University of Nevada, Reno

Applications in Neurorobotics

A thesis submitted in partial fulfillment of the
the requirements for the degree Master of Science in
Computer Engineering

by

Corey M. Thibeault

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

December, 2012

University of Nevada, Reno
Statewide • Worldwide

THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

COREY M. THIBEAULT

entitled

Applications In Neurorobotics

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

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

Dr. Sergiu Dascalu, Ph. D, Committee Member

Dr. Normand Leblanc, Ph. D, Graduate School Representative

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

December, 2012

Abstract

The field of neurorobotics is still in its infancy; however, its intersecting motivations are not. On the one hand, theories of neuroscience that require immersion in the real-world can be embedded in mobile agents creating complex patterns of activity believed to be a requirement for understanding higher-order neural function. On the other, the cognitive capabilities of humans remain unparalleled by artificial agents. Emulating biology is one strategy for creating more capable artificial intelligence. Despite these strong motivations for creating neurorobotic entities technological hurdles still remain at all levels. This thesis presents two different contributions to the field of neurorobotics. The first is aimed at reducing the complexity of coupling spiking neural models with virtual agents. This is accomplished through a set of tools that act to abstract the neuroscience details from roboticists and the mechanical details away from the neuroscientists. The second contribution provides an example of how higher-level cognitive theories of speech processing can be integrated into the neurorobotics paradigm. Extracting the emotional content of a speaker, independent of what is being spoken, is a daily act for most people. The neural basis for this ability remains illusive, however cognitive models have been realized. This class of models can be integrated with the biologically realistic neural simulations in a complementary way to expand the capabilities of a neurorobotic system.

For Dashiel

Acknowledgments

Having just finished writing acknowledgments for my dissertation it feels redundant to put all of the same people here. Nevertheless, my son Dashiel and Amber deserve a huge thanks for their support during this work. As do my mother Lorre, my father John, my stepmother Laura and my grandfather, Stan. Sadly my grandmother Barbara passed away before I completed this work but her influence has never left me.

This work was completed in the UNR Brain Lab and started under the direction of Dr. Phil Goodman. He unfortunately passed away before this thesis was written. However he did get the opportunity to see the work completed.

There were many students in the Brain Lab who helped me along the way. Those people include Josh Hegie, Roger Hoang, Gareth Ferneyhough, Laurence Jayet Bray, Casey Howard, Sridhar Anumandla, Kevin Cassiday, Bryce Prescott, Rashid Makhmudov, and Nick Ceglia.

My committee members, Dr. Frederick C. Harris, Jr., Dr. Sergiu Dascalu and Dr. Normand Leblanc, were kind enough to review this work and provide helpful suggestions in a short amount of time.

I am grateful to all of you...and anyone else I may have missed. Thank You.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is an incredible amount of detail and understanding of the nervous system, from the molecular to the cognitive, that is still unknown. Modern experimental methods are providing historically unprecedented insight into the capabilities of the mind.

Since the development of the Hodgkin-Huxley equations in 1952 [25] the field of computational neuroscience has grown exponentially. Mathematical models of single and populations of neurons have helped expand the field of neuroscience as a whole. Computational modeling allows investigators levels of detail and control over complex neurological structures that are currently impossible at a physiological level. This level of control offers the possibility to observe the effects of invasive surgeries [15, 23, 24, 37], neurological diseases [14, 18, 20, 41], pharmaceuticals [33] as well as the complex dynamics of neural information processing. In addition, these models have found a mutually propitious relationship with robotics research in the form of neurorobotics.

Neurorobotics can be described as the combination of neurological models and robotic agents. Often referred to as embodied modeling, the coupling of computational biology and engineering is a natural one that is beneficial to both disciplines. Historically strategies for embedding artificial intelligence have failed to result in agents with truly emergent properties. Because of this it is still unreasonable to deploy a robotic entity and expect it to learn from its environment the way biological entities can. This is one motivation behind neurorobotics. Similarly, neural models require complex and varied input signals in order to accurately replicate the activity observed *in vivo*. One method for creating this complex stimuli is through immersing a model in real-world stimuli; something that can be accomplished through neurorobotics.

There are number of different neurological models that have been coupled with robotic agents such as rate-based [4, 9, 16], cognitive based [3], and spiking [2, 21, 22, 26]. Although the work presented here is focused on interactions with spiking neural models, this is an arbitrary restriction that is not indicative of its applicability to other areas of computational neuroscience.

This thesis is primarily composed of two published papers that present research in different facets of the neurorobotics paradigm. These are included in their entirety for completeness. The first, [43], presents NCSTools, a software suite developed to help ease the integration of robotics and spiking neural models. This is the contents of Chapter 3. The second, [44], is an example of how researchers can take knowledge of higher level behavioral function and interact effectively with spiking neural models. This is presented in Chapter 4. In addition, Chapter 2 provides a brief introduction to neurons, computational neuroscience and the NeoCortical Simulator. Finally, Chapter 5 closes with some final thoughts and discussion on neurobotic interactions.

Chapter 2

Background

2.1 Computational Neuroscience

There are many different levels of neuroscience research attempting to clarify the function of the brain. From the single molecule studies of biophysics to the behavioral research of cognitive neuroscience, the mysteries of the brain are enjoying elucidation from both bottom-up and top-down approaches. Computational Neuroscience can be described as a complimentary field that spans the spectrum of neuroscience. There are obvious physiological barriers to gathering detailed information of most complex neuronal structures. Beyond the lack of connectivity information, is the lack of non-invasive measurement equipment. Computational neuroscience provides unique and unrivaled access to both the deep structures of the brain as well as the molecular information of single cells.

2.2 Neurons

Neurons are a class of excitable cells that are the primary building block of the nervous system of all animals belonging to the group eumetazoa. These cells are characterized by their electrical and chemical communication mechanisms that transmit information throughout the nervous system. Neurons maintain an ionic concentration gradient along a differentially permeable membrane, resulting in a voltage potential between the inside and outside of the cell. If the membrane voltage of the cell is increased above a threshold an all-or-nothing avalanche of current creates a discrete electrical event that travels along the cell, as shown in Figure 2.1A. These events are referred to as Action Potentials (AP) or spikes. Figure 2.1C illustrates the change in voltage at a discrete location along a theoretical neuron. The voltage change in this location will cause a similar change in the immediate surrounding area. The result is a dynamic change in the membrane voltage that propagates throughout the cell.

When the propagating action potential reaches the synapse located at the end of each of the neuron's axons, illustrated in Figure 2.1B, a chemical signaling cascade is initiated. This cascade results in a quantile release of neurotransmitter which travels down a chemical gradient to a second neuron's postsynaptic cleft. The neurotransmitter activates pores, called ion channels, in the membrane, resulting in a change in the voltage potential of the second cell.

It goes without saying that this is an obvious over-simplification of a complex process. However, this is the essence of communication in the nervous system.

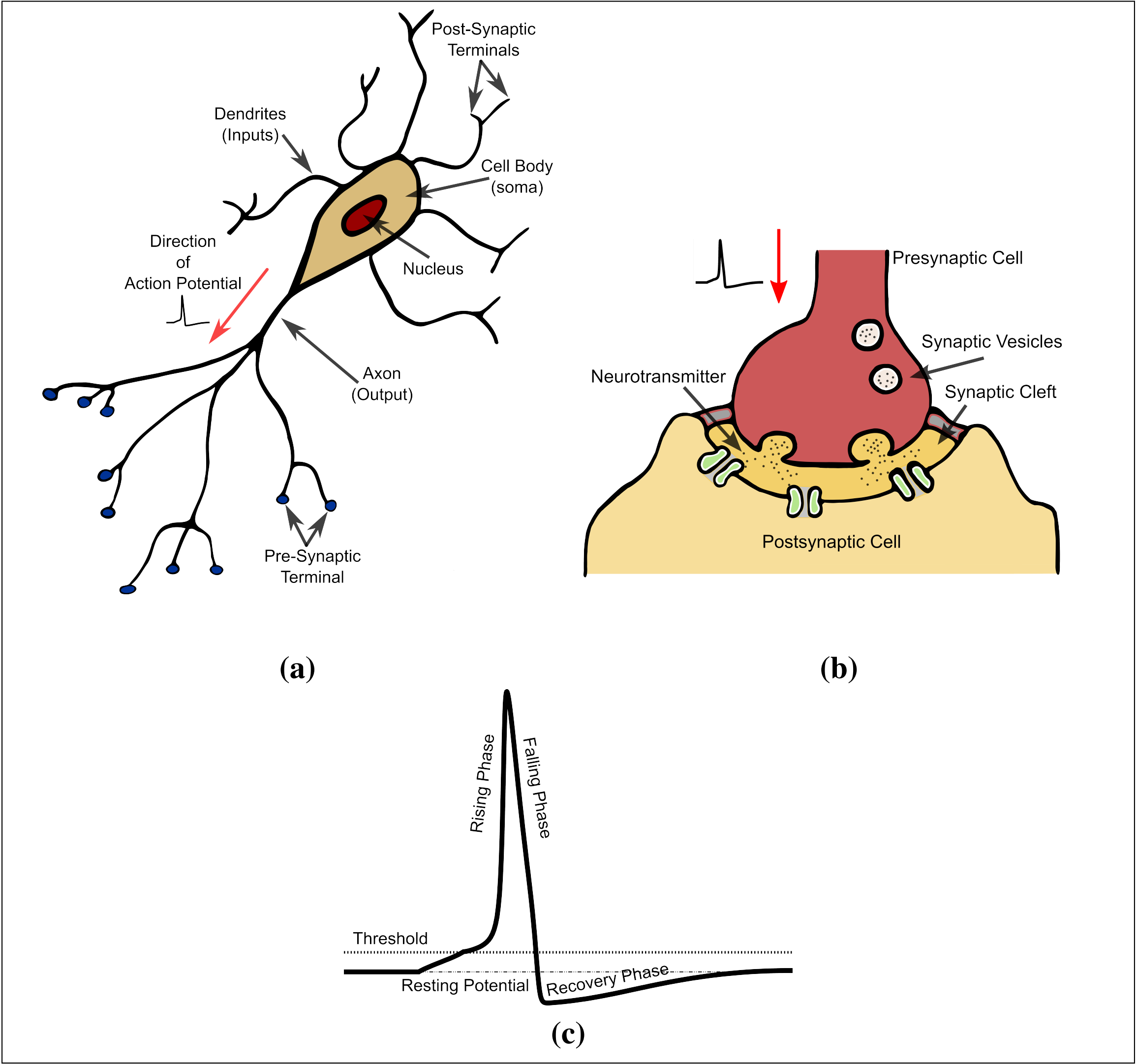

Figure 2.1: (A) Prototypical neuron, (B) Single synapse, (C) Action potential at a single spatial location.

2.3 Spiking Neural Models

Biologically realistic neural simulations generally begin with a model of a single neuron. Of which there are a large number available to computational neuroscientists, each with different levels of computational complexity and biological realism. There is a constant balance between execution time and biophysical plausibility. A balance that more often than not leans in favor of execution time. Even as neuron models are simplified and approximated, the neural structures of interest may require a computationally unreasonable amount of them. In order to further drive the neuroscience research, engineers are creating more optimized simulation environments that take advantage of the latest hardware advances.

The NeoCortical Simulator

The NeoCortical Simulator (NCS) was developed at The University of Nevada, Reno by the Brain Computation Lab under the direction of Dr. Phillip Goodman. From its inception a heavy emphasis was placed on parallelization and performance. In addition, mechanism for getting spiking information out and stimulus in was also extremely important. Despite the focus on performance, NCS provides a number of important biological models. For a review of what NCS refer to Wilson *et al.* [50, 51] and to see how NCS compares to other neural simulators see Brette *et al.* [7]. Its features can be summarized as:

- High-performance Message Passing interface (MPI)-based parallel architecture
- Leaky Integrate-and-fire (LIF) neurons with conductance based synapses and Hodgkin-Huxley channels.
- Hebbian synaptic learning with Short-term plasticity, augmentation, and spike-timing dependent plasticity (STDP).

NCS Neuron Model

The model used in NCS falls under a class of threshold neurons characterized by the absence of an action potential. Since an action potential is an all-or-nothing quality it can be assumed that once the neuron's membrane potential reaches the threshold value that an action potential will occur. Using this approximation the channels responsible for producing the action potential can be ignored and when the threshold is reached the membrane voltage can be reset to the resting potential. This simplification allows researchers to focus on the sub-threshold dynamics while retaining the spiking activity of the model.

At the single cell level NCS solves a limited and slightly reordered form of the Hodgkin-Huxley Model that is similar to Equation (2.1). However, during the numerical integration a constant membrane leak is added. This is explained further below.

$$C_N \frac{dV}{dt} - I_M - I_A - I_{AHP} - I_{input} - I_{syn} + I_{leak} = 0 \tag{2.1}$$

The currents expressed in this equation fall into several different categories that are only briefly explained here. To begin, both I_M and I_{AHP} contribute to the membrane voltage by controlling spike-frequency adaptation [29]. These are small ionic currents that have a long period of activity when the membrane voltage is between rest and threshold.
I_M is the Noninactivating Muscarinic Potassium Current and is defined by

$$I_M = \bar{g}_M S m^P \left(E_k - V\right) \tag{2.2}$$

Where S is a non-dimensional Strength variable added to NCS and P is the power that the activation variable m is raised to. This is essentially decreasing the slope of the activation variable. The change of that activation variable is defined as

$$\frac{dm}{dt} = \frac{m_\infty - m}{\tau_m} \tag{2.3}$$

Where

$$\tau_m = \frac{\epsilon}{e^{\left(\frac{V - V_{1/2}}{\omega}\right)} + e^{-\left(\frac{V - V_{1/2}}{\eta}\right)}}$$

$$m_\infty = \frac{1}{1 + e^{-\left(\frac{V - V_{1/2}}{\xi}\right)}}$$

ϵ is the scale factor.

$V_{1/2}$ satisfies the equation $m_\infty(V_{1/2}) = 0.5$.

ω, η and ξ are slope factors affecting the rate of change of the activation variable m. Notice that (2.2) is different from the traditional equation shown below in Equation (2.4). This reverse of the driving force explains the sign changes in Equation (2.1).

$$I_M = \bar{g}_M m_m (V - E_K) \tag{2.4}$$

I_{AHP} is the current provided by the other small spike-adaptation contributing channel. These are voltage independent potassium channels that are regulated by internal calcium [29].

$$I_{AHP} = \bar{g}_{AHP} S m^P (E_k - V) \tag{2.5}$$

Where S is a non-dimensional Strength variable added to NCS and P is the power that the activation variable m is raised to. The change of that activation variable is defined as

$$\frac{dm}{dt} = \frac{m_\infty - m}{\tau_m} \tag{2.6}$$

$$\tau_m = \frac{\epsilon}{f(Ca) + b}$$

$$m_\infty = \frac{f(Ca)}{f(Ca) + b}$$

Where

ϵ is the scale factor.

b is the backwards rate constant, defined as CA_Half_Min in the NCS documentation.

$f(Ca)$ is the forward rate constant defined by (2.7).

$$f(Ca) = \kappa[Ca]_i{}^{\alpha} \tag{2.7}$$

Internal calcium concentrations are calculated at the compartment level in NCS. Physiologically the calcium concentration of a cell increases when an action potential fires. After the action potential has ended the internal concentration of calcium will diffuse throughout the cell where it is taken up by numerous physiological buffers. In NCS this diffusion/buffering phenomena is modeled by a simple decay equation defined by Equation (2.8).

$$[Ca]_i(t+1) = [Ca]_i(t)\left(1 - \frac{dt}{\tau_{Ca}}\right) \tag{2.8}$$

Where

dt is the simulation time step.

τ_{Ca} is the defined time constant for the Ca decay.

When an action potential fires in NCS the internal calcium concentration is increased by a static value specified in the input file.

The third and final channel type modeled in NCS is the transient outward potassium current or K_a. This channel requires hyperpolarization for its activation; meaning that the channel will open during inhibitory synaptic input [29]. This is defined by (2.9).

$$I_K = \bar{g}_M S m^P h^C \left(E_k - V\right) \tag{2.9}$$

Where as before S is a non-dimensional Strength variable added to NCS, P is the power that the activation variable m is raised to and C is the power that the inactivation variable h is raised to. The change of activation and inactivation variables is defined by (2.10) and

(2.11).

$$\frac{dm}{dt} = \frac{m_\infty - m}{\tau_m} \tag{2.10}$$

$$\frac{dh}{dt} = \frac{h_\infty - m}{\tau_h} \tag{2.11}$$

Where

$$m_\infty = \frac{1}{1 + e^{-\left(\frac{V - V_{1/2m}}{\xi}\right)}}$$

$V_{1/2m}$ satisfies the equation $m_\infty(V_{1/2m}) = 0.5$.

ξ is slope factor affecting the rate of change of the activation variable m.

$$h_\infty = \frac{1}{1 + e^{-\left(\frac{V - V_{1/2h}}{\eta}\right)}}$$

$V_{1/2h}$ satisfies the equation $h_\infty(V_{1/2h}) = 0.5$.

η is slope factor affecting the rate of change of the inactivation variable h.

The time constants τ_m and τ_h are voltage dependent. NCS allows this dependence to be defined using an array of values for both the voltages and time constants. This is defined by (2.12).

$$\tau(V) = \begin{cases} \tau(1) & \text{if } V < V(1), \\ \tau(2) & \text{if } V < V(2), \\ \vdots & \\ \tau(n) & \text{if } V < V(n) \\ \tau(n+1) & \text{else} \end{cases} \tag{2.12}$$

The leakage current is voltage-independent and is modeled by (2.13). Notice that the driving force is expressed using the normal convention. This is the reason the leakage current is subtracted in the membrane voltage equation rather than added, as seen in the traditional membrane voltage equations.

$$I_{leak} = g_{leak}\left(V - E_{leak}\right) \tag{2.13}$$

The synaptic currents are calculated by

$$I_{syn} = \bar{g}_{syn} PSG(t)\left(E_{syn} - V\right) \tag{2.14}$$

The numerical integration scheme employed by NCS is similar to an Eulerian method however, as mentioned above a constant leak term is added to the discretized form of (2.1). To begin the current values defined above are summed

$$I_{Total} = I_M + I_A + I_{AHP} + I_{input} + I_{syn} - I_{leak} \tag{2.15}$$

The new voltage is then calculated as a combination of the defined membrane resting potential, the previously calculated membrane potential, the membrane resistance, capacitive time constant and the total currents.

$$V(t+1) = V_{rest} + \left(V(t) - V_{rest}\right)\left(1 - \frac{\Delta}{\tau_{mem}}\right) + \Delta\frac{I_{Total}}{C_n} \tag{2.16}$$

Rearranging for clarity

$$V(t+1) = V(t) + \left(V_{rest} - V(t)\right)\frac{\Delta}{\tau_{mem}} + \Delta\frac{I_{Total}}{C_n} \tag{2.17}$$

Where

$C_n = \frac{\tau_{mem}}{R_{mem}}$

R_{mem} is the defined resistance of the membrane.

τ_{mem} is the defined capacitive time constant of the membrane.

Notice the form of (2.1) in a simple Eulerian integration scheme would be

$$V(t+1) = V(t) + \Delta \frac{I_{Total}}{C_n} \tag{2.18}$$

The addition of the middle term in Equation (2.17) numerically drives the membrane voltage of the cell back to a predefined resting potential.

The different combinations of the provided sub-threshold channels are capable of replicating many different neuronal cell types. Figure 2.2 presents four different types of GABAergic interneurons found in the brain. These are modeled in NCS using the equations presented above and stimulated with $150 - 350pA$ current injection.

Examples

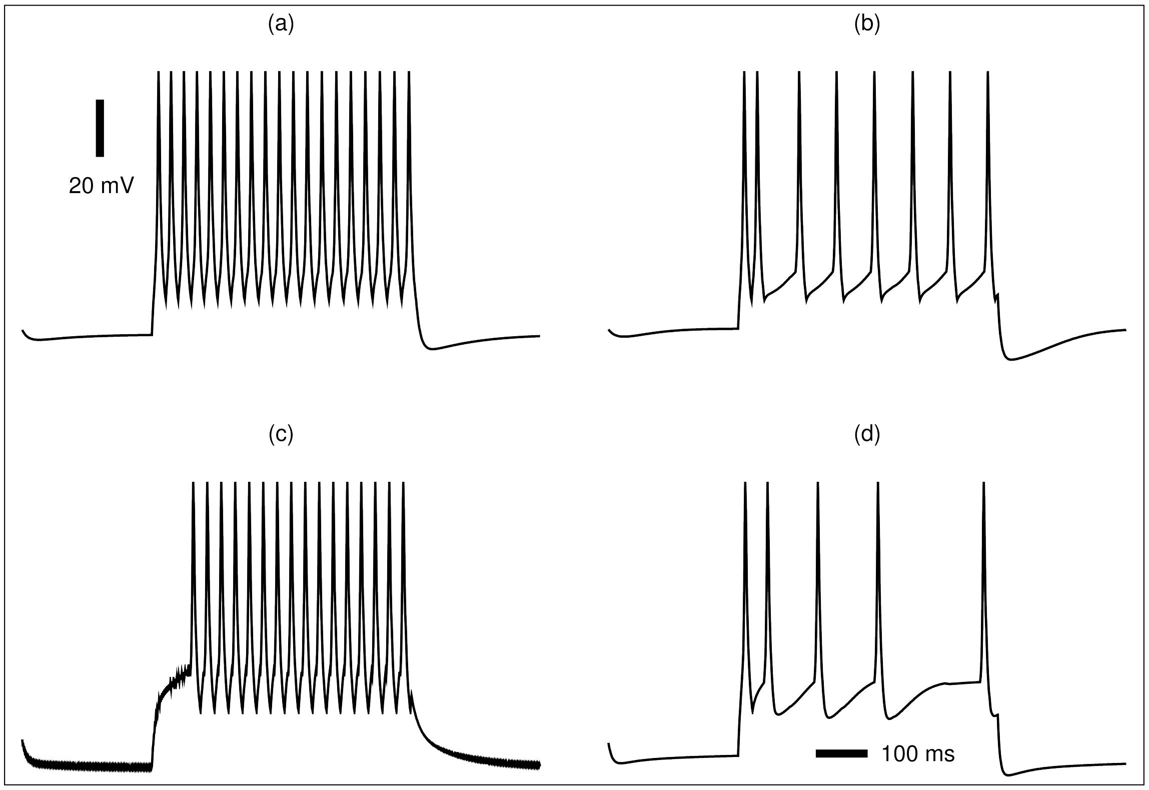

Figure 2.2: Example model neurons under a current-clamp stimulus of 150-350 pA. (a) Classic Non-Accommodating (cNAC) gabaergic interneuron. The neurons reach a steady-state fire rate on the initiation of the stimulus. (b) Bursting Non-Accommodating (bNAC) gabaergic interneuron. These neurons fire a series of bursts at the onset of the current stimulus, followed by a period of steady-state firing. (c) Delayed Non-Accommodating (dNAC) gabaergic interneuron. These are characterized by a delayed rise in membrane voltage after stimulus onset, followed by a period of steady-state firing. (d) Bursting Accommodating (bAC) gabaergic interneuron. These neurons fire a series of bursts on stimulus onset followed by a spike rate that is characterized by the adaptation or dampening of spiking in response to the constant current injection.

Chapter 3

NCSTools

Thibeault, C. M., Hegie, J., and Harris Jr., F. C. (2012). Simplifying neurorobtic development with NCSTools. In *Proceedings ISCA's 27th International Conference on Computers and Their Applications (CATA-2012), Las Vegas, NV.*

Abstract

The combination of biologically realistic neural simulations and robotic agents offers unique opportunities for both computational neuroscience and research in intelligent robotics. A concept that can provide insights into the cognitive developments involved in human robotic interaction as well as provide a pathway to developing truly intelligent agents. In this paradigm spiking neural models are coupled with physical or virtual entities in a closed-loop. The embodied agent provides stimulus to the neural model which in turn provides a filtered and processed view of that world. More often than not the complexity and corresponding computational burden of these models necessitates the use of high-performance computers that must be stored and maintained separate from the entity. In these instances, closing the loop can be an arduous task requiring intimate collaboration

between neuroscientists and engineers. Presented here is a software package for simplifying those interactions while facilitating the communication between neural simulations and abstract remote entities.

3.1 Introduction

Computational neuroscience enjoys a unique role in biological research. At one end it can help validate and quantify experimental results. While at the other, it provides a predictive mechanism for aspects of the nervous system that are unreachable by any other means. In addition to the importance mathematically modeling the nervous system has to physiological research is its value to artificial intelligence and autonomous agents.

By employing spiking neural models as the processing elements for robotic agents, researchers are attempting to explore theories that span the breadth and depth of robotics, AI and neuroscience. Understanding neurological processing often requires complex interactions with the real world (or a virtual one). This is the case in the fields of both social robotics [12, 39] and neurorobotics [10, 30, 49]. These theories generally involve the integration of several sensory modalities as well as complex commands between the agent and the neural architectures controlling it. This integration can be a complex task requiring either expertise in both computer engineering and neuroscience or collaborations between experts in each of these fields. The software, NCSTools, presented here was developed to ease the complexity of interfacing neural models with remote agents as well as abstract the neurological detail from the roboticists and the engineering detail from the neuroscientists.

This paper is laid out with the remainder of this section presenting a minimal background on spiking neural models and neurorobotics as well as introducing NCSTools. This is followed by Section 3.2, which illustrates the design choices made in this project as well as the basic software engineering behind its implementation. Section 3.3 provides a complete example of how NCSTools can be used to speed-up the task of interfacing with a spiking neural simulation. Finally, Section 3.4 concludes with some current applications where NCSTools has been utilized as well as some future directions.

The work presented here is an extension of a presentation given at the Computational Neuroscience Society Meeting in San Antonio, Texas, August 2010 [42].

3.1.1 Spiking Neural Models

Simulating excitable cells involves integrating a set of differential equations that describe the electrical activity along the membrane of the cell. The unique aspects of these cells is that once a threshold voltage has been reached, an all or nothing avalanche of electrical current occurs. This results in a spike of electrical activity, or an action potential, and initiates the communication between neurons. Its effect is felt by all neurons connected to the one that spiked, or fired. Additionally, repeated spike events can alter or grade the effect felt by those downstream neurons. This modification is referred to as synaptic level plasticity and is thought to play a major role in animal learning [38]. By combining the differential equations describing membrane voltages with the synaptic communication, computational neuroscientists hope to reveal details of the nervous system unavailable with current experimental techniques.

3.1.2 Neurorobotics

The field of neurorobotics focuses on the coupling of neural systems with some form of physical actuation. An example of this is the concept of virtual neurorobotics (VNR). This is based around the interoperability of a neural model, a virtual robotic avatar and a human participant [21, 22]. Under all but the most basic scenarios this interoperability is accomplished through an organized network communication system. For this paper an emphasis is placed on robotic and automated agents; however, it should be noted that the tools described here are by no means limited to that application.

3.1.3 NCSTools

NCSTools is a bridge between neuroscientists and engineers using NCS for research. The strength of NCSTools lies in its configuration language. It provides a mechanism for defining interactions between the neural simulation and the agent. These interactions are described using plain text strings and developers simply need to agree on the strings to develop their components. Besides the benefit that neither end is required to know intricacies of the other, this also provides a level of reuse that can reduce development time. The only requirement is that the strings remain consistent. This greatly reduces the development time of a neurorobotic application.

3.2 Design

NCSTools was developed in C++ with a focus on object-oriented design principles. The motivation for its construction was driven by the need for a replacement to the previous software package, Brainstem [35]. Although successful as a proof-of-concept, Brainstem lacked the necessary extensibility and reliability required for rapid use by researchers. Based on these inadequacies, several non-functional requirements were identified before starting the development of NCSTools. These were:

1. **Usability:** The use of NCSTools must be relatively simple. Although its intended users are scientists and engineers, its operation has to make sense based on the task.

2. **Extensibility:** The rapid pace of scientific research necessitates a system that can be readily extended to incorporate new ideas and concepts.
3. **Robustness:** The codebase must be invariant to the application and multitude of configurations.
4. **Reliability:** NCSTools is intended for researchers performing experiments in both neuroscience and robotics. If it is unreliable, meaningful results can be lost.

The component interfaces provide layered abstraction supporting an extensible yet robust code base. Similarly, the configuration language supports overall usability. Throughout this section numerical subscripts that correspond to the non-functional requirements are used to indicate how each design element supports the requirements.

Figure 3.1 presents the system level layout of NCSTools. The major components are described below.

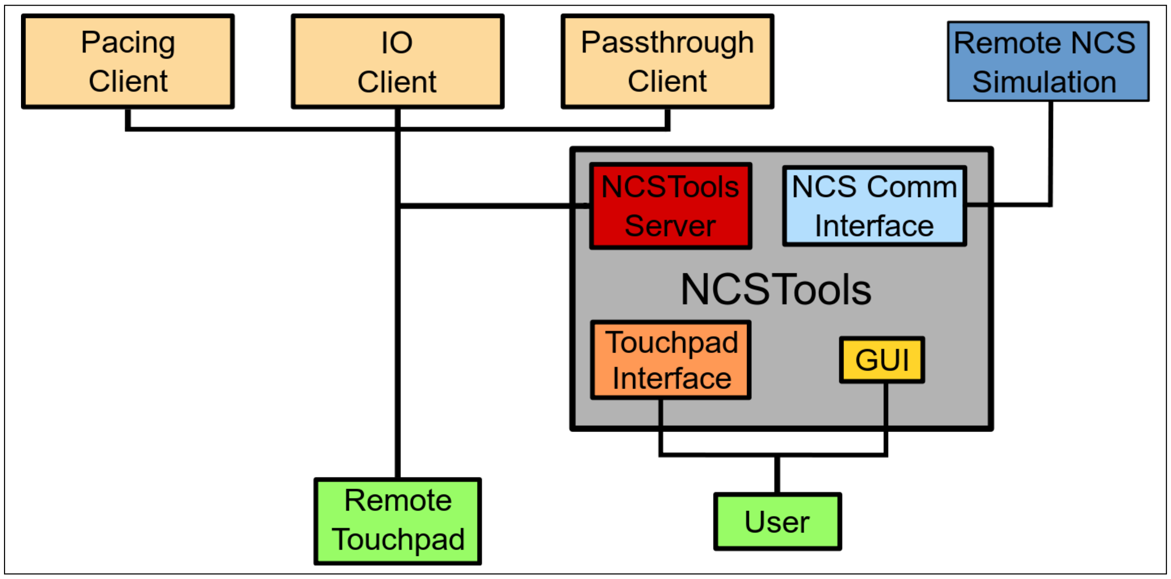

Figure 3.1: System layout. With the exception of the user interfaces, all connections are made using the various network clients described below.

3.2.1 Configuration Language$_1$

The flexibility of NCSTools lies in its configuration language. Most aspects are modifiable at runtime through the input configuration file. This includes control for the definable communication “language”, the GUI, User IO and Data processing. There are a number of configuration examples provided in the subsequent sections however, a complete listing of available options is included in the source code documentation.

3.2.2 Inputs$_{2,3}$

In the context of this paper, inputs are the signals coming from the NCS simulation specified as reports. NCS publishes population level information about the simulation to each of the reports requested. This information can be spike-counts for individual neurons, sampled synaptic weights and neuron voltages.

Input Containers

The base class for all inputs is the input container. This provides the common functionality, such as initialization and communication, as well as the interface for all derived classes.

Windowed Input Containers

Windowed input containers allow the user to specify a window of time over which the spiking activity of any number of neuron populations can be compared in a winner-takes-all pattern. The window of time is determined by how often the NCS simulation sends spiking information. There are currently two types of derived window input containers, one uses a fixed amount of time and the other uses a moving window.

The fixed time container compares the spiking activity over a static window. After the defined period of NCS reports has elapsed the most active population (the one with the

```
input:
{
  # The number of unique Report collections
  NCSReportCollections = 2;
  NCSReportCollection1: {
    num_reports = 500;
    # The total period of counting, includes
    # report collection and recovery.
    period = 500;
    # The number of report outputs.
    NCSReports = 2;
    type = "STANDARD";
    NCSReport1: {
      connection = "to_PMC1";
      command = "point_left";
      };
    NCSReport2: {
      connection = "to_PMC2";
      command = "point_right";
    };
    # Setup the plots for this group.
    plot = "YES";
    plotType = "BAR";
    plotname = "Motor Activity";
    tabIndex = 1;
    plotIndex = 1;
  };
  # This group is used just for displaying
  # information on the GUI
  NCSReportCollection2: {
    type = "RASTER";
    NCSReports = 2;
    NCSReport1: {
      connection = "display_PMC1";
      plotname = "Premotor Cortex 1";
      cells = 10;
      tabIndex = 1;
    };
    NCSReport2: {
      connection = "display_PMC2";
      plotname = "Premotor Cortex 2";
      cells = 10;
      tabIndex = 2;
    };
  };
};
```

Listing 3.1: Input configuration example

highest number of spikes) is selected. The command specified in the configuration file for that report channel is then sent to all connected clients. An example configuration for a fixed time container is presented in Listing 3.1.

The second input container uses a moving window of time. This window is fixed width but progresses in time as the simulation progresses. This continues until the most active population's activity is greater than the next most active by a user specified threshold. At that time the command associated with that report is sent out and the window is reset.

A key feature of both the inputs and the outputs, described in Section 3.2.3 below, is the ability to bind the same command string to multiple containers. This provides a mechanism for starting a coordinated series of events based on a single client command or simulation result.

In addition to sending commands to the connected agents these containers can also be used for plotting in the GUI as described below.

3.2.3 Outputs$_{2,3}$

The NCSTools outputs define the signals traveling to the NCS simulations. These are stimuli sent to populations and can be triggered by text strings from the connected clients or by the built-in touchpad interface described below.

Static Output

The stimulus sent to the neuron population is fixed and defined by the configuration file.

Dynamic Output

The stimulus sent to the neuron population is sent by the client along with the string command.

Timed Output

```
output:
{
  NCSStims = 2;
  NCSStim1: {
    type = "TIMED_OUTPUT";
    command = "saw_red";
    connection = "from_VC1";
    # The static output to send when the client
    # requests it.  This is still needed for
    # the parser even when using Dynamic mode.
    output = "0.2000";
    # How many ticks between stim inputs.
    frequency = 50;
    # The number of times to repeat the input.
    num_outputs = 10;;
  };
  NCSStim2: {
    type = "TIMED_OUTPUT";
    command = "saw_blue";
    connection = "from_childbot_VC1";
    output = "0.0000";
    frequency = 50;
    num_outputs = 10;;
  };
};
```

Listing 3.2: Output configuration example

Timed output containers are used to send the same stimulus a set number of times. Through the configuration file the user can specify how many times to send the stimulus and the interval between successive stimulus. An example of this is presented in Listing 3.2.

3.2.4 Network Communication$_{3,4}$

Aside from the user interface, the connections illustrated in Figure 3.1 represent one or multiple network communication mechanisms. NCSTools not only coordinates the connection to and from the NCS neural simulation but also provides a network server for handling client and remote agent connections. The individual components of the network communication provided by NCSTools are described below.

```
# Define the voServer Connection
voServer: {
  host = "localhost";
  port = "20003";
};
# Define the settings for the NCSTools Server
socketServer: {
  host = "0.0.0.0";
  port = "20001";
};
```

Listing 3.3: Server Configuration Example

NCSTools Server$_4$

NCSTools uses a simple POSIX socket server for client communication. The simplicity of the server helps ensure its reliability and the low-level components help guarantee that the performance of the server does not hinder the overall application.

Client Communication$_{2,3,4}$

Several client implementations are provided for C++, Python and MATLAB/Java. These provide objects that appropriately abstract the interface from the implementation to support the extensibility of both NCSTools and client applications. Both blocking and non-blocking communication is supported.

IO Clients

The IO clients are used for most applications. These provide the input and output mechanisms required to interact with NCSTools.

Pacing Clients

As neural models increase in both size and complexity they often exceed the real-time capabilities of the hardware. Pacing clients are provided to ensure that remote clients do not overfill buffers or lose track of the simulation. These connect to NCSTools to maintain synchronization and receive heartbeats that are output at user specified intervals signaling the current time in the simulation.

Passthrough Clients

Passthrough clients connect to the NCSTools server and receive messages sent to or from other clients connected to the server, including the NCS simulation. These clients can provide users with a complete landscape of the neurorobotic experiment. These can also be used to create context aware clients that can modify their behavior based on the state of the system.

voServer

Communication with NCS is facilitated by the voServer package [28]. This is a minimal publish-subscribe server that provides both binary and ascii data streams. As part of this project a C++ client was developed similar to the NCSTools clients described above. This client is used by all of the IO modules of NCSTools.

3.2.5 Graphical User Interface

The Graphical User Interface (GUI) is an option given to users for visualizing aspects of the neural model in real-time. The GUI was written as a C++ library using Qt [11]. As with all aspects of NCSTools, it is completely configurable through the input file. The user can specify each tab and what information shows up on that tab. An example GUI is given in Figure 3.2.

Plot Types

The GUI is used to visualize the state of the NCS simulation. It provides three plot types:

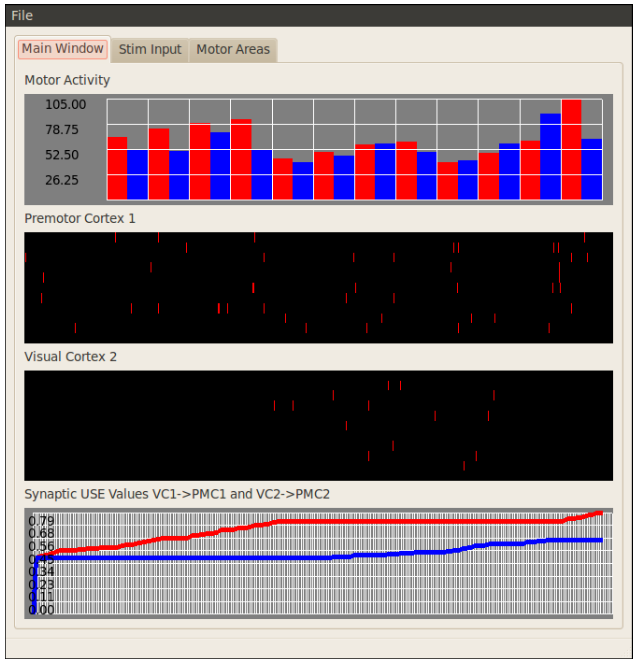

Figure 3.2: Example graphical user interface. This is a single tab that presents the three main types of plots. The tabs and plots are dynamically created based on the configuration file provided at runtime.

Bar Plots

Bar plot are used to visualize the spiking activity of competing neuron populations. It uses the derived instances of the windowed input containers, described in Section 3.2.2.

Raster Plots

Raster plots present the precise spiking activity of the neurons within a population. The X axis is the time, in units of simulation reports, and the Y axis corresponds to the neuron index relative to the population.

Line Plots

Line plots are used to plot the synaptic efficacy over time, again this is in units of simulation time.

```
gui: {
  numTabs = 3;
  tab1: {
    name = "Main Window";
    layout = "HORIZONTAL";
    num_plots = 4;
  };
  tab2: {
    name = "Stim Input";
    layout = "HORIZONTAL";
    num_plots = 2;
  };
  tab3: {
    name = "Motor Areas";
    layout = "GRID";
    num_plots = 4;
  };
};
```

Listing 3.4: GUI Configuration Example

3.2.6 Touchpad Interface

The touchpad interface provides users with a way to bind keyboard inputs, either single keys or new-line terminated strings, to stimulus and control signals. The input is entered through the command line and is fully configurable through the input language. There are three major options for the touchpad that are described below. The touchpad signals can be directed to a named NCS input or to agents connected to the NCSTools server. Similar to the inputs and outputs described above, a particular key binding can be used for any number of different commands. With this a single keyboard input to control many different aspects of the neurorobotic interaction.

Instant: When touchpad bindings are defined as instant their associated actions will occur only once and as soon as the command is received.

Repeated: The repeated bindings are analogous to the time outputs described in Section 3.2.3. These provide a way to repeat an input stimulus over a configured period of time.

```
keybinding2: {
  type = "SERVER_KEY_TIMED";
  voConnection = "v03";
  # The number to ticks to wait before sending output2.
  max_ticks = 2000;
  # The highest value to accept from the keyboard.
  max_input = 10;
  # How many outputs will this keybinding have.
  outputs = 2;
  output1: {
    ServerString = "Saw_Red_Pointed_Left";
    output1 = "SETHEBBIAN syn_EE BOTH;";
    output2 = "SETHEBBIAN syn_EE NONE;";
  };
  output2: {
    ServerString = "Saw_Blue_Pointed_Right";
    output1 = "SETHEBBIAN syn_EE BOTH;";
    output2 = "SETHEBBIAN syn_EE NONE;";
  };
};
```

Listing 3.5: Sample Touchpad Configuration

Timed: The timed bindings provide a mechanism for setting a period of silence before repeating the configured stimulus. This basically creates a window of stimulus followed by a rest period. This window is then repeated for the configured number of times.

Coupling With Input Signals

The unique aspect of the Repeated and Timed bindings is the ability to couple them to signals coming from the connected agents. When a configured signal is received from a connected agent the user is prompted for a numerical value by the touchpad interface. This value is used to determine the graded input values sent to the simulation. This is used in the example given in Section 3.3 below.

3.2.7 Control Interface

In addition to sending stimulus and receiving reports, NCSTools provides access for controlling and modifying a running neural simulation. Commands can be bound to strings from connected agents or to the built-in touchpad functionality. Some of the features this releases include the saving and loading of model states, modification of synapses, adding new stimulus paths and stopping the simulations.

3.3 Example Scenario

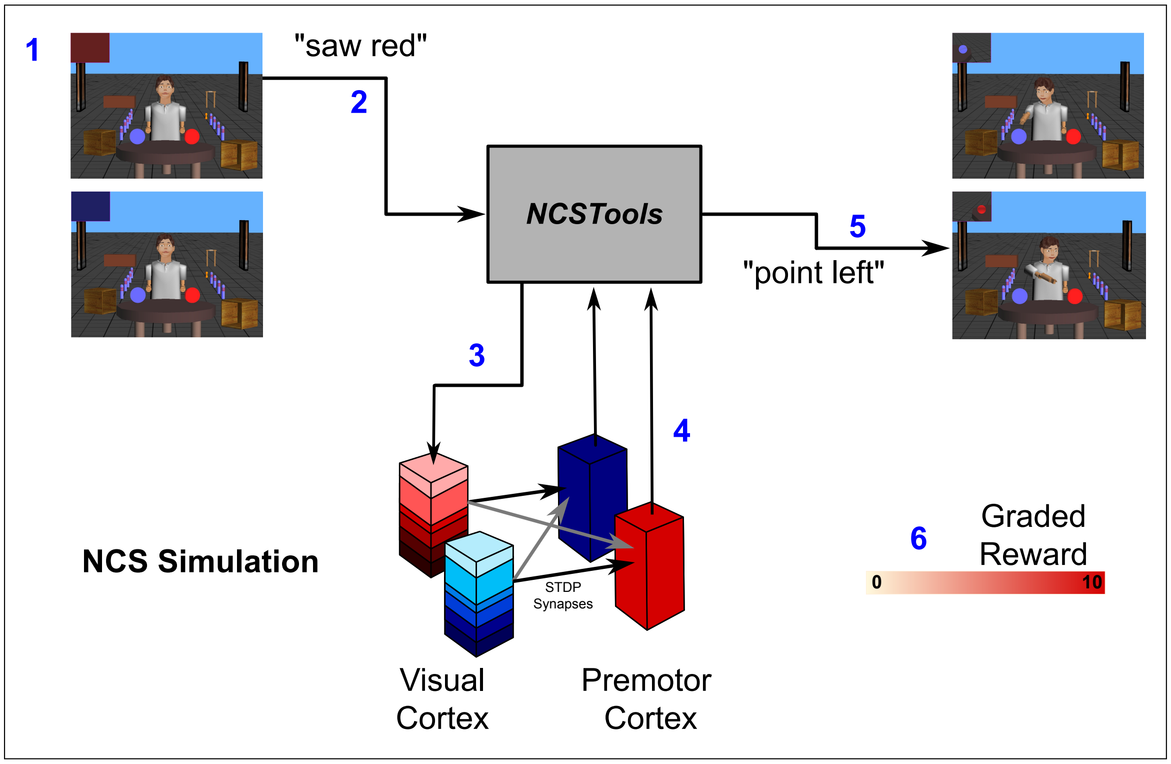

Figure 3.3: Example neurorobotic scenario.

To illustrate NCSTool's role in neurorobotic research, a motivating example is presented in Figure 3.3. In this case the remote agent is a virtual robotic avatar and the interaction is with a camera and the touchpad interface. The steps are:

1.	Camera captures image from user and dominant color is calculated.
2.	The virtual environment sends the defined plain text statement (“saw red”) to NCSTools through the server interface. This uses a static output described in Section 3.2.3.
3.	Based on the configuration NCSTools will stimulate the appropriate regions of the remote NCS Model through the NCS network interface.
4.	The activity of the two premotor regions in the model are monitored and compared as the simulation progresses. A windowed input describe in Section 3.2.2 monitors the activity.
5.	In this case a winner-takes-all calculation is computed and the appropriate plain text statement is sent through the NCSTools server interface to the robotic avatar based on the most active brain region.
6.	The user is then given the opportunity to “reward” the robot if the correct color was identified. The reward is achieved by coupling the output from the agent with the touchpad interface as described in Section 3.2.6.

Although this is a simple example there is still a significant amount of coordination involved.

3.4 Discussion

NCSTools provides a dynamic interface to the neural simulation environment NCS. The abstraction between the neural models and the robotic interface is unique to this project and there have already been several projects that have successfully leveraged NCSTools.

Oxytocin Induced Trust

The work presented by Anumandla *et al.* [2] made extensive use of NCSTools. In this project a human participant interacted with a robotic avatar through a GPU based Gabor processing application, NCSTools and a NCS simulation. The results of this work provided new theories on the role Oxytocin may play in establishing and stabilizing trust between mammals. These theories would not have been possible without a closed loop virtual neurorobotic system.

Emotional Speech Processing

Thibeault *et al.* [44] used NCSTools to coordinate the processing between a speech extraction package and a neural simulation. The speech processing algorithm successfully extracted the emotional aspects of a person's speech pattern to determine the reward stimulus to inject into the model.

Virtual Reality

As part of an unpublished proof-of-concept, NCSTools was utilized for the large-scale visualization of a running neural simulation. The neural model was constructed within NCS along with a corresponding X3D model. The visualization software created a virtual representation of the neuron populations. Through the network interface the voltages of the cells within the model were collected and as the simulation progressed the model neurons would change color to represent the voltage of the cell. In addition, 3D sound was used to signal when a spike was fired. The package was tested successfully on a 3D wall and a 6 sided Cave Automatic Virtual Environment (CAVE).

Future Directions

There are several directions that have been identified for future development of NCSTools.

Real-time Structural Modifications

Providing a mechanism for users to modify aspects of the model including the type of neuron and the connectivity, will greatly increase the rate at which different models can be evaluated. In addition, this would provide a mechanism for actively modifying neurogenesis (the addition of new neurons), and synaptogenesis (the dynamic addition and removal of synaptic connections).

Cluster-Aware Version

The complexity of large-scale neural models generally requires distributed compute clusters. By creating a cluster-aware version of NCSTools, multiple instances can run in parallel while still coordinating communication between instances.

Dynamic Server Configuration

The current version is only modifiable on initialization. Allowing users to modify the configurable aspects of NCSTools will make it a more appealing tool for neurorobotics.

Chapter 4

Emotional Speech Processing

Thibeault, C. M., Sessions, O., Goodman, P. H., and Harris Jr., F. C. (2010b). Real-time emotional speech processing for neurorobotics applications. In *Proceedings ISCA's 23rd International Conference on Computer Applications in Industry and Engineering (CAINE-2010), Las Vegas, NV.*

Abstract

The ability for humans to understand and process the emotional content of speech is unsurpassed by simulated intelligent agents. Beyond the linguistic content of speech are the underlying prosodic features naturally understood by humans. The goal of emotional speech processing systems is to extract and classify human speech for these so called paralinguistic elements. Presented here is a proof-of-concept system designed to analyze speech in real-time for coupled interactions with spiking neural models. Based on proven feature extraction algorithms, the resulting system provides two interface options to running simulations on the NeoCortical Simulator. Some basic tests using new recordings as well as a subset from a published emotional database were completed with promising results.

4.1 Introduction

Much of human communication is not in what is said but how it is spoken. These subtle changes in emotion that exist beyond the linguistic aspects and the perceptual ability to interpret them is fundamental to speech. There have been many studies aimed at parameterizing and classifying such emotions. While many of these investigations have taken advantage of advances in neural networks as well as statistical and probabilistic classification mechanisms, the authors are unaware of such studies employing biologically realistic neural networks. These so called spiking networks strive to model neurons and neural assemblies with as much biological realism as is computationally feasible. The combination of biological neural networks and high-level speech processing proposes a unique opportunity to explore some of the possible neural mechanisms behind emotional expression. Additionally, these networks may aid in the creation of more successful emotional classification tools. This project is a first step towards the combination of emotional speech processing (ESP) and computational neuroscience in a real-time architecture that can be easily modified and extended.

Theory

There has been a wealth of research on the extraction of emotional information from speech. Unfortunately, this work has yet to identify a standard set of features that completely identify a signal. The most common features found in the literature deal with pitch. Additionally, the energy, rate, and frequency components of a signal have been employed with varying rates of success.

Acoustic Properties

Of the acoustic properties researched by the community, pitch appears to be one of the more popular. A product of the tension of the vocal-cords and the corresponding air pressure, pitch changes over time show a direct relation to some basic emotions (e.g. anger and happiness) [17]. Ververidis *et al.* [45], presented a collection of how many acoustic features, including pitch, correspond to some of the basic emotions. Similarly, the intensity of a signal can be used to classify emotional content of a signal.

Although there is considerable empirical evidence supporting the classification of a speech signal based on the acoustic features, there is also wide variation between these studies. Additionally, it has been argued that these features really only provide information about the arousal state of the speaker, rather than their true emotional state [17].

Previous Work

Previous studies on emotional speech recognition use methods ranging from frequency analysis, segmental analysis or prosodic features, as well as analysis of the signal intensity. For comprehensive reviews of the current literature of these methods see [45] and [6]. From these reviews, it becomes obvious that the concept of extracting emotional information independent of the linguistic content is not new. Additionally, there have been several applications identified for these types of classification systems. Some examples are presented in the discussion of Section 4.5 below.

Contribution of This Project

This paper presents an emotional speech processing system offering both real-time performance and a simple programming interface. The remainder of the paper continues with

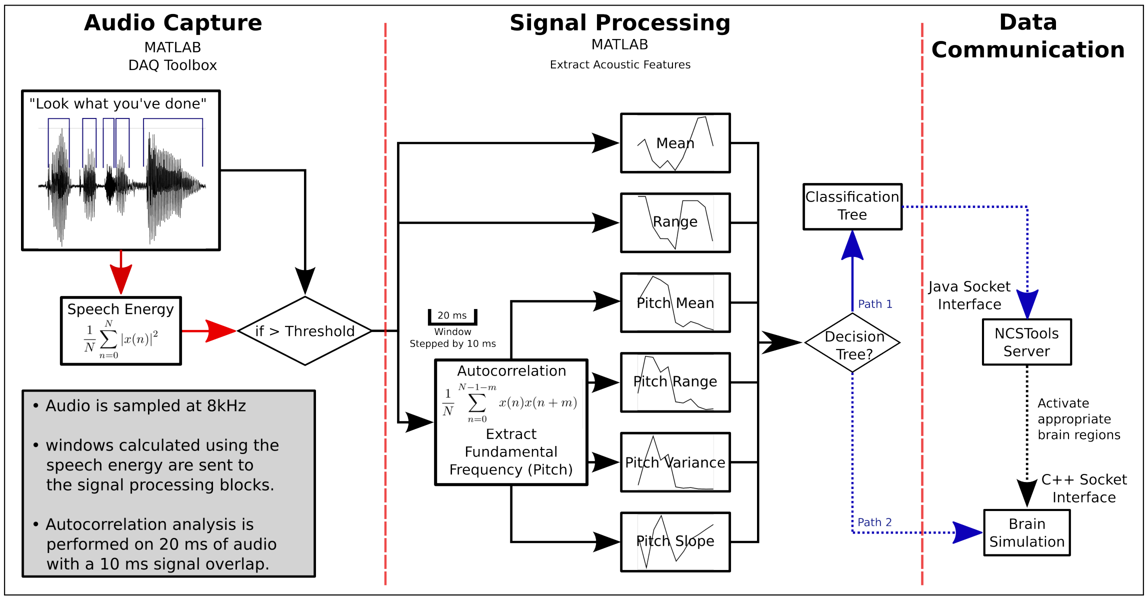
Audio Capture
MATLAB
DAQ Toolbox
"Look what you've done"
Speech Energy
$\frac{1}{N}\sum_{n=0}^{N}|x(n)|^2$
if > Threshold
• Audio is sampled at 8kHz
• windows calculated using the speech energy are sent to the signal processing blocks.
• Autocorrelation analysis is performed on 20 ms of audio with a 10 ms signal overlap.
Signal Processing
MATLAB
Extract Acoustic Features
20 ms
Window
Stepped by 10 ms
Autocorrelation
$\frac{1}{N}\sum_{n=0}^{N-1-m} x(n)x(n+m)$
Extract Fundamental Frequency (Pitch)
Mean
Range
Pitch Mean
Pitch Range
Pitch Variance
Pitch Slope
Classification Tree
Path 1
Decision Tree?
Path 2
Data Communication
Java Socket Interface
NCSTools Server
Activate appropriate brain regions
C++ Socket Interface
Brain Simulation

Figure 4.1: Speech Processing System

Section 4.2, describing the overall system design. Section 4.3 describes the initial testing completed over the course of development. With Section 4.4 giving the results of those tests. Finally, Section 4.5 concludes with a brief discussion of its applications as well as future work.

4.2 Emotional Speech Processing System

The overarching goals of this project were to develop a complete speech processing system that could not only perform in soft real-time but could be modified and extended by users with limited programming skills. The MATLAB environment was chosen mostly for its ease of use. Its interpreted language processing can often be a disadvantage, however for this project some basic optimizations facilitated real-time performance.

The system consisted of three processing sections: Audio Capture, Signal Processing and Data Communication. The speech processing system is diagrammed in Figure 4.1 and explained further below.

4.2.1 Data Capture

The audio stream is captured using the Data Acquisition Toolbox for MATLAB, developed by The Mathworks Inc. Data is captured in 1 second blocks before being sent to a custom callback function. The Analysis begins with an extraction of the significant segments in the speech signal. There are many different methods for segmentation, see [36, 47], here, a simple speech energy calculation is used.

The speech energy of a signal is an average sum of squares calculation that can represent the overall energy in a window of speech. The speech energy can be employed for distinguishing actual speech signals from background noise. Although not employed in this system, the speech energy can additionally be used to help classify the emotional arousal level of the speaker. It can be calculated by Equation (4.1).

$$E(m) = \frac{1}{N}\sum_{n=0}^{N} |x(n)|^2 \tag{4.1}$$

Where N is the number of samples, $x(n)$ is the measured value at time n with respect to the current window and m is the current window being processed. This calculation is completed on 20ms windows of data. The results are stored and compared to a user defined threshold. When that threshold is reached the system will begin extracting sampled data until the threshold is crossed again. Finally, the extracted window of data is sent to the signal processing blocks. Any left-over data will be retained and attached to subsequent recordings.

4.2.2 Signal Processing

The feature extraction begins with a calculation of the mean and range of the raw intensity values. The segment is then run through an autocorrelation algorithm and the fundamental frequencies of 20ms windows is computed. The window is stepped by 10ms allowing overlap of pitch calculations.

Autocorrelation Analysis

The autocorrelation calculation has been shown to faithfully extract the fundamental frequency (pitch) of speech signals. In general the autocorrelation calculation is not appropriate for any continuous function. However, by taking piece-wise windows of the continuous signal, the stationarity assumption can be applied to the individual window [34]. This assumption allows the use of sample autocorrelation calculations on continuous signals and illustrates its appropriateness for this application.

The autocorrelation of a windowed signal, $x(n)$ with N samples, can be defined as Equation (4.2) [19, 34].

$$R(m) = \frac{1}{N} \sum_{n=0}^{N-1-m} x(n)x(n+m) \tag{4.2}$$

Requiring programmatic loops, this can be a computationally expensive calculation. That cost can be reduced by considering the calculation as an ordinary convolution. The autocorrelation can then be computed using the periodogram spectrum defined as Equation (4.3) [34].

$$S(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x(n)e^{-j\omega n} \right|^2 \tag{4.3}$$

This is sometimes referred to as the short-time spectrum. As defined by the Wiener-

Khinchin theorem the combination of $S(\omega)$ and $R(m)$ are a simple Fourier-Transform pair. Additionally, $R(m)$ can be redefined as Equation (4.4) [19].

$$R(m) = \int_{-\pi}^{\pi} S(\omega) cos\omega m, d\omega \tag{4.4}$$

Finally, utilizing the FFT and IFFT functions the autocorrelation of the window can be efficiently calculated using Equation (4.5) [19, 34].

$$R(m) = \frac{1}{N} IFFT\left(|FFT\left(x(n)\right)|^2\right) \tag{4.5}$$

The fundamental frequency, $F0$, of the resulting signal will be represented by the lag location with the greatest amplitude. For emotional speech, the lag is restricted to a range between 50 and 500Hz; this corresponds to the region of pitch perceivable by humans [45].

Four statistical features are extracted from the fundamental frequency analysis. This begins with the mean, range and variance of $F0$. Finally, the slope of the pitch is calculated and fundamental frequency slopes greater than 70Hz are filtered out.

The data capture and signal processing will continue for a user-definable period after the first segment is detected. In testing it was found that a 2-3 second window of processing was sufficient.

At this point the extracted features can be sent to one of two different communications units described in detail below.

4.2.3 Data Communication

There are two options for interacting with an NCS simulation. The choice depends on the role the speech system is playing in a particular investigation.

Path 1

The first communication path is a Java based socket interface from MATLAB to a NCSTools server [42]. NCSTools is a C++ based software system that provides a number of mechanisms for communicating with a running NCS simulation. NCSTools accepts plain text strings from any number of clients connected through the built-in socket server. Through a custom configuration file, users can assign these strings to input stimulus to, or simulation controls of, a running NCS instance. Similarly, NCSTools can be configured to process simulation reports in a number of different ways. The results of which can be sent to connected clients through the server interface. This allows designers of remote tools to interface with a neural-simulation in a way that abstracts them from the details of the model. Thus providing a mechanism of reuse without modification for different models; only the NCSTools configuration needs to be changed.

The use of this path was intended but certainly not limited to coupling with the statistical classification output of the ESP system. As the data is categorized the results can be sent to NCSTools. NCSTools can then activate defined brain regions or dynamically modify Hebbian learning rules. This paradigm provides a means for verbally interacting with a neurorobotic avatar. Additionally, the classified verbal responses can be used for rewarding correct behaviors and discouraging those considered incorrect.

Path 2

The second communication option is a direct socket interface to a running NCS simulation. Similar to the direct NCSTools option above, this is comprised of a Java client interface that connects to the C++ server implementation. This option facilitates future feature classification methods employing biologically realistic spiking neural models; a combination that has significant potential for researchers of both computational neuroscience and speech recognition.

4.3 Testing

To demonstrate the system's capabilities a series of tests were completed. The real-time capture and processing was verified using MATLAB's built-in timing tools. Additionally, MATLAB's Data Acquisition Toolbox provided mechanisms for reporting when the sampled data was not removed before the next sample was ready. Tests were performed on both the classified result (Path 1) and direct data connection (Path 2).

English Recordings

Some initial analysis was completed with non-trained participants speaking the emotionally ambiguous phrase, "Look what you've done,“ while expressing two basic emotions, Joy and Disappointment. These recordings were not analyzed by trained reviewers, so the emotion was categorized by the researcher team only. These tests, however, did provide an excellent test facility for development of the system.

Berlin Emotional Speech Database

Finally, to show how this system can perform on standardized data, a portion of the Berlin Database of Emotional Speech was used [8]. Although in German, EmoDB provides an excellent reference for testing emotional speech algorithms. In emotional speech research there is a lack of freely available databases, especially in English. It was for this reason that EmoDB was utilized.

There is considerable evidence that the classification of the extracted features described above are dependent on the gender of the speaker [5, 45, 48]. This motivated the use of only male recordings for initial testing purposes. Similarly, the performance of many emotional speech recognition systems show a strong speaker dependence [5, 45]. In this project however, it was decided multiple speakers would be allowed but the range of emotions was limited to Anger and Fear. Generally as more emotions are added to the classification system the accuracy will decrease [27, 32, 36]. However, Wu *et al.* [52], accomplished higher recognition rates of 7 of the Berlin Database emotions (disgust was excluded) using long-term spectro-temporal features. Similar results were accomplished by Vlasenko *et al.*, [46], using a combination Gaussian Mixture and Support Vector Machine. The computational cost of these methods would require further investigation for inclusion in real-time system similar to that proposed here. The purpose was not to demonstrate superiority over existing systems but to merely illustrate a classification scheme that can perform accurately in real-time.

Five recordings from both the Anger and Fear sets were randomly selected. The remaining recordings were then analyzed using the emotional speech recognition system. It should be noted here that the recordings were used directly and not sampled by the data acquisition toolbox. As the results of the first tests demonstrated, the system does operate in real-time. After the speech features were extracted, a classification and regression tree was constructed in MATLAB based on the results. Having been successfully employed by other emotional speech processors [40], classification trees can be simple to construct and allow the inclusion of other features not utilized here (e.g. linguistic aspects of the speech signal).

Each segment, as detected by the system, was classified and used as training data in the tree construction. The 10 recordings were then analyzed using the classification tree. Unlike the two to three second analysis period activated by the normal system, the entire

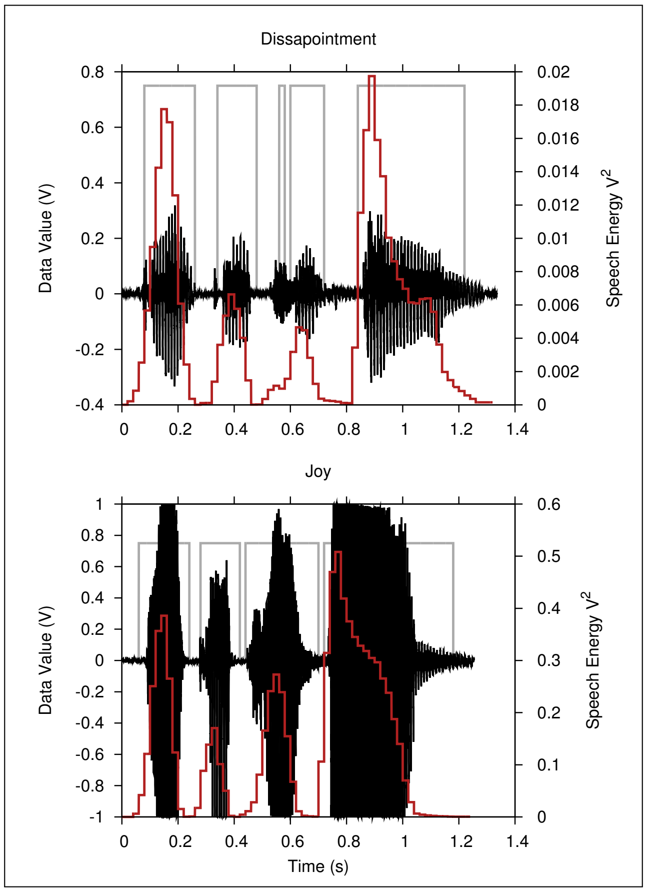

Figure 4.2: Results of male participant speaking the phrase “Look what you’ve done” with the acted emotions, joy and disappointed. Raw data is plotted with black lines. The Speech Energy is plotted in Red. The selected segments are framed by the gray Lines.

recording was used. Each segment was classified, with the result stored locally. When the recording was complete, the system would select the dominant emotion in the recording and send the result to the NCSTools server.

4.4 Results

The results for the English recordings are included here as an illustration of the concepts presented above. In Figure 4.2 the black lines represent the raw data recorded during the

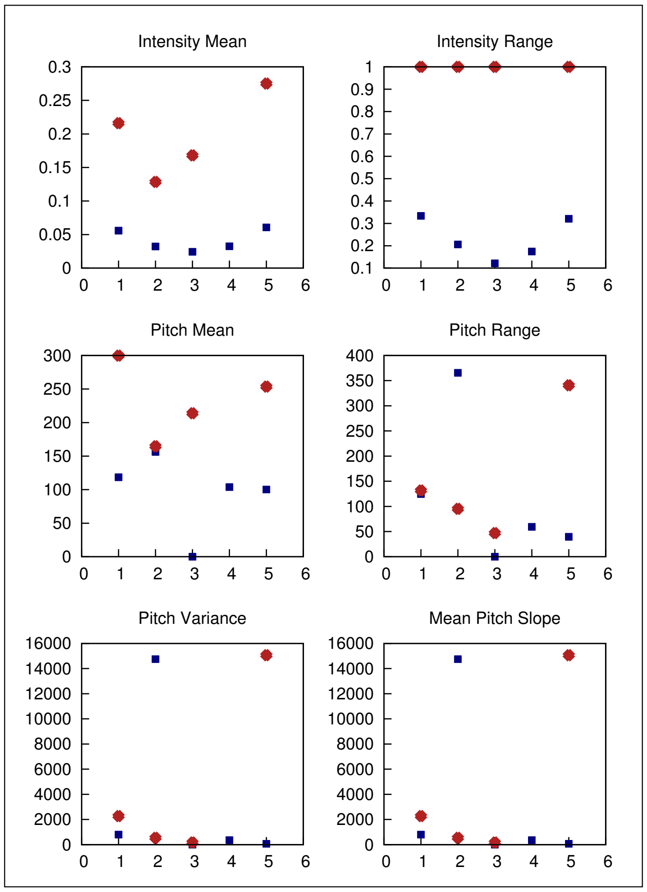

Figure 4.3: Extracted features of male participant. Values from recording classified as Joy are marked in red. The Disappointment values are marked in blue. The x-axis corresponds to the utterance as outlined in Figure 4.2.

training session. The red lines are the speech energy of the signal as calculated by Equation (4.1) and the gray lines frame the automatically calculated segments selected for analysis. The extracted features are given in Figure 4.3. The red marks are from recordings classified as Joy with the blue representing Disappointment. For the intensity mean, intensity range, and pitch mean measurements there is a clear visual separation between the two emotions. The utility of the other measurements, at least in terms of a visual analysis, is unclear. It is likely that trends specific to each emotion will emerge with longer samples as well as more of them.

Table 4.1: Results of Berlin Database [8] Testing.

		Actual	
		Fear	**Anger**
Predicted	**Fear**	4	2
	Anger	1	3

Test results with the Berlin Database were promising. Of the 10 randomly selected recordings, 3 were classified incorrectly. With the ESP system correctly identifying 3 out 5 recordings labeled as angry and 4 out 5 as fear correctly. This is summarized in the confusion matrix labeled Table 4.1. Although only two emotions were trained and classified, this result still illustrates the potential of the design.

4.5 Conclusion

A unique real-time emotional speech recognition system was presented with some promising test results. As a proof-of-concept, this project's results are encouraging and have provided evidence that future work and extended applications will be possible.

Although the system is currently not perfect, there are number of additions that can be made to improve the overall classification performance. Using one of the alternative classifiers should boost the performance and allow for the classification of more than just two emotions. However, adding additional complexity to the classifier will affect performance; a balance between accuracy and performance must be made. For the application presented in Section 3.3, the this classifier would likely be sufficient. Because that decision system is itself noisy, variations in the reward would be easily tolerated and compensated for. A more rigid paradigm however, would require a better performing classifier.

Table 4.2: Incorrectly classified recordings from the Berlin Database [8].

Code	Speaker Info	German Phrase	English Translation	Emotion	Prediction
03b10Ab	male, 31 years	*Die wird auf dem Platz sein, wo wir sie immer hinlegen*	*It will be in the place where we always store it*	Fear	**Anger**
10b02Aa	male, 32 years	*Sie haben es gerade hochgetragen und jetzt gehen sie wieder runter*	*They just carried it upstairs and now they are going down again*	Fear	Fear
11b09Ad	male, 26 years	*Ich will das eben wegbringen und dann mit Karl was trinken gehen*	*I will just discard this and then go for a drink with Karl*	Fear	Fear
12b02Ad	male, 30 years	*Sie haben es gerade hochgetragen und jetzt gehen sie wieder runter*	*They just carried it upstairs and now they are going down again*	Fear	Fear
15a04Ac	male, 25 years	*Heute abend knnte ich es ihm sagen*	*Tonight I could tell him*	Fear	Fear
03a02Wb	male, 31 years	*Das will sie am Mittwoch abgeben*	*She will hand it in on Wednesday*	Anger	**Fear**
10a04Wb	male, 32 years	*Heute abend knnte ich es ihm sagen*	*Tonight I could tell him*	Anger	Anger
11b02Wb	male, 26 years	*Das will sie am Mittwoch abgeben*	*She will hand it in on Wednesday*	Anger	Anger
12a07Wa	male, 30 years	*In sieben Stunden wird es soweit sein*	*In seven hours it will be*	Anger	Anger
15b01Wc	male, 25 years	*Was sind denn das fr Tten, die da unter dem Tisch stehen*	*What about the bags standing there under the table*	Anger	**Fear**

4.5.1 Applications

Beyond the applications to computational neuroscience and neurorobotics discussed previously, several applications for emotional processing have been identified by other researchers. These include call center monitoring [1, 31, 36], Human-Computer Interaction [27], aircraft pilot monitoring [45], and as a therapist diagnostics tool [45]. A possible addition could be applications in law enforcement. The ability to analyze the emotional state of both officers and civilians could provide law enforcement agents with a tool useful in both investigations and stressful situations.

4.5.2 Future Work

As this project progresses from a proof-of-concept to a functional research tool, there are several additions that need to be considered. Some of the more successful techniques referenced here will be considered as replacements for the current algorithms. In addition, novel classification and segmentation concepts must be explored and integrated into the ESP system.

With more comprehensive algorithms the computational cost will inevitably increase. This will eventually lead to a loss of real-time performance and the need to explore new hardware and software platforms.

Chapter 5

Discussion

Despite the benefits of embodied modeling there is a lack of formalized tools for actively researching it. The first contribution of this thesis presented a tool to ease the burden of development on researchers. Although this is an important step, it is still not a solution that can fully support all aspects of neurorobotics. Within the field of computational neuroscience there has been considerable emphasis placed on model interoperability [13]. This movement has gained traction within the community. However, neurorobotics is relatively new, and the idea of standardized components has not yet been fully developed. As these ideas are solidified an obvious extension of the work presented in Chapter 3 would be to support the key standardized inputs.

The second contribution of this thesis dealt with an empirically supported theory of behavior that lacked a clear neural basis. The concept of rationalizing the emotion behind what a person is saying is clear at the cognitive level and this can be a vital component to a neural model. However, incorporating that theoretical knowledge into an embodied agent can be a difficult task. The work presented in Chapter 4 illustrated the process of integrating a high-level behavioral function with a spiking neural-model.

There are number of ways that the performance of the ESP system can be extended. AS mentioned above, increasing the complexity of the classification system will result in improved accuracy. In addition to this, more information about the speech signal can be extracted and used for classification. Some of these features are reviewed in Ververidis *et al.* [45].

Still, all of these deal with the subglottal waveforms; it is not what the speaker is saying that is important but how they are saying it. Coupling this ESP with speech recognition would allow for a complete analysis of the speaker's emotional state. This would require considerably more computational power but in many applications this level of analysis would be required (i.e. aircraft pilot monitoring).

Although, the field of neurorobotics is still new, there is considerable interest in developing brain-based systems. These neurologically inspired designs may someday yield agents that can be truly considered intelligent.

Bibliography

[1] Ang, J., Dhillon, R., Krupski, A., Shriberg, E., and Stolcke, A. (2002). Prosody-based automatic detection of annoyance and frustration in human-computer dialog. In *Proc. ICSLP 2002*, pages 2037–2040.

[2] Anumandla, S., Bray, L. J., Thibeault, C. M., Hoang, R. V., Dascalu, S., Harris, F., and Goodman, P. (2011). Modeling oxytocin induced neurorobotic trust and intent recognition in human-robot interaction. In *International Joint Conference on Neural Networks, IJCNN*, pages 3213–3219.

[3] Asada, M., Hosoda, K., Kuniyoshi, Y., Ishiguro, H., Inui, T., Yoshikawa, Y., Ogino, M., and Yoshida, C. (2009). Cognitive developmental robotics: A survey. *Autonomous Mental Development, IEEE Transactions on*, 1(1):12 –34.

[4] Avery, M. C., Nitz, D. A., Chiba, A. A., and Krichmar, J. L. (2012). Simulation of cholinergic and noradrenergic modulation of behavior in uncertain environments. *Frontiers in Computational Neuroscience*, 6(5).

[5] Benzeghiba, M., Mori, R. D., Deroo, O., Dupont, S., Erbes, T., Jouvet, D., Fissore, L., Laface, P., Mertins, A., Ris, C., Rose, R., Tyagi, V., and Wellekens, C. (2007). Automatic speech recognition and speech variability: A review. *Speech Communication*, 49:763–786.

[6] Bitouk, D., Verma, R., and Nenkova, A. (2010). Class-level spectral features for emotion recognition. *Speech Communication*, 52(7-8):613 – 625.

[7] Brette, R., Rudolph, M., Carnevale, T., Hines, M., Beeman, D., Bower, J. M., Diesmann, M., Morrison, A., Goodman, P. H., Harris, Jr., F. C., Zirpe, M., Natschlager, T., Pecevski, D., Ermentrout, B., Djurfeldt, M., Lansner, A., Rochel, O., Vieville, T., Muller, E., Davison, A. P., El Boustani, S., and Destexhe, A. (2006). Simulation of networks of spiking neurons: A review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398.

[8] Burkhardt, F., Paeschke, A., Rolfes, M., Sendlmeier, W., and Weiss, B. (2005). A database of german emotional speech. In *Interspeech 2005*, pages 1517–1520.

[9] Cox, B. and Krichmar, J. (2009). Neuromodulation as a robot controller. *Robotics Automation Magazine, IEEE*, 16(3):72–80.

[10] Cuperlier, N., Quoy, M., and Gaussier, P. (2007). Neurobiologically inspired mobile robot navigation and planning. *Frontiers in Neurorobotics*, 1(3).

[11] Dalheimer, M. (2002). *Programming with Qt (2nd ed.)*. O'Reilly Media.

[12] Dautenhahn, K. (2007). Socially intelligent robots: dimensions of humanrobot interaction. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1480):679–704.

[13] Djurfeldt, M. and Lansner, A. (2007). Workshop report: 1st INCF workshop on large-scale modeling of the nervous system.

[14] Duch, W. (2007). Computational models of dementia and neurological problems. *Methods in Biology*, 401:305–336.

[15] Feng, X.-J., Shea-Brown, E., Greenwald, B., Kosut, R., and Rabitz, H. (2007). Optimal deep brain stimulation of the subthalamic nucleus a computational study. *Journal of Computational Neuroscience*, 23(3):265–282.

[16] Fleischer, J. G., Gally, J. A., Edelman, G. M., and Krichmar, J. L. (2007). Retrospective and prospective responses arising in a modeled hippocampus during maze navigation by a brain-based device. *Proceedings of the National Academy of Sciences*, 104(9):3556–3561.

[17] Fragopanagos, N. and Taylor, J. (2005). Emotion recognition in human-computer interaction. *Neural Networks*, 18(4):389 – 405.

[18] Frank, M. (2005). Dynamic dopamine modulation in the basal ganglia: A neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. *Journal of Cognitive Neuroscience*, 17(1):51–72.

[19] Furui, S. (2001). *Digital Speech Processing, Synthesis, and Recognition*. Marcel Decker, Inc.

[20] Gangadhar, G., Joseph, D., and Chakravarthy, V. (2008). Understanding parkinsonian handwriting through a computational model of basal ganglia. *Neural computation*, 20(10):2491–2525.

[21] Goodman, P. H., Buntha, S., Zou, Q., and Dascalu, S.-M. (2007). Virtual neurorobotics (VNR) to accelerate development of plausible neuromorphic brain architectures. *Frontiers in Neurorobotics*, 1(1).

[22] Goodman, P. H., Zou, Q., and Dascalu, S.-M. (2008). Framework and implications of virtual neurorobotics. *Frontiers in Neuroscience*, 2(7):123–128.

[23] Guo, Y., Rubin, J., McIntyre, C., Vitek, J., and Terman, D. (2008). Thalamocortical relay fidelity varies across subthalamic nucleus deep brain stimulation protocols in a data-driven computational model. *Journal of Neurophysiology*, 99(3):1477–1492.

[24] Hauptmann, C., Popovych, O., and Tass, P. (2005). Effectively desynchronizing deep brain stimulation based on a coordinated delayed feedback stimulation via several sites: a computational study. *Biological Cybernetics*, 93(6):463–470.

[25] Hodgkin, A. L. and Huxley, A. F. (1952). Propagation of electrical signals along giant nerve fibres. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 140(899):177–183.

[26] Jayet Bray, L. C., Anumandla, S. R., Thibeault, C. M., Hoang, R. V., Goodman, P. H., Dascalu, S. M., Bryant, B. D., and Harris Jr., F. C. (2012). Real-time human-robot interaction underlying neurorobotic trust and intent recognition. *Neural Networks*, 32:130 – 137.

[27] Kim, E. H., Hyun, K. H., Kim, S. H., and Kwak, Y. K. (2009). Improved emotion recognition with a novel speaker-independent feature. *Mechatronics, IEEE/ASME Transactions on*, 14(3):317 –325.

[28] King, J. G. (2005). Brain communication server: A dynamic data transferal system for a parallel brain simulator. Master's thesis, University of Nevada, Reno.

[29] Koch, C. and Segev, I., editors (1998). *Methods in Neuronal Modeling*. The MIT Press, Cambridge.

[30] Krichmar, J., Seth, A., Nitz, D., Fleischer, J., and Edelman, G. (2005). Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. *Neuroinformatics*, 3:197–221. 10.1385/NI:3:3:197.

[31] Lee, C. M. and Narayanan, S. S. (2005). Towards detecting emotions in spoken dialogs. *IEEE Transactions on Speech and Audio Processing*, 13:293–303.

[32] McGilloway, S., Cowie, R., Douglas-Cowie, E., Gielen, S., Westerdijk, M., and Stroeve, S. (2000). Approaching automatic recognition of emotion from voice: A rough benchmark. In *In SpeechEmotion-2000*, pages 207–212.

[33] Migliore, M., Cannia, C., and Canavier, C. (2008). A modeling study suggesting a possible pharmacological target to mitigate the effects of ethanol on reward-related dopaminergic signaling. *Journal of Neurophysiology*, 99(5):2703–2707.

[34] Orfanidis, S. J. (1985). *Optimum Signal Processing An Introduction*. MacMillian Publishing Company.

[35] Peng, Q. (2006). Brainstem: A neocortical simulator interface for robotic studies. Master's thesis, University of Nevada, Reno.

[36] Petrushin, V. A. (1999). Emotion in speech: Recognition and application to call centers. In *Artificial Neural Networks in Engineering (ANNIE 99)*, volume 1, pages 7–10.

[37] Pirini, M., Rocchi, L., Sensi, M., and Chiari, L. (2008). A computational modeling approach to investigate different targets in deep brain stimulation for parkinson's disease. *Journal of Computational Neuroscience*, 26(1):91–107.

[38] Purves, D., Augustine, G. J., Fitzpatrick, D., Hall, W. C., LaMantia, A.-S., McNamara, J. O., and White, L. E. (2008). *Neuroscience*. Sinauer Associates, inc.

[39] Scheutz, M., Schermerhorn, P., Kramer, J., and Anderson, D. (2007). First steps toward natural human-like hri. *Autonomous Robots*, 22:411–423. 10.1007/s10514-006-9018-3.

[40] Tao, J., Kang, Y., and Li, A. (2006). Prosody conversion from neutral speech to emotional speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(4):1145 –1154.

[41] Terman, D., Rubin, J., Yew, A., and Wilson, C. (2002). Activity patterns in a model for the subthalamopallidal network of the basal ganglia. *The Journal of Neuroscience*, 22(7):2963–2976.

[42] Thibeault, C., Harris, F., and Goodman, P. (2010a). Breaking the virtual barrier: real-time interactions with spiking neural models. *BMC Neuroscience*, 11:1–2.

[43] Thibeault, C. M., Hegie, J., and Harris Jr., F. C. (2012). Simplifying neurorobtic development with NCSTools. In *Proceedings ISCA's 27th International Conference on Computers and Their Applications (CATA-2012), Las Vegas, NV.*

[44] Thibeault, C. M., Sessions, O., Goodman, P. H., and Harris Jr., F. C. (2010b). Real-time emotional speech processing for neurorobotics applications. In *Proceedings ISCA's 23rd International Conference on Computer Applications in Industry and Engineering (CAINE-2010), Las Vegas, NV.*

[45] Ververidis, D. and Kotropoulos, C. (2006). Emotional speech recognition: Resources, features, and methods. *Speech Communication*, 48(9):1162–1181.

[46] Vlasenko, B., Schuller, B., Wendemuth, A., and Rigoll, G. (2007). Combining frame and turn-level information for robust recognition of emotions within speech. In *INTERSPEECH-2007*, pages 2225–2228.

[47] Vogt, T. and André, E. (2005). Comparing feature sets for acted and spontaneous speech in view of automatic emotion recognition. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*, pages 474–477. IEEE.

[48] Vogt, T. and Andr, E. (2006). Improving automatic emotion recognition from speech via gender differentiation. In *In Proc. Language Resources and Evaluation Conference (LREC 2006)*.

[49] Wiener, S. I. and Arleo, A. (2003). Persistent activity in limbic system neurons: neurophysiological and modeling perspectives. *Journal of Physiology-Paris*, 97(4-6):547 – 555.

[50] Wilson, E. C. (2001). Parallel implementation of a large scale biologically realistic neocortical neural network simulator. Master's thesis, University of Nevada, Reno.

[51] Wilson, E. C., Goodman, P. H., and Harris Jr., F. C. (2001). Implementation of a biologically realistic parallel neocortical-neural network simulator. *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, March 12-14, 2001*, pages 1–11.

[52] Wu, S., Falk, T., and Chan, W.-Y. (2009). Automatic recognition of speech emotion using long-term spectro-temporal features. In *Digital Signal Processing, 2009 16th International Conference on*, pages 1 –6.