

# Two Simulation Tools for Biologically Inspired Virtual Robotics

David Gamez<sup>1</sup>, Richard Newcombe<sup>2</sup>, Owen Holland<sup>3</sup> and Rob Knight<sup>4</sup>

Department of Computer Science, University of Essex, Colchester, C04 3SQ, UK

{<sup>1</sup>daogam,<sup>2</sup>ranewc,<sup>3</sup>owen,<sup>4</sup>rrknig}@essex.ac.uk

**Abstract** - This paper describes two simulators that have been developed as part of Holland's and Troscianko's project to build a conscious robot. The first simulates a reconfigurable humanoid robot within a dynamic environment. This robot is more biologically realistic than traditional humanoid robots because its movements are transmitted across its whole structure, which poses the same control problems that brains have to solve with biological systems. The second simulator in this paper simulates large numbers of spiking neurons with a flexibility and speed that makes it ideal for developing models of biologically structured neural networks. In combination these simulators will enable researchers to develop models of how the brain controls the human body and they will also be used to create and test controllers for the real CRONOS robot, on which the virtual robot is based. These tools are currently in their final stage of development and will soon be made freely available for non-commercial use.

**Keywords:** Spiking neural networks, robot simulator, virtual robotics, anthropomorphic, PhysX.

## 1 Introduction

This paper outlines two simulation tools that have been developed as part of Holland's and Troscianko's EPSRC Adventure Fund project to build a conscious robot. The overall aim of this project is to build a system that is likely to be attributed some form of phenomenal experience by most of the major theories of consciousness. The hardware robot 'CRONOS' that has been constructed for this project is anthropomorphic in design [5] and includes analogues of the human muscles and skeleton as well as a foveated visual system (see Figure 1).

To accelerate the development of control strategies for CRONOS, a virtual robot 'SIMNOS' has been built to provide a highly reconfigurable simulation of CRONOS in a dynamic environment. This simulation replicates the large numbers of degrees of freedom and general composition of the real robot and it has been created using Ageia PhysX [1], which provides a scalable, highly stable, soft real-time physics engine on desktop PCs and gaming platforms.

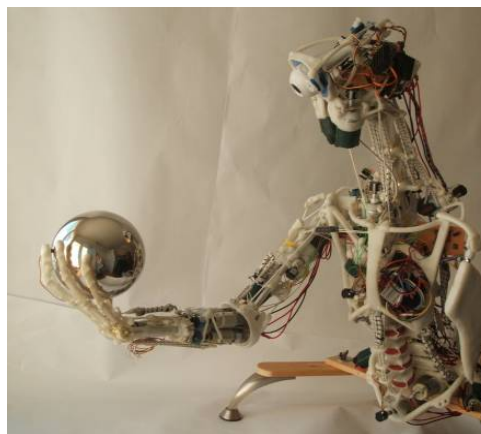


Figure 1. Real CRONOS robot

For the 'brain' of CRONOS we wanted something that could be used to test some of the ideas about consciousness that have emerged from work on its neural correlates. Many of these rely on synchronization within a potentially large number of neurons and event-driven spiking simulators offered the best combination of performance and biological realism for this task. This type of simulator is a relatively recent development and since none of the existing implementations met our needs, the SpikeStream simulator was developed for this project.

This paper starts with the design and performance of SIMNOS and SpikeStream. After covering some of the previous work on virtual robots and spiking neural simulators, it finishes with some applications of these simulation tools and our planned future work.

## 2 The SIMNOS Virtual Robot

### 2.1 Body structure and muscle model

CRONOS' skeletal body components are modelled in SIMNOS as jointed rigid bodies, with spring-damper systems at each joint in place of the un-powered elastic tendons. The muscles on CRONOS consist of a length of Dyneema kiteline wound around a rotary motor axis, situated at the first anchor site of the muscle. The free end of the line attaches to the second anchor site via a piece of elastic shock cord. By winding in the line, after all slack has been removed in both kiteline and cord, the elastic element pulls the anchor points together. The design of the series-elastic actuator restricts the possible

forces developed in the system to pulling on the anchor points.

The simulated CRONOS muscle abstracts from this system and models the muscle with a single parallel spring-damper system with asymmetrical conditioning of  $K_{current}$  and  $B_{current}$ , depending on the displacement of the spring (1) (2), with spring and damper values  $K_{current}$  and  $B_{current}$ ,  $b_r$  = spring resting length,  $b_c$  = current spring length,  $b_d = b_c - b_r$ , and  $b_d' = b_d/dt$ .

$$F_m = b_d \cdot K_{current} - b_d' \cdot B_{current} \quad (1)$$

$$K_{current} = \begin{cases} K_{set} & \text{iff } b_c > b_r \\ 0 & \text{otherwise} \end{cases}, \quad B_{current} = \begin{cases} B_{set} & \text{iff } b_c > b_r \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The winding of the CRONOS motors to alter the length of the shock cord is abstracted as the setting of the resting length of the spring  $b_r$ , with a maximum rate of change per time step. The values of  $K_{set}$  and  $B_{set}$  alter the characteristics of the muscles contraction by changing the forces created when the spring is stretched.

The simulator has been designed so that all the parameters of the robot and its environment can be updated and reloaded during simulation, either using a graphical interface or an XML file. This enables various configurations of muscles, limb proportions and other muscular structural properties to be altered and updated during simulation.

## 2.2 Whole body movements and controller issues

The elastically coupled multi-degree of freedom structure and series-elastic actuators provide robustness to failure induced through unplanned environment impacts [5]. All body loads are transferred throughout the structure and so, unlike traditional robotics platforms, all limb movements and robot-environment interactions are whole body movements.

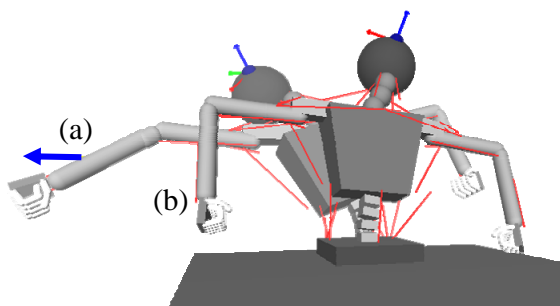


Figure 2 Front views of SIMNOS. Lines are rendered at locations corresponding to the positions of muscles used to actuate limbs. Control of the virtual robot is also possible by specifying joint angles or individual body poses. The overlaid scenes demonstrate the application of an external force to SIMNOS.

To demonstrate a whole body movement, each SIMNOS muscle was set to position the robot upright. An external force was then applied to the right forearm

of the robot, essentially pulling the robot to one side and displacing the muscles from their resting length (Figure 2a). The external force was then removed. Time series data for the lengths of the two biceps and two of the lower torso muscles are plotted in Figure 3.

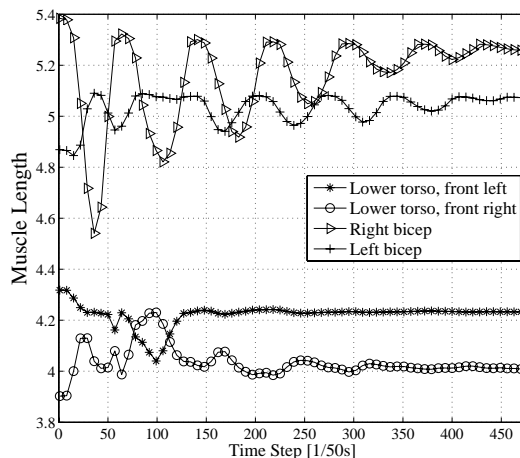


Figure 3. Time series data for muscle lengths during a full body movement produced by an external force applied to the robot

Initially the left bicep oscillates in counter phase to the right. Then the stiffer lower torso muscles stabilise, which affects the bicep oscillation relationship. The asymmetrical oscillation of the individual series-elastic bicep muscles also alters the force transference around the system. As a result of this disruption the new point of equilibrium (Figure 2b) is different from the original. This simple experiment demonstrates that control of SIMNOS and CRONOS must be performed within the context of both the local limb actuation and the composite body movement that results from this.

## 2.3 Sensory data and spike train coding

The sensory data generated by this simulator includes 25 Euler angle values that monitor the relative rotations of thorax-pelvis and head-thorax and every degree of freedom in each hand, arm and shoulder complex. The robot is currently equipped with 41 muscles and the current length is available for each muscle, together with the control values that were issued to it, giving a total of 164 values per time step. The virtual robot is configurable to have either one or two eyes, which provide a continuous visual stream from the virtual environment.

To interact with the SpikeStream simulator a simple model has been developed to convert the real valued sensor data into a time varying spike train. Current theories of neural coding schemes fall under either rate or temporal encoding [2] [14]. Here we utilize a hybrid, spatially distributed, average rate encoding method. This spans the range of a real valued variable with a set of  $N$  broadly tuned 'receptors'. Each receptor,  $n \in \{0..N\}$ , where  $N > 1$ , is modelled with a normalised Gaussian with mean  $\mu_n$  and variance  $\sigma_n^2$  (3) (4), with the values of  $\mu_n$  computed to equally divide the

variable range with a receptor mean at the minimum and maximum of the range.

$$\mu_n = \frac{n}{N-1} \quad (3) \quad \sigma_n = \frac{1}{3(N-1)} \quad (4)$$

Given a real valued variable at time  $t$ , ( $v_t \in [0..1]$ ), the spiking output of each receptor ( $r_n \in \{0,1\}$ ) is computed based on the probability,  $p(n, v_t)$  of that receptor firing (5) (6), where  $c$  is a scaling factor used to control the maximum firing rate of a receptor and  $rand$  is drawn from a uniform distribution. The variance of a receptor is chosen to ensure that  $p(n, v_t) = 1$  when  $\mu_n = v_t$ , with all other receptors having negligible probability.

$$r_n(t) = \begin{cases} 1 & \text{iff } p(n, v_t) > k \\ 0 & \end{cases} \quad (5)$$

$$p(n, v_t) = e^{-\frac{(v_t - \mu_n)^2}{2\sigma_n^2}} \quad k = c \cdot rand[0,1] \quad (6)$$

Given  $N$  spike trains the conversion back to a real value is performed by taking the average normalised firing rate  $fr_n(t)$  for the current time step  $t$  within a given window of  $w$  previous simulation steps for each of the  $N$  spiking signals. The approximated real value at this time,  $\tilde{v}_n(t)$ , is then the sum of the receptor means weighted by the normalised firing rate (7) (8).

$$fr_n(t) = \frac{\sum_{i=t-w}^t r_n(i)}{w} \quad (7) \quad \tilde{v}_n(t) = \sum_{n \in N} \mu_n \cdot fr_n(t) \quad (8)$$

Such a spatially distributed rate encoding provides resilience to noisy signals, with the benefit that increased resolution in spiking representation can be achieved without altering the rate of firing of an individual neuron.

## 2.4 Simulated environment

At present, SIMNOS' environment contains rigid bodies that are either simple geometrical shapes or triangular meshes. Objects in the virtual environment can be loaded and used with two levels of interactive complexity. In the simplest mode a cube or sphere bounds the object and this is used primarily for visualisation purposes. In the second mode an object model is loaded that creates a physical surface for each triangle of the model mesh. Jointed dynamic models can be added to the environment and cloth- and fluid-based objects are also possible within the PhysX framework. These are currently being developed for use in the simulator.

## 2.5 Simulator performance

The possibility of running a simulated version of CRONOS in faster than real time opens up novel modes of control for this robot. For example, SIMNOS could

be used to test the outcome of potential actions of CRONOS to solve control issues and select the best action to perform on the real robot.

To provide a test of the computation time requirements of the simulator, a simple virtual world was developed. During the scene simulation all sensory and motor data in both real valued and spiking stream forms was computed. The simulator was run for 3000 steps. At the end of each time step a newly created sphere was dropped onto the surface of the table where the robot was fixed and muscle parameters were altered with random values, thus providing the maximum computational load. The spheres were created with randomised diameters and random values for density, bounciness and friction. Simulator gravity was set to mimic earth gravity. As the objects fell onto the table and onto the floor they interacted with the robot, the environment and each other.

The computation times for this virtual world were recorded for a number of different time steps,  $ts \in \{1/50, 1/62.5, 1/100\}$  [s], and plotted in Figure 4. Smaller time steps provided a more accurate simulation of the muscles and joint limits when the velocities and forces in use were large.

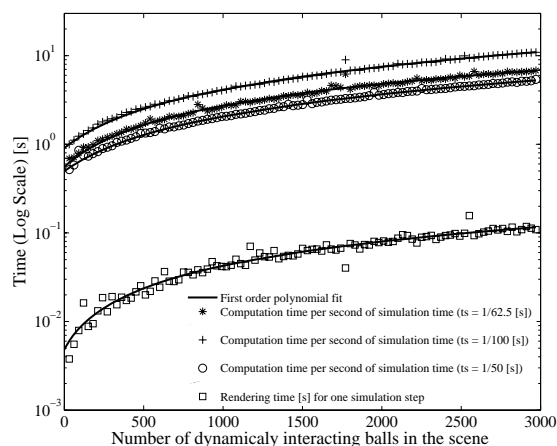


Figure 4. Times for computing the experimental scene on a 1.66 GHz Intel Core Duo notebook with different time step values

The time taken to render a scene in the current implementation is linearly proportional to the number of visible objects at each point in time. Visualisation of each simulation step will be unnecessary for most user requirements for  $ts < 1/50$ . The time required to compute a given time step increases only linearly with the number of interacting objects. Soft real time simulation of the robot in an environment with 300 objects, with full scene rendering for user output, is therefore possible for the larger  $1/50$  s time step value.

The availability of cheap physics processing hardware developed for the PhysX engine will allow large scale environments consisting of thousands of interacting dynamical objects to run in soft real time on

a modern PC, enabling the provision of a rich environment for SIMNOS to explore.

## 3 SpikeStream Simulator

### 3.1 Architecture

The SpikeStream simulator is based on the SpikeNET architecture [3], which allows for the efficient simulation of large numbers of neurons as long as each neuron is relatively unlikely to fire within each time step. The efficiency of this approach comes from the fact that each neuron and synapse is only processed when it receives a spike, with their state being calculated retrospectively for the intervening time. This method does not operate completely asynchronously because it uses the exchange of spike messages to keep the layers running at the same rate, which makes this architecture behave much like a synchronous simulator, but with the performance gain of making the updates event driven. Since the main overhead is in processing the neurons and sending the spikes, the simulator's update speed depends heavily on the level of network activity and at high levels the difference between event driven and synchronous simulators starts to break down.

The original implementation of SpikeNET was quite limited and lacked many important and useful features. For this reason it was decided to create a new simulator with the following key attributes:

1. *Database storage.* The SpikeStream simulator is based around a database that holds all the neurons, connections, parameters, archives and patterns. This makes it very easy to launch simulations across a variable number of machines and provides a great deal of flexibility in the creation of connection patterns.
2. *Parallel Operation.* All of the spike message passing and communication between processes is carried out using PVM so that the simulation can run in parallel across an arbitrary number of machines.
3. *Sophisticated visualisation and editing tools.* An intuitive graphical user interface has been created for the creation and editing of neuron and connection groups and the control of the simulation and archiving. There are also windows to enable the live monitoring of neural activity and the playback of neural activity from archived simulation runs.
4. *Variable delays.* SpikeStream creates variable delays by copying emitted spikes into 250 buffers corresponding to different delay values. At each time step only the spikes in the currently active buffer are sent.
5. *Dynamic synapses.* Each connection between neurons has a unique ID, which is used to route

the spike to a particular synapse class. This makes it possible to model the dynamic response of synapses, as discussed by Maass and Zador [9].

6. *C++ / Qt.* The simulator is written in C++ using Qt for the GUI.
7. *Dynamic class loading.* Neuron and synapse classes are implemented as dynamically loaded libraries, making it easy to try out different neural and synaptic models without recompiling the whole application. Each dynamically loadable class has a parameter table in the database, which makes it easy to change the parameters from the GUI during the simulation run.
8. *Live operation.* Unlike the majority of neural simulation tools, SpikeStream is designed to operate in live as well as simulation time so that it can be used to control real and virtual robots in close to real time, and can also process input from live data sources.
9. *Modular architecture.* The editing and simulating functions of SpikeStream are completely separate from each other and from the database, making it easy for other applications - for example, genetic algorithm software - to modify the neural model in the database without affecting the ability of SpikeStream to display the neural model and run a simulation of it.

### 3.2 Performance tests

The SpikeStream simulator was tested using a variable number of pairs of recurrently connected layers. At the beginning of the simulation run 50% of the neurons were activated in one of the layers in each pair and then the simulation was left to run without any external input for 60 seconds, after which the average update time and neuron fire rate was recorded. By ensuring constant activity in each of the layers, this approach enabled us to evaluate the performance of the simulator on networks of different sizes, thus giving a reasonable idea of how the simulator would perform in a real situation. Two different layer sizes of 5,000 and 10,000 neurons were used to evaluate the effect of layer size on update time. On average there were ten connections from each neuron, leading to around 50,000 and 100,000 connections in each direction for the 5,000 and 10,000 layer sizes. The neural model for these tests was a leaky integrate and fire neuron based on Marian's interpretation of the Spike Response Model [8]. No learning, live monitoring or archiving was used during these tests, which were run on a Pentium IV 3.2 GHz machine.

The performance of SpikeStream was measured in two different ways. The first was the amount of time taken to process each time step. As mentioned in section 3.1, SpikeStream uses a combination of event-based and synchronous simulation and so the update time per time step indicates how close the simulator is running to real time (useful if it is to be used for live operation) and

also measures how long it will take to simulate a given amount of neural time. The limitation of the update time per time step measurement is that when there are no events, the simulator runs very fast and it ceases to be a useful indicator of performance. To solve this problem, the average number of neurons firing per second was also recorded to give an idea about the amount of neural activity that SpikeStream is capable of simulating.

### 3.3 Performance results

The results from the performance tests are shown in Figure 5 and Figure 6.

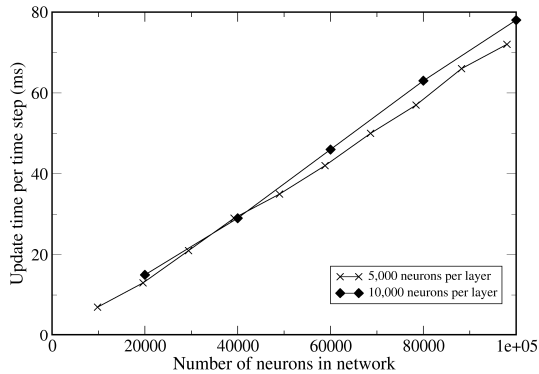


Figure 5. Update time per time step for networks with different numbers of neurons

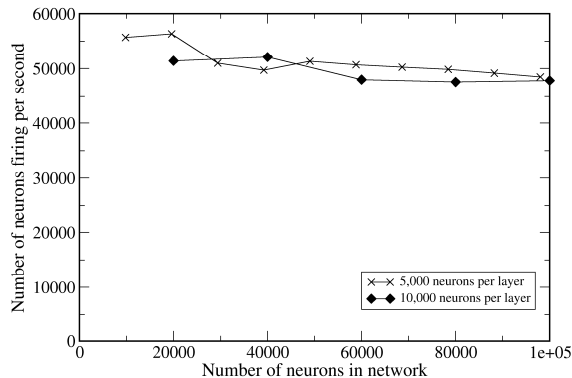


Figure 6. Number of neurons firing per second for networks with different numbers of neurons

The graphs in Figure 5 and Figure 6 show that for small networks of 10,000 neurons SpikeStream can update the network in less than 10 ms for an average firing rate of 5 Hz. As the network size increases to 100,000 neurons, the update time per time step goes up to around 80 ms with each neuron firing at 0.5 Hz. The graph of the average neuron firing rate in Figure 6 highlights the event-driven nature of this simulator since the number of neurons firing per second remains at approximately 50,000 independently of the network size. Since each neuron has an average of ten connections, this amounts to 500,000 spikes per second. In these tests there is constant network activity per layer and so larger networks generate more events, causing the update time per time step to increase linearly with the number of neurons in the network.

## 4 Previous Work

With the introduction of freely available physics engines (e.g. [16]), the increasing physical realism of simulators built on dynamics engines [4][6] has made it possible to develop robot controllers that can be run with little change on the target platform [7]. Generally, however, these simulators have been restricted to widely used, non-biologically inspired robotics platforms. Outside of the mobile robotics community, researchers within biomechanics have created highly realistic emulations of human movement using more computationally intense methods [11]. Recently, bridging the gap between robotic and biologically realistic simulations, the Virtual Soldier project [17], is developing a real-time bio-mechanically correct human to allow the evaluation of soldier behavior.

The main influence for SpikeStream was SpikeNET developed by Delorme and Thorpe [3]. This architecture was covered briefly in section 3.1 and it is claimed that it can handle millions of neurons and hundreds of millions of synaptic weights in close to real time. However, SpikeNET has a number of limitations that considerably reduce its flexibility. These include the fact that it can only process a single spike per neuron during a simulation run, its use of identical topologies and weights for connections of the same type, its use of text files for configuration, the absence of a GUI, a simple learning and neural model and the lack of delays or dynamic synapses. These limitations mean that although SpikeNET is faster on visual processing, SpikeStream is a much more flexible general purpose simulator with the potential to incorporate a greater number of biological features into its neural model. Other influences on SpikeStream include SpikeSNNs created by Marian [8], which has variable delays and semi-biologically realistic learning. However SpikeSNNs relies on a single event queue, which slows it down considerably and makes it unsuitable for parallel operation.

## 5 Applications and Future Work

One application of these simulators and our main future work is to develop models of consciousness that can be used to control both SIMNOS and CRONOS. These will build on the work of Seth et al. [13] and Shanahan [15] and use SpikeStream to create a multi-layered neural simulation that causes SIMNOS or CRONOS to respond preferentially to features of its environment that positively stimulate its emotional system. This neural architecture will then be analyzed for consciousness using some of the major theories.

Some of the research on consciousness stresses the role that internal models may play in both cognition and consciousness (for example Metzinger [12]). Internal models are also important in feed forward movements, where the model is used as a predictor of the future state that can be used to correct the movements much faster than sensory feedback. SIMNOS will enable us to test

these ideas about internal models by using the virtual robot as an internal model of CRONOS (or a second SIMNOS), with the 'internal' simulated robot being updated with information from its real or virtual environment. This would be similar to the Grounded Simulation Model developed for the Ripley robot [10].

All of this work on controllers and consciousness will use both the real and virtual robot, and part of our future work will be to evaluate how easy it is to make the transition between virtual and real robotics.

## 6 Conclusions

This paper has briefly covered two tools for biologically inspired virtual robotics: the SIMNOS virtual anthropomorphic robot and the SpikeStream spiking neural simulator. These will enable researchers to construct complex spiking neural networks and use them to control a virtual robot closely modelled on the human body. Because SIMNOS is based on the real CRONOS robot, it will also be possible to test neural networks that have been developed in the virtual environment on a real robot.

## Acknowledgements

We would like to Renzo De Nardi and Hugo Gravato Marques from the Essex Machine Consciousness Lab for stimulating discussions on many aspects of the project. This work was funded by the Engineering and Physical Science Research Council Adventure Fund (GR/S47946/01).

## References

- [1] <http://www.ageia.com/developers/api.html>, 2006.
- [2] W Bialek, F. Rieke, R. R. de Ruyter van Steveninck, D. Warland, "Reading a neural code", *Science* 252, pp 1854-1857, 1991.
- [3] Arnaud Delorme and Simon J. Thorpe, "SpikeNET: An Event-driven Simulation Package for Modeling Large Networks of Spiking Neurons", *Network: Computational in Neural Systems* 14, pp. 613-627, 2003.
- [4] Brian Gerkey, Richard T. Vaughan and Andrew Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems", *Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003)*, pp. 317-323.
- [5] Owen Holland and Rob Knight, "The Anthropomorphic Principle", in Burn, Jeremy and Wilson, Myra (eds.), *Proceedings of the AISB06 Symposium on Biologically Inspired Robotics*, 2006.
- [6] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator", *IEEE/RSJ International Conference on Unintelligent Robots and Systems*, 2004.
- [7] H. Lipson and J. B. Pollack, "Automatic design and Manufacture of Robotic Life forms", *Nature* 406, pp. 974-978, 2000.
- [8] I. Marian, A biologically inspired computational model of motor control development, MSc Thesis, Department of Computer Science, University College Dublin, Ireland, 2003.
- [9] Wolfgang Maas and Anthony M. Zador, "Computing and Learning with Dynamic Synapses", in Maass, Wolfgang and Bishop, Christopher (eds.), *Pulsed Neural Networks*, The MIT Press, 1999.
- [10] Nikolaos Mavridis and Deb Roy, "Grounded Situation Models for Robots: Bridging Language, Perception, and Action", *AAAI-05 workshop on modular construction of human like intelligence*, 2005.
- [11] S. McGuan, "Human Modeling - From Bubblemen to Skeletons", presented at the 2001 SAE Digital Human Modeling Conference, Arlington, VA, 2001. Available at: <http://lifemodeler.com/>.
- [12] Thomas Metzinger, *Being No One*, Cambridge Massachusetts: The MIT Press, 2003.
- [13] A.K. Seth, J.L. McKinstry, G.M. Edelman and J.L. Krichmar, "Visual binding through reentrant connectivity and dynamic synchronization in a brain-based device" *Cerebral Cortex* 14, pp. 1185 – 1199, 2004.
- [14] M. N. Shadlen, W.T. Newsome, "Noise, neural codes and cortical organization" *Current Opinions in Neurobiology* 4, pp. 569-579, 1994.
- [15] Murray Shanahan, "Consciousness, Emotion, and Imagination; A Brain-Inspired Architecture for Cognitive Robotics", *Proceedings of the AISB05 Symposium on Next Generation approaches to Machine Consciousness*. pp. 26-35, 2005.
- [16] Russell L. Smith, <http://www.ode.org/ode.html>
- [17] Jingzhou Yang, Tim Marler, Kimberly Farrell, Steven Beck, HyungJoo Kim, Karim-Abdel Malek, J.S. Arora, and Kyle Nebel, "Santos: A New Generation of Virtual Humans", *SAE 2005 World Congress*, 2005.