# Towards Robust Autonomous Coverage Navigation for Carlike Robots

Zhong Wang, Ying Shen, Lin Zhang, *Senior Member, IEEE*, Shaoming Zhang, and Yicong Zhou, *Senior Member, IEEE*

***Abstract*—Thanks to their high carrying capacity and strong maneuverability, carlike robots which move with non-holonomic constraints, are frequently utilized in numerous coverage operation fields. In such fields, the robots need to complete the coverage task via autonomous path planning and tracking, which is referred to as a "Coverage Navigation for Carlike Robots" (CNCR) problem. When it comes to CNCR, two folds of challenges are put forward as follows. On one hand, the local path planner should not only consider the non-holonomic constraints of carlike robots but also should meet the high coverage rate required by the coverage task. On the other hand, during CNCR, there must be many U-turns in narrow areas. As a result, Reeds-Shepp-type paths will be planned inevitably. Tracking this sort of multi-segment paths stably is also an urgent problem to be solved in CNCR. To this end, this letter introduces a reference line constraint and a heuristic initialization strategy along with a cubic parametric path smoother to the time elastic band planner and designs a multi-segment-supported path tracker. Experimental results show that the proposed path planner largely reduces the average deviation off the reference line and obtains a high success rate even in complex environments. Besides, the designed path tracker achieves stable and high-precision tracking for multi-segment paths. Real-world experiments reveal that the proposed path planner and tracker are fully competent for CNCR, rendering our sweeping robot to attain an actual average coverage rate of 92.21%. To make our results fully reproducible, relevant codes and data are made available online at https://github.com/csLinZhang/cncr.git.**

***Index Terms*—Autonomous vehicle navigation, motion control, motion and path planning.**

## I. Introduction

IN MANY coverage operation fields like agricultural harvesting, industrial assembly and site cleaning, carlike robots which move with non-holonomic constraints (when the controllable degrees of freedom (DoFs) are less than the total DoFs), are popularly applied owing to their high carrying capacity and strong maneuverability. In such fields, the carlike robots need to complete the full coverage task via autonomous path planning and tracking, which is referred to as a "Coverage Navigation for Carlike Robots" (CNCR) problem.

Usually, a coverage navigation task can be subdivided into a series of point-to-point navigation (P2PN) subtasks according to the coverage path guidance. Reliable pose estimation [1], [2], high-precision map construction [2], [3], semantic environment perception [4], [5], safe-feasible path planning [6], resource optimization [7], [8] as well as accurate-stable path tracking [9] are the key modules in autonomous P2PN.

When it comes to CNCR, more challenges for the related technologies are put forward. Apart from the well-studied localization, mapping, and perception [3], [5], [10], [11], there are at least two crucial problems to be solved. First, the local path planning is no longer a simple point-to-point shortest path-finding problem. In one aspect, it needs to fit the reference path as much as possible for a higher coverage rate. In another aspect, aside from considering obstacle avoidance and time efficiency, the non-holonomic motion constraints also must be complied with (a carlike robot has three DoFs, however, it has only two controllable DoFs, i.e., the acceleration and the steering). Although the time elastic band path planner (TEB) [12], which takes minimizing the time consumed as its optimization objective and models obstacles and motion constraints into a unified graph optimization problem, can meet these requirements to a certain extent, it does not take the reference line deviation into account. Besides, TEB's initialization strategy with straight-line segments makes it difficult to find feasible paths in crowded environments. Second, to turn around in narrow spaces, Reeds-Shepp-type paths [13] will inevitably appear. Although some path tracking methods [14]–[17] have high tracking accuracies for low speed carlike robots, most of them only support forward trackings with constant speeds, leading to failures in the trackings of multi-segment paths which need both forward and backward movements.

Considering the above-mentioned challenges faced by autonomous CNCR, in this letter, the involved path planner and path tracker are carefully designed, and a carlike sweeping robot (as a typical CNCR system) has been built to examine their performance. In summary, our contributions are threefold.

Manuscript received May 26, 2021; accepted September 16, 2021. Date of publication September 29, 2021; date of current version October 11, 2021. This letter was recommended for publication by Editor Dan Popa upon evaluation of the Associate Editor and reviewers' comments. This work was supported in part by the National Natural Science Foundation of China under Grants 61973235, 61936014, and 61972285, in part by the Natural Science Foundation of Shanghai under Grant 19ZR1461300, in part by the Shanghai Science and Technology Innovation Plan under Grant 20510760400, in part by the Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0100, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Ying Shen; Lin Zhang.)*

Zhong Wang, Ying Shen, and Lin Zhang are with the School of Software Engineering, Tongji University, Shanghai 201804, China (e-mail: 2010194@tongji.edu.cn; yingshen@tongji.edu.cn; cslinzhang@tongji.edu.cn).

Shaoming Zhang is with the School of Surveying and Geo-Informatics, Tongji University, Shanghai 201804, China (e-mail: zhangshaoming@tongji.edu.cn).

Yicong Zhou is with the Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: yicongzhou@um.edu.mo).

Digital Object Identifier 10.1109/LRA.2021.3116299

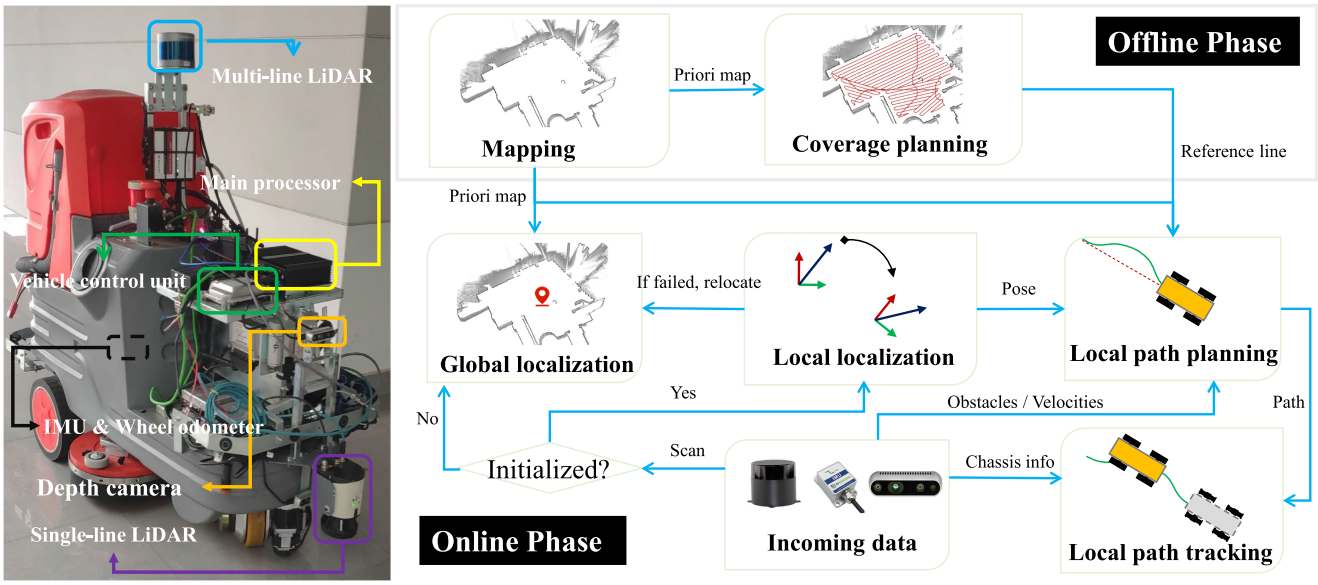


Fig. 1. Our sweeping robot "Raccoon" is a typical CNCR system. Its hardware sensors include two LiDARs, one depth camera, an IMU, and a speedometer. It integrates the modules of mapping, localization, perception, planning, and tracking into a unified software framework, and achieves autonomous coverage navigation.

- When performing planning for CNCR, a key issue is how to make the planned path fit the given reference line while meeting the motion constraints and safety requirements. Therefore, based on TEB [12], this letter introduces a reference line deviation cost term, which greatly reduces the average deviation between the planned path and the reference line. Besides, stable local planning is the premise to ensure the success of the coverage task. To this end, an initialization strategy incorporating the A* algorithm [18] and a path smoother is introduced, which makes the success rate of the local planner reach 98.10% even in complex environments.
- To track the Reeds-Shepp-type curves, based on the geometric model of the carlike robot, the steering angle control mechanism for backward tracking is determined first. Then, the associated speed control scheme is designed by dynamically adjusting the look-ahead distance. The outstanding performance of the resulting path tracker has been corroborated by experiments.
- To examine the actual performance of the designed local planner and path tracker for CNCR, we built an integrated platform of a sweeping robot (as shown in Fig. 1) and carried out extensive coverage navigation experiments in real scenes (such as multiplex polygonal lobbies, narrow long corridors, etc.). The results show that the proposed planner and tracker can effectively adapt to CNCR, making our sweeping robot attain an actual average coverage rate of 92.21%. A running demo can be found online on this website[1].

[1]https://youtu.be/5z6NwxYF-Bw

## II. Related Work

### A. Researches on Point-to-Point Path Planners

Existing point-to-point path planners can be roughly categorized into graph-search based ones, sampling-based ones, curve-interpolation based ones, and numerical-optimization based ones [6]. Among them, graph-search based approaches are the earliest technologies for point-to-point planning. They search the shortest paths in structured graphs to get solutions. A well-known graph-search-based algorithm was proposed by Dijkstra [19]. Inspired by Dijkstra's eminent idea [19], Hart *et al.* put forward a heuristic search strategy, which greatly reduced the search time [18]. Sampling-based schemes were originally introduced to deal with the path-finding problem in high-dimensional space. In 1998, a representative sampling-based approach "rapidly exploring random tree" (RRT) [20] was proposed, and then a series of sampling-based studies have evolved successively, such as RRT* [21] and RRT-Connect [22]. Curve-interpolation based ones obtain feasible paths by interpolating predefined curves conforming to vehicle's motion constraints, such as polynomial, spiral, Bezier curve, etc [23]–[27]. In recent years, methods based on numerical optimization have attracted great attention. They can directly combine motion parameters such as speed, acceleration, steering, and jitter into a unified model, and have been adopted by various autonomous driving systems [12], [28]–[30].

Among all the aforementioned methods based on numerical optimization, TEB [12] is the most promising one, especially for the case of low speed driving. This method shows superior performance in general local point-to-point planning. However, as the algorithm uses a straight line between the starting point and the endpoint as its initial path, its success rate is not high in crowded environments.

### B. Researches on Path Trackers

So far, researches related to path tracking can be summarized into three types, geometric-model based ones, kinematic-model based ones, and dynamic-model based ones [9]. Among them, the geometric-model based approaches are the most classic path trackers. They are devised based on well understood principles and have good practicability, showing impressive performance in low speed vehicles [14]–[17]. Kinematic-model based approaches optimize tracking commands by combining the kinematic constraints of vehicles and adopting chained rules in control theory [31]. Methods based on dynamic models need accurate kinematic and dynamic parameters of vehicles and can build the most extensive complex motion analysis systems, which are mostly used for the control of fast-moving autonomous vehicles [32], [33].

From the above analysis, it is noticeable that the geometric-model based trackers are better choices in low speed driving. But, these kinds of methods were originally designed to trace forward paths with constant speeds. To meet the requirements of Reeds-Shepp-type path tracking, the backward control mechanism of the steering angle needs to be determined. Besides, the fixed look-ahead distance strategies used in the traditional geometric trackers have at least two problems in multi-segment path tracking. One is that the steering angle jumps at the switching point (a point $p_k$ where the direction from its previous point $p_{k-1}$ to $p_k$ is opposite to that from $p_k$ to the next point $p_{k+1}$). The other is that the fixed speed will lead to a serious tracking deviation at the switching point.

## III. $\mathrm{TEB}_{\mathrm{CNCR}}$: A Robust Path Planner for CNCR

Our local path planner $\mathrm{TEB}_{\mathrm{CNCR}}$ is inspired by TEB [12]. In this part, we will first present the model construction of the original TEB and then extend it to meet the requirements of local planning of CNCR.

### A. Time Elastic Band Formulation

TEB takes the shortest time to get to the goal as its optimization objective, which can be formed as,

$$\min_{\mathcal{B}} \sum_{k=1}^{n-1} \Delta T_k^2, \tag{1}$$

subject to,

$$0 \le \Delta T_k \le \Delta T_{max},\ c_1 = c_r,\ c_n = c_f,$$

$$\mathbf{h}_k(c_{k+1}, c_k) = \mathbf{0},\ \mathbf{r}_k(c_{k+1}, c_k) \ge 0,\ \mathbf{o}_k(c_k) \ge \mathbf{0},$$

$$\mathbf{v}_k(c_{k+1}, c_k, \Delta T_k) \ge \mathbf{0},\ \mathbf{a}_k(c_{k+1}, c_k, \Delta T_k) \ge \mathbf{0},$$

where $\mathcal{B} : \{c_1, \Delta T_1, c_2, \Delta T_2, \ldots, c_{n-1}, \Delta T_{n-1}, c_n\}$ represents the set of parameters constituting the optimization problem. Each $c_i$ ($i \in [1, n], n \in Z^+$) stands for a robot configuration on the planned trajectory and each $\Delta T_k$ ($k \in [1, n-1]$) denotes a strictly positive time interval between $c_k$ and $c_{k+1}$. The first configuration $c_1$ is fixed to robot's current pose $c_r$ and the last configuration $c_n$ is assigned to the desired configuration $c_f$. $\mathbf{h}_k(\cdot)$ imposes kinodynamic constraints and $\mathbf{r}_k(\cdot)$ denotes the constraint for the carlike robot's minimum turning radius. $\mathbf{o}_k(\cdot)$ takes obstacles into account. $\mathbf{v}_k(\cdot)$ and $\mathbf{a}_k(\cdot)$ are velocity-related and acceleration-related operators, which limit the robot's motion states to acceptable bounds. The problem defined by Eq. 1 can be regarded as a generalized optimization problem. By transforming it into a corresponding convex dual problem, it can be solved efficiently by employing mature convex optimization technologies (the Karush-Kuhn-Tucker conditions and the Levenberg-Marquardt algorithm).

### B. Reference Line Constraint

The coverage task requires that the robot's locally planned paths should fit the coverage path as much as possible. To meet such a requirement, based on TEB, we introduce a reference line constraint $\mathbf{l}_k(\cdot)$,

$$\mathbf{l}_k(c_k, \tilde{\mathcal{P}}_\perp) = \|c_k, \tilde{\mathcal{P}}_\perp\| \le \tilde{\tau}, \tag{2}$$

to prevent the deviation of the planned path from the reference line, where $\|c_k, \tilde{\mathcal{P}}_\perp\|$ denotes the nearest Euclidean distance between the configuration $c_k$ and the reference path $\tilde{\mathcal{P}}$, and $\tilde{\tau}$ represents the allowable deviation bound.

It should be noted that in the parameter settings, the weight of the constraint $\mathbf{l}_k(\cdot)$ should be lower than the obstacle constraint $\mathbf{o}_k(\cdot)$. By doing so, for $\mathbf{l}_k(\cdot)$ is a soft constraint, it can make the robot give priority to safety and allow a certain degree of deviation off the reference line in obstacle avoidance. When there is no obstacle around, the term $\mathbf{o}_k(\cdot)$ no longer works and the constraint $\mathbf{l}_k(\cdot)$ will take effect.

### C. Heuristic Initialization

Since the original TEB is initialized with the line segment between the robot pose and the goal pose [12], it is easy to fall into local optimal solutions in crowded environments. Therefore, to ensure the stability of the solution, in each planning cycle, we first check whether there are obstacles in a certain range of the current local reference path. If so, A* [18] will be used to search for a safe and collision-free temporary reference path to replace the original one.

Moreover, as the path searched by A* is composed of discrete nodes, it needs to be further smoothed for the robot's stable movement. To this end, we devise an iterative path smoother as shown in Algorithm 1 to smooth the output of A*. The smoother is built atop a cubic function because the cubic polynomial curve is a common data smoothing tool due to its concise expression, high computational efficiency, and second-order differentiability [34]. To improve the universality of the path smoother, we separately perform parametric curve fitting on the path's abscissa and ordinate. The two curve fittings are both achieved by two-fold iterations (line 7 to line 24 of Algorithm 1). In each outer iteration, the cubic polynomial fitting is carried out according to the current anchor points and parameters to update the coefficients of the parametric cubic functions. In each inner loop, for each anchor point, its associated parameter will be updated via iteratively approaching the anchor from its adjacent points. When the curve fittings finish, by resampling new

**Algorithm 1:** Cubic Parametric Path Smoother.

**Input:** Discrete A* path, $\mathbb{P}$
**Output:** The smoothed path, $\mathbb{P}_s$

Assume cubic parametric functions: $x(u) = f_1u + f_2u^2 + f_3u^3, y(u) = f_4u + f_5u^2 + f_6u^3$, where $f_1, \ldots, f_6$ are coefficients and $u$ is the parameter.
Sampling $n$ anchors $\{K_j\}_{j=1}^n$ from $\mathbb{P}$, each anchor has its associated parameter $U_j$.
Set initial value of $U_j$ as $j/n$.
Set the shrinking ratio $\hat{\rho}$.
Denote outer iterated (allowable) times by $N_o$ ($\hat{N_o}$), and inner iterated (allowable) times by $N_i$ ($\hat{N_i}$).
$N_o = 0,\ N_i = 0$
**while** $N_o < \hat{N_o}$ **do**
  Run polynomial fitting with current $\{K_j\}_{j=1}^n$ and $\{U_j\}_{j=1}^n$.
  Fix $U_1$ to 0 and $U_n$ to 1.
  **for all** $i \in (2, \ldots, n-1)$ **do**
    $a = U_{i-1}, b = U_{i+1}$
    $c = b - (b-a) * \hat{\rho}, d = a + (b-a) * \hat{\rho}$
    **while** $N_i < \hat{N_i}$ **do**
      **if**{/*$loss$ returns the difference between the fitted value and the original one.*/}$loss(c, K_i) < loss(d, K_i)$ **then**
        $b = d$
      **else then**
        $a = c$
      **end if**
      $c = b - (b-a) * \hat{\rho}, d = a + (b-a) * \hat{\rho}, N_i \leftarrow N_i + 1$
    **end while**
    $U_i = (b + a)/2$
  **end for**
  $N_o \leftarrow N_o + 1$
**end while**
Resampling $\mathbb{P}_s$ with new parameters in (0,1) from the fitted cubic parametric functions.
**return** $\mathbb{P}_s$

**Algorithm 2:** The Multi-Segment-Supported Path Tracker.

**Input:** path to track $\mathbb{P}$, pose $p_i$, steering angle $\alpha_i$, velocity $v_i$, max velocity $\tilde{v}$, max steering angle $\tilde{\alpha}$
**Output:** steering angle $\alpha_o$ and velocity $v_o$ to control

Set the default look-ahead distance $\tilde{l}$
Find the turning pose $p_t$ in $\mathbb{P}$
$l \leftarrow \tilde{l}$
**if** $\exists p_t$ **then**
  $l \leftarrow min(||p_tp_i||, \tilde{l})$
**end if**
Find the goal pose $p_g$ (the pose $G_f$ or $G_b$ as illustrated in Fig. 2) in $\mathbb{P}$ satisfying $||p_ip_g|| \geq l$
**if** $\nexists p_g$**then**
  $p_g \leftarrow \mathbb{P}_{-1}$, where $\mathbb{P}_{-1}$ is the last pose in $\mathbb{P}$
**end if**
Compute the potential steering $\hat{\alpha_o}$ by Eq. 6 or Eq. 7
$\alpha_o \leftarrow min(\hat{\alpha_o}, \tilde{\alpha})$
Set a constant operation time interval $\tilde{t}$
$\tilde{v}_{max} \leftarrow \tilde{v}$
$\tilde{v}_{min} \leftarrow \tilde{v} \div 3$
$\hat{v} \leftarrow \tilde{v}_{max} + (|\alpha_o| \div \tilde{\alpha})(\tilde{v}_{min} - \tilde{v}_{max})$
Denote the heading difference between $p_i$ and $p_g$ by $\beta$
**if** $\beta \leq \frac{\pi}{2}\ and\ \beta \geq -\frac{\pi}{2}$ **then**
  $v_o \leftarrow \frac{\hat{v}+v_i}{2}$
**else then**
  $v_o \leftarrow \frac{-\hat{v}+v_i}{2}$
**end if**
**return** $\alpha_o, v_o$

parameters, a smoothed path can be obtained and the direction of each point on the path can be determined analytically.

## IV. MssPT: A Multi-Segment-Supported Path Tracker

As discussed in Section II, the geometry-based tracker is a better choice in low speed driving. However, for multi-segment path (such as Reeds-Shepp curves) tracking, there are two problems to be solved. One is that the strategy for controlling the steering angle in backward tracking needs to be determined, and the other is that the associated adaptive speed adjustment mechanism needs to be designed to achieve the purpose of smooth switching. To this end, a multi-segment-supported path tracker MssPT is proposed in this part. The pseudocodes of MssPT are given in Algorithm 2.

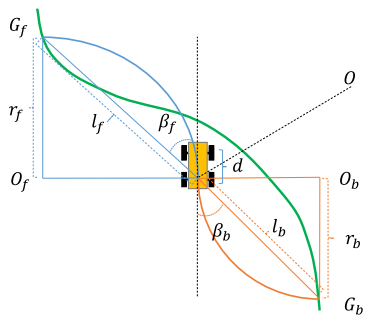


Fig. 2. The geometry of the local path tracking.

### A. Backward Compatible Steering Angle Control

The geometric relationship of the robot motion is illustrated in Fig. 2. In this figure, the robot's current position (the center point of the rear axle) is denoted by $O$ and $d$ denotes the wheelbase of the carlike robot. The geometric model of forward motion is marked by blue, while the backward motion model is drawn in orange. $G_f$ and $G_b$ represent the forward and backward goal points, respectively. $O_f$ ($O_b$) refers to the rotation center of the forward (backward) motion arc and $r_f$ ($r_b$) is the corresponding radius. $\beta_f$ ($\beta_b$) stands for the angle between the forward (backward) goal and the robot's heading direction.

According to the geometric relationships, we can get,

$$\frac{l_b}{sin(2 * \beta_b)} = \frac{r_b}{sin(\frac{\pi}{2} - \beta_b)}. \tag{3}$$

Assume that,

$$k = 2 * \frac{sin(\beta_b)}{l_b}. \tag{4}$$

According to the kinematic model of carlike robots [17], the steering angle for backward tracking can be computed by,

$$\alpha = tan^{-1}(k * d). \tag{5}$$

So, the steering angle can be expressed as,

$$\alpha = tan^{-1}\left(\frac{2 * d * sin(\beta_b)}{l_b}\right). \tag{6}$$

Similarly, the steering angle during forward tracking can be obtained by,

$$\alpha = tan^{-1}\left(\frac{2 * d * sin(\beta_f)}{l_f}\right). \tag{7}$$

So far, we find that the backward steering control formula based on kinematic geometry is consistent with the forward angle control formula in the Pure Pursuit algorithm [14].

### B. Dynamic Steering Angle and Speed Adjustment

To dynamically adjust the steering angle and the speed, the following preview strategy is proposed. We first set a fixed look-ahead distance $l$ as Pure Pursuit [14] does, calculate the distance between the current pose and the potential goal along the path to be tracked, and take the first pose where the distance reaches $l$ as the goal. And then, when there are multiple switching poses, if the distance to the nearest switching pose is less than $l$, this nearest pose will be selected as the goal instead.

Once the goal is determined, the steering angle $\alpha_o$ can be calculated by Eq. 6 or Eq. 7. As for the linear speed, a two-segment linear function (lines 13 to 22 of Algorithm 2) is utilized to dynamically adjust it. Specifically, in the first linear function, to make the rotation velocity of the carlike robot smoothly adjusted, we linearly increase the linear velocity according to the decrease of $\alpha_o$. In the second linear function, to increase the smoothness of robot movement, the speed obtained from the first linear equation is averaged with the current actual speed.

## V. System Design of Our Sweeping Robot

To examine the specially designed path planner and path tracker for CNCR, a consolidated autonomous sweeping robot is constructed. In this part, we introduce it in detail from aspects of how its hardware is configured and how to make the functional modules work organically.

### A. Hardware Configuration

As shown in the left part of Fig. 1, our hardware system is composed of two parts, sensors and processors. The perception data for mapping and localization of our system come from a Robosense RS-16 LiDAR mounted on the roof of the robot. This multi-line sensor has a horizontal FOV of $360^\circ$ and a vertical FOV of $30^\circ$, which can be fed into the localization or mapping module and has the ability of 3D obstacle perception. Due to the limitation of its narrow vertical FOV, there is still a blind spot around the robot about $3m$. Hence, a depth camera is mounted in front of the robot for the avoidance of suspended obstacles, and a single-line LiDAR is equipped in front of the chassis for lower obstacle avoidance. In addition to the perception of the environment, the robot needs to measure its motion status to support safe planning and control. Such information is provided by a sensor suite consisting of the robot's wheel odometer and a YESENSE-YIS10 IMU fixed on the robot's rear axle. Note that although the maximum speed response of the chassis is 1 $m/s$, we set the maximum velocity allowed to 0.5 $m/s$ for the sake of safety.

Our processors consist of two modules. One is the main processor with an i7-8750H CPU which is responsible for processing the computing tasks of all high-level navigation-related modules. The other is the low-level vehicle controller E-ControlSys, whose duty is to connect the high-level navigation system with the low-level vehicle chassis.

### B. Functional Modules

To complete a CNCR task with a maximum coverage rate under a minimum repetition rate, a coverage path needs to be planned in advance to track online. So our CNCR pipeline is carried out in two stages as illustrated in Fig. 1.

At the offline stage, the robot collects data in the operation area by manually controlling and a priori map is built by the mapping module. Based on the map, the coverage planning module generates a global reference path with an optimal coverage rate according to the robot parameters.

At the online stage, the local path planning module obtains a local reference path from the offline planned coverage path in real-time according to the pose provided by the local localization module and fuses the static and dynamic obstacles from sensors' data. Afterward, the module is activated to plan a safe and feasible local path. Subsequently, the path tracking module calculates the best control parameters according to the planned local path and the robot's motion status to move the robot. The global localization module is utilized in system initialization and online relocation. Once the system is initialized, the CNCR task can be completed by repeating the above-mentioned processes of localization, local path planning, and local path tracking.

We adopt the method proposed in [2] to build priori maps and to localize online. As for the coverage reference path generation, we first preprocess the map to remove the scattered and irregular map areas and manually mark the polygon area that needs to perform the coverage task. Then, the polygon is partitioned into several convex sub-polygons by Greene's polygon decomposition approach [35]. For each sub-polygon, a contour pattern is adopted to plan the coverage path. The switching paths between sub-polygons are connected by smoothed A* paths [18]. The path smoother adopted here is the same as the one introduced in Section III-C. To reduce the coupling between modules, all modules should run relatively independently. For this reason, we resort to the communication function of ROS (Robot Operating System) [36] to transmit messages among the modules.

TABLE I
ROBOT PARAMETERS AND EXPERIMENTAL SETTINGS

| $d$ | $\rho$ | $w$ | $l$ | $\tilde{\tau}$ | $\tilde{v}$ | $\tilde{\alpha}$ | $\tilde{l}$ |
|---|---|---|---|---|---|---|---|
| 0.68m | 0.70m | 0.74m | 1.28m | 0.5m | 0.7m/s | 60° | 0.5m |

$d$: wheelbase, $\rho$: minimum turning radius, $w$: width, $l$: length
$\tilde{\tau}$: allowable deviation bound, $\tilde{v}$: max linear velocity, $\tilde{\alpha}$: max steering angle, $\tilde{l}$: default look-ahead distance

TABLE II
AVERAGE OFFSETS OF PLANNED PATHS WITH/WITHOUT THE REFERENCE LINE CONSTRAINT (m)

| | AD $\in [0, \frac{\pi}{3})$ | AD $\in [\frac{\pi}{3}, \frac{2\pi}{3})$ | AD $\in [\frac{2\pi}{3}, \pi)$ |
|---|---|---|---|
| TEB | 0.12 | 0.13 | 0.16 |
| $\text{TEB}_{\text{CNCR}}$ | **0.11** | **0.10** | **0.11** |

AD: heading angle difference between the reference line's starting-pose and ending-pose

TABLE III
SUCCESS RATES OF TEB AND $\text{TEB}_{\text{CNCR}}$

| Obstacle Distribution | TEB | $\text{TEB}_{\text{CNCR}}$ |
|---|---|---|
| 0.5m | 97.30% | **99.97%** |
| 1.0m | 90.10% | **99.60%** |
| 1.5m | 80.17% | **98.10%** |

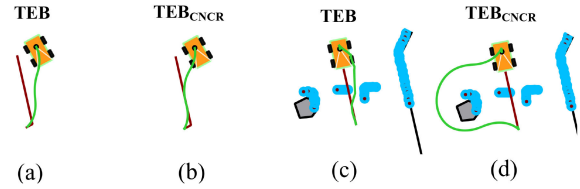


Fig. 3. Planning comparisons between TEB and $\text{TEB}_{\text{CNCR}}$. a/b: without/with $\mathbf{l}_k(\cdot)$. c/d: without/with the heuristic initialization. red line: the reference line. green curve: the planned path. blue dots: the perceived obstacles.

## VI. EXPERIMENTS

In this part, we will first test the performance of the proposed $\text{TEB}_{\text{CNCR}}$ and MssPT separately in the simulation environment, and then carry out complete coverage experiments in real-world scenes by making use of the established sweeping robot "Raccoon". The robot parameters and our planner's / tracker's parameters set in the experiments are listed in Table I. Note that these parameters were kept the same both for simulation and real-world experiments.

### A. Performance Investigation of $TEB_{CNCR}$

To verify the significance of the introduced reference line constraint, given the same reference line, two different feasible paths were planned by TEB and $\text{TEB}_{\text{CNCR}}$. In each planning, a 3 $m$ long reference line was randomly sampled from the coverage path planned according to the contour pattern as introduced in Section V-B. The sampled reference lines were classified into three categories ($[0, \frac{\pi}{3}), [\frac{\pi}{3}, \frac{2\pi}{3}), [\frac{2\pi}{3}, \pi)$) according to the heading angle differences between the starting points and the endpoints. Then, for each category, the average deviation of the planned paths off the reference lines was counted and listed in Table II.

To examine the influence of the presented initialization strategy, the planning success rates of TEB and $\text{TEB}_{\text{CNCR}}$ in complex environments were compared (when the planner returns a feasible path that conforms to all the constraints, it is regarded as a successful planning). The reference lines were still randomly sampled from the coverage path. To simulate complex environments, we first conducted a uniformed sampling longitudinally along the reference line to identify a reference point $p_r$ which was 0.5 $m$ to 1.5 $m$ far away from the starting point. After that, we conducted a uniformed sampling again along the latitudinal direction of $p_r$ to assign an obstacle whose distance to $p_r$ was less than $t_d, t_d \in \{0.5\ m, 1.0\ m, 1.5m\}$. Afterwards, 3,000 planning tests were conducted with TEB and $\text{TEB}_{\text{CNCR}}$ respectively. Meanwhile, their success rates, the ratios of the successful plannings to the total trials, were recorded in Table III.

It should be noted that in the investigation of the initialization strategy, $\text{TEB}_{\text{CNCR}}$ automatically turned off the reference line constraint $\mathbf{l}_k(\cdot)$ for the priority of obstacle avoidance. Besides, the initialization strategy was not enabled in the investigation of $\mathbf{l}_k(\cdot)$ since $\mathbf{l}_k(\cdot)$ is independent of the initialization strategy.

*1) Deviation Reduction With the Reference Line Constraint:* Fig. 3(a) and Fig. 3(b) illustrate the results of the reference line constraint $\mathbf{l}_k(\cdot)$ for the path planners. It can be observed that when the constraint is activated, the path planned by $\text{TEB}_{\text{CNCR}}$ tends to fit the given reference line more closely. But inevitably, $\text{TEB}_{\text{CNCR}}$ behaves more radical, which makes the planned path lose a certain degree of smoothness. In practice, a trade-off can be obtained by adjusting the weight of this constraint.

The quantitative results in Table II demonstrate that the planned paths from $\text{TEB}_{\text{CNCR}}$ are closer to the given reference lines than that obtained by TEB under all types of reference lines. When the angle difference of the reference line is as large as $[\frac{2\pi}{3}, \pi)$, $\text{TEB}_{\text{CNCR}}$ can reduce the average deviation of a single point by 5 $cm$ compared with TEB.

*2) Robustness Improvement With the Heuristic Initialization:* Fig. 3(c) presents that when the robot gets into a narrow space, the original TEB falls into local optimum, and the planned path does not conform to the law of the vehicle's motion constraints. But after a heuristic initialization with a smoothed A* path, $\text{TEB}_{\text{CNCR}}$ successfully finds the global optimal solution as presented in Fig. 3(d).

It can be seen from Table III that the success rate of TEB will decrease significantly with the increase of the complexity of the surrounding environment. When obstacles are distributed within 1.5 $m$ of the reference line, the success rate drops to only 80.17%. By contrast, via incorporating the initialization strategy, the success rate of $\text{TEB}_{\text{CNCR}}$ reaches 98.10% despite in the complex environment.

### B. Functioning and Accuracy Analysis of MssPT

*1) Comparison With Typical Geometric Path Trackers:* According to the conclusions of Snider's work [9], the geometric path trackers can bring about more stable tracking with higher accuracy than the kinematic-model based and dynamic-model based approaches in low speed driving ($\leqq 10\ m/s$). Compared with the two typical geometric path trackers (Pure Pursuit [14]

TABLE IV
ACCURACY AND PROPERTIES OF GEOMETRIC TRACKERS

| | Pure Pursuit | Stanley Tracker | **MssPT** |
|---|---|---|---|
| Straight line (m) | 0.04 | 0.13 | **0.04** |
| Circle curve (m) | 0.07 | 0.12 | **0.06** |
| Eight-shaped curve (m) | 0.07 | failed | **0.06** |
| Multi-segment | ✗ | ✗ | ✓ |
| Auto-speed | ✗ | ✓ | ✓ |

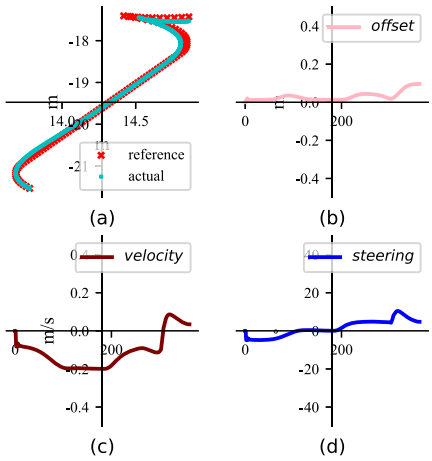


Fig. 4. MssPT's tracking status along a Reeds-Shepp path. (a) The Reeds-Shepp reference line (red) and the actual trajectory (cyan). (b) Offset off the reference line. (c) Robot's linear velocity profile. (d) Robot's steering angle profile.

and Stanley tracker [15]), our MssPT has obviously better characteristics as shown in Table IV, not only supporting multi-segment path tracking but also having the ability of adaptive speed adjustment.

In addition, in order to compare the accuracy of the three trackers, we chose three types of forward paths for tracking experiments, which are straight lines, circle curves, and eight-shaped curves. For each path to be tracked, the offset between its initial point and the robot was set as 0.5 $m$, and the heading angle deviation was set as $30^{\circ}$. The average deviation values of the examined trackers are summarized in Table IV. It can be seen that both MssPT and Pure Pursuit achieve centimeter-level accuracy. In contrast, the accuracy of the Stanley tracker is twice as low, not to mention that it does not converge for the eight-shaped path.

*2) Multi-Segment Path Tacking With MssPT:* To verify the tracking ability of the proposed MssPT for multi-segment curves with forward and backward motion at the same time, typical complex Reeds-Shepp-type paths were chosen in our experiments. Reeds-Shepp paths usually contain two or three curves with different moving directions [13]. If the tracking scheme can control the robot to track such curves accurately, it implies that the tracker has the ability to track complex multi-segment paths. Due to the space limitation, we select one Reeds-Shepp curve as a representative example here. We resorted to OMPL (Open Motion Planning Library) [37] for Reeds-Shepp paths generation and the Stage simulator [38] was utilized to simulate the robot.

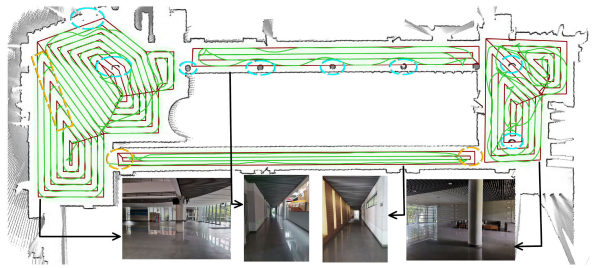

Fig. 5. Coverage navigation in a real-world scene by "Raccoon". The pre-planned coverage paths are drawn in dark-red lines. The actual traveling paths are presented by green curves. Covered areas are highlighted in light green. Dashed blue ellipses correspond to pillars in the real world. Areas enclosed by dashed orange boxes and ellipses are where large orientation differences of reference lines appear.

As shown in Fig. 4, the tracking results of a Reeds-Shepp curve exhibit that the proposed MssPT achieves high tracking accuracy and stability. The position deviations of most of them were less than 0.05 $m$. From the velocity curve, it can be seen that the robot accelerated and retreated steadily. Afterwards, when it reached the switching point of the path, it began to move forward. Thanks to the dynamically adjusted preview distance and the speed generation strategy, the amplitude of the steering angle changed smoothly and there was only one sharp adjustment near the endpoint.

### C. Real-World Experiments With Our Robot "raccoon"

*1) Real-World Performance:* For a given coverage path, the robot must visit all the nodes on this path in order and constantly query the local reference line at its present location. After that, a safe and feasible path is planned and tracked. When all nodes on the coverage path are visited by the robot, the coverage task is completed successfully.

Due to the influence of the localization accuracy, motion constraints and irregular obstacles, the actual trajectory of the robot will deviate from the ideal coverage path to a certain extent. Therefore, in CNCR, the actual coverage rate is the index that can truly reflect the degree of task completion. To evaluate the actual performance of our CNCR system "Raccoon," we conducted real-world experiments in a campus building of Tongji University, with a free area of about 663 $m^2$. The area was partitioned into four sub-areas, including two multiplex lobbies and two narrow-long corridors, named "Lobby-0," "Lobby-1," "Corridor-0" and "Corridor-1". All running data was recorded while performing coverage tasks by "Raccoon" in these sub-areas.

Experimental results show that the actual average coverage rate of "Raccoon" reaches 92.21% in the real-world scenes, which fully corroborates its practicability.

*2) Corner Case Discussion:* Although our Raccoon has achieved pleasing results in real scenes, there are still some corner cases to be considered. According to the actual trajectories shown in Fig. 5, the factors affecting the actual coverage rates in different regions can be summarized into the following two categories. One is the large heading angle difference of the given local reference line (the areas enclosed by the orange dotted box

and ellipses). Owing to the influence of the robot's minimum turning radius, there is a certain coverage blind area inside the turning. The other is the priority to avoid obstacles (shown by the blue dotted ellipses, corresponding to the circular pillars in the actual scene). For this case, $\mathrm{TEB}_{\mathrm{CNCR}}$ prefers to avoid obstacles safely and no longer aims to fit the reference line. Therefore, uncovered areas should be post-processed according to the corresponding situations.

## VII. Conclusion

To achieve robust CNCR, this letter carefully designed the path planning and path tracking modules involved in it. Inspired by TEB, a reference line constraint and a heuristic initialization strategy together with a cubic parametric path smoother were introduced, resulting in $\mathrm{TEB}_{\mathrm{CNCR}}$. Compared with TEB, $\mathrm{TEB}_{\mathrm{CNCR}}$ reduces the average offset between planned paths and reference lines by 5 $cm$ and achieves a success rate of 98.10% even in complex scenes. To track the planned multi-segment paths, a backward control mechanism was derived from the robot's kinematic geometry and the associated speed generation scheme was designed, yielding MssPT. Tracking tests with Reeds-Shepp paths verified its high tracking accuracy for multi-segment paths. To further corroborate the effectiveness of the specially designed $\mathrm{TEB}_{\mathrm{CNCR}}$ and MssPT, an integrated sweeping robot system was built. The experimental results show that the system achieved an average actual coverage rate of 92.21% in real-world complex environments, which fully verifies the practicability and effectiveness of the designed modules. We will devote our efforts to extend our framework to adapt to scenes with highly dynamic obstacles in future.

## References


[1] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.

[2] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LiDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Stockholm, Sweden, 2016, pp. 1271–1278.

[3] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[4] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object detection and tracking, based on DNN, for autonomous vehicles: A review," *IEEE Sensors J.*, vol. 21, no. 5, pp. 5668–5677, Mar. 2021.

[5] X. Zhao, P. Sun, Z. Xu, H. Min, and H. Yu, "Fusion of 3D LIDAR and camera data for object detection in autonomous vehicle applications," *IEEE Sensors J.*, vol. 20, no. 9, pp. 4901–4913, May 2020.

[6] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

[7] K. Tiwari, X. Xiao, A. Malik, and N. Y. Chong, "A unified framework for operational range estimation of mobile robots operating on a single discharge to avoid complete immobilization," *Mechatronics*, vol. 57, pp. 173–187, 2019.

[8] K. Tiwari, X. Xiao, and N. Y. Chong, "Estimating achievable range of ground robots operating on single battery discharge for operational efficacy amelioration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Madrid, Spain, 2018, pp. 3991–3998.

[9] J. M. Snider, "Automatic steering methods for autonomous automobile path tracking," *Robotics Institute*, Carnegie Mellon University, Tech. Rep., 2009.

[10] C. Chien, C. J. Hsu, W. Wang, and H. Chiang, "Indirect visual simultaneous localization and mapping based on linear models," *IEEE Sensors J.*, vol. 20, no. 5, pp. 2738–2747, Mar. 2020.

[11] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, "Deep 3D object detection networks using LiDAR data: A review," *IEEE Sensors J.*, vol. 21, no. 2, pp. 1152–1171, Jan. 2021.

[12] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vancouver, BC, Canada, . 2017, pp. 5681–5686.

[13] 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, 1990.

[14] O. Amidi, "Integrated mobile robot control," Robotics Institute, Carnegie Mellon University, Tech. Rep., 1990.

[15] S. Thrun *et al.*, "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.

[16] M. Samuel, M. Hussein, and M. B. Mohamad, "A review of some pure-pursuit based path tracking techniques for control of autonomous vehicle," *Int. J. Comput. Appl.*, vol. 135, no. 1, pp. 35–38, 2016.

[17] S. F. Campbell, "Steering control of an autonomous ground vehicle with application to the DARPA urban challenge," Ph.D. dissertation, Massachusetts Institute of Technology, 2007.

[18] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci.*, vol. 96, no. 2, pp. 100–107, Jul. 1968.

[19] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[20] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Department of Computer Science, Iowa State University, Tech. Rep., 1998.

[21] S. M. LaValle and J. J. Kuffner Jr., "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[22] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, USA, 2000, pp. 995–1001.

[23] J. hwan Jeon *et al.*, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *Proc. Amer. Contr. Conf.*, Washington, DC, USA, 2013, pp. 188–193.

[24] A. Piazzi, C. L. Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic g2-splines for the iterative steering of vision-based autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 3, no. 1, pp. 27–36, Mar. 2002.

[25] J. Lee and B. Litkouhi, "A unified framework of the automated lane centering/changing control for motion smoothness adaptation," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, Anchorage, USA, . 2012, pp. 282–287.

[26] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, "Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles," in *Proc. IEEE Intell. Vehicle. Symp.*, Orlando, FL, USA, 2014, pp. 510–515.

[27] Z. Liang, G. Zheng, and J. Li, "Automatic parking path optimization based on bezier curve fitting," in *Proc. IEEE Int. Conf. Autom. Logis.*, Zhengzhou, China, 2012, pp. 583–587.

[28] L. B. Cremean *et al.*, "An information-rich autonomous vehicle for high-speed desert navigation," *J. Field Robot.*, vol. 23, no. 9, pp. 777–810, 2006.

[29] T. Gu and J. M. Dolan, "On-road motion planning for autonomous vehicles," in *Proc. Int. Conf.* Intell. Robot. Appl., Montreal, Canada, 2012, pp. 588–597.

[30] D. Kogan and R. Murray, "Optimization-based navigation for the DARPA grand challenge," in *Proc. Conf. Decis. Contr.*, San Diego, CA, USA, 2006, pp. 1–6.

[31] H. Khan, J. Iqbal, K. Baizid, and T. Zielinska, "Longitudinal and lateral slip control of autonomous wheeled mobile robot for trajectory tracking," *Front. Info. Technol. Elect. Eng.*, vol. 16, no. 2, pp. 166–172, 2015.

[32] R. S. Sharp, "Driver steering control and a new perspective on car handling qualities," *Proc. Inst. Mech. Eng. P. J. Sport. Eng. Technol.*, vol. 219, no. 10, pp. 1041–1051, 2005.

[33] R. Rajamani, *Vehicle Dynamics and Control*, *1st ed*. New York, USA: Springer, 2011.

[34] D. Sahu, P. Roy, and B. K. Das, "Real time implementation of cubic b-spline algorithm for electro optical tracking system," in *Proc. Int. Conf. Emerg. Trends Commu. Control, Signal Proc. Comput. Appl.,* Bangalore, India, 2013, pp. 1–5.

[35] B. Chazelle and D. Dobkin, "Decomposing a polygon into its convex parts," in *Proc. Ann. ACM Sympo. Theory Comput.*, New York, NY, USA, 1979, pp. 38–48.

[36] Stanford Artificial Intelligence Laboratory, "Robotic operating system." [Online]. Available: https://www.ros.org

[37] K. Lab, "Open motion planning library." [Online]. Available: https://ompl.kavrakilab.org

[38] R. Vaughan, "Massively multi-robot simulation in stage," *Swarm Intell.*, vol. 2, no. 2-4, pp. 189–208, 2008.