Neuromorphic System Testing and Training in a Virtual Environment based on USARSim

Christopher S. Campbell, Ankur Chandra, Ben Shaw, Paul P. Maglio, Christopher Kello

***Abstract*—Neuromorphic systems are a particular class of AI efforts directed at creating biologically analogous systems functioning in a natural environment. The development, testing, and training of neuromorphic systems are difficult given the complexity and implementation issues of real physical worlds. Certainly, creating physical environments that allow for incremental development of these systems would be time-consuming and expensive. The best solution was to employ a high-fidelity virtual world that can vary in task complexity and be quickly reconfigured. For this we chose to use USARSim---an open source high-fidelity robot simulator with a high degree of modularity and ease-of-use. We were able to accelerate our testing and demonstration efforts by extending the functionality of USARSim for testing neuromorphic systems. Future directions for extensions and requirements are discussed.**

I. Introduction

Artificial intelligence (AI) has always been a vastly broad and multidisciplinary field including everything from decision making agents in economic models to pattern recognition systems, to learning models, to natural language processing algorithms to name a few (see [1]). While some approaches are biologically inspired, many treat the system like a black-box such that only the observable behavior need show intelligent or human-like qualities. In contrast, neuromorphic systems are a particular class of AI systems aimed not at just biologically inspired models, but biologically and psychologically analogous systems. Neuromorphic systems can model many levels including intra-cellular processes, neuron growth and learning, as well as whole brain systems and connections among them. However, due to the complexity of many nervous systems (i.e., mammalian brains), neuromorphic systems work usually involves an attempt to model only a part or subsystem of an entire nervous system.

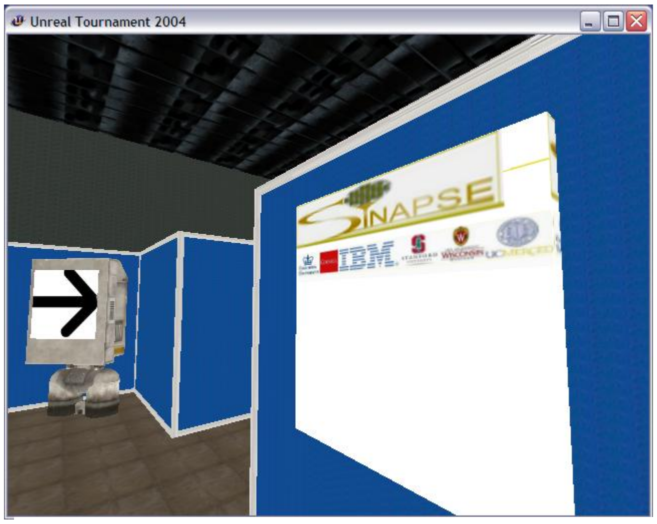


Fig. 1. Synapse 3D Virtual Environment.

Neuromorphic systems research in the software domain has focused on creating simulations of biological systems at all scales. A strong push in this direction was seen in the explosion of research on neural networks and parallel distributed processing in the 1980s [2]. Much of this work was inspired by theoretical considerations of neural organization going back to the 1950s. Such is the case with the Perceptron [3] (a simple feed-forward network of simulated neurons) and Hebbian learning theory [4]. A digital simulation of a "large" 512 neuron neural network was also conducted in IBM Research at this time [5].

In contrast the software simulation, neuromorphic systems research in the hardware domain involves building actual circuits that represent a brain or brain subsystem. A growing interest group has been in existence from the 1980s and the bulk to research focuses on intelligent robotics, low-level perception and motor control.

Neuromorphic technology that replaces programmable systems with learning or adaptive systems would be a significant step forward. If we subscribe to Turing's idea that intelligent systems merely need to demonstrate intelligent behavior, then programmable systems will be sufficient.

Manuscript received July 17, 2009. This work was supported in part by the U.S. Department of Defense Advanced Research Projects Agency (DARPA), Defense Sciences Office (DSO) under "Cognitive Computing via Synaptronics and Supercomputing (C2S2)", DARPA Contract No. 11-09-C-0002. October 2008 – July 2009. Distribution Statement "A" (Approved for Public Release, Distribution Unlimited) by DISTAR Case 14006.

Christopher S. Campbell is a Research Staff Member at the IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120 (408-927-1784; e-mail: ccampbel@almaden.ibm.com).

Ankur Chandra is a Research Software Architect at the IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120 (408-927-2455; e-mail: achandra@us.ibm.com).

Ben Shaw is an Associate Researcher at the IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120 (408-927-2659; e-mail: shawbe@us.ibm.com).

Paul P. Maglio is a Senior Research Manager at the IBM Almaden Research Center, 650 Harry Rd, San Jose, CA 95120 (408-927-2857; e-mail: pmaglio@almaden.ibm.com).

Christopher Kello is an Associate Professor in the School of Social Sciences, Humanities and Arts, at the University of California, Merced, 5200 North Lake Rd., Merced, CA 95343 (e-mail: ckello@ucmerced.edu).

However, there is a growing consensus that intelligence involves more than just the demonstration of intelligent behavior through specific task performance. Rather intelligence requires that good outcomes in task performance are accompanied by additional characteristics such as flexibility, efficiency, generalizability, and creativity. In other words, intelligence is not necessarily just solving a problem, but also the manner in which the problem is solved. Thus neuromorphic systems solve problems in a desirable manner which produces a host of useful qualities:

- managing complex, real-world, dynamic environments
- efficient energy use in terms of power
- efficient information processing
- robustness to damage
- self-organizing and scalable systems
- creative and adaptive use of the environment

The development of neuromorphic systems is without question challenging and complex. Even the simplest mammalian nervous systems have tens of millions of neurons and thousands of interconnected brain structures. To build such a system would require an unprecedented multi-disciplinary team that can work in areas such as computational neuroscience, artificial neural networks, large-scale computation, neuromorphic VLSI, information science, cognitive science, materials science, unconventional nanometer-scale electronics, and CMOS design and fabrication.

II. Building A Neuromorphic System

The Cognitive Computing via Synaptronics and Supercomputing (C2S2) project is a large multi-organization and multi disciplinary effort aimed at creating both the hardware and software components of a neuromorphic system on the scale of a small mammal (i.e., rat). The goal is to create hardware components that behave like biological synapses so the term *synaptronics* is used to refer to the hardware. These collaborating organizations include at IBM Almaden Research Center, Stanford University, Cornell, Columbia University, The University of Wisconsin-Madison, and The University of California-Merced. Each organization may have multiple teams. This effort is in response to a DARPA BAA (i.e., DARPA-BAA 08-28) called Systems of Neuromorphic Adaptive Plastic Scalable Electronics (Synapse) requesting proposals for the development of a neuromorphic system.

There are four main areas if the project and these various teams may work in one or more of these areas. They include:

- *Hardware*: The hardware teams are responsible for building the circuitry for the synaptronic brain from the materials all the way to the full-scale system. They are responsible for creating components that mimic biological synapses showing spike-based information encoding and spike time dependent plasticity (STDP).
- *Architecture:* The architecture teams will evaluate and compile the literature on brain anatomy, physiology, and function. They are responsible for designing the architecture of the synaptronic brain such that it approximates the connectivity, modularity, hierarchical organization, self-organization, reinforcement, and inhibition systems of a biological brain. Processing will should also be distributed, inherently noise-tolerant, and robust to damage.
- *Simulations* : The simulations teams are responsible for creating software to test and explore subsystems to ensure they perform as expected before development of the synaptronic brain. While many of the subsystems can be developed and standard workstations, the large scale simulations will require the use of a supercomputer.
- *Environments*: The environments teams will develop an environment to test, train, and benchmark both the software simulations and the final synaptronic brain. The environments teams will also need to create tests incrementally increase task complexity and intelligent response to evaluate the progress of the project.

The initial coordination plan for C2S2 includes the

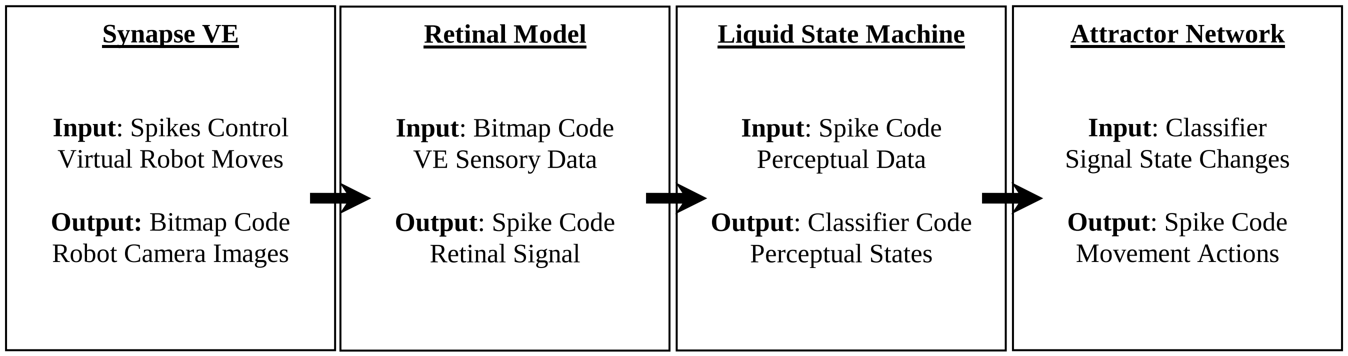

Synapse VE
Input: Spikes Control Virtual Robot Moves
Output: Bitmap Code Robot Camera Images
Retinal Model
Input: Bitmap Code VE Sensory Data
Output: Spike Code Retinal Signal
Liquid State Machine
Input: Spike Code Perceptual Data
Output: Classifier Code Perceptual States
Attractor Network
Input: Classifier Signal State Changes
Output: Spike Code Movement Actions

Fig 2. C2S2 team coordination and data flow through several quasi-biological neuromorphic systems. This effort is an attempt to approximate the future final system design with a software prototype. Each box is a different team.

following. While the hardware team evaluates materials and designs to approximate synapses and brain sub-systems, the simulations and architecture teams start to build a fully simulated prototype of the neuromorphic system. In parallel, the environments team is to create a virtual environment (VE) to bind coordination efforts and integrate simulations.

III. Environment for Benchmarking and Training

A. *Motivation*

Our goal as part of the Environments and testing team was to create a testing and training environment to support all other teams in the development and benchmarking of their components of the system. Our mission was that the environment had to support

1. fast environment development
2. accelerated training
3. incremental testing
4. benchmarking
5. wide range of environment fidelity
6. wide range of task complexity

In response to these requirements, we created the Synapse Virtual Environments Server (Synapse VE Server) as a common interface layer for all the teams working on the C2S2 project. The Synapse VE Server is integrated with a system called the Unified System for Automation and Robot Simulation (USARSim) [6] which provides a framework for creating, controlling, and interacting with a robot in a VE (see Fig 1). While it is true that each C2S2 team could use USARSim to do their own development and testing, providing one common interface has several benefits:

- removes redundant work,
- standardizes the training and benchmarking,
- provides a unified look-and-feel to the project
- serves as a point of integration for the neuromorphic system components

Using USARSim was logical given that it solves many of the problems we would encounter attempting to interface with a VE. Another benefit is that USARSim provides a way to control and monitor a player in a commercially available game called Unreal Tournament 2004 produced by Epic Games [7]. This game is a standard multi-player networked combat-oriented first-person for both Windows and Linux operating systems. Commercially available VEs tend to be of higher quality---e.g., more realistic, provide development tools, and have a pre-built physics engines. The labor and cost of developing our own VE of similar quality would be far beyond the resources of this project. No doubt, if we were forced to created our own VE, it would not have the complexity and realism necessary to test and train neuromorphic systems. Another benefit is that every

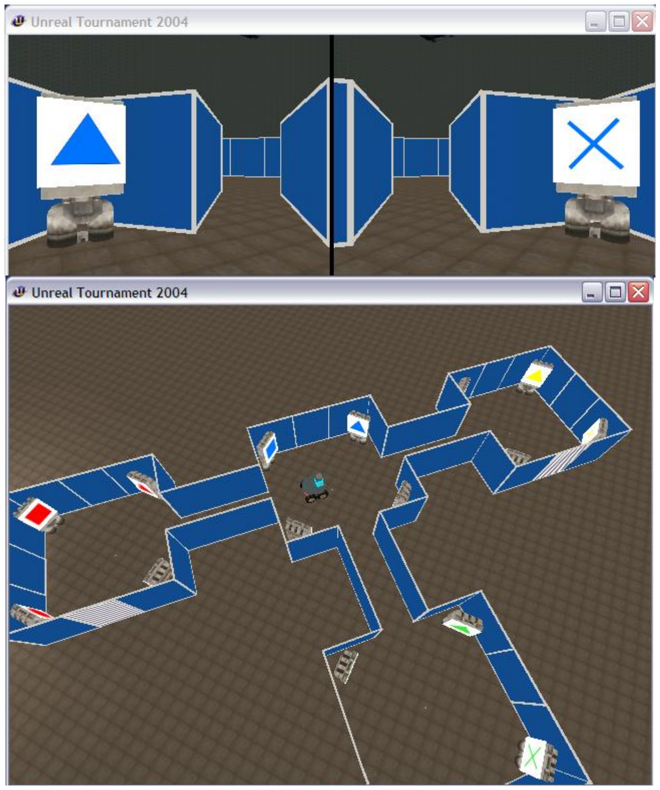


Fig. 3. Top image shows stereoscopic camera view from robot perspective. Bottom image is robot position in a low complexity task.

improvement in the Unreal Tournament game can be realized quickly to the benefit of the C2S2 project. Any new objects and art created for the game can be used immediately.

B. *USARSim*

USARSim was created to be a research and education tool to provide researchers with an easy-to-use robot controller (e.g., Human Robot Interaction [8]) and automation interface. USARSim also makes it easy for students to learn and explore controlling robots. USARSim is currently heavily used in the Robot World Cup Initiative (i.e., Robocup) community---an international community of researchers and educators working to foster intelligent robots research [9]. This is achieved by providing a standardized problem (the Robocup competition) around which a range of technologies can be developed and benchmarked. The Robocup competition has three main components: 1) Robocup soccer, 2) Robocup rescue, and 3) Robocup Junior. USARSim has been developed with these uses in mind so the tool comes with many prebuilt robots, sensors, and arenas. Much of these materials are robot specific such as sensors for sonar distance and laser range finders, for example. Also included is preprioceptive feedback like robot battery power and wheel speed. These are hardly of interest for developing neuromorphic systems yet, they provide an excellent starting point for, say, more biologically relevant proprioceptive feedback (i.e., muscle tension and vestibular).

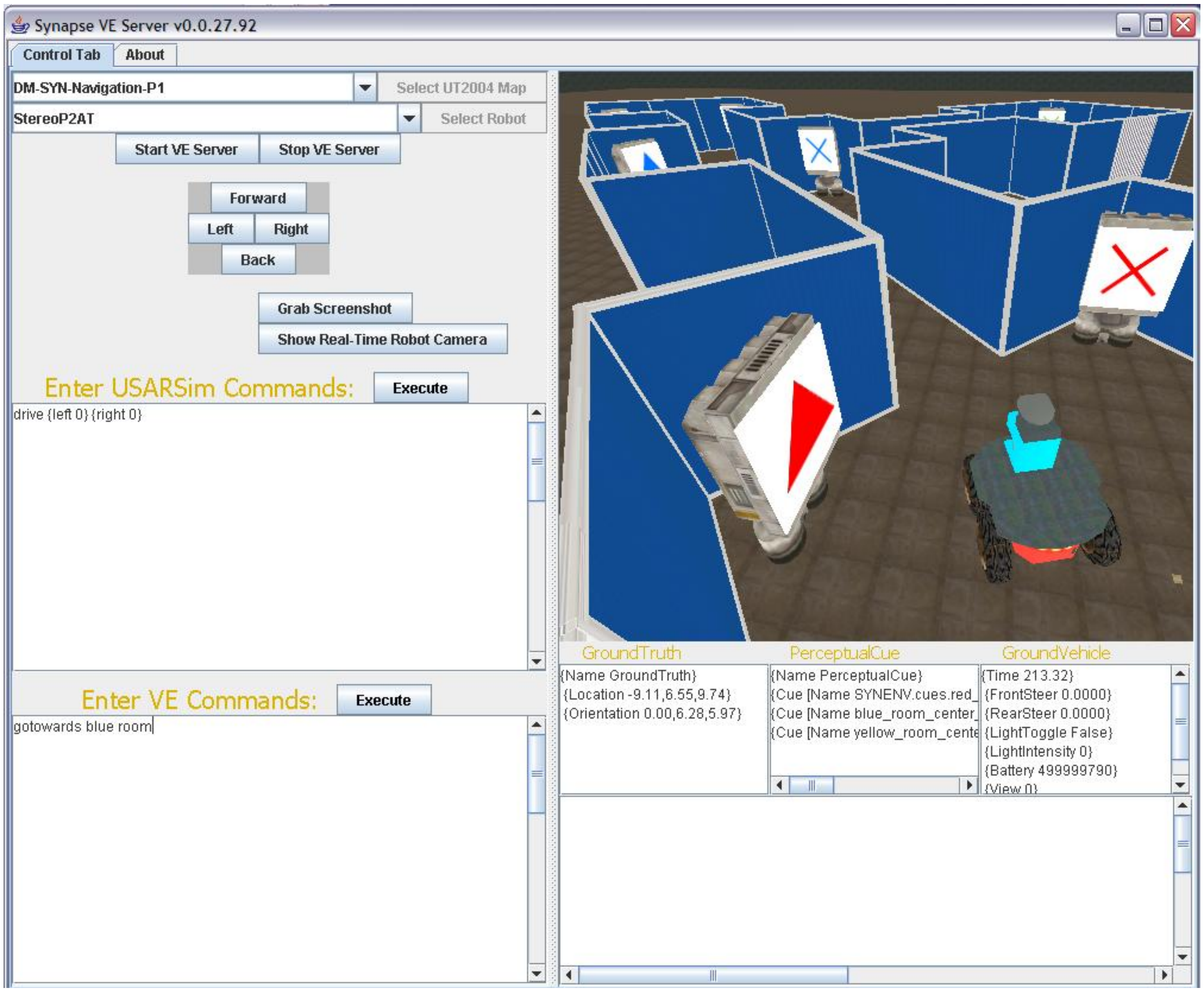
USARSim is already being used for benchmarking robots performing in a search and rescue environment. The National Institute of Standards' (NIST) Reference Test Facility for Autonomous Mobile Robots for Urban Search and Rescue was designed as a physical benchmarking environment. USARSim has the robots, environments, and sensors to recreate this testing facility to a high degree of realism in the virtual world. Realism in an urban search and rescue environment usually includes damaged walls, chairs, and other objects found inside of buildings. It also includes rubble blocked paths, and injured people. The NIST USAR virtual test environment has three levels of increasing difficulty (just like the physical facility) including yellow (i.e., easiest), orange, and red (i.e., most difficult).

Specifically, USARSim uses the Unreal Engine 2.0 through an interface called Gamebots, This interface allows an external application to exchange information, control and monitor, with the engine. While the internals of Unreal Engine 2.0 are proprietary and closed, Epic Games does provide a modding capability for extending "classes" of objects in that run in the engine and the Unreal Virtual Machine. This comes in the form of a Javascript like language called Unrealscript in which objects in the game can be defined and subclassed.

IV. Synapse VE Server

A. *Motivation*

While USARSim is could be used by each team on the C2S2 project individually, the DARPA requirement for project integration required us to create a special layer---i.e.,



DM-SYN-Navigation-P1

StereoP2AT

```
drive {left 0} {right 0}
```

```
gotowards blue room
```

```
{Name GroundTruth}
{Location -9.11,6.55,9.74}
{Orientation 0.00,6.28,5.97}
```

```
{Name PerceptualCue}
{Cue [Name SYNENV.cues.red_
{Cue [Name blue_room_center_
{Cue [Name yellow_room_cente
```

```
{Time 213.32}
{FrontSteer 0.0000}
{RearSteer 0.0000}
{LightToggle False}
{LightIntensity 0}
{Battery 499999790}
{View 0}
```

Fig. 4. Screenshot of the Synapse VE Server GUI. In the upper-left corner the complexity of the task can be selected by the map drop-down list. The robot can also be selected. On the left side of the GUI is all the control functions. The robot can be controlled manually in real-time for testing purposes with the forward, left, right, back buttons. Output is shown on the right with real-time video (third-person or robot camera) and sensor data output at 15 updates per second.

the Synapse VE Server. The purpose of this layer is to provide a simple ready-to-run virtual testing environment where research teams could select task complexity and training runs. Then they would process the input with their part of the system (neuromorphic subsystem) and output control signals to the Synapse VE server. The Synapse VE provides all of the sensors, effectors, environments, and testing experiments needed while gathering all the customized project elements into one place.

Development of the neuromorphic system has necessarily required a layered team approach in which one team develops a base layer (i.e., the VE). Another team takes data from the environment into a retinal model for processing. Another team takes the data from the retinal model into a liquid state machine classifier. Another team takes output from the classifier into a navigation attractor network and so on. As shown in Fig 2, the teams are broken down into general layers to create a software model of the neuromorphic system. Additional layers, details, subsystems, and microcircuitry will be added later.

Another problem with each team creating their own task and training environments is that creating environments is rather complex and time consuming. While Epic Games provides the Unreal Editor (UnrealEd) in order to graphically create environments, the process is really quite complex and error prone. If customized geometry (3D objects) and textures need to be created, it becomes even more difficult requiring a additional set of tools and applications. Each team would probably have to dedicate one person to VE development.

B. *Server Overview*

The Synapse VE Server was designed to provide multiple channels of output from the VE and allow for multiple channels of input (see Fig. 5). This supports ground-truth testing at almost any level of system development. Ideally, the neuromorphic system should be able to transduce raw data into spike-pulse codes and output spike-pulse muscle responses. But, until all the low-level component systems are built this is not a reality. Are we to wait for the entire system to be completed before testing? This seems inefficient. So, the server supports channels of input and output with pre-transformed data. For example, the Synapse VE Server provides a channel of output from the VE that is a raw image showing what the robot is currently viewing. Another simultaneous channel of output is the current visual objects or perceptual cues that the robot is viewing. Any team working on perception and classification can use these channels to test or train their system. Likewise, a team that is working on robot navigation can use the perceptual cues as input and system testing without needing a working perceptual system component to transform the raw image.

C. *Server architecture*

The Synapse VE Server is a stand-alone Java application that was built on a standard client-server model---the

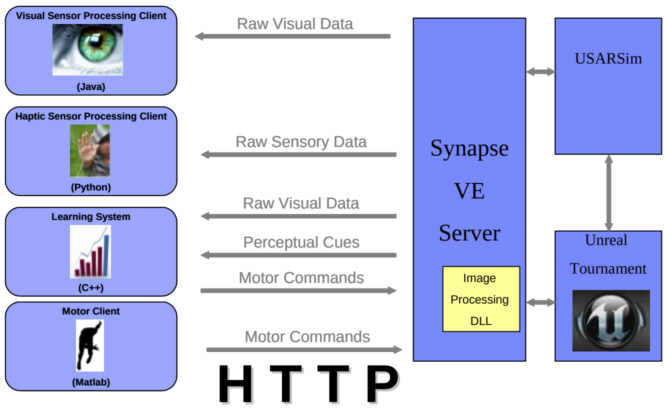


Fig. 5. Synapse VE Server general architecture.

neuromorphic system or subsystem is the client. The client connects to the server via HTTP at port 8080 to send commands in plain text (i.e., post) and receives the requested data. This architecture allows clients to be written in any programming language that can interact with HTTP (most modern languages and systems). This approach also allows the work of the client to be distributed to multiple machines, and also allows for geographic separation of client and server if desired.

A "pull" or request model of data exchange was used as the processing requirements/capabilities of the client would not be known ahead of time. Additional benefits of this model include, 1) off-loading of computation to a separate machine that runs the VE and 2) different APIs are not required for each client language. The Synapse VE Server, however, must run on the same machine as USARSim and Unreal Tournatment as it starts both the Unreal server and client with the needed settings. The server communicates with USARSim through the standard message port (i.e., 3000).

A custom C++ DLL was created to provide raw image data to clients and to improve performance. This was achieved using Hook.dll to pull video data directly from the Microsoft DirectX framebuffer for the Unreal Tournament client thereby capturing the robot camera. The Java Native Interface provides a way for the Synapse VE Server to interface with the custom DLL. Performance tests show that 640x480 24bit image arrays can be captured as fast as 25 times per second (fps) or near real-time. Improvements will be required in this performance as the retinal model becomes more sophisticated. Given that there are about 126 million rods and cones in the human retina, the spatial resolution will have to be much higher.

Thus far, only one customized sensor has been created to detect visual (perceptual) cues in the VE. This sensor, called the *perceptual cue sensor*, was written in Unrealscript as a subclass of the USARSim Sensor class. It was added to the StereoP2AT robot in the USARBots.ini configuration file. This sensor runs in the game engine and scans through all the visible (i.e., the robot's field of view) staticmeshes (i.e., 3D objects) and reports back on the ones with the keyword *cue* in the *label* field. Thus, one can make any staticmesh in the environment a visual cue that is reported by the perceptual cue sensor simply by changing the label to have the word *cue* somewhere in the string. The perceptual cue sensor also computes the angle off center, the absolute x,y,z location, and the x.y.z distance of the visual cue. This information is included in the sensor message that is returned. Figure 3 shows the robot field of view on the top for each of the two cameras and its location in the environment on the bottom.

One of the task environments is shown in Fig. 3 on the bottom. This indoor maze in with four rooms and corridors was created using a set of wall "objects" and other staticmeshes. This environment provides a template to quickly construct a maze with any number of rooms and branches. The Unreal Editor allows whole branches and sections to be selected, copied, and pasted. Also, the environments team can take any image file and put it in the maze as a perceptual object. Thus, almost any type of task or level of complexity can be rapidly created using this template environment.

D. *Interface*

The server has two interfaces. The first is a standard GUI that supports debugging and monitoring for the server (see Fig. 4). The second is a web-based interface that supports remote manual and programmatic control.

The GUI displays a real-time robot camera view at the upper right by acquiring video from the game engine's display window. Movement commands (left, right, forward, back) are used to move the robot. Both the GUI window and the game engine's display window are dynamically updated. The GUI can also be used to send specific commands to the robot and the real-time environment through the two command windows. To send USARSim Commands, the user enters commands in the appropriate window and presses "Execute" (for a list of all commands, see the USARSim manual. To send specific VE Commands, the user enters commands in the window and presses "Execute" (commands include: "face <cue>," which instructs the robot to rotate until it is facing the named perceptual cue; and "gotoward <cue>," which instructs the robot to rotate as above and then proceed toward the named perceptual cue). The GUI also displays sensor information, including "Ground Truth," which displays the robot location (in meters) and orientation (in radians) with respect to the (x,y,z) axes; "Perceptual Cue," which is a list of all labeled cues in the robot's unobstructed field of view with location of each cue (x,y,z) in the environment; and "Ground Vehicle," which contains additional detail on various parameters related to the robot's operation and state.

A web server embedded in the Synapse VE Server allows commands to be sent to the robot in the VE via **http** through a browser window or programmatically. To use this interface, the user launches a web browser and types the following URL: `http://{host_name}:8080/`. The response "welcome to the Synapse VE Server" indicates the server is working correctly. Commands are passed to the robot using the following request syntax: `http://{host_name}:8080/command` where "/command" can be any one of the following: `/robot` to obtain details about the robot, such as location and orientation. Location is given in meters and orientation is in radians on the (x,y,z) axes; `/cues` to obtain a formatted list of all the labeled cues in the robot's unobstructed field of view; `/image.png or /image.jpg` to fetch either a PNG or JPEG image from the current game engine display window; `/forward, /backward, /left, /right or /halt` to move the robot forward, backward, left or right—note that motion continues until a halt command is issued.

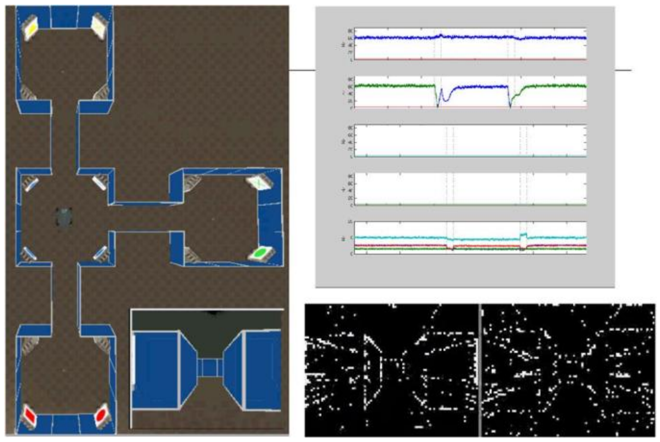


Fig 6. Testing of retinal model and spiking navigation systems in the VE with a low complexity four room/state task. The response of spiking neurons for navigation is shown right-top and the retinal spike response is shown right-bottom.

V. VE Task Complexity

Sensory-motor interaction with an evolving, changing environment is a key to intelligent behavior, as all intelligent must be both situated and embodied. The behavioral tasks for the neuromorphic system fall into three broad cognitive categories, highlighting problems of perception, planning, and navigation. Though the original proposal outlined three separate environments (one for each category of task), we revised the plan such that a single 3D virtual world was used to develop tasks that highlight each of the three kinds of problems. This change made for a uniform and elegant conceptual design of the VE. All tasks are conceived as traversals on state-space graph, with perception as state identification based solely on the current state, navigation as action selection based on the current state and previous states, and planning as action selection based on prediction of the consequences of future possible actions. In the state-space framework, all tasks are versatile, extensible, indefinitely scalable in complexity, and are amenable to objective, quantitative, and comparative performance evaluation. The tasks can be extended to provide interaction over a wide range of space and time scales, and can offer comparison to behavioral studies.

Figure 6 shows a relatively simple task environment with only four states (decision points) shown by the four interconnected rooms. The retinal spiking model takes the raw image data and transforms it into edges and perceptual objects (right-bottom part of Fig. 6). The attractor spiking network takes the perceptual objects as input (spiking input) to detect a) current state and b) next desired state. The attractor network response is shown in the right-top part of Fig. 6.

While the current maze-like task environment appears somewhat artificial, it may be desirable in future work to allow the actual VE to take on different visual characteristics of other, more realistic-naturalistic environments. We have formulated an approach whereby the graph traversal formalization (and its attendant benefits of comparability and quantification) can be maintained and applied to more naturalistically-rendered environments. Such is diagrammatically illustrated in Fig. 7. In this an environment, task complexity is conceived in terms of three dimensions: the number of different perceptual states processed by the agent, the degree of memory (history) and/or prediction (anticipation), and the level of symbolic abstraction involved in the perception-action relation. Actual traversals would involve tasks of systematically-varying perceptual difficulty and complexity, with local and global orientation cues made available or obscured in a controlled (and perhaps dynamic) manner.

VI. Conclusion & Future Work

Our goal as part of the Environments and testing team was to create a benchmarking and training environment to support all other teams in the development of their components of the neuromorphic system. Our mission was that the environment had to support

1. fast environment development
2. accelerated training
3. incremental testing
4. benchmarking
5. wide range of environment fidelity
6. wide range of task complexity

We have made significant strides in achieving fast VE development, incremental testing, and high VE fidelity given the use of the USARSim framework. Because parts of the task environment can be copied and pasted, components are be reused and new task configurations can be quickly created. Environment development is clearly faster than using a physical test facility. VE fidelity too can be easily

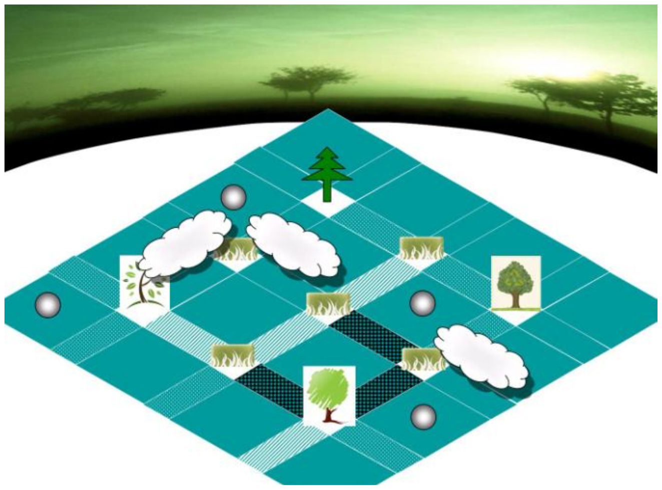


Fig 7. Illustration of a possible “gameboard” structure to underlie future VE renderings of more naturalistic environments while preserving the formalization of state-space graph transitions.

decreased or increased depending on what would provide a challenge for any part of the neuromorphic system. For example, the lighting can be manipulated so that there is no directional lighting or shadows (low fidelity) or there is only directional lighting with many different types of lightsources (high fidelity). Having this capability means that incremental testing is possible.

Still, further technical work is needed to increase fidelity in certain areas. One such area includes increasing the number of input channels to more than just vision and some types of proprioceptive feedback. This would mean adding sensors for hearing, touch, olfactory, and taste channels. Also, these modalities may have to be added to the Unreal Tournament environment as it was not designed to provide olfaction simulation, for example.

This would also mean that the data rate of information for each of these channels was high enough to simulate real-world stimuli. As stated earlier, the current image data rate is only a 640x480 24bit pixel array 25 times per second. That is merely 23 Mbps---a bitrate that is only a small proportion of the data entering the eye. To improve performance the system architecture may need to be redesigned with a faster UDP-based protocol such as RTSP instead of HTTP.

Accelerated training is the least known among all the goals. Technical research needs to be conducted to see 1) if the Unreal Engine can be accelerated and by how much, 2) how USARSim sensors and other components can be accelerated in a unified and consistent manner, 3) how this acceleration interacts with performance characteristics of the server hardware, 4) how much can the input and output data rates be accelerated?

We have defined the environment in terms of transversals in a state-space graph. This graph then could lead to a method for measuring task complexity. However more theoretical work is needed before a quantitative measure of task complexity can be achieved. Also, providing a quantitative measure for task complexity would be central to efforts at benchmarking neuromorphic system performance.

Finally, future work will be aimed at creating a player in the VE that resembles a mammal rather than a robot. The mammal should have articulated joints with fairly realistic skeletal muscle control. This will allow for the development of neuromorphic motor cortex and cerebellum subsystems.

Acknowledgment

We would like to thank Stefano Carpin and the USARSim team for development support and suggestions.

References

[1] Simon, H. A. 1961. Modeling human mental processes. In *Papers Presented At the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference* (Los Angeles, California, May 09 - 11, 1961). IRE-AIEE-ACM '61 (Western). ACM, New York, NY, 111-119.

[2] Rumelhart, D.E., J.L. McClelland and the PDP Research Group (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, Cambridge, MA: MIT Press.

[3] Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain,' Psychological Review, 65: 386-408" (November, 1958).

[4] Hebb, D.., Organization of Behavior, Wiley, 1949 .

[5] Rochester, N., J. H. Holland, L. H. Haibt and W. L. Duda, Tests on a Cell Assembly Theory of the Action of the Brain Using a Large Digital Computer,' IRS Trans. on Information Theory, IT-2; #3, (September, 1956) .

[6] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, C. Scrapper (2007). USARSim: a robot simulator for research and education. Proceedings of the 2007 IEEE Conference on Robotics and Automation, pp. 1400-1405.

[7] Epic games, http://www.unrealtechnology.com/

[8] M. Lewis, J. Wang, and S. Hughes (2007). "USARSim : Simulation for the Study of Human-Robot Interaction", Journal of Cognitive Engineering and Decision Making, (1)1, 98-120.

[9] Robot World Cup Initiative, http://www.robocup.org