Fast adaptive control of snake robots for pole climbing

Zhiyong Jian*, Jianping Huang*, Linlin Liu*, Yuhong Huang†, Long Cheng* and Kai Huang*
*School of Data and Computer Science,Sun yat-sen University, Guangzhou, China †College of Computer, National University of Defense Technology, ChangSha, China

***Abstract*— Snake-liked robots are a class of biomorphic hyperredundant robots consist of many chainconnected active joint modules. Autonomy of this kind of robots is a complex problem. This paper propses an adaptive control framework for snake robots, specially for pole climbing. Our framework can react to environment changes in real time. Experiemental results that with our framework our snake-liked robot can adapt to poles with different diameters.**

***Keywords*—Snake-liked robots; autonomous ability; pipe climbing; weighted regression; machine learning**

I. Introduction

Snake robots are a class of biomorphic hyper-redundant robots [1], designed to imitate the snake creatures biological limbless locomotion with outstanding rapidity, stability, and diversity in wild environment. Typically, these snake robots consist of many chain-connected active joint modules, giving them kinematic versatility, like bending, stretching, and crispation. A considerable number of these robots have been implemented in various fields, such as disaster rescuing, factory maintenance, and terrorism surveillance.

In order to better complete predefined tasks, snake robots are required to obtain capabilities of moving autonomously and behaving self-adaptively [2], e.g., making decision when, where, and how to move based on different situations of itself and environment. Automatically adapting to the environment is, however, not easy. The reasons are multi-folds. First, due to the redundant degrees of freedom, the locomotion of the snake robot is difficult to model, especially including the complex interaction between the robot and the environment. The corresponding control for this multi-degree locomotion is even more complicate. Second, when encountering an environment which is not known beforehand, how to decide a suitable control strategy is obvious. Even for a given control strategy, how to decide its parameters is not straightforward. Third, the runtime procedure of deciding the control strategy and its corresponding parameters must be efficient, i.e., in real time. Otherwise, the desired locomotion of the robot will most probably fail.

To let the robots adapt to an unknown environment, methods that use the sensors to perceive the environment and to embed the environment perception rules has been widely used [3], [4], [5], [6]. Tang et al. proposed a control strategy based on CPG(central pattern generator) model [3]. Rollinson et al. proposed a snake robot adaptive control based on state estimation [4]. As the model in these methods is a gradient model based on state estimation, these they are not suited to the mutation environment. There are researchers that have proposed neural network models combined with physical environment information to determine the control scheme [7], [8], [9], [10]. These models are only the control suggestion and may not make a good effect on real-time motion of the robots.

In this paper, a new framework for adaptive control of snake robots is proposed. Based on data that are collected off-line by unsupervised training, our framework can efficiently adapt new control parameters for environment changes by fast regression. Specifically, we target pole climbing, which demands a fast response to the environment changes. In this scenario, if the control cannot adapt on-time, the robot will fall. To evaluate the effectiveness of the proposed approach, we conduct experiments on a snake robot on different poles. The experimental results show that the overhead of runtime adaptation is slightly more than 100 ms and snake robot can fast adapt to different poles and climb along the poles smoothly. The contributions of this paper are:

- A adaptive control framework for the pole climbing of snake robots. For the offline unsupervised training, a k-means++ method is used to cluster the collected data. During runtime, the Z-axis velocity of the robot is used as a feed-back signal to continuously adapt new control parameters of the robot.
- A transformation of multiple regression problem is proposed to obtain the optimal control parameters in a real-time manner. The entropy variance is utilized to select the parameter which is the most sensitive in current state. Then, by regressing this parameter instead of all the parameters, the multiple regression is transformed into an unit regression problem.
- A set of case studies is conducted and the results demonstrate that our proposed strategy is effectiveness in accomplishing autonomous movement for snake robots.

The rest of this paper is organised as follows. The relative work is shown in Section II and an overview of our work is demonstrated in the Section III. In Section IV and V, proposed strategy is explained in detailed. And then Simulation works are illustrated in Section VI. Section VII shows the conclusion and our future prospects about the proposed

strategy.

II. Overview

Our approach proposed in this paper is shown in Fig. 1. We divide our approach into two main parts: off-line work and runtime execution.

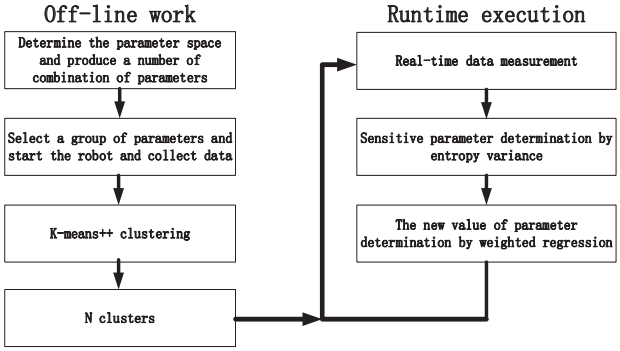


Fig. 1. The overall approach

In off-line work, firstly, we define parameter space and let the robot climb the poles with different diameters by different parameters which are stable during climbing. When the robot are climbing the pole, we record the velocity and joint angles of the robot every two seconds. After that, we combine all the data from climbing different poles together and then conduct clustering [11], [12] on the whole data.

In runtime execution, we will collect the real-time data like joint angles, velocity and the value of parameters every two seconds and then select the data recorded in off-line work to our calculation by clustering the real-time data. After selecting the data we need, we determine the parameter whose change has the greatest effect on the motion of the robot by entropy variance. And finally we get the new value of the selected parameter by weighted regression.

Currently, a widely used of control strategy for snake-liked robots is based on the sinusoidal motion model [13] which is proposed by professor Hirose. After that, Tesch et al. proposed a parametric equation based on the sinusoidal model for a snake-liked robot with three-dimensional athleticism [14]. This parametric equation simplifies the control strategy of the snake-like robots and allows the robot to determine the motion model of the machine with a small amount of control parameters. We take rolling gait [15] as our climbing gait in simulation. The sinusoidal motion model function about rolling gait is shown in the following formula:

$$T_i = \begin{cases} A \cdot \sin(\omega \cdot t + i \cdot \varepsilon) & odd \\ A \cdot \sin(\omega \cdot t + i \cdot \varepsilon + \frac{\pi}{2}) & even \end{cases} \tag{1}$$

By modifying the amplitude A, phase ε, and angular rate ω in Eq.1, the maximum rotation angle of joints, robots' shape and the motion rate of the snake-liked robot are changed. Unlike the biological snake, rolling gait can be performed by snake-liked robot. Rolling gait is an easy but effective gait. Snake-liked robot can move on the ground or climb the pole by rolling gait with suitable parameters.

III. off-line work

In the preprocessing work, we let the robot move along 25 cm and 35 cm poles for a large number of times and we collect and store the data listed in the form like Table I.

TABLE I
RECORDED VALUE DURING TRAINING

Symbol	Definition
$\theta_{m,i}$	The joint angle of the i_{th} joint which is measured
M	Mean of the measured joint angles
υ_z	The Z-axis velocity
A	The value of amplitude
ω	The value of angular rate
ε	The value of phase

We define M as $\frac{\sum_{i=1}^{n}\theta_{m,i}}{n}$ where n is the number of joints. $\theta_m = \begin{bmatrix}\theta_{m,1} & \theta_{m,2} & \cdots & \theta_{m,n}\end{bmatrix}$ is the joint angles which are measured. The control vector is shown as $\begin{bmatrix}A & \omega & \varepsilon\end{bmatrix}$ where A is the amplitude, ω is the angular rate, and the ε is the phase. All of them are applied in Eq.1.

After collecting movement data of a snake-liked robot, we combine the training data obtained in different environment and then cluster the data in order to optimize real-time calculation.

In this research, collected data in preprocessing is a large-scale data set. As k-means++ algorithm has high efficiency and scalability for a large-scale data set, we adopt $kmeans++$ for clustering. We classify training set into N clusters. The clustering process is divided into the following steps:

- Step 1: Decide the value of N by Eq.2.

$$N = arg\min_{N_k} \frac{\sum_{N_k}(S_i - E)^2}{N_k} \tag{2}$$

  where $E = \frac{\sum_N(S_i)}{N}$ and $i \in [1, N]$.
- Step 2: perform k-means++ as **Algorithm 1** shows. K-means++ is an improvement on the k-means algorithm. By function INITIALIZE, we can obtain N initial cluster centers. And then, we iterate function UPDATE to update the cluster centers and the clusters until the cluster centers do not change any more.

With $Kmeans++$, the collected data can be divided into N clusters. After finishing the clustering, the result is stored in two parts:

1) The cluster centers $\mathbf{X}$.
2) The member $P_i, i \in [1, |\boldsymbol{P}|]$ of the cluster center $X_k, k \in [1, N]$

Algorithm 1 k-means++

Input: Training set $\boldsymbol{P}$, The number of clusters N
Output: Set of cluster centers $\boldsymbol{X}$

function INITIALIZE($\boldsymbol{P}, N$)
  $\boldsymbol{X} \leftarrow sample\ a\ point\ randomly\ from\ \boldsymbol{P}$
  while $|\boldsymbol{X}| < N$ **do**
    for $i = 1 \rightarrow |\boldsymbol{P}|$ **do**
      $I \leftarrow arg\min_i \left(\sum_{j=1}^{|\boldsymbol{X}|} \|X_j - P_i\|^2\right)$
      $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \{P_I\}$
      $\boldsymbol{P} \leftarrow \boldsymbol{P} - P_I$
    end for
  end while
  return $\boldsymbol{X}$
end function

function UPDATE($\boldsymbol{P}, \boldsymbol{X}$)
  while X does not change any more **do**
    for $i = 1 \rightarrow |\boldsymbol{P}|$ **do**
      $C_i \leftarrow arg\min_k \left(\|P_i - X_k\|^2\right)$
    end for
    for $k = 1 \rightarrow |\boldsymbol{X}|$ **do**
      $X_k \leftarrow \frac{\sum_i \{C_i = k\} P_i}{\sum_i \{C_i = k\}}$
    end for
  end while
  return $\boldsymbol{X}$
end function

IV. RUNTIME

In robot's running time, we get the real-time data periodically and then we do the following steps. Firstly, we categorize the real-time data based on the clustering result in off-line work. And then we select the most sensitive gait parameter according to the entropy variance. At last, we use the idea of weighted regression to modify the selected gait parameter and keep the other gait parameters unchanged.

A. Parameter selection by entropy variance

1) Real-time data categorization: Every time we get the real-time data, we relegate it to the certain cluster by Eq.3.

$$C = arg\min_k \left(||X_k - P_t||_2\right), \quad X_C \in \boldsymbol{X} \tag{3}$$

$\boldsymbol{X}$ is the cluster center set(**Algorithm 1**). X_C is the closet center to the real-time data vector P_t. And X_k is the k_{th} cluster center of the cluster center set $\boldsymbol{X}$. With $Euclidian\ Distance\ Formula$, we make a prediction on the similarity between two data vectors.

2) The selection of the preponderant data: After categorization of real-time data, we select those preponderant vectors whose Z-axis velocity are bigger than current (Eq.4).

$$\boldsymbol{P_v} = \{P_i | \upsilon_{z,i} \geq \upsilon_{z,P_t}\ ,\ P_i \in \boldsymbol{P_C}\} \tag{4}$$

$\boldsymbol{P_C}$ is all the data vectors which belong to the cluster with the center X_C. $\upsilon_{z,i}$ is the vertical velocity component of the data vector P_i and υ_{z,P_t} is the vertical velocity component of the real-time data vector. $\boldsymbol{P_v}$ is the set of all the preponderant vectors for regression.

3) The selection of the sensitive parameter: We adopt entropy variance as the reference to select the parameter which should be modified. The steps of selecting the sensitive parameter are as follows.

- Step 1: We take the preprocessing operation to discrete the preponderant data (Eq.5).

$$\upsilon_{new,i} = \begin{cases} \left\lfloor \frac{\upsilon_{z,i}}{L_D} \right\rfloor & \upsilon_{z,i} > 0 \\ \left\lceil \frac{\upsilon_{z,i} - L_D}{L_D} \right\rceil & \upsilon_{z,i} \leq 0 \end{cases} \tag{5}$$

  In Eq.5, L_D is the adjustable step length for discretization. We eventually get the velocity discrete sequence:

$$V_{new} = \begin{bmatrix} \upsilon_{new,1} & \upsilon_{new,2} & \upsilon_{new,3} & \upsilon_{new,4} & \cdots & \cdots \end{bmatrix} \tag{6}$$

- Step 2: There are a variety of possible values for each gait parameter. Thus, in order to record all the possible values, we make a set $S_{i,j}$ with the gait parameter i's value is its j_{th} possible value. And the value of i is from 0 to 2 representing A, ω and ε respectively. We calculate the entropy about the vertical velocity when the gait parameter i's value is its j_{th} possible value by Eq.7 as well as the entropy variance of the gait parameter i by Eq.8. In Eq.7, $p(\upsilon_{new,k})$ is the appearance rate of the $V_{i,j}$ in $S_{i,j}$. In Eq.8, $E_{i,j}$ is the mean of velocity when the gait parameter i's value is its j_{th} possible value. And N_i is the number of possible values of the gait parameter i.

$$H(S_{i,j}) = -\sum_{\upsilon_{new,k} \in V_{new}} p(\upsilon_{new,k}) log_2 p(\upsilon_{new,k}) \tag{7}$$

$$Var_i = \frac{\sum_{N_i} (H(S_{i,j}) - E_{i,j})^2}{N_i} \tag{8}$$

- Step 3: We normalize the entropy variance (Eq.9) and randomly select the sensitive gait parameters by Roulette wheel selection algorithm.

$$R_{Var_i} = \frac{Var_i}{\sum Var_i} \tag{9}$$

After the calculation of gait parameters' entropy variance. the sensitive parameter will be found. And then the selected parameter's value will be modified by regression.

B. Assignment to the value of the selected parameter

In this research, we take weighted regression to get the value of the sensitive gait parameter and use the gradient descent method to solve the weighted least squares problem in fitting regression function.

- Step 1: List the fitting prediction function(Eq.10)

$$F_w(P_t) = W^T P_t\,, \quad W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} \tag{10}$$

In Eq.10, W is the coefficient sequence of the fitting equation and m is the number of coefficients where P_t is the real-time collected data vector. Then we can get the error function(Eq.11) which takes the square of error as the estimation with n being the number of data of $\boldsymbol{P_v}$ and $\boldsymbol{Q}$ being the set consisting of the value of selected parameter in P_i.

$$D(W) = \frac{1}{2n}(F_w(\boldsymbol{P_v})^T - \boldsymbol{Q})^T(F_w(\boldsymbol{P_v})^T - \boldsymbol{Q}) \tag{11}$$

$$\boldsymbol{P_v} = \begin{bmatrix} P_{v,1} & P_{v,2} & \cdots & P_{v,n} \end{bmatrix} \tag{12}$$

$$\boldsymbol{Q} = \begin{bmatrix} Q_{v,1} & Q_{v,2} & \cdots & Q_{v,n} \end{bmatrix}^T \tag{13}$$

To get the best-fit coefficient sequence W by the minimum $D(w)$, according to gradient descent method, we turn the Eq.11 into Eq.14.

$$\nabla_w D = \frac{1}{n}\boldsymbol{P_v}(F_w(\boldsymbol{P_v})^T - \boldsymbol{Q}) \tag{14}$$

- Step 2: Perform the weighted operation on preponderant data vector to ensure the estimate result of fitting is good (Eq.16).

$$\nabla_w D = \frac{1}{n}\boldsymbol{P_v} M(F_w(\boldsymbol{P_v})^T - \boldsymbol{Q}) \tag{15}$$

$$M = \begin{bmatrix} \frac{v_{z,1}}{L_s} & 0 & \cdots & 0 \\ 0 & \frac{v_{z,2}}{L_s} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \frac{v_{z,3}}{L_s} \end{bmatrix} \tag{16}$$

In Eq.16, L_s is the learning step and M is the learning rate matrix.

- Step 3: Fit the coefficient vector by Eq.17.

$$W = W - \nabla_w D \tag{17}$$

In this way, the coefficient sequence W is updated.

- Step 4: Iterate the steps above until we acquire the best-fit coefficient. The value of W is stable finally.

When the iteration is stopped, the best-fit coefficient W_{best} will be obtained. By applying W_{best} to Eq.18, we can get the regression value of the sensitive parameter. This value will be used in the robot control. It is worth noting that we only modify the sensitive parameter and others remain the same value.

$$Q_t = F_w(P_t) = W^T P_t \tag{18}$$

V. SIMULATION

We use the rolling gait as the basic gait for simulations. To verify the adaptability of snake-liked robots under our proposed adaptive control, we conduct simulations on poles with different diameters under the robot simulation platform V-REP.

The simulation process can be divided into two categories: training and motion simulation. And we divide simulations into two parts: the adaptable motion along poles having changing diameter and the adaptable motion along straight poles with different diameters.

A. Training process

In the data acquisition process, we let the robot climb along the 25 cm and 35 cm poles under different parameters and collect 25 thousand volumes of training data in total. The interval of amplitude A, phase ε and angular rate ω are $[40, 80]$, $[0, 5]$ and $[1.5, 3]$ respectively. What's more, the step of A is 5, the step of ε is 1 and the step of ω is 0.5. We make the robot climb poles with different combination of parameters and then collect the data. In the preprocessing process, we cluster the training data. We set the number of clusters as 25 by Eq.2 and Fig. 2. The number of data for most of classes is 1000 ± 500 (Fig. 3).

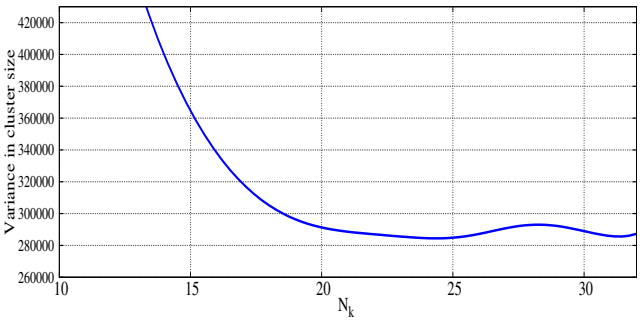


Fig. 2. The variance in cluster size of N_k

B. the autonomous motions along a variable diameter pole

We build a complex pole for simulation in this part. The pole is divided into three sections. The lowest part and the highest part are poles with 35 cm diameter and the middle part is a pole with 25 cm diameter. We record the parametes and the velocity during the climbing in 80 seconds.

Results are shown as Fig. 4. From 0 s to 24 s, the robot changes its shape to adapt to the unknown pipe. So in this phase, its velocity is close to zero. From 24 s to 45 s, the robot climbs along the part with 35 cm diameter. In this phase, A, ε and ω are all changing and A and ε are stable finally. When about 45 s, the robot reaches the interface where the diameter changes. As the diameter changes obviously, the robot can not grasp the middle part

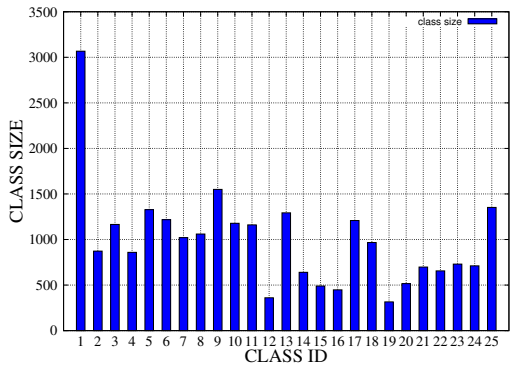


Fig. 3. The result of clustering

immediately, which causes the velocity of the robot fluctuate around zero. The robot autonomously adjusts its parameters by our strategy to continue its climbing motion. From the Fig. 4(g), Fig. 4(h) and Fig. 4(i), it can be found that the parameters are increasing in this phase, which makes the robot continue moving up. Similarly, when about 63 s, the robot meets the second interface between the middle part and the highest part. The robot maintains its motion by autonomously changing the parameters. And when the robot arrives at the highest part, all the parameters are stable at last. As a whole, the velocity changes toward bigger velocity during the motion.

Eventually all control parameters as well as velocity are stable. It shows that under this control strategy, the robot can adjust its parameters autonomously to adapt to the environment.

The simulations show that our strategy is effective to adapt the robot to climb along the variable diameter pole.

C. Contrast experiments on other straight poles

As the unknown environment in the real world is complex and volatile, there may be differences between training environment and actual environment. Our training environments are vertical poles with 25 cm or 35 cm diameter. To ensure that this control strategy can be applied in real world situation, we conduct several independent simulations on straight poles whose diameters are 20 cm, 25 cm 30 cm, 35 cm or 40 cm. The result of these simulations are shown in Fig. 5. Next is the detailed analysis.

By observing the parameter curve we can find that the smaller diameter of the pole is, more time is required to adjust the parameters when the diameter gets smaller. All parameters finally reach a stable value and satisfy the requirements of movement. For the 40 cm diameter pole, the robot is not long enough to form a single ring to wrap the pole. In this case the influence made by phase ε is very weak (Fig. 5(b)). For the 20 cm diameter pole, it is difficult for the robot to climb along the pole if we only increase the amplitude A, because a too large amplitude will cause the snake-liked robot curling up to a high degree, which is not conducive to climb. Therefore, to let the robot climb the pole with small diameters, we need to constantly adjust the phase ε to meet the amplitude A's corresponding requirements. By observing the variation curve of the control parameters of the pole with small diameter, we can found that the phase ε and the amplitude A of the robot are coordinated in the process of self-regulation (Fig. 5(a))(Fig. 5(b)). The simulations show that the robot under our control strategy can catch the unknown pole and adjust itself to a suitable climbing state.

It is worth noting that, for all test environments, the movement velocity of the robot ultimately fluctuates within a constant range (Fig. 5(d)). The diameter of the pole only affects the velocity convergence rate. And also the control parameters of the robot will eventually become stable (Fig. 5). This shows that the influence of the external environment to this control strategy is very weak. What's more, the robot will find the optimal parameter during its motion.

D. Algorithm efficiency

Fig. 6 shows the computing expense of our control strategy and the time spent by the snake-liked robot when climbing poles with different diameters.

We can see that the computation expenses are similar whatever the diameter of pole is, which indicates that the expense of our strategy not only stays in a negligible range but also receives little influence from the environment. Observe that more time is spent by the robot to climb a pole with smaller diameter. The reason is the snake-liked robot needs more time to fit its shape with a pole having a smaller diameter.

In conclusion, the simulations demonstrate that the adaptive control in robot's motion can be realized through our proposed framework.

VI. Conclusion

This paper presents a framework for robot-based adaptive control based on the learing experience of robots. We take Z-axis velocity as feed-back signal and adopt regression to correct the robots' action. We simplify the runtime learning through clustering and transform the multiple regression into unit regression. Experiments show that the scheme is effective. It is noteworthy that this method can be used not only in the case like climbing pipe in this experiment, but also in other robotic applications. We believe that the algorithm can adapt to the other corresponding scene, such as the unmanned vehicle's variable motion, the rugged ground motion of the serpentine robot and the simulated PID control as long as enough training data and clear moving purpose are given.

REFERENCES

[1] G. S. Chirikjian and J. W. Burdick, "The kinematics of hyper-redundant robot locomotion," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 781–793, Dec 1995.

[2] Liljeb, P. Ck, K. Y. Pettersen, Stavdahl, yvind, and J. T. Gravdahl, "Snake robots: Modelling, mechatronics, and control," 2013.

[3] C. Tang, S. Ma, B. Li, and Y. Wang, "A self-tuning multi-phase cpg enabling the snake robot to adapt to environments," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2011, pp. 1869–1874.

[4] D. Rollinson and H. Choset, "Gait-based compliant control for snake robots," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 5138–5143.

[5] C. G. and H. Ranganathan, "Balancing and control of dual wheel swarm robots by using sensors and port forwarding router," in *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, Feb 2017, pp. 532–536.

[6] G. Gerboni, A. Diodato, G. Ciuti, M. Cianchetti, and A. Menciassi, "Feedback control of soft robot actuators via commercial flex bend sensors," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 4, pp. 1881–1888, Aug 2017.

[7] G. Martius, R. Der, and N. Ay, "Information driven self-organization of complex robotic behaviors," vol. 8, p. e63400, 05 2013.

[8] R. Der and G. Martius, "Novel plasticity rule can explain the development of sensorimotor intelligence," vol. 112, 05 2015.

[9] J. P. Cai, L. Xing, M. Zhang, and L. Shen, "Adaptive neural network control for missile systems with unknown hysteresis input," *IEEE Access*, vol. 5, pp. 15 839–15 847, 2017.

[10] A. Vitiello, G. Acampora, M. Staffa, B. Siciliano, and S. Rossi, "A neuro-fuzzy-bayesian approach for the adaptive control of robot proxemics behavior," in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, July 2017, pp. 1–6.

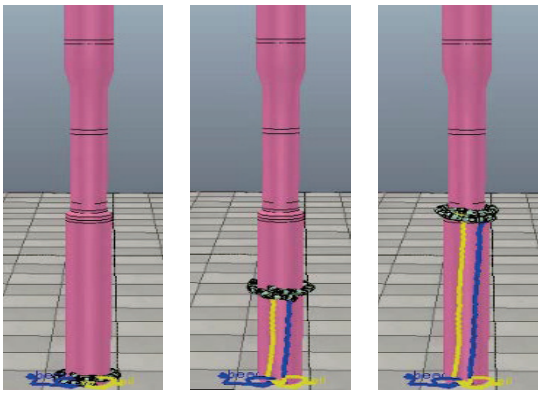
[11] A. Widodo and I. Budi, "Clustering patent document in the field of ict (information amp; communication technology)," in *2011 International Conference on Semantic Technology and Information Retrieval*, June 2011, pp. 203–208.

[12] M. Dundar, Q. Kou, B. Zhang, Y. He, and B. Rajwa, "Simplicity of kmeans versus deepness of deep learning: A case of unsupervised feature learning with limited data," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 883–888.

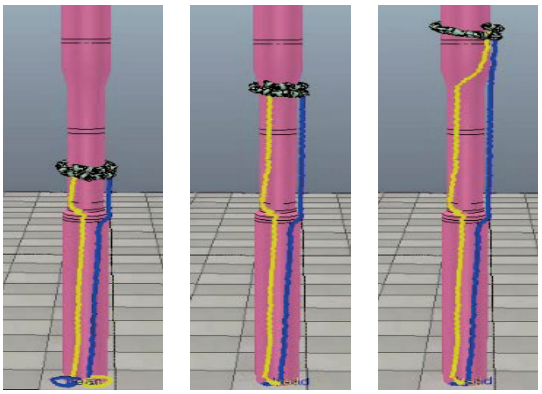
[13] H. Ohno and S. Hirose, "Design of slim slime robot and its gait of locomotion," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, vol. 2, 2001, pp. 707–715 vol.2.

[14] M. Tesch, K. Lipkin, I. Brown, R. L. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and scripted gaits for modular snake robots," vol. 23, pp. 1131–1158, 06 2009.

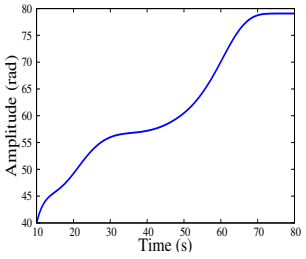
[15] F. Enner, D. Rollinson, and H. Choset, "Motion estimation of snake robots in straight pipes," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5168–5173.



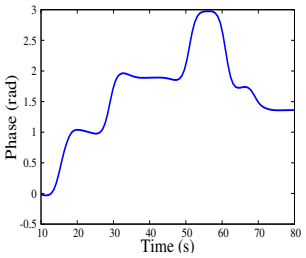
(a) t=24s (b) t=35s (c) t=47s



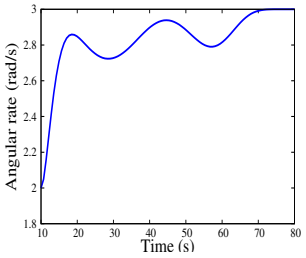
(d) t=56s (e) t=63s (f) t=69s



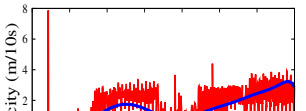
(g) Amplitude versus Time

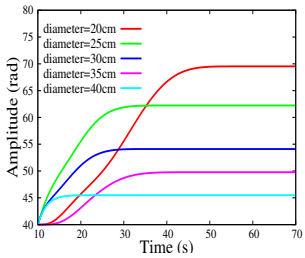


(h) Phase versus Time

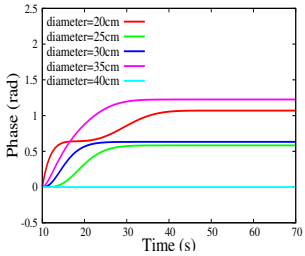


(i) Angular rate versus Time

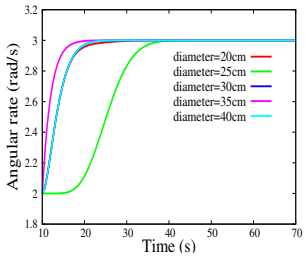




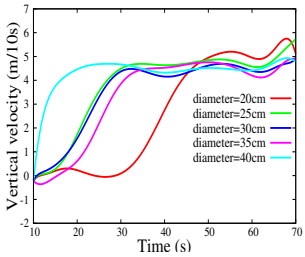
(a) Amplitude versus Time



(b) Phase versus Time

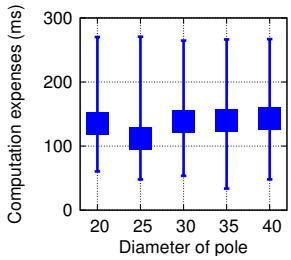


(c) Angular rate versus Time

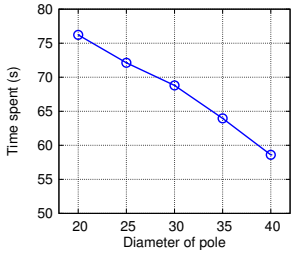


(d) velocity versus Time

Fig. 5. Curves about the robot climbing the poles with different diameters



(a) computing expenses in climbing



(b) time spent in climbing

Fig. 6. Worst, best, and average case computation expenses and the total time spent of climbing the five meters high pole with different diameters