2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Perception-Aware Path Finding and Following of Snake Robot in Unknown Environment

Weixin Yang, Gang Wang, and Yantao Shen

Abstract— **In this paper, we investigate the perception-aware path finding, planning and following for a class of snake robots autonomously serpentining in an unmodeled and unknown environment. In the work, the onboard LiDAR sensor mounted on the head of the snake robot is utilized to reconstruct the local environment, by which and the modified rapidly-exploring random tree method, a feasible path from the current position of the robot to a local selected target position can be obtained. Next, the parametric cubic spline interpolation path-planning method and potential functions are applied to make the path more smooth so as to prevent the multi-link and elongated robot body from hitting obstacles. To steer, a time-varying line-of-sight control law is designed to ensure that the robot moves to the local target position along the generated path by the perception-aware method. The robot will repeatedly perform the above search-find-move strategy until it reaches the final predefined target point. Simulation and experimental results demonstrate a good performance of the proposed perception-aware approach, that is, the elongated and underactuated snake robot is capable of autonomously navigating in an unknown environment.**

I. Introduction

The inspiration for snake robots stems from natural snakes. Snakes demonstrate superior mobility capabilities and can move over virtually any kind of terrain, containing narrow and confined areas [1]. As the demand increases for snake robots in numerous applications that range from fire fighting, rescue and search, and the inspection of subsea oil and gas filling for vehicles, the need for high intelligence of snake robots is clearly apparent; therefore, an autonomous control method and its task-orientated path following strategy are expected. In order to enable snake robots to operate autonomously and interact with the environment with less constrains induced by its elongated and underactuated body, it needs to perceive information about the environment surrounded, which can be used in their planned actions accordingly.

In recent years, snake robots research has been focused on understanding the basic principles of snake movement and the development of locomotion control technology [2]. However, it is noted that most current control strategies require the desired trajectories or paths to be known *a priori*. Hence, exact knowledge of the global environment in which the robot moves are needed to generate these trajectories or paths. Such a requirement restricts the autonomy level of the developed strategies to a relatively small class of applications. For the more practical consideration, it is highly desirable to make the motion planner taking environment representation as input, i.e., employ the fusion of different sensor data and previously stored knowledge to build an environmental map. Specifically, the snake robot can find and plan the path in real-time, relying only on the local environmental information that can be perceived. Then the robot is able to use the tracking controller to adjust the possible deviation between the path and the actual environment.

This work aims to improve the autonomy level of snake robots, and to enhance their adaptability by providing a perception-aware path finding, planning and tracking framework to ensure snake robots reach the predefined target point in an unknown environment. Unlike the mapping and navigation strategies using the combination of laser radar, infrared sensors, and ultrasonic sensors, we only use an onboard LiDAR sensor on the robot head as the "eye" to perceive environmental information encountered by the robot during the motion. In detail, our framework consists of three implementation procedures. First, the LiDAR sensor is utilized to perceive environmental data around the robot. Then, considering the challenges faced by applying the simultaneous localization and mapping (SLAM) method to the snake robot navigation, we create the local costmap using the scanned LiDAR information with the hector mapping, by which and a modified rapidly-exploring random tree, a feasible but rough path from the current position of the robot to a local selected target position can be on-line generated. In addition, we take into account the underactuated and high-DoF characteristics of the multi-link snake robot and its elongated body constrains, and apply the parametric cubic spline interpolation path-planning method and potential functions to smooth the rough path to be a feasible one. Finally, a time-varying line-of-sight control law is designed to steer the snake robot towards and follow the generated feasible path to the local target position. The robot will repeat the above on-line searching, planning and control steps until it approaches the final predefined target point.

Note that, contrary to the existing control schemes designed based on exact knowledge of the environment [3], [4], [5] and in accordance with the autonomy and technology readiness assessment (ATRA) framework adapted from [6], our work trends towards increasing the autonomy level of snake robot to a self-decision making and evaluation level, which performs the path finding, planning and autonomous navigation based on locally searched environmental information. In addition, by considering underactuated nature of the snake robot and its multi-link elongated body configuration with many degrees of freedom (DoF) [7], the highly efficient

W. Yang, G. Wang, and Y. Shen is with the Department of Electrical & Biomedical Engineering, University of Nevada, Reno, NV 89557, USA ytshen@unr.edu

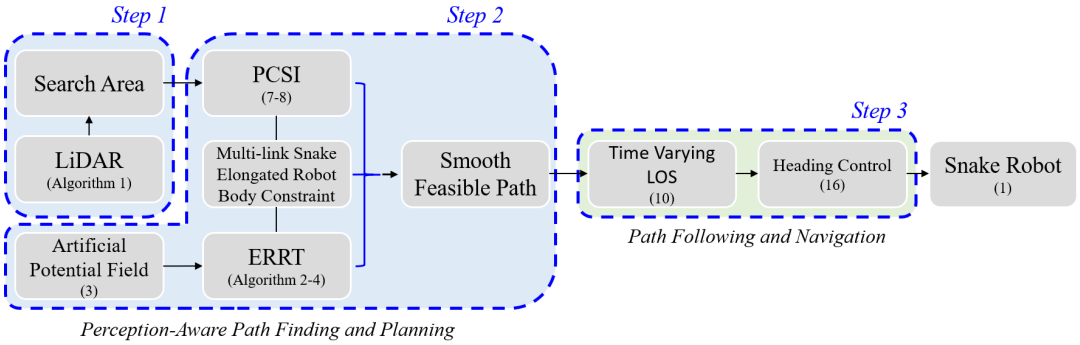


Fig. 1. Architecture of the proposed perception-aware path following scheme.

ERRT-PCSI path planner is well leveraged with the real-time LiDAR scanning based hector mapping to efficiently generate the smooth collision avoidance path for the snake robot serpentining in an unknown environment. Compared with SLAM based methods [8], the proposed perception-aware method has the advantage of less-computing yet high-efficiency for the class of snake robots.

II. Problem Statement and Perception-Aware Path Following

The kinematic model of snake robot can be of the following form [3], [4]

$$\begin{aligned} \dot{\theta} &= v_\theta, \dot{p}_x = v_t \cos\theta - v_n \sin\theta, \\ \dot{p}_y &= v_t \sin\theta + v_n \cos\theta, \dot{v}_\theta = -f_1 v_\theta + \frac{f_2}{N-1} v_t \bar{e}^T \phi \end{aligned} \tag{1}$$

where $(p_x, p_y) \in R^2$ represents the center of mass of the snake robot. N is the number of links, and we specify the $N = 8$ in the experiment. The ith link angle, $i = 1, \ldots, N$, of the snake robot is denoted by $\theta_i \in R$. The joint angle $\phi_i, i = 1, \ldots, N-1$, is given by $\phi_i = \theta_{i+1} - \theta_i$. The joint vector is described as $\phi = [\phi_1, \ldots, \phi_{N-1}] \in R^{N-1}$. The heading angle is calculated as the average of all joint angles $\theta = \frac{1}{N}\sum_{i=1}^{N} \theta_i \in R$. $v_\theta \in R$ is the angular velocity. $v_t \in R$ and $v_n \in R$ are, respectively, the tangential and normal direction velocity of the robot. $\bar{e} = [1, \ldots, 1]^T \in R^{N-1}$, and f_1 and f_2 are positive constant friction parameters.

The primary control objective is to force the snake robot to reach the target point from the starting position. The main obstacle lies in two aspects. The first is that the robot possesses many degrees of underactuation and has highly coupled nonlinear dynamics. The second is that the motion environment is unmodeled and lacks global information for planning paths. Thus, before we move the robot, LiDAR scanning algorithm will be introduced to reconstruct the unstructured local environment. The overall perception-aware path following architecture is shown in Fig. 1. Step 1 uses LiDAR and artificial potential field to reconstruct the local environment. Step 2 applies ERRT and PCSI to generate a smoothly feasible path from the current point to a local selected target position based on locally searched information. Steps 1 and 2 are for path finding and planning. Step 3 refers to the time-varying LOS based controller to steer the snake robot toward and subsequently along the generated path.

III. Perception-Aware Path Finding and Planning

A. LiDAR Scanning Based Hector Mapping

The LiDAR carried by the snake robot is a 360-degree scanning sensor that receives data within a range of 12 meters around the robot. In order to make the robot move more efficiently toward the given target point, we introduce a reward function. The function is adopted to calculate the reward value of each point explored by the LiDAR, and the point p_h (not necessarily only one) with the highest reward value can be identified. Then, the number of p_h can be obtained in each quadrant, and the quadrant with the largest number of p_h is where the snake robot is going. The LiDAR scanning algorithm rules out 3 impossible quadrant and guides the robot to the direction most likely to reach the target which places a base for the following path planning algorithms. The detailed algorithm is given below.

Algorithm 1 LiDAR Scanning Algorithm.

1: $LiDAR\_Scan(Position, Angle)$
2: **for** $j = 1, j < 5, i++$ **do**
3: $dead\_end(\text{j}) \leftarrow is\_quadrant\_dead\_end()$
4: **end for**
5: **if** $!dead\_end()$ **then**
6: **for** $i = 1, i < 360, i++$ **do**
7: $connected\_flag(\text{i}) \leftarrow Connected(Position, Target)$
8: $angle\_weight(\text{i}) \leftarrow Angle\_weight(Angle, Target)$
9: $distance\_weight(\text{i}) \leftarrow Distance\_weight(Position, Target)$
10: $reward\_value(\text{i}) \leftarrow dead\_end(\text{i}) \times (a_r \times angle\_weight(\text{i}) + b_r \times connected\_flag(\text{i})c_r \times distance\_weight(\text{i}))$
11: **return** $reward\_value$
12: **end for**
13: **end if**

In the above algorithm, line 1 is to use LiDAR scanning to

get all points within the range of 12 meters and 360-degree information. A full rotation of the LiDAR is divided into four quadrants, and line 3 judges which of these cannot be passed (each quadrant contains 90 points, and it is unreachable if the graph that consists of these 90 points has a closed boundary). Line 7 determines whether each point obtained by the LiDAR can directly connect the final predefined target point, i.e., there is no obstacle between the scanning point and the target within the LiDAR range. Line 8 is the angle information. The scanning point needs to be in the direction of the vector formed by the robot and the target point. For simplicity, we only consider the situation when the snake robot moves. For example, if the target point is in the 30-degree direction of the robot, the point LiDAR_Position in the -90 to 90-degree direction of the robot will get more reward. Line 9 represents the distance information. The distance between LiDAR_Position and the target is added as a weight to the reward function. Line 10 is the addition of the total weight. The parameters a_r, b_r, and c_r denote the angle information, the connectivity and the distance information, respectively. In practice, we can obtain different forward strategies by choosing different weights. Finally, line 11 returns the reward value for path planning.

B. The Executive Rapidly-Exploring Random Tree

To pursue the path planning for snake robots, we modify the basic RRT [9] by introducing a waypoint cache, and the path cost, as the executive rapidly-exploring random tree (ERRT).

As presented in Algorithm 2, ERRT can adequately address the path planning problems for the snake robot motion with the small computational load. In the processing of the ERRT, all the states will be updated in the cache whenever the path is found as described in Algorithm 2, from line 10 to line 19. Furthermore, we take the path cost into account such that to allow each new node can be selected more closely to the target waypoint. By including the path cost into the calculation, the algorithm can preferentially push the lower-cost paths to the target. Then, we introduce an additional measure of Υ to estimate the path cost from the potential nodes to the target node, which is computed as

$$\Upsilon = 1 - \frac{\lambda_{vertex} - \lambda_{opt}}{\lambda_{max} - \lambda_{opt}}. \tag{2}$$

Here λ_{vertex} denotes the sum of the integrated cost along the path and the estimated path cost from this node heading toward to the target node, λ_{opt} is the estimated cost of the optimal path from the initial point to the target point, and λ_{max} represents the maximum path cost through all possible nodes. The value of the measurement Υ is in direct proportion to the quality of the path, in comparison to the optimal path. The algorithm structure is described in Algorithm 2, from line 20 to line 25.

The ERRT improves planning efficiency and the quality of the generated path. Compared with basic RRT as described in Table I, which indicates that the computing efficiency of ERRT is around three times higher than that of the basic RRT. The comparison of basic RRT and our ERRT is shown in Fig. 2.

Algorithm 2 The ERRT Algorithm.

Require: The initial point X_{init}, and the target point X_{target}

Ensure: The planned path $finalpath$

1: $T_{init}(X_{init})$
2: **for** $k = 1$ **to** n **do**
3: $p = randomin[0.0 \cdots 1.0]$
4: $i = randomin[0 \cdots NumOfWayPoint - 1]$
5: $X_{rand} \leftarrow Random\_State(obstacles)$
6: $X_{near} \leftarrow Nearst\_Neighbor(X_{rand}, T)$
7: **if** $collision$ **then**
8: Back to 5
9: **end if**
10: **if** $0 < p < P_{target}$ **then**
11: $X_{rand} \leftarrow target$
12: **end if**
13: **if** $P_{target} < p < P_{target} + P_{WayPoint}$ **then**
14: $X_{rand} \leftarrow WayPointCache[i]$
15: **end if**
16: **if** $P_{target} + P_{WayPoint} < p < 1$ **then**
17: $X_{rand} \leftarrow RandomNode()$
18: **end if**
19: $X_{new} \leftarrow X_{rand}$
20: $T_{add\_vertex}(X_{new})$
21: $X_{add\_edge}(X_{near}, X_{new})$
22: $Calculate\_Path\_Cost(X_{new})$
23: **if** $arrivetarget(T, X_{target})$ **then**
24: $finalpath \leftarrow export(T)$
25: **end if**
26: **end for**

TABLE I
PERFORMANCE COMPARSION OF RRT AND OUR ERRT

Method	Total tree nodes	Total path cost	Computation time(s)
RRT	943	117.87	4.32
ERRT	334	41.75	1.53

Besides, we employ an artificial potential function to prevent the spanning tree from reaching obstacles. The function takes the following form [10]

$$U = \frac{1}{2}\mu\left(\frac{1}{\rho(p_c, p_{obs})} - \frac{1}{\rho_0}\right) \tag{3}$$

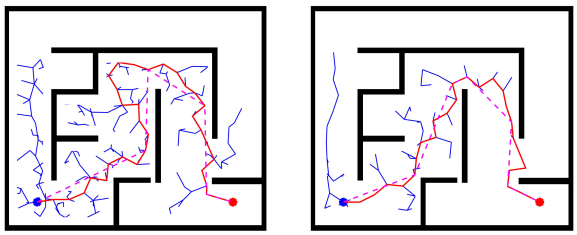
where μ is a positive scaling factor, $\rho(p, p_{obs})$ is the minimal distance between the robot and the obstacle, p is the position of the robot, p_{obs} denotes the point on the obstacle so that the distance between this point and the robot is minimal, and ρ_0 represents the width of the robot. For further improving the ERRT's feasibility, we redesign ERRT search step size l_s such that l_s dynamically changes according to the exploration complexity. l_s is proposed as

$$l_s = l_{\min} + \frac{k(l_{\max} - l_{\min})}{1 + e^{-sp_e}}U \tag{4}$$

where $l_{\min}$ and $l_{\max}$ are, respectively, the shortest step and the longest step, $k \in (0,1)$ and $s \in (0,1)$ are constants that can adjust the speed of exploration, p_e is the position error. Each sample point $\bar{p} = [x_s, y_s]^T$ is generated with respect to reference position $p_d = [x_d, y_d]^T$ and heading θ_d by

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} r\cos\beta \\ r\sin\beta \end{bmatrix} \tag{5}$$

with $r = \delta_r|n_r| + r_0$ and $\beta = \delta_\theta|n_\theta| + \theta_d$, where n_r and n_θ are random numbers obeying Gaussian distributions, δ_r and δ_θ are the standard deviation in the radial direction and the standard deviation in the radial direction, respectively. The forward direction has been obtained through the LiDAR scanning algorithm. We then use the Gaussian distribution sample scheme to get sampling points for the ERRT so that the path planning algorithm can reach the target point quickly. The visualization of the Gaussian distribution sampling strategy is demonstrated in Fig.4, as the yellow dots.



(a) (b)

Fig. 2. Tree expansions comparison for RRT and our ERRT illustrating path optimality. Path from the blue initial point to the red target point is shown in red. (a) Solution from basic RRT, (b) solution from our ERRT.

Note that due to the exotic property of the RRT path, the generated path by ERRT is still not applicable for snake robot locomotion. The PCSI method is then proposed to smooth the generated path for the following controller design as presented in the next section.

C. Parametric Cubic Spline Interpolation Based Curve Path Smoothing

The snake robot is able to search a feasible path from the initial point to the target point through the proposed ERRT. However, the path is non-smooth and multi-bended. PCSI is then employed to smooth the path consisted of the waypoints picked from the waypoints cache. The path is defined by the coordinates $p_{wpt}(i) = (x_{wpt}(i), y_{wpt}(i)) \in R^2$. For path generation, we use PCSI to produce a path for each pair of successive waypoints in the form of $(p_{wpt}(i), p_{wpt}(i+1)), i = 1, \ldots, n-1$, which is called a spline. A generated path is a sequence of order splines. This requires the introduction of an independent variable s with the definition $\dot{s} = f(s,t)$. The smooth path f_d between two waypoints $(x_{wpt(i)}, y_{wpt}(i))$ and $(x_{wpt(i+1)}, y_{wpt}(i+1))$, is described as $f_d(s) = (x_d(s), y_d(s))$, where $(x_d(s), y_d(s))$ is the position of the smooth path

$$x_d(s) = a_3(s-s_i)^3 + a_2(s-s_i)^2 + a_1(s-s_i) + a_0 \tag{6}$$

$$y_d(s) = b_3(s-s_i)^3 + b_2(s-s_i)^2 + b_1(s-s_i) + b_0 \tag{7}$$

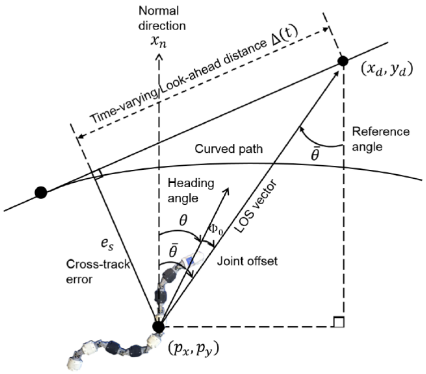


Fig. 3. Time-varying LOS guidance law.

Our mission becomes to find the root of the polynomials (6) and (7). The constant parameters $a_0, a_1, a_2, a_3, b_0, b_1, b_2$, and b_3 have to be determined to construct the cubic polynomial of each spline.

To satisfy the two necessary constraints presented in our work [5] and the other two boundary constraints, the values of the constant parameters a_0, a_1, a_2, a_3 can be calculated. Furthermore, we can obtain the constant parameters b_0, b_1, b_2, and b_3 for the cubic polynomial 7 with the same manner by simply replacing x with y.

IV. Time-Varying LOS Based Steering Controller Design

Based on the ERRT-PCSI planned path, a time-varying LOS guidance law is introduced to steer the snake robot as shown in Fig.3. Inspired by the snake sinusoidal creeping locomotion [11], the ith joint angle of the robot is described as

$$\phi_i = A\sin(\omega t + (i-1)\delta) + \phi_o \tag{8}$$

To steer the snake robot to the generated path, a time-varying LOS guidance law interpreted as the saturated control law is adopted as

$$\bar{\theta} = -\arctan(K_p e_s) \tag{9}$$

where $e_s = p_y - y_d$ is the cross-track error, and $K_p = 1/\Delta$ with $\Delta > 0$ being the lookahead distance. For better steering performance, we designed the time-varying lookahead distance $\Delta = (\Delta_{\max} - \Delta_{\min})e^{-K_\Delta e_s^2} + \Delta_{\min}$, where $K_\Delta > 0$ is the convergence rate, and $\Delta_{\max}$ and $\Delta_{\min}$ are, respectively, the upper and lower bounds of the lookahead distance. The time-varying LOS guarantees that the snake robot is always directed toward the designed path generated by the ERRT-PCSI. The stability of the proposed time-varying LOS guidance law can be found in our previous work [5].

We specify the control objective which pertains to drive the snake robot approaching the ERRT-PCSI designed path using the time-varying LOS steering law. More specifically, we intend to minimize the distance between the snake robot trajectory and the designed path in the normal direction. The cross-track error e_s is defined as the normal line from the point $(x_d(s), y_d(s))$ on the designed path through the

snake robot actual position (p_x, p_y). Multiplying the e_s with the transition matrix in the global coordinate, we have the expression for e_s

$$e_s = -(p_x - x_d(\bar{s}))\sin(\alpha) + (p_y - y_d(\bar{s}))\cos(\alpha) \quad (10)$$

where $\alpha = \arctan(y_d'(s), x_d'(s))$. Thus, the control objective becomes

$$\lim_{t\to\infty} e_s(t) = 0. \quad (11)$$

In the sequel, we design the joint offset ϕ_o such that the heading angle θ converges to the LOS guidance law defined in (9). To start, we define the heading angle error variable as

$$\psi = \zeta - \bar{\zeta} \quad (12)$$

where $\zeta = \theta + \ell\dot{\theta}$ and $\bar{\zeta} = \bar{\theta} + \ell\dot{\bar{\theta}}$ with $\ell > 0$. Taking the time derivative of ψ can obtain that

$$\begin{aligned} \dot{\psi} &= v_\theta + \ell(-f_1 v_\theta + \tfrac{f_2}{N-1} v_t \bar{e}^T \phi) - \dot{\bar{\zeta}} \\ &= \ell(-f_1 v_\theta + f_2 v_t \phi_o) + \xi \end{aligned} \quad (13)$$

with $\xi = v_\theta - \dot{\bar{\zeta}} + \frac{\ell f_2 v_t}{N-1}\sum_{i=1}^{N-1} \alpha \sin(\omega t + (i-1)\delta)$. We choose a Lyapunov function candidate at this step as $V = \psi^2/2$. Taking the derivative of V along (13) gives

$$\dot{V} = \ell(-f_1 v_\theta + f_2 v_t \phi_o)\psi + \xi\psi. \quad (14)$$

We choose the joint offset ϕ_o as

$$\phi_o = \tfrac{f_1}{f_2 v_t} v_\theta + \tfrac{1}{f_2 \ell v_t}(-\eta\psi - \xi) \quad (15)$$

where η is a positive control gain. Using (15), the time derivative of V becomes

$$\dot{V} = -\eta\psi^2 \leq 0. \quad (16)$$

The design of the joint coordinates ϕ_i given by (8) is complete.

V. Simulation and Experiments

A. Simulation Study

The performance of the perception-aware path following method is examined in MATLAB. The controller proposed in Section IV is adopted to navigate the snake robot to the target. First, the proposed LiDAR-based map searching method finds the highest rewards quadrants that are most likely to approach the target. The angle information a_r, the connectivity b_r, and the distance information c_r are given as 1.0, 0.8, and 0.6, respectively. Second, we use the ERRT to generate an exotic path from the starting point to the highest rewards quadrant with the Gaussian distribution sampling strategy. The Gaussian sampling strategy ensures ERRT to expand nodes to the highest rewards quadrant. The standard deviation δ_r and δ_θ are selected as 12 and $\pi/2$, respectively. The offsets are set as $r_0 = 3$ and $\theta_d = 0.4\pi$. Various maneuvers are generated simply by changing these parameters according to the robot location and the approaching strategy. Varying step improves the ERRT searching speed, and potential function creates a boundary tolerance area. The shortest step l_{min} and longest step l_{max} are assigned values like $l_{min} = 2$, $l_{max} = 10$, respectively. For the potential function, the parameters are given as $\mu = 0.4$ and $\rho_0 = 2.5$. Then, the PCSI method is adopted to smooth the ERRT path. Considering the computation load and the algorithm execution efficiency, we pick three waypoints from the ERRT for PCSI smoothing. The snake robot in our Robotics and Biomimetics Laboratory has a length of $L = 100$cm, and eight links of mass $m = 1.08$kg. In order to avoid singularity, we give the snake robot initial tangent and normal velocity as $v_t(0) = 20$cm/s and $v_n(0) = 12$cm/s, respectively. The LOS guidance method is initialized with setting convergence rate $K_\Delta = 2$, and the constant proportional gain $K_p = 0.3$. The Newton-Raphson method is used to find the roots of the cubic polynomials (6) and (7).

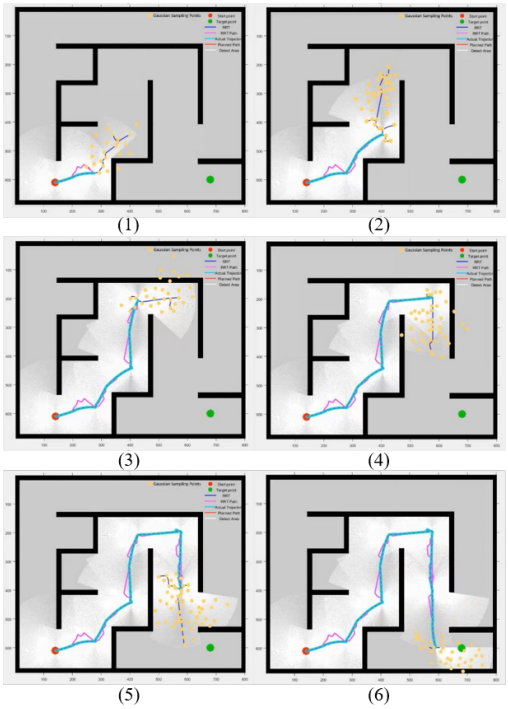


Fig. 4. The perception-aware path planning and following simulation result (sequentially displayed). The grey area is unknown, and the white space is LiDAR scanned area; the Gaussian distribution sample points are depicted by the yellow dots; the red dot and the green dot denote the start and the target point, respectively; the blue line is the ERRT exploration leaf, while the pink line is the ERRT generated path; finally, the dash red line is the planned smooth path, and the cyan line represents the snake robot actual trajectory.

The simulation result is illustrated in Figure 4. As manifested by the theoretical analysis, the LiDAR scanning algorithm biases the search area toward the target. The ERRT generates a feasible but exotic path for the snake robot plotted with a pink line. Then, a red dash line is formed by the PCSI method, which is available for the snake robot locomotion. The cyan line shows the trajectory of a snake robot, which shows the performance of the time-varying LOS based controller.

B. Experiments

The proposed perception-aware planning and tracking control algorithm is implemented with experiments in a card-

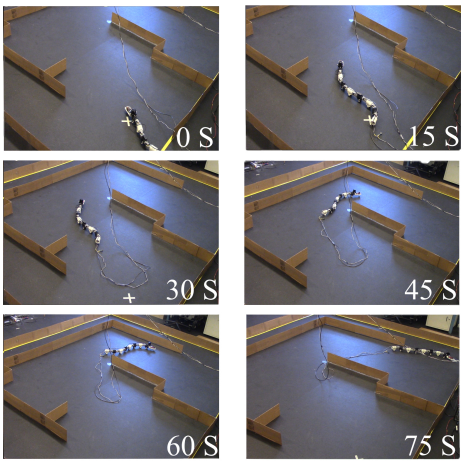


Fig. 5. Experimental snapshots of perception-aware path following.

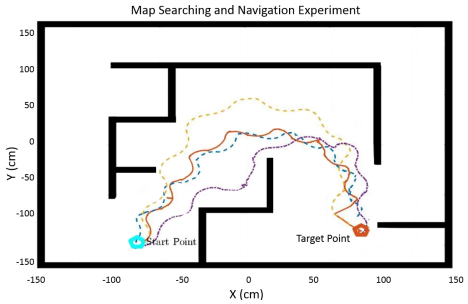


Fig. 6. Perception-aware path following results (4 trails).

board built maze. The size of the maze is 300cm×400cm. It is worth noting that the LiDAR scanning is continuous in the experiment, which refers to the continuous scanning data in each step the snake robot moves. A LiDAR system is mounted on the head of the snake robot to acquire the local map information at a scanning frequency of 10Hz, and the angular resolution is 0.391°. The hector mapping can be utilized to create a costmap and estimate 2D position. Our ERRT path planner provides feasible path information to the move_base node for navigation. It is foreseeable that the generated paths will be slightly different due to the random propriety of ERRT. The proposed framework uses a local planner to accomplish its global navigation task, which allows the autonomous navigation of a snake robot in a maze. In our setup, the high-level computer is running a Linux distribution. The robot operation system (ROS) is the middleware that ensures communication between different entities of the snake robot through various nodes. With the proposed LiDAR scanning algorithm and ERRT method, the snake robot can find a path to the target and follow it. The parameters for initializing the snake robot are $A = 13.5$cm, $\omega = 25°/\text{s}$, $\delta = 20°$, $\theta = 30°$, $v_\theta = 0°/\text{s}$, $v_\phi = 0°/\text{s}$. To avoid slippage, the snake robot moves slowly at the speed of $v_t = 5\text{cm/s}$ and $v_n = 0\text{cm/s}$ during the straight-line motion and the speed of $5°/\text{s}$ during turning motion. The snapshots of the real-time path finding and following experiment are presented in Fig. 5. The entire locomotion lasts 75 seconds. We conducted the perceptive-aware path finding and following four times to test the algorithm reliability and the method's performance. The snake robot successfully reached the target point in all four trails. Note that four-trail trajectories are not overlaid due to the small differences on every-time perception and robot driving that reflect the real-world experimental conditions, as shown in Fig. 6. The results also reflect the random property of the ERRT, but the ERRT is still reliable to plan a feasible path for the snake robot to reach the target.

VI. Conclusions

In this work, we have presented a perception-aware path finding, planning and following framework to ensure the snake robot autonomously approaches a predefined target point without requiring exact knowledge of the encountered environment *a priori*. The proposed framework makes a class of snake robots possible to fulfill missions in the unknown environment with complexity. The efforts on the new method that well leverages the LiDAR scanning based real-time mapping, ERRT-PCSI path planner, with time-varying LOS based steering, take the autonomy of the snake robot to a higher level. Since the path can be planned in real-time based on local scanned environmental information, the proposed approach has the potential to incorporate other desirable features such as avoiding moving objects. Both simulation and experimental results were included to demonstrate the performance of the proposed perception-action autonomous scheme for snake robots.

References

[1] S. Hirose and M. Mori, "Biologically inspired snake-like robots," in *International Conference on Robotics and Biomimetics*, 2019.

[2] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, *Snake robots: modelling, mechatronics, and control*. Springer Science & Business Media, 2012.

[3] P. Liljeback, I. U. Haugstuen, and K. Y. Pettersen, "Path following control of planar snake robots using a cascaded approach," vol. 20, no. 1, 2012, pp. 111–126.

[4] G. Wang, W. Yang, Y. Shen, H. Shao, and C. Wang, "Adaptive path following of underactuated snake robot on unknown and varied frictions ground: theory and validations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4273–4280, 2018.

[5] W. Yang, G. Wang, H. Shao, and Y. Shen, "Spline based curve path following of underactuated snake robots," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[6] F. Kendoul, "Towards a unified framework for uas autonomy and technology readiness assessment (atra)," in *Autonomous Control Systems and Vehicles*. Springer, 2013, pp. 55–71.

[7] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[8] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard, and D. Nardi, "Fast and accurate slam with rao–blackwellized particle filters," *Robotics and Autonomous Systems*, vol. 55, no. 1, pp. 30–38, 2007.

[9] G. Haitao, Z. Qingbao, and X. Shoujiang, "Rapid-exploring random tree algorithm for path planning of robot based on grid method," *Journal of Nanjing Normal University (Engineering and Technology Edition)*, vol. 2, no. 14, 2007.

[10] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[11] S. Ma, H. Araya, and L. Li, "Development of a creeping snake-robot," in *2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (ICRA)*, 2001, pp. 77–82.