Adaptive Motor Primitive and Sequence Formation in a Hierarchical Recurrent Neural Network

Rainer W. Paine **Jun Tani**

RIKEN Brain Science Institute
Laboratory for Behavior and Dynamic Cognition
2-1 Hirosawa, Wako-shi, Saitama, 351-0198 JAPAN
rpaine@brain.riken.jp; tani@brain.riken.jp

Abstract

This study describes how complex goal-directed behavior can be obtained through adaptation in a hierarchically organized recurrent neural network using a genetic algorithm. Robot simulations showed that different types of dynamic structures self-organize in the lower and higher levels of the network for the purpose of achieving complex navigation tasks. Behavior primitives are switched in a top-down way through lower level parametric bifurcation structures. In the higher level, a topologically ordered mapping of initial cell activation states to motor-primitive sequences self-organizes by utilizing the initial sensitivity characteristics of nonlinear dynamical systems. The biological plausibility of the model's essential principles is discussed.

1. Introduction

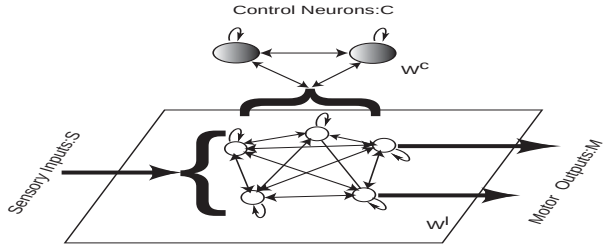
It is widely believed that behavior systems develop certain hierarchical or level structures for achieving goal-directed complex behaviors, and that these structures should self-organize through interactions with the environment. It is reasonable to assume that an abstract event sequence is represented in a higher level, while its detailed motor program is generated in a lower level. Arbib (1981) proposed the idea of movement primitives (also referred to as perceptual-motor primitives, motor schemas, or motor programs), which are a compact representation of action sequences for generalized movements that accomplish a goal. Evidence of such primitives in animals has been found (Giszter et al., 1993; Mussa-Ivaldi et al., 1994), and human studies also indicate their role in complex movement generation (Thoroughman & Shadmehr, 2000). Once such motor primitives develop early in the life of an organism, diverse behaviors can emerge by learning to combine them in a multitude of complex sequences.

A movement primitive can be formalized as a "control policy", encoded using a few parameters in the form of a parameterized motor controller, for achieving a particular task (Schaal, 1999). The motor primitives are routine motor programs that repeatedly appear in the sequences of the motor patterns. Once such motor primitives develop early in the life of an organism, diverse behaviors can emerge by learning to combine them in a multitude of complex sequences.

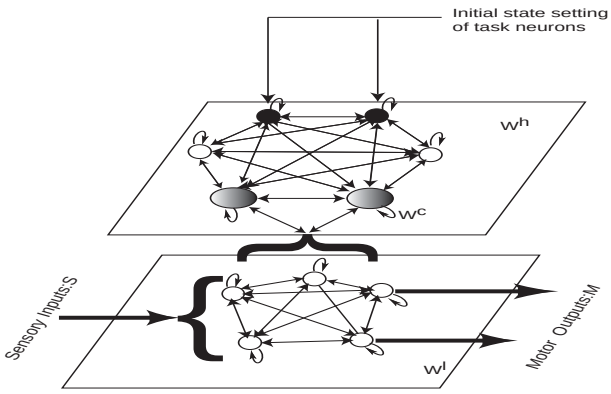
Mataric (2002) showed examples of behavior generation through primitive combination in the context of imitation learning using a virtual humanoid robot. In her computational architecture, a cluster of motor primitives was organized in a lower level. Then, a Hidden Markov model in the higher level learned how the primitives were combined into sequences in order to recognize and regenerate the human instructor's behavior patterns. In related work, Amit and Mataric (2002) used Self-Organizing Maps (SOMs) to hierarchically control postural and oscillatory movement primitives in a simulated robotic arm.

Hochreiter and Schmidhuber (1997) proposed a so-called long-term and short-term memory connectionist model for hierarchical sequence learning which focuses on the problem of sequence segmentation. The essential idea in this study was to learn long and complex sequences by dividing them into chunks of sub-sequences.

Tani and Nolfi (1999) extended the idea of the Mixture of Experts (Jacobs et al., 1991; Jordan & Jacobs, 1994) by introducing level structures. In their experiments with a simulated mobile robot, the robot learned to perceive sensory-motor flow as hierarchically articulated. In contrast to this local representation scheme utilizing expert modules for representing primitives, Tani (2003) proposed a distributed representation scheme where multiple primitives can be embedded in a single recurrent neural network (RNN) in terms of a "forward model" (Kawato et al., 1987). Each primitive can be accessed by a control parameter called the "parametric bias" (PB). The higher level RNN combines the lower level primitives in sequences by learning and sending the corresponding PB sequences to the lower level RNN.



(a)



(b)

Figure 1: Conceptual diagram of network architecture.

However, these neural network schemes seem to have some potential drawbacks. One major problem is that the network cannot be adapted dynamically through trial and error. This is due to the fact that the above mentioned neural network approaches depend heavily on a supervised learning scheme using teaching signals. In contrast, reinforcement learning schemes using "macro-actions", or sequences of primitives, can learn to effectively combine primitives to solve sequential tasks simply through environment-mediated rewards. However, inappropriate use of macro-actions may retard learning. Appropriate macro-actions must generally be tuned manually for specific environments, although work is being done to automate the process (McGovern et al, 1997, 2001).

Another problem is the setting of time constant parameters for the network dynamics at each level. The time constant in the lower level has to be determined based on the shifting frequency of the lower primitives. The higher level time constant must further be determined based on those of the lower level primitives as well as task features. Such parameter setting is usually done manually by experimenters, which requires certain a priori knowledge about the task environment. In order to overcome these problems, this paper introduces a novel scheme using an evolutionary adaptation mechanism through a genetic algorithm (GA). The evolutionary robotics community has shown that the GA scheme allows for the self-organization of dynamic adaptive behaviors in sensory-motor systems (Tucci et al., 2002; Nolfi & Floreano, 2000).

In this paper, we study the dynamic adaptation process of multiple levels of continuous time recurrent neural networks (CTRNNs) applied to a navigation task using a simulated robot. Through the experiments, we will demonstrate that a hierarchically organized network can perform well in adapting to complex tasks through combination with a GA. We will focus on how motor primitives are self-organized in the lower level, and how they are manipulated in the higher level.

2. Methods

2.1 General Model

The neural network model utilized in the current paper consists of two levels of fully connected CTRNN (Yamauchi & Beer, 1994; Blynel & Floreano, 2002). The lower level network, as shown in Figure 1, receives sensory inputs and generates motor commands as outputs. This network is supposed to encode multiple sensory-motor primitives, such as moving straight down a corridor, and turning left or right at intersections or to avoid obstacles in the navigation task adopted in this study.

A set of external neural units, called the "control neurons", are bidirectionally connected to all neurons in the lower level network. The control neurons influence lower level network functions and favor the generation of particular motor primitives. Through evolution of both the lower level internal synaptic weights (W_l) and the interface weights (W_c) between the control neurons and the lower level neurons, a mapping between the control neurons' activities and the sensory-motor primitives stored in the lower level network is self-organized. Modulation of the control neurons' activities causes shifts between generating one primitive and another. The scheme is analogous to the idea of the parametric bias in Tani (2003) and the command neuron concept (Aharonov-Barki et al., 1999; Edwards et al., 1999; Teyke et al., 1990). How might more complex tasks, such as navigation in an environment, be generated? Such tasks require generating sequences of motor primitives. We propose that a higher level network may modulate the activities of the control neurons through time to generate sequences of lower level movement primitives (Figure 1b). The higher level network evolves to encode abstract behavior sequences utilizing the control neurons.
It is assumed that the desired sequences will be generated if adequate nonlinear dynamics can be self-organized in the higher level network. As will be described in detail later, the robot becomes able to navigate to multiple goal

positions when starting from the same initial position in the maze environment. Therefore, the higher level network has to encode multiple sequence patterns, which have to be retrieved for the specified goal.

We utilize the initial sensitivity characteristics of nonlinear dynamic systems in order to initiate different sequences. When the robot is placed at the initial position in the environment, the internal values of all the higher level neurons are set to 0.0, except for two neurons called the task neurons (Figure 1b). The initial values of the task neurons are set corresponding to the specified goal positions. These goal-specific initial task neuron activities were evolved through the same genetic algorithm that yielded the network's synaptic weights.

We assume that an appropriate sequence pattern that enables the robot to navigate to the kth goal can be generated by setting adequate initial activities (γ_k) for the task neurons when the higher level internal connective weights (W_h) are adequately generated through evolution. This idea of utilizing the initial sensitivity of the network dynamics is similar to the studies of Nishimoto & Tani (2003) and Blynel (2003). Further, Tanji & Shima (1994) showed in the monkey that the pre-Supplementary Motor Area neuronal activities during the motor preparation period might encode abstract behavior sequences through the initial state values of the network dynamics. Nishimoto & Tani (2003) showed that a Recurrent Neural Network (RNN) learns to generate various action sequences by setting different initial context unit activities using the back-propagation learning method. Blynel (2003) found that goal positions can be "remembered" through the activations of hidden neurons in a reinforcement learning task when Genetic Algorithm evolution is applied to a single level CTRNN.

Evolution of the two-level network used here goes through two phases. In the first phase (Experiment 1), the network shown in Figure 1a evolves to perform collision-free left and right turns in a T maze environment. Both the lower level internal synaptic weights (W_l) and the control neuron interface synaptic weights (W_c) are evolved while adequate activation values of the control neurons are determined for the left and right turns. At this stage, there is no higher level network with task neurons. Instead, the neuronal activation bias (θ, Equation 2) of the two control neurons is free to evolve differently for the left and right tasks. The synaptic weights are identical for the two turn directions.

In the second phase (Experiment 2), the evolved lower level network is extended by adding and evolving the higher level network (Figure 1b). The task of the robot is to find ways to reach multiple goals from the same starting position. In this phase, only the higher level internal connective weights (W_h) and goal-specific task neuron initial activities (γ_0) are evolved. The bottom level internal weights and control neuron interface synaptic weights (W_c) are kept constant from the first phase's T maze task. Thus, collision free left or right turning does not need to be re-evolved. Instead, the second learning phase focuses solely on goal finding through the appropriate turn sequence generation.

2.2 Continuous-Time Recurrent Neural Networks

All neurons in the simulations presented here used the following equations and parameters for a CTRNN, based on those of Blynel & Floreano (2002). In Equation 1, γ_i is the internal activation state (cell potential) of the i^{th} neuron. τ is the time constant of the neuron. It affects the rate of neuronal activation in response to the k^{th} external sensory neuronal activation, I_k, and signals from the j^{th} presynaptic neuron with activity A_j. The signal from the presynaptic neuron is weighted by weights w_{ij}, and the sensory input is weighted by w_{ik}. N is the number of neurons in the network, and S is the number of sensory receptors which send input signals to the network.

$$\frac{d\gamma_i}{dt} = \frac{1}{\tau_i}(-\gamma_i + \sum_{j=1}^{N} w_{ij}A_j + \sum_{k=1}^{S} w_{ik}I_k) \quad (1)$$

The presynaptic neuronal activity (A_j) is defined in Equations 2 and 3. θ is a bias term and σ is the standard logistic function, defined in Equation 3.

$$A_j = \sigma(\gamma_j - \theta_j) \quad (2)$$

$$\sigma(x) = 1/(1+e^{-x}) \quad (3)$$

Numerical integration was carried out using the Forward Euler method. The update rule of the neuronal activation state γ_i for each integration time step is given by Equation 4. n is the iteration step number and Δt is the time step interval, defined as 0.2.

$$\gamma_i(n+1) = \gamma_i(n) + \frac{\Delta t}{\tau_i}\left(-\gamma_i(n) + \sum_{j=1}^{N} w_{ij}A_j(n) + \sum_{k=1}^{S} w_{ik}I_k\right) \quad (4)$$

Except for the previously mentioned task neurons, whose initial activation is dependent on the movement goal, the neuronal activation, γ_i, is initialized to 0 at the start of integration, $\gamma_i(0) = 0$.

2.3 Genetic Encoding

The following parameters are genetically encoded within the following ranges. τ is the time constant (Equation 1). θ is the activation bias (Equation 2). $\boldsymbol{w}$ is the synaptic weight,

and $\gamma_{task}(0)$ is the initial activation of the task-dependent neurons of the higher level network.

$$\tau \in [1,70]; \theta \in [-1,1]; w \in [-5,5]; \gamma_{task}(0) \in [-10,10]$$

Each parameter is encoded using 5 bits and the following encoding rule to generate analog parameter values. Thus, each parameter value is generated by a linear scaling of the analog value within a given range to the range of 0 to 31 in binary code. The encoded parameter value (P_e) from the 5 bit string is given in Equation 5. The analog parameter value (P_a) is given by Equation 6 for an analog parameter value in the range $P_a \in [\min, \max]$.

$$P_e = \sum_{i=1}^{5} bit_i \cdot 2^{5-i} \quad (5)$$

$$P_a = \min + \frac{P_e \cdot (\max - \min)}{2^5 - 1} \quad (6)$$

In the T maze experiment, the network consists of eight inputs, five bottom level units, and two control neurons, for a total of 515 bits encoding τ, θ, and ***w***. In the Eight-Goal-Maze experiment, the network is as above, except that it also includes 4 higher level neurons. The task neurons' initial activity ($\gamma_{task}(0)$) is also encoded, yielding a genome length of 715 bits.

2.4 Genetic Algorithm

A standard genetic algorithm (GA) with mutation but without crossover was employed to evolve the weights and parameters (τ, θ, *w*, $\gamma_{task}(0)$) of the network (Mitchell, 1998; Goldberg, 2002). The mutation rate per bit was set at 2% for all simulations reported here. (Higher mutation rates of up to 10% were tried, but the fitness of the resulting populations became progressively more unstable without improvements in the best robot performance). The population consisted of 80 robots. The twenty robots with the best fitness reproduced each generation. Each of the reproducing robots made 4 copies of itself with mutation. Of the best robot's offspring, one was an exact copy of the parent without mutation. Simulations were generally run for 50 to 200 generations, and the performance of the best robots was analyzed.

2.5 Network Architecture

In Experiment 1, the network depicted in Figure 1a consists of 8 sensory inputs, scaled to a range of 0 to 1, which are sent to the bottom level of 5 neurons. The bottom network receives from and sends signals to the two control neurons. In Experiment 2, the control neurons are further connected to the higher level of 4 neurons (Figure 1b). Two of the higher level neurons are the "task neurons", whose initial activities determine the particular turn sequence which will be generated to reach a goal.

The outputs of the first two bottom level neurons are taken as motor command signals to the simulated robot wheels. The actual neuronal outputs are analog signals, but the robot controller can use only integer speed commands. The analog signals are therefore rounded to the nearest integer. The simulated wheel speed is in the range of 0 to 4, corresponding to speeds of 0 to 0.32cm per second. Note that the wheels are allowed to turn only in a forward direction in these experiments for simplicity.

2.6 Experimental Setup

All experiments reported here were executed using a simulated Khepera II robot in the Webots 3 robot simulator. Simulations were run on a Plathome computer with a 2.0 GHz processor. Simulations ran at about 40 times real Khepera speed. Data were analyzed using Matlab Release 13.

Inputs to the neural network consisted of the signals from seven infra-red proximity sensors (2 left, 2 front, 2 right, and 1 rear) and one downward facing ground sensor (illustrated by rays protruding from robot in Figure 2). The input was modified by randomly adding or subtracting 5% of its value as noise. All sensor inputs to the network were scaled to a range of 0 to 1. In all experiments, the robot was repositioned to the starting point and the trial was ended if the robot either collided with a wall or reached a goal area. Goal areas were defined by differently colored floors. The floor sensor was used by the controller to detect when the robot found a goal, triggering the appropriate fitness reward, ending of the trial, and repositioning of the robot.

3. Experiments

3.1 Experiment 1: T-Maze Task

Experiment 1 is designed to evolve a bottom level network which contains movement primitives of left and right turning behavior at intersections as well as collision-free straight movement in corridors. The same lower level and control neuron weights are used for both right and left turns. The only difference between the left and right turn controllers is in the bias values (θ) of the two control neurons. Intuitively, this might correspond to different sets of cortical "control" neurons becoming associated with each of the lower level movement primitives. Parallel connections from the bottom level to the control neurons might develop, yielding the same weights to both sets of control neurons. Intrinsic differences in the control neurons' responses (as through the different θ bias values used here) to the lower level signals would determine with

which motor primitives each set of control neurons became associated.

The evolutionary runs consisted of up to 200 generations with 2 epochs and 3 trials per robot of the 80 robot population. Each trial was run for 500 time steps, starting at the same position at the bottom of the T maze. Different bias values (θ) evolved in the control nodes for the left and right turning tasks in epochs 1 and 2, respectively. All other parameters were identical in the left and right turning tasks. In epoch 1, fitness was awarded to robots that turned to the left at the intersection based on the following fitness rule. In epoch 2, fitness was awarded to robots that turned to the right. Each robot ran 3 trials per epoch.

Experiment 1 uses a two-component fitness rule (Equation 7). The first component (F_{OA}) consists of a reward for straight, fast movements with obstacle avoidance. The fitness rule of Floreano & Mondada (1994) was adopted for this purpose and is shown in Equation 8. V is the wheel speed scaled to a range of 0 to 1. $\overline{V}$ is the average speed of the two wheels. ΔV is the absolute value of the difference in speeds between the two wheels, and is used to reward straight movements. S_{max} is the maximum robot sensor value, scaled to a range of 0 to 1, and is used to reward obstacle avoidance.

$$F = F_{OA} + F_{goal} \tag{7}$$

$$F_{OA} = \overline{V} \cdot (1 - \sqrt{\Delta V}) \cdot (1 - S_{\max}) \tag{8}$$

The second component of the fitness rule, F_{goal}, rewards the robot for finding a goal. The goal is located to the left of the intersection for epoch 1, and to the right for epoch 2. The robot is linearly rewarded, based on its position, for approaching and reaching the goal. Greater reward per time step is received linearly as the robot approaches the goal, starting at the middle of the top of the T maze.

At the start of each trial, the robot was placed at the same starting position at the bottom of the T maze. Three different starting orientations (facing $135^{o,}$ $90^{o,}$ and 45^{o}; that is, left, straight, and right, respectively) were employed, one for each of the three trials per epoch. Further, motor noise was added to the wheel speed commands sent to the robot. The integer speed command was increased or decreased by one with a probability of 20% on each simulation time step. The varying starting orientations and motor noise were used to ensure that the robot would experience wall collisions early during evolution, so that controllers with obstacle avoidance would be more likely to evolve. Both the bottom level and control neurons were free to evolve in this experiment.

3.2 Experiment 2: Eight-Goal Task

The Eight-Goal maze depicted in Figure 2 is a combination of T maze-like environments with eight different goals at the ends of each T maze component. Thus, combinations of the same turn primitives evolved in experiment 1 should allow the robot to reach the different goals. In experiment 1, different sets of control neurons with differing internal dynamics (due to differing θ bias values) became associated with particular motor primitives, as will be described later. In experiment 2, it is shown how the activity of a *single* set of control neurons can be modulated over time to generate a *sequence* of motor primitives. Further, it is shown how varying only the initial activation of the task neurons in the higher level network can lead to the generation of different network activation time courses by which multiple primitive sequences are generated. Thus, the routes to multiple goals can effectively be stored in a single network, with a single set of synaptic weights and corresponding initial activation values of the task neurons.

The bottom level genome, including the weights for the connections to the control neurons, from experiment 1 was used in this experiment and held constant. Only a single set of synaptic weights and parameters in the higher level network, and multiple sets of the initial task neuron activities, were free to evolve. The experiment consisted of up to 200 generations, with 12 epochs per generation and 2 trials per epoch. Each of the 12 epochs evaluated a different set of higher level task neuron initial activities, using the fitness rule described below. Further, each task neuron set was run for two trials, in order to evaluate the stability of the robot's goal-finding ability. Robots which found multiple goals repeatedly were rewarded more than those which found them only intermittently. Further, robots which found some stable goals tended to be rewarded more

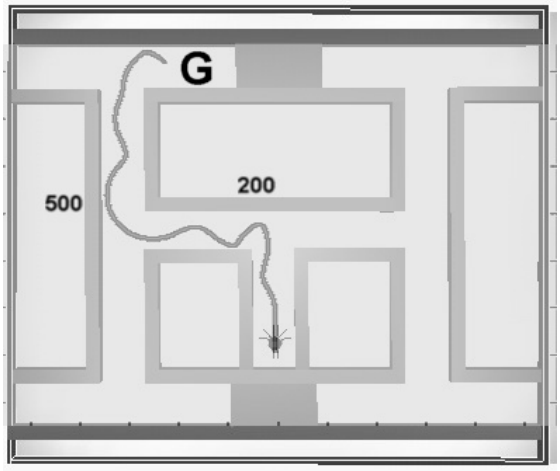


Figure 2: Eight-Goal Maze environment, showing trajectory for a Left, Right, Right sequence. G= Goal location. Turns occur at steps 200 and 500 in the neuronal activity traces for this sequence in Figure 4.

than robots that found more goals haphazardly. As in experiment 1, a trial was ended when the robot either found a goal or collided with a wall. The robot was then repositioned to the same starting point. Since the bottom level genome from experiment 1 was used here, obstacle avoidance and turn primitives were present in the first generation. Therefore, only one starting orientation was

used (90°, i.e., straight up) and no motor noise was added to the speed commands sent to the robot wheels (but sensor noise was still present).

In experiment 2, the fitness rule (F) consists solely of a reward for finding goals consistently (Equation 9).

$$F = \sum_{g=1}^{Ngoals} \max_{i=1:Ntasks} \left(\sum_{trial=1}^{2} R_{gi} \right) \quad (9)$$

Here, a fixed reward (R) per new goal (g) found is given to the robot. In experiment 2, Ngoals = 8, and Ntasks = 12.

Each robot has 12 sets of task neuron initial activities (i) which are evaluated. Each set has two trials in which to find a goal. A robot which finds a goal on both trials receives twice the reward of finding the goal on only one trial. If a different set of task neuron initial activities leads to the same goal, then the reward is the maximum of the reward given to the two different task neuron sets. Thus, a robot with multiple task neuron sets that find a goal on only one of the two trials will receive less reward than a robot with one task neuron set that finds the goal on both trials.

4. Results/Analysis

4.1 Experiment 1: T-Maze Task

Collision avoidance and left and right turning behavior emerged within 63 generations. Left turns were generated for one set of control neuron bias values, and right turns were generated for another set. Although both left and right turns could be generated, the robot exhibited oscillatory movements after collision avoidance. That is, it turned away from one wall too much and headed towards the opposite wall instead of straightening its path through the lower part of the T maze. (As mentioned previously, the robot started from three different orientations, leftward, straight up, and rightward, requiring it to avoid wall collisions early during each trial.) The controller was therefore allowed to evolve further. By generation 189, fewer fluctuations occurred after collision avoidance and the lower level genome was used for experiment 2.

Control neuron 0 appears to be critical in determining whether a left or right turn is generated. Turns occur at approximately time step 200, at which time the control neuron activity is 0.2 for the left turn and 0.6 for the right turn. The relation between turn behavior and the activities of control neurons 0 and 1 is further quantified in the phase plot of Figure 3. 441 trials with different sets of the two control neurons' activities, held constant for each trial, (A=[0:1], step size = 0.05) were run and the resulting turn directions recorded. Note that this figure also applies to the activity of the control neurons evolved in experiment 2, since the same bottom level genome, including the synaptic weights between the bottom level and control neurons, was used in both experiments 1 and 2. The turn phase plot of Figure 3 shows a clear bifurcation, or appearance of new movement behavior with the control neuron activity change, with two distinct regions of stability for left (black) and right (white) turns, for the corresponding combinations of control neuron 0 and neuron 1 activities. The gray squares indicate unstable regions in which wall collisions occur. In both experiments 1 and 2, the evolved control neuron weights tend to suppress the activity of control neuron 1. The phase plot shows that the smallest region of instability between left and right turns occurs for such small control neuron 1 activities. Further, the weights from control neuron 0 to the two motor output nodes (w_{jk}= 5.0, -3.4) of the bottom level have a greater magnitude than the weights

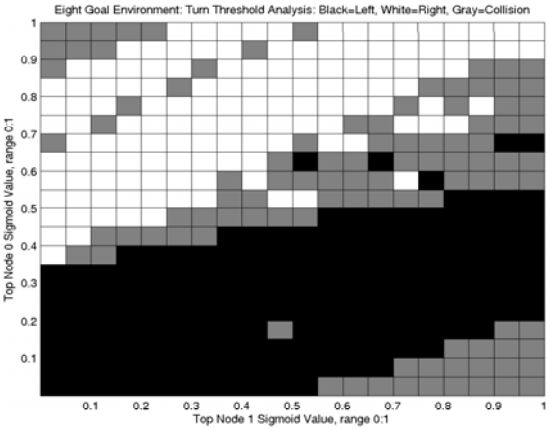


Figure 3: Phase analysis of turn direction as a function of control neuron activation. X and Y axes: neuronal activities of control neurons 1 and 0, respectively. Black = Left turn, White = Right turn, Grey = Collision with wall.

from control neuron 1 (w_{jk}=-1.1, 2.0). Thus, control neuron 0 has the dominant effect on the turn direction of the lower level's output neurons. It therefore appears that the lower level, which receives and processes all sensory inputs, has a dominant role in collision avoidance. Collision avoidance competes with the control neurons' "turn" signals. In the unstable regions of the phase plot (grey in Figure 3), the control neurons' "turn" commands inappropriately override the lower level's collision avoidance, triggering collisions when the robot turns toward and collides with a wall.

4.2 Experiment 2: Eight-Goal Task

The best robot became able to reach up to 7 different goals stably within 24 generations by evolving the higher level network and control neurons. The turn sequence for each goal was determined by a particular set of initial task neuron activities ($\gamma_{task}(0)$). Since 12 different sets of $\gamma_{task}(0)$ were evaluated per robot, multiple sets would sometimes lead to the same goal. Each $\gamma_{task}(0)$ set was evaluated for the stability of its goal. Some values of $\gamma_{task}(0)$ led to repeatable goal-finding performance, while others were

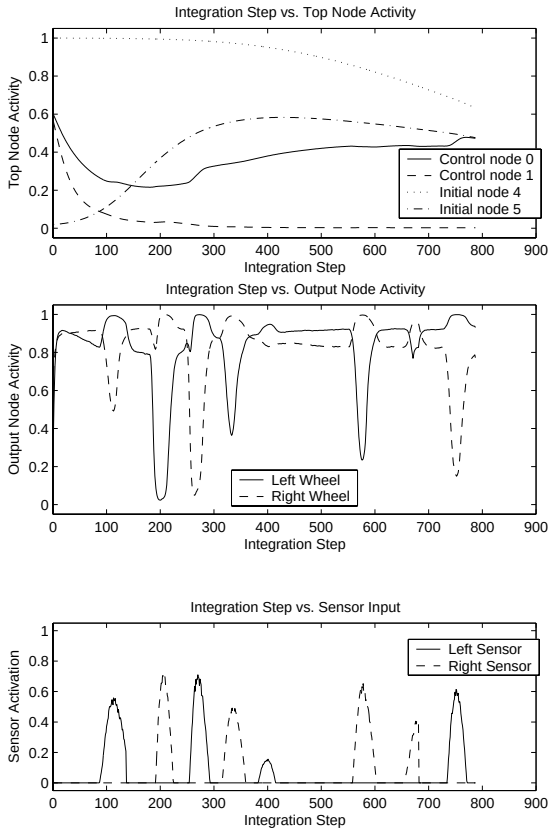


Figure 4: Left-Right-Right turn sequence. Top: Neuronal activity of control and task (initial node) neurons; Middle: Lower level motor output node activity; Bottom: Left and right sensor activities.

unstable, leading to different goals on different trials, or even to wall collision. Although evolution led to controllers which could find all 8 goals, the best controller could only find 7 goals stably. The definition of stability used here is reaching a particular goal, with the corresponding evolved $\gamma_{task}(0)$ values, on at least 70% of trials.

Different $\gamma_{task}(0)$ activities led to differently fluctuating patterns in the control neuron activities (Figure 4). As in the results of experiment 1 reported above, control neuron 0 had the greatest influence on turn direction, and exhibited the greatest fluctuations during various turn sequences. Left turns were generated when its activity was below a threshold of approximately 0.35, and right turns were generated when the activity was above the threshold. The activity of control neuron 1 tended to be suppressed, leading to a smaller region of instability at the transition from left to right turns in the phase plot of Figure 3. Figures 2 and 4 show the trajectory and neuronal activities, respectively, for a left-right-right turn sequence. The results shown here are for a controller which learned to reach six goals stably in 25 generations. They show large amplitude fluctuations of the control neurons due to a small evolved value of the time constant τ in Equation 1.

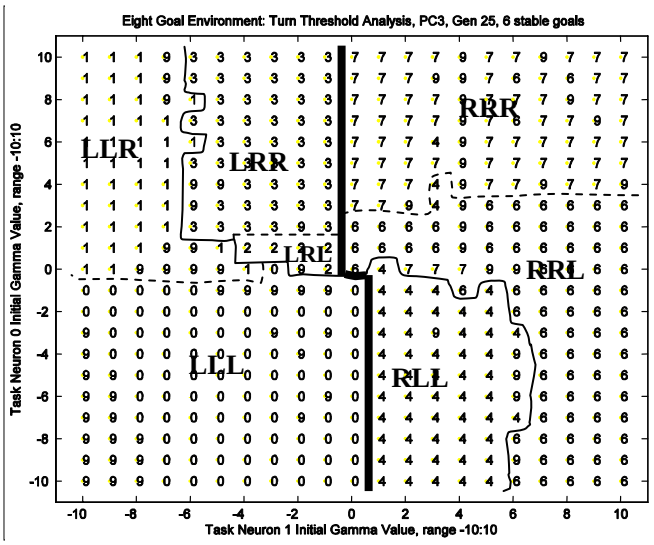


Figure 5: (a) Phase analysis of three-turn sequence generation as a function of task neuron initial activity, $\gamma_{task}(0)$. X and Y axes: Initial activities of task neurons 1 and 0, respectively. Plotted numbers correspond to sequences (L=Left, R=Right) as in Table 1.

The amplitude of the control neurons' fluctuations is also significant because larger amplitude fluctuations may render the controller more robust to noise. Although 5% sensor noise was used throughout these experiments, motor noise (increasing or decreasing the wheel speed command by 1 unit with 20% probability) was used only in experiment 1 in order to facilitate the development of obstacle avoidance. Motor noise was also tested during separate evolutionary runs in experiment 2, and found to decrease the number of stable goals found from a maximum of 7 (without motor noise) to 5 (with motor noise). Note that the total number of goals found was eight, with or without motor noise. Thus, additional noise added to the wheel motor commands increases the instability of the turn sequences.

Goal instability was most often seen in the final turn direction of the three-turn sequences learned. This final turn instability can be appreciated by noting that the control neuron activity tends to hover around the turn threshold near the end of the trial. As seen in the phase plot of Figure 3, this region is unstable.

Figure 5 shows an analysis of the movement sequences generated for the range of task neuron initial activities, $\gamma_{task}(0) \in [-10,10]$, in the evolved controller of experiment 2. 441 sets of initial task neuron activities were tested and the resulting turn sequences recorded. The numbers in the figure correspond to movement sequences as labeled in the figure, e.g., LRL for left, right, and left turns. The sequence patterns self-organize into well-defined, topologically ordered clusters in the $\gamma_{task}(0)$ space. First, the $\gamma_{task}(0)$ space is grossly clustered based on the first turn direction, left or right, of the movement sequence, as shown by a thick solid line in Figure 5. Each of these two clusters is then further divided into sub-clusters, depending

SEQUENCE	BINARY CODE	INTEGER VALUE
LLL	000	0
LLR	001	1
LRL	010	2
LRR	011	3
RLL	100	4
RLR	101	5
RRL	110	6
RRR	111	7
Collision	---	9

Table 1: Movement Sequence representation used in Figure 5. L=Left Turn; R=Right Turn; 0=Left Turn; 1=Right Turn.

on the second turn direction of the movement sequence, as shown by a solid line. These sub-clusters are still further divided into smaller clusters, depending on the third turn as shown by the dashed lines.

Thus, turn sequences are hierarchically ordered into progressively smaller regions of the initial task neuron activity space, as additional turns are added. In other words, as the complexity of the movement sequence increases, so too does the initial sensitivity to the task neuron activities.

This mapping of initial task neuron activity to particular sequences is an emergent property of the evolved controller. Different evolutionary runs yield different cluster patterns, but the general trend of distinct, topologically ordered sequence regions remains. This self-organized, topologically ordered mapping of sequences, with increasing initial sensitivity to task node activities as movement sequence complexity increases, is notable in such a small network, and is reminiscent of the fractal distribution of sequences mapped in the parameter space of Nishimoto & Tani (2003). Indeed, it would be interesting to see if fractal structure could be found in controllers branching out to larger numbers of goals.

5. Discussion/Conclusion

The work presented here describes a novel hierarchical model of behavioral sequence memory and generation. It recalls in general terms the hierarchical organization of movements in the primate spinal cord, brainstem, and cortical regions. Different types of dynamic structures self-organize in the lower and higher levels of the network. A parametric bifurcation in the control neurons' interaction with the lower level allows top-down behavioral switching of the primitives embedded in the lower level. Utilizing the initial sensitivity characteristics of nonlinear dynamic systems (Fan et al., 1996), a topologically ordered mapping of initial task neuron activity to particular behavior sequences self-organizes throughout the development of the network. The interplay of task-specific top-down and bottom-up processes allows the execution of complex navigation tasks.

One unique feature of the current model is the hierarchical organization of the network and its training. The bottom level network represents movement primitives, such as collision avoidance and turning at intersections. Since it must directly deal with quickly changing environmental stimuli, its time constants have become small through adaptation so that the neuronal activity of the output neurons ($\tau_0 = 1$, $\tau_1 = 1$ in Equation 1) can change rapidly to drive the robot's movement in real time. In contrast, the higher level represents sequences of the lower level primitives over longer time spans. Accordingly, the task neuron time constants have adapted to be large ($\tau_{task0} = 70$, $\tau_{task1} = 52$ in Equation 1) so that neuronal activity changes much more gradually and is less affected by short-term sensory changes.

The neurons of the higher level receive no direct sensory inputs, but are gradually influenced by them through the control neurons, which are fully connected to the input-receiving bottom level. This system is reminiscent of the organization of sequence generation in primates, as is elucidated by the studies of Tanji & Shima (1994) and Ninokura et al. (2003). In the former study, cellular activity in monkeys' supplementary motor area (SMA) was found to be selective for the sequential order of forthcoming movements, much as the task neurons' initial activities determine future movement order in the current model. In the latter study, distinct groups of cells in the lateral prefrontal cortices (LPFC) of monkeys were found to integrate the physical and temporal properties of sequentially reached objects, in a manner analogous to integration of higher level sequential information and lower level sensory input by the control neurons in the present model.

Although other models of sequence generation have been trained in a modular fashion because it was felt necessary to achieve the task (Yamauchi & Beer, 1994), the current work begins by explicitly evolving simple movement primitives, such as straight movements, collision avoidance, and turning at corners. The next level of the hierarchy subsequently develops to utilize the lower level primitives in complex movement sequences. One can envision further levels of complexity, with higher levels representing sequences of sequences for different sets of tasks, in a manner analogous to the "chunking" phenomenon observed in human memory of data sequences (Sakai et al., 2003). The beauty of this system is that the synaptic connections need not grow without bound as the number and complexity of sequences increases. As shown here, a *single* network can represent *multiple* complex movements through modulation of the activities of a small number of "task" neurons.

One may argue that hierarchical structure has been imposed on our system, and that no such structure is absolutely needed to complete the task. Indeed, Tucci et al. (2002) showed that the sequence generation task, which

Yamauchi & Beer (1994) felt required a modular approach, could indeed be generated with a single, non-modular, network. Further, Siegelmann & Sonntag (1995) showed that first and higher order recursive networks are computationally equivalent. However, the theoretical possibility that one giant first order network can carry out the same tasks as modular, hierarchically structured systems implies nothing about the relative ease with which either system can be generated artificially or biologically.

Although the initial sensitivity of the movement sequences generated to task neuron activations was an emergent feature of the system found by self-organization of network parameters through a genetic algorithm, the model architecture was predetermined, and the details of the network training influenced the specific functions that were assumed by different components of the architecture. Given that the current network architecture is loosely based upon the primate motor system's hierarchical design, one might expect it to perform better than a less biologically plausible giant first-order network that encompasses both simple movement primitives as well as their combination into complex sequences. This assumption will be tested in future work.

One may further question the need for hierarchical training of the current architecture. Given that humans show a clear progression of movement learning, from simple to complex movement patterns as they develop (Needlman, 2003), one might assume that there is an advantage, either in learning rate or the final skill level attained, to such an incremental, hierarchical learning organization.

Future work will test whether such complex movements can be learned "from scratch", without an externally imposed succession of increasingly difficult tasks. Further, it will be of great interest to see whether a similarly intricate dynamic structure self-organizes in the task-dependent neuronal activity of a network without such hierarchical movement learning.

Acknowledgments

The authors wish to thank Dario Floreano and Jesper Blynel, at EPFL in Switzerland, for their help and hospitality during the early stages of this work. Thanks are also due to Miki Sagara for her organizational skills and help in the preparation of this manuscript. This work would also not have been possible without the technical assistance of Yuuya Sugita and Ryunosuke Nishimoto.

References

Aharonov-Barki, R., Beker, T., Ruppin, E. (1999)
Spontaneous Evolution of Command Neurons, Place Cells and Memory Mechanisms in Autonomous Agents
In: *Advances in Artificial Life*, ECAL '99, vol. 1674, Lecture Notes in Artificial Intelligence, Springer Verlag.

Amit, R., Mataric, M. J. (2002)
Parametric Primitives for Motor Representation and Control
Int. Conf. on Robotics and Automation (ICRA), Washington DC, May 11-15, 2002.

Arbib, M. A. (1981)
Perceptual structures and distributed motor control
In: Brooks, V. B. (Ed.), *Handbook of Physiology*, Section 2: The Nervous System (Vol. II, Motor Control, Part 1), pp. 1449-1480, American Physiological Society.

Blynel, J. (2003)
Evolving Reinforcement Learning-Like Abilities for Robots
In: Tyrrell, A, Haddow, P.C., and Torresen, J (Eds.), *Evolvable Systems: From Biology to Hardware: 5th International Conference*, ICES 2003.

Blynel, J., Floreano, D. (2002)
Levels of dynamics and adaptive behavior in evolutionary neural controllers
In: Hallam, B., Floreano, D., Hallam, J., Hayes, G., Meyer, J.A. (Eds.), *From Animals to Animats 7: Proceedings of the Seventh International Conference on Simulation of Adaptive Behavior*. MIT Press, Bradford Books, Cambridge, MA.

Edwards, D. H., Heitler, W. J., Krasne, F. B. (1999)
Fifty years of a command neuron: the neurobiology of escape behavior in the crayfish
Trends in Neurosciences, 22(4), 153-161.

Fan, J., Yao, Q., Tong, H. (1996)
Estimation of Densities and Sensitivity Measures in Nonlinear Dynamical Systems
Biometrika, 83, 1, 189-206.

Floreano, D., Mondada, F. (1994)
Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot
In: Cliff, D., Husbands, P., Meyer, J., Wilson, S.W. (Eds.), *From Animals to Animats 3: Proceedings of the Third Conference on Simulation of Adaptive Behavior*. MIT Press, Bradford Books, Cambridge, MA.

Giszter, S. F., Mussa-Ivaldi, F. A., Bizzi, E. (1993)
Convergent force fields organized in the frog's spinal cord
Journal of Neuroscience, 13(2): 467-491.

Goldberg, D.E. (2002)
The Design of Innovation: Lessons from and for Competent Genetic Algorithms
Kluwer Academic Publishers, Boston, MA.

Hochreiter, S., Schmidhuber, J. (1997)
Long short-term memory
Neural Computation, 9(8), 1735-1780

Jacobs, R., Jordan, M., Nowlan, S., Hinton, G. (1991)

Adaptive mixtures of local experts
Neural Computation, 3(1), 79-87.

Jordan, M., Jacobs, R. (1994)
Hierarchical mixtures of experts and the EM algorithm
Neural Computation, 6(2), 181-214.

Kawato, M., Furukawa, K., Suzuki, R. (1987)
A hierarchical neural network model for the control and learning of voluntary movement
Biological Cybernetics, 57, 169-185.

Mataric, M. J. (2002)
Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics
In: Dautenhahn, K., Nehaniv, C. L. (Eds.), *Imitation in Animals and Artifacts*, pp. 391-422, MIT Press.

McGovern, A., Barto, A. G. (2001)
Accelerating Reinforcement Learning through the Discovery of Useful Subgoals
Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics, and Automation in Space: i-SAIRAS 2001

McGovern, A., Sutton, R. S., Fagg, A. H. (1997)
Roles of macro-actions in accelerating reinforcement learning
Proceedings of the 1997 Grace Hopper Celebration of Women in Computing, 13-18.

Mitchell, M. (1998)
An Introduction to Genetic Algorithms
MIT Press, Cambridge, MA.

Mussa-Ivaldi, F. A., Giszter, S. F., Bizzi, E. (1994)
Linear combination of primitives in vertebrate motor control
Proceedings of the National Academy of Sciences, USA, 91: 7535-7538.

Needlman, R.D. (2003)
Growth and Development
In: Behrman, R. E., Kliegman, R. M., Jenson, H. B. (Eds.)
Nelson Textbook of Pediatrics, 17th Ed., Part II, Chapter 10, p. 33. W. B. Saunders Publishing

Ninokura, Y., Mushiake, H., Tanji, J. (2003)
Integration of Temporal Order and Object Information in the Monkey Lateral Prefrontal Cortex
Journal of Neurophysiology, 10, 1152.

Nishimoto, R., Tani, J. (2003).
Learning to Generate Combinatorial Action Sequences Utilizing the Initial Sensitivity of Deterministic Dynamical Systems
Proc. of the 7th International Work-Conference on Artificial and Natural Neural Networks (IWANN'03). Springer, pp. 422-429.

Nolfi, S., Floreano, D. (2000)
Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines
MIT Press, Bradford Books, Cambridge, MA.

Sakai, K., Kitaguchi, K., Hikosaka, O. (2003)
Chunking during human visuomotor sequence learning
Experimental Brain Research, 152(2), 229-242.

Schaal, S. (1999)
Is imitation learning the route to humanoid robots?
Trends in Cognitive Sciences, 3, 233-242.

Siegelmann, H.T., Sontag, E.D.(1995)
On the computational power of neural nets
Journal of Computer and System Sciences, 50(1), 132-150.

Tani, J. (2003)
Learning to generate articulated behavior through the bottom-up and the top-down interaction processes
Neural Networks, 16, 1, 11-23.

Tani, J., Nolfi, S. (1999)
Learning to perceive the world as articulated: An approach for hierarchical learning in sensory-motor systems
Neural Networks, 12: 1131-1141

Tanji, J., Shima, K. (1994)
Role for supplementary motor area cells in planning several movements ahead
Nature, 371, 413-416.

Teyke, T., Weiss, K. R., Kupfermann, I. (1990)
An identified neuron (CPR) evokes neuronal responses reflecting food arousal in Aplysia
Science, 247, 85-87.

Thoroughman, K. A., Shadmehr, R. (2000)
Learning of action through combination of motor primitives
Nature, 407, 742-747

Tucci, E., Quinn, M., Harvey, I. (2002)
An Evolutionary Ecological Approach to the Study of Learning Behavior Using a Robot-Based Model
Adaptive Behavior, 10(3/4), 201-222.

Yamauchi, B., Beer, R.D. (1994)
Sequential behavior and learning in evolved dynamical neural networks
Adaptive Behavior, 2(3), 219-246.