University of Nevada, Reno

Brainstem: A NeoCortical Simulator Interface for Robotic Studies

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science with a major in Computer Science

by

Qunming Peng

Dr. Frederick C. Harris, Jr., Thesis Advisor

December 2006

We recommend that the thesis

prepared under our supervision

by Qunming Peng

entitled

Brainstem: A NeoCortical Simulator Interface for Robotic Studies

be accepted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

_____

Frederick C. Harris, Jr., Ph. D., Advisor

_____

Monica Nicolescu, Ph. D., Committee Member

_____

Philip Goodman, M.D., Graduate School Representative

_____

Marsha H. Read, Ph. D., Associate Dean, Graduate School

December 2006

Abstract

"A Hybrid Neuromorphic/AI Socially Interactive Robotic Sentry" is an ongoing project that was proposed by Goodman Brain Computation Lab, University of Nevada, Reno. The goal of this project is to build neuromorphic systems that could make decision in real-time based on brain physiology. The entire system will include the following components. First, NCS, running on a Remote Beowulf Cluster, is the software that provides the computation power to run the cortical simulation. Secondly, AIBO, robotic dog provided by Sony. Third, a laptop with a wireless adapter, which contains the AI components and models brainstem functionality. We call this last component Brainstem. This thesis will talk about the development of the software for Brainstem.

Dedication

To my parents: Peng Zhirong, Huang Sanying

and my wife: Dan Xu

Acknowledgements

I would like to thank my advisor, Dr. Frederick C. Harris, Jr. for the advice, support and technical guidance throughout this work.

Also I would like to thank Dr. Goodman for providing such a great opportunity to work on this project in the Brain Lab. His time spent in discussion and instructions is greatly appreciated.

In addition, thanks to Rich Drewes for helping me a lot, especially on Linux stuff.

Finally, I would like to thank Dr. Monica Nicolescu for serving on my thesis committee.

Contents

List of Figures

List of Tables

Chapter 1: Introduction

It has been five years since the development of first version of the NeoCortical Simulator (NCS) in Goodman Brain Computation Lab at the University of Nevada, Reno. Many components and functionality have been added to NCS. We are now developing applications using NCS. It was in this context that “A Hybrid Neuromorphic/AI Socially Interactive Robotic Sentry” was proposed. Its goal is to make a robotic dog which could make decision real-time based on brain physiology. We could look at the robotic dog as body and NCS as the brain. Therefore, a system was needed to connect the robotic dog and its brain. This is where Brainstem fits.

Brainstem, in physiology, is relay center between the brain and the body. It is responsible for collecting signals from all over the body and sending commands from the brain to the body. Meanwhile, it can make some low-level decisions itself, such as waking up. The Brainstem in our project is designed to function in the same fashion as the real brainstem. It acts as the relay center between NCS and the robotic dog.

This thesis is organized as follows: the background is given in Chapter 2. Then program design is presented in Chapter 3 and the graphical user interface is introduced in Chapter 4. At the end, conclusions and future work are given in Chapter 5.

Chapter 2: Background

Since the Brainstem is a relay center between the NCS and the robotic dog, some background material is needed. This chapter presents background on robotics, neural biology, NCS, the robotic dog and the physiology of a brainstem.

2.1 Robotics

2.1.1 Overview

Encyclopedia Britannica defines robotics as the “Design, construction, and use of machines (robots) to perform tasks done traditionally by human beings.” Robots are already widely used in industries such as automobile manufacturing to do simple repetitive tasks. Recently, research in Robotics has been productive. Its application domain has been extended to entertainment [1], fire fighting [2], law enforcing [3], and even medical surgery [4], etc. In a word, Robotics play more and more important role for human beings.

2.1.2 Artificial Intelligence (AI)

At the early development of Robotics, due to the simplicity of tasks, traditional control methods were enough to make the robots do what we wanted them to do. With the need to work in unknown environments, the robot has to adapt to the environment and to be capable of learning new knowledge. Most aspects of current robotics involve artificial intelligence. The most typical AI theories are neural networks (NN), fuzzy logic (FL), evolutionary computation (GA – genetic algorithms) and the hybrid forms of them.

Neural networks [5] represent massively parallel distributed networks with the ability to learn through interaction with the environment. It is adaptive and self-organizing. NN are already applied to the control of non-linear and time-varying systems that contain uncertainties.

Fuzzy control systems [6] based on mathematical formulation of fuzzy logic have the capability of representing human knowledge as a set of fuzzy rules. The fuzzy robot controllers use human knowledge in the format of if-then rules, while the fuzzy engine computes the proper control action.

Genetic algorithms represent an approach to obtain global optimization based on the mechanics of natural selection and natural genetics [7]. For example, the hierarchical trajectory generation method [8] provides efficient control for robots by optimizing the total energy of all actuators.

All of the methods mentioned above were verified to be efficient at controlling the motion of a robot, usually based on some complex model. There are still needs to make robots have intelligence in order to handle real world problems, especially having memory and learning ability. So recently research has begun to focus on biological neural systems.

2.1.3 Hybrid Neuromorphic / AI

The term neuromorphic was coined in the late 1980's by Carver Mead: "systems that are based on the organizing principles of the nervous system" [9]. Because a neuromorphic system is based on a biological neural model it is more robust and consistent in dealing with real-world problems than a pure Artificial Intelligent system.

"A Hybrid Neuromorphic/AI Socially Interactive Robotic Sentry" is an ongoing project that was proposed by Goodman Brain Computation Lab. Its goals are to build neuromorphic systems to exhibit and explain the following three aspects of intelligent behaviors [9]:

1. Socially meaningful interactions with animals and humans
2. Robustness: to variability in the timing of responses by animals, to changes in the sensory environment, and to competing perturbations and response options
3. Decision making in real-time based on brain physiology

In the growth of a child, usually he or she will learn more and more from the world around them, including nature, people, and books. Although we are still not clear how the brain develops, we do believe that the tutor or supervisor plays an important role in the learning process. It is kind of like Artificial Intelligence. The brain needs to learn supervised by AI. This is why our project focuses on Hybrid Neuromorphic / AI.

As shown in Figure 2.1, the entire system will include the following components:

* NCS, short for NeoCortical Simulator, is software running on remote beowulf cluster. It is the actual brain of AIBO. It's able to communicate via the Internet with the other components.
* A robotic dog provided by Sony, we call it AIBO
* A laptop with a wireless adapter, which contains the AI components and models brainstem functionality, we call it Brainstem.

NCS and AIBO will be discussed in more detail in the following sections. After that, the physiology of Brainstem will be presented. The Brainstem's software design and development will be presented in the next two chapters.

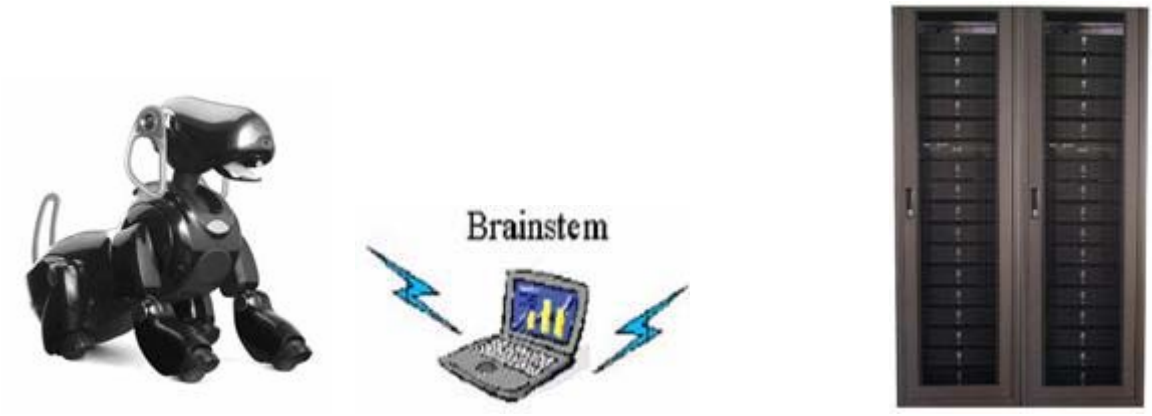

Figure 2.1 Compartments of Entire System

2.2 NeoCortical Simulator (NCS)

2.2.1 Overview

NCS is the simulator of a brain based on a biologically realistic neural network. It has been under development in the last 5 years [10]. Basically, NCS is a biological neural network. It is based on the biology of brain, so we need to review some of that.

2.2.2 Biology

Neurons

Figure 2.2 shows the components of a neuron. The basic component of brain is neuron, which is a unique type of cell found in the brain and body that is specialized to process and transmit information. A typical neuron consists of a cell body (soma) and many extensions called dendrites. It is primarily the surfaces of dendrites that exchange chemical information with other neurons. There is a long dendrite different from all the others, it is called an axon. The purpose of an axon is to transfer electro-chemical signals

to other neurons, sometimes over a long distance. Usually the axon is protected by a myelin sheath. The exchange of information to other neurons happens at the end of axon, which is named axon foot or bouton.

Between the ending of axon and the dendrite of other neuron, there are tiny gaps called the synapse, which will be discussed later.

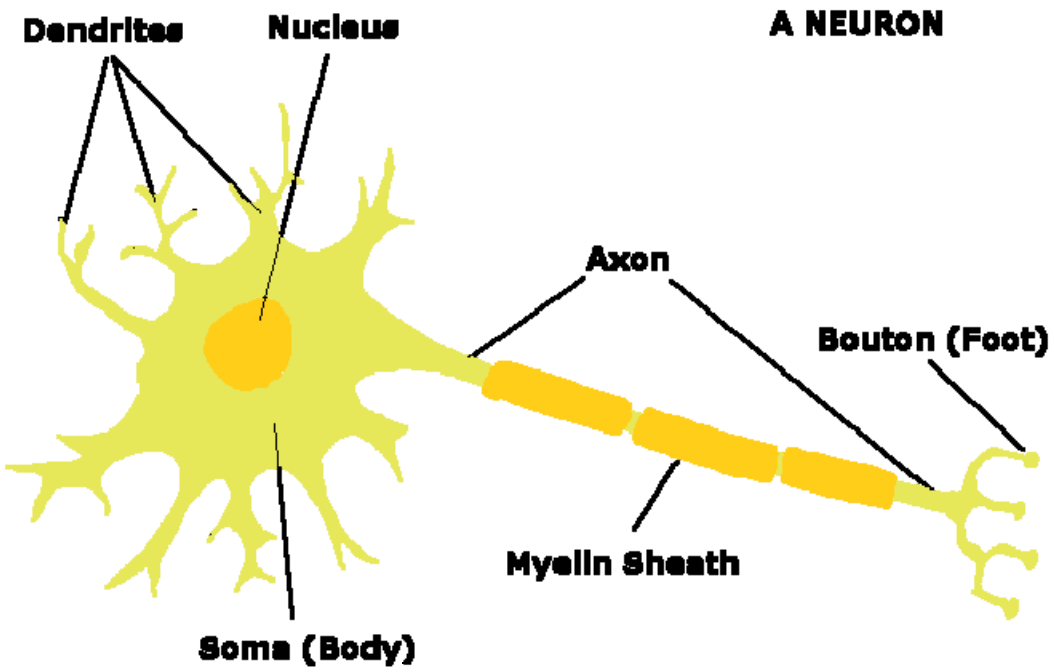

Figure 2.2 A Neuron [11]

Action Potential

When chemicals contact the surface of a neuron, the balance of ions inside and outside of the membrane of cell is changed. Once the change reaches a threshold value, it crosses the cell's membrane to the axon, causing an action potential there.

On the surface of the axon, there are lots of channels which act like gates. They open when action potentials come across and allow ions to come in and out. This produces a transfer of energy called an action potential which is illustrated in Figure 2.3.

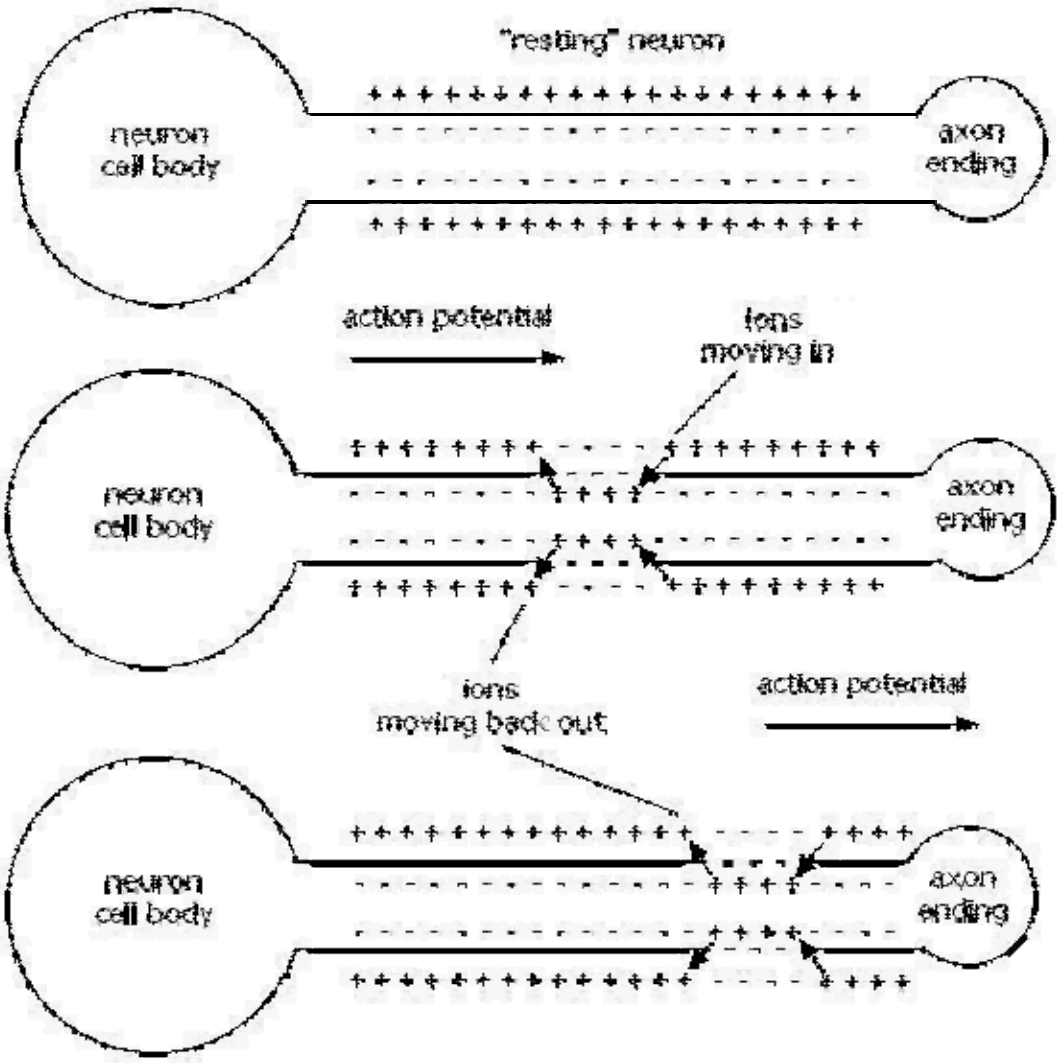

Figure 2.3 Action Potential [11]

Synapse

Information from one neuron flows to another neuron across a synapse. The synapse is a small gap separating neurons.

Figure 2.4 shows a synapse between the axon of a neuron and the dendrite of another neuron. It consists of:

1. A presynaptic ending that contains neurotransmitters, mitochondria, and other cell organelles,
2. A postsynaptic ending that contains receptor sites for neurotransmitters and,
3. A synaptic cleft or space between the presynaptic and postsynaptic endings.

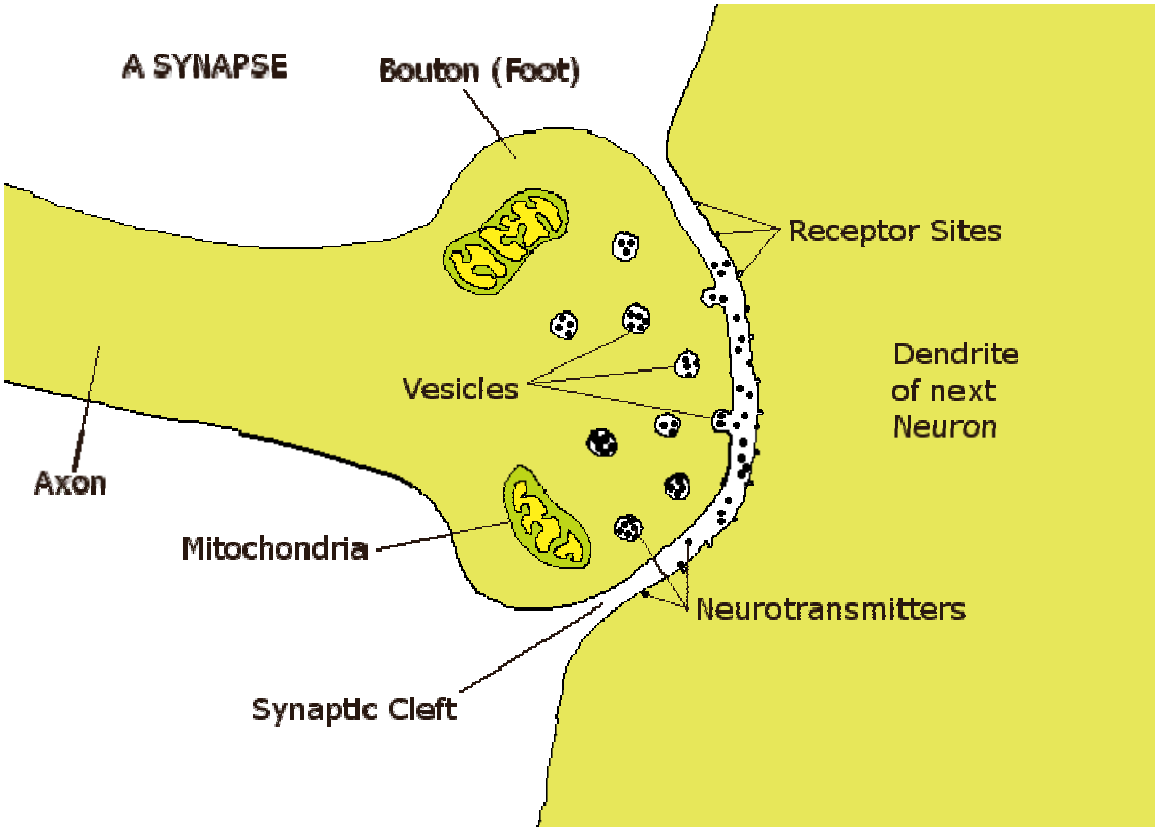

Figure 2.4 A Synapse [11]

2.2.3 Applications

Many optimizations have been made to NCS since its first version. The latest version NCS5, is fully developed to simulate the biology of the brain, including neurons,

membrane channels, synapse dynamics such as facilitation, depression, synaptic augmentation, and Hebbian STDP (spike-timing dependent plasticity) learning [12 - 17].

So first of all, we used the NCS to simulate experiments which are hard to do in the real-world. One experiment did pattern reorganization on an audio signal [18]. One simulated biological experiment took a look at how spike-timing and membrane dynamics of biological neurons may encode information [19, 20]. Figure 2.5 [20] shows a simulated experiment which replicates the behavior of various neurons: classic non-accommodating (cNAC), classic accommodating (cAC), bursting non-accommodating (bNAC), bursting accommodating (bAC), delayed non-accommodating (dNAC), delayed accommodating (dAC), classic stuttering (cSTUT), and bursting stuttering (bSTUT).

Secondly, applying NCS to establish intelligent system has been studied for a couple of years. The robotic project CARL was a prototype attempt [21, 22]. Due to the weight and size issues, it is hard for robots to carry a multiple-processor supercomputer. Inaba presented a robot using wireless communication to gain the benefits of faster, more complex processing from a multiprocessor system [23].

CARL went further by expanding the two-system model into a three-layer hierarchal robotic control system. These layers were the Body, the Brainstem, and the Cortex that associate their behaviors with corresponding biological structures. The configuration is depicted in Figure 2.6. The body layer corresponds to the robot body, which interacts with environment and collects original stimulus. The Brainstem layer is a local machine which is in charge of some instinctive response and relaying information between body and cortex. The Cortex layer corresponds to remote computer cluster which provides powerful computation for making decisions.

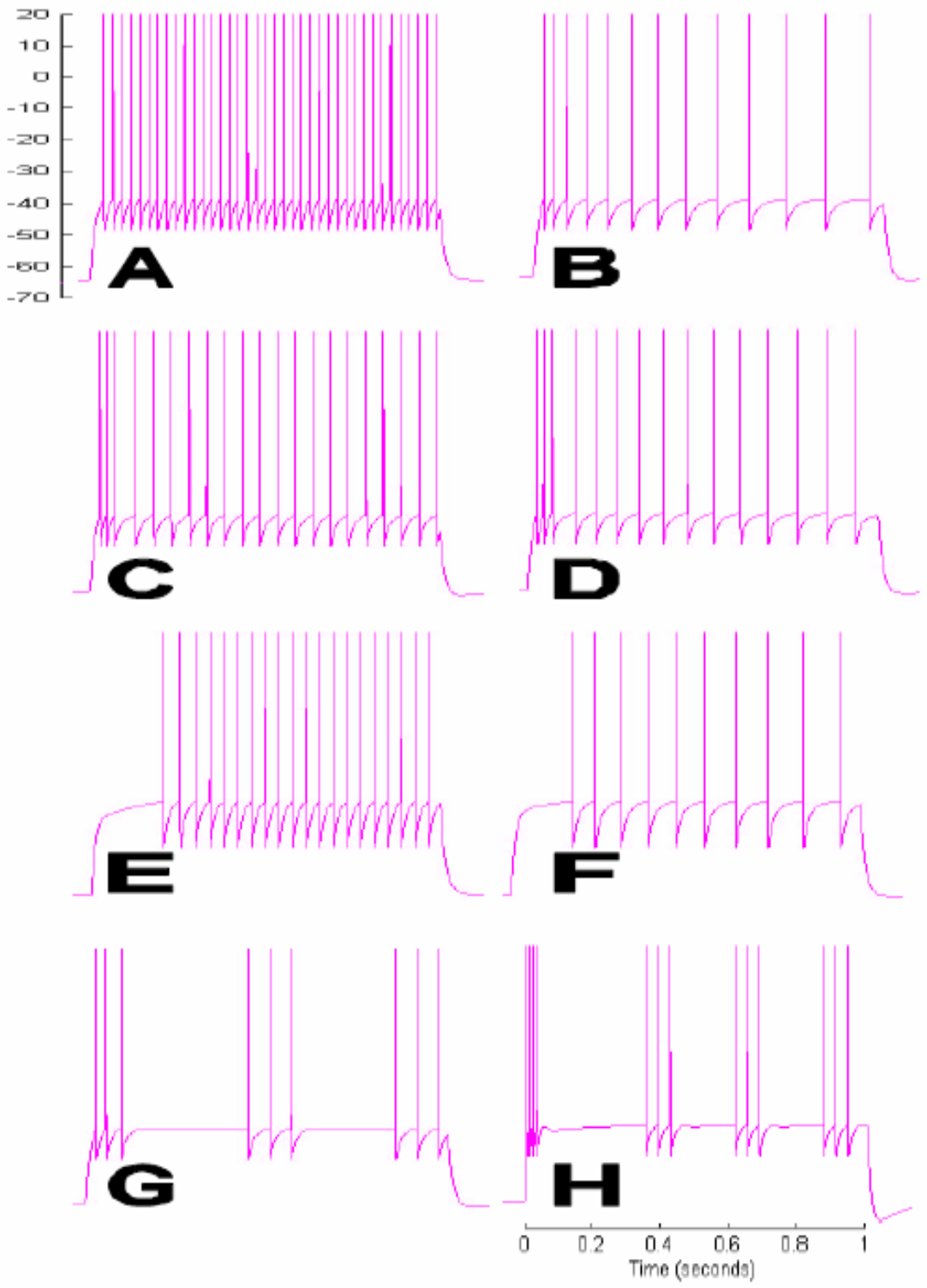

Figure 2.5 Simulated response to step current of 150-300 pA [20].
A. cNAC B. cAC C. bNAC D. bAC E. dNAC F. dAC G. cSTUT H. bSTUT

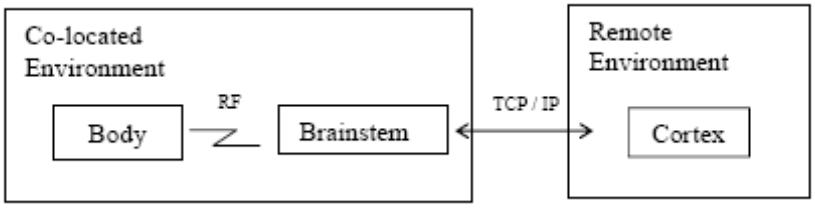

Figure 2.6 The hierarchical robotic system concept [22]

In the CARL project, the robot was built by our lab and was just a prototype. We needed a much more suitable robot to satisfy the criteria listed earlier.

Finally we found AIBO from SONY. This is a well-developed robotic dog. It will be introduced in more detail in Section 2.3. Also, in the CARL project, the Brainstem layer was just a prototype and needed to be fully developed.

2.3 AIBO

The AIBO robot is the name which Sony has given to its family of robots that are designed with the goal of interacting with human being. The name itself is a play on the words “artificial intelligence” (AI) and “robot”. The AIBO robot combines a body (hardware) and software that allows it to move and display the lifelike attributes of emotion, instinct, and learning. The configuration of hardware is as follows and shown in Figure 2.7 [24].

1) Stereo microphones: allow the AIBO to listen to the surrounding environment
2) Head distance sensor: measures the distance between the AIBO and other objects
3) Color camera: detects the color, shape, and movement of nearby objects.
4) Mouth: picks up toys and expresses emotions.

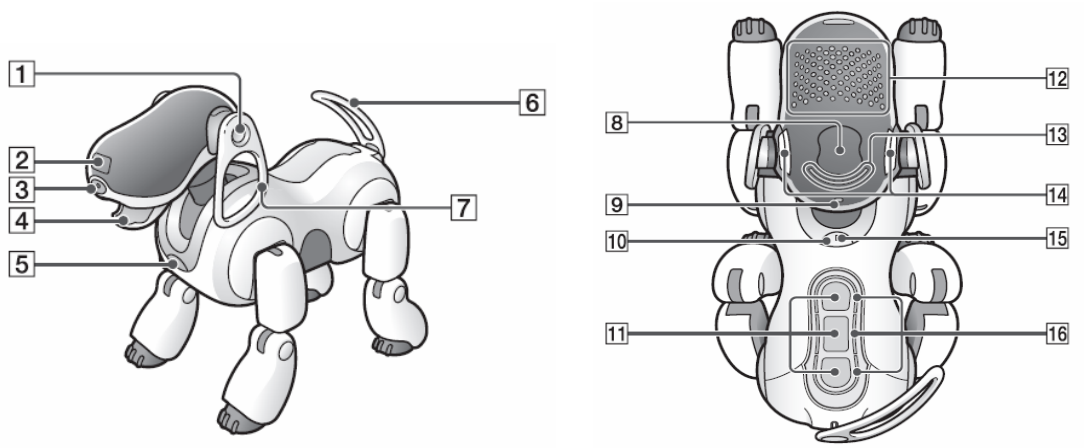

Figure 2.7 Hardware of AIBO [24]

5) Chest distance sensor: measures the distance between the AIBO robot and other objects.
6) Tail: moves up, downs, left and right to express the AIBO's emotions.
7) Ears: Indicate the AIBO's emotions and condition
8) Head sensor: detects and turns white when you gently stroke the AIBO's head.
9) Wireless light (on the back of AIBO's head): blue light for use with the wireless LAN function.
10) Pause button: when pressed, the AIBO activity will pause or resume
11) Back sensors (front, middle, and rear): detect and turn white when you gently stroke the AIBO's back
12) Face lights: these lights turn various colors to show the AIBO's emotions and conditions.
13) Head light: detects and turns white when you touch the head sensor. Turns orange when an AIBO's joint is jammed.

14) Mode indicators (inner side of ears): these indicate the present mode and condition of the AIBO robot

15) Operation light: During operation: turns green.

16) Back light (front, middle, and rear): detect and turn white when you gently touch the AIBO's back sensors.

The software structure in an AIBO is shown in Figure 2.8. It is developed using

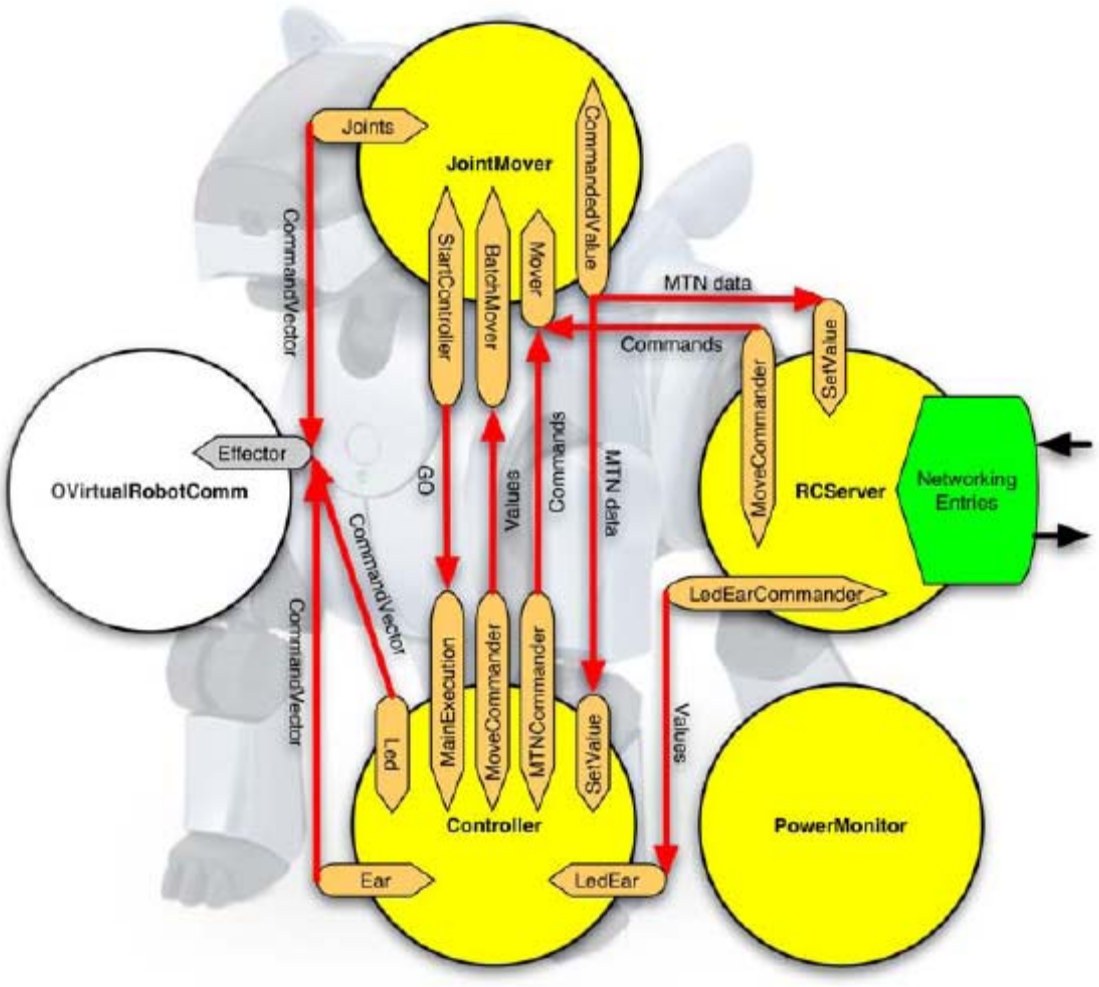

Figure 2.8 Software of AIBO [24]

the OPEN-R SDK which is also provided by Sony. It consists of some objects built with an Object Oriented Design. The main object is called Controller, which controls ear, LEDs and the execution of commands. The JointMover object is in charge of the movement of AIBO. RCSever is an object where we could customize the interface with an outside program, which communicates with the AIBO through network communication.

2.4 Brainstem

In physiology, the brainstem is the lower extension of the brain where it connects to the spinal cord. The brainstem can be broadly divided into the Midbrain, the Pons and the Medulla Oblongata which is displayed in Figure 2.9. Neurological functions located in the brainstem include those necessary for survival (breathing, digestion, heart rate, blood pressure) and for arousal (being awake and alert). It also relays the information between the brain and the body.

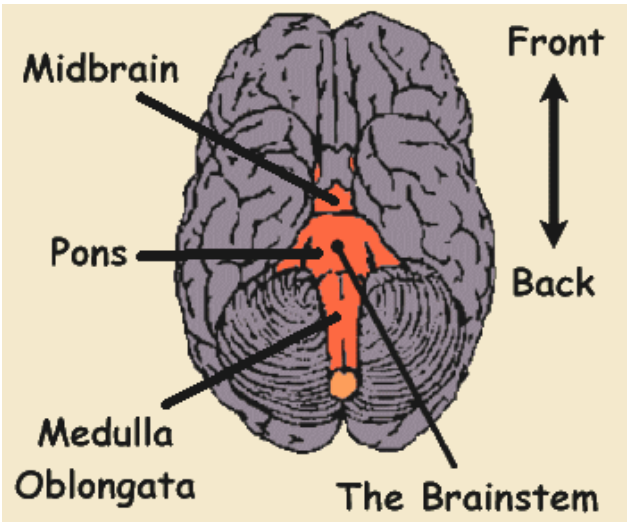

Figure 2.9 Brainstem [25]

According to the three-layer structure for robotic system in Figure 2.6., the brainstem layer is needed for connecting the body (robot) and the cortex (NCS), which will give it functionalities similar to the real brainstem. These are neurological functions for arousal and relaying information between the body and the brain. The next chapter will describe the software design of Brainstem.

Chapter 3: Program Design

3.1 Overview

The whole Brainstem program consists of two parts: the background program and graphic user interface. The background program is designed to implement the functionality for basic control of the AIBO and relaying information between NCS and AIBO. The user interface will be used to assist the user in knowing the running status of the background program and controlling it to some extent. In this chapter, we present several parts of the program design from the initial scenario to software design, final implementation of background program.

3.2 Initial Scenario and Software Design

The original scenario for AIBO has several steps, they are:

1) AIBO resting on-guard

2) Ambient noise or motion

3) AIBO rises to attention

4) AIBO cortex processes visual activity for several seconds

5) AIBO cortex reaches decision on motor action; performs

6) AIBO receptive to reward (stroking of back) for several seconds

7) Human rewards (or not)

8) AIBO resumes resting on-guard state

3.2.1 Sequential Structure

Based on the scenario mentioned in the last section, we designed a flowchart shown in Figure 3.1. At the beginning, an AIBO stays waiting for any audio noise or apparent video difference. As soon as it detects some audio noise or apparent video difference, it begins to grab video clips and process them using a Gabor filter to produce stimuli. The stimuli are sent to NCS though network communication. After that, Brainstem starts a waiting timer. While the NCS is doing the computation, the AIBO could only wait without doing any necessary action. When the waiting timer expires, Brainstem reads the generated report from the NCS and parses it to know what command should be sent to the AIBO. Then command is sent to the AIBO giving it some action to do. As soon as the AIBO finishes the execution of command, Brainstem starts to listen to the back sensors on the AIBO. If some stokes happens, the NCS will be rewarded. Otherwise, Brainstem makes the AIBO go to the initial waiting stage.

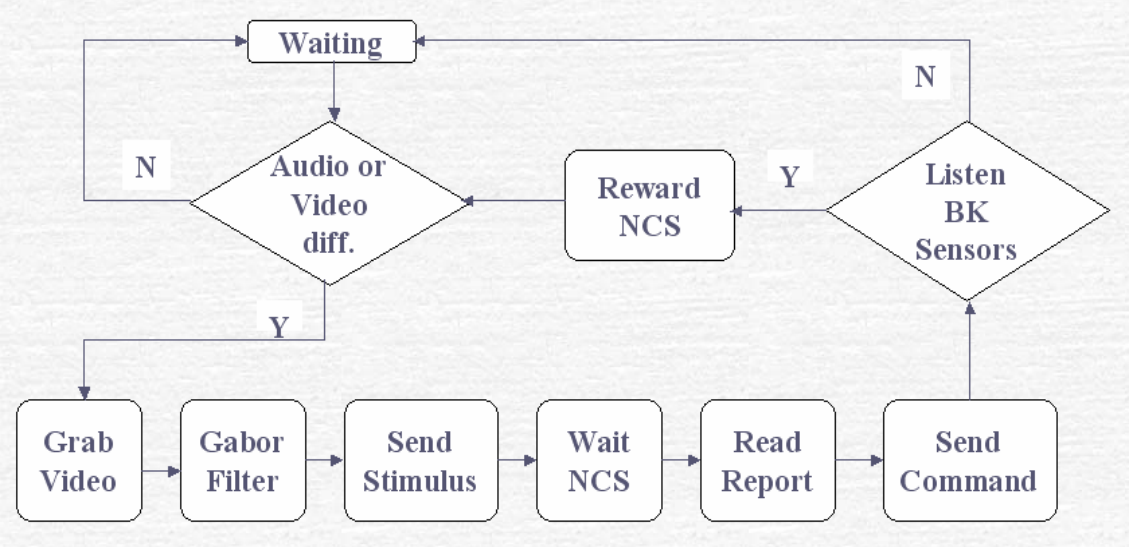

Figure 3.1 Flowchart of software structure

Using the flowchart, the initial version of Brainstem using Matlab was implemented during the summer of 2005. There are some limits of the initial version:

1) Its structure is sequential, so at each block it can only do one thing. For example, in the block of WaitNCS, it cannot collect any sensor signal.
2) It read an image from AIBO and saved the image to hardisk before processing. So it took a lot of I/O time.

3.2.2 Multiple-threaded Structure

The corresponding functionalities for Brainstem to implement for each step in the original requirements are list following.

1) Control AIBO to resting state
2) Capture video and sound from the outside environment
3) Control AIBO to alert state
4) Process video signal and send the results to NCS for post-processing
5) Receive decision from NCS
6) Receive signal from touch sensor and process it
7) Nothing for Brainstem
8) Control AIBO to resting state again

So overall, the functionalities for Brainstem is to control AIBO's action; collect video, audio, touch signal from AIBO and process them; Send data to NCS; Read decision from NCS.

Moreover, due to the limitations of sequential structure, a multiple-threaded structure was developed. Each block in Figure 3.2 is an individual thread. The SigView block is added to show the original signal collected from AIBO. Since all the blocks are different

threads, technically they could run at the same time. This will avoid the limitations of the sequential structure.

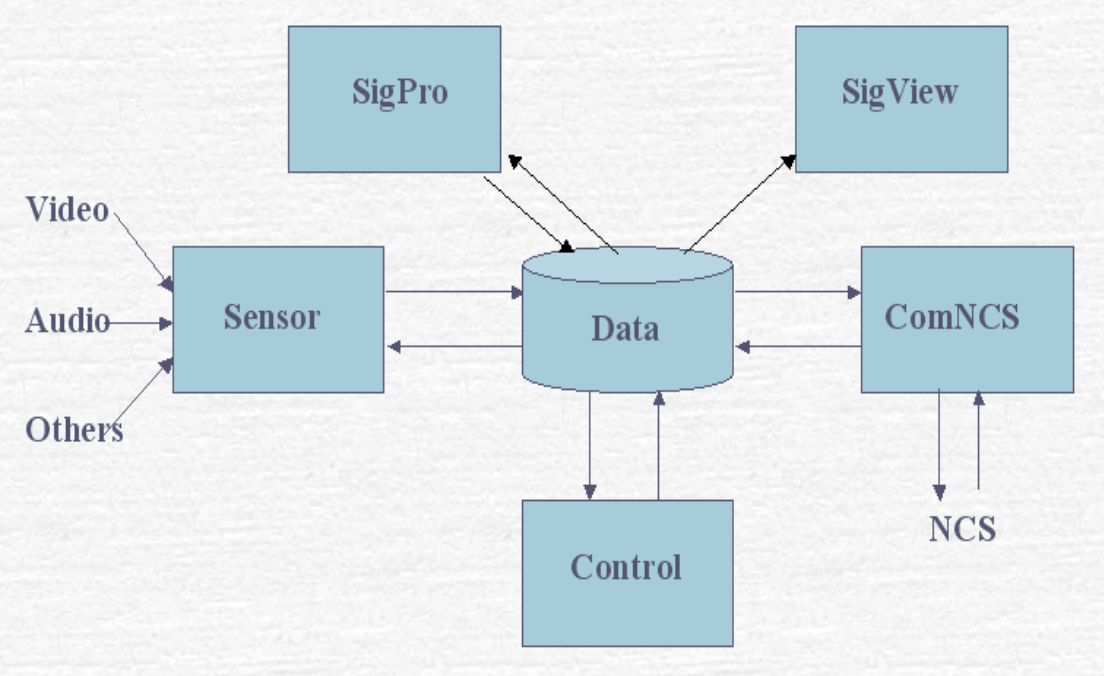

Figure 3.2 Multiple-threaded Schema

3.2.3 System Schema based on Pipe-line Model

To be more modular, the structure mentioned in last section was redesigned in more detail using a Pipeline Model [26]. An automotive assembly line is a classic example of a pipeline. Each car goes through a series of stages on its way to the exit gates. At any time many cars are in some stage of completion.

In Figure 3.3, there are four pipelines. For the sensor signals, it will go though the SENSOR module, PROCESS module and SENDSTIMULUS module, exit to BCS (Brain Communication Server), which is a server program attached to NCS. As for the control flow, reports from NCS will be read, parsed and sent to AIBO. All queues, including the SensorDataQueue, the StimulusQueue and the ReportQueue are self-blocking queues. It

means a queue will block the reading threads if the queue is empty, until there is some element available. And also, a queue will block the writing threads if the queue is full, until it is not full. Since the most time-consuming module PROCESS could be running while the Brainstem communicate with BCS and AIBO through network ports, overall computational efficiency is improved over the sequential structure.

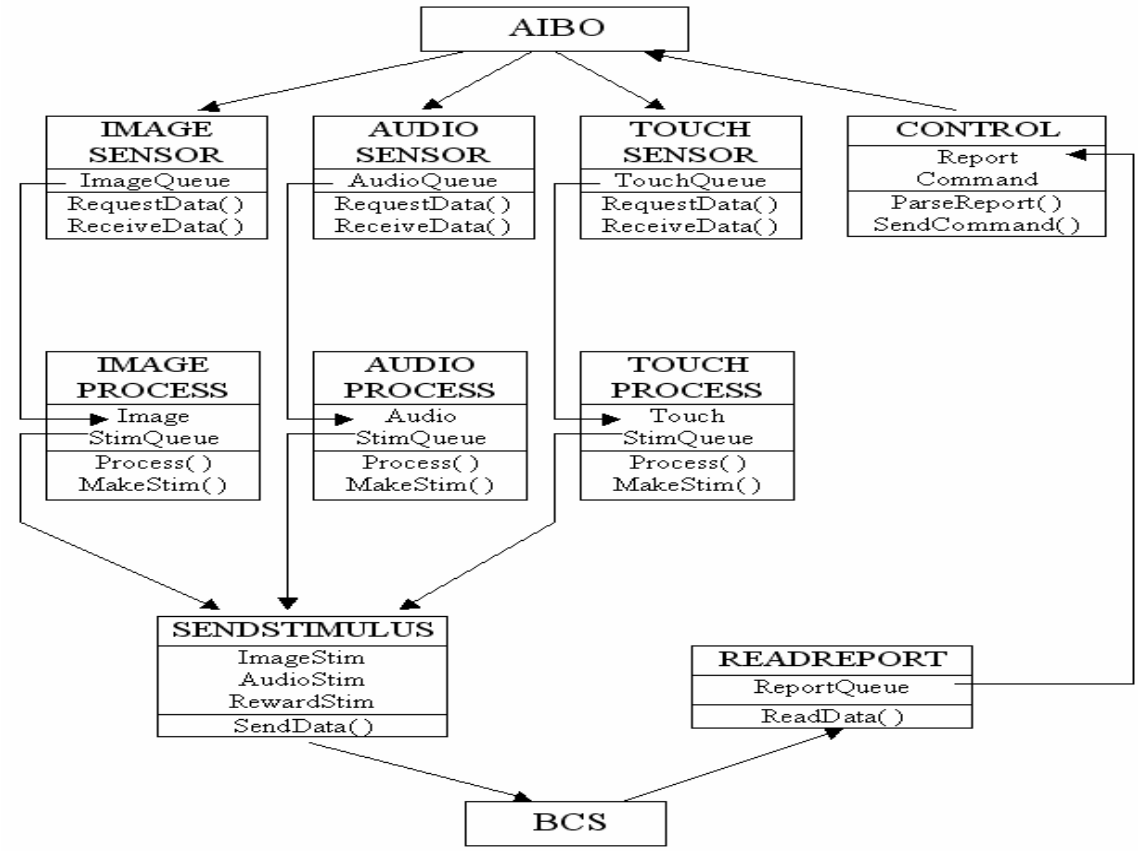

Figure 3.3 Pipe-Line Model

3.3 Interface with other systems

3.3.1 AIBO

The communication protocol between AIBO and Brainstem is wireless TCP/IP. On the AIBO side, one port number is assigned to each sensor for the outside program to collect

signal by network transfer. Currently, the ports numbers are listed in Table 3.1. Furthermore, there is one port for receiving control command from the outside. The port number is 54321. All these port numbers are set in the program on the AIBO side.

Table 3.1 Port number for each sensor on AIBO

Sensor	Port number
Video	60080
Audio	53197
Touch	24680

3.3.2 NCS

The communication between NCS and Brainstem is also based on the TCP /IP protocol. There is a separate server program which runs independently of NCS. We call it BCS [27]. It receives data packets though TCP/IP communication and sends them to NCS. All the connections between BCS and NCS, or BCS and Brainstem have information about what the connection is for. In this way BCS will know how to handle the data received, such as send to NCS or Brainstem. So actually BCS is the bridge server between the NCS and the outside world.

3.4 Implementation

As mentioned in the design section, the whole program will consist of multiple threads. They cooperate with each other to implement all the functionalities. In this sub-section, all those threads will be introduced one by one.

3.4.1 ImageSensor

As mentioned before, AIBO is a server which could provide sensor signal to requesting clients. Figure 3.4 is the flowchart for the thread of ImageSensor. It establishes the connection with AIBO through the image port first. After that, “GET / 1.0\r\n\r\n” will be sent to the connection. AIBO will respond with “HTTP/1.0 200 OK\r\nServer: W3AIBO/0.1\r\nContent-Type: image/jpeg\r\n\r\n”. Followed by this response, the image size will be available on the AIBO side. Finally, Brainstem could start to receive the image data and push it into the image queue.

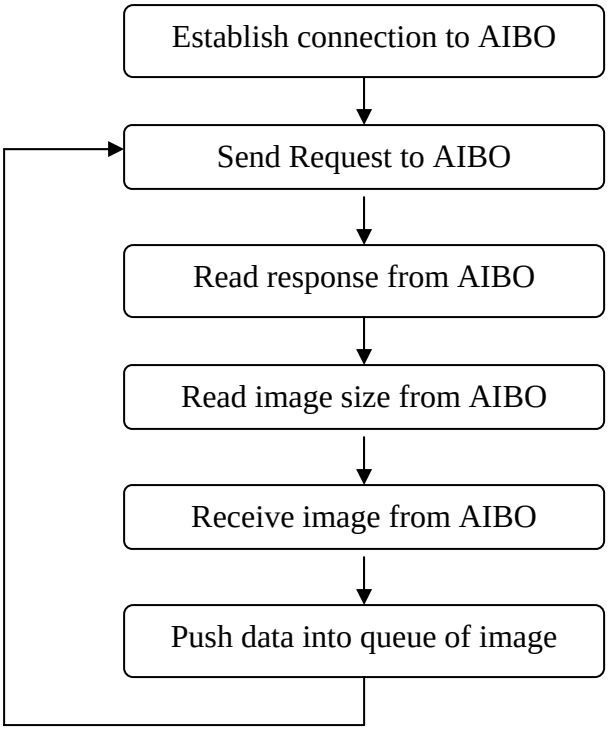

Figure 3.4 Flowchart of ImageSensor

3.4.2 ImageProcess

Once an image has been received ImageProcess takes it off the data queue and a Gabor filter is applied to process the original image to obtain stimuli. Usually a Gabor filter is

applied in motion estimation, the process of estimating the velocities of objects moving in 3-D space and their projections onto a 2-D image plane [28]. It is a methodology developed in the late 1960's. With the growing digital technology, it has attracted the interest of researchers again.

Gabor filters are defined by harmonic functions modulated by a Gaussian distribution. They bear some similarities to Fourier filters, but (by the Gaussian damping terms) are limited to certain frequency bands. As an example, in two dimensions and using a polar coordinate system with coordinates r and θ:

$$G(r,\theta) = \exp(-i\omega(\theta-\theta_0))\exp(-(r-r_0)^2/\sigma_r^2)\exp(-(\theta-\theta_0)^2/\sigma_\theta^2) \quad (1)$$

$$\sigma_r = \frac{1}{2}\sqrt{2\ln\sqrt{2}} \quad (2)$$

$$\sigma_\theta = \frac{\pi}{2\cdot num\_or} \quad (3)$$

$$r_0 = \frac{N}{2^{scale}} \quad (4)$$

$$\theta_0 = \frac{\pi}{num\_or}\cdot i, \quad i = 0,1,2,...,num\_or-1 \quad (5)$$

where num_or is the number of orientations, N is the size of filter, scale is the scale factor.

To apply the Gabor filter to a 2-dimensional image, the procedure is as follows.

1) Get the original image OriIm

2) Make filter $G(r,\theta)$

3) GaboredIm = real(ifft2(OriIm .* fftshift(abs($G(r,\theta)$))))

[Note] IFFT2 is Two-dimensional inverse discrete Fourier transform; FFTSHIFT Shift zero-frequency component to center of spectrum; '.*' is the arithmetic operator of array multiplication.

Using Matlab, 160 x 160 Gabor filters in four orientations are implemented. They are scaled and displayed as images by Imagesc (a function provided by Matlab) in Figure 3.5. One filter is a two-dimensional matrix. The blue background means the represented points that have small values. For those red points, it means that on those locations, the filter values are much bigger.

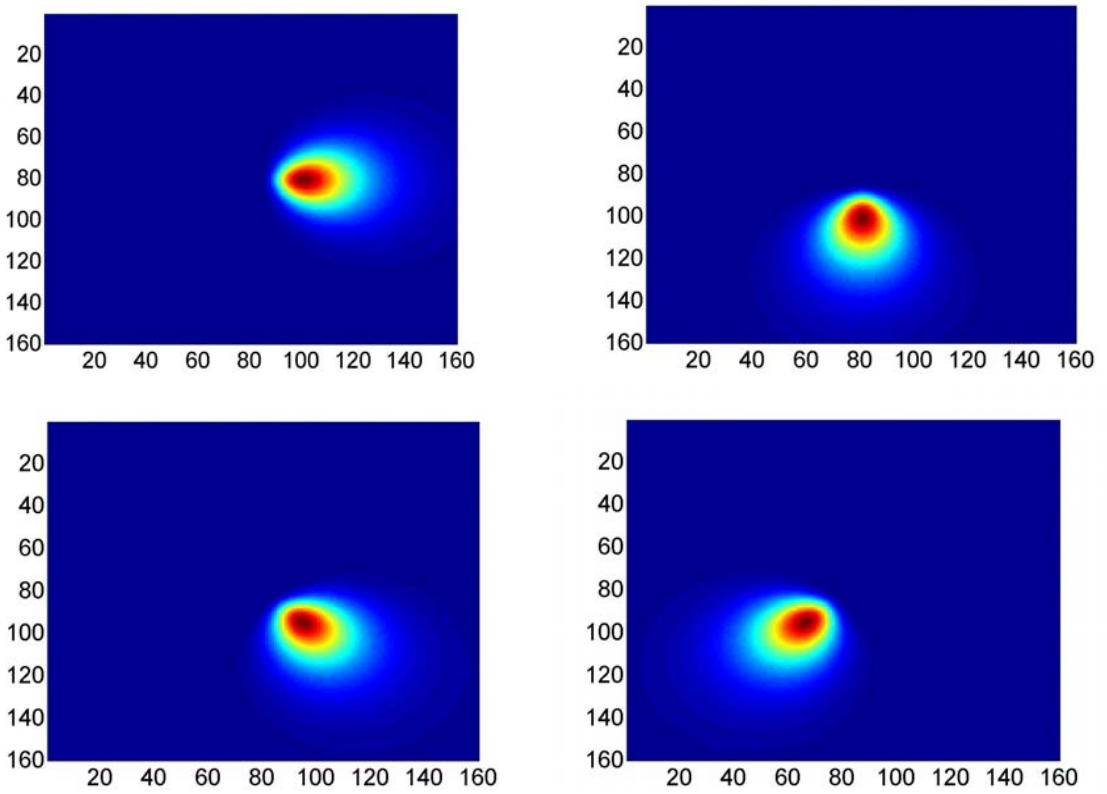

Figure 3.5 Gabor filter

Figure 3.6 shows an example for Gabor filtering. The left one is the original image. The right four images are the gabored images in four different orientations. It could be seen that after the Gabor processing, the features in specific orientations are extracted out well. The background information is filtered out.

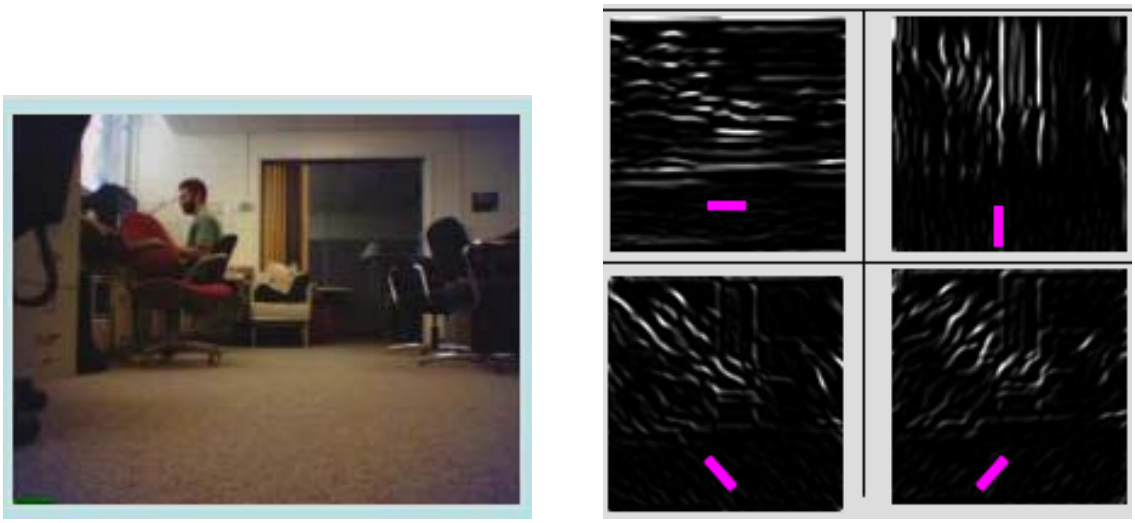

Figure 3.6 Sample results of Gabor filtering

3.4.3 TouchSenosr and TouchProcess

Figure 3.7 shows the flowchart for the TouchSensor thread. Similar to the thread of ImageSensor, TouchSensor builds the connection to the specific port of the program running on AIBO. After that, similar to the ImageSensor, “GET / 1.0\r\n\r\n” will be sent to AIBO to request data. Once the request is received by AIBO, the Brainstem could start to receive signal from touch sensor and push them to the data queue.

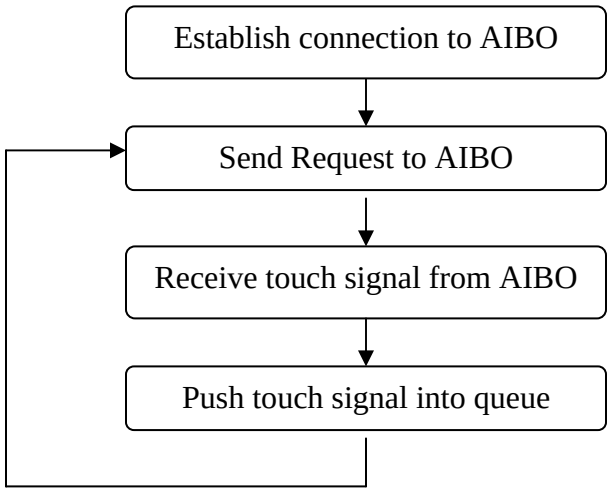

Figure 3.7 Flowchart of TouchSenosr

Figure 3.8 shows the sample sensor data returned by AIBO and the data legend. The format is determined by the program on the AIBO side. From the view point of Brainstem, it uses this data format to parse the sensor data received from AIBO.

[33] val 0 0 0 sig 0 0 0 0 0 [32] val 0 0 0 sig 0 0 0 0 0 [31] val 0 0 0 sig 0 0 0 0 0 [3] val 0 0 0 sig 0 0 0 0 0

[33] identifies the frontmost back sensor
[32] is the middle back sensor
[31] is the rear back sensor
[3] indicates the head sensor

The four integers after val become greater than zero if they are touched

Figure 3.8 Sample data returned by AIBO and data legend

3.4.4 SendStim and ReadReport

The establishment of a connection between Brainstem to NCS is different than the connection between Brainstem to AIBO. All the connections for different stimuli are connected to NCS through BCS using the same port. Socket programming ensures different socket number for every connection. Figure 3.9 shows the flow chart of SendStim thread. It builds the connection to NCS first. Then it starts a loop of reading stimuli from the data queue and sending stimuli to NCS.

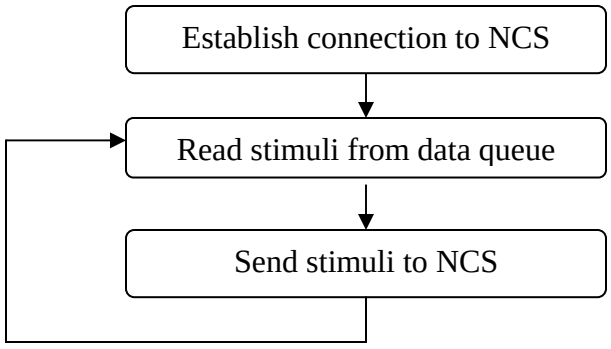

Figure 3.9 Flowchart of SendStim

In Figure 3.10, the thread of ReadReport is shown. It connects to NCS first. After that, it starts the loop to read reports from NCS and push them into the data queue.

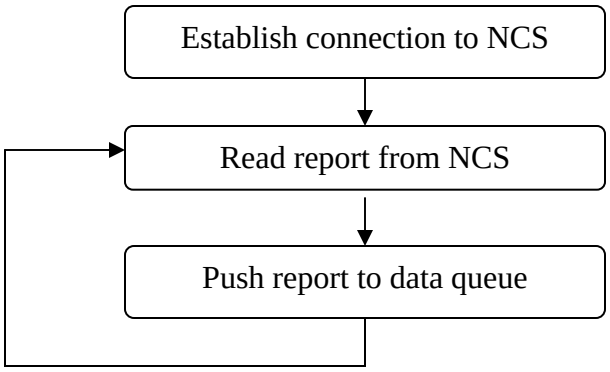

Figure 3.10 Flowchart of ReadReport

3.4.5 Control

Figure 3.11 shows the flowchart for the Control thread. It reads available reports from the data queue and parses the report to get commands that are to be executed. When the command is found, Brainstem will send the command to AIBO though the connection to AIBO. This connection is established at the beginning of thread.

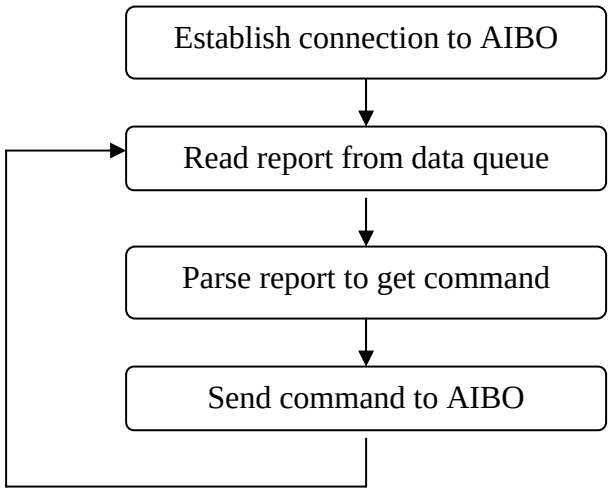

Figure 3.11 Flowchart of Control

Chapter 4: Graphic User Interface

4.1 Design

To make the program friendlier, a GUI was designed as shown in Figure 4.1.

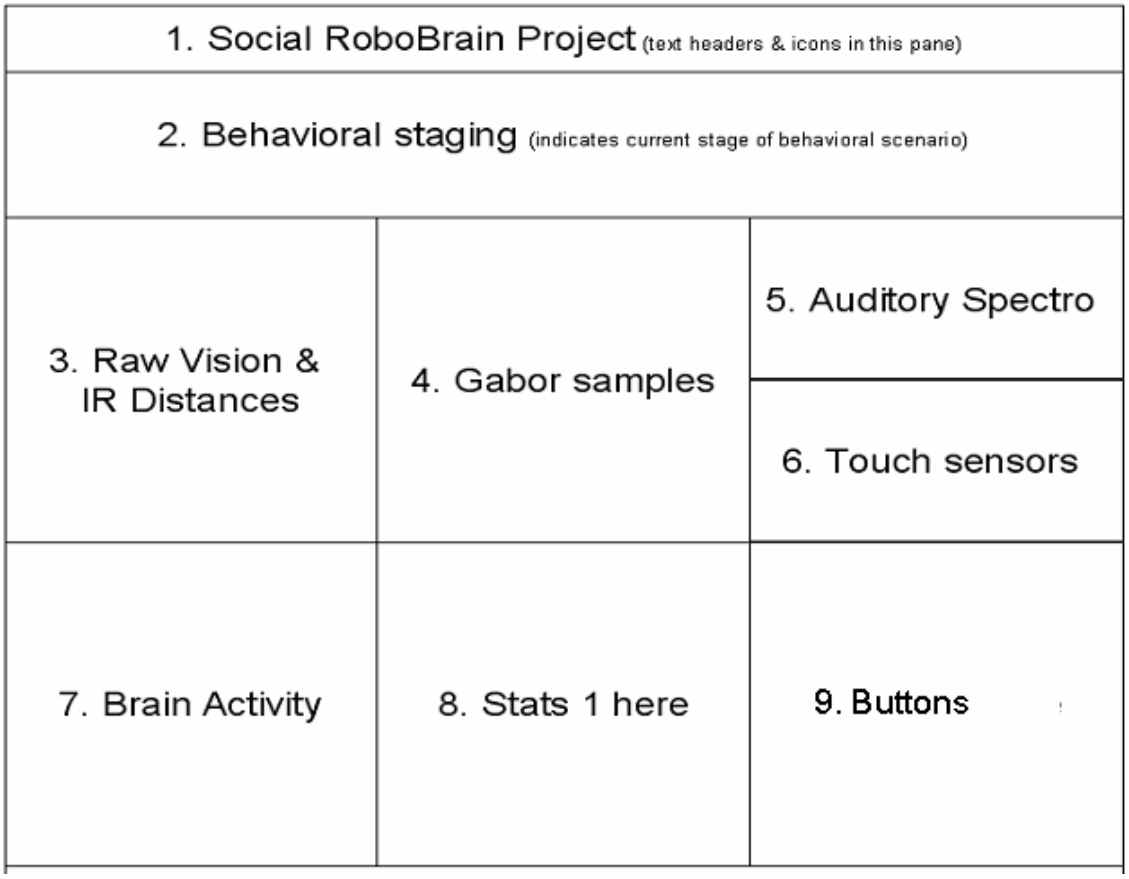

Figure 4.1 GUI Original Design

In part 1, the project title will be displayed. Part 2 indicates the current stage of the behavioral scenario. The stages are asleep, alerted, observing, action, and reward, which were mentioned in the original requirements of Chapter 3. Raw image vision from AIBO will be put inside the sub-window of part 3. Also the information from distance sensors will be there. The vision and distance information will be updated in real time. Gabored

images in four orientations are displayed in part 4. They are updated in real time also. The spectrum of auditory signal is designed to be displayed in part 5. The information in part 6 will indicate whether AIBO is being touched. In part 7, the activity of simulated brain is posted. It includes the voltage wave of membranes of neuron cells and the activity of different locations in the simulated brain. In part 8, the statistics results of simulated scenario are shown. The last part, part 9, contains some buttons which will be useful for the user to interact with the background program.

Based on Figure 4.1, a mock-up GUI view is presented in Figure 4.2.

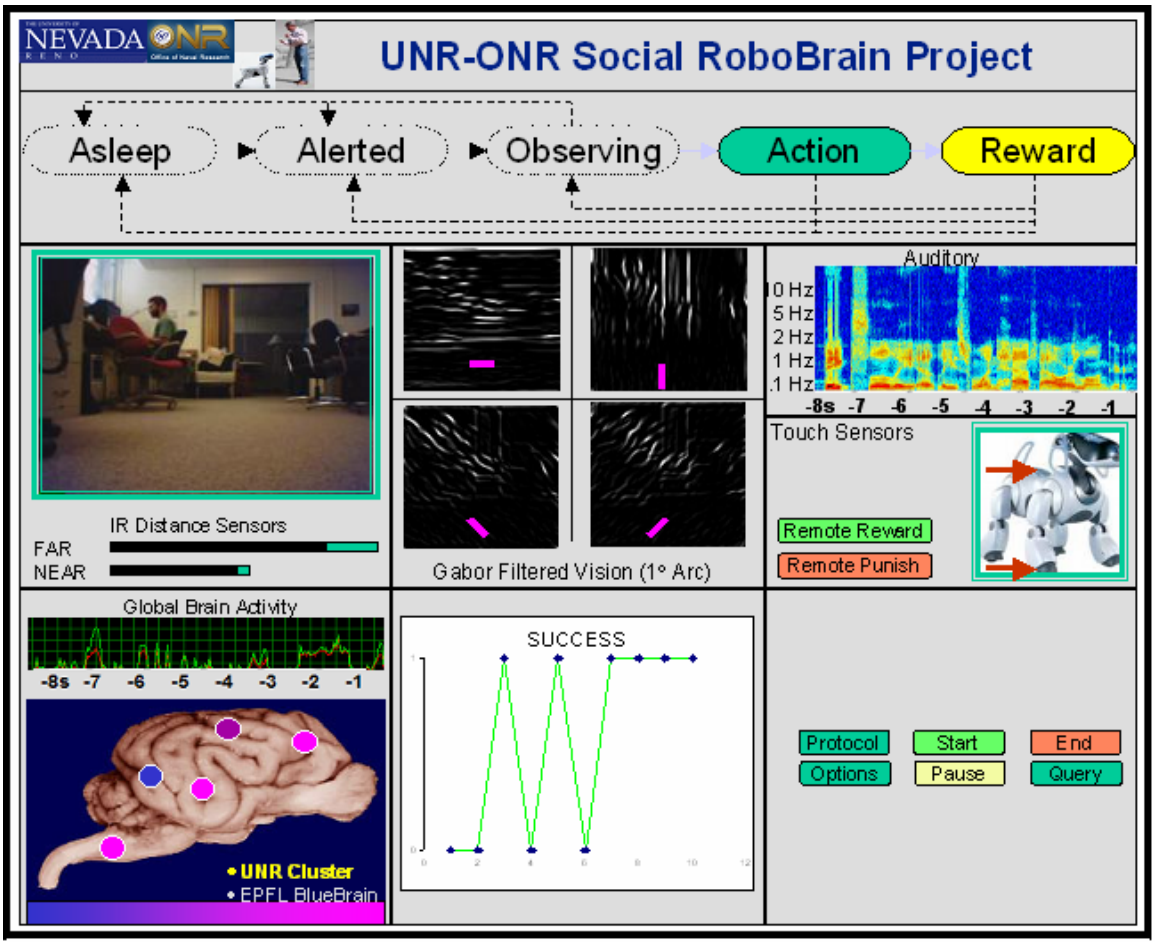

Figure 4.2 Mock-up GUI View

4.2 Development

4.2.1 Development Tool

Since the platform which we selected for running the Brainstem is Linux, a GUI development tool on Linux was needed. QT [29] is currently the most popular and convenient tool. It helps the developers for high-performance, cross-platform application development. It also includes a C++ class library and tools for cross-platform development.

Figure 4.3 QT: cross-platform GUI development tool [29]

4.2.2 Menu

QT is based on an Object-Oriented-Design. So to add menus, you just need to create the object of a menu and add the actions. The segment codes in Figure 4.4 create the menu displayed in Figure 4.5.

```
// declare pointer of file menu
QMenu *fileMenu;
// declare pointer of help menu
QMenu *helpMenu;

// add file menu to menu bar
fileMenu = menuBar()->addMenu(tr("&File"));
// add help menu to menu bar
helpMenu = menuBar()->addMenu(tr("&Help"));
// add quit action to file menu
fileMenu->addAction(quitAct);

// add About action to help menu
helpMenu->addAction(aboutAct);
// add QuitAbout action to help menu
helpMenu->addAction(aboutQtAct);
```

Figure 4.4 Code segment to create sample menu

The main menu includes two sub-menus: file menu and help menu. There is a Quit action in the file menu. As for the help menu, it includes two actions: About and QuitAbout.

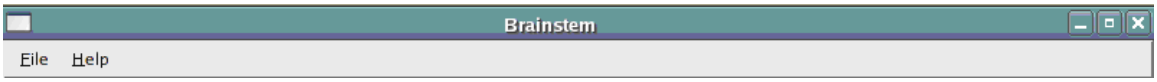

Figure 4.5 Main menu for the Brainstem

4.2.3 Splitter

The class of QSplitter is helpful to create splitter windows. The option of Vertical in the constructor of QSplitter will create splitter windows up and down. On the other hand, the option of Horizontal will create splitter windows side by side. The following code segment in Figure 4.6 creates the splitter windows in Figure 4.7. Splitter_01 is created from the main window and its style is up and down. Splitter_02 is created from Splitter_01. Its style is also up and down. Since Splitter_04 and Splitter_05 are both created from Splitter_03, so actually they with Splitter_03 together consist a big splitter

parallel to Splitter_01 and Splitter_02. Meanwhile, because the style for Splitter_04 and Splitter_05 is Horizontal, they are side by side with Splitter_03.

```
Splitter_01 = new QSplitter(Qt::Vertical, this);
Splitter_02 = new QSplitter(Qt::Vertical, Splitter_01);
Splitter_03 = new QSplitter(Qt::Vertical, Splitter_01);
Splitter_04 = new QSplitter(Qt::Horizontal, Splitter_03);
Splitter_05 = new QSplitter(Qt::Horizontal, Splitter_04);
```

Figure 4.6 Code segment to create splitters

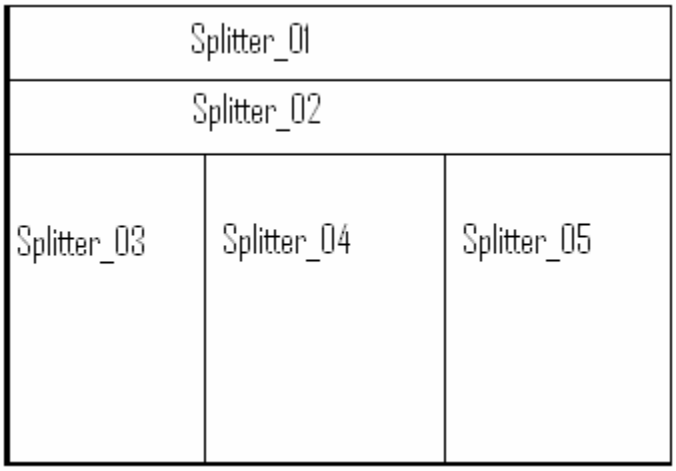

Figure 4.7 Example of splitter

4.2.4 Image View

To show an image in a window, we create an object of QLabel and set the image to be shown in it. For example, the following segment code in Figure 4.8 shows how to display an image in the first splitter window in Figure 4.7. The resulted image view is shown in Figure 4.9. It just displays the project title in the sub-window of Splitter_01.

```
QLabel *imageLabel = new QLabel(Splitter_01);
QImage image("project_title.JPG");
imageLabel->setPixmap(QPixmap::fromImage(image));
```

Figure 4.8 Code segment to create image view

Figure 4.9 Example of image view

4.2.5 Buttons

To add buttons, a layout object which contains the buttons needs to be created. Buttons can be instanced by the class of QPushButton and then be placed into the layout. Eventually, connect the event handing functions to the buttons. The following segment code in Figure 4.10 shows how to add buttons. The Figure 4.11 shows the layout of buttons. The buttons of Scenario, Start, End, Options, Pause and Query are added into the layout.

```
// Create the layout to contain buttons
QGridLayout *layout = new QGridLayout;
// Create and put Scenario button in the layout
buttonScenario = new QPushButton(tr("Scenario"));
layout->addWidget(buttonScenario);
// Create and put Start button in the layout
buttonStart = new QPushButton(tr("Start"));
layout->addWidget(buttonStart);
// Create and put End button in the layout
buttonEnd = new QPushButton(tr("End"));
layout->addWidget(buttonEnd);
// Create and put Options button in the layout
buttonOptions = new QPushButton(tr("Options"));
layout->addWidget(buttonOptions);
// Create and put Pause button in the layout
buttonPause = new QPushButton(tr("Pause"));
layout->addWidget(buttonPause);
// Create and put Query button in the layout
buttonQuery = new QPushButton(tr("Query"));
layout->addWidget(buttonQuery);
// connect the event handling functions to Options button.
connect(buttonOptions, SIGNAL(clicked()), this, SLOT(options()));
```

Figure 4.10 Code segment to create buttons

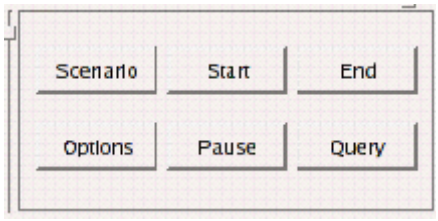

Figure 4.11 Example of Buttons

4.3 Results

The current GUI implementation is shown in Figure 4.12. The project title, behavioral staging, raw vision, Gabor samples, brain activity, touch sensor information and buttons in the original design are implemented. The others which include distance information, auditory spectrum and stats are left as the future work.

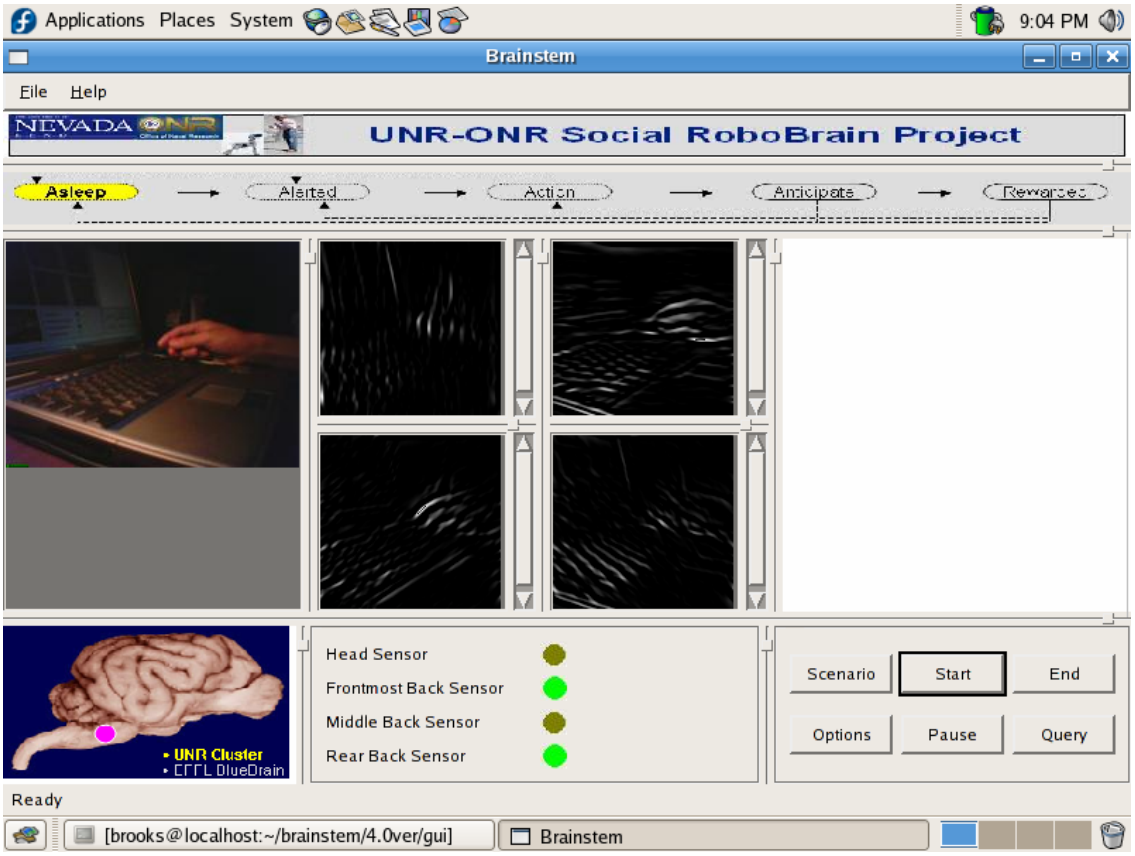

Figure 4.12 Current Implementation of GUI

The raw video shown in the GUI is from the ImageSensor thread and the Gabored images are from the ImageProcess Thread. The touch sensor information is from TouchSensor thread. These threads are all mentioned in Chapter 3.

Figure 4.13 shows the dialog for options. It makes adjusting settings of program more convenient. “No NCS” is useful when we need to test only the communication between AIBO and Brainstem. When “No NCS” is selected, the SendStimulus thread reads stimuli from stimuli queue but throws them away instead of sending to NCS. Checkbox of “Process Image”, “Process Audio” and “Process Touch” determine whether Brainstem needs to process the image, audio, touch signal respectively.

Figure 4.1314 Dialog for the button of options

Chapter 5: Conclusions and Future Work

5.1 Conclusions

First of all, the scenario for the Brainstem which is mentioned in Section 3.2, listed below again, is implemented.

1) AIBO resting on-guard

2) Ambient noise or motion

3) AIBO rises to attention

4) AIBO cortex processes visual activity for several seconds

5) AIBO cortex reaches decision on motor action; performs

6) AIBO receptive to reward (stroking of back) for several seconds

7) Human rewards (or not)

8) AIBO resumes resting on-guard state…

Second, we expanded the program to be capable of sending commands to AIBO directly. This allows for local AI to take over some low-level control functionality. Furthermore, since the program structure is multiple-threaded, it is convenient to be expanded. Moreover, a graphical user interface is implemented to make the Brainstem friendlier, more interactive. This GUI was presented in the Section 4.3.

5.2 Future Work

5.2.1 Audio Signal Processing

Currently, only the vision stimuli are sent to the NCS. No audio stimuli are sent to the NCS because how to process audio signal to produce stimuli is still being studied. Some

prototype methodology was mentioned in the dissertation of J. B. Maciokas [30].

5.2.2 Graphical User Interface

Some of the sub-windows in current GUI are blank. Their content depends on the development of core model for NCS-AIBO interaction. For the implementation of user interface, please refer to Chapter 4: Graphical User Interface.

5.2.3 Distance Sensor Signal Collecting and Processing

In the program of AIBO, the entire signal from sensors is packed to be sent out. For the current version, only data from three back touch sensors and the head sensor are being sent out. We just need to uncomment some code to turn the other sensors on.

The code of AIBO could be found at http://brain.cse.unr.edu/~king/aiboFiles/Robot/Cycle3Dog.zip. Code segments are given in Appendix C. It is not hard to turn on any sensor.

On the Brainstem side, collecting sensor signal from AIBO is done with the threads of TouchSensor and TouchProc. Current output data format for the sensor signal was already discussed in Section 3.4.3. After other sensors are turned on, the data from new opened sensors will be appended to the current package.

5.2.4 Artificial Intelligent Thread

To be able to make some low-level decisions, an artificial intelligent thread need to be added. It is supposed to hook up with the ImageSensor, AudioSensor and TouchSensor threads to do some artificial intelligent analysis of the raw data and make low-level decisions. This will allow more general AI interaction as opposed to a fixed scenario.

Appendix A: Commands to AIBO

The following definition is from the program of AIBO, also defined in the file of ./bkg/global.h

```
#define SIT             'a'
#define SIT_GREET       'b'
#define WALK            'c'
#define EAR             'd'
#define HEAD_TILT       'e'
#define PAW             'f'
#define GROWL           'g'
#define REST            'h'
#define TAIL            'i'
#define BARK            'j'
```

Appendix B: File Tree

./bkg/ (Background program)

---- main.cpp: main file for program if running without GUI

---- InitBkg.cpp: Initialize global variable, connection with AIBO and NCS

---- DataQueue.cpp: Definitions for all data queues

---- SocketFun.cpp: Functions for establish socket connection

---- rjpg8c.c: JPEG functions

---- jmysrc.c: Data source manager for rjpg8c

---- Matrix.cpp: Class of matrix

---- ImageSensor.cpp: functions for the thread of ImageSensor

---- ImageProc.cpp: functions for the thread of ImageProcess

---- TouchSensor.cpp: functions for the thread of TouchSensor

---- TouchProc.cpp: functions for the thread of TouchProcess

---- SendStim.cpp: functions for the thread of SendStim

---- ReadReport.cpp: functions for the thread of ReadReport

---- Control.cpp: functions for the thread of Control

./gui/ (Forground program)

---- main.cpp: main file for GUI

---- mainwindow.cpp: code for GUI and actual running function for all threads

---- moc_mainwindow.cpp: generated automatically by qmaker

Appendix C: Code Segment for Sending Sensor Signal on the AIBO Side

File location: CycleDog\SensorObserver7\SensorObserver7\SensonObserver7.cc

```
void
SensorObserver7::PrintERS7Sensor( int index, OSensorFrameVectorData* sensorVec)
{
   //print data to the sending buffer
   char *ptr;

   //move pointer to front of send buffer and reset the size value
   ptr = (char*)connection[index].sendData;
   connection[index].sendSize = 0;

   //Whenever data is written, we advance the pointer forward by the number
   // of bytes written (that's why the number of bytes written is returned)

   // BODY ----------------------------------------------------------------
   /*
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[ACC_X]);      //acceleration x
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[ACC_Y]);      //acceleration y
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[ACC_Z]);      //acceleration z
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[BODY_PSD]);   //body psd?
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[WLAN_SW]);    //wireless lan switch
   */
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[BACK_SW_F]);  //front back sensor
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[BACK_SW_M]);  //middle back sensor
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[BACK_SW_R]);  //rear back sensor

   // HEAD ----------------------------------------------------------------
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[HEAD_SENSOR]); //head sensor
   /*
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[CHIN_SW]); //chin switch
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[HEAD_PSD_NEAR]); //head psd (near)
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[HEAD_PSD_FAR]);  //head psd (far)
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[HEAD_TILT1]); //head tilt 1
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[HEAD_PAN]);  //head pan
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[HEAD_TILT2]); //head tilt2
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[MOUTH]);  //mouth

   // RIGHT FRONT LEG -----------------------------------------------------
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RFLEG_J1]); //joint 1
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RFLEG_J2]);  //joint 2
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RFLEG_J3]);  //joint 3
   ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[RFLEG_SW]);  //

   // LEFT FRONT LEG ------------------------------------------------------
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LFLEG_J1]);  //joint 1
   ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LFLEG_J2]);  //joint 3
```

```
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LFLEG_J3]);  //joint 3
    ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[LFLEG_SW]);  //

    // RIGHT REAR LEG ------------------------------------------------------
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RRLEG_J1]);  //joint 1
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RRLEG_J2]);  //joint 2
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[RRLEG_J3]);  //joint 3
    ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[RRLEG_SW]);  //

    // LEFT REAR LEG -------------------------------------------------------
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LRLEG_J1]);  //joint 1
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LRLEG_J2]);  //joint 2
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[LRLEG_J3]);  //joint 3
    ptr += PrintSensorValue(ptr, index, sensorVec, ers7idx[LRLEG_SW]);  //

    // TAIL ----------------------------------------------------------------
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[TAIL_TILT]);  //tilt
    ptr += PrintJointValue(ptr, index, sensorVec, ers7idx[TAIL_PAN]);  //pan
    */

    //end data with newline character
       //in the future, we may want to compute length and send that first
       //so the client can just do a read X bytes instead of using 'readline'.  Is there
       //a significant speed difference?
    sprintf( ptr, "\n" );
    ptr += 1;
    connection[index].sendSize+=1;
}
```

References

[1] Modi K. P., “An application of human robot interaction: Development of a ping-pong playing robotic arm”, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, 2005, v 2, pp. 1831-1836.

[2] Fang Hairong, “Application study of fire-fighting robot for railway tunnel fire”, Proceedings of the 2004 International Symposium on Safety Science and Technology, 2004, v 4, pp. 1627-1631

[3] Nguyen H.G., “Robotics for law enforcement: Applications beyond explosive ordnance disposal”, Proceedings of SPIE - The International Society for Optical Engineering, 2001, v 4232, pp. 433-454

[4] Cavusoglu, “Robotics for telesurgery: Second generation Berkeley / UCSF laparoscopic telesurgical workstation and looking towards the future applications”, The Industrial robot, 2003, v 30, n 1, pp. 22 – 29

[5] Ramelhart D.E. and McClelland J.L., “Parallel Distributed Processing (PDP)”, Exploration in the Microstructure of Cognition, 1986, v 1, MIT Press, Cambridge, MA, USA

[6] Zimmermann, H.J., "Fuzzy Sets Theory and Its Applications", Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990

[7] Goldberg D.E., “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wisely, MA, USA, 1989

[8] Arakawa T. and Fukuda T, “Natural motion generation of biped locomotion robot using hierarchical trajectory generation method consisting of GA, EP layers”, Proceedings of IEEE International Conference on Robotics and Automation, Albuquerque, USA, 1997, pp. 211-216

[9] SUMMER PROJECT 2005: A Hybrid Neuromorphic/AI Socially Interactive Robotic Sentry, [last accessed on September 28th, 2006]; Available from: http://brain.cse.unr.edu/share/summer05/pages/overview.html

[10] P. Goodman and F. C. Harris Jr., "Parallel Beowulf Brain-Robotics Simulation", Research Proposal to ONR., submitted 2001

[11] The neuron, [last accessed on September 28th, 2006]; Available from: http://www.ship.edu/~cgboeree/theneuron.html

[12] E. C. Wilson. Parallel implementation of a large-scale biologically realistic parallel neocortical-neural network simulator [Master's Thesis]. University of Nevada: Reno. 2001.

[13] E. C. Wilson, P. H. Goodman, and F. C. Harris Jr., "Implementation of a Biologically Realistic Parallel Neocortical-Neural Network Simulator," in the *Proc. of the 10th SIAM conference on Parallel Process for Sci. Comput.*, March 2001.

[14] E. C. Wilson, P. H. Goodman, and F. C. Harris Jr., "A Large-Scale Biologically Realistic Cortical Simulator," in the *Proceedings of SC 2001*, Denver, Colorado, 2001.

[15] J. Frye. Parallel Optimization of a NeoCortical Simulation Program [Master's Thesis]. University of Nevada: Reno, NV. December 2003.

[16] J. Frye, J. G. King, J. C. Wilson, and F. C. Harris Jr., "QQ: Nanoscale Timing and Profiling," in the *Proceeding of the 19th IEEE International Parallel & Distributed Processing Symposium*, 2005.

[17] P. Goodman and F. C. Harris Jr., "Durip04: Parallel Beowulf Computing / Brain / Robotics, Phase III," Research Proposal to ONR. Submitted 2004.

[18] P. Goodman, E. C. Wilson, J. B. Maciokas, F. C. Harris, A. G. Gupta, J. L. Louis, H. Markram, “Large-scale parallel simulation of physiologically realistic multicolumn sensory cortex”, Technical Paper, University of Nevada, Reno, 2001

[19] J. Maciokas, P. Goodman, and F. Harris, "Large-scale spike-timing-dependant-plasticity model of bimodal (audio-visual) processing." Technical Paper. Brain Computation Lab, University of Nevada, Reno, NV. 2002.

[20] J. B. Maciokas, P. H. Goodman, and J. L. Kenyon, "Accurate Dynamical Model of Interneuronal GABAergic Channel Physiologies." Technical Paper. University of Nevada, Reno. 2004.

[21] J. C. Macera, Design and implementation of a hierarchical robotic systems; A platform for artificial intelligence investigation, [Master's Thesis]. University of Nevada, Reno, 2003.

[22] J. C. Macera, P. H. Goodman, F. C. Harris Jr., R. Drewes, and J. B. Maciokas, "Remote-neocortex control of robotic search and threat identification," *Robotics and Autonomous Systems*, 2004, vol. 46, pp. 97-110

[23] M. Inaba, S. Kagami, F. Kanehiro, Y. Hoshino, and H. Inoue, "A platform for robotics research based on the remote-brained robot approach," *The International Journal of Robotics Research*, October 2000, vol. 19, pp. 933-954

[24] User's Guide: basic for ERS-7, [last accessed on June 11, 2006], Available from http://esupport.sony.com/US/perl/model-documents.pl?mdl=ERS7

[25] Brainstem, [last accessed on September 28th, 2006], available from: http://www.mult-sclerosis.org/brainstem.html

[26] Bradford Nichols, Dick Buttlar and Jacqueline Proulx Farrell, "Pthreads Programming", O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA, USA, 1998, p35

[27] James G. King, Brain Communication Server: A Dynamic Data Transferal System for A Parallel Brain Simulator, [Master's thesis], University of Nevada, Reno, 2005

[28] Tabatabai, J. Ali, "Motion estimation methods for video compression - a review," Journal of the Franklin Institute, Aug, 2000, v 126, n 8, pp. 1411-1441

[29] Qt – Cross-Platform C++ Development, [last accessed on September 28th, 2006]; Available from: http://www.trolltech.com/products/qt/features/index

[30] J. B. Maciokas, Towards an Understanding of the Synergistic Properties of Cortical Processing: A Neuronal Computational Modeling Approach, [Doctoral Dissertation], University of Nevada, Reno, 2003