Article

Continual Learning for Addressing Optimization Problems with a Snake-Like Robot Controlled by a Self-Organizing Model

Jong-Chen Chen

Department of Information Management, National Yunlin University of Science and Technology,
Yunlin 64002, Taiwan; jcchen@yuntech.edu.tw; Tel.: +886-921-717-966

Received: 3 June 2020; Accepted: 10 July 2020; Published: 15 July 2020

Abstract: We have entered a new era, “Industry 4.0”, that sees the overall industry marching toward an epoch of man–machine symbiosis and intelligent production. The developers of so-called “intelligent” systems must attempt to seriously take into account all possible situations that might occur in the real world, to minimize unexpected errors. By contrast, biological systems possess comparatively better “adaptability” than man-made machines, as they possess a self-organizing learning that plays an indispensable role. The objective of this study was to apply a malleable learning system to the movement control of a snake-like robot, to investigate issues related to self-organizing dynamics. An artificial neuromolecular (ANM) system previously developed in our laboratory was used to control the movements of an eight-joint snake-like robot (called *Snaky*). The neuromolecular model is a multilevel neural network that abstracts biological structure–function relationships into the system’s structure, in particular into its intraneuronal structure. With this feature, the system possesses structure richness in generating a broad range of dynamics that allows it to learn how to complete the assigned tasks in a self-organizing manner. The activation and rotation angle of each motor are dependent on the firing activity of neurons that control the motor. An evolutionary learning algorithm is used to train the system to complete the assigned tasks. The key issues addressed include the self-organizing learning capability of the ANM system in a physical environment. The experimental results show that *Snaky* was capable of learning in a continuous manner. We also examined how the ANM system controlled the angle of each of *Snaky*’s joints, to complete each assigned task. The result might provide us with another dimension of information on how to design the movement of a snake-like robot.

Keywords: continuous optimization; computational intelligence; evolutionary learning; self-organizing dynamics

1. Introduction

The advancement of artificial intelligence, especially deep learning, coupled with the recent development of the Internet of Things, has once again received great attention by the public. Robots with autonomous learning capabilities have become one of the most important robotics trends throughout the world. However, the performance of a robot operating in the actual environment, regardless of how well it is designed, is sometimes quite different from the expected performance in its design environment. To minimize this difference, so-called “smart systems” should possess a certain degree of self-correction or self-learning capability. In response to the need to correct this problem, most traditional artificial intelligence approaches must carefully consider and evaluate all possible situations in real world operations, to ensure that everything is under control. Without doubt, in such conditions, a carefully designed system should perform as expected, as all the problems related to the operational environment

should have been clearly defined in a predetermined manner. However, as we all know, most traditional artificial intelligence systems, based on people's "rigid appeal", are realized with computer programs or codes. That is, in terms of programming, roboticists must accurately design the required objects (or "symbols") and the rules (algorithms) that operate on these objects. A noticeable problem is that, when the code of a computer system is slightly modified, the results (system functions) it produces may be quite unpredictable.

Higushi et al. [1] therefore emphasized that people should think about how to add plasticity characteristics into a computer system, to increase its malleability. Similarly, Vassilev et al. [2] put forward the so-called "neutral principle", which states that, when adding a number of similar functions into a system, the possibility of providing solutions increases, as well. Thompson and Layzell [3] suggested that relative adaptability might be improved if strict requirements for computer systems could be appropriately released.

Compared with computer systems, biological systems exhibit a relatively better "adaptation" capability, as they are comparatively more capable of continuing to function (or operating) in an uncertain or even in an unknown environment. This is because the fitness of organisms can be thought of as presenting with a relatively gentle surface; that is, its function (fitness) changes gradually when the structure changes slightly.

The artificial neuromolecular (ANM) system [4–6], a biologically motivated information processing architecture, possesses a close "structure/function" relation. Unlike most artificial neural networks, its emphasis is on intra-neuronal dynamics (i.e., information processing within neurons). Through evolutionary learning, we have shown that the close "structure/function" relation facilitates the ANM system in generating sufficient dynamics for shaping neurons into special input/output pattern transducers that meet the needs of a specific task [4,5].

In this study, the ANM system was linked with the navigation problem of a snake-like robot. It should be noted that our goal was absolutely not to build a state-of-art snake-like robot but to use it as a test domain for investigating the perpetual learning capability of the system. More importantly, we are comparatively more interested in addressing the issue related to learning in an uncertain or unknown environment, the so-called adaptability problem proposed by Conrad [7]. An ambiguous or ill-defined problem domain would fall into the category of an unknown or partially known environment. A noisy or changing domain would fall into the category of a statistically uncertain or unpredictable environment.

Previously, the research team of this group used a quadruped robot controlled by the ANM system, to address the issue of adaptability [6]. The movement of the robot was sometimes uncertain, as the robot was composed of wooden strips fused together with simple disposable chopsticks. Apart from the robot itself, controlling the robot to move forward or to make a left/right turn was also uncertain due to variable friction between the robotic legs and the test ground. Another constraint of the robot was that each leg was only allowed to produce a limited front and rear swing. The only way for the robot to generate appropriate movements was to constantly change its center of gravity during the movement process. Finding changes in the center of gravity at different times may not be possible in a simple prior calculation. The sum of all constraints put on the test made it infeasible for system designers to find solutions to solve the movement problems in advance, even through careful design and analysis.

In nature, some animals can move quickly on land and in the water, without feet, which is hard for humans to imagine and understand. What is even more surprising is that these animals show considerable adaptability to different environments and terrains. Because of this, some scholars have put considerable effort into understanding the snake's movement and trying to create a robot that mimics it. The steps involved are to first study the snake's motion curve and then to apply mechanical operation principles to create snake-type motion [8]. These steps include deriving the velocities of different snake segments to perform rectilinear motion [9–11], using the position of the motor and leaning against the rotation of the wheel to cause the snake-shaped robot to move

forward [12], and using Watt-I planar linkage mechanism to control a biped water-running robot to generate propulsion force [13]. In short, most of the above research involves designing and calculating the operation of snake-type robots by studying the operating principle and motion patterns of these robots. Moreover, as pointed out by Liljebäck et al. [14], the majority of literature on snake robots so far has focused on locomotion over flat surfaces. However, in the real world, there are many unidentified or unknown external factors (noise). Small changes in these factors can cause a well-designed system to become completely inoperable. Shan and Koren [15] have proposed a motion planning system for a mechanical snake robot to move in cluttered environments, without avoiding obstacles on its way; the robot would instead “accommodate” them by continuing its motion toward the target, while being in contact with the obstacles.

Unlike the above study from our group, here, through an autonomous learning mechanism, we trained a snake-type robot (herein referred to as *Snaky*) to learn how to complete the specified snake mission. The core learning mechanism (*Snaky*’s brain) is the artificial neuromolecular system (ANM) developed by one of the authors Reference [4] decades ago. We note that this robot has eight joints and four control rotary motors. Initially, the ANM system randomly generates different parameter sets for motor control. For each learning session, the system evaluates the performance of each set, selects some of the best-performing sets, and finally copies from the best-performing sets to lesser-performing sets, with some alterations. The learning continues until the robot completes the assigned tasks or is stopped by the system developer. It should be noted that *Snaky* is definitely not a high-precision robot; that is, given the same input, its output behavior may be different for each run of experiments. However, we must emphasize that the goal of this research was definitely not to build a state-of-art snake-type robot. Instead, we were trying to use the uncertainty of the robot, as well as its interaction with the environment, as a test bed for studying the continuous optimization problem in the ANM system. Basically, if the entire system is capable of showing that it learns in a continuous manner when the complexity of the task assigned increases, we can call it a “success”.

Section 2 introduces the architecture of the proposed model, the evolutionary learning mechanisms, our application domain, and the input–output interface. Section 3 discusses the experimental results. The final section presents the concluding remarks.

2. ANM Architecture

2.1. General Overview

The concept of the entire ANM system comes from the operation of the brain’s nervous system [4,16,17]. The main structure of the ANM system is a central processing subsystem (CPS) consisting of a group of neurons that emphasize memory manipulation (MM) and another group of neurons that emphasize information processing (IP). An IP neuron is responsible for integrating signals from different external neurons in space and time into a series of spatiotemporal output signals. The role of an MM neuron is to select appropriate IP neurons that would engage in input/output signal transduction (note: the neuronal activity of other unselected IP neurons is ignored). The ANM system combines these two neurons to produce an effective collaborative learning mechanism [4]. Figure 1 shows the conceptual architecture of the ANM system.

Several scholars [18–20] have proposed that the cytoskeleton may play a role in signal integration. The IP neurons in this study are motivated by the dynamics that reflect molecular processes believed to be operative in real neurons, in particular processes connected with second messenger signals and cytoskeleton–membrane interactions. However, it must be emphasized that the dynamics within the neuron are more complex than are currently understood. If we want to understand and simulate these dynamics in depth, the whole study would be forced to purely emphasize mimicking the internal dynamics of neurons, which would require considerable computing resources. The method used in this study extracts the information processing inside the neurons in a focused and functional way,

assuming that there are different constituent molecules on the cytoskeleton and that they have different transmission speeds and different degrees of mutual influence.

Figure 1. Architecture of the artificial neuromolecular (ANM) system.

2.2. Conceptual Architecture of an IP Neuron

The cytoskeleton of an IP neuron is represented by a 2D wrap-around cellular automaton [19,20], as shown in Figure 2. Each grid cell may have some basic cellular molecules (CUs), each of which represents a unit of signal transmission and information integration. It is currently assumed that there are three types of CUs (represented by C_1 , C_2 , and C_3). There are six possible states of a CU: quiescent (q_0), active with increasing levels of activity (q_1 , q_2 , and q_3), activating (q_4), and refractory (q_r). A CU in activating state q_4 will return to state q_r at the next update time, which in turn will go to state q_0 at its following update time. A CU in state q_r is not affected by its neighbors, ensuring unidirectional signal propagation. A CU in any of the active states will go to its comparatively lower state at the next update time if it receives no signal (e.g., it goes to state q_2 if it was in state q_3 , to state q_1 if in state q_2 , and to state q_0 if in state q_1). Except if it is in state q_r , a CU will enter state q_4 if a neighboring CU of the same kind is in state q_4 .

This study also assumes that two adjacent CUs of different types can interact with each other through a linker protein (MAP), depending on the nature of the molecular types. To design the signal integration capability of a molecule, two arbitrary assumptions are made. One is that an activating C_1 -type molecule has the greatest influence on the other two types (C_2 and C_3) of molecules. In contrast, an activating C_3 -type molecule has the least influence on the other two types (C_1 and C_2) of molecules. The influence of a C_2 -type molecule is in between that of the other two types (C_1 and C_3). The other assumption is that C_3 -type molecules have the fastest signal transmission speed, whereas C_1 -type molecules have the slowest signal transmission speed. In summary, C_1 -type molecules possess the slowest signal transmission but have the greatest influence on other molecules, whereas C_3 -type molecules possess the fastest signal transmission but have the least influence on other molecules.

In the cytoskeleton, each cellular molecule may be associated with a read enzyme, which is responsible for receiving external signals. Whenever a read enzyme receives an external signal, it is activated and simultaneously activates the cellular molecules at the same location, thereby activating a

signal in the cytoskeleton. In addition, the neuron fires when certain combinations of signals arrive at the place where a *readout* enzyme is located (only for C_1 -type molecules in the present implementation).

Figure 2. Conceptual architecture of an information processing (IP) neuron. Each grid location, referred to as a site, has at most one of three types of components: C_1 , C_2 , or C_3 . Some sites may not have any component at all. A *readin* enzyme converts an external signal into a cytoskeletal signal. Specific combinations of cytoskeletal signals will activate a *readout* enzyme, which in turn causes the neuron to fire. The neighbors of an edge site are determined in a wrap-around fashion. An *MAP* links two neighboring components of different types together.

2.3. Evolutionary Learning at the IP Neuron Level

In the present implementation, there are 256 IP neurons, which are divided into eight subnets, each comprising 32 IP neurons. The input/output interface of these subnets (to be described in Section 3) is the same so that each subnet will receive the same input signals and generate the same output behavior if the firing pattern of the IP neurons is also the same. The algorithm of the evolutionary learning at the IP neuron level is shown in Figure 3. Evolutionary changes at each level (parameter) are summarized as follows:

- (1) At the *readout* enzyme pattern, a change at this level may alter the pattern of inputs that determines the initiation of cytoskeletal signals inside an IP neuron.
- (2) At the C_i -type pattern (increasing or decreasing the number of its constituent molecules), a change at this level may alter the signal configuration of an IP neuron.
- (3) At the *MAP* pattern, a change at this level may alter the influence of different types of cytoskeletal signals, which in turn may modify the signal configuration of an IP neuron.
- (4) At the *readout* enzyme pattern: a change at this level may alter the output of an IP neuron.

At present, we have configured the operation mode of the system to allow only one level of evolutionary learning at a certain time, and all other levels of learning are turned off. After learning for a fixed period of time, the system will turn off this level of learning and turn on other levels of learning. In this way, the system takes turns opening each level of learning.

1. **Generate** at random the initial *MAP*, *component-type*, *readin enzyme*, and *readout enzyme* patterns of each neuron of each subnet.
2. **Evaluate** the performance of each competing subnet and **select** the three best-performing subnets.
3. **Copy** the *MAP*, *component-type*, *readin enzyme*, and *readout enzyme* patterns of each neuron from any one of the three best-performing subnets to a lesser-performing subnet.
4. **Vary** the *MAP*, *component-type*, *readin enzyme*, and *readout enzyme* patterns of each neuron in the lesser-performing subnets.
5. **Go to Step 2** unless the stopping criteria are satisfied.

Figure 3. Evolutionary learning at the IP neuron level.

2.4. Evolutionary Learning at the MM Neuron Level

As mentioned earlier, the role of the control neurons in the CPS is to select appropriate IP neurons to participate in the integration of information. The role of MM neurons is to control IP neurons in the manner that only the neurons selected are allowed to be controlled in order to engage in input/output information processing (otherwise the information processing of unselected neurons will be ignored). The whole architecture (Figure 4) of the control mechanism is divided into three layers: high-level MM neurons, low-level MM neurons, and information processing neurons. Each low-level MM neuron is responsible for controlling the corresponding IP neurons in different subnets (this control relationship does not change during the learning process). Each subnet has 32 IP neurons, and therefore, there are a total of 32 low-level MM neurons. Evolutionary learning at the MM level is shown in Figure 5.

Figure 4. The architecture of the control mechanism of memory manipulation (MM) neurons.

Figure 5. (a) Low-level MM neurons controlled by each high-level MM neuron are activated in sequence for evaluating their performance. Assume the MM neurons controlled by R_2 achieve better performance. (b) The pattern of neural activities controlled by R_2 is copied to R_1 . (c) R_1 controls a slight variation of the neural grouping controlled by R_2 , assuming some errors occur during the copy process.

3. Problem Domain

3.1. Snaky's Structure

The snake-type robot used in this study (Figure 6) has a total of eight joints. Each joint has two degrees of freedom (DOFs), which are controlled by two motors (one responsible for horizontal motion and the other for vertical motion). In this study, only four horizontally controlled motors in the eight joints of the snake-type robot were used.

Figure 6. Snake robot and two orthogonally connected servo motors.

The snake-type robot is equipped with a 32-bit Cortex M3 core Armino servo motor controller, a single-board computer that accepts (reads) the motor control signals (the rotation angles of the four motors) sent from the ANM system through a Bluetooth communication module (HL-MD08R-C1A, UART Bluetooth module). The latter transmit these signals to each motor in sequence to control the movement of the *Snaky* robot. In this study, a wireless mouse was placed under the head of *Snaky*, to capture its coordinates every 100 milliseconds, from which we plotted the movement trajectory and measured the distance it moved. The distance will be used for fitness evaluation.

3.2. Interface between ANM and Snaky

As mentioned earlier, the ANM system has 256 IP neurons, divided into eight comparable subnets. All the IP neurons in different subnets that are similar in terms of their inter-neuronal connections and intra-neuronal structures are grouped as a bundle. The 256 IP neurons could thus also be grouped into 32 bundles. In addition, the neurons in the same bundle also have the same patterns of connections with the external inputs and with the motors (Figure 7). The above setup ensures that each subnet receives the same inputs from the receptor neurons and that the outputs of the system are the same when the firing patterns of each subnet are the same.

Figure 7. Interface between the ANM system and *Snaky*.

In the following text, we will explain how to map (transform) the outputs of the ANM system to rotation angles for the four motors. As mentioned earlier, the IP neurons of the ANM system are grouped into 32 bundles that are further divided into four classes, with each class controlling the rotation of a single motor. In our present implementation, the rotation angle of each motor was determined by the time difference between the first two firing IP neurons of the same class. The greater the time difference is, the greater the angle at which the motor rotates. A simple implementation would involve a linear relationship between the rotation angle and the time difference. However, in such a design, changes in the angle of each motor would be very sensitive to the time difference. Therefore, our somewhat arbitrary assumption was that the motor rotation change and the time difference should exhibit a nonlinear relationship similar to a sigmoid function (Equation (1)).

$$\text{Rotation angle} = ((1/(1 + e^{-(15 * \Delta t)})) - 0.5) * 2 * (A_{\max} - A_{\min}) + A_{\min} \quad (1)$$

where Δt is the time difference of the first two firing neurons of the same class, and $A_{\max}$ and $A_{\min}$ are the maximum and minimum angles of rotation, respectively, for each motor.

Due to the limitations of the Armino servo motor controller, only one motor can be activated at a time (the operating time of each motor is the same). Thus, these four motors are activated in sequence (one at a time). The activation sequence of the four motors (M_1 , M_2 , M_3 , and M_4) currently used in this study starts with M_1 and proceeds to each subsequent motor, until the last motor, M_4 , is activated. The entire process from activating M_1 to activating M_4 is referred to as an operational step. For each run of the experiment, we observed *Snaky*'s performance after it completes four operational steps. Once the ANM system determines the set of rotation angles for the four motors, they are sequentially assigned to each motor. However, the assignment is not constant across all operation steps but instead is implemented in a rolling manner. For example, as shown in Table 1, if the ANM system determines that the angles of rotation for the four motors are A_1 , A_2 , A_3 , and A_4 , the four motors, M_1 , M_2 , M_3 , and M_4 , will be A_1 , A_2 , A_3 , and A_4 , respectively, in the first operational step, S_1 . In the second operational step (S_2), M_1 , M_2 , M_3 , and M_4 will instead be assigned A_2 , A_3 , A_4 , and A_1 , respectively. It is emphasized here that there is no particular academic or physical justification for the abovementioned setup. However, the adoption of this randomness provides a test environment for this study, to explore whether the ANM system has an autonomous learning ability to complete the assigned tasks, allowing the snake to advance toward the assigned target without predetermining its solution in advance. From another perspective, although this randomness is not supported by prior research, the answers found by the system may provide insight into another way of using machines to produce snake-like motions. As stated earlier, the ANM system had eight competing subnets that each must perform independently. For each run of the experiments, we had to ensure that the setup of the environment and *Snaky* should be the same. That is, both the environment and *Snaky* were set up manually. We identified the completion of the assessment of the eight subnets as a learning cycle. Each of the experiments (to be described in the next section) was terminated after 64 cycles. In total, 512 runs were required for each experiment, and its total physical time was approximately 8 h.

Table 1. Rotation angle of each individual motor for each operational step.

Step \ Motor	M_1	M_2	M_3	M_4
S_1	A_1	A_2	A_3	A_4
S_2	A_2	A_3	A_4	A_1
S_3	A_3	A_4	A_1	A_2
S_4	A_4	A_1	A_2	A_3

4. Experiments

Two types of experiments were performed in this study. The first asked *Snaky* to move to a certain target position. In this study, three independent tasks were performed separately. One asked *Snaky* to move to position *L* (i.e., move to the left), another to move to position *M* (i.e., move forward), and the other to move to position *R* (i.e., move to the right). The second asked *Snaky* to move to a certain target position, as described above, but with some limitations on the range of motor rotation angle. Our intuition is that the second experiment would be comparatively more difficult than the first one because the possible solutions that the ANM system could explore are relatively limited when more constraints are put on the system. Because of this, in the second type of experiment, the distances of the target positions were comparatively shorter than those in the first experiment. Consequently, we also had the robot perform three independent tasks in this second type of experiment. One asked the robot to move to position *l* (i.e., move to the left), the second to move to position *m* (i.e., move forward), and the third to move to position *r* (i.e., move to the right). Figure 8 shows the position of the above six target locations in a two-dimensional space. For all the above experiments, the fitness of the system is based on the distance between the snake robot and its designated target. The shorter the distance is, the better the performance of the system.

Figure 8. Positions of the six target locations.

4.1. General Learning

The goal of this experiment was to train *Snaky* to move to each of these three assigned positions (i.e., positions *L*, *M*, and *R* in Figure 8) independently. Figure 9 shows that the learning performance of *Snaky* varied as learning preceded. There are two reasons for this behavior. One of the reasons is that *Snaky* itself is not a highly accurate machine. The other is that *Snaky*'s test floor was not a completely flat platform but instead a lattice with some kind of tidal groove. Based on the abovementioned reasons, the contact friction between *Snaky* and the floor was somewhat uncertain. Therefore, for each run of the experiment in the learning process, the resulting output may be different, even if all variables were controlled under the same conditions. However, even though the performance oscillates, it falls mostly within a certain range of variations. The most important thing to note is that the learning curve of the robot gradually improves over time. This implies that the system can not only overcome the noise but also shows the ability to continuously learn. For each set of motor rotation angles obtained for each of the above experimental results, we repeated the test times times, to check the similarity of *Snaky*'s movement trajectories. The results (Figure 10) show that when *Snaky* uses the same set of motor rotation angles, it can advance toward its assigned target and produce a similar movement trajectory.

Figure 9. Learning to move to the left, forward, and to the right.

Figure 10. *Snaky* movement trajectories.

In the following section, we analyse the rotation angles of the four motors obtained from each of the above experimental results. Table 2 lists the rotation angle of each motor used by *Snaky* toward each of the three designated positions after 64 generations of learning. If the angle of rotation was negative, the motor was turned to the left (only four of the horizontally controlled motors in the eight joints of the snake-type robot were used). In contrast, if it was positive, the motor was turned to the right. The experimental result shows that *Snaky* will advance toward the designated target with a combination of relatively large angles of motor rotations. In other words, it will rotate at a large angle in one direction and then at a large angle in the other opposite direction. As shown in Table 2, in the turning left task, *Snaky* first turned to the right at a large angle and then turned to the left at a large angle. The above results are very similar to the results obtained from the task where *Snaky* turned to the right (where *Snaky* turned left at a large angle and then right at a large angle). This result can be explained by the fact that *Snaky* takes a similar approach to the motion of swaying to produce forward motion. The most interesting result is the forward-target task. The results show that *Snaky* uses two consecutive “first right to left” cross-angle rotations. This practice can be explained by the fact that it swings left and right with large angles, to produce forward motion, and simultaneously uses an almost equal left and right swing angle to achieve unbiased forward movement.

Table 2. Rotation angle of each individual motor.

Motor	M_1	M_2	M_3	M_4
<i>L</i>	81.8	−23.2	−89	−6
<i>M</i>	78	−76.7	77.2	−71.3
<i>R</i>	32.9	89.9	−59.3	−88

As mentioned earlier, for each run, we allowed *Snaky* to complete the four operational steps of the action and then evaluated its mobile performance (i.e., learning performance). The practical approach taken was to sequentially assign the rotation angles of the four motors obtained via the ANM system to the individual motors of each operational step in a rolling manner. When we combine the results of the four motor rotations assigned to the four operating steps in turn, the results are even more impressive. For example, in the case of moving toward the *L* position, at S_1 , M_1 first approaches the right turn at

approximately 80 degrees, and then M_2 and M_3 turn left by approximately 80 degrees. As a result of the above combination, as shown in Figure 11, *Snaky* changes with relatively large angles, to produce a moving action similar to the English letter V. Similarly, at S_2 , M_2 first rotates at a large angle of approximately 80 degrees with a left turn and then M_3 and M_4 rotate to the right at an approximately 80-degree angle. The result of the combination of S_1 and S_2 is a large left turn, followed by a large right turn, to produce a movement and balance effect. Similarly, from S_3 to S_4 , *Snaky* first generates a large angle right turn with the M_1 and M_2 motors and then mixes M_3 and M_4 to turn left, at a large angle, to form a final left turn movement. The combination of the rotation angles of all four motors (M_1 , M_2 , M_3 , and M_4) forms an inverted V-shape.

Figure 11. Rotation angles (y -axis) of different motors (x -axis) obtained via the ANM system (general learning).

Another task in this study asked *Snaky* to move toward the R position. The results show that, similar to the previous task, the result of training through the ANM system caused M_1 , M_2 , M_3 , and M_4 to produce relatively large angles to make *Snaky* rotate in one direction and then to rotate with opposite angles to cause movement in the opposite orientation. The difference between the movements for the L and R tasks is that *Snaky* turns to the left first at Step 1, S_1 , and then generates a right turn when at S_2 and S_3 , and finally turns left slightly at S_4 (note: the result of correcting an excessive right turn). In terms of overall operational action, *Snaky* produces a right-turning motion shift.

The last task required *Snaky* to move straight toward the front M position. As a result, it is shown that the four motors (M_1 to M_4) form an uppercase English letter N -shape at S_1 , an inverse N -shape at S_2 , an N -shape at S_3 , and an inverse N -shape at S_4 . This result shows that *Snaky* uses an N -shape and an inverse N -shape in an interlaced way, to generate forward motion.

4.2. Constrained Learning

Unlike the above experiment, the goal of this experiment was to train *Snaky* to move to the assigned locations with limitations on the range of motor rotation angles. Our intuition was that the second experiment would be more difficult than the first one because the possible solutions from which the ANM system can search are relatively limited. In this experiment, independent tasks were performed to reach three different target locations (i.e., positions *l*, *m*, and *r*), as shown in Figure 8.

Similar to the first experiment, the learning performance oscillates but falls mostly within a certain range of variations in the learning process, even if all variables are controlled under the same conditions (Figure 12). A particularly striking difference from the previous experimental results can be seen in the task where *Snaky* is required to move forward. Figure 12 shows that its learning performance decreases at a relatively slow rate. When we compare the distances actually moved, it can be clearly seen that, when the snake's motor rotation angle is limited, it is relatively difficult to learn how to reach the target, because of the small range of solutions that can be searched. However, what is certain is that the system can still perform continuous learning. In the process of learning, the performance of the system increases and decreases, but as the number of learning cycle increases, the performance of the system shows a trend of continuous improvement. Overall, through autonomous learning, the ANM system can find a set of rotation angles in cooperation with the four motors, to move the snake robot to a specified target point.

Figure 12. Learning to move to the left, forward, and to the right.

Table 3 lists the rotation angle for each motor used by *Snaky* to reach each of the three designated positions after 64 generations of learning. As with the first experiment, we tested the motor rotation angles five times for each task through the ANM system. The results show that, when *Snaky* uses the same rotation angles, it can advance toward its assigned target and produce a similar movement trajectory each time (Figure 13).

Table 3. Rotation angle of each individual motor.

Motor	M_1	M_2	M_3	M_4
<i>l</i>	−44.9	43.1	12.5	35.7
<i>m</i>	−44.1	44.9	38.5	5.2
<i>r</i>	42.2	44.9	−44.5	−2.9

Figure 13. Snaky movement trajectories.

When the range of motor angles is limited, the experimental results (Figure 14) show that the four motors (M_1 , M_2 , M_3 , and M_4) are rotated in a manner similar to the English letter *N*, *V*, or *U* (as well as in an inverse *N*-, inverse *V*-, or inverse *U*-shape). When the task involves moving toward position *l*, from S_1 to S_4 , the combination of the rotation angles for the four motors (M_1 , M_2 , M_3 , and M_4) forms an uppercase English letter *N*-shape or an inverse *N*-shape.

Figure 14. Rotation angles (y -axis) of different motors (x -axis) obtained via the ANM system (constrained learning).

In contrast, when moving toward position *m* or toward position *r*, the combination of angles consists of *N*-, *V*-, or *U*-shaped rotations (including inverse *N*-, *V*-, or *U*-shapes). In other words, the way the robot moved did not resemble the way it moved in the previous experiment.

5. Conclusions

In recent years, the use of “machine learning” in the processing of big data has increased everyone’s attention to artificial intelligence. Autonomous learning plays a very important role in the field of artificial intelligence, especially in the case of a problem that is difficult to solve with systemic algorithms. This study uses a snake-type robot to explore the design of ground motion problems, which is somewhat compatible with the difficulty of obtaining a solution to a problem in advance by

using a systematic algorithm. First, *Snaky* is not a high-precision robot, and the angles of rotation of each motor have some degree of error; second, the floor on which *Snaky* performed its tasks was not flat. During each movement, it came into contact with elements with different resistances, due to the different moving positions. Because of this, the resistance faced by snake-type robots in contact with the ground is not uniform but varies over time. Combining the above two factors, this study wants to emphasize again that *Snaky* itself and its interactions with the environment contend with uncertainty of considerable interference, which provides a suitable experimental platform for us to explore the issue of autonomous learning.

This study applied a molecular-like neural system and an evolutionary learning algorithm to allow a snake-like robot to search for the rotation angles for its motors, in a self-learning manner, in order to move toward a target point. It is added here that the rotation angle of each motor must be matched to the angles of the other three motors, in order to produce effective movement. The results of the whole experiment show that, through autonomous learning, the snake-type robot can learn to move to reach the target in a continuous manner and can use different motion combinations to reach farther distances. The preliminary results of this study demonstrate that different combinations of motions can be used to create additional combinations of paths, to reach different locations at equal distances from the starting point, or to use more combinations of motions to meet a different level of objectives. For the topic of continuous learning, this research can be further explored, in the future, by increasing the difficulty of the test environment, such as increasing the slope of the ground or changing the flatness of the ground (including regular or irregular). On the other hand, this research should continue to explore how the system can overcome this problem through self-learning when one or more of its motors malfunction.

Ethical Approval: This article does not contain any studies performed by the author that used human participants.

Funding: This study was in part funded by Taiwan Ministry of Science and Technology (Grant MOST 107-2221-E-224-039-MY3).

Conflicts of Interest: The author has no conflict of interest.

References

1. Higuchi, T.; Iwata, M.; Keymeulen, D.; Sakanashi, H.; Murakawa, M.; Kajitani, I.; Takahashi, E.; Toda, K.; Salami, N.; Kajihara, N. Real-world applications of analog and digital evolvable hardware. *IEEE Trans. Evol. Comput.* **1999**, *3*, 220–235. [\[CrossRef\]](#)
2. Vassilev, V.K.; Fogarty, T.C.; Miller, J.F. Smoothness, ruggedness and neutrality of fitness landscapes: From theory to application. In *Advances in Evolutionary Computing*; Ghosh, A., Tsutsui, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 3–44. [\[CrossRef\]](#)
3. Thompson, A.; Layzell, P. Analysis of unconventional evolved electronics. *Commun. ACM* **1999**, *42*, 71–79. [\[CrossRef\]](#)
4. Chen, J.-C.; Conrad, M. Evolutionary learning with a neuromolecular architecture: A biologically motivated approach to computational adaptability. *Soft Comput.* **1997**, *1*, 19–34. [\[CrossRef\]](#)
5. Chen, J.-C. Problem Solving with a Perpetual Evolutionary Learning Architecture. *Appl. Intell.* **1998**, *8*, 53–71. [\[CrossRef\]](#)
6. Chen, J.C. A study of the continuous optimization problem using a wood stick robot controlled by a biologically-motivated system. *J. Dyn. Syst. Meas. Control Trans. ASME* **2015**, *137*, 53–71. [\[CrossRef\]](#)
7. Conrad, M. *Adaptability*; Plenum Press: New York, NY, USA, 1983.
8. Nansai, S.; Yamato, T.; Iwase, M.; Itoh, H. Locomotion Control of Snake-Like Robot with Rotational Elastic Actuators Utilizing Observer. *Appl. Sci.* **2019**, *9*, 4012. [\[CrossRef\]](#)
9. Hirose, S. *Biologically Inspired Robots: Snake-Like Locomotors and Manipulators*; Oxford University Press: Oxford, UK, 1993.
10. Sato, M.; Fukaya, M.; Iwasaki, T. Serpentine locomotion with robotic snakes. *IEEE Control Syst.* **2002**, *22*, 64–81. [\[CrossRef\]](#)
11. Wu, X.; Ma, S. CPG-based control of serpentine locomotion of a snake-like robot. *Mechatronics* **2010**, *20*, 326–334. [\[CrossRef\]](#)

12. Chen, L.; Ma, S.; Wang, Y.; Li, B.; Duan, D. Design and modelling of a snake robot in traveling wave locomotion. *Mech. Mach. Theory* **2007**, *42*, 1632–1642. [\[CrossRef\]](#)
13. Xu, L.; Mei, T.; Wei, X.; Cao, K.; Luo, M. A Bio-Inspired Biped Water Running Robot Incorporating the Watt-I Planar Linkage Mechanism. *J. Bionic Eng.* **2013**, *10*, 415–422. [\[CrossRef\]](#)
14. Liljeback, P.; Pettersen, K.Y.; Stavdahl, O.; Grasdahl, J.; Grasdahl, J.T. A review on modelling, implementation, and control of snake robots. *Robot. Auton. Syst.* **2012**, *60*, 29–40. [\[CrossRef\]](#)
15. Shan, Y.; Koren, Y. Design and motion planning of a mechanical snake. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 1091–1100. [\[CrossRef\]](#)
16. Chen, J.-C.; Conrad, M. A multilevel neuromolecular architecture that uses the extradimensional bypass principle to facilitate evolutionary learning. *Phys. D Nonlinear Phenom.* **1994**, *75*, 417–437. [\[CrossRef\]](#)
17. Chen, J.-C.; Conrad, M. Learning synergy in a multilevel neuronal architecture. *BioSystems* **1994**, *32*, 111–142. [\[CrossRef\]](#)
18. Liberman, E.; Minina, S.; Mjakotina, O.; Shklovsky-Kordy, N.; Conrad, M. Neuron generator potentials evoked by intracellular injection of cyclic nucleotides and mechanical distension. *Brain Res.* **1985**, *338*, 33–44. [\[CrossRef\]](#)
19. Matsumoto, G.; Tsukita, S.; Arai, T. Organization of the axonal cytoskeleton: Differentiation of the microtubule and actin filament arrays. In *Cell Movement: Kinesin, Dynein, and Microtubule Dynamics*; Warner, F.D., McIntosh, J.R., Eds.; Alan R. Liss: New York, NY, USA, 1989; pp. 335–356.
20. Hameroff, S.; Dayhoff, J.; Lahoz-Beltra, R.; Samsonovich, A.; Rasmussen, S. Models for molecular computation: Conformational automata in the cytoskeleton. *Computer* **1992**, *25*, 30–39. [\[CrossRef\]](#)

© 2020 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).