Received August 4, 2020, accepted August 18, 2020, date of publication August 24, 2020, date of current version September 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3018731

Knowledge-Biased Sampling-Based Path Planning for Automated Vehicles Parking

YIQUN DONG[1], YUANXIN ZHONG[2], AND JIAJUN HONG[2]
[1]Department of Aeronautics and Astronautics, Fudan University, Shanghai 200433, China
[2]Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48105, USA

Corresponding author: Yiqun Dong (yiqundong@fudan.edu.cn)

ABSTRACT We consider automated vehicles operation in constrained environments, i.e. the automated parking (AP). The core of AP is formulated as a path planning problem, and Rapidly-exploring Randomized Tree (RRT) algorithm is adopted. To improve the baseline RRT, we propose several algorithmic tweaks, i.e. reversed RRT tree growth, direct tree branch connection using Reeds-Shepp curves, and RRT seeds biasing via regulated parking space/vehicle knowledge. We prove that under these tweaks the algorithm is complete and feasible. We then examine its performance (time, success rate, convergence to the optimal path) and scalability (to different parking spaces/vehicles) via batched simulations. We also test it using a real vehicle in a realistic parking environment. The proposed solution presents itself more applicable when compared with other baseline algorithms.

INDEX TERMS Automated vehicles, automated parking, sampling-based path planning, knowledge-based biasing.

I. INTRODUCTION

A. MOTIVATION

Recent advances in automated vehicles (AVs) have shown radical impacts on our societal lives. Among the many scenarios that the AVs have been deployed/tested in, most of them focus on open roads interacting with other road users. However, the operation in constrained environments, e.g. automated parking (AP), is also important. In this scenario, the spaces are limited, vehicle movements are constrained to kinematic (minimum turning radius) and geometric (collision checking) constraints. Vehicle start pose and parking angle/sizes also varies. How to develop a robust AP solution still remains an open question.

Essentially, the operation of AP hings on three folds: to perceive the parking environment, to find the parking path, and to execute the path. For the low-speed vehicle parking control, the path execution has been well addressed [1], [2]. For perception, we propose to rely on AV's onboard sensors [3], and formulate the parking path finding as a planning problem. The path planning therefore is the core. We hope to develop a *complete* and *feasible* algorithm that yields *good performance* (high success rate, short planning time, fair convergence to the optimal path). We also hope this algorithm has an *unified* (wide scalability) form; i.e., it can be applied to *all* vehicles (with different sizes) that start from *any* pose in *all* parking spaces (with different parking angles/sizes).

B. RELATED WORK

Many of the path planning work for AP adopts geometric-based methods; curve fitting are widely used to concatenate a path that guides the vehicle from the start to the parking pose [4]–[9]. In these methods, however, the vehicle must start from a specific point/region, which does not satisfy our formulation for a unified parking solution.

Robotic planning algorithms have also been widely used. In [10], given global map of the parking space, A* was used to find a geometrical path. Trajectory smoothing is then performed to satisfy the kinematic constraints. Other methods that rely on pre-known knowledge of the parking space includes the artificial potential field [11] and state roadmap [12]. These solutions, however, usually hinge on careful tuning of the algorithmic parameters (cost in A*, the potential field, etc.), which is sensitive to the parking spaces. In real cases the parking spaces may vary significantly. Scalability of these algorithms is unknown.

We adopt Rapidly-exploring Randomized Tree (RRT) [13] as our solution. In this algorithm, randomized seeds are generated. The RRT tree is biased to these seeds, performing an exploration of the environment. As proved in [2], [13], RRT satisfies the vehicle's constraints. Given sufficient planning time, the algorithm is also guaranteed to find the path that starts from any poses (so long as it exists). In literature

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaosong Hu.

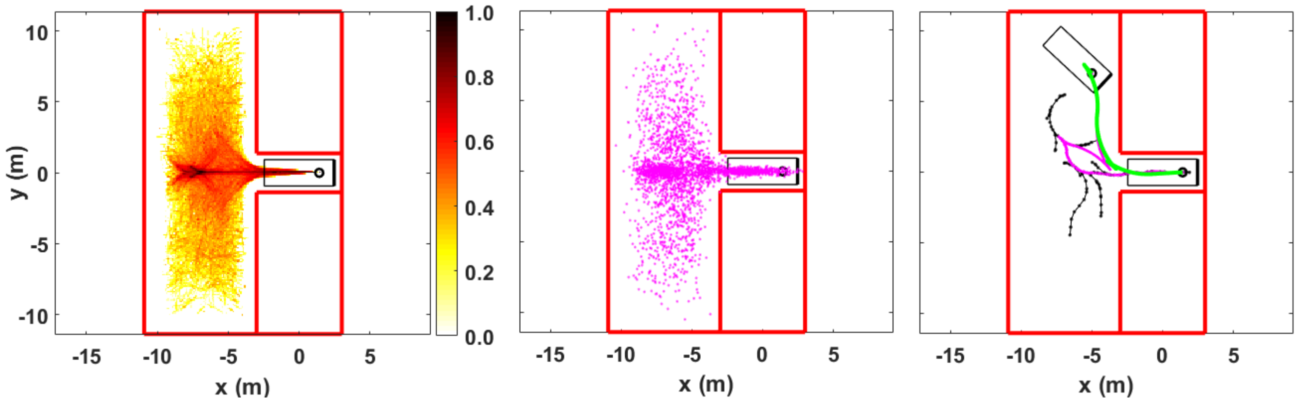

FIGURE 1. Illustration of the proposed algorithm for 90^o parking. Left: heatmap of the clustered optimal parking paths. Middle: RRT seeds generation are biased to the hot area of the heatmap. Right: RRT tree's growth (black) is guided by the generated seeds, exploring more into the hot areas where the optimal paths are more possibly to happen. The tree is also reversely grown from the parking spot, and last branch connected via Reeds-Sheep curves. After a feasible path (magenta) is found, post optimization is performed to obtain the optimized path (green).

RRT-based AP solutions have appeared. In [14] the authors discussed perpendicular, parallel, and inclined parking. But as admitted in the paper, the adopted algorithm was slow, and the planned path not smooth enough. In [15] orientation of the RRT seed was specified. But the orientation is dependent on the parking environment; the authors discussed perpendicular parking only. It is unknown how to generalize the algorithm to other parking spaces. In [16] the authors proposed two improvements over the baseline RRT. But no justification as to feasibility and efficiency of the algorithm was disclosed. Lastly in [17] the authors implemented RRT via reversely growing a tree from the parking spot. Branches on the tree, however, were pre-assigned. Completeness of the algorithm was unclear.

In literature there exist efforts dedicated to improving the baseline RRT, which may render the algorithm more suitable to the constrained AP problem. To begin with, while the randomized RRT seeds generation guarantees its completeness, the bad positioning of the seeds may lure the tree growth into narrow passages, significantly slowing down the planning. To overcome this issue, in [18] the authors proposed RRT-VF; only the tree nodes with higher viability are extended. In [19] MARRT was formulated. The RRT tree grows along the medial axis of the working space. In [20] Skilled-RRT was proposed. Local seeds were generated uniformly within a neighborhood along the skeleton of the work space. In [21] the seeds generation was biased to the Voronoi Graph of the work space. Lastly, in [22] the authors followed the new thrust force of Deep Learning and adopted expert knowledge-trained conditional variational autoencoder (CVAE) in biasing the seeds. Similar to these researches, we summarize the optimal parking paths that originate from random start poses in different parking spaces. We then cluster these paths and use it as pre-knowledge in generating the seeds.

Other researches focused on the extend function of RRT, which is used to navigate the vehicle from a certain tree node to the generated seed. In [23] a fixed-final-state-free-final-time controller that optimally connects the tree node and seed was proposed. In [24] a modified controller that manipulates both lateral heading and longitudinal speed was adopted. Other works include [25] wherein clothiod curves are used to fit the vehicle's trajectory. In our solution we follow [26], and use the pure-pursuit controller to navigate the vehicle. We also adopt Reeds-Shepp curves [27] to greedily connect every newly added node to the goal pose directly.

Works that alter the tree growth direction in RRT also exist. Apart from the reversed tree in [17], other efforts along this line include [25], [28], wherein two trees are rooted in start and goal poses, respectively. The two trees are simultaneously grown till a certain node on one tree falls close enough to a node on the other tree. We adopt similar approach in our solution; via tests, however, it is found that reversed single-tree growth is faster than bidirectional dual-tree growth. Test results are detailed in following contents.

C. OVERVIEW OF THIS PAPER

See Fig. 1 for an illustration of our RRT-based solution for the automated parking problem. The contributions in this paper are summarized as:

- We propose several algorithmic tweaks to the baseline RRT, i.e., seeds biasing using the expert parking knowledge in different parking spaces, Reeds-Shepp curves directly connecting the tree node and goal pose, and reversed tree growth that originates from the parking pose.
- We perform mathematical analysis for the proposed algorithm, and prove that it retains the completeness as well as feasibility claims of the baseline RRT.
- We examine the proposed solution's performance via batched simulations. We prove that all the three tweaks help to elevate the algorithm's performance. We compare the proposed algorithm to two other RRT variants, and show that it claims shorter planning time, higher success rate, and better convergence to the optimal path. Lastly we justify our algorithm's scalability to different vehicles. We also prove its applicability via a realistic parking problem using a real self-drivable vehicle.

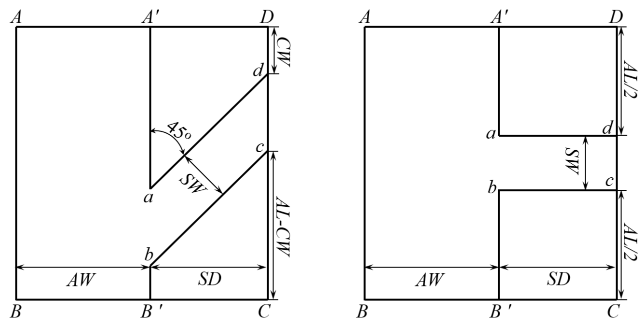

FIGURE 2. Layout for one-way (left, 45°) and two-way (right, 90°) parking; *CW*: car width.

TABLE 1. Parking spaces standards in the municipal documents.

Angle (°)	Stall Width (ft)	Stall Depth (ft)	Aisle Width (ft)
90	8.5∼9.0	18.0∼19.7	23.0∼26.0
75	8.5∼9.0	18.5∼19.5	21.2∼23.0
60	8.5∼9.0	18.0∼21.7	14.0∼18.0
45	8.5∼9.0	17.0∼20.3	11.0∼16.0
30	8.5∼9.0	16.4∼17.7	9.8∼14.0
0	9.0∼10.0	22.0∼24.6	12.0∼23.0

The remainder of this paper is organized as follows: Section II illustrates the general problem formulation. Section III details the algorithm we propose. We also present the associated mathematical analysis. Section IV examines and analyzes the performance of the proposed algorithm. Finally in Section V a general conclusion of this paper is delivered.

II. PROBLEM FORMULATION

A. PARKING SPACES

The problem formulation starts with defining the parking spaces. We review municipal documents from several cities, and allocate 6 most common cases, i.e. 0^o, 30^o, 45^o, 60^o, 75^o, and 90^o parking. The characterization of these parking spaces involves 3 variables: stall width (*SW*), stall depth (*SD*), and aisle width (*AW*), see Fig. 2. We summarize the standards for each variable in the documents, and list them in Table 1. We then specify the parking space sizes by randomly sampling from between these listed thresholds.

Vehicle's goal pose (position, heading) and aisle length (*AL*) also needs to be specified. We put the vehicle's goal position in the center of the parking spot. As for heading, for parking below 75^o, the aisle is usually one way. We thus consider head-in parking only. For 75^o and 90^o parking, two-way aisle may exist. We specify the heading as, if the vehicle starts with pointing into the spot, a head-in parking is defined, otherwise reverse parking. For *AL*, in 75^o and 90^o, we assign 10m on both sides of the spot. For other cases, the spot is put at a certain distance (*CW*) to the far end of *AL*=20m.

B. AGENT VEHICLES

Three agent vehicles are used in our studies, see Table 2. We use all three in batched simulation studies. We also adopt Sedan for the realistic test. All the three vehicles are front

TABLE 2. Agent vehicle specifics.

Agent	Vehicle Width	Vehicle Length	Wheel Base	δ_{max}
Sedan	1.86m	4.93m	2.83m	30^o
Compact	1.80m	4.14m	2.60m	32^o
Truck	2.19m	5.89m	3.71m	32^o

wheel steerable. Given that the speed in parking is relatively low, we adopt the non-slip kinematic bicycle model:

$$\begin{cases} \dot{x} = v cos\theta \\ \dot{y} = v sin\theta \\ \dot{\theta} = v tan\delta / L \end{cases} \tag{1}$$

wherein x and y denote position of the vehicle's rear axle center (anchor point), θ heading (counter-clockwise positive), v speed (forward positive), δ the steering angle (counter-clockwise positive), and L the wheel base. In (1), δ is subject to the maximum angle δ_{max}, i.e. $|\delta| \leq \delta_{max}$. Also we use $v = 2m/s$ throughout our studies, and the steering command is sent to the vehicle at a frequency of $25Hz$.

C. PROBLEM STATEMENT AND RESEARCH OBJECTIVES

We denote the parking space as $\chi \in R^d$ ($\overset{\frown}{ABCD}$), $\mu \in R^m$ the control space of the vehicle, χ_{obs} the obstacles ($\overset{\frown}{A'adD} \cup \overset{\frown}{bB'Cc}$), and $\chi_{free} = \chi / \chi_{obs}$ the free space. The kinematics (Σ) of the vehicle is described in (1). Following [29], the vehicle is small-time controllable from χ if for any moment T, the states reachable from χ before T contains a neighbourhood of χ. We take the assumption that the vehicle is small-time controllable. Our research objectives are then summarized as:

- **Completeness and feasibility**: In the real parking problem, a path $\tau \subset \chi_{free}$ that connects from the start $x_{start} \in \chi_{free}$ to goal $x_{goal} \in \chi_{free}$ exists. Our solution must be able to find one. The vehicle motion is also subject to both geometrical and kinematic constraints (χ_{obs}, Σ). The algorithm must consider such and yield a path that is executable.
- **Improved performance**: We are focused on finding a feasible path. We hope it be fast and reliable (time short and success rate high). We also hope this path converges better to the optimal one (shorter length).
- **Scalability**: Lastly, we hope the above claims hold for all parking spaces (χ and x_{goal} at different angles/sizes), different vehicles, and any potential start poses ($x_{start} \in \chi_{free}$), in both simulation and real environments.

III. SAMPLING-BASED PARKING SOLUTION

A. THE RRT-BASED SCHEME

We adopt RRT algorithm as our solution, see Algorithm 1. The overall algorithmic flow involves 3 stages, i.e. initialization (lines 1-2), RRT tree growth (lines 3-13), and path connection/optimization (lines 14-15). The baseline RRT hinges on the randomized seeds-biased exploration of the environment, which is asymptotically complete if the tree nodes number approaches infinity. We also adopt CL-RRT [26],

Algorithm 1 RRT-Based Solution

```
Input: $\Sigma$, $\chi$, $x_{start}$, $x_{goal}$
Output: $\tau$
1:  $(q_{root}, q_{goal}) \leftarrow Initialize(x_{start}, x_{goal})$
2:  $\zeta.AddNode(q_{root})$
3:  $\Omega_{RS} \leftarrow$ RSConnect($\zeta$.AddNode, $q_{goal}$)
4:  while not $\Omega_{RS}$ and $\zeta$.AddNode $\notin \Omega(q_{goal}, \epsilon)$ do
5:      $x_{rand} \leftarrow$ GenerateSeed($\chi$)
6:      $q_{near} \leftarrow NearNode(\zeta, x_{rand})$
7:      $q_{new} \leftarrow Navigate(\Sigma, q_{near}, x_{rand}, \Delta q)$
8:      $\zeta.AddNode(q_{new})$
9:      $\zeta.AddNavigate(q_{new})$
10:     $\Omega_{RS} \leftarrow$ RSConnect($\zeta$.AddNode, $q_{goal}$)
11: end while
12: $\zeta.AddNode(q_{goal})$
13: $\zeta.AddRSConnect(q_{goal})$
14: $\tilde{\tau} \leftarrow$ LinkFeasiblePath($\zeta$)
15: return: $\tau \leftarrow$ OptimizedPath($\tilde{\tau}$)
```

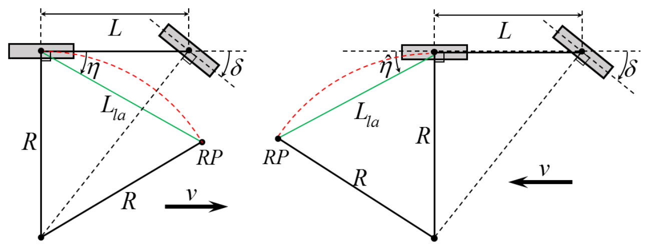

FIGURE 3. Pure-pursuit controller; left: forward drive, right: reverse drive. *RP* is the desired reference point, and red dashed line the expected vehicle trajectory.

feasibility is thus guaranteed. In this part we first briefly introduce the baseline functions in RRT.

NearNode(ζ, x_{rand}) selects from RRT tree (ζ) and returns the closest node q_{near} to x_{rand} (the generated random seed). In our studies, we use 2D Euclidean distance metric.

Navigate($\Sigma, q_{near}, x_{rand}, \Delta q$) expands the tree by driving the vehicle from q_{near} to x_{rand} till a collision is encountered, or the predefined maximum expansion step Δq is reached. We adopt pure-pursuit controller in the vehicle navigation, see Fig. 3. The steering command is given by:

$$\begin{cases} \delta = -tan^{-1}(2Lsin(\eta)/L_{la}) & \text{:forward} \\ \delta = -tan^{-1}(2Lsin(\hat{\eta})/L_{la}) & \text{:reverse} \end{cases} \tag{2}$$

wherein L_{la} is the look-ahead distance, η and $\hat{\eta}$ the angular biasing of *RP* with respect to the vehicle, and R the turning radius. All angles defined are counter-clockwise positive.

To guarantee stability of the controller, L_{la} must be larger than $v\tau_a$, wherein τ_a is the time constant of the steering actuator. Through extensive simulations and field tests, we specify $L_{la} = 1.0m$ in our studies. The defining of Δq denotes how large each stride is taken in expanding the RRT tree. If it is too large, the connected paths may all end up with colliding with the obstacles, or too small a slow planning. We adopt batched simulation-based studies in specifying Δq. Details are illustrated in next section.

AddNode(q_{new}) and **AddNavigate**(q_{new}) add the nodes and branches onto the tree. In our work, we encode the agent's position and heading in the nodes. We also store the previous and immediate next node. The added branch is the navigated path subject to collision-check and kinematic constraints.

LinkFeasiblePath(ζ) links and returns the feasible path ($\tilde{\tau}$) based on the grown RRT tree. In lines 3-4, the tree expansion may be terminated via checking Ω, wherein $\Omega(q_{goal}, \epsilon)$ represents a close neighborhood of q_{goal}, ϵ a predefined small value, and Ω_{RS} indicates if a direct Reeds-Shepp connection exists (detailed in following contents). After the tree growth is terminated, all tree nodes are added subject to the kinematic/collision constraints. We concatenate from the last node (q_{goal}) backwards to the root (q_{root}). We also link each two consecutive nodes via the tree branches, which is the navigated path in each step of the tree expansion.

B. REVERSED RRT TREE GROWTH

Line 1 in Algorithm 1 initializes the RRT tree growth. For the parking problem, the logical flow demands growing a tree from x_{start} to x_{goal}. Based on our studies, the last part of this path finding (squeezing into the spot) is a narrow passage problem, which slows down the planning. We thus adopt a reversed tree growth, i.e. we assign x_{goal} to q_{root}, and grows the tree reversely from the parking spot to the wider aisle, till x_{start} is connected onto the tree.

Claim 1: Via the reversed RRT tree growth, the algorithm is still complete, i.e. it is able to find a path that connects from x_{goal} to x_{start}.

Proof 1: The proof is obvious as the completeness in [13] does not depend on q_{root} or q_{goal}, i.e. the algorithm is complete for any $q_{root} = x_{goal} \in \chi_{free}$ and $q_{goal} = x_{start} \in \chi_{free}$.

Claim 2: After the tree is grown, the returned path $\tilde{\tau}$ that connects from q_{root} (x_{goal}) to q_{goal} (x_{start}) is reversed. This reversed path (τ) is also feasible.

Proof 2: We discuss geometrical and kinematic feasibility separately. For geometrical feasibility, the path $\tilde{\tau}$ is connected from x_{goal} to x_{start}. Essentially, $\tilde{\tau} = \{x_i\}_{i=1,\ldots,n} \subset \zeta$, wherein n is the number of discrete vehicle poses in $\tilde{\tau}$. The geometrical feasibility claim holds at $\forall x_i \in \tilde{\tau}$ which is performed via collision checking between vehicle body and χ_{obs}. Concatenating the $\{x_i\}$ forward or backward does not affect the pointwise collision checking results. The geometrical feasibility thus still holds in τ. ■

Proof 3: For kinematic feasibility, given that the speed in parking is low, we neglect the vehicle's dynamic constraints. We also adopt the assumption that the vehicle is small-time controllable (following [29]). For a certain point $x_i \in \tilde{\tau}$, there exists a next point x_{i+1} which is located at neighborhood of x_i, and reachable from x_i given a certain progressive time $\overrightarrow{t}$. Considering a retrospective time $\overleftarrow{t}$, x_i is also reachable from x_{i+1}, given that one simply needs to opposite the time differentials in Equation (1). Repeating this reachability check from x_{start} to x_{goal}, all $x_i \in \tilde{\tau}$ are reachable from its previous/next neighbor. Reversing $\tilde{\tau}$ thus does not change its kinematic feasibility claim.[1] ■

C. DIRECT CONNECTION VIA REEDS-SHEPP CURVES

In the baseline RRT, the tree is expanded till a node reaches into $\Omega(q_{goal}, \epsilon)$. However this navigation-based expansion is usually slow. In our studies, we propose to use Reeds-Shepp (RS) curves [27] to connect to q_{goal} directly, see lines 3, 10, and 13. Given Table 2, we establish the RS curves using the vehicle's minimum turning radius. Each time a node is added, we connect it to q_{goal} via RS curves directly. We perform collision checking throughout this curve. If any collision is found, we continue to expand the tree (lines 4-10) till a certain node (q_{rs}) is able to connect to q_{goal} with a collision-free RS curve. We add q_{goal} as the last tree node, and the RS curve as the branch linking the last two nodes.

Claim 3: The RS curve-based connection does not change the completeness and feasibility claim of the RRT algorithm.

Proof 4: Completeness of the RRT still holds as in line 4 we continue to use the original terminate rule (ζ.**AddNode** $\notin \Omega(q_{goal}, \epsilon)$). The connection via RS curve hence expedites the tree growth as the computation for RS curve is faster than the navigation-based expansion. For feasibility, we separate $\tilde{\tau}$ into two parts, i.e. from q_{root} to q_{rs} ($\tilde{\tau}_1$), and q_{rs} to q_{goal} ($\tilde{\tau}_2$). In $\tilde{\tau}_1$, the tree expansion follows the original RRT. The feasibility claims holds naturally. In $\tilde{\tau}_2$, we note that we consider minimum turning radius as the kinematic constraints. As the RS curve is established using this radius, it thus represents a shortest-distance connection in the vehicle's (kinematic) configuration space from q_{rs} to q_{goal}, which justifies the kinematic feasibility. We also perform collision checking throughout the RS curve to guarantee the geometrical feasibility. Combining the analysis for both $\tilde{\tau}_1$ and $\tilde{\tau}_2$, feasibility claim of the path ($\tilde{\tau}$) therefore holds. ■

D. KNOWLEDGE-BIASED SAMPLES GENERATION

In line 5 of Algorithm 1, **GenerateSeed**(χ) generates the probing samples x_{rand} which guides the growth of the RRT tree. In the baseline RRT, most of these samples are uniformly distributed in the work space [13]. While this guarantees the feasibility and completeness, the cost is the planning unnecessarily slow, and tree growth may be stuck in narrow passages/corners. In real parking problems, the sizes, angles, and kinematic constraints of both environments and vehicles are usually regulated. Following previous studies [18], [20], [22], we extract these knowledge to bias the RRT seeds generation.

We summarize the parking knowledge via expert demonstrations. For different parking angles, we uniformly sample from between the thresholds in Table 1, and assign different sizes to the 6 parking spaces. We then investigate the parking starting from a randomized $x_{start} \in \chi_{free}$. We adopt RRT to solve this problem till a feasible path is found.

[1]This is also verified by the realistic parking test–the reversely grown RRT path is successfully executed by the vehicle.

We run post-processing to find the optimal path. For each parking angle, we repeat this for 2,000 times, see the clustered heatmap in Fig. 1 and 5. There are certain "hot" areas wherein the optimal path occurs more (e.g. in front of the spot), and "cold" areas wherein the path barely goes into (e.g. corners of the aisle). We hope to generate more samples from the hot areas, and less in the cold.

We adopt Gaussian Mixture Model (GMM) in abstracting the heatmap knowledge. The GMM distribution is written as:

$$\Psi\left(x_i \mid \Theta\right) = \sum_{k=1}^{K} \alpha_k \cdot \psi_k\left(x_i \mid \mu_k, \Sigma_k\right) \tag{3}$$

wherein $x_i \in R^2$ are the parking paths. The GMM is expressed as a linearly weighted combination of K Gaussian Components (GC) ψ_k, wherein α_k is the weighted coefficients for the k-th GC subject to $\alpha_k > 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. Also $\Theta = \{(\alpha_k, \mu_k, \Sigma_k)\}_{k=1}^{K}$ is the parameters set for the GMM, wherein $\mu_k \in R^2$ and $\Sigma_k \in R^2$ are the mean and covariance matrices of the k-th GC component, respectively.

Given x_i, GMM is typically solved via Expectation Maximization (EM) algorithm. Essentially, the EM algorithm is an iterative procedure which converges to a maximum-likelihood estimation of the parameter set Θ in (3). This process consists of the expectation step (E-step) and maximization step (M-step). In E-step, the posterior probability is defined as:

$$P_{ik}\left(x_i \mid \Theta\right) = \frac{\alpha_k \psi_k\left(x_i \mid \mu_k, \Sigma_k\right)}{\sum\limits_{j=1}^{K} \alpha_j \psi_j\left(x_i \mid \mu_j, \Sigma_j\right)} \tag{4}$$

The M-step then proceeds to update Θ via:

$$\begin{cases} \alpha_k = \dfrac{1}{n} \sum\limits_{i=1}^{n} P_{ik}\left(x_i \mid \Theta\right) \\ \mu_k = \dfrac{\sum\limits_{i=1}^{n} \left[P_{ik}\left(x_i \mid \Theta\right) x_i\right]}{\sum\limits_{i=1}^{n} P_{ik}\left(x_i \mid \Theta\right)} \\ \Sigma_k = \dfrac{\sum\limits_{i=1}^{n} \left[P_{ik}\left(x_i \mid \Theta\right)\left(x_i - \mu_k\right)\left(x_i - \mu_k\right)^T\right]}{\sum\limits_{i=1}^{n} P_{ik}\left(x_i \mid \Theta\right)} \end{cases} \tag{5}$$

till the log-likelihood function $L(X \mid \Theta)$ defined in below converges to a local maximum value:

$$\begin{aligned} L\left(X \mid \Theta\right) &= \log \Pi_{i=1}^{n} \Psi\left(x_i \mid \Theta\right) \\ &= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \alpha_k \psi_k\left(x_i \mid \mu_k, \Sigma_k\right) \end{aligned} \tag{6}$$

In (6), $X = \{x_i\}_{i=1,2,\ldots n}$ is the parking paths.

We need to specify an initial value for Θ and K in solving the GMM. In our studies we adopt randomized initial values

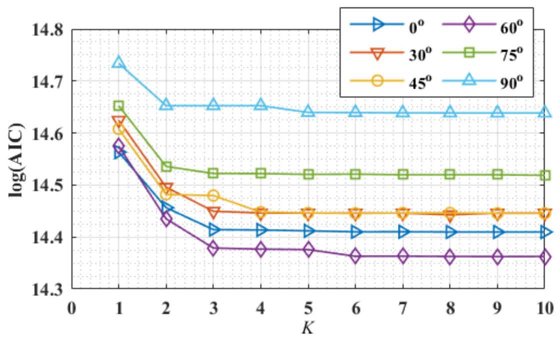

FIGURE 4. GMM components number selection based on AIC.

for Θ. For K, we fully regress the GMM at different K, see Fig. 4. In vertical axis we plot AIC values [30]. Note that we hope the GMM to be both accurate (AIC low) and computationally economic (K small). We thus select $K = 6$ for all parking angles. The regressed GMM is then adopted in generating the seeds, see Fig. 1 and 5, wherein the magenta seeds locate closely to the hot areas in the heatmap. Also on our computational platform, it takes 6E-4 seconds to generate these (2,000) samples, which is considered computationally efficient.

To guarantee completeness of the algorithm, following [13], [22], we include a small portion of seeds that follow the original randomized generation manner in RRT. Prior to the tree expansion, we generate N samples, wherein λN follows the GMM, and the rest $(1 - \lambda)N$ are generated randomly. Following [22] we adopt $\lambda = 95\%$.

Claim 4: The biased seeds generation does not change completeness and feasibility claim of the RRT algorithm.

Proof 4: Note the navigation-based tree expansion and geometric-based collision checking remain the same. This seeds biasing therefore does not affect the feasibility claim. In the biased seeds, we refer to [22], and note that we adopt $(1 - \lambda)N$ samples which follows the original RRT. Given the proof in [13], we thus adjust any references to n (the number of the randomized samples in the original RRT) to $(1 - \lambda)N$ (the number of the randomized samples in our algorithm). Referring to [22], adjusting these samples will only improve the solution or lower the randomness in the RRT exploration, and will not jeopardize the completeness claim. ■

E. POST-PROCESSING FOR OPTIMIZATION

Following [2], [31], we perform optimization after the feasible path is found. We randomly select two indexes on the path, and connect them via a RS curve. If the curve is collision free, it is used to replace the original path. Or if not, we mark this RS connection as a failure, and continue to check another two random indexes. This optimization repeats till a predefined maximum continuous failure number is reached. Given that we are always trying to tune the path via RS curve, the length is being refined in a non-increasing manner. We then adopt the final path as the optimized result.

IV. EXPERIMENTAL TESTS AND ANALYSIS

A. EXPERIMENTAL SETUP

In Algorithm 1 the RRT tree is grown till a collision free RS curve is connected to the goal point, or a certain branch on the tree reaches into a neighborhood of the goal. In our studies, we define this neighborhood as distance and heading difference both lower than 0.01 (unit meter or radius). We also define success as planning being completed within 30 seconds. We perform batched simulation-based analysis to specify Δq, see Fig. 6. We randomly sample from Table 1 to characterize the parking spaces. We then start from a randomized pose, and deploy the RRT (with RS connection) to find the parking path. The results in Fig. 6 are averaged from a batch of 100 simulation runs. Fig. 6 corresponds to the physical understanding of Δq, i.e., it represents the "resolution" in exploring the environment. Too large Δq will cause the tree to collide onto obstacles (hence longer time and lower success rate), and too small Δq may render the exploration too slow. Certainly in our algorithm we adopt RS curves which expedites the planning. Planning time and success rate at lower Δq remains steady as most of the RRT tree growth may be completed via RS connections. Based on Fig. 6, we select $\Delta q = 0.48m$ throughout our studies.

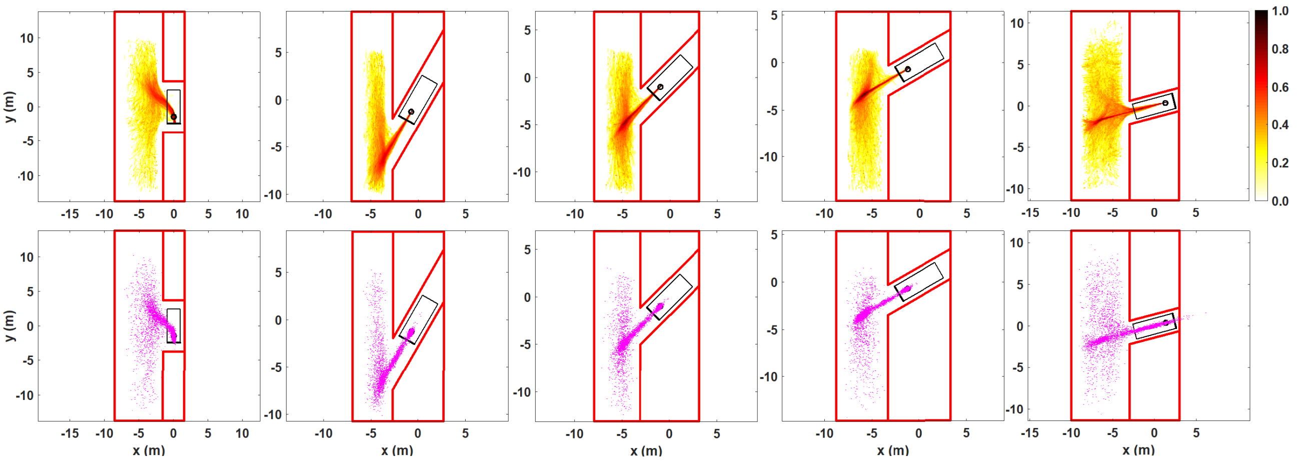

FIGURE 5. Optimal path heatmap (top) and samples generations (bottom) for 0^o, 30^o, 45^o, 60^o, and 70^o parking (left to right).

TABLE 3. Comparative study of the algorithms in different parking environments.

	Average Time (s)	Success Rate (%)	Normalized Time (s)	Optimality (-)		Average Time (s)	Success Rate (%)	Normalized Time (s)	Optimality (-)
0^o parking angle					**30^o parking angle**				
RRT	19.3864	23	84.2886	3.6330	RRT	15.9774	22	72.6245	3.6016
F-RRT-RS	10.8430	75	14.4573	3.5082	F-RRT-RS	11.7788	45	26.1751	3.0547
R-RRT-RS	10.5288	58	18.1531	1.5873	R-RRT-RS	11.6092	80	14.5115	1.7922
GMM-F-RRT-RS	6.0991	87	7.0105	2.9766	GMM-F-RRT-RS	3.3855	81	4.1796	2.6081
GMM-R-RRT-RS	2.0652	92	2.2448	1.2893	GMM-R-RRT-RS	3.8215	91	4.1995	1.3650
GMM-R-RRT*-RS	4.9846	73	6.8283	1.2224	GMM-R-RRT*-RS	4.0834	70	5.8335	1.2044
BiRRT*	19.6542	14	140.3871	1.6919	BiRRT*	33.0223	48	68.7964	2.3466
POSQ-RRT*	16.8061	16	105.0381	3.0040	POSQ-RRT*	16.7210	18	92.8944	3.2494
45^o parking angle					**60^o parking angle**				
RRT	18.2453	34	53.6625	3.2572	RRT	34.9846	9	388.7174	3.4346
F-RRT-RS	13.5288	55	24.5978	2.6899	F-RRT-RS	33.9088	36	94.1911	2.3503
R-RRT-RS	8.9728	83	10.8106	1.8723	R-RRT-RS	12.3225	70	17.6036	1.8698
GMM-F-RRT-RS	7.2889	83	8.7818	2.4491	GMM-F-RRT-RS	18.3639	74	24.8161	2.2938
GMM-R-RRT-RS	4.6502	92	5.0546	1.4086	GMM-R-RRT-RS	8.9537	82	10.9191	1.8486
GMM-R-RRT*-RS	5.2508	68	7.7217	1.2869	GMM-R-RRT*-RS	9.2613	51	18.1595	1.3043
BiRRT*	28.4283	32	88.8385	2.2510	BiRRT*	26.3320	44	59.8454	2.1658
POSQ-RRT*	24.8252	9	275.8351	3.6234	POSQ-RRT*	36.0759	10	360.7594	2.9074
75^o parking angle					**90^o parking angle**				
RRT	31.4585	25	125.8339	4.0625	RRT	33.9833	30	113.2776	3.9940
F-RRT-RS	24.7412	55	44.984	2.1190	F-RRT-RS	17.5765	68	25.8478	1.8596
R-RRT-RS	12.183	61	19.9721	1.7650	R-RRT-RS	8.5154	83	10.2595	1.7427
GMM-F-RRT-RS	11.3411	75	15.1215	1.8105	GMM-F-RRT-RS	10.8942	78	13.9669	1.7879
GMM-R-RRT-RS	5.2378	87	6.0205	1.4412	GMM-R-RRT-RS	5.7672	91	6.3376	1.5091
GMM-R-RRT*-RS	6.0659	55	11.0298	1.3664	GMM-R-RRT*-RS	5.9097	59	10.0164	1.4380
BiRRT*	25.2595	31	81.4824	1.9230	BiRRT*	12.2060	10	122.0600	2.1246
POSQ-RRT*	31.1866	13	239.8967	3.9190	POSQ-RRT*	39.6408	19	208.6360	3.0422

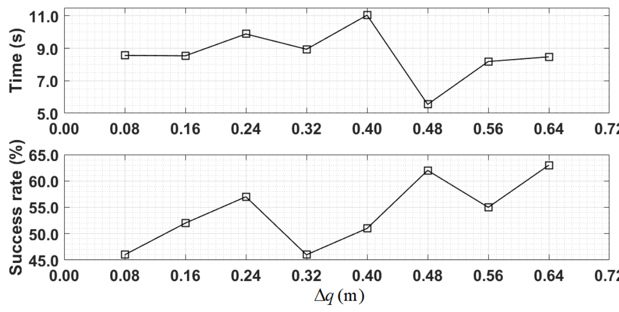

FIGURE 6. To decide Δq based on planning time and success rate.

B. PERFORMANCE OF THE PROPOSED ALGORITHM

Batched simulations are conducted, see Table 3. We run these tests on our platform (i7-8700 CPU, 16GB RAM) in MATLAB 2019b using Sedan, results averaged from 100 runs. The algorithm is executed till a feasible path is found, of which the mean time and success rate are allocated. Normalized time is defined as mean time divided over success rate. We run post-optimization to acquire the optimized path, and define optimality as feasible path length normalized by the optimized one. All values in Table 3 are lower the better except success rate, which is higher the better.

1) THE ADOPTION OF RS CONNECTION

The effects of RS connection are concluded by comparing the results of baseline RRT and F-RRT-RS/R-RRT-RS, wherein "R" represents reversed tree, and "F" the normal. The average time drops in the RS-connected RRT, and success rate higher. As direct RS connection is shorter than navigation-based tree growth, optimality of RS-connected path is also better.

2) REVERSED TREE GROWTH

Comparing F-RRT-RS and R-RRT-RS, in $30^o - 90^o$ the reversed tree claims better results. Moreover, optimality of R-RRT-RS is better: the vehicle firstly moves out of the narrow passage, which renders it easier to connect the shorter RS curves. Although the performance in 0^o parking is not excessively better, given the results in other cases we show the reversed tree growth improves the algorithm.

3) GMM-BASED BIASING

The last tweak we adopt is the GMM biasing, i.e. GMM-F-RRT-RS/GMM-R-RRT-RS. Note that in the discussion of reversed tree growth, both normal and reversed tree show fair results. We thus keep both in this round of comparison. As shown in the table, GMM successfully improves both time and success rate. Also as GMM biases the tree growth close to the optimal paths, it claims an overall better optimality. Lastly, the reversed algorithm in 0^o parking is better: the GMM helps to regulate (suppress) the randomness in the RRT, and the algorithm's performance for all cases tends to be unified.

4) EFFECTS OF GMM TO OPTIMALITY

GMM also improves the optimality of the path, see GMM-R-RRT-RS and GMM-R-RRT*-RS, wherein the latter adopts RRT* in tuning the tree branch. The GMM-baised RRT solution claims better optimality over other baseline algorithms, which is the closest to the RRT*-based scheme. Time of GMM-R-RRT-RS, however, is faster than the RRT*-based scheme.

5) COMPARISON TO TWO OTHER RRT VARIANTS

Via the aforementioned discussion, we conclude that reversed RRT tree growth enhanced with RS connection and GMM

TABLE 4. Performance of the proposed algorithm for different vehicles.

	Average Time (s)	Success Rate (%)	Normalized Time (s)	Optimality (-)		Average Time (s)	Success Rate (%)	Normalized Time (s)	Optimality (-)
0^o parking angle					**30^o parking angle**				
Compact	1.1029	95	1.1609	1.2067	Compact	2.6140	94	2.7808	1.2293
Truck	4.2000	90	4.6667	1.4999	Truck	4.0470	89	4.5472	1.4707
45^o parking angle					**60^o parking angle**				
Compact	2.5199	94	2.6808	1.2731	Compact	2.0025	92	2.1766	1.2449
Truck	7.5223	89	8.4520	1.7142	Truck	13.2016	82	16.0995	1.5347
75^o parking angle					**90^o parking angle**				
Compact	2.8049	95	2.9525	1.2716	Compact	2.1617	96	2.2518	1.3613
Truck	6.1379	86	7.1371	1.6152	Truck	7.6197	90	8.4663	1.7026

biasing yields the best results. To further examine its performance, we compare the algorithm with two other RRT variants, i.e. BiRRT* [28] and POSQ-RRT* [24].

BiRRT* adopts the RRT* scheme, which is slower than the baseline RRT. However the dual tree growth in BiRRT* may expedite the planning, and in some cases BiRRT* is indeed faster than the baseline RRT (in 60^o, 75^o, and 90^o parking angles). Compared to our solution (GMM-R-RRT-RS), however, BiRRT* is slower and success rate lower in all cases. Optimality of GMM-R-RRT-RS is also better than BiRRT* in all parking angles.

In POSQ-RRT*, the major improvement is the POSQ extension function. It manipulates the steering (heading) in the lateral plane, and also creates a smoother speed profile in navigating the vehicle. However, this controller needs heuristics-based tuning (which was not clarified in [24], the authors simply list their choice of the parameters). In our work we adopt $K_\rho = 4$, $K_\phi = -2$, $K_\alpha = 2$, $K_v = 3.8$ and $\gamma = 0.15$ (see [24] for the parameters definition). POSQ-RRT* is generally slower even than the baseline RRT as it adopts the RRT* scheme, and POSQ controller is not designed to expedite it. Overall, compared to both our solution and the baseline RRT, POSQ-RRT* is slower, success rate lower, and optimality not excessively better.

C. SCALABILITY: TO DIFFERENT VEHICLES

We adopt Sedan to generate the optimal parking path in regressing the GMM knowledge. In this part we check this Sedan-based GMM's scalability to other vehicles, see Table 4. Compact and Truck are used, and results averaged from 100 simulation runs. Comparing Table 4 to Table 3, the algorithm yields similar performance with respect to both time, success rate, and optimality. Scalability of the proposed algorithm to different vehicles is considered acceptable.

D. SCALABILITY: FEASIBILITY IN REAL PARKING PROBLEMS

Due to briefness we present one realistic test for 90^o parking. We adopt a self-drivable Lincoln MKZ in our test. This vehicle is equipped with RTK GPS to provide accurate positioning information. ROS middleware is installed to communicate with the vehicle's CAN bus (to send and receive steering/throttle/brake messages). We adopt PI controller to manipulate throttle and brake for the longitudinal speed control. Lateral steering angle is updated via the

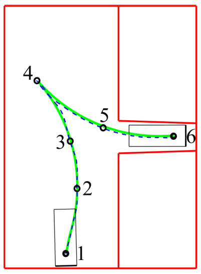

FIGURE 7. Illustrative results for the realistic test. Green thick: path from our algorithm; blue dashed: vehicle's real trajectory. Both corresponds well, which justifies the feasibility of the planned path. We also mark 6 milestones. See the associated video frames in Fig. 8.

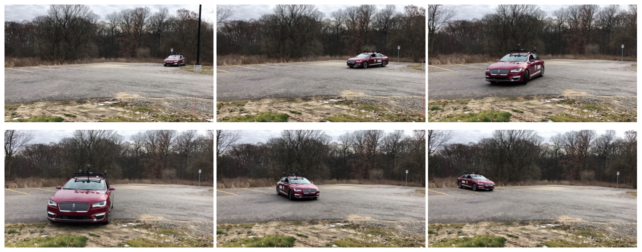

FIGURE 8. Recorded milestone video frames in the realistic parking test. From top to bottom, left to right: 1-6 as shown in Fig. 7.

aforementioned pure-pursuit controller (for both forward and backward motion).

In the test, we measure the size of a realistic parking spot, we then build up the work space in Fig. 7. Note the two lines to the port/starboard sides of the parking spot are not in parallel, which does not follow the standards in Table 1 and reflects the challenges in realistic parking. We deploy GMM-R-RRT-RS to find the feasible path, post-optimization is then adopted. See Fig. 7, the vehicle is driven from milestone 1, moves through 2 and 3 till 4, wherein the vehicle slows down to halt. It then reverses through 5 to the parking spot at 6. Associated video frames are listed in Fig. 8, vehicle response history is also plotted in Fig. 9. The path yielded from our parking solution presents itself feasible. Also although the parking space is not regulated (parking lines not in parallel) and differs from the standards (Table 1) that our GMM knowledge roots in, our solution still proves to be applicable.

E. RESULTS SUMMARY

To summarize, the simulation and realistic test results in this chapter present the superiority of our parking solution as:

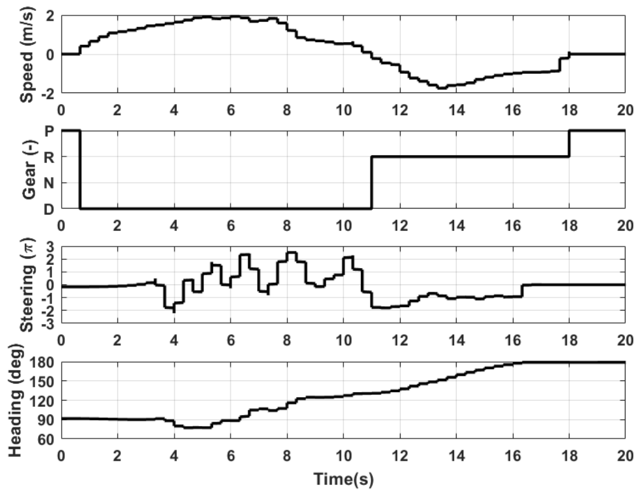

FIGURE 9. Vehicle response history in the realistic parking test.

- **Improved performance**: Via Table 3 we show that the algorithmic tweaks we propose are effective. Our proposed parking solution exhibits better performances (planning time shorter, success rate higher, path optimality better) when compared with both the baseline RRT, and two other RRT variants.
- **Scalability to different environments and vehicles**: Implicitly in Table 3, we consider different (randomized) vehicle start poses and parking environments. We show that the proposed algorithm yields better performances. Via Table 4, we check the algorithm's scalability to different vehicles, which is also promising.
- **Scalability to realistic parking problems**: We lastly justify the algorithm's applicability via a real self-drivable vehicle in a realistic parking test.

V. CONCLUSION

In this paper we propose an unified solution for the automated vehicles parking problem. To guarantee the completeness and feasibility in different parking environments, we adopt RRT-based scheme. To overcome some deficiencies of the baseline RRT algorithm, we propose several algorithmic tweaks, i.e. reversed RRT tree growth, direct tree branch connections via Reeds-Shepp curves, and regulated knowledge-based RRT seeds biasing. We prove that under these tweaks, the algorithm is still complete and feasible. We then perform batched simulation-based comparative studies to examine its performance. We also check its scalability to different vehicles, and testify the applicability to real vehicles in realistic parking problems. Via the presented results, we show that our algorithm is better and promising for real applications.

REFERENCES

[1] H. Peng and M. Tomizuka, "Preview control for vehicle lateral guidance in highway automation," in *Proc. Amer. Control Conf.*, Jun. 1991, pp. 3090–3095.

[2] Y. Dong, Y. Zhang, and J. Ai, "Experimental test of unmanned ground vehicle delivering goods using RRT path planning algorithm," *Unmanned Syst.*, vol. 5, no. 1, pp. 45–57, Jan. 2017.

[3] Y. Dong, Y. Zhong, W. Yu, M. Zhu, P. Lu, Y. Fang, J. Hong, and H. Peng, "Mcity data collection for automated vehicles study," 2019, *arXiv:1912.06258*. [Online]. Available: http://arxiv.org/abs/1912.06258

[4] M. F. Hsieh and U. Ozguner, "A parking algorithm for an autonomous vehicle," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 1155–1160.

[5] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar, "Geometric continuous-curvature path planning for automatic parallel parking," in *Proc. 10th IEEE Int. Conf. Netw., Sens. Control (ICNSC)*, Apr. 2013, pp. 418–423.

[6] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking with geometric continuous-curvature path planning," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 465–471.

[7] Z. Liang, G. Zheng, and J. Li, "Automatic parking path optimization based on Bezier curve fitting," in *Proc. IEEE Int. Conf. Autom. Logistics*, Aug. 2012, pp. 583–587.

[8] S. Ma, H. Jiang, M. Han, J. Xie, and C. Li, "Research on automatic parking systems based on parking scene recognition," *IEEE Access*, vol. 5, pp. 21901–21917, 2017.

[9] T.-H. Hsu, J.-F. Liu, P.-N. Yu, W.-S. Lee, and J.-S. Hsu, "Development of an automatic parking system for vehicle," in *Proc. IEEE Vehicle Power Propuls. Conf.*, Sep. 2008, pp. 1–6.

[10] R. Kummerle, D. Hahnel, D. Dolgov, S. Thrun, and W. Burgard, "Autonomous driving in a multi-level parking structure," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 3395–3400.

[11] Y. Dong, Y. Zhang, and J. Ai, "Experimental test of artificial potential field-based automobiles automated perpendicular parking," *Int. J. Veh. Technol.*, vol. 2016, pp. 1–10, Oct. 2016.

[12] H. Fuji, J. Xiang, Y. Tazaki, B. Levedahl, and T. Suzuki, "Trajectory planning for automated parking using multi-resolution state roadmap considering non-holonomic constraints," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 407–413.

[13] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[14] L. Han, Q. H. Do, and S. Mita, "Unified path planner for parking an autonomous vehicle based on RRT," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 5622–5627.

[15] S. Shin, J. Ahn, and J. Park, "Desired orientation RRT (DO-RRT) for autonomous vehicle in narrow cluttered spaces," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4736–4741.

[16] B. Lee, Y. Wei, and I. Y. Guo, "Automatic parking of self-driving car based on lidar," *ISPRS - Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vols. XLII–2/W7, pp. 241–246, Sep. 2017.

[17] Z. Feng, S. Chen, Y. Chen, and N. Zheng, "Model-based decision making with imagination for autonomous parking," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 2216–2223.

[18] M. Kalisiak and M. van de Panne, "Faster motion planning using learned local viability models," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 2700–2705.

[19] J. Denny, E. Greco, S. Thomas, and N. M. Amato, "MARRT: Medial axis biased rapidly-exploring random trees," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 90–97.

[20] Y. Dong, E. Camci, and E. Kayacan, "Faster RRT-based nonholonomic path planning in 2D building environments using skeleton-constrained path biasing," *J. Intell. Robot. Syst.*, vol. 89, nos. 3–4, pp. 387–401, Mar. 2018.

[21] J. Wang and M. Q.-H. Meng, "Optimal path planning using generalized Voronoi graph and multiple potential functions," *IEEE Trans. Ind. Electron.*, early access, Jan. 1, 2020, doi: 10.1109/TIE.2019.2962425.

[22] B. Ichter, J. Harrison, and M. Pavone, "Learning sampling distributions for robot motion planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7087–7094.

[23] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5054–5061.

[24] L. Palmieri and K. O. Arras, "A novel RRT extend function for efficient and smooth mobile robot motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 205–211.

[25] M. Sobue and H. Fujimoto, "RRT-based path planning considering initial and final pose under curvature constraints for nonholonomic wheeled robot," in *Proc. IECON-45th Annu. Conf. IEEE Ind. Electron. Soc.*, vol. 1, Oct. 2019, pp. 724–731.

[26] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *Proc. AIAA Guid., Navigat. Control Conf. Exhib.*, Aug. 2008, p. 7166.

[27] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific J. Math.*, vol. 145, no. 2, pp. 367–393, Oct. 1990.
[28] M. Jordan and A. Perez, "Optimal bidirectional rapidly-exploring random trees," Comput. Sci. Artif. Intell. Lab., Tech. Rep. MIT-CSAIL-TR-2013-021:1–12, 2013.
[29] L. Palmieri, S. Koenig, and K. O. Arras, "RRT-based nonholonomic motion planning using any-angle path biasing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2775–2781.
[30] H. Akaike, "Information theory and an extension of the maximum likelihood principle," in *Selected Papers of Hirotugu Akaike*. Springer, 1998, pp. 199–213.
[31] Y. Dong and Y. Zhang, "Application of RRT algorithm to unmanned ground vehicle motion planning and obstacle avoidance," in *Proc. Int. Conf. Intell. Unmanned Syst.*, vol. 11, 2015.

YIQUN DONG was born in Jiangsu, China, in 1990. He received the B.E. and Ph.D. degrees in aerospace engineering from Fudan University, Shanghai, China, in 2010 and 2016, respectively.

From 2014 to 2016, he was an Exchange Ph.D. Student with the Department of Mechanical, Industrial, and Aerospace Engineering, Concordia University, Montreal, QC, Canada. From 2016 to 2017, he was a Postdoctoral Research Fellow with the Department of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. From 2017 to 2019, he was a Postdoctoral Research Fellow with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI, USA. Since 2019, he has been an Associate Professor with the Department of Aeronautics and Astronautics, Fudan University. He has published more than 30 highly-cited journal/conference papers. His research interests include path planning, decision making in intelligent systems, aerospace engineering, and automated vehicles.

YUANXIN ZHONG was born in Hubei, China, in 1997. He received the B.E. degree in vehicle engineering from Tsinghua University, Beijing, China, in 2018. He is currently pursuing the Ph.D. degree in mechanical engineering with the University of Michigan, Ann Arbor, MI, USA.

Since 2018, he has been a Research Assistant with the Vehicle Dynamics Laboratory, University of Michigan. He is also exploring sensor fusion and temporal fusion of different onboard sensors. His research interests include path planning, lidar-based perception for robots, and autonomous vehicles using deep learning techniques.

JIAJUN HONG was born in Shanghai, China, in 1996. He received the B.E. degree in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2018, and the M.S. degree in mechanical engineering and electrical and computer engineering from the University of Michigan, Ann Arbor, MI, USA, in 2020. His research interests include path planning algorithms, perception system for robots, and machine learning.