2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Bayesian Deep Predictive Coding for Snake-like Robotic Control in Unknown Terrains

William Ziming Qu*,[1], Jessica Ziyu Qu*,[1], Li Li, Jie Yang, Yuanyuan Jia[2]

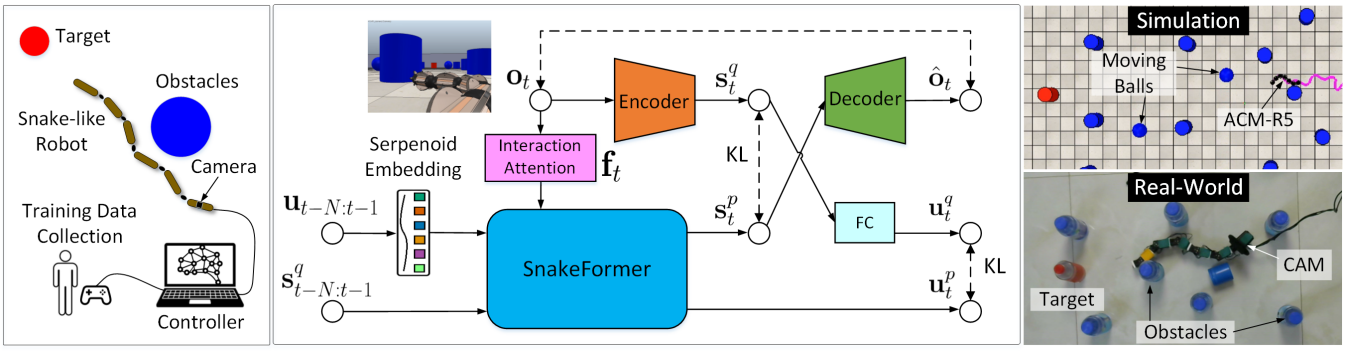

Fig. 1: **SnakeFormer Overview**. **Left:** Task setting. A snake-like robot avoids obstacles while moving towards a target. **Middle:** Training structure. The model includes a VAE module compressing observation $\mathbf{o}_t$ to latent state $\mathbf{s}_t^q$, and an enhanced Transformer conducting spatiotemporal prediction, realizing gait control with serpenoid embedding, and handling path planning and obstacle avoidance by interaction attention mechanism. Here, $\hat{\mathbf{o}}_t$ is the reconstructed image, $\mathbf{s}_t^p$ the predicted state, $\mathbf{f}_t$ the interaction attention, $\mathbf{u}_{t-N:t-1}$ the N-step actions, $\mathbf{s}_{t-N:t-1}^q$ the N-step states, $\mathbf{u}_t^p$ and $\mathbf{u}_t^q$ the predicted and estimated actions, respectively, with KL divergence being used for regularization. **Right:** Experiments in both simulation and real-world scenarios.

***Abstract*— Effectively modeling the spatio-temporal interactions both internally and externally is a challenge in controlling multi-linked snake robots. This paper presents an effective method based on deep predictive coding: SnakeFormer, to address the aforementioned issue. The main contributions include: 1) Deriving a variational free energy function with two innovative regularization terms through Bayesian probabilistic analysis, offering a novel perspective to simulate the interactions between agent and the environment; 2) Introducing an interaction-attention model within a Transformer structure for predicting dynamics, and collaboratively addressing path planning and obstacle avoidance tasks. 3) By incorporating serpenoid embedding and optimizing self-attention computations, the gait stability and motion efficiency are improved. Preliminary experiments and comparative analysis with baseline models fully validate the effectiveness and generalizability of the method.**

I. Introduction

Snake robots, due to their exceptional adaptability, have a wide range of applications in disaster relief, industrial inspection, and medicine [1]. However, their complex mechanics and multiple degrees of freedom make precise control for efficient navigation and obstacle avoidance challenging..

Early research focused on their mechanical structure and basic control algorithms for serpentine movement [2]. Facing sensor noise and frictional collisions, these methods struggled, leading researchers to explore more complex gait generation methods, such as collision handling algorithms [3], CPG-based methods [4], and distributed reinforcement learning [5]. These modeling methods deepened our understanding of snake robots' dynamic behavior, improving their ability to avoid obstacles. Yet, their dynamical models often rely on physical rules or deterministic functions, limiting adaptability and generalization. Effectively modeling the uncertainties of environmental interactions remains a highly worthwhile topic for in-depth research.

Bayesian modeling [6] combines objective observations with subjective priors and has been widely applied across many fields, serving as a powerful tool for robot-environment interactions [7]. Various world models [8] estimate the physical world's nonlinear dynamics. Model-based deep reinforcement learning [9] has recently gained attention in robotics, with policy learning based on long-term rewards being an effective method for exploring unknown environments. However, these methods have issues with large search spaces, slow convergence, and limited interpretability and generalization [10]. Predictive coding techniques [11] offer a new perspective on the interaction between robots and their environment, achieving stable and efficient control effects without designing reward functions [12]. Research on snake robots in this domain is scarce, and commonly used recursive structures, such as RNNs and LSTMs, show limitations in parallelism, global information integration, and model capacity [13]. The difficulties of obstacle avoidance in complex scenarios for snake robots have driven researchers

[1]Canadian Academy, 658-0032, Japan. *Authors contributed equally.
[2]Kyoto University, 606-8501, Japan, jiayuanusa@gmail.com
This work was supported by JSPS KAKENHI Grant Number 23K16976.

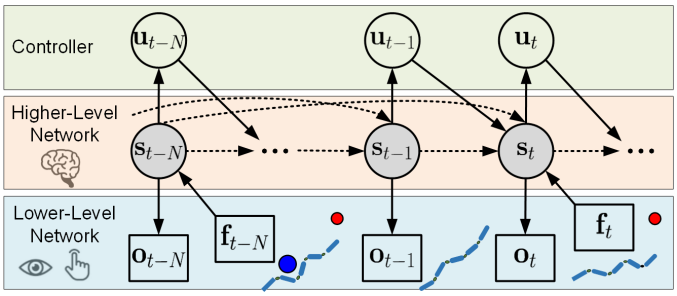

Fig. 2: The probabilistic graphic model for SnakeFormer.

to design more effective network structures.

The Transformer model [14], with its self-attention and multi-head design, can handle long-term dependencies, surpassing the limitations of traditional recurrent neural networks and demonstrating great potential [15]. Researchers have begun exploring its application in robotics [16]. However, the direct use of traditional transformer models often leads to a high risk of overfitting. This challenge is even more significant for snake robots, which consist of multiple joints and operate in complex and uncertain environments.

To address these challenges, we developed a unique SnakeFormer architecture for controlling and navigating snake robots. Its main advantages include: 1) Proposing a unified Bayesian probabilistic framework to model the internal and external spatio-temporal interaction between robot and environment; 2) Combining a Variational Autoencoder (VAE) with the Transformer architecture for end-to-end feature extraction and latent space prediction, specifically targeting the obstacle navigation problem; 3) Designing an enhanced dynamic interaction attention model and serpenoid embedding, achieving stable and efficient gait coordination among the robot's multiple links while navigating obstacles.

The rest of the paper is structured as follows: Section II details the research methodology; Section III delves into training and testing experiments; the final section concludes with discussions and summaries.

II. Methodology

Based on the foundation of Bayesian spatio-temporal analysis, we designed the SnakeFormer model and demonstrated the details of network implementation and training steps.

A. Mathematical Modeling

We employ the Probabilistic Graphical Model (PGM) [17] to simulate the obstacle avoidance process of snake-like robots. As shown in Fig. 2, $\mathbf{s}_t$ represents the latent state of robot at moment t. The connections between states at different time steps represent the dynamics, which assumes to be an N-step Markov chain $p(\mathbf{s}_t|\mathbf{s}_{t-N:t-1})$. This multi-step state transition allows the model to infer over a long time span, better adapting to changes and uncertainties in the environment. The arrow from $\mathbf{s}$ to observation $\mathbf{o}$ represents the generative relation $p(\mathbf{o}|\mathbf{s})$. Since the camera moves in a winding manner, observations cannot fully reflect the true state of the system, making the whole process a partially observable Markov decision process (POMDP). The environmental interaction $\mathbf{f}$ is dynamically changing. The definition of action $\mathbf{u}_t$ will be discussed in Section II-D. The directed edge from $\mathbf{s}_t$ to $\mathbf{u}_t$ represents the control policy. Actions sent to the robot will affect the state at the next moment. In predictive coding theory [11], the higher-level predictive network based on state transition simulates the brain of a cognitive robot, which updates the prediction with lower-level sensory input [8].

For snake robot control, we need to dynamically estimate the latent state $\mathbf{s}$ and actions $\mathbf{u}$ based on observations $\mathbf{o}$ and interactions $\mathbf{f}$, i.e., the posterior probability distribution $p(\mathbf{u}_{1:t}, \mathbf{s}_{1:t}|\mathbf{o}_{1:t}, \mathbf{f}_{1:t})$, where $\mathbf{u}_{1:t}$ represents the action sequence, and so on. Using conditional probabilities and Bayes' theorem, this distribution can be derived as follows:

$$
\begin{aligned}
& p(\mathbf{u}_{1:t}, \mathbf{s}_{1:t} \mid \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) \\
= \; & \frac{p(\mathbf{o}_t \mid \mathbf{s}_t) p(\mathbf{u}_{1:t}, \mathbf{s}_{1:t}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t})}{p(\mathbf{o}_{1:t}, \mathbf{f}_{1:t})} && (1) \\
= \; & \frac{p(\mathbf{o}_t \mid \mathbf{s}_t)}{p(\mathbf{o}_{1:t}, \mathbf{f}_{1:t})} \cdot p(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t}) \\
& \cdot p(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t}) && (2) \\
= \; & \frac{p(\mathbf{o}_t \mid \mathbf{s}_t)}{p(\mathbf{o}_{1:t}, \mathbf{f}_{1:t})} \cdot p(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{f}_t) p(\mathbf{f}_t) \\
& \cdot p(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) && (3) \\
= \; & \frac{p(\mathbf{o}_t \mid \mathbf{s}_t) p(\mathbf{f}_t) p(\mathbf{u}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t}, \mathbf{f}_t)}{p(\mathbf{o}_t, \mathbf{f}_t \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1})} \\
& \cdot p(\mathbf{s}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t-1}, \mathbf{f}_t) \\
& \cdot p(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) && (4)
\end{aligned}
$$

where we applied Bayes' theorem and the Markov property $p(\mathbf{o}_t \mid \mathbf{u}_{1:t}, \mathbf{s}_{1:t}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t}) = p(\mathbf{o}_t \mid \mathbf{s}_t)$, $p(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t}) = p(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{f}_t)$ and $p(\mathbf{f}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) = p(\mathbf{f}_t)$ in (1) and (3). In (4), we separated actions and states based on the conditional probability multiplication and applied the N-step assumption. Ultimately, we obtained a temporal iterative expression, which offers us a new way to rethink and understand the interrelated coupling among interaction with the environment, action decision-making, and iterative latent space states.

B. Variational Free-Energy Principle

Estimating the posterior distribution obtained is crucial for solving specific application problems. Various optimization techniques, such as variational inference [18] and deep reinforcement learning [9], can be utilized. We employ the variational free energy principle as a paradigm. For this purpose, we first introduce an approximate posterior distribution,

$$
\begin{aligned}
& q(\mathbf{u}_{1:t}, \mathbf{s}_{1:t} \mid \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) \\
= \; & q(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) \\
& \cdot q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) && (5) \\
= \; & q(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{o}_t) q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) && (6) \\
= \; & q(\mathbf{u}_t \mid \mathbf{s}_t) q(\mathbf{s}_t \mid \mathbf{o}_t) q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}). && (7)
\end{aligned}
$$

where the conditional probability assumptions $q(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1}, \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) = q(\mathbf{u}_t, \mathbf{s}_t \mid \mathbf{o}_t)$ and $q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t}, \mathbf{f}_{1:t}) = q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1})$ are used in (6). In (7), the action and state are separated by the conditional probability multiplication.

To minimize the variational free energy, let's first examine the KL divergence between the two posterior distributions,

$$\begin{aligned} & D_{\mathrm{KL}}[q_t \parallel p_t] \\ = \; & \mathbb{E}_q[\log q(\mathbf{s}_t \mid \mathbf{o}_t) - \log p(\mathbf{s}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t-1}, \mathbf{f}_t) \\ & + \log q(\mathbf{u}_t \mid \mathbf{s}_t) - \log p(\mathbf{u}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t}, \mathbf{f}_t) \\ & + \log q(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) - \log p(\mathbf{o}_t \mid \mathbf{s}_t) \\ & - \log p(\mathbf{u}_{1:t-1}, \mathbf{s}_{1:t-1} \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1}) - \log p(\mathbf{f}_t) \\ & + \log p(\mathbf{o}_t, \mathbf{f}_t \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1})] \qquad (8) \\ = \; & D_{\mathrm{KL}}[q(\mathbf{s}_t \mid \mathbf{o}_t) \parallel p(\mathbf{s}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t-1}, \mathbf{f}_t)] \\ & + D_{\mathrm{KL}}[q(\mathbf{u}_t \mid \mathbf{s}_t) \parallel p(\mathbf{u}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t}, \mathbf{f}_t)] \\ & - \mathbb{E}_q[\log p(\mathbf{o}_t \mid \mathbf{s}_t)] + D_{\mathrm{KL}}[q_{t-1} \parallel p_{t-1}] \\ & - \mathbb{E}_q[\log p(\mathbf{f}_t)] + \mathbb{E}_q[\log p(\mathbf{o}_t, \mathbf{f}_t \mid \mathbf{o}_{1:t-1}, \mathbf{f}_{1:t-1})] \qquad (9) \end{aligned}$$

where the last three terms are unrelated to the current action $\mathbf{u}_t$ or state $\mathbf{s}_t$, and thus do not impact the optimization result. Therefore, the variational free energy $\mathcal{F}_t$ can be defined as follows,

$$\begin{aligned} \mathcal{F}_t = & \underbrace{-\mathbb{E}_q[\log p(\mathbf{o}_t|\mathbf{s}_t)]}_{\textbf{Prediction Error}} \\ & + \underbrace{D_{\mathrm{KL}}[q(\mathbf{s}_t \mid \mathbf{o}_t) \parallel p(\mathbf{s}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t-1}, \mathbf{f}_t)]}_{\textbf{State Regularization}} \\ & + \underbrace{D_{\mathrm{KL}}[q(\mathbf{u}_t \mid \mathbf{s}_t) \parallel p(\mathbf{u}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t}, \mathbf{f}_t)]}_{\textbf{Action Regularization}}. \end{aligned} \qquad (10)$$

Next, we minimize the variational free energy $\arg\min_\phi \mathcal{F}_t$ using deep neural networks, where ϕ represents the network parameters to be estimated.

C. Variational Autoencoder Details

The encoder consists of a Convolutional Neural Network (CNN) with two hidden layers. Each layer contains 32 filters of size 3×3 with a stride of 1, followed by a ReLU function. The input image $\mathbf{o}_t$ has a resolution of $128 \times 128 \times 3$. The loss function used is $E_q[\log q(\mathbf{s}_t^q|\mathbf{o}_t)]$ as shown in (10).

The decoder is a deconvolutional neural network that takes the predicted latent state $\mathbf{s}_t^p$ as input. It generates the reconstructed image observation $\hat{\mathbf{o}}_t$ through a fully connected layer, two transposed convolutional layers with 32 filters of size 3×3 and a stride of 1, followed by a ReLU function. It uses the expected log-likelihood $E_q[\log p(\mathbf{o}_t|\mathbf{s}_t^p)]$ as the loss function, aiming to minimize the error between reconstructed image and the original input.

D. SnakeFormer Implementation

As shown in Fig. 3, the proposed SnakeFormer introduces three main improvements:

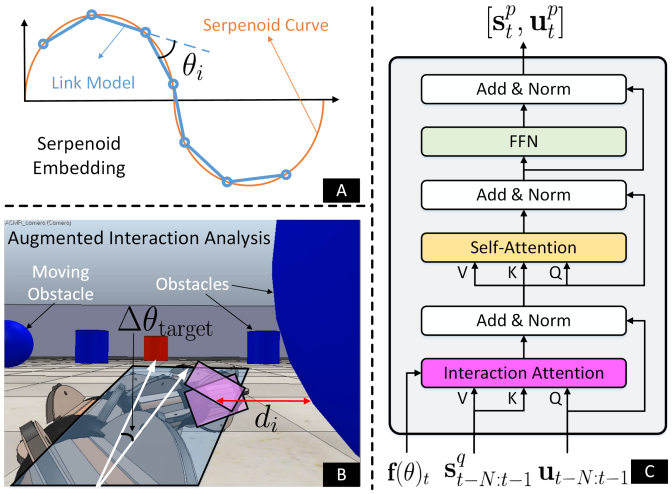

Fig. 3: **SnakeFormer**. **A)** Serpenoid embedding; **B)** Interaction analysis including attraction by the target and the repulsion with obstacles; **C)** Interaction-attention based transformer.

Serpenoid Embedding: As illustrated in Fig. 3A), we adopt the link model and serpenoid curve [2] to simplify the association within the robot:

$$\theta^i = \alpha_t^i \sin(\omega t + i\beta) + \gamma_t^i \qquad (11)$$

where θ represents the actuator angles, α the curve amplitude, ω the angular velocity, i the joint index, β the phase, and γ the offset. We adopts an angle control, meaning that at time t, the controller sends $\boldsymbol{\theta}_t = \{\theta_1, \cdots, \theta_I\}$ to the robot's servos, where I is the total number of joints. For simplicity, we fix ω and β, controlling only α_t^i and γ_t^i for each joint, i.e., $\mathbf{u}_t = \langle \Delta\boldsymbol{\alpha}_t, \Delta\boldsymbol{\gamma}_t \rangle$, where $\Delta\boldsymbol{\alpha}_t$ and $\Delta\boldsymbol{\gamma}_t$ are the amplitude vector and offset vector for each joint, respectively. As shown in Fig.1, the action sequence $\mathbf{u}_{t-N:t-1}$ goes through serpenoid embedding before entering the SnakeFormer model.

Augmented Interaction Analysis: Although the latent state implicitly contains the environmental interaction with robot, we found that its direct use may lead to issues like insufficiently agile obstacle avoidance and slow posture recovery after collisions. Therefore, leveraging certain priors for explicit interaction analysis is crucial.

As shown in Fig.3B, we utilize YOLO8 [19] to detect target, obstacles, and robotic links. Then, we fit a quadrilateral around the robot to find the midline as body direction. The angle $\Delta\boldsymbol{\theta}_{\text{target}}$ between this direction and the line from the camera to the target represents the corrective steering angle the robot should adjust. Simultaneously, we determine whether to activate the local obstacle avoidance based on the distance d_i between each link and the nearest obstacle. If d_i is less than a predefined threshold d_{TH}^i, potential collision avoidance is needed. Note that each link has its own threshold d_{TH}^i, mainly to account for distance distortion under the camera's perspective. The corrective steering angle

is defined as,

$$\Delta\theta_i = \text{sign}(x^i_{\text{link}} - x^j_{\text{obstacle}}) \cdot \frac{\vartheta_0}{d_i^2 + \epsilon} \tag{12}$$

where ϑ_0 is a preset constant, and ϵ a small number to prevent division by zero. When an obstacle j is located to the right side of link i, $\Delta\theta_i$ is negative, signifying that a shift to the left is needed; conversely, $\Delta\theta_i$ is positive, meaning the joint needs to shift to the right. Thus, we obtain the enhanced interaction turning vector, $\mathbf{f}(\theta) = \Delta\boldsymbol{\theta}_{\text{target}} + [0, \cdots, \Delta\theta_i, \cdots, 0]^T$.

Interaction-Attention Based Transformer: Building on the above analysis, we designed the interaction attention for transformer decoder as follows:

$$\text{InterAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{(\mathbf{Q} + \mathbf{f}(\theta))\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{13}$$

As shown in Fig.3C), we also swapped the order of the cross-attention and self-attention modules, allowing the attention scores to focus more on features of environmental interaction.

To better implement the regularization in (10), we also use another Fully Connected network (FC) to generate corresponding actions $\mathbf{u}^q_t$ from the encoder output $\mathbf{s}^q_t$, estimating $q(\mathbf{u}_t|\mathbf{s}_t)$. Note that in (10), although $p(\mathbf{u}_t \mid \mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t}, \mathbf{f}_t)$ includes $\mathbf{s}_t$, when estimating $\mathbf{u}^p_t$, $\mathbf{s}^p_t$ is not directly used. However, since both share the same network training and there is an internal self-attention mechanism, the prediction of $\mathbf{u}^p_t$ implicitly contains the information of $\mathbf{s}^p_t$.

E. Training Workflow and Algorithm Pseudocode

The entire training process comprises three parts:

Data Collection: In simulation, we utilized the Coach-Based Reinforcement Learning method (CBRL) [20] to collect data, including robot state information (such as joint angles and velocities) and environmental observation, as well as control signals. This method provides basic obstacle avoidance and path planning, performing well with fixed obstacles. The limitations include: 1) Under rapid impacts from moving obstacles, it may lead to delayed avoidance and loss of direction after severe collisions; 2) With significant environmental changes, the original strategy may not apply, leading to frequent collisions or longer paths. To address these, we further employed human expert assisted control for corrections, to collect more effective training data.

Additionally, we used domain randomization to enhance the generalization ability, allowing the model to "see" as many scenario variations as possible during training. This includes changes in lighting, texture, color for visual domain randomization; changes in friction coefficients, obstacle weight, moving speeds for physical parameter randomization; and adding image noise, joint sensor bias for sensor noise randomization.

Simulation Teaching: This process involves two steps: the first step is the pre-training of the VAE model, learning an effective low-dimensional representation of the environment. The second step is training the SnakeFormer. Using predictive coding training strategy, the learning of multiple networks is achieved by minimizing the variational free energy $\mathcal{F}_t$ in (10). Algorithm 1 presents the pseudocode for the above process.

Algorithm 1 Training Process of the Proposed SnakeFormer

Input: Training dataset $\mathbf{o}_{1:T}$, $\mathbf{u}_{1:T}$, define learning rate ρ
Initialize the network: VAE + SnakeFormer + Controller
while $t < T$ **do**
  Estimate the latent state: $\mathbf{s}^q_t = \text{Encoder}_{\phi_1}(\mathbf{o}_t)$
  Calculate the InteractionAttention $\mathbf{f}_t$
  Make the predication for both state and action:
  $\langle \mathbf{s}^p_t, \mathbf{u}^p_t \rangle = \text{SnakeFormer}_{\phi_2}(\mathbf{u}_{t-N:t-1}, \mathbf{s}_{t-N:t-1}, \mathbf{f}_t)$
  Evaluate the action: $\mathbf{u}^q_t = \text{FC}_{\phi_3}(\mathbf{s}^q_t)$
  Reconstruct the image: $\hat{\mathbf{o}}_t = \text{Decoder}_{\phi_4}(\mathbf{s}^p_t)$
  Compute the free energy: $\mathcal{F}_t$ by (10)
  Update networks using Adam optimizer
  $t \leftarrow t + 1$
end while

Sim2Real Transfer Learning:Considering the discrepancy between simulation and real-world scenarios, we first re-trained the VAE model to enhance its reconstruction capability. The YOLO8 module was also updated to adapt to the actual environment. Subsequently, we evaluated the model's performance in navigation and obstacle avoidance, and further fine-tuned the model as needed.

III. Experimental Results

Experiments were conducted in both simulated and the real world, initially comparing the baseline model CBRL [20] and an efficient obstacle avoidance method for snake robots, the Distributed A3C (DA3C) [5]. Ablation studies were also carried out on the core component, SnakeFormer, comparing the performance differences with a VAE method using only an FC controller (VAEF) and a method replaced with LSTM (LSTM-VAE).

A. Simulation

In the V-REP platform [21], we selected the ACM-R5 robot, with 8 joints and max torque of 10 Nm. We established a scenario as illustrated in Fig.1. The robot starts from the right side, and try to pass through a randomly placed array of obstacles to reach the target. Compared to the baseline model CBRL [20], it has two notable differences: 1) only fixed obstacles are present during training, but multiple rolling spheres are randomly placed during testing to simulate uncertainty in real world; 2) the stationary global perspective camera is used only for recording but not for training, with only a level-view camera mounted on the tail link used for observation. As previously mentioned, training data were captured by CBRL [20] and manual assistance. A total of 120 trajectories ranging from 30 seconds to 1 minute were collected, recording various types of data including video observations, actions, joint parameters, and actual locations, amassing a total of 112689 image frames.

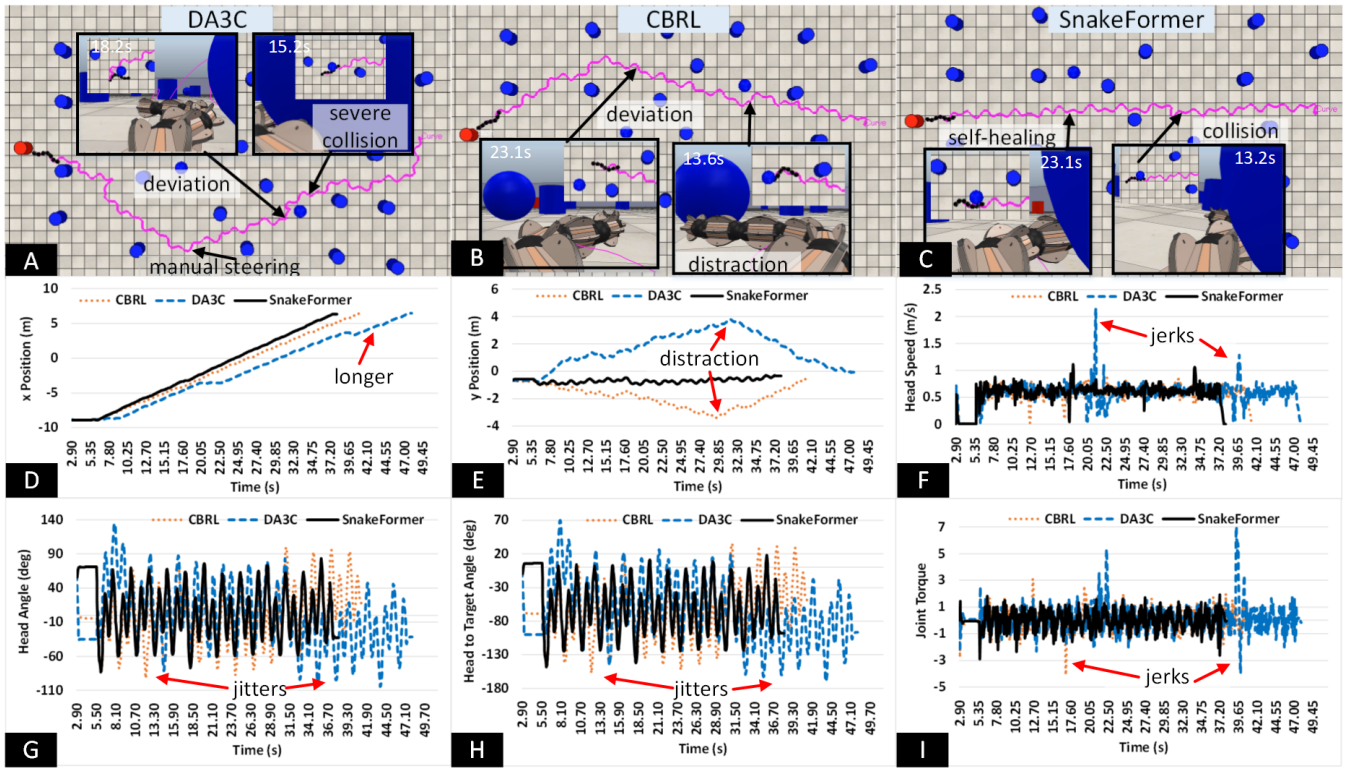

Fig. 4: Comparative Performance Analysis: SnakeFormer vs. DA3C and CBRL. A-C: Trajectories and snapshots; D-E: x,y positions; F: Snake head speed; G: Head angle; H: Change in angle between snake head and target; I: Head servo torque.

SnakeFormer V.S. Baseline Methods: SnakeFormer leverages latent space regularization, enhanced interaction attention, and self-attention gait control to achieve navigation and obstacle avoidance through end-to-end learning. To validate the algorithm's effectiveness, we compared its performance with DA3C and CBRL who use overhead cameras to capture the robot position and link posture. Since DA3C lacks target planning capability, we manually correct the direction after deviations. Fig.4 presents typical results where A-C show overall trajectories and close-up shots. DA3C's path is the most tortuous, showing good local joint deformation and obstacle avoidance for fixed obstacles but often failing to dodge fast-moving balls, with poor self-recovery. CBRL, with coach guidance, shows improved trajectory planning, but encounters deviations and slow gait recovery when meeting unforeseen obstacles like balls. In contrast, SnakeFormer demonstrates more direct and shorter trajectories (Fig.C-E), quickly dodging approaching balls. Even after collisions, it quickly recovers its gait through spatiotemporal coordination, significantly improving stability and efficiency. Fig.F-H show changes in head speed, orientation angle, and head to target angle, with SnakeFormer outperforming other methods in stability and coherence, showing smaller oscillations in curves, highlighting its obstacle avoidance, strong path planning, and effective gait control against uncertain collisions. Fig.I emphasizes SnakeFormer's advantage in torque, maintaining low and stable loads, whereas other methods show larger peaks that may cause joint wear.

TABLE I: Performance Comparison with Baseline Models

Average Metric	DA3C	CBRL	SnakeFormer	Optimization
# of Collisions	21	9	6	$33.3\% \sim 71.43\%$
Travel Time (s)	63.34	48.96	39.72	$18.9\% \sim 37.3\%$

We further compared the statistical data of the three methods across 20 test runs. As shown in TableI, SnakeFormer had fewer average collisions and shorter travel time, fully demonstrating its significant improvements in handling environmental interaction and motion efficiency.

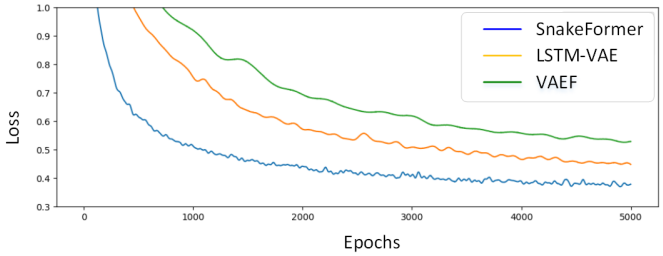

Fig. 5: Ablation Study: convergence speed curves

Ablation Study: To delve deeper into the origins of SnakeFormer's advantages, we conducted ablation studies with VAEF and LSTM-VAE methods. The LSTM network includes two hidden layers, each with 128 neurons, with

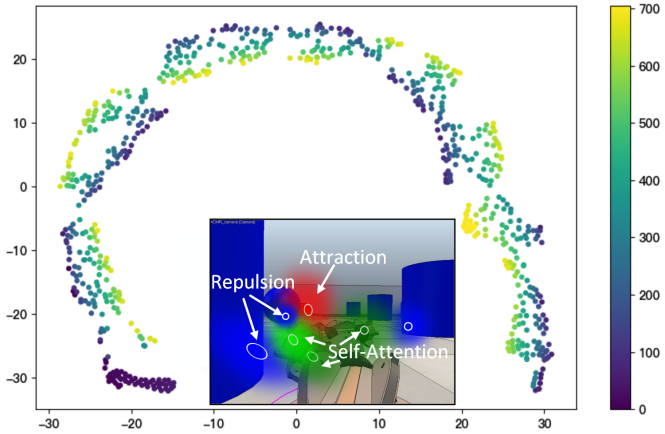

Fig. 6: Latent state visualization and multiple attentions

TABLE II: Ablation Study of SnakeFormer Structure

Performance Metric	VAEF	LSTM-VAE	SnakeFormer
Converge Epochs	13000	8000	5000
Average Collisions	35	22	5
Navigation Period (s)	97.4	67.3	41.6

inputs being the state-action sequences, excluding interaction attention. All experimental data were obtained on a PC equipped with an 8-core i7 processor, 64GB RAM, and four RTX 2080Ti GPUs. Table II presents average statistical results from 10 runs, and Fig.5 shows typical training curves for the three methods. It is observed that VAEF converges slower due to the lack of effective priors and regularization terms. Although LSTM-VAE can capture local dependencies, it requires more training epochs and exhibits moderate stability and efficiency. SnakeFormer showed rapid and stable convergence, with the trained model possessing more precise obstacle avoidance and efficient motion capabilities.

To better display the learned latent states, we used the t-SNE technique [22] to reduce them to a 2D plane. As shown in Fig.6, colors represent the sequential order, with the sample points being uniformly and centrally distributed, showing clear swinging characteristics. This is mainly due to the effective constraint of two regularization terms in the free energy during parameter optimization, the guarantee of gait coordination by the serpenoid embedding, and the flexibility brought by enhanced interaction attention. As illustrated in the subplot, intuitively, it's as if various objects in the scene each have their own potential fields, and SnakeFormer provides a spatiotemporal modeling mechanism that implicitly simulates these potential fields, allowing for effective coordination and mutual influence.

TABLE III: Details of the Real Snake Robot

Parameters	**Metric**
Dimensions (mm^3)	$75 \times 35 \times 65$
Mass (g)	72
Range of Yaw Angle (degrees)	$[-180, +180]$

TABLE IV: Transfer Learning & Generalization Comparison

Average Metric	Zero-Shot	Few-Shot	New Scene
# of Collisions	8.6	4.4	5.6
Travel Time (s)	43.3	39.5	40.2

B. Real-World Scenario

To validate SnakeFormer's generalization capability in the real world, as shown in Fig.7, we conducted experiments using a similar 6-link snake-like robot as in [20], with specifications listed in Table.III. The last link had a Logitech Webcam 920 mounted. The scene included a red bottle as target and randomly placed blue bottles as obstacles, plus rolling cylinders to add uncertainty. We then tested the proposed model for zero-shot, few-shot transfer learning, and its generalization in new scenes. Table.IV presents statistical data from 15 tests. The model trained in simulation could still work well but resulted in some additional collisions, with typical close-ups shown in Fig.7A. This is mainly because environmental and robot changes indeed make the original control parameters less adaptive. For this reason, we further collected data in new scenes, and just 7 episodes of additional fine-tuning enhanced the model's adaptability, with a noticeable reduction in collisions, as shown in Fig.7B, even demonstrating good response to fast-moving obstacles. Fig.7C illustrates the result in a different scene, and Fig.D-I show changes of servo current, angle, and joint speed, indicating stable overall performance without significant jumps and fewer collisions. This is mainly due to the effectiveness of the SnakeFormer model in spatiotemporal modeling, where serpenoid embedding and self-attention ensure coordinated gait, and interaction attention achieves timely and effective obstacle avoidance. Despite the cluttered environment, the learned strategy could effectively plan and avoid obstacles, and quickly adjust the path after collisions.

IV. Conclusion

This paper presents a SnakeFormer method based on deep predictive coding to model the complex coupled interactions in obstacle avoidance navigation task of snake-like robots, which significantly enhances the controller's flexibility, stability, and robustness. Future research includes further integrating reinforcement learning to boost generalization capability, adding sensors to improve perception, and generalizing this method to 3D snake-like robot control.

References

[1] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "Snake robot locomotion in environments with obstacles," *IEEE/ASME Transactions on Mechatronics*, vol. 17, pp. 1158–1169, 2012.

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

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

[4] X. Wu and S. Ma, "CPG-based control of serpentine locomotion of a snake-like robot," *Mechatronics*, vol. 20, pp. 326–334, 03 2010.

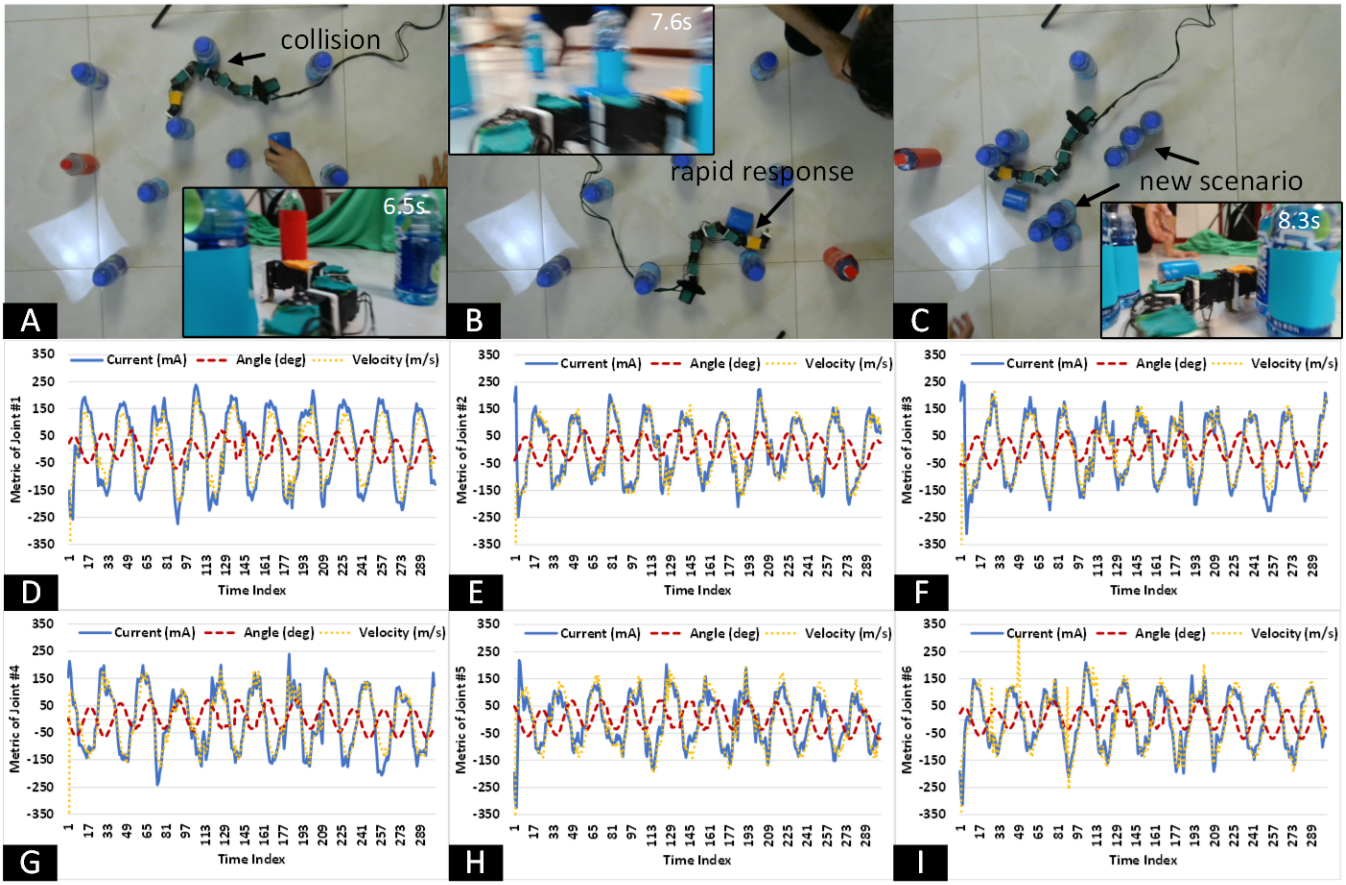

Fig. 7: Performance of SnakeFormer control over a snake robot in complex real-world environment.

[5] 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.

[6] D. Barber, *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[7] 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.

[8] T. Taniguchi, S. Murata, M. Suzuki, D. Ognibene, P. Lanillos, E. Ugur, L. Jamone, T. Nakamura, A. Ciria, B. Lara, and G. Pezzulo, "World models and predictive coding for cognitive and developmental robotics: Frontiers and challenges," *Advanced Robotics*, vol. 37, no. 13, pp. 780–806, 2023.

[9] 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.

[10] J. Qu, W. Z. Qu, L. Li, and Y. Jia, "Reinforcement learning based multi-layer bayesian control for snake robots in cluttered scenes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[11] A. Ororbia and D. Kifer, "The neural coding framework for learning generative models," *Nature Communications*, vol. 13, p. 2064, 04 2022.

[12] K. Suzuki, H. Ito, T. Yamada, K. Kase, and T. Ogata, "Deep predictive learning : Motion learning concept inspired by cognitive robotics," 2023.

[13] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[15] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM computing surveys (CSUR)*, vol. 54, no. 10s, pp. 1–41, 2022.

[16] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[17] 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.

[18] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[19] [Online]. Available: https://github.com/ultralytics/ultralytics

[20] 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.

[21] E. Rohmer, S. P. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 1321–1326.

[22] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.