I, Anand Krishnan Balasubramanian, hereby submit this original work as part of the requirements for the degree of Master of Science in Electrical Engineering.

It is entitled:
A Computational Model for Building Modular Animals: Design and Configuration of the Decision Network

Student's name: Anand Krishnan Balasubramanian

This work and its defense approved by:

Committee chair: Ali Mirai, PhD
Committee member: Raj Bhatnagar, PhD
Committee member: Carla Purdy, PhD
A Computational Model for Building Modular Animals: Design and Configuration of the Decision Network

A thesis submitted to the

Graduate School of University of Cincinnati

In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In the School of Electronic & Computing Systems

Of the College of Engineering and Applied Science

December 2012

by

Anand Krishnan Balasubramanian

B.E. in Electronics & Instrumentation Engineering

Anna University, Chennai, India

Thesis Advisor and Committee Chair: Dr. Ali A. Minai
Abstract

The factors behind the ability of animals to perform complex tasks in a dynamic environment are a fascinating area of research. In order to effectively build adaptable and robust artificial intelligent systems, it is important to fully understand how complex behavior emerges through self-organization and embodiment in the actual animals. Biologically-inspired modeling has emerged as the most effective way of building artificial animals capable of complex behaviors in the recent times, and is the primary research focus in this thesis.

The following hypotheses are the fundamental basis of this work: 1) Artificial intelligent systems built using developmental and/or evolutionary learning techniques are more efficient in performing complex tasks in dynamic environment; 2) Embodiment plays a significant role in emergence of intelligent behavior in complex animals; 3) Modularity is one of the major factors that favors the ‘evolvability’ of systems; 4) Complex animals arise through systematic reconfiguration and duplication, composition, reorganization and reallocation of modules in simpler modular animals through evolution.

This thesis extends a previously developed framework for configuring and simulating modular animals [1] to add a control structure that receives sensory input and generates neural control signals for the motor system. The framework also allows the possibility of simulating developmental processes to systematically build complex animals. The ultimate objective is to
develop an environment for evolving animals that would be capable of highly complex and intelligent behavior.
Acknowledgements

I would like to sincerely thank my advisor Dr Ali Minai whose knowledge and expertise helped me through this thesis. I would be always grateful for his guidance, support and patience, without which this would never have materialized. I would also like to thank Dr Raj Bhatnagar and Dr Carla Purdy for agreeing to make time amidst their busy schedule to be on my thesis defense committee and review my work.

I am extremely grateful to my parents, my sister, brother in law and so many other family members for always being there for me and providing their unconditional love and support. This thesis is dedicated to them.

I am also extremely thankful to my lab mates for always willing to help me with the technical challenges I faced during my research.

Most importantly I would like to thank all my friends, colleagues and especially my roommates for all the wonderful memories.
# Table of Contents

1 Introduction .................................................................................................................. 1

1.1 Research objective ................................................................................................. 4

2 Background and Literature Review .............................................................................. 7

2.1 Developmental and Evolutionary Methods .............................................................. 7

2.2 Role of Embodiment in Control .............................................................................. 10

3 Model .......................................................................................................................... 13

3.1 Model Extensions .................................................................................................... 16

3.2 Sensory Map ........................................................................................................... 18

3.3 Action Map ............................................................................................................. 21

3.4 Selector Network .................................................................................................... 23

3.5 Motor System Network .......................................................................................... 28

3.6 GUI Guide ............................................................................................................... 29

3.6.1 Step 1: Initialization ......................................................................................... 30

3.6.2 Step 2: Module Specification ............................................................................ 31

3.6.3 Step 3: Connectivity Specification ................................................................. 31

3.6.4 Step 4: Motor System to Spinal Weight Specification ..................................... 32
3.6.5  Step 5: Decision Network Configuration .................................................. 32
3.6.6  Step 6: Simulation Time Specification ....................................................... 36
3.6.7  Step 7: Spinal Segment Maximum Angle Specification .............................. 36
3.6.8  Step 8: Run Simulation .............................................................................. 37
3.6.9  Data Analysis .............................................................................................. 37
4   Results and Discussion .................................................................................... 38
  4.1  Simulation Setup ........................................................................................... 38
  4.2  Developing Complex Animals ....................................................................... 39
    4.2.1  Increasing the Rotational Flexibility of Each Segment ............................. 39
    4.2.2  Increasing the Temporal Duration for Input Sensory States ...................... 45
    4.2.3  Increasing the Number of Segments ....................................................... 51
  4.3  Significance of Decision Network Parameters in Animal Behavior ............... 61
  4.4  Significance of Attractors ............................................................................ 62
5   Conclusion and Future Work ........................................................................... 64
6   References ....................................................................................................... 67
List of Figures

Figure 3-1: Model architecture ........................................................................................................ 14
Figure 3-2: Spinal Module ............................................................................................................... 15
Figure 3-3: Decision Network Architecture .................................................................................... 20
Figure 3-4: Activity Bump in the Sensory Map ............................................................................... 21
Figure 3-5: State-Action Mapping ................................................................................................. 23
Figure 3-6: Mapping between neurons of Action Map and Selector Network ......................... 27
Figure 3-7: Hidden layer activities of two attractor patterns which differ by just one bit in the output layer .................................................................................................................. 28
Figure 3-8: Spiking pattern in the Cortical Network for all possible Selector Outputs .......... 29
Figure 3-9: GUI Main Interface .................................................................................................... 30
Figure 3-10: Decision Network Interface ...................................................................................... 32
Figure 4-1: Time lapse screenshots at different simulation instances for an 8 segmented animal configured to oscillate in thrashing mode ......................................................................................... 40
Figure 4-2: Force profiles for individual segments of an 8 segmented animal configured to oscillate in thrashing mode ........................................................................................................... 41
Figure 4-3: Angle profiles for individual segments of an 8 segmented animal configured to oscillate in thrashing mode ...................................................................................................... 42
Figure 4-4: Time lapse screenshots at different simulation instances for the 8 segmented animal configured for thrashing mode with increased angular flexibility ............................. 43
Figure 4-5: Force profiles for individual segments of the 8 segmented animal configured for *thrashing* mode with increased angular flexibility ................................................................. 44

Figure 4-6: Angle profiles for the individual segments of the 8 segmented animal configured for *thrashing* mode with increased angular flexibility ........................................... 45

Figure 4-7: Time lapse screenshots at different simulation instances for a 12 segmented animal configured to oscillate in *thrashing* mode................................................................. 46

Figure 4-8: Force profiles for individual segments of a 12 segmented animal configured to oscillate in *thrashing* mode ......................................................................................... 47

Figure 4-9: Angle profiles for individual segments of a 12 segmented animal configured to oscillate in *thrashing* mode ......................................................................................... 48

Figure 4-10: Time lapse screenshots at different simulation instances for the 12 segmented animal configured for *thrashing* mode with increased sensory state temporal distribution 49

Figure 4-11: Force profiles for individual segments of the 12 segmented animal configured for *thrashing* mode with increased sensory state temporal distribution...................... 50

Figure 4-12: Angle profiles for individual segments of the 12 segmented animal configured for *thrashing* mode with increased sensory state temporal distribution...................... 51

Figure 4-13: Time lapse screenshots at different simulation instances for an 8 segmented animal configured to oscillate in *spiral* mode ................................................................. 53

Figure 4-14: Force profiles for individual segments of an 8 segmented animal configured to oscillate in *spiral* mode ............................................................................................ 54
Figure 4-15: Angle profiles for individual segments of an 8 segmented animal configured to oscillate in spiral mode ................................................................. 55

Figure 4-16: Time lapse screenshots at different simulation instances for a 12 segmented animal configured to oscillate in spiral mode ................................................................. 56

Figure 4-17: Force profiles for individual segments of a 12 segmented animal configured to oscillate in spiral mode .......................................................................................... 57

Figure 4-18: Angle profiles for individual segments of a 12 segmented animal configured to oscillate in spiral mode .......................................................................................... 58

Figure 4-19: Time Lapse screenshots at different simulation instances for a 16 segmented animal configured to oscillate in spiral mode ................................................................. 59

Figure 4-20: Force profiles for individual segments of a 16 segmented animal configured to oscillate in spiral mode .......................................................................................... 60

Figure 4-21: Angle profiles for individual segments of a 16 segmented animal configured to oscillate in spiral mode .......................................................................................... 61

Figure 4-22: Effect of noise on true attractor convergence.................................................. 63
1 Introduction

Even very simple animals have the ability to perform with ease tasks that would be considered computationally extremely complex for an artificial system. This can be observed specifically in cases where the task involves coordination and control of motor systems, such as locomotion, pursuit, evasion, etc. Not only do animals perform these tasks with ease, they also adapt them to constantly changing complex environments, and do so in real time with minimal error. The physical basis of these remarkable capabilities is an active area of research and the major motivation for the work in this thesis.

It is critical to remember that the animals capable of complex motor control have gone through millions of years of evolution: The physical framework underlying complex behavior has emerged over millions of years and is, to a significant degree, encoded genetically, i.e., the brains and bodies of animals are capable of complex behavior because of their genetically encoded structures and processes. Animals (and before them other organisms) started off with relatively simple capabilities and have only gradually developed more complex ones through the systematic complexification of their physical forms and by building more complex processes on the infrastructure of simpler, more primitive ones. This occurs both during evolution at the level of species and during development at the level of single organisms. The animals that we see around us did not arise ex nihilo in their fully complex forms, nor were the complex capabilities of perception, cognition and behavior they possess too learned from scratch. This view contrasts
starkly with the traditional engineering approach that divides the building of intelligent systems
into two components – physical design and learning. Thus, complex robots are first designed and
built purely for their physical attributes and then asked to learn complicated tasks with their
complicated bodies, creating an impossible problem. In contrast, animals begin life with
relatively simple bodies and learn simple tasks within a limited repertoire of possibilities. Then,
in time, as their bodies (including brains) become more complex, they learn more complex tasks
using the earlier ones as constraints and as building blocks. A similar bootstrapping process
occurs during evolution from simpler to more complex animals. It has been shown that these
evolutionary and developmental approaches to learning can be much more efficient than
traditional ones for building intelligent systems [2]. However, a useful framework for doing this
requires deep understanding of the principles underlying productive complexification in
biological systems.

Recent research in systems biology and evolutionary developmental biology has shown that the
fundamental principle underlying successful complexification is modularity: The encapsulation
of generically useful functions into unitary functional or physical entities that can be used as
building blocks for more complex forms [3] [4]. It can be shown that modularity enables both
richness and robustness simultaneously, which is the main “trick” for successful increase in
complexity [5] [6]. Most of the growth and change in the system comes from the composition
and ramification of existing modules whose internal functionality is already “guaranteed” by
previous validation, so that each level of complexity has a certain foundational level of reliability.

Another key part of this perspective is that all control processes in the complexifying animals are emergent rather than top-down. They are the consequence of the complexification and co-evolve with it – again in contrast with traditional engineering, where complex control algorithms are developed top-down when more complex machines are built.

The research presented in this thesis is motivated by the following fundamental hypotheses:

- All animals are hierarchical, multi-scale networks of physical and functional modules.
- Perception, cognition, memory, control and action in animals are emergent coordination modes of the natural dynamics of these networks in the context of their environments.
- New, more complex animal forms arise primarily through the proliferation, recomposition, ramification and hierarchical encapsulation of modules evolved in simpler animals, with very occasional emergence of truly novel modules into the repertoire. Evolutionary selection acts on these forms, resulting in the continual addition of new functional modules into the mix available for future evolution.

This view of living systems has led to the idea that they are autopoeietic – self-creating [7].
The work in this thesis does not focus on addressing these issues directly, but is part of a long-term effort to build a platform where these issues can be explored.

1.1 Research objective

In this thesis, the main objectives are the following.

- Add a control framework to the existing simulator for building modular animals.
- Extend the GUI to include a facility to customize the control structure.
- Demonstrate how the system can be used to systematically build and simulate more complex animals.

The original simulator included only a multi-segmented body and a neural control architecture corresponding roughly to the spinal cord and the motor cortex. The new work adds higher level control structures that can link sensory inputs to the motor system, and extends the graphical user interface (GUI) for the system to allow simulation of significantly more complex animals.

The new control, structure, termed the Decision Network, has the following functions:

- Receive the current sensory input from the environment and map it onto a sensory state of the system.
- Decide on an appropriate action for the current sensory state.
• Generate neural control signals inputs to the motor network/spinal system of the animal that results in the generation of the selected action.

Typically, the sensory input would involve both - input from the external environment and from the animal’s own body (e.g., posture information). While the new system provides the neural architecture for representing sensory stimuli and mapping them to control signals, it does not yet include the learning processes that would be needed to configure the sensory map or learn appropriate actions. These will be included in future development of the system.

The key features of the implemented system are modularity and emergent control. The user can configure animals of arbitrary size and modular multiplicity (within limits), specify sensory states and define how they map to desired actions. The decision network uses a recurrent competitive attractor network architecture that provides both stability and robustness.

The user is provided the option of customizing the following decision network parameters from the GUI.

• The number and form of embedded attractor patterns, each corresponding to a motor state of the animal.
• The size of the sensory map and the number of sensory states.
• The state- action pairs, i.e., the mapping from sensory states to attractor patterns and, therefore, actions.
For simulation purposes, the user also can specify the number of Input signals that would be sent as input during the course of the simulation time, and also the sensory states that need to be activated for each of these Input signals.

Chapter 3 explains the system in more detail. It also covers the steps involved in the configuration of simulations, and how these configurations are mapped to an animal network. Chapter 4 focuses on how complex animals can be specified systematically using the GUI and how changing some parameters while retaining the others can lead to animals with different behaviors. Chapter 5 presents conclusions and suggestions for future work.
2 Background and Literature Review

It is becoming increasingly clear that truly intelligent artificial systems require more than just complex algorithms. Rather, they must embody principles that underlie living systems that enable these systems to generate intelligent behavior. As discussed in Chapter 1, two of the central principles involved are modularity and bottom-up self-organization. While these principles are important in all complex adaptive systems, in living systems they are essential to explaining how animals come to have the forms they do, and how these forms enable their behavior. In this thesis, the main objective is to provide a framework in which an experimenter can systematically build modular, self-organizing animals with emergent behaviors, and explore these behaviors by varying control signals and the forms. This chapter reviews some previous work and background material related to this.

2.1 Developmental and Evolutionary Methods

There has recently been a lot of interest in using the principles of developmental and/or evolutionary learning techniques to build artificial systems capable of performing complex tasks in dynamic environments. Some success has been achieved in evolving complex animals systematically from simpler ones. Dellaert and Beer were able to evolve complete, though simple, animals capable of autonomously following a path. They also demonstrated that an agent that underwent the developmental process is more robust in adapting to dynamic changes in its environment [8] [9] [10]. Beer et al. used a “biologically inspired” approach to build a
distributed neural control system for the movement of a robotic hexapod that helped it move over complex and irregular terrains using minimal computational resources [11].

Sims [12] developed a framework in which virtual animals evolved from primitive forms into different complex creatures capable of specific functions like locomotion or jumping. In the same work, he also showed that applying an evolutionary constraint coupled with optimization through selection led to more interesting and complex behaviors in animals [13]. Cangelosi et al. [14] have suggested that the paired learning of knowledge and action benefits greatly from a developmental approach, since the acquisition of skills and their integration in this case happens as a concurrent gradual process [15]. A very similar claim has also been made by Lee et al. [16], who have stressed on the importance of an artificial system starting off with “constraints” at an infant stage in making the system more adaptive and robust as it becomes more complex. Brooks et al. [17] have built a humanoid robot with twenty one degrees of freedom and have been successful in getting it to exhibit sophisticated behaviors by gradually increasing the complexity of the system, starting from an infant stage. Pfeifer and Gomez have shown that applying developmental learning principles to their complex robot required minimal control signals in order to generate a vast set of different behavioral patterns [18].

In a recent survey by Lungarella et al. [19], they concluded that instead of engineering “intelligence” into an artificial system, it should just be equipped with a basic configuration from which the system by itself would develop and simultaneously learn such that the intelligence can
be observed as it grows. Similarly Gomez and Eggenberger Hotz [20] demonstrated that an adaptive robotic system performed better and was much more adaptable when it was given the opportunity to explore all the action domains during its testing in real environment, as opposed to simulation testing where most of its available actions were preloaded. Federici built a system of spiking neural networks and showed that when the system was subjected to learning through a developmental process, it discovered the globally optimum control mechanism and became naturally fault tolerant [21]. The latter was specifically tested at every stage of development by intentionally inducing noise into the system and it was noticed that the system’s fault repair mechanism also improved as it grew more complex.

Metta et al. [22] have shown that when two systems, one built by the traditional engineering method and the other using simulated artificial development, were presented the same goal, the latter was able to reach the goal with much better control than the former even without much knowledge of the kinematic parameters of the system. This once again proves that systems which undergo a developmental learning process will be able to adapt to a dynamic environment very easily even without having a complete knowledge of it.

Elman [23] has proven that “starting small” (i.e.) with developmental restrictions is not really a drawback for a system, but in fact it helps it acquire all the skills needed for conquering complex domains. The reason for this is that having limited degrees of freedom while starting helps the system explore all possible action domains that could be achieved with those constraints. Smith
and Gasser [24] have suggested that people developing artificial intelligent systems must look more into how babies acquire their skills and apply those principles in their work.

2.2 Role of Embodiment in Control

Embodiment is the idea that perception, cognition, action and intelligence are not just attributes of the brain but emerge only when the brain comes with a body capable to sensing and action. The attributes ascribed to the brain are said to be “grounded” in the body which, in turn, is embedded in its environment. It is now widely believed that embodiment is the key to building truly intelligent systems [25]. In particular, embodiment helps greatly in mitigating the complexity of control – the so-called degrees of freedom problem [26] – by exploiting pre-coordinated behaviors and attributes latent in the embodiment. For example, passive dynamic walkers can produce complex locomotor behaviors simply under the influence of gravity [27] [28] or wind power [29].

In a review of embodied approaches by Pfeifer, Lungarella and Iida [30], it was observed that using standard control engineering principles with a generic, centralized control leads to systems do not adapt well in the real world complexities. In contrast, an embodied and specific approach seems to work much better in practice. Lungarella and Sporns [31] also found that information storage and processing in sensorimotor networks is distributed across various components, thus making it dependent on the embodiment and this information acquisition, storage and retrieval are also concurrently affected by learning.
It has been shown in many instances that, with properly configured embodiment and control systems, robots can perform a large number of complex tasks with the manipulation of relatively few control variables, relying on the coordinated degrees of freedom latent in the embodiment to generate the action. The best example illustrating this claim is the “Salamander” model created by Ijspeert et al. [32] in which they were able to toggle the locomotion mode of the model between ‘walking’ and ‘swimming’ merely by changing the oscillation frequency of the driving signal. Similarly, Iida et al. [33] were also able to generate several interesting locomotion behaviors in a simple four legged robot by varying just one control parameter. Schmitz et al. [34] were able to verify that a physical robot was able to keep a steady position and run on a treadmill with varying speeds with just one control parameter being manipulated, thus confirming its adaptive and robust nature. Beer et al. [35] were able to produce different gaits on a hexapod robot by manipulating the level of activity in a single neuron. All these cases clearly prove that systems learn through the continuous interactions of its control network, its body and the environment, and in order for the learning to have maximum effect, it is imperative that the central control system and the physical structure complement each other. All the cases listed above show that when that is done properly, a wide range of intelligent behaviors can be obtained using minimal control.

An important aspect of the embodied approach is that it takes an integrated view of perception, cognition and action rather than placing sharp boundaries between them. It is the animal as a whole that interacts with and responds to its environment. Cham and Cutkosky [36] found that
even a small change in the timing of activation and deactivation of the actuators had a significant effect on the dynamics of limb movements, indicating that sensorimotor information could be spatially and temporally encoded across all components of the system. Paine and Tani [37] have shown that having a hierarchical, modularized control structure with minimal interference between various individual components leads to greatly improved system performance in terms of adaptability. Sporns and Lungarella [38] have shown that, for an embodied system, intelligent behavior depends largely on the coupling dynamics between the system’s environment, its physical modules and its central nervous system.

It has also been shown that modularity is one of the most important factors in building ‘evolvability’ in complex systems [39]. The main explanation behind why modularity favors evolvability is that the individual characters can evolve independently and become more adaptable with little or no interference from or dependence on others [40] [41]. Hartwell et al. [42] have shown the importance of modularity not just in the survival of an individual, but also in the process of selection in terms of evolving gene pools. It can be seen that modular approach in developmental learning helps in emergence of more complex behaviors as the system grows.

The next chapter describes a simulation environment in which animals with specific modular embodiment can be built and their emergent behaviors observed. In the current version, development and evolution can only be simulated manually, but the system provides a framework for automating these as well.
3 Model

As explained in the previous chapters, the main objective of this thesis is to extend the framework developed by Salunke [1] to include a decision network that operates as the high level control structure for simulated modular animals. A brief summary of that model is given first as reference for the subsequent description of the additions.

Figure 3-1 shows the high level architecture of the overall model animal. The animal may have single or multiple effectors, corresponding to spinal segments, as specified by the user. As in real vertebrates, the spinal segments are physically linked and controlled by muscle movements. The first segment is fixed at one end, while the subsequent segments are linked to the previous ones. Each segment has a dedicated spinal neural network and has two muscles on either side which determine that segments’ rotational angle. The underlying spinal circuits receive their input from a system called the motor system, which corresponds roughly to the motor cortex and the brainstem in higher and lower vertebrates, respectively. These neural networks have been implemented using the spiking neuron models developed by Izhikevich [43] [44].

The motor system, in turn, receives input from a higher level system termed the decision network system – corresponding to cognitive control component of the nervous system. The decision network gets processed input from the environment via the sensory system. The sensory system and the decision system were not implemented in the simulator developed by Salunke [1].
The neural network for a single spinal segment is shown in Figure 3-2 [1]. The cortical output from the motor system is received as input in the Integrative Stage and is converted into a Central Pattern Generator (CPG). The pattern generated by the CPG is then converted into force/angle by the effector system. Another important component is the stretch receptor, which provides feedback on the amount of muscle stretch.
The cortical network consists of several modules, each of which is a recurrent neural network with several embedded attractor patterns. Each module connects to one or more spinal segments, and each attractor pattern in the module, when activated, provides input of a specific strength to
the integrative stage of the targeted spinal segment. The motor system modules can potentially connect to each other as well. For all the equations of the part implemented by Salunke – the spinal segments and the motor system – the reader is referred to [1].

3.1 Model Extensions

The research in this thesis adds a neural network implementation of the decision system and a neural, albeit simplistic implementation of the sensory system. Together, the functions provided by these systems are:

- Map the current sensory input that is received from the environment and the animal’s own body into a sensory state.
- Decide on the right action for that state.
- Generate the corresponding neural signals that are to be sent to the motor system, where the signals are converted to input for the spinal segments.

The sensory system is implemented simply as a two-dimensional map of neurons called the sensory map and used for the purpose of storing all the sensory states that can be perceived by the animal. When the sensory network receives an input from the environment, it is mapped to the nearest known/stored sensory state through a locally excitatory, globally inhibitory map dynamics. The encoded sensory state then becomes an input to the decision network (i.e., the decision system), which comprises an Action Map and a selector network as described below.
The input from the current sensory state is received by the switcher, and is mapped to a decision state represented by a single winning neuron in the competitive switcher network. This, in turn, triggers an activity pattern in the selector network, which is then stabilized by attractor dynamics. The selector network has a set of distinct activity patterns, each of which corresponds to a specific combination of states for each of the motor system modules, thus making it a distinct motor state of the animal. The neurons of the Action Map have a one to one mapping with the patterns of the selector network. Thus, the winning unit of the Action Map selects the pattern associated with that unit. Based on the output of the selector network, the corresponding attractors are triggered in the motor modules, and then sent as input to the spinal CPG network, which generates the appropriate spiking patterns. These spiking patterns are then converted to force/angle values by the effector.

While the current framework does not implement the sensory network, it provides the option of manually specifying the number of sensory states that can be perceived by the animal for simulation purposes. This gives the decision network information about the number of sensory states, making it possible to configure the state-action mapping. When the sensory network is implemented, the framework can be very easily extended to accommodate the learning algorithms that automatically map the inputs to the closest matching sensory states in the map.

The graphical user interface (GUI) has also been changed so as to provide the user the option of customizing and configuring the decision network. Specifically the part in the GUI where the
user had to choose from a set of pre-formulated patterns to be sent as input to the cortical network has now been replaced with a new component titled “Configure Decision Network”. This additional component provides the user the option for customizing the following decision network parameters.

- The number of attractor patterns in the selector network, and the structure of each pattern.
- The size of the sensory map, and the number of sensory states to be stored in the map.
- The mapping for each of the sensory states to the desired attractor pattern.

It is to be noted that in the subsequent versions when learning is implemented, the third option would not be needed anymore as the animal would have learned by itself to do the state-action mapping.

Figure 3-3 gives a high level view of the architecture of the decision network.

3.2 Sensory Map

The sensory map is a two dimensional grid of neurons with the dimension \( k \times k \) where \( k \) is specified by the user as the size of map. Each of the sensory states that can be perceived by the animal corresponds to a region on the map, and the neurons of that region would ideally be dedicated specifically for that state. This is achieved by selecting \( M \) distinct neurons which are sufficiently well separated from each other in terms of Euclidean distance. Here \( M \) corresponds to the number of sensory states. Thus, there would be one such selected neural unit for each of
the states, and that neuron is termed the state’s centroid. This centroid along with its neighboring units forms the representation for that particular state, which takes the form of an activity bump. When a particular state is chosen as input, the neurons in the bump for that state are activated to a degree that depends monotonically on their Euclidean distance from the state’s centroid. Typically the neural activity is distributed normally with the peak at the central unit and is determined by the formula:

\[
x_i^j = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(i-i^*)^2}{2\sigma^2}}
\]

Where
\( x_i^j \) = excitation for the neuron \( i \) when state \( j \) is selected.
\( \sigma \) = standard deviation, typically \( \sigma = 1 \).
\( i^* \) = centroid unit for the state \( j \).

Figure 3-4 shows the activity bump corresponding to a particular sensory state, when that state is chosen as the sensory input.
Figure 3-3: Decision Network Architecture
3.3 Action Map

The Action Map has \( m \) neurons, where \( m \) is the number of attractor patterns in the selector network (specified by the user). Each neuron in the system is dedicated to exactly one of the patterns in the network, and the excitation of these neurons happens on a mutually exclusive or on a ‘winner take all’ (WTA) basis. This means that any point of time, only one of the switcher neurons would be excited or “on”, while the rest would have to be “off”. The neuron that gets the
most excitation from the sensory map is considered the winner and it is set to “on”, with rest set to “off”. The pattern associated with that neuron then becomes the *action pattern* for that state.

The main advantage of having the Action Map in the architecture is that it makes good use of modularity. Considering the scenario when the same pattern is to be chosen for more than one sensory state, the presence of the Action Map in between the sensory map and the selector network makes the entire process much easier than when they are directly coupled. This would be especially beneficial where the number of sensory states and the number of patterns is large.

Figure 3-5 shows how different neurons in the action map are mapped to different sensory states in the sensory map.
3.4 Selector Network

Currently the number of possible attractor motor states for a single motor module is seven. The number of module neurons that send input to the spinal network is 140, where each motor state has a set of 20 dedicated neurons that get activated for that motor state. The framework also provides the scope for developers to encode more motor states if needed in the future. As the animal becomes more complex in terms of number of physical segments, the state space for the action to be taken for each segment can also increase. Thus, combining the available states for each segment and the total number of segments in the system, it can be seen that the total state
space would become extremely large. Not only would it be a very complex task to accommodate all of these action states in the ‘state-action’ mapping, there is also a much bigger challenge of redundancy. Given $S$ modules and $m$ attractors in each module, the motor system can produce $m^S$ different motor states. Thus 5 modules with 7 states each encode 16807 motor states. However, many of these either produce useless movements or end up placing the animal in the same positional state as many others. By allowing the animal to selectively stabilize a few useful and distinct states and minimizing the chances of recalling the rest, the selector network reduces the problems posed by the combinatorial explosion systematically without precluding its beneficial uses.

Essentially, the selector network provides the users (and thus, the animals) the option of choosing their own action space. Users decide on a set of motor state combinations for all spinal modules, which they think would produce distinct useful behavior in the animal through positional movement. These combinations are stored as attractors in the selector network. So when the decision network decides on a particular combination of modules as the right action for a given sensory state, the selector network associatively recalls the pattern that has the closest match with the desired combination and triggers the corresponding action.

The neural units of selector network are organized into $S$ rows, each of which has $m$ output neurons and $m_h$ hidden neurons, giving a total of $n = m + m_h$ neural units per row and $Sn$ in all. All neurons in the network are laterally connected to the others in all rows. Each stored pattern in
the network must have exactly one output unit out of $m$ activated in each row, indicating which attractor the module corresponding to that row should activate. It should be recalled that one of the attractor in each motor module encodes the “no action” decision, while the rest indicate stimulus of varying degree to the two muscles for the spinal segment to which the module projects. Thus, the dynamics of the selector network is row-wise competitive while the connections between rows encode the desired combination of active units to be included in the attractor. The number of hidden neurons is chosen as a function of the number of parameters, and each attractor is given a randomly chosen activity pattern over the hidden neurons in order to make each attractor statistically different. These random hidden patterns are sparse (i.e., only a small fraction of the $m_h$ hidden neurons are set to 1). This also ensures greater separation between the overall $(m+m_h)$-bit attractor patterns. Attractors are encoded in the network by setting the weights between neurons in the using the covariance learning rule:

$$w_{pq} = \frac{1}{n} \sum_{i=1}^{m} (a_p^i - \bar{a}_p)(a_q^i - \bar{a}_q)$$

Where,

- $w_{pq}$ = weight from neuron $q$ to neuron $p$.
- $a_p^i$ = activity of neuron $p$ for the pattern $i$.
- $a_q^i$ = activity of neuron $q$ for the pattern $i$.
- $\bar{a}_p$ = mean activity of neuron $p$ over all patterns.
- $\bar{a}_q$ = mean activity of neuron $q$ over all patterns.
- $m$ = total number of patterns.
- $n$ = total number of neurons.
The selector network acts on the modules through the weights from its output neuron to the downstream modules. These are set such that, if output neuron $i$ is active in row $j$ of the selector network, the weights from $i$ to the neurons of motor system module $j$ are such that they receive an input pattern very close to the desired attractor state in the motor system module. However, it should also be noted that noise and other actors (e.g., interference between possibly non-orthogonal attractors) can make this imprecise and the selector network requires a few time steps after triggering to settle to the desired attractor. Thus, there may be a transient that the downstream systems must be buffered from.

Figure 3-6 shows the mapping between the neurons in the Action map and Selector network. The significance of the hidden layer can be observed from Figure 3-7 where the two patterns differ by just one bit in the output layer, but are noticeable different in the hidden layer, thus ensuring the right convergence to the desired attractor.
Figure 3-6: Mapping between neurons of Action Map and Selector Network
Figure 3-7: Hidden layer activities of two attractor patterns which differ by just one bit in the output layer

3.5 Motor System Network

The output from the selector network is sent to the cortical network. Based on the selected motor state for each of the spinal modules, the corresponding motor system neurons are activated and sent as input to the spinal network. Figure 3-8 displays the firing pattern in the motor system network for each of the possible output states in the selector network for a single motor system module.
3.6 GUI Guide

As mentioned before, the motivation behind developing the system was to provide users a platform for systematically configuring artificial animals and studying them through simulation. For this purpose, the framework developed by Salunke [1] was extended to include a customizable decision system. This, in turn necessitated changes in the graphical user interface (GUI), and making these changes to produce a user-friendly GUI is a major component of the present thesis. This has also resulted in changes to the original GUI, leading to a richer, more integrated interface. A very important addition to the system is that it now allows users to save animals that they find interesting in a way that the simulation can be restarted when desired.
Figure 3-9 provides a basic idea of the look and feel of the GUI.

The following steps help the user to create and observe animals systematically.

3.6.1 Step 1: Initialization

In this step, the user is provided the option of either creating a new animal from scratch or loading a previously saved configuration. In case the former is chosen, the user will need to configure all the parameters consistently. In case there are any discrepancies like a
dimensionality mismatch or if any of the minimum/maximum limit conditions not being met, an error message is displayed. If the user chooses to load a previously saved configuration, all the subsequent steps are bypassed and the simulation can be run directly. Alternatively, the user can also modify the configuration and then run the simulation. In this step, the user also has the option of saving the animation alone in a video file.

3.6.2 Step 2: Module Specification

In this step, the user is provided the option of setting the number of motor and spinal modules associated with the animal. The number entered as the $S$ parameter will finally be the number of segments that the animal will have. It is to be noted that unless the user clicks on the “Form Matrix” button, the connection matrix needed for the next step is not created. In order to have a reasonable simulation time, it is better to limit the number of both cortical and spinal modules to 20 at most.

3.6.3 Step 3: Connectivity Specification

This step is used to configure the dynamics within the modules of the motor and spinal systems, as well as the interactions between the modules of the two systems. Thus three different matrices are created – Motor to Motor, Spinal to Spinal and Motor to Spinal. In order to limit variation, spinal modules are only allowed to affect the behaviors of modules below them (i.e., farther away from the cortical structures). Users are also provided the option to randomize the connections in all three matrices.
3.6.4 Step 4: Motor System to Spinal Weight Specification

This step allows the user to modify the weights connecting the motor system output to the spinal modules. However, it is to be noted that the default values for these weights have been configured for the most natural performance in terms of system behavior and modifying them could lead to unexpected or spurious animal behavior.

3.6.5 Step 5: Decision Network Configuration

Clicking on the button “Configure Decision Network” opens a new GUI, in which the user is provided the option to configure the decision network parameters.

![Decision Network Interface](image)

Figure 3-10: Decision Network Interface
Figure 3-10 provides the basic look and feel of the Decision Network GUI.

### 3.6.5.1 Selector Attractor Specification

The user is provided the option of selecting the number of attractor patterns to be stored in the selector network. This corresponds to the number of motor system activations that are possible and thus the number of distinct motor behaviors the animal will have. For computational purposes and to make sure that the animal is capable of different types of behavior, the minimum number of patterns that have to be stored in the network is three. An error message would be displayed if a number entered less than that is entered. Once the user specifies the number of patterns and clicks on “Set”, the “Attractor Pattern” matrix is made available. This matrix has
rows equal to the number of patterns and columns equal to the number of motor system modules. Since each motor module embeds seven attractors, the user can enter numbers between 1 and 7 in each of the cells. Each row in the input entered by the user corresponds to a particular pattern. For example, if the animal has five motor modules, the input “1 3 1 5 7” means that, in his motor state, motor module 1 should recall attractor 1, motor module 2 attractor 3 and so on.

It is advisable to make the patterns as varied as possible for the following two reasons:

- Having a large number of extremely similar patterns would lead to false attractors in some cases.
- The more varied the patterns are, the more distinct the animal’s behaviors are likely to be.

The user also has the option of randomizing the patterns. Another precaution to be taken by the user in this step is that if an existing model had been loaded in Step 1 and if the value of $S$ (number of motor modules) had been changed in Step 2, then the matrix dimension would also change. If the user tries to run the simulation without making this change, an error message would be displayed. Nevertheless, the user just has to click on the “Set” button for the new matrix to be formed, which can then be configured.

3.6.5.2  Sensory Map and Sensory States Specification

This step allows the user to specify the size of the sensory map. The map is a $k \times k$ grid where $k$ is a number specified by the user. After clicking on “Set”, the user is provided with the option of
specifying the number of sensory states. The number of sensory states has to be at least two; otherwise an error message would be displayed. Error message would also be displayed if the map size is too small to save the minimum number of states or if there is a mismatch between the map size and the number of states, i.e., if a map with a larger size would be needed to store the desired number of states. This is based on the need to minimize interference between the activity bumps associated with different states by keeping them well-separated.

3.6.5.3 Sensory State to Attractor Configuration

Once the map size and the number of states are set, the user is shown the “State to Attractor Mapping” matrix, in which the attractor pattern that needs to be chosen for each of the sensory states is specified and the sensory state associated with the corresponding element of the switcher network. This effectively makes this step the “state-action” mapping part of the framework. If the user enters a number greater than the number of patterns previously specified, an error message is displayed. The user is also provided the option to randomize the mapping configuration.

3.6.5.4 Temporal Input Specification

In order to simulate and observe the animal’s behavior, in this step, the user is asked to provide the sequence of sensory states that the simulated animal will be stimulated with. Once the simulator is completely implemented, this information would come from the animal’s sensing of its environment, but the user needs to specify this in the current version. The user may choose
any sequence from the known sensory states. For example, if the animal has four sensory states A, B, C and D, a sequence could be A → C → A → D → B → B → A. In the current version of the system, the animal spends equal time in each state, but this will be modified in later versions. The currently active sensory state provides input to the switcher network, thereby triggering the appropriate selector attractor, leading to the corresponding action.

3.6.5.5 Return and Reset

Once all the parameters are set, clicking on “Return” stores all the parameters, closes the Decision Network GUI and returns the user to the Parent GUI. Users can also click on “Reset” if they want to get rid of the current configuration and choose new parameters instead.

3.6.6 Step 6: Simulation Time Specification

In this step, the user specifies how long they would like the simulation to run. It is to be noted that the time entered here does not mean a real unit of time, but rather the number of iterations. However, a single time step corresponds approximately to 30 milliseconds.

3.6.7 Step 7: Spinal Segment Maximum Angle Specification

The maximum angle up to which each spinal segment can turn is specified in this step by the user. Increasing this value would make the animal more flexible and could lead to much more complex – but possibly unnatural – behaviors.
3.6.8 Step 8: Run Simulation

Having configured the spinal network and the decision network parameters, the user can now click on the “Run” button to simulate and observe the animal. The animal movement is animated in the axis window, while simultaneously the data for generating plots is also collected. The user can also decide to “Abort” the simulation in the middle if needed, but this would make the option of “Save Movie” impossible.

3.6.9 Data Analysis

Once the simulation is complete, the user can also view the plots for analyzing the animal’s behavior. Using the drop down menus, the user can see plots of the various neural activities in all regions of the animal as well as other factors like the force and angle profiles etc. These plots can also be saved if needed. Another feature in the GUI is that clicking on “Abort” merely stops the animation, but all these data plots would still available for analysis.
4 Results and Discussion

The Decision network system presented in the previous chapter was added to the current framework [1] and was simulated to verify that a user would be able to systematically specify complex animals and observe their behavior. The results are presented and discussed in this chapter.

4.1 Simulation Setup

In order to demonstrate the use of the framework in helping the user systematically develop and observe complex animals, two distinct test cases were considered. In each case, the animal was given the objective of generating a distinct pattern of rhythmic oscillating mode. The two modes that were considered were termed thrashing and spiral. For both these modes, the decision network was configured by providing the right set of attractor patterns, mapping them with the appropriate sensory states and providing an input of alternating sequences so as to get a cyclic pattern.

The animals were also systematically made more complex in both the modes and their behaviors were observed. The systematic increase in complexities was achieved in the following two ways:

1. By manipulating the system parameters, thus altering the dynamics by which the input signals were converted to muscle movements. Specifically the following two parameters were manipulated.
• The rotational angle limit for a single spinal module was increased, thus making the entire body of the animal more flexible. Now the animal would be able to get into more complex positional states that were previously not possible.

• The temporal duration of the sensory states was increased, thus having the animal receive a sustained recurring input for an extended period of time, increasing the force/angle values for each segment.

2. By making the animal physically more complex by adding more segments to the existing animal. With more segments and joints, now the animal would have more degrees of freedom, providing a larger positional state space.

In the former case there was no need to modify the decision network configuration, since the manipulations were done on the spinal network and the animal was observed to exhibit more complex behaviors for the same decision network configuration. However, in case of the latter, the attractor pattern matrix for the selector network had to be modified in order to accommodate for the dimension changes owing to the addition of new segments.

4.2 Developing Complex Animals

4.2.1 Increasing the Rotational Flexibility of Each Segment

As mentioned in the previous section, an animal was built with a certain number of spinal modules, and its behavior was observed. Then the animal was made more complex by increasing the maximum limit for the rotational angle for each segment, while retaining the same number of
spinal modules. This was achieved by changing the *maximum angle* value mentioned in Section 3.6.7. With this increase in the angular flexibility for each spinal segment, the whole animal was able to explore a larger positional state space, and hence perform more complex tasks, for the same set of cortical input received from the decision network, which was unchanged. The results for an 8 segmented animal that was made to oscillate in the *thrashing* mode are shown below.

Figure 4-1 shows the animal position at different points during the simulation. The corresponding force and angle profiles for each of the segments are shown in Figure 4-2 and Figure 4-3 respectively.

---

**Figure 4-1:** Time lapse screenshots at different simulation instances for an 8 segmented animal configured to oscillate in thrashing mode
Figure 4.2: Force profiles for individual segments of an 8 segmented animal configured to oscillate in *thrashing* mode
The animal was then made more complex by having the *maximum link angle* value increased and its movements were observed. Now more force would act on the motor muscles causing greater angular rotation on each spinal segment since the force function was implemented as a function of the maximum link angle. Figure 4-4 provides the position of animal at different instants of simulation and the corresponding force and angle profiles are provided in Figure 4-5 and Figure 4-6 respectively.

It could be clearly seen from the force and the angle profiles that for the same decision network configuration, the animal was able to exhibit more complex behaviors with increase in the rotational angle as observed in the Time Lapse screenshots.
Figure 4-4: Time lapse screenshots at different simulation instances for the 8 segmented animal configured for thrashing mode with increased angular flexibility
Figure 4-5: Force profiles for individual segments of the 8 segmented animal configured for *thrashing* mode with increased angular flexibility
4.2.2 Increasing the Temporal Duration for Input Sensory States

Similar to the method explained in the previous section, an animal was created with a certain number of segments and its behavior was observed. Then the animal was made more complex by increasing the temporal duration for each of the sensory inputs that were provided to the decision network. This was achieved by increasing the value for the simulation length parameter that described in section 3.6.6. Since the current simulator allocates equal time to each sensory state, increasing overall simulation duration by a factor $K$ increases the duration of each sensory state by the same factor. Providing the same sensory input for an extended duration allows the spiking pattern for each of the motor neurons to persist longer, applying the same force for longer
durations. The results for a 12 segmented animal that was made to oscillate in the *thrashing* mode are shown below.

Figure 4-7 shows the animal position at different instants, and the corresponding force and angle profiles for the individual segments are shown in Figure 4-8 and Figure 4-9 respectively.

The animal was then made more complex by having the temporal duration for each input sensory state increased, and the behavior was observed. Figure 4-10 shows the animal position at different times and Figure 4-11 and Figure 4-12 show the force and angle profiles.

Figure 4-7: Time lapse screenshots at different simulation instances for a 12 segmented animal configured to oscillate in *thrashing* mode
Figure 4-8: Force profiles for individual segments of a 12 segmented animal configured to oscillate in *thrashing* mode.
Figure 4-9: Angle profiles for individual segments of a 12 segmented animal configured to oscillate in *thrashing* mode
Figure 4-10: Time lapse screenshots at different simulation instances for the 12 segmented animal configured for thrashing mode with increased sensory state temporal distribution.
Figure 4-11: Force profiles for individual segments of the 12 segmented animal configured for thrashing mode with increased sensory state temporal distribution
It can be seen from the force and angle profiles that the same pattern of inputs acting for an increased period of time produced more complex behaviors in the animal.

### 4.2.3 Increasing the Number of Segments

In this case, the complexity in an animal was increased by making it physically more complex, i.e., an animal was created with a certain number of spinal segments, and gradually the number of spinal modules was increased by increasing the *spinal modules* parameter described in Section 3.6.2 and changing the corresponding matrices accordingly. Now the animal had more links and segments that added more degrees of freedom, allowing the animal the opportunity to get into physical positions that were not possible for an animal with fewer segments.
To demonstrate this idea, an 8 segmented animal was developed to oscillate in spiral mode and its behavior was observed. This animal was then made more complex by having the number of segments increased to 12 and subsequently 16 and their behavior was observed. Correspondingly, as the animal’s segments were increased, the attractor pattern matrix for the selector network in the decision network was also modified in order to accommodate the dimensionality of the additional segments. However, the new matrix was given the same structure of pattern as the older one in order to keep animals consistent, and the other decision network parameters were left unchanged. The results are shown below.

Figure 4-13 shows the 8 segmented animal’s position at different simulation instants while Figure 4-14 and Figure 4-15 provide the force and angle profiles for the individual segments.
Figure 4.13: Time lapse screenshots at different simulation instances for an 8 segmented animal configured to oscillate in *spiral* mode.
Figure 4-14: Force profiles for individual segments of an 8 segmented animal configured to oscillate in *spiral* mode
Figure 4-15: Angle profiles for individual segments of an 8 segmented animal configured to oscillate in *spiral mode*

Figure 4-16 shows the 12 segmented animal’s position at different points in the simulation while Figure 4-17 and Figure 4-18 provide the force and angle profiles for the individual segments.
Figure 4-16: Time lapse screenshots at different simulation instances for a 12 segmented animal configured to oscillate in spiral mode
Figure 4-17: Force profiles for individual segments of a 12 segmented animal configured to oscillate in *spiral mode*
Figure 4-18: Angle profiles for individual segments of a 12 segmented animal configured to oscillate in *spiral mode*

Figure 4-19 shows the 16 segmented animal’s position at different simulation instances while Figure 4-20 and Figure 4-21 provide the force and angle profiles for the individual segments.

It can be seen from the force and angle profiles in all three cases that for the same type of patterns that are provided as motor inputs from the motor network, the animal was able to exhibit more complex behaviors as seen in the time lapse screenshots.
Figure 4-19: Time Lapse screenshots at different simulation instances for a 16 segmented animal configured to oscillate in spiral mode
Figure 4-20: Force profiles for individual segments of a 16 segmented animal configured to oscillate in *spiral mode*.
4.3 Significance of Decision Network Parameters in Animal Behavior

The number and the type of patterns that are configured in the selector network have the most crucial role in having the animal effectively behave as desired. The reason for this is that the attractor patterns form the set of all possible action states for the animal – a repertoire of motor possibilities. The types of behaviors that the animal can produce in a sequence of sensorimotor states depend on the richness of this repertoire.

Once the selector network is configured properly, the next major step is to map the state-action pairs. Since there is no scope in the current framework for the animal to learn the right set of state-action mappings, it will have to be provided manually by the user. Hence it is also equally
important for the user to define the number of sensory states and map the pattern to be chosen for each state. Finally, having configured the patterns and mapping them to the appropriate states, the user can now choose the number and the type of sensory states that would be presented to the decision network as sensory inputs during the course of the simulation, and to specify their sequence.

4.4 Significance of Attractors

The main reason behind having the selector network converge as a recurrent attractor network is to make the system more robust and stable. Having the selector output settle into one of the attractor patterns after a transient state based on the output from the Action map makes the network more tolerant to any kind of noise in the signals from the sensory map and/or action map. Particularly in scenarios where overlapping regions in the sensory map are mapped to similar patterns in the selector network, the dynamics of the attractor networks ensure the convergence towards the right pattern for a given state. This was verified by introducing noise in the action map output for a specific sensory state and verifying if the selector network converged to the desired attractor pattern for that input.

The noise was generated as a normal distribution with mean zero and a standard deviation $\eta$ and added to the output signal generated from the action map and sent as input to the selector network. The parameter $\eta$ was varied as a fraction of the magnitude of selector input (i.e.) from 0
to 1 and the fraction of true attractor convergence was plotted. The plot can be seen in Figure 4-22.

![Effect of Noise in Selector Input on True Attractor Convergence](image)

**Figure 4-22: Effect of noise on true attractor convergence**

It could be observed that even for a noise as high as almost the magnitude of the input the attractor network had an 80% convergence, indicating its stability and robustness in adapting to complex dynamic changes in its environment.
5 Conclusion and Future Work

The main motivation behind the work in this thesis was to provide a framework for configuring and exploring modular animals as well as simulating and observing the effect of development in them. This thesis particularly focused on extending the framework to provide the user the option of configuring the decision network and setting its parameters. The key feature that was utilized in the model was the use of modularity in order to clearly define and decentralize the functionality of individual components.

The main ideas that govern the motivation behind developing this framework are:

- Interaction of simple modules leads to the emergence of complex behaviors in animals.
- Artificial systems built using developmental and/or evolutionary learning techniques can learn to perform complex tasks in dynamic environments [8] [9] [10].
- The complexity that arises with control of a complex system can be greatly reduced by the appropriate use of embodiment [25] [26].
- Modularity plays a vital role in evolvability of complex systems.

Additionally, the graphical user interface was also extended with a separate component added for providing the users a simple yet effective portal for customizing the decision network parameters. This new interface, coupled with the existing interface for configuring the motor network parameters, allows users to utilize the full scope of the framework for developing and
analyzing modular segmented artificial animals. The interface was also extended to have the
decision network parameters saved along with the motor network parameters when they choose
the “Save Model” option. The previously configured decision network parameters are loaded
along with the motor network parameters when a saved configuration is reloaded.

This framework and interface would be used as the base of a multi-stage project, in order to
further explore the possibilities of developing fully functional, autonomous, modular, self-
organizing animals capable of emergent behavior through the process of developmental and/or
evolutionary learning. Currently the sensory inputs are manually provided to the system and the
state-action mappings are explicitly stated. In future work, these issues would be addressed and
the animal would be expected to inherently develop all controls necessary for complex behaviors
through biologically plausible learning processes.

Specific areas of work that could be incorporated in the near term include:

- Including a full, neurally implemented sensory system that would learn sensory states
  through self-organization and learn useful mappings from sensory to motor states through
  reinforcement learning – possibly in the context of specific tasks.
- Using the system to explore the effects of specific complexification processes on animals
  with canonical architectures. This would be useful in developing theories of how motor
  behaviors observed in nature may have arisen through evolution and/or development.
• Integrating the simulation system with a larger evolutionary and/or developmental simulation system to study issues in those areas.

• Using the system as a test bed for developing neural algorithms for segmented modular robots.

• Developing animals with limbs attached to the side that would act as hands or legs [32].

• Developing animals that are not anchored and are capable of locomotion.

• Extending the framework to place the animal in a 3-dimensional environment, possibly using a physical simulation environment such as ODE.
6 References


[26] N. A. Bernstein, On dexterity and its development, In M. L. Latash and M. Turvey (Eds.), *Dexterity and its development* (pp. 3-244 ), Mahwah, New Jersey: Lawrence Erlbaum Associates, 1996


