

Motor Primitive and Sequence Self- Organization 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

Acknowledgments

The authors wish to thank Dario Floreano and Jesper Blynell, 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.

Send Reprint Requests to

Rainer W. Paine, M.D., Ph.D.

RIKEN Brain Science Institute

Laboratory for Behavior and Dynamic Cognition

2-1 Hirosawa, Wako-shi, Saitama, 351-0198 JAPAN

Phone: (81) (0)48-462-1111

Fax: (81) (0)48-467-7248

Email: rpaine@bdc.brain.riken.jp

Motor Primitive and Sequence Self-Organization in a Hierarchical Recurrent Neural Network

Abstract

This study describes how complex goal-directed behavior can be obtained through adaptation processes in a hierarchically organized recurrent neural network using a genetic algorithm (GA). Our experiments, using a simulated Khepera robot, 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. The parametric bifurcation structures that appear in the lower level explain the mechanism of how behavior primitives are switched in a top-down way. 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.

Keywords: Motor Primitive, Sequence, Self-Organization, Genetic Algorithm, Hierarchical, Recurrent Neural Network, Initial Sensitivity, Robot

List of symbols

A	Presynaptic neuron activity
F	Total fitness awarded to robot
F_{OA}	Fitness awarded for obstacle avoidance, and fast, straight movements.
F_{goal}	Fitness awarded for goal finding
γ	Internal activation state (cell potential)
$\gamma_{task}(0)$	Initial task neuron activation states
I	External sensory neuronal activation
P_e	Encoded parameter value from 5 bit string
P_a	Analog parameter value
R	Reward for finding a single new goal
S_{max}	Maximum robot sensor value, scaled to a range of 0 to 1
σ	Logistic function
τ	Time constant of a neuron
θ	Neuronal activation bias
V	Wheel speed scaled to a range of 0 to 1.
\bar{V}	Average speed of the two wheels.
ΔV	Absolute value of the difference in speeds between the two wheels
w	Synaptic weight

W_c	Control neuron interface synaptic weights
W_l	Bottom level internal weights
W_h	Higher level internal weights

1. INTRODUCTION

It is widely believed that humans 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. Fikes et al. (1972) described "generalized robot plans" as a way of solving future problems by incorporating past experience. 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).

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 were 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, 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.

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", analogous to 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). The GA scheme used here could be considered a type of automated trial-and-error controller optimization tool, although it must generally be run off-line.

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. Such parameter setting is usually done manually by experimenters, which requires certain a priori knowledge about the task environments. 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 RNNs 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. The emphasis of the work is on the self-organization of control structures at each level. The parametric bifurcation mechanism that emerges in the lower level explains how behavior primitives are switched. On the other hand, 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. Self-organization is a crucial principle in the organization of biological systems. There is much evidence that the brain uses this principle as well, which is the hypothesis in this article. Animal data (Tanji & Shima, 1994), discussed later, are suggestive that initial sensitivity to the activities of key neurons in SMA plays a role in sequence generation similar to that of the task neurons in the current model. However, the details of our model's implementation remain in the neural networks/machine learning area of study.

Our goal is therefore both to "explain biology", at least in terms of key general concepts, and to develop a

"better" robotic controller. Several essential questions will be discussed in this paper.

One interesting question is how motor primitives can be utilized not as fixed functions, but as ones that can flexibly develop and adapt to changes in many different environments. Another focus will be on the dynamic interactions between the levels. Sun et al. (2001) modeled bottom-up skill learning in a two level hybrid connectionist/reinforcement/symbolic learning system in which procedural knowledge was acquired in the lower level before or simultaneously with the acquisition of declarative knowledge in the top level. If the higher level is to encode abstract event sequences and the lower level is to encode detailed sensory-motor patterns, how can these top-down and bottom-up processes be reconciled if certain conflicts take place between them? Further, how can the higher level self-organize to make the best use of the lower-level primitives for different tasks? The proposed scheme is introduced in the next section.

2. Methods

2.1. General Model

The neural network model utilized in the current paper consists of two levels of fully connected continuous time recurrent neural networks (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_1) 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-Barkai 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 activity values of the task neurons determine the subsequent turn sequence, and the goal which is found. 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 k^{th} 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 analogous to the studies shown by 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 an 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 GA 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_1) and the control neuron interface synaptic weights (W_c) are evolved while adequate activation values of the control neurons are determined for the left-turn and the right-turn. 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 continuous-time recurrent neural network (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.

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

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.

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

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). 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 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. Average population performance was lower (e.g., 3 goals found) but the sheer performance of the GA in searching the network parameter space is not the main point of this article. Rather, it is the performance

and capabilities of the network, exemplified in the "best" controller, which is of interest.

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.

3. Experiments

All experiments reported here were executed using a simulated Khepera II robot in the Webots 3 robot simulator available from www.cyberbotics.com. 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.

3.1. 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 simulated robot environment is depicted in Figure 2. 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. \bar{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.

$$F = F_{OA} + F_{goal} \quad (7)$$

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

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, as shown at the top of Figure 2. 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° , 90° , and 45° ; 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 8 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 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).

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

Here, a fixed reward (R) per new goal (g) found is given to the robot. In experiment 2, $N_{goals} = 8$, and $N_{tasks} = 12$. 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.

3.3. Experiment 3: Adaptation to a Novel Environment

One question that arises from the previous experiment is how the higher level network may adapt to use the lower level's motor primitives in different environments. For example, one might ask if the left turning behavior at T

intersections is dependent on the particular wall configuration of the environment. Would the right turn primitive continue to function for turns significantly greater than 90° ? In order to evaluate the ability of the higher level network and control neurons to adapt to use the lower level primitives evolved in experiment 1 in a different environment, the following experiment was conducted. While keeping the bottom level genome evolved in experiment 1 constant, the control and higher level neurons' genomes were free to evolve to direct the robot to a goal in a novel environment. The wall configuration at the first intersection was changed to include a right wall, and the angle of the second right turn was increased to 130° .

The fitness function was the same as in experiment 2, except that $N_{goals} = 1$, and $N_{tasks} = 1$. Since only one goal was present, only one set of task neuron initial activities was evolved. The experiment was allowed to run for 200 generations, with 1 trial per generation. As in the prior experiments, each trial ended either when the goal was reached or when a wall collision occurred, triggering repositioning of the robot at the starting point for the next trial (Figure 11).

3.4. Experiment 4: Sequence Retention in an Enlarged Environment

An interesting question is whether travel time or distance is intrinsically represented in the dynamics of the evolved neural network, or whether the turn sequence can be generated over any arbitrary distance and time. In order to answer this question, the following experiment was conducted.

The size of the eight-goal environment from experiment 2 (Figure 8) was doubled. More specifically, the corridor distances were doubled, while the corridor widths remained the same. The evolved controller from experiment 2 was then used to control the robot in this larger environment. The initial task neuron values which led to the six stable goals in experiment 2 were then loaded into the controller, and the resulting movement sequences were observed.

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 (e.g., 0.74, -0.87), and right turns were generated for another set (e.g., -1, 0.23) (Figures 2-4). 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 experiments 2 through 4.

As seen in Figures 3 and 4, 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 5. 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 5 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 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. The control neurons can exert parametric control of the primitives. For example, a successively smaller or larger control neuron activity may lead to turns that are progressively closer to the wall. The unstable regions of Figure 5 indicate that the turns got too close to the wall, leading to collisions. Collision avoidance competes with the control neurons' "turn" signals. In the unstable regions of the phase plot (Figure 5), 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 (~ 2 days of run time) 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 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.

The progress of evolution is depicted in Figure 7, where the dashed line indicates the average population fitness, and the solid line indicates the fitness of the best robot of each generation.

Different $\gamma_{task}(0)$ activities led to differently fluctuating patterns in the control neuron activities (Figures 9 and 10). 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 5. Figures 8 and 9 show the trajectory and neuronal activities, respectively, for a left-right-right turn sequence. Figure 10 shows the control and task neuronal activities for five additional turn sequences. The results shown here are for a controller which learned to reach six goals stably ($\geq 70\%$ of the time) in 25 generations. As stated earlier, another controller evolved to reach 7 goals stably. The results of the 6-stable-goal controller are shown here because they demonstrate greater amplitude fluctuations of the control neurons, due to a smaller evolved value of the time constant τ in Equation 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). Setting the initial conditions of top level neurons (while keeping the

task neuron initial activities constant) to random values in the activity parameter range $(-10:10)$ also led to sequence instability. 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.

With or without motor noise, 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 the decrease in the amplitude of control neuron fluctuations over time (Figures 10c and d). As the fluctuation amplitude decreases, the control neuron activity tends to hover around the turn threshold. As seen in the phase plot of Figure 5, this region is unstable.

Figure 6a 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 described in Table 1. The sequences are labeled in the figure, e.g., LRL for left, right, and left turns. It is observed that the sequence patterns are arranged in 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 6a. Each of these two clusters is then further divided into sub-clusters, depending 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.

The hierarchical ordering of turn sequences into progressively smaller regions of the initial task neuron activity space, as additional turns are added, is represented in Figure 6b. 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.

4.3. Experiment 3: Adaptation to a Novel Environment

In contrast to the larger number (24) of generations required to evolve multiple goal-directed movement sequences, a controller with the ability to navigate to the single goal in the novel environment of experiment 3 emerged quickly, within only four generations (Figure 13). Note that the two principal differences between the environments of experiments 2 and 3 are the presence of a right side wall at the first intersection and a greater turn angle (130° versus 90° previously) at the second intersection. The ability of the controller to successfully adapt to this novel environment is not surprising when one examines the behavior of the robot. A strategy of following a left/right wall prior to turning left/right at an intersection emerges. For the initial left turn, the robot immediately veers left and follows the left wall until reaching the intersection where the left sensors no longer detect the wall. The competition between

obstacle avoidance and the leftward influence of the control neurons then shifts in favor of the left turn. The activity of control neuron 0 then increases above the right turn threshold of 0.4 (Figure 12) and right wall following occurs until the next intersection. The robot then turns right, and continues to turn right until the right-facing sensors detect the wall and obstacle avoidance again balances the right-turn influence of the control neurons.

This wall-following strategy contrasts with the turning strategy observed in experiment 2, in which the robot tended to remain in the center of a corridor and to turn only after running nearly straight into a wall at the next intersection (Figure 8).

As was done in experiment 1, Figure 5, the relation between movement behavior and the activities of control neurons 0 and 1 is quantified in the phase plot of Figure 14. 441 trials, each starting at the same place (bottom right corner, as shown by the robot picture in Figure 11), were run. The robot had a choice of either turning left at the first intersection (time step 200 in Figure 11), or else proceeding straight "up" the corridor. A barrier was placed to prevent the robot from reaching the second intersection (at time step 450 in Figure 11), for simplicity of analysis. In each trial, a different set of

the two control neurons' activities, held constant throughout the trial, ($A=[0:1]$, step size = 0.05) was tested and the resulting movement behavior was recorded. As in Figure 5, a clear bifurcation of movement behavior relative to control neuron activity can be seen in Figure 14. In Figure 14, however, the two movement behaviors are either a left turn or straight movement at the first intersection (Figure 11). Note the smaller region of instability between left turns and straight movement in Figure 14 compared to the instability region between left and right turns of Figure 5. The smaller instability region in Figure 14 is likely due to the simpler nature of the task (differentiating left turns and straight movements, versus left and right turns in Figure 5). Also note the large amplitude fluctuation of control neuron 0's activity in the adaptation task's left-right turn sequence (Figure 12) compared to that in the left-right-right sequence of Figure 9. As mentioned previously for experiment 2, larger amplitude fluctuations should make the transition between movement behaviors less sensitive to noise. Collisions are less likely since the control neuron activity quickly moves away from potentially unstable threshold regions.

4.4. Experiment 4: Sequence Retention in an Enlarged Environment

When six sets of task neuron initial activities ($\gamma_{task}(0)$), corresponding to the six stable goals found in experiment 2, were loaded into the controller of the robot when placed in an enlarged environment, only two out of the six $\gamma_{task}(0)$ sets led to the robot reaching the same goal as in the smaller environment of experiment 2 (Table 2). In Table 2, the movement sequences in the original environment of experiment 2 are shown in the left column, and sequences in the enlarged environment of experiment 4 are in the right column. These results indicate that the internal dynamics of the higher level neurons appear to proceed at their own rate, relatively independent of external inputs. For example, the Right-Right-Left (RRL) sequence turned into an RLL sequence after doubling the maze size since the second right turn was missed due to the longer corridor. The control neurons' outputs then passed the turn threshold, leading to a premature left turn when the robot finally arrived at the intersection (Figures 15 and 10d).

This result is not surprising, since no direct sensory input reaches the higher level network. It would be interesting to explore in future research the capability of

the control neurons, which are connected more closely to sensory signals via their connections with the bottom level network, to modify the higher level activity based on environmental changes.

5. Discussion

The work presented here describes a hierarchical model of behavioral sequence memory and generation in a single distributed network. 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 important 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. The primitives used here are intentionally simple compared to those in multi-dimensional tasks such as arm movements. Superposition of more complex primitives and tasks could be used as well. However, the simpler model described here is sufficient to describe the concept of a distributed neural network controller that uses initial sensitivity of its dynamics for sequence representation. Since the bottom level 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 ($\tau_{\text{task}0} = 70$, $\tau_{\text{task}1} = 52$ in Equation 1) so that neuronal activity changes much more gradually and is less affected by short-term sensory changes. These emergent multi-time-scale dynamics are a common feature of biological memory systems. The role of different time

scales in the synchronization and bifurcation of coupled systems, similar to the bifurcations of control and task neuron activities of figures 5 and 6a, was studied by Fujimoto and Kaneko (2003). The need for the neural integration of information over multiple time scales in order to take advantage of various-duration events and environmental regularities was also demonstrated in evolutionary robotics by Nolfi (2002) and by Precup and Sutton (1997) in the area of reinforcement learning.

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.

A total of 11 neurons were used in the current network. Although an arbitrarily large network could theoretically generate similar performance, the search space of network parameters (weights, time constants, etc.) would increase and yield ever slower performance of the GA. Imposing left-right symmetry on the lower network might increase the GA search speed by decreasing the search space. However, such a network symmetry constraint might adversely affect performance for more complex and asymmetric primitives, sequences, and environments. A network of too few neurons would not contain any solutions in its parameter space that would allow a given number of goals/sequences to be found. For example, when only 4 neurons were used in the top level

during model development, task neuron activity was unimodal, such that certain turn combinations were never found.

Two task neurons were used to encode primitive sequences since two primitives were encoded in the bottom level. It is not necessarily the case that the number of task neurons must always equal the number of primitives, but such a relationship was not investigated here.

Although the initial sensitivity of the movement sequences generated to task neuron activations (Figures 6, 9, and 10) 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, i.e., separate

training of the bottom and top levels. This task separation was done in analogy to the presumed sequential development of motor primitives in young humans, which are combined into sequences only later, presumably when the cortex has further developed. In another analogy, one could compare it to the evolutionary development of lower brain stem structures for "primitive" movements, with later evolution of cortical structures (the "top level") allowing for complex combinations of the primitives. Nevertheless, it would be interesting to learn/evolve both levels simultaneously and see what types of primitive and sequence organization emerge. 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.

Although other examples of evolved robot controllers can generate multiple sequences of movements to reach various goals, they generally require dynamic synaptic weight changes (Ziemke & Thieme, 2002), sustained neural activity between tasks (Blynel, 2003), and/or a change in sensory-motor mappings between different tasks (Ziemke & Thieme, 2002).

Different sensory-motor mappings for different complex tasks are often reasonable assumptions, as in the different mappings between sensation and hand movements in piano-playing versus communicating in sign-language. However, assuming different sensory-motor mappings, and hence different neuronal network modules and synaptic weights, for sequential tasks that require only different combinations of a set of movement primitives would quickly exhaust the supply of neurons in the brain. Further, simply switching among completely separate motor-primitive modules (Tani & Nolfi, 1999) for different tasks would greatly limit the diversity of possible movements, again requiring a proliferation of slightly different modules to accomplish similar tasks.

Wolpert and Kawato (1998) propose that an explosion in the number of motor primitive modules needed for arbitrary movements could be avoided through a linearly weighted

combination of a given set of modular outputs. However, one question with their model is how generalization can be achieved simply through linear interpolation among the modules. It is proposed that certain kernel modules have to be self-organized through their mutually interactive computations for the purpose of attaining the generalized internal representation.

Although a modular system can avoid the stability-plasticity dilemma (Grossberg, 1982), or the catastrophic forgetting (French, 1991; McCloskey & Cohen, 1989) of old primitives when new ones are learned, its abilities to generalize to novel environments, create sufficiently diverse combinations of primitives, decide when to create new modules, and select the appropriate module combinations for different tasks and environments are questionable.

Requiring fast, dynamic changes of synaptic weights during different parts of a movement sequence is also not a biologically plausible means of movement sequence representation and generation. As a purely artificial mechanism in the context of delayed response tasks, it is an effective solution, although it requires significantly more training (1000 vs. 200 generations) than the current model (Ziemke & Thieme, 2002).

Blynel (2003) described a novel, evolved network that allowed a simulated robot to "remember" the location of a goal that had been found through exploration of the two arms of a T maze. The goal location was represented by the continuous, dynamic activity of the network's neurons. Further, it was shown that the initial activity of a single neuron in the network determined whether the robot turned left or right at the intersection. There is therefore some similarity between the direction-determining neuron in Blynel's network and the control neurons in the current model. The key difference lies in the need to maintain the pattern of neural activity between trials in order to find a goal in Blynel's model. If different, "distractor" tasks were to be interposed between successive trials, the robot would forget the turn direction that it had previously learned. Thus, although both the Blynel model and the current one represent different turn directions through the dynamic activity of their networks, only the current model proposes a means whereby sequences of turns can be both generated and then regenerated at arbitrary later times through the modulation of task neuron activities without the need for synaptic weight change.

Despite the theoretical limitations of strictly modular systems outlined above, they remain a useful tool

in designing robot controllers. The reinforcement learning (RL) community has made successful use of learning modules for specific simple tasks, which can then be combined in a variety of ways to generate more complex tasks and sequences of tasks. For example, McGovern et al. (1997), use macro-actions, groups of simpler primitive movements, to accelerate the search for rewarding sequences of finite state movements while showing that inappropriate use the macros can also slow it. Unfortunately, most macros must be hand-designed, and it is hard to predict whether particular macros will enhance or worsen task performance. However, without them the search for rewarding movement sequences is greatly lengthened. Indeed, much research has been done on accelerating the search phase of reinforcement learning, increasing robustness of models to environmental uncertainty, and increasing solution transfer across problems through numerous types of modular and hierarchical architectures, (Subgoals: Sutton et al., 1999, McGovern and Barto, 2001; Hierarchical Abstract Machines: Parr and Russell, 1997; MAXQ: Dietterich, T., 2000), and enabling more complex goal-oriented movement sequences to be generated. Genetic algorithms have also been used to help optimize RL parameters (Mesot et al., 2002). Most of this work has focused on artificial discrete time, finite state

environments, or "grid-worlds", with minimal non-linearities and non-stationarity. Doya et al. 2002, have addressed non-linear and non-stationary continuous control of a pendulum through their "multiple model-based" RL, which shares use of a softmax function for module selection and combination with the "multiple paired forward-inverse models" of Wolpert and Kawato (1998) mentioned previously.

5.1. Limitations and Directions for Future Research

The present model does not address the development of synaptic connections through error-driven learning in biological systems, but instead uses a genetic algorithm to find a single synaptic weight pattern that is effective in completing multiple tasks. Although error-driven learning algorithms, such as backpropagation through time (Haykin, 1994; Werbos, 1990), have been successfully used to train recurrent neural networks, the manual selection of appropriate parameter values is a daunting task. Until more effective and/or biologically plausible training methods are developed, genetic algorithms remain one of the most powerful tools in training neural network robot controllers for complex movement tasks.

As seen in the results of experiment 4, one limitation of the current model is its inability to respond effectively to environmental changes without further weight modification. When the length of the corridors was doubled, the robot usually failed to reproduce the turn sequence which had been learned in the smaller environment. The dynamics of the higher level network were essentially independent of the external environment. The current model is effectively an open-loop, top-down controller that executes sequences by rote at a fixed speed. The work of Smith et al. (2002) is relevant to this time/speed scaling, or "temporal adaptivity" dilemma. They found that Gas Nets, using internal neuron dynamics modeled on biological gaseous neuromodulators, are better able to adapt to environmental and movement speed changes than traditional neural networks that rely solely on the movement between fixed equilibrium points to switch speeds, or turns as in our model's top level. When environmental changes prevented the robot from turning at the usual time, the top network activity continued to progress toward the next turn (top level equilibrium point) in the sequence, skipping a turn instead of merely delaying it until the next intersection. Also of relevance to this problem is the work of Ijspeert et al. (2003), which describes a novel

form of on-line environmentally mediated modification of previously learned attractor dynamics in response to environmental changes.

The higher level network's influence on the outputs of the lower level is disproportionately large compared to the lower level's influence on it. The higher level's relative isolation from the "real world's" sensory input is in stark contrast to the rich flow of both physical and temporal sensory information which is integrated in the primate lateral prefrontal cortex during the learning of movement sequences (Ninokura et al., 2003). Although both the model's control neurons and primate LPFC neurons integrate both temporal sequence and physical sensory information, the monkey can modulate the speed of its sequence generation, whereas the current model cannot. Future work will therefore explore the possibility of better modulating the activity of the higher level through bottom-up connections in a way which reflects environmental changes.

6. Conclusion

This work has demonstrated an approach to adaptive, goal-directed, behavioral sequence representation in a

self-organized, hierarchical, recurrent neural network controlling a simulated mobile robot. Different types of dynamic structures self-organize in the lower and higher levels of the network for the purpose of achieving complex navigation tasks. Top-down behavioral switching emerges through parametric bifurcation of lower level activity via control neurons. 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 dynamic systems. Task-neuron modulation could be effected by even higher level networks which could represent sequences of sequences for different sets of ever more complex tasks. This research serves as an example of how complex dynamic structures with initial sensitivity and task-dependent temporal activity may self-organize to control simpler structures that encode movement primitives. Such structures may be analogous to those which encode movement sequences in biological neural networks, and may be a promising direction for research into mobile robot navigation.

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.
- Beer, R.D., Gallagher, J.C. (1992). Evolving dynamical neural networks for adaptive behavior. Adaptive Behavior, 1, 1, 92-122.
- Blynel, J. (2003). Evolving Reinforcement Learning-Like Abilities for Robots. In A. Tyrrell, P.C. Haddow, and J.

Torresen: 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. Cambridge, MA: MIT Press.

Dietterich, T., (2000). Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. Journal of Artificial Intelligence Research, 13, 227-303.

Doya K., Samejima K., Katagiri K., Kawato M. (2002). Multiple model-based reinforcement learning. Neural Computation, 14, 1347-1369

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.

Fikes, R. E., Hart, P. E., Nilsson, N. J. (1972). Learning and Executing Generalized Robot Plans. *Artificial Intelligence*, 3, 251-288.

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*. Cambridge, MA: MIT Press.

French, R. M. (1991). Using semi-distributed representation to overcome catastrophic forgetting in connectionist networks. *Proceedings of the 13th Annual Cognitive Science Society Conference* (pp. 173-178). Hillsdale, NJ: Lawrence Erlbaum Publishers,.

Fujimoto, K., Kaneko, K. (2003). How Fast Elements can Affect Slow Dynamics. *Physica D: Nonlinear Phenomena*, 180, 1-2.

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*. Boston, MA:
Kluwer Academic Publishers.

Grossberg, S. (1982). *Studies of Mind and Brain: Neural
Principles of Learning, Perception, Development, Cognition,
and Motor Control*. Boston Studies in the Philosophy of
Science, Vol. 70. Dordrecht, Holland: D. Reidel Publishing
Co.

Harvey, I., Husband, P., Thompson, A., and Jakobi, N.
(1997). Evolutionary Robotics: the sussex approach.
Robotics and Autonomous Systems, 20, 205-224.

Haykin, S. (1994). *Neural Networks: A Comprehensive
Foundation*. Prentice-Hall Publishers.

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

Ijspeert, A., Nakanishi, J., Schaal, S. (2003). Learning attractor landscapes for learning motor primitives. In: Becker, S., Thrun, S., Obermayer, K. (Eds.), *Advances in Neural Information Processing Systems*, 15. Cambridge, MA: MIT Press.

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).

Cambridge, MA: MIT Press.

Mesot, B., Sanchez, E., Pena, C., Perez-Uribe, A. (2002).

SOS++: Finding Smart Behaviors Using Learning and

Evolution. In Standish, Abbass, Bedan (Eds.), Artificial

Life VIII, MIT Press, pp. 264-273.

McCloskey, M., Cohen, N. J. (1989). Catastrophic

interference in connectionist networks: The sequential

learning problem. The Psychology of Learning and

Motivation, 24, 109-165.

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.

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. Cambridge, MA: MIT Press.

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. Cambridge, MA: MIT Press.

Nolfi, S., (2002). Evolving Robots able to Self-localize in the Environment: The Importance of Viewing Cognition as the Result of Processes Occurring at Different Time Scales. *Connection Science*, 14, 3, 231-244.

Parr, R., Russell, S. (1998). Reinforcement Learning with Hierarchies of Machines. In: Jordan, M. I., Kearns, M. J., Solla, S. A., (Eds.), *Advances in Neural Information Processing Systems*, 10. Cambridge, MA: MIT Press, pp. 1043-1049.

Precup, D., Sutton, R. S. (1997). Multi-time Models for Temporally Abstract Planning. In: Jordan, M. I., Kearns, M. J., Solla, S. A., (Eds.), *Advances in Neural Information Processing Systems*, 10. Cambridge, MA: MIT Press, pp. 1050-1056.

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.

Smith, T., Husbands, P., Philippides, A., O'Shea, M. (2002). Temporally Adaptive Networks: Analysis of GasNet Robot Control Networks. In Standish, Abbass, Bedan (Eds.), *Artificial Life VIII*, MIT Press, pp. 274-282.

Sun, R., Merrill, E., Peterson, T. (2001). From Implicit Skills to Explicit Knowledge: A Bottom-Up Model of Skill Learning. *Cognitive Science*, 25, 2, 203-244.

Sutton, R., Precup, D., Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112, 181-211.

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.

Werbos, P. (1990). Backpropagation through time: what it does and how to do it. *Proc. IEEE*, 78, 1550-1560.

Wolpert, D. M., Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11, 1317-1329.

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

Ziemke, T., Thieme, M. (2002). Neuromodulation of Reactive Sensorimotor Mappings as a Short-Term Memory Mechanism in Delayed Response Tasks. *Adaptive Behavior*, 10, 3/4, 185-199.

Figure and Table Captions

Figure 1: Conceptual diagram of network architecture. The bottom level receives sensory inputs, interacts with control neurons, and sends motor commands to the robot wheels. The top level modulates the control neurons' activities over time to generate movement sequences, sensitive to the initial activities of the task neurons.

w^l =lower level weights, w^c =control neuron weights, w^h =higher level weights. (a) Initially, the bottom level and control neurons evolve in a T-maze environment. (b) The top level is then added in an 8-goal-maze environment and evolves while the bottom level parameters remain constant.

Figure 2: T maze environment with left and right turn trajectories. Turns occurred at integration step=200, corresponding to the neuronal activity traces in figures 3 and 4. The graph at the top of the environment illustrates the linear fitness function dependent on the robot's position at the top of the environment. Fitness increases as the robot approaches the left or right ends of the T.

Figure 3: T maze left turn. Top: Neuronal activity of control neurons; Middle: Lower level motor output node activity; Bottom: Left and right sensor activities.

Figure 4: T maze right turn. Top: Neuronal activity of control neurons; Middle: Lower level motor output node activity; Bottom: Left and right sensor activities.

Figure 5: Experiments 1 and 2. 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. 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 plotted. Black = Left turn, White = Right turn, Grey = Collision with wall.

Figure 6: (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 2, respectively. Plotted numbers correspond to sequences as in Table 1. Sequences also indicated by L=Left, R=Right. Note the emergent topological ordering, such that adjacent sequence regions differ by one bit, or one turn. Sequence regions differing only in the third turn (right-most letter) are separated by a dashed line, while those differing in the second turn are separated by a solid line. Right- and Left-beginning sequences cluster together on right and left sides of the figure, respectively, and are separated by a thicker solid line. (b) Hierarchical tree-structure of sequence organization in (a). Top: first turn; Middle: second turn; Bottom: third turn.

Figure 7: Population (bottom curve) and Best Robot (top curve) fitness vs. generation of evolution. The maximum fitness at generations 25 and 26 corresponds to the finding of six stable goals by the best robot.

Figure 8: 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 9.

Figure 9: 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.

Figure 10: Control and Task neuron (initial node) activities for the following turn sequences. L=Left, R=Right. a) LLL; b) LLR; c) RLL; d) RRL; e) RRR

Figure 11: Environment and trajectory for experiment 3. G=Goal. Turns occur at steps 200 and 450, corresponding to the neuronal activity in figure 12.

Figure 12: Experiment 3. Top: Neuronal activity of control and task (initial node) neurons; Middle: Lower level motor output node activity; Bottom: Left and right sensor activities.

Figure 13: Experiment 3. Population (bottom curve) and Best Robot (top curve) fitness vs. generation of evolution. The goal was found in generation 4.

Figure 14: Experiment 3. Phase analysis of movement behavior as a function of control neuron activation. Black = Left turn, White = Straight movement, Grey = Collision with wall. Note the smaller region of instability between left turn and straight movement regions compared with the instability between left and right turns of Figure 5.

Figure 15: Experiment 4. Control and Task neuron (initial node) activities when the distance to the goal is doubled. In the original smaller environment of Experiment 2, this set of initial task neuron activities led to a Right-Right-Left turn sequence. When the distance to the goal was doubled, a Right-Left-Left sequence occurred since the activities of control neurons 0 and 1 passed the turn threshold prior to reaching the second intersection. The

second turn occurred at step 450 in the original environment of figure 10d, and at step 1000 in the double-length environment.

Table 1: Movement Sequence representation used in Figure 6. L=Left Turn; R=Right Turn; 0=Left Turn; 1=Right Turn.

Table 2: Movement sequences in the original environment of experiment 2 (left column) and in the enlarged environment of experiment 4 (right column). L=Left turn; R=Right turn. Note that only 2 of 6 sequences were preserved after enlarging the environment.