Doctoral Dissertation

A Bayes Estimation Based Control Method for Snake Robots

March 2022

Doctoral Program in Advanced Mechanical

Engineering and Robotics

Graduate School of Science and Engineering

Ritsumeikan University

JIA Yuanyuan

Doctoral Dissertation Reviewed
by Ritsumeikan University

A Bayes Estimation Based Control Method for Snake Robots
(ヘビ型ロボットのためのベイズ推定に基づく制御方法)

March 2022

2022 年 3 月

Doctoral Program in Advanced Mechanical Engineering and Robotics

Graduate School of Science and Engineering

Ritsumeikan University

立命館大学大学院理工学研究科

機械システム専攻博士課程後期課程

JIA Yuanyuan

ジア　ユアンユアン

Supervisor：Professor MA Shugen

研究指導教員：馬　書根教授

Abstract

A Bayes Estimation Based Control Method for Snake Robots

by

Yuanyuan JIA

Doctor of Philosophy in Department of Robotics

Ritsumeikan University, Biwako-Kusatsu Campus

Professor Shugen MA, Chair

Snake robot control in a cluttered environment has received tremendous attention because of its wide practical applications such as survival rescue, pipeline inspection, medical surgery, firefighting, and surveillance. Because of the complexity associated with a highly redundant structure, many degrees of freedom, occlusions, and environmental clutters, robust and efficient control of snake robots remains a challenging task. To date, conventional deterministic snake robot control does not perform well when obstacles are in close proximity or present collisions. In such circumstances, modeling the interactions among robot modules and the interactions between the robot and the environment, establishing a correspondence between modules and observations, and decreasing the computational complexity to achieve real-time implementation are critical problems.

In this dissertation, we investigate issues towards solving these problems. Specifically, the thesis comprises five fundamental contributions. The first is a centralized Bayesian control method, which extends the shape-based compliant control and achieves the robust performance of snake robot control in a complicated environment. Secondly, a dynamically decoupled Bayesian formulation is proposed for snake robot control with respect to robot modules and interacted obstacles. The characteristics of this formulation are that it avoids the common practice of using complex fully coupled interaction analysis and performing difficult joint data association. It extends the Bayesian control framework by modeling interactions in terms of stimulus functions. The third contribution is a fully distributed framework using multiple collaborative agents for multiple robot modules with significant and persistent interactions with clutter surroundings. In this framework, we propose

modeling the interactions among modules and the environment by an inter-module likelihood and a module-based virtual external force density. Fourthly, we have proposed a centralized coach-based Bayesian method for snake robot control. Instead of applying density propagation analysis directly for controller design, we use it to expedite the training process of a reinforcement learning process. Experimental results show that it is a very promising direction in combining the advantages of two research areas for snake robot control. Finally, we apply the distributed Bayesian control as a coach to the training process of reinforcement learning for the snake robot. Compared with the centralized framework, it further expedites the training speed due to exploiting parallel computing.

The five Bayesian controllers have been demonstrated on both simulation and real-world data to verify the proposed methods. As a result, they can achieve robust real-time locomotion for snake robots in an unstructured environment. While all these systems can function effectively, some are more stable and efficient than others in various situations. To reveal their advantages and disadvantages, thorough investigations in terms of characteristics of architecture, complexity analysis, and performance comparison have been studied. Among these methods, the centralized framework is more suitable for a small system with a limited number of robot links, while decoupled and distributed structures are better for more complicated snake robots with more body links. Meanwhile, the two coach-based approaches stand on the shoulders of other frameworks and provide an innovative way to bridge the reinforcement learning with stochastic Bayesian control together.

Contents

List of Figures

List of Tables

Acknowledgements

First of all, I would like to express my deep appreciation to my advisor, Professor Shugen Ma, for his invaluable guidance and support during the whole course of my doctoral studies. He taught me not only how to do research work, but how to think in a right way as well. Discussions with Professor Ma are always fruitful. His in-depth knowledge in robotics, wise advice and inspiring attitude helped me shape and present this work as it is today.

I wish to thank my dissertation committee, Professor Miyano and Professor Tokuda for their unwavering support and assistance on my dissertation and helping me complete this important milestone of my academic life.

I am lucky to work in a wonderful lab. I truly benefit from the awesome team work of our Lab. Thanks to Prof. Tian, Prof. Li, Dr. Du, Dr. Qiu, Mr. Oka, Mr. Sun, and Ms. Cao for giving me excellent suggestions and helpful comments. Thanks to Prof. Zhang for his help when I prepared response to journal reviews. Thanks to Prof. Kakogawa for sharing codes when I started my research from scratch. Thanks to Mr. Kawabata for technical suggestions when I set up hardware constructions.

A profound gratitude goes to my husband, who has always been there and encouraged me whenever I got stuck in research or felt down in stress. His constructive attitude of always solution finding, helped me get back on track, and cut many detours.

Finally I would like to thank my beloved Mom, Dad who helped me to complete this tough task under pressure. They live far away in Beijing but are so close in other ways.

Chapter 1

Introduction

1.1 Motivation and Research Background

Over the recent years, people made great success in developing numerous types of autonomous mobile robots, to overcome the difficulty of conventional robots in rugged environment. Although in human survival training, it is common to require versatile ability, which includes swimming underwater, traversing rocky mountain and desert, tunnels and even more, there still exists a gap for a single robot to accomplish all the above mobility. The chance to bridge this gap probably lies in snake-like robots. By imitating biological snakes, snake robots are likely to integrating versatile mobility within a uniform body mechanism.

Snake robotic control [5] is a system that contributes to the locomotion of robots. This commonly involves mechanical aspects and algorithms that make it possible to control robots. Inspired by biological snake movement, snake-like robots usually consist of serially connected joint links capable of bending in one or multiple planes. Though these biomorphic hyper-redundant robots can vary greatly in size and design, all snake robots share the following properties: 1) the small cross section to length ratio allows them to move into, and maneuver through narrow and tight spaces; 2) the ability to change the shape of their body allows them to perform a wide range of behaviours, such as stepping stairs or climbing trees; 3) the chaining redundancy makes them resistant to failure since they can continue to operate even if parts of their bodies are destroyed. These unique charac-

teristics make snake robot control a principal research area within robotics community over the last decade. The increasing interest is attributed to the numerous applications including survival rescue, medical surgery, fire fighting, sanitation, agriculture, surveillance, and maintenance of complex and possibly dangerous structures or systems such as nuclear plants or pipelines, intelligent services, media, exploration, research, education, etc.

Most existing snake robot control systems rely on a particular mechanical design and deterministic analysis of kinematics of robots, which are not suitable for practical applications where uncertainties present not only inside a snake robot itself but also in unstructured environments. Stochastic analysis of snake robot control problem remains a very challenging task for robotics community because of the difficulties embedded in this tough problem, some of which are as follows:

(1) How to model the high redundant structure with a large number of DOFs and solve the critical uncertainty problem in a stochastic way?

(2) How to handle the interactions between the surrounding clutters, the changing environment, and the robot during navigation?

(3) How to efficiently control multiple actuators simultaneously while still keeping the proper gait in real-time?

(4) How to learn the robot's kinematics automatically and quickly without prior assumptions?

(5) How to achieve an optimal balance between exploration and exploitation in the locomotion?

The purpose of this thesis is trying to answer these questions and to establish novel efficient frameworks for snake robot control in unknown environment. The approach is rooted in ideas from statistics, graphical model theory, probability density estimation, machine learning and control engineering. The problem is to control multiple links of a snake robot simultaneously, as they collaborate together to move through clutter scenes, which contains targets and obstacles.

This chapter presents a brief review of background materials relevant to this thesis. Shape-based gait control, probability graphical modeling, Bayesian network, probability density estimation and reinforcement learning theory as well, are covered.

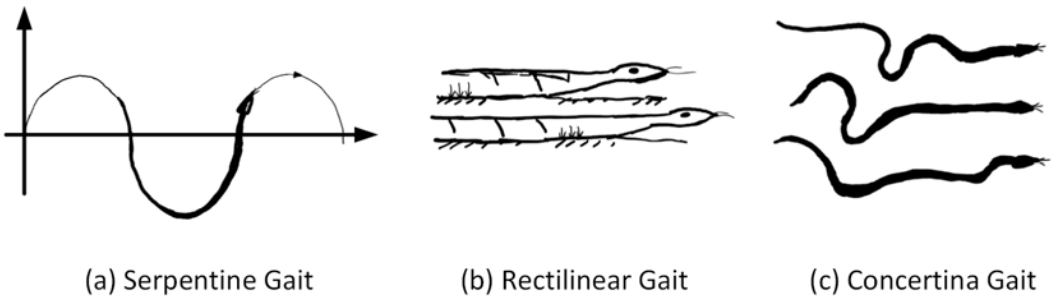

Figure 1.1. The typical gaits of snake robot locomotion.

1.1.1 Shape-Based Gait Control

Research of snake robot has received a lot of attentions and been conducted for several decades. Early locomotion studies of snakes were reported by Gray [6] in 1940s. Different gaits such as serpentine locomotion, rectilinear locomotion [7], concertina locomotion [8] and sidewind locomotion were studied as shown in Fig. 1.1 by investigation of biological snakes. Serpentine locomotion is an effective transverse wave where snakes exploit it in a flat environment most commonly. Rectilinear locomotion is generated when alternating locomotion of ribs and muscles, where biological snakes use it when tracking preys. Concertina locomotion consists of gripping or anchoring with portions of the body while pulling or pushing other sections in the direction of movement. Sidewinding locomotion is 3D spiral locomotion, which is achieved by a combination of serpentine locomotion and rectilinear locomotion. Various snake-like robot kinematic models have been studied based on different gaits. Hirose developed the first snake robot in 1972 [5]. He used sinusoidal functions and tuned amplitude, frequency and phase differences to derive the robot to a certain direction. The trace of body of natural snakes is called a "serpenoid curve" which guarantees that the distribution of a strain in the muscle becomes a smooth function. Ma proposed the serpentine curve in [9], where a mathematical model of the muscle characteristics of snakes is employed to derive the resulted form of the body shape during lateral undulation. He proved that the serpentine is more efficient than serpenoid curve in simulation results. Moreover, snake-like robot can create a variety of gaits by extracting waveform from spine curves [10]. Furthermore, other gaits such as spiral gait when climbing trees was investigated by researchers. In some situations, a biological snake may also have a jump, swing around the body, sliding shock, etc. When snakes climb up along a rod or move by

lateral gait, their patterns are 3D spiral curves. Generally speaking, shape-based gait control is an intuitive and computationally simple way to balance the tradeoff between coordinating the motion of internal degrees of freedom and reacting to unknown features, connecting high-level planning to low-level control. The advantage of this kind of approaches is that the control can be executed in a lower dimensional space through shape functions. More detailed reviews of shape-based gait control of snake robots can be found in [11] and [12].

1.1.2 Probability Graphical Modeling

Probability Graphical Model (PGM) theory has emerged from a mix of log-linear and covariance selection theories, sharing the concepts of path analysis, independence and conditional independence. Graphical models are amazing probability models for the analysis of multivariate random observations, where the independence structure is characterized by a conditional independence graph [1]. In this section, we briefly introduce the concepts of graphical model theory used in this thesis. For details, we also refer readers to [1].

A graph G is a mathematical object that consists of two sets, a set of vertices, V, and a set of edges, E, consisting of pairs of elements taken from V. There is a *directed edge* or *arrow* between vertices i and j in V if the set E contains the ordered pair (i, j); vertex i is a *parent* of vertex j, and vertex j is a *child* of vertex i. There is an *undirected edge* or *line* between these vertices if E contains both pairs, (i, j) and (j, i). The graph is *undirected* if all edges are undirected. Vertices i and j are *adjacent* if the undirected edge between i and j is in E, and a line connects them in the diagram of the graph. Let $a \subseteq K$ denote a subset of vertices of the graph. The *neighbors* of a are all those vertices in V, but not in a, that are adjacent to a vertex in a. The *conditional independent graph* of X is the undirected graph $G = (K, E)$ where $K = (1, 2, ..., k)$ and (i, j) is *not* in the edge set E if only if $X_i \perp\!\!\!\perp X_j | X_K \setminus \{i, j\}$, where "$\perp\!\!\!\perp$" denotes the independence of two random variables; "|" means "given" or "conditional"; and "\" is "not including".

Separation Theorem. *If X_a, X_b and X_c are vectors containing disjoint subsets of variables from X, and if, in the independence graph of X, each vertex in b is separated from each vertex in c by the*

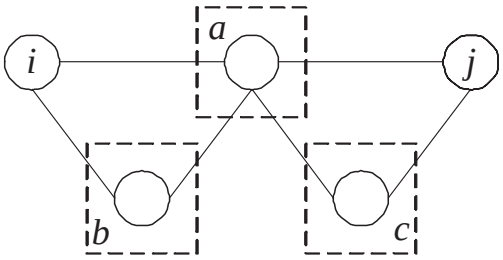

Figure 1.2. Conditional independent graph.

subset a, then

$$X_b \perp\!\!\!\perp X_c | X_a, \tag{1.1}$$

which means given X_a, X_b is conditionally independent with X_c.

The converse of the separation theorem is immediate: if it is true that whenever a separates i and j that $i \perp\!\!\!\perp j|a$ then it is true that, whenever i and j are not adjacent, $i \perp\!\!\!\perp j$ |rest. Fig. 1.2 illustrates the separation theorem, where we can easily have $i \perp\!\!\!\perp j|V \setminus \{i, j\}$ and $X_b \perp\!\!\!\perp X_c|X_a$.

The most important tool for interpreting independence graphs is the *Markov property*, that, for all disjoint subsets a, b and c, of V, whenever b and c are separated by a in the graph, then X_b and X_c are independent given X_a alone: $X_b \perp\!\!\!\perp X_c|X_a$.

In many studies of several interacting variables there is a striking lack of symmetry in the roles played by the variables that corresponds to a notion of causality and the premiss that if X causes Y then Y cannot cause X. A neat way to portray the relationship that 'X effects Y' is by means of a directed graph and its diagram of Fig. 1.3 together with the conditional probability density $p(Y|X)$.

Figure 1.3. Directed graph

The *directed independence graph* of X is the directed graph $G^{\rightarrow} = (V, E^{\rightarrow})$, where $V = \{1, 2, ..., v\}$, $V(j) = \{1, 2, ..., j\}$ and the edge (i, j), with $i \rightarrow j$, is *not* in the edge set $E^{\rightarrow}$ if and only if $j \perp\!\!\!\perp i \,|V(j) \setminus \{i, j\}$. This is the same definition used for the undirected independence graph

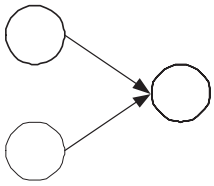

Figure 1.4. Forbidden Wermuth configuration

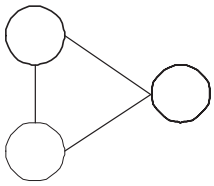

Figure 1.5. Moral graph

with the conditioning set modified, from the 'rest' comprising all past and future variable, to just the past.

We wish to elucidate the Markov properties of a directed independent graph but hope to avoid much further work in proving such properties by exploiting the relationship of the directed graph to independence statements elicited from its associated undirected graph. A directed graph satisfies the *Wermuth condition* if no subgraph has the configuration of Fig. 1.4. The *moral graph* associated with the directed graph $G^{\rightarrow} = (V, E^{\rightarrow})$ is the undirected graph $G^m = (V, E^m)$ on the same vertex set and with an edge set obtained by including all edges in $E^{\rightarrow}$ together with all edges necessary to eliminate the forbidden Wermuth configurations from $G^{\rightarrow}$. It is termed a moral graph because it 'marries parents'. For example, Fig. 1.5 shows the moralized graph of Fig. 1.4. It has been proven that *the directed independence graph* $G^{\rightarrow}$ *possess the Markov properties of its associated moral graph,* G^M [1, p.76].

1.1.3 Bayesian Network

Many reported controllers of snake robots [13] [14] [15] are deterministic systems. However, in practical applications, sensors are usually disturbed by different noises. The observation is commonly stochastic with uncertainties. Therefore, the state can only be observed partially. In other

words, a control problem with uncertainties should be regarded as a stochastic process with noisy observations [16]. Deterministic mathematical modeling of dynamic systems is usually problematic. A statistical approach is needed to evaluate unknown parameters [17].

Kalman-type filtering has been widely used for solving stochastic problems [18]. It has been used for state estimation of snake robots by Rollinson et al. [19]. Classic Kalman-type filters assume a Gaussian distribution, which makes the solution too ideal and not suitable for practical application due to uncertainties and noises. Some statistical methods have also been reported for dynamic robot systems, such as [20], and [21] [22].

Bayesian Networks (BN) have received a lot of interests in the research domain of machine learning, pattern recognition and artificial intelligence. There also exists an analogy between Bayesian network and interactions between genes in biology [23]. The cell consists of many interacting components. These components affect each other in some consistent fashion. If random sampling of the system is considered, some states become more likely. In this context, the state of a cell denotes the concentration of proteins and metabolites in various compartments and so on. And the likelihood of a state can be specified by the joint probability distribution on each of the cells components. Furthermore, reconstruction of Bayesian network models from physiologically relevant primary cells might be applied to understanding native-state tissue signaling biology, complex drug actions, and dysfunctional signaling in diseased cells [24].

Different with most classical control methods, the BN models can be learned from sample data if the model does not have an analytical form. During the training process, missing information can be handled. Therefore, BN can achieve superior performance comparing with other self adaptive systems in different control tasks. BN is a directed acyclic graph model structured by some nodes and directed edges. The nodes represent random variables and the directed edges represent the relationship between the nodes. Every node has its own probability distribution. A BN represents the joint probability distribution of n random variables X,

$$p(X_1, X_2, \ldots, X_n) = p(X_1) \prod_{i=2}^{n} p(X_i | X_1, \ldots, X_{i-1}) \tag{1.2}$$

Since not all random variables in $X_1, \ldots, X_{i-1}$ has an influence on X_i, eq. 1.2 can be written as

$$p(X_1, X_2, \ldots, X_n) = p(X_1) \prod_{i=2}^{n} p(X_i | Pa(X_i)) \tag{1.3}$$

where $Pa(X_i)$ is the random vector called the parents of X_i, which includes the nodes with an influence on X_i [25]. The main task of BN is to do probabilistic inference using inference algorithm. One of the most important operations on BNs is the calculation of marginal distribution. Given a full distribution $p(X)$ with $X = X_1, \ldots, X_n$, an arbitrary distribution $p(X|C)$ with $C \subset X$ can be calculated by integration over all variables in C,

$$p(X|C) = \int_C p(X) dC. \tag{1.4}$$

For many control systems, the variables change with time. Dynamic Bayesian Networks (DBN) can be used to model and control such kind of linear dynamic systems [25]. The well-known Kalman filter and Hidden Markov Model which are widely used in prediction and tracking problems of control systems are special DBNs [25]. Modeling with DBNs is equivalent to learning a probability distribution which represents the data as well as possible. A more detailed description of the algorithms used for BNs is given in [25].

1.1.4 Density Estimation Techniques

Many approaches [26] can been applied for the estimation of the probability densities in Bayesian network such as Expectation Maximization (EM) [27], Variational Inference (VI) [28], Monte Carlo (MC) [29], etc. In this thesis, we exploit Sequential Monte Carlo (SMC) method and Bayesian Neural Network (BNN) [30] for density estimation.

Sequential Monte Carlo Method

The conventional SMC method [31] is to estimate the desired density $p(\cdot)$ by a set of random samples with associated weights. For example, a density $p(x_t|z_{1:t})$ can be estimated,

$$p(x_t|z_{1:t}) = \sum_{n=1}^{Ns} w_t^n \delta(x_t - x_t^n) \tag{1.5}$$

where $\{x_t^n, n = 1, \ldots, N_s\}$ is a set of samples with associated weights $\{w_t^n, n = 1, \ldots, N_s\}$, $\sum_n w_t^n = 1$. However, samples from the posterior are not directly available. The filter can use an *importance sampling* technique [32] to generate samples where a proposal *importance density* $q(\cdot)$ is first used to generate samples. Each sample is assigned weight by

$$w_t^n \propto \frac{p(\cdot)}{q(\cdot)} \tag{1.6}$$

In eq. (1.6), "$\propto$" means "is proportional to". Then the difference between the desired density and the proposal importance density can be made up by

$$E[x_t w_t(x_t)] = \int_{x_t} x_t \frac{p(\cdot)}{q(\cdot)} q(\cdot) dx_t = E[x_t] \tag{1.7}$$

That is, the estimate $E[x_t]$ can be approximated instead by $E[x_t w_t(x_t)] \approx \sum_{n=1}^{N} x_t^n w_t(x_n^t)$.

For the Bayesian control problem, by substituting the posterior density into (1.6), the weight is calculated as

$$w_t^n \propto w_{t-1}^n \frac{p(z_t|x_t^n)p(x_t^n|x_{t-1}^n)}{q(x_t^n|x_{t-1}^n, z_t)} \tag{1.8}$$

where $q(x_t^n|x_{t-1}^n, z_t)$ is a proposal importance density from which the samples x_t^n are easily generated. The most common choice of importance density in the conventional sample filter is the prior

$$q(x_t|x_{t-1}^n, z_t) = p(x_t|x_{t-1}^n) \tag{1.9}$$

Substitution of eq. (1.9) into eq. (1.8) yields

$$w_t^n \propto w_{t-1}^n p(z_t|x_t^n). \tag{1.10}$$

Bayesian Neural Network

Bayesian neural network has received much attention recently due to its ability to model uncertainty comparing to conventional deep learning approaches such as CNN, RNN etc. However, due to the high computation complexity, little research has been done in physical robot control area [33]. The difficulty lies in not only the network design but also training data collection. Since it's usually very time consuming and cost expensive in real word applications. This explains the reason why little work has been reported using BNN for snake control although it's very promising in terms

of capturing the uncertainty during locomotion and interaction with environment. Moreover, the inherent difficulty of high dimensionality of snake robots makes the problem very challenging.

One of the most distinguishing characteristics about Bayesian is that parameters are probability distributions but not fixed weights. The Bayesian neural network decomposes complicated uncertainty into model misspecification, model uncertainty, and inherent sensing noise. Dropout is a well-established procedure to regularize a neural network and limit over-fitting [30]. The dropout as a Bayesian approximation proposes a simple approach to qualify the neural network uncertainty. It employs dropout during both training and testing. It develops a new theoretical framework casting dropout in deep neural network (NNs) as approximate Bayesian inference in deep Gaussian processes. It mathematically shows that these multiple passes are equivalent to Monte Carlo sampling. Thus, the first and second moment (mean and variance) provides the network's output and uncertainty respectively. The dropout rate is a hyper-parameter that needs to be tuned. A small dropout rate eliminates the Monte Carlo sampling utility. A big dropout rate can lead to divergence or at least require more iterations to converge. The key idea is making dropout do the same thing in both training and testing process: at test time, repeating a few hundreds of times conducting the same input into the network with random dropout; after that, taking means of prediction. There are uncertainties not only in the model itself, but also weight too. Dropout provides a new and handy way to estimate the uncertainty with minimal changes in most existing networks.

1.1.5 Deep Reinforcement Learning

Robotics is still dominated by complex processing stacks. It has potential to gain a revolution similar as what happened in the area of computer vision recently in which area powerful gradient-driven end-to-end optimisation can clear a path directly from pixels to torques [33]. Deep Reinforcement learning (DRL) becomes a hot research area because it is capable of learning end-to-end robotic control tasks. However, the current accomplishments have been mostly in simulation, rather than on actual robots [34] [33]. Little research has been done applying DRL for snake robot control. Sartoretti et al. [35] proposed to leverage Asynchronous Advantage Actor-Critic (A3C) algorithm, one low-level agent per body portion, in order to learn decentralized control policies of a snake

robot. A3C is a popular learning algorithm. It can allow agents to train a stochastic policy rather than a deterministic one, where the former is more robust to noise and uncertainty. Snake robot portions can respond independently against local forcing detected by torque sensors. Windows are anchored between zero curvature points of the serpenoid curve. Consequently, shape parameters in each window can be updated by a trained agent. The A3C meta-agent represents the whole robot, while the low-level worker agents act as the window, each selecting local adaptation with the environment. The state space is defined by 7-tuple $\eta_i(t) = \langle \tau(t), \beta_i(t)^T, \mu_{ext}(j,t)^T, \beta_0^T \rangle$, where τ is a normalized phase of the serpenoid's wave, β_j the current shape parameters of window j, μ_{ext} the external torque readings, β_0 the nominal value. An action is defined as a 2-tuple $a = \langle a_A, a_\omega \rangle$ with $a_A \in \{-\triangle_A, 0, +\triangle_A\}$, and $a_\omega \in \{-\triangle_\omega, 0, +\triangle_\omega\}$. At the end of each episode, each worker updates the global network and then collects the new state of the global weights. An entropy-based loss function is used to update the policy. An off-policy is exploited to train the agents by replaying experiences.

Several problems remain open for the reported method: 1) Although different agents share a common meta-agent during training, reinforcement learning methods such as A3C algorithm essentially only exploit multiple agents to speed up the learning. In other words, there is no explicit modeling of interactions among these agents. If each window has only one joint, then the modules of the snake robot will become completely independent and thus lose the serpenoid gait; 2) Because of the limitations of experience replay, the snake robot can only change its gait based on the training samples. Namely, it may fail in an unseen situation. 3) The initialization state is very sensitive and must be learned. It suffers from error propagation problem since all portions are independent. In case the serpenoid wave is broken, the robot can not recover the gait.

Although being a very promising research direction, reinforcement learning for robotics still faces many challenges. Firstly, existing DRL research has made a lot of efforts to solve the problems which are analytically intractable by different estimation algorithms and data-driven methods. Nevertheless, it is usually impractical to make the assumption that the state can be completely observable. Moreover, observations are commonly noisy due to disturbance and inherent physical constraints in robotics. Keeping the environmental state containing raw observations and uncer-

tainty on its estimates is usually necessary [33]. Moreover, exponentially increasing computation demands more data to cover the complete search of state-action space as the number of dimensions grows. Therefore, how to achieve efficient training with less samples is critical for the successful applications of DRL in robot control. Furthermore, robotic systems often need to handle the high dimensionality of states and actions because of the many degrees of freedom in most modern robots, which gives another challenge for the use of reinforcement learning in robotics. Finally, although model-based methods have potential to alleviate the above problems, designing a sufficiently accurate model of the robot and its surroundings is also challenging due to complex mechanical interactions, which is difficult to be estimated accurately. Even minor model errors may make the simulated robot diverge far from the real-world physical system rapidly because of the accumulation of errors. When a control policy is learned by using an inaccurate forward model, its behavior can not be easily transferred without significant modifications.

1.2 Main Contributions

This thesis mainly comprises four fundamental contributions to the objectives:

The first is a coherent and systematic set of formulations using graphical model theory and Bayes rule to analyze the interaction and collaboration among multiple snake robot links and external objects. These formulations lead to Bayesian conditional density propagation rules and meaningful probability densities which can be easily interpreted and modeled.

The second contribution is different mechanisms to decouple the model for a deeper investigation. Specifically, three different controllers with a centralized state representation, a dynamically decoupled interaction handling scheme and a fully distributed structure, are studied respectively.

The third contribution is that different likelihood and interaction models are proposed to estimate the probability densities within various implementations. Most of them do not need any explicit analytical form and can be implemented easily without prior knowledge.

The fourth contribution is that we apply the Bayesian model into reinforcement learning and propose two coach-based approaches which greatly expedite the training process. It reflects our

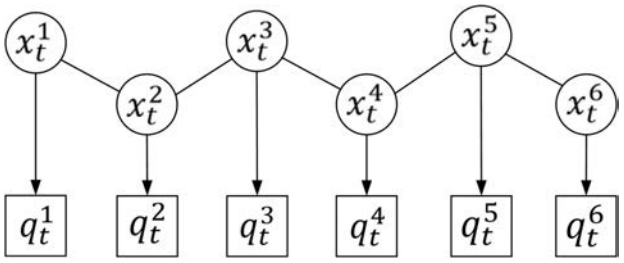

Figure 1.6. Bayesian network for a six-modular snake robot.

preliminary efforts to merge the learning theory with computationally efficient Bayesian framework together.

The proposed Bayesian framework may not only be used as a controller directly but also play an important role to improve the performance for other state-of-the-art control methods of snake robots.

1.3 Overview

1.3.1 Definitions and Symbols

We use a Bayesian network [1] as shown in Fig. 1.6 to model a snake robot with six links, which has two kinds of nodes: circle nodes in the hidden state layer and square nodes inside observation layer. Each circle node represents a snake robot *link*. The undirected edges indicate physical correlations among snake robot links. Each individual state has its observation, where the directed edge between two parties represents the local observation likelihood.

Different snake robot models [9] can be used for gait control. We choose the serpenoid curve [5] in this work,

$$\theta_t^l = A_t^l \sin(\omega t + l\beta) + \gamma_t^l, \tag{1.11}$$

where θ_t^l is the joint angle for actuator l at time t, A_t^l the corresponding amplitude, ω the temporal frequency determining the wave's moving speed, β the phase, γ_t^l the corresponding angular offset, $l = 1, \ldots, L$ the link index of a snake robot, L is the total number. The link state is chosen as

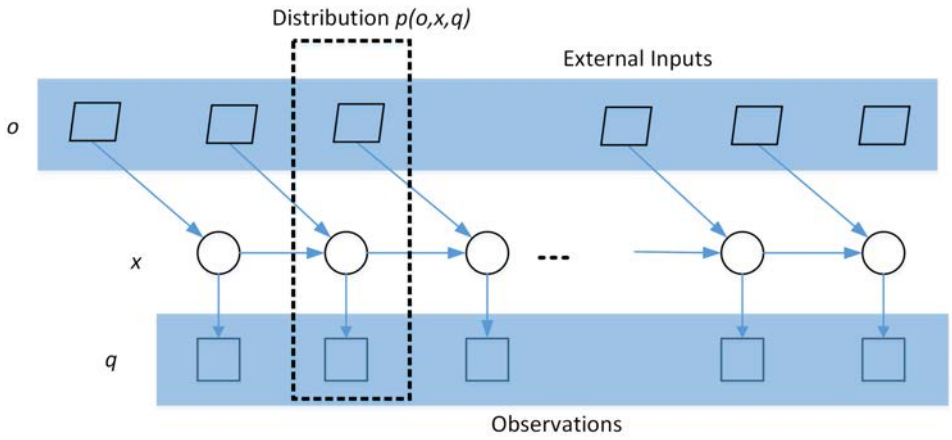

Figure 1.7. Dynamic Bayesian network controller

$\mathbf{x}_t^l = (A_t^l, \gamma_t^l)$ for simplicity though other parameters such as velocity, acceleration, torque, could be included if needed. When these parameters are changing, different shapes would be generated. Moreover, we denote the observation of $\mathbf{x}_t$ by $\mathbf{q}_t$.

Control problem of a robot can be described by a graphical model such as Fig. 1.7 [25]. The circle nodes represent the state sequence $\{\mathbf{x}_t, t = 0, 1, \ldots\}$ of a robot where t is the time index. The nodes $\{\mathbf{q}_t, t = 1, 2, \ldots\}$ represent the observations. The nodes $\{\mathbf{o}_t, t = 1, 2, \ldots\}$ represent inputs. In this work, they are referred to as interactions from environmental objects such at obstacles and targets. The directed edges between states characterize the system dynamics $p(\mathbf{x}_t|\mathbf{x}_{0:t-1})$ (or called state transition). The directed edges between the state and the associated input characterize the likelihood $p(\mathbf{o}_t|\mathbf{x}_{t+1})$. We use $\mathbf{x}_{0:t}$, $\mathbf{q}_{1:t}$ and $\mathbf{o}_{1:t}$ to represent the set of states, observations, and inputs up to time t individually, where $\mathbf{x}_0$ is the initialization prior. Two general assumptions are usually made. The first is that the robot dynamics form a temporal first-order Markov chain, so that $p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The second is that observation $\mathbf{q}_t$ is assumed to be independent both mutually and with respect to the dynamical process, namely, $p(\mathbf{q}_t|\mathbf{x}_{0:t}) = p(\mathbf{q}_t|\mathbf{x}_t)$. The objective of control is to predict the posterior density based on the set of all available inputs and observations. The parameters of a Bayesian network might be inferred, or a training algorithm may be used.

In order to keep the serpenoid shape [5] for the snake robot, a joint state representation can be usually adopted as $\mathbf{X}_t = (\mathbf{x}_t^1, \ldots, \mathbf{x}_t^l, \ldots, \mathbf{x}_t^L)$. It encodes the hidden information of the whole

snake robot body. $\mathbf{Q}_t = (\mathbf{q}_t^1, \ldots, \mathbf{q}_t^l \ldots, \mathbf{q}_t^L)$ denotes the set of observations of all robot links at time t. $\mathbf{O}_t$ denotes the set of all interacted environmental objects for the whole body of snake robot at time t. Moreover, the set of all history states up to time t is denoted by $\mathbf{X}_{0:t}$ where $\mathbf{X}_0$ is the initialization prior. Similarly, we define the group of all observations as $\mathbf{Q}_{1:t}$, the group of all interacted environmental objects by $\mathbf{O}_{0:t}$ and $\mathbf{O}_0$ is the initialization prior.

1.3.2 Outline of This Thesis

The purpose of this research is to develop robust Bayesian controllers for the locomotion of a snake-like robot in cluttered scenes. The contents of this thesis are organized as follows.

Chapter 2 introduces a simulation platform and a designed snake-like robot *JAW*-I for real-world experiments. The hardware specification and software description are all provided. The proposed control algorithms in the following chapters were verified by the simulator and physical testing using the snake robot *JAW*-I.

Chapter 3 outlines a centralized Bayesian solution for snake robot control. Different with the existing shape-based compliant control, it puts the control problem inside a statistical field. Specifically, it gives an explicit mathematical Bayesian derivation. A sequential density update rule is presented. Several probability densities with different physical meanings have been unified in a consistent framework. In particular, we propose two input influence densities, which simulates the cumulative effect of various external forces that the snake robot undergoes. Moreover, the observation likelihood model is used to provide an effective closed-loop feedback. The proposed approach give an innovative way to handle challenging tasks of snake robot control in complicated environment handling uncertainties in a unified Bayesian framework. The performance has been demonstrated on both simulation and real-world data.

Chapter 4 presents a method which avoids the common practice of using a complex fully coupled snake robot model and performing kinematics and dynamics analysis for control in cluttered environments. Instead, we introduce a dynamically decoupled Bayesian formulation with respect to interacted snake robot links and environmental objects, which requires much lower complexity

for efficient and robust control. When a snake robot does not interact with obstacles, it runs by a simple serpenoid controller with centralized state propagation. When it exhibits interaction with environments, defined as close proximity or collision with targets and/or obstacles, we extend the conventional Bayesian framework by modeling such interactions in terms of stimuli. The proposed "*multi-neural-stimulus function*" represents the cumulative effect of both external environmental influences and internal constraints of the snake robot. It implicitly handles the "*unexpected collision*" problem and thus solve the difficult data association and shape adjustment problems for snake robot control in an innovative way. Preliminary experimental results have demonstrated promising performance of the proposed method comparing with the state-of-the-art.

Chapter 5 describes a fully distributed Bayesian framework which can greatly reduce the computation complexity by exploiting efficient parallel computing. Different with the centralized framework which uses a definition of high dimensional state and thus faces exponentially increased computational cost with link number, this decentralized solution adopts multiple controllers, one for each robot link, simultaneously. Some decentralized control solutions have been reported, for example, using central pattern generators. However, it still lacks enough studies of the dynamical correlation among snake links as well as environmental interactions. The proposed approach has several major contributions: 1) A novel graphical model is presented to set up the snake robot control problem; 2) A completely distributed probabilistic updating rule is derived mathematically for each robot link within a stochastic process, where the uncertainty has been covered and simulated; 3) An inter-link likelihood and a link-based environmental input function are introduced to simulate the interaction among snake robot links and with environmental objects. Superior performance has shown validity of the proposed method on both simulations and real world tests.

Chapter 6 presents a centralized coach-based method for snake robot control using Bayesian analysis, which successfully combine the advantages of both stochastic density propagation analysis and reinforcement learning. It can effectively expedite the training speed with much less episodes than existing methods. The main contributions of the proposed method include: 1) a unified graphical model for Bayesian analysis, which combines a coach layer to guide the RL agent, embedding prior knowledge and handling uncertainty in model parameters; 2) an explicit stochastic formu-

lation of robot-environment interaction with different uncertainties; 3) an efficient and effective learning process for snake robot control, which can make the path planning and obstacle avoidance simultaneously. The proposed probabilistic dynamics model allows the agents to consider transition uncertainty throughout planning and prediction, greatly improving the data efficiency. We have reported both simulation and real-world experimental results comparing with state-of-the-art.

Chapter 7 extends the coach-based reinforcement learning control framework by exploiting a distributed architecture, which has been demonstrated in providing more complexity during the interaction handling. It can further expedite the training convergence speed by using parallel computing. The advantage of distributed topological structure is successfully combined in a coherent formulation with RL. Simulation results have shown the promising performance.

Finally, in chapter 8, a full discussion and comparison of the proposed methods is given. After that, conclusion of the thesis as well as an outline of some promising future research are provided in chapter 9.

Chapter 2

Simulation and Experimental Platform

The goal of this research is to propose robust Bayesian controllers for a snake-like robot. To verify the control systems, simulation and physical experiment platforms are necessary. Therefore, a 3-D virtual dynamic simulation environment has been developed. For physical experiment, a snake-like robot, JAW-I, is designed to verify the proposed controllers.

2.1 V-REP Based Simulation Setup

To verify the proposed control approaches, a simulator has been developed in Virtual Robot Experimentation Platform (V-REP) [36]. V-REP is a versatile and scalable robot simulation frame-

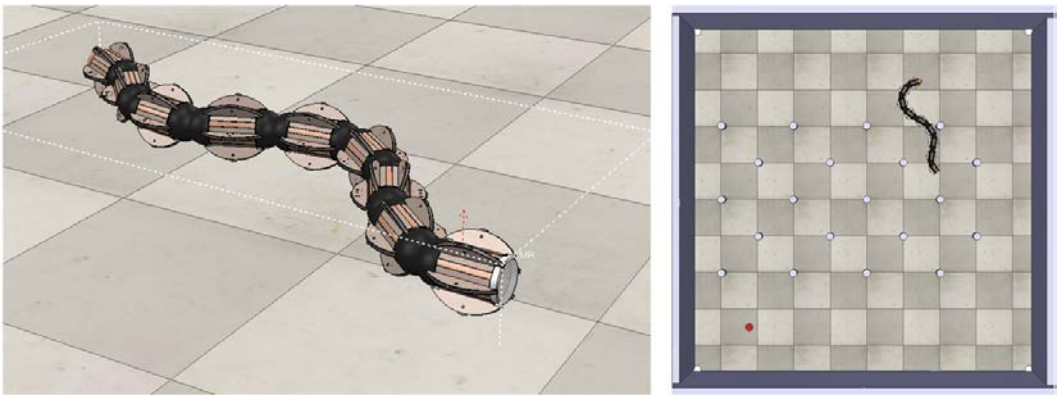

Figure 2.1. The used ACM-R5 robot and simulation scenario in V-REP.

Table 2.1. Physical parameters of the simulated robot

Number of joints	8
Size of link (mm)	$124 \times 139 \times 201$
Mass of link (g)	300
Moment of inertia (kgm^2)	1.612×10^{-3}
Max torque (Nm)	10
Joint angle (deg)	$[-90, 90]$

work. Specifically, we used ACM-R5 snake robot as shown in Fig. 2.1, which is composed of 8 joints. The physical parameters used are given in Table 2.1. The anisotropy in friction is achieved by equipping passive wheels on each link. The time step is set to be 10ms. Instead of using torque as input, we calculated the angle of each joint that is realized by the controller. 20 cylinder obstacles were set inside a 5m $\times$ 5m scene. Two sensors including a camera and LiDAR are used to give observations. The camera provides bird-view RGB images, which can be represented by a tensor of $\mathbb{R}^{128\times128\times3}$. For the LiDAR, we project the point cloud to the ground plan getting a 2D image represented by a tensor of $\mathbb{R}^{128\times128\times3}$. These two sensors can provide complementary information. LiDAR point clouds can give accurate spatial information of target and obstacles in 360 degrees of view while the bird-view camera image of each robot link is good at providing information of the local surroundings.

2.2 Experimental Platform - *JAW*-I

2.2.1 Physical Prototype

The overview of designed snake robot *JAW*-I is shown in Fig. 2.2. Similar as the robot model reported in [37], it has six actuators chained together in a serial configuration. The links are composed of 3D-printed connection blocks and separated by DYNAMIXEL XL430-W250 servo motors (manufactured by ROBOTIS Inc., Seoul, South Korea). For simplicity, only yaw joints rotating on a vertical axis between adjacent links are adopted in the tests. Therefore, the snake-like robot can

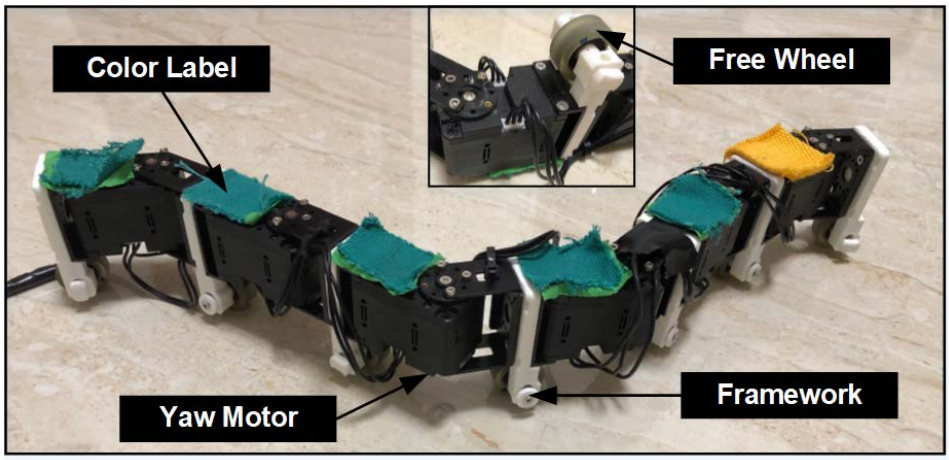

Figure 2.2. Overview and mechanical structure of the snake robot *JAW*-I for real-world tests.

Table 2.2. Mechanical specifications of the snake robot *JAW*-I

Items	Details
Number of joints	6
Link Size(mm^3)	$75 \times 35 \times 65$
Weight of link (g)	72
Motion range of yaw angle (deg)	[-180, +180]
Width of wheel (mm)	9
Radius of wheel (mm)	11
Weight of wheel (g)	25

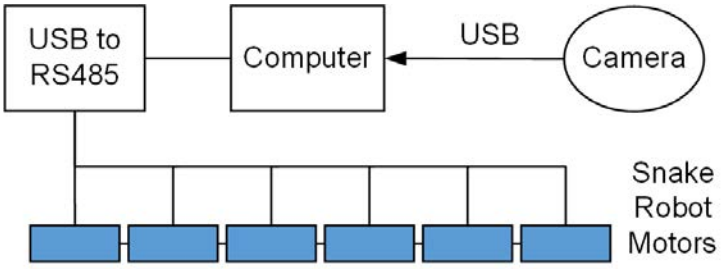

Figure 2.3. The experimental setup for real-world testing.

move on a two-dimensional horizontal plane. Labels of different colors are used to indicate links. Specifically, yellow for robot head and green for body. A single free wheel with a rubber cover is mounted at the bottom of each link to realize asymmetric friction in normal and tangential directions. Suspension mechanisms by compression coil springs are mounted at the bottom of each link. The specifications of the robot are shown in Table 2.2.

2.2.2 Control System

The control system design of the snake-like robot *JAW*-I is shown in Fig. 2.3. The robot is controlled by signals sent from a Linux computer through an RS-485 communication module. Similar as [2] [15], an overhead camera is used to monitor the scene, providing observations for each robot link. Tags of different colors are used to label objects, specifically, red for targets, blue for obstacles. A Convolutional-Neural-Network learning based video detector [38] was exploited to estimate the object position and robot link's shape variation in real time. Consequently, the stimulus density and measurement likelihood could be dynamically calculated.

2.3 Summary

In this chapter, we introduced the design of the simulation and experimental platform. The motion of a snake robot can be simulated in an virtual dynamical environment V-REP. A sensor-driven snake robot *JAW*-I is implemented to perform desired serpenoid locomotion in unstructured scenarios. In the following chapters, the proposed controllers will be verified by the designed simulator and the physical snake-like robot *JAW*-I on the real-world data.

Chapter 3

Stochastic Centralized Bayesian Control

A snake robot is characterized by features of a redundant structure, multiple joints, and a modular framework that enable a wide range of locomotion patterns and adaptation to environment. Such kind of characteristics makes snake-like robot control a challenging task in practical applications. A natural and widely adopted choice is based on a centralized solution which regards the articulated snake robot as a whole. The global controller can coordinate the multiple links together presenting rhythmic gait. However, due to the uncertainty incurred by noisy perceptions, unknown environments, and dynamical collisions, robust control of snake robots is still an open problem. In this chapter, we discuss this issue and investigate the snake robot control in a stochastic domain.

Comprehensive surveys on existing methods of snake robot control can be found in [11] [12] [39]. For snake robot applications, sophisticated control mechanisms, such as model-based approaches [13] and Central Pattern Generators (CPG) [12] were used. Nevertheless, when snake robots move in unstructured terrain or collide with barriers, these approaches are prone to failure. The movement of snake robots using contact with objects as an aid has been examined by several studies. In [40], Transeth et al. present a hybrid model based on robot dynamics and the contact force with obstacles. Obstacle-assisted snake robot approaches often depend on assumptions of precise sensing and modeling, in which the convexity and number of friction models are often predetermined. These assumptions restrict their applications in unstructured terrains, since it is usually difficult to predict and hard to model the constraints. Some study investigates approaches of neural

networks [12] to model the interaction mechanism of biological snakes with objects and utilizes machine learning approaches [12]. Obstacle avoidance for snake robots has been addressed in different ways. A neurally controlled steering strategy for a snake robot's collision-free behavior is presented in [41]. Tanaka et al. [13] studied self-collision avoidance and provided a range-sensor-based technique to achieve whole body collision avoidance semi-autonomously. A shape-based compliant controller [14] [15] has recently been used to provide a computationally efficient method to snake robots that addresses various challenges associated with this complex work. Even if these methods can handle external interactions in theory, they requires accurate measurement of torques or forces, as well as a nice evaluation of the controller's gain matrices, which confines their application, particularly in unknown complex environments, in which the dynamics are changing and difficult to model reliably.

The majority of known controllers, such as those in [15], are deterministic. However, deterministic mathematical modeling of robotic systems is often inaccurate due to different kinds of uncertainties, say, sensor noise, input disturbance, a discrepancy between control signals and mechanical actuations, vagueness, and model incompleteness etc. Because deterministic simulation of system dynamics is often inaccurate, a statistical technique is required to approximate unknown parameters and assess their accuracy. To tackle stochastic situations, Kalman filtering is widely utilized [18]. The Kalman filter was used by Rollinson et al. [19] to estimate states of snake robots. Traditional Kalman filters, however, assume a Gaussian noise distribution, which is not valid in many cases. For various aspects of dynamic robot systems such as control and navigation, advanced statistical approaches such as Bayesian network [1] have been examined. As far as we know, nonetheless, not quite enough work has been done using Bayesian network in stochastic field for snake robots.

We present a Bayesian network-based snake robot controller in this chapter. Some state-of-the-art techniques are unified into a coherent formulation. In particular, we deduce an unique conditional density propagation strategy for snake robot control in unstructured environment with disturbances. Furthermore, we explicitly cope with shape variation and obstacle interaction issues in a unique style by calculating observation probability, the input influence density, and state transition in a sequential importance sampling process.

3.1 Methodology of the Centralized Framework

A novel Bayesian formulation for snake robot control is presented next in this section. First, a brief overview of shape-based compliant control is provided.

3.1.1 Shape-Based Compliant Control

Shape-based compliant control uses an admittance controller to adjust the snake robot's shape parameters by the external torques F_t [15]. If $\beta_t = (A_t, \gamma_t)^T$ is defined, in which A and γ are the amplitude and angular offset respectively in the serpenoid curve model [5], the compliant controller is defined by

$$M\ddot{\beta}_t + B\dot{\beta}_t + K\beta_t = F_t, \tag{3.1}$$

in which $M, B, K \in \mathbb{R}^{2\times 2}$ respectively denote the effective mass, damping, and spring constant matrices. This technique may appropriately adjust the snake robot's body in response to obstacles by incorporating the serpenoid model into the shape function and tuning the parameters of the aforementioned controller.

There are still a few challenges ahead for this controller: 1) M, B and K, the optimal gain matrices of a snake robot are usually hard to determine; 2) External torque F_t is difficult to accurately measure during collisions because it varies in magnitude, angle, direction and so on; 3) It lacks an observer design, despite the fact that the external force may alter the snake robot's posture implicitly. Thus, it's hard to precisely approximate the controller state. To this end, we suggest a more sophisticated shape-based control mechanism.

3.1.2 Shape-Based Bayesian Network Modeling

In this chapter, a joint state representation is adopted as $\mathbf{X}_t = (\mathbf{x}_t^1, \ldots, \mathbf{x}_t^l, \ldots, \mathbf{x}_t^L)$ for snake robot representation where the link state is chosen as $\mathbf{x}_t^l = (A_t^l, \gamma_t^l)$ by the serpenoid curve. Considering the locomotion problem, a dynamical Bayesian network as shown in Fig. 3.1 is used. In this figure, states of the whole body of the snake robot are denoted by circle nodes, the corresponding observations are represented by square nodes. Totally three consecutive time frames are shown.

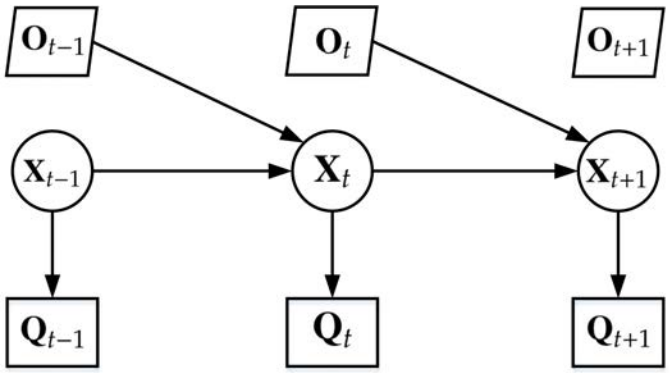

Figure 3.1. Dynamical modeling of snake robot locomotion.

The state transition that is assumed to be a *Markov* chain, is denoted by an arrow between adjacent states. We used an additional layer to indicate the input $\mathbf{O}_t$, which represents obstacles and targets in the environment during interaction. From Fig. 3.1, we can derive the Markov Properties, or, conditional independence properties as below, by using the *Separation Theorem* [1].

$$p(\mathbf{O}_t|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t}) = p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) \tag{3.2a}$$

$$p(\mathbf{Q}_t|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t-1}) = p(\mathbf{Q}_t|\mathbf{X}_t) \tag{3.2b}$$

$$p(\mathbf{X}_{t+1}|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}) = p(\mathbf{X}_{t+1}|\mathbf{X}_t) \tag{3.2c}$$

3.1.3 Bayesian Conditional Density Propagation

In comparison with Kalman filters and difference equations, the dynamic Bayesian network derived above has the benefit for statistical investigation. In particular, It offers much modeling flexibility for stochastic processes. A prediction issue is studied in our work to design a controller. All the history of evolution of uncertainties is captured by $p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$, the joint posterior, which is derived to achieve the prediction.

$$
\begin{aligned}
& p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\
&= p(\mathbf{O}_t|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t}) p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) && (3.3) \\
&= p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) && (3.4) \\
&= \frac{p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{Q}_t|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t-1})}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})} p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t-1}) && (3.5) \\
&= \frac{p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{Q}_t|\mathbf{X}_t) p(\mathbf{X}_{t+1}|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t})}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})} p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) && (3.6) \\
&= \frac{p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{Q}_t|\mathbf{X}_t) p(\mathbf{X}_{t+1}|\mathbf{X}_t)}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})} p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) && (3.7) \\
&= c_t p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{Q}_t|\mathbf{X}_t) p(\mathbf{X}_{t+1}|\mathbf{X}_t) p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}). && (3.8)
\end{aligned}
$$

We applied the *Markov* property (3.2a) in eq. (3.4). The *Markov* property (3.2b) is also used in eq. (3.6). The property (3.2c) is applied in eq. (3.7). We take the denominator as a normalization constant in eq. (3.7), because this item does not contain states.

The interactions between the robot and the environment is explicitly simulated by the stochastic Bayesian formulation above. We can see that four factors regulate the posterior at time t: (1) the input influence density $p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t)$; (2) the observation likelihood $p(\mathbf{Q}_t|\mathbf{X}_t)$; (3) the state transition density $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$; (4) the posterior $p(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})$ at the previous time $t-1$.

3.2 Density Modeling and Sequential Importance Sampling Implementation

The *sequential importance sampling* (SIS) [42] is adopted as a paradigm to approximate the posterior obtained above. The underlying concept of SIS approximation [42] is to manage a weighted sample set $\{\mathbf{X}_{0:t}^n, w_t^n\}_{n=1}^{N_s}$ to approximate the posterior density, in which $\{\mathbf{X}_{0:t}^n, n = 1, \ldots, n_s, \ldots, N_s\}$ are the samples, $\{w_t^n, n = 1, \ldots, n_s, \ldots, N_s\}$ the normalized weights associated, and $\sum_n w_t^n = 1$. Based on the *importance sampling theory* [42], the samples $\mathbf{X}_{0:t}^n$ can be produced from an importance density $f(\cdot)$ with associated importance weights:

$$
w_t^n \propto \frac{p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}^n|\mathbf{Q}_{1:t})}{f(\cdot)}. \tag{3.9}
$$

As to the sequential case, if the importance density $f(\cdot)$ is factorized as,

$$\begin{aligned} & f(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ = & f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}, \mathbf{Q}_{1:t}) f(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t}) \\ = & f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{X}_t) f(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}). \end{aligned} \tag{3.10}$$

where the *Markov* properties $f(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t}) = f(\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})$ and $f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{O}_{0:t-1}, \mathbf{X}_{0:t}, \mathbf{Q}_{1:t}) = f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{X}_t)$ from Fig. 3.1 are applied. Then by substituting eq. (3.8) and eq. (3.10) into eq. (3.9), we have

$$w_t^n \propto w_{t-1}^n \frac{p(\mathbf{O}_t|\mathbf{X}_{t+1}^n, \mathbf{X}_t^n) p(\mathbf{Q}_t|\mathbf{X}_t^n) p(\mathbf{X}_{t+1}^n|\mathbf{X}_t^n)}{f(\cdot)}. \tag{3.11}$$

We can approximate the dynamics $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$ by applying a random walk model. How to estimate the input influence density $p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t)$, the observation likelihood $p(\mathbf{Q}_t|\mathbf{X}_t)$ and the choice of the importance density $f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{X}_t)$ are critical and will be developed next.

3.2.1 Input Influence Model

The interaction between environmental inputs and predicted state is modeled by $p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t)$, the input influence density. The approximation of this probability is very important in different applications. An Artificial Potential Field (APF) method was adopted to model snake robot locomotion in [43]. This inspired us to develop two efficient input influence models to manage obstacle interaction and target searching. The two interactions are regarded as independent, therefore,

$$p(\mathbf{O}_t|\mathbf{X}_{t+1}, \mathbf{X}_t) = p(\mathbf{O}_{a,t}|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{O}_{r,t}|\mathbf{X}_{t+1}, \mathbf{X}_t) \tag{3.12}$$

in which $p(\mathbf{O}_{a,t}|\mathbf{X}_{t+1}, \mathbf{X}_t)$ formulates the influence by virtual attraction force from the goal, $p(\mathbf{O}_{r,t}|\mathbf{X}_{t+1}, \mathbf{X}_t)$ models influence of the accumulative repulsion force resulted from obstacles. We formulate the probability densities as below.

Target Influence Model

The objective of a goal searching task is to arrive at the target as soon as possible. Thus, we use an attraction potential model to instruct the robot to explore the optimal route [43]. Meanwhile, the attraction potential function for sample $\mathbf{X}_{t+1}^n$ can be calculated by

$$p(\mathbf{O}_{a,t}|\mathbf{X}_{t+1}^n, \mathbf{X}_t^n) = \frac{1}{\alpha_a}\exp\left\{-\frac{d_{a,n,t}^2}{\sigma_a^2}\right\} \tag{3.13}$$

in which α_a is a constant, σ_a is a prior constant characterizing the maximal distance of attraction. $d_{a,n,t}$ represents the distance of the robot and the goal. For example, it can be an Euclidean distance $d_{a,n,t} = \|\mathbf{O}_{a,t} - \Delta\mathbf{X}_{t+1}^n\|$, where $\Delta\mathbf{X}_{t+1}^n = \mathbf{X}_{t+1}^n - \mathbf{X}_t^n$. The target is static in the original APF model [43]. However, the goal in our model can be movable during the locomotion of the robot.

Obstacle Influence Model

[43] used a repulsive potential model to control snake robot. In a similar way, for sample $\mathbf{X}_{t+1}^n$, the repulsive potential function is represented by

$$p(\mathbf{O}_{r,t}|\mathbf{X}_{t+1}^n, \mathbf{X}_t^n) = 1 - \frac{1}{\alpha_r}\exp\left\{-\frac{d_{r,n,t}^2}{\sigma_r^2}\right\} \tag{3.14}$$

in which α_r denotes a normalization constant, σ_r indicates a prior constant representing the maximal effective repulsive distance, $d_{r,n,t}$ indiates the distance between the obstacle $\mathbf{O}_{r,t}$ and $\mathbf{X}_{t+1}^n$, for instance, we can use an Euclidean distance to measure $d_{r,n,t} = \|\mathbf{O}_{r,t} - \Delta\mathbf{X}_{t+1}^n\|$ where $\Delta\mathbf{X}_{t+1}^n = \mathbf{X}_{t+1}^n - \mathbf{X}_t^n$. Comparing with the conventional APF approach in [43], in which all obstacles are handled together, We only consider neighboring obstacles according to necessities of the sample $\mathbf{X}_{t+1}^n$. In this way, the model is more practical for real applications because the distributions and whole number of obstacles are difficult to predetermine and may be subject to dynamic varying. Furthermore, such a mechanism is also effective to tackle the issue of *local optimum* when the potential field is calculated, because we only take neighbouring obstacles into account.

During the locomotion in an unstructured terrain, the snake robot moves in a region with potentials resulted from obstacles and the target. An instance of an input influence potential field for

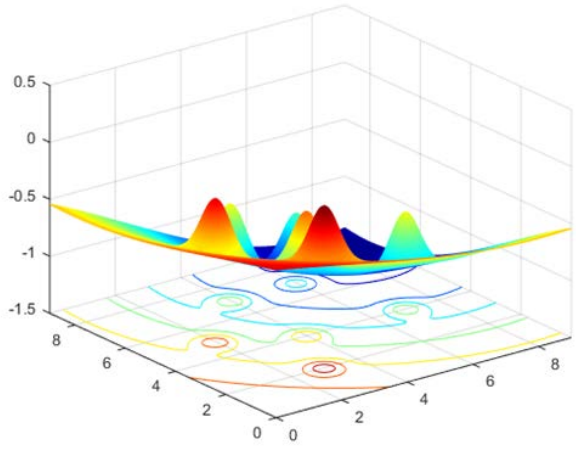

Figure 3.2. The input influence potential field with obstacles and one target.

a cluttered environment, is shown in Fig. 3.2. As shown, it seems to be a large slope, in which the goal is at the bottom. The influence of objects is local, and target attraction influence is global. Every obstacle functions like a *hill*. And the robot locomotion is like traversing over a hilly area to arrive at the goal. Instead of being static, the potential field gets dynamically updated during the whole navigation.

3.2.2 Measurement Likelihood Model

Thanks to the embedded serpenoid function, the robot is able to produce rhythmic gaits and locomote with state transitions. Benefiting from these interaction functions, the robot can make responses, vary its state and shape in unstructured environment. Nevertheless, the robot would be lack of state estimation and have low reliability if there is no design for observer. We use the observation likelihood density $p(\mathbf{Q}_t|\mathbf{X}_t)$ to approximate the uncertainties between the robot's state and its observation. This density is used as a feedback to handle the above issues. Modeling this probability density is challenging and very important. We got inspired by the nice shape variation of real snakes when they locomote in complex environment. A biological snake is aware of its internal shape and would make body deformation smoothly. This is why it seldom overreacts when making swift transient deformation in response to external perturbations. The objective of an effective

observation feedback model should be similar to those observed in the biological snakes, which can be obtained by an observation likelihood function for sample $\mathbf{X}_t^n$,

$$p(\mathbf{Q}_t|\mathbf{X}_t^n) = \frac{1}{\alpha_m}\exp\left\{-\frac{d_{m,n,t}^2}{\sigma_m^2}\right\} \tag{3.15}$$

where α_m indicates a normalization constant, σ_m indicates a prior constant characterizing the maximal effective distance, $d_{m,n,t}$ represents the distance between $\mathbf{X}_{t+1}^n$ and the corresponding observation $\mathbf{Q}_t$, for example, a Bhattacharyya distance is accepted in our implementation. This model offers the system the ability of simulating biological snakes' shape deformation when making responses to environmental disturbances.

In conclusion, the proposed mechanism can accomplish robust snake robot control due to the factors: 1) The serpenoid curve model guarantees the snake robot moving in a rhythmic pattern; 2) The state transition density introduces motion randomness and thus endows the robot's locomotion with more possibilities; 3) The input influence incurs shape deformation and motion variation; 4) The observation likelihood provides a closed-loop feedback and thus retains the transformation in a fast and smooth way. All these four reasons are necessary to achieve a robust control for snake robots. When no obstacle or target appears, the input influence density will be uniformly distributed. The controller will be degraded to a serpenoid model, if the state transition also adopts a uniform distribution.

3.2.3 Importance Density

The efficiency of a sequential importance sampling based approach highly relates with the selected importance density $f(\cdot)$. When $f(\cdot)$ is close to the true posterior, the samples become more effective. A natural choice of the importance density is the state dynamics $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$. In our implementation, we use this choice for simplicity.

Table 3.1. Brief description of CenBC algorithm

- $j = 0$, Sampling $\mathbf{X}_{t+1}^n$ from $f(\mathbf{O}_t, \mathbf{X}_{t+1}|\mathbf{X}_t)$
- Initial Weighting, $w_{t+1}^n \sim \mathbf{X}_{t+1}^n$
- Initial Prediction, $\hat{\mathbf{X}}_{t+1,j}$=$\sum_{n=1}^{N_s} w_{t+1}^n \cdot \mathbf{X}_{t+1}^n$
- Loop $j = 1 : J$ # Re-sampling Scheme
  - Input Influence Weighting, $p_1(\cdot)$
  - Measurement Likelihood Weighting, $p_2(\cdot)$
  - Updating Weights, $w_{t+1}^n = w_{t+1}^n \cdot p_1(\cdot) \cdot p_2(\cdot)$
  - Normalizing Weights
  - Updating Predication, $\hat{\mathbf{X}}_{t+1,j}$=$\sum_{n=1}^{N_s} w_{t+1}^n \cdot \mathbf{X}_{t+1}^n$
- End Loop j

3.3 Experimental Results

The proposed Centralized Bayesian Controller (CenBC) was compared with the Shape-based Compliant Controller (SCC) [15] on both simulation and real world data. Table 3.1 presents a brief description of the proposed algorithm for one time slot. Five hundred samples were used to predict the joint state density in our experiments.

3.3.1 Simulation Results

We designed a simulation in Matlab for a thorough comparison. Fig. 3.3 shows the experimental scenario with one target and thirteen obstacles, which are all randomly arranged. A potential field is generated by calculating the proposed input influence models. Although sharing similarities with APF [43], we only consider the obstacle in the neighbor of the snake robot during the locomotion. By doing this, the local optimum problem commonly annoying APF-type methods could be successfully avoided. Three trajectories are illustrated in Fig. 3.3. The blue dash curve is generated

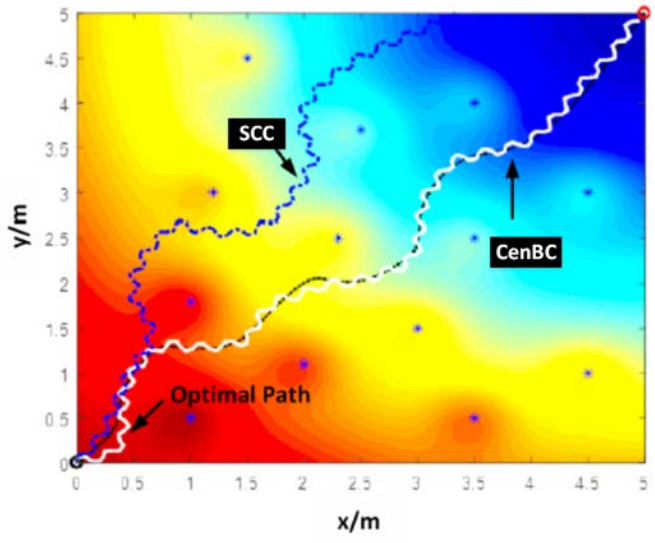

Figure 3.3. Comparison of simulation results using CenBC (black) and SCC (blue).

by the head of a snake robot using SCC. The white solid curve shows the results of CenBC. The black solid curve is an optimal path computed by a gradient-descent algorithm based on the input influence potential field. In experiments, we find that the performance of SCC is sensitive to the initialization status. Different initial direction and position may lead to various trajectories. As illustrated in Fig. 3.3, the SCC method falsely misses the target direction in the middle and runs outside the field. Although the external forces are partially modeled, the collision is still uncontrollable in terms of contacting direction, degree, and the phase of snake robot's serpenoid gait. However, benefiting from the closed-loop design, CenBC can achieve much more robust performance. As long as the target can be detected, the snake robot tends to move for it. When it approaches to an obstacle, the effect of repulsion force will be triggered. The sample close to the obstacle will have a smaller weight while the sample away to the obstacle will be given a larger weight. Such a *reward-far-punish-close* scheme will deform the snake robot's shape and motion curve.

Fig. 3.4 shows the RMSE of SCC and CenBC on ten tests with the same target and similar initial position. The curves are calculated by computing summation of the difference between each method's trajectory and the optimal path. It can be seen that CenBC is more robust. The errors of CenBC are caused by three factors: 1) Due to the different initial status, it may need an adjustment process at the beginning; 2) The uncertainty may cause a bias during sharp turnings; 3) The

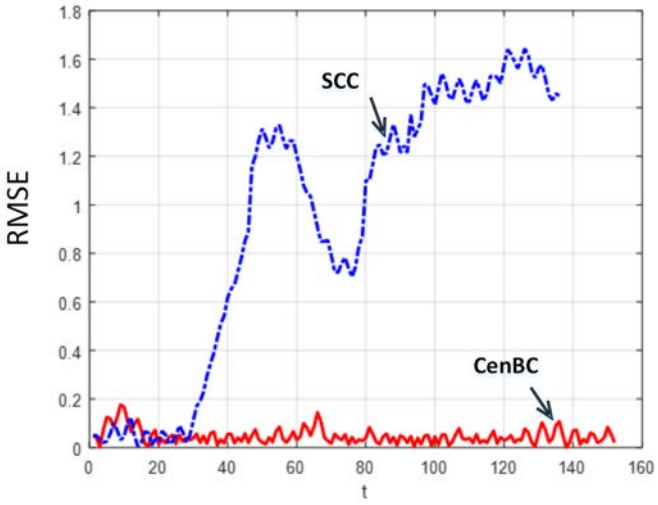

Figure 3.4. Comparison of Root-Mean-Square-Error (RMSE) using CenBC and SCC methods.

environmental friction and interference may further generate additional deviations. These kinds of uncertainties are all unpredictable in the real-world applications but need to be considered. This is why a stochastic framework like the proposed CenBC is desirable.

3.3.2 Real-World Data

The snake-like robot *JAW*-I is used to test the performance of the proposed CenBC. A learning based video detector was exploited to find the position of these objects in real-time. Then, the input influence density and observation likelihood could be dynamically calculated, specifically, observation $\mathbf{Q}_t$ was estimated by a link model [12] between centers of adjacent links.

Fig. 3.5 illustrates the performance using both SCC and the proposed CenBC in a real-world test, where one target and more than twenty obstacles are randomly set in the scene. It challenges many algorithms because of the presence of moving targets and obstacles. SCC (1st row) suffered from the *unexpected collision* problem during the interaction with obstacles. The reason is mainly because of the complexity of environmental interactions. For example, the friction situation, collision angle, and strength of contacting force are all not handled explicitly in the controller. In the experiments, we found that the controller was sensitive to the initial pose. Moreover, although the shape was deformed by portions, it could not respond to moving obstacles in time due to lack of

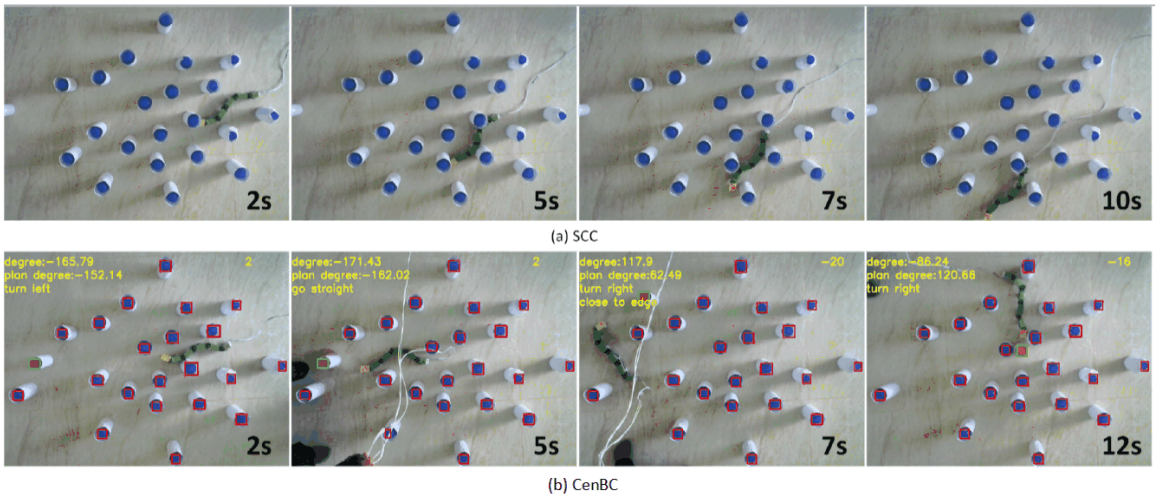

Figure 3.5. Performance comparison of SCC and CenBC for unstructured environments.

a closed-loop design. Furthermore, without a target influence model, it usually missed the desirable direction after spontaneous collision with obstacles and finally went out of the scene quickly. However, the proposed CenBC (2nd row) performed superiorly even with moving obstacles and targets. In most situations with different initialization status, the proposed CenBC method presented an impressive performance. Although a joint state space is adopted, the snake robot's shape is not exclusively changed as a whole. On the contrary, different portions can be flexibly stimulated by external inputs similar as [4]. Specifically, due to the various position along a snake robot, each link may receive different interaction forces even when reaching to the same obstacle. In such a situation, the distances defined in input influence models are quite different among individual links and thus play an important role to change the snake robot's shape. Thus, the proposed framework endows the entire body with a particular ability to deform locally by environmental features. Compared with the deterministic controller such as SCC, the snake robot's locomotion using CenBC is more *active* and *agile*. The reason may be the stochastic modeling inherent in the non-Gaussian density propagation. Benefiting from keeping multi-hypotheses of both motion and shape variations, the snake robot can respond to instant moving target and obstacles quickly.

3.4 Summary

In this chapter, a centralized Bayesian control framework has been proposed for a snake-like robot. It models the interaction with environmental objects using probability density propagation. two input influence densities are proposed to model the cumulative effect of various external forces that the snake robot undergoes. Moreover, the observation likelihood model is exploited to give a more robust closed-loop feedback. Preliminary experimental results have demonstrated promising performance in unstructured circumstances.

Chapter 4

Dynamically Decoupled Bayesian Control

Snake robot locomotion in a cluttered environment introduces additional challenges beyond the uncertainty problem in the previous chapter. It is a complicated and computationally expensive task because the motion model is discontinuous due to the frequent physical contact with obstacles, and the contact force cannot be determined solely by contact positions. Moreover, the highly redundant structure with a large number of DOFs makes control and navigation further difficult. [12]. In these scenarios, how to dynamically model the high-level gait-based interactions inside a snake robot and the low-level interactions between the robot link and environmental objects is important for snake robot control. In this chapter, we regard the snake robot as a hierarchical articulated entity, which has both global gaits and local adaptability simultaneously.

Many approaches have been studied to circumvent the problems inherent in snake robot control. Cao et al. proposed an adaptive path following scheme to control a three dimensional snake robot to navigate along certain paths [44]. However, these approaches usually fail when a snake robot moves in unstructured environment or presents collisions with obstacles. In these circumstances, modeling surrounding interactions and shape adjustments are the most critical problems. Without an effective scheme to model the interaction between the robot and neighboring conditions, and solve shape variation problem, conventional control of snake robots commonly suffers from "*unexpected*

collision" problem, where the robot contacts with obstructions either undesirably in an obstacle-free behavior or by unwanted direction, false angle or unsuitable torque in obstacle-aided situations.

Most existing methods rely on accurate modeling and sensing assumptions, which limit their performance in unstructured environments where constraints commonly are hard to foreknow and difficult to model [45] [46] [47]. Several studies [48] began to examine semi-autonomous collision avoidance of a snake robot based on range sensor data. Bayes filters have been studied by Thrun et al. for probabilistic robotics in [49]. Yu et al. proposed a Bayesian method to acquire the estimation of human impedance and motion intention in a human-robot collaborative task [50]. A Bayesian framework for the active multi-modal perception of 3D structure and motion is presented by Ferreira et al. in [51]. Nevertheless, snake robot control with Bayesian filtering has not been fully investigated, especially for the interaction with environments. In [2], we introduced a shape-based Bayesian controller for a snake robot to navigate in cluttered environment, where virtual forces generated by external objects are modeled.

In this chapter, we propose a dynamically decoupled Bayesian framework for snake robot control having intensive interaction with environment. Different with conventional Bayes filter [49], it dynamically splits the interaction with respect to interacted robot links and associated environmental objects, which effectively decreases model complexity and implicitly handles the "*unexpected collision*" problem in an innovative way.

4.1 Methodology of the Decoupled Framework

Next, a probabilistic graphical model (PGM) is presented, and Bayesian formulation for snake robot control is discussed.

4.1.1 Probabilistic Graphical Modeling

Conventional models of snake robots [5] could not thoroughly handle the dynamic interactions between the robot and objects in the environment. Recently, researchers have widely adopted probabilistic sequential analysis for control and prediction tasks [52] [53]. A probabilistic graphical

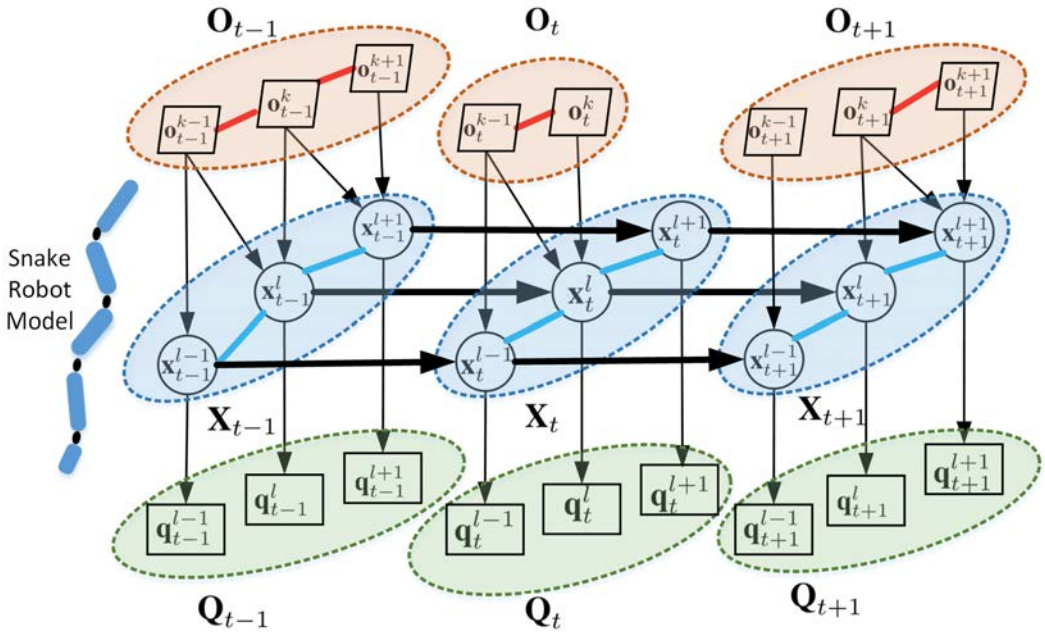

Figure 4.1. A PGM for snake robot control having intensive interactions with the environment

model is developed to model the mutual interplay as shown in Fig. 4.1, which has three consecutive time slices with multiple layers. Environmental object tier (red) is indicated by **O**, robot hidden state tier (blue) by **X**, and observation tier (green)by **Q**. Fig. 4.1 only illustrates three adjacent modules and their neighboring obstacles for simplicity. The same type of nodes are grouped and encompassed by dashed ellipses. The grouping relationship dynamically changes with the robot's journey because of the varied correlation among surrounding objects. In Fig. 4.1, a link model is used to denote the snake robot, and a circle node **x** is used to indicate the state of each link. Quadrilateral nodes represent the observation **q**, and Parallelogram nodes in tier **O** indicate objects in environment. In Chapter 3, we have defined the joint state representation as $\mathbf{X}_t = (\mathbf{x}_t^1, \ldots, \mathbf{x}_t^l, \ldots, \mathbf{x}_t^L)$. The state transition is indicated by directed bold edge between consecutive states. This state transition is considered as a *Markov* chain [1]. Moreover, their physical constraints are indicated by blue undirected edges among link states.

Information of an environmental object, for example, size, position, and status relative to a robot link, can be encoded in node $\mathbf{o}_t^k$. Here, k indicates the object index. The set of all environmental objects that interact with snake robot at time t, is represented by $\mathbf{O}_t = (\mathbf{o}_t^1, \ldots, \mathbf{o}_t^k \ldots, \mathbf{o}_t^{K_t})$. The number of involved objects is K_t, which would vary with the changing environment during the

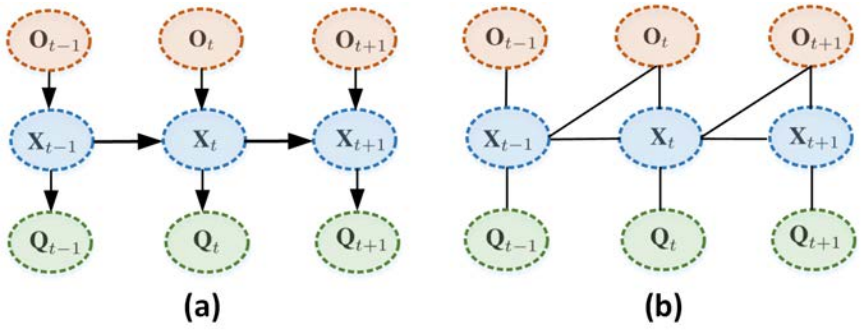

Figure 4.2. The probabilistic graphical model for Bayesian analysis. (a) is the clustered model, (b) the corresponding moralized graph based on *Separation Theorem* [1].

locomotion. $K_t = 0$ means that no object is in close approximity, and the robot would move by itself. The controller estimates K_t dynamically. If two objects interact with the same robot link, an undirected red edge in between will connect them. That is to say, if obstacles $\mathbf{o}_t^k$ and $\mathbf{o}_t^{k-1}$ are very close to the same link $\mathbf{x}_t^l$, they are deemed as being coupled. Such a relationship list for all the detected objects is recorded and updated by the controller as time goes by. The stimulus is denoted by a directed edge from an environmental object to a state. Interactions of this type is also varying dynamically. For instance, at time $t-1$, robot links l and $l+1$ interact with obstacle k, while at time t, only link l interacts with obstacle k.

State $\mathbf{x}_t^l$ generates observation $\mathbf{q}_t^l$, which could be decoded from LiDAR point cloud or camera images. The local observation likelihood is represented by the directed edge from a state $\mathbf{x}_t^l$ to the corresponding $\mathbf{q}_t^l$. Since sensors give stochastic noisy measurement and environmental interactions have disturbances, we can only partially observe links' state. This also accounts for why we formulate snake robot control with uncertainties as a stochastic problem with noisy observations.

The proposed model has the following differences in comparison with the existing approach [49]: i) a new tier which encodes hidden environmental information; ii) an innovative scheme to handle dynamically correlations among environmental objects; iii) a deeper investigation of link-level interactions between robot modules and environmental objects.

A two-step solution is adopted for further analysis: we derive a clustered Bayesian formulation for the whole robot; next, we use an innovative decoupling algorithm to deeply investigate various interactions.

4.1.2 Clustered Bayesian Formulation

It is problematic when conventional Bayes filters [25] [49] are directly applied for snake robot control, because interactions with the environment are not handled well, particularly on link level of snake robots. To obtain clear description of the proposed Bayesian framework, details of internal nodes and edges within each ellipse group can be temporarily ignored, and the model from Fig. 4.1 is converted to a clustered graph as shown in Fig. 4.2(a), in which edges connecting the same groups are merged. From the moralized graph of Fig. 4.2(b), the following conditional independence properties can be derived by applying the *Separation Theorem* [1]:

$$p(\mathbf{O}_{t+1}|\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t}) = p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t) \tag{4.1a}$$

$$p(\mathbf{Q}_t|\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t-1}) = p(\mathbf{Q}_t|\mathbf{X}_t) \tag{4.1b}$$

$$p(\mathbf{X}_{t+1}|\mathbf{O}_{0:t}, \mathbf{X}_{0:t}) = p(\mathbf{X}_{t+1}|\mathbf{X}_t) \tag{4.1c}$$

where eq. (4.1a) shows that the posterior density of environmental objects given its corresponding states is conditionally independent of other nodes based on *Markov* property from the moralized undirected graph [1] of Fig. 4.2(b). Eq. (4.1b) indicates that the observation $\mathbf{Q}_t$ is conditionally independent with all other nodes given corresponding state $\mathbf{X}_t$. In a similar way, when given $\mathbf{X}_t$ in the property (4.1c), $\mathbf{X}_{t+1}$ is conditionally independent with $\mathbf{O}_{0:t}$.

By dynamically predicting the posterior $p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$, an efficient controller is designed. We can decouple it with respect to interacted environmental objects $\mathbf{O}_{0:t+1}$,

$$p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) = \int p(\mathbf{O}_{0:t+1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) d\mathbf{O}_{0:t+1}. \tag{4.2}$$

where we apply marginalization [1]. We can further derive the joint posterior $p(\mathbf{O}_{0:t+1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$ by,

$$\begin{aligned} & p(\mathbf{O}_{0:t+1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ &= p(\mathbf{O}_{t+1}|\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t}) p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ &= p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t) p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \end{aligned} \tag{4.3}$$

where the property of eq. (4.1a)is exploited. Next the probability density $p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$ is

inferred as below,

$$\begin{aligned} & p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ = & \frac{p(\mathbf{Q}_t|\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t-1})p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}, \mathbf{Q}_{1:t-1})}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})p(\mathbf{Q}_{1:t-1})} \\ = & \frac{p(\mathbf{Q}_t|\mathbf{X}_t)}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})} p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t-1}) \end{aligned} \tag{4.4}$$

where the property eq. (4.1b)is applied. Moreover, we can infer the density $p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t-1})$ by Bayes rule as,

$$\begin{aligned} & p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t-1}) \\ = & p(\mathbf{X}_{t+1}|\mathbf{O}_{0:t}, \mathbf{X}_{0:t})p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) \\ = & p(\mathbf{X}_{t+1}|\mathbf{X}_t)p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) \end{aligned} \tag{4.5}$$

where the conditional density property eq. (4.1c) is applied. Thus, by firstly substituting eq. (4.5) to eq. (4.4) and then into eq. (4.3), we have

$$\begin{aligned} & p(\mathbf{O}_{0:t+1}, \mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ = & \frac{p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)p(\mathbf{Q}_t|\mathbf{X}_t)p(\mathbf{X}_{t+1}|\mathbf{X}_t)}{p(\mathbf{Q}_t|\mathbf{Q}_{1:t-1})} p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) && (4.6) \\ = & \frac{1}{c_t^c} p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)p(\mathbf{Q}_t|\mathbf{X}_t)p(\mathbf{X}_{t+1}|\mathbf{X}_t)p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}) && (4.7) \end{aligned}$$

where the denominator in eq. (4.6) is deemed as a normalization constant c_t^c , which is not related to the state $\mathbf{X}_t$. By substituting eq. (4.7) back into eq. (4.2), we further have,

$$\begin{aligned} & p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ = & \frac{1}{c_t^c} \int p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)p(\mathbf{Q}_t|\mathbf{X}_t)p(\mathbf{X}_{t+1}|\mathbf{X}_t)p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})d\mathbf{O}_{0:t+1} && (4.8) \\ = & \frac{1}{c_t^c} \int p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)d\mathbf{O}_{t+1} \cdot p(\mathbf{Q}_t|\mathbf{X}_t)p(\mathbf{X}_{t+1}|\mathbf{X}_t) \cdot \int p(\mathbf{O}_{0:t}, \mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})d\mathbf{O}_{0:t} && (4.9) \\ = & \frac{1}{c_t^c} p(\mathbf{Q}_t|\mathbf{X}_t)p(\mathbf{X}_{t+1}|\mathbf{X}_t) \cdot \underbrace{\int p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)d\mathbf{O}_{t+1}}_{\textit{Multi-Neural-Stimulus Function}} \cdot \underbrace{p(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})}_{\textit{Posterior Without Integral}}. && (4.10) \end{aligned}$$

where in eq. (4.9), we use the property that $\mathbf{O}_{t+1}$ is conditionally independent of $\mathbf{O}_{0:t}$ when given $\mathbf{X}_{t+1}$ and $\mathbf{X}_t$ as illustrated in the model of Fig. 4.2(b), which separate the integral.

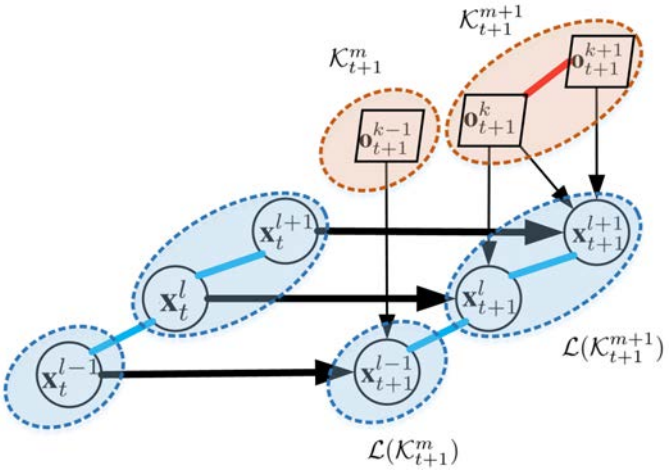

Figure 4.3. The decoupled graphical model for the analysis of interactions at time $t+1$. Interacted environmental objects and states have been grouped together.

Regarding the derived Bayesian filter in eq. (4.10), there are four densities: 1) $p(\mathbf{Q}_t|\mathbf{X}_t)$ is the observation likelihood; 2) the state transition is indicated by $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$; 3) the probability marginalization over $\mathbf{O}_{0:t}$ is used again to get a recursive formulation $p(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})$; 4) an integral with respect to $\mathbf{O}_{t+1}$ is introduced, which is a "*Multi-Neural-Stimulus Function*" $p(\cdot)_{\text{MNSF}}$. In this function, each link is modeled as a neuron and the interaction as stimulus. It also analyzes the accumulative effect of various interactions between the snake robot and environmental objects. First, a deep Bayesian analysis of $p(\cdot)_{\text{MNSF}}$ is given, and then the estimation details of all densities are discussed.

4.2 Decoupling of Interactions

To further lower the complexity of interactions, and study the working mechanism of "*multi-neural-stimulus function*", the graphic model Fig. 4.1 is decoupled with respect to the environmental objects $\mathbf{O}_{t+1}$ when given states $\mathbf{X}_t$ and $\mathbf{X}_{t+1}$. As illustrated in Fig. 4.3, we remove all other unrelated nodes and edges because of the conditional independence of $p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t)$ based on the *Separation Theorem* [1]. Furthermore, we group the correlated environmental objects, such as $\mathcal{K}_{t+1}^{m} = \{k-1\}$, $\mathcal{K}_{t+1}^{m+1} = \{k, k+1\}$, where m is the set index and $\mathcal{K}$ a set of object indices. We also cluster the corresponding states, for example, $\mathcal{L}(\mathcal{K}_{t+1}^{m}) = \{l-1\}$ and $\mathcal{L}(\mathcal{K}_{t+1}^{m+1}) = \{l, l+1\}$, in which $\mathcal{L}$ denotes a set of state indices. As stated above, this grouping dynamically varies with the robot

locomotion. For example, $K_{t+1} = 5$ if 5 obstacles are interacting with robot links at time $t+1$. And we have obstacle set number $M = 3$, and obstacle set $\mathcal{K}^1_{t+1} = \{1\}$, $\mathcal{K}^2_{t+1} = \{2\}$ and $\mathcal{K}^3_{t+1} = \{3,4,5\}$, if they can be divided into three groups $\{\mathbf{o}^1_{t+1}\}$, $\{\mathbf{o}^2_{t+1}\}$ and $\{\mathbf{o}^3_{t+1}, \mathbf{o}^4_{t+1}, \mathbf{o}^5_{t+1}\}$ based on the interaction with different robot links $\{\mathbf{x}^1_{t+1}\}$, $\{\mathbf{x}^3_{t+1}\}$ and $\{\mathbf{x}^4_{t+1}, \mathbf{x}^5_{t+1}, \mathbf{x}^6_{t+1}\}$. Therefore, $\mathcal{L}(\mathcal{K}^1_{t+1}) = \{1\}$, $\mathcal{L}(\mathcal{K}^2_{t+1}) = \{3\}$, $\mathcal{L}(\mathcal{K}^3_{t+1}) = \{4,5,6\}$. Based on the decoupled graphical model, we propose a two-step scheme to analyze *multi-neural-stimulus function*.

4.2.1 Decoupling with Respect to Interacted Environmental Objects

It is likely that a snake robot simultaneously interacts with multiple objects, some of which are independent while others may have correlations, such as in close proximity or connecting together. Hence it is desirable to decouple the function $p(\cdot)_{\text{MNSF}}$ with respect to each involved environmental object,

$$
\begin{aligned}
& p(\cdot)_{\text{MNSF}} \\
&= \int p(\mathbf{O}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t) d\mathbf{O}_{t+1} \\
&= \int \prod_{\mathcal{K}^m_{t+1} \in \bar{\mathcal{K}}_{t+1}} p(\mathbf{o}^{\mathcal{K}^m_{t+1}}_{t+1}|\mathbf{X}_{t+1}, \mathbf{X}_t) d\mathbf{O}_{t+1} && (4.11) \\
&= \prod_{\mathcal{K}^m_{t+1} \in \bar{\mathcal{K}}_{t+1}} \int p(\mathbf{o}^{\mathcal{K}^m_{t+1}}_{t+1}|\mathbf{x}^{\mathcal{L}(\mathcal{K}^m_{t+1})}_{t+1}, \mathbf{x}^{\mathcal{L}(\mathcal{K}^m_{t+1})}_{t}) d\mathbf{o}^{\mathcal{K}^m_{t+1}}_{t+1} && (4.12) \\
&\approx \prod_{\mathcal{K}^m_{t+1} \in \bar{\mathcal{K}}_{t+1}} \sum_{k=1}^{\mathcal{K}^m_{t+1}} \underbrace{p(\mathbf{o}^k_{t+1}|\mathbf{x}^{\mathcal{L}(k)}_{t+1}, \mathbf{x}^{\mathcal{L}(k)}_{t})}_{\textit{Decoupled Stimulus Density}} . && (4.13)
\end{aligned}
$$

where $\bar{\mathcal{K}}_{t+1} = \{\mathcal{K}^1_{t+1}, \ldots, \mathcal{K}^m_{t+1}, \ldots, \mathcal{K}^M_{t+1}\}$ is the set of all groups of environmental objects which currently interact with the snake robot at time $t+1$, $\mathbf{x}^{\mathcal{L}(\mathcal{K}^m_{t+1})}_{t+1}$ the states interacting with objects $\mathbf{o}^{\mathcal{K}^m_{t+1}}_{t+1}$, $\mathcal{L}(\mathcal{K}^m_{t+1})$ the set of indices of corresponding states, $\mathcal{L}(k)$ the set of indices of states having interactions with object k. Take Fig. 4.3 for an instance, $\mathcal{L}(k) = \{l, l+1\}$ since object $\mathbf{o}^k_{t+1}$ interacts with two states $\mathbf{x}^l_{t+1}$ and $\mathbf{x}^{l+1}_{t+1}$ simultaneously.

Through the derivations above, the interactions can be separated by a set of decoupled stimulus densities between each environmental object and its related states. And each of them can be further split with respect to robot links.

4.2.2 Decoupling with Respect to Robot Links

As for each environmental object, it may simultaneously interact with multiple robot links. Therefore, we desire to decouple the stimulus density for each interacted robot link,

$$
\begin{aligned}
& p(\mathbf{o}_{t+1}^{k} | \mathbf{x}_{t+1}^{\mathcal{L}(k)}, \mathbf{x}_{t}^{\mathcal{L}(k)}) \\
= \; & p(\mathbf{o}_{t+1}^{k} | \mathbf{v}_{t+1}^{\mathcal{L}(k)}) && (4.14) \\
= \; & \prod_{l'=0}^{L(k)-2} \overbrace{\frac{p(\mathbf{v}_{t+1}^{l'+1} | \mathbf{v}_{t+1}^{l'}, \mathbf{o}_{t+1}^{k})}{\underbrace{p(\mathbf{v}_{t+1}^{l'+1} | \mathbf{v}_{t+1}^{l'})}_{\textit{Link-Constraint}}}}^{\textit{Stimulus-Propagation}} \cdot \underbrace{p(\mathbf{o}_{t+1}^{k} | \mathbf{v}_{t+1}^{0})}_{\textit{Object-Link-Stimulus}} && (4.15)
\end{aligned}
$$

where l' the link index within $\mathcal{L}(k)$, $L(k)$ the total number of robot links in group $\mathcal{L}(k)$, and $\mathbf{v}_{t+1} = \{\mathbf{x}_{t+1}, \mathbf{x}_t\}$ is the state increment.

As shown in eq. (4.15), the stimulus density is finally decoupled by three probability densities: 1) the initial interaction between the environmental object $\mathbf{o}_{t+1}^{k}$ and state increment $\mathbf{v}_{t+1}^{0}$, which is modeled by the "*object-link-stimulus*" $p(\mathbf{o}_{t+1}^{k} | \mathbf{v}_{t+1}^{0})$; 2)the inherent correlation between adjacent snake robot links l and $l+1$ is denoted by the "*link-constraint*" $p(\mathbf{v}_{t+1}^{l'+1} | \mathbf{v}_{t+1}^{l'})$; 3) the signal transmission along the body of the snake robot is simulated by the "*stimulus-propagation*" $p(\mathbf{v}_{t+1}^{l'+1} | \mathbf{v}_{t+1}^{l'}, \mathbf{o}_{t+1}^{k})$. This decoupling mechanism of two steps, is essentially different with Bayes filters method [49], bacause the proposed *multi-neural-stimulus function* models the interaction between the robot and the environment, as well as constraints within the snake robot. It only considers the currently involved interaction, and makes the control strategy more robust and efficient by effectively narrowing down the state space.

4.2.3 Simulation with Sequential Monte Carlo Method

It is difficult to calculate the posterior densities inferred above without any approximation, because they usually do not have analytical forms and are usually non-Gaussian owing to the inherent uncertainties. Variation Inference, Gaussian Mixture model, kernel density estimation [54] and other density approximation approaches can be utilized in our framework. As a paradigm, the Sequential Monte Carlo (MC) method [55] is adopted in our implementation. Particularly,

we use a weighted sample set $\{\mathbf{X}^n_{0:t+1}, w^n_{t+1}\}^{N_s}_{n=1}$ to estimate the posterior $p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$. The samples are $\{\mathbf{X}^n_{0:t+1}, n = 1, \ldots, N_s\}$, the associated normalized weights are $\{w^n_{t+1}, n = 1, \ldots, N_s\}$, and $\sum_n w^n_{t+1} = 1$.

$\{\mathbf{X}^n_{0:t+1}, w^n_{t+1}\}^{N_s}_{n=1}$ is defined to approximate the posterior $p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$, in which the samples are $\{\mathbf{X}^n_{0:t+1}, n = 1, \ldots, N_s\}$, the associated normalized weights are $\{w^n_{t+1}, n = 1, \ldots, N_s\}$, and $\sum_n w^n_{t+1} = 1$. $\mathbf{X}^n_{0:t+1}$ can be sampled from a functional density $f(\cdot)$ with associated weights by using the MC theory [55],

$$w^n_{t+1} = c^w_t p(\mathbf{X}^n_{0:t+1}|\mathbf{Q}_{1:t})/f(\cdot). \tag{4.16}$$

where c^w_t is a constant. For the sequential case, the function $f(\cdot)$ is chosen to factorize by

$$\begin{aligned} & f(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t}) \\ = & f(\mathbf{X}_{t+1}|\mathbf{X}_{0:t}, \mathbf{Q}_{1:t}) f(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t}) \\ = & f(\mathbf{X}_{t+1}|\mathbf{X}_t) f(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1}). \end{aligned} \tag{4.17}$$

where we apply the conditional independence properties $f(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t}) = f(\mathbf{X}_{0:t}|\mathbf{Q}_{1:t-1})$, and $f(\mathbf{X}_{t+1}|\mathbf{X}_{0:t}, \mathbf{Q}_{1:t})$
$= f(\mathbf{X}_{t+1}|\mathbf{X}_t)$ from Fig. 4.2. We then substitute eq. (4.10) and eq. (4.17) into eq. (4.16), and get

$$w^n_{t+1} = w^n_t \frac{c^c_t p(\mathbf{Q}_t|\mathbf{X}^n_t) p(\mathbf{X}^n_{t+1}|\mathbf{X}^n_t)}{c^w_t f(\mathbf{X}^n_{t+1}|\mathbf{X}^n_t)} p(\cdot)_{\text{MNSF}}. \tag{4.18}$$

where

$$p(\cdot)_{\text{MNSF}} = \underbrace{\prod_{\mathcal{K}^m_{t+1} \in \tilde{\mathcal{K}}_{t+1}} \sum_{k=1}^{\mathcal{K}^m_{t+1}} \prod_{l'=0}^{L(k)-2}}_{\textit{Deep Decoupling}} \underbrace{\overbrace{\frac{p(\mathbf{v}^{l'+1,n}_{t+1}|\mathbf{v}^{l',n}_{t+1}, \mathbf{o}^k_{t+1})}{p(\mathbf{v}^{l'+1,n}_{t+1}|\mathbf{v}^{l',n}_{t+1})}}^{\textit{Stimulus-Propagation}}}_{\textit{Link-Constraint}} \cdot \underbrace{p(\mathbf{o}^k_{t+1}|\mathbf{v}^{0,n}_{t+1})}_{\textit{Object-Link-Stimulus}}. \tag{4.19}$$

We can deeply decouple the interaction $p(\cdot)_{\text{MNSF}}$ with respect to interacted environmental objects as well as robot links. We will discuss how to estimate the densities in eq. (4.18) as below.

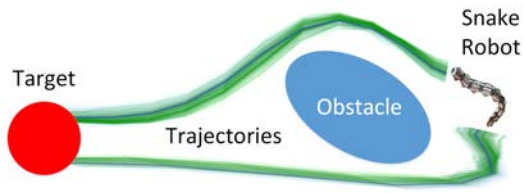

Figure 4.4. Training scene of the BNN-based stimulus density estimation. 50000 samples were generated from 25 trials.

4.2.4 Learning-Based Density Estimation

To model the effect that the snake robot experiences during locomotion, we use a neuron model motivated by *Neuroscience* [56]. We regard each robot link as a neuron, which can sense environmental conditions, produce proper stimulus, and propagate to its neighbor with inherent constraints. Please note that this multi-neural-stimulus function is the weighting density for state samples in implementation, but not the control signal. We can learn these densities from training data by neural network methods. In our experiments, the Bayesian Neural Network (BNN) [30] is adopted as a paradigm. The BNN represents model uncertainty with the use of a distribution over the weights of the NN. The densities of a BNN is usually intractable. Therefore, an approximating solution is to use variational inference, in which a distribution is found in a tractable family which minimizes Kullback-Leibler (KL) divergence to the true density. Particularly, the dropout scheme [30] is used as a variational Bayesian approximation. We learn the distributions from about 50000 samples from 25 trials by using a snake robot as illustrated in Fig. 4.4.

Object-Link-Stimulus

The modeling complexity of $p(\mathbf{o}_{t+1}^{k}|\mathbf{v}_{t+1}^{0})$ relies on different factors including the property of terrain scenes, environmental objects, link motion patterns etc. We aim to find an estimate $\hat{p}(\mathbf{o}|\mathbf{v})$ of the true conditional density $p(\mathbf{o}|\mathbf{v})$ given a dataset of observation $\mathcal{D} = \{(\mathbf{o}_n, \mathbf{v}_n)\}_{n=1}^{N}$ drawn from the joint distribution $(\mathbf{o}_n, \mathbf{v}_n) \sim p(\mathbf{o}, \mathbf{v})$. The KL divergence objective is formulated as expectation over

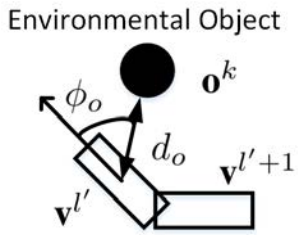

Figure 4.5. Compute the relative posture of a robot link with an interacted object in environment.

$p(\mathbf{v})$,

$$\begin{aligned} &\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}[\mathcal{D}_{KL}(p(\mathbf{o}|\mathbf{v})\|\hat{p}(\mathbf{o}|\mathbf{v}))] \\ &= \mathbb{E}_{(\mathbf{v},\mathbf{o})\sim p(\mathbf{v},\mathbf{o})}[\log p(\mathbf{o}|\mathbf{v}) - \log\hat{p}(\mathbf{o}|\mathbf{v})] \end{aligned} \tag{4.20}$$

The neural network weights $\mathbf{w}_o$ are calculated by fitting through maximum likelihood estimation,

$$\mathbf{w}_o^* = \underset{\mathbf{w}_o}{\operatorname{argmax}} \sum_{n=1}^{N} \log\hat{p}_{\mathbf{w}_o}(\mathbf{o}|\mathbf{v}) \tag{4.21}$$

The input of BNN is chosen as $\langle\phi_o, d_o\rangle$, describing the relative posture between a robot link and an environmental object as illustrated in Fig. 4.5, where d_o the normalized distance from robot link to the object, ϕ_o is the angle between the link $\mathbf{v}^{l'}$ and object $\mathbf{o}^k$. In this way, the information of both $\mathbf{v}^{l'}$ and $\mathbf{o}^k$ is implicitly encoded in this input definition. The output of BNN is defined as a normalized summation of trajectories in a predefined neighbor to indicate the possibility of a particular position of the snake robot in the state space. Regarding the structure, the NN consists of two fully connected layers with hidden units number 256. The uncertainty in parameters of the NN induces prediction uncertainty by marginalizing over the estimated probability density utilizing Monte Carlo integration.

Link-Constraint of the Snake Robot

Estimation of $p(\mathbf{v}_{t+1}^{l'+1}|\mathbf{v}_{t+1}^{l'})$ should adapt to various applications, which models the inherent correlation between the analyzed link l' and its neighbor $l'+1$. Various methods [57] [58] [59] have been reported to incorporate different constraint information. Nonetheless, these kinematic analysis has high computational requirements with the number of links and may fail in crowded scenario

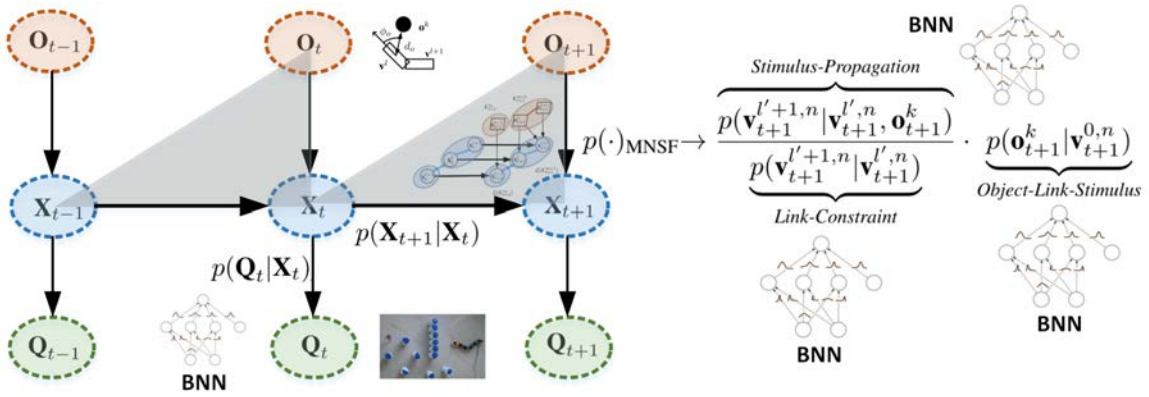

Figure 4.6. The graph of algorithm implementation. The gray area indicates the *multi-neural-stimulus function* $p(\cdot)_{\text{MNSF}}$, whose densities can be learned by BNNs. The objective of control is to recursively predict the state $\mathbf{X}_{t+1}$ based on historical information. Measurement $\mathbf{Q}_t$ can be decoded from sensor inputs such as camera image or LiDAR point cloud.

with many obstacles, due to the uncertainty inherent in the robot and the surroundings. Moveover, we introduce a "*shape-based spring-joint*" model by learning using BNN from the training data. The input of NN is chosen as $\mathbf{v}^{l'}$ with the output $\mathbf{v}^{l'+1}$. It consists of two fully connected layers with hidden units number 256. The neural network weights $\mathbf{w}_v$ are fitted through maximum likelihood estimation,

$$\mathbf{w}_v^* = \underset{\mathbf{w}_v}{\text{argmax}} \sum_{n=1}^{N} \log \hat{p}_{\mathbf{w}_v}(\mathbf{v}^{l'+1}|\mathbf{v}^{l'}). \tag{4.22}$$

Stimulus-Propagation

The effect of a stimulus signal passing along adjacent snake robot links when facing an environmental object is modeled by the density $p(\mathbf{v}_{t+1}^{l'+1}|\mathbf{v}_{t+1}^{l'}, \mathbf{o}_{t+1}^{k})$, which can be learned from training data by using Bayesian Neural Network. $\langle \phi_o, d_o \rangle$ is selected as input, $\mathbf{v}^{l'+1}$ the output. The network parameters $\mathbf{w}_p$ are fitted via maximum likelihood estimation,

$$\mathbf{w}_p^* = \underset{\mathbf{w}_p}{\text{argmax}} \sum_{n=1}^{N} \log \hat{p}_{\mathbf{w}_p}(\mathbf{v}^{l'+1}|\mathbf{v}^{l'}, \mathbf{o}_{t+1}^{k}). \tag{4.23}$$

Measurement Likelihood

The density $p(\mathbf{Q}_t|\mathbf{X}_t)$ can be calculated as follows, by using the *Markov* property from the graphical model of Fig. 4.1, in which the link observations $\mathbf{q}_t$ are conditionally independent if the corresponding states $\mathbf{x}_t$ are given,

$$p(\mathbf{Q}_t|\mathbf{X}_t) = \prod_{l=1}^{L} p(\mathbf{q}_t^l|\mathbf{x}_t^l), \tag{4.24}$$

The observation likelihood of body link l, is denoted by $p(\mathbf{q}_t^l|\mathbf{x}_t^l)$, which encodes observation uncertainties due to noisy observations, and serves as a feedback for the controller. Regarding the BNN training process, the input uses camera images for the observation $\mathbf{q}_t^l$ with the output of corresponding state $\mathbf{x}_t^l$. Convolutional layers encode sensor images into features of size 256, and followed by two fully connected layers with hidden units number 256. The network parameters $\mathbf{w}_c$ are fitted via maximum likelihood estimation,

$$\mathbf{w}_c^* = \underset{\mathbf{w}_c}{\operatorname{argmax}} \sum_{n=1}^{N} \log \hat{p}_{\mathbf{w}_c}(\mathbf{q}_t^l|\mathbf{x}_t^l). \tag{4.25}$$

State Transition and Importance Function

The same choice of state transition and importance function as the conventional Bayes filter [49] is used to have a better comparison. Given no prior knowledge, the state transition $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$ has been chosen as a random walk model. The efficiency of a sequential Monte Carlo based method highly depends on the sampling function $f(\cdot)$. If $f(\cdot)$ is in close proximity to the true posterior, the samples are more effective. A natural choice of this function is the state dynamics $p(\mathbf{X}_{t+1}|\mathbf{X}_t)$, which is accepted in our implementation.

Algorithm Implementation

The implementation process is illustrated by Fig. 4.6. The gray area indicates the *multi-neural-stimulus function* $p(\cdot)_{\text{MNSF}}$, whose densities are learned by Bayesian Neural Networks as we stated

Table 4.1. Algorithm Paradigm of Decoupled Bayesian Control

- Sampling $\mathbf{X}_{t+1}^n, n = 1, \ldots, N_s$, from $f(\mathbf{X}_{t+1}|\mathbf{X}_t)$
- Initial Weighting $w_{t+1}^n \sim \mathbf{X}_{t+1}^n$ by equation (4.18).
- Initial Prediction, $\hat{\mathbf{X}}_{t+1}=\sum_{n=1}^{N_s} w_{t+1}^n \cdot \mathbf{X}_{t+1}^n$
- Loop $r = 1 : R$ # Re-sampling Scheme
  - Interaction Weighting,

    $p_1(\cdot)$ by equation (4.19)
  - Measurement Likelihood Weighting,

    $p_2(\cdot) = p(\mathbf{Q}_t|\mathbf{X}_t^n)$ by equation (4.24)
  - Updating Weights, $w_{t+1}^n \approx w_{t+1}^n \cdot p_1(\cdot) \cdot p_2(\cdot)$
  - Weight Normalization
  - Prediction Updating, $\hat{\mathbf{X}}_{t+1}=\sum_{n=1}^{N_s} w_{t+1}^n \cdot \mathbf{X}_{t+1}^n$
- End Loop r
- Compute the control signal θ for actuators by serpenoid curve with shape parameters in the predicted state $\hat{\mathbf{x}}_{t+1}$.

above. We aim to recursively predict the state $\mathbf{X}_{t+1}$ based on historical information. This can be achieved by estimating the posterior $p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$ as shown in eq.(4.10) . The only input for the controller is observations $\mathbf{Q}_{1:t}$, which can be estimated from sensor signals such as camera images or LiDAR point cloud. The environmental object tier $\mathbf{O}_{1:t}$ encodes the complexity of surroundings and the relative status with the snake robot, which can be decoded from sensor observations too. The correlation graph inside the gray area, as shown in Fig. 4.3, is varying and dynamically approximated by the controller during robot locomotion so as to calculate the function $p(\cdot)_{\text{MNSF}}$. When the posterior $p(\mathbf{X}_{0:t+1}|\mathbf{Q}_{1:t})$ is computed, the joint angles can be calculated and sent to actuators for proper gaits generation.

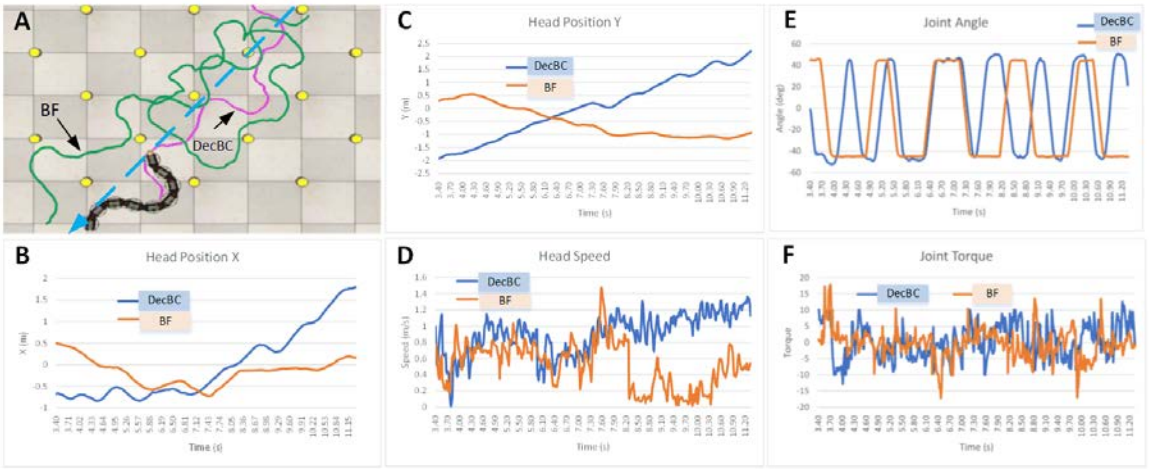

Figure 4.7. Comparisons of simulation results for obstacle avoidance using the proposed DecBC and the conventional BF. A is the overall the navigation trajectories, B and C the coordinate of the head position with respect to time, D the robot head speed, E and F the joint angle and torque, respectively.

Pseudo-code of the proposed algorithm for predicting $\mathbf{X}_{t+1}$ at time t is presented in table 4.1, which is recursively called during the whole process. We use a re-sampling mechanism to achieve stable performance and alleviate the degeneracy commonly seen in MC methods. The number of re-sampling times R is selected as 3 in our experiments. The total number of samples, N_s, ranges from 200 to 500. Samples are updated by different weights, where various interactions and the robot's own dynamics are considered. And then we calculate the state prediction by the weighted mean. Next, the control signal of each robot joint θ_t^l would be calculated via the serpenoid curve and sent to corresponding actuators.

4.3 Experimental Results

We conducted simulation and real world experiments to show the performance of the Decoupled Bayesian-based Controller (DecBC) in comparison with the state-of-the-art.

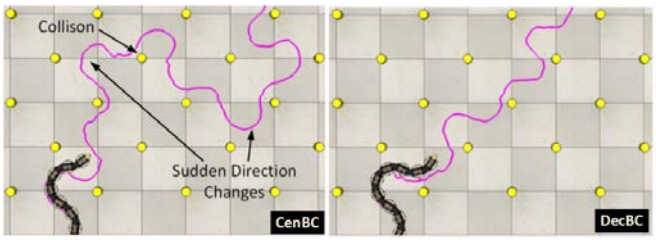

Figure 4.8. Comparison of using CenBC [2] and DecBC, where DecBC achieved more efficient result with a shorter trace.

4.3.1 Simulation Analysis

Simulations were performed by using ACM-R5 snake robot in platform V-REP. Performance of DecBC were first compared with the conventional BF [49]. In the original BF, only the head link was directly controlled by it while the rest links follow this gait, since the original BF was not designed for snake robots. To obtain a fair comparison, both the methods used the same state transition model and observation likelihood functions. In 4.7, the results of two methods are illustated, where the robot moves from the top-right to bottom-left as shown by the blue dashed arrow.

The overall movement of DecBC and BF are illustrated by pink and green trajectories in Fig. 4.7, which shows that BF suffers from "*unexpected collision*" problem frequently due to lacking enough scheme to deal with crowded obstacles. Nevertheless, with the equipment of the decoupled interaction mechanism, the robot using DecBC effectively swings through the peg array. The corresponding coordinates of head position are shown in Fig. 4.7B and 4.7C. As shown in In Fig. 4.7D, the head speed of BF remarkably reduces when the robot gets stuck due to collisions while DecBC does not. The variation of one joint's angle and torque are shown in Fig. 4.7E and 4.7F. As shown in Fig. 4.7E, by using either methods, the snake robot can keep the serpenoid gaits well. Whereas, BF undergoes wider delay because of frequent unexpected collisions. The same problem is reflected by the overshoots of torque in Fig. 4.7F, in which DecBC has a performance far superior to BF.

To demonstrate the effectiveness of the proposed DecBC framework, the performance of DecBC and Centralized Bayesian-based Controller (CenBC) [2] by using the same number of samples, are further compared. The resulted trajectories on the testing `ARRAY` scenario are presented in Fig. 4.8. Both methods can reach the final target successfully, but trajectory length, or the control

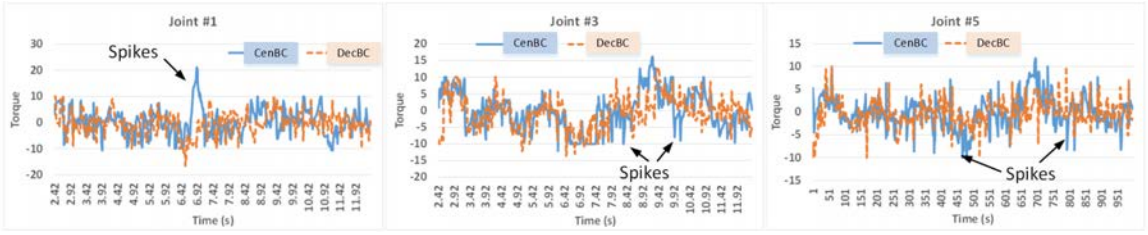

Figure 4.9. Measured torque of actuators No. 1, 3, 5 using CenBC [2] and DecBC where CenBC has more spikes due to collisions or sharp changes of directions. DecBC achieved much less jerky curves.

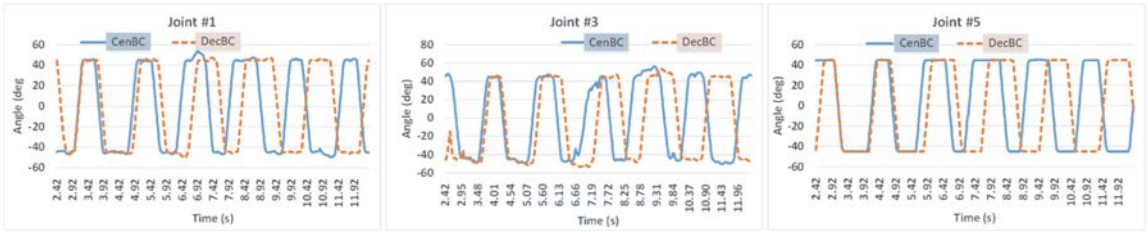

Figure 4.10. Measured joint angles of actuators No. 1, 3, 5 using CenBC [2] and DecBC where DecBC got smoother curves than CenBC.

efficiency is quite different. DecBC is apparently more efficient with a shorter trajectory. The robot of using DecBC achieved more robust results, while the CenBC experienced a few collisions during the journey. This result verifies that the DecBC framework is more efficient than the centralized framework in that it needs less samples to hand the interaction densities. To analyze the motor power consumption, the variation of actuator torque is investigated, because they are correlated in a positive manner. Torque values can be obtained from the platform V-REP. The measured torque of joint actuators No.1, 3, 5 using CenBC [2] and DecBC, are shown in Fig. 4.9. DecBC has a much less jerky curve with fewer power consumption, while the overall consumed energy of CenBC is higher with more spikes due to collisions or sharp changes of directions. The actual joint angles of actuators No.1, 3, 5 using CenBC [2] and DecBC are shown in Fig. 4.10. Both the methods obtained similar periodic performance due to using the shape-based model. Nonetheless, DecBC obtains more robust results with a smoother curve than CenBC.

As far as we know, because of the high DOFs and the challenging obstacle interaction problem, little study has been done using sequential Monte Carlo methods directly for snake robot control. To have a further comparison with the state-of-the-art, two more related approaches Particle Filter

Table 4.2. Comparison of performance and computational complexity

Method	Optimal Sample No.	Spent Cycles	Unexpected Collisions	Computational Complexity
BF	2000	53	18	$O(N_s)$
PFC	1000	37	16	$O(N_s K)^{\dagger}$
BSC	300	33	14	$O(N_s K_p L)^{*}$
CenBC	300	31	8	$O(N_s K L)^{\dagger}$
DecBC	300	27	3	$O(N_s \mathcal{K} \mathcal{L})^{\ddagger}$

* $K_p = n_T \frac{2\pi}{\omega_0 \Delta t}$, where n_T is the number of periods in each episode of Q-learning, Δt the discrete time step-size, ω_0 input torque frequency [60].

† K is the max number of detected environmental objects which is much less than K_p, L the total number of links of the snake robot.

‡ $\mathcal{K}$ is the number of interacted environmental objects, which is much less than K; $\mathcal{L}$ is the number of interacted links, which is usually less than L too.

based Controller (PFC) [61] and Bio-inspired-learning of Sensorimotor Control (BSC) [60] were chosen. PFC was implemented for the head link of the snake robot similar as BF, although it was not originally developed for snake robot control. Comparison of the average quantitative performance as well as the computational complexity are given in Table 8.2. We ran the experiments for the five methods by ten tests respectively, on similar setup of Fig. 4.7. The computer is equipped with Intel i7-1065G7 CPU and 16GB RAM. We study the resampling algorithms for complexity comparison. In this table, BF has the least computational complexity theoretically but needs a much more samples to achieve relatively stable results because it lacks a sufficient scheme to deal with obstacle collision problems. As we can see, PFC has additional computational cost on solving obstacle detection and avoidance, though it needs much less samples. Collisions occurred mostly by the rest robot links because there is no particular interaction handling scheme except the head link. The other three methods introduce more complexity to deal with the environmental interaction for the body links of snake robot. Particularly, CenBC is less complex than BSC since K_p is usually much larger than K, for instance, $K_p = 3140$ in [60] where K is normally less than 100. DecBC has the least computational complexity among these three centralized approaches because it only considers the interacted obstacles and robot links when necessary due to the decoupled framework. For a fair comparison of performance, we applied the same number of samples for these three methods. As we can see, equipped with the proposed decoupled formulation and interaction model in eq. (4.19), DecBC achieved the most robust and efficient performance in terms of less cycles and unexpected collisions to reach the same target than the other two approaches.

4.3.2 Real-World Data

We used the snake robot *JAW*-I for implementation. The performance of using DecBC on a real-world test is presented in Fig. 4.11. As we can see, one target and more than twenty obstacles are randomly set in the scene. It is very challenging due to the presence of movable targets and obstacles. The DecBC method achieved a robust performance, smoothly traversing through peg array and successfully reaching the target. Though we adopted a centralized state space, the snake robot's shape was not changed as a whole. Instead, external objects can stimulate different body links. The

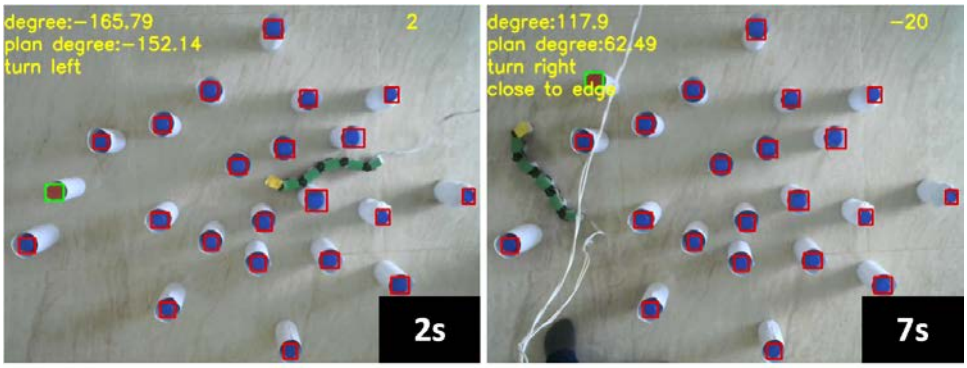

Figure 4.11. Results of the proposed DecBC for unstructured RANDOM scenario with a shiftable target.

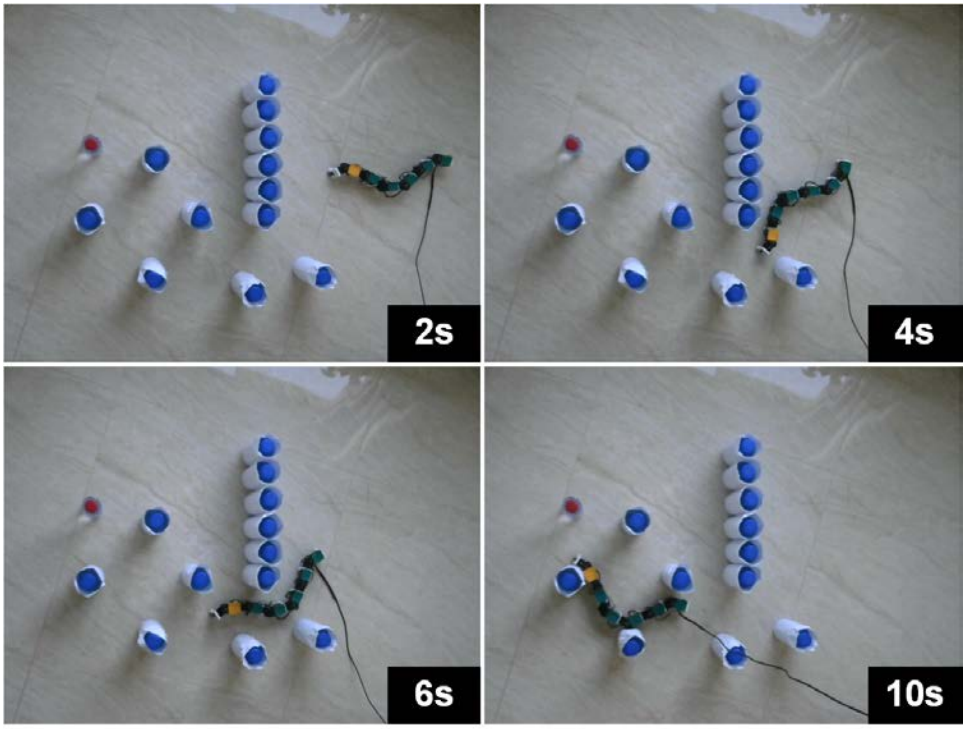

Figure 4.12. Performance of DecBC on testing scene WALL−LIKE with various obstacles and a sharp turn.

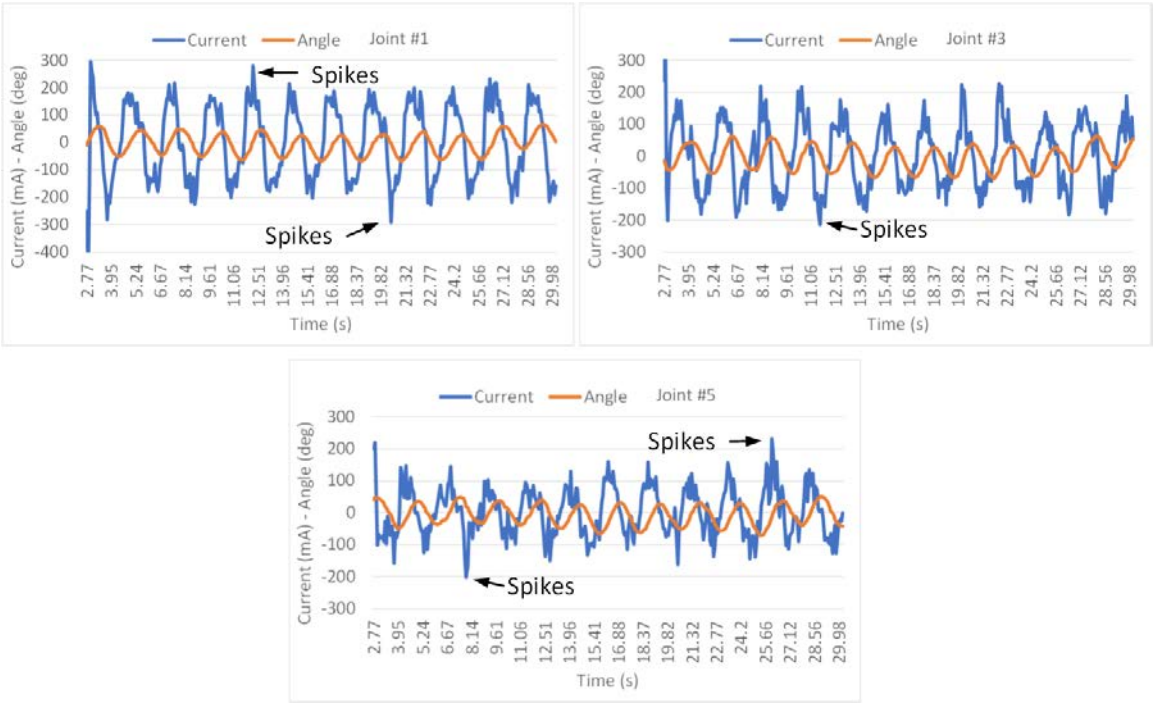

Figure 4.13. Measured results of joint actuators No. 1, 3, 5 on the real world data RANDOM.

snake robot can respond to shifting obstacles and targets quickly by keeping multi-hypotheses of both motion and shape variations. Fig. 4.13 shows the results of measured joint current and angle of actuator No. 1, 3, and 5. The power consumption in motors by the current is investigated because they are positively correlated. As we can see, even with the uncertainties of environmental interactions during the locomotion, the proposed controller present relatively smoothing shape variation. Some slight overshoots of current can be observed in Fig. 4.13, which are introduced by the collisions or direction changes after obstacle interactions.

The obstacles in real-world scenes are very complicated. We further studied the performance of DecBC with more scenarios. For instance, a situation with WALL-LIKE obstacles and a sharp turn on the way to the target is shown in Fig. 4.12. DecBC could achieve a successful performance with these challenges as shown in Fig. 4.12, by benefiting from the proposed decoupled Bayesian framework which models various interactions.

4.4 Summary

In this chapter, we have presented a decoupled Bayesian framework for snake robot control in dynamically changing surroundings. The interactions between the robot and environment are simulated by a Bayesian dynamic graphical model. Benefiting from the decoupling mechanism, the proposed probabilistic propagation formulation provides an innovative way to model the uncertainty during locomotion in cluttered terrain. The proposed "*multi-neural-stimulus function*" represents the cumulative effect of both external environmental influences and internal constraints of the snake robot. It implicitly handles the "*unexpected collision*" problem and thus solve the difficult data association and shape adjustment problems for snake robot control in an innovative way. Preliminary experimental results have demonstrated promising performances of the DecBC approach for different unstructured scenarios.

Chapter 5

Interactively Distributed Bayesian Control

The centralized representation discussed in the previous two chapters has inherent difficulty when the body length of snake robot is getting longer or the computational resource is limited on the mobile platform. In these cases, a decentralized solution is desirable where the snake robot is regarded as a multi-agent system. In this chapter, we focus on how to design distributed intelligence to achieve better predictability and more prompt control performance for snake robot control.

The advantage of centralized snake robot control methods is that they can keep the snake robot gait effectively in different aspects [2]. Nevertheless, most of them have limited environmental adaptability or a large requirement of computation cost, which makes them difficult for many practical applications. To further reduce the complexity, researchers are interested in various decentralized control solutions for snake robot control in cluttered environments [45]. Central Pattern Generator (CPG) [62] has been a hot research area and received a lot of attentions. In order to simulate the biologic interaction strategy of snakes with obstacles and exploit machine learning techniques, researchers considered neural network based methods [63] [64]. Sartoretti et al. [4] take this idea a step further, using the A3C algorithm by learned agents. It does not yet offer an explicit mathematical description of the interaction with the world, though this technique points to a promising direction. Compared with the centralized approaches, the reported decentralized meth-

ods have presented a promising way to improve the control performance. However, many additional challenges have come up too. For instance, both physical constraints among snake robot links and the various frequent interactions with environment still need to be well modeled. Without effective schemes, most existing methods still suffer from error collision problems, where the snake robot may be easily stuck by obstacles. Furthermore, many learning-based approaches have limited adaptation to dynamical environment. When the variation is beyond the training data, they may fail due to the confined learning experience.

In this chapter, we propose a fully distributed framework by extending our previous Bayesian methods. Compared with the previous centralized method, it can greatly reduce the complexity by parallel computation. Different with other approaches, it provides a new perspective to model the various interactions among snake robot links and environmental objects.

5.1 Methodology of the Distributed Framework

A graphical design is firstly presented for the distributed snake robot control. Then, we give a strict mathematical derivation for the whole stochastic process. Finally, the details of probability density approximation are discussed.

5.1.1 Bayesian Network Modeling

Dynamic Bayesian network [1] is a useful tool to analyze the complicated problems such as snake robot control. In particular, we present an example for three joints with three time steps in Fig. 5.1, which has three levels: the hidden state level (yellow), the observation level (green), and the environmental input level (pink). The circle nodes represent snake robot links while the square nodes indicate their observations. Four different kinds of edges are exploited: 1) the undirected ones show the constraints among snake robot links; 2) the directed ones from a link state to its observation represent the local likelihood; 3) the directed edges from an environmental input to a state represents the interactions such as collisions; 4) the directed edges among link state show the robot dynamics, which is assumed as a Markov chain. $\mathbf{x}_t^i$, where $i = 1, \ldots, L$, denotes the link state

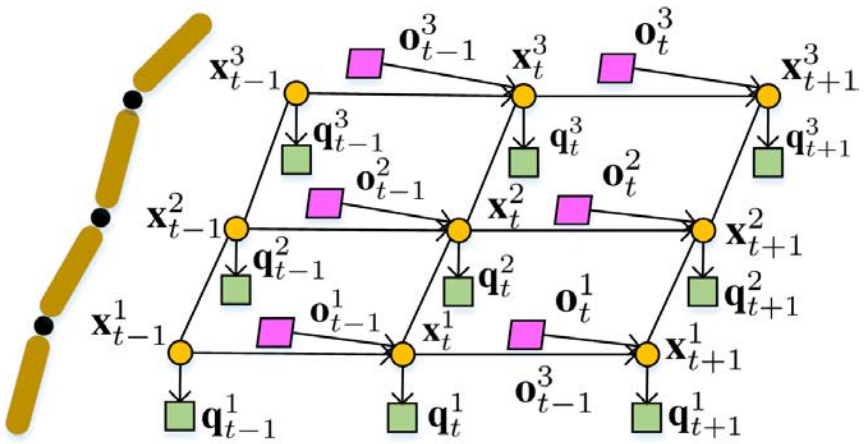

Figure 5.1. Bayesian network modeling for a snake-like robot with three joints.

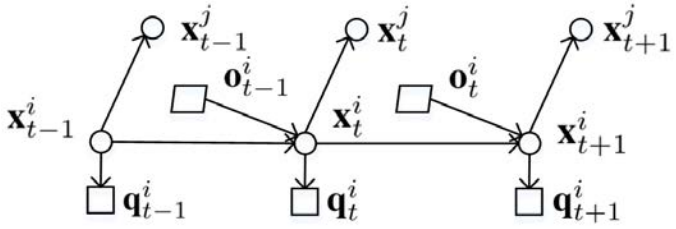

Figure 5.2. Dynamical distributed model decomposition for a link of snake robot.

for link i, where L is the total number of all snake robot links. The corresponding observation is $\mathbf{q}^i_t$, which can be estimated by a learning based video detector. Specifically, we define the state as a two tuple by $\mathbf{x}^i_t = (A^i_t, \gamma^i_t)$, where A^i_t and γ^i_t are the amplitude and offset in the serpenoid curve [5]. Furthermore, we denote the group of all environmental inputs to time t by $\mathbf{o}_{1:t}$.

The graphical model in Fig. 5.1 has too many couplings and thus is difficult for a direct analysis. By applying graphical decomposition rule [65], we can split the model with respect to each robot link as shown in Fig. 5.2. Such kind of decomposition is meaningful since all information is kept when we process all links in parallel. In the decomposed graphical model such as Fig. 5.2 for link i, all other nodes and links are ignored temporarily except the directed edge and nodes of link i. $\mathcal{A}(i)$ denotes the set of all adjacent links of i, where $j \in \mathcal{A}(i)$ is used to show an example. Correspondingly, the set of observations is denoted as $\mathbf{q}^{\mathcal{A}(i)}$.

5.1.2 Bayesian Sequential Updating Rule

The dynamic graphical model designed above endows good flexibility for sophisticated probability density analysis comparing with traditional Kalman filters. Since we accept a decentralized control scheme, one controller for each link, the posterior $p(\mathbf{o}^i_{0:t}, \mathbf{x}^i_{0:t+1}|\mathbf{q}^i_{1:t})$ is desirable to be recursively estimated based on the evolutional history of various uncertainties:

$$
\begin{aligned}
& p(\mathbf{o}^i_{0:t}, \mathbf{x}^i_{0:t+1}|\mathbf{q}^i_{1:t}, \mathbf{q}^{\mathcal{A}(i)}_{1:t}) \\
&= \frac{p(\mathbf{o}^i_t|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t}, \mathbf{q}^{\mathcal{A}(i)}_{1:t})}{p(\mathbf{q}^i_{1:t}, \mathbf{q}^{\mathcal{A}(i)}_{1:t})} p(\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t}, \mathbf{q}^{\mathcal{A}(i)}_{1:t}) && (5.1) \\
&= \frac{p(\mathbf{o}^i_t|\mathbf{x}^i_{t+1}, \mathbf{x}^i_t) p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1})}{p(\mathbf{q}^i_{1:t}, |\mathbf{q}^{\mathcal{A}(i)}_{1:t})} p(\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}) && (5.2) \\
&= \frac{p(\mathbf{o}^i_t|\mathbf{x}^i_{t+1}, \mathbf{x}^i_t) p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t)}{p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1})} p(\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}|\mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}) && (5.3) \\
&= \frac{p(\mathbf{o}^i_t|\mathbf{x}^i_{t+1}, \mathbf{x}^i_t) p(\mathbf{x}^i_{t+1}|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t}, \mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1})}{p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1})} p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t) p(\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t}, |\mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}) && (5.4) \\
&= c_t \cdot p(\mathbf{o}^i_t|\mathbf{x}^i_{t+1}, \mathbf{x}^i_t) p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t) p(\mathbf{x}^i_{t+1}|\mathbf{x}^i_t) p(\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t}|\mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}). && (5.5)
\end{aligned}
$$

where we use the Bayes rule in eq. (5.1). Two Markov properties $p(\mathbf{o}^i_t|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t}, \mathbf{q}^{\mathcal{A}(i)}_{1:t}) = p(\mathbf{o}^i_t|\mathbf{x}^i_{t+1}, \mathbf{x}^i_t)$ and $p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t+1}, \mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}) = p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t)$ are applied in eq. (5.2) and eq. (5.3), respectively, which can be easily verified from the decomposed graphical model in Fig. 5.2. A constant c_t is used to replace the denominator of (5.4) because there is no state $\mathbf{x}^i$ in it. In eq. (5.5), another Markov property that $p(\mathbf{x}^i_{t+1}|\mathbf{o}^i_{0:t-1}, \mathbf{x}^i_{0:t}, \mathbf{q}^i_{1:t-1}, \mathbf{q}^{\mathcal{A}(i)}_{1:t-1}) = p(\mathbf{x}^i_{t+1}|\mathbf{x}^i_t)$ is used. The density $p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t)$ in eq. (5.5) can be viewed as a likelihood function for all related observations including $\mathbf{q}^i_t$ and $\mathbf{q}^{\mathcal{A}(i)}_t$. We give a further derivation for it as follows,

$$
\begin{aligned}
& p(\mathbf{q}^i_t, \mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t) \\
&= p(\mathbf{q}^i_t|\mathbf{x}^i_t) p(\mathbf{q}^{\mathcal{A}(i)}_t|\mathbf{x}^i_t) && (5.6) \\
&= p(\mathbf{q}^i_t|\mathbf{x}^i_t) \prod_{j \in \mathcal{A}(i)} p(\mathbf{q}^j_t|\mathbf{x}^i_t) && (5.7) \\
&= p(\mathbf{q}^i_t|\mathbf{x}^i_t) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}^j_t|\mathbf{x}^i_t, \mathbf{x}^j_t) p(\mathbf{x}^j_t|\mathbf{x}^i_t) d\mathbf{x}^j_t && (5.8) \\
&= p(\mathbf{q}^i_t|\mathbf{x}^i_t) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}^j_t|\mathbf{x}^j_t) p(\mathbf{x}^j_t|\mathbf{x}^i_t) d\mathbf{x}^j_t && (5.9)
\end{aligned}
$$

where in eq. (5.6) and eq. (5.7), Markov properties that observations are conditionally independent given $\mathbf{x}_t^i$ are applied. In eq. (5.8), the adjacent state of $\mathbf{x}_t^j$ appears by introducing an integral operation. In eq. (5.9), the Markov property that the observation depends on its own $\mathbf{x}_t^i$ is exploited to simplify the derivation. By substituting eq. (5.9) back into eq.(5.5), we can get a new form of the posterior density,

$$\begin{aligned}
&p(\mathbf{o}_{0:t}^i, \mathbf{x}_{0:t+1}^i | \mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) \\
&= c_t \cdot p(\mathbf{o}_t^i | \mathbf{x}_{t+1}^i, \mathbf{x}_t^i) p(\mathbf{q}_t^i | \mathbf{x}_t^i) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}_t^j | \mathbf{x}_t^j) p(\mathbf{x}_t^j | \mathbf{x}_t^i) d\mathbf{x}_t^j \\
&\cdot p(\mathbf{x}_{t+1}^i | \mathbf{x}_t^i) p(\mathbf{o}_{0:t-1}^i, \mathbf{x}_{0:t}^i | \mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)})
\end{aligned} \tag{5.10}$$

By marginalization [1], the above posterior can be further simplified,

$$\begin{aligned}
&p(\mathbf{x}_{0:t+1}^i | \mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) \\
&= \int p(\mathbf{o}_{0:t}^i, \mathbf{x}_{0:t+1}^i | \mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) d\mathbf{o}_{0:t}^i
\end{aligned} \tag{5.11}$$

$$\begin{aligned}
&= \int c_t \cdot p(\mathbf{q}_t^i | \mathbf{x}_t^i) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}_t^j | \mathbf{x}_t^j) p(\mathbf{x}_t^j | \mathbf{x}_t^i) d\mathbf{x}_t^j \\
&\cdot p(\mathbf{o}_t^i | \mathbf{x}_{t+1}^i, \mathbf{x}_t^i) p(\mathbf{x}_{t+1}^i | \mathbf{x}_t^i) \\
&\cdot p(\mathbf{o}_{0:t-1}^i, \mathbf{x}_{0:t}^i | \mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)}) d\mathbf{o}_{0:t}^i
\end{aligned} \tag{5.12}$$

$$\begin{aligned}
&= c_t \cdot p(\mathbf{q}_t^i | \mathbf{x}_t^i) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}_t^j | \mathbf{x}_t^j) p(\mathbf{x}_t^j | \mathbf{x}_t^i) d\mathbf{x}_t^j \\
&\cdot p(\mathbf{x}_{t+1}^i | \mathbf{x}_t^i) \int p(\mathbf{o}_t^i | \mathbf{x}_{t+1}^i, \mathbf{x}_t^i) d\mathbf{o}_t^i \\
&\cdot \int p(\mathbf{o}_{0:t-1}^i, \mathbf{x}_{0:t}^i | \mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)}) d\mathbf{o}_{0:t-1}^i
\end{aligned} \tag{5.13}$$

$$\begin{aligned}
&= c_t \cdot p(\mathbf{q}_t^i | \mathbf{x}_t^i) \prod_{j \in \mathcal{A}(i)} \int p(\mathbf{q}_t^j | \mathbf{x}_t^j) p(\mathbf{x}_t^j | \mathbf{x}_t^i) d\mathbf{x}_t^j \\
&\cdot p(\mathbf{x}_{t+1}^i | \mathbf{x}_t^i) \int p(\mathbf{o}_t^i | \mathbf{x}_{t+1}^i, \mathbf{x}_t^i) d\mathbf{o}_t^i \\
&\cdot p(\mathbf{x}_{0:t}^i | \mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)}).
\end{aligned} \tag{5.14}$$

From eq. (5.11) to eq. (5.12), we substitute eq. (5.10). In eq. (5.13), the Markov property that $\mathbf{o}_t^i$ is independent with $\mathbf{o}_{0:t-1}^i$ is used to simplify the derivation, which can be easily verified from the graphical model in Fig. 5.2.

The derived sequential updating rule in eq. (5.14) clearly simulates different correlations and

interactions inside the snake robot itself and between the robot and environment. As we can see, there are five distinguishing densities: i) the inter-link likelihood $p(\mathbf{x}_t^j|\mathbf{x}_t^i)$; ii) the observation likelihood $p(\mathbf{q}_t^i|\mathbf{x}_t^i)$ and $p(\mathbf{q}_t^j|\mathbf{x}_t^j)$; iii) the state dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$; iv) the environmental input function $p(\mathbf{o}_t^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i)$; v) the previous posterior $p(\mathbf{x}_{0:t}^i|\mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)})$.

Compared with the centralized formulation in [2], this newly proposed method has three important advantages: 1) It avoids using the computationally expensive joint state representation but chooses a distributed state space, which can exploit the benefits of parallel computing; 2) The derived "inter-link likelihood" could be used to simulate the physical correlation among snake robot links; 3) The derived "environmental input function" can be used to model the interactions between the snake robot and its surrounding objects.

5.2 Density Estimation

Different density estimation techniques could be used to approximate the derived probability densities. Monte Carlo (MC) method [66] is chosen in this work for a better comparison with our previous centralized solution. Specifically, The posterior density $p(\mathbf{x}_{0:t+1}^i|\mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)})$ could be approximated by a group of samples with associated weights $\{\mathbf{x}_{0:t}^{i,n}, \omega_t^{i,n}\}_{n=1}^{N_s}$ where $\{\mathbf{x}_{0:t}^{i,n}, n = 1, \ldots, N_s\}$ are samples, and $\{\omega_t^{i,n}, n = 1, \ldots, N_s\}$ the weights. When the samples are drawn from an importance function $f(\cdot)$, their weights $w_t^{i,n}$ are proportional to $p(\mathbf{x}_{0:t+1}^{i,n}|\mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)})/f(\cdot)$ based on the importance sampling theory. For the sequential case, $f(\cdot)$ could be further factorized by the following equations,

$$\begin{aligned} & f(\mathbf{x}_{0:t+1}^i|\mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) \\ = & f(\mathbf{x}_{t+1}^i|\mathbf{x}_{0:t}^i, \mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) f(\mathbf{x}_{0:t}^i|\mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) \\ = & f(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i) f(\mathbf{x}_{0:t}^i|\mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)}). \end{aligned} \tag{5.15}$$

where the Markov property $f(\mathbf{x}_{t+1}^i|\mathbf{x}_{0:t}^i, \mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)})$
$= f(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i)$ and $f(\mathbf{x}_{0:t}^i|\mathbf{q}_{1:t}^i, \mathbf{q}_{1:t}^{\mathcal{A}(i)}) = f(\mathbf{x}_{0:t}^i|\mathbf{q}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{\mathcal{A}(i)})$ is used to simplify the equation. Therefore,

the weights can be calculated as,

$$w_t^n = w_{t-1}^n \frac{p(\mathbf{q}_t^i|\mathbf{x}_t^{i,n}) \prod_{j\in\mathcal{A}(i)} \int p(\mathbf{q}^j|\mathbf{x}_t^j) p(\mathbf{x}_t^j|\mathbf{x}_t^{i,n}) d\mathbf{x}_t^j}{f(\mathbf{x}_{t+1}^{i,n}|\mathbf{x}_t^{i,n})}$$
$$\cdot p(\mathbf{x}_{t+1}^{i,n}|\mathbf{x}_t^{i,n}) \int p(\mathbf{o}_t^i|\mathbf{x}_{t+1}^{i,n}, \mathbf{x}_t^i) d\mathbf{o}_t^i \tag{5.16}$$
$$\approx w_{t-1}^n \frac{p(\mathbf{q}_t^i|\mathbf{x}_t^{i,n}) \prod_{j\in\mathcal{A}(i)} \sum_{n'=1}^{N_s'} p(\mathbf{q}^j|\mathbf{x}_t^{j,n'}) p(\mathbf{x}_t^{j,n'}|\mathbf{x}_t^{i,n})}{f(\mathbf{x}_{t+1}^{i,n}|\mathbf{x}_t^{i,n})}$$
$$\cdot p(\mathbf{x}_{t+1}^{i,n}|\mathbf{x}_t^{i,n}) \sum_{k=1}^{K} p(\mathbf{o}_t^{i,k}|\mathbf{x}_{t+1}^{i,n}, \mathbf{x}_t^i). \tag{5.17}$$

As we can see, we use a summation to estimate the integral of $\mathbf{o}_t^i$ where $k = 1, \ldots, K$ is index of environmental objects such as target and obstacles, $n' = 1, \ldots, N_s'$ the samples. A Gaussian random walk model is used to estimate the state dynamics $p(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i)$. The importance function $f(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i)$ is chosen as the state dynamics, which is a common selection. Moreover, we have carefully designed various models to estimate the other densities as follows.

5.2.1 Estimation of Inter-Link Likelihood

The derived inter-link likelihood $p(\mathbf{x}_t^j|\mathbf{x}_t^i)$ indicates the internal constraints among links of the snake robot. If there is no such kind of interaction modeling, the distributed framework will have multiple independent agents. Each agent controls one robot link. Such a situation will make the snake robot lose its serpenoid gait because different links can not collaborate with each other rhythmically in the navigation. We choose a learning method with Bayesian Neural Network (BNN) [30] for this density estimation, which can simulate the uncertainty by a distribution over the network weights.

If a sample set $\{(\mathbf{x}_t^{j,n}, \mathbf{x}_t^{i,n})\}_{n=1}^N$ is drawn from the distribution $(\mathbf{x}_t^{j,n}, \mathbf{x}_t^{i,n})$, the objective is to find an estimate $\hat{p}(\mathbf{x}_t^j|\mathbf{x}_t^i)$ of the true conditional density $p(\mathbf{x}_t^j|\mathbf{x}_t^i)$. In particular, the dropout scheme [30] is adopted. The neural network weights $\mathbf{w}_{\text{BNN}}$ are estimated by a maximum likelihood estimation,

$$\mathbf{w}_{\text{BNN}}^* = \underset{\mathbf{w}_{\text{BNN}}}{\text{argmax}} \sum_{n=1}^{N} \log \hat{p}_{\mathbf{w}_{\text{BNN}}}(\mathbf{x}_t^j|\mathbf{x}_t^i) \tag{5.18}$$

We have learned the likelihood function from 20 trials with 40000+ samples, which are produced by a snake robot similar as [2]. Since there are different interactions with environmental objects during

the learning process, the training data can model the inter-link function with uncertainties. The prediction uncertainty can be generated by the uncertainty of BNN weights through marginalising over the approximate density with Monte Carlo integration.

5.2.2 Estimation of Environmental Input Function

The density $p(\mathbf{o}_t^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i)$ models the interaction between state $\mathbf{x}_{t+1}^i$ and the environmental objects $\mathbf{o}_t^i$, which includes both target and obstacles. It can be further factorized by the conditional independence property since the target is independent with obstacles, $p(\mathbf{o}_t^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i) = p_a(\mathbf{o}_{a,t}^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i)p_s(\mathbf{o}_{s,t}^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i)$, where $p_a(\cdot)$ models the attraction from the target, and $p_s(\cdot)$ models the stimuli by obstacles.

Similar to [2], we propose two distributed paradigms for link i. For the target attraction function, we have

$$p_a(\mathbf{o}_{a,t}^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i) = \frac{1}{\rho_a}\exp\left\{-\frac{\|\mathbf{o}_{a,t}^i - \Delta\mathbf{x}_{t+1}^i\|^2}{\sigma_a^2}\right\} \tag{5.19}$$

where ρ_a is a normalization constant, σ_a a standard variation, $\|\mathbf{o}_{a,t}^i - \Delta\mathbf{x}_{t+1}^i\|$ the Euclidean distance between two arguments, $\Delta\mathbf{x}_{t+1}^i = \mathbf{x}_{t+1}^i - \mathbf{x}_t^i$.

For the stimulus from obstacles, we have

$$p_s(\mathbf{o}_{s,t}^i|\mathbf{x}_{t+1}^i, \mathbf{x}_t^i) = 1 - \frac{1}{\rho_s}\exp\left\{-\frac{\|\mathbf{o}_{s,t}^i - \Delta\mathbf{x}_{t+1}^i\|^2}{\sigma_s^2}\right\} \tag{5.20}$$

where ρ_s is a normalization constant, σ_s a standard variation, $\|\cdot\|$ also the Euclidean distance between two arguments.

5.2.3 Estimation of Observation Likelihood

The distribution $p(\mathbf{q}_t^i|\mathbf{x}_t^i)$ indicates the likelihood from $\mathbf{x}_t^i$ to $\mathbf{q}_t^i$. In control theory, this density gives a feedback, which can be observed for a better estimation of the state. Without such a closed-loop design, the controller may be not stable especially when facing obstacle collisions since the potential bias can not be rectified in time during the snake robot's locomotion. Many methods have been reported for the estimation of the observation likelihood in the literature. We choose the

CNN-based learning algorithm [38] to detect the observations in our implementation. After that, the likelihood $p(\mathbf{q}_t^i|\mathbf{x}_t^i)$ is modeled as a Gaussian distribution $N(|\mathbf{q}_t^i - \mathbf{x}_t^i|, \sigma_o^2)$, where $|\mathbf{q}_t^i - \mathbf{x}_t^i|$ is the mean, σ_o the standard variance.

5.2.4 Estimation of State Dynamics

The state transition $p(\mathbf{x}_{t+1}^i|\mathbf{x}_t^i)$ predicts the state one step further based on the current state. It can be used to simulate the systematic uncertainty inside a snake robot's dynamics. For instance, there are many spur gears and electric components in the snake robot's servo motors, which have inherent disturbances due to the non-negligible friction and/or inertia resistance. Such kind of uncertainties are hard to be calculated without a statistical estimation. Without any prior knowledge and loss of generality, this state transition density can be model by a Gaussian random walk.

5.3 Experimental Performance

We have demonstrated the performance of the proposed Distributed Bayesian Controller (DisBC) in different situations including both simulations and real-world scenarios.

5.3.1 Simulation Analysis

We performed simulations by using ACM-R5 snake robot in V-REP. 20 cylinder obstacles were set inside a 5m $\times$ 5m scene with walls around.

Path Following

We first tested the performance of DisBC for path following. The results are shown in Fig. 5.3. The robot moves straightly from right to left, where the green curve shows the locomotion trajectory. As we can see, the control seems to work reasonably well in the physical simulation, despite some small variations. Fig. 5.3B to 5.3F show the recorded position, speed, joint angle and torque of the robot's head with time. Although a decentralized control framework is adopted, one

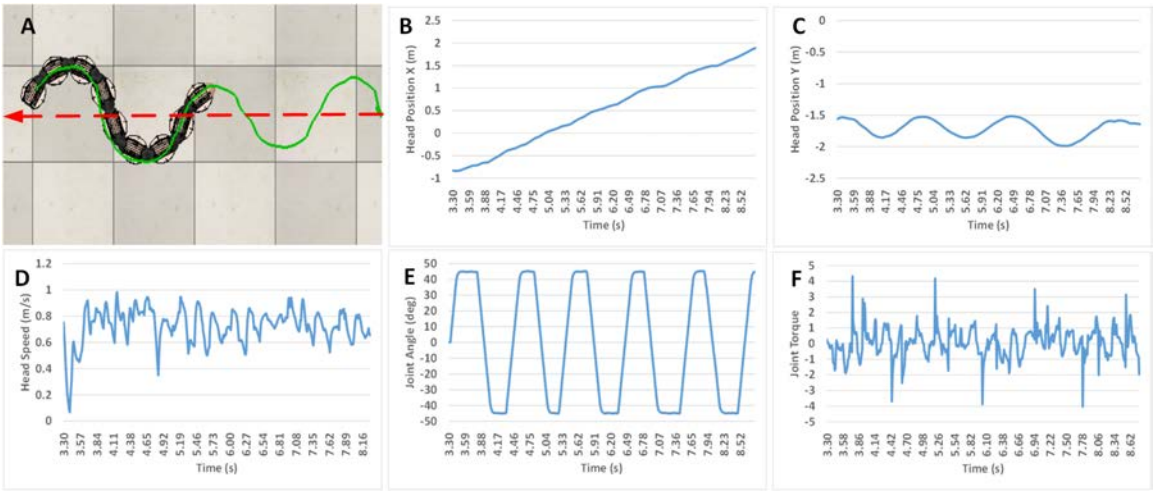

Figure 5.3. Simulation results of path following using ACM-R5.

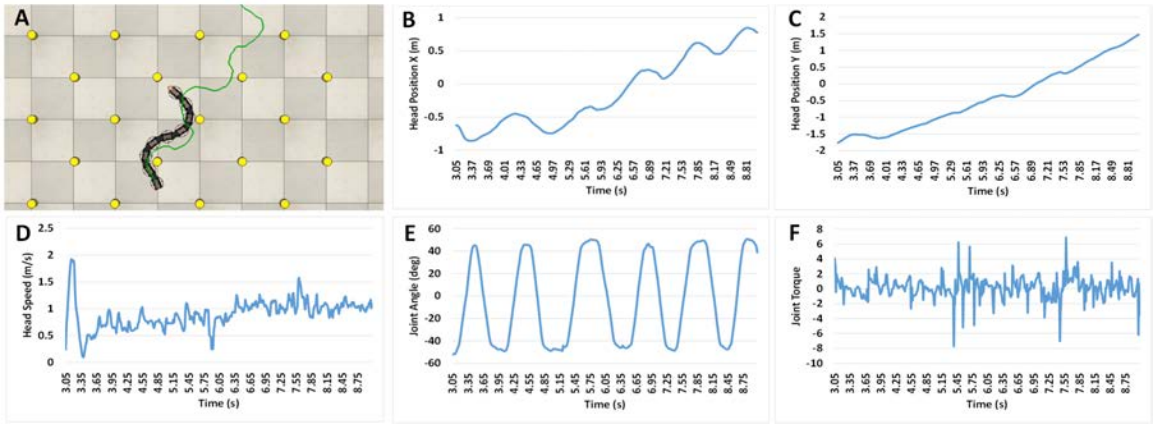

Figure 5.4. Simulation results of obstacle avoidance using ACM-R5.

controller per robot link, it still works well because all controllers can collaborate with each other based on the proposed Bayesian updating rule. The robot can trace the desirable path, tolerating the environmental uncertainty as shown in Fig.5.3B and 5.3C. Moreover, the serpenoid gait is kept successfully as can be seen from Fig.5.3A, 5.3B, and 5.3C.

Obstacle Avoidance

To test the performance of the proposed DisBC for environment with obstacles, we have designed an experiment using a scene as shown in Fig. 5.4A. It also reports the overall trajectory

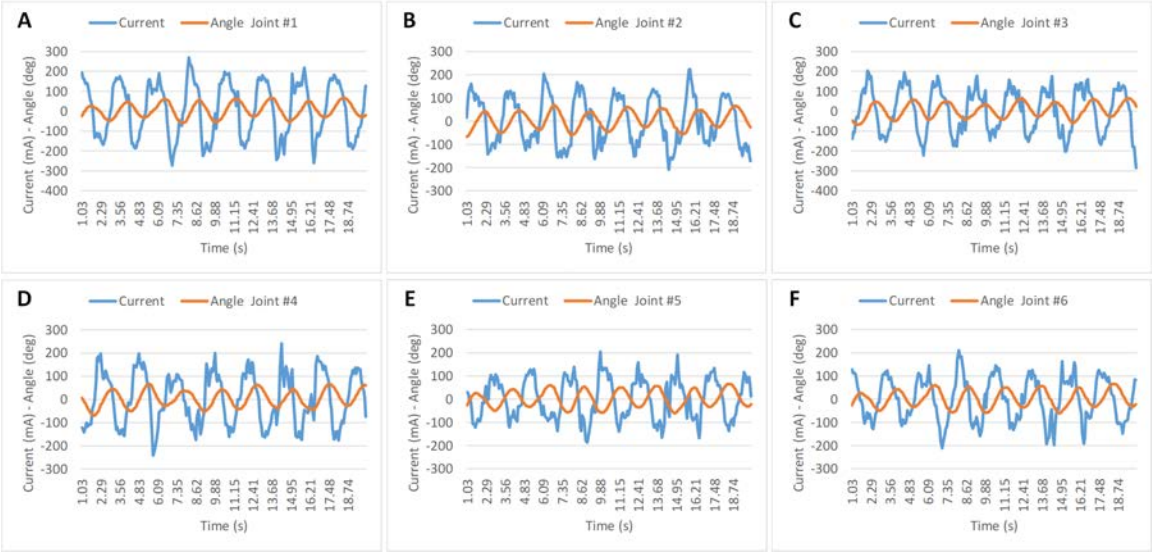

Figure 5.5. Quantitative performance of both current and angles for all snake robot links.

of one test where the snake robot swings from the top-right to bottom-left. Equipped with the interaction mechanism, the robot can effectively move through the peg array as shown in Fig. 5.4B and 5.4C. The initial rapid change is shown in Fig. 5.4D and 5.4E as the snake robot converges to the serpenoid gait at the beginning. The overshoot of torque in Fig. 5.4F occurs when the direction changes dramatically after avoidance of obstacles. Overall, the serpenoid gait is kept well. Furthermore, the movement is relatively smooth.

5.3.2 Real-World Data

In order to demonstrate the effectiveness of the DisBC method, we have used the snake robot *JAW*-I in different carefully designed scenarios. Similar to [15], we have exploited an overhead camera to watch the scene and observe the robot. Consequently, we have used a CNN-based video detector [38] to measure the obstacle position and robot link's shape variation dynamically. By the proposed models and density estimation functions, we can further update posterior density for the robot states, which are then used to calculate the control signals.

We have placed 10+ beverage bottles and one target randomly in a rectangle area. Both the target and obstacles are movable either manually or by collisions. This setup introduces a number

of different uncertainties and is suitable to test the stability of the proposed DisBC in a dynamical scene, which is very common in practical application. We have tested it with 40+ trials. With the results, we have done a thorough analysis of the DisBC controller both qualitatively and quantitatively.

Fig. 5.6 presents some snapshots in two different scenes. As we can see, although the target was changed, the proposed DisBC method can still successfully lead the snake robot to reach it fast and smoothly. Moreover, even with crowded obstacles, the controller could handle the uncertainties well and keep the gait during the environmental interactions.

Fig. 5.5 shows the performance of both current and angle for all robot links in the scene of 5.6(a). The power consumption in motors can be evaluated by the current. There are different types of uncertainties during the robot's navigation. The DYNAMIXEL XL430-W250 servo motor itself has different spur gears and electric components, which easily give various inherent uncertainties of power consumption and frictions. Moreover, the frequent collisions and interactions with obstacles and the ground also introduce additional uncertainties. This can explain why a stochastic analysis is really desirable. As shown in the figure, the DisBC controllers can be run in parallel very well due to modeling the inter-link interaction as well as the environmental inputs. The overall movement of the snake robot is smooth without any failure or large jerky during the navigation. It can successfully swing through the peggy array and keep the gait very well as can be seen from the reported curves. A few moderate overshoots of current appears in Fig. 5.5D and 5.5E. They are induced by the obstacle interactions. Benefiting from the stochastic modeling of snake robot motion, the inter-link interaction and the environmental inputs, the entire performance is relatively stable. Particularly, the controller is robust to the initial position and shape, which normally gives a lot of trouble for many reporting methods. Additionally, although we accept a purely distributed solution, one controller per link in parallel simultaneously, the different links can coordinate with each other by not only the serpenoid model but also the different interaction schemes.

To verify the performance of the proposed inter-link model, the environmental input function, as well as the distributed solution, we have given a detailed comparison between DisBC with the state-of-the-art Centralized Bayesian Controller (CBC) [2]. The same observation likelihood and

Table 5.1. Quantitative comparison between DisBC and two controllers using the CenBC method

Method	Time(s)	Sample No.	Cycles	Unexpected Collisions
DisBC	17.11	90	31	3.6
CenBC[1]	51.41	300	33	3.8
CenBC[2]	19.06	90	46	18.8

Figure 5.6. Results using the proposed DisBC method on real-world data.

state transition have been used in the implementation. Similar setup has been used as shown in Fig. 5.6. The comparisons are given in Table 8.2 where we averaged the results of six trials using two methods, respectively. The running platform is a desktop with Intel i7-1065G7 CPU and 16GB RAM.

As shown from the table, more samples can usually give more stable control results. However, a large number of samples can easily introduce a very higher computational complexity using the same centralized method of CBC. With a relatively comparable performance, CBC[1] requires much more samples than DisBC, which is expected because the parallel computing can greatly decrease the running time. This shows the promising advantage of the distributed framework especially for the highly redundant snake robot. Meanwhile, DisBC achieved much more efficient and robust results than CBC[2] with much less cycles and unexpected collisions with the same number of samples. The reason lies in the proper modeling of various interactions inside the snake robot itself and with the environment.

5.4 Summary

In this chapter, a fully distributed Bayesian framework, one agent per link, is proposed for the snake robot. Compared with the centralized framework, it may need additional computation cost for the cooperation and communication. However, such a tradeoff is definitely worthy especially when the snake robot has a very long body length. It avoids to use the high dimensional centralized state representation, thus solving the exponentially increased computational cost. Overall, the proposed DisBC approach has the following distinguishing contributions: 1) A fully distributed graphical representation and decomposed modeling of the snake robot; 2) a mathematically derived Bayesian updating rule of modeling not only the uncertainties but also the various interactions inside the robot and outside the body with surroundings; 3) likelihood models for inter-link interaction and environmental input functions are designed. Both simulation and real-world tests have shown the promising performance.

Chapter 6

Centralized Coach-Based Bayesian Control

A robot is an essentially active agent that interacts with the physical world, and it generally does so under uncontrollable or harmful circumstances [33]. On the basis of inadequate and uncertain information, robots must perceive, decide, plan, and execute actions. Deep reinforcement learning has been regarded as a valuable tool by the robotics community, and it had begun to be used and advanced. However, not enough research has been done for snake robot control due to challenges of the highly redundant structure, and frequent interactions inside the robot and with the environment during the locomotion. In this chapter, we study the possibility of applying state-of-the-art Bayes estimation into RL domain for robust snake control.

The majority of available snake robot control methods still rely on precise modeling and sensing assumptions, which restrict performance in unknown situations [11]. Scientists have recently begun to use neural network-based approaches and machine learning techniques to model actual snake interactions with obstacles [63]. The use of reinforcement learning to create stable controllers has sparked a lot of attention. For various robot applications, numerous model-free methods were used [3] [34]. Despite its immense promise, RL's implementation in robotics is hampered by the arduous training process caused by data shortage and poor convergence. Because robotics involves complicated physical systems, samples could be costly owing to the considerable time it takes to

accomplish tasks, the requirement for user intervention, and the need for repairs and maintenance. It is typically hard to use RL for snake robot control without an effective scheme since the large D-OF makes the problem much more challenging. By generalizing observed transitions, model-based RL enables agents to develop appropriate policies with a fewer number of trials. A probabilistic model of the agent's ignorance about the world can increase data efficiency even further, allowing it to pick actions under uncertainty [67]. Prior information is encoded and uncertainty in model parameters is represented in Bayesian RL [68]. It offers a potential way to tackling the problems of slow convergence in RL. Most disclosed solutions, however, still dedicated to numerical modeling and seldom applicable to practical robot control.

To address the aforementioned challenges, we offer a centralized coach-based principled paradigm in this chapter. When compared to traditional model-free RL, it uses a guided procedure to teach the agent, especially during the start of training, substantially reducing convergence time and using far less data. The suggested technique is more tractable and efficient to implement than conventional model-based RL algorithms. Furthermore, the system and environment's uncertainties are all explicitly treated inside a unified stochastic framework.

6.1 Methodology of the Centralized Coach-Based Framework

We need a way to embed prior knowledge and mature experience into the RL's formulation for snake robot control in order to perform an efficient policy search with a small number of training episodes. It should maintain a probability distribution across model parameters to explicitly represent uncertainty, and perform actions that optimize the expected long-term reward with respect to the distributions. The policy calculation may then examine both the reward and contribution of actions to learn unknown model parameters, resulting in an appropriate trade-off between exploration and exploitation. As a result, we decided to frame the notion as a coach-based RL architecture.

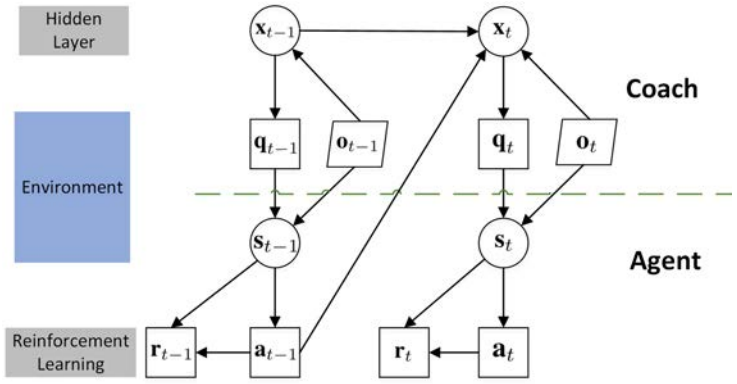

Figure 6.1. The probabilistic graphical model for coach-based reinforcement learning.

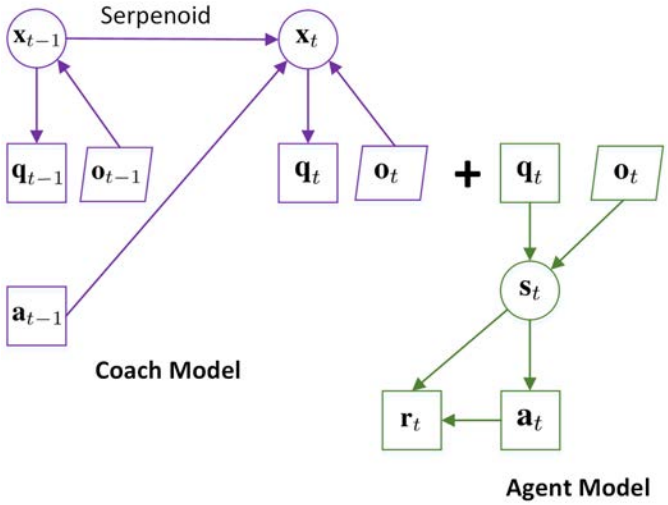

Figure 6.2. The coach model (purple) and agent model (green) after graphical decomposition.

6.1.1 Probabilistic Graphical Modeling

As shown in Fig. 6.1, we propose a Probabilistic Graphical Model (PGM) [69] to facilitate the analysis of the snake robot control issue. The designed model is a Partially Observable Markov Decision Process (POMDP) [70]. It has two coupled parts: the agent and the coach. Unlike traditional model-based RL algorithms, which have complicated dynamics to predict, making convergence problematic for robot applications, we use a coach-guided formulation to speed up the training process. The traditional environment state in RL is further split into three different types of nodes: the agent state $\mathbf{s}_t$, the observation $\mathbf{q}_t$, and the external interactive objects such as targets and obstacles $\mathbf{o}_t$, where t is the time index. The observation $\mathbf{q}_t$ includes the location, direction, and link angles of

the snake robot, which can be estimated by a simulator or an image detector. The external objects $\mathbf{o}_t$ includes the location of all currently interacted targets and obstacles. Although the number of detected obstacles is dynamically changing, the nearest one is used as discussed in Section 6.1.2. By doing this, we provide a more explicit modeling and deeper analysis of the environment. The coach's hidden states $\mathbf{x}_t$ can then be partially observed by $\mathbf{q}_t$. We denote the set of all agent states up to time t by $\mathbf{s}_{0:t}$, where $\mathbf{s}_0$ is the initialization prior, the action of an agent by $\mathbf{a}_t$, the set of all actions up to time t by $\mathbf{a}_{1:t}$, the reward of agent by $\mathbf{r}_t$, and the set of all rewards up to time t by $\mathbf{r}_{1:t}$. Similarly, we further denote the set of all coach states up to time t by $\mathbf{x}_{0:t}$, where $\mathbf{x}_0$ is the initialization prior, the set of all observations up to time t by $\mathbf{q}_{1:t}$, and the set of all external interaction up to time t by $\mathbf{o}_{1:t}$. Formally, the POMDP for RL agent is a tuple $\langle S, A, R, \nu \rangle$, where S is the set of agent states, A the set of actions, $R(\mathbf{s}_t, \mathbf{a}_t)$ the reward received when taking action $\mathbf{a}_t$ in state $\mathbf{s}_t$, and ν the discount factor. Different with common practice of importing the action $\mathbf{a}_{t-1}$ into the next step agent state $\mathbf{s}_t$ directly, we modulate it with coach's state $\mathbf{x}_t$. Both $\mathbf{x}_t$ and $\mathbf{a}_t$ are defined in shape space as discussed later in Section 6.1.2. After they are fused together, the correspondingly generated control signal is then sent to actuators. The robot will next interact with external objects $\mathbf{o}_t$, which will give the observation $\mathbf{q}_t$. The command may not be implemented precisely owing to system and environment uncertainties. Therefore, we use a re-evaluation process based on observation $\mathbf{q}_t$ and external objects $\mathbf{o}_t$ to compute the agent state $\mathbf{s}_t$. The dynamics is indicated by the directed edges between the coach's states. The edge between state and action show the policy π_p. We inherit the commonly accepted Markov assumption, i.e. conditional independence, throughout the whole graphical model. It provides a potential framework for dealing with uncertainty in a cohesive manner by representing unknown parameters using probabilistic distributions.

Compared to working from basic trial-and-error as in model-free learning, being able to plan based on a model is usually far more sample-efficient. Training a good model, on the other hand, is often challenging, and cumulative errors from model deficiencies usually result in low agent performance. As a result, many early deep RL successes (such as DQN and A3C) would prefer to use a model-free form. Despite its enormous potential, direct application of the PGM in Fig. 6.1 by a model-based RL method is quite difficult. For snake robots, the presented coach-based strategy can provide a better balance between exploration and exploitation than typical model-based RL

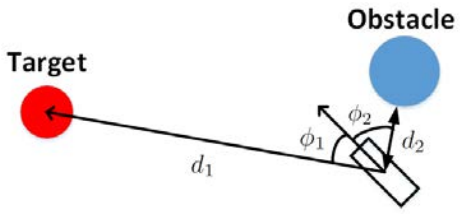

Figure 6.3. Determining the robot's position with relation to the goal and the closest obstacle.

methods. Two models, one for coach (purple) and the other for agent (green), could be built using graphical decomposition theory [65], as illustrated in Fig. 6.2. This graphic breakdown is valid because all of the knowledge from Figure 6.1 is preserved without loss of any nodes or edges. As shown in Fig. 6.2, the dynamics makes use of the serpenoid model to aid in the analysis. It may be used to steer policy searching, resulting in a significant increase in convergence speed. After decomposition, the agent model is kept as a model-free RL process for a simple implementation with a modified input state. This approach is referred to as "Centralized Coach-based Reinforcement Learning" (CCRL).

Overall, the aforementioned RL issue is addressed, in which an agent operates in a stochastic environment by sequentially selecting actions across a succession of time steps in order to reduce a cumulative cost. This coach-guided policy search approach consists of two critical aspect: The first aspect involves estimating a dynamics model using prior knowledge and updating data, where a Bayesian approach is critical due to the uncertainty inherent in practical scenarios; the second aspect involves learning the parameters $\mathbf{w}_\pi$ of an optimal policy function π_p that maximizes the reward expectation.

6.1.2 Shape-Based State and Action Space

We chose the serpenoid model in eq. (1.11), because it has a lower state dimensionality, which helps speed up learning convergence [5]. The coach state is denoted as $\mathbf{x}_t = \langle \gamma_t, \alpha_t \rangle$. A 4-tuple expression is used for the agent state. It represents the robot's and the environment's relative position, as seen in Fig.6.3,

$$\mathbf{s} = \langle d_1, \phi_1, d_2, \phi_2 \rangle \tag{6.1}$$

where d_1 the distance between the target and the robot for determining if the movement is approaching the destination, the angle in between is ϕ_1, d_2 the distance between the nearest obstacle and the robot, and the angle in between is ϕ_2.

To keep the action space dimensionality low, we choose $\mathbf{a} = \langle \Delta\gamma, \Delta\alpha \rangle$, where $\Delta\gamma$ and $\Delta\alpha$ and are the offset and amplitude increments respectively. Particularly, constant increments are defined for $\Delta\gamma \in \{-\Delta_\gamma, 0, +\Delta_\gamma\}$, and $\Delta\alpha \in \{-\Delta_\alpha, 0, +\Delta_\alpha\}$.

6.2 Centralized Coach Modeling By Bayesian Density Propagation

The purpose now is to recursively calculate the control signal $\mathbf{x}_t$ using the coach model in Fig. 6.2. This could be accomplished by first estimating the posterior distribution $p(\mathbf{x}_{0:t}|\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1})$, and then using the external information $\mathbf{o}_t$ to compute the agent's state $\mathbf{s}_t$,

$$
\begin{aligned}
& p(\mathbf{x}_{0:t}|\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1}) \\
&= \frac{p(\mathbf{x}_t, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{x}_{0:t-1}, \mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2})}{p(\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1})} p(\mathbf{x}_{0:t-1}, \mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}) && (6.2) \\
&= \frac{p(\mathbf{x}_t, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{x}_{t-1})}{p(\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1})} p(\mathbf{x}_{0:t-1}, \mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}) && (6.3) \\
&= \frac{p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1})}{p(\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1})} p(\mathbf{x}_{0:t-1}, \mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}) && (6.4) \\
&= \frac{p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1})}{p(\mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2})} p(\mathbf{x}_{0:t-1}|\mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}) && (6.5) \\
&= k_c \cdot p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{x}_{0:t-1}|\mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}). && (6.6)
\end{aligned}
$$

We apply the Bayes rule in eq. (6.2). In eq. (6.3), we use the conditional independence property $p(\mathbf{x}_t, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{x}_{0:t-1}, \mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2}) = p(\mathbf{x}_t, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{x}_{t-1})$ from the coach model of Fig. 6.2. In eq. (6.4), the Markov property $p(\mathbf{x}_t, \mathbf{q}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_{t-1}|\mathbf{x}_{t-1}) = p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1}) p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1})$ is applied. In eq. (6.5), we get a recursive updating rule with posterior $p(\mathbf{x}_{0:t-1}|\mathbf{q}_{1:t-2}, \mathbf{a}_{1:t-2}, \mathbf{o}_{1:t-2})$ for time $t-1$. We also use a constant k_c to denote the denominator in eq. (6.6), since it does not contain the state $\mathbf{x}$. Because it has to know $p(\mathbf{x}_{0:2}|\mathbf{q}_1, \mathbf{a}_1, \mathbf{o}_1)$, the start of the updating rule in eq. (6.6) should meet $t \geq 3$. In this implementation, the initialization of variables $\mathbf{x}_{0:2}$, $\mathbf{q}_1, \mathbf{a}_1$, and $\mathbf{o}_1$ are preset.

This formula expresses the coach dynamics' uncertainty propagation clearly. Three probability densities are crucial to updating the posterior, as shown in eq. (6.6): the observation function $p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1})$, the interaction function $p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1})$ and the modulated dynamics $p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1})$. The next section discusses how to approximate these factors.

6.2.1 Observation Function

The probability from the coach state $\mathbf{x}_{t-1}$ to its observation $\mathbf{q}_{t-1}$ is modelled by probability density $p(\mathbf{q}_{t-1}|\mathbf{x}_{t-1})$. To estimate this observation density, various approaches have been proposed. To approximate this function, we employ the BNN-based learning method [30].

6.2.2 External Interaction Function

The interaction of the state $\mathbf{x}_{t-1}$ and the external objects $\mathbf{o}_{t-1}$ is represented by the function $p(\mathbf{o}_{t-1}|\mathbf{x}_{t-1})$. We use the BNN-based technique [30] to learn the function from training data.

6.2.3 Dynamics Modulation

We describe the modulated dynamics using a multivariate Gaussian, as illustrated in Fig. 6.2, so as to integrate the control signals from the agent and the coach,

$$\begin{aligned} & p(\mathbf{x}_t, \mathbf{a}_{t-1}|\mathbf{x}_{t-1}) \\ &= \frac{1}{2\pi|\Sigma_d|^{1/2}} \exp\Big\{ -\frac{1}{2}(\mathbf{x}_t - \mu)^T \Sigma_d^{-1}((\mathbf{x}_t - \mu)\Big\} \end{aligned} \tag{6.7}$$

where μ is the mean, Σ_d the covariance as follows,

$$\mu = \begin{bmatrix} \alpha_{t-1} + k_\alpha \Delta\alpha_t \\ \gamma_{t-1} + k_\gamma \Delta\gamma_t \end{bmatrix}, \quad \Sigma_d = \begin{bmatrix} \sigma_\alpha^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix} \tag{6.8}$$

In eq. (6.8), k_α and k_γ are modulation coefficients. When $k_\alpha = 0$ and $k_\gamma = 0$, only the coach is evaluated and the agent's actions are ignored. When k_α or k_γ increases, the involvement of the agent in the learning process becomes more prominent.

6.3 Agent Modeling By RL

The posterior $p(\mathbf{x}_{0:t}|\mathbf{q}_{1:t-1}, \mathbf{a}_{1:t-1}, \mathbf{o}_{1:t-1})$ calculates the modulated control signal, which is then supplied to actuators for implementation. As a result of its interactions with the surrounding environment, the robot will produce the observation $\mathbf{q}_t$. The two arrows from $\mathbf{o}_t$ to $\mathbf{x}_t$ and $\mathbf{x}_t$ to $\mathbf{q}_t$ in Fig. 6.2 represent the aforementioned procedure. It's challenging to represent such interplay simply using analytical formulation because of the uncertainty in the environment and the robot. We can, however, approximate this interaction using the partially observed information. The distance d_1 and the angle ϕ_1 denote the position between the robot and the target, as illustrated in Fig. 6.3. Their integration has the potential to direct the robot to the goal. The distance d_2 and the angle ϕ_2 denote the robot's relative location to the closest obstacle. They may collaborate and play a key part in avoiding obstacles. The RL network then imports this 4-tuple state to train policies.

6.3.1 Actor-Critic Network

The stochastic value function and policy function are calculated using two deep networks, one for Critic and another for Actor. The Actor networks are built using two hidden layers, but the Critic network uses just one. To add some non-linearity to activate the neural networks, rectifier linear Relu unit is utilized. An A2C-based reinforcement learning approach [34] is used in our solution to illustrate the performance of the suggested scheme and to compare it to the state of the art [35]. Alternatively, other model-free reinforcement learning approaches might be applied. Utilizing gradient descent and an entropy-based loss function as below, the agent learns a policy $\pi_p : S \rightarrow A$,

$$f_{\pi_p}(\mathbf{w}_A)=\log\Big\{\pi_p(\mathbf{a}_t|\mathbf{s}_t;\mathbf{w}_A)\Big\}\Big\{\mathbf{R}_t - \mathbf{V}(\mathbf{s}_t;\mathbf{w}_A)\Big\} + \kappa \cdot \mathbf{H}\Big\{\pi_p(\mathbf{s}_t;\mathbf{w}_A)\Big\}, \tag{6.9}$$

in which $\mathbf{R}$ denotes the discounted estimated return through time t, $\mathbf{H}$ is the entropy component to promote exploration, and κ represents the constant regulating exploitation and exploration. The expectation of discounted cumulative reward is set as the policy's value $\mathbf{V}$

$$\mathbf{V} = E\left(\sum_{t=0}^{\infty} \nu_t R(\mathbf{s}_t, \mathbf{a}_t)\right), \tag{6.10}$$

where the expectation is with respect to the random variable $\mathbf{s}_t$ at time t, ν_t the discount coefficient. The goal is to choose an optimal policy π_p^* with maximal value.

Strategy of Reward

Unlike reward design in [4], which simply incorporates obstacle-aided shape change, we combine obstacle avoidance with path planning. As a result, we suggested this reward function,

$$\mathbf{r}_t = r_1 + r_2 + r_3 \tag{6.11}$$

where

$$r_1 = -k_1, \tag{6.12}$$

$$r_2 = \begin{cases} 0 \\ \frac{-k_2}{d_2+1}, d_2 < \text{th}_2 \end{cases} \tag{6.13}$$

$$r_3 = \begin{cases} 0 \\ k_3, d_1 < \text{th}_1 \end{cases} \tag{6.14}$$

In our test, parameters $k_1 = 0.01$, $k_2 = 0.1$, $k_3 = 20$, $\text{th}_1 = 5$, $\text{th}_2 = 50$. The penalty $r1$ is used to penalize any extra motion. It motivates the robot to go to the destination as quickly as possible. The second return, is used for obstacle avoidance. The third item r_3 is a bonus given when the robot arrives at the destination.

6.3.2 Process of Learning

We also use a commutative mechanism throughout the training phase to produce a good balance between exploitation and exploration. In particular, the RL agent is alternately executed with and without coach assistance. In our tests, such a technique may provide a more stable result than just running the coach-based RL. The coach might not always be the global optimal option, though

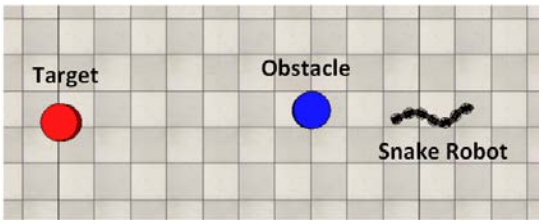

Figure 6.4. Scenario of Training with one obstacle and one target.

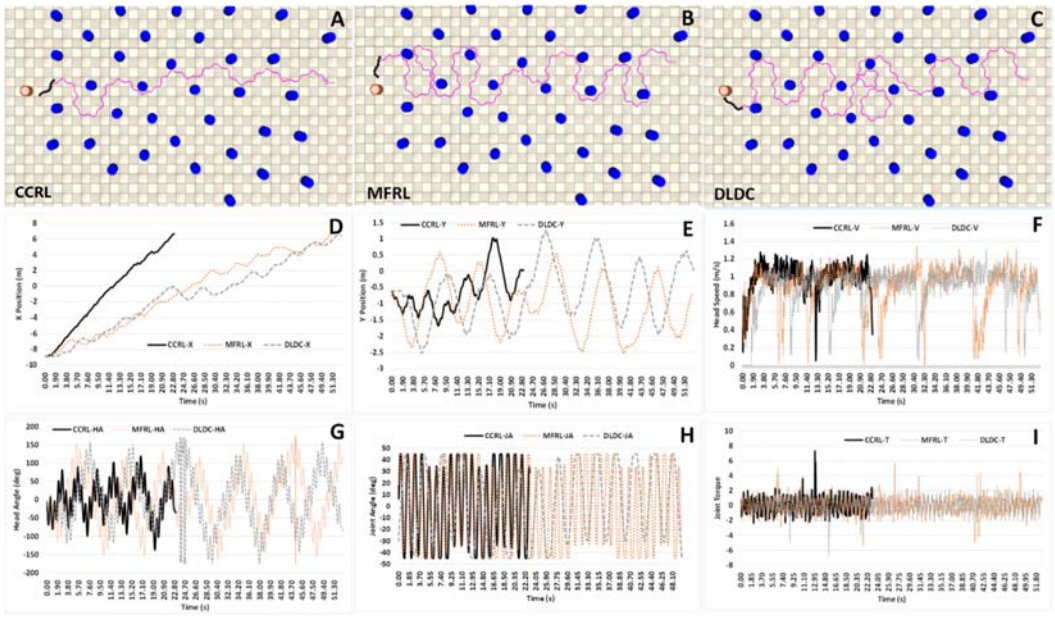

Figure 6.5. Simulation results using the proposed CCRL with comparisons of MFRL [3] and state-of-the-art DLDC [4].

it might provide mature expertise for learning. As a result, exploration is promoted by a balance between these two ways.

6.4 Experimental Analysis

With comparisons of Distributed Learning of Decentralized Control (DLDC) and Model-Free RL (MFRL) [3] techniques, both simulation and real-world data were implemented to assess the performance of the CCRL method.

Table 6.1. Comparative analysis of training methods

Item	CCRL	MFRL	DLDC
Episodes	120 ~ 260	1.2 ~ 1.7k	11 ~ 14k
Time (min)	31 ~ 65	250 ~ 300	700 ~ 800

6.4.1 Simulation Results

To do simulations, the ACM-R5 in the Virtual Robot Experimentation Platform (V-REP) is used. Each actuator is controlled by the joint angle θ . Because the robot interacts with obstacles and the target, joint angles are altering in response to dynamic environment. As illustrated in Fig. 6.3, the relative position of the robot and the surroundings is dynamically evaluated. The learnt policy may then be used to compute actions. The proper actuator angles may then be calculated and fed to the controller. Despite the fact that various sensor noises may affect the state estimation, the suggested technique can still manage them effectively thanks to the Bayesian mechanism. Fig. 6.4 depicts a training situation with a target and an obstacle. Table 6.1 shows episode number required to achieve stable convergence. As shown in Table 6.1, the suggested CCRL can accomplish significantly more training efficiency thanks to the coach model's acceleration. Only due to more sophisticated reward design does our version of MFRL converge faster than DLDC. It may, for example, encourage the robot to approach the destination while dodging impediments. The original DLDC reward design, on the other hand, focus on transforming the robot's shape without taking route planning into account. In our implementation, it is found that this reward tends to fall in a local minimum that the snake just crawls around without approaching further. This local minimum may account for the long convergence for training. More sophisticated reward function design may help to reduce training time, but is susceptible to poor generalization for DLDC. This is to be considered in our research in future.

We used a 12m x 20m scenario with obstacles and a target to show the efficiency of the CCRL in complex environment. We assume that the setting is unknown beforehand, and only dynamically approximated as the robot moves through the scene. In general, there are three types of uncertainty in this situation: 1) The robot's internal uncertainty, such as inherent joint friction, current varia-

tion, and so on. 2) uncertainty in the external world, including interactions such as collisions with obstacles that aren't anticipated; 3) data uncertainty resulted from sensor noise. With the aid of Bayesian-based RL mechanism, the CCRL achieved promising results in coping with these various uncertainties.The results of the robot moving from the right to the left are shown in Fig. 6.5. To obtain a fair comparison, we adjusted the reward functions of DLDC and MFRL, adding route planning abilities similar to r_3 in (6.14). In Fig. 6.5A, the pink curve depicts CCRL's entire trajectory. The snake robot smoothly goes through the array without being stuck. The trajectories of DLDC and MFRL, which are substantially less efficient than CCRL, are shown in Fig. 6.5B and 6.5C. This is further supported by Fig. 6.5D and 6.5E, in which the CCRL trail is obviously the shortest. Because of unpredictable collisions, DLDC had to take detours. Fig. 6.5F compares the speed of robot head of 3 separate algorithms, and we observe that CCRL experienced less collisions with obstacles. Fig. 6.5G compares the angle changes of robot head, and the suggested CCRL had the highest stable performance. Because they applied similar training data but acquired different policies, the three approaches had similar serpenoid curves with different lengths, as shown in Fig. 6.5H. We can also see from the torque fluctuations of robot head joint in Fig. 6.5I that CCRL experienced less collisions due to fewer curve spikes, demonstrating the performance of the learnt control policy. Similar trajectories of DLDC and MFRL suggest that pure model-free RL might lead policy search to local optimum, not as efficient as the coach-based algorithm. Despite the fact that CCRL has a torque overshoot owing to collision with obstacle in Fig. 6.5I, its direction could still be maintained by using the Bayesian mechanism, which manages uncertainty well than other techniques.

Table 6.2. Quantitative comparison of different approaches

Average Item	CCRL	MFRL	DLDC	Improvements*
Routing Time (s)	23.10	52.41	52.95	55.9 ~ 56.4%
Collisions	2	10	8	75 ~ 80%

* Performance improvement of CCRL in comparison with MFRL and DLDC.

By running the simulated test five times, Table 6.2 provides a quantitative average comparison of three approaches. As can be seen, the robot's average routing time adopting CCRL is much

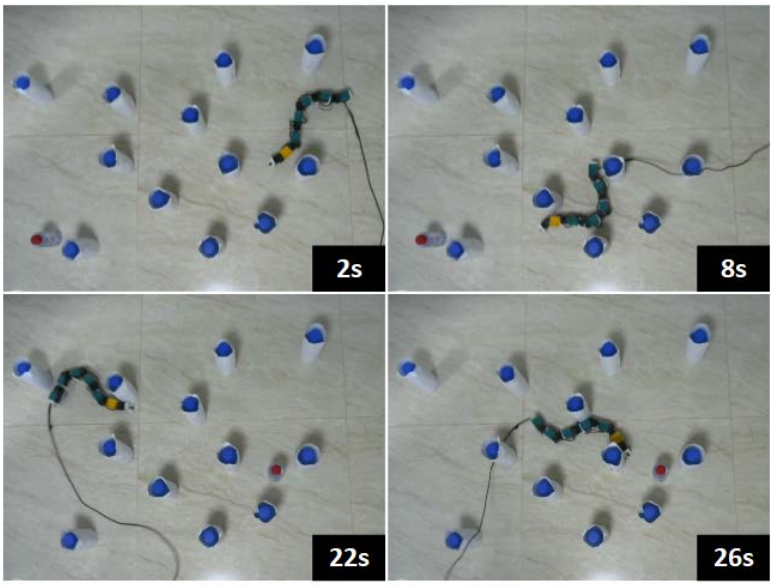

Figure 6.6. Results using the proposed CCRL method on the real-world data.

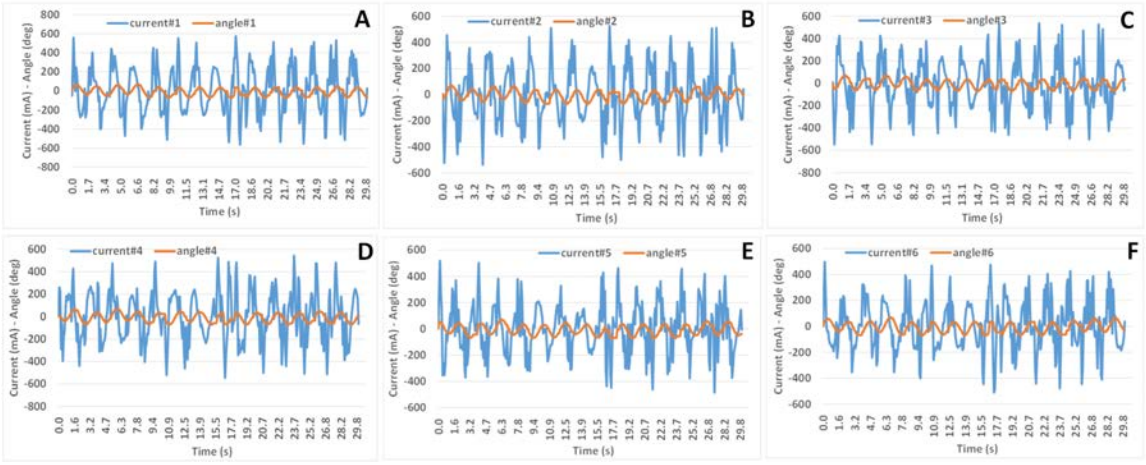

Figure 6.7. Measured results of all joints of the snake robot on real-world data.

smaller than that of DLDC and MFRL. With the same destination, it obtains a performance increase of more than 50% in terms of navigation efficiency. Furthermore, DLDC and MFRL have far more collisions while CCRL has only two, demonstrating that the presented approach could considerably increase obstacle avoidance performance in such testing circumstances.

6.4.2 Real-World Results

The suggested CCRL approach was tested in the real world using the physical robot *JAW*-I. A camera of bird's view, similar to [15], is put in place to watch the whole scenario. The object location and shape variation of robot links are determined in real time by using a CNN-based video detector [38].

In an unstructured environment with one movable goal and a set of unfixed bottle-like objects, we examined the performance of the suggested CCRL technique. As previously stated, the robot itself, as well as external interactions with the environment and inherent sensor noise, comprises various uncertainties. In practice, collision angles and forces, are difficult to estimate precisely. C-CRL, nonetheless, demonstrated its promising performance in tolerating those uncertainties. First, we used V-REP to train a strong policy. Then we transferred it into real world environment and did additional training for about thirty episodes to make it more adaptive to the actual terrain. With a half-hour, the whole procedure was completed. To test the robustness, more than 20 experiments were conducted, and it performed well. During the navigation, the serpenoid gait was well maintained with appropriate shape changes adapting to these pegs, as seen in Fig. 6.6. Despite the fact that the temporal frequency and link phase in (**??**) are fixed as constants, the joint angle θ is changing dynamically due to frequent interactions with the real world, causing the body wave to reshape in response. Additionally, due to the uncertainties imposed by collisions, noise and electrical disturbance, the real phase of the robot's body wave may vary a little during locomotion. By means of a coach-based Bayesian formulation, the performance is both efficient and effective. The guidance significantly reduces the search space and ensures a robust control policy. The Bayesian framework can tolerate various uncertainty, resulting in stable transfer and generalization. Just a few minor standstill were found, mostly during intense obstacle involvement. It could be the result of unknown frictions or unlearned conditions. The measurement results of joint current and angle can be seen in Fig. 6.7, in which the target moved once at about 12.8 seconds. The electrical current represents the motor's power usage as well as the torque it produces. As shown, the joint angles are quite smooth with nice serpenoid curve, allowing the robot to effectively avoid collisions and approach the

goal, even if it has moved. As anticipated, the electrical current fluctuation is regular with rhythmic patterns. Collisions with obstructions produce some minor overshoots in energy consumption.

6.5 Summary

We introduced a centralized coach-based Bayesian controller for snake robots that utilizes reinforcement learning in this chapter. RL often requires a large number of episodes, making its deployment on real robots costly and difficult. Due to the extra issue of considerable redundancy of freedom, there has been little study on snake robot control using reinforcement learning. We use a stochastic process with probabilistic density propagation to characterize the uncertainty in both the external environment and the robot itself. The suggested technique achieves robust obstacle avoidance and target searching with substantially reduced convergence time by integrating a shape-based coach method into a model-free RL strategy inside a uniform formulation. Preliminary tests verify the effectiveness of the controller.

Chapter 7

Distributed Coach-Based Bayesian Control

Deep reinforcement learning is a very active research area. Unfortunately, problems such as data inefficiency, exploration-exploitation trade-off, and multi-task learning are still very challenging. Thus, distributed RL received more and more attention recently, which could be run on many machines simultaneously. In this chapter, we regard the snake robot as a multi-agent system and investigate the issue of applying Bayes estimation into a distributed RL framework for efficient snake robot control.

A full review about distributed RL can be found in [71]. A new framework for distributed reinforcement learning called Acme is proposed by Deepmind team to increase reproducibility in RL [72]. Pawar and Maulik discussed a deep RL method of using a distributed formulation to optimize and control system parameters in [73]. Sartoretti et al. exploited A3C algorithm to provide a distributed learning method for a single articulated robot in [74]. Zhang and Cai used Double-DQN and PPO of RL to train a snake in [75]. Bing et al. proposed a perception-action coupling target tracking controller for a snake robot via RL in [76]. In [77], we proposed a coach-based RL method using a joint state representation for the whole body of a snake robot. A Bayesian control framework is set up to speed up the process of training and greatly save the convergence time. It

shows an interesting research direction to reduce the complexity and difficulty encountered in the slow convergence of RL.

In this chapter, we extend our previous centralized coach-based framework by a distributed architecture. Compared with the central solution, it adopts multiple agents, one for each snake robot link, to guide training of RL. Such a framework can greatly decrease the convergence time by using much less data. The rest of the chapter is organized as follows. Section 7.1 provides the methodology of the proposed distributed coach-based framework. Section 7.2 derives the distributed density propagation rule. Section 7.3 discusses the distributed RL agents. Experimental results are reported in Section 7.4.

7.1 Methodology of the Distributed Coach-Based Framework

The purpose of a distributed coach-based RL method is to go one step further based on the work in the previous chapter by taking the benefit of parallel computation. The practice in distributed Bayesian control in the early chapter can be transferred to fulfill this goal.

To exploit the parallel computation capability of GPU and further expedite the training process, a completely decentralized formulation is really desirable for RL-based method for snake robot control. Nevertheless, both the physical constraints among robot links and the interaction with environmental objects have to be well modeled. The uncertainty should also be treated in a stochastic way by density propagation so that optimal actions can be chosen by maximizing the expectation of long-term reward. All these ideas have been casted by a newly proposed RL architecture, adopting purely distributed and coach-based methods as follows.

7.1.1 System Structure

The system formulation has been shown in Fig. 7.1, where a six-link snake robot model is shown. Each robot link is simulated by a coach module. The dash line between two adjacent robot links represents the physical constrain. Although a meta-coach is modeled for the whole robot body, it has been analyzed by a purely distributed way as further discussed in Fig. 7.2. Moreover, we use

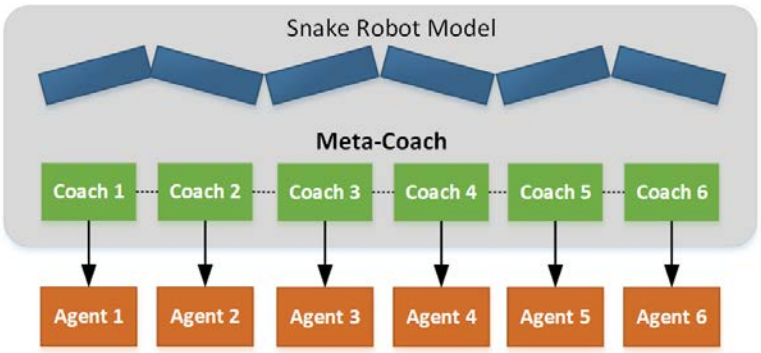

Figure 7.1. The structure of the proposed distributed framework for snake robot control using coach-based RL method, where each robot link is modeled by a coach and trained by an RL agent. The dash lines between robot links represent their physical constraints.

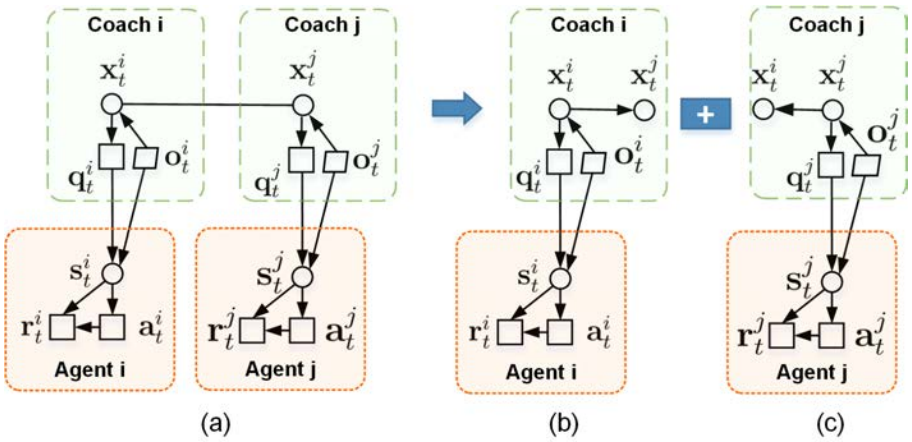

Figure 7.2. The spatial graphical modeling and decomposition with respect to robot links. **x** is the coach's hidden state, **q** the observation, **o** the external interactive objects, **s** the agent state, **a** the agent action, **r** the agent reward, t the time index. (a) The static coupled model of two adjacent robot links; (b) The decomposed model for robot link i; (c) The decomposed model for robot link j.

multiple RL agents, one for each robot link, to learn the control policy simultaneously in the training process.

The Probabilistic Graphical Model (PGM) [69] is a good tool to formulate the control problem of snake robot control. Like [78], we model the coach and agent modules by PGMs in Fig. 7.2 for two adjacent links i and j, where (a) is the coupled model, (b) and (c) the decomposed models, respectively. **x** is the coach's hidden state, **q** the observation, **o** the external interactive objects, t the time index. Similarly, **s** is the state of RL agent, **a** the corresponding action, **r** the associated reward. The undirected edge between $\mathbf{x}_t^i$ and $\mathbf{x}_t^j$ represents their physical interaction. Since we

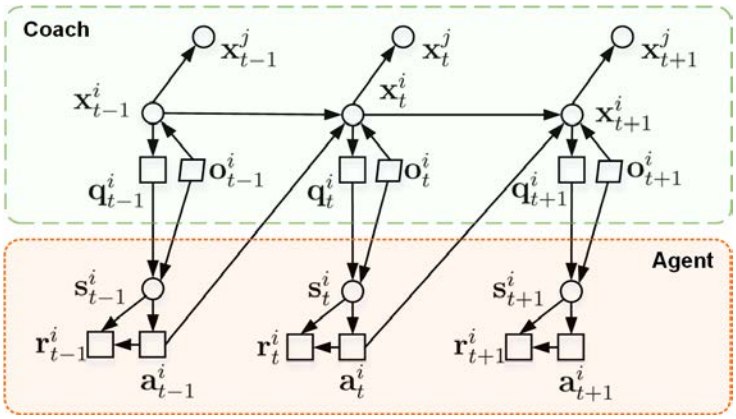

Figure 7.3. The dynamical graphical modeling for robot link i.

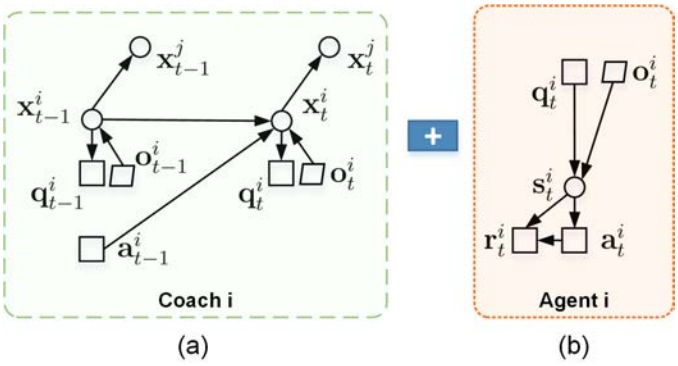

Figure 7.4. The graphical decomposition with respect to coach and agent models.

consider all agents in parallel, it can be split into two directed edges without any information loss by graphical decomposition theory [65]. In the model of robot link i, for instance, the directed edge from $\mathbf{x}_t^i$ to $\mathbf{q}_t^i$ shows the observation likelihood. The observation includes the information of the robot's direction, location coordinates, and link angles. The directed edge from $\mathbf{o}_t^i$ to $\mathbf{x}_t^i$ indicates the interaction between environmental objects to the robot link. There may be different number of obstacles around the snake robot. For simplicity, only the nearest one is chosen in this work. More complicated computational scheme could be used instead. The hidden states $\mathbf{x}_t^i$ of robot coach could be estimated partially from the observations $\mathbf{q}_t^i$. The state $\mathbf{s}_t^i$ of RL agent can be calculated from both observation $\mathbf{q}_t^i$ and interactive object $\mathbf{o}_t^i$, which are indicated by the two directed edges from $\mathbf{q}_t^i$ and $\mathbf{o}_t^i$ to $\mathbf{s}_t^i$.

In order to facilitate a sequential analysis, we further provide a dynamical graphical model for

robot link i as shown in Fig. 7.3, where three time frames are presented. In most RL methods, the action $\mathbf{a}^i_{t-1}$ is usually imported directly into the robot agent's state $\mathbf{s}^i_t$ for the next step. Differently, we connect it with the coach state $\mathbf{x}^i_t$ instead. The directed edge between two states such as $\mathbf{x}^i_t$ and $\mathbf{x}^i_{t+1}$ represents the coach dynamics. The set of all coach's states up to time t is denoted by $\mathbf{x}^i_{0:t}$, where $\mathbf{x}^i_0$ is the initialization prior. Similarly, we denote the set of all observations up to time t by $\mathbf{q}^i_{1:t}$. Correspondingly, the set of all external interactions up to time t can be denoted by $\mathbf{o}^i_{1:t}$. Moreover, the set of all agents' states up to time t is represented by $\mathbf{s}^i_{0:t}$, where the initialization prior is $\mathbf{s}^i_0$. $\mathbf{a}^i_{1:t}$ is the set of all actions up to time t, $\mathbf{r}^i_{1:t}$ the set of all rewards up to time t.

The PGM in Fig. 7.3 is very complicated for a direct analysis due to the various couplings. Therefore, we exploit the graphical decomposition [65] again with respect to coach and agent models. The result is shown in Fig. 7.4 for time t and $t-1$, where (a) is called a "*Link Coach Model*" (LCM) and (b) a "*Link Agent Model*" (LAM). No information is lost since all nodes and edges are kept comparing with the original model in Fig. 7.3. As shown in Fig. 7.4(a), the control signal $\mathbf{x}^i_t$ is first estimated and then sent to robot link's actuator after $\mathbf{x}^i_{t-1}$ and $\mathbf{a}^i_{t-1}$ are fused together with the interaction from link $\mathbf{x}^j$ at time t. After that, each robot link will have interactions with external environment, for example, attracted by targets and colliding with obstacles, which are all indicated by $\mathbf{o}^i_t$. Finally, the observation $\mathbf{q}^i_t$ is produced. During this process, the control command may not be exactly executed because of different noises and uncertainties. Thus, the agent state $\mathbf{s}_t$ has to be estimated based on both observation $\mathbf{q}^i_t$ and external objects $\mathbf{o}^i_t$ as shown in Fig. 7.4(b). As we can see, the agent model is still a model-free RL process overall where the edge between state and action represents the RL policy π_p. The reward $\mathbf{r}^i_t$ is computed based on both $\mathbf{s}^i_t$ and $\mathbf{a}^i_t$ as shown by the two directed edges in Fig. 7.4(b). In this RL process, an agent acts in a stochastic environment due to the unknown environment and system uncertainties. The control policy can be learned by minimizing a long-term cumulative cost. The graphical model adopts the Markov assumption, namely, conditional independence property. The following Markov properties can easily be verified from the decomposed graphical models in Fig. 7.4 by the theory in [69],

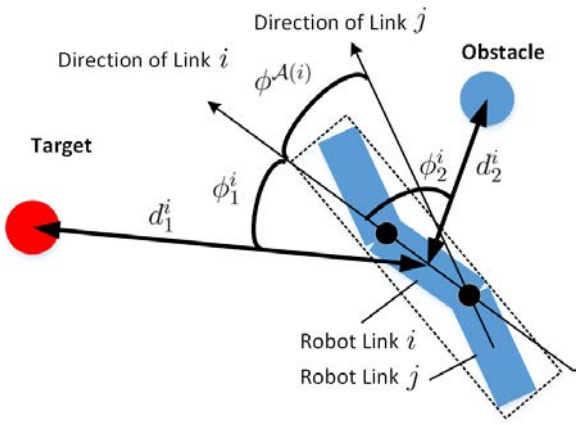

Figure 7.5. The relative pose between robot links and environmental objects.

7.1.2 Definition of State and Action Space

The serpenoid model [5] is chosen to simulate the snake gait. Correspondingly, we denote the robot's coach state as $\mathbf{x}_t^i = \langle \alpha_t^i, \gamma_t^i \rangle$ for robot link i.

A 5-tuple agent state definition is chosen to depict the relative status between the robot link i and its environmental objects as shown in Fig. 7.5. We denote

$$\mathbf{s}^i = \langle \phi_1^i, d_1^i, \phi_2^i, d_2^i, \phi^{\mathcal{A}(i)} \rangle \tag{7.1}$$

where ϕ_1^i is the angle between the snake robot link i and the navigation target as shown in Fig. 7.5. d_1^i is the normalized distance between the robot link i and the target, which is used to estimate the proximity of snake robot to the destination. ϕ_2^i indicates the angle between the link i and its nearest obstacle, where d_2^i evaluates the corresponding distance. $\phi^{\mathcal{A}(i)}$ is the angle between an adjacent link j and the robot link i where $j \in \mathcal{A}(i)$. $\mathcal{A}(i)$ is the set of all adjacent links of i where we only illustrate one adjacent link j here for simplicity. As we can see, this definition is quite different with the joint state representation in [78] which uses the whole body (dash rectangle) to calculate the relative pose between snake robot and external objects. The action is defined as $\mathbf{a}^i = \langle \Delta\alpha^i, \Delta\gamma^i \rangle$, where $\Delta\alpha^i$ is the amplitude increment and $\Delta\gamma^i$ the offset increment for the control signal in **??**. Particularly, we use a constant increment Δ_α for amplitude $\Delta\alpha \in \{-\Delta_\alpha, 0, +\Delta_\alpha\}$, and another constant increment Δ_γ for offset $\Delta\gamma \in \{-\Delta_\gamma, 0, +\Delta_\gamma\}$ similarly as [78].

7.2 The Stochastic Bayesian Derivation for Coach Model

In order to analyze the coach model in Fig. 7.4, we would like to estimate the control signal $\mathbf{x}_t^i$ recursively. Specifically, the posterior density $p(\mathbf{x}_{0:t}^i|\mathbf{a}_{1:t-1}^i, \mathbf{o}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)})$ is firstly to be derived where $\mathcal{A}(i)$ represents all indices of adjacent links of i.

$$
\begin{aligned}
& p(\mathbf{x}_{0:t}^i|\mathbf{a}_{1:t-1}^i, \mathbf{o}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)}) \\
&= \frac{p(\mathbf{x}_t^i, \mathbf{q}_{t-1}^{i,\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{x}_{0:t-1}^i, \mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)})}{p(\mathbf{a}_{1:t-1}^i, \mathbf{o}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)})} p(\mathbf{x}_{0:t-1}^i, \mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}) && (7.2) \\
&= \frac{p(\mathbf{x}_t^i, \mathbf{q}_{t-1}^i, \mathbf{q}_{t-1}^{\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)}{p(\mathbf{a}_{1:t-1}^i, \mathbf{o}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)})} p(\mathbf{x}_{0:t-1}^i, \mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}) && (7.3) \\
&= \frac{p(\mathbf{q}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_t^i, \mathbf{a}_{t-1}^i|\mathbf{x}_{t-1}^i)}{p(\mathbf{a}_{1:t-1}^i, \mathbf{o}_{1:t-1}^i, \mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)})} p(\mathbf{q}_{t-1}^{\mathcal{A}(i)}|\mathbf{x}_{t-1}^i) p(\mathbf{x}_{0:t-1}^i, \mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}) && (7.4) \\
&= \frac{p(\mathbf{q}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_t^i, \mathbf{a}_{t-1}^i|\mathbf{x}_{t-1}^i)}{p(\mathbf{q}_{t-1}^{i,\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)})} p(\mathbf{q}_{t-1}^{\mathcal{A}(i)}|\mathbf{x}_{t-1}^i) p(\mathbf{x}_{0:t-1}^i|\mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}) && (7.5) \\
&= k_c \cdot p(\mathbf{q}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_t^i, \mathbf{a}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{q}_{t-1}^{\mathcal{A}(i)}|\mathbf{x}_{t-1}^i)p(\mathbf{x}_{0:t-1}^i|\mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}). && (7.6)
\end{aligned}
$$

In eq. (7.2), we use the Bayes rule. In eq. (7.3) and eq. (7.4), the following two Markov properties from the LCM of Fig. 7.4(a) are used,

$$
\begin{aligned}
& p(\mathbf{x}_t^i, \mathbf{q}_{t-1}^{i,\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{x}_{0:t-1}^i, \mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}) \\
&= p(\mathbf{x}_t^i, \mathbf{q}_{t-1}^i, \mathbf{q}_{t-1}^{\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_t^i, \mathbf{q}_{t-1}^i, \mathbf{q}_{t-1}^{\mathcal{A}(i)}, \mathbf{a}_{t-1}^i, \mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i) \\
&= p(\mathbf{q}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{o}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{x}_t^i, \mathbf{a}_{t-1}^i|\mathbf{x}_{t-1}^i)p(\mathbf{q}_{t-1}^{\mathcal{A}(i)}|\mathbf{x}_{t-1}^i).
\end{aligned}
$$

Finally, a density updating rule from the posterior density $p(\mathbf{x}_{0:t-1}^i|\mathbf{a}_{1:t-2}^i, \mathbf{o}_{1:t-2}^i, \mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)})$ at time $t-1$ is derived in eq. (7.5). A constant k_c is further used to represent the denominator in eq. (7.6) since there is no hidden state $\mathbf{x}^i$ in it. The recursive rule in eq. (7.6) has to begin with $t \geq 3$ because it requires $p(\mathbf{x}_{0:2}^i|\mathbf{a}_1^i, \mathbf{o}_1^i, \mathbf{q}_1^{i,\mathcal{A}(i)})$ where the initial values of $\mathbf{x}_{0:2}^i$, $\mathbf{a}_1^i$, $\mathbf{o}_1^i$, and $\mathbf{q}_1^{i,\mathcal{A}(i)}$) are all predefined. This mathematically derived rule clearly describes the propagation of uncertainties in the dynamics of LCM. Compared with the framework in [78], there are two major difference: i) a completely distributed rule for each robot link instead of the whole body; ii) a newly derived density

$p(\mathbf{q}_{t-1}^{\mathcal{A}(i)}|\mathbf{x}_{t-1}^{i})$, which could be expanded further,

$$
\begin{aligned}
& p(\mathbf{q}_{t}^{\mathcal{A}(i)}|\mathbf{x}_{t}^{i}) \\
= & \prod_{j\in\mathcal{A}(i)} p(\mathbf{q}_{t}^{j}|\mathbf{x}_{t}^{i}) && (7.7)\\
= & \prod_{j\in\mathcal{A}(i)} \int p(\mathbf{q}_{t}^{j}|\mathbf{x}_{t}^{i},\mathbf{x}_{t}^{j}) p(\mathbf{x}_{t}^{j}|\mathbf{x}_{t}^{i}) d\mathbf{x}_{t}^{j} && (7.8)\\
= & \prod_{j\in\mathcal{A}(i)} \int p(\mathbf{q}_{t}^{j}|\mathbf{x}_{t}^{j}) p(\mathbf{x}_{t}^{j}|\mathbf{x}_{t}^{i}) d\mathbf{x}_{t}^{j} && (7.9)
\end{aligned}
$$

where we firstly use the Markov properties in eq. (7.7). In eq. (7.8), the state $\mathbf{x}_{t}^{j}$ appears by introducing an integral. Moreover, the Markov property that the observation is only decided by its own state $\mathbf{x}_{t}^{i}$ is adopted in eq. (7.9). By substituting eq. (7.9) into eq. (7.6), we can get,

$$
\begin{aligned}
& p(\mathbf{x}_{0:t}^{i}|\mathbf{a}_{1:t-1}^{i},\mathbf{o}_{1:t-1}^{i},\mathbf{q}_{1:t-1}^{i,\mathcal{A}(i)}) \\
= & k_c \cdot \underbrace{p(\mathbf{q}_{t-1}^{i}|\mathbf{x}_{t-1}^{i})}_{\textit{observation function}} \cdot \underbrace{p(\mathbf{o}_{t-1}^{i}|\mathbf{x}_{t-1}^{i})}_{\textit{environment interaction}} \cdot \underbrace{p(\mathbf{x}_{t}^{i},\mathbf{a}_{t-1}^{i}|\mathbf{x}_{t-1}^{i})}_{\textit{modulated dynamics}} \\
& \cdot \prod_{j\in\mathcal{A}(i)} \int \underbrace{p(\mathbf{q}_{t}^{j}|\mathbf{x}_{t}^{j})}_{\textit{observation function}} \cdot \underbrace{p(\mathbf{x}_{t}^{j}|\mathbf{x}_{t}^{i})}_{\textit{link interaction}} d\mathbf{x}_{t}^{j} \\
& \cdot p(\mathbf{x}_{0:t-1}^{i}|\mathbf{a}_{1:t-2}^{i},\mathbf{o}_{1:t-2}^{i},\mathbf{q}_{1:t-2}^{i,\mathcal{A}(i)}), && (7.10)
\end{aligned}
$$

where four different kinds of probability distributions are important to calculate the posterior density: the observation functions for both link i and its adjacent links, the environment interaction, the modulated dynamics, and the link interaction. A number of various density estimation techniques can be exploited to simulate these distributions. Here we give some paradigms.

7.2.1 Observation Function

Both $p(\mathbf{q}_{t-1}^{i}|\mathbf{x}_{t-1}^{i})$ and $p(\mathbf{q}_{t}^{j}|\mathbf{x}_{t}^{j})$ are the likelihood between a coach state i or j to its associated observation. Similar to [78], the BNN-based learning method [30] is chosen to approximate these two functions by training data.

7.2.2 Environment Interaction

The density $p(\mathbf{o}^i_{t-1}|\mathbf{x}^i_{t-1})$ simulate the environmental interaction between the state $\mathbf{x}^i_{t-1}$ of robot link i and the external object $\mathbf{o}^i_{t-1}$, which could be stimulated by obstacles or targets. The BNN-based algorithm in [30] is adopted to learn this function as well from training data.

7.2.3 Modulated Dynamics

Similar to [78], the modulated dynamics can be modeled as a multi-variate Gaussian distribution,

$$\begin{aligned} & p(\mathbf{x}^i_t, \mathbf{a}^i_{t-1}|\mathbf{x}^i_{t-1}) \\ = & \frac{1}{2\pi|\Sigma_d|^{1/2}}\exp\Big\{-\frac{1}{2}(\mathbf{x}^i_t-\mu)^T\Sigma_d^{-1}((\mathbf{x}^i_t-\mu)\Big\} \end{aligned} \tag{7.11}$$

where μ is the multi-variate mean, Σ_d the multi-variate covariance as follows,

$$\mu = \begin{bmatrix} \alpha_{t-1} + k_\alpha \Delta\alpha_t \\ \gamma_{t-1} + k_\gamma \Delta\gamma_t \end{bmatrix}, \quad \Sigma_d = \begin{bmatrix} \sigma_\alpha^2 & 0 \\ 0 & \sigma_\gamma^2 \end{bmatrix},$$

As we can see, k_α and k_γ are all coefficients where only the coach itself works when $k_\alpha = 0$ and $k_\gamma = 0$. When k_α or k_γ is getting larger, the agent action becomes more critical in the learning procedure.

7.2.4 Link Interaction

The density $p(\mathbf{x}^j_t|\mathbf{x}^i_t)$ models the physical constraints between robot link state $\mathbf{x}^j_t$ and $\mathbf{x}^i_t$. It can also be learned by the BNN-based learning algorithm [30] with collected training data.

7.3 Learning Details for the Agent Model

After the posterior $p(\mathbf{x}^i_{0:t}|\mathbf{q}^i_{1:t-1}, \mathbf{a}^i_{1:t-1}, \mathbf{o}^i_{1:t-1})$ is estimated, the robot control signal can be calculated and sent to the actuator of robot link i. By interacting with the surroundings, the observation

$\mathbf{q}_t^i$ could be measured for the snake robot. As can be seen in Fig. 7.4(a), the two directed edges from $\mathbf{o}_t^i$ to $\mathbf{x}_t^i$ and $\mathbf{x}_t^i$ to $\mathbf{q}_t^i$ represent these processes, respectively. By the definition of eq. (7.1), the agent state can be calculated and then used for the RL training.

7.3.1 The Network for Reinforcement Learning

The A2C-based RL method [34] is chosen as a paradigm in our implementation for a better comparison with other methods. For the network architecture, we use two deep neural networks for the RL training process. Specifically, one is used as Actor to estimate the the stochastic polity function and another plays as Critic to estimate the value function, respectively. For the Actor network, we have chosen two hidden layers. However, for Critic network, only one hidden layer is selected. As a common practice, the ReLU function is introduced in order to get non-linearity after each hidden layer.

A four tuple $\langle \mathbf{S}^i, \mathbf{A}^i, \mathbf{R}^i, \nu^i \rangle$ is used to denote the RL training process. In particular, $\mathbf{S}^i$ represents the set of agent states of link i. $\mathbf{A}^i$ is the set of all actions for robot link i. $\mathbf{R}(\mathbf{s}_t^i, \mathbf{a}_t^i)$ is the corresponding reward. ν^i is the discount factor in the RL. The optimal policy $\pi_p^i : \mathbf{S}^i \rightarrow \mathbf{A}^i$ is learned by gradient descent with an entropy-based loss function,

$$\begin{aligned} f_{\pi_p^i}(\mathbf{w}_{\mathbf{A}}^i) = & \log\Big\{\pi_p^i(\mathbf{a}_t^i|\mathbf{s}_t^i; \mathbf{w}_{\mathbf{A}}^i)\Big\}\Big\{\mathbf{R}_t^i - \mathbf{V}^i(\mathbf{s}_t^i; \mathbf{w}_{\mathbf{A}}^i)\Big\} \\ & + \kappa \cdot \mathbf{H}\Big\{\pi_p^i(\mathbf{s}_t^i; \mathbf{w}_{\mathbf{A}}^i)\Big\}. \end{aligned} \tag{7.12}$$

where $\mathbf{H}$ is the entropy factor used to encourage exploration, $\mathbf{w}_{\mathbf{A}}^i$ the network weights, κ the constant adjust the tradeoff between exploration and exploitation. The policy value $\mathbf{V}^i$ is defined by

$$\mathbf{V}^i = E\left(\sum_{t=0}^{\infty} \nu_t^i \mathbf{R}(\mathbf{s}_t^i, \mathbf{a}_t^i)\right), \tag{7.13}$$

where the objective is to find out the optimal policy π_p^* having a maximum value.

7.3.2 The Design of Reward Function

For a better comparison, the same reward function has been used for all link agents similar as [78],

$$\mathbf{r}_t^i = r_1^i + r_2^i + r_3^i \tag{7.14}$$

where

$$r_1^i = -k_1, r_2^i = \begin{cases} \frac{-k_2}{d_2^i+1}, & d_2^i \leq \text{th}_2 \\ 0, & d_2^i > \text{th}_2 \end{cases}, r_3^i = \begin{cases} k_3^i, & d_1^i \leq \text{th}_1 \\ 0, & d_1^i > \text{th}_1. \end{cases}$$

Specifically, the reward r_1^i is a punishment factor to reduce the movement and push the snake robot to approach the target quickly. The factor r_2^i is used to decrease collisions by keeping distance from obstacles. The third factor is a final reward when reaching the target.

For the parameters, we empirically set $k_1 = 0.01$, $k_2 = 0.1$, $k_3 = 20$, $\text{th}_1 = 5$, $\text{th}_2 = 50$.

7.3.3 Learning Details

As talked in the previous chapter, we found that a commutative scheme can make the trained policy more stable because it can achieve a good balance between exploration and exploitation. Therefore, in this work, we still make the RL agent take turns with and without the coach model.

7.4 Experimental Results

The proposed Distributed Coach-based RL (DCRL) method has been tested by both simulations and real-world data with comparisons of Model-Free RL (MFRL) [3], the Centralized Coach-based RL (CCRL) [78], and Distributed Learning of Decentralized Control (DLDC) method [4].

7.4.1 Simulation Analysis

V-REP [36] is used as the simulation platform for experimental verification. The ACM-R5 snake robot is exploited. We varied the number of links from 5 to 12 during the tests. During the

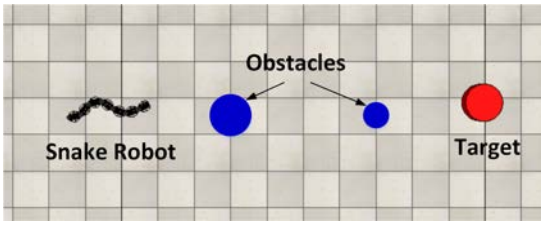

Figure 7.6. The training scenario adopted in our implementation. It has one target and two obstacles with different sizes.

navigation, the relative status between each robot link and environmental objects is first dynamically estimated. After that, the learned policy for each snake robot link can generate an action based on the input, which can be further used to compute the desirable actuator angle. Although all actuator joints are controlled independently, they are inherently correlated by two ways: 1) the serpenoid model guarantees the phase collaboration among robot links which keeps the predominant gait of snake robot; 2) the interaction between adjacent links has been not only embedded inside the learned policies during training process but also triggered by the same external objects.

The training scenario is shown in Fig. 7.6 where two obstacles with different sizes and one target are used. Compared with the one target and one obstacle training scene in [78], we introduce an additional obstacle in order to cover more abundant interaction situations. In our implementation, we found that the previous training scenario could not work well for the proposed distributed framework. In our new formulation, each robot link learns its own policy. If lacking enough interaction cases, the learned policy will be biased although it can make a convergence. In other words, the performance of inference will be not stable if the training scene is too simple. The policy may fail when facing unlearned situations. This new scene would slow down the training speed than before as expected. However, the sacrifice of additional computation is not only necessary but also worthy.

Table 7.1. Comparison of Training Performance*

Item	MFRL	DLDC	CCRL	DCRL
Episodes	1.6 ~ 2.4k	15 ~ 19k	193 ~ 354	136 ~ 238
Time (min)	350 ~ 490	880 ~ 950	52 ~ 61	26 ~ 44

* The performance was achieved by using the ACM-R5 snake robot with 8 links.

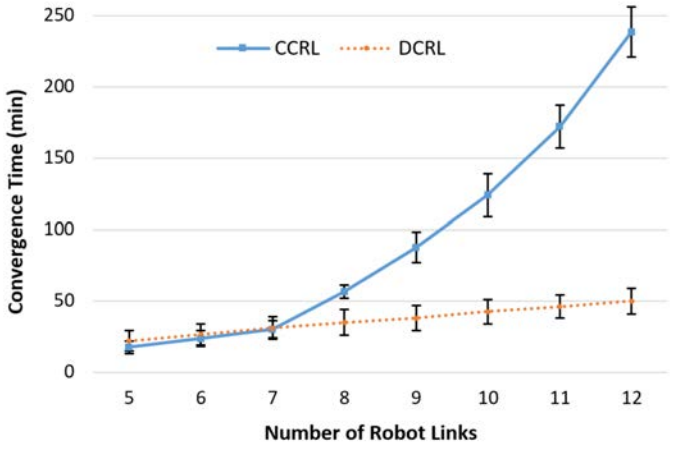

Figure 7.7. Performance comparison of training speed using CCRL and DCRL.

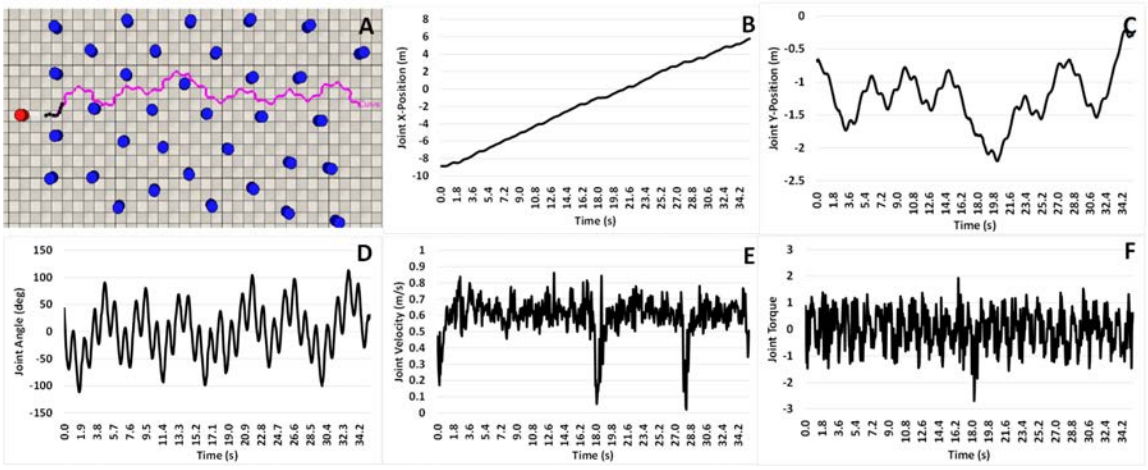

Figure 7.8. Simulation results using the proposed DCRL on the scene of `PEGGY-ARRAY`.

Table 7.1 presents the comparison of training performance using different methods. The results were achieved by using a snake robot with 8 links. As we can see, DLDC is most data starving and time consuming in order to achieve a relative robust convergence.

Due to using a more complicated reward design, MFRL converges quickly. The efficient training achieved by CCRL is because of using the coach model. The proposed DCRL achieved the best results in terms of a least episode demand and a fastest convergence time benefiting from the distributed formulation.

In order to give a deeper investigation of the newly proposed DCRL comparing with its cen-

tralized counter party, we varied the number of robot links and tested their training performance as shown in Fig. 7.7. For each setup, we tested five times and calculated the averaging time as well as the variance. As can be seen, the convergence speeds are quite similar when the body length is short with a limited number of robot links. CCRL is even better than DCRL when the number of links is small, say less than seven. However, when link number gets larger, the convergence time of CCRL slows down dramatically than DCRL due to the exponentially increased computational complexity by the centralized state space. However, the training speed of DCRL only varies linearly in terms of the link number because of the smaller state space and the parallel computing. This shows the great advantage of such a distributed solution especially when the snake robot has a very long body length.

Table 7.2. Quantitative Comparisons of Performance*

Performance	MFRL	DLDC	CCRL	DCRL
Time Used (s)	63.97	64.71	24.14	25.66
Collisions	23	15	10	4

* The performance was achieved by the ACM-R5 snake robot with 10 links.

For the inference performance, we first tested the learned policy of DCRL in a terrain called `PEG-ARRAY` which can be seen in Fig. 7.8A, where a target and obstacles are unknown in advance and only dynamically estimated during the locomotion. It is challenging due to the frequent interactions among robot links as well as with environments. Although using multiple independent controllers, one for each link, during the navigation, the proposed DCRL achieved robust performance in tolerating these various uncertainties inside robot itself and from sensor noise by the proposed collaborative-training-decentralized-execution scheme. Fig. 7.8 shows the experimental results, where A shows the overall movement trajectory, B and C the joint position, D the joint angle, E the joint speed, and F the joint torque. As we can see, the snake robot can successfully pass through the peg array efficiently and robustly with a smooth trace. The serpenoid gait is well kept as shown in Fig. 7.8D. Very few collisions happen as indicated by the spikes in 7.8E and F.

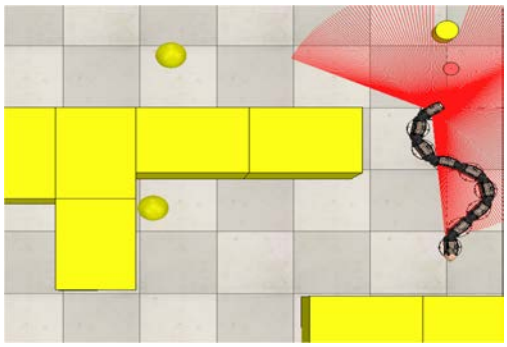

Figure 7.9. Results on complex data CORRIDOR. It has many crowed obstacles, some of which are movable.

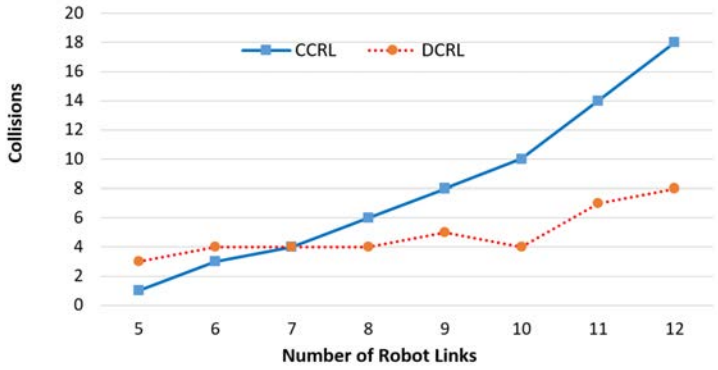

Figure 7.10. Performance comparison of CCRL and DCRL during the navigation process on CORRIDOR.

We have compared the quantitative performance with different methods in detail. Table 7.2 gives the analysis of both averaging routing time and collisions of five runs using each control method. As we can see, both MFRL and DLDC spent a much longer period due to the detours after collisions. Comparing with CCRL, DCRL achieved a more robust performance with even less collisions and comparable routing speed. This is mainly because the independent controller of each robot link provides more flexibility to adjust its local shape.

One more scenario CORRIDOR shown in Fig. 7.9 is further used to demonstrate the effectiveness of generalization and make a thorough comparison with CCRL like [78]. It is more challenging due to the narrow space and small movable obstacles. We varied the number of snake robot links and tested the robustness of learned policy in inference. Fig. 7.10 shows the performance comparison using the CCRL and DCRL.

Commonly, it is very difficult to achieve a collision-free locomotion in unknown scene with

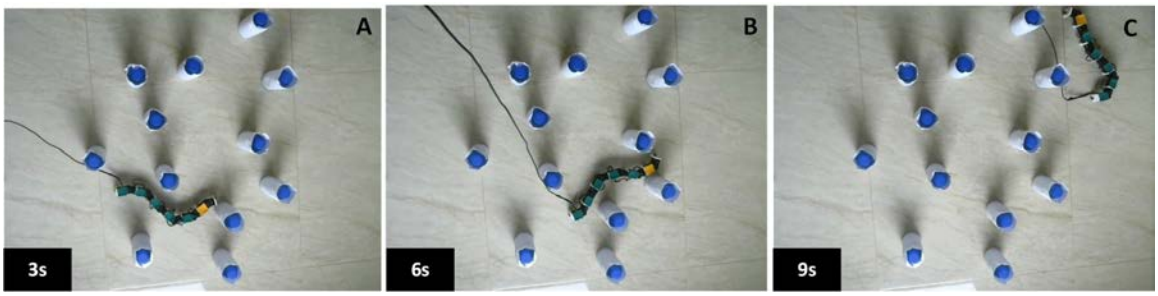

Figure 7.11. Real-world experimental results of DCRL method.

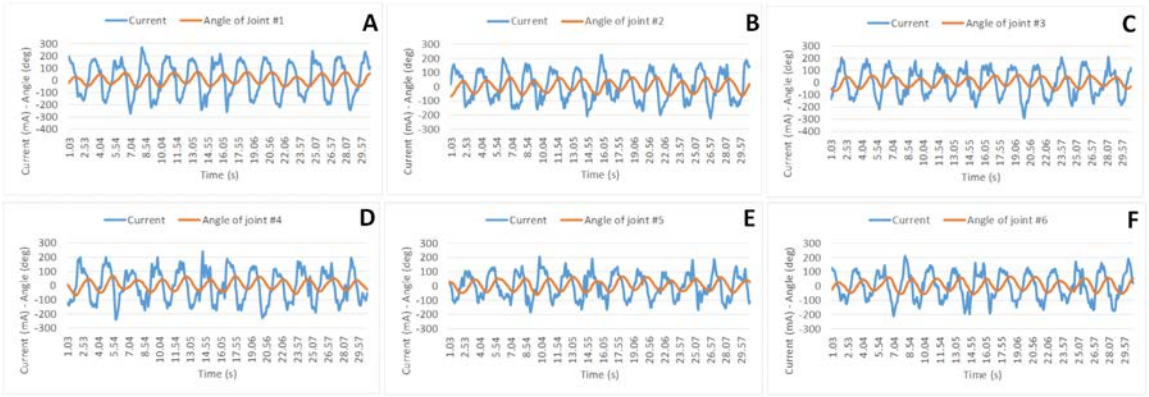

Figure 7.12. The quantitative result for all snake robot links using the proposed DCRL approach.

many obstacles, some of which are even shiftable. Sometimes, suitable collisions may be helpful to aid the movement of the snake robot. However, hard and/or frequent collisions are definitely undesirable. The goal of this work is to design schemes for obstacle avoidance. As we can see, there is not much difference when the number of robot links is small. When it is beyond seven, the collisions of CCRL get much more than DCRL. This is mainly because the joint state representation makes the complexity increase exponentially in terms of the number of robot links. One central controller is difficult to coordinate a large number of robot links and fit to complicated environment changes. On the contrary, DCRL doesn't suffer from this problem and can keep relatively stable performance even when the robot length is getting large.

7.4.2 Experimental Results in the Real World

The physical snake robot *JAW*-I was used to verify the proposed DCRL in real-world. A testing scenario having one shiftable target and a number of bottle-like obstacles are adopted to simulate cluttered environments. For the training process, we firstly transferred the policy learned from simulation and then re-trained it in the real-world since the physical parameters of snake robots are different in these two situations. However, we find that the training policy is mostly sensitive to the number of robot links but not the configuration parameters. In other words, not too much additional training, say 20-30 episodes, can guarantee a relatively stable performance for the new environment in our implementation. More than 15 trials were tested to demonstrate the performance where we found the proposed DCRL could efficiently avoid the obstacles and reach the target in this challenging scene. Compared with CCRL, it can adjust the local shape more flexible by the multiple independent controller, one for each robot link. With the serpenoid model and the collaborative training by multiple coaches, the gait could be kept very well as shown in Fig. 7.11, where some snapshots are given. In Fig. 7.12, the measured results are provided for the joint currents and angles. As we can see, the curves are relatively smooth and rhythmical as expected, which demonstrates the robust performance of the proposed DCRL.

7.5 Summary

In this chapter, we have proposed a distributed Bayesian controller for snake robots. Specifically, it combines multiple coach models, one for each robot link, with multiple agents within a unified reinforcement learning framework. Compared with centralized solutions, it greatly reduces the computational complexity and expedites the training convergence time when the link number of snake robot increases. By the proposed collaborative-training-decentralized-inference scheme, the interactions both inside robot links and with environmental objects are well learned and handled during the locomotion.

The mathematically derived distributed Bayesian propagation rule explicitly models the different uncertainties inside the snake robot and with the surroundings within a consistent stochastic

framework. We have also done detailed experiments to demonstrate the performance comparing with state-of-the-art.

Chapter 8

Discussion of the Methods

8.1 Characteristics of Topological Architectures

We have presented five Bayesian control methods for snake robot locomotion in unstructured scenarios. From the perspective of topological structure, they can be classified into three categories: centralized vs decoupled vs distributed as shown in Fig. 8.1.

Chapter 3 and 6 adopt a centralized controller by using a joint state representation for the snake robot. Such an architecture is the most intuitive and easy one to be selected. In this design, all links can collaborate together as a whole by one controller where each agent of the robot link sends its observation to a master and receives the command back as can be seen in Fig. 8.1(a). The large

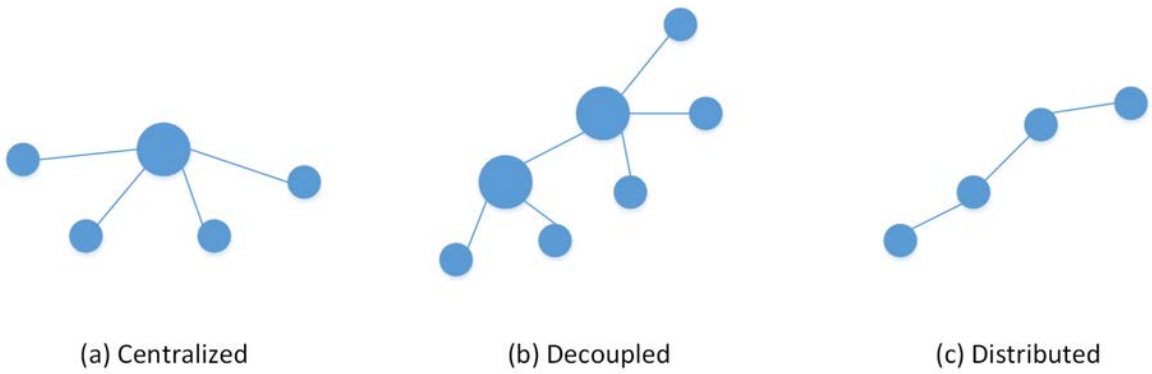

Figure 8.1. The topological structures of centralized, decoupled, and distributed systems.

circle node represents the master controller while each small node is a slave agent controlling its corresponding robot link actuator. Such kind of centralized systems has many advantages. For example, it is usually easy to develop and quick to update. Moreover, it is commonly cost-efficient especially for small systems up to a certain limit where nodes can be easily detached. On the contrary, its limitations are obvious too. Such an architecture is generally very sensitive to the central node whose failure tends to cause the entire system to fail. Only vertical scaling on the central master is possible while horizontal scaling contradicts the single central unit characteristics of this system.

Chapter 4 chooses a decoupled way which doesn't have one central owner as shown in Fig. 8.1(b). Instead, there are multiple grouping owners which can listen for connections from other nodes. Every group of link nodes of the snake robot can make their own decision. However, the final behavior of the system is the aggregate of the decisions of the individual ones. Compared with centralized systems, such an architecture shares advantages of high availability and more autonomy where some nodes can always be available without bottleneck situations. One link's error would not make the whole controller fail. However, this mechanism shares a disadvantage where the additional collaboration cost has been introduced. It is usually not beneficial to build and operate small decoupled systems because of the low cost ratio.

Chapter 5 and 7 presented methods using a distributed system. It is similar to a decoupled one in that it doesn't have a single central controller. By going a step further, it eliminates the centralization where every node makes its own decision based on the local information as shown in Fig. 8.1(c). The significant point is that each node has to pick a place to discover after negotiation with the other nodes in its communication range. For a distributed system, nodes fail independently without having a severe effect on the entire system, which increases the system's robustness. Both horizontal and vertical scaling is possible. But it also has limitations as well. Usually, such kind of systems are difficult to design. It may be difficult for a node to get a global view of the system and hence take informed decisions based on the states of other nodes. In other words, it may be difficult for the whole system to achieve consensus without a specifical design.

While all these systems can function effectively, some are more stable and efficient than others

by different designs in various situations. Either way, they face the same challenges: fault tolerance, maintenance costs, and scalability etc. In Table 8.1, we give a comparison of the above issues. Overall, with a small number of robot links, say less than 5, centralized methods usually is a good choice considering the development cost. With the increase of link amount, say between 6 to 10, the decoupled scheme shows its advantage in achieving a tradeoff among system complexity, robustness, and efficiency. When the link number of snake robot is large, the distributed architecture will dominate the performance.

Table 8.1. Comparison of centralized, decoupled, and distributed architectures

System Structure	Proposed Method	Development Convenience	Scalability	Maintenance Cost	Fault Tolerance
Centralized	CenBC, CCRL	H	L	L	L
Decoupled	DecBC	M	M	M	M
Distributed	DisBC, DCRL	L	H	H	H

* H denotes high, M medium, L low; CenBC represents centralized Bayesian controller; DecBC the decoupled Bayesian controller, DisBC the distributed Bayesian controller, CCRL the centralized coach based reinforcement learning, DCRL the distributed coach based reinforcement learning.

8.2 Complexity Analysis

The complexity of the proposed methods are evaluated. Since the resampling is the most time consuming part for Bayesian density estimation, we provide a comparison of different resampling algorithms using these methods. However, for the memory requirement, we consider not only for resampling, but also for the complete process.

As we can see in Table 8.2, CenBC is the most complex method due to using the joint state representation and a centralized controller. DecBC has less computational complex in the resampling because of only considering the interacted environmental objects and robot link. DisBC further de-

Table 8.2. Complexity comparison for the proposed methods

Method	Computational Complexity	Overall Memory Requirements
CenBC	$O(N_s KL)^{\dagger}$	$O(N_s^2)$
DecBC	$O(N_s \mathcal{K} \mathcal{L})^{\ddagger}$	$O(N_s^2)$
DisBC	$O(N_s K)^{*}$	$O(N_s^2 L)$
CCRL	$O(N_s KL)$	$O(N_s^2)$
DCRL	$O(N_s K)^{*}$	$O(N_s^2 L)$

† K is the max number of detected environmental objects, L the total number of links of the snake robot.

‡ $\mathcal{K}$ is the number of interacted environmental objects, which is much less than K; $\mathcal{L}$ is the number of interacted links, which is usually less than L too.

* complexity for one worker.

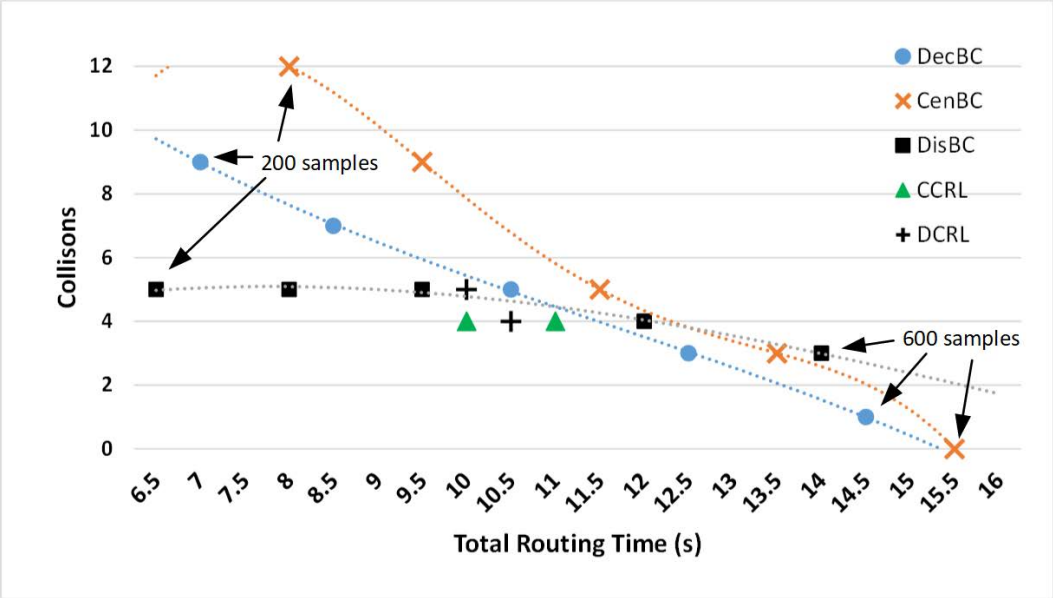

Figure 8.2. Performance comparison of the proposed methods.

creases the complexity for each work with a parallel computing scheme. However, it sacrifices the overall memory requirements. Both two RL based methods CCRL and DCRL actually have additional complexity beyond the listed for training process due to reinforcement learning. Nevertheless, they are commonly more efficient in the inference phase than most SMC based methods. Compared with CCRL, DCRL has less computational complexity in the resampling part during training phase. However, it needs more memory requirements because of the parallel computing.

8.3 Performance Comparison

In order to compare the performance of the proposed five methods, we use the same scenario in Fig. 7.9. Five tests for each approach were implemented by assigning a total number of samples from 200 to 600, and adjusting parameters. Although all of them could successfully pass through the corridor in real-time and reach the target finally, their navigation efficiency and robustness varied as presented in Fig. 8.2. As we can see, CenBC achieved a collision-free locomotion when the number of samples is large enough. However, when only having a small amount of samples, its robustness decreases dramatically than others. On the contrary, DisBC is not as sensitive as CenBC to the

number of samples, whose curve is relatively flat. DecBC achieved a tradeoff performance between CenBC and DisBC. The performance of both CCRL and DCRL is independent with samples in the inference phase as can be seen. The robustness of CCRL is slightly better than DCRL because we used the same reward functions and provided same number of training episodes for the two methods. However, as compared in Chapter 7, DCRL can greatly expedite the training speed than CCRL by exploiting parallel computing.

Chapter 9

Conclusions and Future Work

9.1 Contributions

In this dissertation, we have studied issues related to snake robot control. We have proposed several novel approaches for real-time stochastic control using Bayesian methods.

We have presented a stochastic centralized Bayesian-based controller for snake robots by modeling the interaction with environmental objects using probability density propagation. Preliminary experimental results have demonstrated promising performance in unstructured circumstances. For the future work, we would like to investigate more sophisticated interaction models and extend the proposed framework to more problems such as simultaneous localization and mapping.

Moreover, we have proposed a dynamically decoupled Bayesian framework for snake robot control in changing surroundings. The interactions between the robot and environment are simulated by a Bayesian dynamic graphical model. Benefiting from the decoupling mechanism, the proposed probabilistic propagation formulation provides an innovative way to model the uncertainty during locomotion in cluttered terrain. Preliminary experimental results have demonstrated promising performances of the proposed approach for challenging unstructured scenarios.

Furthermore, we have given an interactively distributed Bayesian controller for snake robots by modeling the inter-module interaction and the external virtual force with environmental ob-

jects using a probabilistic density propagation formulation. Preliminary experimental results have demonstrated promising performance on real world data.

Finally, two coach-based Bayesian controllers have been presented for snake robots using reinforcement learning. The uncertainty lying in both the system and environment is modeled by a stochastic process with probabilistic density propagation. By combining two shape-based coach methods, one centralized and one distributed, and a model-free RL approach within a unified formulation, the proposed methods can achieve robust target searching and obstacle avoidance with dramatically saved convergence time. Preliminary experimental results have shown promising performance.

9.2 Obtained Results and Conclusions

The proposed approaches have been tested both on simulation and real-world experiments. The control performance have been carefully analyzed qualitatively and quantitatively comparing with state-of-the-art methods. Through the reported results, the following conclusions can be obtained:

(1) We have verified that Bayes estimation is an effective method to handle the uncertainties in stochastic snake robot control. Most existing methods of snake robot control are still deterministic. Uncertainty is inherent and lies in almost everywhere due to sensor noise, input disturbance, discrepancy between control signal and mechanical actuation, vagueness, and model incompleteness etc. By applying the Bayesian density propagation and estimation, the proposed framework can greatly improve the control robustness and efficiency.

(2) We have studied different architectures in modeling the high redundant structure, complex dynamics, and the complicated interaction of snake robot control. Overall, the centralized solution is more accurate theoretically since it exploits a joint state representation and considers the snake robot as a whole. Without considering the computation complexity or when the number of robot links is small, it is good choice because of the relatively simple controller design and the easier deployment requirement. Decoupled solution is more flexible since it considers both high-level global behavior and low-level local interactions simultaneously. However, additional design and computational cost

have to be paid. Such a method is suitable when frequent collisions happen during the snake robot's navigation. Finally, the proposed distributed solution is more computationally efficient. It can exploit GPU's power to achieve fast implementation by parallel computing. Meanwhile, it is more proper for the situation where the body length is very long. Every coin has its two sides. Usually, there is no perfect solution but the most suitable one, all depending on the conditions. Each kind of architectures has its own advantages and disadvantages.

(3) We have shown that coach-guided RL is a promising framework in solving snake robot control with state-of-the art AI technologies. Compared with the conventional "supervised" model-based snake robot control methods, RL-based approaches do not need too much prior knowledge and can learn intelligence by robot itself from behaviors. Therefore, it provides a more natural way to design controllers for practical real-world applications. The proposed coach-based solutions can effectively speed up the training convergence and thus remove one of the largest barriers to realizing the true potential of RL for snake robot control.

9.3 Future Work

A number of promising avenues of research are suggested by the work presented in this thesis, some of which are detailed as follows:

• **Decoupled Bayesian Control.** (1) More efficient interaction models are under investigation to deal with various challenging obstacles; (2) Although BNN-based supervised learning method has shown its power for density estimation in our implementation, it still suffers from the bias problem sometimes due to the quality of training data. We plan to apply unsupervised learning methods and further improve the performance; (3) Other density estimation methods could be exploited to simulate the proposed framework for a better comparison; 4) More sensors can be tested to give a detailed observation of the environment, for instance, embedding a camera on each side of the snake robot module and getting a panoramic view of the surroundings.

• **Distributed Bayesian Control.** (1) Although the basic performance of DBC has been verified by the real-world experiments since we think they are better than simulations to prove the method, a

deeper investigation through various simulations is still desired to further validate the effectiveness of the proposed method; (2) State-of-the-art machine learning techniques may have potentials to be exploited for training better densities in the framework; (3) More sophisticated interaction models are needed for different types of irregular obstacles.

• **Coach-Based Bayesian Control.** (1) More sample efficient learning algorithm are still under investigation to further improve the training efficiency; (2) The sensitivity of state dimensionality with control performance is an interesting research topic for future study; (3) Different types of reinforcement learning algorithms could be applied to get better performance, for example, multi-agent hierarchical RL learning; (4) The current implementation trains one policy for multiple agents, one for each snake robot module, distributed during the locomotion. The interactions are not modeled during the inference. We plan to further investigate more sophisticated reward function including such kind of information and may give each module a different policy for a better control performance in the future.

Bibliography

[1] J. Whittaker, *Graphical Models in Applied Multivariate Statistics*. Wiley, 1990.

[2] Y. Jia and S. Ma, "A Bayesian-based controller for snake robot locomotion in unstructured environments," in *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[4] G. Sartoretti, W. Paivine, Y. Shi, Y. Wu, and H. Choset, "Distributed learning of decentralized control policies for articulated mobile robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, 2019.

[5] S. Hirose, P. Cave, and C. Goulden, *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators*. Oxford University Press, 1993.

[6] J. Gray, "The mechanism of locomotion in snakes." *The Journal of experimental biology*, vol. 23 2, pp. 101–20, 1946.

[7] K. Wang, W. Gao, and S. Ma, "Snake-like robot with fusion gait for high environmental adaptability: Design, modeling, and experiment," *Applied Sciences*, vol. 7, p. 1133, 2017.

[8] B. Jayne, "What defines different modes of snake locomotion?" *Integrative and Comparative Biology*, vol. 60, pp. 156 – 170, 2020.

[9] S. Ma, "Analysis of snake movement forms for realization of snake-like robots," *Proceedings 1999 IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3007–3013, 1999.

[10] M. Travers, P. Schiebel, C. Gong, H. Astley, D. Goldman, and H. Choset, "Shape-constrained whole-body adaptivity," in *Proceedings of Symposium on Adaptive Motion of Animals and Machines*, June 2015.

[11] J. Liu, Y. Tong, and J. Liu, "Review of snake robots in constrained environments," *Robotics Auton. Syst.*, vol. 141, p. 103785, 2021.

[12] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A review on modelling, implementation, and control of snake robots," *Robotics and Autonomous Systems*, vol. 60, pp. 29–40, 2012.

[13] M. Tanaka and K. Tanaka, "Shape control of a snake robot with joint limit and self-collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, pp. 1441–1448, 2017.

[14] J. Whitman, F. Ruscelli, M. J. Travers, and H. Choset, "Shape-based compliant control with variable coordination centralization on a snake robot," *IEEE Conf. on Decision and Control (CDC)*, pp. 5165–5170, 2016.

[15] M. J. Travers, J. Whitman, and H. Choset, "Shape-based coordination in locomotion control," *The International Journal of Robotics Research*, vol. 37, pp. 1253 – 1268, 2018.

[16] M. Rafieisakhaei, S. Chakravorty, and P. R. Kumar, "On the use of the observability gramian for partially observed robotic path planning problems," *2017 IEEE 56th Annual Conference on Decision and Control*, pp. 1523–1528, 2017.

[17] S. Gamse, F. Nobakht-Ersi, and M. A. Sharifi, "Statistical process control of a kalman filter model," *Sensors*, vol. 14, no. 10, pp. 18 053–18 074, 2014.

[18] Y. Pei, S. Biswas, D. S. Fussell, and K. Pingali, "An elementary introduction to kalman filtering," *Commun. ACM*, vol. 62, pp. 122–133, 2017.

[19] D. Rollinson, H. Choset, and S. Tully, "Robust state estimation with redundant proprioceptive sensors," in *ASME 2013 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, 2013.

[20] C. Stachniss and W. Burgard, "Particle filters for robot navigation," *Foundations and Trends in Robotics*, vol. 3, pp. 211–282, 2013.

[21] R. Deventer, J. Denzler, and H. Niemann, "Control of dynamic systems using bayesian networks," in *Proceedings of the IBERAMIA/SBIA*, 2000, pp. 33–39.

[22] J. Wang, J. yang Liu, S. kun Pang, and H. Yi, "Dynamic bayesian network controller design and its application on ship autopilot," 2017.

[23] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using bayesian networks to analyze expression data," *Journal of computational biology*, vol. 7, no. 3-4, pp. 601–620, 2000.

[24] K. Sachs, O. Perez, D. Pe'er, D. A. Lauffenburger, and G. P. Nolan, "Causal protein-signaling networks derived from multiparameter single-cell data," *Science*, vol. 308, no. 5721, pp. 523–529, 2005.

[25] B. J. v. d. Z. Ajith Abraham, *Innovations in Intelligent Systems (Studies in Fuzziness and Soft Computing)*. Springer, 2004.

[26] P. J. Green, A. Seheult, and B. Silverman, "Density estimation for statistics and data analysis." *Applied statistics*, vol. 37, p. 120, 1988.

[27] S.-T. Tseng, H.-T. Chiang, and J. Lehnert, "Parametric density estimation using em algorithm for collaborative spectrum sensing," *2008 3rd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom 2008)*, pp. 1–6, 2008.

[28] D. Nott, S. L. Tan, M. Villani, and R. Kohn, "Regression density estimation with variational methods and stochastic approximation," *Journal of Computational and Graphical Statistics*, vol. 21, pp. 797 – 820, 2012.

[29] P. L'Ecuyer, F. Puchhammer, and A. B. Abdellah, "Monte carlo and quasi-monte carlo density estimation via conditioning." *arXiv: Statistics Theory*, 2019.

[30] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *The International Conference on Machine Learning*, 2016, pp. 1050–1059.

[31] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Baysian tracking," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[32] A. Doucet, S. J. Godsill, and C. Andrieu, "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statist. Comput.*, vol. 10, pp. 197–208, 2000.

[33] N. Sünderhauf, O. Brock, W. J. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and P. I. Corke, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, pp. 405 – 420, 2018.

[34] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[35] G. Sartoretti, Y. Shi, W. Paivine, M. Travers, and H. Choset, "Distributed learning for the decentralized control of articulated mobile robots," in *IEEE Int'l Conf. on Robotics and Automation*, 2018, pp. 3789–3794.

[36] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1321–1326.

[37] A. Kakogawa, S. Jeon, and S. Ma, "Stiffness design of a resonance-based planar snake robot with parallel elastic actuators," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1284–1291, 2018.

[38] M. Najibi, M. Rastegari, and L. S. Davis, "G-CNN: An iterative grid based object detector," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2369–2377, 2016.

[39] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[40] A. A. Transeth, R. I. Leine, C. Glocker, K. Y. Pettersen, and P. Liljebäck, "Snake robot obstacle-aided locomotion: Modeling, simulations, and experiments," *IEEE Transactions on Robotics*, vol. 24, pp. 88–104, 2008.

[41] X. Wu and S. Ma, "Neurally controlled steering for collision-free behavior of a snake robot," *IEEE Transactions on Control Systems Technology*, vol. 21, pp. 2443–2449, 2013.

[42] A. Gelman and X.-L. Meng, "Simulating normalizing constants: From importance sampling to bridge sampling to path sampling," *Statistical Science*, vol. 13, no. 2, pp. 163–185, 1998.

[43] E. Kelasidi, K. Y. Pettersen, and J. T. Gravdahl, "A waypoint guidance strategy for underwater snake robots," *22nd Mediterranean Conference on Control and Automation*, pp. 1512–1519, 2014.

[44] Z. Cao, D. Zhang, and M. Zhou, "Direction control and adaptive path following of 3-d snake-like robot motion," *IEEE Transactions on Cybernetics*, pp. 1–8, 2021.

[45] T. Kano, T. Sato, R. Kobayashi, and A. Ishiguro, "Local reflexive mechanisms essential for snakes' scaffold-based locomotion." *Bioinspiration biomimetics*, vol. 7, no. 4, p. 046008, 2012.

[46] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "Hybrid modelling and control of obstacle-aided snake robot locomotion," *IEEE Transactions on Robotics*, vol. 26, pp. 781–799, 2010.

[47] D. Yagnik, J. Ren, and R. Liscano, "Motion planning for multi-link robots using artificial potential fields and modified simulated annealing," *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pp. 421–427, 2010.

[48] M. Tanaka, K. Kon, and K. Tanaka, "Range-sensor-based semiautonomous whole-body collision avoidance of a snake robot," *IEEE Transactions on Control Systems Technology*, vol. 23, pp. 1927–1934, 2015.

[49] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[50] X. Yu, W. He, Y. Li, C. Xue, J. Li, J. Zou, and C. Yang, "Bayesian estimation of human impedance and motion intention for humancrobot collaboration," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1822–1834, 2021.

[51] J. Ferreira, J. Lobo, P. Bessiere, M. Castelo-Branco, and J. Dias, "A bayesian framework for active artificial perception," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 699–711, 2013.

[52] S. Choe, H. Seong, and E. Kim, "Indoor place category recognition for a cleaning robot by fusing a probabilistic approach and deep learning," *IEEE Transactions on Cybernetics*, pp. 1–12, 2021.

[53] M. M. Arzani, M. Fathy, A. A. Azirani, and E. Adeli, "Switching structured prediction for simple and complex human activity recognition," *IEEE Transactions on Cybernetics*, pp. 1–12, 2020.

[54] B. W. Silverman, *Density estimation for statistics and data analysis*. New York: Chapman and Hall, 1986.

[55] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Baysian tracking," *IEEE Trans. Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[56] L. Squire, D. Berg, F. E. Bloom, S. Du Lac, A. Ghosh, and N. C. Spitzer, *Fundamental neuroscience*. Academic Press, 2012.

[57] F. Enner, D. Rollinson, and H. Choset, "Simplified motion modeling for snake robots," in *2012 IEEE international conference on robotics and automation*, pp. 4216–4221.

[58] A. A. Transeth, *Modelling and control of snake robots*. Fakultet for informasjonsteknologi, matematikk og elektroteknikk, 2008.

[59] M. Tanaka and F. Matsuno, "Control of snake robots with switching constraints: Trajectory tracking with moving obstacle," *Advanced Robotics*, vol. 28, pp. 415–429, 2014.

[60] T. Wang, A. Taghvaei, and P. G. Mehta, "Bio-inspired learning of sensorimotor control for locomotion," in *2020 American Control Conference (ACC)*, 2020, pp. 2188–2193.

[61] E. Baklouti, N. B. Amor, and M. Jallouli, "Particle filter localization and real time obstacle avoidance control based on 3d perception for wheelchair mobile robot," in *2016 IEEE 8th International Conference on Intelligent Systems (IS)*, 2016, pp. 735–740.

[62] X. Wu and S. Ma, "CPG-based control of serpentine locomotion of a snake-like robot," *Mechatronics*, vol. 20, no. 2, pp. 326–334, 2010.

[63] W. Ouyang, C. Li, W. Liang, Q. Ren, and P. Li, "Motion control of a snake robot via cerebellum-inspired learning control," *IEEE International Conf. on Control and Automation (ICCA)*, pp. 1010–1015, 2018.

[64] L. Zhao, Q. Xiao, Z. Cao, R. Huang, and Y. Fu, "Adaptive neural network tracking control of snake-like robots via a deterministic learning approach," *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 2710–2715, 2017.

[65] S. L. Lauritzen, *Graphical models*. Clarendon Press, 1996, vol. 17.

[66] P. D. Hoff, *Monte Carlo Approximation*. New York, NY: Springer New York, 2009, pp. 53–65.

[67] M. Deisenroth and C. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*. Omnipress, 2011, pp. 465–472.

[68] M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar, "Bayesian reinforcement learning: A survey," *arXiv preprint arXiv:1609.04436*, 2016.

[69] F. Pernkopf, R. Peharz, and S. Tschiatschek, "Introduction to probabilistic graphical models," in *Academic Press Library in Signal Processing*. Elsevier, 2014, vol. 1, pp. 989–1064.

[70] T. Jaakkola, S. P. Singh, and M. I. Jordan, "Reinforcement learning algorithm for partially observable markov decision problems," in *Advances in neural information processing systems*, 1995, pp. 345–352.

[71] M. R. Samsami and H. Alimadad, "Distributed deep reinforcement learning: An overview," *CoRR*, vol. abs/2011.11012, 2020. [Online]. Available: https://arxiv.org/abs/2011.11012

[72] M. Hoffman, B. Shahriari, J. Aslanides, G. Barth-Maron, F. Behbahani, T. Norman, A. Abdolmaleki, A. Cassirer, F. Yang, K. Baumli *et al.*, "Acme: A research framework for distributed reinforcement learning," *arXiv preprint arXiv:2006.00979*, 2020.

[73] S. Pawar and R. Maulik, "Distributed deep reinforcement learning for simulation control," *Machine Learning: Science and Technology*, vol. 2, no. 2, p. 025029, 2021.

[74] G. Sartoretti, W. Paivine, Y. Shi, Y. Wu, and H. Choset, "Distributed learning of decentralized control policies for articulated mobile robots," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1109–1122, 2019.

[75] R. Cai and C. Zhang, "Train a snake with reinforcement learning algorithms," 2020.

[76] Z. Bing, C. Lemke, F. O. Morin, Z. Jiang, L. Cheng, K. Huang, and A. Knoll, "Perception-action coupling target tracking control for a snake robot via reinforcement learning," *Frontiers in Neurorobotics*, vol. 14, p. 79, 2020.

[77] Y. Jia and S. Ma, "A coach-based bayesian reinforcement learning method for snake robot control," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2319–2326, 2021.

[78] Y. JIA and S. Ma, "A decentralized bayesian approach for snake robot control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6955–6960, 2021.

Published Papers During Doctoral Course

Journal Papers:

1. **Yuanyuan Jia** and Shugen Ma, A Coach-Based Bayesian Reinforcement Learning Method for Snake Robot Control, ***IEEE Robotics and Automation Letters***, vol.6, pp. 2319 - 2326, 2021.

2. **Yuanyuan Jia** and Shugen Ma, A Decentralized Bayesian Approach for Snake Robot Control, ***IEEE Robotics and Automation Letters***, 2021.

International Conference Papers:

1. **Yuanyuan Jia**, and Shugen Ma, A Decentralized Bayesian Approach for Snake Robot Control, In *Proc. of the 2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (**IROS'21**)*, Prague, Czech Republic, Sep., 2021.

2. **Yuanyuan Jia**, and Shugen Ma, A Coach-Based Bayesian Reinforcement Learning Method for Snake Robot Control, In *Proc. of the 2021 IEEE Int. Conf. on Robotics and Automation (**ICRA'21**)*, Xi'an, China, Jun., 2021.

3. **Yuanyuan Jia**, and Shugen Ma, A Bayesian-Based Controller for Snake Robot Locomotion in Unstructured Environments, In *Proc. of the 2020 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (**IROS'20**)*, Las Vegas, USA, Oct., 2020.