University of Nevada, Reno

Design and Implementation of a Hierarchical Robotic System: A Platform for Artificial Intelligence Investigation

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

By

Juan C. Macera

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

December 2003

We recommend that the thesis
prepared under our supervision by

Juan C. Macera

entitled

Design and Implementation of a Hierarchical Robotic System: A Platform for Artificial Intelligence Investigation

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

MASTER OF SCIENCE

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

____
Dr. Dwight Egbert, Ph.D., Committee Member

____
Dr. Monica N. Nicolescu, Ph.D., Committee Member

____
Dr. Philip H. Goodman, Ph.D., At-Large Member

____
Dr. Marsh H. Read, Ph.D., Associate Dean, Graduate School

December 2003

Abstract

Robots need to act in the real world, but they are constrained by weight, power, and computation capability. Artificial intelligence (AI) techniques try to mimic the clever processing of living creatures, but they lack embodiment and a realistic environment. This thesis introduces a novel robotic architecture that provides slender robots with massive processing and parallel computation potential. This platform allows the investigation and development of AI models (the brain) in interaction with its body and environment. Our robotic system distributes the processing on three biologically correlated layers: the Body, Brainstem, and Cortex; on board the robot, on a local PC, and on a remote parallel supercomputer, respectively. On each layer we have implemented a series of intelligent functions with different computational complexity, including a binaural sound localization technique, a bimodal speech recognition approach using artificial neural networks, and the simulation of biologically realistic spiking neural networks for bimodal speech perception.

To my mother,
a virtuous woman.

Acknowledgments

I would like to thank Dr. Frederick Harris Jr., my advisor, for his encouragement and valuable guidance throughout the thesis implementation. I am very grateful to Dr. Phil Goodman, director of the Brain Computation Laboratory, for his support and motivation during the project development. I express my sincere gratitude to Dr. Dwight Egbert, for leading me at the beginning of the project and for being part of my thesis committee. I also thank Dr. Monica Nicolescu for serving on my thesis committee. Finally, I would like to thank my brother Jose Macera, for his support when most needed, and to my whole family, for their constant and unconditional love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter is to give an overview of the thesis. First we present the problem background, next we provide a glance of our proposal, and finally describe the thesis organization.

1.1 Problem Background

The creation of intelligence is the intersection and ultimate goal of two popular science fields: artificial intelligence (AI) and computational neuroscience. The first field tries to achieve it *via* computational and mathematical techniques, and the second one through biologically realistic neuronal models. Even though they use different approaches to mimic the functioning of the brain of living creatures, both of them need also to imitate the way living creatures interact with their environments. In real life, every brain has a body and every body is placed in an environment. We share the assertion of Chiel and Beer [5], that intelligent models will arise only when these three elements, brain-body-environment, act together.

Although computational intelligent systems combined with robotic platforms are a good way to deal with the brain-body-environment concern, many drawbacks constrain its success. The main problem of these intelligent robotic systems is the limited computational power of the robot brain, which consists of a simple CPU. In these configurations it is not possible to perform investigations that require massive and parallel computation such as evolutionary algorithms and spiking neural networks (SNN). Another problem with stand-alone robotic systems is their lack of versatility. In order to upgrade the robot brain, physical contact is required (*e.g.*, the removal and installation of hardware and/or software). Such upgrades are not possible if the robot is unreachable or is performing long and non-stoppable experiments. Another disadvantage of stand-alone robotic systems is the inability to monitor in real time the robot metrics, the environment data, and the development of AI techniques in study.

1.2 Proposal Approach

Considering that the main purpose of robotic systems is to interact intelligently and effectively with the environment and that the main purpose of AI systems is to provide intelligence to real life entities like robots, we propose a robotic model that meets these goals, successfully dealing with the robot-intelligence-environment or body-brain-environment problems of current stand-alone robots. Our proposal is a remote-brained robot with hierarchical processing distribution.

Our remote-brained approach is demonstrated with a high-precision, miniature, autonomous robot (dubbed CARL), whose processing capability was distributed on three layers: (1) on-board the robot, (2) on a local PC or laptop, and (3) on a remote computer cluster. We refer to these three layers as the Body, Brainstem, and Cortex, respectively. In this processing layout, the robot is provided with two main features: (1) a slender and dynamic body that interacts effectively with its environment and (2) the ability to process high-level AI techniques that usually require massive computation. Figure 1.1 depicts this idea.

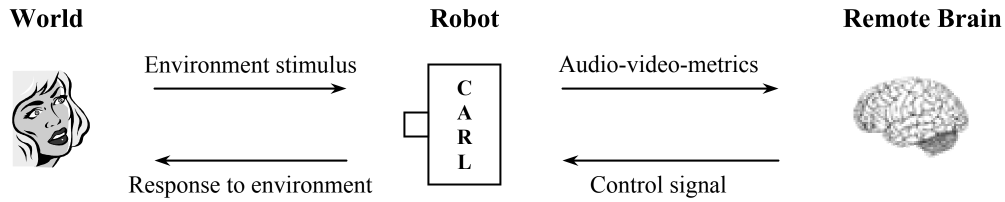

Figure 1.1: Robotic proposal depiction showing its remote processing capability and its practical interaction with the environment.

In addition to the processing distribution, we propose a robotic functionality with a biological correlation. In this approach reactive processing, which requires minimum computation, is executed on the Body; instinctive processing, which requires medium computation, is performed on Brainstem; and cognitive processing, which requires massive computation, is executed on Cortex. These features will make CARL an excellent prototype for robotics and AI experimentation. To that end, we developed a variety of intelligent functions on each layer (*e.g.*, obstacle avoidance, sound localization, speech perception, and speech recognition) by using sophisticated AI techniques such as audio and image processing, artificial neural networks, and spiking neural networks.

1.3 Thesis Structure

This thesis is organized as follows. In Chapter 2 the rationale of the three-layer system (Body-Brainstem-Cortex) is presented, followed by the implementation of the communication backbone. Chapter 3, Chapter 4, and Chapter 5 detail the architecture and functions of the Body, Brainstem, and Cortex, respectively. The following three chapters detail novel AI applications to be used by the hierarchical system. In Chapter 6 we present our methodology and implementation for binaural sound localization. Chapter 7 portrays the implementation of a novel bimodal speech recognition system using artificial neural networks. In Chapter 8 we present an approach to design and train spiking neural networks for bimodal speech perception. The evaluation of the complete system is provided in Chapter 9, and in Chapter 10 we present our conclusions and future work.

Chapter 2

The Hierarchical Robotic System

The ultimate goal of any robotic system is to interact with the natural world as natural as living creatures do by means of artificial intelligence techniques. This chapter describes the backbone of a novel robotic control system that would make this goal attainable. Section 2.1 discusses the current limitations of robotic systems, Section 2.2 describes the novel proposal, and Section 2.3 presents the implementation of the system infrastructure.

2.1 Limitations of Robotic Systems in the Real World

Two main issues constrain current robotic systems from fruitful interaction with the real world. The first issue is the lack of versatility for experimentation on different environments, and the second issue is the lack of computing power when massive processing is required. These limitations are discussed below.

2.1.1 Brain, Body, and Environment

Artificial Intelligence (AI) is a research field that tries to understand and model the intelligence of humans and living creatures. The creation of intelligence is the utmost goal of all AI techniques and algorithms, such as artificial life, machine learning, artificial neural networks, and genetic algorithms. However, any AI investigation and simulation will not resemble the objective (*i.e.*, the brain function) unless it also mimics the body's interaction with the environment. Any serious AI investigation would require successful interaction between the brain, the body, and the environment (*i.e.*, processor, robot body, and the real world) [5].

Although this triplet, brain-body-environment, offers the best test bed for AI research, its realization is constrained by many factors. From the computational perspective, intensive processing is the principal issue that restrains robotic interaction with its environment. At present, many relevant artificial intelligent tasks, such as computer vision or experiential learning, require complex techniques and algorithms. In order to meet timing demands, these algorithms must be executed using parallel programming techniques on multiprocessor systems. Therefore, standalone mobile robots will be restrained by computation capability. On the other hand, because of weight and size issues, robots with onboard multiprocessing potential will be constrained in environmental interaction.

2.1.2 Remote-brained Robots

Remote-brained robotics is a solution for this dilemma. This approach, originally proposed by Inaba *et al.* [15], consists of dividing the functions of a robotic system into a brain and a body separated physically from each other. The resulting framework would be a slender robot body that easily interacts with the environment and a powerful brain that is executed on a co-located multiprocessor system, both of them radio frequency (RF) linked.

Even though the approach of Inaba and colleagues could provide maximum processing power, its weakness is that it restricts the robot to a specific environment. The brain and body are separated and RF linked, but they must be co-located, for instance in the same building, because RF technology provides reliable data transmission over only short distances. This co-located model is illustrated in Figure 2.1.

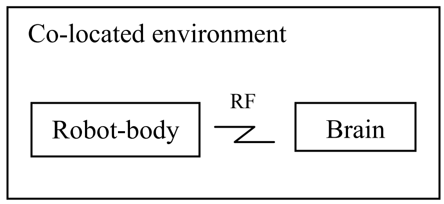

Figure 2.1: Remote-brained system with shared environment.

With recent advances in data communication technology, the location attachment problem can be alleviated. At present, the Internet, wireless networking, and high-speed data transfer techniques allow placing the robot body and brain in different environments, as depicted in Figure 2.2. Within this framework, a robotic system can take advantage of interacting with different environmental settings, such as AI laboratories or simulation fields, while preserving its computation power.

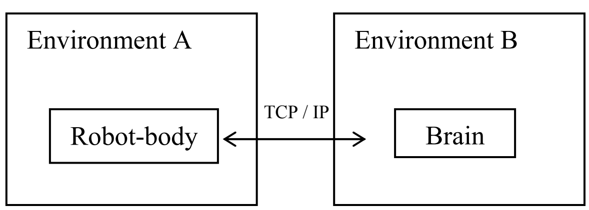

Figure 2.2: Remote-brained system with independent environments.

2.2 The Hierarchical Control System Approach

Although a remote-brained architecture is powerful for AI investigation, this thesis proposes a better approach: a three-layer hierarchical robotic control system. In this approach, processing and control are distributed on three layers. These layers will be referred to as *Body, Brainstem,* and *Cortex*, and their biological analogy will be explained later. The first layer, the Body, has small processing capability but great potential for data capture and transmission. The second layer, Brainstem, has higher computation power. It is a local PC or laptop linked to the Body *via* RF. The third layer, Cortex, is a remote computer cluster, which is connected to Brainstem over the Internet and is intended for massive parallel processing. This configuration is depicted in Figure 2.3.

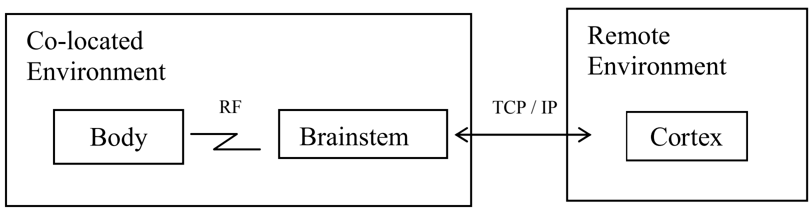

Figure 2.3: The hierarchical robotic system concept.

2.2.1 The Biologic Correlation: Reactive, Instinctive, and Cognitive Control

We chose to call the layers of the robotic system with biologically significant names because we intended to correlate our approach with the control and processing strategy of living creatures. The cortex, brainstem, and body each play a unique role when living creatures interact with their environment [27]. In biology, the cerebral *cortex* is largely responsible for higher brain functions, including sensation, voluntary muscle movement, thought, reasoning, and memory. The *brainstem* is part of the brain system located between the cerebrum and the spinal column. The brainstem relays information between the peripheral nerves and spinal cord to the upper parts of the brain. The main functions of the brainstem include alertness, breathing, and other autonomic functions. The *body* is the entire material or physical structure of a living creature that interacts with the environment by sending and receiving signals. The body captures stimuli data; the brainstem pre-processes this data; and the cortex post-processes the brainstem output to make an intelligent decision.

Our three-layer robotic system tries to separate and mimic the functions of the body, the brainstem and the cerebral cortex. The biological correlation helps to define the functionality and purpose of each layer. The robotic data processing is distributed as follows: Data processing for *reactive control* is computed by microcontrollers on the Body, data processing that involves *instinctive control* is executed on Brainstem, and data processing for *cognitive control* is performed on Cortex. This distribution of tasks is depicted in Figure 2.4. At present, Cortex is a research platform for biologically realistic neural network modeling at the Brain Computation Laboratory at the University of Nevada, Reno.

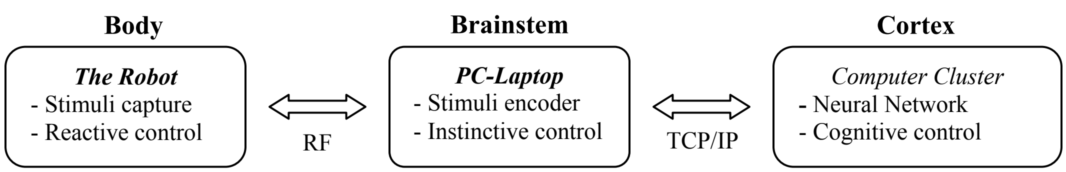

Figure 2.4: Processing and control distribution with biological correlates.

2.2.2 System Characteristics

Processing for reactive, instinctive, and cognitive control requires different computation complexity and power. For this reason we distribute our system on three computational levels of differing capacities. Task execution distributed according to its complexity is the foundation and innovation of our system. A comparison of the computational power at each level is presented in Table 2.1. Here *S(n)* is the speed up of the computer cluster when working in parallel as a function of *n*, the number of nodes

Table 2.1: Hardware comparison between the robotic system layers.

Layer	Processor	Processor Speed	Execution Rate	Memory
Robot body	Parallax BS2-IC	20 MHz	~4000 inst./s	2 K EEPROM 64 B RAM
Brainstem	Pentium 4	2.2 GHz	R	512 MB RAM
Cortex	Xeon 2.2 128 nodes	2.2 GHz per node	$R\ .\ S(n)$	2 GB RAM per node

used. Brainstem and one node of Cortex have approximately the same execution rate (R), however n nodes of Cortex working in parallel would have an execution rate of $R\ .S(n)$.

Theoretically this novel architecture will surpass remote-brained models in speed and practicability for two reasons: (1) onboard processing will be concentrated on maximizing data capture and transmission, and (2) Brainstem will allow the processing of medium-level tasks locally rather than remotely, thus eliminating Internet transmission latency. Table 2.2 shows the transmission speed between layers. As we can see in this table, row data such as audio or video is massive and requires high speed when transmitting from the Body to Brainstem. Data between Brainstem and Cortex will be preprocessed and requires slower transmission speed.

Table 2.2: Data transmission speed comparison between layers.

Link	Transmission Speed
Robot body – I/O (serial)	50 kbps
Robot body – Brainstem (RF)	Audio – Video (USB): 12 Mbps - 2.4 GHz RF Metrics (serial): 9.6 kbps - 315 MHz RF
Brainstem – Cortex (TCP/IP - DSL)	500 kbps
Between Cortex nodes (ETH)	2 Gbps

In summary, our hierarchical robotic system configuration will provide the following advantages:

- Provides maximum processing capability.
- Massive parallel processing potential.
- Limber body: light weight and small volume.
- Less power consumption onboard.

- Dynamic interaction to the environment.
- Maximize onboard data capture and communication.
- Flexibility to experiment on different environments.
- Feasibility to monitor the body locally and remotely.

2.2.3 System Functions

In order to provide the robotic system the ability to interact with the environment we developed a series of intelligent applications. These functions were distributed on the three-layer system according to their complexity, as Table 2.3 shows, and the most important ones are detailed in this thesis. First, a system to control the robot locomotion over the Internet was implemented. This served as the communication backbone of the robotic system and is described in Section 2.3. Next, we built a system for sound localization and robot navigation. This is covered in Chapter 6. Our third development was a bimodal speech recognition system using ANN and sequential programming. This is described in Chapter 7. Finally, a bimodal speech perception approach using SNN and parallel programming was tested. This is covered in Chapter 8.

Table 2.3: Distribution of functions over the three-layer system.

Application	Body	Brainstem	Cortex
Obstacle avoidance & navigation routines	X		
Binaural sound localization	X	X	
Navigation to sound target	X	X	
Robotic control over the Internet	X	X	X
Bimodal speech recognition (ANN)	X	X	X
Bimodal speech perception (SNN)	X	X	X

2.3 The Hierarchical Communication Backbone

To verify the viability of the three-layer model, we assembled the communication backbone of the hierarchical robotic system and tested it by controlling the robot locomotion over the Internet. This communication infrastructure is depicted in Figure 2.5.

Two communication links were necessary: near and distant. The near communication system was implemented using proprietary protocols *via* RF transceivers, linking the robot body and Brainstem. The distance communication system was implemented using TCP/IP protocols over the Internet, linking Brainstem and Cortex.

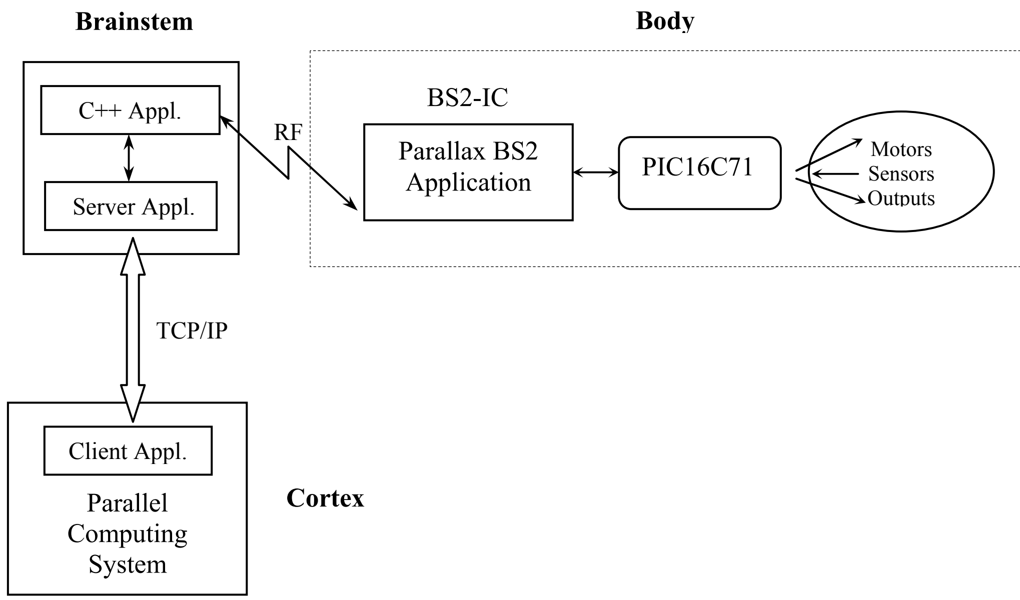

Figure 2.5: Communication architecture of the three-layer system.

2.3.1 Body – Brainstem Link

To provide a wireless link, the robot was integrated with a module for radio frequency communication: Parallax RF-433. This module consists of two transceivers: one is linked to the robot main processor (BS2-IC), and the other is linked to the PC (*i.e.*, Brainstem). Figure 2.6 depicts our RF link architecture.

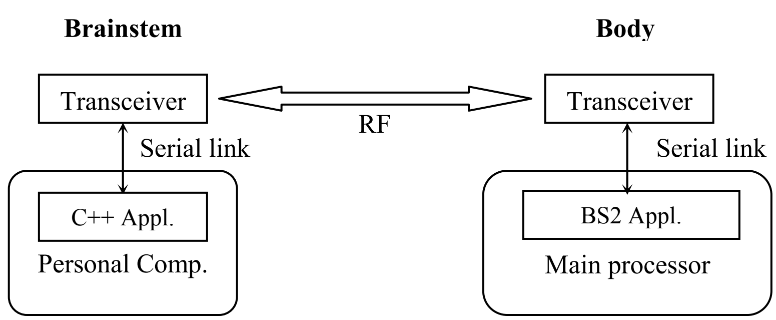

Figure 2.6: Communication architecture between the Body and Brainstem.

Both transceivers communicate *via* a serial port. This hardware configuration provides a bi-directional communication up to 250 feet. Each transceiver sends and receives serial data at 9600 baud (*N*, 8, 1) with logic levels between 0 and +5 volts [28]. Sensory metrics are sent from the robot to Brainstem, and control commands are sent from Brainstem to the robot.

RF application on robot body

On board the robot, an RF program was implemented using proprietary language: Parallax Basic Stamp 2 (BS2). BS2 provides built-in commands for the serial communication between the transceiver and the main microprocessor (see Figure 2.6). SERIN and SEROUT are the BS2 commands for serial transmission. The communication protocol at the application level consists of the following commands:

T: Transmit data packet
R: Request data packet
E: Request (and reset) error count
I: Initialize PIC
V: Request PIC firmware version

The serial command to transmit data from the robot to Brainstem has the following format: First the port of communication to the transceiver is included (13\12, 32768), followed by the command of transmission: “T”. Afterwards a number representing the data size to transmit is included (maximum data length is 10 bytes), followed by the data bytes to transmit. For example, to send two sensor variables of one-byte size, the serial command would be:

```
SEROUT 13\12, 32768, ["T", 2, SENSOR1_TO_BRAINSTEM, SENSOR2_TO_BRAINSTEM]
```

The received data packet is requested with the R command. The data format is as follows: the first byte is a number representing the number of data bytes (byte_count);

next are the data bytes (CMD_FROM_BRAINSTEM), followed by the error count (errors). If a data packet is requested with the R command and there is no data in the receive buffer, a zero will be returned as the byte count, and the error count will follow as usual. The following commands are used to receive a data packet:

```
rx:
SEROUT 13\12, 32768, ["R"]
SERIN 13\12, 32768, 100, rx, [byte_count, (CMD_FROM_BRAINSTEM), errors]
```

The 100 and rx are for the SERIN timeout. If the SERIN command has to wait more than 100 ms for a data byte, the program flow redirects to the 'rx:' tag.

In summary, our onboard RF application is a program that repeatedly reads instructions from Brainstem and writes sensor metrics to Brainstem. Capturing sensor information and executing locomotion commands are also functions of this program.

RF application on Brainstem

On Brainstem, an RF program was implemented in C++. We used communication protocols at the transmission level, writing and reading data directly to and from the serial port of the local PC where the transceiver was attached (see Figure 2.6). This communication protocol is described below.

The Parallax RF-233 transceivers communicate between themselves in a packet-type format. These transceivers are designed for a master-slave relationship in a bi-directional fashion and can send, receive, verify, and re-send data if necessary. All data transmitted between transceivers are formatted into a variable length data packet as depicted in Figure 2.7.

Packet #	**Data Count**	**Data Value 1**	----	**Data Value *n***	**Checksum 1**	**Checksum 2**
Byte 1		Byte 2		Byte *n*+1	Byte *n*+2	Byte *n*+3

Figure 2.7: Data packet format for transceiver's communication.

In this communication protocol, Byte 1 consists of two pieces of data, the packet number and the data count. The packet number is a value from 1 to 15 (0 is an illegal value). The packet number is the ID of the packet relative to the previously transmitted packet and is used to verify that no duplicate packets are mistaken for new data. The data count is a value from 0 to 15 representing the number of data values in this packet. Packets can contain from 1 to 16 bytes of data values (at least 1 data value is required). Thus, the number of data values is actually the data count + 1. Bytes 2 through n+1 are the actual data values where n = data count + 1. Bytes n+2 and n+3 marks the end of the packet and consist of checksum values. Two different methods can be used: XOR algorithm (90% efficient) or Cyclic-Redundancy-Check algorithm (99% efficient).

In summary, this RF application is a program that interacts with a user to request metrics and send locomotion commands to and from the robot. If the user is in a remote location, an extra Internet application is required. This is described below.

2.3.2 Brainstem – Cortex link

In order to control the robot over the Internet, a client-server application was implemented in C using sockets (winsock32 library). The server application will run on the local PC (Brainstem) and the client will run on a head node at the computer cluster (Cortex).

The server application will interact with both the RF application on Brainstem and the client application on Cortex (see Figure 2.5). In summary, this program waits for a remote connection over the Internet, receives commands and requests from the client, and forwards them to the RF application on the same PC. Commands would be for locomotion control and requests for sensor status retrieval. The server also will send acknowledged information to the client. This process is illustrated in Figure 2.8.

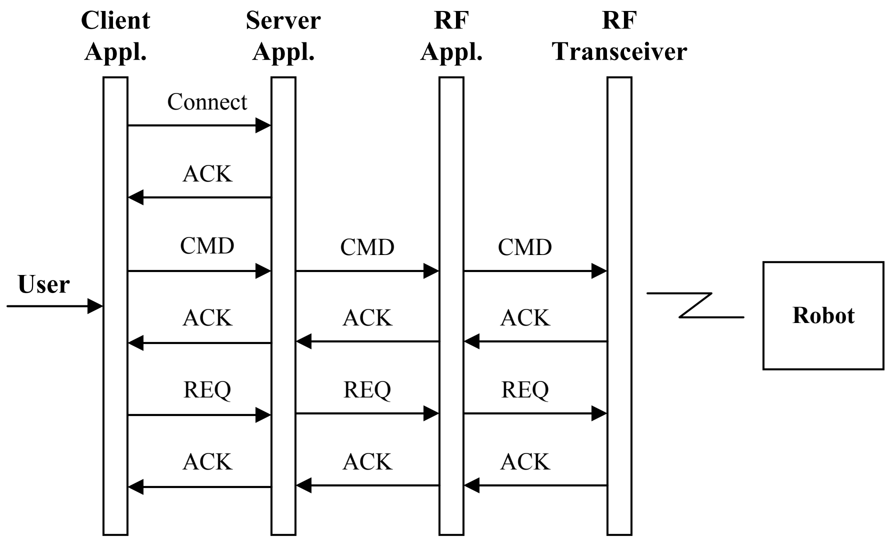

Figure 2.8: Communication dynamics between Brainstem and Cortex applications.

The functions of the client program are to initiate connection with the server, interact with the user through the keyboard, send user requests to the server, and read feedbacks from the server. To synchronize the client-server application, the server must run first, listening on a specific port number:

ServerAppl <Port_Number>

Then the client application must be executed specifying the IP address and the port number of the server application:

ClientAppl <Server_IP_Address> <Port_Number>

We tested the whole system by controlling the robot from Cortex using a PC keyboard. Five locomotion commands were used: forward, backward, left, right, and stop. We were also able to retrieve sensory information. The time responses of the experiments, from keystroke to robot locomotion response, were satisfactory (registered between 0.001 to 0.02 seconds approximately).

2.4 Chapter Summary

In this chapter we presented our hierarchical robotic approach. We discussed a technique that allows robots to be used for massive computation, while being environment independent. This approach, which mirrors the strategy of living creatures, distributes the processing work according to its complexity on three layers: reactive processing, on the Body; instinctive processing, on Brainstem; and cognitive processing, on Cortex. We will discuss the architecture and functions of the Body, Brainstem and Cortex in Chapter 3, 4, and 5 respectively.

Chapter 3

The Body: Architecture and Functions

The robot body is the first layer of the hierarchical robotic system and is the one that interacts with the environment. To put in practice our control model, we acquired a weightless, factory-made mobile robot, which we dubbed CARL, and incorporated with an auditory-vision system and wireless link. This chapter describes CARL's electronics in Section 3.1 and its onboard functions in Section 3.2.

3.1 Hardware Architecture

CARL is a high-precision, miniature, programmable, autonomous robot. It is a wheeled robot based on a dual-processor architecture inspired by the biology of the nervous system [7]. The secondary processor, or I/O processor, is a PIC16C71 – RISC, factory programmed to control speed and heading. This processor communicates I/O and position information to the primary processor on board, a Parallax BS2-IC microcontroller, which allows sensor capture and navigation. CARL's design offers high navigation accuracy, responsiveness, and ease of command.

CARL features four true 8-bit A/D conversion ports, a speaker, four CdS light sensors, and a thermistor to perceive temperature variations. Four ultra-bright LEDs permit object proximity detection and status indication. Highly responsive bumper sensors detect objects around the entire front and sides of the robot. CARL has an efficient drive system that operates in near silence. Programmable PWM-controlled motors can deliver speeds from a crawl to more than 2 ft/s. IR encoders monitor the motion of each wheel. The robot is approximately 7 inches in diameter and 4.5 inches tall. The assemblage of CARL's drive and sensory system is presented in Figure 3.1, and its hardware schematics are given in Appendix 1.

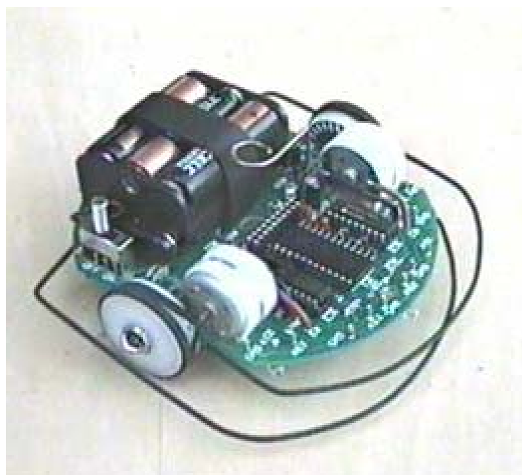

Figure 3.1: CARL robot before assembling audio-video system and RF transceiver.

CARL's drive system is elegant in its simplicity. Flat belts drive the wheels directly from each motor shaft, eliminating gears, pulleys, and servo hardware and minimizing expense. This unique direct belt drive is extremely reliable, and even during complex movements CARL operates in near silence. In addition, CARL's reversible DC motors are a low RPM, high-torque design delivering tremendous response. They are accurately positioned on the robot's frame by areas routed directly into the printed circuit board and positively locked in place with nylon ties. The motors are controlled by a feedback circuit served by infrared photo interrupters reading indexes placed directly on the inside of each wheel. The photo interrupters are carefully located under the robot where they are best protected from ambient light.

To provide auditory and vision capability, CARL was integrated with a color video camera and stereophonic microphones [4]. Audio and video are captured through separate channels and fed to our audio/video sender device. The sender device is an X10-VT32A, which is linked to Brainstem *via* radio frequency as depicted in Figure 3.2. This wireless technique allows us to capture audio-video data within a 100 meters distance [1]. The methodology for capturing audio and video from the PC *via* software is explained in Section 6.2.1 and Section 7.2 respectively.

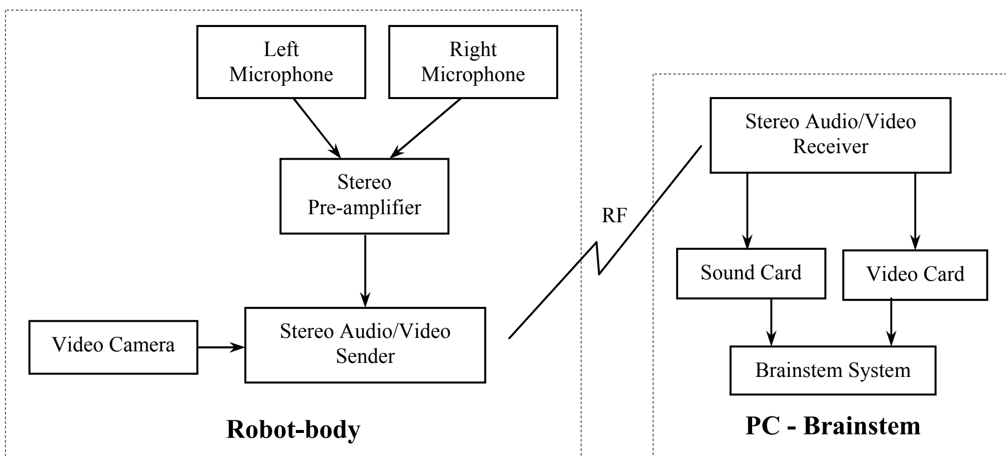

Figure 3.2: Wireless audio-video hardware configuration.

To provide the wireless link, CARL was integrated with a radio frequency transceiver device, a Parallax RF-433 [28]. This device allows bi-directional data transmission and is used to send sensory information to and receive locomotion commands from Brainstem. Brainstem contains another transceiver that can be separated up to 100 meters for reliable data interchange. Details of the transceiver communication protocol were discussed in Section 2.3.

3.2 Onboard Functions

CARL has two principal goals: to interact efficiently with the environment and to perform processing for reactive control. Interaction with the environment is divided in two parts. The first part consists of capturing auditory and visual information and sending it *via* wireless link to Brainstem. The audio-video system is totally independent of the robot's main processor. This setup allows real-time streaming to the local computer. The second part requires the use of the main processor and consists of capturing sensory metrics and commanding the locomotion of CARL.

The role of the main processor when interacting with the environment is summarized schematically in Figure 3.3. Locomotion commands are received through the transceiver and are delivered to the I/O processor for execution after simple evaluation. Metric requests from Brainstem are also received through the transceiver and then forwarded to the I/O processor. Subsequently, metrics are transmitted in opposite directions in succession.

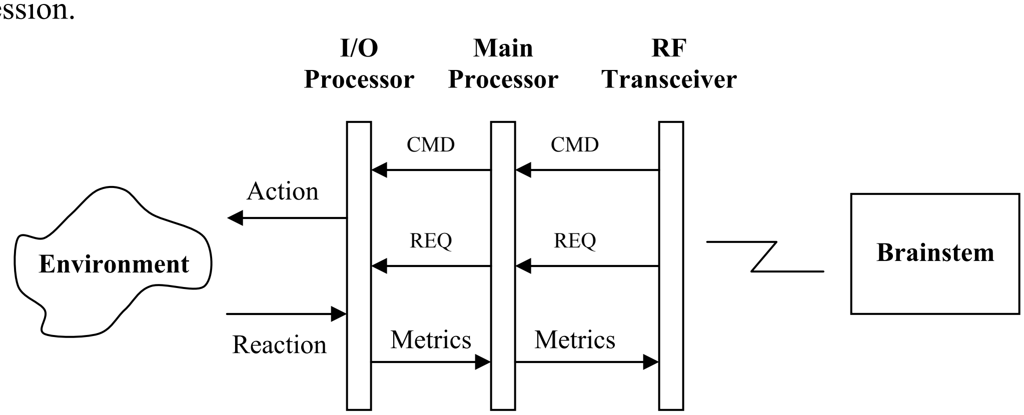

Figure 3.3: Role of CARL's processors when interacting with the environment and Brainstem.

The main processor uses the proprietary language Parallax Basic Stamp 2 (BS2) [2] to communicate to both the RF transceiver device and the I/O processor *via* the serial port. The protocols and examples of communication between the main processor and the RF transceiver were presented in Section 2.3.1. Here we describe the protocol of communication between the main processor and the I/O processor.

The main processor communicates with the I/O processor in serial data format at 50 kbps. Port 15 is used for communication, and port 14 is used for flow control (see the hardware schematic in Appendix 1). The I/O processor requires about 80 ms to startup, so a *pause* command should be included at the beginning of each program. SEROUT is the BS2 command to send locomotion directives to the I/O processor. For example, to make CARL go forward 100 encoder counts, the program should read:

```
SEROUT 15\14, 0, ["F", 100]
```

To receive serial data from the I/O processor, a SEROUT request must be sent, followed immediately by a SERIN command. For example, to request and receive the distance traveled in encoder counts of the left wheel, the program should read as follows:

```
DISTANCE VAR WORD
DL VAR DISTANCE.LOWBYTE
DH VAR DISTANCE.HIGHBYTE
SEROUT 15\14, 0, ["["]
SERIN 15\14, 0, [DL, DH]
```

In this code, ‘[’ is a reserved character in BS2 to request the left wheel distance traveled, and DL and DH are 8-bit variables. More information regarding this communication protocol is available in [7]. The effective interaction between the main processor and the I/O processor provides CARL with the following basic onboard functions:

Motion - The user program can specify a direction and percentage of power for each of the left and right gear motors, giving the user complete direct control. Complex motor commands can also be sent such as ‘F 2000’. At this command, CARL will start moving forward at the current cruise speed (set with the ‘S’ command) while constantly monitoring its wheel photo interrupters to keep the robot on the correct heading and also to slow down and stop once the distance count of 2000 has been reached. All motor functions, including an automatic ramp-up of the motor voltage if a stalled wheel is detected, are performed entirely in the background. During this time, the user program can continue to communicate with the I/O processor, requesting A/D port conversions on the external re-configurable sensors, distance traveled, or current heading values. The user program can even give the I/O processor a new heading in the middle of a running mode.

Navigation - The current robot heading and distances traveled by each of the wheels are constantly monitored by the I/O processor. These values can be individually read and reset by the user program. The main processor’s built-in trigonometric functions give user programs the ability to calculate complex movements and position reckoning. The light sensors can be used to line up on a beacon and synchronize the I/O processor's internal heading with the environment.

Sensor input - Four true 8-bit, analog-to-digital conversion ports are provided along the edges of CARL. These ports have been carefully positioned and duplicated to allow various and useful orientations of the flexible sensors. The I/O processor performs fast conversions on any or all of the ports and returns the value to the user program when requested. Readings on all four A/D ports can be obtained in a fraction of the time required for a single RCTIME conversion by the main processor. These sensor readings are returned as values between 0 and 255, representing the voltage level at the port during the conversion. When CdS cells are plugged into any or all of the ports, the value will represent the light level striking the face of the sensor at that time. These readings are useful for many purposes such as motion detection, line following, light-seeking, and non-contact obstacle avoidance. Any combination of these resistive sensors may be plugged into the four A/D ports.

Reactive control - Finally, CARL is equipped with a set of onboard programs intended for the reactive control of the robot. These are functions designed to interact quickly with the environment when low-level processing is required. These functions are small BS2 programs that make effective use of the I/O components of the robot body. Examples of these applications are: obstacle avoidance, light tracking or avoidance, navigation routines, and heat tracking or avoidance.

3.3 Chapter Summary

This chapter demonstrated the feasibility of transforming a standard robot into a wireless controlled robot, which becomes the Body of our three-layer model. We also described CARL's rich functionality, which make realizable its primary onboard goal of processing for reactive control. The reliable bi-directional capture and transmission of data (audio, video, commands, requests, and metrics) makes possible a successful interaction with both the environment and Brainstem. Brainstem is the second layer of the system and is presented in detail in Chapter 4.

Chapter 4

Brainstem: Architecture and Functions

Brainstem is the second layer in the hierarchical robotic system, and we use the term to refer to both the hardware and software that is co-located and RF linked with CARL. Brainstem provides the novelty to the robotic control approach and is the cornerstone of the system. This chapter presents the hardware components of Brainstem and the functions supported by the system.

4.1 Hardware Architecture

The major hardware components of Brainstem comprise a personal computer, a RF transceiver device, and a wireless audio-video receiver. The personal computer used for our experiments has an Intel Pentium 4 processor with a speed of 2.8-GHz, 533-MHz of system bus, 512 MB of RAM, and a 100 GB hard drive. The radio frequency transceiver is a Parallax RF-433 connected over a serial port to the PC, whose protocol of communication was discussed in Section 2.3.1. The audio-video receiver on Brainstem is an X10 VR30A. This receiver has two audio output channels, which are connected to the PC through the *line-in* on the stereo soundcard. Video output from the receiver feeds an analog-to-digital converter, an ATI TV-Wonder device [33], which delivers digital video data to the PC *via* USB. The audio-video system on Brainstem is depicted in Figure 3.2. In addition to these components, the PC was equipped with a 10BaseTX/100BaseTX Ethernet for network connection on campus, and with a DSL modem for Internet connection off campus.

4.2 Brainstem Functions

Brainstem is the novelty of the hierarchical system, and its rich functionality makes this system more efficient when compared with common remote-brained robots. This section details those functions.

Data communication manager

The system was implemented such that all information transmitted between CARL and Cortex must pass through Brainstem. As Figure 4.1 depicts, Brainstem is the data communication manager of the hierarchical system. As such, Brainstem will be responsible for the following tasks: (1) controlling CARL's navigation by sending locomotion commands, which would be Cortex or Brainstem outputs, (2) receiving audio-video data from CARL, used for auditory and visual processing, (3) accessing metrics from CARL and Cortex, used for monitoring purposes and for instinctive/cognitive processing assessment, and (4) injecting preprocessed data to Cortex, such as feature vectors for a neural network simulation.

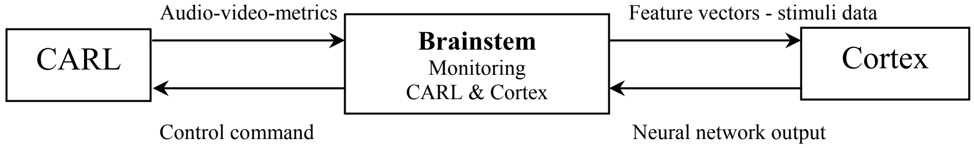

Figure 4.1: Brainstem managing the data communication of the system.

Monitoring CARL and Cortex

An important feature of Brainstem is that it provides the feasibility to monitor both CARL and Cortex. This is a useful and unique feature of the system. Monitoring CARL's metrics allows humans to supervise the performance of CARL in real time and to assist the robot if necessary. Monitoring Cortex's metrics consist of screening the progress of remote massive computation, such as spiking neural network simulations, and would be especially valuable when experimenting with new AI models or when old AI models encounter new challenges. The supervision of the complete system at work and in real-time will be essential when experimenting with new technologies and novel AI methods.

Instinctive processing

In addition to serving as the communication controller between CARL and Cortex, Brainstem is also responsible for processing computational tasks of medium level complexity. We call this processing for instinctive control. At present, Brainstem is equipped with two intelligent processes: sound localization and navigation to target. The sound localization system is part of the auditory system in development, which allows the robot to determine the direction of any sound originated in the environment. Once the direction of sound is determined, the navigation-to-target system steers the mobile robot to the sound source. Periodic sound measurements and reactive movements are the basis of this technique. Chapter 6 details both techniques.

Building data for cognitive processing

At present, Cortex can simulate two neural network models –one using conventional artificial neurons and the other using biologically realistic spiking neurons. The first model was implemented for bimodal speech recognition and is detailed in Chapter 7. The second model is an example of bimodal speech perception and is described in Chapter 8. Both Cortex's simulators require pre-processed inputs. In general, researching on new neural network models requires training and simulation, and both of these steps use pre-processed data. However, when running the network model on robotic systems, pre-processing the data input must be synchronized with other systems and in real time, which is not required when training the model. This requirement is provided by Brainstem using two applications –a feature vector generator program for artificial neural network simulation and a stimuli generator program for spiking neural network simulation, as depicted in Figure 4.2.

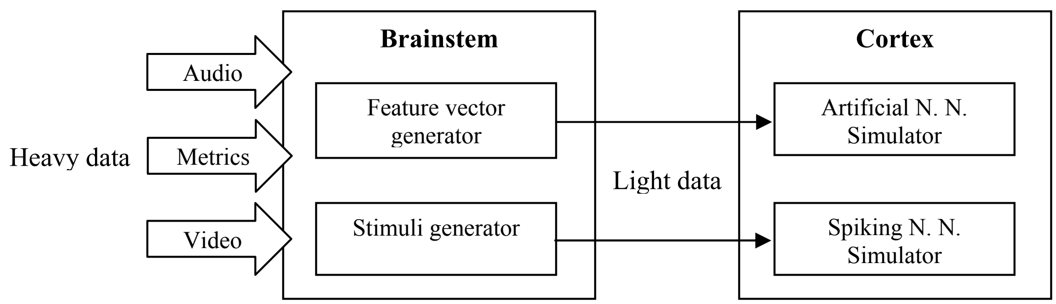

Figure 4.2: Brainstem transforms raw data for high-level processing in Cortex.

Transforming heavy data into light data before sending it to Cortex will speed up the data transmission and improve the system performance, which otherwise would be the bottleneck of the system.

4.3 Chapter Summary

The rich functionality that Brainstem provides makes this system an invaluable aid to the hierarchical system. In summary, Brainstem provides data communication, monitoring aid, local processing, and data transformation. None of these functions require parallel computing, and therefore Brainstem can be co-located with the robot body in a portable PC or laptop. Functions that require massive processing will be executed on Cortex as discussed in Chapter 5.

Chapter 5

Cortex: Architecture and Functions

Cortex is a powerful cluster of computers linked to the Brainstem system *via* TCP/IP protocol. The goal of Cortex is to deal with CARL's high-level decisions and control tasks that require intensive computation, such as bimodal (audio-video) or multimodal (*i.e.*, multisensory) processing. This chapter describes the hardware configuration and functions of Cortex.

5.1 Hardware Architecture

Cortex is a supercomputer system constructed in the summer 2001 and managed by researchers of the Brain Computation Laboratory at the University of Nevada, Reno. Cortex is a parallel computer system whose main purpose is to be the simulation platform for biologically realistic spiking neural networks. This is described in the next section. Currently, the computer cluster has a total of 128 processors with 256GB of RAM and more than a terabyte of disk. Cortex initially consisted of 30 dual Pentium III 1-GHz processor nodes with 4GB of RAM per node and was upgraded in the summer of 2002 with 34 dual Xeon 2.2 GHz processor nodes with 4GB of RAM per node.

Myrinet 2000 [26] handles the intensity of communication that occurs in the fine-grain parallel model. This high-bandwidth/low-latency interconnection network gives a much higher level of connectivity than would have been otherwise possible and is the key factor to running large-scale NCS models. In addition to the Myrinet 2000 interconnection network, the cluster is also connected with an HP 4108 Ethernet switch. The original 30 nodes have 100TX ports, and the new 34 nodes have 1000TX ports [13].

The management of Cortex's cluster is performed by Rocks [30], a Linux distribution based on Red Hat Linux that has been specifically designed and created for the installation, administration, and use of Linux clusters. Rocks was developed by the Cluster Development Group at the San Diego Supercomputing Center.

5.2 Cortex Functions

Spiking neural network simulator

Here we describe a spiking neural network simulator, dubbed NCS –Neocortical Simulator, implemented in the Brain Computation Laboratory at the University of Nevada, Reno. NCS is part of a research project that focuses on understanding the principles of mammalian neocortical processing using large-scale computer simulations. The primary objective of this project is to create the first large-scale, synaptically realistic cortical computational model under the hypothesis that the brain encodes and decodes information through timing of action potential pulses rather than through average spiking rates. This approach, which includes channels and biological accuracy on column connectivity, is discussed in more detail in [36, 37, 38].

NCS was designed using object-oriented design principles and coded in C++. In this design, a “brain" (an executing instance of NCS) consists of objects, such as cells, compartments, channels, and the like, which model the corresponding cortical entities. The cells, in turn, communicate *via* messages passed through synapse objects. Input parameters allow the user to create many variations of the basic objects in order to model measured or hypothesized biological properties.

The operation and reporting of NCS is based on parameters specified in a text input file. In this way, a user can rapidly model multiple brain regions merely by changing input parameters. The user specifies the design using biological entities: a brain consists of one or more columns, each column contains one or more layers, each layer contains cells of specified types, and so on. By changing only the input file, this simulator can model large numbers of cells and various connection strengths, which affect the number of synapse objects and the amount of communication. The design also allows the modeling of very large numbers of channels and external stimuli.

Artificial neural networks on parallel computers

Although Cortex was created to be a platform for NCS, it is also an excellent test bed for experimenting with artificial neural networks. At present, Cortex is equipped with a sequential ANN simulator supported by Matlab [24], however this can be extended to a parallel fashion. This approach would be the ultimate solution to ANNs that require massive computation with time related problems. Although this function is not implemented in Cortex, we mention it in this chapter for completeness and because it is a consideration for future work. A combination of the neural network toolbox and the MPI toolbox both under Matlab would be the practical solution for the parallel computing approach.

Bimodal speech perception using SNN

This section describes a particular high-level decision process performed on Cortex by NCS. For this work, NCS was trained in advance to recognize a number of phrases, some of which were classified as threatening and some as neutral. Bimodal perception was chosen in order to support and link a related research project exploring the potential for increased phrase recognition accuracy when combining lip position information with sound information into a bimodal neocortical speech recognition strategy. Details of this learning algorithm are available in a technical report from our lab [20], and a summary is described in Chapter 8.

Bimodal speech recognition using ANN

A speech recognition system that uses both audio and video clues was implemented using an ANN. This system is not a parallel programming application neither requires massive computation. It is a sequential program implemented using ANN toolbox under Matlab, which we run on Cortex in order to test the performance of the hierarchical system; and also to parallel results with the SNN counterpart. The methodology of the speech recognition system is based on image processing the speech spectrogram, and lips reading the video information. This is detailed in Chapter 7.

5.3 Chapter Summary

It is believed that NCS research could lead to a major revolution in our understanding of the cortical dynamics of perception and learning [38]. Currently NCS supports simulations with more than 6 million synapses per node. The hardware configuration of Cortex is crucial, considering that connectivity between cells drives everything from memory and CPU usage to latency in internodal communication. This chapter concludes the study of the three-layer robotic system. The next three chapters detail the AI methods used in our experimentations.

Chapter 6

Binaural Sound Localization

Localizing the direction of a sound is a basic feature of many living creatures. It is an ability that allows creatures to navigate, find prey, or avoid danger. The importance of this biological feature motivated us to implement a sound localization system for our hierarchical robotic platform. The sound is captured by CARL, and the processing is executed on Brainstem. This chapter presents the principles, implementation, and experimentation of this functionality.

6.1 Principles of Sound Localization

The way humans localize a sound source is primarily by using cues derived from differences between the inputs received by the two ears. Two of such cues are known as interaural time difference (ITD) and interaural intensity difference (IID) [9]. In the human brain, the IID function is performed by a brainstem structure called the lateral superior olive, and the ITD function is performed by a brainstem structure called the medial superior olive [27]. In this paper we implement correlates of these functions in conventional software (*i.e.*, in an algorithmic programming language) on our Brainstem computer.

John Strutt, in his work on what he called the *Duplex Theory* [8], presents two consistent approaches that use the ITD and IID cues for finding the azimuth angle (θ) of the origin of a sound. The first method uses the IID cue. It is based on the *head-shadow effect* [8] and the significant signal level difference between the two ears. The down side of this method is that it is highly frequency dependent. At low frequencies, where the wavelength of the sound is long relative to the head diameter, there is hardly any difference in sound pressure at the two ears. However, at high frequencies, where the wavelength is short, the difference would be about 20 dB or greater. With this method, the azimuth angle can be estimated from the ratio of right-ear to left-ear spectral amplitudes, weighted by the spectral energy.

The second method uses the ITD cue. It is based on the speed of sound and the time difference that the sound takes to arrive at each ear (or microphone), as Figure 6.1 depicts.

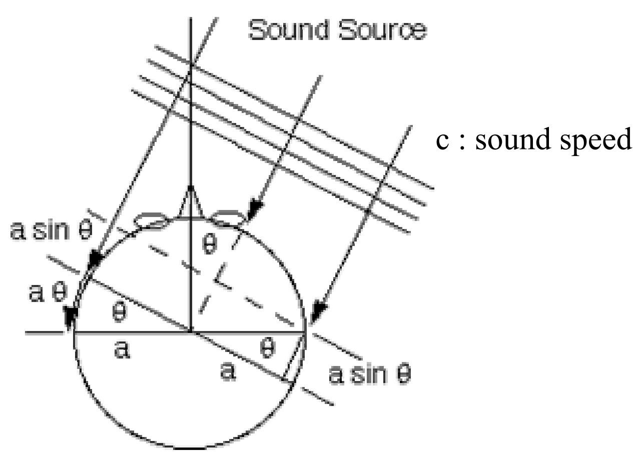

Figure 6.1: Sound direction localization by ITD.

From Figure 6.1 we can deduce an equation to find the azimuth angle (θ) if the ITD were known.

$$\text{ITD} = (a / c) \cdot (\theta + \sin\theta), \quad -90° \le \theta \le +90° \qquad \text{Equation 1}$$

According to [3], one way to compute ITD is to perform a cross correlation between the signals received at each ear. The cross-correlation function is defined as follows:

$$C(\tau) = \int Sc(t) \,.\, Si(t-\tau)\,dt \qquad \text{Equation 2}$$

where Sc and Si are the signals received at the contra-lateral and ipsi-lateral ears, respectively. Ideally, the lag-location of the peak of the cross-correlation function would correspond to the ITD.

6.2 Sound Localization Implementation

Based on this theory, we have implemented a sound localization system for the robotic infrastructure. This consists of two procedures: stereo data acquisition and binaural processing.

6.2.1 Stereo data acquisition

This procedure consists of capturing stereo audio information in digital format from the sound card of the Brainstem computer. Capturing reliable audio information requires the following steps: system setup, calibration, and trails. System setup refers to installing the adequate software and hardware and attaching the microphones. In our case, the stereo microphones are on CARL, and the software and hardware are on Brainstem. The calibration step consists of adjusting the microphones so that noise is avoided, and both audio channels capture audio data at the same intensity level, under similar circumstances. This adjustment can be made *via* software or hardware. The purpose of trails is to ensure that audio data are not saturated, to filter any residual noise, and to verify that audio data is ready for intelligent processing. Trails are performed *via* software.

Aided by the Data Acquisition Toolbox of Matlab [23], we were able to proceed successfully with all these steps of sound capture. The Matlab key code for the acquisition of stereo audio data from a sound card is as follows:

1. Create a device object - Create the analog input object ‘ai’ for a sound card.

```
ai = analoginput( 'winsound' );
```

2. Add channels - Add left and right channels to ‘ai’ (stereo).

```
chan = addchannel( ai, 1:2 );
```

3. Set property values - Each trigger will last 0.02 sec. with a rate of 16000 samples/s.

```
duration = 0.02;
set( ai,'SampleRate',16000 );
ActualRate = get( ai,'SampleRate' );
set( ai,'SamplesPerTrigger',duration*ActualRate );
```

4. Define trigger condition - Sound is captured when the loudness is above 0.4 Volt.

```
set( ai, 'TriggerConditionValue', 0.4 );
set( ai, 'TriggerChannel', ai.channel(1) );
```

5. Define trigger delay - Sound data starts 0.25 sec. before the trigger condition event.

```
set( ai, 'TriggerDelay', -0.25 );
set( ai, 'TriggerDelayUnits', 'seconds' );
set( ai, 'TimeOut', 3600 );
```

6. Acquire data - Start the audio stereo acquisition. The ‘ai’ stops automatically.

```
start( ai );
[data ,time] = getdata( ai );
leftchannel =  data( :,1 );
rightchannel = data( :,2 );
```

7. Clean up - When ‘ai’ is no longer needed, remove it from memory.

```
delete( ai );
clear ( ai );
```

6.2.2 Binaural processing

We developed applications that compare both left and right signals and determine the direction of sound. We experimented with both IID and ITD methods independently.

Two equivalents approaches were used for the IID method. The first approach works in the time domain and consists of comparing the interaural energy of left and right channels. This energy comparison is depicted in Figure 6.2. The top row of this figure shows the left and right waveform of the original data. From this data is not possible to determine which microphone is closer to the sound source. The bottom row shows the left and right energy plots calculated from the original data (V^2), where the total energy received at each microphone can be estimated by calculating the area under each energy plot. This output tells us that the right microphone is closest to the sound source.

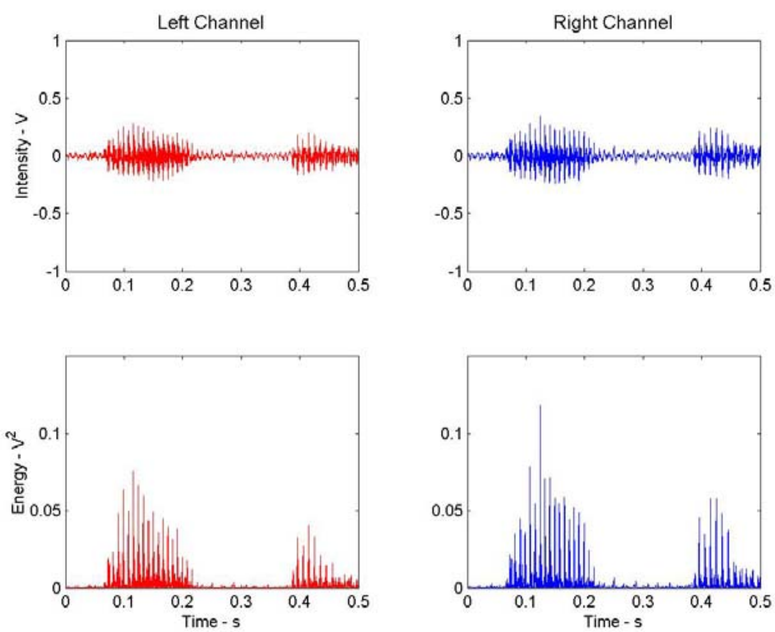

Figure 6.2: Interaural energy comparison in the time domain.

The second IID approach works in the frequency domain, and consists of comparing the interaural energy received by the left and right microphones. This is illustrated in Figure 6.3 for the same audio data. The top row of this figure shows the left and right spectrogram of the original data. In the spectrograms we can visualize the energy level at different frequencies during the period of the sound. The energy of each channel can be calculated from the spectrogram by adding all the energy components per frequency. This energy plot is presented in the bottom row of Figure 6.3. Another faster method would be calculating the fast Fourier transform of the signal and multiplying it by its complex conjugate [25]. The total energy received at each microphone can be estimated by calculating the area under each energy plot. Whichever microphone has higher total energy would be closest to the sound source, in this case the right microphone.

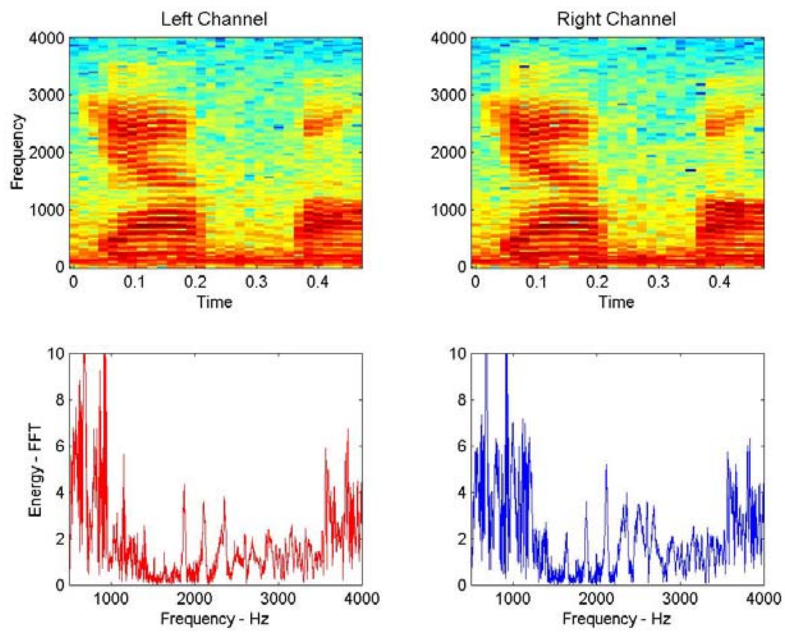

Figure 6.3: Interaural energy comparison in the frequency domain.

Although IID methods are simple and straightforward for calculating the direction of sound, they are not reliable under real-life conditions. This drawback is discussed in the next section. The other method in consideration is the ITD, which after experimentation was found to be more resilient. This method requires cross correlation between the left and right signals. We implemented the cross correlation application using Equation 1 and the algorithm provided by [3] in C. This was tested using the signals captured by CARL and digitized by the acquisition system on Brainstem. Our technique allows comparing two similar signals recorded at different phases. As a result, the phase or time difference between both signals can be determined. Figure 6.4 shows the cross correlation output plotted in the time domain. When both signals are closely correlated, the plot looks symmetric, and a 'peak' close to $x = 0$ should be present. The time difference between both signals will be ABS(x_{peak}). The sign of x_{peak} will indicate if the signal is coming from the left or right side, and the value of the time difference will help to calculate the azimuth using Equation 2.

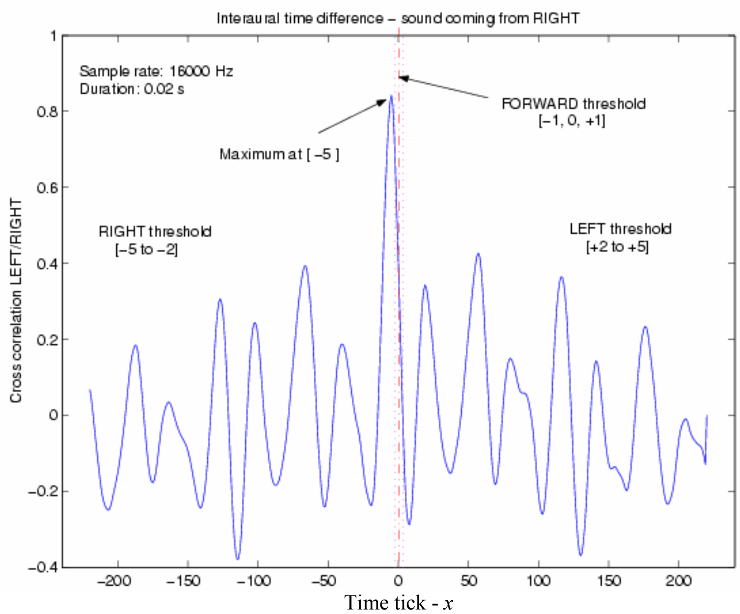

Figure 6.4: Sound localization methodology by cross correlation of binaural information.

6.3 Experimentation

This section describes the experimentation and evaluation of the sound localization applications developed for the robotic system. We consider this processing, which is executed on Brainstem, an *instinctive feature* of the hierarchical system. Through this feature CARL will be able to move toward the origin of sound. Considering this objective, we set up our experimenting conditions as follows: The experiments took place in our laboratory, a common office subject to background noise and echo. The sound source was a desktop PC speaker located 80 cm from CARL's stereo microphones. The type of sound used was speech, which was previously recorded and played by a PC in loop. The sound output was calibrated between -0.5 and + 0.5 volt of intensity.

Under these conditions, 300 sound localizations experiments were executed with each technique. Each experiment consisted of capturing 0.5 second of stream data at 8000 Hz and determining the general sound origin: the LEFT, CENTER or RIGHT side of CARL. We executed 100 trails for each direction. Figure 6.5 and Figure 6.6 present our results for ITD by cross correlation and IID by energy comparison in the frequency domain, respectively. IID in the time domain showed similar results to the frequency domain, and these results are not presented in this thesis.

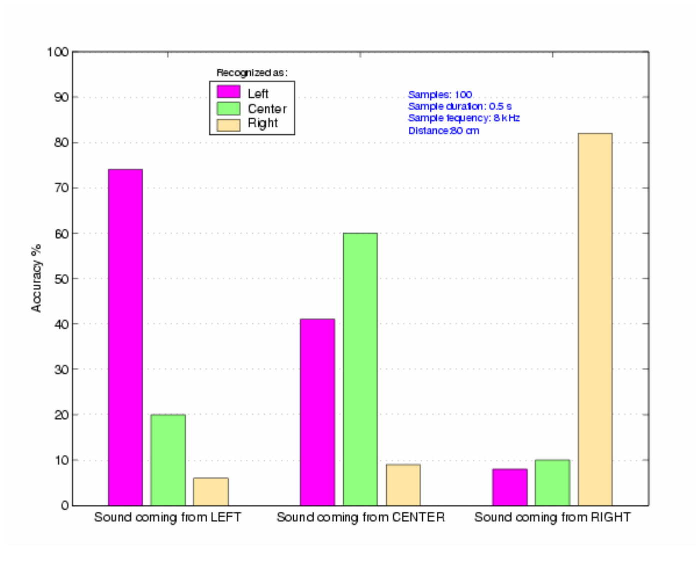

Figure 6.5: Localization accuracy using IID technique.

As we can see in these results, the ITD method was more efficient than the IID method. The IID technique was sensitive to frequency variations, microphone calibration, background noise, and echo. However, the ITD method demonstrated high resiliency to these conditions and reported 91% of efficiency on average.

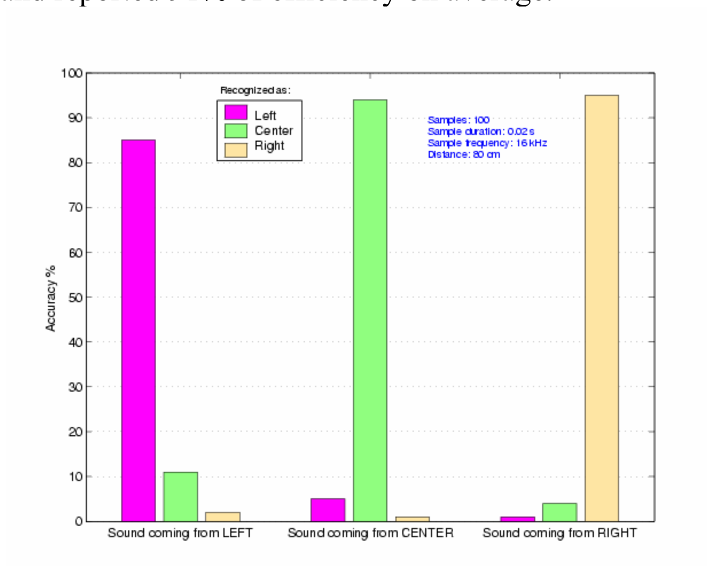

Figure 6.6: Localization accuracy using ITD technique.

6.4 Chapter Summary

We found and implemented a robust sound localization technique, which using the ITD as its primary cue provides 91% of efficacy under natural conditions. Through this application, the robotic system is able to determine if a sound in the environment is coming from the left, center, or right side of CARL. This feature is used as an aid for the navigation of the robot and is evaluated in Chapter 9.

Chapter 7

Bimodal Speech Recognition Using ANN

In this chapter we present the development of a speech recognition system that is integrated with our robotic system. This is a bimodal system that would train an ANN with auditory and visual clues in order to control the locomotion of CARL. Section 7.1 presents the auditory recognition approach, Section 7.2 describes the visual recognition model, and Section 7.3 summarizes our findings.

7.1 Speech Recognition by Image Processing its Spectrogram

This section introduces a novel speech recognition technique, which is based on processing a visual representation of speech. Section 7.1.1 impart the theory that supports our approach, Section 7.1.2 details our methodology and implementation, and Section 7.1.3 provides the evaluation of the approach.

7.1.1 Founding Principles

Speech is voice modulated by the throat, tongue, lips, *etc*. This modulation is accomplished by changing the form of the cavity of the mouth and nose through the action of muscles that move their walls [11]. Physically, speech consists of air pressure variations produced in the vocal tract. Recording speech and synthesizing it is well understood for phoneticians; however, reverse mapping speech signals is not an easy task. In order to understand speech and capture some aspects of it on paper or on a computer screen, researchers have developed instrumental analysis of speech. This instrumentation includes X-ray photography, air-flow tubes, electromyography, spectrographs, mingographs, laryngographs, *etc*. [10]. The ultimate goal for these researchers was to *visualize the speech* in some way so that they could capture phonetic clues.

At present, great phonetic insights have been achieved with the aid of these visual tools. For instance, an experienced spectrogram reader has no trouble identifying the word "compute" from the visually salient patterns in the image representation. Here rests the foundation and novelty of our speech recognition system. If humans extract phonetic cues by the use of their visual system, we reasoned that it would be possible to extract similar cues by the use of image processing and computer vision techniques. This chapter implements a speech recognition system based on speech visualization data. There are many ways to visualize speech on computer systems, and all of them are closely related. Understanding these visualization techniques is the key of the success of our approach. We consider these techniques in the remainder of this section.

Speech waveform (oscillogram)

Physically, the speech signal (actually, all sound) is a series of pressure changes that occurs between the sound source and the listener. The most common representation of the speech signal is the oscillogram, often called the waveform (see Figure 7.1). The oscillogram is the temporal domain depiction of speech and represents the fluctuations in air pressure over time. The major time domain parameters of interest are duration and amplitude.

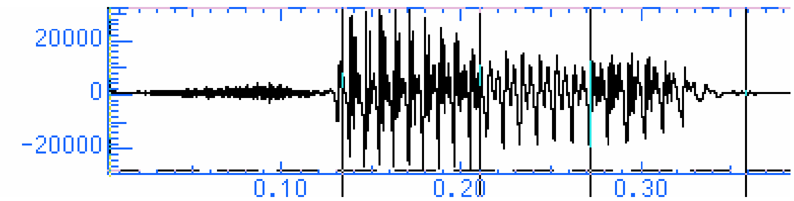

Figure 7.1: Visual representation of speech in the time domain: the waveform. *y*: amplitude; *x*: time.

Fundamental frequency (pitch F0)

Another representation of the speech signal is the one produced by pitch analysis. Pitch analysis tries to capture the fundamental frequency of the sound source by analyzing the final speech utterance. A graph of fundamental frequency is depicted in Figure 7.2.

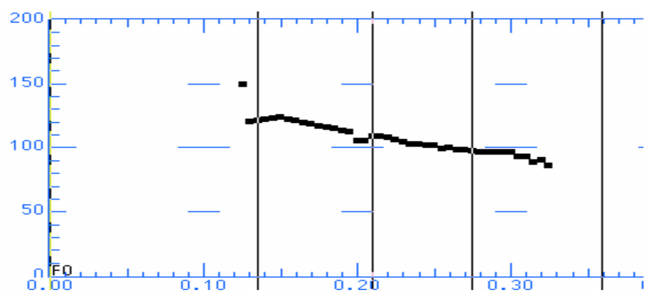

Figure 7.2: Speech fundamental frequency. The dominating frequency of the sound produced by the vocal chords.

Speech is normally looked upon as a physical process consisting of two parts: a product of a sound source (the vocal chords) and filtering (by the tongue, lips, teeth, *etc.*). The fundamental frequency is the dominating frequency of the sound produced by the vocal chords. It is the strongest correlate to how the listener perceives the speaker's intonation and stress.

Speech spectrum

According to general theories, each periodical waveform may be described as the sum of simple sine waves, each with a particular amplitude, frequency, and phase [22]. The spectrum depicts the distribution of frequency and amplitude at a moment in time. In this representation, the horizontal axis represents frequency, and the vertical axis represents amplitude. If we want to plot the spectrum as a function of time, we need a way of representing a three-dimensional diagram. One such representation is the spectrogram. By the use of Fourier techniques, it is possible to totalize the amplitudes for every frequency component within a period of time. This would be the representation of speech in the time domain, or the spectrum of the speech. A speech spectrum is depicted in Figure 7.3.

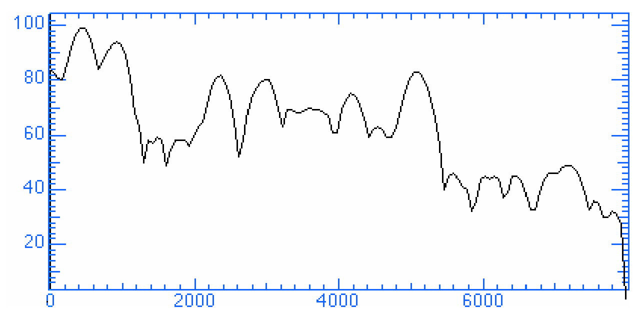

Figure 7.3: Two-dimensional representation of speech in the frequency domain: the spectrum. *y*: amplitude; *x*: frequency.

Speech spectrogram

In a spectrogram, the time axis is the horizontal axis, and frequency is the vertical axis. The third dimension, amplitude, is represented by shades of darkness. The spectrogram can be considered as a number of spectrums in a row, looked upon "from above", where the highs in the spectra are represented with dark spots in the spectrogram. Figure 7.4 is an example.

The spectrogram allows the visual analysis of phonetic cues. In the unvoiced fricative sounds, the energy is concentrated high up in the frequency band, and quite disorganized (noise-like) in its appearance. In other unvoiced sounds (*e.g.,* the plosives) much of the speech sound actually consists of silence until strong energy appears at many frequency bands, as an "explosion". The voiced sounds appear more organized. The spectrum highs (dark spots) form horizontal bands across the spectrogram. These bands represent frequencies where the shape of the mouth gives resonance to sounds. The bands are called formants and are numbered from the bottom up as F1, F2, F3, *etc*. The positions of the formants are different for different sounds, and they can often be predicted for each phoneme.

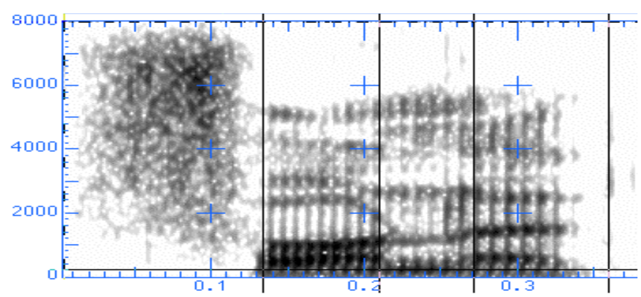

Figure 7.4: Three-dimensional representation of speech: the spectrogram. y: frequency; x: time; darkness: amplitude.

Speech waterfall spectrogram

The waterfall spectrogram is another way of viewing the three-dimensional plot of time, frequency, and amplitude. Figure 7.5 is an example of this representation. This picture is looked upon diagonally with time and frequency along the bottom axes. The amplitude is visible for each spectrum. The formants can be seen as sequential high points in each spectrum.

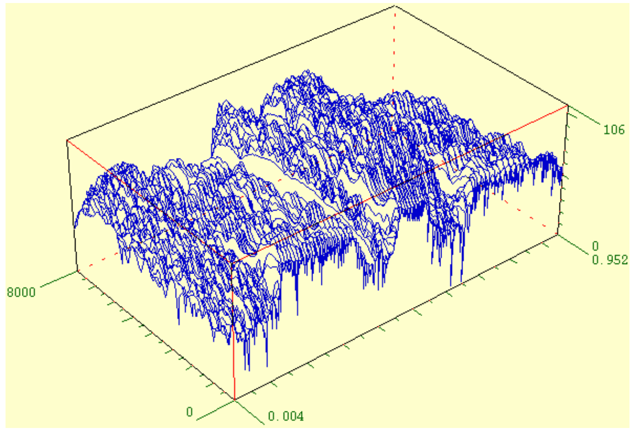

Figure 7.5: Three-dimensional representation of speech: the waterfall spectrogram. x: time; y: frequency; z: amplitude. Source: [10].

7.1.2 Speech Recognition Implementation

In this subsection we describe the implementation of a novel single speaker speech recognition system. First, we explain *how* and *when* we capture our speech samples for real time operation. Then, we present our methodology for processing the speech signal and extracting the feature vectors. Finally, we detail our neural network design and how we trained it.

Speech data acquisition

Acquiring data for speech recognition systems consists of recording speech samples in digital format so that it can be processed *via* software. Our speech recognition system captures speech data from CARL's microphones, which is transmitted to Brainstem. In addition, considering that our speech recognition application is integrated with our robotic system, real time factors need to be considered. The main concerns here are to define when to start the capture of speech data, and when to stop the capture. We solve the first issue by defining a minimum level of loudness *via* software, so that when the audio input is above this limit, the system starts capturing data. In addition, to avoid the loss of data in those words that start with amplitude below the limit, a delay feature is included. In our case the delay is -0.3 second. We solve the second issue by setting a constant length of speech capture. In our case the time is 0.75 second. The data acquisition application was implemented in Matlab. Figure 7.6 shows the data acquisition output for three different speech inputs. These are the speech waveforms and will be used to demonstrate our methodology.

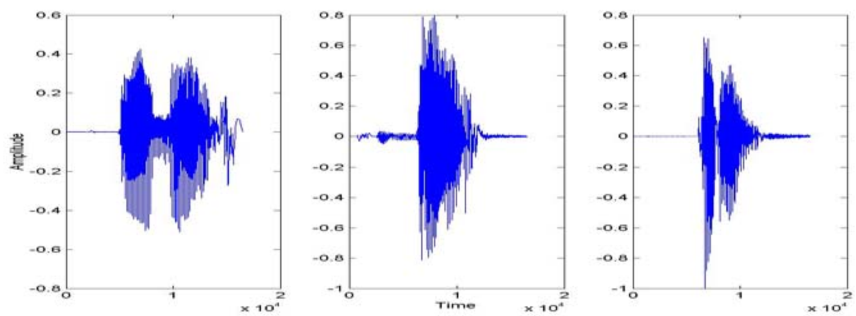

Figure 7.6: Waveform of three different speech samples. Trigger value: 0.4 volt; trigger delay: -0.3 sec.; data length: 0.75 sec.

Speech processing

Speech visualization is the basis of our speech recognition methodology. The hypothesis is that if humans can read phonetic cues by visually inspecting a speech representation and accordingly classify them, then computer systems could do a similar job by the use of computer vision techniques and by image processing the representation. The speech spectrogram is the representation that includes all the features of speech. Although there exist many types of spectrograms such as RASTA and PLP [17], we decided to develop our experiment using standard spectrograms. The first processing step is to extract the spectrograms from the speech signals captured. Figure 7.7 shows the spectrograms of three different speech samples. These are single words, and we chose a variety of words to demonstrate that our approach is word-independent. Before extracting the spectrogram, we found two important speech data requirements: speech capture must be sampled at high frequency (this is for high spectrogram resolution, we use 22000 Hz), and all speech samples must be standardized at the same limits of amplitude.

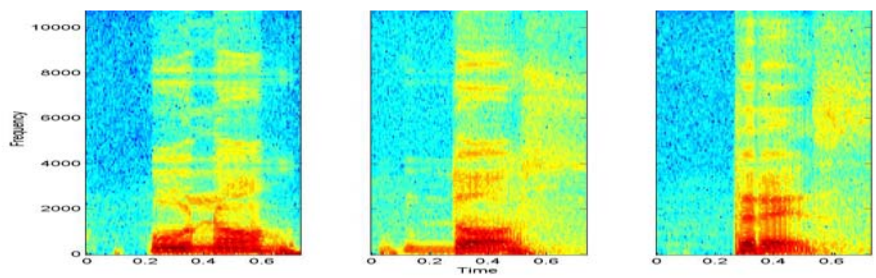

Figure 7.7: Spectrograms extraction of three different, and standardized words using Matlab: specgram(speech_data, 256, sample_rate).

Matlab creates speech spectrograms using Fourier analysis windowed across the time domain, so that all spectrogram row data consists of complex numbers. Therefore in order to image process the spectrogram it is necessary to convert the spectrogram into an image. We achieved this by converting the complex numbers into absolute numbers. Next, we image process the spectrogram so that every noise is removed and speech formants and other cues are stressed, as depicted in Figure 7.8. The image processing of the spectrogram consisted of very-low-pass filtering, low-pass filtering, and complete threshold. As a result, the speech shape is preserved and presented at the same amplitude across the time and frequency. Identical words presented similar visual cues, and different words showed different visual cues.

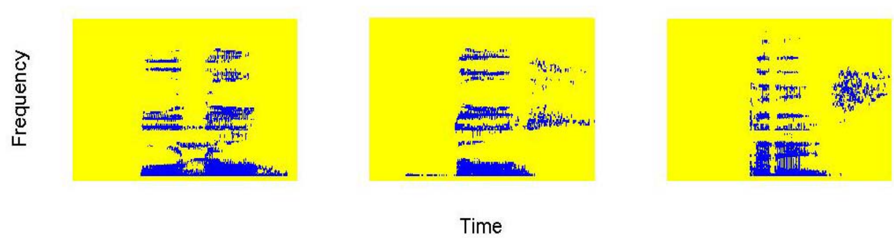

Figure 7.8: Image processed spectrogram results of the three different words. Noise is removed and important cues are stressed.

Feature vector extraction

The output of the speech-processing step is a simplified visual representation of speech, where similar speech samples can be recognized and dissimilar speech samples can be differentiated. This differentiation is a primary and practical condition before building feature vectors for artificial neural networks. Our first attempt to extract the feature vectors from the image-processed spectrogram was in the frequency domain. We totalized in a vector all the amplitudes values across the time for every frequency component. Next, to reduce data redundancy, we down-sampled the vector. Feature vector results are presented in Figure 7.9.

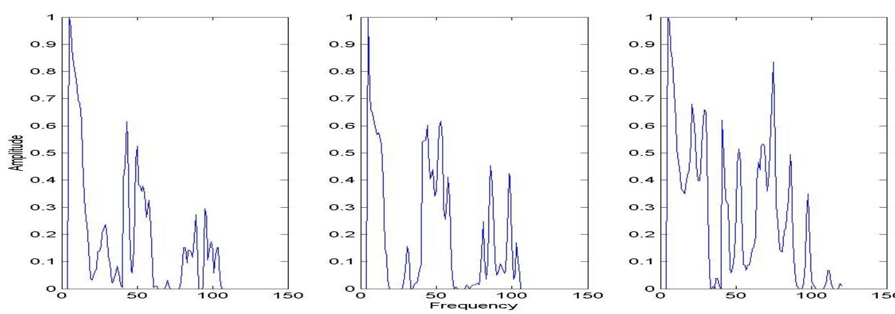

Figure 7.9: Feature vectors extracted from the image-processed spectrograms in the frequency domain.

Although this feature vector is valid, it does not give us the complete signature of the image processed spectrogram because cues in the time domain are missing. Another feature vector can be extracted similarly in the time domain. The sequential integration of both vectors will be our final feature vector used to train our neural network. The composed feature vector has a length of 210, 120 belong to frequency cues and 90 to time cues. These final vectors are depicted in Figure 7.10. We also found similarities when plotting feature vector samples of the same speech.

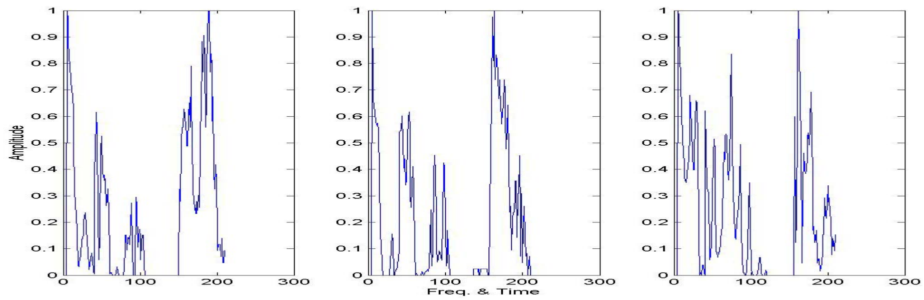

Figure 7.10: Final feature vectors composed by cues in the frequency and time domain.

Our feature vector set for neural network training consists of five words of robot locomotion control: GO, BACK, STOP, LEFT and RIGHT. We created twenty feature vectors for each word.

Neural network design

Although there are many neural network alternatives, we decide to use the multi-layer feed-forward network model trained using the back-propagation algorithm. We chose this model and algorithm because they are practical, effective, and supported by Matlab. The feed-forward model consists of layers of nodes (see Figure 7.11). Inputs are applied to the first layer, and outputs are retrieved from the nodes at the output layer. Between adjacent layers, nodes are fully interconnected, but within the same layer there is no connection.

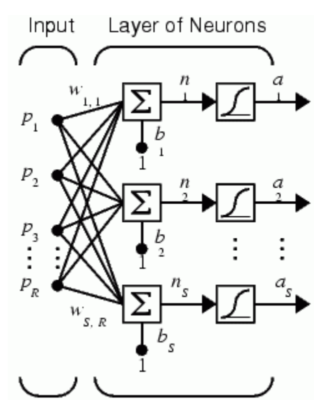

Figure 7.11: A single-layer feed-forward network. R: number of elements in input vector; S: number of neurons in layer; $a = \mathrm{f}\,(W.p + b)$. Source: [24].

We created our neural network model using the toolbox of Matlab [24]. In Matlab the function `newff` creates a feed-forward network. It requires four inputs and returns the network object. We used the following script to create our network model:

```
net=newff(min_max,[25,1],{'tansig','purelin'},'trainrx');
```

The first input (*min_max)* is an *R* by 2 matrix of minimum and maximum values for each of the *R* elements of the input vector. The second input ([25,1]) is an array containing the sizes of each layer. We use two layers: the first one has 25 neurons and the second one has one neuron. The third input ({'tansig','purelin'}) is a cell array containing the names of the transfer functions to be used in each layer. The final input is the name of the algorithm function to train the network. We use a derivation of the back-propagation method: *trainrx*.

Neural network training

The back-propagation algorithm consists of error-back propagation that allows supervised training of multi-layers of nodes [29]. This method is a gradient-search technique that minimizes a cost function between the desired outputs and those generated by the net. The aim is to establish a functional relationship for a given problem by adjusting the weights between neurons. After selecting some initial values for the weights and internal thresholds, input/output patterns are presented to the network repeatedly and, on each presentation, the states of all nodes are computed starting from the bottom layer and moving upward until the states of the nodes in the output layer are determined. At this level, an error is estimated by computing the difference between the outputs of the nodes and the desired outputs. The variables of the net are then adjusted by propagating the error backwards from the top layer to the first layer.

With standard back-propagation, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate [6]. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. However this sensitivity can be improved if we allow the learning rate to change during the training process. An *adaptive learning rate* will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is responsive to the complexity of the local error surface. Another method that will provide a faster convergence is *back-propagation with momentum*. This method allows a network to respond not only to the local gradient but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum, a network may get stuck in a shallow local minimum. With momentum, a network can slide through such a minimum.

We train our neural network with an algorithm that combines both approaches above mentioned: the back-propagation with momentum and the adaptive learning rate. This is provided by Matlab as the *trainrx* training function. The Matlab script to train the feed-forward network consists of:

```
%define initial learning rate
net.trainParam.lr = 0.001;

%define goal performance
net.trainParam.goal = 1e-4;

%define momentum, to ignore small features in the error surface
net.trainParam.mc = 0.9;

%define variable learning rate
net.trainParam.lr_inc = 1.02;
```

```
%define number of epochs of training
net.trainParam.epochs = 20000;

%start training the neural network
[ tr_net ]=train( net, fv_all, tgt );
```

The last script starts the neural network training for speech recognition of robot control words. It uses the *train* function with three inputs. The first input is the feed-forward network (*net)*. The second input (*fv_all*s) is a matrix that encloses all the speech feature vectors: GO, BACK, STOP, LEFT and RIGHT, 20 samples of each of them. Each column of *fv_all* is one feature vector. The third input (*tgt*) is the target of the network. It is an array of 100 columns, where each element is a number that identifies the target word.

7.1.3 Approach Evaluation

In this section we evaluate our speech recognition approach by analyzing the quality of the generated feature vectors and by monitoring the ANN training process. Figure 7.12 presents a plot of 20 feature vectors for the keyword "STOP", captured and processed in real time.

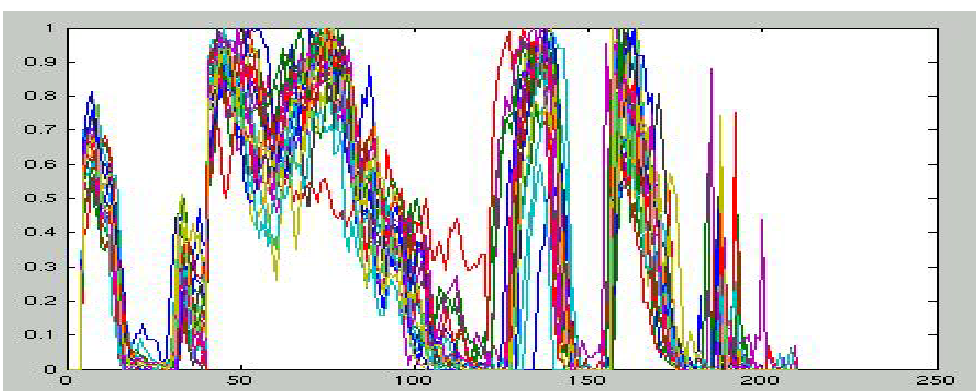

Figure 7.12: Plot of 20 feature vectors for the keyword "STOP". Same speaker.

As we can devise in this figure, despite the fact that the speech samples were captured at different conditions, i.e. voice intonation, voice level, distance from the microphone, and random noise, we were able to extract a consistent pattern. Having a similar pattern for the same keyword, but dissimilar to other keywords is a guarantee for a successful ANN training. From the plot, we can also notice that although the vectors have similar pattern, they have different phase. This was expected and it is because the triggering of the capture is different for every sample. We expect this to be alleviated by the neural network model.

We trained our neural network model for the recognition of the five keywords using 20 speech samples of each one from the same speaker. As Figure 7.13 depicts, the training process was favorable along the epochs, and after 20000 epochs the average error between the outputs and targets was reduced from 1100 to 0.00091. Although the goal was 0.00010, we cannot predict the system performance unless we test the recognition system in real work. The training process took about 3 minutes on a Cortex node.

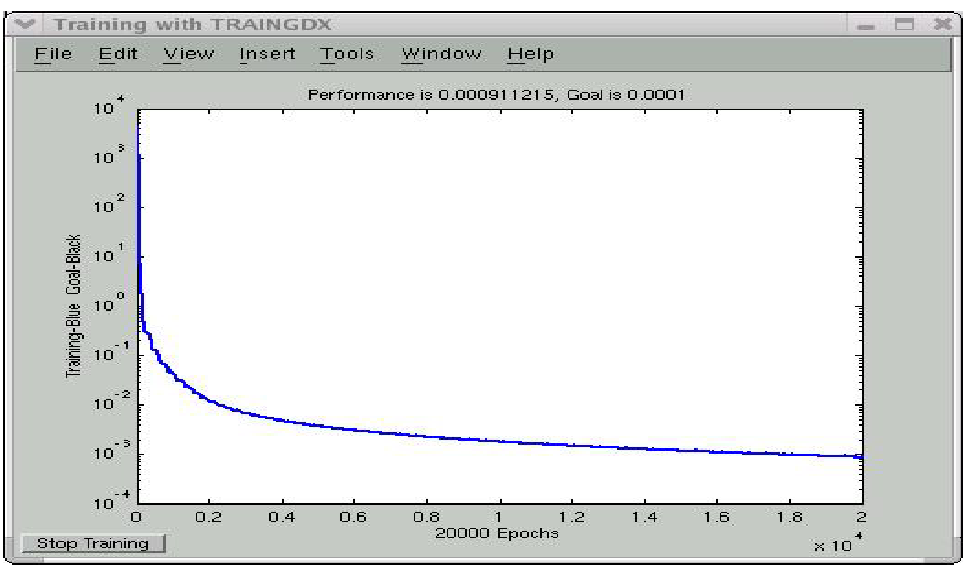

Figure 7.13: Training progress of the feedforward network using backpropagation algorithm with momentum and variable learning rate.

7.2 Mouth-video Processing for Speech Recognition Support

In this section we describe a supplementary technique to enhance the performance of the speech recognition system presented in Section 7.1. This technique processes video frames of the speaker's mouth to extract cues that characterize a particular spoken word. These cues would become the feature vectors for the training of an artificial neural network. Our approach assumes the face and mouth localization of the target speaker. Therefore we focus on the image processing of the mouth frames and on the extraction of feature vectors.

Our method is motivated by the fact that humans can decipher words by visually analyzing the mouth of a speaker. Although our analysis could be more complex, we simplify it by considering the mouth as one ellipse whose diameters change in time. Therefore, the image-processing job consists of extracting the lips from every mouth frame. We performed real-time experiments with the aid of Matlab toolboxes along with VFM (vision for Matlab) software [34]. The result for each speech sample consisted of 7 or 8 frames showing lip contours in black and white. From each set of frames we created one feature vector that would be the signature of the speech sample. From each processed frame we extracted the diameters in the x and y directions of the imaginary ellipse. The sequential variation of these two parameters across the 8 frames shaped our feature vector. We created feature vectors for neural network training from videos of the same speech samples used for the speech recognition system of Section 7.1.

7.3 Chapter Summary

Through this chapter we demonstrated the feasibility to image-process the speech spectrogram and extract simple and consistent patterns for speech recognition using ANN. Another important finding is our self-activated technique to capture speech data in real time. The performance of these finding are evaluated in Chapter 9 when they are integrated with the hierarchical system for robotic control by speech commanding. Finally, through our mouth-video processing approach we expect to strengthen the recognition system. We were able to extract unique visual cues, but its integration with the auditory module is still under research.

Chapter 8

Bimodal Speech Perception Using SNN

This chapter presents a spiking neural network example applied to bimodal speech perception. This is a research project in development by the Brain Computation Laboratory and has been used to test our hierarchical robotic system [19,20]. This approach consists of customizing NCS, which resides on Cortex, with biologically realistic parameters to recognize auditory and visual speech cues.

8.1 Data Acquisition and Spike Encoding

Audio-video-interleave (.avi) movies were recorded from ten volunteers speaking the following three sentences: “Attack with gas bombs,” “He is a loyal citizen,” and “I’m not entirely sure.” Each .avi was recorded at 25 frames per second with audio digitization of 11 KHz. Recordings were truncated to 1.6 seconds of audio and 40 frames of video to keep the sentences the same length. Auditory signals were processed using a short-time Fourier transform (STFT). A STFT decomposes the auditory signal into 129 frequency bands and provides the power of each frequency as a function of time (Figure 8.1).

By moving a narrow window (2.5ms) independently for each frequency across time, a probability of spiking is computed from the power within each window (normalized to the maximum power across all windows of all frequencies). The tonotopic representation of the cochlea is closer to a logarithmic scale, and the Fourier transform is a linear manipulation. In order to minimize the difference between cochlear processing and the STFT, a larger proportion of cells were encoded at lower frequencies than higher frequencies. Our auditory cortex included three columns. The first column received the first 20 frequency bands, the second column received the next 40 frequency bands, and the final column received the remaining 69 frequency bands.

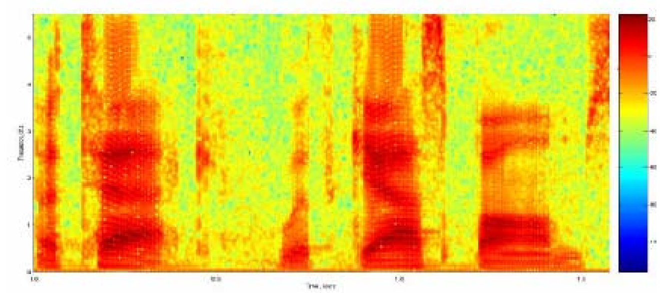

Figure 8.1: Spectrogram of the spoken sentence, "Attack with gas bombs". Vertical axis-auditory frequency (0-5.5 KHz in 129 bands). Horizontal axis–time in seconds (1.6 s). Pseudocolor legend–signal power in dB [-120, 25].

Visual signals were first whitened and then processed using Gabor analysis. The receptive field properties of primary visual cortex (VI) simple cells resemble Gabor-like properties [35], minimizing the tradeoff between frequency information and spatial information. Figure 8.2 shows two frames of an .avi movie before and after Gabor-filtering using horizontally oriented high- and low-band-pass filters. Frames a & d, before filtering. Frames b & e, after filtering with high-band-pass filter. Frames c & f, after filtering with low-band-pass filter. Frame g is the high-band-pass filter used (30x30). Frame h, is the low-band-pass filter used (30x30). In order to preserve the retinotopic mapping, the filtered image was broken down into 5x5 subregions. The average intensity within a subregion was used as the probability of spiking for a group of cells encoding that position.

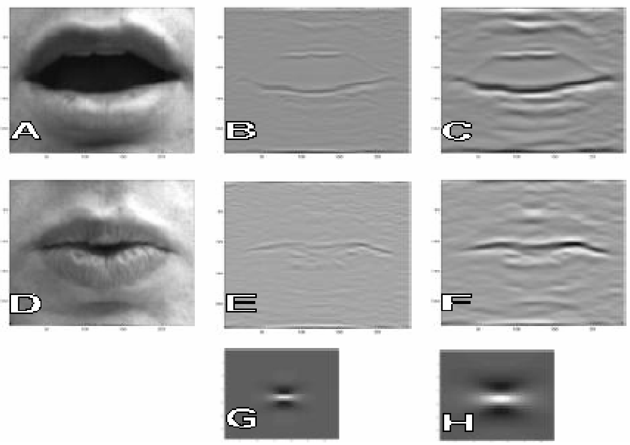

Figure 8.2: Two frames (240x240) from an .avi movie before and after horizontal Gabor-filtering. Source: [19, 20].

8.2 Network Design

Our network was made up of ten columns (6 visual, 3 auditory and 1 association). Each primary sensory column comprised two layers: an input layer (IV) and an output layer (II/III). Layer IV included 300 excitatory cells. Layer II/III included 300 excitatory and 75 inhibitory cells. Layer IV excitatory cells connected to layer II/III excitatory cells with a 10% probability. Layer II/III excitatory cells connected with each other and to inhibitory cells with a 10% probability. Inhibitory cells connected to excitatory cells within layer II/III with a 5% probability. The association column was made up of one input layer (IV) similar to the output layers of the primary sensory columns. The excitatory cells of layer II/III for the six visual and three auditory columns each connected with layer IV of the association column using a 1% probability. Simulations typically took approximately three to five minutes to process a three-second recording. Details of cell design and channel design are presented in [19, 20].

8.3 Network Training

Learning and training were designed to take advantage of the synaptic properties observed in neocortical tissue. Both short-term transient and long-term Hebbian-like synaptic changes were modeled. In order to mimic the feedback projections of the frontal cortex, training was accomplished by selectively injecting a unique subset of cells with current for each sentence presented to the network.

Our synapse model included reversal potential, conductance, A (absolute strength, or product of quantal size and number of release sites), U (mean probability of release), D and F (the time constants to recover from depression and facilitation, respectively). Details of parametric equations are completely characterized in [32] and [21]. F1 synapses predominately facilitate (F:D, 9.04 ± 1.85), F2 synapses depress (D:F, 40.6 ± 4.6), and F3 synapses are mixed (2.82 ± 4.6); further details can be found in [12].

8.4 Results

Spike-coded visual and auditory representations in primary sensory cortices demonstrated unique patterns for the three sentences. When output layers of these primary cortices interacted in multimodal association cortex, there was again preservation of unique spiking patterns.

Figure 8.3 shows the change in synaptic strength (USE) after successive sentence presentations for the rewarded *vs*. nonrewarded neurons. Rewarded neurons where given direct current injection during sentence presentation to bring their membrane potential closer to threshold. Further analysis of these results is available in [19, 20].

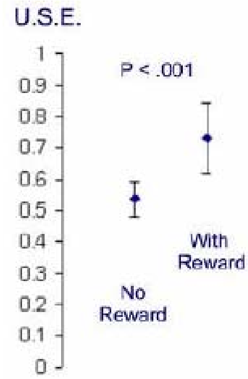

Figure 8.3: Utilization of synaptic efficacy. Mean USE (± 1 std) after 7 presentations of spoken sentence "Attack with gas bombs" among excitatory neurons in multimodal association cortex. Source: [19, 20].

8.5 Chapter Summary

In this chapter we presented an approach to design and train spiking neural networks for bimodal speech perception. When the network was tested using the three spoken sentences, it showed unique spike-coded patterns of the visual and auditory representations in primary sensory cortices. When output layers of these primary cortices interacted in multimodal association cortex, there was again preservation of unique spiking patterns. We use this trained network in real experimentation for robotic threat identification in Chapter 9.

Chapter 9

Project Evaluation

In this section we evaluate the entire hierarchical robotic system by performing two integrated experiments that use the AI functions provided at each layer. The main experiment is a robotic search and threat identification. This uses all three layers and SNN for high-level decisions. The other experiment is a robotic locomotion control by speech commanding. This also uses all three layers of the system and ANN for high-level processing.

9.1 Robotic Search and Threat Identification

By experimenting the complete hierarchical system we intend to demonstrate its effectiveness to dynamically interact with the environment and its ability to perform tasks of different levels of complexity. Robotic search and threat identification is an experiment that consists of CARL looking for a threat in the environment. This integrated experiment comprises many AI tasks sequenced as illustrated in Figure 9.1 and distributed in the three-layer system as depicted in Table 9.1. Initially CARL navigates on its environment by making use of its onboard reactive features such as random navigation and object avoidance.

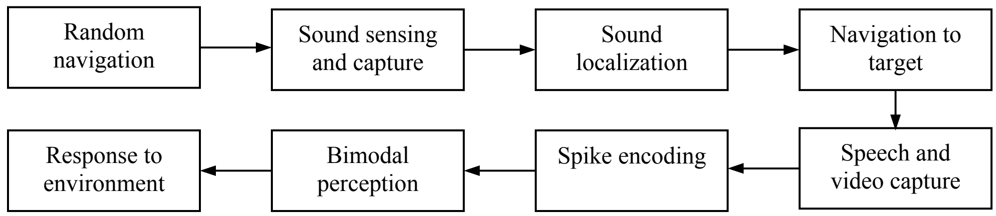

Figure 9.1: Task sequence of the integrated experiment.

When a sound from the environment is above a threshold, the robotic system captures the signal and tries to localize its origin. This is performed by the instinctive functionality of Brainstem. When the sound source is defined, CARL starts navigating towards the target until a touching sensor is activated, which indicates its encounter. The target is an animated human speaking mouth played on a LCD screen. At this point, CARL starts capturing speech and mouth frames from the target for threat identification, and delivers them to Brainstem. On this layer, these auditory and visual data are spike encoded, and delivered to Cortex. Here, NCS is executed using the bimodal perception model for threat identification. Finally, the cognitive function of Cortex outputs a signal that characterizes the target as: friend, foe, or unknown. According to this CARL gives a response to the environment.

Table 9.1: Distribution of tasks of the integrated experiment.

	Body	**Brainstem**	**Cortex**
Functions performed at this control level	Random navigation, object avoidance, data-metrics capture.	Sound localization, preparing spike codes from audio and video data for input to neocortex simulation.	Neocortical simulator software (NCS). Speech perception using audio and video (lip reading).
Location of this control function in the robotic system	On board the mobile robot, called CARL.	On nearby desktop class computer, called “Brainstem", connected to robot *via* wireless RF.	On remote large-scale parallel computer, called "Cortex", connected to Brainstem *via* the Internet.

Our experiments took place in the Brain Computation Laboratory facilities under office conditions of noise and echo; where CARL, Brainstem, and the target (the speaking LCD mouth) were co-located. The configuration of the arena of experimentation is illustrated in Figure 9.2. Short video clips demonstrating CARL's ability to localize sound, navigate to target and capture speech and mouth frames are available online at http://www.cs.unr.edu/~macera/threatID.html.

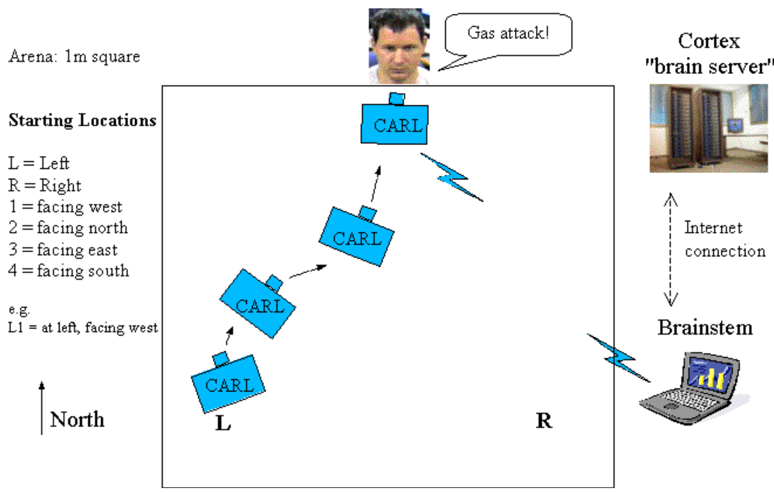

Figure 9.2: Robotic search and threat identification experiment.

Sound localization and navigation to target

Ten trials of navigation to target were performed from each starting location (left and right), and for each of the four orientations (see Figure 9.2), making a total of 80 trials. A trial consisted of interleaved events of sound localization and robot locomotion. Although ITD and IID are generally complementary techniques for estimating a sound direction, we found ITD to be considerably more robust and less subject to calibration errors and errors due to noise or echoes. For these reason our experiments were conducted using ITD alone.

Each trial comprised multiple individual left/right/center ITD computations, resulting in an incremental rotation, or movement toward the target if the ITD orientation remained unchanged. A trial was considered successful if CARL's front bumper made contact with the target and the middle 80% of the imaged lip was visible from CARL's onboard camera. CARL successfully navigated toward and contacted the target mouth region in 75 of 80 trials, as depicted in Table 9.2 (χ^2=20.3, P<0.0001, based on the number of possible endings along the edge of a meter square table surface). Each navigation experiment took between 25 to 30 seconds.

Table 9.2: Results of 80 experiments of navigation to target.

		LEFT - CARL Orientation				**RIGHT - CARL Orientation**			
		L1	L2	L3	L4	R1	R2	R3	R4
Exp. #	**1**	OK	OK	OK	OK	OK	OK	OK	OK
	2	OK	OK	OK	OK	OK	OK	**FAIL**	OK
	3	OK	**FAIL**	OK	OK	OK	OK	OK	OK
	4	OK	OK	OK	OK	OK	OK	OK	OK
	5	OK	OK	OK	OK	OK	OK	OK	OK
	6	OK	OK	**FAIL**	OK	OK	OK	OK	OK
	7	OK	OK	OK	OK	OK	**FAIL**	OK	OK
	8	OK	OK	OK	OK	OK	OK	OK	OK
	9	OK	OK	OK	**FAIL**	OK	OK	OK	OK
	10	OK	OK	OK	OK	OK	OK	OK	OK

Threat assessment using bimodal speech perception

After a successful localization of target, 1.6 seconds of audio and 23 frames of video are captured by CARL. Three sentences that respectively would typify a friend, foe or unknown are used: “He is a loyal citizen”, “Attack with gas bombs” and “I am not entirely sure”. These data are sent to Brainstem for spike encoding generation, which takes out 3.5 seconds during simulation. Figure 9.3 shows samples of audio-video capture after successful target localization.

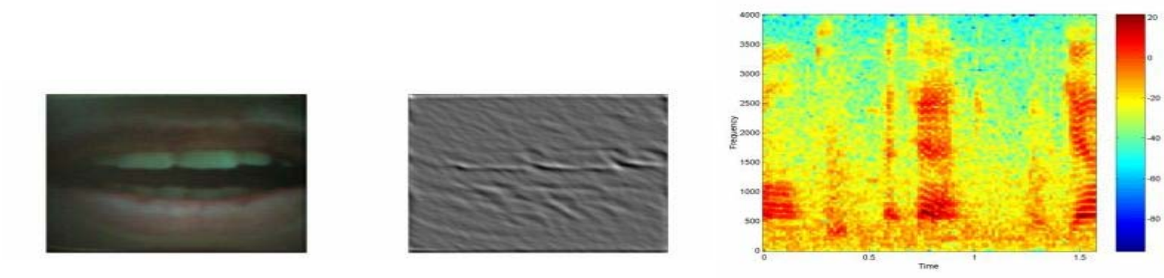

Figure 9.3: Left: Mouth frame sample capture by CARL. Center: Same frame after Gabor analysis. Right: STFT output of the speech captured.

When the stimuli data is ready, Brainstem establishes a TCP/IP connection with Cortex and streams the data towards it. Then the NCS program is invoked and injected with both the auditory-visual data and the ‘input file’. The input file defines the network specifically for bimodal speech perception and initializes the neo-cortical model according to the state of the network previously trained using the approach described in Chapter 8.

Figure 9.4 shows result examples of the bimodal perception of three spoken sentences obtained from NCS. The first column shows three sentences modified from the TIMIT corpus. Columns two and three show the spiking response of neurons driven from the

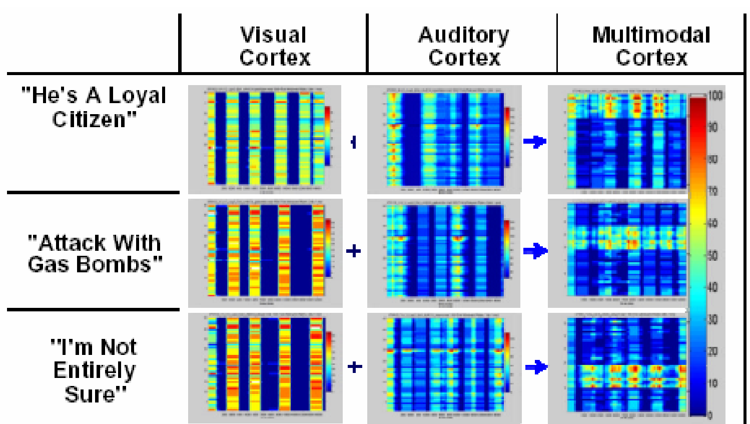

Figure 9.4: Bimodal speech perception results executed by NCS on Cortex. Pseudocolor windowed spike rate plots in response to spoken sentences. Length: 1.6 seconds; *y*: windowed spike frequency; *x*: time. Source: [19, 20].

visual and auditory transformations. The fourth column is the response of associative multimodal cortex during reward depolarization of selected neurons for each sentence.

As we can see in this figure, spike-coded visual and auditory representations in primary sensory cortices demonstrated unique patterns for the three sentences. When output layers of these primary cortices interacted in multimodal association cortex, there was again preservation of unique spiking patterns (Figure 9.4, fourth column).

Response to environment

After the neocortical simulation, an interpretation of the resulting spike code will determine accordingly if the target is a friend, foe, or unknown. At present the threat assessment is determined in terms of neuronal perception, further research is in progress at the Brain Computation Laboratory for speech threat identification. When a threat is pseudo-identified, a command will be passed back to Brainstem to take action on CARL. If the target is recognized as friend, the robot repositions and resumes searching. If the target is identified as foe, CARL rapidly backs away to escape. If the target is identified as unknown, the robot reposition and continue monitoring the target. Video clips of CARL's responses are available online at http://www.cs.unr.edu/~macera/threatID.html.

9.2 Robot Locomotion Control by Speech Commanding

This integrated experiment consists of controlling the CARL's locomotion through speech commands using our speech recognition system described in Chapter 7.1. There are two methods to achieve this. The first method is using the three robotic layers. CARL would capture the speech command and sent it to Brainstem. Brainstem would generate the feature vectors and sent it to Cortex. Finally, Cortex would simulate the ANN in sequential mode and sent back the recognized command for robot movement. The second method is using two robotic layers, CARL and Brainstem. In this case the ANN simulation would be performed in Brainstem. Considering that speech recognition is a high level cognitive function, we decided to perform our experiments using the first method. Under this configuration, we tested in real time the locomotion control of CARL using speech commands from approximately 1 meter of distance in the Brain Lab environment. Our system responded in average with 94% of effectiveness for five speaker dependent words commands (GO, BACK, STOP, LEFT and RIGHT – 20 trials each). Figure 9.5 shows an example of the efficacy in capturing and generating the feature vectors of the BACK command.

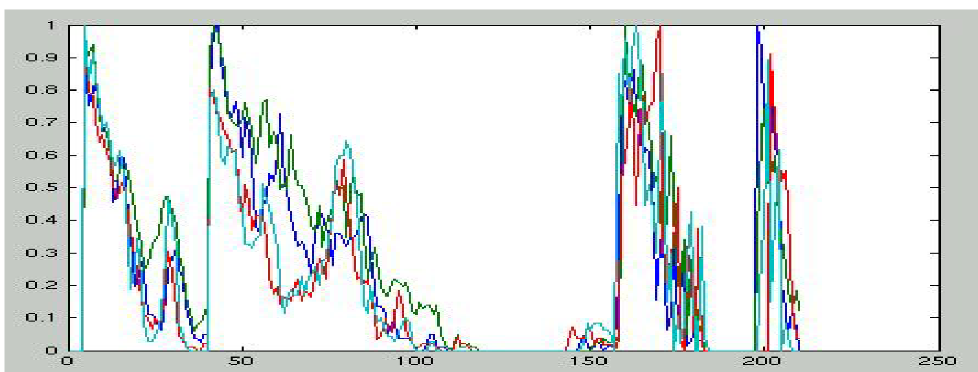

Figure 9.5: Feature vectors plot of "BACK" showing capture precision and pattern consistency of four real time trails.

9.3 Chapter Summary

In this chapter we integrated the three-layers of the robotic system Body, Brainstem and Cortex and put it at work in a common task: the intelligent control of CARL robot. The integration of three dissimilar computing systems, with reliable linkage and synchronization, is the main achievement of this project. In addition, we were able to distribute AI functions across the robotic layers according to their complexity and perform them effectively. We were able to capture auditory and visual signals efficiently, localize the sound origin, identify threat targets, recognize speech commands, and perform CARL navigation effectively.

Chapter 10

Conclusions

In this chapter we summarize our work, then we describe the contribution of this project, and finally we recommend some future work.

10.1 Project Summary

We have implemented a novel robotic architecture that helps to develop and test artificial intelligence models in the real world. This robotic architecture distributes the computational task on three layers that are remotely located but wireless linked: the Body (on board the robot), Brainstem (on a local PC), and Cortex (on a parallel computer cluster). Initially, in order to prove the feasibility of the proposal, we implemented the communication backbone of the system and tested it by controlling CARL, the robot, over the Internet from a remote location. The locomotion of CARL was successfully controlled in time, and precision from Cortex and the metrics of the robot were accurately monitored on Brainstem. Next, CARL was equipped with stereo auditory and visual capability by hardware integration and software implementation. The robotic platform was then used for a series of artificial intelligent functions, implemented on each layer with different computation complexity. On the Body, we implemented a simple navigation and object avoidance system. On Brainstem, we constructed a binaural sound localization and navigation-to-target system. On Cortex, we implemented a speech recognition system using ANN and we tested a bimodal speech perception approach using biologically realistic spiking neural networks. Finally, we evaluated the entire robotic system with two integrated experiments: (1) robotic search and threat identification and (2) robotic locomotion control by speech commanding.

10.2 Project Contribution

The first contribution of this project is the provision of a novel and effective platform for AI investigation. Through our biologically correlated robotic system, we successfully mimic two strategic operations of intelligent living creatures: (1) the effective interaction with the real world by modeling the brain, body, and environment conjunctively, and (2) the distribution of processes according to their complexity: reactive processing on the Body, instinctive processing on Brainstem, and cognitive processing on Cortex. Although the concept of remote-brained robotics has been explored previously by Inaba and colleagues in [15], in that work the brain and body were separated, both conceptually and physically, our system is novel in that it incorporates three-level hierarchical processing intended to model the efficiency of human neurological perceptual processing and decision making. In this configuration, task selection and allocation are relevant and contribute to effective robot responsiveness.

The hierarchical robotic system has been a valuable platform for our own AI research. With this platform we quickly and effectively implemented a sound localization and navigation-to-target system. Although we did not focus on the precise localization of sound such as [16], this function provides CARL with auditory perception and an effective way to track mobile sound targets, resembling again an important feature of biological entities. Another important achievement is our novel and practical speech recognition algorithm using an ANN. Some speech recognition algorithms focus on developing new mathematical models to represent the speech spectrogram, such as Perceptual Linear Predictive (PLP) analysis and RelAtive SpecTraAl (RASTA) processing [17], and others focus on the estimation of the short-term spectral envelope, such as filter banks, cepstral processing, and linear predictive coding (LPC) [11]. Our approach simply image processes standard spectrograms in order to stress the visually perceptible formants of the speech. Our experiment using speech to control the locomotion of CARL demonstrated a 94% effectiveness for speaker-dependent trails.

From a cognitive-science perspective, our remote-brained robotic system's massive parallel processing and its embodiment of perceptual decisions make our system a valuable platform for investigating new types of artificial intelligence such as applied neurocomputing [14] and evolutionary agents [31], where the active and strong relationship between the brain, body, and environment is fundamental for neural model development [5]. From a general perspective, our system is also notable for its ability to map many-to-many robots and "cortices" *via* a distributed communication network (here, the Internet). Each CARL could potentially communicate with many Cortex-like clusters globally distributed. In turn, each Cortex could simultaneously control (hence coordinate) many CARL robots. This approach would yield not only flexible distribution of the computational load across a dynamic problem-solving environment, but also redundancy that could sustain the system in the event of focal destructive events.

10.3 Future Work

Our robotic platform, along with the neo-cortical simulator, provides a great avenue for future investigation on biologically realistic neuronal modeling [18]. Certainly, complex brain functions such as cognition and memory will be harder to model unless we have a better understanding of neuron dynamics at both a micro level and a macro level, and unless we conceptualize the basic building blocks (structures) that make the neural system behave reasonably. While current techniques try to solve the neural puzzle by analyzing a huge search space, a challenging and promising future work is to analyze small spiking neural structures that evolve over time to bigger and complex structures with biological significance. This could be accomplished by means of evolutionary techniques applied to neural systems that interact with the body and the environment. Tracing the neural network during its evolution would lead us to identify hypothetic building blocks of neural systems. Our robotic platform provides the elements and computational power to accomplish this proposal.

To take advantage of the computational power of the robotic system, it would be valuable to provide Cortex with parallel implementations of ANN and genetic algorithms. These features would speed up the development of AI utilities for the robotic system and would help to brainstorm and experiment with AI models that combine ANN and genetic algorithms, a field little explored.

At present Brainstem monitors the vision, auditory, and metrics of CARL; however, Brainstem's functionality should be extended to monitor the processing of Cortex. The future work related to the mouth-video processing presented on Section 7.2 for the support of the speech recognition approach presented on Section 7.1 is to experiment combinations of ANN and feature vectors from both approaches in order to find a bimodal speech recognition technique of higher performance. Finally, with respect to the Body, we suggest to upgrade the onboard processor and memory in order to enrich the reactive functions and to take advantage of emergent wireless Ethernet technologies.

Bibliography

[1] Audio/Video Sender System, VK54A. Module manual. X10. ftp://ftp.x10.com/pub/manuals/vk54a-om.pdf, accessed on 11-04-03.

[2] Basic Stamp 2 (BS2). Programming language documentation. Parallax Inc. http://www.parallax.com/html_pages/tech/faqs/prgm_info.asp, accessed on 11-04-03.

[3] P. Bourke. Cross correlation. Technical Report. Swinburne University of Technology. http://astronomy.swin.edu.au/~pbourke/analysis/correlate, accessed on 11-04-03.

[4] N. C. Braga. *Robotics, Mechatronics and Artificial Intelligence*. Newnes, Boston, MA, 2002.

[5] H. J. Chiel and R. D. Beer. The brain has a body: Adaptive behavior emerges from interactions of nervous system, body and environment. *Trends in Neurosciences*. Elsevier, Amsterdam, Netherlands, 1997.

[6] W. Chou and B. H. Juang. *Pattern Recognition in Speech and Language Processing*. CRC Press, Boca Raton, Florida, 2003.

[7] Descartes. Operating and assembly manual. Living Machines. 310 E. Locust, Lompoc, CA 93436. http://www.robotalive.com, accessed on 11-04-03.

[8] R. O. Duda. Sound localization research. Technical Report. San Jose State University. http://www-engr.sjsu.edu/~duda/Duda.Research.html, accessed on 11-04-03.

[9] M. Erturk, C. P. Brown, D. J. Klein, and S. A. Shamma. A neuromorphic approach to the analysis of monaural and binaural auditory signals. Technical Report. Institute for Systems Research & Dept. of Electrical Engineering. University of Maryland, MD, 2002.

[10] M. Filipsson. Speech analysis tutorial. Technical Report. Dept. of Linguistics and Phonetics. Lund University, Lund, Sweden, 2003.

[11] B. Gold and N. Morgan. *Speech and Audio Signal Processing, Processing and Perception of Speech and Music*. John Wiley & Sons, Inc., New York, NY, 2000.

[12] A. Gupta, Y. Wang, and H. Markram. Organizing principles for a diversity of GABAergic interneurons and synapses in the neocortex. *Science*, 2000, 287(5451): pp. 273-278.

[13] F. C. Harris Jr., J. Baurick, J. Frye, J. G. King, M. C. Ballew, P. H. Goodman, and R. Drewes. A novel parallel hardware and software solution for a large-scale biologically realistic cortical simulation. Technical Report. Brain Computation Laboratory, University of Nevada, Reno, 2002.

[14] J. J. Hopfield and C. D. Brody. What is a moment? "Cortical" sensory integration over a brief interval, *Proceeding of the National Academy of Science*, December 2000, Vol. 97, No. 25, pp. 13919-13924.

[15] 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, No. 10, pp. 933-954.

[16] R. E. Irie. Robust sound localization: An application of an auditory perception system for a humanoid robot. MS Thesis. Massachusetts Institute of Technology, Cambridge, MA, 1995.

[17] C. M. Jones. Speech and natural language processing. Technical Report. Heriot-Watt University, Edinburgh. http://www.cee.hw.ac.uk/~cmj, accessed on 11-04-03.

[18] C. Koch and I. Segev. *Methods of neuronal modeling*. MIT Press, 2nd edition, Cambridge, MA, 1998.

[19] J. C. Macera, P. H. Goodman, F. C. Harris, Jr., R. Drewes, and J. Maciokas. Remote-neocortex control of robotic search and threat identification. *To appear: Robotics and Autonomous Systems.* Elsevier, Amsterdam, Netherlands.

[20] J. Maciokas, P. H. Goodman, and F. C. Harris, Jr. Large-scale spike-timing-dependent-plasticity model of bimodal (audio/visual) processing, Technical Report. Brain Computation Laboratory. University of Nevada, Reno, 2002.

[21] H. Markram, *et al*. Potential for multiple mechanisms, phenomena and algorithms for synaptic plasticity at single synapses. *Neuropharmacology*, 1998, 37(4-5): pp. 489-500.

[22] J. H. McClellan, R.W. Schafer, and M.A. Yoder. *Digital Signal Processing First: A Multimedia Approach*. Prentice Hall, Englewood Cliffs, NJ, 1998.

[23] MATLAB Release 12.1. Data acquisition toolbox documentation. The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098. http://www.mathworks.com/access/helpdesk/help/toolbox/daq/daq.shtml, accessed on 11-04-03.

[24] MATLAB Release 12.1. Neural network toolbox documentation. The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098. http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet.shtml, accessed on 11-04-03.

[25] MATLAB Release 12.1. Signal processing toolbox documentation. The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098. http://www.mathworks.com/access/helpdesk/help/toolbox/signal/signal.shtml, accessed on 11-04-03.

[26] Myricom Inc. Creators of Myrinet. 325 N. Santa Anita Ave. Arcadia, CA 91006. http://www.myrinet.com, accessed on 11-04-03.

[27] D. Purve, G. J. Augustine, D. Fitzpatrick, L. C. Katz, A. LaMantia, J. O. McNamara. *Neuroscience*, Sinauer Associates Inc., Sunderland, MA USA, 1997.

[28] Radio Frequency Transceiver, RF-433. Module manual. Parallax Inc. http://www.parallax.com/dl/docs/prod/comm/27986-8.pdf, accessed on 08-15-02.

[29] M. G. Rahim. *Artificial Neural Networks for Speech Analysis/Synthesis*. AT&T Bell Laboratories. Murray Hill. New Jersey, 1994.

[30] Rocks Cluster Distribution. San Diego Super Computing Center. http://rocks.npaci.edu, accessed on 11-04-03.

[31] E. Ruppin. Evolutionary autonomous agents: A neuroscience perspective. *Nature Reviews Neuroscience*, Vol. 3, No. 2, February 2002, pp. 132-141.

[32] W. Senn, H. Markram, and M. Tsodyks. An algorithm for modifying neurotransmitter release probability based on pre- and postsynaptic spike timing. *Neural Computation*, 2001, 13(1): pp. 35-67.

[33] TV Wonder USB. Technical Report. ATI Technologies. http://www.ati.com/products/tvwonderusb/index.html, accessed on 11-04-03.

[34] Vision For Matlab, VFM. Intelligent Systems Laboratory. Technion. http://www.cs.technion.ac.il/Labs/Isl/Vision4Matlab/vision_for_matlab.htm, accessed on 11-04-03.

[35] M. A. Webster and R.L. De Valois. Relationship between spatial-frequency and orientation tuning of striate-cortex cells. *J. Opt. Soc. Am. A.*, 1985, 2(7): pp. 1124-1132.

[36] E. C. Wilson. Parallel implementation of a large scale biologically realistic neocortical neural network simulator. MS Thesis. University of Nevada, Reno, 2001.

[37] E. C. Wilson, P. H. Goodman, and F. C. Harris, Jr. Implementation of a biologically realistic parallel neocortical-neural network simulator. In Micheal Heath, *et al*., Editor, *Proc. of the 10th SIAM Conf. on Parallel Process*. For Sci. Comput., Portsmouth, Virginia, March 2001.

[38] E. C. Wilson, F. C. Harris, Jr., and P. H. Goodman. A large-scale biologically realistic cortical simulator. In Charles Slocomb, *et al*., editor, *Proc. of SC 2001*, Denver, CO, November, 2001.

Appendix 1

Schematics of CARL's sensor, drive and processing system. Source: [7].