Adaptive Behavior, Vol.13, No.3, pp.211-225, 2005

How hierarchical control self-organizes in artificial adaptive systems

Running Title: **Hierarchical control in adaptive systems**

Rainer W. Paine, Jun Tani*

RIKEN Brain Science Institute, Laboratory for Behavior and Dynamic Cognition, 2-1 Hirosawa, Wako-shi, Saitama, 351-0198, Japan

*To whom correspondence should be addressed. Email: *tani@brain.riken.jp*,

Tel. 81-(0)48-462-1111; Fax: 81-(0)48-467-7248

Abstract

Diverse, complex, and adaptive animal behaviors are achieved by organizing hierarchically structured controllers in motor systems. The levels of control progress from simple spinal reflexes and central pattern generators through to executive cognitive control in the frontal cortex. Various types of hierarchical control structures have been introduced and shown to be effective in past artificial agent models, but few studies have shown how such structures can self-organize. This study describes how such hierarchical control may evolve in a simple recurrent neural network model implemented in a mobile robot. Topological constraints on information flow are found to improve system performance by decreasing interference between different parts of the network. One part becomes responsible for generating lower behavior primitives while another part evolves top-down sequencing of the primitives for achieving global goals. Fast and slow neuronal response dynamics are automatically generated in specific neurons of the lower and the higher levels, respectively. A hierarchical neural network is shown to outperform a comparable single-level network in controlling a mobile robot.

Key words: Neural Network, Hierarchy, Self-organization, Initial Sensitivity, Genetic Algorithm, Evolutionary Robotics.

1. Introduction

It is well known that diverse, complex, and adaptive animal behaviors are achieved by organizing hierarchically structured controllers in motor systems. The levels of control progress from simple spinal reflexes, central pattern generation in the spinal cord and brainstem, their elaborated coordination in sensory-motor cortex, through to executive cognitive control in the frontal cortex (Kelly, 1991). In developing artificial agents such as autonomous robots, various types of multi-level structures have been introduced and shown to be effective (Blynel & Floreano, 2002; Paine & Tani, 2004; Tani, 2003; Tani & Nolfi, 1999; Yamauchi & Beer, 1994; Ziemke & Thieme, 2002). However, few studies have shown how such effective hierarchical controllers can self-organize through system adaptation without explicitly designing them. The current paper reports that multiple hierarchically structured controllers can self-organize through evolution of a simple artificial neural network implemented in an artificial sensory-motor system. It is shown that a simple constraint on information flow in the network reinforces stable self-organization of each level.

Much research has been done in the field of "machine learning" with numerous types of multi-level architectures in order to accelerate the search phase of reinforcement learning, increase the robustness of models to environmental uncertainty, increase solution transfer across problems, and to generate more complex goal-oriented movement sequences (Dietterich, 2000; Doya, Samejima, Katagiri, & Kawato, 2002; McGovern & Barto, 2001; Parr & Russell, 1998; Sutton, Precup, & Singh, 1999). However, in most of these studies, each level has to be learned separately by utilizing external cues, such as subgoals.

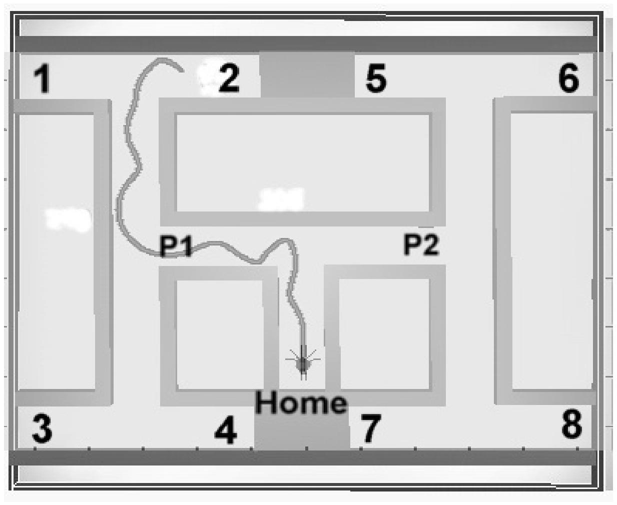

Figure 1 Maze environment. Shows trajectory for a Left-Right-Right sequence. A simulated mobile robot learns to reach 8 different goals starting from a home position. If the robot reached positions P1 or P2 as a function of BN activities, left or right turns, respectively, were recorded for Figure 4.

The current work explores the possibility that multiple levels of hierarchical control may self-organize through simple adaptation processes in artificial neural networks.. In our experiments, a mobile robot explores the maze environment as shown in Figure 1. The task of the robot is to find navigation paths reaching as many different goals as possible from a start position, while synaptic connectivity weights of the network controlling the robot are evolved for better task performance using a standard genetic algorithm (GA) (Holland, 1975; Lipson & Pollack, 2000; Ruppin, 2002). The robot must acquire different levels of behavioral skills in order to achieve the goal. First, the robot should become able to perform collision-free maneuvering while safely turning left or right at corners. Second, the robot should become able to navigate to the goals by means of sequencing left or right turns at corners. Our experiments will show how this sort of two-level hierarchical structure can self-organize.

The robot controller is implemented with a continuous time recurrent neural network (CTRNN) whose synaptic connectivity is evolved by a GA (Yamauchi, & Beer, 1994). We test two types of CTRNNs as shown in Figure 2. Figure 2A is called a “bottleneck” network since there is a restriction of information flow between the upper and lower parts of the network. The neural activations can propagate to the other level only through the bottleneck neurons (BNs). There are sensory inputs and wheel-motor outputs in the lower part of the network. Figure 2B is a standard fully connected CTRNN.

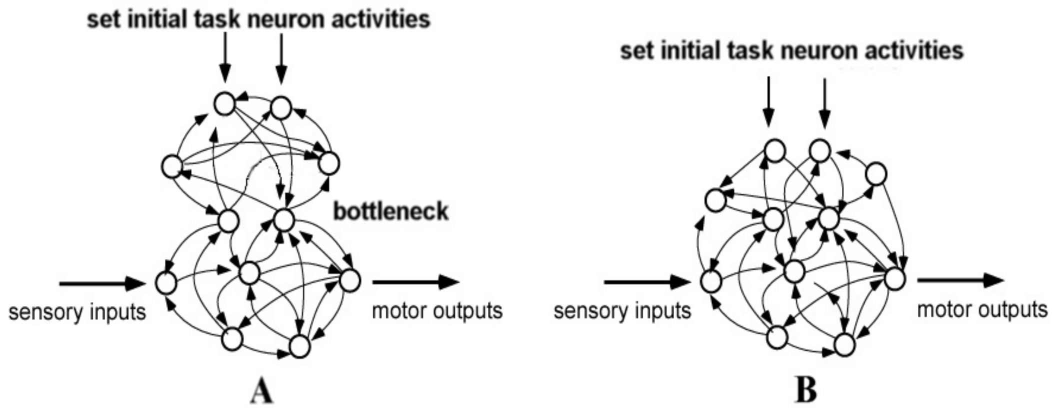
set initial task neuron activities
bottleneck
sensory inputs
motor outputs
A
set initial task neuron activities
sensory inputs
motor outputs
B

Figure 2 CTRNN. Bottleneck network in (**A**) and a standard, fully connected CTRNN in (**B**).

There are special neurons called “task” neurons (TNs) in both types of networks. The idea here is that different initial internal activity states (γ in Equation 1, analogous to cell membrane potentials or firing frequencies in biological neural networks) set in the TNs would lead the robot to reach different goal positions, acting as action programs by utilizing the initial sensitivity characteristics of the global neural network dynamics (Nishimoto & Tani, 2004). Blynel (2003) found a similar initial-state coding in an evolved network that allowed a simulated robot to “remember” the location of a single goal that had been found through exploration of the two arms of a T maze. It was shown that the initial activity state 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 goal-determining neuron in Blynel's network and the TNs in the current model. The current model expands on the use of goal-determining initial neural states to represent multiple turn sequences.

Sets of the two TN's initial states at time *t*=0, given by the $M \times 2$ matrix $\gamma_{M,2}^{task(0)}$, are evolved along with the connective weights and neuron parameters of the network. The fitness function maximizes the number of different goals achieved within *M*=12 sessions of robot travel. The robot starts each session from the home position, while set with one of 12 differently evolved TN initial states. The initial states of all other neurons are set to zero. Thus, a single network with a single set of parameters (synaptic weights, time constants, activation biases) can encode multiple movement sequences through different TN initial states.

The simulation experiments consist of 20 runs of evolution for each type of network. Each run has 200 generations with an 80 robot population per generation. The bottleneck network has 11 neurons, and the fully connected CTRNN consists of 9 neurons, with a total of 81 synaptic weights in each network. The same number of synaptic weights are chosen in each network to equalize the search space of the GA. Thus, any performance differences that might arise should be due to differences in network structure, and not to the learning algorithm employed. Experiments with an 11 neuron fully connected network yielded similar results. The simulated robot controllers are then tested in a real Khepera II robot.

2. Methods

2.1 Simulations of Hierarchical "Bottleneck" versus Fully Connected Controllers

Robot simulations were carried out using *Webots 4* software (Cyberbotics Ltd., www.cyberbotics.com) to simulate a Khepera II robot (built by K-Team, www.k-team.com). All neurons in the simulations presented here use the following equations and parameters for a Continuous Time Recurrent Neural Network (CTRNN) (Yamauchi & Beer, 1994). In Equation 1, γ_i is the internal activity 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 kth external sensory neuronal activation, I_k, and signals from the jth presynaptic neuron with activity A_j. The signal from the presynaptic neuron is weighted by weights $w^p{}_{ij}$, and the sensory input is weighted by $w^s{}_{jk}$. N is the number of neurons in the network, and R is the number of sensory receptors which send input signals to the network.

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

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.

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

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

The numerical integration is carried out using the Forward Euler method. Except for the task neurons, whose initial activity states, $\gamma_{M,2}^{task(0)}$, are obtained through evolution, the neuronal activation, γ_i, is initialized to 0 at the start of integration.

The bottleneck CTRNN has 5 neurons in the lower part, 2 BNs, and 4 neurons in the upper part, including 2 TNs. The standard CTRNN has 9 neurons, including 2 TNs. Both networks receive inputs from the Khepera robot's 8 infra-red proximity sensors and generate motor outputs to the two wheels, determining left and right wheel speeds. The sensory inputs are sent to the 5 neurons in the lower part of the bottleneck network, and to 5 arbitrarily chosen neurons in the fully connected network (including output but not task neurons). The input is modified by randomly adding or subtracting up to 5% of its value as noise. Further, motor noise is added to the wheel speed commands sent to the robot. The integer speed command is increased or decreased by one with a probability of 20% on each simulation time step. The motor noise helps ensure that the robot experiences wall collisions early during evolution, so that controllers with obstacle avoidance are more likely to evolve quickly.

A standard genetic algorithm (GA) with mutation, but without crossover, is employed to evolve the weights and parameters (τ , θ, w, $\gamma_{M,2}^{task(0)}$) of the network (Goldberg, 2002; Mitchell, 1998). The ranges of the parameters are set as follows.

$$\tau \in [1,70]\,; \theta \in [-1,1]\,; w \in [-5,5]\,; \gamma_{M,2}^{task(0)} \in [-10,10]$$

Each parameter is encoded using 5 bits .The mutation rate per bit is set at 2% for all simulations. The population consists of 80 robots. The twenty robots with the best fitness reproduce each generation (*elitism*). Each parent makes 4 copies of itself with mutation. Of the best robot's offspring, one is an exact copy of the parent without mutation.

Each session consists of 2 trials to examine the ability to repeatedly reach the same goal despite sensory noise on different trials. Robots which find the same goal on both trials are rewarded more than those which find two different goals or only one goal. Once goal-finding ability evolves, motor noise is set to 0 to better allow evaluation of the controller's ability to find the same goal on repeated trials. A trial is ended when the robot either finds a goal, collides with a wall, or uses up the 2000 time steps allocated per trial. The robot is then repositioned to the same starting point.

After 12 sessions, each robot is evaluated based on how many different goals it can reach repeatedly. A two-component fitness rule is used to evaluate each robot (Equation 4). The first component (F_{OA}) consists of a reward for straight, fast movements and obstacle avoidance. A fitness rule (Floreano & Mondada, 1994) is adopted for this purpose and is shown in Equation 5. 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, a measure of the distance to the nearest obstacle. It is scaled to a range of 0 to 1, and is used to reward obstacle avoidance.

$$F = F_{OA} + F_{goal} \tag{4}$$

$$F_{OA} = \overline{V} \cdot (1 - \sqrt{\Delta V}) \cdot (1 - S_{\max}) \tag{5}$$

The second component of the fitness rule, F_{goal}, rewards the robot for reaching goals consistently (Equation 6).

$$F_{goal} = \sum_{g=1}^{Ngoals} \max_{i=1:Nsessions} (R_{g1} + R_{g2}) \tag{6}$$

Here, a fixed reward (R) per new goal (g) reached repeatedly on two trials is given to the robot. Ngoals = 8, and Nsessions = M = 12. When $g2=g1$, $R_{g2}=R_{g1}=10$. When $g2 \neq g1$, $R_{g2}=0$. In other words, a greater reward is given for finding the same goal on both trials than for finding two different goals or only one goal for a particular pair of TN initial states. Total fitness is averaged over the robot's run time. F_{OA} is scaled to a range of 0:1 while R=10 to ensure that goal-finding, once it emerges, is the dominant force in evolution.

3. Results

The following sections present the results of simulations (sections 3.1-3.2) and real robot performance (section 3.3). Section 3.1 statistically compares the performance of the bottleneck and fully connected networks. Sections 3.2-3.2.2 analyze the bottleneck network and the effect of its topological constraints on the evolved localization of specific functions to the separate levels of its architecture. Section 3.2.3 examines the evolved neuronal time constants of bottleneck and fully connected networks and how their common features contribute to task execution. The insights gained from these sections help explain the functional differences between the bottleneck and fully connected networks, as described in 3.2.4. Finally, transfer to the real robot is briefly discussed in section 3.3.

3.1 Performance of the Hierarchical "Bottleneck" and Fully Connected Controllers

It was found that the best performance was obtained in the bottleneck network. In twenty evolutionary runs, the mean number of different goals reached was 5.1 for the bottleneck CTRNN, and 2.3 for the standard CTRNN (p=0.015, two-tailed Student's paired t-Test indicating significantly different means). The bottleneck CTRNN found five or more goals on 14 of 20 runs, whereas the standard CTRNN found them on only 6 runs.

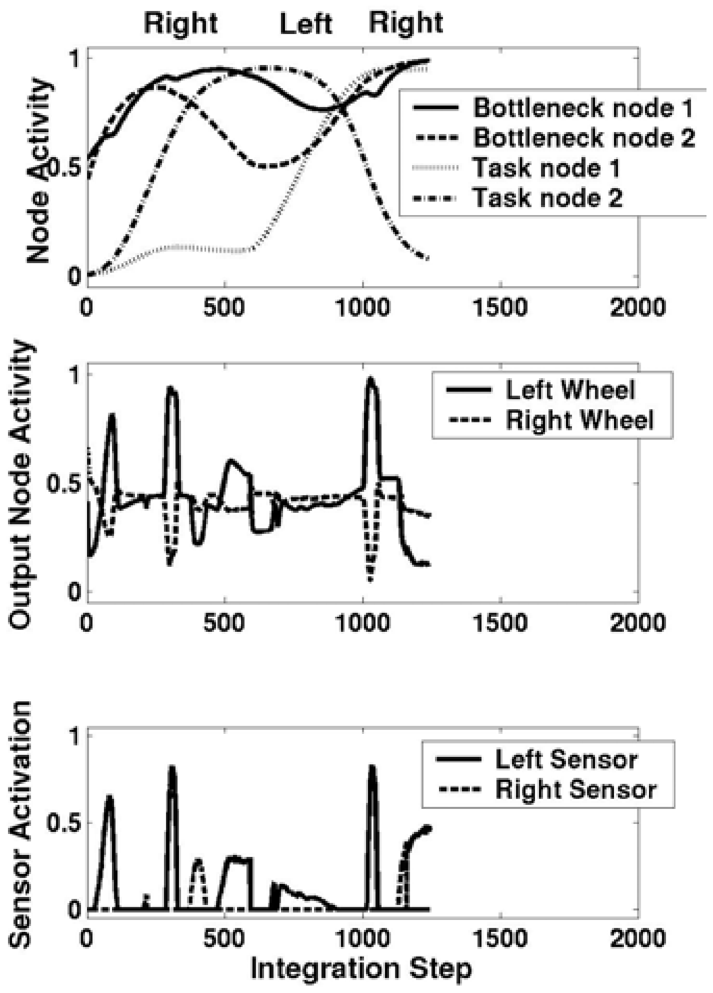

Figure 3 Neuronal activity for a Right-Left-Right turn sequence in the bottleneck network. Turns occur at the times shown by the arrows. **Top**: Neuronal activity of bottleneck and task neurons with slower evolved time constants to encode turns and sequences, respectively; **Middle**: Activities of motor output nodes that evolved faster neuronal time constants to respond to immediate task demands relative to sensor inputs (**Bottom**).

3.2 Analysis

The temporal neuronal activation profiles for an evolved bottleneck network, which found 6 different goals, are shown in Figure 3. The profiles correspond to a Right-Left-Right turn sequence, starting from the home position, that reaches goal 6 of Figure 1. The top row shows the activation profiles of two TNs and two bottleneck neurons (BNs, Figure 2A). The bottom row shows the profiles of the two motor output neurons in the lower part of the network. Observe that the time constants for the motor neurons are much faster than those of the TNs and BNs. The activation profiles of the BNs correlate with right and left turns, denoted by arrows in the top figure. For the right turn, both BNs have high activation values, while BN-2 takes a low value and BN-1 slightly decreases for the left turn. TN-2 shows a similar type of activation profile as the BNs, while the profile of TN-1 seems more complex, with medium activity levels correlated with left turns and both low and high levels with right turns.

3.2.1 Lower Level network and Bottleneck Neurons

We conducted further analysis of the functions of BNs in the same network. A phase space analysis for the BNs, focusing on the cornering behavior at corridor intersections, can be seen in Figure 4. It shows how the cornering behavior varies when the activations of two BNs are externally clamped to various values. Specifically, an evolved controller was tested by fixing 441 sets of BN states (21x21 combinations of the two BN activities (γ in equation 1) ranging from -10 to 10, at 1-unit intervals). For each pair of BN activities, the robot began a trial in the same home position as in Figure 1. If the robot reached position P1 or P2 in Figure 1, then a left or right turn was recorded, respectively. In other words, the upper level of the bottleneck controller was cut off, allowing a direct test of the BNs' effect on robot movement and demonstrating that obstacle avoidance and turning behavior are controlled by the lower level. One may observe that the BNs' activation space is divided into three regions, white, black and grey, which correspond to right turns, left turns and collisions with the walls, respectively.

These left and right turns, encoded in the lower level of the network, act as reusable motor patterns or "primitives" (Arbib, 1981; Fikes, Hart, & Nilsson., 1972; Giszter Mussa-Ivaldi, & Bizzi, 1993; Thoroughman & Shadmehr, 2000). As demonstrated in Figure 4, the bottleneck neurons' activities bias the robot to turn in a particular direction, in a manner analogous to command neurons (Ruppin, 2002) and the "parametric bias" of Tani (2003).

Analysis of the robot's behavior during the generation of Figure 4 yields some insights into the role of the BNs. Obstacle avoidance must be balanced with the tendency to veer in a particular direction due to the BNs' influence. For example, when the BN activities are set in the "Left" region of the figure, the robot starts veering left while near the Home position of Figure 1. When it gets "too close" to the wall, obstacle avoidance takes over causing it to turn away from the wall. A balance between obstacle avoidance and the BNs' leftward bias yields a wall following strategy on the left side of the corridor, leading the robot to make a left turn at the intersection. However, when the BN activities are set in the black collision region on the left of Figure 4, the robot again starts veering to the left, but collides with the wall because the BN bias is too strong, overriding obstacle avoidance. Further, collisions can occur at the transition between left and right BN activity regions due to a balance between the BN turn biases, leading the robot to head straight "up" from the home position to collide with the facing wall at the first intersection. In effect, it can't "decide" which direction to turn and hits the wall in front of it. Finally, a last cluster of collisions occurs in the lower right region of Figure 4 where the rightward BN bias on turning direction is too strong, again overriding obstacle avoidance. Collisions do not occur during the evolved controller’s movements since BN activities stay out of the collision region of Figure 4. These collisions were induced by artificially clamping BN activities as described previously to generate the figure.

3.2.2 Higher Level network with Task Neurons

As the lower level’s role in encoding turns and obstacle avoidance was demonstrated by cutting off the upper level, the upper level’s role in organizing the sequence is demonstrated first by cutting off the lower level, and then by mapping TN initial states to the resulting turn sequence. Figure 5, when compared to the top of Figure 3, shows that the TN and BN activity profiles are largely unaffected by disconnecting the upper from the lower level (setting all weights between the BNs and the lower level to zero). In other words, the upper level essentially exerts open-loop top-down control of the turn primitives in the lower level. Although this strategy is successful in the static environment used here, greater bottom-up modulation would be needed to preserve sequences in more realistic and variable environments (see Figure 15 of Paine & Tani, 2004).

A phase space analysis of the task neuron initial states is depicted in Figure 6, where the regions represent the initial states which result in the robot reaching different goals. These regions are labeled by the corresponding turn sequence, e.g., LRR for a Left-Right-Right turn sequence. The different numbers in the figure correspond to different turn sequences, as described in the legend of Figure 6.

In some runs in which a given number of goals were found by evolution, additional goals were found in the TN phase space analysis. Although the genetic algorithm searched only 12 different sets of TN states per robot of an 80 robot population in each generation, Figure 6 was generated by testing an evolved controller with 441 sets of TN initial states (21x21 combinations of two TN initial states ranging from -10 to 10, at 1-unit intervals). For the particular controller shown here, these 441 TN states yielded the same six sequences as had been found in the 12 TN states during evolution.

It is observed that the sequence patterns are arranged in clusters in the TN initial state space. First, the 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 6. Each of these two clusters is then further divided into topologically ordered 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. These smallest clusters neighbor each other and share the first two turns of their sequences in common. In other words, the turn sequences are hierarchically ordered into progressively smaller regions of the initial TN activity space as additional turns are added. As the complexity of the movement sequence increases, so too does the initial sensitivity to the TN activities.

This hierarchically ordered 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, ordered sequence regions remains. This trend was also found in Paine & Tani (2004), and is reminiscent of the fractal distribution of sequences mapped in the initial state space of Nishimoto & Tani (2004). Indeed, it would be interesting to see if fractal structure could be found in controllers branching out to larger numbers of goals.

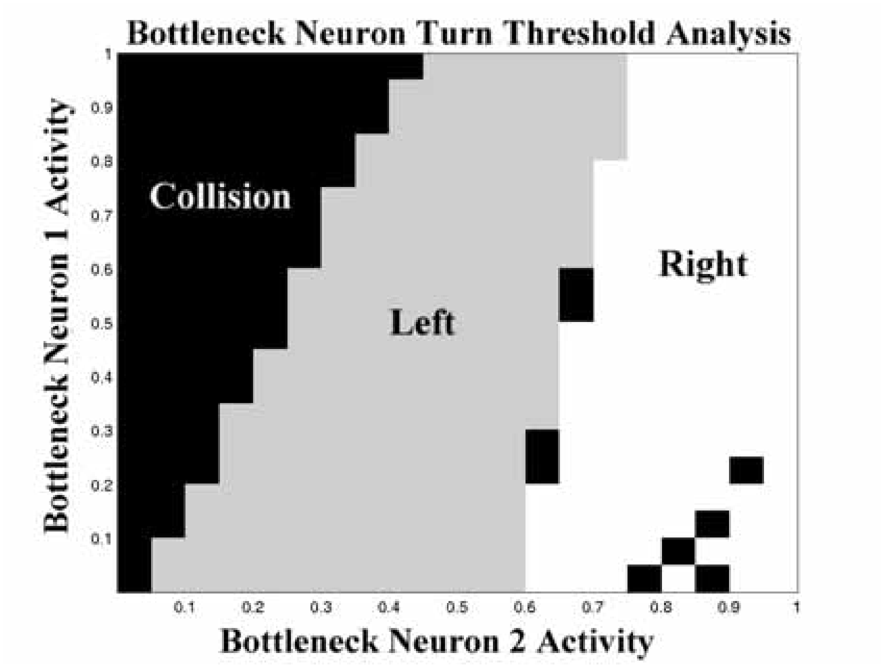

Figure 4 Phase space analysis of turn direction as a function of bottleneck neuron activities. BN activities were held fixed throughout a trial and the robot's behavior was observed. The robot began each trial at the home position of Figure 1. A left turn (grey) was recorded if the robot reached point P1 in Figure 1, and a right turn (white) if it reached P2. If the robot hit a wall of the maze, then a collision (black) was recorded.

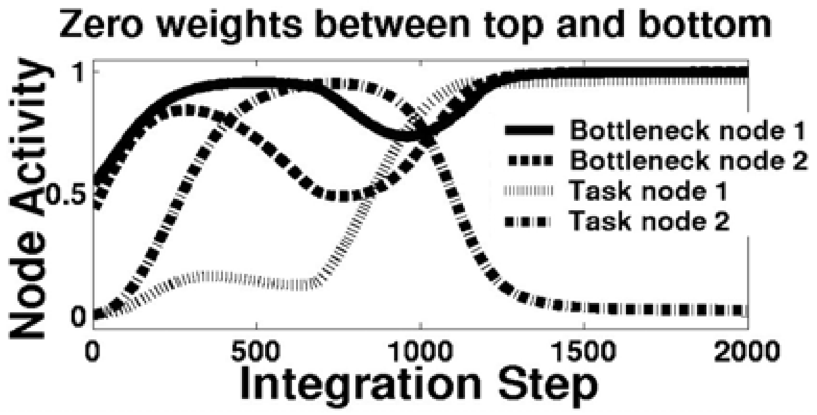

Figure 5 Disconnecting the upper level. Compare to the BN and TN activity profiles at the top of Figure 3 for a normal Right-Left-Right sequence (reaching goal 6 of Figure 1 at a time step of about 1250) using the bottleneck network. The very similar profiles here are generated when the weights between the BNs and the lower level are set to zero. The relative independence of the upper level from the lower indicates top-down open-loop control of turn sequences.

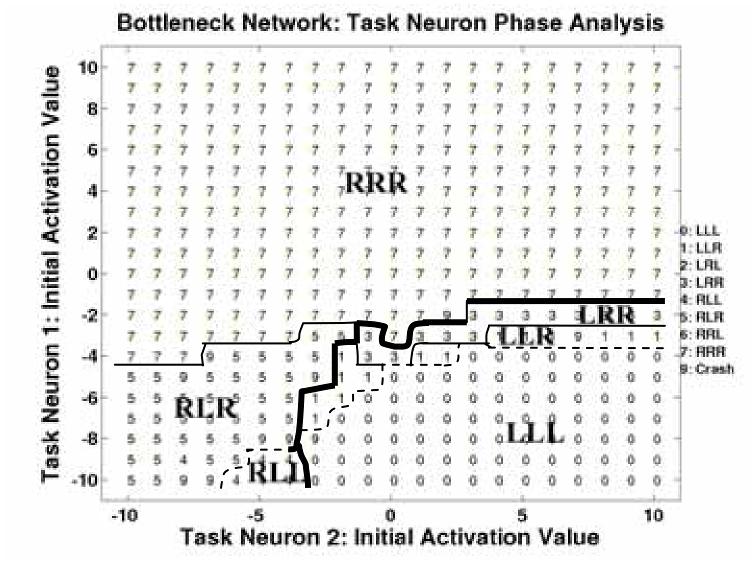

Figure 6 Phase space analysis of three-turn sequence generation as a function of task neuron initial activity, $\gamma_{M,2}^{task(0)}$. X and Y axes: Initial activities of task neurons 1 and 2, respectively. Plotted numbers correspond to sequences as in the legend on the right. Letters represent turns of the sequences (L=Left, R=Right). Six goals of the maze in Figure 1 were found.

3.2.3 Evolved Time Constants

Since obstacle avoidance requires rapid motor adjustments to avoid wall collisions, the motor output neurons (Figure 3, Middle; Figure 7) evolve small time constants (median value of τ=1 over 20 runs), allowing fast neuronal responses to changing sensor inputs. Since controlling turning behavior is a longer-duration task, the BNs evolve larger time constants for slower neuronal responses (median value of τ=41). Finally, the TNs, whose initial states determine future sequences, evolve the largest time constants (median value of τ=52) for turn sequence control, which is stable despite rapidly changing pre-synaptic signals, over the entire duration of the task (Figure 3, Top; Figure 7). The distributions of time constants in the bottleneck, 11-neuron, and 9-neuron fully connected networks are shown in Figure 7. The striking feature of this comparison is that TNs have significantly larger time constants than output neurons ($p<<0.02$, two-tailed, paired t-test), and tend to have the largest time constants of each network.

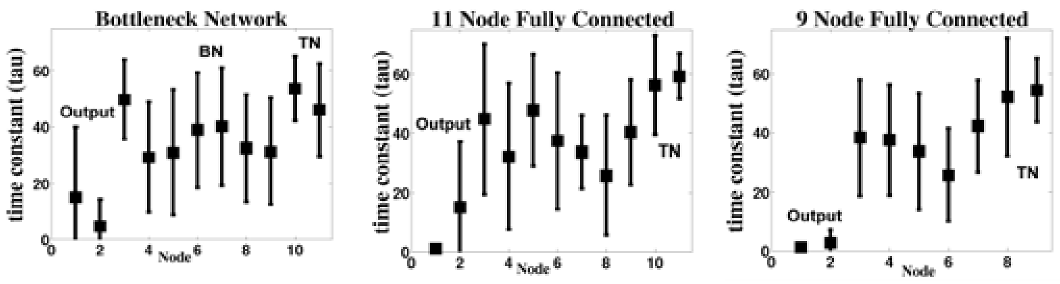

Figure 7 Time constant (τ) analysis of the bottleneck and fully connected networks. Squares mark means across runs. Error bars mark +/- one standard deviation. Note low output neuron (nodes 1-2) and high TN (last two nodes of each network) values. BN marks bottleneck neurons (nodes 6-7).

3.2.4 Analysis Conclusions

The analyses of Figures 4 through 6 suggest that levels of control with separate responsibility for local (lower level) or global (upper level) task components self-organize by utilizing the topological constraint on the synaptic connectivity of the bottleneck network. The fully connected networks share the bottleneck network's respective fast and slow, output and task neuron, temporal responses. Such local and global function-specific neuronal activation time-scales evolved in all three of the networks studied here. However, the lack of separation between the local and global processes causes them to interfere with one another and limit the numbers of sequences that can be stored in the fully connected networks. In effect, the connections from the rest of the network contribute additional noise to the long-term sequence representation, limiting the numbers of goals found. As discussed later, topological constraints have also been shown to both increase information storage and the robustness to noise in scale-free networks (Torres, Munoz, Marro, & Garrido 2004).

3.3 Implementation in a Real Khepera II Robot

In order to test the validity of the simulated controllers of section 2.1, they were transferred to a real Khepera II robot running in an eight-goal maze. Up to six goals were found by the bottleneck network controlling a real robot. The same controller had found seven goals in simulation. More accurate simulated robot and environmental modelling should further improve transfer from simulation to the real robot, as would evolution using the real robot itself.

4. Discussion

The following sections will discuss the generality of the presented ideas as compared and reconciled with prior related work. Future possible research extensions will also be discussed. Special focus will be given to the problems of local versus distributed representation of behavioral modules, issues of architectural levels and time constants, and the biological relevance of the work.

4.1 Local versus Distributed Representations

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. However, a modular system's ability to generate large numbers of different movement sequences through module combinations highlights the difficulty of representing multiple sequences in a fully distributed system. Perhaps a hybrid system using modules of distributed representations could harness the strengths of both approaches while avoiding their individual weaknesses.

4.2 Levels and Time Constants

An important feature of the bottleneck network is its emergent hierarchical functional organization. The lower level, with direct access to sensory inputs and control of motor output commands, automatically acquires a representation of movement primitives, such as collision avoidance and turning at intersections. These tasks require fast adjustments to compensate for rapidly changing sensory inputs. As a result, the output neurons evolve fast time constants. In contrast, the task neurons are farther removed from the rapidly changing sensory inputs and evolve slower neuronal activation time constants to control the longer-term task of sequence control. 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 divisions of bottleneck and task neuron activities of Figures 4 and 6, 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 has also been demonstrated in evolutionary robotics (Nolfi, 2002), in reinforcement learning (Precup & Sutton, 1997), and in neural network music composition (Todd, 1991; Mozer, 1992; Eck & Schmidhuber, 2002).

In Mozer’s (1992) model of learning variable-duration musical phrases, time constants of the network’s hidden units are manually tuned to optimize network performance. The hidden units are then able to respond at different rates that match the temporal regularities of the music. In contrast, the time constants governing neuronal response rates are genetically encoded in Nolfi’s (2002) work (as in the present model). In Nolfi’s experiments, a mobile robot evolves to self-localize in its environment by detecting regularities in its sensory-motor flow at different time scales. Neurons with variable activation rates and threshold activation functions evolve to detect events extending over variable lengths of time.

The current work uses the different temporal responses of TNs and output neurons to represent long and short-term components of a sensory-motor task. Similarly, in Tani's (2003) recurrent neural network with parametric bias, the top-down and bottom-up interaction mechanisms between the higher and the lower levels use careful tuning of the time constant parameters. Rapid switching of movement primitives occurs in the lower level, while they are combined sequentially over longer periods of time in the higher level.

A different approach to learning precise timing is used by Eck & Schmidhuber (2002) to learn the temporal structure of blues music and compose novel melodies with the same temporal characteristics. They make use of the Long Short-Term Memory (LSTM) recurrent neural network (RNN) (Gers, Schraudolph, & Schmidhuber, 2002). Instead of using time constants to vary neurons' temporal responses to inputs of different time scales, they store neural activation values and input-based error signals over time for future use in training the network. This network explicitly maintains activation and error signals, used to train the RNN's weights, over variable times in "memory cells" via trainable gates. The gates determine the type and longevity of stored information.

In contrast to the explicit encoding of neuronal response-time parameters of Nolfi (2002), Tani (2003), and the present work, or Gers et al.'s (2002) storage of training signals over time, Ziemke and Thieme (2002) use environmentally triggered synaptic weight changes to modulate sensory-motor mappings over time. Modulations occur after opening a gating, or "decision neuron", in response to environmental stimuli in multiple T-maze delayed response tasks. For example, seeing a light in a particular location can trigger a change in the sensory-motor mapping that switches the robot's turning direction at an intersection. As opposed to the top-down, open-loop control in the present work which maintains temporal modulation independently of external stimuli (Figure 5), Ziemke & Thieme’s controller is a bottom-up, stimulus-driven network with no internal representation of time. One could also imagine a system in which environmental cues help determine TN initial states, just as sound cues help determine the initial activities of sequence-determining supplementary motor area neurons in the primate brain (Tanji & Shima, 1994).

4.3 Automatic Sub-Division of Levels

As it is hard to predict whether particular subdivisions of a task into smaller sub-tasks or primitives will enhance or worsen task performance in reinforcement learning (McGovern, Sutton, & Fagg, 1997), it is also hard to predict where in a neural network an information bottleneck should occur. For example, placing it “too close” to the output or task neurons in the simple 11-neuron network used here might limit the ability of the network to influence, and be influenced by those critical components. Ideally, the algorithmic or architectural subdivision of a task to enhance performance should be automated. For example, McGovern & Barto (2001) discuss automating the process of “sub-goal” selection to break a large task into smaller units in reinforcement learning algorithms.

Since the weights of the fully connected network used here were “optimized” through a genetic algorithm, why didn’t the topological constraints of the bottleneck network automatically emerge from the fully connected network? In other words, if zeroing some of the weights of most nodes in the network, i.e., to form separate top and bottom levels, significantly enhances performance, shouldn’t BNs automatically be generated in the fully connected network through evolution? The fact that BNs were not automatically found in the present work is likely due to the inability of the GA to fully search the weight space of the network, especially since no explicit mechanism of severing connections was encoded in the genome short of coincidentally setting many weights of many nodes to precisely 0.0.

In contrast to the static architectures with evolved activation parameters used here, one may also evolve the architecture of a neural network to optimize it for a particular task. Early work in such evolution was done by Miller, Todd, & Hegde (1989), who represented network connections in a bit string genotype. More recently, Stanley & Mikkulainen (2004) explicitly encode connection "disabling" in their NeuroEvolution of Augmenting Topologies (NEAT) method, which starts with a simple network and gradually adds nodes and connections through evolution. They also test "simplifying coevolution", in which initially complex networks prune connections and nodes during evolution. They found that starting with a simple network and adding additional structural complexity only as "needed" through evolution yielded the best performance. However, even with such explicit pruning of connections, no emergent topological constraints were reported to subdivide the evolved networks into different functional levels. A network architecture with separately functioning groups of neurons which enhance task performance represents a small region of the large solution space searched by these evolutionary algorithms, contributing to the difficulty of evolving successful hierarchical architectures.

The Barabasi-Albert model (Albert & Barabasi, 2002) of growing networks with "scale-free topologies" deals extensively with the relation between topology and performance, as further highlighted in the work of Torres et al. (2004). Biological neural networks are not fully connected, "infinite range", networks with every neuron connected to every other neuron. Torres et al. investigated "scale-free networks", which attempt to mimic biological networks' finite connectivity by combining a majority of "boundary" nodes with small connectivity, with a minority of "hub" nodes having large connectivity. The hub nodes in scale- free networks bear a resemblance to the BNs in the current network. The bottleneck network was found to outperform the fully connected network in the present work by encoding turn-primitive control through the BNs. Similarly, Torres et al. found that scale-free neural networks tend to outperform Hopfield-like networks with the same number of synapses, distributed randomly over the network, by storing more relevant information into noise-resistant hubs.

4.4 Generality of the Model

How generalizable are the results presented here to other tasks and other kinds of controllers? As described below, elements of the TN and BN behavior representations, hierarchical task decomposition, and architectural constraints presented here have been researched in various forms. Past work ranges from the recurrent neural network with "parametric bias" models (Tani, 2003), reinforcement learning, to the scale-free Barabasi-Albert growing networks (Albert & Barabasi, 2002; Torres, Munoz, Marro, & Garrido 2004) discussed previously, which can be applied to a wide variety of sensory-motor control tasks.

Using a small number of parameters to control the dynamics of RNNs, generating diverse behavior sequences, has also been extensively studied in the "recurrent neural network with parametric bias" (RNNPB) models, e.g., Tani (2003). The current study of the BN network shares general concepts with the RNNPB, where the BN corresponds to the parametric bias (PB). The PB parameters of these models adaptively modulate the encoding of different self-organized behavior patterns in single RNNs, with nonlinear maps between the PB space and behavior patterns. The current work focuses on whether PB-like functions can emerge through a GA without the explicit programming of the RNNPB. It has been shown that they can, at least in the current navigation task. Since the RNNPB has been applied successfully to a range of tasks ranging from human-robot interaction (Ogata et al., 2004), robotic arm movement imitation (Ito

& Tani, 2004), to the acquisition of language semantics (Sugita & Tani, 2005), it is likely that the BN scheme can also be successful in various task domains.

Hierarchical learning in robot navigation has also been investigated using multiple levels of RNNs (Tani & Nolfi, 1999). However, hierarchical task decomposition is not limited to neural network models. The reinforcement learning (RL) community has made successful use of hierarchical 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, Sutton, & Fagg (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 of macro-actions 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.

The use of TN initial states to encode multiple movement sequences in a single RNN, while limited to simple 3-turn sequences in the current work, could be extended to encode more sequences and more complex actions by adding higher levels to modulate the TNs of the lower levels. For example, one could envision a level above the current bottleneck network's upper level to encode 6-turn sequences by modulating TN activities over longer periods of time. 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, Kitaguchi, & Hikosaka, 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.

4.5 Biological Relevance

The work presented here describes a model of behavioral sequence memory and generation in a single distributed network. Hierarchical structure in the neural network controller is shown to improve performance in a simulated mobile robot, in terms of both learning speed and the number of movement sequences learned. The model recalls in general terms the hierarchical organization of movements in the primate spinal cord, brainstem, and cortical regions. Simple motor primitives, consisting of turns and obstacle avoidance, are generated by lower-level structures, analogous to the spinal cord and brainstem. In animals, such primitives consist of various reflexes and rhythmic locomotor and scratching behaviors (Giszter et al., 1993; Ghez & Krakauer, 2000). Different types of dynamic structures self-organize in the lower and higher levels of the model network. As the brainstem helps integrate sensory information for the control of primitives, so the BNs in the model evolve to integrate robot sensor information and control turn-primitives. As shown in Figure 4, top-down modulation of the bottleneck neurons' interaction with the lower level allows behavioral switching of the primitives embedded in the lower level. A cat with a severed cervical spinal cord can still generate rhythmic locomotor movements, but its cerebral cortex cannot modulate those movements to reach a goal. "Cutting the spinal cord" of the model by severing the upper level similarly preserves the turn-primitives, but prevents the top-down modulation needed for goal-directed sequences. Utilizing the initial sensitivity characteristics of nonlinear dynamic systems (Fan, Yao, & Tong, 1996), a mapping of initial task neuron activity state 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. In contrast, interference between top-down and bottom-up processes appears to hamper the performance of a fully connected network controller that lacks hierarchical function segregation.

The current model shows how different turning behaviors can be triggered by modulating a small number of nodes (BNs) in a network. These nodes are similar to the “command neurons” that switch between different behaviors in evolved autonomous agents and in animals (Ruppin, 2002; Aharonov-Barki, Beker & Ruppin, 2001). Just as command neurons can modulate behaviors by triggering different activity patterns in a given artificial or biological neural network, so the BNs trigger different turning behaviors in the bottom level of the current model.

The organization of sequence generation in primates has been studied extensively, as in the studies of Tanji & Shima (1994) and Ninokura, Mushiake, & Tanji (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 activity states 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 bottleneck neurons in the present model.

However, although it might be tempting to make a direct mapping between the components of the simple networks presented here and specific brain structures, the complexity of the neuro-anatomical and physiological data makes such conclusions premature. For example, other studies (Crutcher & Alexander, 1990; Tanji & Kurata, 1982) have also reported SMA cell activities correlated with immediate movement execution. Further, Lu and Ashe (2005) found that cells in the primary motor cortex (M1), a major site of motor output via the corticospinal tract, can also encode sequences of upcoming movements. Thus, the experimental data suggest a complex system of motor sequence control including SMA, pre-SMA, as well as M1. Additional studies will be needed to further elucidate the contributions of these regions to sequential behaviors.

4.6 Summary

The research presented here addresses the question of whether hierarchical control structure is needed to complete sequential movement tasks. Tucci, Quinn, & Harvey (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 hierarchically structured systems implies nothing about the relative ease with which either system can be generated artificially or biologically. The experiments reported here show that a fully connected network controller had greater difficulty evolving to control a complex movement task than a hierarchically organized controller with comparable numbers of nodes or connections. Hierarchical organization was able to improve performance in a sequential movement task. One should note, however, that the sequential movement task examined here was easily divisible into simpler sub-tasks, or primitives. Further, the fixed bottleneck architecture used helped determine which control functions were adopted by which parts of the network. In this case, that particular grouping of control “primitives” was beneficial. However, it has also been shown that the use of groupings of primitives that are inappropriate for a particular task can retard learning of that task (McGovern et al., 1997). Future work will focus on how hierarchical control architectures may emerge automatically when they enhance performance of a particular task, but be suppressed when they might hinder it.

5. Conclusion

This work has demonstrated an approach to encoding goal-directed, behavioral sequences in a self-organized recurrent neural network controlling simulated and real mobile robots. It further examined how hierarchical segregation of control can emerge in a given architecture and enhance controller performance. Different types of dynamic structures self-organize in different levels of the network for the purpose of achieving complex navigation tasks. Neuronal response time constants are automatically generated relative to the task demands, with slower responses for longer-term movement sequencing and faster responses for short-term wheel commands and obstacle avoidance behavior. Top-down behavioral switching emerges through modulation of lower level activity via bottleneck neurons. In the higher level, a mapping of initial cell activity states to motor-primitive sequences self-organizes by utilizing the initial sensitivity characteristics of nonlinear dynamic systems. It was shown how those two levels of function evolved as relatively independent systems that interact only via the bottleneck, yielding more successful evolution than the examined fully connected networks. 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.

Acknowledgments

The authors wish to thank Yuuya Sugita and Ryunosuke Nishimoto for their technical assistance. Thanks are also due to Miki Sagara for her organizational skills and to Igor Paromtchik for his helpful comments. We also wish to express our gratitude to Dario Floreano and Jesper Blynel, at EPFL in Switzerland, for discussions and hospitality during the early stages of this work.

References

Aharonov-Barki, R., Beker, T., Ruppin, E. (2001). Emergence of memory-driven command neurons in evolved artificial agents. *Neural Computation*, **13**, 691-716.

Albert, R., Barabasi, A. (2002). Statistical mechanism of complex networks. *Reviews of Modern Physics*, **74**(1), 47–97.

Arbib, M. A. (1981). Perceptual structures and distributed motor control. In Brooks, V. B. (Ed.), *Handbook of Physiology*, Vol. II, (pp. 1449-1480). American Physiological Society.

Blynel, J. (2003). Evolving Reinforcement Learning-Like Abilities for Robots. In Tyrrell, A., Haddow, P.C., Torresen, J. (Eds.), *Evolvable Systems: From Biology to Hardware: 5th International Conference, ICES 2003,* Lecture Notes in Computer Science (pp. 320-331), Berlin: Springer Verlag.

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* (pp. 272-281). Cambridge, MA: MIT Press, Bradford Books.

Crutcher, M. D., Alexander, G.E., (1990). Movement-related neuronal activity selectively coding either direction or muscle pattern in 3 motor areas of the monkey. *Journal of Neurophysiology*, **64**(1), 151-163.

Cyberbotics, Ltd. http://www.cyberbotics.com, *Webots* Commercial Mobile Robot Simulation Software.

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.

Eck, D., Schmidhuber, J. (2002). Learning the long-term structure of the blues. In *Proceedings of the International Conference on Artificial Neural Networks*, Lecture Notes in Computer Science. Berlin: Springer Verlag.

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, Bradford Books.

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.

Gers, F. A., Schraudolph, N. N., Schmidhuber, J. (2002). Learning Precise Timing with LSTM Recurrent Networks. *Journal of Machine Learning Research*, **3**, 115-143.

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.

Ghez, C., Krakauer, J. (2000). The Organization of Movement. In Kandel, E. R., Schwartz, J. H., Jessell, T. M. (Eds.), *Principles of Neural Science,* Fourth Edition (pp. 653-673). New York: McGraw-Hill,.

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.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Ito, M., Tani, J. (2004). On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, **12**(2), 93-115.

Kelly, J. P. (1991). The Neural Basis of Perception and Movement. In Kandel, E. R., Schwartz, J. H., Jessell, T. M. (Eds.), *Principles of Neural Science,* Third Edition (pp. 283-295). Norwalk, Connecticut: Appleton and Lange.

Lipson, H., Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature*, **406**, 974-978.

Lu, X., Ashe, J. (2005). Anticipatory Activity in Primary Motor Cortex Codes Memorized Movement Sequences. *Neuron*, **45**, 967-973.

Miller, G. F., Todd, P. M., Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379-384). San Mateo, CA: Morgan Kaufmann.

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*, 13-17.

Miller, G. F., Todd, P. M., Hegde, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, J. D. (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379-384). San Mateo, CA: Morgan Kaufmann.

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

Mozer, M. C. (1992). Induction of Multiscale Temporal Structure. In Moody, J. E., Hanson, S. J., Lippmann, R. P. (Eds.), *Advances in neural information processing systems IV* (pp. 275-282). San Mateo, CA: Morgan Kaufmann.

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. (2004). Learning to generate combinatorial action sequences utilizing the initial sensitivity of deterministic dynamical systems. *Neural Networks*, **17**(7), 925-933.

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.

Ogata, T., Sugano, S., Tani, J. (2004). Open-end human robot interaction from the dynamical systems perspective: Mutual adaptation and incremental learning. *Lecture Notes in Artificial Intelligence*, **3029**, 435-444.

Paine, R. W., Tani, J. (2004). Motor primitive and sequence self-organization in a hierarchical recurrent neural network. *Neural Networks*, **17**, 1291-1309.

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* (pp. 1043-1049). Cambridge, MA: MIT Press.

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* (pp. 1050-1056). Cambridge, MA: MIT Press.

Ruppin, E. (2002). Evolutionary autonomous agents: a neuroscience perspective. *Nature Reviews Neuroscience*, **3**, 132-141.

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

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

Stanley, K. O., Mikkulainen, R. (2004). Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research*, **21**, 63-100.

Sugita, Y., Tani, J. (2005). Learning Semantic Combinatoriality from the Interaction between Linguistic and Behavioral Processes. *Adaptive Behavior*, **13**, 33-52.

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., Kurata, K. (1982). Comparison of movement-related activity in 2 cortical motor areas of primates. *Journal of Neurophysiology*, **48**(3), 633-653.

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

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

Todd, P. M. (1991). A connectionist approach to algorithmic composition. In Todd, P. M., and Loy, D. G. (Eds.), *Music and connectionism* (pp. 173-193). Cambridge, MA: MIT Press.

Todd, P. M., Werner, G. M. (1999). Frankensteinian approaches to evolutionary music composition. In Griffith, N., Todd, P. M. (Eds.), *Musical networks: Parallel distributed perception and performance*. Cambridge, MA: MIT Press, Bradford Books.

Torres, J. J., Munoz, M. A., Marro, J, Garrido, P. L. (2004). Influence of topology on the performance of a neural network. *Neurocomputing*, **58-60**, 229-234.

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.

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.

Author Biographies

Rainer Paine received MD and PhD degrees from Boston University, USA, where he worked with Stephen Grossberg on the neural mechanisms of handwriting movement control. Following an internship studying hippocampal motor sequence representation with Yoko Yamaguchi at RIKEN, Japan, Rainer joined Jun Tani's lab at RIKEN as a postdoctoral researcher. Rainer's interests include sequential movement control, embodied cognition, and spinal oscillator circuits. Email: *rpainemail-riken@yahoo.com*, Post: 78 West Millbury Rd., Sutton, MA 01590, USA.

Jun Tani received the B.S. degree in mechanical engineering from Waseda University, M.S. degree in electrical engineering from Michigan University and Dr. Eng. from Sophia University. He is currently a team leader in Lab. For Behavior and Dynamic Cognition, Brain Science Institute, RIKEN in Tokyo, Japan. He is interested in embodied cognition, complex adaptive systems and brain science. Email: *tani@brain.riken.jp*, Post: RIKEN Brain Science Institute, Laboratory for Behavior and Dynamic Cognition, 2-1 Hirosawa, Wako-shi, Saitama 351-0198, Japan.