University of Nevada, Reno

Adaptive Path Following of the Biomorphic Hyper-Redundant Snake Robot in Unconstructed Environment

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in
Electrical Engineering

by

Weixin Yang

Yantao Shen, Ph.D., Dissertation Adviser

December-2019

University of Nevada, Reno

THE GRADUATE SCHOOL

We recommend that the dissertation
prepared under our supervision by

Weixin Yang

entitled

Adaptive Path Following of the Biomorphic Hyper-Redundant Snake Robot in Unconstructed Environment

be accepted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

Yantao Shen, Ph.D., Committee Chair

M. Sami Fadali, Ph.D., Committee Member

Hao Xu, Ph.D., Committee Member

Hung La, Ph.D., Committee Member

Wanliang Shan, Ph.D., Committee Member

Matteo Aureli, Ph.D., Graduate School Representative

David W. Zeh, Ph.D., Ph.D., Dean, Graduate School

December-2019

Abstract

The efficient movement of biological snakes in various environments is the result of a long evolution. It is a challenge for researchers to make snake robots gain high efficiency, athletic agility, and environmental adaptability of snakes due to the complicated motion control issues caused by under-actuation, high-dimension nonlinearity, and uncertainties in kinematics, dynamics and interactions with complex environments.

This work relies on a custom-built snake robot that can mimic the locomotion capabilities of snakes, such as serpentine, sidewinding, and rectilinear motions, which stems from our prior efforts on its modular mechanical design, advanced electronic system, and efficient driving capabilities. Based on this robot system, we moved forward to study the closed-loop control strategy for a class of snake robots and implemented pathfinding and following, which is the challenging and practical problem in control of snake robots.

In this dissertation, we first investigate straight line path-following problems for a class of planar underactuated snake robots. For this purpose, an adaptive controller that can autonomously drive the snake robot moving on ground with unknown and varied friction coefficients is designed and validated. Combining with time-varying Line of Sight (LOS) and Parametric Cubic Spline Interpolation (PCSI) path planning methods, snake robots tracking arbitrary paths is further explored and the

comprehensive curve path following is realized. Last, the perception-aware pathfinding and following for snake robots in unmodeled and unknown environments are also studied, and a simple but efficient control methodology is developed. The proposed method is validated by extensive simulations and experimental results.

More importantly, in this dissertation, we addressed to achieve a generic adaptive controller for motion control of a class of snake robots with uncertain dynamics. The advanced controller is proved to be stable and also it does not require a high gain for reaching ideal control performance. With the proposed controller and path following methods, snake robots are capable of improving mobility in different environments, which promises the potential of using snake robots in diverse real-world applications, like the earthquake rescue, narrow space surveillance and even overwater or underwater explorations.

Keywords: Bio-inspired, Snake Robot, Pathfinding, Path-following, Model-based Control, Adaptive Controller, Underactuated.

To my fimily

Acknowledgements

This dissertation would not have been possible without the help, support and guidance of many individuas. To start I would like to thank Dr. Yantao Shen, my advisor for allowing me to join his lab when it was first established. His patience and training in all aspects of being a good research scientist are greatly appreciated and will not be forgotten. I also would like to thank my committee members; Dr. M. Sami Fadali, Dr. Hao Xu, Dr. Hung La, Dr. Matteo Aureli, and Dr. Wanliang Shan for all their enthusiasm, suggestions and constructive criticism.

Being in Dr. Shen's lab I had privilege of working with Dr. Gang Wang, the research associate, and three exceptional students, Cong Peng, Mehdi Rahimi, and Na Zhao. A special thank to Dr. Gang Wang for helping on the control system design. I really enjoyed the conversations on research and their good nature when things got difficult. And I also would like to thank Ming Feng, Zejian Zhou, the members in Dr. Hao Xu's lab for their technical help in using lab equipment and facilities for the experimental implementation.

Lastly, I would like to thank my family for their incredible support, encouragement and understanding. I have been blessed by my parents and their guidance, advice and support was always there when I needed it most. Thank you to my girlfriend, Sirui Zhang, for her smile on bad days and for her encouragement.

Contents

1 Introduction **1**

1.1 Research Background and Overview 1

1.2 The State-of-art of Snake Robots 2

1.2.1 Current Snake Robots 2

1.2.2 Snake Robot Control 4

1.3 Methodology Review 14

1.3.1 Snake Robot Kinematics and Dynamics 14

1.3.2 Parametric Cubic Spline Interpolation Based Curve Path Generation 15

1.3.3 ERRT Based Path Generation 16

1.3.4 Time-varying LOS Guidance Law 17

1.3.5 Backstepping Enabled Path Follow Control 19

1.4 Research Problems, Challenges, and Our Solutions 19

1.5 Contributions 20

1.6 Structure of the Dissertation 21

2 The Snake Robot: Design and Locomotion Performance Validation 23

2.1 Snake Robot Mechanical Design 24

2.1.1 Snake Robot Segment Design 24

2.1.2 Snake Robot Transmission System Design 30

2.2 Snake Robot Electronic System Design 36
2.2.1 Electronic Control Modules 37
2.2.2 Electronic Design of the Snake Robot Control Modules 42
2.2.3 Softwares and Programming Environment 48
2.3 Snake Robot Locomotion Performance Validation 54

3 Snake Robot Kinematics and Dynamics 61
3.1 Kinematics 62
3.2 Gait Pattern 65
3.3 Dynamics 68

4 Adaptive Path Following of Snake Robots 73
4.1 Adaptive Straight Path Following 73
4.1.1 Control Objective 74
4.1.2 Adaptive Backstepping Controller Design 75
4.1.3 Simulations and experiments 83
4.1.3.1 Simulation Study 83
4.1.3.2 Experimental Study 88
4.2 Parametric Curve Path Following Control 95
4.2.1 Parametric Cubic Spline Interpolation Based Curve Path Generation 97
4.2.2 Time-varying LOS based Guidance Law Design 104
4.2.2.1 Time Varying LOS-Based Steering 104
4.2.2.2 Controller Design 106
4.2.3 Monotonic Curve Path Following 109
4.2.4 Closed-loop Curve Path Following 115
4.2.5 Cross Curve Path Following 119

5 Perception-Aware Pathfinding and Following of Snake Robots in Unknown Environments **128**

5.1 Problem Statement 129

5.2 LIDAR Searching Algorithm 130

5.3 Executive Rapidly-Exploring Random Tree Path Generation 131

5.4 Path Smooth and Control Objective 135

5.5 Perception-Aware Pathfinding and Following 138

5.5.1 Simulation Study 138

5.5.2 Experimental Validation 141

6 Conclusions and Future Work **144**

References **155**

List of Figures

1.1 The world's first snake robot developed by Prof. Shigeo Hirose in 1972 [19] 5
1.2 The snake robot ACM R3 developed at Tokyo Institute of Technology [44] 5
1.3 The snake robot S5 developed by Dr. Gavin miller [2] 6
1.4 The snake robot ACM R5 developed at Tokyo Institute of Technology [65] 7
1.5 The snake robot Uncle Sam developed at Carnegie Mellon University [63] 8
1.6 A snake robot with a miniature joint mechanism developed at Tokyo Institute of Technology [23] 8
1.7 The OmniTread snake robot developed at the University of Michigan, the robot has pneumatic joints and is covered by motorized tracks [5] 9
1.8 A snake robot with a skin drive propulsion system developed at Carnegie Mellon University [42] 10
1.9 The snake robot Kulko developed at the Norwegian University of Science and Technology. Each joint module is covered by force sensors in order to measure contract forces from the environment [34] 10
1.10 Snake robot kinematics [21]. 15

1.11 PCSI generated smooth curth path. 16
1.12 ERRT path generation method example. 17
1.13 Time-varying LOS guidance diagram. 18
1.14 The research roadmap of the dissertation. 22

2.1 Natural snake skeleton [62] 25
2.2 Snake robot body upper part 27
2.3 Snake robot body bottom part 27
2.4 Passive wheel 28
2.5 Passive wheel's rubber ring 28
2.6 Whole snake robot body part 29
2.7 Two separate parts of servomotor compartment 31
2.8 Slots of the servomotor compartment 32
2.9 Power transmission gear set 32
2.10 Gear sets connection 33
2.11 Single joint of snake robot 34
2.12 Snake robot head 34
2.13 The whole snake robot 35
2.14 Atmel SAMD21 MCU 39
2.15 16-channel PWM controller 40
2.16 9 DOF smart absolute orientation sensor 41
2.17 Digi XBee module allows for wireless transmission 43
2.18 MCU peripheral circuit schematic 45
2.19 SPX3819M5-L3-3 voltage regulator 46
2.20 Wireless communication chip XBee peripheral circuit 46
2.21 BNO055 absolute orientation sensor 47
2.22 BNO055 absolute orientation sensor 48

2.23 PCB sanity 49
2.24 Snake robot control PCB design 50
2.25 The connection board schematic 51
2.26 The connection board PCB layout 52
2.27 Snapshot of snake robot serpentine locomotion 55
2.28 Trajectory plot of serpentine locomotion 56
2.29 x coordinates of the serpentine locomotion 56
2.30 y coordinates of the serpentine locomotion 57
2.31 Snapshot of snake robot sidewinding locomotion 58
2.32 Trajectory plot of sidewinding locomotion 58
2.33 x coordinates of the sidewinding locomotion 59
2.34 y coordinates of the sidewinding locomotion 59

3.1 Snake robot kinematic model 64
3.2 Snake robot motion simulation with angle offset ϕ_0 The joint offset $\phi_0 = -0.3rad$ when $t = 120s$, and $\phi_0 = -0.6rad$ when $t = 165s$. . . . 67
3.3 For turning angle offset ϕ_0, the snake robot makes a sharp turn at waypoint $EP4$. 67
3.4 Symbols of snake robot kinematics and dynamics. (Adopted from [33]) 68

4.1 Structure of the proposed adaptive path following controller. 78
4.2 Trajectory of p_y. 85
4.3 Trajectories of θ and $\bar{\theta}$. 86
4.4 Velocities v_t and v_n. 86
4.5 Parameter estimates. 87
4.6 Comparative simulation result. 87
4.7 Snake robot employed in the experiment. 88
4.8 Trajectory of p_y in the comparative simulation. 88

4.9 Trajectories of θ and $\bar{\theta}$ in the comparative simulation. 89
4.10 Path following of the snake robot under the proposed adaptive controller on two different frictions terrains. 92
4.11 Trajectory of p_y. 93
4.12 Trajectories of θ and $\bar{\theta}$. 93
4.13 Parameter estimates. 94
4.14 Comparative experiment result. 94
4.15 Snake robot tracking trajectory on a monotonic curve. 110
4.16 Lookahead distance varies to achieve appropriate steering. 111
4.17 The errors e_s and c_s converge to zero in a short time. 111
4.18 Monotonic curve path following experiment of the snake robot under the proposed controller. The first frame is the customized mechanical snake robot we used in the experiment. All these frames are taken from an experimental movie. 113
4.19 Monotonic curve path following experiment repeated four times where all trajectories pass through five waypoints. 116
4.20 Monotonic curve path following experiments results: e_s, c_s, and Δ. . . 117
4.21 Snake robot tracking trajectory on a closed-loop curve. 117
4.22 The simulated time-varying LOS on a closed-loop curve. 118
4.23 Closed-loop path following simulation results: e_s and c_s. 118
4.24 Closed-loop curve path following experiment repeated three times with the first and last waypoint joined. 119
4.25 Closed-loop curve path following experimental results: e_s, c_s, and Δ. 120
4.26 Closed-loop path following experiment of the snake robot under the proposed controller. All frames are taken from a movie recording of the experiment. 121

4.27 Snake robot tracking trajectory on a cross-line curve. 122
4.28 Simulated time-varying LOS on a cross-line curve. 123
4.29 Cross-line path following simulation results: e_s and c_s. 123
4.30 Cross-line path following experiment of the snake robot under the proposed controller. All frames are taken from a movie recording of the experiment. 125
4.31 Cross-line curve path following experiment repeated three times. The snake robot passes through all waypoints. 126
4.32 Cross-line curve path following experimental results: e_s, c_s, and Δ. . . 127

5.1 Gaussian distribution sampling strategy. The red dots represent sample points. 136
5.2 Tree expansions comparison for RRT and ERRT illustrating path optimality. Path from the blue initial point to the red target point is shown in red. (a) Basic RRT solution, (b) ERRT solution. 136
5.3 Flow chart of the proposed perception-aware path following scheme. . 139
5.4 LIDAR based path following simulation 140
5.5 Experiment snapshots 142
5.6 ERRT path search experiment repeated four times 143

Chapter 1

Introduction

1.1 Research Background and Overview

Snakes have survived on earth for millions of years and have developed many special locomotion modes to provide traversability in irregular environments. Bio-inspired snake robots are robotic mechanisms developed to mimic the excellent mobility capabilities of biological snakes. Snake robots typically consist of several joint modules with the same mechanical structure, which allows such a device to possess many degrees of freedom for moving in many complex and unstructured terrains. Snake robots have therefore great potential in various applications in irregular and unstructured terrains, where a high degree of flexibility is needed. 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 installations, the need for high intelligence of snake robots is obvious; therefore, an autonomous control method and its path following strategy are expected. Unfortunately, due to the underactuated nature of snake robots as well as its many degrees of freedom, stable closed-loop control of the robot is a challenging problem.

1.2 The State-of-art of Snake Robots

1.2.1 Current Snake Robots

Inspired by the unique locomotion methods of the biological snake, snake robots have the potential to provide mobility in different environments. The snake robots typically consist of serially connected joint modules capable of bending in one or more planes. Many degrees of freedom make snake robots challenging to control but give them the ability to travel in irregular and challenging environments where conventional wheeled robots cannot reach. Research on snake robots has been conducted for several decades. The early snake robot's locomotion analysis has been reported by Gray [15] in the 1940s. And Hirose developed the first snake robot in the world in 1972 [18]. In the last 20 years, there have been many types of research focusing on the analysis of snake robot's locomotion methods. Many types of bionic locomotion have been developed based on this research, which provide many suggestions for building a model to develop and control the snake robots. Although the majority of the literature on snake robots has focused on locomotion over flat surfaces, there is a growing need for a snake robot that can move in the challenging environments encountered in practical applications. Meeting this need is essential to realize snake robots' potential in the future.

Research on snake robots is inspired by the robust motion capabilities of biological snakes. These amazing creatures have emerged through millions of years of evolution to adapt to most of the different environments in the earth. Biomechanical studies of snakes are, therefore, relevant to research on snake robots. One of the earliest analytical studies of snake locomotion methods was proposed by Gray in 1940s, he proposed a mathematical description of force acting when the snake is moving. His studies gave properties of snake locomotion. One of Gray's conclusions was that the

forward motion of a snake on a flat surface requires the existence of an external forces acting in the average direction to the snake body. Hirose studied the biological snake and built the mathematical model as a continuous curve that cannot move sideways. A famous result proposed by Hirose is the serpentine curve (the most common form of snake locomotion) equation. Hirose discovered that a close approximation to the shape of a biological snake during lateral undulation is given by a planar curve whose curvature varies sinusoidally. Hirose also proposed mathematical descriptions of how external factors, such as ground friction or temperature, affect the snake locomotion efficiency and the shape of the locomotion curve.

The world's first snake robot was developed by Hirose as early as 1972. The snake robot is shown in Fig. 1.1, which equipped with passive wheels. The friction between passive wheels and ground enables Hirose's snake robot to move forward on the flat surface. Several other snake robots with passive wheels have been proposed over the years. Illustrations are shown in Fig. 1.2, Fig. 1.3, and Fig. 1.4. Some of the robots only have locomotion on a flat surface, while others have the ability to move their links both horizontally and vertically. The two degrees of freedom of each modular link enables the snake robot to have 3D locomotion, which improves the robot's adaptability to environments. Some robots have shielded joint modules that enable motion in some special environments like mud and dust, and even enable locomotion under water. A common feature of these snake robots is that, generally, they are able to move across relatively flat surfaces since the passive wheels block the motion in the cluttered environment. Such mechanisms are therefore limited to run in the lab environment but not for practical application in a challenging environment.

Some snake robots do not have wheels,emphi.e., robots with straight links interconnected by motorized joints. Despite its lack of wheels, the snake robot maintains anisotropic ground friction since the underside of each link has edges, or grooves, that

run parallel to the link. These robots can, therefore, move forward by lateral undulation through purely planar motion. Robots whose ground friction properties are isotropic, on the other hand, can move forward during lateral undulation by resorting to sinus lifting, *i.e.*, by slightly lifting the peaks of the body wave curve from the ground. However, snake robots with isotropic friction are mostly used for studying gaits other than lateral undulation, such as gaits based on sidewinding, inchworm motion, or lateral rolling. Illustrations shown in Fig. 1.5 and Fig. 1.6 focuses on the development of small, light-weight, and strong joint actuation mechanisms, which are important for many future applications of snake robots. There are also some snake robots equipped with active propulsion along the body. Such snake robots are equipped with motorized wheels in each link or by installing tracks along the body of the snake robot, or by employing a screw drive mechanism as shown in Fig. 1.7 and Fig. 1.8, respectively.

The environment sensing for snake robots in previous research is limited. Early in 1972, Hirose equipped a snake robot with contact switches, which enable the snake robot to have basic obstacle avoidance ability. In the latter research, some snake robots with active wheels are equipped with a three-axial force sensor. The translational forces are measured by the force sensor on the wheel axis based on optical range measurements. The snake robot, shown in Fig. 1.9, is in ball-shape joint modules equipped with force sensors mounted underneath the shell of each module.

1.2.2 Snake Robot Control

Research endeavors on bio-inspired snake robots have extensively appeared for over 40 years since Hirose *et al.* designed the world's first snake robot in 1972 [17]. Some seminal works are [55, 38, 39], and [10]. And a comprehensive exposition of state

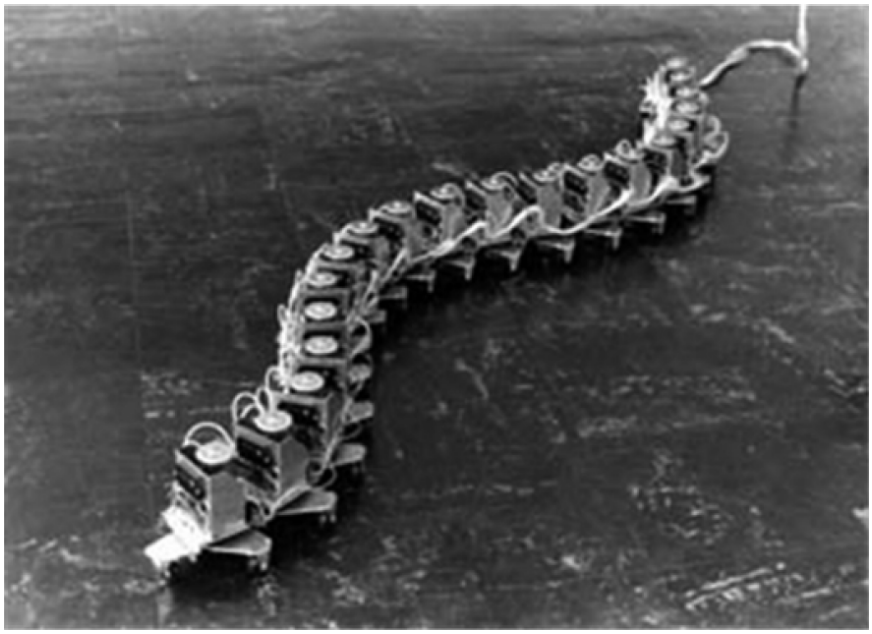


Figure 1.1: The world's first snake robot developed by Prof. Shigeo Hirose in 1972 [19]

Figure 1.2: The snake robot ACM R3 developed at Tokyo Institute of Technology [44]

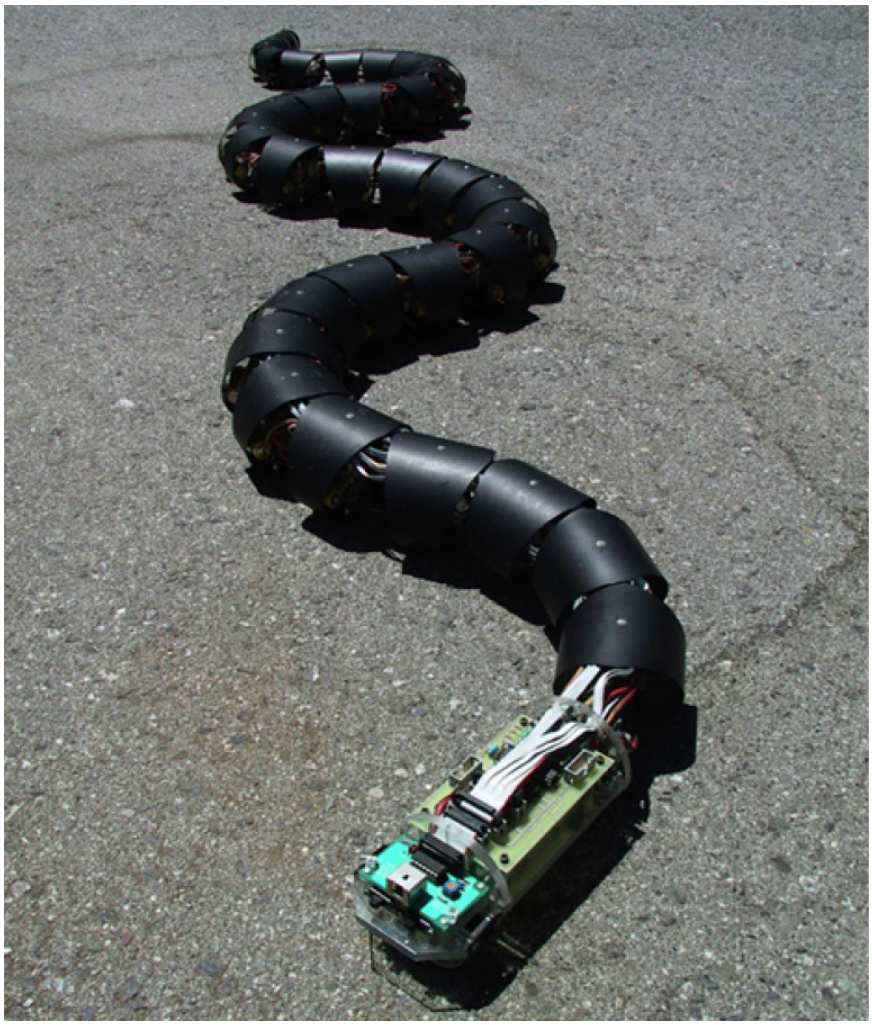


Figure 1.3: The snake robot S5 developed by Dr. Gavin miller [2]

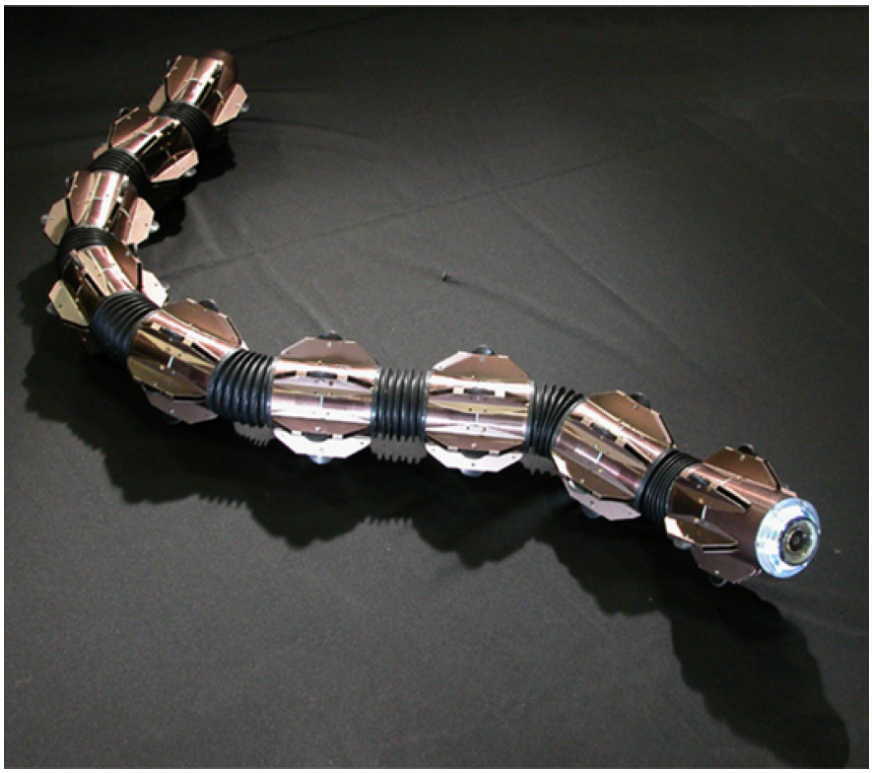


Figure 1.4: The snake robot ACM R5 developed at Tokyo Institute of Technology [65]

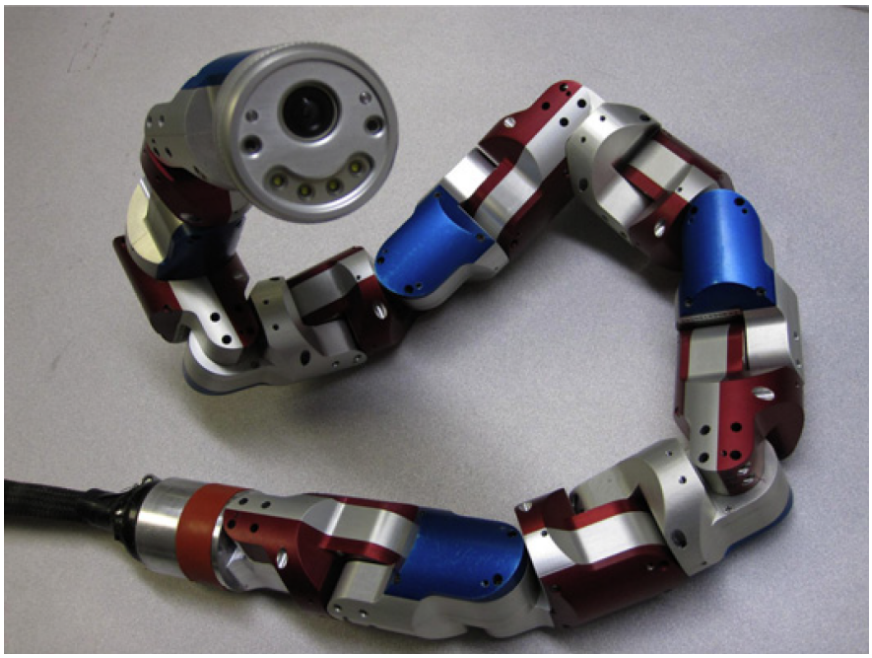


Figure 1.5: The snake robot Uncle Sam developed at Carnegie Mellon University [63]

Figure 1.6: A snake robot with a miniature joint mechanism developed at Tokyo Institute of Technology [23]

Figure 1.7: The OmniTread snake robot developed at the University of Michigan, the robot has pneumatic joints and is covered by motorized tracks [5]

Figure 1.8: A snake robot with a skin drive propulsion system developed at Carnegie Mellon University [42]

Figure 1.9: The snake robot Kulko developed at the Norwegian University of Science and Technology. Each joint module is covered by force sensors in order to measure contract forces from the environment [34]

of the art on bio-inspired snake robots can be found in survey papers [49, 60], and references therein.

Controlling the planar snake robot is a challenging problem due to its intricate dynamic model, which is highly coupled and possesses at least three degrees of under-actuation. In the available studies, much attention has been paid for the snake robots to duplicate the four main types of snake locomotion [39, 20, 47], which is actually the open-loop control. However, autonomous closed-loop mobility is often required in the applications of the robots. In the available studies, much attention has been paid to open-loop control for the snake robots to duplicate the four main types of snake locomotion [39, 20, 47]. However, autonomous closed-loop mobility is often required in practical robot applications. The (from a control perspective) practical path following problem for snake robots has been considered very rarely. There are some efforts concerning the mathematical models of snake robots to improve control efficiency in environments, which makes them close to real-life applications [40, 37]. As far as closed-loop control is concerned, further research is still required due to the intricate dynamics of the snake robots that is highly coupled and carries many degrees of under-actuation. In the context of uncertain dynamics, various solutions have been obtained in robot control (see, e.g., [32]). Nevertheless, to the best of our knowledge, the results on the control of uncertain snake robots are still limited due to their inherent underactuated characteristic and high-coupling nonlinear dynamics. In [45], it is assumed that the uncertainty is bounded by a known constant and a sliding mode control is given to solve the velocity tracking and head-angle control problem of the planar snake robot. Recently, controllability and stability analysis of planar snake robot locomotion is intensively investigated in [36],any asymptotically stabilizing control law for a planar snake robot to an equilibrium point must be *time-varying*. A control method for straight line path following problem of snake robots

is proposed with the help of the Poincaré map. However, as emphasized in [49], Poincaré map analysis is based on simulations and is not always feasible as there is no general method to construct a Poincaré map. To overcome this obstacle, in light of the stability of the cascaded system, the path following control of two-dimensional snake robots is studied, and a perfect following performance is achieved and validated via simulations as well as experiments in [50]. A cascaded system with Line of Sight (LOS) based navigation is studied to solve the path following problem for a two-dimensional snake robot with known constant frictions [33]. Furthermore, by using virtual holonomic constraints (VHCs) and reduction theorems, the problem of path following control for a planar snake robot is approached in [43]. The VHCs are also employed in neural-network control for the path following of snake robots [8]. More recently, the neural-network-based central pattern generator (CPG) brings a snake robot closer to a biological snake. The CPG generates rhythmic patterns by the coupled nonlinear oscillators instead of digital signals, which leads to the smooth locomotion of a snake robot [64, 9].

Despite recent progress towards the control of planar snake robots, specific problems remain open. The results above are obtained requiring the assumption that the friction coefficients in the dynamic model of the snake robot are precisely known. However, friction forces are usually unknown and variable in practical operating environment. In addition, even for the same snake robot these coefficients depend on the interactive terrains [3]. Assuming that friction coefficients are known and time-invariant constants results in an inconsistency between both theoretical and in-field performances. Therefore, research in snake robots without *a priori* knowledge of friction coefficients is of more practical significance. With further development of this method for solving problems with unknown and varied friction, an adaptive control algorithm has been provided in [61]. Nevertheless, these control approaches [33, 61]

are limited to the straight line paths following. Alternative solutions to curved path following are presented in [53, 43], by employing hierarchical structure and maneuvering control.

Path-planning is the prerequisite of path following, which pertains to regulating a route when robots are moving from one location to the destination. Practically, the path-planning initialization is to introduce several fixed points in space from a start point to an end point, namely the waypoints. Then the desired curve path is generated by joining successive straight lines that pass through these waypoints. However, the aforementioned straight line path following strategy is not applicable to the curve situation because such a path has a discontinuous first derivative *(velocity function)* at each waypoint. One solution, namely Dubins path, is promulgated in [56]. The Dubins method still considers a straight line between two waypoints but achieves turning by inscribing circles at the waypoint. This solution, to the best of our knowledge, gets little attention because the generated path cannot go through the waypoint, which may lead to unpredictable problems with obstacle avoidance. Considering the aforementioned requirements (velocity continuity and passing through the waypoint), Bézier curve is proposed in order to form a smooth curve in [22]. However, it also brings a high computational load to the onboard control system. More recently, Parametric Cubic Spline Interpolation (PCSI) and Cubic Hermite Interpolation (CHI) have also been extensively studied in path-planning research [4], [59]. Regarding the special characteristics of the snake robot mechanical structure, PCSI is able to produce a smooth curve at the turning point, thus it is able to reduce the power consumption of the servomotor.

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 is needed to generate these trajectories or paths. Such a

requirement restricts the application of the developed strategies to a relatively small class of applications. For most practical applications, it is highly desirable to explore a path following strategy and design control algorithms by which the snake robot can real-time plan the path, relying only on available local environmental information. Algorithms for obstacle detection and path planning using a point cloud generated by the LIDAR are studied in [30, 24], however, they cannot be applied to non-holonomic robots.

1.3 Methodology Review

1.3.1 Snake Robot Kinematics and Dynamics

First, the kinematics of the snake-shaped robot were established. The simplified models were proposed in [50] and [54], which greatly simplifies the existing planar snake-like robot motion model. The model only captures the basic characteristics of the snake's movement, so the complexity is greatly reduced compared to the original model used in the analysis [35]. The kinematics of the snake robot is shown in Fig. 1.10. A complex and accurate model of the snake robot is considered in a Lagrangian framework in [33]. Basically, the snake robot is composed of n rigid links and $n-1$ joints. Each link has a passive wheel on its center, which does not slip nor slide sideways. With the coordinate of each center of mass of a link, it is possible to specify the kinematic map of the snake robot [21]. Moreover, a simplified snake robot is developed [37] to capture only these essential properties of snake locomotion, thereby significantly reducing the complexity compared to the original model. The dynamics of the planar snake robots will be expressed in Section 1.10.

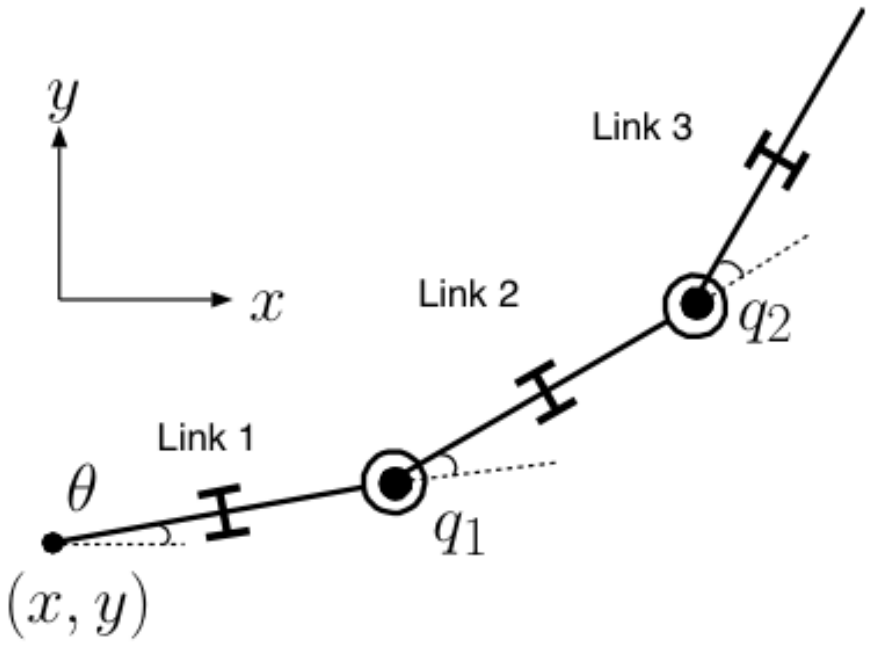


Figure 1.10: Snake robot kinematics [21].

1.3.2 Parametric Cubic Spline Interpolation Based Curve Path Generation

The previously described snake robot kinematics and dynamics are accurate enough to control the snake robot. However, before we are going to design the controller, we need to regulate a path which is suitable for the snake robot to traverse. Such that we use the parametric cubic spline interpolation based curve path generation.

The path of the snake robot is specified in terms of n waypoints. The waypoints $p_{wpt}(i)$, $i = 1, \ldots, n$ is defined by the coordinates $p_{wpt}(i) = (x_{wpt}(i),\ y_{wpt}(i)) \in R^2$, and the set of waypoints can be expressed as

$$\begin{aligned} p_{wpt} \quad = \quad & \{(x_{wpt}(1), y_{wpt}(1)), (x_{wpt}(2), y_{wpt}(2)), \\ & \ldots, (x_{wpt}(n), y_{wpt}(n))\}. \end{aligned} \tag{1.1}$$

The mission of path planning is to generate a desired path that from the starting

point $p_{wpt}(1)$ to the terminal point $p_{wpt}(n)$. PCSI and CHI are popular choices for curve fitting for ease of data interpolation. The particular difference between these two methods is how the slopes at waypoints are processed. In this work, the PCSI is employed for the path generation for each pair of successive waypoints in the form of $(p_{wpt}(i), p_{wpt}(i+1)), i = 1, \ldots, n-1$, which is also called a spline. A generated path is a sequence of order splines. The generated curves are plotted in Fig. 1.11.

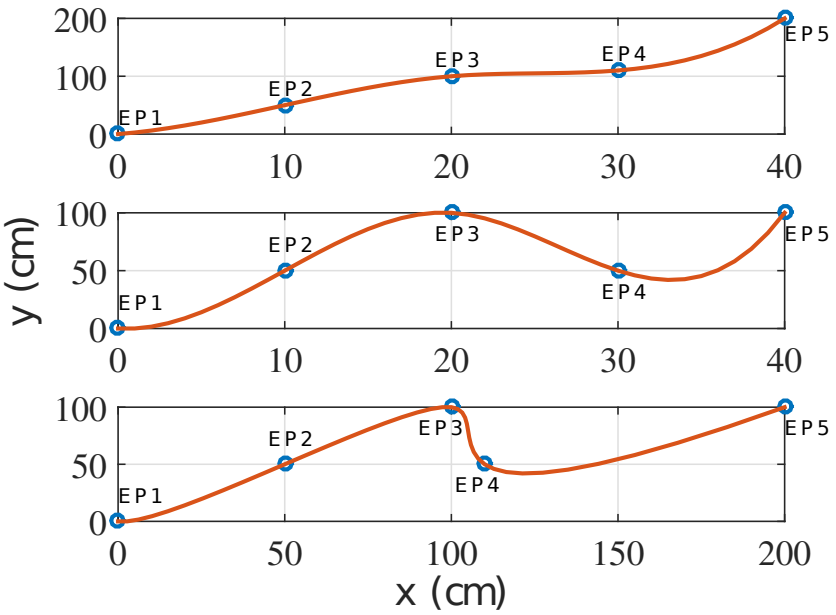


Figure 1.11: PCSI generated smooth curth path.

1.3.3 ERRT Based Path Generation

ERRT can adequately address the path planning problems in a high-dimensional space with complicated constraints. Thus, the path planning algorithm based on ERRT can avoid modeling of space by collision detection of sampling points in state space. Thereby ERRT can find a feasible path from the starting point to the target point, which is applicable to solve the path planning problem of snake robots in the

complex and unconstructed environment. The example of ERRT generated path is demonstrated in Fig. 1.12.

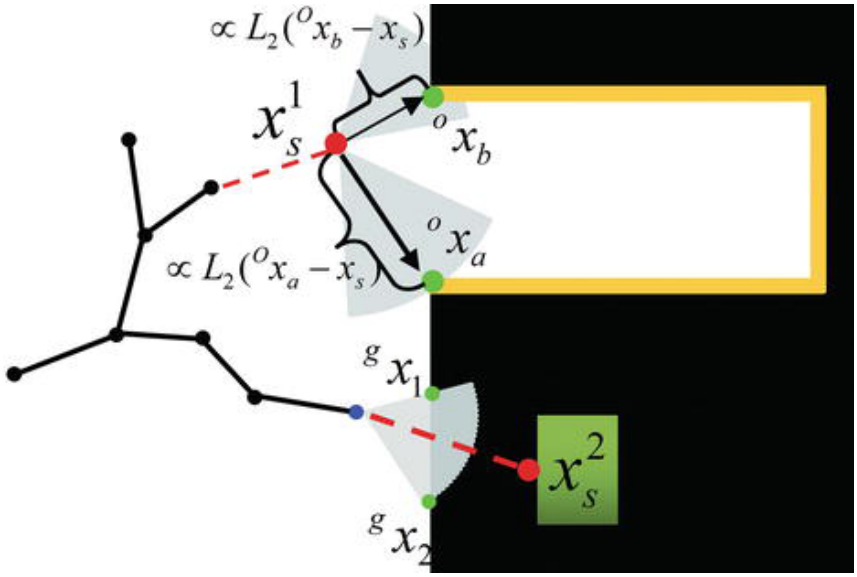


Figure 1.12: ERRT path generation method example.

1.3.4 Time-varying LOS Guidance Law

To steer the snake robot to the desired path generated by the PCSI method in, considering that the sideslip of snake robot may occur when friction varies, an integral LOS based guidance law interpreted as the saturated control law is utilized [6]

$$\bar{\theta} = -\arctan\left(K_p e_s - K_i \int_0^t e_s(\tau)\, d\tau\right), \tag{1.2}$$

where $e_s = p_y - y_d$ is the cross-track error, $K_p = 1/\Delta$ with $\Delta > 0$ being the lookahead distance. $K_i > 0$ represents the integral gain. The reference steering angle is now proportional to the e_s with a constant gain K_p. Furthermore, integral control is effective to compensate the steering angle if sideslip occurs [13].

For dynamically controlling when turning, a time-varying lookahead distance is

considered as $\Delta = (\Delta_{\max} - \Delta_{\min})e^{-K_\Delta e_s^2} + \Delta_{\min}$, 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. Typically, $\Delta_{\max} = 1.3L$ and $\Delta_{\min} = 0.4L$, where L is the length of snake robot.

The LOS guidance ensures the snake robot to be directed toward the desired endpoint (x_i, y_i) until it converges to the curve path, such that the control objective (4.1) is satisfied. The time-varying LOS is shown in Fig. 1.13.

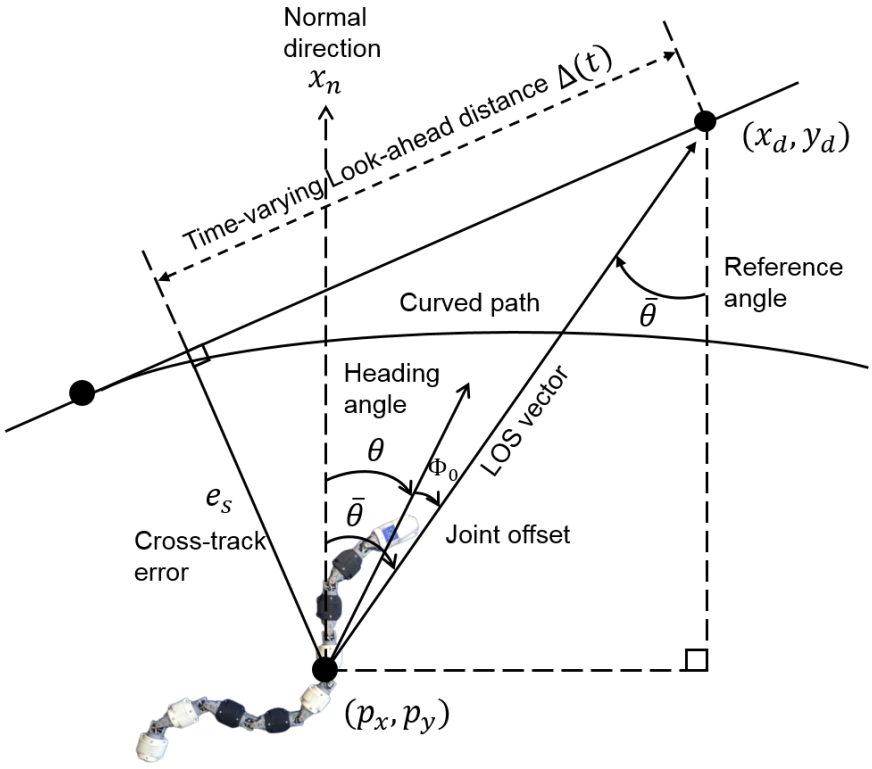


Figure 1.13: Time-varying LOS guidance diagram.

1.3.5 Backstepping Enabled Path Follow Control

Due to the fact that the snake robot model (1.10) possesses the higher-order nonlinear dynamics, the controller design will be performed by following a step-by-step procedure known as backstepping technique [28]. The detailed design procedure is given as follows.

Step 1: In this step we design the joint offset ϕ_o such that the heading angle θ converge to the LOS guidance law (1.2).

Step 2: With the purpose of making the joints track $\bar{\phi}_i$, we define the link angle error $z_2 = \phi - \bar{\phi}$. Such that we make the actual joint angle follows the reference joint angle.

Step 3: We set the actuator force control input to make the actual joint velocity following the desired joint velocity.

1.4 Research Problems, Challenges, and Our Solutions

We consider the path following problem for a class of planar underactuated bio-inspired snake robots in a complex environment with varying friction coefficients. The snake robot has nonlinear dynamics $\mathbf{x} = \dot{f}(\mathbf{x}, \mathbf{u})$, where $\mathbf{x}$ is the states and $\mathbf{u}$ is the control inputs of the snake robot. Several constraints are imposed on the snake robot such as control saturation, actuator lag, and speed bounds. These are represented by $\mathbf{x} \in \mathbf{X}$ and $\mathbf{u} \in \mathbf{U}$.

Our investigation starts on a path planning problem that generates a desired path that from the starting point $p_{wpt}(1)$ to the terminal point $p_{wpt}(n)$. PCSI is employed for the path generation for each pair of successive waypoints in the form

of $(p_{wpt}(i), p_{wpt}(i+1)), i = 1, \ldots, n-1$, which is also called a spline. A generated path is a sequence of order splines. By introducing the four constraints, we can solve the PCSI cubic equations to generate a smooth path for snake robots. The LIDAR-based ERRT allows snake robots a path finding ability by employing an initial point as the root node to generate a random extension tree with randomly sampling the leaf nodes. Then, we follow three steps to design the path following controller. Step 1: steer the snake robot towards the desired path where a time-varying LOS law is utilized. Step 2: design the controller to make the actual joint angle follows the reference joint angle. Step 3: design the actuator torque vector τ to make the actual joint velocity following the desired joint velocity.

Note that, the operating environment is dynamic and uncertain, so the path planning must be generated online. In addition, to quickly react to the sudden change in situational awareness, the planning interval must be as short as 0.1 seconds.

1.5 Contributions

In this dissertation, we consider the path following problems of planar underactuated bio-inspired snake robots on ground with uncertain friction coefficients. Benefit from the PCSI path planner and the time-varying LOS guidance, snake robots can traverse from one location to the destination through the arbitrarily planned smooth curve. Specifically, based on the backstepping technique, we design a novel adaptive controller that can compensate for unknown and varied friction coefficients online. It is proved via LaSalle-Yoshizawa theorem that the following errors converge to zero asymptotically, and all the parameter estimates are bounded. Furthermore, a perception-aware path planning and tracking framework that makes the snake robot reach a pre-defined target point in an unconstructed environment is proposed and validated.

The main contributions of this dissertation can be summarized as follows:

- The uncertain friction coefficients of the snake robot are taken into account in the controller design, which, however, have not been discussed up to now in the relevant works. With compensating for uncertain frictions, our control methods do not require high-gain control to achieve an ideal control performance compared with the maneuvering control of planar snake robots.

- Our proposed method can make the snake robot follow arbitrary curved paths, which is more advanced in real-world applications.

- The improved LOS guidance method with a time-varying lookahead distance is proposed, which helps achieve a faster convergence to the planned path than that with constant lookahead distance in existing work.

- Compared with SLAM based mapping and navigation methods, the proposed perception-aware method is a less complicated but efficient path searching and navigation algorithm and possesses high operating efficiency in an the sense of low computation.

- A modified rapidly-exploring random tree searching method that can adaptively change the size of exploration steps, is proposed to improve the exploration efficiency of the path finding for path following.

1.6 Structure of the Dissertation

The organization of this dissertation is as follows. Chapter 2 introduces the snake robot mechanical design and the associated electrical control system. Chapter 3 describes the kinematics and dynamics of the underactuated snake robots. Chapter 4 presents the path-following methods for a class of bio-inspired underactuated snake

robots in known/modeled environment. Including the adaptive straight path following, parametric curve path following. Chapter 5 is about the LIDAR searching based perception-aware pathfinding and following in unknown/unmodeled environment. Simulation and experimental results are illustrated on an 8-link snake robot to validate our theoretical findings and investigations. We conclude the dissertation in Chapter 6, and also point out the future work in this Chapter. Corresponding to the structure, to guide for outlining the research work of this dissertation, the research roadmap of the dissertation is illustrated in Fig. 1.14.

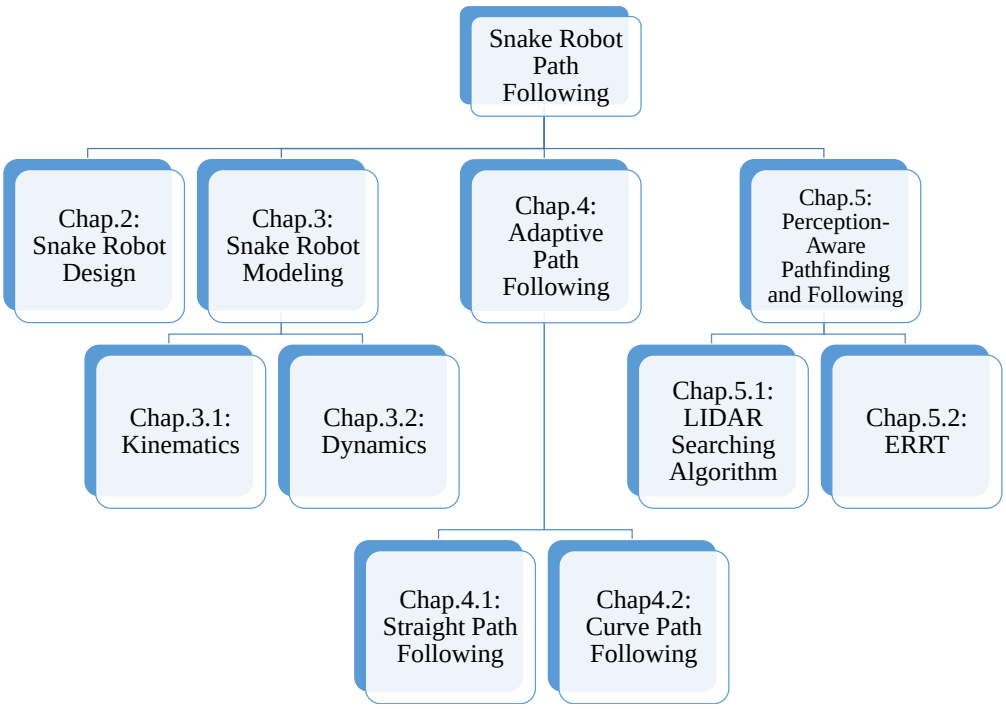


Figure 1.14: The research roadmap of the dissertation.

Chapter 2

The Snake Robot: Design and Locomotion Performance Validation

Snakes have lived on the earth for hundreds of millions of years. They have evolved a variety of locomotion patterns in order to survive in different landscapes. Our goal is to use mechanical design to replicate as much as possible the way snakes move in the natural world. Through the observation of natural snakes, we found that most snakes use the S-shape to swing forward, and some snakes in the desert use the Sidewinding method. The unique shape of the snake robot and its ability to navigate in highly complicated environments make them suitable not only for rescue missions in urban search but also for environmental monitoring missions in the ever-changing terrain. However, only strong, well-designed, reliable snake-shaped robots can accomplish these tasks.

2.1 Snake Robot Mechanical Design

Mechanical snake robots are a class of super-redundant mechanical devices that move through internal shape changes. Many factors can affect the design of mechanical snakes, such as size, power, and weight. Mechanical snake designs require complex mechanical and electrical structures to meet these constraints. We have designed a modular architecture that allows for the replication of multiple snake motion patterns, even in the event of failure of one or more servo motors. The architecture combines sophisticated mechanical design with state-of-the-art electronics and software that can be used by machine snakes to perform tasks efficiently. We considered the common faults of other modular robots and designed them accordingly. The machine snake body part is an elliptical cylinder that is the intermediate connecting parts used to connect the two power mechanisms. In order to facilitate maintenance and disassembly, the snake body is divided into upper and lower parts and assembled with screws.

The natural snake body consists of a huge number of vertebrae and other bones shown in Fig. 2.1. The body is swung by the contraction of muscles to move forward aginst the friction. Depending on the type of snake, the number of vertebrae of snakes is between 140 and 500. A large number of vertebrae guarantee a lot of degrees of freedom for natural snakes, which in turn forms a variety of sports patterns unique to snakes.

2.1.1 Snake Robot Segment Design

In our design, the mechanical snake is a multi-joint multi-degree of freedom, which can realize a mechanical structure that is bent in two vertical planes. The actuator is used to replace the muscle to generate kinetic energy. Taking into account the practical application of mechanical snakes and the installation of electronic control

Figure 2.1: Natural snake skeleton [62]
A natural snake has numerous vertebrates that enable the snake to generate motion in multiple degrees of freedom.

components, such as detection sensors for environmental monitoring and transport functions in complex geomorphology, a cavity is added between the two joints, which is the body part of the snake. The upper part shown in Fig. 2.2, has a large internal space that can be used to install the necessary electrical devices and sensors. The bottom part touches the ground as shown in Fig. 2.3, the wall of the bottom part is thicker than the upper part and can carry the weight of the whole snake robot. Assemble snake robot body has an elliptical cross section geometry to approximate the typical body shape of a snake, and prevent the snake robot from rolling during movement. There are two passive wheels,shown in Fig. 2.4, mounting groove at the bottom of the bottom part to allow the snake to pass through the smooth ground. Each passive wheel is equipped with rubber tires, shown in Fig. 2.5, which increases the lateral friction of the snake robot for better movement on smooth surface. Furthermore, a pair of passive wheels are mounted at a 45 degree angle to further increase friction. With these 4 separate accessories, we can assemble a complete snake robot body part as shown in Fig. 2.6. The short body length leaves space for electrc installation and also guarantees high degree of flexibilty. A pair of passive wheels with rubber tires increase friction for efficient locomotion on the flat plane. The final snake robot body design is 30mm in length, and has an elliptical cross section with a semi-major axis of 61.5mm, semi-minor axis of 48.08mm.

In order to achieve the gait sequence of natural snakes, the design of the joints is crucial. The joints of the snake robot must be able to implement a board set of S-shaped poses. Each joint must exhibit the ability to bend 90 degrees in both x and y plane without significant distortion or rolling. The snake should be modular, as the movement would require at least one pair of antagonistic actuators to be connected in series, so each actuator must have a connection mechanism that does not require excessive use of rigid components. To meet these requirments, servo motor DS75K

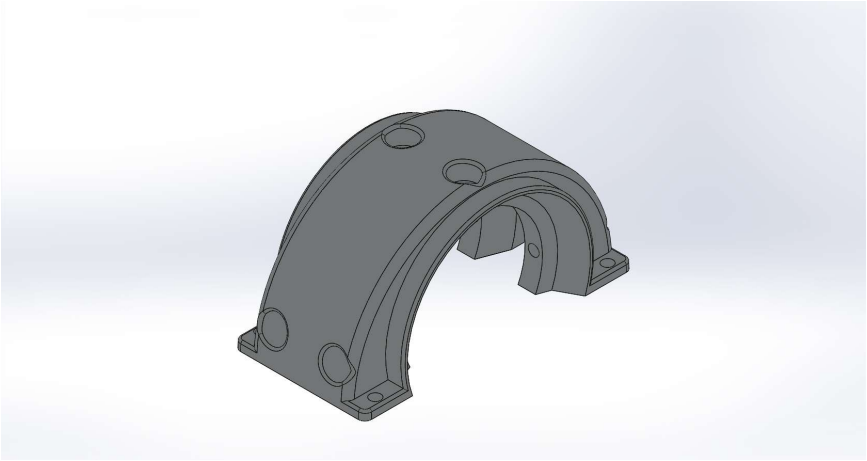


Figure 2.2: Snake robot body upper part
The snake robot upper part has large internal space for device installation.

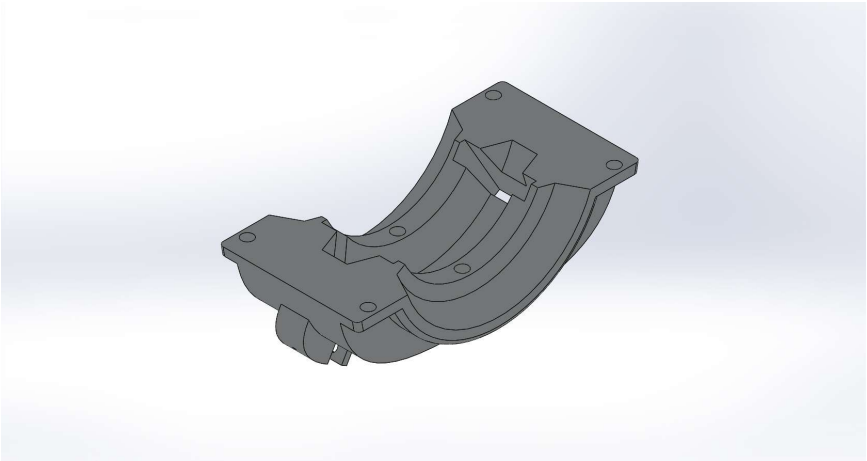


Figure 2.3: Snake robot body bottom part
The snake robot bottom part wall is thicker than the upper one to carry weight.

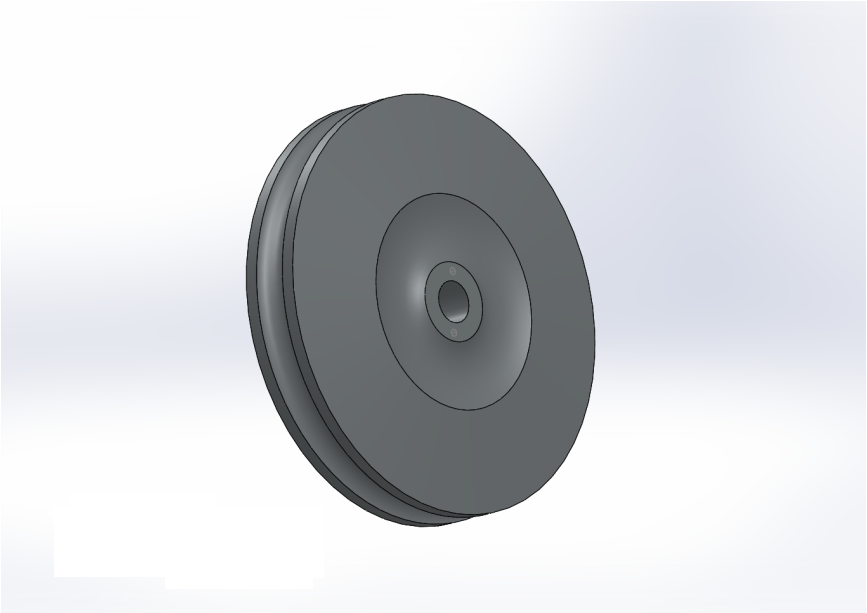


Figure 2.4: Passive wheel
A pair of passive wheels are mounted at a 45 degree angle.

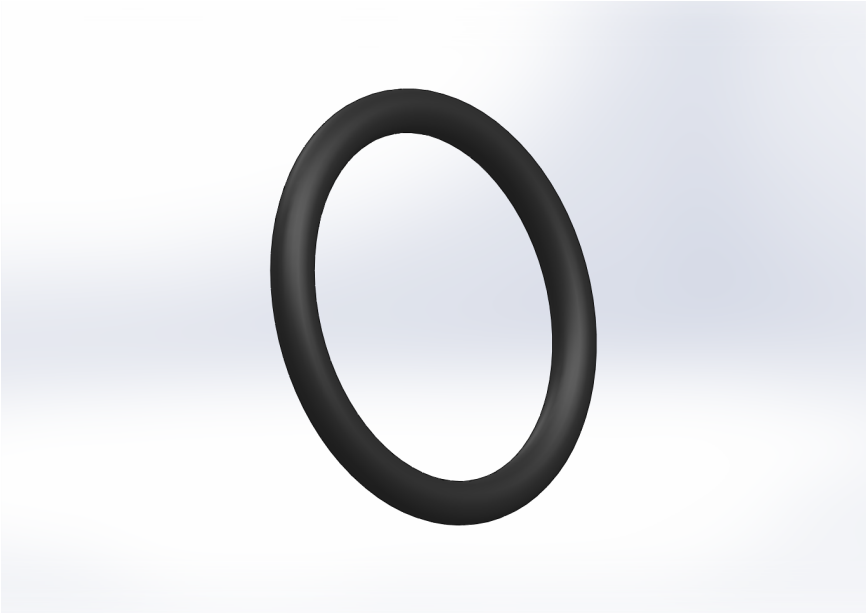


Figure 2.5: Passive wheel's rubber ring
The rubber ring is used to increase friction.

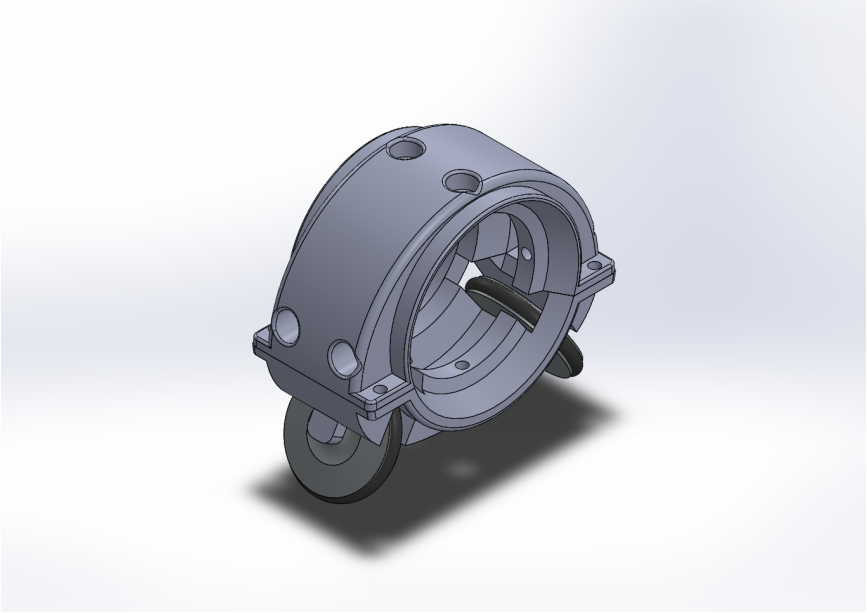


Figure 2.6: Whole snake robot body part
Short body length ensures high degree of flexibility in the movement of the snake robot. The passive wheels allow the snake robot ot move efficiently on the plane.

from MKS is selected to provide bending power to both x and y direction. DS75K is a powerful servo operating at 5V voltage with 1 cell LiPo battery and is able to provide torque of 2.4kg/cm with operating speed of 0.13sec/60 DS75K has small size of 23mm wide, 9mm thick, and 16.7mm high for easy installation. The reason why snakes can perform flexible movements with high degrees of freedom is that snakes have a lot of vertebrates. When designing snake robot joints, we try to mimic the joint characteristics of natural snakes so that the length of joints is as short as possible. Therefore, the servo motor DS75K is horizontally placed in the servomotor compartment so that the joint length is significantly reduced. The joints of snake snakes are used to connect all snake robot bodies and need to withstand the torsional forces generated by servo motor rotation. So we thicken the walls of the joints so that the joint can withstand two perpendicular directions of torque. Similar to the body design of the snake robot, the joint is modularized. It is divided into two

detachable parts, as shown in Fig. 2.7 to facilitate the installation of the transverse servo motor. Considering that the servo motor will withstand tremendous torque during the movement of the snake robot, the servomotor power transmission gear set may be damaged, so the detachable modular design facilitates installation and maintenance. In addition, the channel for passing the signal line, the power line, and the space for mounting the board is reserved in the servomotor compartment, as shown in Fig. 2.8. The reasonable electrical layout makes the signal transmission unaffected by the noise of the servomotor and also improves the stability of the operation of the electronic system. For example, an electronic system can provide a stable control output when a mechanical snake moves in different motions. The ball bearings are used on both sides of the servomotor compartment to guide the rotary motion of the cylindrical gear shaft and to withstand the load transmitted to the snake robot by the circumference.

2.1.2 Snake Robot Transmission System Design

The snake robots swing the joints in the x and y plane through the rotation of servomotors. We use the standard spur gear set to transmit the servomotor torque to the snake robot joint. Spur gear transmission is a mechanical transmission that uses two gear teeth to mesh with each other to transmit power and motion. The spur gear set has high transmission efficiency, the closed transmission efficiency is 96%∼99%, and the structure is compact which requires small space size. The snake robot requires to swing the body with high angular velocity to motion, which means that the transmission gear set needs to output the servo motor torque at a high angular speed. We choose the 1st gear as the spur gear set with the transmission ratio of 1. The secondary gear is the power transmission of the increasing gear set with the output ratio of 0.75. That is, the first gear group has the same number of teeth;

the second gear set can output a larger angular velocity. The power transmission gear set contains 4 sets of gears as shown in Fig. 2.9. Considering that the snake robot joints are only twisted within a certain angle, that is, -45°∼45°. Gears 1, 2, 3, and 4 have different numbers of teeth, which are 24, 12, 9, and 32, respectively.h are 24, 12, 9, and 32, respectively. The different gear counts of the gear set are designed to meet the power transmission requirements, making the gears tightly meshed and saving space in the servomotor compartment. As shown in Fig. 2.10, the gear 1 is coaxial with the servo motor, the gear 2 is coaxial with the gear 3, and the gear 4 is coupled to the bearings of the servomotor nacelle to drive the snake robot to twist.

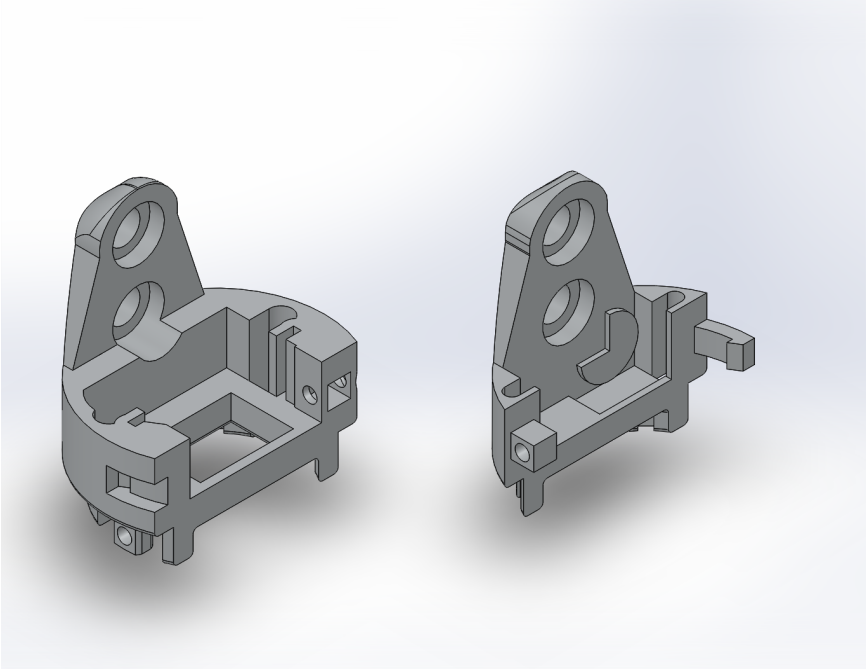


Figure 2.7: Two separate parts of servomotor compartment
Modularized servomotor compartment makes installation and maintenance easy.

The joint assembly diagram of the snake robot is shown in the Fig. 2.11. The entire snake robot consists of 8 jdentical joints and a snake robot head. Like snakes, the snake robot head contains all the necessary electronics to control body movements and sense external information. Therefore, the snake robot head is divided into two

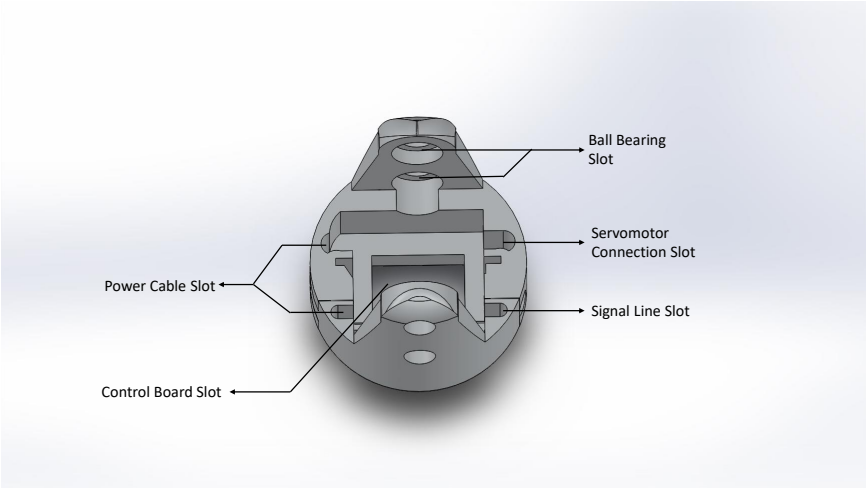


Figure 2.8: Slots of the servomotor compartment

Reasonable allocation of servomotor compartment space to install more electronic systems.

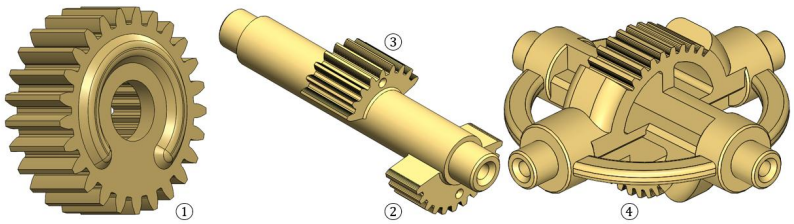


Figure 2.9: Power transmission gear set

Four sets of gear transmit power from servomotor to the snake robot joint. Gear 1 directly connects the servomotor, gear 2 and gear 3 are coaxial, and gear 4 connects the ball bearings of snake robot joint.

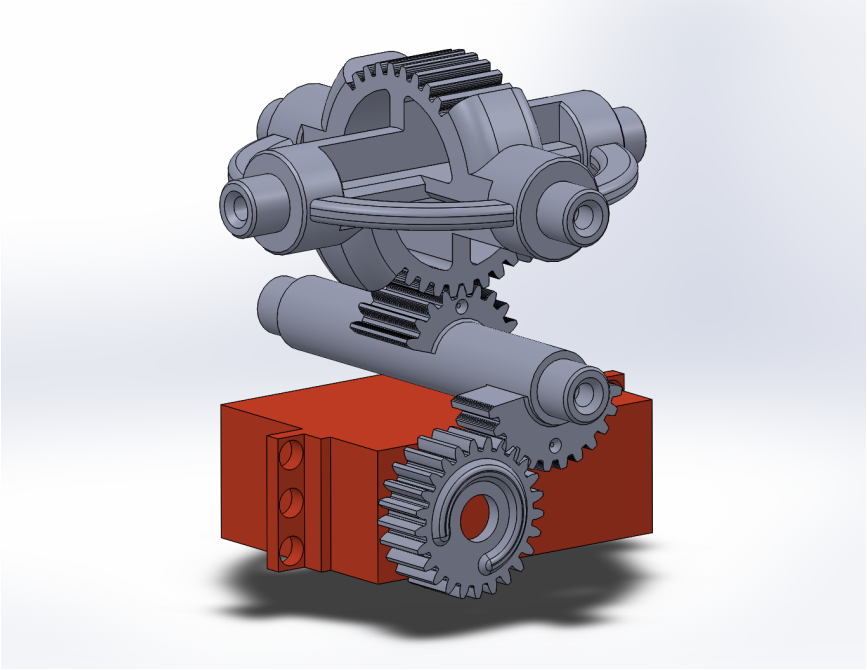


Figure 2.10: Gear sets connection
Spur gear set connection diagram.

parts as shown in Fig. 2.12. The lower part is used to install the electronic control board, and the upper part is used to connect the sensor that the snake robot senses the surrounding environment. The design of each vacancy of the snake robot head corresponds to the circuit board, which is convenient for installation and debugging. We also have a universal wheel mounting groove under the snake robot head. We abandoned the two passive wheel designs that are the same as the snake body and used a universal wheel instead, because we want to leave more space for electronics and it will not affect snake robot kinematic model.

The whole snakerobot consists eight the same joint shown in Fig. 2.13.

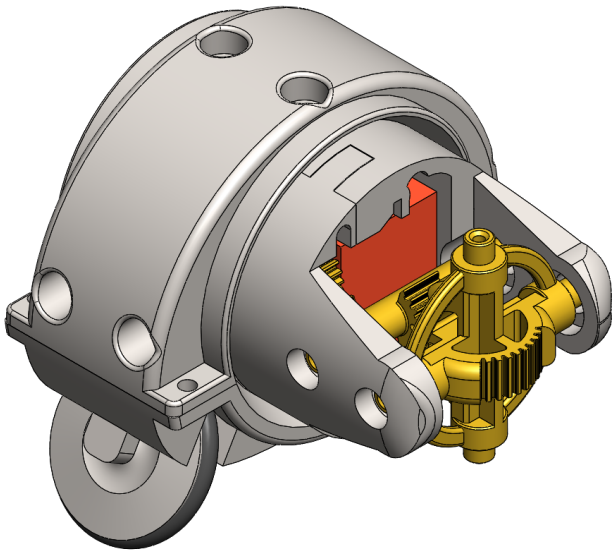


Figure 2.11: Single joint of snake robot
A snake robot consists of 8 the same joint.

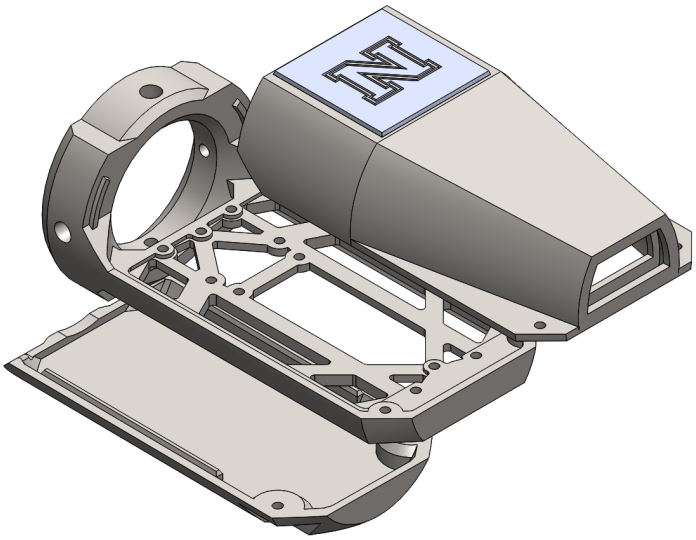


Figure 2.12: Snake robot head
A snake robot head is separated into 3 parts to contains more electrical components.

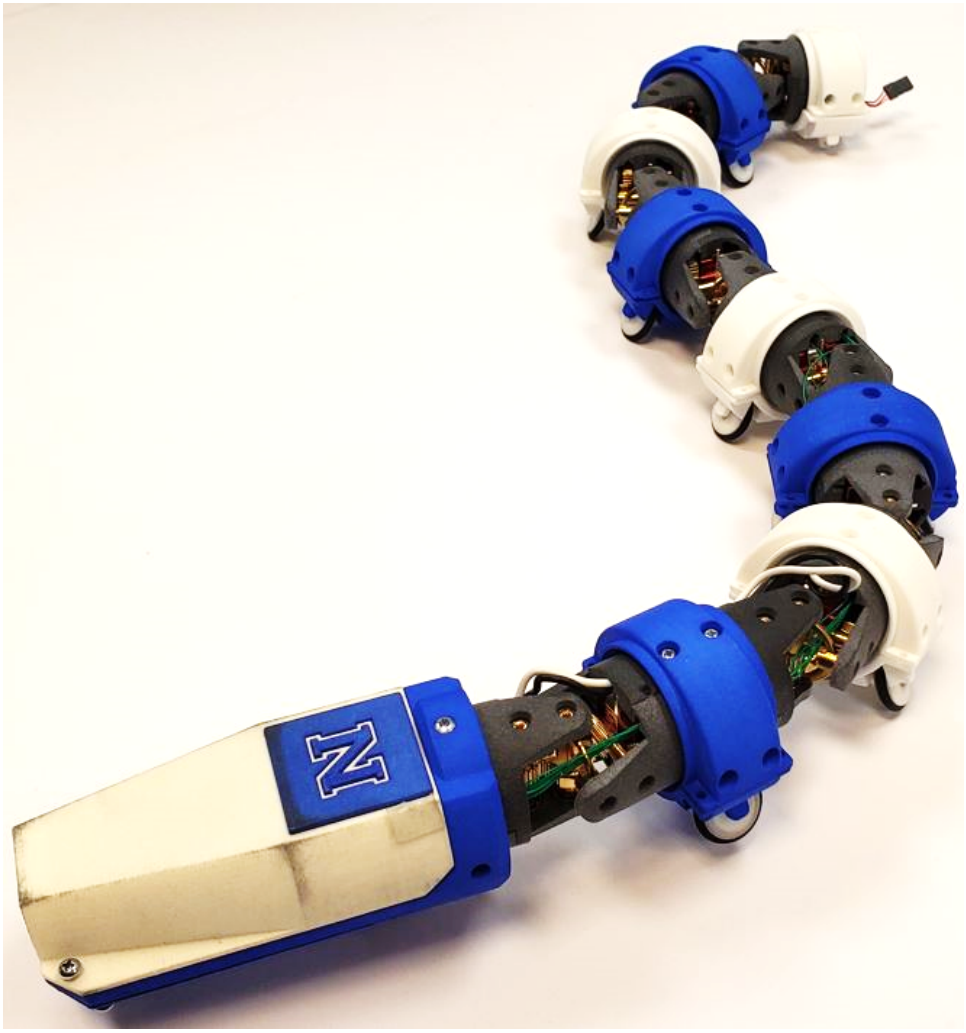


Figure 2.13: The whole snake robot
The whole snake robot consists 8 the same joins.

2.2 Snake Robot Electronic System Design

Snake robots are a class of super-redundant mechanical devices that move through internal shape changes. Due to the multi-degree of freedom of the snake robot and the compact mechanical design features, the design of the electronic control system for the snake robot has become a problem. Many factors can influence the design of electronic control systems for snake robots, such as size, power, and working environment. So we designed complex electronic systems to meet these limitations. This electronic system allows for open-loop remote control of snake robots, closed-loop automatic control, and data acquisition and transmission. Damage to one or more servomotors is also taken into account without causing the entire snake robot to smash, and this multi-redundant electronic design does not result in increased power or signal disturbances. We integrate all the necessary electronic modules on a single circuit board through intelligent circuit design and PCB layout to be mounted on the head of the snake robot. Each joint receives motion signals and transmits power through signals and power adapter boards. The integrated design and flexible routing allow the individual components on the board to work independently without interfering with each other. All the IO ports that are not used by the microcontroller are taken out, which ensures the expandability of the electronic control system. When designing the electronic control system PCB, we considered the compact space inside the snake robot. We used 3D-PCB design technology to arrange a reasonable layout for each chip and the external circuit and the quad flat package (QFP) is used as well. The 3D-PCB design technology ensures the stability of a single system by using the 3D model of the chip to properly place the external circuitry recommended by the chip manual on the board. This stable-priority and structure-compact design ensure that the various electronic modules on the chip work normally without interference. The thickness of the entire PCB is also less than 5 mm. The electronic control

board chips and electronic components all use the QFP, which poses a considerable challenge for our manual soldering. However, the machine welding can solve this problem.

2.2.1 Electronic Control Modules

Based on our requirements for microcontroller computing power, microcontroller chip resources, and microcontroller size, we chose Atmel ATSAMD21G18A-F (AMD21) as the system micro-control unit. This ARM chip has many features that we need. The Atmel® SAMD21 is a low-power microcontroller using the 32-bit ARM® Cortex®-M0+ processor, and 48 pins with 256KB flash and 32KB of SRAM. The SAMD21 devices operate at a maximum frequency of 48MHz. This MCU is designed for intuitive and straightforward migration. It can be applied to devices with the same peripheral module, hex compatible code, same linear address mapping, and pin compatible.

The Atmel SAMD21 shown in Fig. 2.14 provide the following features we need in the electronic control system design. First, it supports in-system programmable flash, which allows us to update the control program instantly. Then the programmable interrupt controller accepts interruption events to change the snake robot motion state it is moving. 48 I/O pins not only satisfy on-board electronic components control but also meet the sensors' expansion requirements. The 32-bit real-time clock and calendar enable real-time running monitoring, which reduces the probability of losing control. There are five 16-bit Timer/Counters (TC) that can be used to generate time sequences to control other modules. This MCU provides one full-speed USB2.0 embedded host and device interface for program updates. This feature also allows us to use a thin standard micro-USB port for program download instead of a bigger size of JTAG port. For connection of peripheral electronic devices, four

Serial Communication Modules (SERCOM) are available on-board that each can be configured to act as a USART, UART, SPI, I^2C. The on-board serial port always provides highly reliable data transmission, even though the software-simulated serial port is available to use with associated libraries. The Analog to Digital Converter (ADC) is also prepared in this MCU, one twenty-channel 350ksps 12-bit ADC with programmable gain is used to monitor servomotors running states. The SAMD21 has accurate and low-power external and internal oscillators that maintain a high CPU frequency while reducing power consumption. The Atmel SAMD21 supports Arduino programming, which is an open-source electronics platform based on easy-to-use software. Compatible with Arduino means the supports of many available libraries such that we can focus on logic programming. It is worth noting that the Atmel SAMD21 is woking with the voltage of 3.3V; however, other on-board devices require 5V voltage. We use a 5V to 3.3V voltage regulator SPX3819M5-L-3-3 to provide power to MCU.

The multi-degree-of-freedom motion mode of the snake robot is formed by twisting two servo motors perpendicular to each other on each joint. Our snake robot has eight joints and 16 degrees of freedom, which means we need to control 16 servo motors by Pulse Width Modulation (PWM). There are two ways to generate the required PWM; one uses 16 I/O ports to simulate the PWM signal through the system clock division. This method occupies a large number of system resources, and the accuracy of the generated PWM signal depends on the operation speed and clock accuracy of the MCU. Another method is to use the proprietary 16-channel PWM controller PCA9685, as shown in Fig. 2.15. The PCA9685 is an I^2C-bus controlled 16-channel PWM controller. Each channel output has its own 12-bit resolution fixed frequency individual PWM controller that operates at a programmable frequency from a typical of 40Hz to 1000Hz with a duty cycle that is adjustable from 0% to 100% to allow

Figure 2.14: Atmel SAMD21 MCU
Atmel SAMD21 provies many features to control a snake robot.

the servomotor to be set to a specific rotation angle. All outputs are set to the same PWM frequency. Each servomotor can be set at its individual PWM controller value, which allows the complex motion of the snake robot. This PWM controller communicates with the MCU through the I^2C port with only two wires, which are I^2C clock pin and I^2C data pin. Both of wires can either use 3V or 5V logic voltage. The maximum output current is only 25mA, so it is not a good idea to use the MCU 5V pin to power the servos. Electrical noise and brownouts from excess current draw can cause the MCU to act erratically, rest or overheat. External power source with maximum 10A current supply is used to power all 16 servos. In our latest design, a snake robot consists of eight joints with 16 servers. In the case of more joints with over 16 servos required, multiple PWM controllers (up to 62) can be chained to control still more servos with I^2C port by assigning a unique address to each controller.

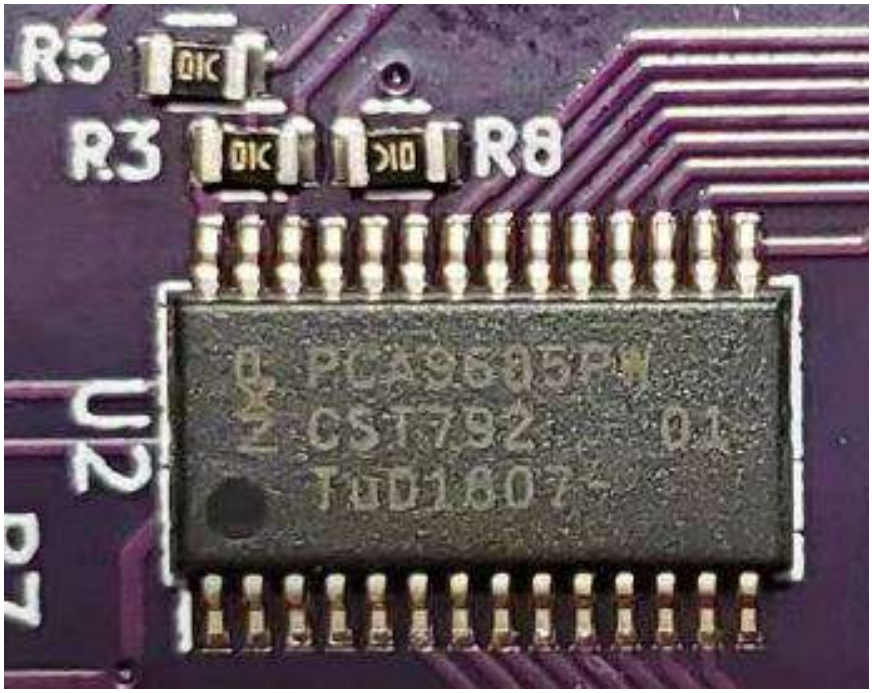


Figure 2.15: 16-channel PWM controller
Each servo is controlled separately using the PWM controller.

The environment in which a snake robot works may be a GPS-denied closed environment, such as collapsed house ruins, closed pipes, and caves. In these GPS-denied environments, snake robots can only perceive external environments through their sensors, such as vision systems and inertial measurement units (IMU). We have reserved space for the vision system at the head of the snake robot, and the IMU is integrated into the control circuit. We chose BNO055 as the IMU to provide attitude information for the snake robot, as shown in Fig. 2.16. BNO055 is a smart sensor with 9 degrees of freedom, which can output three-axis orientation based on a 360-degree sphere, three-axis angular velocity, and three-axis acceleration. The most attractive point is that BNO055 does not need to process the collected data; it can do sensor fusion itself so that we can get useful pose data directly. BNO055 uses the I^2C port for data transmission. Thanks to the rich scalability of the MCU, we can use the I^2C port on MCU to collect data from this IMU. If the later added sensors also need to take the reuse I^2C port, we can also control different components by assigning unique addresses.

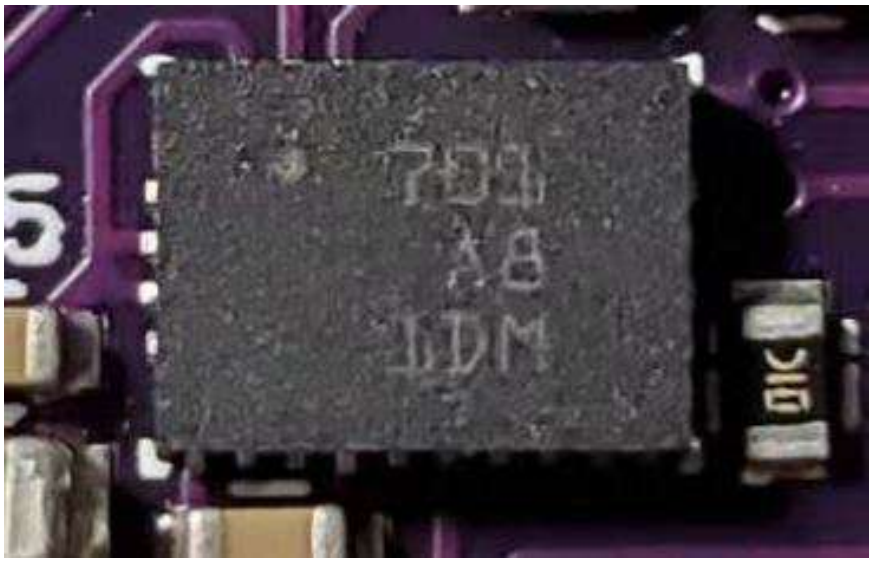


Figure 2.16: 9 DOF smart absolute orientation sensor
BNO055 provides orientation, angular velocity and acceleration information.

The fourth major chip is a wireless transmission chip, Digi XBee3 module, as shown in Fig. 2.17, which is used for the command release of open-loop control and transmission of information in closed-loop control. The Digi XBee3 has the size of 13mm*19mm. With its reduced size, weight, and power consumption, it is ideal for wireless transmission in a compact snake robot head. Digi XBee3 offers a low-power microcontroller capable of multi-level programmability, including dual-mode radios, multi-module instant networking, and switching between a variety of protocols. It also has the feature of easy over-the-air (OTA), which changes to devices in the field for bug fixes and new features. OTA function also supports programming the MCU wireless. Snake robot has many locomotion modes and running in complex environments, which requires different communication configurations. Digi XBee3 can be dynamically reconfigured based upon the situation. In addition, it has built-in support for advanced I/O, including I^2C, SPI to drive sensors, and actuators. The connection between XBee and the MCU is established by serial port. Although the MCU has an additional serial port for XBee module, the PCB only has one micro-USB port. Such that we have to program MCU and XBee separately. Furthermore, Digi XBee3 has a perfect development kit and Arduino library, which saves us time in snake robot development.

2.2.2 Electronic Design of the Snake Robot Control Modules

The previous section introduced the design of the snake robot electronic control system, including Atmel-SAMD21 MCU, 16-channel PWM controller PCA9686, 9 DOF smart absolute orientation sensor BNO055 and wireless transmission chip Digi XBee. All chips can generally work without interference through intelligent circuit design and the elegant PCB layout. Each chip has a peripheral circuit design recommended by the chip datasheet. We designed the peripheral driver circuit for each chip in

Figure 2.17: Digi XBee module allows for wireless transmission
XBee module supports OTA function benefiting for snake robot reconfiguration.

strict accordance with the requirements of the chip manual and tested it through the breadboard before integration. After completing the test, we use Altium Designer to draw the circuit schematic and perform the PCB layout. PCB layout is the most precise and most technically demanding in the circuit design process. A separate system usually works, but you never know what happens when you integrate multiple systems. Here are a few of our techniques in the PCB design process to ensure the stability of the snake robot controller. Fortunately, the chips we choose are all low-frequency digital chips that don't cause too much interference.

The MCU is the core device, and its peripheral circuit is shown in the Fig. 2.18. We placed the MCU on the top of the PCB near the Micro-USB interface, and the crystal oscillator was placed next to the MCU to ensure the stable operation of the MCU. A decoupling capacitor is added between the MCU power supply and the ground line to provide a large current for the chip and to remove noise. It is to affect the chip with as little power noise as possible, and the noise generated by the chip

does not affect the power supply. All the pins that are not used by the MCU are taken out and placed on both sides of the PCB. On the left side is the digital IØ, and on the right is the ADC interface. The digital signal and the analog signal must be separated because the analog signal is highly sensitive, and the high-frequency digital signal dramatically affects the quality of the analog signal. In order to ensure the scalability of the snake robot electronic control system, we have introduced an I^2C interface and an SPI interface. The chip power supply and the servomotor supply use the same power interface in parallel. We use the SPX3819M5-L3-3, shown in Fig. 2.19, to generate 3.3V to power all chips on board. All ground and power lines are widened to meet onboard power requirements.

The schematic diagram of XBee is shown in the Fig. 2.20. The wireless transmission chip XBee uses serial port 1 (pin 3 and pin 4) to communicate with the MCU. The MCU supports four serial ports for simultaneous communication. However, there is only one Micro-USB interface on the PCB. Considering that the configuration of XBee and MCU serial communication does not occur at the same time, we connect the MCU serial port 0 and the XBee serial port 1. In order to configure XBee to the appropriate parameters, in the Arduino program, we use the example SerialPassthrough to achieve the purpose of configuring XBee on the computer. The SerialPassthrough example is to pass the data read by the MCU serial port 0 to the XBee serial port 1. In order to prevent the data read by the MCU serial port 0 from affecting the transmission and reception of XBee data, all the interfaces except the serial port 1 and the power cable are grounded.

IMU, as shown in the Fig. 2.21, as the most critical attitude feedback device for snake robots, which needs a good operating environment. The IMU is placed in the center of the PCB, close to the wireless transmission chip XBee. Placing the IMU in the center of the PCB ensures that IMU data is not affected by I/O. The short

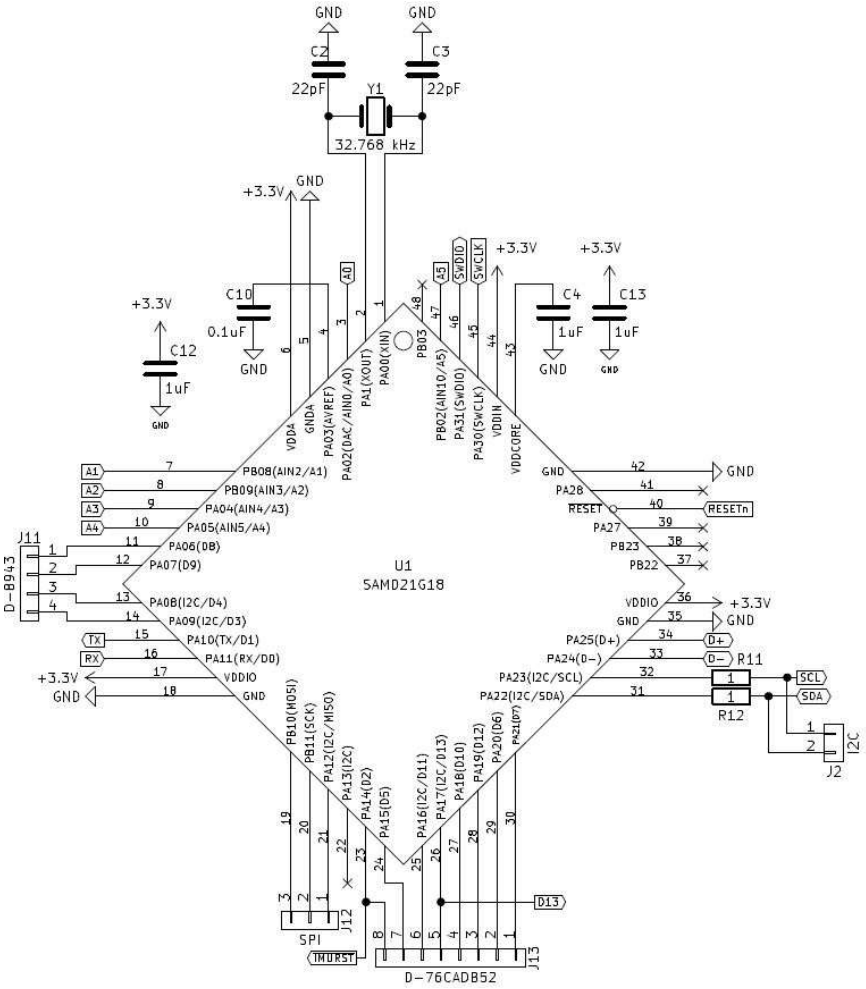


Figure 2.18: MCU peripheral circuit schematic
Precise wiring ensures stable operation of the MCU.

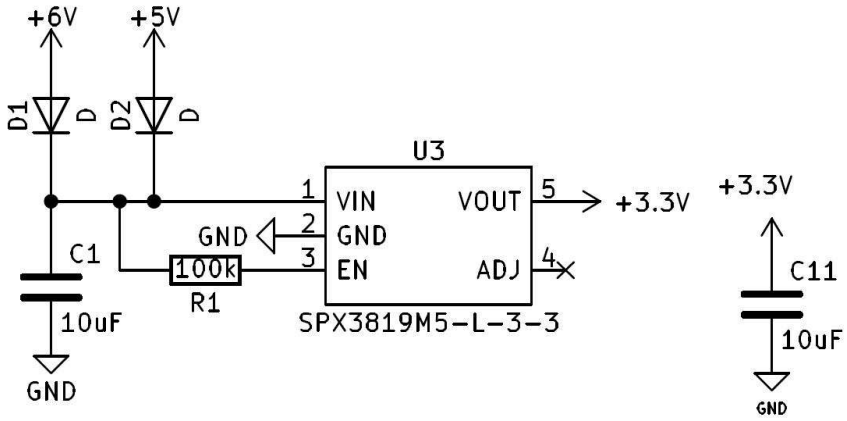


Figure 2.19: SPX3819M5-L3-3 voltage regulator
5V voltage input is regulated to 3.3 for MCU power supply.

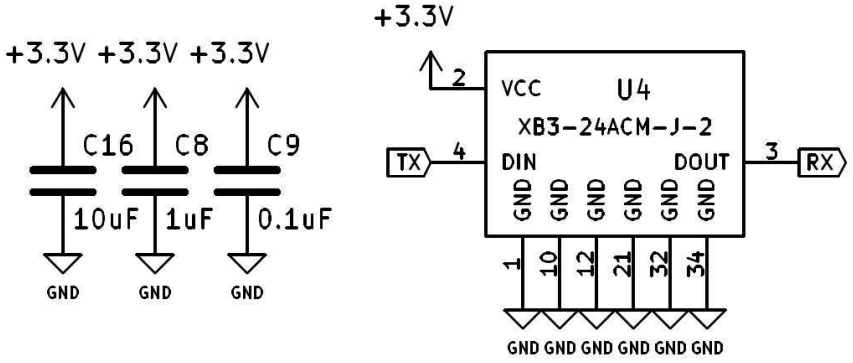


Figure 2.20: Wireless communication chip XBee peripheral circuit
In case of disturbances from serial port, all other pins are grounded except serial port and power lines.

distance from the XBee chip ensures the reliability of data transmission. The crystal oscillator is placed next to the IMU to avoid other signal interference from the clock signal.

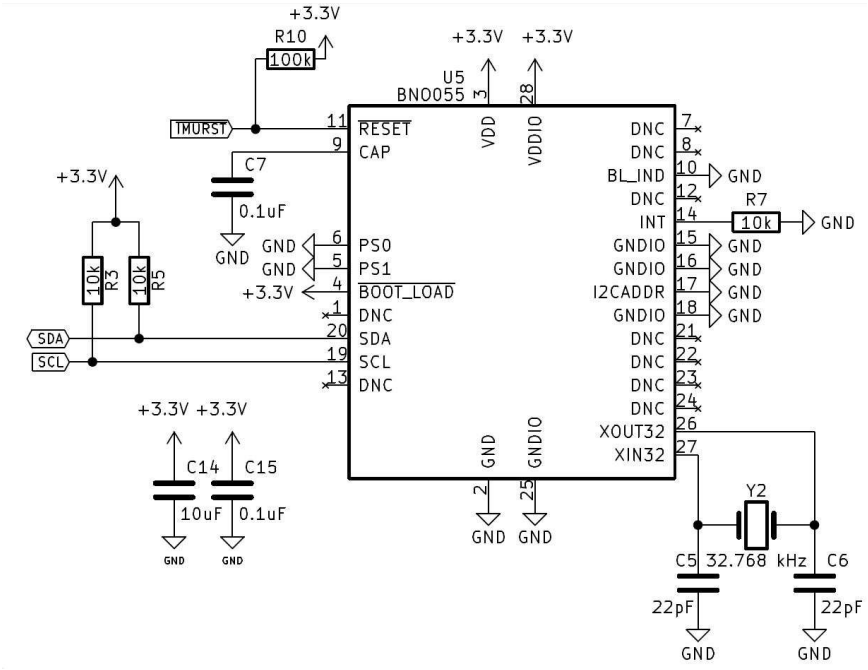


Figure 2.21: BNO055 absolute orientation sensor
IMU is reasonably placed to reduce the impact of interference on snake robot attitude data.

PWM schematic is shown in Fig. 2.22. As the final execution chip of the control command, the PWM controller is placed at the end of the PCB to facilitate the DuPont cable connection. A snake robot consists of 8 joints with 16 servos, thus we only put one PWM controller on PCB.

Taking into account the characteristics of each chip, we carefully arranged the location of each chip. And through proper routing, the interference between the various modules is significantly reduced, so that the entire system as a snake robot controller can operate normally. As mentioned before, the use of QFP for all chips and electronic components makes manual soldering more difficult. Although we use

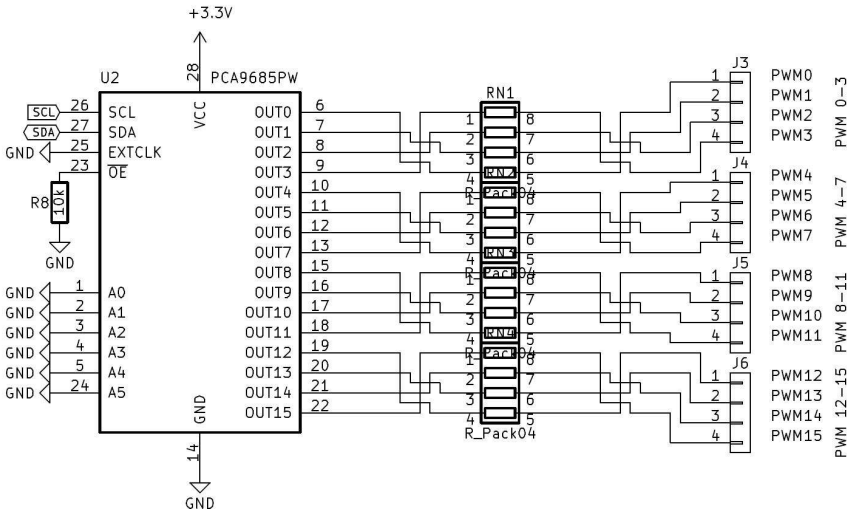


Figure 2.22: BNO055 absolute orientation sensor
IMU is reasonably placed to reduce the impact of interference on snake robot attitude data.

a mold to brush each pin and then heat-weld, it still has a success rate of 80%. PCB size and chip position is shown in Fig. 2.23, PCB layout is shown in Fig. 2.24 (a). After the soldering is completed, the PCB is shown as Fig. 2.24 (b).

The electrical connection of each joint is also achieved through the PCB. Compared to the controller PCB of the snake robot head, the electrical connection PCB is much simpler. The main function of the connection PCB is to transmit power and control signals for each servo motor. Each connection PCB has two sets of power interfaces and four sets of PWM interfaces for controlling four servo motors of the two joints. The schematic of the connection board is shown in the Fig. 2.25. PCB layout as shown in Fig. 2.26.

2.2.3 Softwares and Programming Environment

The snake robot control system software design is divided into two layers, the low-level chip driver and the high-level algorithm implementation. For the low-level chip driver programming, we use Arduino integrated development environment (IDE) to

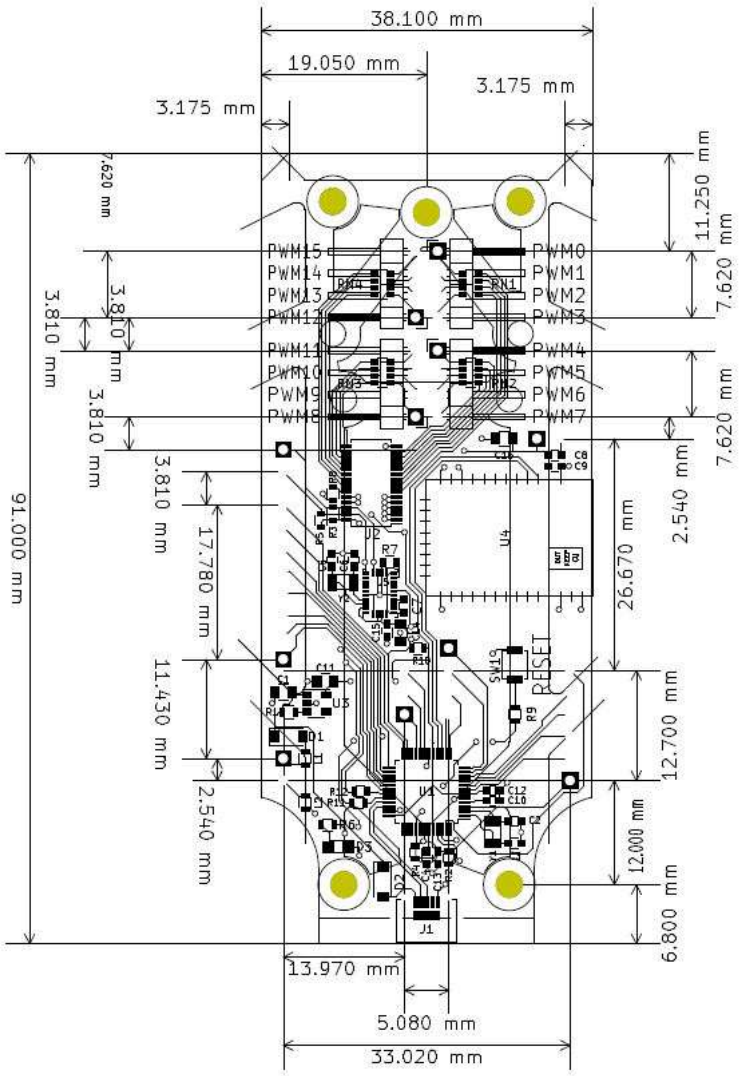


Figure 2.23: PCB sanity
This figure shows the size of each element, and how we place those electronic elements.

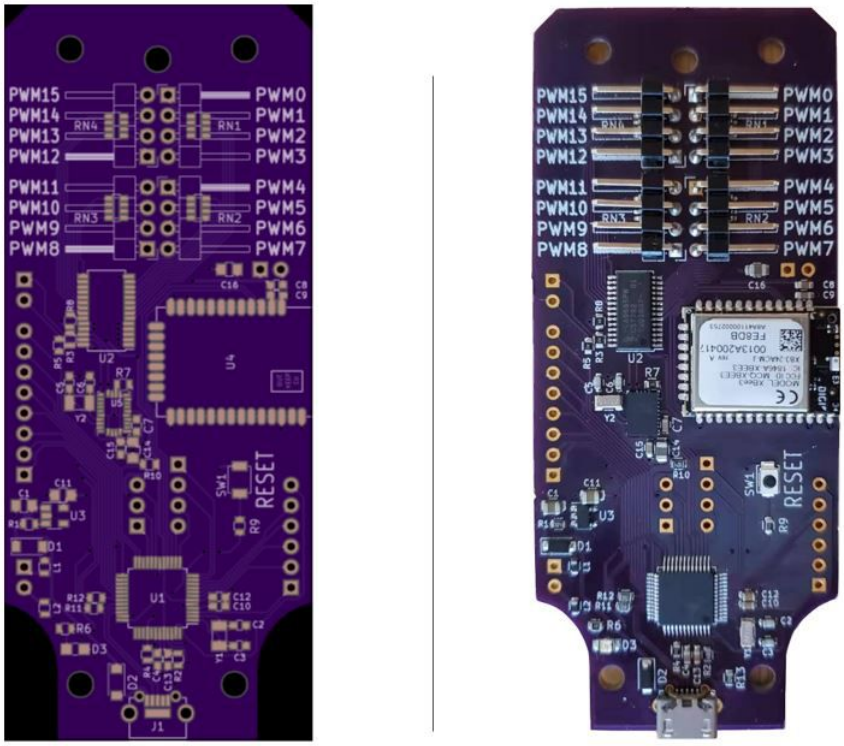


Figure 2.24: Snake robot control PCB design
(a) is the PCB layout is carefully designed for the special working environment of the snake robot. (b) is the completed PCB, Each element is QFP, making the thickness of the entire PCB less than 5mm.

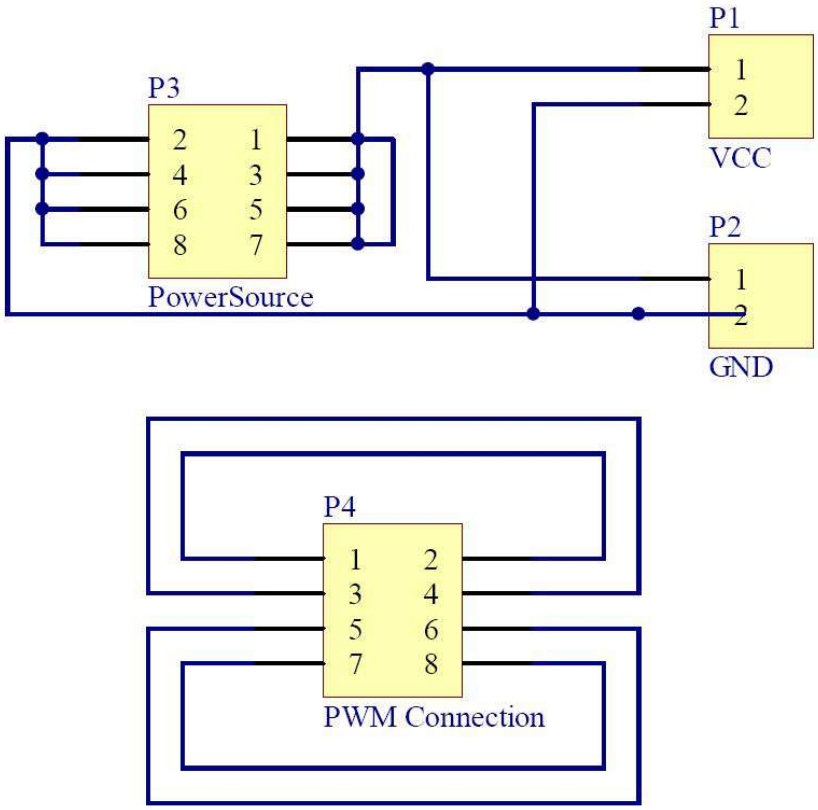


Figure 2.25: The connection board schematic
This board is used to trnsmit power and control signal to servos.

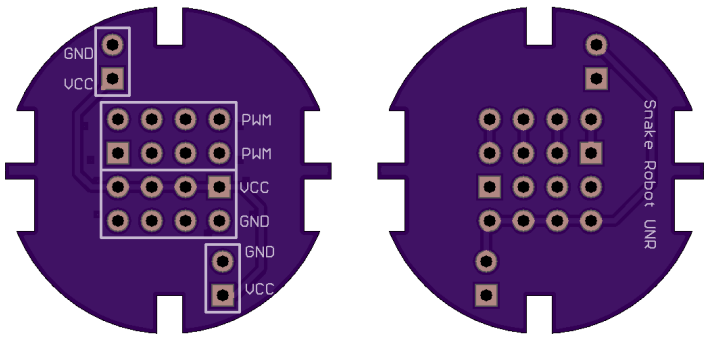


Figure 2.26: The connection board PCB layout
This board is used to trnsmit power and control signal to servos, it fits the shape of servomotor compartment.

code each chip. Arduino is an open-source electronics platform based on easy-to-use hardware and software. The Atmel-SAMD21 is campatible with Wiring [51] based arduino programming language and Processing [46] based Arduino software. Arduino IDE allows us easily program the I/O of the hardware with the supports of many libraries. For developers, it is a good thing to use the chip's functionality without having to spend extra time debugging individual modules. For example, we use the library "AdafruitPWMServoDriver" to control all 16 channel servos. We use the library "AdafruitBNO055" to read data from the IMU BNO055. We reasonably allocate MCU resource to a different module, MCU-SAMD21 can use these libraries to control each module in a time-sharing manner. Wireless transmission chip Digi XBee uses one serial port to communicate with others. The XBee also has the support of the Arduino library. However, we do not use it for necessary wireless communication. A software named XCTU is used for XBee configuration. XCTU has unique features like graphical network view, which graphically represents the XBee network along with the signal strength of each connection. Such that we can

config a communication network for a snake robot through the XBee easily.

Another software essential for the snake robot closed-loop control application is Robot Operating System (ROS) [25]. ROS is robotics middleware, which provides service designed for heterogeneous computer cluster such as low-level device control. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor data, control, state, planning, actuator, and other messages. ROS can be employed in multiple platforms such as Linux, Windows, and some SOC systems, supporting control of all devices of a snake robot through node message; even ROS is not a real-time operating system. We decide to deploy ROS in Ubuntu operating system for high-level path planning algorithm calculation. Benefit from the high-speed computation of the laptop and reliable wireless communication, a snake robot can use many high-level path planning and navigation algorithms to motion in various environments. Since ROS is an open-source operating system, ROS has a number of extensions that can be used to scale applications as needed. For example, Gazebo [26] for simulation, YoLo3 [52] for visual recognition, and TensorFlow [1] for machine learning. Depending on the application of the snake robot, we can use different expansion packs to enhance the task adaptability of the snake robot.

The snake robot's trajectory can be obtained by the motion tracking system, the Motive OptiTrack. OptiTrack is a real-time motion capture system with multiple cameras and components, offering high-performance optical tracking. OptiTrack can obtain the real-time robot's coordinates in x, y, and z-axis, and also the yaw, pitch, and roll angle data with a frequency of 120Hz. OptiTrack has a professional community to support and develop the software platform of this system. ROS package is available to collect data and use it in the application algorithm. So with the OptiTrack, we can easily take robot's coordinates in real-time for the path follow-

ing algorithm. In the later LIDAR-based path following experiment, the trajectory captured by the OptiTrack is used as path reference.

2.3 Snake Robot Locomotion Performance Validation

The serpentine locomotion is the most common snake movement. We can simulate the snake's wave motion by changing the relative angle of the snake robot, as follows:

$$\phi_i = \alpha sin(\omega t + (i-1)\delta) + \phi_0 \tag{2.1}$$

where $\alpha, \delta, and \phi_0$ are the parameters that determine the shape of the serpentine curve realized by the snake robot, and ω specifies how fast the serpentine wave propagates along the body. Our hypothesis is that the serpentine gait is achieved when the shape change of (2.1) is coupled with the environment through directional friction characteristics. In this section, we experimentally verify that this assumption is correct. Another purpose here is to confirm Hirose's [17] observation of the snake robot with wheels: the speed and direction of the snake-shaped movement are mainly determined by ω and δ.

The experiment has been conducted to verify the serpentine locomotion mode. We consider the snake robot with $N = 8$ links of mass $m = 0.135$kg. The model parameters are adopted based upon the snake robot in the Robotics and Biomimetics Laboratory of University of Nevada. We choose the gait parameters in (4.51) as $\alpha = 0.1$m, $\omega = 120°/$s, and $\delta = 40°$. The experiment snapshot is presented in Fig. 2.27. It can be seen from the figure that the snake robot has always maintained a complete sinusoidal curve during its movement, pushing the snake robot forward. In order to observe the motion trajectory of the snake robot more intuitively, we

have added marks to the first joint, the second joint and the last joint of the snake robot, and drawn the motion trajectory of the snake robot as shown in Fig. 2.28. Point 1 and point 3 are the markers on the first two joints, point 2 is the snake robot tail trajectory. It is obviously shown with the motion curve. Point 1 and point 3 basically are sinusoidal waves with the same amplitude and frequency. And they have different angle shift δ as described in (2.1). Point 2 has the most different features compared with others. The point 2 curve also obeys a sinusoidal manner, however, with different amplitudes. The reason is the snake robot tail does not have the physical constraints as other joints. Such that, slide slip happens throughout the locomotion on the last joint. Other two figures Fig. 2.29 and Fig. 2.30 present the x and y coordinates of the motion. A snake robot shows different features in x and y plane. We can conclude the high-efficiency motion along the x axis of the snake robot due to the rising curves of x without any drawback, which indicates a movement of continuous progress. The curve at y axis shows the sinusoidal motion manner.

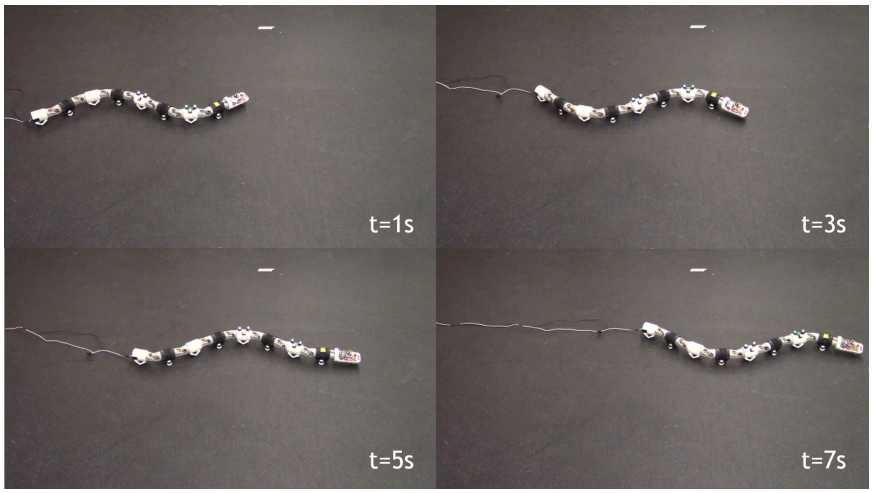


Figure 2.27: Snapshot of snake robot serpentine locomotion

The next experimental validation conducts another widely observed snake locomotion mode, the sidewinding locomotion. In the sidewinding mode, the snake robot

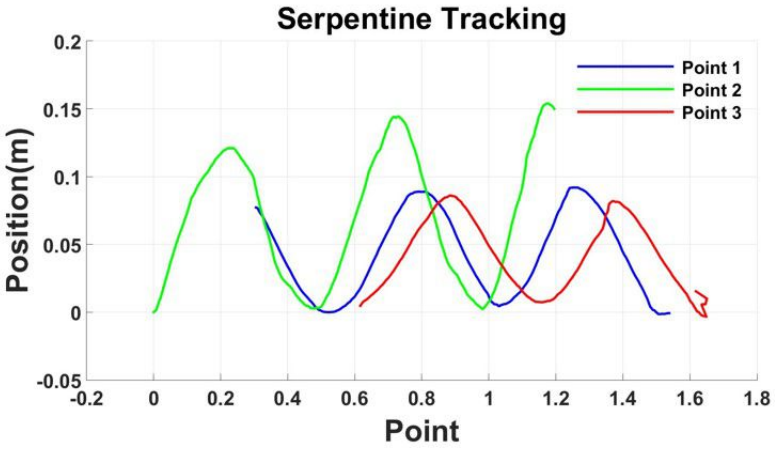


Figure 2.28: Trajectory plot of serpentine locomotion

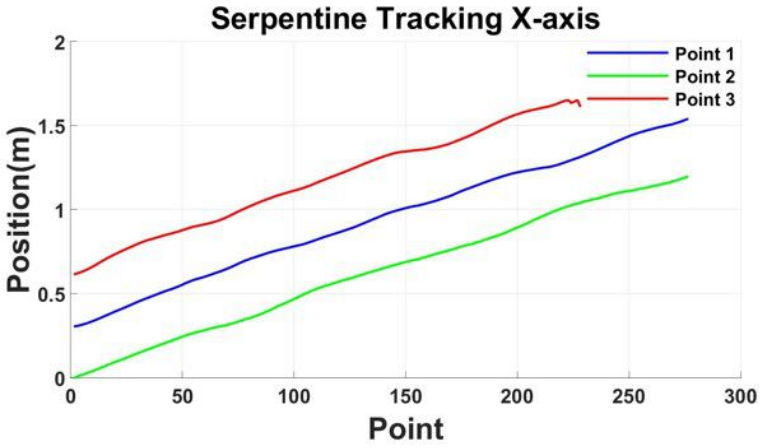


Figure 2.29: x coordinates of the serpentine locomotion

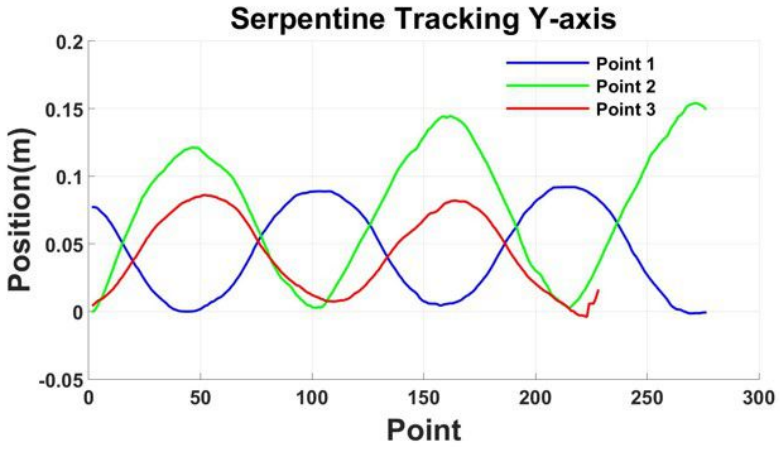


Figure 2.30: y coordinates of the serpentine locomotion

maintains two to three static contact (local rolling/peeling) with the substrate at all times. During this movement, each segment of the robot is gradually laid to the ground contact, peeled off into an arched segment, and then transferred forward into a new ground contact segment. The sidewinding movement mode only requires two or three points of the snake robot to touch the ground, and the snake robot is moved forward by the body's swing in 3-dimension space. Sidewinding is suitable for snakes moving on the unstructured ground, and it also gives snake robots a certain obstacle crossing function. During motion control, we assign each joint in the x plane a sinusoidal wave, while assigning a cosine wave to the joints bending in the y plane. Such that the snake robot is able to move in 3-dimension space. The snapshots of the snake robot sidewinding locomotion are shown in Fig. 2.31. Unlike serpentine locomotion, the sidewinding motion efficiency depends on the contact points. When the snake robot swings the body, the contact points with physical constraints generate anisotropic friction force to drive the snake robot to move side by side. It is clearly depicted in Fig. 2.32; the snake robot head has the lowest displacement because the head only has a universal wheel to provide anisotropic friction, while other joints have two passive wheels split with 90° to provide bigger friction force. The

sidewinding locomotion efficiency also depends on the ground structure. The sand ground or other ground with a high friction coefficient can provide anchor points for snake sidewinding, that is why we can normally see the sidewinding locomotion in the desert.

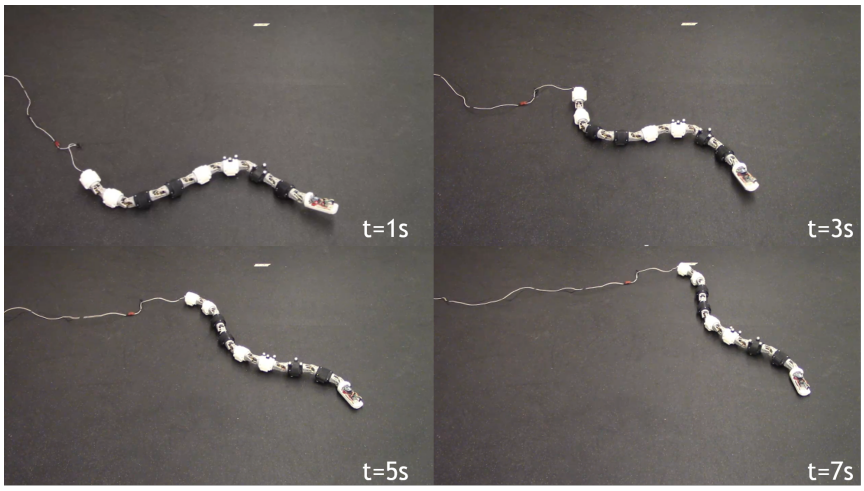


Figure 2.31: Snapshot of snake robot sidewinding locomotion

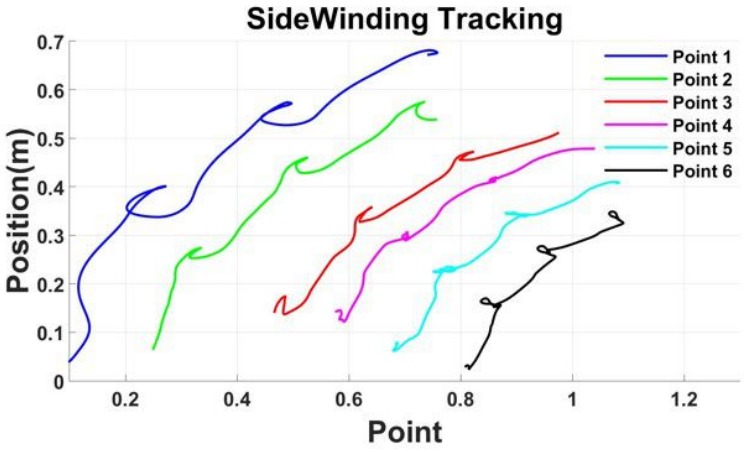


Figure 2.32: Trajectory plot of sidewinding locomotion

In this chapter, we introduce the model design of the snake robot, including the body, the servomotor compartment, and the transmission system. The overall design

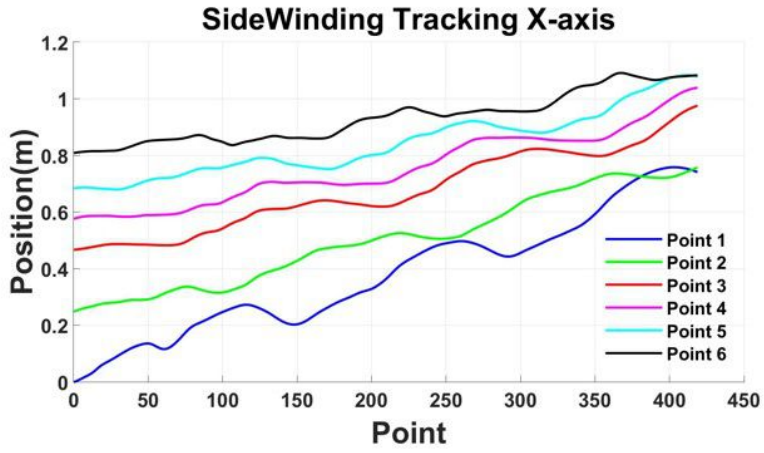


Figure 2.33: x coordinates of the sidewinding locomotion

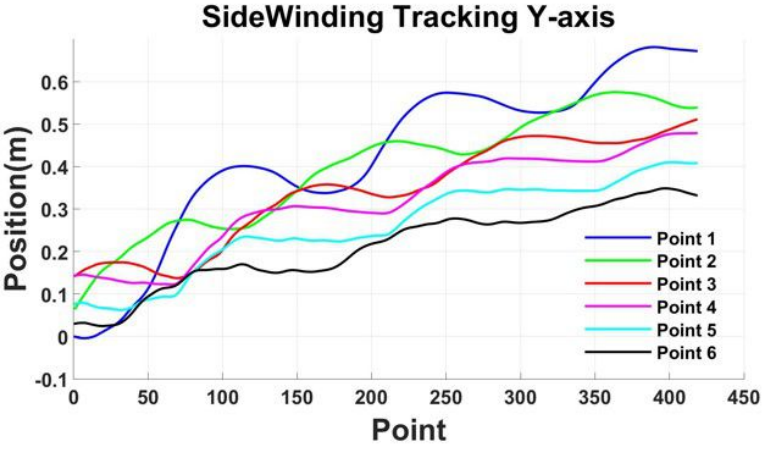


Figure 2.34: y coordinates of the sidewinding locomotion

principle follows the joint characteristics of natural snakes, making each joint of the snake robot as short as possible, which can generate more freedom of movement, and the snake robot can be more flexible. Shorter joints mean shorter moments to reduce wear on the servomotor, especially when the snake robot moves quickly. Each component is modular and detachable for easy maintenance and installation of the snake robot. The design of the snake robot electronic control system also follows the principle of high integration, so that the electronic control PCB is only 5 mm thick and can be mounted to the snake robot head. Each subsystem is reasonably placed in different positions of the PCB according to its characteristics, and the subsystems are allowed to work without interference by precise wiring. The external connection of multiple sets of interfaces guarantees the scalability of the snake robot's electronic control board to meet the needs of future functional expansion. We use the open-source software platform Arduino to complete the low-level driver programming of each subsystem, using ROS to achieve path-planning, object tracking, path following, and other high-level algorithms. We have mastered the hardware design and software operation of the snake robot so that we can make corresponding changes according to the needs of the task. We have mastered the hardware design and software operation of the snake robot so that we can make corresponding changes according to the needs of the task and enhance the adaptability of our snake robots under different application requirements. In order to better control the customized snake robot, experiments have been conducted to validate the snake robot locomotion efficiency which laid the solid base for further control. In the next chapter, we will introduce the kinematics modeling and dynamic modeling of the customized snake robot.

Chapter 3

Snake Robot Kinematics and Dynamics

The contribution of this dissertation is to solve the problem of path following control of a class of planar underactuated snake robots without nonholonomic velocity constraints by using virtual holonomic constraints. We constrain the state evolution of the system to a metaspace submanifold, called a constrained manifold. The manifold is defined by the geometric relationship between the generalized coordinates of the system. This geometric relationship is called a virtual holonomic constraint. The proposed feedback control law is designed to be an exponentially stable constrained manifold, *i.e.*, to perform a virtual holonomic constraint so that the snake robot can converge to the desired path. As far as we know, model-based motion control of snake robots based on a formal stability proof has received little attention in the literature. In this dissertation, we use a non-simplified, more complex, and more accurate snake robot model, and utilize virtual holonomic constraints to design path following control for a class of planar underactuated snake robots. In the following chapters, we propose a model-based straight-line path following control method, a curve path following control method, and a perception-aware path finding and path

following method. All three control methods rely on accurate snake robot kinematics and dynamics models. In this chapter, we derive the kinematic model and equation of motion for the snake robot under the Lagrangian framework.

3.1 Kinematics

The kinematics of the planar snake robot is illustrated in Fig. 3.1 [54]. We use partial feedback linearization to transform the model to a simpler form for model-based control design. In order to achieve the control objectives, we write the control equations of the system in an implementable form. This is usually done by selecting a local coordinate map and writing a system equation for these coordinates. According to the illustration of the snake robot in Fig. 3.1, we choosse the generalized coordinate vector of the N-link snake robot as $x = [\phi_1, \phi_2, \cdots, \phi_{N-1}, \theta, p_x, p_y] \in R^{N+2}$, where $\phi_i \in \{1, \cdots, N-1\}$ denotes the i^{th} joint angle, θ denotes the snake robot head angle, and the pair (p_x, p_y) are the coordinates of the Center of Mass(CM) of the snake robot with respect to (w.r.t) the global $x-y$ axes. Using these coordinates, we can specify the kinematics of the robot. We select the first N elements of x, namely $(\phi_1, \phi_2, \cdots, \phi_{N-1}, \theta)$, to describe the angular coordinates. The shape variables, *i.e.* $\phi_s = (\phi_1, \phi_2, \cdots, \phi_{N-1})$, describe the internal structure of the robot and are directly controllable. The position variables, *i.e.* $p_s = (\theta, p_x, p_y)$, are passive DOF of the system. The coupling between the shape variable and the position variable results in a displacement of the position variable, according to the cyclic motion of the shape variable, *i.e.* the *gait pattern*. The snake robot velocities are v_n and v_t in normal direction and tangential direction respectively.

The snake robot moves on a horizontal plane with a total of $N+2$ degrees of freedom. We define the robot's motion relative to the two coordinate systems shown in Fig. 3.1. The $x-y$ coordinate system is a fixed global coordinate system. The

$v_n - v_t$ frame is always aligned with the snake robot, that is, the v_t axis and v_n axis always point to the tangential and normal directions of the robot, respectively. The origins of both coordinate systems are fixed and coincide. We indicate the tangential or positive robot direction of the v_t axis, and the normal direction of the v_n axis. Note that the $v_n - v_t$ frame is not fixed to the robot body. However, if a body frame fixed to the robot is defined, its orientation will be the same as that of the $v_n - v_t$ frame.

The position of the snake robot is described by the coordinates of its center of mass in the global frame p_x, p_y. The direction of the global coordinate system of the robot is represented by the angle that the rotates from the positive direction of x-axis in a counterclockwise direction. The angle between the global x axis and the v_t axis is also the same, because the $v_t - v_n$ coordinate system is always aligned with the robot. The position described in a frame that is always aligned with the snake robot is inspired by a coordinate transformation similar to that of the robot.

The relationship between the $v_t - v_n$ frame and the global frame is given by

$$\begin{aligned} p_t &= p_x cos\theta + p_y sin\theta \\ p_n &= -p_x sin\theta + p_y cos\theta \end{aligned} \tag{3.1}$$

The forward velocity v_t and normal velocity v_n of the CM of the snake robot can be written as

$$\begin{aligned} v_t &= \dot{p_x} cos\theta + \dot{p_y} sin\theta \\ v_n &= -\dot{p_x} sin\theta + \dot{p_y} cos\theta \end{aligned} \tag{3.2}$$

Differentiating (3.1) with respect to time and insering (3.2) gives

$$\begin{aligned} \dot{p_t} &= v_t + p_n \dot{\theta} \\ \dot{p_n} &= v_n + p_t \dot{\theta} \end{aligned} \tag{3.3}$$

We rewrite equations (3.1), (3.2, and (3.3) to obtain the snake robot kinematic model

$$\begin{aligned} \dot{\theta} &= v_\theta, \dot{p}_x = v_t \cos\theta - v_n \sin\theta, \\ \dot{p}_y &= v_t \sin\theta + v_n \cos\theta, \dot{v}_\theta = -f_1 v_\theta + \frac{f_2}{N-1} v_t \bar{e}^T \phi \end{aligned} \tag{3.4}$$

The heading angle is 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.

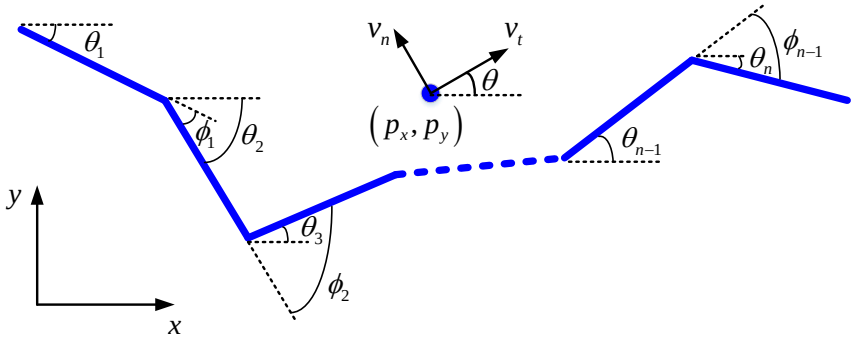


Figure 3.1: Snake robot kinematic model
An illustration of the N-link snake robot. Kinematic parameters of the snake robot.

Most of the literature on snake robots and similar movable multi-body robot structures (such as eel robots) used the Newton-Eulerian formula to derive the equations of motion, *i.e.* where the equations of linear motion and angular motion of the individual link are described separately [57]. This is due to the fact that it is usually difficult to integrate the anisotropic external dissipating forces acting on these complex robotic structures, *i.e.* the ground friction, into their Euler-Lagrange equations. However, ground friction has been shown to play a vital role in the movement of snake robots. In this dissertation, we derive the equations of snake robot motion in a Lagranigian framework, *i.e.* we analyze the snake robot motion curve with sinusoidal curvature changes along the body axis [19, 66]. The change in the curvature of the

snake robot motion curve is described with the following equation.

$$k(s) \quad = \quad \frac{2K_n\pi\phi}{L} sin \frac{2K_n\pi}{L} S \ , \tag{3.5}$$

where K_n is the number of the sinusoidal shape (S-shape); L is the wavelength of the curve; ϕ is the joint angle. Thus, we can describe the snake robot S-shape locomotion, called serpentine locomotion, with the curvature function. It is assumed that the snake robot holds its shape on the serpentine curve. The relative angle $\theta_i = (\phi_i - \phi_{i-1})$ is derived by integrating the curvature function (3.5). Assuming $L = nl$, where l is the length of individual joint, we have

$$\begin{aligned} \theta_i = \int_{S+S_{pi}+\frac{1}{2}l}^{S+S_{pi-1}+\frac{1}{2}l} k(u)du = \int_{S+il+\frac{1}{2}l}^{S+(i-1)l+\frac{1}{2}l} k(u)du \\ = -2\alpha sin(\frac{K_n\pi}{n}) sin(\frac{2K_n\pi}{nl}s + \frac{2K_n\pi}{n}i), i = 1, 2, \cdots, n-1, \end{aligned} \tag{3.6}$$

Defining $b = \frac{2K_n\pi}{nl}, s = ct, ct + bl, ct + 2bl, \cdots, ct + (N-1)bl, A = -2\alpha sin(\frac{K_n\pi}{n}), \omega = bc, \beta = bl$, where c is a constant, gives the serpentine locomotion equation:

$$\theta_i = Asin(\omega t + i\beta) \tag{3.7}$$

3.2 Gait Pattern

In the previous section, we show how a snake robot is moving forward with serpentine locomotion. We now study how the snake robot performs steering movements when advancing with serpentine locomotion. The latter can be considered as a wave composed of horizontal waves transmitted from the snake's head to its tail. Serpentine locomotion is the most commonly used movement of snakes on the horizontal plane. As in Equation (3.7), the serpentine motion mode of snakes is controlled by the sine

wave motion of the i^{th} joint, $i \in \{1, \cdots, N-1\}$. Rewrite formula (3.7) as follows

$$\phi_i = \alpha sin(\omega t + (i-1)\delta) + \phi_0 \tag{3.8}$$

where α and ω are the amplitude and frequency, respectively, of the sinusoidal joint motion and δ determines the phase shift between the joints. The parameter ϕ_0 is a joint angle offset that controls the overall direction of locomotion. The effect of this parameter is given in Fig. 3.2 and Fig. 3.3, which demonstrate the snake robot's turning motion with $N = 10$ links of length $l = 0.14$m. Fig. 3.3 shows the trajectory of the snake head during the motion, while Fig. 3.2 is the turning angle. The snake robot is under the control of (3.8) with $\alpha = 30°$, $\omega = 70°/s$, and $\delta = 40°$. The offset angle is set to $\phi_0 = -0.3rad$ during time interval $t \in [110, 130]$ and $\phi_0 = 0.6rad$ in the time interval $t \in [160, 170]$. Fig. 3.3 clearly shows that the snake robot turns smoothly at point EP3 with small angle offset $\phi_0 = -0.3rad$. while at point EP4, the snake robot makes a sharp turn when $\phi_0 = 0.6rad$.

It can be seen from the figure that a positive (resp. negative) average joint angle causes the snake robot to rotate counterclockwise (resp. clockwise). We also found that the speed of the direction change is related to the amplitude of the average joint angle. In summary, in the serpentine locomotion of a snake robot described in (3.4), as long as the average joint angle is 0, the overall direction of movement is unchanged. When the average joint angle is positive (resp. negative), the movement direction changes counterclockwise (resp. clockwise). The magnitude of the average joint angle determines how quickly the direction of motion changes.

The dynamic model is described in the next sub-section.

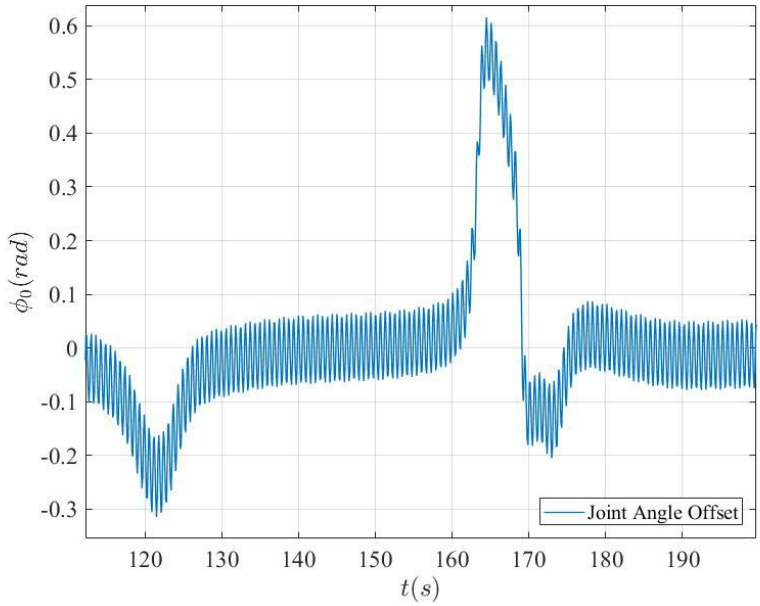


Figure 3.2: Snake robot motion simulation with angle offset ϕ_0 The joint offset $\phi_0 = -0.3rad$ when $t = 120s$, and $\phi_0 = -0.6rad$ when $t = 165s$.

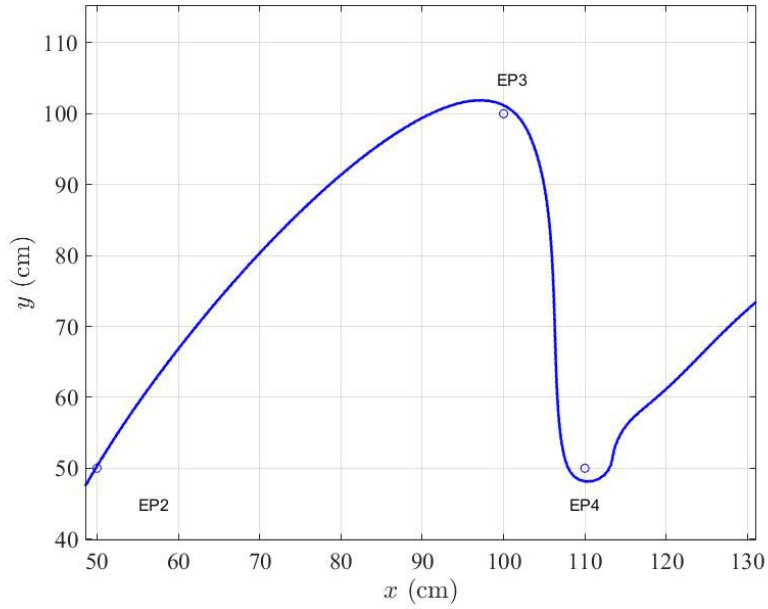


Figure 3.3: For turning angle offset ϕ_0, the snake robot makes a sharp turn at waypoint $EP4$.

3.3 Dynamics

This section presents a model of the accelerations of the snake robot. [33, 50, 61]

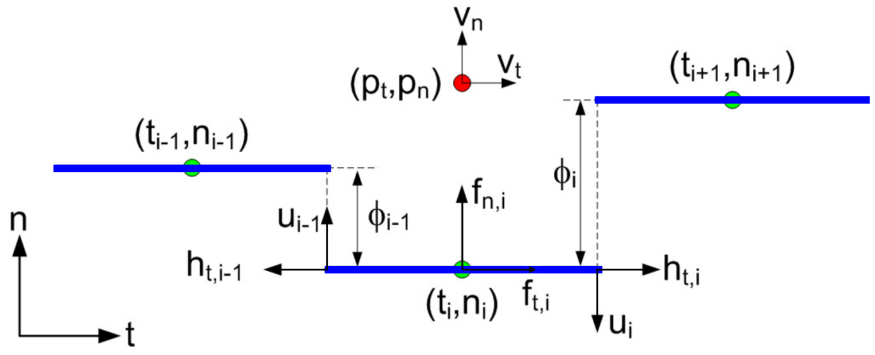


Figure 3.4: Symbols of snake robot kinematics and dynamics. (Adopted from [33])

We can find the force balance for link i from Fig. 3.4 by

$$\begin{aligned} m\ddot{t}_i &= f_{t,i} + h_{t,i} - h_{t,i-1} \\ m\ddot{n}_i &= f_{n,i} + u_i + u_{i-1} \end{aligned} \tag{3.9}$$

where $f_{t,i}$ and $f_{n,i}$ are the ground friction forces defined by

$$\begin{aligned} f_{t_i} &= -c_1\dot{t}_i + c_2(\phi_{i-1} + \phi_i)\dot{n}_i \\ f_{n_i} &= -c_1\dot{n}_i + c_2(\phi_{i-1} + \phi_i)\dot{t}_i \end{aligned} \tag{3.10}$$

The coefficient of viscous friction c_1 is a measure of the friction component, which resists tangential and normal link motions, while c_2 determines the magnitude of the tangential friction component and the normal friction component caused by the normal and tangential link speeds, respectively. $h_{t,i}$ and $h_{t,i-1}$ are the joint constraint forces on link i from link $i+1$ and link $i-1$, respectively, and u_i and u_{i-1} are the actuator forces at joint i and joint $i-1$, respectively. The combined constraint forces $h_{t,i}$ and $h_{t,i-1}$ prevent the relative motion of the connecting rod in the tangential

direction, and the actuator forces u_i and u_{i-1} to cause the relative motion of the connecting rod in the normal direction. The force balance of all links can be written in matrix form

$$\begin{aligned} m\ddot{\mathbf{t}} &= \mathbf{f}_t + \mathbf{D}^T\mathbf{h}_t \\ m\ddot{\mathbf{n}} &= \mathbf{f}_n + \mathbf{D}^T u \end{aligned} \tag{3.11}$$

where $\mathbf{D}$ is the matrix

$$D = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \in R^{(N-1)\times N}, \tag{3.12}$$

$h_t = (h_{t,1}, \cdot, h_{t,N-1}) \in R^{N-1}$, and $u = (u_1, \cdot, u_{N-1}) \in R_{N-1}$. Premultiplying (3.11) by $\frac{1}{m}\mathbf{D}$ gives

$$\mathbf{D}\ddot{n} = \frac{1}{m}\mathbf{D}f_n - \frac{1}{m}\mathbf{D}\mathbf{D}^T u \tag{3.13}$$

By differentiating the holonomic constraint $\mathbf{D}\mathbf{n}+\phi = 0$ twice with respect to time, we show that $\mathbf{D}\ddot{n} = -\ddot{\phi}$. Therefore, we can write the snake robot body shape dynamics as

$$\ddot{\phi} = -\frac{1}{m}\mathbf{D}\mathbf{f}_n + \frac{1}{m}\mathbf{D}\mathbf{D}^T u \tag{3.14}$$

Inserting the friction force normal direction, $\mathbf{f}_n = -c_1 v_n \mathbf{e} + c_1 \mathbf{D}\dot{\phi} + c_2 v_t diag(\mathbf{A}^T\phi)\mathbf{e}$

into (3.14), where matrix $\mathbf{A}$ is given as

$$A = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \in R^{(N-1)\times N}, \tag{3.15}$$

and using the easily verifiable relations $\mathbf{De} = \mathbf{0}$, $\mathbf{D}\overline{\mathbf{D}} = \mathbf{I}_{N-1}$, and $\mathbf{D}diag(\mathbf{A}^T\phi)\mathbf{e} = -\mathbf{A}\mathbf{D}^T\phi$, we have

$$\ddot{\phi} = -\frac{c_1}{m}\dot{\phi} + \frac{c_2}{m}v_t\mathbf{A}\mathbf{D}^T\phi + \frac{1}{m}\mathbf{D}\mathbf{D}^T u \tag{3.16}$$

The tangential acceleration and the normal acceleration of the snake-shaped robot CM are represented by v_t and v_n, respectively, and the sum of the tangential and normal forces applied to the connecting rod divided by the mass Nm of the snake-shaped robot, respectively. This is written as

$$\begin{aligned} \dot{v}_t &= -\tfrac{1}{Nm}(e^T m\ddot{(t)}) = \tfrac{1}{Nm}e^T\mathbf{f}_t, \\ \dot{v}_n &= -\tfrac{1}{Nm}(e^T m\ddot{(n)}) = \tfrac{1}{Nm}e^T\mathbf{f}_n \end{aligned} \tag{3.17}$$

where we note that when the link accelerations are added, the joint restraint force, h_t, and the actuator force u are eliminated, namely $e^T\mathbf{D}^T = 0$. Inserting the friction force in the tangential direction $\mathbf{f}_t = -c_1v_t\mathbf{e} + c_1\overline{\mathbf{D}}\dot{\phi} + c_2v_t diag(\mathbf{A}^T\phi)\mathbf{e}$ and the friction force in the normal direction $\mathbf{f}_n = -c_1v_n\mathbf{e} + c_1\mathbf{D}\dot{\phi} + c_2v_t diag(\mathbf{A}^T\phi)\mathbf{e}$ into (3.17), and using the easily verifiable relations $e^T diag(\mathbf{A}^T\phi e) = 2e^{-T}\phi, e^T\overline{\mathbf{D}} = \mathbf{0}$, and $e^T diag(\mathbf{A}^T\phi)\overline{\mathbf{D}}$, we get

$$\begin{aligned} \dot{v}_t &= -\tfrac{c_1}{m}v_t + \tfrac{2c_2}{Nm}v_n\bar{e}^T\phi - \tfrac{c_2}{Nm}\phi^T A\bar{D}v_\phi, \\ \dot{v}_n &= -\tfrac{c_1}{m}v_n + \tfrac{2c_2}{Nm}v_t\bar{e}^T\phi, \end{aligned} \tag{3.18}$$

The apparent difference between a snake robot with rotating joints and a snake robot with moving joints in the simplified model is the absolute orientation of the robot. Snake robots with rotating joints do not have a clear direction because each link has an independent link angle. However, the orientation of a robot with moving joints is clearly defined by the scalar angle, which is also the angle of all the links. This difference must be taken into account when we model angular acceleration, $\ddot{\theta}$, for a snake robot with moving joints. The model requires that when the average $\frac{1}{N-1}e^{-T}\phi$ of the joint coordinates is non-zero, the direction of the forward movement changes (that is, θ changes). A model that fits this attribute is

$$\ddot{\theta} = -c_3\dot{\theta} + \frac{c_4}{N-1}v_t\mathbf{e}^{-}T\phi \tag{3.19}$$

The rotation of the snake robot is realized by a viscous friction torque determined by the friction coefficient c_3. In addition, the joint coordinates produce a mean value of the moment the robot passes the coefficient c_4, and the expansion also passes the forward speed v_t. The induced torque must be multiplied by v_t since the snake robot would otherwise have a constant angular velocity while still lying on non-zero average co-ordination. Although the $\ddot{\theta}$ model is simplified, when the coefficients of c_3 and c_4 are correctly selected, the behavior of the model is very similar to that of a snake robot with rotating joints.

We can write the complete model of the snake robot as

$$\begin{aligned}
\dot{\phi} &= v_\phi,\ \dot{\theta} = v_\theta, \\
\dot{p}_x &= v_t \cos\theta - v_n \sin\theta, \\
\dot{p}_y &= v_t \sin\theta + v_n \cos\theta, \\
\dot{v}_\phi &= -\tfrac{c_1}{m} v_\phi + \tfrac{c_2}{m} v_t A D^T \phi + \tfrac{1}{m} D D^T u, \\
\dot{v}_\theta &= -c_3 v_\theta + \tfrac{c_4}{N-1} v_t \bar{e}^T \phi, \\
\dot{v}_t &= -\tfrac{c_1}{m} v_t + \tfrac{2c_2}{Nm} v_n \bar{e}^T \phi - \tfrac{c_2}{Nm} \phi^T A \bar{D} v_\phi, \\
\dot{v}_n &= -\tfrac{c_1}{m} v_n + \tfrac{2c_2}{Nm} v_t \bar{e}^T \phi,
\end{aligned} \tag{3.20}$$

where N represents the number of links, the link angle of the ith, $i = 1, \ldots, N$ link of the snake robot is denoted by $\theta_i \in R$, while the joint angle of joint $i, i = 1, \ldots, N-1$ is given by $\phi_i = \theta_{i+1} - \theta_i$, $\phi = [\phi_1, \ldots, \phi_{N-1}] \in R^{N-1}$ is the joint angle vector, $\theta = \frac{1}{N}\sum_{i=1}^{N} \theta_i \in R$ denotes the heading angle of the robot, $(p_x, p_y) \in R^2$ is the planar position of the center of mass, $v_\phi \in R^{N-1}$ is the joint velocity vector, $v_\theta \in R$ is the angular velocity, $v_t \in R$ and $v_n \in R$ are, respectively, the tangential and normal direction velocity of the robot, $u \in R^{N-1}$ is the vector of actuator torque, m is the mass of a link, $\bar{e} = [1, \ldots, 1]^T \in R^{N-1}$, $\bar{D} = D^T(DD^T)^{-1} \in R^{N\times(N-1)}$, and matrix (A) and (D) are given by (3.15) and (3.12), respectively. The parameters $c_i, i = 1, 2, 3, 4$ in (3.20) are positive scalar friction coefficients. Generally speaking, the precise values of c_i are difficult to determine due to the inevitable measurement noise. Moreover, different terrains over which the snake robots carry out tasks also affect the value of c_i. In the next chapter, we develop model-based control methods for a class of planar snake robot. Moreover, we solve the control problems of the planar snake robot without prior knowledge of the friction coefficient.

Chapter 4

Adaptive Path Following of Snake Robots

This section investigates path following problems for a class of planar underactuated bio-inspired snake robots. Path following for multi-DOF robots, like snake robots and eel robots, has always been a hard problem, especially in environments with unknown and variable friction coefficient. In this section, we investigate the straight-line path following problem and the curve path following problem for a class of planar underactuated snake robot with external positioning device. An adaptive controller is employed to drive the snake robot in the environment with unknown and variable friction coefficients. Simulations and experiments on 8-link snake robots are carried out to illustrate the effectiveness of the proposed methods.

4.1 Adaptive Straight Path Following

This section investigates the straight path following problem with unknown and varied friction coefficients using the adaptive controller [61]. Existing works usually design controllers that require knowledge of the exact values of friction coefficients.

The latter depend on the specific operating terrain and may not always be known *a priori*. With the aid of backstepping control, we present a novel adaptive controller that can actively compensate for unknown and varied friction coefficients. Moreover, it is proved via LaSalle-Yoshizawa theorem that the tracking errors converge to zero asymptotically and that all the parameter estimation errors are bounded. Our contribution is significant for too reasons. First, the uncertain friction coefficients of the snake robot are taken into account in the controller design, which has not been discussed up to now in the relevant works [36, 33, 43], and [8]. Second, the cascaded systems stability method [33] and the reduction theorems used in [43, 45], which play vital roles in their control system design and analysis, cannot be applied to our case because uncertain dynamics is involved. Furthermore, the adaptive controller proposed in this dissertation is applicable to the systems considered in [33] as well as to others for which their methodology is not applicable. Simulations and experiments on 8-link snake robots are carried out to illustrate the effectiveness of the proposed controller.

4.1.1 Control Objective

The goal of the control task is to design an adaptive controller for the planar snake robot described in (3.20). The control task is to design an actuator torque vector u of the form $u = u(\phi, \theta, p_y, v_\phi, v_\theta, v_t, v_n) \in R^{N-1}$ for the snake robot (3.20) in the presence of unknown friction coefficients $c_i, i = 1, 2, 3, 4$ so that it can converge to and finally track a straight-line path while maintaining a heading parallel with the path, i.e.,

$$\lim_{t\to\infty} p_y(t) = 0 \text{ and } \lim_{t\to\infty} \theta(t) = 0. \tag{4.1}$$

To achieve the control objective, we invoke the coordinate and state transformation similarl to those in [11, 14]

$$\begin{aligned} \bar{p}_x &= p_x + \epsilon \cos\theta, \\ \bar{p}_y &= p_y + \epsilon \sin\theta, \\ \bar{v}_n &= v_n + \epsilon v_\theta, \end{aligned} \tag{4.2}$$

where $\epsilon = -\frac{2(N-1)c_2}{Nmc_4}$. Then, we have

$$\begin{aligned} \dot{\phi} &= v_\phi,\ \dot{\theta} = v_\theta, \\ \dot{\bar{p}}_y &= v_t \sin\theta + \bar{v}_n \cos\theta, \\ \dot{v}_\phi &= -\tfrac{c_1}{m} v_\phi + \tfrac{c_2}{m} v_t A D^T \phi + \tfrac{1}{m} D D^T u, \\ \dot{v}_\theta &= -c_3 v_\theta + \tfrac{c_4}{N-1} v_t \bar{e}^T \phi, \\ \dot{v}_t &= -\tfrac{c_1}{m} v_t + \tfrac{2c_2}{Nm} (\bar{v}_n - \epsilon v_\theta) \bar{e}^T \phi - \tfrac{c_2}{Nm} \phi^T A \bar{D} v_\phi, \\ \dot{\bar{v}}_n &= X v_\theta + Y \bar{v}_n, \end{aligned} \tag{4.3}$$

where $X = \epsilon(\frac{c_1}{m} - c_3)$ and $Y = -\frac{c_1}{m}$.

Remark 1 *The coordinate and state transformation (4.2) requires the parameter ϵ, which is actually a function of unknown coefficients c_2 and c_4. However, as noted in [50], c_4 is only influenced by the propulsion coefficient c_2. Thus, it is reasonable to suppose that $\frac{c_2}{c_4}$ is an available constant and independent of the specific working environment. Hence, in this dissertation, we treat ϵ as a known constant such that the transformation (4.2) can be realized.*

4.1.2 Adaptive Backstepping Controller Design

The gait pattern of the snake robot is presented in (3.7). Inspired by the sinusoidal lateral undulatory gait introduced in [57] and [55], the reference trajectory for the

i^{th} joint angle of the robot is described in (4.51).

Assumption 1 *The snake robot executes lateral undulation and keeps a nonzero and positive forward velocity v_t. Furthermore, we assume that there exist two known positive constants $\underline{V}$ and $\bar{V}$ such that $\underline{V} \leq v_t(t) \leq \bar{V}, \forall t \geq 0$.*

Remark 2 *Assumption 1 is standard and is also needed for the control of snake robots even in the absence of uncertainties [33, 50, 37, 27]. The validity of Assumption 1 can be verified by inspecting the equations of motion in (4.3).*

To design an adaptive controller, the backstepping design methodology [29] is adopted. The detailed design procedures are included below.

Step 1: To steer the snake robot toward the desired straight path, we utilize the line-of-sight guidance law [14]

$$\bar{\theta} = -\arctan\left(\frac{\bar{p}_y}{\triangle}\right), \tag{4.4}$$

where $\triangle$ is a positive design parameter that satisfies $\triangle > \frac{|X|}{|Y|}(1 + \frac{\bar{V}}{\underline{V}})$.

Remark 3 *Because the friction coefficients c_1 and c_3 are unknown, X and Y are unknown but bounded constants. Thus, in practical applications, we can always preselect a large enough $\triangle$ such that $\triangle > \frac{|X|}{|Y|}(1 + \frac{\bar{V}}{\underline{V}})$.*

The heading angle error variable is defined as

$$e_\theta = s_\theta - s_{\bar{\theta}}, \tag{4.5}$$

where $s_\theta = \theta + \lambda_\theta \dot{\theta}$ and $s_{\bar{\theta}} = \bar{\theta} + \lambda_\theta \dot{\bar{\theta}}$ with λ_θ is a positive constant. Taking the time

derivative of e_θ, we obtain

$$\begin{aligned}\dot{e}_\theta &= v_\theta + \lambda_\theta(-c_3 v_\theta + \tfrac{c_4}{N-1} v_t \bar{e}^T \phi) - \dot{s}_{\bar{\theta}} \\ &= v_\theta + \lambda_\theta \left(-c_3 v_\theta + c_4 v_t \phi_o + \tfrac{c_4 v_t}{N-1} \bar{e}^T (\phi - \bar{\phi})\right) \\ &\quad -\dot{s}_{\bar{\theta}} + \tfrac{\lambda_\theta c_4 v_t}{N-1} \textstyle\sum_{i=1}^{N-1} \alpha \sin(\omega t + (i-1)\delta).\end{aligned} \tag{4.6}$$

Where, $\bar{\phi} = [\bar{\phi}_1, \cdots, \bar{\phi}_{N-1}]^T \in R^{N-1}$, we choose the joint offset ϕ_o as

$$\begin{aligned}\phi_o &= \tfrac{\hat{d}_3}{v_t} v_\theta + \tfrac{\hat{d}_4}{\lambda_\theta v_t}(-k_\theta e_\theta - v_\theta + \dot{s}_{\bar{\theta}}) \\ &\quad -\tfrac{1}{N-1} \textstyle\sum_{i=1}^{N-1} \alpha \sin(\omega t + (i-1)\delta),\end{aligned} \tag{4.7}$$

where $k_\theta > \frac{1}{4}$ is the constant control gain, $\hat{d}_3$ and $\hat{d}_4$ are, respectively, the estimates of $\frac{c_3}{c_4}$ and $\frac{1}{c_4}$. The adaptive tuning laws for $\hat{d}_3$ and $\hat{d}_4$ are designed as

$$\dot{\hat{d}}_3 = -k_3 v_\theta e_\theta, \dot{\hat{d}}_4 = -k_4(-k_\theta e_\theta - v_\theta + \dot{s}_{\bar{\theta}}) e_\theta, \tag{4.8}$$

where k_3 and k_4 are positive constant tuning gains. Substituting (4.7) into (4.6) yields

$$\begin{aligned}\dot{e}_\theta &= -k_\theta e_\theta + \lambda_\theta c_4 v_\theta(\hat{d}_3 - \tfrac{c_3}{c_4}) + \lambda_\theta \tfrac{c_4 v_t}{N-1} \bar{e}^T (\phi - \bar{\phi}) \\ &\quad +c_4(-k_\theta e_\theta - v_\theta + \dot{s}_{\bar{\theta}})(\hat{d}_4 - \tfrac{1}{c_4}).\end{aligned} \tag{4.9}$$

We construct the following Lyapunov function candidate

$$V_1 = \frac{1}{2} e_\theta^2 + \frac{\lambda_\theta c_4}{2k_3}(\frac{c_3}{c_4} - \hat{d}_3)^2 + \frac{c_4}{2k_4}(\frac{1}{c_4} - \hat{d}_4)^2. \tag{4.10}$$

Differentiating V_1 with respect to time and using (4.9), we get

$$\begin{aligned}\dot{V}_1 &= -k_\theta e_\theta^2 + \tfrac{\lambda_\theta c_4 v_t}{N-1} e_\theta \bar{e}^T (\phi - \bar{\phi}) - \lambda_\theta c_4 v_\theta e_\theta (\tfrac{c_3}{c_4} - \hat{d}_3) \\ &\quad -c_4(-k_\theta e_\theta - v_\theta + \dot{s}_{\bar{\theta}}) e_\theta (\tfrac{1}{c_4} - \hat{d}_4) \\ &\quad +\tfrac{\lambda_\theta c_4}{k_3}(\tfrac{c_3}{c_4} - \hat{d}_3)(-\dot{\hat{d}}_3) + \tfrac{c_4}{k_4}(\tfrac{1}{c_4} - \hat{d}_4)(-\dot{\hat{d}}_4).\end{aligned}$$

In view of (4.8), we can further obtain

$$\dot{V}_1 = -k_\theta e_\theta^2 + \frac{\lambda_\theta c_4 v_t}{N-1} e_\theta \bar{e}^T (\phi - \bar{\phi}). \tag{4.11}$$

Noting that $|\bar{e}^T(\phi - \bar{\phi})| \leq \|\phi - \bar{\phi}\|_1 \leq \sqrt{N-1}\|\phi - \bar{\phi}\|_2$, (4.11) can be further relaxed to

$$\begin{aligned} \dot{V}_1 &\leq -k_\theta e_\theta^2 + \tfrac{\lambda_\theta c_4 v_t}{\sqrt{N-1}} |e_\theta| \|\phi - \bar{\phi}\|_2 \\ &\leq -(k_\theta - \tfrac{1}{4}) e_\theta^2 + \tfrac{\lambda_\theta^2 c_4^2 v_t^2}{N-1} \|\phi - \bar{\phi}\|_2^2, \end{aligned} \tag{4.12}$$

where we have used the Young's inequality $\frac{\lambda_\theta c_4 v_t}{\sqrt{N-1}} |e_\theta| \|\phi - \bar{\phi}\|_2 \leq \frac{1}{4} e_\theta^2 + \frac{\lambda_\theta^2 c_4^2 v_t^2}{N-1} \|\phi - \bar{\phi}\|_2^2$ to obtain the second inequality.

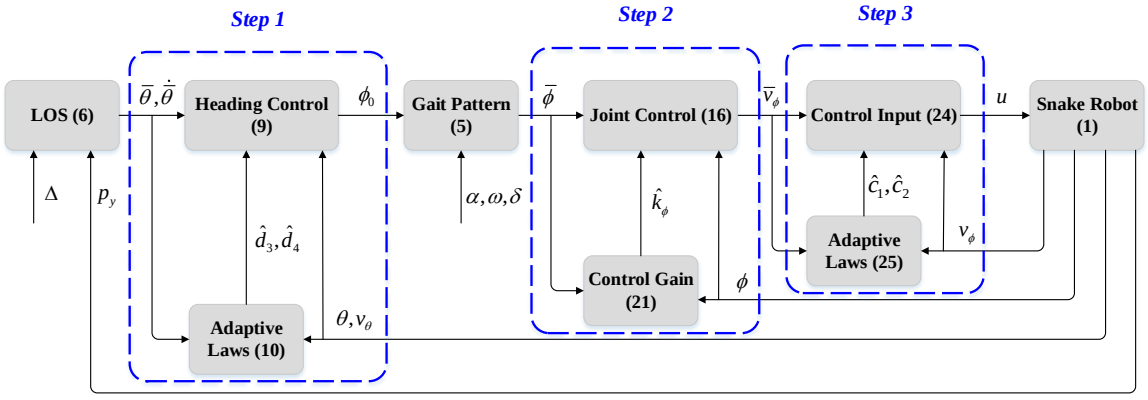


Figure 4.1: Structure of the proposed adaptive path following controller.

Step 2: The following design is to make the joint angle error between ϕ and $\bar{\phi}$ as small as possible. The error variable e_ϕ is introduced as

$$e_\phi = \phi - \bar{\phi}.$$

Computing the derivative e_ϕ, we can obtain

$$\dot{e}_\phi = v_\phi - \dot{\bar{\phi}}. \tag{4.13}$$

Choose the following virtual controller $v_{\bar{\phi}}$ as

$$v_{\bar{\phi}} = -\hat{k}_\phi e_\phi + \dot{\bar{\phi}}, \tag{4.14}$$

where $\hat{k}_\phi$ is a time-varying control gain to suppress the unknown bounded function $\frac{\lambda_\theta^2 c_4^2 v_t^2}{N-1}$. Then, (4.13) can be rewritten as

$$\dot{e}_\phi = -\hat{k}_\phi e_\phi + (v_\phi - v_{\bar{\phi}}). \tag{4.15}$$

Define a Lyapunov function candidate at this step as

$$V_2 = V_1 + \frac{1}{2} e_\phi^T e_\phi + \frac{1}{2\gamma}(k_\phi - \hat{k}_\phi)^2, \tag{4.16}$$

where γ is a positive constant tuning gain, k_ϕ is a positive scalar to be determined later. Differentiating V_2 with respect to time and using (4.12) and (4.15) can obtain

$$\begin{aligned} \dot{V}_2 &= \dot{V}_1 + e_\phi^T \dot{e}_\phi + \tfrac{1}{\gamma}(k_\phi - \hat{k}_\phi)(-\dot{\hat{k}}_\phi) \\ &\leq -(k_\theta - \tfrac{1}{4}) e_\theta^2 - (\hat{k}_\phi - \tfrac{\lambda_\theta^2 c_4^2 v_t^2}{N-1}) e_\phi^T e_\phi \\ &\quad + \tfrac{1}{\gamma}(k_\phi - \hat{k}_\phi)(-\dot{\hat{k}}_\phi) + e_\phi^T (v_\phi - v_{\bar{\phi}}). \end{aligned} \tag{4.17}$$

Choose k_ϕ to be sufficiently large such that $k_\phi \geq \frac{\lambda_\theta^2 c_4^2 \bar{V}^2}{N-1} + \sigma_\phi$ with σ_ϕ being a positive constant. Thus, (4.17) becomes

$$\begin{aligned} \dot{V}_2 &\leq -(k_\theta - \tfrac{1}{4}) e_\theta^2 - \sigma_\phi e_\phi^T e_\phi - (\hat{k}_\phi - k_\phi) e_\phi^T e_\phi \\ &\quad + \tfrac{1}{\gamma}(k_\phi - \hat{k}_\phi)(-\dot{\hat{k}}_\phi) + e_\phi^T (v_\phi - v_{\bar{\phi}}). \end{aligned} \tag{4.18}$$

Choosing

$$\dot{\hat{k}}_\phi = \gamma e_\phi^T e_\phi, \tag{4.19}$$

we deduce that

$$\dot{V}_2 \;\leq\; -(k_\theta - \tfrac{1}{4})e_\theta^2 - \sigma_\phi e_\phi^T e_\phi + e_\phi^T(v_\phi - v_{\bar{\phi}}). \tag{4.20}$$

Step 3: In this step, we can design the actuator torque vector u to make the error $e_v = v_\phi - v_{\bar{\phi}}$ as small as possible. Taking the time derivative of e_v and using (4.3) we obtain

$$\dot{e}_v = -\frac{c_1}{m}v_\phi + \frac{c_2}{m}v_t A D^T\phi + \frac{1}{m}DD^T u - \dot{v}_{\bar{\phi}}. \tag{4.21}$$

We propose the following adaptive controller for (3.20) with unknown friction coefficients

$$u = m(DD^T)^{-1}(\bar{u} + \frac{\hat{c}_1}{m}v_\phi - \frac{\hat{c}_2}{m}v_t A D^T\phi), \tag{4.22}$$

where $\hat{c}_1$ and $\hat{c}_2$ are, respectively, the estimates of c_1 and c_2, and

$$\bar{u} = -k_v e_v + \dot{v}_{\bar{\phi}} - e_\phi$$

with k_v being the positive control gain. The adaptive tuning laws for $\hat{c}_1$ and $\hat{c}_2$ are designed as

$$\dot{\hat{c}}_1 = -k_1\frac{e_v^T v_\phi}{m}, \dot{\hat{c}}_2 = k_2\frac{v_t e_v^T}{m}AD^T\phi, \tag{4.23}$$

where k_1 and k_2 are positive constant tuning gains.

Substituting (4.22) into (4.21) yields

$$\dot{e}_v = -k_v e_v - \frac{v_\phi}{m}(c_1 - \hat{c}_1) + \frac{v_t}{m}AD^T\phi(c_2 - \hat{c}_2) - e_\phi. \tag{4.24}$$

Choose the following Lyapunov function candidate

$$V_3 = V_2 + \frac{1}{2}e_v^T e_v + \frac{1}{2k_1}(c_1 - \hat{c}_1)^2 + \frac{1}{2k_2}(c_2 - \hat{c}_2)^2. \tag{4.25}$$

Differentiating V_3 with respect to time and using (4.20) and (4.24) yields

$$\begin{aligned} \dot{V}_3 &= \dot{V}_2 + e_v^T \dot{e}_v + \tfrac{1}{k_1}(c_1 - \hat{c}_1)(-\dot{\hat{c}}_1) + \tfrac{1}{k_2}(c_2 - \hat{c}_2)(-\dot{\hat{c}}_2) \\ &\leq -(k_\theta - \tfrac{1}{4})e_\theta^2 - \sigma_\phi e_\phi^T e_\phi - k_v e_v^T e_v \\ &\quad + e_v^T(-\tfrac{v_\phi}{m}(c_1 - \hat{c}_1) + \tfrac{v_t}{m}AD^T\phi(c_2 - \hat{c}_2)) \\ &\quad + \tfrac{1}{k_1}(c_1 - \hat{c}_1)(-\dot{\hat{c}}_1) + \tfrac{1}{k_2}(c_2 - \hat{c}_2)(-\dot{\hat{c}}_2). \end{aligned} \tag{4.26}$$

Then substituting (4.23) into (4.26), we have that

$$\dot{V}_3 \leq -(k_\theta - \frac{1}{4})e_\theta^2 - \sigma_\phi e_\phi^T e_\phi - k_v e_v^T e_v, \tag{4.27}$$

which is obviously negative definite.

Now we have completed the controller design procedure for the snake robot with unknown friction coefficients $c_i, i = 1, 2, 3, 4$. The overall adaptive path following control structure is shown in Fig. 4.1.

The main result of this dissertation can be stated in the following theorem. The LaSalle-Yoshizawa theorem [29] that is used in our stability analysis is recalled next.

Lemma 1 *Let $x = 0$ be an equilibrium point of $\dot{x} = f(x, t)$ and suppose $f : R^n \times R_+ \to R^n$ is locally Lipschitz in x uniformly in t. Let $V : R^n \times R_+ \to R_+$ be a continuously differentiable function such that*

$$\gamma_1(|x|) \leq V(x, t) \leq \gamma_1(|x|),$$
$$\dot{V}(x, t) = \tfrac{\partial V}{\partial t} + \tfrac{\partial V}{\partial x}f(x, t) \leq -W(x) \leq 0,$$

$\forall t \geq 0$, $\forall x \in R^n$, where γ_1 and γ_2 are class of $\mathcal{K}_\infty$ functions and W is a continuous

function. Then, all solutions of $\dot{x} = f(x,t)$ *are globally uniformly bounded and satisfy*

$$\lim_{t\to\infty} W(x(t)) = 0.$$

In addition, if $W(x(t))$ *is positive definite, then the equilibrium* $x = 0$ *is globally uniformly asymptotically stable.*

Theorem 1 *For the system (3.20), under Assumption 1, the path following controller defined by (4.51), (4.7), and (4.22) with adaptive tuning laws (4.8), (4.19), and (4.23) achieves the control objective (4.1).*

Proof 4.1.1 *By the definition of* V_3 *in (4.25) along with (4.10) and (4.16), it can be concluded from (4.27) that* e_θ, e_ϕ, e_v, $(c_1 - \hat{c}_1)$, $(c_2 - \hat{c}_2)$, $(\frac{c_3}{c_4} - \hat{d}_3)$, $(\frac{1}{c_4} - \hat{d}_4)$, *and* $(k_\phi - \hat{k}_\phi)$ *are bounded. Since* $c_i, 1, 2, 3, 4$ *and* k_ϕ *are positive constants, the boundedness of the estimates* $\hat{c}_1$, $\hat{c}_2$, $\hat{d}_3$, $\hat{d}_4$, *and* $\hat{k}_\phi$ *can be ensured. In addition, applying Lemma 1 to (4.27) can result in* $\lim_{t\to\infty} e_\theta(t) = 0$, $\lim_{t\to\infty} e_\phi(t) = 0$, *and* $\lim_{t\to\infty} e_v(t) = 0$. *Recalling (4.5) and* $s_\theta = \theta + \lambda_\theta \dot{\theta}$ *and using Lemma 3 in [58], it can be easily verified that* $\lim_{t\to\infty}(\theta(t) - \bar{\theta}(t)) = 0$ *and* $\lim_{t\to\infty}(v_\theta(t) - \dot{\bar{\theta}}(t)) = 0$. *Therefore, employing the similar arguments of Theorem 1 in [7], we have* $\lim_{t\to\infty} \bar{p}_y(t) = 0$ *and* $\lim_{t\to\infty} \bar{v}_n(t) = 0$ *due to the asymptotic convergence of* $(\theta(t) - \bar{\theta}(t))$ *and* $(v_\theta(t) - \dot{\bar{\theta}}(t))$. *It then follows from (4.4) that* $\lim_{t\to\infty} \bar{\theta}(t) = 0$. *As an immediate result,* $\lim_{t\to\infty} \theta(t) = 0$. *Finally, by (4.2), we can draw a conclusion that* $\lim_{t\to\infty} p_y(t) = 0$, *which completes the proof.*

Remark 4 *In [50] and [7], path following control problems of underactuated mechanisms with known dynamics such as snake robots and surface vessels were studied. The cascaded systems stability theory was employed to deduce the asymptotic convergence of the tracking errors. However, the absence of uncertainties in these systems prevents the application of such a scheme in path following problems for snake robots with unknown friction coefficients that are addressed in this dissertation. To tackle*

this issue, a novel methodology combining the backstepping with adaptive control is introduced to actively compensate for unknown friction coefficients.

4.1.3 Simulations and experiments

In this section, some simulations and experiments are carried to illustrate the performance of the proposed adaptive controller.

4.1.3.1 Simulation Study

The model of the snake robot (3.20) and the path following controller are defined by (4.51), (4.7), and (4.22) with adaptive tuning laws (4.8), (4.19), and (4.23) are conducted by the MATLAB 'ode45' method. We consider the snake robot with 8 links. The model parameters are adopted based upon the snake robot in the Robotics and Biomimetics Laboratory of University of Nevada as shown in Tab. 4.1. The control parameter in (4.5) is selected as $\lambda_\theta = 2.2$. The control gains in (4.7) and (4.22) are chosen as $k_\theta = 0.3$ and $k_v = 17$. The adaptive tuning gains in (4.8), (4.19), and (4.23) are $k_1 = 0.0002$, $k_2 = 0.05$, $k_3 = k_4 = 0.0005$, and $\gamma = 0.6$. To obtain $\dot{\bar{\theta}}$, $\ddot{\bar{\theta}}$, $\dot{\bar{\phi}}$, and $\ddot{\bar{\phi}}$ that are required for the controller (4.51), (4.7), and (4.22), 3rd-order low-pass filter reference models are imposed, whose details can be found in Chapter 5 in [12]. The parameters of the reference models are selected as $\omega_n = \pi/3$ and $\zeta = 1$.

All the initial conditions of estimates are set to zero. Simulation results are obtained and showed in Figs. 4.2-4.5. In spite of the presence of unknown friction coefficients $c_i, i = 1, 2, 3, 4$, Theorem 1 implies that the state variables p_y and θ converge to zero.

Further, to underline the capability of the proposed controller to accommodate unknown friction coefficients that vary with the operating terrain, we compare the cascaded approach controller [33] and our method when applied to the 8-link snake

Table 4.1: Table

Symbol	Description	Value
N	Number of links	8
m	Mass of a link	0.135kg
c_1, c_2, c_3, c_4	Friction coefficients	0.45, 3, 0.5, 20
δ	Lookahead distance	1.2m
ϕ	Joint coordinates	0°
θ	Orientation of the robot	90°
$(p_x, p_y) \in R$	CM position of the robot	(0,1)m
v_ϕ	Joint velocities	$0^\circ/s$
v_θ	Angular velocity of the robot	$0^\circ/s$
(v_t, v_n)	Translational velocity of the robot	(0.2m/s, 0m/s)
α	Amplitude of gait	0.1m
ω	Frequency of gait	$120^\circ/s$
δ	Phase shift	40°

robot. Specifically, the friction coefficients are set as

$$\begin{cases} \underbrace{c_1 = 0.45, c_2 = 3, c_3 = 0.5, c_4 = 40}_{\text{Terrain 1}}, & \text{If } x \leq 18\text{m} \\ \underbrace{c_1 = 0.54, c_2 = 1.8, c_3 = 1.75, c_4 = 24}_{\text{Terrain 2}}, & \text{If } x > 18\text{m} \end{cases}$$

The friction coefficients required for the controller [33] are fixed at $c_1 = 0.45$, $c_2 = 3$, $c_3 = 0.5$, and $c_4 = 40$ over both terrains, while the specific values of these coefficients are not needed in our method. The simulation is run for 100s. Comparative results are shown in Fig. 4.6. It can be observed that, under both control methods, the snake robot can perfectly track the desired path, i.e., $x-$axis, over Terrain 1. The asymptotic convergence over Terrain 2 can still be achieved under our control method after a short delay. The simulation results confirms that the proposed controller tracks the desired path with unknown friction coefficients.

The comparative simulation with different parameters has been conducted to demonstrate the situation, which is closer to the real experiment. The comparative

simulation friction coefficients are c_1=0.3, c_2=2.4, c_3=0.4, and c_4=16. The friction coefficients are selected smaller than the first simulation. Furthermore, the tangential velocity is $v_t = 1.5m/s$ instead of $0.2m/s$ is closer to the snake robot moving velocity. Other parameters are the same as the first simulation. All the initial conditions of estimates are set to zero. Simulation results are obtained and shown in Fig. 4.8 and Fig. 4.9.

It can be easily observed from comparative simulation results that both the trajectory p_y and the snake robot heading angle converge to zero at around 40s, which is much longer than the first simulation. The comparative simulation is closer to the experiment; the snake robot can not converge to the desired path as quick as the simulation due to the mechanical limitation, *i.e.* mechanical snake robot can not move with very high speed. This simulation well explains the gap between the simulation and the experiment.

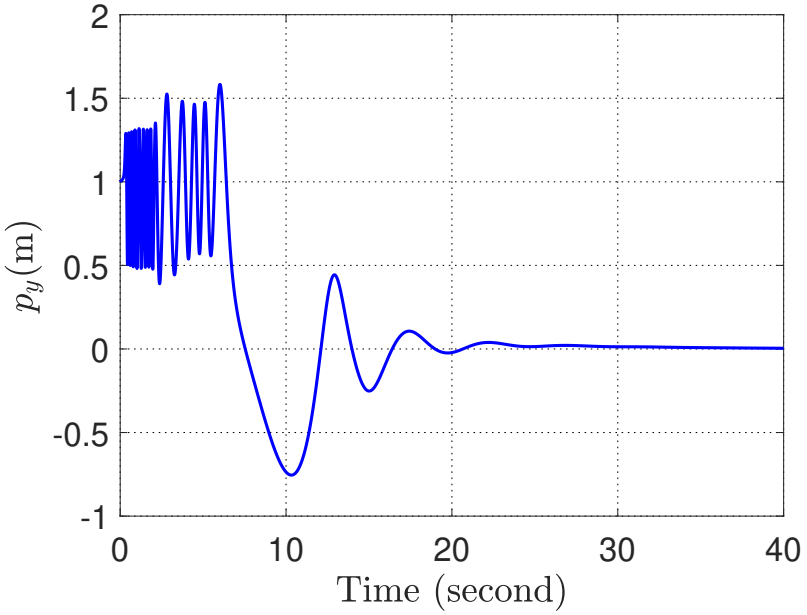


Figure 4.2: Trajectory of p_y.

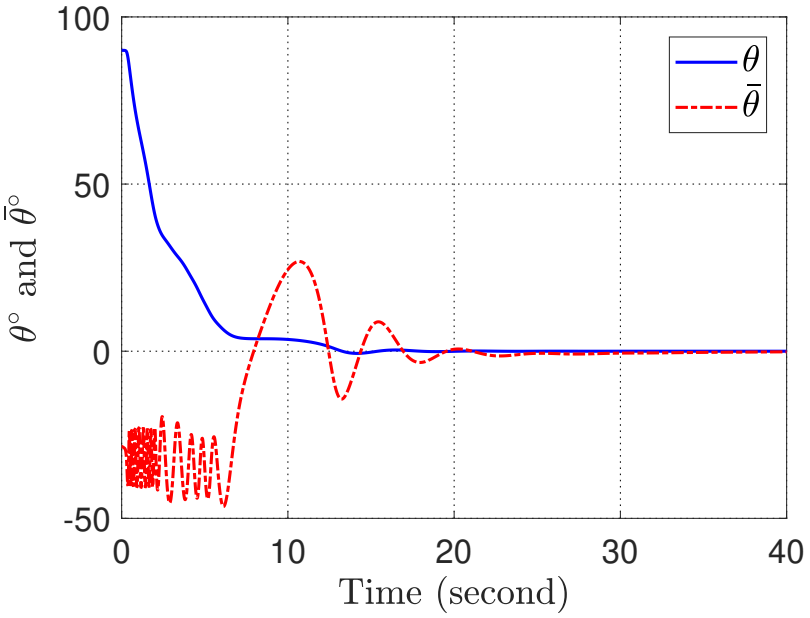


Figure 4.3: Trajectories of θ and $\bar{\theta}$.

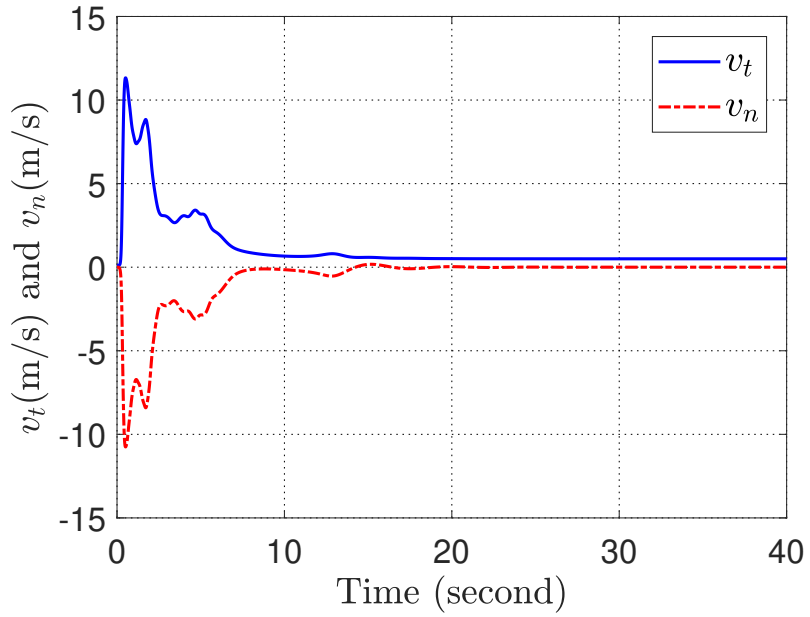


Figure 4.4: Velocities v_t and v_n.

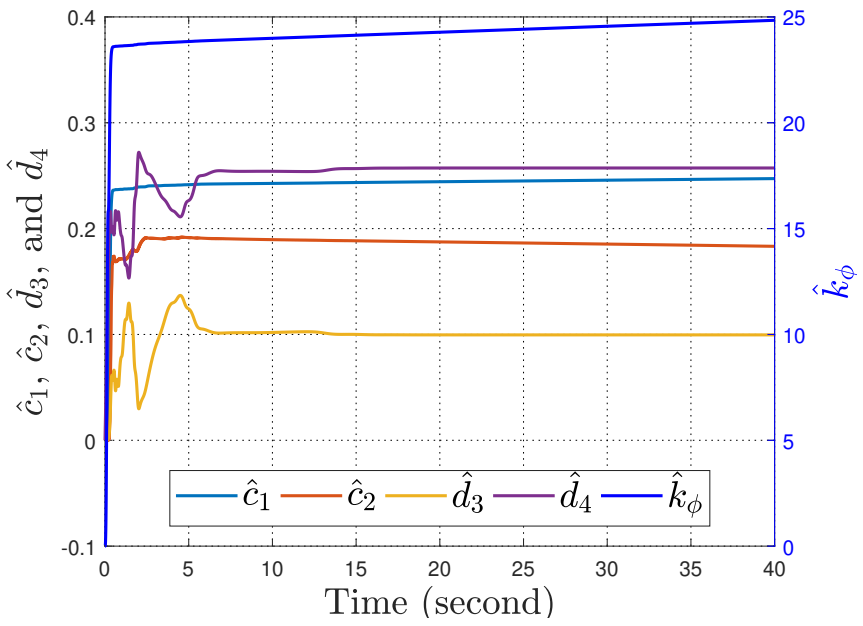


Figure 4.5: Parameter estimates.

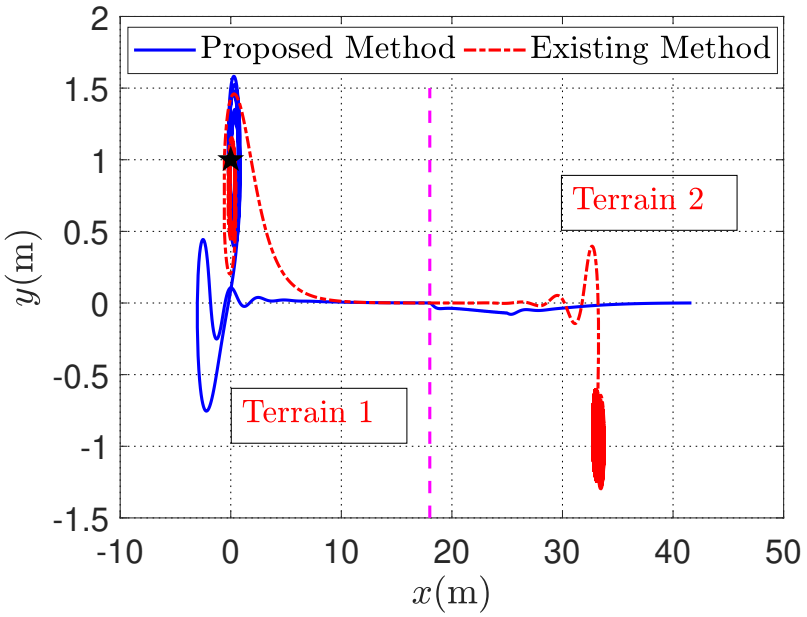


Figure 4.6: Comparative simulation result.

4.1.3.2 Experimental Study

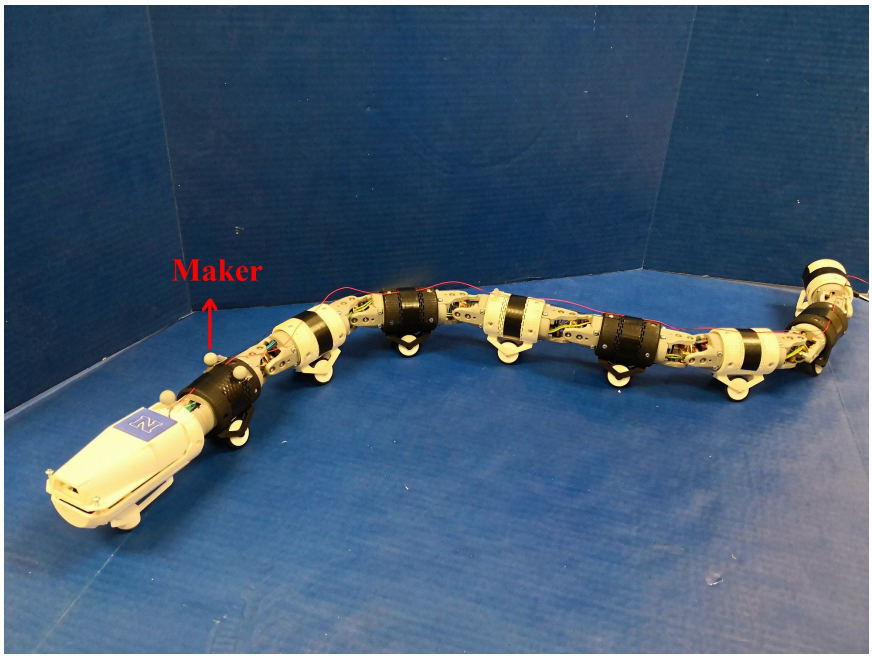


Figure 4.7: Snake robot employed in the experiment.

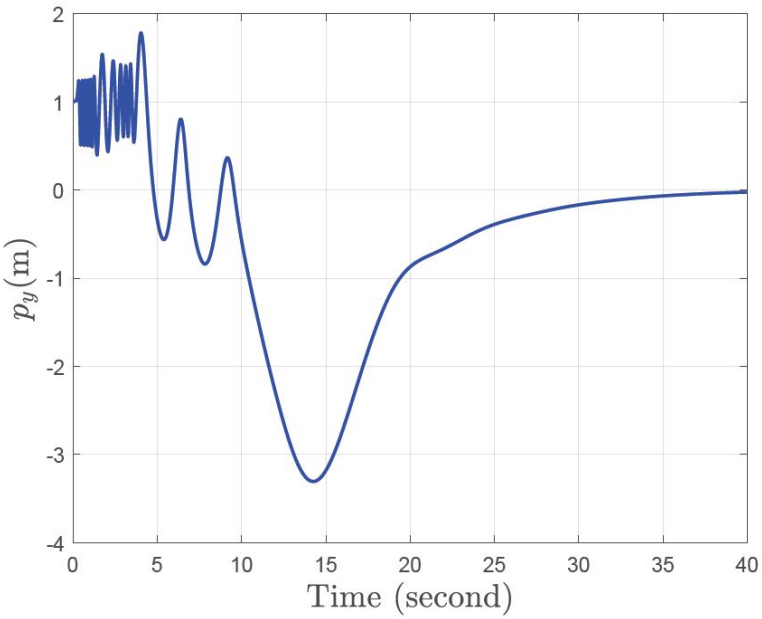


Figure 4.8: Trajectory of p_y in the comparative simulation.

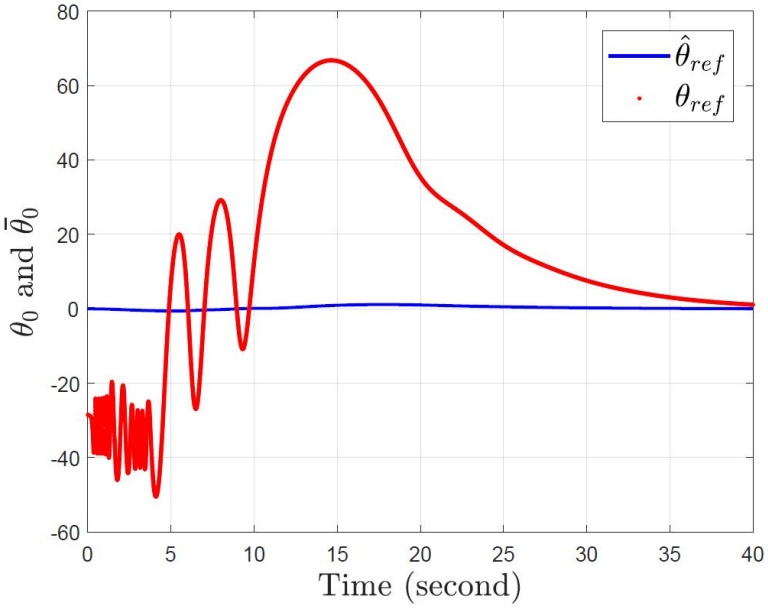


Figure 4.9: Trajectories of θ and $\bar{\theta}$ in the comparative simulation.

Experimental setup: The performance of the adaptive controller presented in Section 4.1.2 was investigated experimentally for the straight line path. The customized mechanical snake robot employed in the experiment is pictured in Fig. 4.7, of which the detailed physical parameters have been reported in the simulation section. The snake robot containing the metal gears is computer-assisted designed and built by the 3D printer, whose module is equipped with a pair of passive wheels to provide anisotropic ground friction properties during motion. A pair of snake robot wheels is assembled with tires and split in 90° to decrease the sliding motion. Two pulse-width-modulation controlled servo motors are invoked for each gear to generate locomotion in the $x-y$ and the $y-z$ planes (in this experiment, the snake robot only moved in the $x-y$ plane). An absolute orientation sensor mounted on the snake robot head is used to measure the snake robot heading angle as θ. During the experiments, we set the coordinate transformation parameter ϵ in (4.2) to $\epsilon = 0$, i.e., $\bar{p}_y = p_y$ as in [33]. To obtain the accurate position information (p_x, p_y), a high-speed motion track-

ing system, the OptiTrack, was configured to capture the three fluorescent spherical markers (12.7mm in diameter) mounted on the snake robot head, shown in Fig. 4.7. This system consisting of 13 cameras and with a sampling rate of 120 frames per second can render the tracking resolution to the millimeter level.

The control objective is to force the snake robot to converge to and finally tracking the desired straight line. The desired line, i.e., the global x-axis, is marked with a yellow line on the floor (see, e.g., Fig. 4.10). The initial conditions of the snake robot are $(p_x, p_y) = (-2.1520, 1.0748)$m, $\theta = -8.1^\circ$, $\phi = 0^\circ$, $v_\theta = 0^\circ$/s, $v_\phi = 0^\circ$/s, $v_t = 0.05$m/s, and $v_n = 0$m/s. Like [33], we only carried out the adaptive joint angle controller given by (4.7), since accurate torque control cannot be gauged by the servo motors installed in the snake robot. The guidance distance $\triangle$ in (4.4) is set as $\triangle = 52.5$cm, which is calculated from half of the length of the snake robot. The gait parameters in (4.51) for the joint angle are $\alpha = 13.5^\circ$, $\omega = 25^\circ$/s, and $\delta = 20^\circ$. The control parameter λ_θ in (4.5) is selected as $\lambda_\theta = 5$. The control gain in (4.7) is chosen as $k_\theta = 40$. The adaptive tuning gains in (4.8) are $k_3 = 0.008$ and $k_4 = 0.0005$. The initial conditions of estimates $\hat{d}_3$ and $\hat{d}_4$ are set to zero.

Experimental results: The trajectory of p_y over the same terrain is pictured in Fig. 4.11 with a dashed line. To demonstrate the capacity of the proposed controller to accommodate unknown and varied friction coefficients, the experimental site is specified with two different frictions terrains. The tangential and normal direction friction coefficients between the robot and the rough terrain (the black ground in Fig. 4.10, i.e., $x \leq 1.5$m) are, respectively, 0.37 and 0.61, and those of the glossy terrain (the green ground in Fig. 4.10, i.e., $x > 1.5$m) are, respectively, 0.21 and 0.34. The experiment results are obtained and presented in Figs. 4.10-4.13. The experimental results with the proposed adaptive controller are provided in Fig. 4.10, which shows that the robot converged perfectly to and moved along the desired path

despite the variation of the friction coefficients. The trajectories of heading angle θ and the reference angle $\bar{\theta}$ are shown in Fig. 4.12, which shows that θ oscillates nicely around $\bar{\theta}$. The controller [33] without the adaptive law was also implemented, and the comparative experimental results are plotted in Fig. 4.14. Under both control methods, the snake robot can practically track the desired path, i.e., $x-$axis, over Terrain 1. Better tracking performance over Terrain 2 is achieved wit our adaptive control method. The experimental result are consistent with the simulation results (e.g., the section of the x-axis from 5m to 30m in Fig. 4.6 and support the conclusion that the proposed adaptive controller can handle ground with unknown friction effectively.

Remark 5 *Compare with the simulation results, the experimental curves have gaps with the simulations due to the mechanical limitation. For instance, the snake robot is moving forward with a sinusoidal manner rather than a smooth curve in the simulation, as shown in Fig. 4.12 and Fig. 4.3. Such that the cross-track errors and the heading errors are also changing with a sinusoidal way. The error vibrations around zero conduct the friction force estimation updating all the time Fig. 4.13 but not converging to a particular value, as shown in the simulation. In addition, the snake robot consists of 16 servo motors, which ideally have the same mechanical features. However, a tiny difference of each servo may be accumulated to significant errors, such that our proposed controller keeps updating the friction force estimation to compensate for the errors caused by the servos. Besides, the snake robot moving velocity also affects the convergence rate, as shown in the simulation results in Fig. 4.8 and Fig. 4.9. Both the trajectory at y-axis and the heading angle spend more time to converge to zero due to the slow velocity. We can conclude from the gaps between the simulation and experimental results that the proposed controller can drive the snake robot approaching the desired path with varied and unknown friction coef-*

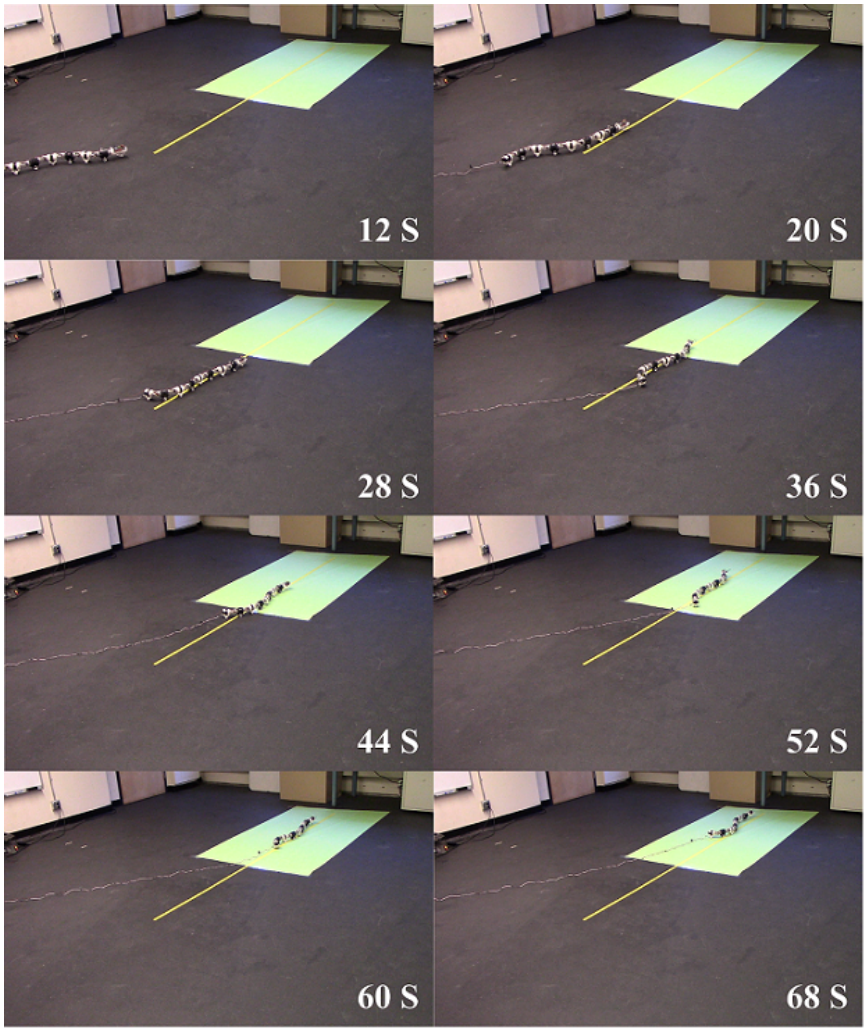


Figure 4.10: Path following of the snake robot under the proposed adaptive controller on two different frictions terrains.

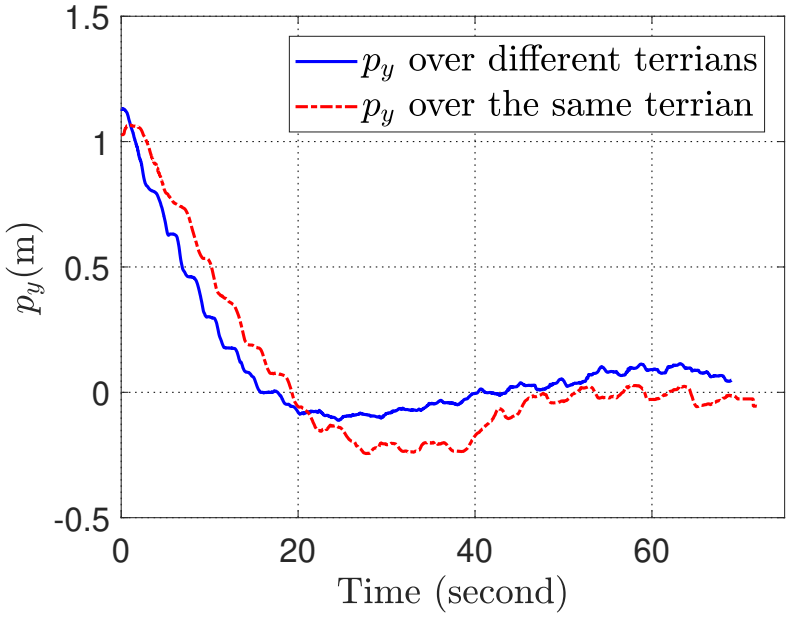


Figure 4.11: Trajectory of p_y.

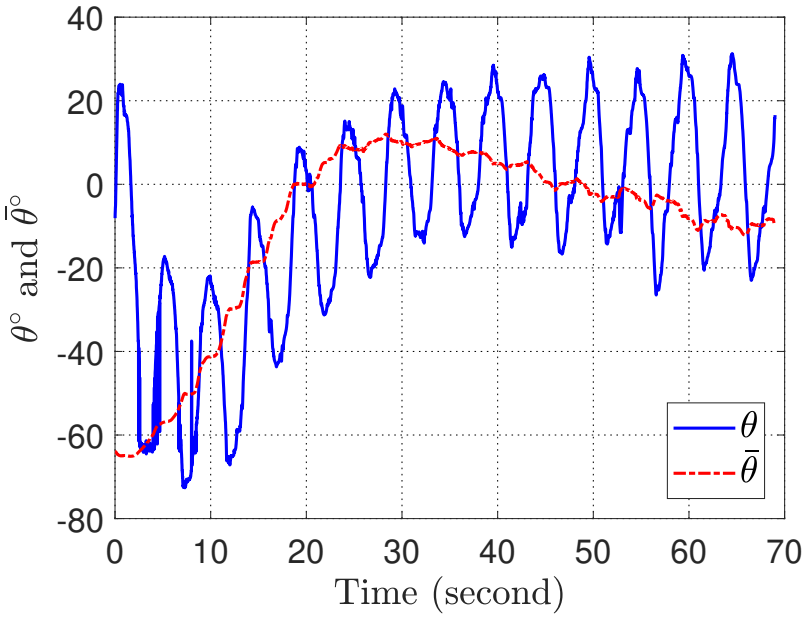


Figure 4.12: Trajectories of θ and $\bar{\theta}$.

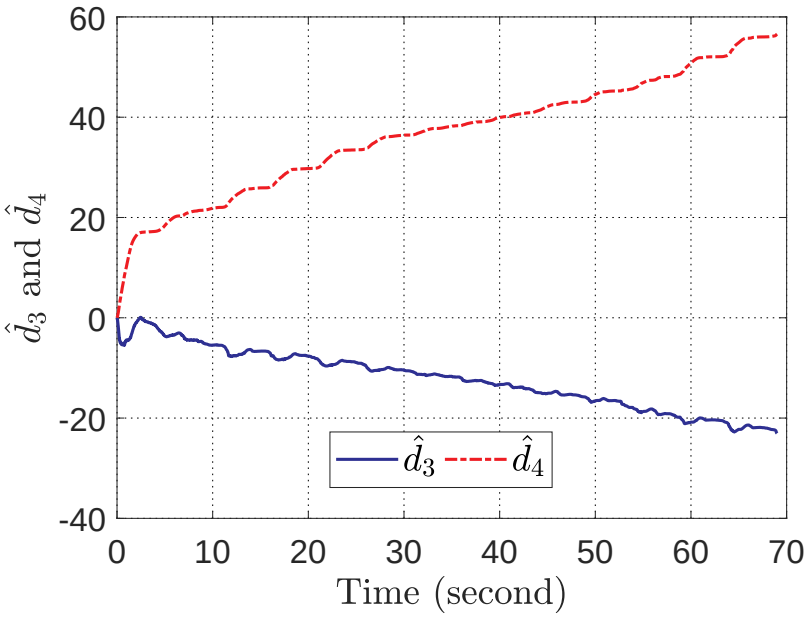


Figure 4.13: Parameter estimates.

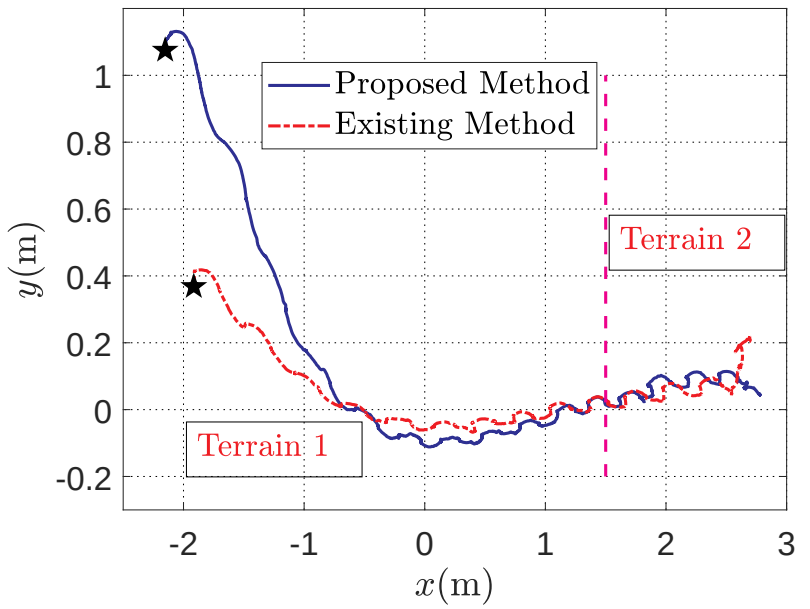


Figure 4.14: Comparative experiment result.

ficient. However, the snake robot can not converge to the straight path as perfect as the simulation, converge to the path quickly with zero errors*, due to the mechanical limitation and different parameter selection.*

In this section, an innovative adaptive path following control method has been presented for planar underactuated snake robots to following a straight path. All friction coefficients have been assumed to be unknown. The proposed controller can compensate for unknown friction coefficients actively and ensure the asymptotic convergence of following errors and the boundedness of parameter estimates. Simulations and experiments on 8-link snake robots were conducted to demonstrate the effectiveness of the proposed controller. Moreover, in the next section, we will discuss the curve path following problem for the snake robot, which makes the snake robot control more helpful in practical applications.

4.2 Parametric Curve Path Following Control

This section investigates a curve path following problem for a class of planar underactuated bio-inspired snake robots. Existing studies mainly focus on straight line or monotonic curve path following problems in relatively simple environments. Based on a time-varying line-of-sight (LOS) guidance law and parametric cubic spline interpolation (PCSI) path-planning method, we propose a solution that can make the snake robot follow an arbitrarily formed curve path on a variety of terrains. The improved LOS helps the snake robot to steer aggressively at a sharp turning point. To avoid the side-slip caused by the ground friction change, an integral controller is introduced in the design of the heading reference. Simulations and experiments on an 8-link custom-built snake robot are conducted, and the results demonstrate and validate the effectiveness of the proposed curve path following algorithm.

As mentioned in Section 4.1, a cascaded system with LOS based navigation is studied to solve the path following problem for a two-dimensional snake robot with unknown and varied frictions. An adaptive control algorithm was proposed in our earlier work [61]. Nevertheless, these control approaches [33, 61] are limited to straight-line path following. Alternative solutions that can also be used for curved path following are presented in [53, 43], are based on hierarchical structure and maneuvering control.

Path-planning, which determines a route when robots are moving from an initial location to their destination, must precede path following. Path-planning selects several fixed points in space from a start point to an end point, namely the waypoints. Then the desired curve path is generated as a series of successive straight lines that pass through the waypoints. However, the aforementioned straight line path following strategy is not applicable to curve following because such a path has a discontinuous first derivative *(velocity function)* at the waypoint. They are many method to generate a continuous curve in two-dimensional, like Dubins path [56], B*é*zier curve [22]. However, these path generation techniques either do not pass through the waypoint or require heavy computation. PCSI and Cubic Hermite Interpolation (CHI) have been extensively studied in path-planning research [4], [59]. For the snake robot mechanical structure, PCSI can produce a smooth curve at the turning point and thus reduce power consumption of the servomotor.

We investigate the curve path following problems of planar underactuated bio-inspired snake robots. Specifically, based on the backstepping technique and PCSI generated path, we design a stable controller with a time-varying LOS guidance system by which the snake robot can traverse from its inital location to the destination through a planned smooth curve. Moreover, the sideslip of the snake robot, caused by the change of friction, is eliminated by the integral control of the LOS method

[41]. We conducted experiments using our algorithm where the snake robot was able to follow an arbitrarily formed curve path, *e.g.,* a closed-loop path, and a cross curve.

4.2.1 Parametric Cubic Spline Interpolation Based Curve Path Generation

The path of the snake robot is specified in terms of n waypoints. The waypoints $p_{wpt}(i)$, $i = 1, \ldots, n$ are defined by the coordinates $p_{wpt}(i) = (x_{wpt}(i), y_{wpt}(i)) \in R^2$, and the set of waypoints can be expressed as

$$\begin{aligned} p_{wpt} \quad = \quad & \{(x_{wpt}(1), y_{wpt}(1)), (x_{wpt}(2), y_{wpt}(2)), \\ & \ldots, (x_{wpt}(n), y_{wpt}(n))\}. \end{aligned} \tag{4.28}$$

The mission of path planning is to generate a desired path that from the starting point $p_{wpt}(1)$ to the terminal point $p_{wpt}(n)$. PCSI and CHI are popular choices for curve fitting for ease of data interpolation. The two methods use different formulas to compute the slopes at waypoints. In this dissertation, the PCSI is employed for the path generation for each pair of successive waypoints in the form of $(p_{wpt}(i), p_{wpt}(i+1)), i = 1, \ldots, n-1$, which is also called a spline. A generated path is a sequence of order splines. This entails the introduction of the independent variable s, which is defined as $\dot{s} = f(s, t)$. For the formulation of one separate spline, the corresponding path variable is denoted as $s_i, i = 1, \ldots, n-1$. Based on [4], the cubic polynomials of a desired curve f_d between two waypoints $(x_{wpt(i)}, y_{wpt}(i))$ and $(x_{wpt(i+1)}, y_{wpt}(i+1))$, is denoted as $f_d(s) = (x_d(s), y_d(s))$, where $(x_d(s), y_d(s))$ is the position of the desired curve

$$x_d(s) = a_3(s - s_i)^3 + a_2(s - s_i)^2 + a_1(s - s_i) + a_0, \tag{4.29}$$

$$y_d(s) = b_3(s - s_i)^3 + b_2(s - s_i)^2 + b_1(s - s_i) + b_0, \tag{4.30}$$

where s_i is the value of the independent variable s at the beginning of the spline. The constant parameters are $a_0, a_1, a_2, a_3, b_0, b_1, b_2$, and b_3. The first-order derivative of $f_d(s)$ with respect to s is $\dot{f}_d(s) = (\dot{x}_d(s), \dot{y}_d(s))$

$$\dot{x}_d(s) = 3a_3(s - s_i)^2 + 2a_2(s - s_i) + a_1, \tag{4.31}$$

$$\dot{y}_d(s) = 3b_3(s - s_i)^2 + 2b_2(s - s_i) + b_1. \tag{4.32}$$

Furthermore, the second-order derivative of $f_d(s)$ with respect to s is $\ddot{f}_d(s) = (\ddot{x}_d(s), \ddot{y}_d(s))$

$$\ddot{x}_d(s) = 6a_3(s - s_i) + 2a_2, \tag{4.33}$$

$$\ddot{y}_d(s) = 6b_3(s - s_i) + 2b_2. \tag{4.34}$$

The CHI needs the first-order derivative $\dot{f}_d(s)$ to be continuous at the waypoint $p_{wpt}(i)$. The PCSI is developed from standard cubic spline interpolation (CSI), and it has the second-order derivative $\ddot{f}_d(s)$ continuous at the waypoint $p_{wpt}(i)$. In addition, PCSI also allows choosing boundary conditions for special curve [4]. For instance, in the case of a closed-loop, the first and last waypoints are subjected to the same first and second derivative compatibility constraints. Practically, PCSI allows generating arbitrary smooth curves for snake robot to move from one position to the destination. Considering the specificity of snake robot mechanical structure, we choose PCSI to generate curve paths. A cubic polynomial (4.29) is considered to obtain the governing equation for cubic spline, In order to solve the four unknown parameters of (4.29) (we similarly for solve the four unknown parameters of polynomial (4.30)), four constraints must be satisfied:

1. The path through the waypoints $(x_{wpt}(i), y_{wpt}(i))$ and $(x_{wpt}(i+1), y_{wpt}(i+1))$ must satisfy $x_d(s_i) = x_{wpt}(i)$ and $x_d(s_{i+1}) = x_{wpt}(i+1)$.

2. At the waypoints, the slopes (first derivatives) must be equal, which means that the slope between two intervals at the waypoint must be continuous, *i.e.,* $\dot{x}_d(s_i) = \dot{x}_d(s_{i+1})$.

3. The slopes of the first and the last interval at the waypoint are equal, that is, $\dot{x}_d(s_1) = \dot{x}_d(s_n)$.

4. The second derivatives of the first and the last intervals at waypoint are equal, namely, $\ddot{x}_d(s_1) = \ddot{x}_d(s_n)$.

For PCSI, the first two conditions must be used in the parameter calculations while the last two constraints can be specified by different curves. In contrast, CHI has only one constraint to customize. PCSI is selected to illustrate a closed-loop path by determining the third and fourth conditions, which are used later in the experiment. The governing equation for cubic splines can be derived by considering the first two splines $x_d(s_1)$ and $x_d(s_2)$, and the corresponding s_i are $x_{wpt}(1)$ and $x_{wpt}(2)$, respectively. The 4 unknowns can be obtained by imposing the constraint that $\ddot{y}_d(s) = \ddot{y}_d(s_1)$ at $x_{wpt}(1)$ and $\ddot{x}_d(s) = \ddot{x}_d(s_2)$ at $x_{wpt}(2)$, which can be expressed as

$$\ddot{x}_d(s_1) = 6a_3(x_{wpt}(1) - x_{wpt}(1)) + 2a_2, \tag{4.35}$$

$$\ddot{x}_d(s_2) = 6a_3(x_{wpt}(2) - x_{wpt}(1)) + 2a_2. \tag{4.36}$$

By substituting (4.35) into (4.36), the values of a_3 and a_2 are calculated:

$$a_2 = \frac{\ddot{x}_d(s_1)}{2}, \tag{4.37}$$

$$a_3 = \frac{\ddot{x}_d(s_2) - \ddot{x}_d(s_1)}{6h_1}, \tag{4.38}$$

where $h_i = x_{wpt}(i+1) - x_{wpt}(i)$ is the point space. Next, we utilize the constraint

that each polynomial must pass through the waypoints, *i.e.,* $x_d(s_1) = x_{wpt}(1)$ and $x_d(s_2) = x_{wpt}(2)$

$$\begin{aligned} x_{wpt}(1) =& a_3(x_{wpt}(1) - x_{wpt}(1))^3 + a_2(x_{wpt}(1) - \\ & x_{wpt}(1))^2 + a_1(x_{wpt}(1) - x_{wpt}(1)) + a_0, \\ x_{wpt}(2) =& a_3(x_{wpt}(2) - x_{wpt}(1))^3 + a_2(x_{wpt}(2) - \\ & x_{wpt}(1))^2 + a_1(x_{wpt}(2) - x_{wpt}(1)) + a_0, \end{aligned} \tag{4.39}$$

which leads to

$$a_1 = \frac{x_{wpt}(2) - x_{wpt}(1)}{h_1} - \frac{h_1(\ddot{x}_d(s_2) - 2\ddot{x}_d(s_1))}{6}, \tag{4.40}$$

$$a_0 = x_{wpt}(1). \tag{4.41}$$

Finally, we impose the constraint that the slope must be equal at the waypoint, *i.e.,* $\dot{x}_d(s_1) = \dot{x}_d(s_2)$ at $x_{wpt}(2)$. That is

$$\begin{aligned} & 3a_3(x_{wpt}(2) - x_{wpt}(1))^2 + 2a_2(x_{wpt}(2) - x_{wpt}1)) + a_1 \\ & = 3a_3^*(x_{wpt}(2) - x_{wpt}(2))^2 \\ & + 2a_2^*(x_{wpt}(2) - x_{wpt}(2)) + a_1^*, \end{aligned} \tag{4.42}$$

where a_1^*, a_2^*, and a_3^* are the parameters of spline two. From the above equations (4.43) we can obtain the governing equation for PCSI

$$\begin{aligned} & h_1\ddot{x}_d(s_1) + 2(h_1 + h_2)\ddot{x}_d(s_2) + h_2\ddot{x}_d(s_3) \\ =& 6\left[\frac{x_{wpt}(3) - y_{wpt}(2)}{h_2} - \frac{x_{wpt}(2) - x_{wpt}(1)}{h_1}\right]. \end{aligned} \tag{4.43}$$

The PCSI governing equation results in a general tri-diagonal set of linear equations $\mathcal{A}\mathcal{X} = b$, where $\mathcal{X}$ is the unknown second derivatives $\ddot{x}_d(s_i)$, $\mathcal{A}$ is the left-hand

side of the governing equation (4.43), and b is the right-hand side of (4.43). For simplification, an equal point space (*i.e.,* $h_1 = h_2 = \cdots = h_i = h$) is used

$$\begin{bmatrix} ? & & & & & & 0 \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & \ddots & & \\ & & & \ddots & & & \\ & & & & \ddots & 1 & \\ & & & & 1 & 4 & 1 \\ 0 & & & & & & ? \end{bmatrix} \begin{bmatrix} \ddot{x}_d(s_1) \\ \ddot{x}_d(s_2) \\ \vdots \\ \vdots \\ \vdots \\ \ddot{x}_d(s_{n-1}) \\ \ddot{x}_d(s_n) \end{bmatrix} = 6/h^2 \begin{bmatrix} ? \\ x_{wpt}(3) - 2x_{wpt}(2) = x_{wpt}(1) \\ x_{wpt}(4) - 2x_{wpt}(3) = x_{wpt}(2) \\ \vdots \\ \vdots \\ x_{wpt}(n) - 2x_{wpt}(n-1) = x_{wpt}(n-2) \\ ? \end{bmatrix}. \tag{4.44}$$

The first and the last equations (denoted by ?) represent the boundary conditions of the free ends of spline that must be determined. For demonstrating an experimental closed-loop path passing through 5 waypoints, the first boundary condition, $\dot{x}_d(s_1) = \dot{x}_d(s_5)$, can be used to derive the following equation

$$\begin{aligned} &2h\ddot{x}_d(s_1) + h\ddot{x}_d(s_2) + h\ddot{x}_d(s_4) + 2h\ddot{x}_d(s_5) \\ =&\frac{6(x_{wpt}(2) - x_{wpt}(1) - x_{wpt}(5) + x_{wpt}(4))}{h^2}. \end{aligned} \tag{4.45}$$

The last boundary condition, $\ddot{x}_d(s_1) = \ddot{x}_d(s_5)$, can be written as

$$\ddot{x}_d(s_1) - \ddot{x}_d(s_5) = 0. \tag{4.46}$$

Substituting equations (4.45) and (4.46) in the tri-diagonal linear equation (4.44), the values of second derivatives of $\ddot{x}_d(s)$ are calculated. Furthermore, we can determine all the coefficients of the polynomials $x_d(s)$ through equations (4.37), (4.38), (4.40), and (4.41). Replacing x with y, we can obtain the parameters of $y_d(s)$ in the same manner.

This section presents a path generation method PCSI in details. The PCSI allows two degrees of freedom of the boundary conditions to be specified to form different curves. For instance, a so-called *nature* boundary condition is used to achieve the total minimum curvature $\ddot{x}_d(s) = \ddot{y}_d(s) = 0$, a *zero slope* boundary condition, $\dot{x}_d(s) = \dot{y}_d(s) = 0$, and the boundary condition we used in the experiment of the closed-loop path following $\ddot{x}_d(s_1) = \ddot{x}_d(s_n)$. The PCSI is employed in this dissertation to: 1) generate a curved path for snake robot with continuous first and second order derivatives between the successive waypoints, which ensures a smooth velocity function along the path, and 2) prove the stability of the time-varying LOS described in Section 4.2.2 when the snake robot is converging to the desired curved path.

With the snake robot kinematic model described in section (3.1) and the aforementioned snake robot dynamic model (3.3), the snake robot curve path following control objective is presented. The control objective is to drive the snake robot to approach the curved path generated in Section 4.2.1 using the LOS steering law. First, the values of the independent variable s of the desired curve path $f_d(s) = (x_d(s), y_d(s))$ must be determined. The Newton-Raphson method is employed to find the root of the third-order cubic polynomial for $\bar{s}$. The x-coordinates of the snake robot $p_x(t)$ is matched with the x-coordinates of the desired curve $x_d(\bar{s})$

to set the along-track error $a_s(t) = p_x(t) - x_d(\bar{s})$ to zero. The position of snake robot at time t is $(p_x(t), p_y(t))$. Our control objective is to minimize the normal distance between the snake robot and the desired curve path which is defined as the magnitude of the cross-track error $e_s(t) = p_y(t) - y_d(s)$. This subsection is to derive the analytical expression of $e_s(t)$.

Since we only consider the problem of minimizing the distance in the normal direction, the normal line through the point $(x_d(s), y_d(s))$ can be expressed by

$$p_y(t) - y_d(s) = -\frac{\dot{x}_d(s)}{\dot{y}_d(s)}(p_x(t) - x_d(s)). \tag{4.47}$$

Then we calculate the x-coordinate of the line going through the point $(p_x(t), y_d(\bar{s}))$ with a slope of $\dot{y}_d(\bar{s})$. Following equation (4.47), we have $\dot{y}_d(\bar{s})(p_y(t) - y_d(\bar{s})) = -\dot{x}_d(\bar{s})(p_x(t) - x_d(\bar{s}))$. This equation is used to iteratively solve for the roots of the third-order cubic function to obtain $\bar{s}$. It is shown in Section 4.2.1 that the slope (first derivative $\dot{y}_d(\bar{s})$) is continuous at the waypoint except for the initial and the final points. The Newton-Raphson method must quickly converge with

$$\bar{s}_{j+1} = \bar{s}_j - \frac{y_d(\bar{s}_j)}{\dot{y}_d(\bar{s}_j)}, \tag{4.48}$$

where j is the iteration number. By choosing a reasonable initial value close to the true root, the algorithm converges rapidly after a few iterations. The normal line from the point $(x_d(\bar{s}), y_d(\bar{s}))$ on the path through the point $(p_x(t), p_y(t))$ defines the cross-track error $e_s(t) = p_y(t) - y_d(\bar{s})$. Multiplying the cross-track error by the transition matrix in the global coordinate system, we have the expression of $e_s(t)$

$$e_s = -(p_x(t) - x_d(\bar{s}))\sin(\alpha) + (p_y(t) - y_d(\bar{s}))\cos(\alpha), \tag{4.49}$$

where $\alpha = \arctan(\dot{y}_d(\bar{s}), \dot{x}_d(\bar{s}))$. As mentioned at the beginning of this section, the control goal is to minimize the cross-track error with the help of the LOS system, which is

$$\lim_{t\to\infty} e_s(t) = 0. \tag{4.50}$$

We use the Newton-Raphson method to minimize the cross-track error. We program the snake robot to have an initial velocity to avoid singularity when we use the Newton-Raphson method to find the root of the cubic polynomials in Section 4.2.1.

4.2.2 Time-varying LOS based Guidance Law Design

4.2.2.1 Time Varying LOS-Based Steering

Before presenting the procedure of snake robot curve path following controller, the snake robot gait pattern is described. Inspired by the snake sinusoidal creeping locomotion [48, 39], the reference for the i^{th} joint angle of the robot is described as

$$\bar{\phi}_i = A_m \sin(\omega t + (i-1)\delta) + \phi_o, \tag{4.51}$$

where A_m and ω denote, respectively, the amplitude and frequency of the sinusoidal locomotion, δ is the phase shift between the joints, and ϕ_0 is the joint offset to steer the snake robot.

To steer the snake robot to the desired path generated by the PCSI method of Section 4.2.1 and because sideslip of the snake robot may occur when friction varies, an integral LOS based guidance law interpreted as the saturated control law is utilized [13]

$$\bar{\theta} = -\arctan\left(K_p e_s - K_i \int_0^t e_s(\tau)\, d\tau\right), \tag{4.52}$$

where $e_s = p_y - y_d$ is the cross-track error, $K_p = 1/\Delta$ with lookahead distance $\Delta > 0$,

and $K_i > 0$ is the integral gain. The reference steering angle is now proportional to e_s with a constant gain K_p. The LOS steering law ensures that the snake robot's heading angle is directed toward the waypoint (x_n, y_n) until the snake robot converges to the designed curve path. Furthermore, integral control compensates for the steering angle if sideslip occurs [13].

To achieve smooth snake robot steering control, a time-varying lookahead distance is considered as

$$\Delta = (\Delta_{\max} - \Delta_{\min})e^{-K_\Delta e_s^2} + \Delta_{\min} \tag{4.53}$$

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. Typically, $\Delta_{\max} = 1.3L$ and $\Delta_{\min} = 0.4L$, where L is the length of the snake robot.

LOS guidance ensures the snake robot is directed toward the desired waypoint (x_i, y_i) until it converges to the curve path and the control objective (4.50) is satisfied. Differentiating the cross-track error (4.49) with respect to time, we have

$$\begin{aligned}
\dot{e}_s =& -[\dot{p}_x - \dot{x}_d(\bar{s})]\sin(\alpha) + [\dot{p}_y - \dot{y}_d(\bar{s})]\cos(\alpha) \\
& - [p_x - x_d(\bar{s})]\cos(\alpha)\cdot\dot{\alpha} - [p_y - y_d(\bar{s})]\sin(\alpha)\cdot\dot{\alpha} \\
=& \underbrace{-\dot{p}_x\sin\alpha + \dot{p}_y\cos\alpha}_{term_1} + \underbrace{\dot{x}_d(\bar{s})\sin\alpha - \dot{y}_d(\bar{s})\cos\alpha}_{term_2} \\
& \underbrace{-\dot{\alpha}([p_x - x_d(\bar{s})]\cos\alpha + [p_y - y_d(\bar{s})]\sin\alpha).}_{term_3}
\end{aligned} \tag{4.54}$$

Equation (4.54) is divided into 3 terms. Transforming term 1 in amplitude-phase form, we have $term_1 = \sqrt{\dot{p}_x^2 + \dot{p}_y^2}\sin\alpha = V\sin\alpha$. Similarly, term 2 is expressed as $term_2 = \sqrt{\dot{x}_d^2 + \dot{y}_d^2}\sin(\alpha + \varsigma)$, where ς is designed as $\varsigma = \arctan(-v_n, v_t) = -\alpha$ thus term 2 equals zero. Term 3 is the along-track error which is also zero. If the snake

robot tracks the curve path perfectly, replace α by $\bar{\theta}$

$$\alpha = -\arctan\left(K_p e_s - K_i \int_0^t e_s(\tau)\, d\tau\right). \tag{4.55}$$

If the integral part of the equation (4.55) is a non-zero term, we obtain $\dot{e}_s = V\sin(\arctan(e_s + K_i e_{int})/\Delta))$. Using the trigonometric identities, we have $\dot{e}_s = -V\frac{e_s+K_i e_{int}}{\sqrt{\Delta^2+(e_s+K_i e_{int})^2}}$, where the derivative of e_{int} is expressed as

$$\dot{e}_{int} = \frac{V e_s}{\sqrt{\Delta^2 + (e_s + K_i e_{int})^2}} \tag{4.56}$$

We construct the following Lyapunov function candidate $L_1 = e_s^2/2 + K_i e_{int}^2/2$, whose derivative with respect to time is

$$\begin{aligned}\dot{L}_1 &= -V\frac{e_s^2 + e_s K_i e_{int}}{\sqrt{\Delta^2 + (e_s + K_i e_{int})^2}} + K_i e_{int}\dot{e}_{int} \\ &= \frac{-V e_s^2}{\sqrt{\Delta^2 + (e_s + K_i e_{int})^2}} \\ &\quad - K_i e_{int}\left(\frac{e_s V}{\sqrt{\Delta^2 + (e_s + K_i e_{int})^2}} - \dot{e}_{int}\right)\end{aligned} \tag{4.57}$$

Substituting equation (4.56) into equation (4.57), we finally have

$$\dot{L}_1 = -V\frac{e_s^2}{\sqrt{\Delta^2 + (e_s + K_i e_{int})^2}} \leq 0 \tag{4.58}$$

4.2.2.2 Controller Design

Because the dynamics of the snake robot modeled by (3.20) is high-order and non-linear with a mismatched condition, the control design is performed by following a step-by-step procedure known as backstepping technique [28]. The detailed design procedure is given as follows.

Step 1: Design the joint offset ϕ_o such that the heading angle θ converge to the LOS guidance law (4.52). To do so, the heading angle error variable is defined as

$$z_1 = \zeta_\theta - \zeta_{\bar{\theta}}, \zeta_\theta = \theta + \kappa\dot{\theta}, \zeta_{\bar{\theta}} = \bar{\theta} + \kappa\dot{\bar{\theta}}, \tag{4.59}$$

where κ is a positive constant. Taking the time derivative of z_1, we obtain

$$\begin{aligned} \dot{z}_1 &= v_\theta + \kappa(-c_3 v_\theta + \tfrac{c_4}{N-1} v_t \bar{e}^T \phi) - \dot{\zeta}_{\bar{\theta}} \\ &= \kappa(-c_3 v_\theta + c_4 v_t \phi_o + \tfrac{c_4 v_t}{N-1} \bar{e}^T(\phi - \bar{\phi})) + \eta, \end{aligned} \tag{4.60}$$

where $\eta = v_\theta - \dot{\zeta}_{\bar{\theta}} + \frac{\lambda c_4 v_t}{N-1} \sum_{i=1}^{N-1} \alpha \sin(\omega t + (i-1)\delta)$.

We choose a Lyapunov function candidate for this step as $L_2 = z_1^2/2$. Taking the derivative of L_2 along (4.60) gives

$$\dot{L}_2 = \kappa(-c_3 v_\theta + c_4 v_t \phi_o + \frac{c_4 v_t}{N-1} \bar{e}^T(\phi - \bar{\phi}))z_1 + \eta z_1. \tag{4.61}$$

We choose the joint offset ϕ_o as

$$\phi_o = \tfrac{c_3}{c_4 v_t} v_\theta + \tfrac{1}{c_4 \kappa v_t}(-\lambda_1 z_1 - \eta), \tag{4.62}$$

where λ_1 is a positive control gain. Substituting (4.62) into (4.61) gives

$$\dot{L}_2 = -\lambda_1 z_1^2 + \frac{\kappa c_4 v_t}{N-1} \bar{e}^T(\phi - \bar{\phi}) z_1. \tag{4.63}$$

Now we have completed the design of the joint reference coordinates $\bar{\phi}_i$ given by (4.51).

Step 2: Define the link angle error $z_2 = \phi - \bar{\phi}$. and select the Lyapunov function

candidate L_3 for this step as $L_3 = L_2 + \frac{1}{2} z_2^T z_2$. From (3.20) and (4.63), we have

$$\dot{L}_3 = -\lambda_1 z_1^2 + \frac{\kappa c_4 v_t}{N-1} \bar{e}^T z_2 z_1 + z_2^T (v_\phi - \dot{\bar{\phi}}). \tag{4.64}$$

We then choose the virtual joint velocity $\bar{v}_\phi$ as

$$\bar{v}_\phi = -\frac{\kappa c_4 v_t}{N-1} \bar{e} + \dot{\bar{\phi}} - \lambda_2 z_2. \tag{4.65}$$

where λ_2 is a positive constant. Substituting (4.65) into (4.64) yields

$$\dot{L}_3 = -\lambda_1 z_1^2 - \lambda_2 z_2^T z_2 + z_2^T (v_\phi - \bar{v}_\phi). \tag{4.66}$$

Step 3: Set the actuator forces as follows

$$u = m(DD^T)^{-1}(\dot{\bar{v}}_\phi - \lambda_3 z_3 - z_2 + \frac{c_1}{m} v_\phi - \frac{c_2}{m} v_t A D^T \phi), \tag{4.67}$$

where $z_3 = v_\phi - \bar{v}_\phi$, λ_3 is a positive constant.

Choose the Lyapunov function for the overall system as

$$L_4 = L_2 + L_3 + \frac{1}{2} z_3^T z_3. \tag{4.68}$$

From (4.63), (4.66), and (4.67), we can obtain

$$\dot{L}_4 = -\lambda_1 z_1^2 - \lambda_2 z_2^T z_2 - \lambda_3 z_3^T z_3 \leq 0. \tag{4.69}$$

Theorem 2 *For the system (3.20), the path following controller defined by (4.62) and (4.67) meets the control objective.*

Proof. Applying the LaSalle-Yoshizawa theorem to (4.69), it follows that $\lim_{t\to\infty} z_1(t) =$

0. Recalling (4.59), we conclude that $\lim_{t\to\infty} \theta(t) - \bar{\theta}(t) = 0$. The rest of the proof is the same as that of [61], but omitted here.

4.2.3 Monotonic Curve Path Following

Simulation Setup: Using the snake robot dynamics (3.20) studied in Section 3.3, with the integral time-varying LOS guidance law (4.52), the performance of the proposed controller (4.62) is investigated through MATLAB simulation. We illustrate the curve path following function by passing through five waypoints, which comprise three types of curves by the PCSI method introduced in Section 4.2.1, *i.e.,* monotonic curve, closed-loop curve, and cross-line curve. For a snake robot, these three types of curves meet the path planning requirements of most of practical applications. The snake robot in our Robotics and Biomimetics Laboratory has a length of $L = 100$cm, 8 links of mass $m = 1.08$kg. The initial tangent and normal velocity of snake robot are given by $v_t(0) = 20$cm/s and $v_n(0) = 12$cm/s. To initialize the LOS guidance system, the parameters are chosen as: convergence rate $K_\Delta = 2$, proportional constant gain $K_p = 0.3$ and integral gain $K_i = 0.01$. Newton-Raphson method is used to find the roots of the third-order cubic function for the independent path variable s. The algorithm must converge after few iterations of equation (4.48). We use an initial value of $s = 100$, and stop the iteration after the change is less than 10^{-3}.

Experimental Setup: The performance of the controller proposed in Section 4.2.2 is evaluated experimentally in this subsection. Corresponding to the simulations of three types of curve paths following, the snake robot must follow the three curves to validate the effectiveness of the proposed path following system. A mechanical snake robot in the first frame of Fig. 4.18 is employed in experiments whose physical parameters are given in the simulation section. Each joint has a power system that includes two pulse-width-modulation (PWM) controlled servomotors DS450. Two sets

of metal gears drive each joint to generate motion both on $(x-y)$ and $(y-z)$ planes (in this experiment, only $(x-y)$ plane motion is considered). Two passive wheels assembled with tires are equipped with a split angle of 45° to provide anisotropic ground friction properties during the motion. An Arduino-M0 micro-controller along with a ZigBee wireless transmission chip, and an Inertial-Measurement-Unit (IMU) chip are integrated in a single customized Printed Circuit Board (PCB) to provide the control signal for each servomotor.

The first simulation demonstrates that a snake robot follows a monotonic curve consisting of 5 waypoints, *i.e.*, $x_{wpt}(i+1) > x_{wpt}(i), i = 1, \ldots, 4$. This could occur at a scenario where the snake robot passes through obstacles to reach its destination. For the aforementioned snake robot dynamics, the initial position is set as $(p_x(0), p_y(0)) = (5, 25)$. The simulated snake robot's path following trajectory, time-varying LOS and tracking error (including cross-track error and along-track error) are plotted in Fig. 4.15, Fig. 4.16, and Fig. 4.17, respectively. In the simulation, the snake robot

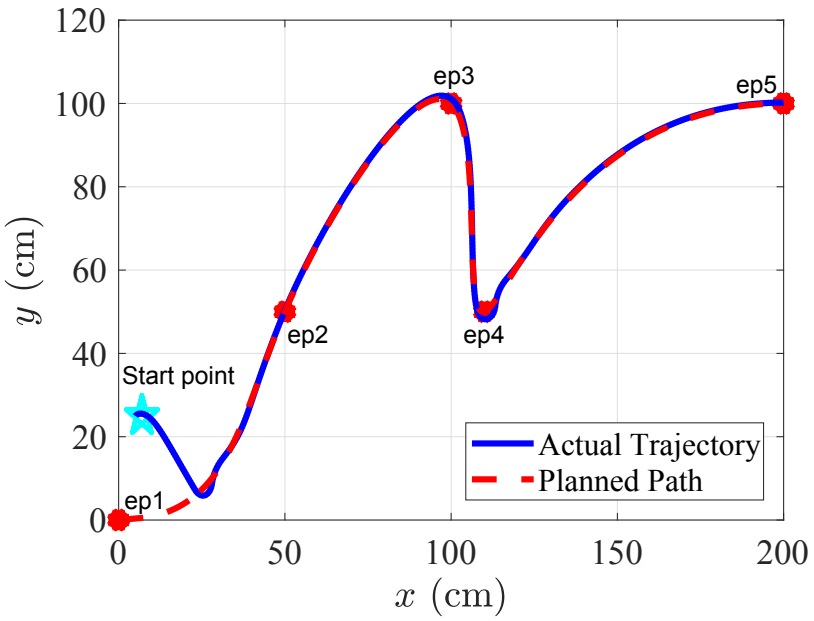


Figure 4.15: Snake robot tracking trajectory on a monotonic curve.

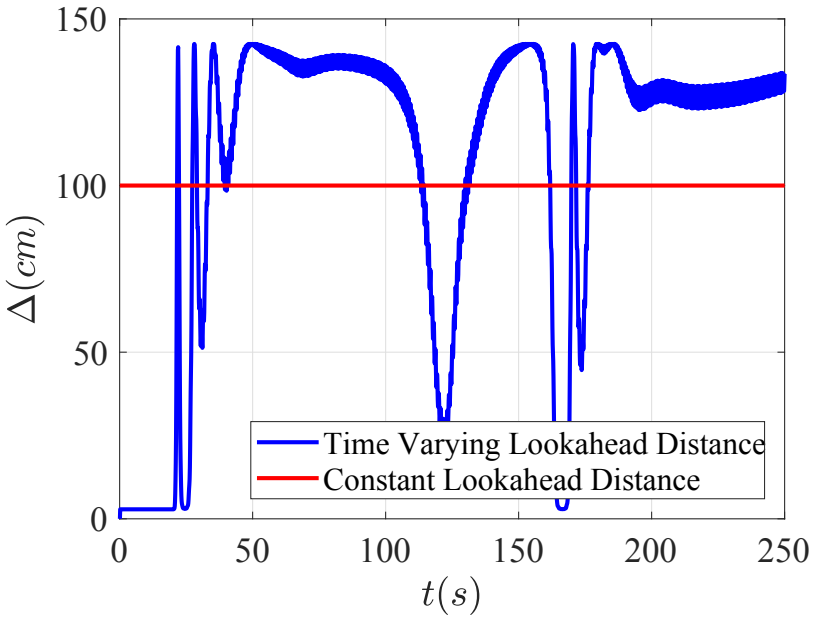


Figure 4.16: Lookahead distance varies to achieve appropriate steering.

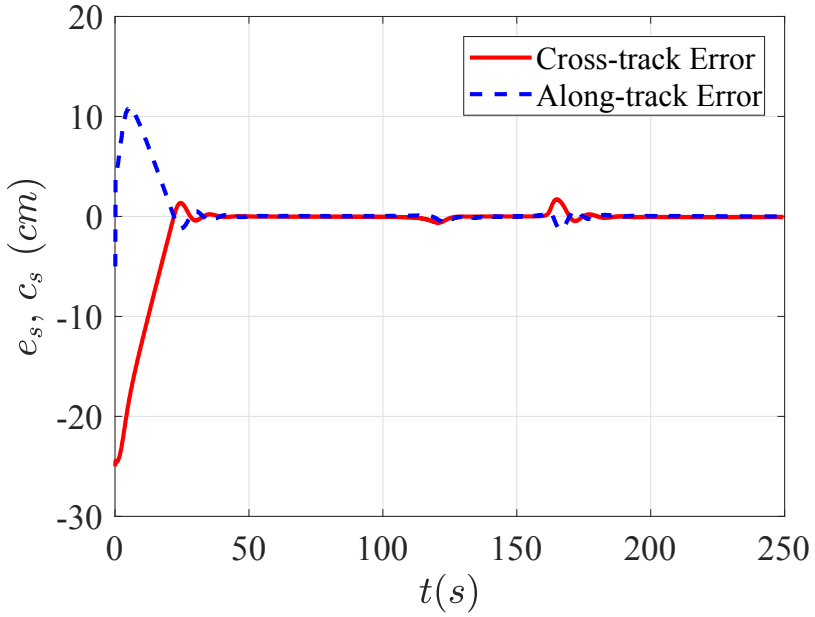


Figure 4.17: The errors e_s and c_s converge to zero in a short time.

converges to the desired monotonic curve path within $30s$ and continues to follow the smooth curve generated by the PCSI. The curvature changes dramatically at waypoint $ep3$ and $ep4$. With time-varying LOS, the snake robot achieves aggressive steering at waypoint ep4 as shown in Fig. 4.16. The snake robot follows the path with a small error. The slope of the path connecting ep3 and ep4 has a different sign from the slope of the path connecting ep4 and ep5, which leads to a small tracking error at the waypoint ep4. The tracking error shown in Fig. 4.17 also validates that the snake robot does not change its position and orientation very fast and demonstrates the effectiveness of the path following control. The proposed method improves performance when more difficult maneuvering is required. Ignoring the along-track error (it is indeed zero all the time due to the Newton-Raphson method), the cross-track error is almost zero all the time except at the steering waypoint ep4. As for the shape of the path, the PCSI produced curve passes through all the selected waypoints and the x-coordinates of the curve monotonically increase between each pair of waypoints without wiggling of zigzagging. This is attributed to the aforementioned first-order constraint and the second-order constraint in Section 4.2.1, which results in a more practical and tractable path.

The experiments are conducted to demonstrates the snake robot path following function in a monotonic curve route. Due to the difference of the coordinate system, five points are set as ep1 $= (-218, 8.19)$cm, ep2 $= (-123, -105.2)$cm, ep3 $= (17, -15)$cm, ep4 $= (143, 38)$cm, and ep5 $= (280, 0)$cm. To obtain accurate snake robot position information, a high-speed motion capture system, OptiTrack, is used. OptiTrack is able to capture the motion of snake robot at the rate of 120Hz with three fluorescent spherical markers (12.7mm in diameter) mounted on the snake robot head, shown in the first picture of Fig. 4.18.

The algorithm runs on the computer with Ubuntu 16.04 operation system for

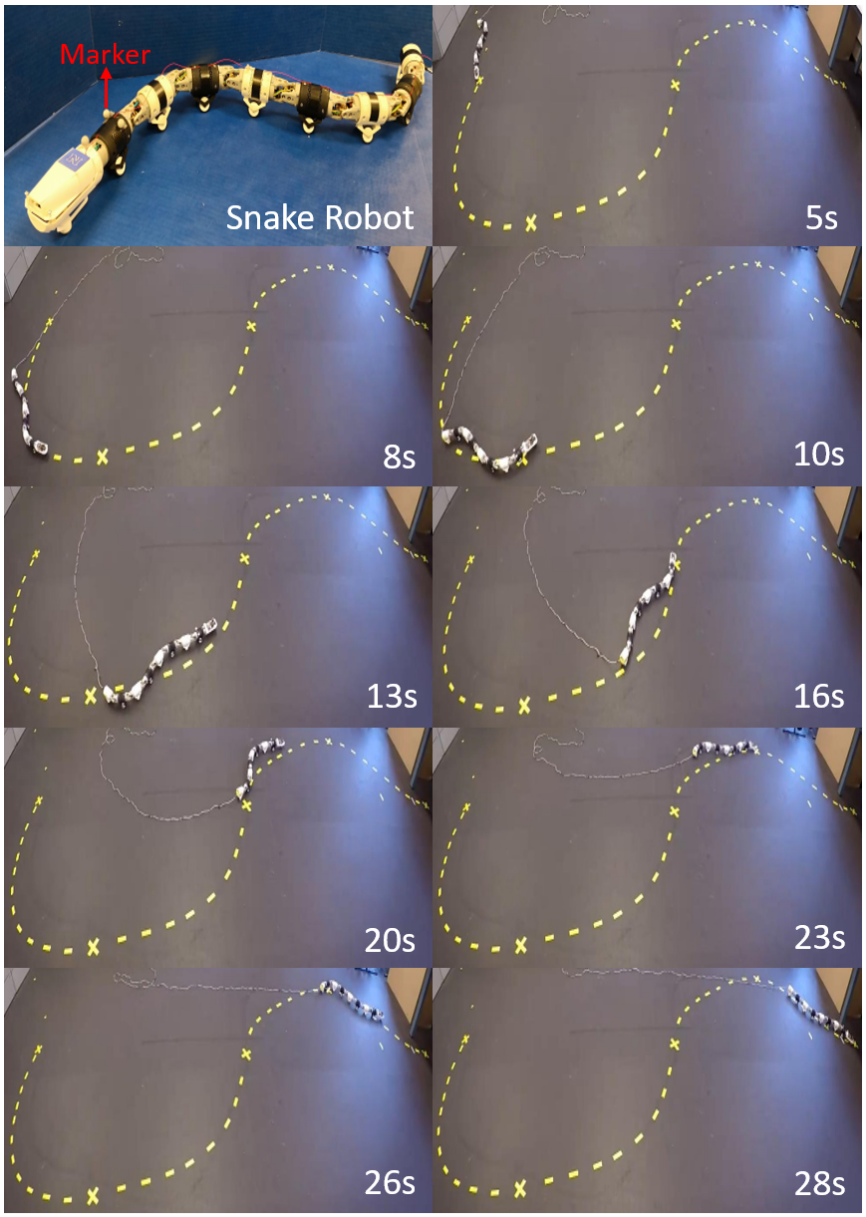


Figure 4.18: Monotonic curve path following experiment of the snake robot under the proposed controller. The first frame is the customized mechanical snake robot we used in the experiment. All these frames are taken from an experimental movie.

instant data feedback. The control objective is to force the snake robot to approach the desired monotonic curve path, which is marked as a yellow dash line on the floor. For experimental repeatability, we manually select the start point $p_0 = (-220, 60)$cm although the algorithm converges from any start point. The snake robot is initialized with parameters of equation (4.51) that $A_m = 13.5$cm, $\omega = 25°/s$, $\delta = 20°$, $\theta = 30°$, $v_\theta = 0°/s$, $v_\phi = 0°/s$, $v_t = 5$cm/s, and $v_n = 0$cm/s. The Newton-Raphson method introduced in Section 4.2.1 is employed to find the roots of the third-order cubic equations (4.29-4.30) for s. As indicated in the Newton-Raphson equation (4.48), $\dot{y}_d(\bar{s})$ and $\dot{x}_d(\bar{s})$ cannot be zero to avoid singularity. Referring to equations (4.32), (4.34), and (4.40), the system must satisfy three conditions to avoid singularity:

1. Point space $h_i \neq 0$,
2. x-coordinates of two successful waypoints are not equal, *i.e.,* $x_{wpt}(i+1) \neq x_{wpt}(i)$,
3. y-coordinates of two successful waypoints are not equal, that is, $y_{wpt}(i+1) \neq y_{wpt}(i)$,

The snake robot is programmed to avoid a singularity. All other parameters are exactly the same as those in the simulation.

Experimental results: The monotonic curve path following experimental results with the proposed algorithm are presented in Fig. 4.18. The curve trajectory of the snake robot passing through all five waypoints is shown in Fig. 4.19. As expected, the trajectories of four repeated experiments almost completely coincide except at the steering waypoints ep2 and ep4. The slope sign at waypoint ep2 is opposite to that at ep4, which results in a different deviation. this can also be verified from the cross-track error e_s shown in Fig. 4.20. The e_s error departs from zero at 7s and 20s but goes back to zero after 25s. It can be concluded that the proposed controller

converges quickly with no significant error. It is worth noting the shape change of e_s before the first approach. The simulation indicates that it is normal to see a sudden increment or drop in the error before the snake robot reaches the desired path. That is because the actual heading of the snake robot is not equal to the desired heading, and the heading dynamic will take more time to converge. This conclusion also can be found in the plot of time-varying LOS Δ shown in Fig. 4.20. Δ decreases right after it reaches the maximum at around 2.5 second, because the snake robot deviate a bit after it reaches the desired path, the Δ varies to help the snake robot turning. Since the cross-track error e_s is deviates from zero at 7s and 20s depicted in Fig. 4.20, the Δ decreases at these two points to achieve aggressive steering angles. The proposed method promises improved performance to provide good maneuverability for snake robots. As the Newton-Raphson method indicates, the along-track error is kept near zero as shown in Fig. 4.20. Due to the compensation of integral action and the advanced design of tire equipped passive wheels, sideslip does not occur during the experiments. The results of the experiments are consistent with the simulation and confirm the effectiveness of the proposed algorithm.

4.2.4 Closed-loop Curve Path Following

The snake robots' closed-loop path following algorithm shows that the snake robot passes through 5 waypoints and finally returns to the original point, *i.e.,* the first waypoint and the last waypoint are joined. In the closed-loop path following simulation, the snake robot dynamic parameters and control coefficients are the same as that in the first simulation, except that the initial position is selected as $(p_x(0), p_y(0)) = (0, 0)$. The tracking trajectory of the simulation results are shown in Fig. 4.21. The PCSI generated path is separated into two parts by the waypoint ep3. The x-coordinates of the first part of the path from ep1-ep3 monotonically increase. In

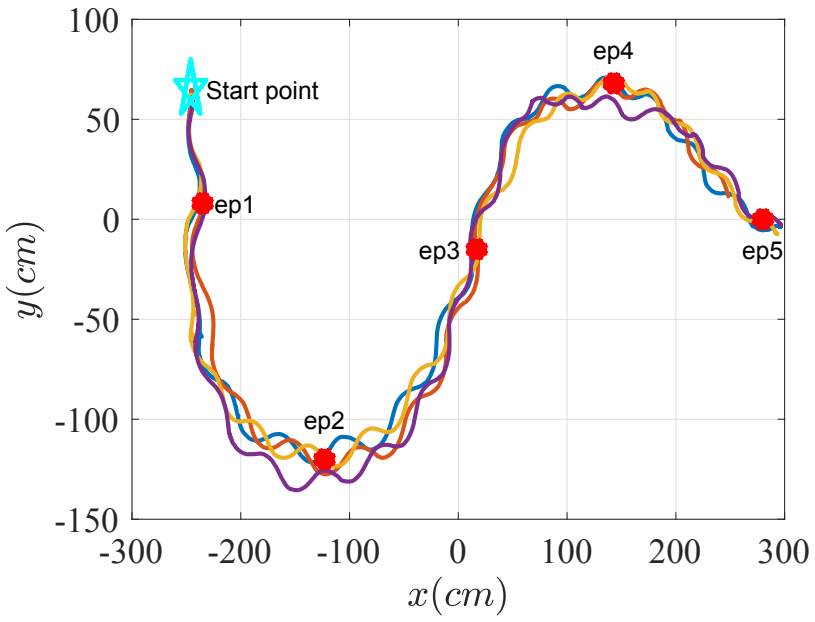


Figure 4.19: Monotonic curve path following experiment repeated four times where all trajectories pass through five waypoints.

contrast, the x-coordinates of the second part, *ep*3-*ep*5 decrease. The sign of slope also changes at the waypoint ep3, which leads to a small tracking error. The corresponding time-varying LOS, shown in Fig. 4.22, varies from $\Delta = 140$cm to $\Delta = 40$cm to achieve aggressive steering at waypoint ep3. This is critical for applications of the snake robot, especially when the snake robot is working in a challenging environment. With a variable LOS, the snake robot is able to hit the desired path sligtly faster than the constant lookahead distance. It also can be verified in Fig. 4.21, that the cross-track slightly deviates from zero and back to zero in a few seconds. This simulation demonstrates potential possibilities for the snake robot to explore an unstructured environment.

Additional experiments validate the snake robot closed-loop path following function as shown in Fig. 4.26. The snake robot follows the regulated closed-loop path, which is the yellow dashed line in the figure. The experiment was repeated three

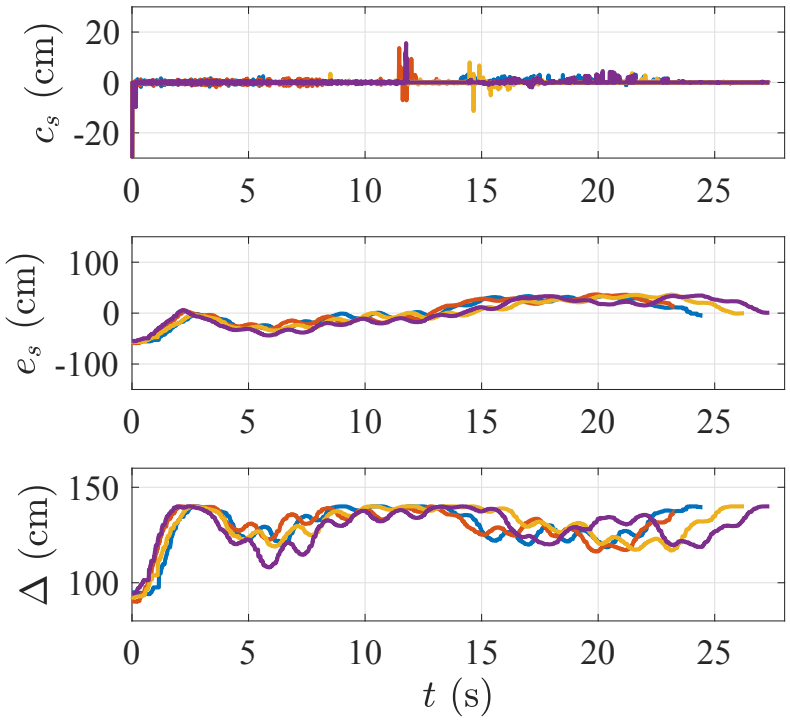


Figure 4.20: Monotonic curve path following experiments results: e_s, c_s, and Δ.

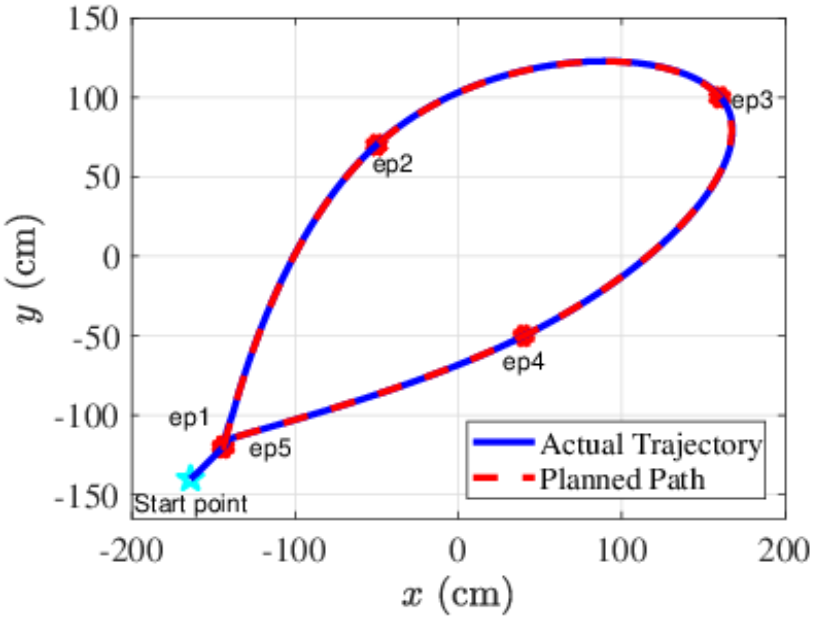


Figure 4.21: Snake robot tracking trajectory on a closed-loop curve.

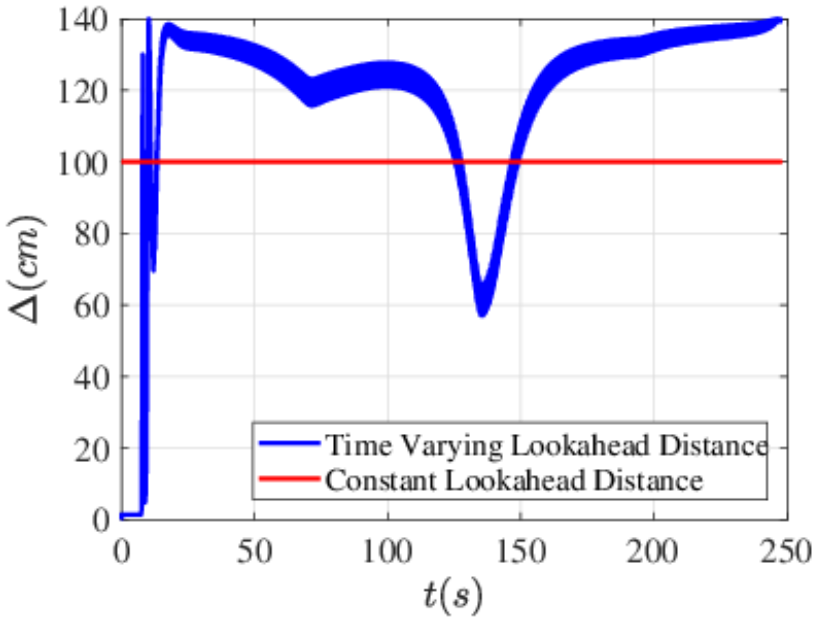


Figure 4.22: The simulated time-varying LOS on a closed-loop curve.

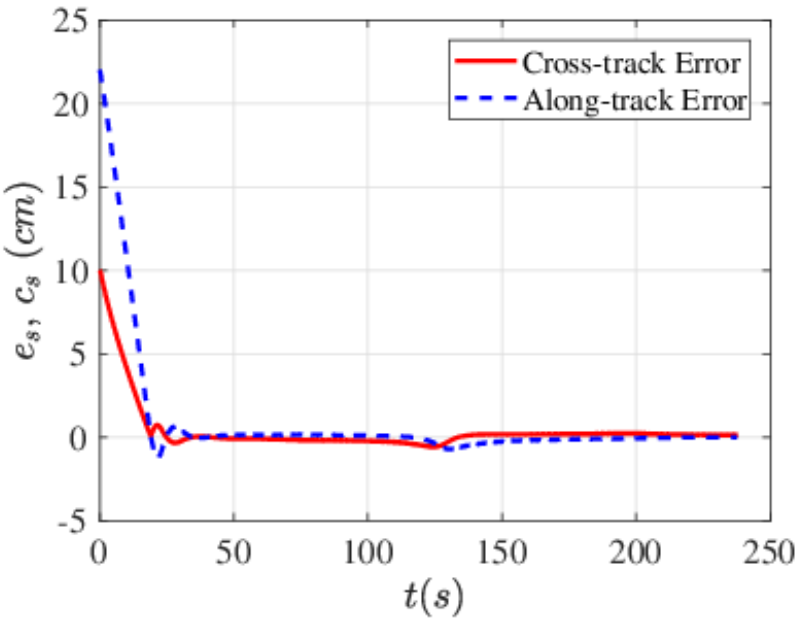


Figure 4.23: Closed-loop path following simulation results: e_s and c_s.

times to verify the effectiveness of the proposed LOS guidance law. The results are shown in Fig. 4.24. Although the blue trajectory deviates slightly from curves of the other two experiments, the peak value of the deviation is even smaller than the other two curves shown in the first two pictures of Fig. 4.25. Overall, the experimental results for closed-loop path following with time-varying LOS highly agree with the simulation results.

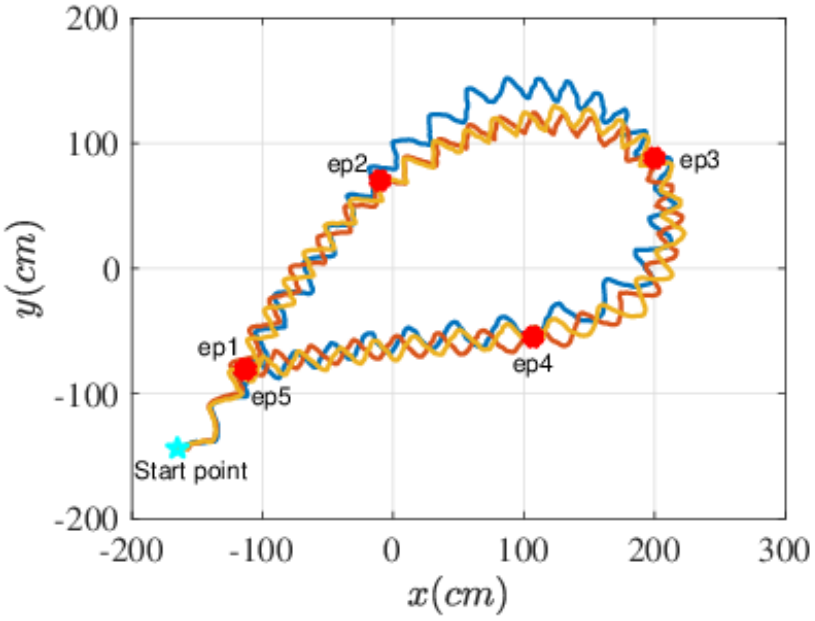


Figure 4.24: Closed-loop curve path following experiment repeated three times with the first and last waypoint joined.

4.2.5 Cross Curve Path Following

The third simulation demonstrates that the snake robot passes through all four quadrants to form a cross-line. Furthermore, a challenging start point is given as $(p_x(0), p_y(0)) = (160, 20)$, which is above the first waypoint ep1 as shown in Fig. 4.27 so that the difference between the snake robot's initial heading angle and the desired angle is over 90°. We import the snake robot dynamics and coefficients from

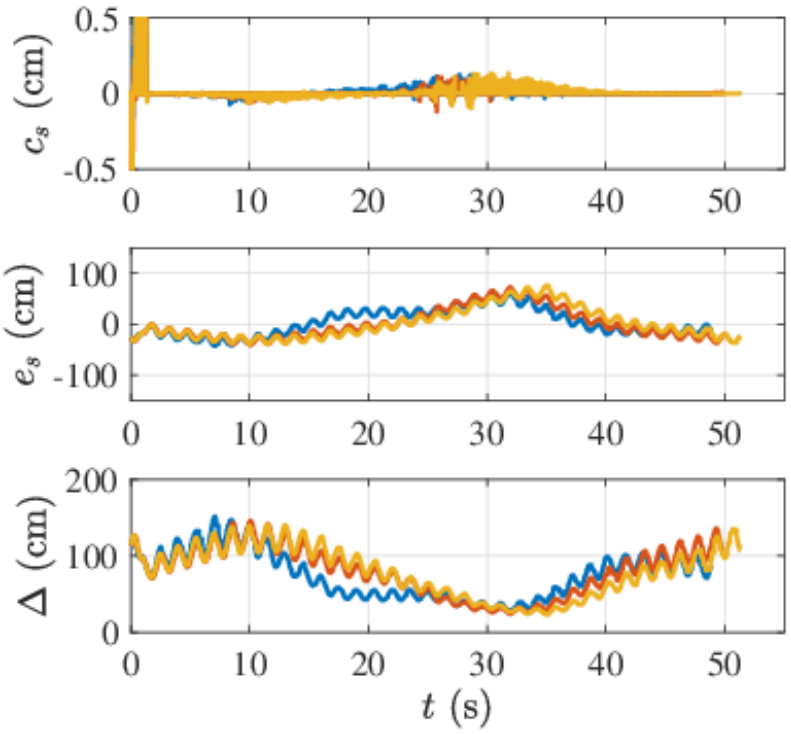


Figure 4.25: Closed-loop curve path following experimental results: e_s, c_s, and Δ.

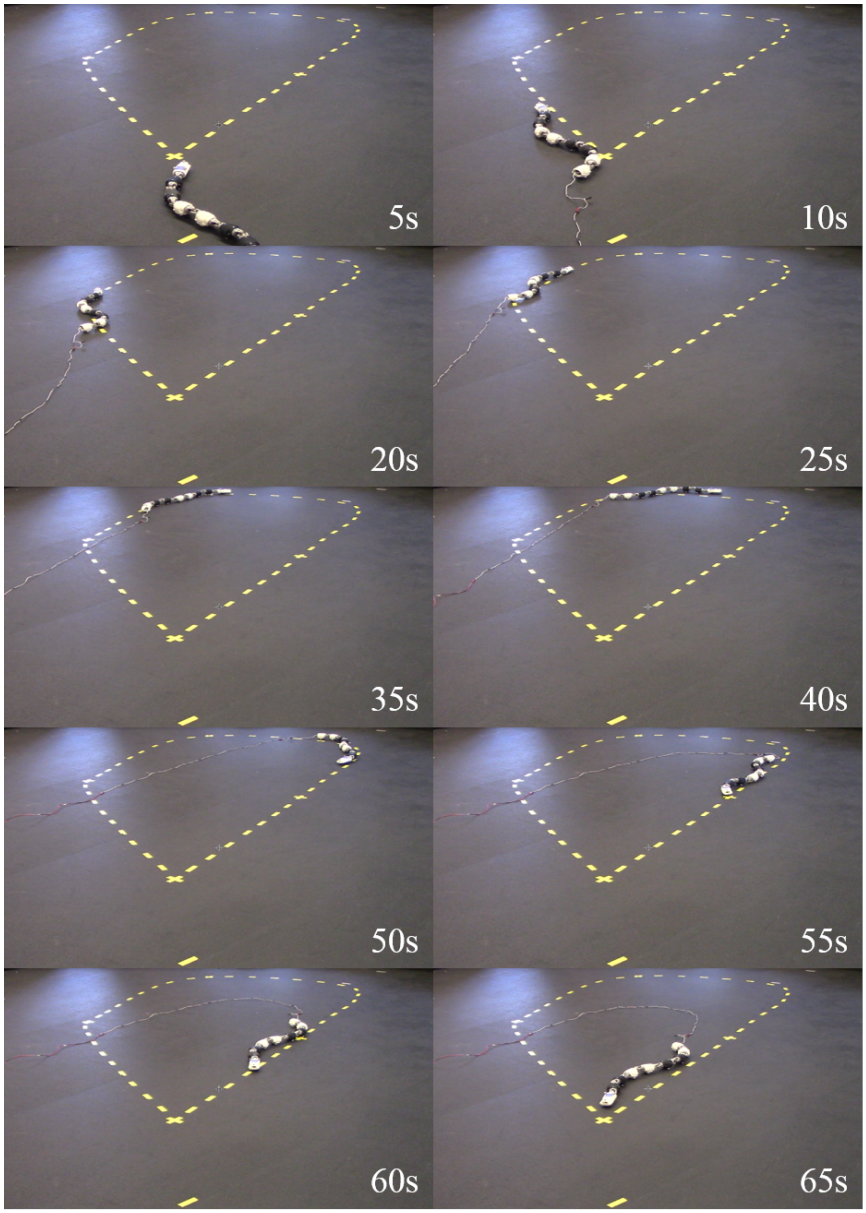


Figure 4.26: Closed-loop path following experiment of the snake robot under the proposed controller. All frames are taken from a movie recording of the experiment.

the second simulation. As expected, the snake robot's LOS vibrates sharply in the first $15s$, which varies from $\Delta = 140$cm to $\Delta = 6$cm as shown in Fig. 4.28. The corresponding along-track error and cross-tracking error shown in Fig. 4.29 are also higher than the first two simulations in the first $15s$. This is because the proposed controller is trying to regulate the snake robot heading to follow the desired angle. The overshoot occurs when the snake robot first hits the desired path. Subsequently, the snake robot follows the desired path continually with the help of varied LOS as can be verified by the zero cross-track error shown in Fig. 4.29.

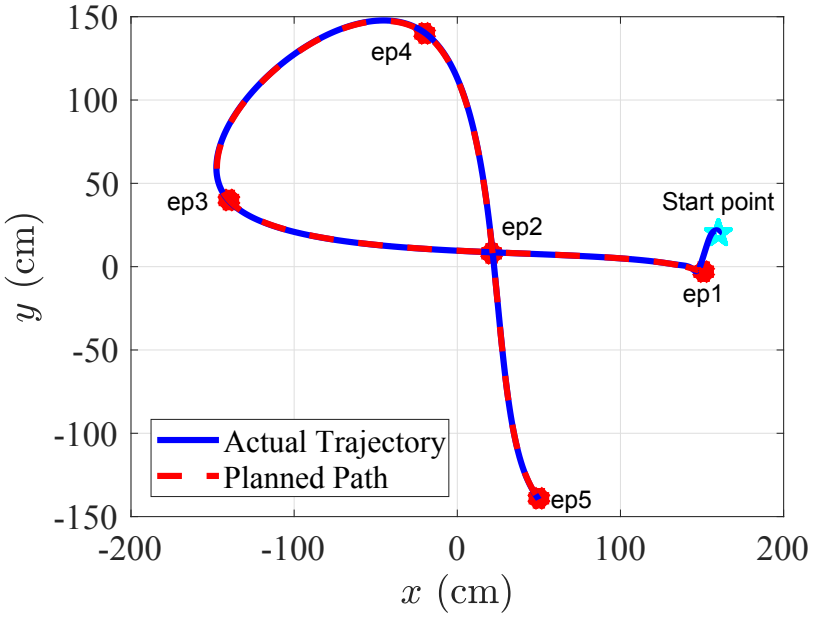


Figure 4.27: Snake robot tracking trajectory on a cross-line curve.

The experiments implement the cross-line path following function for the snake robot. We adjust the position of the waypoints based on the requirements of the motion capture coordinate system. Comparing to the cross-line path following simulation, the start point is set in the fourth quadrant instead of the first quadrant to make the snake robot pass through all 4 quadrants of the local coordinate system. The experimental results presented in Fig. 4.30 show good tracking performance for

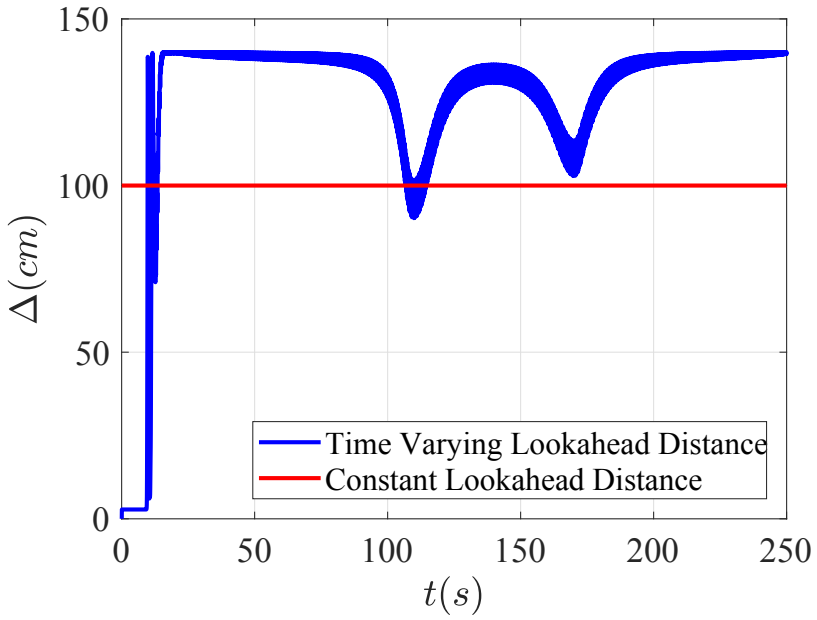


Figure 4.28: Simulated time-varying LOS on a cross-line curve.

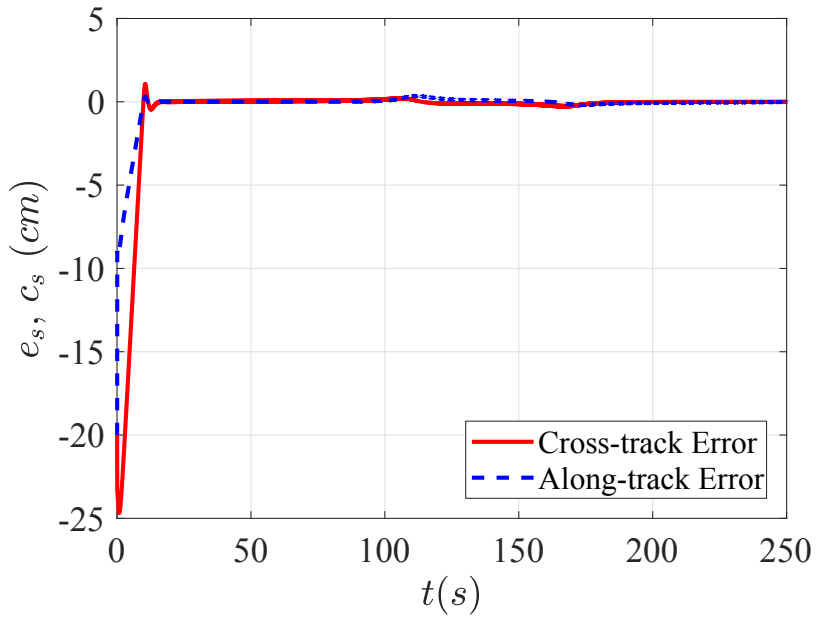


Figure 4.29: Cross-line path following simulation results: e_s and c_s.

the snake robot with the cross-line path following algorithm. The snake robot passes through all 5 regulated waypoints in a shorter time than the closed-loop path experiment. The motion capture results are pictured in Fig. 4.27. The stability of the controller is experimentally validated by the superimposed tracking curves. The cross-line path has high curvature that exerts pressure on the snake robot steering system. As expected, the cross-tracking error e_s is around zero before the third waypoint ep3 because the curvature between the waypoints ep1 and ep4 is small. The amplitude of the curvature oscillates and the sign of curvature changes after the waypoint ep3. The corresponding time-varying LOS decreases to the lowest boundary to keep the snake robot following the desired cross-line. With the regulation of the proposed controller, e_s returns to zero when the snake robot is on the way to the fifth waypoint ep5. The changes in e_s with time-varying LOS are depicted in Fig. 4.32.

It is worth noting the shape vibrations at the beginning of the three demonstrated simulations. The snake robot aggressively turns in the first 30s before converging to the desired curve. It is normal to see snake robots to steer aggressively, and the correlated Δ vibrate dramatically, especially when the snake robot first approach the desired path. The reason is that the snake robot's heading angle is not equal to the desired angle when it hits the path and he snake robot heading takes time to converge. The snake robot's trajectory will deviate a bit when it first reaches the desired path, and the Δ reacts to adjust the heading of the snake robot.

We present a new curve path following algorithm for planar underactuated snake robots in this section. The algorithm is an innovative combination of the improved LOS guidance law and the PCSI path planning method, which can drive multi-degrees-of-freedom snake robots to follow curve paths while eliminating sideslip. Both simulation and experiments conducted on an 8-link custom-built snake robot demonstrate the effectiveness of the proposed algorithm, that is, the snake robot can follow

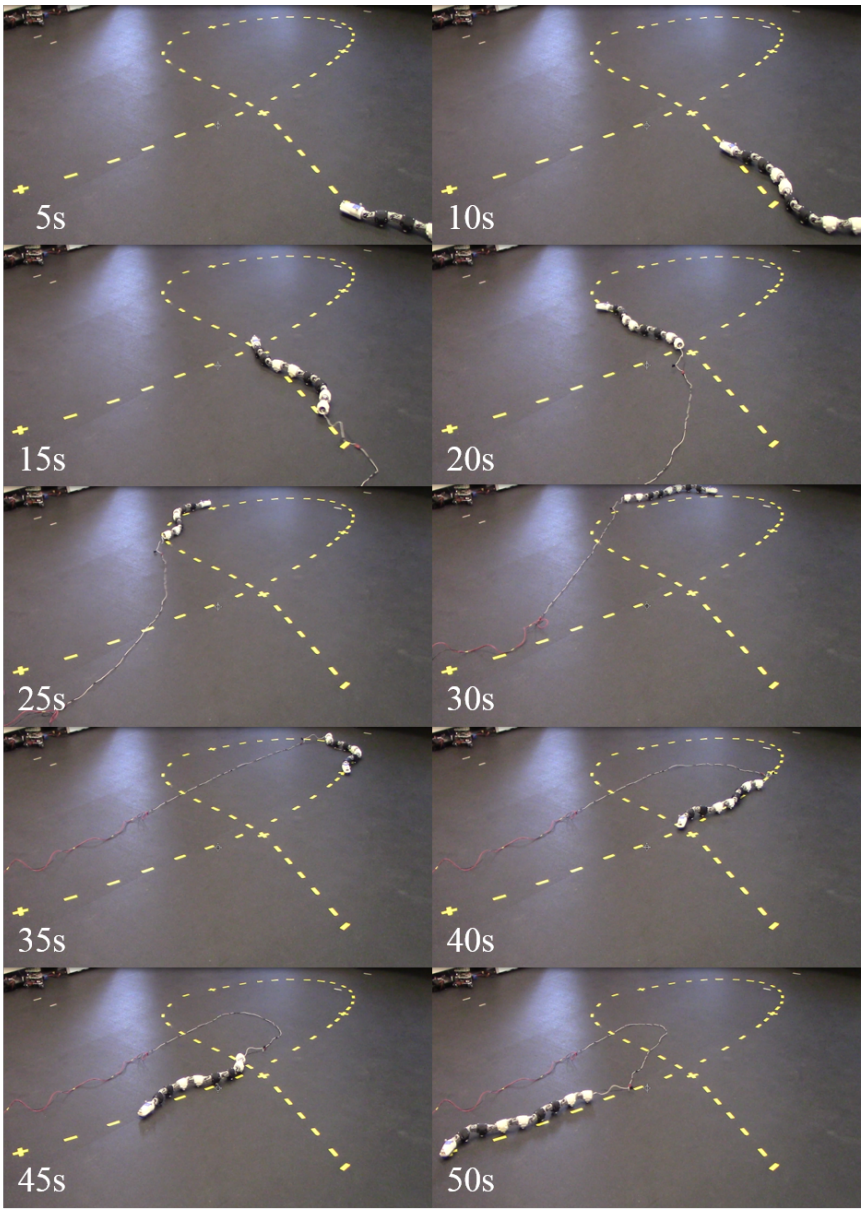


Figure 4.30: Cross-line path following experiment of the snake robot under the proposed controller. All frames are taken from a movie recording of the experiment.

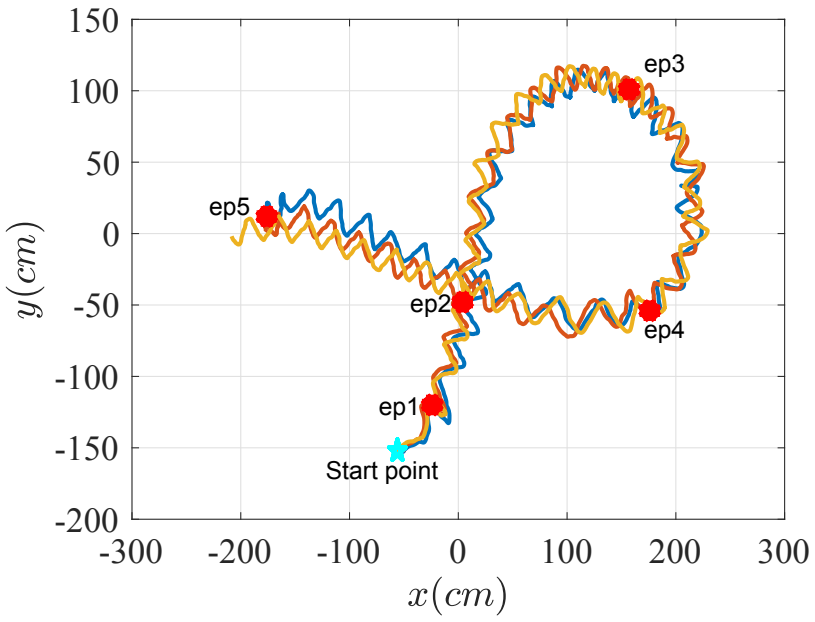


Figure 4.31: Cross-line curve path following experiment repeated three times. The snake robot passes through all waypoints.

the planned curve paths (waypoints) very well. Future work will focus on the snake robot path following in more challenging environments with complicated curve paths on rough terrains.

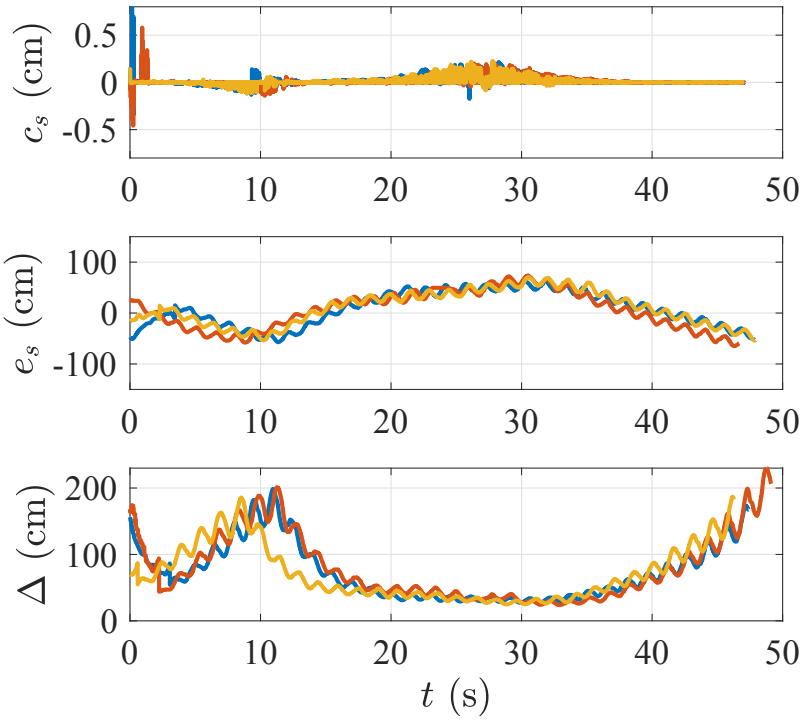


Figure 4.32: Cross-line curve path following experimental results: e_s, c_s, and Δ.

Chapter 5

Perception-Aware Pathfinding and Following of Snake Robots in Unknown Environments

In the previous two sections 4.1 and 4.2, we investigated different path following problem of a planar snake robot, namely, straight path following and 3 types of curve path following. Simulations and experiments demonstrate the efficiency of the proposed methods. However, path following is achieved with the support of an external motion tracking system, *OptiTrack* described in Section 2.2.3, which limits the applicability of the snake robot. In this section, we investigate perception-aware planning and tracking control for a class of snake robots in an unmodeled and unknown environment. First, the onboard LIDAR sensor mounted on the head of the snake robot is utilized to reconstruct a model of the local environment. This is used by the modified rapidly-exploring random tree to construct a feasible path from the current position of the robot to a local selected target position. Then, the parametric cubic spline interpolation path-planning method and potential functions are applied to make the path more smooth and to prevent the robot from hitting obstacles.

A time-varying line-of-sight control law is adopted to ensure that the robot moves to the local target position following the generated path by the perception-aware method. The robot repeatedly performs the above search strategy until it reaches the final predefined target point. Simulation and experimental results are provided to illustrate the performance of the proposed path-planning and control approach.

5.1 Problem Statement

The kinematic model of snake robot, described in (3.1) can be presented in the following form [33, 61]. [33, 61]

$$\begin{aligned} \dot{\theta} &= v_\theta, \dot{p}_x = v_t \cos\theta - v_n \sin\theta, \\ \dot{p}_y &= v_t \sin\theta + v_n \cos\theta, \dot{v}_\theta = -f_1 v_\theta + \frac{f_2}{N-1} v_t \bar{e}^T \phi \end{aligned} \tag{5.1}$$

where $(p_x, p_y) \in R^2$ is the position of the center of mass of the snake robot and N is the number of links, with $N = 8$ in our experiment. The i^{th} 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 $\phi = [\phi_1, \ldots, \phi_{N-1}] \in R^{N-1}$. 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 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 to the path planning and

control of the robot, the LIDAR scanning algorithm to reconstruct the unstructured local environment is introduced.

5.2 LIDAR Searching Algorithm

The LIDAR carried by the snake robot is a 360-degree scanning sensor that can receive data within a range of 12 meters around the robot. In order to make the robot move more efficiently toward the target point, we introduce a reward function. The function is adopted to calculate the reward value of each point explored by the LIDAR and identify the point p_h (not necessarily unique) with the highest reward value. 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 directed. The detailed LIDAR scanning algorithm is given below.

Algorithm 1 LIDAR Scanning Algorithm.

1: $LIDAR\_Scan(Position, Angle)$
2: **for** $j = 1, j < 5, i++$ **do**
3: $died\_end$(j) $\leftarrow$ $is\_quadrant\_died\_end()$
4: **end for**
5: **if** !$died\_end()$ **then**
6: **for** $i = 1, i < 360, i++$ **do**
7: $connected\_flag$(i) $\leftarrow$ $Connected(Position, Target)$
8: $angle\_weight$(i) $\leftarrow$ $Angle\_weight(Angle, Target)$
9: $distance\_weight$(i) $\leftarrow$ $Distance\_weight(Position, Target)$
10: $reward\_value$(i) $\leftarrow$ $died\_end$(i) $\times$ (a_r $\times$ $angle\_weight$(i) + b_r $\times$ $connected\_flag$(i)–c_r $\times$ $distance\_weight$(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-degrees. 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 scan can be directly connected to the final pre-defined target point, i.e., there is no obstacle between the point and the target. Line 8 is the angle information. The scanning point must 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 to obtain 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.

5.3 Executive Rapidly-Exploring Random Tree Path Generation

Traditional path planning algorithms include the artificial potential field method, the fuzzy rule method, genetic algorithm, neural network, simulated annealing, and ant colony optimization. However, these methods model obstacles in a specific space. Since the computational complexity is exponentially related to the robot's degrees of freedom, they are not suitable for solving the path planning problem of multi-link snake robots with many degrees of freedom. RRT can adequately address path planning problems in a high-dimensional space with complicated constraints. Thus, the path planning algorithm based on RRT can avoid space modeling by collision detection using sampling points in state space. RRT can find a feasible path from

the starting point to the target point and solve the path planning problem of snake robots in a complex and unconstructed environment. The basic RRT [16] is an efficient planning method in multidimensional space and uses an initial point as the root node to generate a random extension tree by randomly sampling the leaf nodes (see Algorithm 2).

Algorithm 2 Basic RRT 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: $X_{rand} \leftarrow Random\_State(obstacles)$
4: $X_{near} \leftarrow Nearst\_Neighbor(X_{rand}, T)$
5: **if** *collision* **then**
6: Back to 3
7: **end if**
8: $X_{new} \leftarrow X_{rand}$
9: $T_{add\_vertex}(X_{new})$
10: $X_{add\_edge}(X_{near}, X_{new})$
11: **if** $arrivetarget(T, X_{target})$ **then**
12: $finalpath \leftarrow export(T)$
13: **end if**
14: **end for**

We modify the RRT by introducing a waypoint cache for snake robot path planning. In the processing of the RRT, all the states are updated in the cache whenever the path is determined. Due to the modification of the basic RRT, the new algorithm can search and save a feasible path from the initial point to the target point for snake robots. The waypoint cache modification of RRT is shown in Algorithm 2.

Furthermore, we take the path cost into account to select each new node more close to the target waypoint. By including the path cost in the calculation, the algorithm can select the lower cost paths to the target. The generated path is more close to the feasible path rather than basic RRT. RRT shapes the probability distribution to select a potential node based on its Voronoi region and the path progress toward

Algorithm 3 Waypoint Cache Extension.

```
p = randomin[0.0 ··· 1.0]
i = randomin[0 ··· NumOfWayPoint − 1]
if 0 < p < P_target then
    X_rand ← target
    if P_target < p < P_target + P_WayPoint then
        X_rand ← WayPointCache[i]
        if P_target + P_WayPoint < p < 1 then
            X_rand ← RandomNode()
        end if
    end if
end if
X_new ← X_rand
```

the node. We introduce an additional measure Υ to estimate the path cost from the potential nodes to the target node, which is computed as

$$\Upsilon = 1 - \frac{\lambda_{vertex} - \lambda_{opt}}{\lambda_{max} - \lambda_{opt}}. \tag{5.2}$$

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} is the maximum path cost through all possible nodes. Each parameter is calculated via the distance. The value of the measurement Υ is a direct measure of the condition of the path, in comparison to the optimal path. The algorithm structure is described in Algorithm 3. All extensions of previous RRT work improve replanning efficiency and

Algorithm 4 Path Cost Extension.

```
X_new ← X_rand
T_add_vertex(X_new)
X_add_edge(X_near, X_new)
Calculate_Path_Cost(X_new)
if arrivetarget(T, X_target) then
    finalpath ← export(T)
end if
```

Table 5.1: Performance Comparsion of RRT and ERRT

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

the quality of generated path. The novel execution extended RRT is called as ERRT. ERRT is successfully applied to the snake robot to demonstrate its ability to search for a feasible path in unconstructed domains more efficiently and dynamically than basic RRT. Table 5.1 shows that the computational efficiency of ERRT is around three times higher than that of basic RRT. The comparison of basic RRT and our ERRT is shown in Fig. 5.2.

We employ an artificial potential function to prevent the spanning tree from reaching obstacles. The function takes the following form [31]

$$U = \frac{1}{2}\mu\left(\frac{1}{\rho(p_c, p_{obs})} - \frac{1}{\rho_0}\right) \tag{5.3}$$

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 closest point on the obstacle to the robot, and ρ_0 is the width of the robot.

For traditional RRT, the exploration step size l_s for each exploration is fixed, which leads to inefficient RRT exploration. To solve this issue, we redesign l_s such that it dynamically changes according to the exploration complexity. More specifically, l_s changes according to the relative distance between the current position of the robot and the target point. That is, when the robot is far away from the target point, l_s becomes longer to enable it to explore the surrounding environment more quickly, and when the robot is close to the target point, l_s becomes shorter so that

the target can be accurately found. l_s is proposed as

$$l_s = l_{\min} + \frac{k(l_{\max} - l_{\min})}{1 + e^{-sp_e}} U \tag{5.4}$$

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

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} r\cos\beta \\ r\sin\beta \end{bmatrix} \tag{5.5}$$

with $r = \delta_r |n_r| + r_0$ and $\beta = \delta_\theta |n_\theta| + \theta_d$, where n_r and n_θ are Gaussian random numbers, δ_r and δ_θ are the standard deviations in the radial direction and the standard deviation in the tangential direction, respectively. The forward direction is obtained through the LIDAR scanning algorithm. We then use random Gaussian sampling to obtain sampling points for the ERRT so that the path planning algorithm can reach the target point quickly. The Gaussian distribution sampling strategy is demonstrated in Fig. 5.1.

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

5.4 Path Smooth and Control Objective

The snake robot can search for a path from the initial point to the target point through the proposed ERRT. However, the path is non-smooth and cannot be fol-

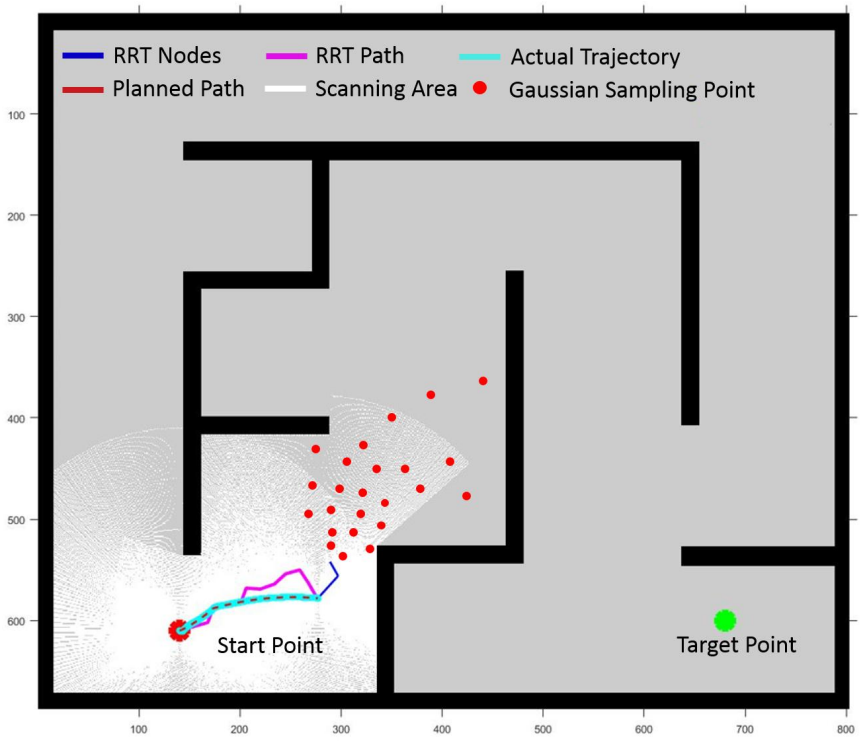
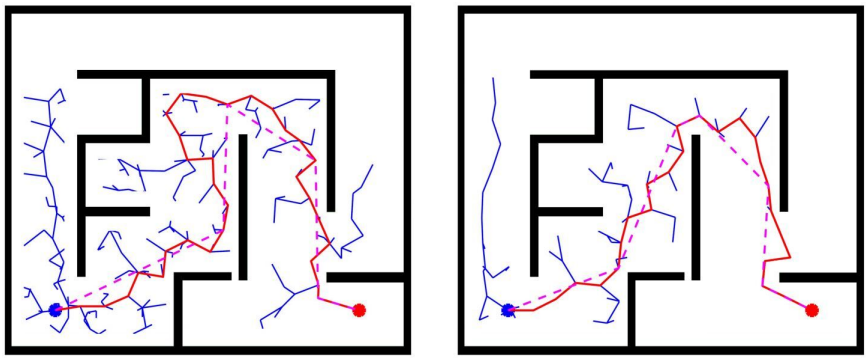


Figure 5.1: Gaussian distribution sampling strategy. The red dots represent sample points.



(a) (b)

Figure 5.2: Tree expansions comparison for RRT and ERRT illustrating path optimality. Path from the blue initial point to the red target point is shown in red. (a) Basic RRT solution, (b) ERRT solution.

lowed by a snake robot. PCSI, described in Section 4.2.1, is employed to smooth the path consisting of the waypoints picked from the waypoints cache. Based on the ERRT-PCSI planned path, a time-varying LOS guidance law Section 4.2.2 is introduced to steer the snake robot.

We specify the control objective to drive the snake robot to follow the ERRT-PCSI designed path using the time-varying LOS steering law. More specifically, we 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's actual position (p_x, p_y). Multiplying e_s with the transition matrix in the global coordinate, we have the expression for e_s of (4.49)

In the sequel, we design the joint offset ϕ_o such that the heading angle θ converges to the LOS guidance law of (4.52). To start, we define the heading angle error variable as

$$\psi = \zeta - \bar{\zeta} \tag{5.6}$$

where $\zeta = \theta + \ell\dot{\theta}$ and $\bar{\zeta} = \bar{\theta} + \ell\dot{\bar{\theta}}$ with $\ell > 0$. The time derivative of ψ is

$$\begin{aligned} \dot{\psi} &= v_\theta + \ell(-f_1 v_\theta + \tfrac{f_2}{N-1} v_t \bar{e}^T \phi) - \dot{\bar{\zeta}} \\ &= \ell(-f_1 v_\theta + f_2 v_t \phi_o) + \xi \end{aligned} \tag{5.7}$$

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 as $V = \psi^2/2$. Taking the derivative of V along (5.7) gives

$$\dot{V} = \ell(-f_1 v_\theta + f_2 v_t \phi_o)\psi + \xi\psi. \tag{5.8}$$

We choose the joint offset ϕ_o as

$$\phi_o = \frac{f_1}{f_2 v_t} v_\theta + \frac{1}{f_2 \ell v_t}(-\eta\psi - \xi) \tag{5.9}$$

where η is a positive control gain. Using (5.9), the time derivative of V becomes

$$\dot{V} = -\eta\psi^2 \leq 0. \tag{5.10}$$

The design of the joint coordinates ϕ_i given by (4.51) is complete.

Theorem 3 *For the snake kinematic model (5.1), the path following controller defined by (5.9) meets the control objective (4.50).*

Proof. Applying the LaSalle-Yoshizawa theorem to (5.10) shows that $\lim_{t\to\infty} \psi(t) = 0$. By additionally noting (5.6), we have $\lim_{t\to\infty} \theta(t) - \bar{\theta}(t) = 0$. The rest of the proof is the same as that of [61] but is omitted here.

5.5 Perception-Aware Pathfinding and Following

5.5.1 Simulation Study

The performance of the LIDAR scanning algorithm, the ERRT path-planning, and PCSI path smoothing algorithm are examined through MATLAB simulation. The controller proposed in Section 5.4 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. In the simulation, we choose the strategy to approach the target quickly. 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 ERRT to generate a path from the starting point to the

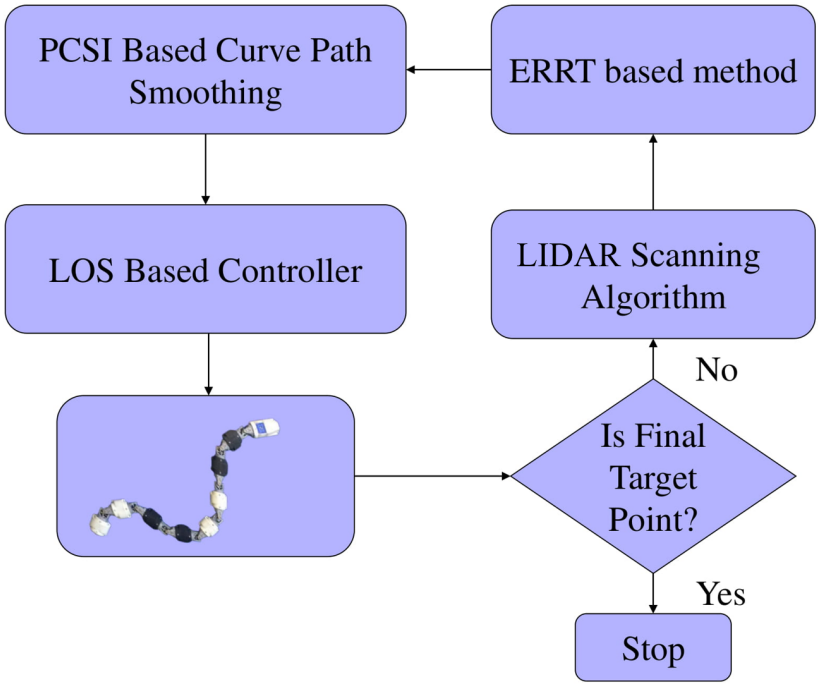


Figure 5.3: Flow chart of the proposed perception-aware path following scheme.

highest rewards quadrant with Gaussian sampling. The Gaussian sampling ensures that ERRT expands 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 by changing these parameters according to the robot location and the approaching strategy. Varying the step size improves the ERRT searching speed, and the 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 $\rho_0 = 2.5$. 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

velocity $v_t(0) = 20$cm/s and normal velocity $v_n(0) = 12$cm/s. The LOS guidance method is initialized with convergence rate $K_\Delta = 2$ and constant proportional gain $K_p = 0.3$. The Newton-Raphson method is used to find the roots of the cubic polynomials (4.29) and (4.30). With the Newton-Raphon method, path following can converge within a few iterations. The simulation results are shown in Figure 5.4.

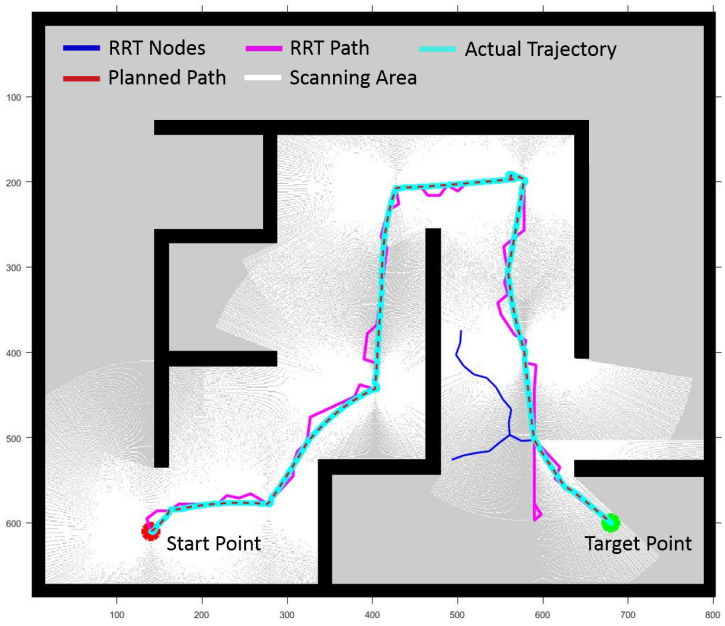


Figure 5.4: LIDAR based path following simulation
The perception-aware path planning and following simulation results. The red dot is the starting point and the green dot is the target point. The grey area is unknown, and the white space is LIDAR scanned. RRT node results are depicted by blue line, and the RRT path is shown with the pink line; the red line is PCSI smoothed; the snake robot's actual trajectory is the broad cyan line.

As manifested by the theoretical analysis, the LIDAR scanning algorithm directs the search area toward the target. The ERRT generates a path for the snake robot, plotted with a pink line. A feasible path for snake locomotion, shown as a red dashed line, is formed by the PCSI method. The cyan line shows the trajectory of the snake robot.

5.5.2 Experimental Validation

The proposed search based algorithm is implemented with experiments in a maze built by paperboard. The labyrinth is the same as that utilized in the simulation with the dimension of 300×400cm. A LIDAR system called RpLIDAR s1 is mounted on the head of the snake robot to obtain the local map information with 10Hz scanning frequency, such that 9200 points are collected in one second. When the LIDAR is scanning with 10Hz rate, the angular resolution is 0.391∘. As a result of the high update rate from the LIDAR, sector mapping can be utilized to create a cost map and to estimate 2D position. Our ERRT path planner provides feasible path information to the move_base node for navigation. The generated paths are slightly different due to the random propriety of ERRT. The proposed controller 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 (Ubuntu 16.04). 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, $\omega = 25°/$s, $\delta = 20°$, $\theta = 30°$, $v_\theta = 0°/$s, $v_\phi = 0°/$s. To avoid slippage, the snake robot travels slowly at a speed of $v_t = 5$cm/s, and $v_n = 0$cm/s during the straight-line motion and at a speed of $5°/s$ during turning motion. Other parameters are the same as those used in the simulation.

The results of the path searching experiments are presented in Figure 5.6. The entire locomotion lasts 75 seconds due to low movement speed. We repeat the operation four times to test the algorithm's reliability. The snake robot successfully reaches the target four times, even though the four times trajectories are different,

as shown in Fig. 5.6. The results show that the randomness of the ERRT does not affect its reliable path planning for the snake robot. In this section, we presented

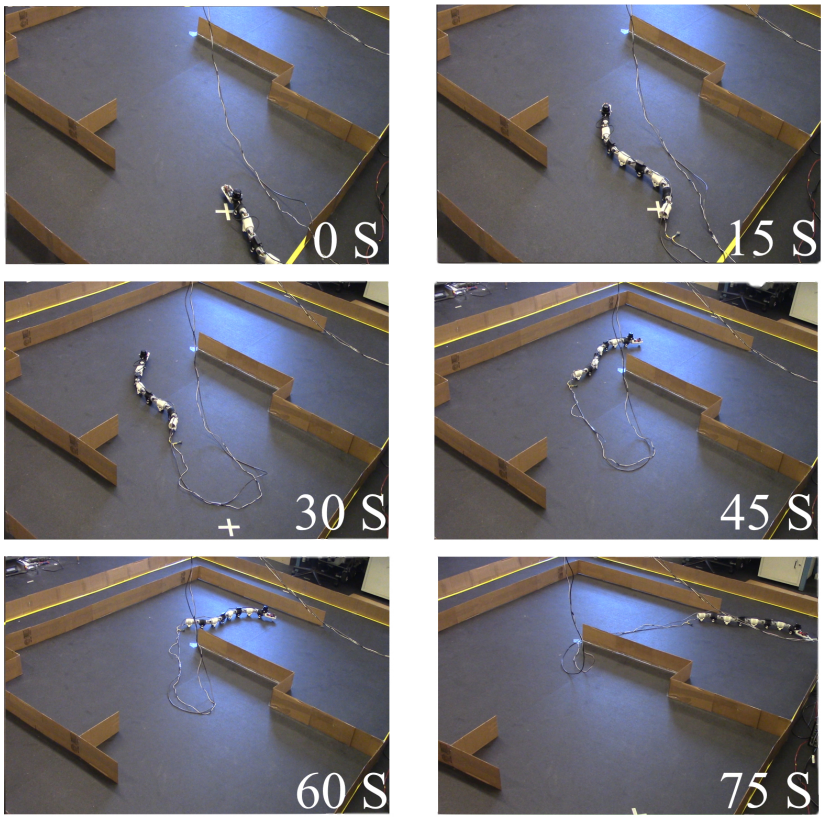


Figure 5.5: Experiment snapshots

a perception-aware path planning and tracking framework to make the snake robot approach a pre-defined target point without requiring exact knowledge of the environment. The proposed framework consists of the following four main components: (1) LIDAR scanning algorithm, which reconstruct the local environment; (2) ERRT, which find a feasible path from current point to a local selected target position based on locally searched information; (3) PCSI based curve path smoothing solution, which provide a smooth desired path for the snake robot; and (4) LOS based controller, which steers the robot toward and subsequently along the generated path. As the path is planned in real time based on local scanned environmental information, the

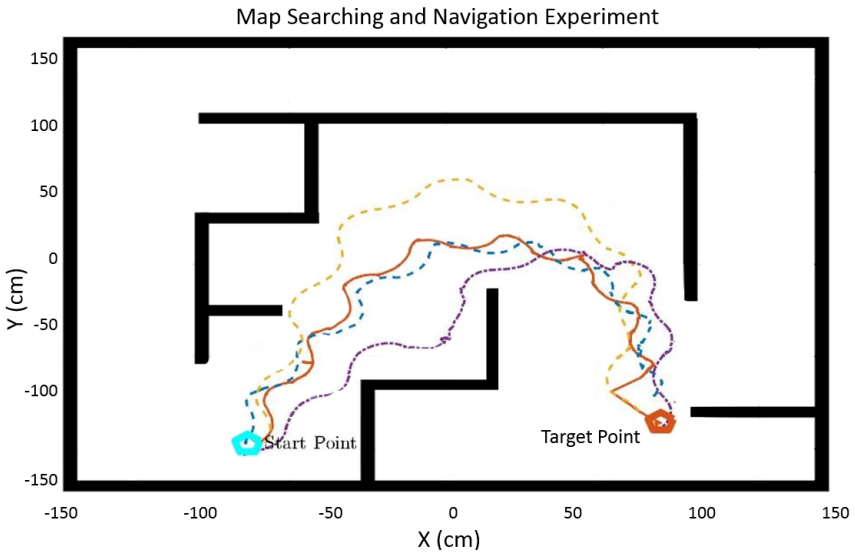


Figure 5.6: ERRT path search experiment repeated four times

proposed approach has the potential for incorporating other desirable features such as avoiding moving objects. Experimental and simulation results are included to illustrate the performance of the proposed framework.

Chapter 6

Conclusions and Future Work

In this dissertation, we explored the path following solutions of snake robots in a variety of situations, including the straight-line path following with unknown and various friction coefficients; path following under arbitrarily generated curve in a known environment; and a perception-aware path planning and tracking in an unknown environment. To deal with the problems in the path following, we firstly proposed an innovative adaptive path-following controller to compensate for unknown friction coefficients and ensure the asymptotic convergence of path tracking. An improved LOS guidance law and the PCSI path-planning method were further investigated in order to drive the snake robot to follow an arbitrarily planned curve path and eliminate the sideslip. Last, we used a LIDAR searching algorithm to real-time reconstruct the local environment and generate a live path with ERRT algorithm, so that the snake robot can reach the target by finding and following the live path. Extensive simulation and experimental results are demonstrated to prove the performance of the proposed methods and solutions.

Our **future work** can be pointed out as follow. First of all, we realized that the simplified model cannot completely reflect the system of the snake robot we used. In the simulation, we did not add the gait of the snake robot to the kinematics

modeling, this results in that the snake robot can quickly make the tracking error converge to zero, that means, the snake robot followed the desired path without gait patterns in the simulation. While in the experiment, the snake robot follows the path with the generated gait and friction estimation. This makes the differences between simulation and experimental results. In order to eliminate the differences, a more sophisticated model needs to be defined and applied to both simulations and experiments for validation. In addition, the serpentine gait of the snake robot needs to be included in the simulation model. In future work, we will improve the model accuracy by analyzing experimental outcomes, and will update and further verify the model in both simulations and experiments. In addition to updating the snake robot mathematical model, we will also optimize the snake robot's mechanical structure and electronic control module. Being an effort, we will make each joint of the snake robot shorter, which conforms to the skeletal structure of natural snakes so as to increase the motion smoothness and certain flexibility of the snake robot. Each joint can be equipped with a more powerful servo motor and precision gear system to reduce motion errors caused by the servo system.

We will continue to explore the autonomous navigation function in complex environments and expand the snake robot's locomotions from two-dimension space to three-dimension space, making it more adaptative to environments and real-world tasks. For this purpose, we will facilitate more motion modes of snake robots, such as sidewinding, rectilinear locomotion modes, combined with machine learning methods, to achieve autonomous navigation in three-dimension and complex environments.

Overwater and underwater snake robots will be an extension of current research. There are many applications of overwater and underwater snake robots, including hydrological information detection and underwater ocean bridge quality detection. In this extension, both mechanical and electronic systems of the snake robot need to

be improved for a waterproofed consideration. To deal with the complex overwater or underwater environment, the robot gait pattern and path following controller need to be re-designed to fit the water-relevant environments based on hydrodynamics and swimming motion dynamics of the snake robot.

Snake robots certainly offer great potential in many autonomous applications. We believe that our research efforts will advance the progress of snake robots and their applications.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] Joseph Ayers Joel L Davis Alan et al. *Neurotechnology for biomimetic robots*. MIT press, 2002.

[3] Brian Armstrong-Helouvry, Pierre Dupont, and Carlos Canudas De Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1083–1138, 1994.

[4] R.H. Bartels, J.C. Beatty, and B.A. Barsky. *An Introduction to Splines for Use in Computer Graphics and Geometric Godeling*. Morgan Kaufmann Series in Computer Graph Series. M. Kaufmann Publishers, 1987.

[5] Johann Borenstein and Malik Hansen. Omnitread ot-4 serpentine robot: new features and experiments. In *Unmanned Systems Technology IX*, volume 6561, page 656113. International Society for Optics and Photonics, 2007.

[6] E. Borhaug, A. Pavlov, and K. Y. Pettersen. Integral LOS control for path following of underactuated marine surface vessels in the presence of constant

ocean currents. In *2008 47th IEEE Conference on Decision and Control*, pages 4984–4991, Dec 2008.

[7] Even Børhaug, Alexey Pavlov, Elena Panteley, and Kristin Y Pettersen. Straight line path following for formations of underactuated marine surface vessels. 19(3):493–506, 2011.

[8] Zhengcai Cao, Qing Xiao, Ran Huang, and Mengchu Zhou. Robust neuro-optimal control of underactuated snake robots with experience replay. *IEEE transactions on neural networks and learning systems*, 29(1):208–217, 2017.

[9] Alessandro Crespi and Auke Jan Ijspeert. Amphibot II: An amphibious snake robot that crawls and swims using a central pattern generator. In *Proceedings of the 9th international conference on climbing and walking robots*, number BIOROB-CONF-2006-001, pages 19–27, 2006.

[10] Alessandro Crespi and Auke Jan Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. 24(1):75–87, 2008.

[11] KD Do and Jie Pan. Global tracking control of underactuated ships with off-diagonal terms. In *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, volume 2, pages 1250–1255. IEEE, 2003.

[12] Thor I Fossen. *Marine control systems: guidance, navigation and control of ships, rigs and underwater vehicles*. Trondheim: Marine Cybernetics, 2002.

[13] Thor I. Fossen, Morten Breivik, and Roger Skjetne. Line-of-sight path following of underactuated marine craft. *IFAC Proceedings Volumes*, 36(21):211 – 216, 2003.

[14] E Fredriksen and Kristin Ytterstad Pettersen. Global κ-exponential way-point maneuvering of ships: Theory and experiments. *Automatica*, 42(4):677–687, 2006.

[15] J. Gray. The mechanism of locomotion in snakes. *Journal of Experimental Biology*, 23(2):101–120, 1946.

[16] Guo Haitao, Zhu Qingbao, and Xu Shoujiang. Rapid-exploring random tree algorithm for path planning of robot based on grid method. *Journal of Nanjing Normal University (Engineering and Technology Edition)*, 2(14), 2007.

[17] S. Hirose. *Biologically inspired robots: Snake-like locomotors and manipulators*. Oxford: Oxford University Press, 1993.

[18] Shigeo Hirose and Makoto Mori. Biologically inspired snake-like robots. In *2004 IEEE International Conference on Robotics and Biomimetics*, pages 1–7. IEEE, 2004.

[19] Shigeo Hirose and Hiroya Yamada. Snake-like robots [tutorial]. *IEEE Robotics & Automation Magazine*, 16(1):88–98, 2009.

[20] David L. Hu, Jasmine Nirody, Terri Scott, and Michael J. Shelley. The mechanics of slithering locomotion. *Proceedings of the National Academy of Sciences*, 106(25):10081–10085, 2009.

[21] Masato Ishikaway, Katsuya Owaki, Masahide Shinagawa, and Toshiharu Sugie. Control of snake-like robot based on nonlinear controllability analysis. In *2010 IEEE International Conference on Control Applications*, pages 1134–1139. IEEE, 2010.

[22] K.G. Jolly, R. Sreerama Kumar, and R. Vijayakumar. A Bézier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57(1):23 – 33, 2009.

[23] Tetsushi Kamegawa, T Yarnasaki, Hiroki Igarashi, and Fumitoshi Matsuno. Development of the snake-like rescue robot" kohga". In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 5081–5086. IEEE, 2004.

[24] Soren Kammel and Benjamin Pitzer. Lidar-based lane marker detection and mapping. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1137–1142. IEEE, 2008.

[25] Jackie Kay. Proposal for implementation of real-time systems in ros 2, 2016.

[26] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[27] Anna M Kohl, Kristin Ytterstad Pettersen, Eleni Kelasidi, and Jan Tommy Gravdahl. Planar path following of underwater snake robots in the presence of ocean currents. *IEEE Robotics and Automation Letters*, 1(1):383–390, 2016.

[28] Miroslav Krstic, Ioannis Kanellakopoulos, Petar V Kokotovic, et al. *Nonlinear and adaptive control design*. New York: Wiley, 1995.

[29] Miroslav Krstic, Ioannis Kanellakopoulos, and Peter V Kokotovic. *Nonlinear and adaptive control design*. New York: Wiley, 1995.

[30] Jacoby Larson and Mohan Trivedi. Lidar based off-road negative obstacle detection and analysis. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 192–197. IEEE, 2011.

[31] Jean-Claude Latombe. *Robot motion planning*. Norwell, MA: Kluwer, 1991.

[32] F L Lewis, Suresh Jagannathan, and A Yesildirak. *Neural network control of robot manipulators and non-linear systems*. London: CRC Press, 1998.

[33] Pål Liljeback, Idar U Haugstuen, and Kristin Y Pettersen. Path following control of planar snake robots using a cascaded approach. *IEEE Transactions on Control Systems Technology*, 20(1):111–126, 2011.

[34] Pål Liljebäck, Kristin Y Pettersen, and Øyvind Stavdahl. A snake robot with a contact force measurement system for obstacle-aided locomotion. In *2010 IEEE International Conference on Robotics and Automation*, pages 683–690. IEEE, 2010.

[35] Pål Liljebäck, Kristin Y Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. Controllability analysis of planar snake robots influenced by viscous ground friction. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3615–3622. IEEE, 2009.

[36] Pål Liljeback, Kristin Y Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. Controllability and stability analysis of planar snake robot locomotion. *IEEE Transactions on Automatic Control*, 56(6):1365–1380, 2010.

[37] Pål Liljebäck, Kristin Y Pettersen, Øyvind Stavdahl, and Jan Tommy Gravdahl. A simplified model of planar snake robot locomotion. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2868–2875. IEEE, 2010.

[38] Shugen Ma. Analysis of snake movement forms for realization of snake-like robots. In *Proc. IEEE Int. Conf. Robot. & Automat.*, volume 4, pages 3007–3013, 1999.

[39] Shugen Ma. Analysis of creeping locomotion of a snake-like robot. *Advanced Robotics*, 15(2):205–224, 2001.

[40] F. Matsuno and H. Sato. Trajectory tracking control of snake robots based on dynamic model. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005.

[41] L. McCue. Handbook of marine craft hydrodynamics and motion control. *IEEE Control Systems Magazine*, 36(1):78–79, 2016.

[42] James C McKenna, David J Anhalt, Frederick M Bronson, H Ben Brown, Michael Schwerin, Elie Shammas, and Howie Choset. Toroidal skin drive for snake robot locomotion. In *2008 IEEE International Conference on Robotics and Automation*, pages 1150–1155. IEEE, 2008.

[43] Alireza Mohammadi, Ehsan Rezapour, Manfredi Maggiore, and Kristin Y Pettersen. Maneuvering control of planar snake robots using virtual holonomic constraints. *IEEE Transactions on Control Systems Technology*, 24(3):884–899, 2016.

[44] M. Mori and S. Hirose. Development of active cord mechanism acm-r3 with agile 3d mobility. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 3, pages 1552–1557 vol.3, Oct 2001.

[45] Joyjit Mukherjee, Sudipto Mukherjee, and Indra Narayan Kar. Sliding mode control of planar snake robot with uncertainty using virtual holonomic constraints. *IEEE Robotics and Automation Letters*, 2(2):1077–1084, 2017.

[46] Joshua Noble. *Programming interactivity: a designer's guide to Processing, Arduino, and OpenFrameworks*. " O'Reilly Media, Inc.", 2009.

[47] Y. Ohmameuda and Shugen Ma. Control of a 3-dimensional snake-like robot for analysis of sinus-lifting motion. In *Proceedings of the 41st SICE Annual Conference.*, volume 3, pages 1487–1491 vol.3, 2002.

[48] Jim Ostrowski and Joel Burdick. Gait kinematics for a serpentine robot. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1294–1299. IEEE, 1996.

[49] Kristin Y. Pettersen. Snake robots. *Annual Reviews in Control*, 24:19–44, 2017.

[50] Kristin Y Pettersen, Ãyvind Stavdahl, and Jan Tommy Gravdahl. *Snake robots: modelling, mechatronics, and control*. Springer London, 2012.

[51] Casey Reas and Ben Fry. *Processing: a programming handbook for visual designers and artists*. Mit Press, 2007.

[52] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[53] Ehsan Rezapour, Andreas Hofmann, and Kristin Y Pettersen. Maneuvering control of planar snake robots based on a simplified model. In *Proc. IEEE Int. Conf. Robot. Biomimetics*, pages 548–555. IEEE, 2014.

[54] Ehsan Rezapour, Kristin Y Pettersen, Pål Liljebäck, Jan T Gravdahl, and Eleni Kelasidi. Path following control of planar snake robots using virtual holonomic constraints: theory and experiments. *Robotics and biomimetics*, 1(1):3, 2014.

[55] Masashi Saito, Masakazu Fukaya, and Tetsuya Iwasaki. Modeling, analysis, and synthesis of serpentine locomotion with a multilink robotic snake. *IEEE control systems magazine*, 22(1):64–81, 2002.

[56] Madhavan Shanmugavel, Antonios Tsourdos, Brian White, and Rafał Żbikowski. Co-operative path planning of multiple UAVs using Dubins paths with clothoid arcs. *Control Engineering Practice*, 18(9):1084 – 1092, 2010.

[57] Shugen. Analysis of creeping locomotion of a snake-like robot. *Advanced Robotics*, 15(2):205–224, 2001.

[58] Shize Su and Zongli Lin. Distributed consensus control of multi-agent systems with higher order agent dynamics and dynamically changing directed interaction topologies. 61(2):515–519, 2016.

[59] Lianfang Tian and Curtis Collins. An effective robot trajectory planning method using a genetic algorithm. *Mechatronics*, 14(5):455–470, 2004.

[60] Aksel Andreas Transeth, Kristin Ytterstad Pettersen, and P Liljeback. A survey on snake robot modeling and locomotion. *Robotica*, 27(7):999–1015, 2009.

[61] Gang Wang, Weixin Yang, Yantao Shen, Haiyan Shao, and Chaoli Wang. Adaptive path following of underactuated snake robot on unknown and varied frictions ground: theory and validations. *IEEE Robotics and Automation Letters*, 3(4):4273–4280, 2018.

[62] Wikiversity. Nature snake skeleton. https://www.popsugar.com/family.

[63] Cornell Wright, Aaron Johnson, Aaron Peck, Zachary McCord, Allison Naaktgeboren, Philip Gianfortoni, Manuel Gonzalez-Rivero, Ross Hatton, and Howie Choset. Design of a modular snake robot. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2609–2614. IEEE, 2007.

[64] Xiaodong Wu and Shugen Ma. CPG-based control of serpentine locomotion of a snake-like robot. *Mechatronics*, 20(2):326 – 334, 2010.

[65] Hiroya Yamada. S. development of amphibious snake-like robot acm-r5. In *the 36th International Symposium on Robotics (ISR 2005), Tokyo*, 2005.

[66] Weixin Yang. Biomorphic hyper-redundant snake robot: Locomotion simulation, 3D printed prototype and inertial-measurement-unit-based motion tracking. Master Thesis, *University of Nevada, Reno*, 2016.