Reinforcement Learning Approach to Generate Goal-directed Locomotion of a Snake-Like Robot with Screw-Drive Units

Sromona Chatterjee*, Timo Nachstedt*, Florentin Wörgötter*, Minija Tamosiunaite*,
Poramate Manoonpong*†, Yoshihide Enomoto‡, Ryo Ariizumi‡and Fumitoshi Matsuno‡
*Bernstein Center for Computational Neuroscience
Third Institute of Physics, University of Göttingen, Germany
Email: chatterji.sromona.86@gmail.com, poma@mmmi.sdu.dk,
m.tamosiunaite@if.vdu.lt, worgott@gwdg.de
†Maersk Mc-Kinney Møller Institute
University of Southern Denmark, Denmark
‡Department of Mechanical Engineering and Science
Graduate School of Engineering, Kyoto University, Japan

***Abstract*—In this paper we apply a policy improvement algorithm called Policy Improvement using Path Integrals (PI^2) to generate goal-directed locomotion of a complex snake-like robot with screw-drive units. PI^2 is numerically simple and has an ability to deal with high dimensional systems. Here, this approach is used to find proper locomotion control parameters, like joint angles and screw-drive velocities, of the robot. The learning process was achieved using a simulated robot and the learned parameters were successfully transferred to the real one. As a result the robot can locomote toward a given goal.**

I. Introduction

Research in the domain of snake-like robots has been ongoing for several decades [1], [2], [3]. This is because such robots can be used as experimental platforms to study locomotion or motor coordination problems [4], [5]. They can be also employed for search and rescue operations [6]. Typically, a sake-like robot consists of several segments connected in a serial manner. A conventional way to generate its locomotion is undulation movements, which imitate real snake′s movements [3], [4]. However, this type of locomotion requires a width for undulations, which is larger than the width of the robot. Therefore, this might become a problem in narrow spaces.

From this point of view, we have developed a new type of snake-like robot using a screw-drive mechanism [7]. The robot is composed of screw-drive units with passive wheels. The screw-drive units are connected by active joints serially. It has four screw-drive units and three active joints. With this construction, undulation is not required to move the robot since propulsion is generated by rotating the screws. Additionally, unlike most existing snake-like robots, the robot can move in any direction by a proper combination of screw angular velocities. As a continuation of the robot development, this article presents goal-directed locomotion control of the robot. Due to the robot structure which can be formed in different shapes (e.g., straight line or zigzag shape) and the switching of its passive wheels in contact with the ground, finding proper control parameters in a continuous state space for generating goal-directed locomotion becomes complicated and challenge. According to this, we apply a special type of reinforcement learning, called Policy Improvement using Path Integrals (PI^2) [8], for the task. This method is selected because it can deal with the problem of the ”curse of dimensionality” in a stable way [9] and has been shown to be successful for different robot learning tasks [9], [10], [11]. Here, it is, for the first time, applied to the nonstandard snake-like robot. In the work at hand, it is used as a model-free learning mechanism to find a proper combination of seven locomotion control parameters (four screw unit angular velocities and three yaw joint angles) for moving toward a given goal in a given time. This article thus also demonstrates how PI^2 can be formulated for a new application like goal-directed locomotion control of a many degrees-of-freedom system, like the nonstandard snake-like robot with screw-drive units.

II. SNAKE-LIKE ROBOT WITH SCREW DRIVE UNITS

Fig 1. shows the basic structure of the snake-like robot with screw-drive units [7]. The robot has three active joints and four screw-drive units. Each joint has two degrees of freedom (pitch and yaw angles) and has two servo motors for this. The joint angles have a range of $\pm\frac{\pi}{2}$ rad. Each screw-drive unit has one DC motor, one screw part, and an encoder. The motor drives the rotation of screw unit around its rotation axis. Each screw-drive unit has eight blades attached to it where each blade has four alternately attached passive wheels with rubber ring to it. A screw unit is said to be left or right screw unit depending on inclination of blade (α). If $\alpha > 0$ then it is called left screw unit and if $\alpha < 0$ then referred as right screw unit. In addition, the head of the robot is provided with ball bearing and ground contact for stability. The rotation of screw units generates propulsion such that the robot can move in any direction. In our experimental cases, pitch angle is always zero as all the screw units have contact with the ground by some wheels and flat ground is considered.

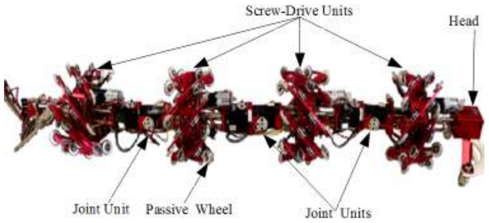


Fig. 1. Snake-like robot with screw-drive units.

TABLE I. NOTATIONS USED

Notation	Description
w	Robot's state vector
u	Robot's control input vector
ϕ_i	Joint angle
$\dot{\phi}_i$	Joint angular velocity
$\dot{\theta}_i$	Screw angular velocity
(x, y)	Position of robot head
ψ	Orientation of first unit with robot head
U_1	Parameter vector learned for a prefixed robot shape
U_2	Parameter vector used for learning all seven control parameters
r	Cost function
(x_G, y_G)	Goal position
n	Total number of updates
t	Update index
$\tau(a)$	Trajectory generated using locomotion control parameter set a
K	Total number of noisy trajectories or roll-outs
k	k^{th} roll-out
$\epsilon_{t,k}$	Noise applied in k^{th} trajectory of the t^{th} update
$\epsilon_{\dot{\theta}_i(t,k)}$	Noise applied to screw velocity
$\epsilon_{\phi_i(t,k)}$	Noise applied to joint angle
$I_{t,k}$	Final cost at the end of k^{th} noisy trajectory in the t^{th} update
$P_{t,k}$	Probability weighting for k^{th} noisy trajectory in the t^{th} update

III. THE SETUP FOR GOAL-DIRECTED LOCOMOTION CONTROL

Here, we use Policy Improvement using Path Integrals (PI^2) as a learning mechanism to find a proper combination of control parameters for goal-directed locomotion of the robot. In the following section, we present the formulation of PI^2 for the task. In PI^2 implementation here the parameter vector to be learned, say U, is updated at the end of every update t. Each update consists of K noisy trajectories or roll-outs. n updates are performed to obtain the final parameter vector which will make the robot locomote toward a given goal. U contains locomotion control parameters like screw velocities and joint angles. TABLE 1. gives the notations used in this paper.

A. Policy Formation

Our snake-like robot with screw-drive units follows the kinematic model as described in equations (1-3) [7], where, w is the state vector and u is the control input vector for the robot system.

$$A\dot{w} = Bu \tag{1}$$

$$w = [x, y, \psi, \phi_1, \phi_2, \phi_3] \tag{2}$$

$$u = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3] \tag{3}$$

(x, y) is the position of the head of the robot. ψ is the orientation of the first screw-drive unit with respect to robot head. ϕ_1, ϕ_2, ϕ_3 are the three yaw joint angles in radian (rad) which are relative orientations of one screw unit to the previous. $\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4$ are the angular velocities in radians/sec (rad/s) of the first, second, third and fourth screw-drive units from the head respectively. $\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3$ are the angular velocities of the three joints in rad/s starting from $\dot{\phi}_1$ which is the angular velocity of the first joint from the head. A and B are the system matrices and depend on system configurations like screw-drive unit radius, inclination of blade (α), length of one screw-drive unit [7].

The learning of locomotion control parameters like, screw velocities and joint angles, for goal-directed locomotion is executed . For instance, if an initial head position of the robot is at (x_0, y_0) and the goal to be reached is G (x_G, y_G), the final state vector w_{goal} on reaching goal should have the head position as (x_G, y_G). ψ is automatically adjusted with robot kinematics. Two parameter vectors representing the control policy are described by equations (4) and (5).

$$U_1 = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4] \tag{4}$$

$$U_2 = [\dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3, \dot{\theta}_4, \phi_1, \phi_2, \phi_3] \tag{5}$$

The parameter vector to be learned is selected according to the learning problem. U_1 is used for experiments when joint angles are fixed while U_2 is used for experiments where all seven control parameters, four screw-drive velocities $\dot{\theta}_i$ (i= 1, 2, 3, 4) and three joint angles ϕ_i (i= 1, 2, 3), are learned.

The control policy following the kinematic model in equations (1-3) is represented by U_1 and U_2 of equations (4-5). Note that angular velocities of joint units, $\dot{\phi}_i$ (i=1, 2, 3), are set to zero. After learning final values of U_1 and U_2 with PI^2, the learned parameters have been successfully transfered to the real robot for executing goal-directed locomotion in a real environment. As a consequence, the robot will start from the initial position (x_0, y_0) and then move toward the target position or desired goal (x_G, y_G).

B. Cost Function and Exploration Noise

A cost function or an objective function is the key component which influences convergence of learning. For our task here, we use the Euclidean distance as our cost function r:

$$r(x, y) = \sqrt{(x - x_G)^2 + (y - y_G)^2} \tag{6}$$

It basically provides the distance in meter (m) between a reached robot head position (x, y) and a given goal position (x_G, y_G). The parameter vector learned reaches it's final value when the cost is almost zero; i.e., the goal is reached and the task is completed. With the final parameter vector the robot locomotes toward the given goal.

Noise is the only open parameter in PI^2 and needs to be designed in a task based manner. For our learning problem we select random values ζ from a normal distribution N(0,1) which has zero mean and standard deviation of 1. These drawn values, ζ, are then dynamically adjusted according to the noise-free cost r_{t-1} obtained at the end of the previous update. When $r_{t-1} > 3$ m then the noise is drawn as: $\epsilon_{t,k} = (\exp \frac{-1}{r_{t-1}})/L \cdot \zeta, \zeta \in \mathbb{N}(0, 1)$. Here, L=10 meter, k is the noisy roll-out number, t is the update number, $\epsilon_{t,k}$ is the noise during k^{th} noisy trajectory or roll-out of the t^{th} update. When $0.5 < r_{t-1} \leq 3$ m then the noise is adjusted as: $\epsilon_{t,k} = 0.05\zeta, \zeta \in \mathbb{N}(0, 1)$. When the noise-free cost r_{t-1} is very low and $\leq$ 0.5 m, then the noise is adjusted as: $\epsilon_{t,k} = 0.025\zeta, \zeta \in \mathbb{N}(0, 1)$.

Let the noise applied to screw velocities be $\epsilon_{\dot{\theta}_i(t,k)}$ (i=1, 2, 3, 4) and the noise applied to joint angles be $\epsilon_{\phi_i(t,k)}$ (i=1, 2, 3). All these seven noise distributions follow above description of $\epsilon_{t,k}$ in every noisy trajectory $\tau_{t,k}$. All these seven noise patterns are different in a trajectory as $\zeta \in \mathbb{N}(0,1)$ is randomly drawn in the definition of $\epsilon_{t,k}$.

C. Implementation of PI^2 to the Learning Problem

The main steps for PI^2 applied to our learning task are discussed here. Firstly, the parameter vector, U_1 or U_2, according to the learning task is selected. Secondly, the number of roll-outs K per update needs to be fixed. We choose $K = 40$. According to the parameter vector to be learned, in every roll-out k the trajectory

$$\begin{aligned}\tau_{t,k}(&\dot{\theta}_1 + \epsilon_{\dot{\theta}_1(t,k)}, \dot{\theta}_2 + \epsilon_{\dot{\theta}_2(t,k)}, \dot{\theta}_3 \\ &+ \epsilon_{\dot{\theta}_3(t,k)}, \dot{\theta}_4 + \epsilon_{\dot{\theta}_4(t,k)}, \phi_1, \phi_2, \phi_3),\end{aligned} \tag{7}$$

or

$$\begin{aligned}\tau_{t,k}(&\dot{\theta}_1 + \epsilon_{\dot{\theta}_1(t,k)}, \dot{\theta}_2 + \epsilon_{\dot{\theta}_2(t,k)}, \dot{\theta}_3 + \epsilon_{\dot{\theta}_3(t,k)}, \dot{\theta}_4 + \\ &\epsilon_{\dot{\theta}_4(t,k)}, \phi_1 + \epsilon_{\phi_1(t,k)}, \phi_2 + \epsilon_{\phi_2(t,k)}, \phi_3 + \epsilon_{\phi_3(t,k)})\end{aligned} \tag{8}$$

with noisy parameters is simulated for 10 s starting from robot start position (x_0, y_0). Trajectory is generated using the robot kinematic model in (1-3) and locomotion control parameters (screw velocities and joint angles). Here, $\dot{\theta}_i + \epsilon_{\dot{\theta}_i(t,k)}$ (i=1,2,3,4) give the noisy screw velocity parameters. $\phi_i + \epsilon_{\phi_i(t,k)}$ (i=1,2,3) give the noisy joint angles. So, equation (7) uses noisy screw velocities and noise-free prefixed joint angles ϕ_i as locomotion control parameters, as needed for learning U_1. Equation (8) uses both noisy screw velocities and noisy joint angles as required for learning U_2. Based on the reached position $(x_{t,k}, y_{t,k})$ of the robot head at the end of this roll-out, final cost for this roll-out is calculated by evaluating the cost function as: $I_{t,k} = r(x_{t,k}, y_{t,k})$. In this way, all K noisy roll-outs from robot start position within one update process t are completed and corresponding $I_{t,k}$ is stored for each roll-out. Now, an exponential value is calculated on $I_{t,k}$ for each roll-out as:

$$S(\tau_{t,k}) = \exp^{-\lambda \frac{I_{t,k} - \min\limits_k(I_{t,k})}{\max\limits_k(I_{t,k}) - \min\limits_k(I_{t,k})}}. \tag{9}$$

The constant factor λ in equation (9) is identical in all our learning tasks: $\lambda = 30$. Now, the probability weighting $P_{t,k}$ for each roll-out is calculated based on above as follows:

$$P_{t,k} = \frac{S(\tau_{t,k})}{\sum_{l=1}^{K} S(\tau_{t,l})}. \tag{10}$$

Based on these weighting $P_{t,k}$, the corrections for the parameter vector are calculated as:

$$\triangle\dot{\theta}_i = \sum_{k=1}^{K} P_{t,k} \cdot \epsilon_{\dot{\theta}_i(t,k)}, \tag{11}$$

$$\triangle\phi_i = \sum_{k=1}^{K} P_{t,k} \cdot \epsilon_{\phi_i(t,k)}. \tag{12}$$

From equations (11), (12) the complete update vector is constructed as:

$$\triangle U_1 = [\triangle\dot{\theta}_1, \triangle\dot{\theta}_2, \triangle\dot{\theta}_3, \triangle\dot{\theta}_4], \tag{13}$$

or

$$\triangle U_2 = [\triangle\dot{\theta}_1, \triangle\dot{\theta}_2, \triangle\dot{\theta}_3, \triangle\dot{\theta}_4, \triangle\phi_1, \triangle\phi_2, \triangle\phi_3]. \tag{14}$$

The locomotion control parameter vector at the end of an update t is thus given by: ${U_1}^{(t)} = U_1 + \triangle U_1$ or ${U_2}^{(t)} = U_2 + \triangle U_2$. At the end of each update process t one noise-free trajectory with updated parameter vector ${U_1}^{(t)}$ or ${U_2}^{(t)}$ is simulated to calculate the noise-free cost $r_t = r(x_t, y_t)$, where, (x_t, y_t) is the reached robot head position with updated parameters. If the cost is smaller than a predefined threshold, no more update is required. Otherwise the whole update process is repeated for next update $t+1$. So, new parameter vector at the end of next update will be ${U_1}^{(t+1)}$ or ${U_2}^{(t+1)}$. This iterative loop continues until final learned values of parameter vector $U_1^{(n)}$ or $U_2^{(n)}$ are obtained that makes the robot head reach the target goal (x_G, y_G).

While updating U_1 and U_2 care has been taken to fix the range of joint angles and screw unit velocities in order to exclude robot instability situations. One such situation can be that the learned joint angles cannot make the robot go in a shape where at any instant $\phi_1 = \phi_2 = \phi_3 = 1.57$ rad (90°). For this, joint angles are limited within +1 rad and -1 rad. Further, screw-drive velocities are limited within +1 rad/s and -1 rad/s considering limitations. Thus, the search space is reduced and learned values of the parameters are obtained within the defined range to maintain robot stability. Hence, with this implementation locomotion control parameters like screw velocities and joint angles are learned by the robot to locomote toward a given goal.

IV. EXPERIMENTS AND RESULTS

For all following experiments, we select one parameter vector out of equations (4) and (5) depending on our learning task. The parameter vector is initialized to zero. Experiment 1 and 2 learn U_1 for fixed joint angles. Experiment 3 learns to control seven parameters and uses U_2 as the parameter vector. In all experiments testing on the real robot is also done. We first learn the control parameters by simulation and then transfer them to the real robot to test its behavior. The experiments are repeated for many different goals and multiple times to assure that stable learning is achieved. Robot length is around 0.9 m and all goal positions are in meter (m). Further, the screw-drive velocities are limited within ±1 rad/s. Joint angles are also limited within ±1 rad. In all experiments the robot starts from (0 m, 0 m) position. Snapshots of real robot experiments for different goals can be found in Fig 6 - Fig 9.

A. Learning Robot Control for Straight-Line Shape

In experiment 1 we restrict the robot shape to be prefixed as straight-line with ϕ_i(i=1, 2, 3) = 0 rad and four screw unit velocities $\dot{\theta}_i$ are learned for this body shape and a given goal. So, the parameter vector U_1 is learned. Fig 2 (a) shows the goal (-3 m, -3 m) with a small blue circle that is reached in this experiment. It can be seen that the robot (depicted by four attached dashed squares) starts from the position (0 m, 0 m) and finally reaches the goal. Fig 2 (b) shows learning curves

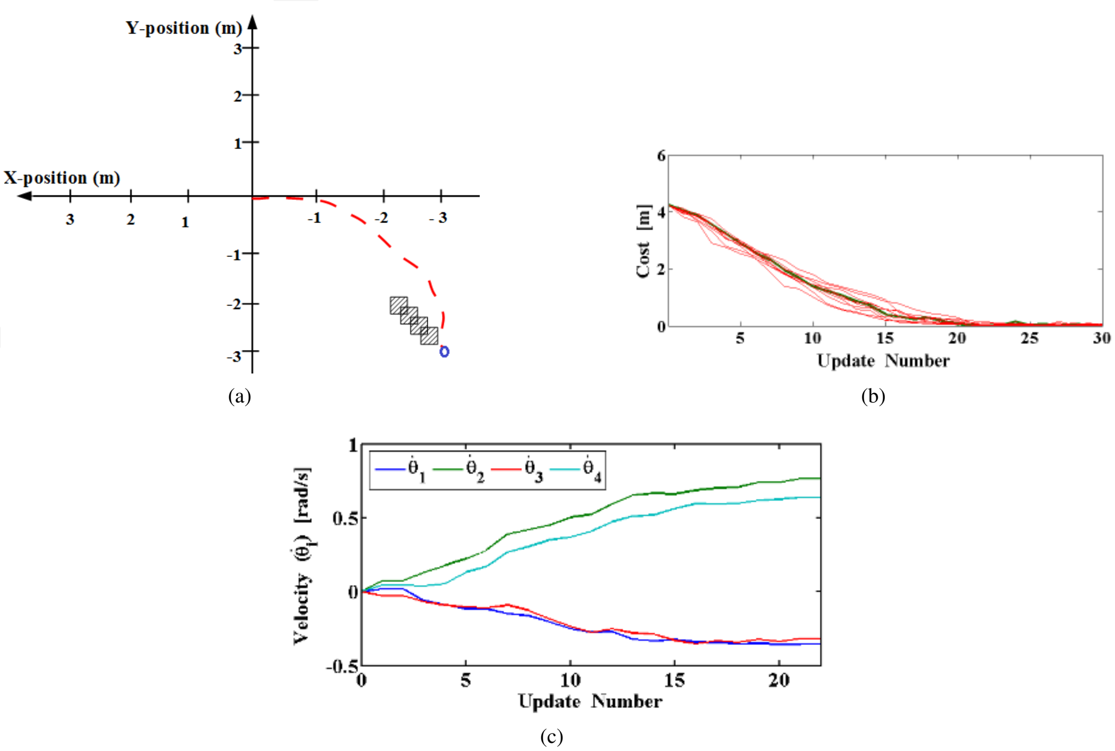


Fig. 2. Experiment 1: Straight-line body shape. (a) Robot head reaches the goal position (-3 m, -3 m) that is shown by a small blue circle. Start position is (0 m, 0 m). Final followed trajectory by the robot in red-dashed line. (b) 10 learning curves for this goal position with straight body shape showing convergence for every run. It takes around 20 updates for the average run in bold to converge to the lowest cost. (c) Learning of angular velocities for four screw units $\dot{\theta}_i$ (i=1,2,3,4). Learning gives final screw unit velocities and stabilizes after goal is reached at around 20 updates. Final values obtained at the end and convergence of learning are: $\dot{\theta}_1$ = -0.35 rad/s, $\dot{\theta}_2$ = 0.77 rad/s, $\dot{\theta}_3$ = -0.3 rad/s, $\dot{\theta}_4$ = 0.65 rad/s.

for this goal. The learning experiment is repeated 10 times for same goal and shows convergence to lowest cost every time. Fig 2 (c) gives the evolution of velocities during the learning process. Snapshots of the corresponding real robot experiment for this learning are shown in Fig 6. We invite readers to see also the supplementary video of this experiment at http://manoonpong.com/RAAD2014/suppl1.mpg.

B. Learning Robot Control for Any Fixed Shape

The experiment in this section demonstrates learning of four screw unit velocities $\dot{\theta}_i$ (i=1,2,3,4) when robot has any possible shape. Parameter vector learned is U_1. The presented experiment is done for the prefixed zigzag robot shape.

In experiment 2 the joint angles are prefixed as $\phi_1 = 0.5$ rad, $\phi_2 = -0.5$ rad, $\phi_3 = 0.5$ rad prior to learning to form a zigzag body shape and $\dot{\theta}_i$ is learned for this robot shape. The goal for this experiment is (2 m, -2 m), as shown in Fig 3 (a). It can be seen that the robot reaches the goal from start position (0 m, 0 m). Fig 3 (b) shows 10 learning curves for the same goal which converge to lowest cost every time. Fig 3 (c) shows how the velocities are learned for the given goal. Snapshots of the corresponding real robot experiment for this learning are shown in Fig 7. We invite readers to see also the supplementary video of this experiment at http://manoonpong.com/RAAD2014/suppl2.mpg.

C. Learning Robot Control - Generalized

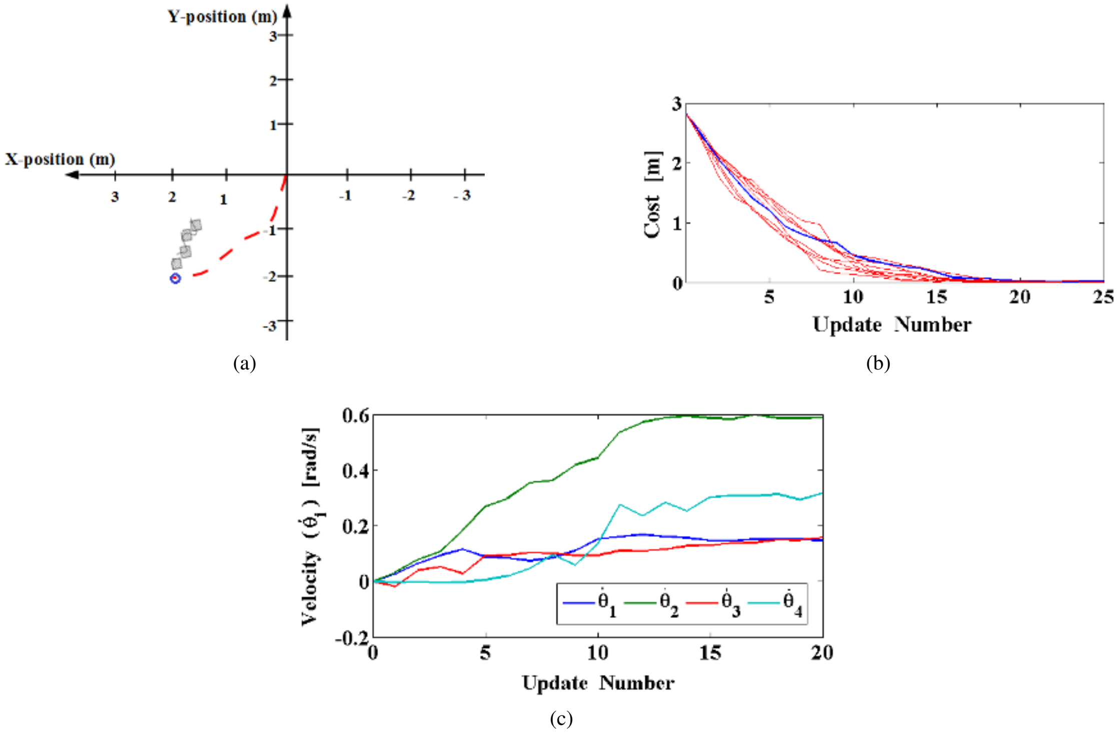
Here, the experiments demonstrate that the robot learns all of its seven control parameters, $\dot{\theta}_i$ (i=1,2,3,4) and ϕ_i (i=1,2,3) for locomoting toward a given goal. So, the parameter vector learned is U_2.

The goal for experiment 3 is (-3 m, -1 m), as shown in Fig 4 (a) with a blue circle. It can be seen that the robot starts from (0 m, 0 m) and finally reaches the goal (-3 m, -1 m) using learned robot control parameters. Fig 4 (b) shows learning curves for the same goal converging to lowest cost for all 10 runs. Fig 4 (c) shows evolution of the joint angles during learning process. Fig 4 (d) shows evolution of the screw-drive velocities during learning process. The learning of joint angles and velocities are seen to stabilize once convergence is achieved. Snapshots of the corresponding real robot experiment for this learning are shown in Fig 8. We invite readers to see also the supplementary video of this experiment at http://manoonpong.com/RAAD2014/suppl3.mpg.

Additionally, snapshots of another real robot experiment for goal (2 m, 2 m) where the robot also learns all seven control parameters for this goal are shown in Fig 9. We invite readers to see also the supplementary video of this experiment at http://manoonpong.com/RAAD2014/suppl4.mpg.

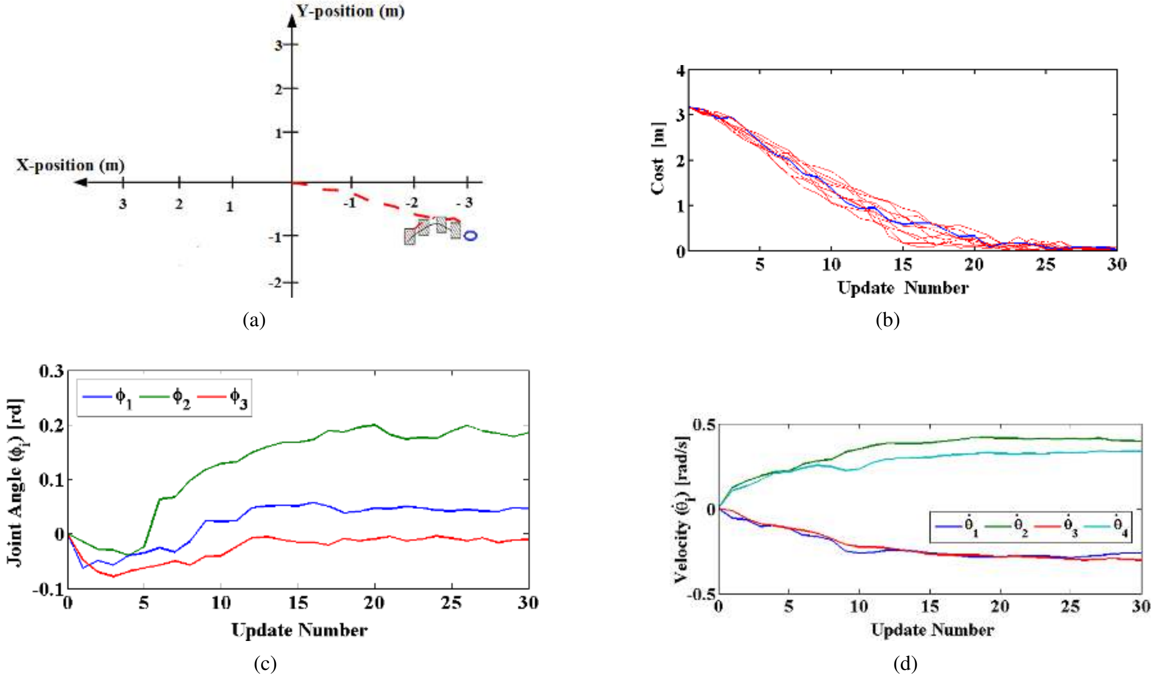
D. Different Starting Positions

Experiment 4 shows two scenarios for learning same goal (-2 m , 2 m) with straight-line robot shape but with different start positions. It shows that learning converges irrespective of the



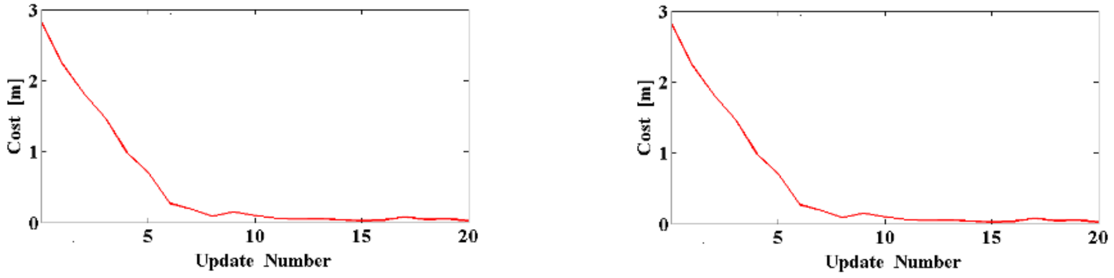
(a) (b) (c)

Fig. 3. Experiment 2: Zigzag body shape. (a) Goal position (2 m, -2 m) is shown by the small blue circle. The robot reaches the goal. Final followed trajectory by the robot shown in red-dashed line. (b) The 10 time run statistics of learning curves for this goal with shape as: ϕ_1 = 0.5 rad, ϕ_2 = -0.5 rad, ϕ_3 = 0.5 rad. Learning converges at around 16 updates for the average run in bold. Convergence to lowest cost is seen for all runs. (c) The learning of velocities $\dot{\theta}_i$ (i=1,2,3,4) of screw-drive units for this goal position and robot body shape. Learned values obtained at end of convergence are: $\dot{\theta}_1$ = 0.16 rad/s, $\dot{\theta}_2$ = 0.61 rad/s, $\dot{\theta}_3$ = 0.14 rad/s, $\dot{\theta}_4$ = 0.30 rad/s.



(a) (b) (c) (d)

Fig. 4. Experiment 3: Learning all seven locomotion control parameters. (a) Learned goal position is (-3 m, -1 m) shown with a small blue circle. Final followed trajectory (in red-dashed line) of the robot during locomoting toward the goal can be seen in this figure. (b) Learning experiment repeated 10 times for this same goal giving 10 learning curves. Convergence to lowest cost for every learning curve is observed. Learning curve of the average run is in bold and shows convergence at around 20 updates. (c) Three learned joint angles ϕ_i. Final learned values are: ϕ_1 = 0.04 rad, ϕ_2 = 0.16 rad, ϕ_3 = -0.02 rad. (d) Learned velocities of screw units $\dot{\theta}_i$. Final learned values are: $\dot{\theta}_1$ = -0.26 rad/s, $\dot{\theta}_2$ = 0.42 rad/s, $\dot{\theta}_3$ = -0.29 rad/s, $\dot{\theta}_4$ = 0.34 rad/s.



(a)

(b)

Fig. 5. Experiment 4: Different starting positions of the robot for the same goal is shown here. Learned goal position is (-2 m, 2 m) for both scenarios. (a) Learning curve for the goal converges at around 8 updates for start position (0 m, 0 m). (b) Learning curve for the same goal converges at around 17 updates for start position (3 m, 3 m). This shows that the learning mechanism can deal with different initial positions without any problem.

start position. For the first case robot starts from (0 m, 0 m) position and for the second it starts from (3 m, 3 m). For both scenarios same experimental set-up is employed. In both cases U_1 is initialized to zero and learned. In Fig 5 both scenarios are shown in (a) and (b). It can be seen that learning converges in both cases and the robot reaches the goal.

In the first scenario, shown in Fig 5 (a), for (0 m, 0 m) start position and (-2 m , 2 m) goal position learning converges at around 8 updates with initial cost of 2.8 m that has falls to almost zero. Learned velocities for this scenario are: $\dot{\theta}_1$ = -0.55 rad/s, $\dot{\theta}_2$ = 0.22 rad/s, $\dot{\theta}_3$ = -0.44 rad/s, $\dot{\theta}_4$ = 0.18 rad/s. In the second scenario, shown in Fig 5 (b), for (3 m, 3 m) start position and (-2 m , 2 m) goal position learning converges at around 17 updates with initial cost of 5 m that falls to almost zero with learning. Learned screw velocities for this second scenario are: $\dot{\theta}_1$ = -0.58 rad/s, $\dot{\theta}_2$ = 0.48 rad/s, $\dot{\theta}_3$ = -0.58 rad/s, $\dot{\theta}_4$ = 0.42 rad/s.

V. Conclusion

We have successfully generated goal-directed locomotion for the complex snake-line robot with screw-drive units by learning a proper combination of locomotion control parameters; i.e., screw velocities and joint angles using PI^2. Proper control parameters were also found when the robot was configured with different shapes (i.e., straight-line, zigzag, arc, etc.). Real robot experiments show that using learned control parameters enables the robot to successfully reach a given goal. In some experiments a small deviation (i.e., approximately 25 cm) from the goal is observed. The deviation is due to real conditions like friction, cabling, etc. In future, we are planning to handle such deviations by integrating sensory feedback to the robot in order to adjust its movements during locomotion. In this way, the robot will be able to perform online error correction. Taken together this study suggests how reinforcement learning approach with PI^2 helps to learn motor control in a coordinated way for a high degree-of-freedom system, like this nonstandard snake-like robot; thereby generating goal-directed locomotion.

Experimental results show that the robot can move in any direction with different body configurations by a proper combination of learned screw velocities and joint angles in order to reach different goals. In this way, a large set of behavioral primitives are generated. So, based on this framework, in the future we will use certain sets of learned control parameters as motor primitives with an additional mechanism and sensory feedback to properly chain or combine them for generalization as well as generating new locomotion behaviors in an online manner. This way the robot will be able to deal with an unknown situation, a complex environment, or morphological change (like, body damage).

Acknowledgment

This research was supported by Emmy Noether grant MA4464/3-1 of the Deutsche Forschungsgemeinschaft, Bernstein Center for Computational Neuroscience II Goettingen (BCCN grant 01GQ1005A, project D1) and by the HeKKSaGOn network.

References

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

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

[3] P. Liljebck, K.Y. Pettersen, . Stavdahl and J.T. Gravdahl, *A Review on Modeling, Implementation, and Control of Snake Robots*. Robotics and Autonomous Systems, 60(1), pp. 29–40, 2012.

[4] Z. Lu, S. Ma, B. Li, and Y. Wang, *Serpentine Locomotion of a Snakelike Robot Controlled by Cyclic Inhibitory CPG Model*. Proc. IEEE International Conference on Intelligent Robots and Systems (IROS 2005), pp. 96–101, 2005.

[5] K. Inoue, T. Sumi, and S. Ma, *CPG-based control of a simulated snake-like robot adaptable to changing ground friction*. Proc. IEEE International Conference on Intelligent Robots and Systems (IROS 2007), IEEE, pp. 1957–1962, 2007.

[6] G. Miller, *Snake robots for search and rescue, in: Neurotechnology for Biomimetic Robots*. MIT Press, Cambridge, MA, USA, London, pp.271–284.(Chapter), 2002.

[7] H. Fukushima, S. Satomura, T. Kawai, M. Tanaka, T. Kamegawa, and F. Matsuno, *Modeling and Control of a Snake-Like Robot Using the Screw-Drive Mechanism*. IEEE Trans. Robot., vol. 28, no. 3, pp.541–555, 2012.

[8] E.Theodorou, J.Buchli and S.Schaal, *A Generalized Path Integral Control Approach to Reinforcement Learning*. Journal of Machine Learning Research, vol. 11, pp. 3137–3181, 2010.

[9] E.Theodorou, J.Buchli and S.Schaal, *Reinforcement learning of motor skills in high dimensions: A path integral approach*. Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 2397–2403, 2010.

[10] F.Stulp and O.Sigaud, *Robot Skill Learning: From Reinforcement Learning to Evolution Strategies*. Paladyn, Journal of Behavioral Robotics, vol. 4, issue 1, pp. 49–61, September 2013.

[11] F.Stulp and S.Schaal, *Hierarchical Reinforcement Learning with Movement Primitives*. 11th IEEE-RAS International Conference on Humanoid Robots, pp. 231–238, 2011.

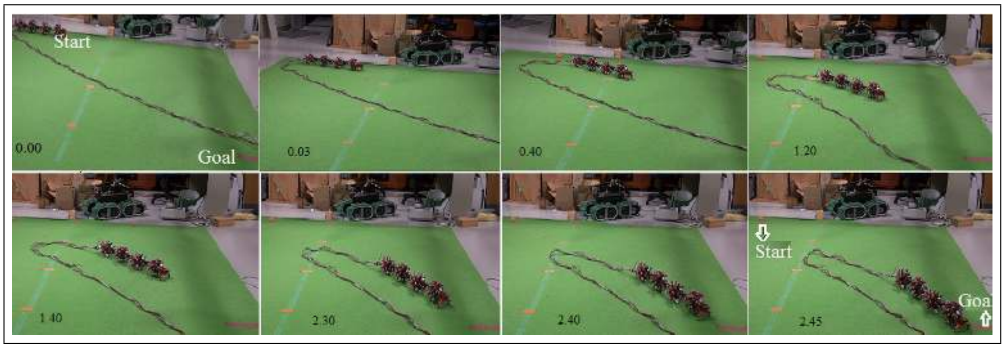


Fig. 6. Snapshots of real robot experiment for goal (-3 m, -3 m) where the robot locomotes toward the goal with a straight-line body shape. Start position is (0 m, 0 m). The robot reaches the goal with learned screw velocities obtained using PI^2. A small deviation of about 25 cm from the goal is observed.

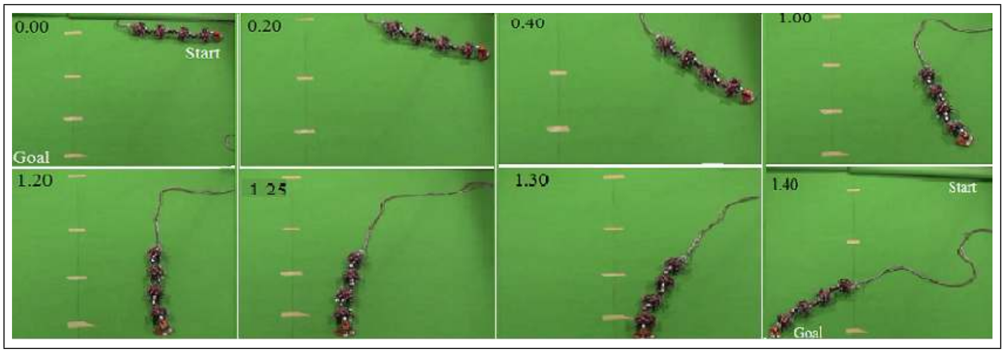


Fig. 7. Snapshots of real robot experiment for goal (2 m, -2 m) where the robot locomotes toward the goal with a zigzag body shape. Start position is (0 m, 0 m). The robot reaches the goal with learned screw velocities obtained using PI^2.

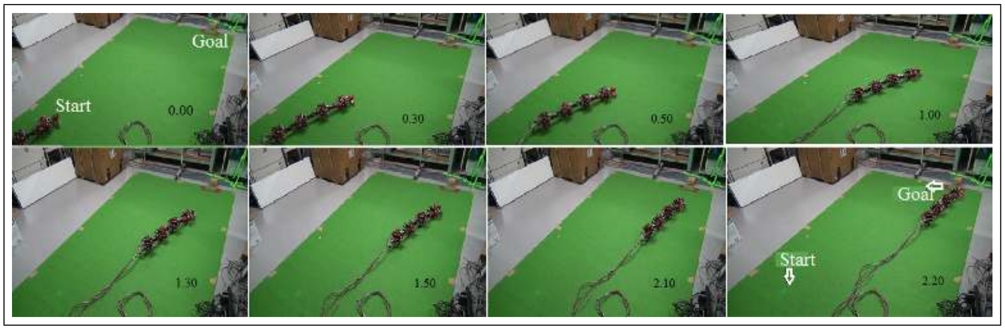


Fig. 8. Snapshots of real robot experiment for goal (-3 m, -1 m) where the robot locomotes toward the goal when all seven locomotion control parameters, four screw velocities and three joint angles, are learned using PI^2. Start position is (0 m, 0 m). The robot reaches the goal with learned screw velocities and joint angles.

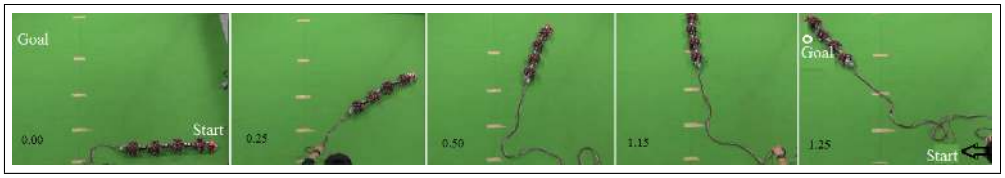


Fig. 9. Snapshots of real robot experiment for goal (2 m, 2 m) where the robot locomotes toward the goal when all seven locomotion control parameters, four screw velocities and three joint angles, are learned. Start position is (0 m, 0 m). The robot reaches the goal with learned screw velocities and joint angles obtained using PI^2. A small deviation of about 20 cm from the goal is observed.