase state sampling step is to follow the modified reference path while satisfying nonholonomic constraints and keeping the selected paths collision-free. The multiple state space sampling method produces a curvature-continuous, collision-free, kinematic-feasible path, which satisfies all the requirements imposed on the path from the outset.

C. SPEED PLANNING ALONG THE FIXED PATH

Speed planning plays an important role in guaranteeing the ride comfort and safety of the vehicle. Dynamic constraints of the vehicle are also enforced through the speed profile selection, by considering the slip circle. As a path with continuous curvature is provided, this section focuses on finding a minimum time speed profile traveling along a fixed path subject to the vehicle dynamics constraints, slip circle constraints, and actuator limits. The determination of an optimal speed profile along a fixed path has been shown to be a convex optimization problem [66], [67].

A workspace path, r, of the body point, b, at the center of the rear axle with footprint, $\mathcal{A}$, is defined as $r : [0, s_f] \to \mathbb{R}^2$. In order to provide a general solution, we use the workspace path (9) to represent a fixed path, with the orientation and curvature encoded implicitly by the path. The relationship between the arclength s and the corresponding time t is formed as the function $s = f(t)$, therefore the time parameterized workspace path $\tilde{r}(t) = (\tilde{x}(t), \tilde{y}(t)), t \in [0, t_f]$ can be easily acquired by substituting in for s.

$$r(s) = (x(s), y(s)), s \in [0, s_f] \tag{9}$$

Since the path, $r(s)$, is known, the speed vector $\vec{\boldsymbol{v}}$ in Cartesian coordinates can be calculated as below,[4]

$$\vec{\boldsymbol{v}} = \dot{r}(s) = r'(s)\dot{f}, \tag{10}$$

where $r'(s)$ is the unit tangent vector of the path $r(s)$ at s that represents the direction of the speed of a car by assuming no sliding, $\dot{f}$ is the corresponding longitudinal speed of the car in ego frame. Let $\theta(s)$ represent the heading of the car at s of the path r, we get

$$r'(s) = \big(\cos(\theta(s)), \sin(\theta(s))\big) = \big(x'(s), y'(s)\big). \tag{11}$$

The acceleration vector $\vec{\boldsymbol{a}}$ in Cartesian coordinates system is

$$\vec{\boldsymbol{a}} = \ddot{r}(s) = r''(s)\dot{f}^2 + r'\ddot{f}, \tag{12}$$

where $\ddot{f}$ is the longitudinal acceleration and $r''(s)$ is the principle normal vector of the path, which is also called the curvature vector. The 2-norm of the $r''(s)$ is the scalar of the curvature.

$$\kappa = \|r''\| \tag{13}$$

The control force is defined as $\boldsymbol{u} = (u_{long}, u_{lat})$, where u_{lat} is the lateral force and u_{long} is the longitudinal force in ego frame. The dynamics of the car are given by

$$R\boldsymbol{u} = m\ddot{r}, \tag{14}$$

where

$$R = \begin{bmatrix} \cos(\theta(s)) & -\sin(\theta(s)) \\ \sin(\theta(s)) & \cos(\theta(s)) \end{bmatrix}$$

is the rotation matrix that maps forces from the ego frame to the global Cartesian coordinate system, m is the mass of the car. The control forces are limited by the friction circle as below

$$\|\boldsymbol{u}\| \le \mu\, mg, \tag{15}$$

where μ is the coefficient of friction between the tires and the road surface. The longitudinal force upper bound can be calculated according to the maximum longitudinal acceleration by $u_{long} \le m a_{max}^{long}$. There is no need to add a lateral acceleration constraint since it has been already limited by (15) and the longitudinal acceleration, although a user-specified lateral acceleration upper bound a_{max}^{lat} can be added to satisfy ride comfort as follows,

$$\|\boldsymbol{u}\| \le (a_{max}^{lat})^2 + (a_{max}^{long})^2 \le (\mu\, mg)^2. \tag{16}$$

We now replace the $\ddot{f}$ with a function $\alpha(s)$, $\dot{f}^2$ with a function $\beta(s)$ according to [66],

$$\alpha(s) = \ddot{f}, \quad \beta(s) = \dot{f}^2. \tag{17}$$

Then, $\dot{\beta}(s) = 2\ddot{f}\dot{f} = 2\alpha(s)\dot{f} = \beta'\dot{f}$. Thus,

$$\beta'(s) = 2\alpha(s), s \in [0, s_f]. \tag{18}$$

The objective function $T = \int_0^T 1dt$ is the total time travelling along the fixed path from 0 to s_f. Substitute the time variable t with arclength s and a convex optimization problem is posed

$$\begin{aligned} \underset{\alpha(s),\beta(s),\boldsymbol{u}(s)}{\textbf{minimize}}\ & T = \int_{f(0)}^{f(t_f)} \frac{1}{\dot{f}} ds = \int_0^{s_f} \beta(s)^{-\frac{1}{2}} ds \\ \textbf{s.t.}\ & (14), (18) \\ & \big(\alpha(s), \beta(s), \boldsymbol{u}(s)\big) \in \Big\{ \big(\ddot{r}(s), \dot{r}^2(s), \boldsymbol{u}(s)\big) \mid \\ & \qquad \|\boldsymbol{u}\| \le \mu\, mg, u_{long} \le m a_{max}^{long} \Big\} \end{aligned} \tag{19}$$

where $\dot{r}^2(s) = \big(r'(s)\big)^2\beta(s)$ and $\ddot{r}(s) = r'\alpha(s) + r''\beta(s)$. We use the MTSOS solver [67] to solve this problem and refer readers to [67] for more details.

[4]The prime $'$ and the dot $\cdot$ denote derivatives with respect to the arclength, s, and the time, t, respectively for a curve throughout the paper.

D. COLLISION CHECKING

Collision checking is one of the most time-consuming processes in sampling-based motion planning module. For collision checking, we employ a similar method to that used in [25] and [38], but with an alternate footprint approximation strategy and a different collision checking routine to improve accuracy and efficiency, respectively. By applying the distance transform explained in [68] to the occupancy grid map, a clearance map that represents the distance to the closest obstacle is generated, as shown in Fig. 12. The footprint of the car is approximated by a cluster of circles. Given the footprint of a car, we first calculate a circumcircle of the footprint rectangle called the bounding circle, $C_{bounding} \in \mathbb{R}^2$ and decompose the whole rectangle to four equal sized squares aligned with corner points and two rectangles with the same size shown in Fig. 13(b), then cover the small rectangles using circumcircles that are called footprint circles $C_{footprint} \in \mathbb{R}^2$. The collision checking routine is divided to two phases: the broad phase and the narrow phase. In the broad phase, if the distance from the center of the $C_{bounding}$ to the closest obstacle is greater than its radius, it is deemed collision-free and collision checking terminates. Otherwise, the collision checking proceeds to the narrow phase. The collision is detected if the distance from the center of any footprint circle $C_{footprint}$ to the closest obstacle is less than its radius.

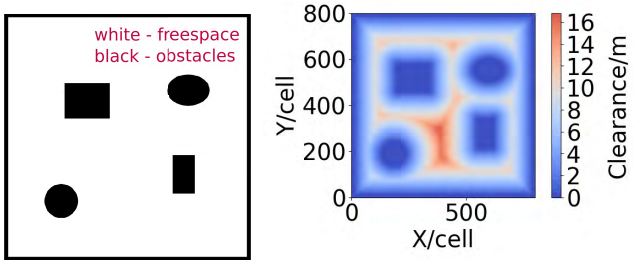


FIGURE 12. (a) The grid map. (b) The clearance map by applying distance transform algorithms to the grid map.

Since obstacles are represented in an occupancy grid map that can be easily generated from a practical perception system, our method works well on obstacles with general shapes in a realistic on-road autonomous driving system. Compared to the fast collision checking method used in [69], our method is able to avoid computationally-intensive calculations of $SE(2)$ configuration space obstacles with respect to all possible rotations along the path. In addition, the six-circle approximation method maintains lower approximation errors in the x direction and in overall area, as compared to the four-circle decomposition strategy used in [25], [38], and [70], while still keeping the number of collision checking circles small, as seen in Fig. 13(d). Although our method has a greater error than that of four-circle approximation method in the y direction, the absolute error remains relatively low (0.1 m), which is acceptable in practice.

Algorithm 5 CollisionFree(*state*)

Data: *state* The vehicle state that includes (x, y, θ) in map frame.
Result: *true* if the car state is collision-free, *false* otherwise
Parameter: $\{C_{footprint}\}$ The footprint circle set.
$C_{bounding}$ The bounding circle of the car footprint.
Operation: `Position` Acquire the position element of a circle (C) object.
`Radius` Acquire the radius element of a circle object.
`Transform` Acquire the position in map frame of a point represented in ego frame by performing the homogeneous transformation between map and ego frame.
`Clearance` Acquire the clearance value according to the position in the map.

```
p^E_bounding = Position(C_bounding); // Position in ego frame
p^M_bounding = Transform(state, p_bounding); // Position in map frame
d_bounding = Clearance(p^M_bounding);
if d_bounding > Radius(C_bounding) then // Broad phase checking
    return true;
else
    foreach C^i_footprint ∈ {C_footprint} do // Narrow phase checking
        p^E = Position(C^i_footprint);
        p^M = Transform(state, p^E);
        d = Clearance(p^M);
        if d ≤ Radius(C^i_footprint) then
            return false;
return true
```

In order to evaluate the performance of the proposed collision checking algorithm, we conducted benchmark tests by checking 1 million random $SE(2)$ states for collision within an 800×800 cell grid map with different numbers of random obstacles. Footprints of the first 1500 random states and 50 random obstacles are presented in an example benchmark test setup,[5] as shown in Fig. 14(a). As we use more circles to check collision, our improved footprint approximation with a single phase checking routine consumes more time to return results comparing to the four-circle method for all the cases. Thanks to the two-phase collision checking routine, our overall method maintains lower runtimes than the four-circle method used in [25], [38], and [70] in the presented test scenarios with obstacle density below 23% although

[5] As the performance of our method will not be affected by the shape of obstacles, we use disks whose radii are 3 m to generate random obstacles.

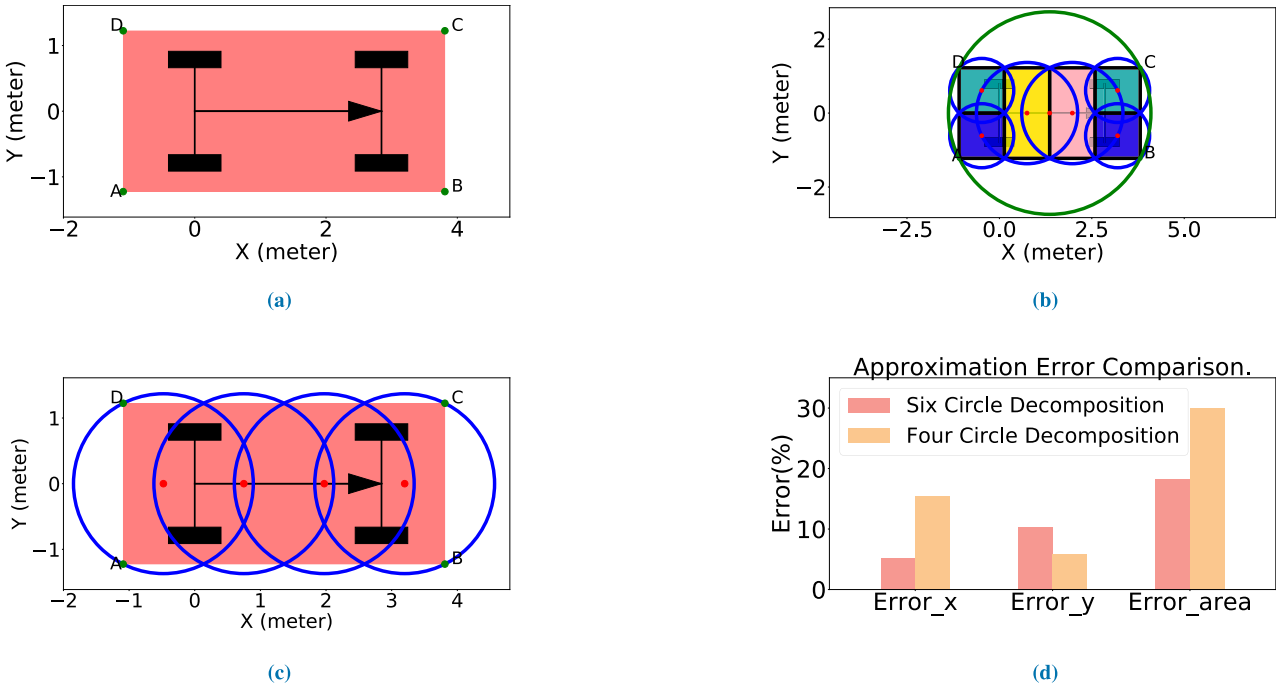


FIGURE 13. (a) The footprint of a car. (b) Six-circle decomposition. (c) Four-circle decomposition. (d) Dimension and area errors of both decomposition strategies in percentage.

we use more circles to approximate the footprint, as seen in Fig. 14(b). By leveraging the dimension information of the clearance, our method only need to check one bounding circle when the vehicle is far away from obstacles, which avoids unnecessary checking of several circles and saves up to 83.33% time for one collision checking. In practice, a large portion of waypoints on the path are far from obstacles, our method is able to gain great performance boosts, especially in scenarios with a low obstacle density. In worst case, we need to check 7 circles. So it's possible that our method may be slower than four-circle method on average when the obstacle density increases to a certain threshold. But the threshold is way more higher than that of most of realistic on-road driving scenarios.

III. SIMULATION RESULTS

In order to highlight the performance and features of the proposed algorithm, we evaluate its performance in multiple challenging simulated scenarios, in presence of multiple obstacles. The proposed planning algorithm is implemented in C++ running on a PC with an Intel Xeon E3 processor at 2.8GHz and 8GB RAM in a Linux system. We compare the proposed algorithm with the hybrid A* path planning algorithm from Autoware[6] [71] and the conjugate-gradient (CG) descent path smoother of [35]. We compare with the hybrid A* as it is a well tested planner and is used for both on-road driving and free-space driving scenarios in Autoware. The hybrid A* planner employs a customized collision checking method by checking grid cells of the footprint of the car for collision in the grid map. It is also an approximated collision checker due to the discrete grid representation. For the CG path smoother, we consider the obstacle avoidance term, the change of headings term, and the square of curvatures term mentioned in [35] as objectives to smooth the path of hybrid A* using the CG descent method. The corresponding coefficients of the aforementioned three terms are w_o, w_s, w_k, respectively. For all the following experiments, the CG path smoother uses 122 way points (roughly with 0.5 m for an interval) to perform the smoothing. We also implement the single phase state space sampling planners using two different path generation methods, such as the spiral path generation from [20] and [21] for red curves in Fig. 15 and the curvilinear path generation from [25] and [51] for orange curves in Fig. 15, for comparison. Unfortunately, both fail the easiest test scenario (Case A) that we present in this section, as shown in Fig. 15. There is no feasible path in their path candidates. Thus we only provide detailed results of the hybrid A* planner and the CG path smoother to compare with ours in the following test scenarios.

The vehicle model parameters are based on the dimensions of the Lincoln MKZ, shown in Fig. 16. The kinematics and dynamics of the vehicle are simulated in the high-fidelity V-REP simulator [72]. The global path and the road boundaries for traversable region extraction are predefined by Lanelet maps [73]. The proposed planner communicates with other support modules through ROS[7]. The vehicle model

[6]https://github.com/CPFL/Autoware

[7]http://www.ros.org/

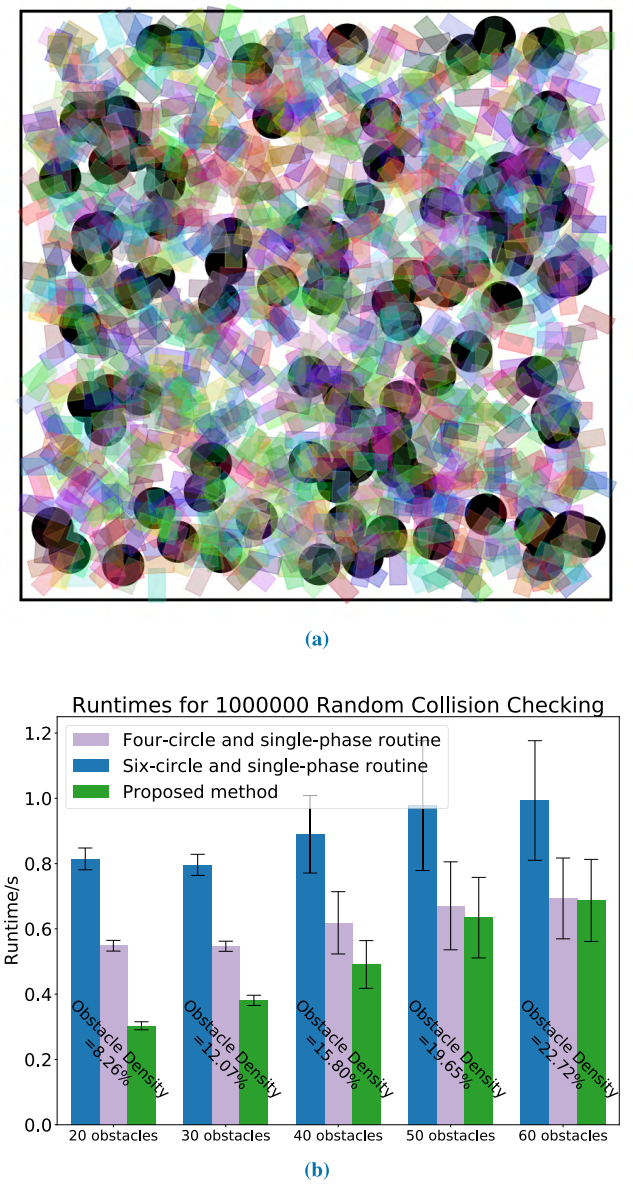


FIGURE 14. (a) An example of benchmark tests. The corresponding obstacle density is close to 20%. Colored rectangles depict car footprints and black disks represent obstacles. (b) Runtimes of the proposed two phase collision checking routine and single phase collision checking routine with the same 6 circle approximation for tests in (a).

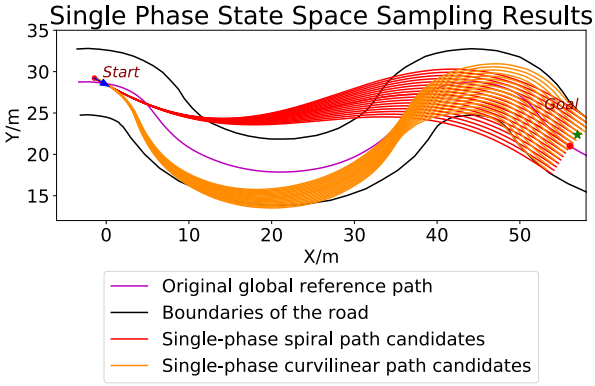


FIGURE 15. Single phase state space sampling failures with a long planning horizon along a curvy road.

FIGURE 16. Lincoln MKZ vehicle platform.

TABLE 1. Parameter setting.

Parameter	Description	Setting	Unit
w	Car width	2.45	m
l	Car length	4.9	m
wb	Car wheelbase	2.8448	m
tr	Car track	1.5748	m
r_{min}	Minimum turning radius of the car	5.0	m
R_{min}	Minimum circle radius for space explore	1.6	m
R_{max}	Maximum circle radius for space explore	10.0	m
res	Resolution of the grid map	0.1	m
m	The mass of the car.	1500.0	kg
μ	The friction coefficient	0.7	1
a_{max}^{long}	The max longitudinal acceleration of the car.	5.0	m/s^2
j	Children node number in each expand in Fig. 5	36	1

and other parameters used by the proposed planner are listed in Table 1. The curvature bounds are $\kappa_L = -0.2$ m^{-1} and $\kappa_U = 0.2$ m^{-1} according to r_{min}.

A. CURVY ROAD DRIVING WITHOUT OBSTACLES.

We set up a challenging test scenario on a curvy, single lane road with a width of 8 m. Fig. 18, Fig. 19, and Fig. 20 illustrate the scenario along with resulting paths, heading profiles, and curvature profiles respectively for the proposed planner, and the hybrid A* planner and its CG path smoother. It should be noted that the CG path smoother needs an initial path that is collision-free to get a result, or it may not be able to provide a solution. The resulting path of the hybrid A* planner is used as the input of the CG path smoother. Since the obstacle avoidance term in the CG path smoother is a soft constraint, the smoothed path is not guaranteed to be collision-free. Thus we conduct three experiments with different parameter settings (see Fig. 17) to find the best feasible path to compare with and show the limitations of their method as well. The S-CG path is not collision-free since the coefficient w_o is zero. The L-CG path is sub-optimal due to a relative large coefficient $w_o = 0.5$ for the obstacle avoidance term. Therefore, we only compare the results of the M-CG path with that of hybrid A* planner and ours. The length of the

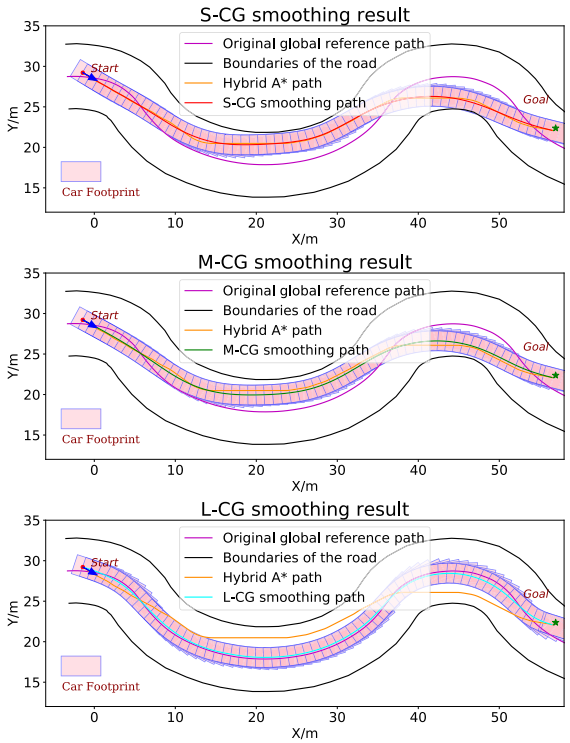


FIGURE 17. CG paths with different coefficient settings. S-CG: $w_s = 100, w_k = 1, w_o = 0$. M-CG: $w_s = 100, w_k = 1, w_o = 0.08$. L-CG: $w_s = 100, w_k = 1, w_o = 0.5$.

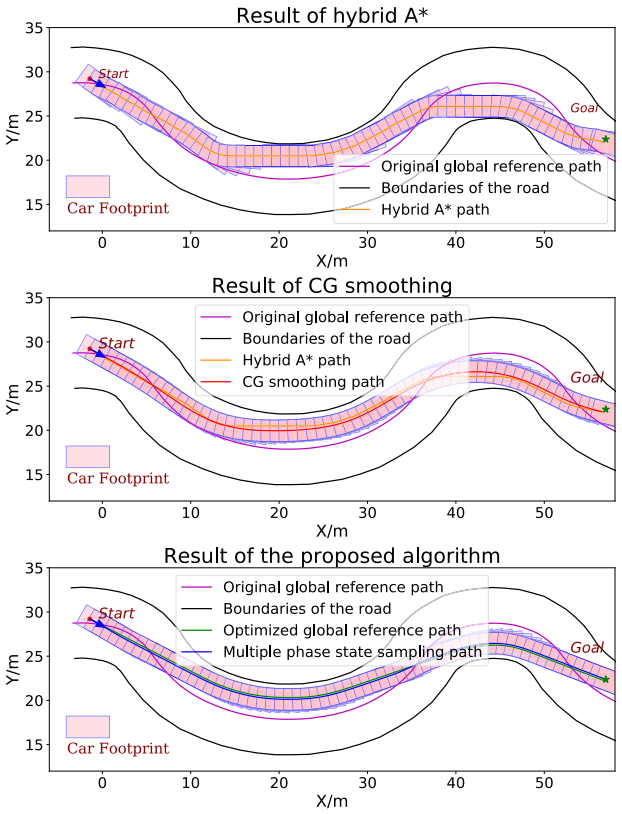


FIGURE 18. Path results for the scenario without obstacles.

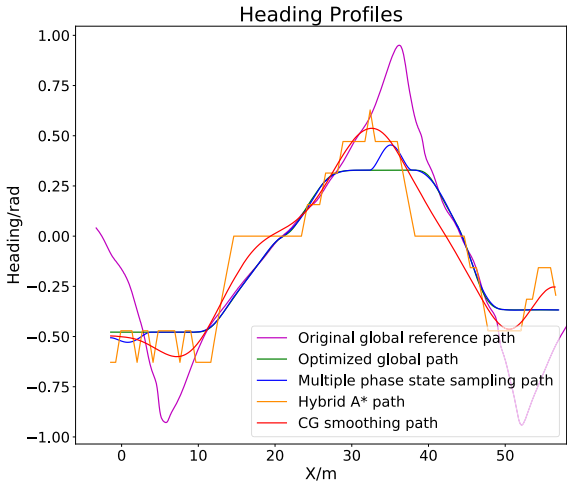


FIGURE 19. Heading profile results for the scenario without obstacles.

original global path (GP) from start to goal is 67.07 m. All the listed planning results in Fig. 18 show some path shortening capacity of the tested planners when compared to the original global path. The length reductions of the optimized global path (OGP) and multiple state sampling path (MSSP) are 7.75% and 7.28%, respectively. The hybrid A* planner and the CG path smoother show similar results in terms of path length, providing 6.99% and 7.1% reductions, respectively. The proposed planner demonstrates a notable path smoothing feature according to the curvature profiles in Fig. 20, which is not observed in the hybrid A* planner. In constrast, the hybrid A* planner generates a serrated curvature profile due to its use of discretized motion primitives to construct the path. The CG path smoother shows a similar result with ours visually in terms of smoothness based on the hybrid A* path. We also provide quantitative measurements of smoothness of the resulting paths using (1), shown in the **Smoothness** column in Table 3 (the lower, the better). The smoothness of the proposed planner's path is only 13.94% of that of the hybrid A* planner and 82.11% of that of the CG path smoother. It should be noted that the smoothness of CG paths can be improved by tuning the coefficients, and may ultimately perform better than our method. However, it requires a major tuning effort to identify such coefficients and the best coefficients may be environment-dependent. It is hard or even impossible to find a single suitable coefficient set for the scenarios presented, or for autonomous driving in general. In addition, the collision-free property and kinematic-feasibility are not guaranteed according to the problem formulation of the CG path smoother [35]. This scenario demonstrates both the path shortening and smoothing capacity of the proposed method within the given traversable region.

B. CURVY ROAD DRIVING WITH DENSE OBSTACLES.

In the second scenario, we placed several consecutive static obstacles close together on the curvy road to investigate the

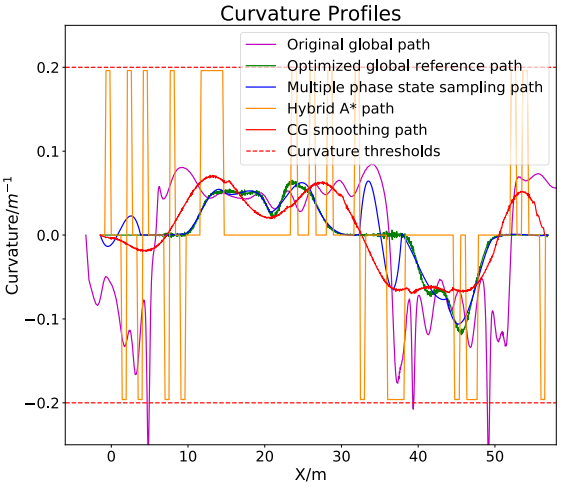


FIGURE 20. Curvature profile results for the scenario without obstacles.

TABLE 2. Path shortening and smoothing.

Path	Length (m)	Percentage of GP Length	Max Curvature (m^{-1})
GP	67.07	100%	0.2885
HAP	62.38	93.01%	0.1961
M-CG	62.31	92.90%	0.0713
OGP	61.87	92.25%	0.1188
MSSP	62.19	92.72%	0.1061

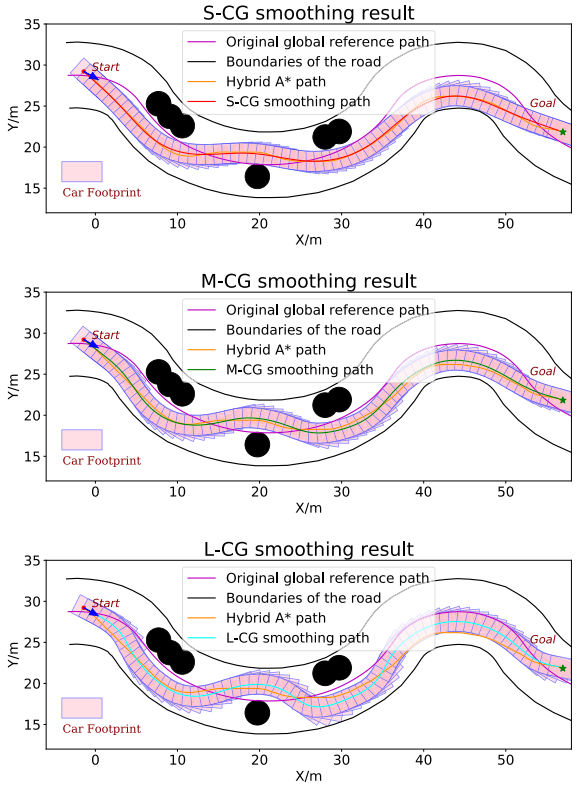


FIGURE 21. CG paths with different coefficient settings. S-CG: $w_s = 100, w_k = 1, w_o = 0$. M-CG: $w_s = 100, w_k = 1, w_o = 0.2$. L-CG: $w_s = 100, w_k = 1, w_o = 1$.

obstacle avoidance ability and resulting path quality of the proposed planner, as shown in Fig. 22, Fig. 23, and Fig. 24. Similar to the previous tests, we conduct three experiments for the CG path smoother using different parameter settings to get a reasonable well result to compare with, as shown in Fig. 21. The results of S-CG and L-CG both collide with the obstacle or boundary of the road due to the reasons that have been stated in last subsection. Thus only detailed results of the M-CG path smoother are presented in the following comparison. As the hybrid A* planner cannot exactly reach the goal state with discretized motion primitives, we set a 0.3 m tolerance on the position dimension to prevent unbounded search or failures for the hybrid A* planner. All the listed planners are able to generate collision-free, kinematically-feasible paths to reach the goal, as depicted in Fig. 22. According to results in Table 3, the path length of the hybrid A* planner is slightly shorter (0.18 m) than that of the proposed planner in this scenario. However, the path endpoint of the hybrid A* did not exactly reach the goal according to the x axis data in Fig. 22, Fig. 23, and Fig. 24 due to the position tolerance. Thus, the deviation in length can be ignored. The CG path smoother shows a slightly longer path than ours. Again, the proposed planner outperforms the hybrid A* planner in smoothness, time and memory usage aspects and outperforms the CG path smoother in smoothness, path length and memory usage aspects. The smoothness scalar of the proposed planner is 0.4377, roughly half of that

TABLE 3. Performance.

Scenario	Planner	Length (m)	Smoothness	Max Curvature m^{-1}	Time (ms)	Memory (MB)
Fig. 18	Hybrid A*	62.38	0.7476	0.1961	291.54 ± 7.11	~3072
	M-CG	62.31	0.1269	0.0713	57.30±5.77	60-90
	Proposed	62.19	0.1042	0.1061	51.93 ± 5.7	~20
	Speed Planner	-	-	-	14.55 ± 1.4	-
Fig. 22	Hybrid A*	65.59	0.8786	0.1961	239.40 ± 6.69	~3072
	M-CG	65.88	0.4657	0.1514	26.15±1.77	60-90
	Proposed	65.77	0.4377	0.1894	53.3 ± 4.7	~20
	Speed Planner	-	-	-	14.15 ± 1.6	-

of the hybrid A* planner and 93.99% of that of the CG path smoother.

To compare the runtime, we ran all the planners 1000 times for the same scenario with the same start and goal, then calculated the mean and the standard deviation of the runtime for a single planning loop in both cases. To keep the comparison fair, we only compare the running time of path generation part of the both planners since there is no speed planning module available for the hybrid A* planner and the CG path smoother. The running time of the speed planning layer for our planner is also listed separately for reference. The related results are presented in Table 3. The proposed planner only consumed 17.81% and 22.26% of time of the hybrid A* planner in Fig. 18 and Fig. 22 cases with the presented planning horizon, respectively. It should be noted the runtimes of the CG path smoother listed in Table 3 only show the time consumed in the smoothing process. The final runtime for the

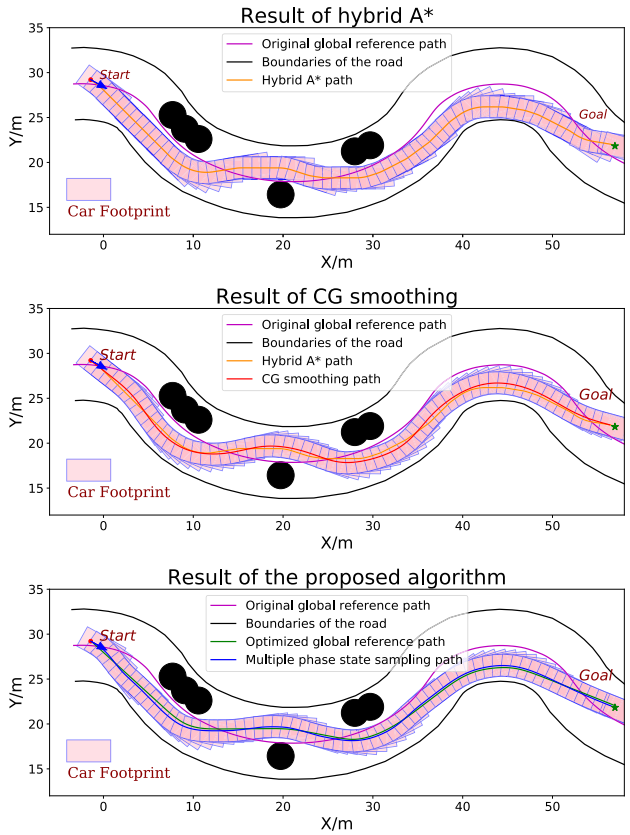


FIGURE 22. Path results for the scenario with multiple obstacles.

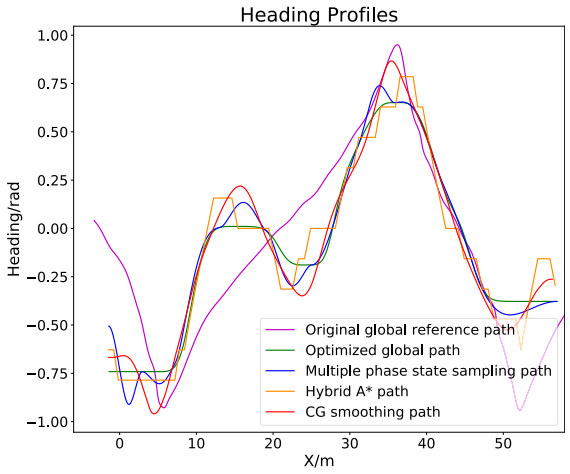


FIGURE 23. Heading profile results for the scenario with multiple obstacles.

result of the CG path smoother should be the summation of runtimes of the hybrid A* path generation part and the CG path smoothing part. Thus, the proposed method also outperforms the overall CG path smoother in run-time performance. Since the first running of the hybrid A* planner took over 2000 ms for setting up, we removed it from the results for fairness. As the hybrid A* planner needs to build a huge 3D search space to explore, it took up to 3072MB memory while

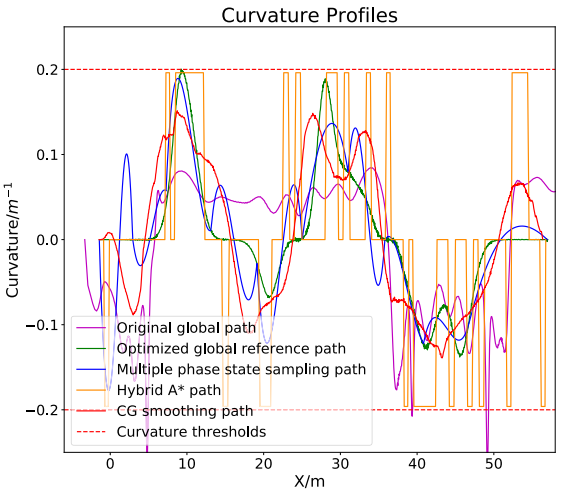


FIGURE 24. Curvature profile results for the scenario with multiple obstacles.

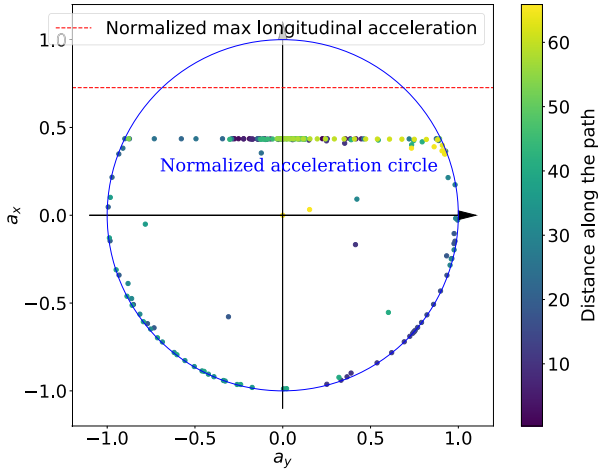


FIGURE 25. The g-g diagram of the resulting trajectory.

the proposed planner requires only 20MB memory. The CG path smoother itself takes memory beween 60MB and 90MB roughly for one planning cycle. For both planning scenarios, the optimized global path generated by the global path modification (GPM) layer is shorter and smoother than the final path. This is due to the fact that the GPM layer searches a 2D workspace and optimizes the path without considering the nonholonomic constraints and current heading of the car. The GPM provides the shortest path through the environment subject to the given constraints, but non-collision and kinematic feasibility of the path are not guaranteed. The small deviation between the optimized global path and final path, as well as the curvature profiles in Fig. 18 and Fig. 22 justify the three layer decomposition presented in this work, as each layer refines the previous result without drastically altering it.

Interested readers are encouraged to view a video of the proposed planner for this scenario, available at https://vimeo.com/257666203. An MPC controller is integrated into the system to track the resulting final path

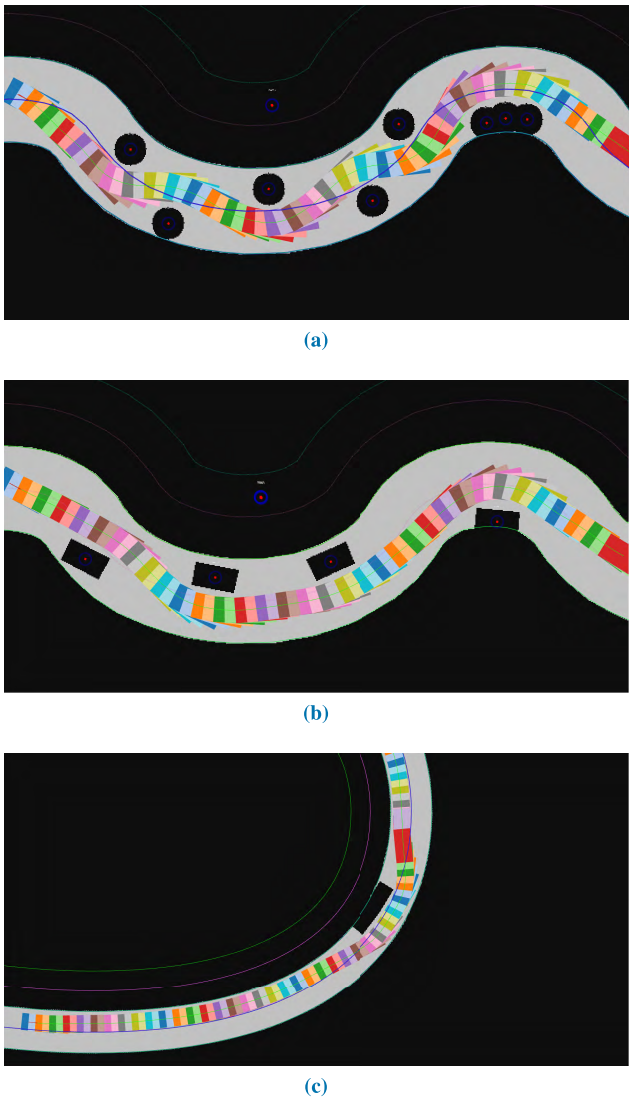


FIGURE 26. Other challenging on-road driving scenarios.

in the video. Note that we did not deliberately tune the weight matrix of the MPC controller for this case. The car is still able avoid the consecutive obstacles. In addition, the grid map data is noisy between frames in simulation, and the proposed planner handled it well through rapid re-planning. We also include several planning results of different on-road driving scenarios without quantity analyses, shown in Fig. 26.

In terms of dynamic constraints of cars, the normalized longitudinal and lateral accelerations results of the speed planning portion are plotted with a friction circle ("g-g" diagram [74], [75]) in Fig. 25 for the obstacle free scenario of Fig. 18. As depicted in Fig. 25, the accelerations at every point of the trajectory were limited within the friction circle and all the longitudinal accelerations are below the max longitudinal acceleration threshold. As the objective of speed planning is minimum time of travel along the path, most acceleration points tend to stay close to the acceleration limits imposed on the solution.

IV. CONCLUSION

In this paper, we have presented a novel and efficient hybrid trajectory planning framework to handle geometry constraints, nonholonomic constraints and dynamics constraints nicely in a human-like and layered fashion. The proposed method employs a derivative-free global path modification algorithm to refine the reference path and extract high-order state information in free space for state sampling, which provides correct guidance information for sampling and increases sampling efficiency as well. By extending a single phase state space sampling to a multi-phase state space sampling, our method is able to approximate complex motions without bringing in too much computation burden. The proposed method also exploits a more accurate and efficient collision checking algorithm to filter out sampling paths in collision. The results show our method is able to generate curvature-continuous, kinodynamic-feasible, smooth and collision-free trajectories in real-time while using fewer computational resources and outperforming the hybrid A* planner, the CG path smoother and the single phase state space sampling method in multiple challenging planning scenarios.

Although we have addressed the trajectory planning problem with several essential constraints for autonomous driving in highly constrained environments, our method does *not* consider dynamic obstacles explicitly. By fast re-planning, our method is able to avoid some of the dynamic obstacles in a reactive way. But it's not ideal and safety-guaranteed. We will consider dynamic obstacles in both spatial and temporal domain in our future research and investigate how to manipulate lateral motions and longitudinal motions of cars to avoid dynamic obstacles smartly.

ACKNOWLEDGMENT

The authors would also like to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transp. Res. C, Emerg. Technol.*, vol. 60, pp. 416–442, Nov. 2015.

[2] R.-H. Zhang, Z.-C. He, H.-W. Wang, F. You, and K.-N. Li, "Study on self-tuning tyre friction control for developing main-servo loop integrated chassis control system," *IEEE Access*, vol. 5, pp. 6649–6660, 2017.

[3] Z. He, L. Zheng, L. Lu, and W. Guan, "Erasing lane changes from roads: A design of future road intersections," *IEEE Trans. Intell. Veh.*, vol. 3, no. 2, pp. 173–184, Jun. 2018.

[4] J. H. Reif, "Complexity of the mover's problem and generalizations," in *Proc. IEEE Annu. Symp. Found. Comput. Sci. (FOCS)*, Oct. 1979, pp. 421–427.

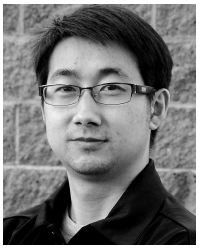
[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[6] J. Barraquand and J.-C. Latombe, "On nonholonomic mobile robots and optimal maneuvering," in *Proc. IEEE Int. Symp. Intell. Control (ISIC)*, Sep. 1989, pp. 340–347.

[7] B. Li and Z. Shao, "Simultaneous dynamic optimization: A trajectory planning method for nonholonomic car-like robots," *Adv. Eng. Softw.*, vol. 87, pp. 30–42, Sep. 2015.

[8] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
[9] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robot. Autom.*, vol. 14, no. 1, pp. 166–171, Feb. 1998.
[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[11] S. M. LaValle and J. J. Kuffner, Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[12] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.
[13] J.-H. Ryu, D. Ogay, S. Bulavintsev, H. Kim, and J.-S. Park, "Development and experiences of an autonomous vehicle for high-speed navigation and obstacle avoidance," in *Frontiers of Intelligent Autonomous Systems*. Berlin, Germany: Springer, 2013, pp. 105–116.
[14] J. H. Jeon *et al.*, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *Proc. Amer. Control Conf. (ACC)*, 2013, pp. 188–193.
[15] Y. Chen, H. Peng, and J. W. Grizzle, "Fast trajectory planning and robust trajectory tracking for pedestrian avoidance," *IEEE Access*, vol. 5, pp. 9304–9317, 2017.
[16] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 1478–1483.
[17] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE Access*, vol. 2, pp. 56–77, 2014.
[18] F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche, "Driving with tentacles: Integral structures for sensing and motion," *J. Field Robot.*, vol. 25, no. 9, pp. 640–673, 2008.
[19] A. Broggi, P. Medici, P. Zani, A. Coati, and M. Panciroli, "Autonomous vehicles control in the VisLab intercontinental autonomous challenge," *Annu. Rev. Control*, vol. 36, no. 1, pp. 161–171, 2012.
[20] T. Howard, C. Green, A. Kelly, and D. Ferguson, "State space sampling of feasible motions for high-performance mobile robot navigation in complex environments," *J. Field Robot.*, vol. 25, nos. 6–7, pp. 325–345, 2008.
[21] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
[22] M. Wang, T. Ganjineh, and R. Rojas, "Action annotated trajectory generation for autonomous maneuvers on structured road networks," in *Proc. IEEE Int. Conf. Autom., Robot. Appl. (ICARA)*, Dec. 2011, pp. 67–72.
[23] P. Resende and F. Nashashibi, "Real-time dynamic trajectory planning for highly automated driving in highways," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2010, pp. 653–658.
[24] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. J. Robot. Res.*, vol. 31, no. 3, pp. 346–359, 2012.
[25] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 740–753, Apr. 2016.
[26] L. Ma, J. Yang, and M. Zhang, "A two-level path planning method for on-road autonomous driving," in *Proc. IEEE Int. Conf. Intell. Syst. Design Eng. Appl. (ISDEA)*, Jan. 2012, pp. 661–664.
[27] T. Gu and J. Dolan, "On-road motion planning for autonomous vehicles," in *Intelligent Robotics and Applications*. Berlin, Germany: Springer, 2012, pp. 588–597.
[28] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *Int. J. Robot. Res.*, vol. 28, no. 8, pp. 933–945, 2009.
[29] S. Kammel *et al.*, "Team AnnieWAY's autonomous system for the 2007 DARPA urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 615–639, 2008.
[30] A. Bacha *et al.*, "Odin: Team VictorTango's entry in the DARPA urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 467–492, 2008.
[31] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. IEEE/RSJ Int. Workshop Intell. Robots Syst. (IROS)*, Oct. 2009, pp. 1879–1884.
[32] M. N. Pivtoraiko, "Differentially constrained motion planning with state lattice motion primitives," Robotics Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. CMU-RI-TR-12-07, 2012.
[33] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 4889–4895.
[34] J. Bohren *et al.*, "Little ben: The ben franklin racing team's entry in the 2007 DARPA urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 598–614, 2008.
[35] D. Dmitri, *et al*, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.
[36] D. Kogan and R. Murray, "Optimization-based navigation for the DARPA grand challenge," in *Proc. 45th IEEE Conf. Decis. Control (CDC)*, San Diego, CA, USA, 2006.
[37] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Int. J. Robot. Res.*, vol. 22, nos. 7–8, pp. 583–601, Jul. 2003.
[38] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—A local, continuous method," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2014, pp. 450–457.
[39] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 205–210.
[40] J. Pan, L. Zhang, D. Manocha, and U. Hill, "Collision-free and curvature-continuous path smoothing in cluttered environments," in *Robotics: Science and Systems VII*, vol. 17. Cambridge, MA, USA: MIT Press, 2012, p. 233.
[41] M. Elbanhawi, M. Simic, and R. Jazar, "Randomized bidirectional B-spline parameterization motion planning," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 2, pp. 406–419, Feb. 2016.
[42] M. Matthew, "Parallel algorithms for real-time motion planning," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2011.
[43] J.-W. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation," in *Proc. IEEE Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2012, pp. 282–287.
[44] A. Piazzi and C. G. L. Bianco, "Quintic G^2-splines for trajectory planning of autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, Oct. 2000, pp. 198–203.
[45] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 613–626, Feb. 2018.
[46] C. Chen, "Motion planning for nonholonomic vehicles with space exploration guided heuristic search," Ph.D. dissertation, Fakultät für Inf., Tech. Univ. Munich, Munich, Germany, 2016.
[47] T. Gu, J. Snider, J. M. Dolan, and J.-W. Lee, "Focused trajectory planning for autonomous on-road driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2013, pp. 547–552.
[48] T. Gu, J. M. Dolan, and J.-W. Lee, "Runtime-bounded tunable motion planning for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, 2016, pp. 1301–1306.
[49] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.
[50] E. W. Weisstein. (Mar. 2, 2018). *SSS Theorem*. [Online]. Available: http://mathworld.wolfram.com/SSSTheorem.html
[51] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mech. Syst. Signal Process.*, vol. 100, pp. 482–500, Feb. 2018.
[52] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 561–568, Jun. 2010.
[53] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita, "Bézier curve based path planning for autonomous vehicle in urban environment," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2010, pp. 1036–1042.
[54] J. Funke and J. C. Gerdes, "Simple clothoid paths for autonomous vehicle lane changes at the limits of handling," in *Proc. ASME Dyn. Syst. Control Conf.*, 2013, p. V003T47A003.
[55] D. Knowles, "Real time continuous curvature path planner for an autonomous vehicle in an urban environment," California Inst. Technol., Pasadena, CA, USA, Tech. Rep., 2006, vol. 1.
[56] S. Gim, L. Adouane, S. Lee, and J.-P. Dérutin, "Clothoids composition method for smooth path generation of car-like vehicle navigation," *J. Intell. Robotic Syst.*, vol. 88, no. 1, pp. 129–146, 2017.

[57] H. Marzbani, R. N. Jazar, and M. Fard, "Better road design using Clothoids," in *Sustainable Automotive Technologies*. Springer, 2015, pp. 25–40.
[58] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing* (Mathematics Visualization). New York, NY, USA: Springer, 2002.
[59] R. T. Farouki and T. Sakkalis, "Real rational curves are not 'unit speed,'" *Comput. Aided Geometric Des.*, vol. 8, no. 2, pp. 151–157, 1991.
[60] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2011.
[61] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006.
[62] J. H. Mathews, *Numerical Methods for Mathematics, Science and Engineering*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1992.
[63] L. Piegl and W. Tiller, *The NURBS Book*. New York, NY, USA: Springer-Verlag, 1997.
[64] B. K. P. Horn, "The curve of least energy," *ACM Trans. Math. Softw.*, vol. 9, no. 4, pp. 441–460, Dec. 1983.
[65] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, 1957.
[66] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Trans. Autom. Control*, vol. 54, no. 10, pp. 2318–2327, Oct. 2009.
[67] T. Lipp and S. Boyd, "Minimum-time speed optimisation over a fixed path," *Int. J. Control*, vol. 87, no. 6, pp. 1297–1311, 2014.
[68] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2012.
[69] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2010, pp. 518–522.
[70] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep./Oct. 2015, pp. 250–256.
[71] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, "An open approach to autonomous vehicles," *IEEE Micro*, vol. 35, no. 6, pp. 60–68, Nov./Dec. 2015.
[72] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: A versatile 3D robot simulator," in *Proc. Simulation, Modeling, Program. Auton. Robots*, 2010, pp. 51–62.
[73] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2014, pp. 420–425.
[74] W. F. Milliken and D. L. Milliken, *Race Car Vehicle Dynamics*. Warrendale, PA, USA: SAE, 1995.
[75] R. S. Rice, "Measuring car-driver interaction with the G-G diagram," SAE Tech. Paper 730018, 1973.



YU ZHANG received the B.S. degree in automotive engineering from China Agriculture University, Beijing, China, in 2012. He is currently pursuing the Ph.D. degree in mechanical engineering with the School of Mechanical Engineering, Beijing Institute of Technology, Beijing.

His research interests include simulation, real time, and optimal motion planning and control for autonomous vehicles.

HUIYAN CHEN received the Ph.D. degree from the Beijing Institute of Technology, Beijing, China, in 2004. He has been holding a position with the Beijing Institute of Technology since 1981, where he is currently a Professor at the Intelligent Vehicle Research Center.

His research interest covers intelligent vehicle and information technologies.

STEVEN L. WASLANDER received the B.Sc.E. degree in applied mathematics and mechanical engineering from Queen's University, Kingston, ON, Canada, in 1998, and the M.S. and Ph.D. degrees in aeronautics and astronautics degree from Stanford University, Stanford, CA, USA, in 2002 and 2007, respectively. He is currently an Associate Professor with the Institute for Aerospace Studies, University of Toronto, Toronto, ON, Canada. He is also the Director of the Toronto Robotics and Artificial Intelligence Laboratory and the Waterloo Autonomous Vehicles Laboratory.

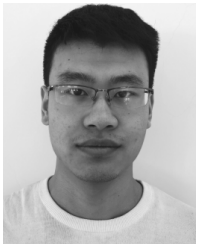
His main research interests include perception, navigation, and control of self-driving cars and autonomous aerial rotorcraft with a focus on simultaneous localization and mapping, object detection, optimal motion planning, and multirobot coordination.

JIANWEI GONG received the Ph.D. degree in mechatronical engineering from the School of Mechanical Engineering, Beijing Institute of Technology, Beijing, China, in 2002. He is currently a Professor with the Intelligent Vehicle Research Center, School of Mechanical Engineering, Beijing Institute of Technology.

His research interest covers planning, control, and humanlike driving for autonomous vehicles.

GUANGMING XIONG received the Ph.D. degree in mechatronical engineering from the Beijing Institute of Technology, Beijing, China, in 2005. He is currently an Associate Professor with the School of Mechanical Engineering, Beijing Institute of Technology.

His main research interests include intelligent vehicles, mobile robotics, machine vision, and multi-vehicle coordination.



TIAN YANG received the B.S. degree in automation from the Beijing Institute of Technology, Beijing, China, in 2017, where he is currently pursuing the M.S. degree in mechanical engineering with the School of Mechanical Engineering.

His research interests include optimal control and real-time motion planning for autonomous driving.

KAI LIU received the B.S. and M.S. degrees in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2008 and 2010, respectively, where he is currently pursuing the Ph.D. degree in mechanical engineering with the School of Mechanical Engineering. He was a Visiting Scholar with The Ohio State University, Columbus, OH, USA.

His research interests include autonomous driving, vehicle dynamics modeling, intelligent vehicle systems, and model predictive control.