Glove-TalkII: Mapping Hand Gestures to Speech Using Neural Networks – An Approach to Building Adaptive Interfaces

by

S. Sidney Fels

A thesis submitted in conformity with the requirements
for the degree of doctor of philosophy
Graduate Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4

 This research is supported by Natural Sciences and Engineering Research Council (NSERC) and the Institute for Robotics and Intelligent Systems (IRIS).

Glove-TalkII: Mapping Hand Gestures to Speech
Using Neural Networks –
An Approach to Building Adaptive Interfaces

S. Sidney Fels, Ph. D., Dept. of Computer Science, University of Toronto, 1994

Abstract

Glove-TalkII is a system which translates hand gestures to speech through an adaptive interface. Hand gestures are mapped continuously to 10 control parameters of a parallel formant speech synthesizer. The mapping allows the hand to act as an artificial vocal tract that produces speech in real time. This gives an unlimited vocabulary in addition to direct control of fundamental frequency and volume. Currently, the best version of Glove-TalkII uses several input devices (including a Cyberglove, a 3-space tracker, a keyboard and a foot-pedal), a parallel formant speech synthesizer and 3 neural networks. The gesture-to-speech task is divided into vowel and consonant production by using a gating network to weight the outputs of a vowel and a consonant neural network. The gating network and the consonant network are trained with examples from the user. The vowel network implements a fixed, user-defined relationship between hand-position and vowel sound and does not require any training examples from the user. Volume, fundamental frequency and stop consonants are produced with a fixed mapping from the input devices. One subject has trained to speak intelligibly with Glove-TalkII. He speaks slowly with speech quality similar to a text-to-speech synthesizer but with far more natural-sounding pitch variations.

Acknowledgements

A great many thanks to my supervisor, Geoff Hinton for his guidance, support and assistance with this undertaking. Thanks to Bill Buxton for his enthusiasm and discussions along the way with this project in addition to being on my committee. Thanks to my other committee members, Allan Jepson, Michael Jordan, Keith Oatley and K.C. Smith, for spending the time to critique my work and guide me through the process.

Several people must also be thanked for their invaluable assistance with some necessary hardware and software. First, thanks to Mike Ruicci for patiently helping while prototyping different circuits to get all the hardware working. Eric Lewis generously provided the text-to-speech software (free of charge!). Mark Huckvale magnanimously gave me the software version of the JSRU parallel formant synthesizer to get me started (free too!). Thanks also to Jim Kramer, who developed the Cyberglove. Without it, the project would not have been possible at all.

Many visitors had a major influence on the direction of the research and the eventual design. Much thanks goes to John Bridle, Mike Jordan, David Mackay, Mike Mozer and Ray Watrous. It's difficult to thank-you all enough.

A big thank-you must go to Mike Revow for his work with the graphics code and assistance getting the JSRU synthesizer working with the SGI box. This is in addition to the thanks for being an office mate and having to listen to the significant amount of noise that was produced from the synthesizer during development and demos. Likewise, thanks to my other office mates, Drew and Maureen.

The Connectionist Research Group at U. of Toronto have all contributed with discussions and ideas which most likely have seeped unconsciously into the design of Glove-TalkII. So, thanks to all of you, past and present; Sue Becker, Michelle Craig, Peter Dayan, Brendan Frey, Radford Neal, Steve Nowlan, Sageev Oore, Tony Plate, Carol Plathan, Dave Plaut, Ed/Carl Rasmussen, Mike Revow, Maureen Smith, Evan Steeg, Drew van Camp, Chris Williams and Rich Zemel. Together, you are a great group to work with. Thanks goes to the Input Research Group (IRG) headed by Bill at U. of Toronto for providing terrific discussions on the design and implications of this research. Thanks also goes to the group at Institute of Perception Research (IPO) in Eindhoven; especially Barry Eggen and Jack Gerrisson for helping me with understanding speech better.

Special thanks must also be sent to Sageev, since he's the "subject" who learned to speak with Glove-TalkII. Because of his talent, focus and effort he was able to speak, demonstrating nicely how Glove-TalkII works.

Thanks must go to my family and friends who provided lots of support and necessary distraction. And finally, and importantly, a heart felt thanks to my partner and proof-reader, Kirsty.

Contents

List of Figures

[1] The independence assumption is not necessary.

List of Tables

Chapter 1

Introduction

Imagine playing a special type of piano. As you practice your pieces, the key strokes you make are stored. After your practice, the keyboard is altered automatically so that the mistakes you made on the pieces are minimized. When you return the next day for another practice session, the keyboard has changed so that the difficult fingerings for some of the pieces are much easier, allowing you to master pieces quickly. This imaginary piano uses an *adaptive* interface.

Interfaces between users and devices are usually static. Typical examples of static interfaces are computer keyboards, musical-instrument controls, and industrial control panels. If the interface is poorly designed, users will not be able to learn to use the device (or will at least require extensive training to learn to use it). In such instances, the usual practice is to redesign the interface and send out a new version. This is a costly procedure. Further, the new version may still not suit enough users, especially if the users vary, so different models of the same device may be required. For devices that require "high-bandwidth" control, the danger of providing a poor mapping for any particular user increases due to the increased control space.

One alternative to building a fixed interface in such cases is to use an *adaptive interface*. An adaptive interface changes its behaviour to improve a user's performance based upon experience with the user. Neural networks provide natural models for building adaptive interfaces. A key characteristic of neural networks is that they can learn most input-to-output function from examples. Once trained, they run fast enough for real-time control, even on serial machines. For a device to have an adaptive interface, it must be able to change its model of the world. A device's model of the world contains three main parts: a user model, a domain model, and a communication model. Changing any of these three models (based on experience with the user) causes the interface between user and device to adapt.

This dissertation begins with an investigation of various strategies for adapting a computer's model of the world. The interface designer's choice of strategy determines an algorithm most appropriate for the task. This dissertation also explores the use of neural networks for implementing adaptive interfaces. Specifically, the research focuses on implementing a gesture-to-speech device using neural networks to illustrate their use in the creation of adaptive interfaces.

Gesture-to-speech mappings form a spectrum based on the granularity of speech. The spectrum provides insight about the bandwidth required to control the chosen gesture-to-speech device (example devices and their position within the spectrum are discussed in chapter 3). A pilot project, called Glove-Talk, used neural networks to implement an adaptive interface translating gestures to words. Based on the promising results of that project, neural networks have been used to implement a more complex adaptive interface: Glove-TalkII, a gesture-to-formant device.

Glove-TalkII uses a Cyberglove, a polhemus sensor and a foot pedal to measure 25 parameters of a user's hand at a 60-100 Hz rate. The adaptive interface maps these parameters to 10 control parameters at 100 Hz which drive a parallel formant synthesizer. Speaking with Glove-TalkII can be likened to playing a musical instrument, except that the user is producing intelligible speech instead of music. The model of the sounds produced is based on articulatory configurations rather than a spatial mapping. Glove-TalkII can therefore be considered as an artificial vocal tract (AVT). How well one masters the device determines the quality of the speech. After only a few hours of practice using Glove-TalkII, most speakers can produce many of the phoneme sounds in the English language, as well as some rudimentary words.

The research described in this dissertation investigates how neural networks can learn to map gesture space to formant space, while making the interface adaptive. Ideally, an adaptive interface should automatically make optimal use of the bandwidth available between the user and the device. Such an adaptive interface would have the following capabilities:

- able to provide a natural mapping between gestures and speech
- able to produce natural-sounding speech (including accent and intonation)

- able to adapt automatically to new users
- able to provide a device which makes learning to speak easy
- unlimited vocabulary

Several different techniques are investigated to implement the adaptive interface for Glove-TalkII. The best of these techniques is given in-depth analysis (discussed in chapter 4).

In evaluating Glove-TalkII, several qualitative and quantitative questions must be addressed:

1. Does the interface capture the constraints of the user's hand?
2. Does the interface adequately capture the user's model of the gesture-to-speech task?
3. Is the mapping invertible (i.e. given some speech utterance is it possible to determine the hand gesture necessary to produce it)?
4. Does the mapping cover the space of speech (i.e. can the user make all the possible sounds for the language he wants to speak)?
5. Is it easy to learn to speak with the interface?

1.1 Structure of Dissertation

This dissertation begins in chapter 2 with a discussion of interfaces and the strategies that make them adaptive. Chapter 2 provides a framework for interface design at an abstract level, thus providing a reference point for comparing Glove-TalkII to other interfaces.

Chapter 3 investigates the gesture-to-speech task along with various approaches that implement gesture-to-speech devices. Chapter 3 also introduces the spectrum of speech mappings and discusses the role of neural network architectures to implement adaptive interfaces between gestures and speech. The history of human controlled talking machines, which begins, remarkably, in the 1700's, along with other attempts to create artificial vocal tracts provides a context for Glove-TalkII.

Chapter 4 is the main chapter in this dissertation. It discusses the current implementation of Glove-TalkII. In the investigation, many different models are explored for use in mapping between gesture and speech. Each model can be considered either as a separate device or an individual instantiation of Glove-TalkII. For the purposes of chapter 4, however, only the *best* model is used as the basis for discussion of Glove-TalkII. The quantitative and qualitative comparison of other techniques considered during the development of Glove-TalkII is left to later chapters in an attempt to provide a clearer view of the issues. A qualitative description of the process of speaking with Glove-TalkII follows in section 4.3. Section 4.3 is especially important as it describes the experiences of a subject learning to speak with Glove-TalkII. The gesture vocabulary learned by the interface is also illustrated. This initial mapping has been learned by all the different network models considered for Glove-TalkII as a basis for comparing them quantitatively. In addition, details of the hardware and software tools used in Glove-TalkII are discussed.

Sections 5.1, 5.2 and 5.3 describe various neural network architectures considered for possible use in implementing the adaptive interface in Glove-TalkII. While the qualitative analysis for each network provides a guide in choosing which network is best, ultimately the "proof is in the pudding". Accordingly, a qualitative analysis of each network must be used to determine which performs best perceptually. Informal listening results used to evaluate speaking performance are therefore included in each of the sections. Section 5.4 discusses obtaining the training and test data sets used for adapting the interface, and for establishing quantitative performance measures.

Chapter 6 discusses various alternative strategies tried while implementing Glove-TalkII. Chapter 6 examines informal comparisons between these alternative strategies and the method currently used in Glove-TalkII. In addition, a technique called the constrained auto-encoder method provides a novel mechanism which automatically creates the gesture-to-speech mapping.

In order to learn to use an adaptive interface, the user must have some way to obtain the inverse mapping. That is, there must be some mechanism which tells her how to manipulate the device to get a desired response. Getting the inverse mapping may be as simple as trial and error, or as complicated as a graphical teacher. In chapter 7, the task of obtaining the inverse mapping of Glove-TalkII is investigated. Depending on the method used to implement Glove-TalkII, the difficulty of getting the inverse mapping may be more or less problematic. These issues are discussed, where appropriate, throughout the dissertation and summarized in chapter 7.

Chapter 8 discusses specific difficulties remaining with Glove-TalkII and provides some possible solutions to enhance it. Finally, general issues that pertain to the implementation of adaptive interfaces are summarized.

Chapter 2

Adaptive Interfaces

An interface is the method by which information is exchanged between a user and a device. The interface is generally chosen by the designer of the device. If the designer chooses a static interface, the user must then adapt his own behaviour to suit the interface. This would not be the case, however, if the interface could adapt automatically. An *adaptive* interface changes its parameters based on experience with the user and so can improve performance.

Figure 2 shows a block diagram of a user's task model and a device's task model (in this case, the device is a computer). Both the user and the computer have three-part task models. The user's model includes a device, a domain, and a communication model. The user's device model is what the *user* thinks the *device* thinks about the task. The user's domain model is what the user *believes* about the task being performed. The user's communication model defines her knowledge of how information is transferred to the computer. To illustrate these, consider the *vi* text editor. In *vi*, a user can enter 'insert mode" by pressing the letter *i* when he is in "command mode". The mode the user thinks he is in is part of the user's *device* model. The knowledge that the letter '*i*' initiates "insert mode" from "command mode" would be part of his *domain* model since it is knowledge about *vi*. If the arrow keys move the cursor in the editor (which normally requires using the keys *j, k, l and h*), they are included in the *communication* model. When a user learns to use a device, all three of the device, the domain, and the communication components adapt.

Similarly, the device has its own model of the task, composed of a user model, a domain model and a communication model. The user model consists of the knowledge the device (e.g. a computer) has about the user and what the user knows. The domain model is everything the computer knows about the task. The communication model informs the device how information transfers between user and device. These concepts can again be illustrated through the example of the *vi* text editor. A computer's user model might include the fact that the user knows about the *i* command but not about the *a* command. The computer's domain model includes all the commands and current modes in the editor. Its communication model would include the gain of the cursor's movement when the arrow key is used. As with a human user, the computer can change its three models to change the way information is transferred, and thereby make an interface adaptive.

The strategies humans use to adapt to their environment may lend insight into computer (or other device) adaptation strategies. Table 2.1 shows human adaptation strategies paired with an analogous computer strategy. An

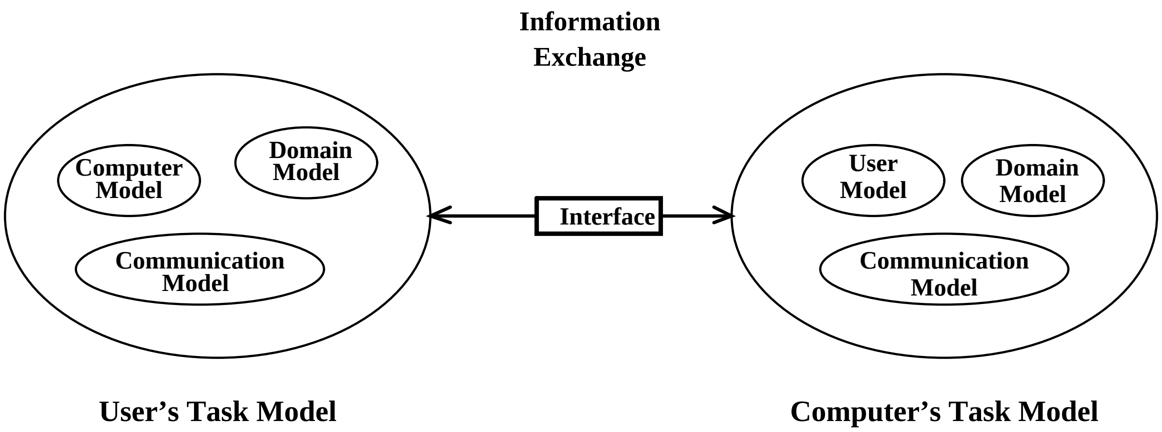


Figure 2.1: Interface: information is transferred between the User and the Device. The User's model of the world has three sub-models, as does the device's model of the world.

People		Computer	
Strategy	Example	Strategy	Example
genetic	natural selection	preprogram or reprogram	setting function keys; manufacturer product upgrades
simple stimulus-response	smell food → salivate	discrimination learning	one-to-one mapping of hand shape-to-words based on training examples
complex stimulus-response (context dependent)	learning different meanings for one word	multiple interpretations of single stimulus	providing different help text depending on expertise of user
rule formation (planning and model formation)	add "s" to make plural	rule formation (to mediate response)	user always needs help text for file manipulation
meta-rules and meta-knowledge (analogy, metaphor, etc.)	time flies like an arrow; fruit flies like a banana	generalization (interpolation, extrapolation, analogy)	user uses mouse in polar coordinates for task A, so use polar for task B

Table 2.1: Human and Computer Adaptation Strategies

Human/Evolutionary	Computer
Natural Selection	Designed systems
Genetic Engineering	Adaptable/tailorable
Tropism/reflexes	Adaptive
Operant conditioning	Self-regulating
Internal evaluation	Self-mediating
Abstraction	Self-modifying

Table 2.2: Taxonomy of Adaptive Systems in [6]

example of each strategy is given in the table. Browne et al. [6] suggests another taxonomy of user adaptation strategies and computer adaptation strategies. Their taxonomy is shown in table 2.2. A simpler view of the above taxonomies is useful. Computer adaptation strategies can be placed on a continuum based on the degree of adaptation. On one end are systems which are non-adaptive. At the other end are systems which require no user-supervision to adapt. In the middle are systems which require user supervision. The continuum is shown in figure 2.2. This continuum does not specifically have analogous counterparts with human adaptation schemes, however, it does provide the interface designer with intuition for selecting techniques for implementing his adaptive interface design once he determines where his system is placed on the continuum. As will become clear, different schemes for implementing Glove-TalkII fall at different locations on this continuum.

Computer adaptation strategies are used to adjust a combination of the computer's three models. Much research remains necessary to identify algorithms which will automatically adjust the computer's model to improve the user's performance. Glove-Talk [11] [12] demonstrated that neural networks are suitable for implementing a discrimination strategy. Potentially, neural networks could be used effectively for all the strategies. Adhoc techniques have been used for multiple interpretations in a system called the UNIX Consultant [7]. Other formal algorithms have been considered for various strategies (see [6]) in different adaptive interfaces.

To use any of the adaptation strategies listed in table 2.1, the user must be characterized. Further, such a characterization must be operationalized. Operationalization should be based on information that comes through

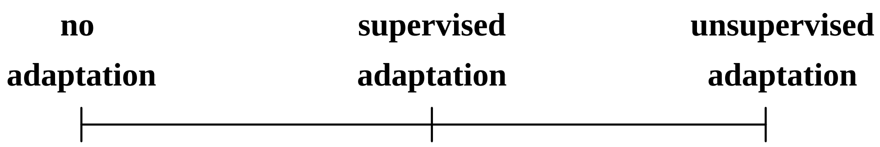


Figure 2.2: Continum of computer adaptation systems

the interface. The information that comes from the user can be broadly classified into four types: instructive, advisory, tutorial and corrective. These types of information can be used to measure user characteristics. Notice also that these are precisely the same types of information users get from the computer in order to adapt to the computer. For example, the Unix consultant [7] defines the user as either expert, intermediate, beginner or novice. The user's level is deduced from the statements she makes to the computer.

To summarize, there are five main points to consider when designing an adaptive interface:

1. characterizing the user
2. measuring the characteristics of the user
3. determining which of the device's models to change
4. ascertaining how to change these models (i.e. which strategy to use)
5. choosing the appropriate algorithm to change the model.

Additionally, obtaining an inverse model for user training may be very important.

In this proposed research, a gesture-to-formant device provides a test-bed for designing an adaptive interface. The five points listed above are important to consider in order to make a working device which is easy to use. In addition, the design process itself will shed light on the pitfalls of adaptive interface systems. In the next chapter, the gesture-to-speech task will be discussed in general. In particular, the gesture-to-formant task is shown to be a "real world" problem which may greatly benefit from a well designed adaptive interface.

Chapter 3

Glove-TalkII and the Spectrum of Gesture–to–Speech Mappings

Converting hand gestures to speech can be accomplished in many ways. Figure 3.1 shows a spectrum based on the granularity of speech each gesture represents. The designer's choice made within this spectrum dictates the bandwidth required from the user. In addition, granularity chosen within the spectrum determines the type of synthesizer, the vocabulary size, speech sound lag plus other features of such a device (see [11] for more details).

On the right is a coarse scheme where each gesture represents a word. By gesturing, a string of isolated words is sent to a text-to-speech synthesizer. In order for the speech to sound fluent the user has to make gestures within the time it takes for the previous word to be spoken. For a speaking rate of 100 words per minute the user has about 600 msec per gesture. In addition, the vocabulary is necessarily fixed. One advantage of this system is that the user can easily learn small vocabularies, since she does not have to produce gestures at a high rate, and there are only a small, fixed number of gestures to be memorized. However, as the vocabulary increases, learning to use such an interface gets harder. Therefore, an adaptive interface becomes a helpful addition to this device. Glove-Talk [11] implements an adaptive interface that does gesture-to-word classification. Notice that the spectrum continues from left to right, past the word-level system. Possibilities off to the right side include American Sign Language (ASL) recognition systems and gesture-to-phrase mappings.

Moving left across the spectrum, there is a decrease in the amount of time the user has to make a gesture that produces fluent-sounding speech. For example, with finger spelling (where gestures are mapped to letters), the user has to spell out each word. Of course, the user has an unlimited vocabulary, but, the time required to say a word is proportional to its length. Speakers of ASL can conveniently use the device if gestures are mapped to the ASL alphabet. It is less obvious that an adaptive interface would be useful for the gesture-to-letter mapping, since only a small set of gestures would need to be recognized. However, due to the speed required to finger spell fluent speech, an adaptive interface may offer improved performance. Examples of finger spelling devices with adaptive interfaces include [21] and [10]. A related scheme maps gestures to phonemes, where words are now phonetically spelled out. Three main differences exist between finger spelling and phoneme generation. First, phoneme spelling requires a larger number of gestures than finger spelling (there are approximately 40 phoneme). Second, with finger spelling, a word cannot be spoken until the last letter has been gestured[1]. Third, finger spelling is well-defined in ASL; however,

[1] Conversely, with phoneme spelling, the sounds can usually be made while the phonemes are being gestured, except in cases where a successive phoneme is required (i.e. after a burst phoneme a vowel or a consonant sound is required). In exceptional cases, a lag of one

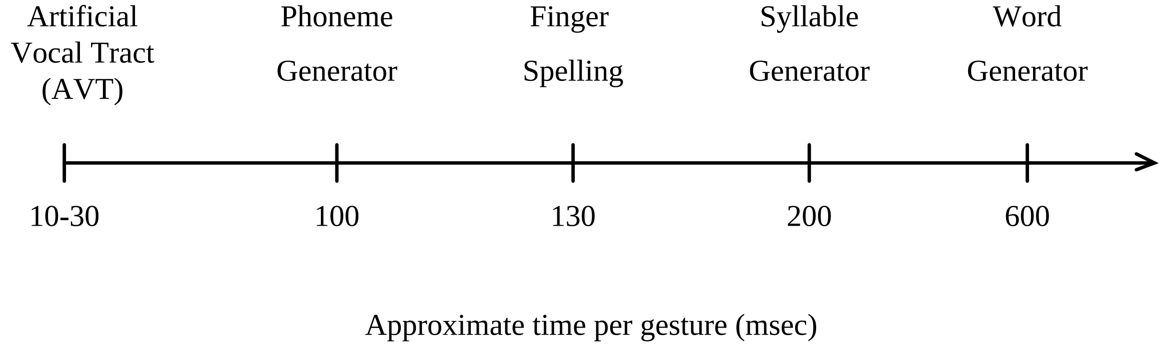


Figure 3.1: Spectrum of gesture-to-speech mappings based on granularity of speech

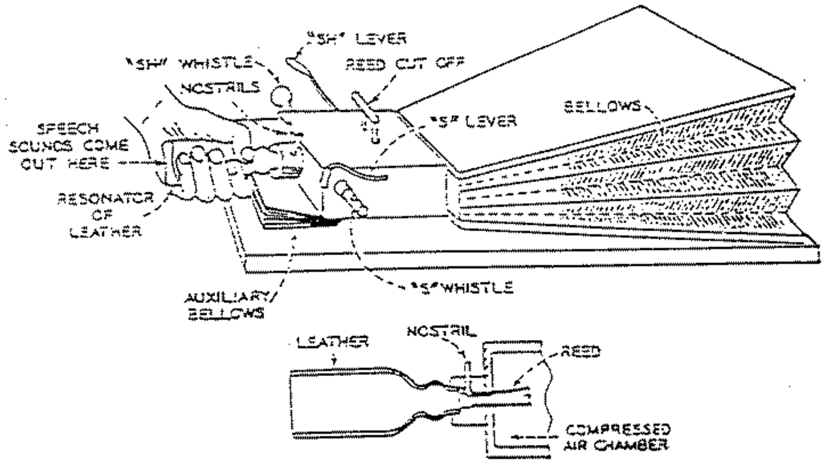


Figure 3.2: Diagram of von Kempelen's gesture-to-speech device - taken from [15].

phoneme spelling has no such standard, so users would have to learn a new gesture language.

At the far left of the spectrum, gestures are used to control an artificial vocal tract (AVT). The AVT does not necessarily have to contain the actual articulators, but could hold synthesizer parameters that change at about the rate articulators do, such as formant parameters or linear predictive code (LPC) coefficients. Perhaps the best analogy for this type of device appears as a musical instrument. Imagine that a musical chord is a vector of speech parameters and that the user controls the device to make speech instead of music. Such a device has an unlimited vocabulary. The user must gesture at the same rate as the underlying parameters move in order to speak. An interesting feature of the AVT device is that the mapping between gestures and speech is a topological mapping between gesture space and speech space. This contrasts with the previously mentioned devices which require discrete classifications. Numerous attempts have been made to create an AVT device. In 1791, Wolfgang von Kempelen[2] built an AVT. Figure 3.2 shows a schematic of a reconstruction by Wheatstone of von Kempelen's device. Bellows were used to blow air over a reed which excited the hand-varied leather resonator. The user's other hand controlled four separate constricted passages.

In the late 1800's, Alexander Graham Bell and his brother Melville Bell built a more biologically motivated speaking device.

> Following their father's advice, the boys attempted to copy the vocal organs by making a cast from a human skull and molding the vocal parts in guttarpercha. The lips, tongue, palate, teeth, pharynx, and velum were represented. The lips were a framework of wire, covered with rubber which had been stuffed with cotton batting. Rubber cheeks enclosed the mouth cavity, and the tongue was simulated by wooden sections – likewise covered by a rubber skin and stuffed with batting. The parts were actuated by levers controlled from a keyboard. A larynx 'box' was constructed of tin and had a flexible tube for a windpipe. A vocal-cord orifice was made by stretching a slotted rubber sheet over tin supports.
>
> Bell says the device could be made to say vowels and nasals and could be manipulated to produce a few simple utterances (apparently well enough to attract the neighbours)[3].

The next attempt appeared at the World's Fairs in 1939 (New York) and 1940 (San Francisco). Figure 3.3 shows a schematic of the system called Voder (Voice Operation Demonstrator) [4].

> The 'resonance control' box of the device contains ten contiguous bandpass filters which span the speech-frequency range and are connected in parallel. All the frequency filters receive excitation from either the noise source or the buzz (relaxation) oscillator. The wrist bar selects the excitation source, and a foot pedal controls the pitch of the buzz oscillator. The outputs of the bandpass filters pass through potentiometer gain controls and are added. Ten finger keys operate the potentiometers. Three additional keys provide a transient excitation of selected filters to simulate stop-consonant sounds.
>
> ... Although the training required was quite long (on the order of a year or more) the operators were able to 'play' the machines – literally as though they were organs or pianos – and to produce intelligible speech.

gesture will occur before the speech is heard.

[2] Figure 3.2 from [15] and description taken from [4].

[3] taken from [15], original reference, Bell, A.G., 1922.

[4] Figure 3.3 from [15] and text from [4]; original reference [9].

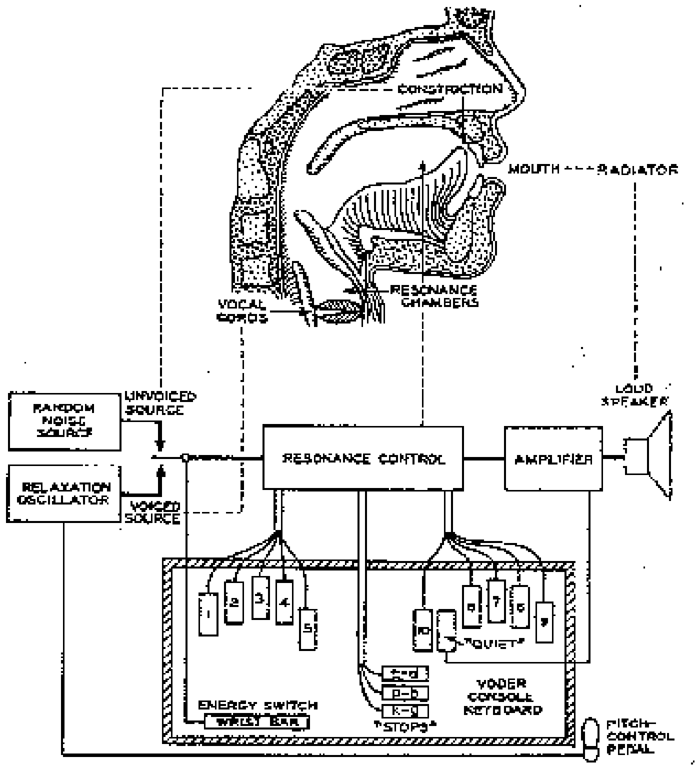


Figure 3.3: Block Diagram of the Voder – taken from [15].

One observer described the Voder-produced speech as "robotic, monotone and slow but still intelligible". More recently, another less ambitious attempt has been made to map gestures to speech at this fine-grained level. Pausch et al. and Girson et al. [27] [16] have developed a device intended for children with cerebral palsy. Their device uses a joystick and an articulatory representation of the cascade formant synthesizer [20]. The joystick controls the position of the tip and back of the tongue parameters. The values of these parameters are then sent to the synthesizer to produce speech. "The current implementation can be used to produce crude, monotone speech by using a joystick to navigate the tongue position grid".[5] It was very difficult for the user to learn to speak with their system, which demonstrated the need for an adaptive mapping between joystick and articulatory parameters. Each of the above devices specifies one way to perform the mapping between gestures and speech sounds. It is not clear which one is best, since each device either restricts the variety of speech or is difficult to learn. Ideally, an AVT system would compromise and allow the user to speak any words she wants, *and* be easy to learn. Finding an optimal trade-off is difficult. An adaptive interface attempts to provide a better trade-off than a hand-designed interface would. Adapting to the user maximizes the bandwidth between the user and the device, allowing the speech space to be covered completely. At the same time, the user's cognitive model of the device and constraints on possible gestures can be captured and used to determine a mapping. As a result, a user's learning time will be minimized without compromising speech quality.

Glove-TalkII, an AVT developed for this dissertation, translates gestures measured by a Cyberglove, polhemus device and a foot pedal, to parameters driving a formant synthesizer. The adaptive interface between them acts according to a mapping learned from the user. It is hoped that the constraints of each individual user's hand together with the influence of the user's cognitive model of the task, are captured in the interface to make learning easy while not compromising speech quality and vocabulary size. Glove-TalkII should demonstrate the possibility of finding a suitable "trade-off" through an adaptive interface. Neural network technology provides an excellent means by which to meet the needs of such an interface.

The next chapter describes the final Glove-TalkII system. The final system uses the best of the architectures that were empirically assessed.

[5] See [27].

Chapter 4

The Final Glove-TalkII System

4.1 Overview

The final[1] Glove-TalkII system converts hand gestures to speech, based on a gesture-to-formant[2] model. The gesture vocabulary is based on a vocal-articulator model of the hand. As discussed in chapter 3.1, at this mapping level the user must move his hand relatively quickly to produce continuous intelligible speech. At high speed, it is hard to exactly achieve target configurations so most of the system utilizes an adaptive interface between the user and the speech synthesizer. Neural networks implement the adaptive interface. To run fast enough on a standard workstation, the networks must be small. By dividing the mapping tasks into independent subtasks, a substantial reduction in network size and training time is possible (see Glove-Talk [11]).

Three error backpropagation networks, a preprocessor and a postprocessor implement the interface which converts gestures to control parameters of a parallel formant speech synthesizer. Two of the neural networks train on data obtained from the user, thereby making the interface adaptive. Some aspects of the mapping are fixed; a block diagram of the system is shown in figure 4.1.

Important features of this system include the three neural networks labeled vowel/consonant decision (V/C), vowel, and consonant. The V/C network is trained to decide whether the user wants to produce a vowel or a consonant sound. The vowel network is trained to convert the current gesture to the appropriate vowel sound. Likewise, the consonant network is trained to produce consonant sounds. Nine keys on the keyboard designate nine stop consonants (B, D, G, J, P, T, K, CH, NG), because the dynamics of such sounds proved too fast to be controlled by the user using a one-to-one mapping scheme. The foot pedal provides a volume control by adjusting the speech amplitude. The mapping is fixed. The fundamental frequency, which is related to the pitch of the speech, is determined by a fixed mapping from the user's hand height. The output of the system drives a parallel formant speech synthesizer.

After training, Glove-TalkII is used as follows: to initiate speech, the user forms the hand shape of the first sound she intends to produce. She depresses the foot pedal and the speech comes out of the synthesizer. By co-ordinating hand motions, vowels and consonants of various qualities are produced in a continuous fashion. Words are formed by making the correct motions; for example, to say "hello" the user forms the "h" sound, depresses the foot pedal and quickly moves her hand to produce the "e" sound, then the "l" sound and finally the "o" sound. The user has complete control of the timing and quality of the individual sounds. The articulatory mapping between gestures and speech is decided apriori. Even though the designer may instruct the user on what mapping to use, the adaptive interface learns the *user's* interpretation of the mapping.

In the next section, the initial mapping between hand gestures and speech is discussed in detail. First though, an introduction to some basic concepts of speech is provided to help the reader understand the networks' learning task.

The goal of the Glove-TalkII system is to allow users to speak intelligibly. Quantifying Glove-TalkII's success is difficult for several reasons:

- intelligibility is subjective and listeners adapt to peculiarities of speech.
- the speech quality improves with practice
- minimal speech intelligibility requires much practice

[1] In this chapter, only the best system developed thus far is discussed and described as Glove-TalkII. Refer to subsequent chapters for details of different techniques which were considered.

[2] The term formant parameters will be used to indicate all the control parameters of the parallel formant speech synthesizer; though technically, not all the control parameters are related to formants i.e. voicing and fundamental frequency. The context of the use of the term formant should make its use clear.

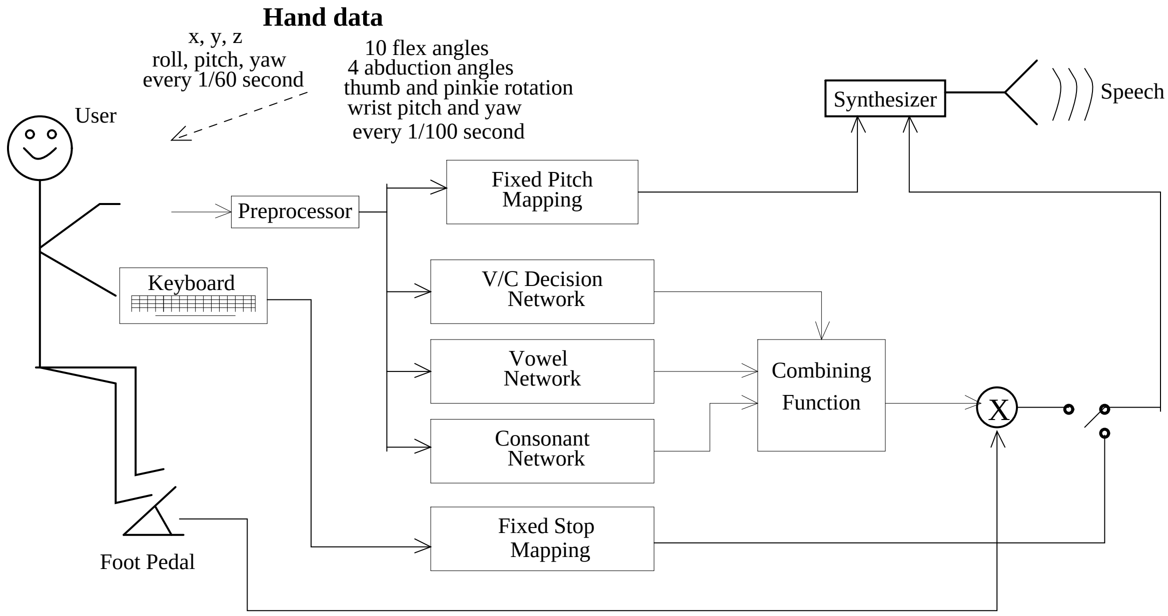


Figure 4.1: Block diagram of Glove-TalkII: input from the user is measured by the Cyberglove, polhemus, keyboard and foot pedal, then mapped using neural networks and fixed functions to formant parameters which drive the parallel formant synthesizer [31].

For the above reasons, the main tool for determining Glove-TalkII success is a subjective account of Glove-TalkII speech. Section 4.3 provides a qualitative assessment of the best speech quality produced by a single subject.

Quantitative comparisons between different versions of the networks within Glove-TalkII can be performed. Even though the perceptual performance cannot be guaranteed from quantitative analysis, the poorest networks can be discarded based upon such analysis. Sections 5.1, 5.2 and 5.3 discuss quantitative analyses of different architectures for the V/C, vowel and consonant networks respectively, all considered in implementing the adaptive interface. From this analysis, the best networks are evaluated qualitatively to produce the final Glove-TalkII system. Chapter 5 also describes how the networks are trained. These sections discuss the implementation details of each of the network blocks within figure 4.1.

The rest of chapter 4 provides implementation details of the final system without the exact network structures. These details include the specifics of hardware, software, and non-neural network additions which make the system work.

4.2 Glove-TalkII Vocabulary Model - an Artificial Vocal Tract

The current implementation of Glove-TalkII learns the user's interpretation of a hand-designed gesture-to-formant mapping. The user's interpretation of the initial mapping is captured during training sessions. These training sessions are described in section 5.4. The initial gesture-to-formant mapping is based on an articulatory model of speech sounds in English. For this reason, before the mapping is described, a brief introduction to articulatory phonetics and speech synthesis provides the reader with insight into the decisions made for the mapping.

4.2.1 An Introduction to Articulatory Speech and Synthesis

In this version, Glove-TalkII is intended for English speakers[3]. Thus, the initial mapping must be able to map hand gestures to all the different sounds in English, called phonemes[4]. Combinations of these sounds together form words and sentences. Figures 4.2 and 4.3 show the symbols for the phonemes of English. In fact, our vocal apparatus can make about 100 different phonemes[5]. These have been categorized and studied extensively and the reader is encouraged to refer to [22] for a more detailed study of this field. For our purposes, it is only important to be able

[3]It is a simple matter to adjust the final version of Glove-TalkII for any language (see section 5.5 for discussion of necessary extension).

[4]Phonemes are indicated by enclosing them in //, i.e. the "ee" sound in the word "heed" is represented by the phoneme /i/. Throughout the text, phonemes that are used by the text-to-speech synthesizer are distinguished by capital letters, i.e. the phoneme /i/ is represented by EE.

[5]Each phoneme can have many variations called allophones.

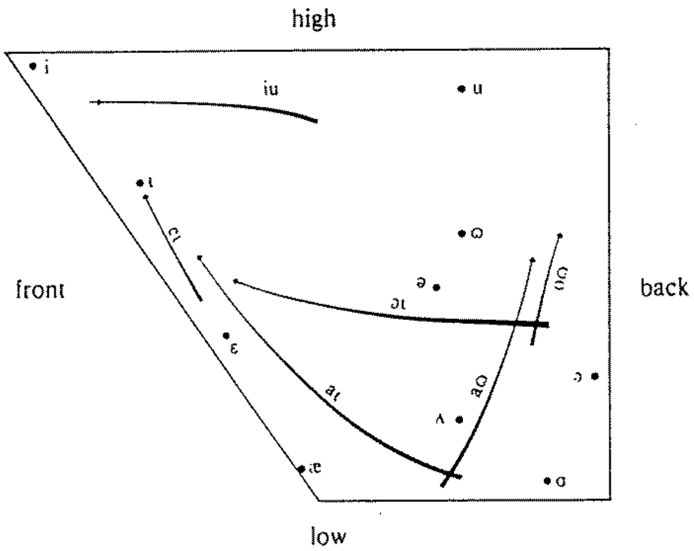


Figure 4.2: Relative auditory qualities of the vowels as a function of tongue positions. Notice the dipthongs are trajectories through this space. Also, the axis on this graph are a mirror image with 180 degrees rotation of the initial vowel mapping used in Glove-TalkII. Figure taken from [22]. The symbols in the figure represent phonemes and should be thought of as enclosed in //.

to make all the sounds used in English; which has about 40 phonemes. These can be categorized roughly on an articulatory basis.

Consider a mid-saggital view of a typical vocal tract. Air is pushed through the vocal cords. If they are pulled tight they begin to vibrate at some frequency producing voiced speech [6]. If they are not pulled tight a hiss-like sound is produced - called unvoiced speech. To illustrate the difference, feel your throat when saying the long vowel "e" (voiced) and compare that with the sound "sh" (unvoiced). The sound produced at the vocal cords travels up into the vocal tract. The positions and movements of the articulators within the tract then shape the sound to produce phonemes. This view of speech is commonly called the source-tract model [22].

Vowels constitute a common phonological distinction. Vowels are created when the flow of air is unobstructed through the vocal tract. All vowel sounds are voiced, i.e. vocal cords vibrate during vowel production. The different vowel sounds correspond roughly to the positions of the body of the tongue[7]. As illustrated in figure 4.2, when the tongue is high in the mouth and towards the front, the sound /ee/ is produced as in the word "heed". Other configurations produce other vowel sounds. Importantly for Glove-TalkII, vowels can be produced by continuous variations along two dimensions: tongue height and tongue back. Further, not all vowel sounds are static configurations of the articulators, rather some require coordinated movements between two static positions. These are called dipthongs. An example of a dipthong is the phoneme /ai/ which is heard in the word "hi". To produce this sound the speaker moves his tongue from the low-back position to a high-front position. Remember that these articulatory descriptions are only approximate; the target configuration does not need to be achieved precisely for the listener to hear particular phonemes.

Obstructions in the vocal tract create another sound source, categorized as consonants. Within the consonant grouping, a further distinction is made by the type of obstruction. The type of obstruction is classified with two parameters: the place of articulation (where the main constriction is) and the manner of articulation (how much constriction exists). If the vocal tract is completely closed off, the sound is called a stop consonant; for example the sounds /b/, /d/, /g/, /p/, /t/, /k/, /ch/, /j/ and /ng/ are stop consonants. The particular sound produced depends upon where the closure is. For instance, if the lips are closed and a voiced sound is pushed through the lips a /b/ sound is produced. If the obstruction causes enough turbulence to make a hissing type sound the result is a fricative as in /f/, /v/, /th/, /dh/, /z/, /s/, /sh/, and /zh/. Again, depending upon where the constriction takes place a different phoneme will be produced; for example, /f/ is produced when an unvoiced sound encounters a constriction between the lower lips and the upper teeth. If the oral cavity closes while the nasal passage is left open, the sound produced becomes a nasal consonant as in /m/, /n/. If the constriction does not cause a sound source but is still voiced then the sound will be somewhere between a vowel and a consonant called an approximant: i.e., /l/, /r/, /w/ and /y/. A special category, called aspirant, occurs when the air flow is unvoiced and unobstructed by

[6] The frequency of the vocal cord vibrations is called the fundamental frequency which is related to the pitch of the speech.

[7] Lip rounding also plays some role.

Place of articulation

Manner of articulation	bilabial	labio-dental	dental	alveolar	palato-alveolar	palatal	velar
nasal (stop)	m			n			ŋ
stop	p b			t d			k g
fricative		f v	θ ð	s z	ʃ ʒ		
(central) approximant	(w)			ɹ		j	w
lateral (approximant)				l			

Figure 4.3: Voicing, Manner and Place of Articulation as Primary Parameters of American English Phonemes - taken from [22]. The symbols in the figure represent phonemes and should be thought of as enclosed in //.

the vocal tract. The phoneme /h/ is the only example of an aspirant used in English. For a designer of a mapping between hand gestures and phoneme sounds, the important characteristic to exploit becomes the roughly continuous change in both the manner and place of articulation for the various phonemes. Further, each phoneme is associated with a target set of *formants*, described in section 4.4 and appendix C. Figure 4.3 shows the relationship between consonant sounds and their manner and place of articulation. Notice the place of articulation in this table starts on the right-hand side with the highest point in the vocal tract (labial) going all the way to the beginning of the vocal tract at the glottis. Loosely, the manner of articulation begins at the top with a full obstruction to gradually no obstruction – the vowel sounds. The reason for putting nasals before stop consonants is that nasal closures are held for longer periods of time (and usually tighter) than the equivalent stop consonants (test this for yourself by saying /d//ee/ and compare that with /n//ee/.

4.2.2 The Initial Mapping – Glove-TalkII Gesture Vocabulary

The gesture language was designed according to the analysis of English described above. The main idea was to use continuous changes in finger positions to correspond roughly to the changes in manner and place of articulation as well as the degree of voicing. To this effect, the mapping which is based on the vocal tract analogy is as follows:

- For vowel sounds, the hand is placed in an open configuration as illustrated in table 4.1. The main articulators are the tongue position (front↔back) and tongue height (high↔low). The equivalent hand parameters are approximately the X and Y positions of the hand in the horizontal plane. The exact mapping is shown in figure 4.4. The exact mapping has been adjusted to correspond to the changes in formant frequencies F1 and F2 which corresponds roughly to the position and height of the tongue. Remember, to produce dipthongs, the user is required to make a trajectory through the vowel space. To make an IE[8] sound for instance, the user's hand must start below and to the right of the A sound, which is in the middle of the mapping and sweep to the user's left and away from her, towards the EE sound. Some of the common dipthongs are illustrated with arrows on the mapping. The phonemes are labeled with the corresponding text-to-speech phonemes because this mapping trains the vowel network (see section 5.2 and 5.4).

- An obstruction of the vocal tract is created by moving either the index or middle finger towards the thumb. Obstructions in the vocal tract produce consonants; thus, the closure of the hand produces the fricatives, approximants and nasal sounds. The exact mapping is fairly complex, as illustrated in table 4.1.

- The ring finger determines the degree of voicing. A bent ring finger produces an unvoiced source, while an unbent ring finger produces a voiced sound. This mapping coincides with the view that an open hand configuration produces vowels which are voiced. If the ring finger is held in opposition to the thumb, it produces a purely unvoiced sound with no obstructions. The aspirant H is the sound produced.

[8] Phonemes used in Glove-TalkII are derived from a text-to-speech synthesizer and are identified using capital letters to distinguish them from IPA phonemes indicated by //.

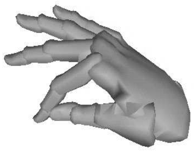
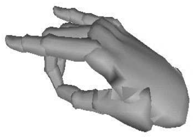
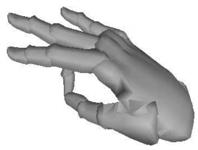
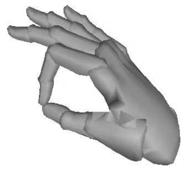
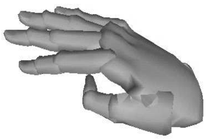
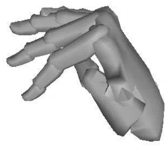
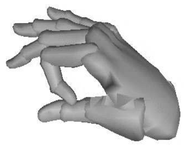
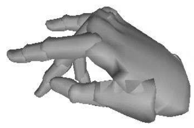
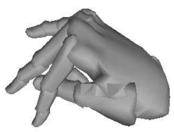
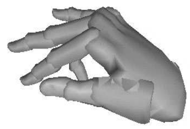
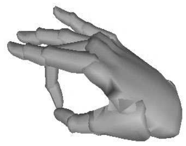
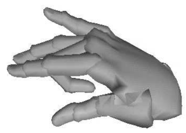
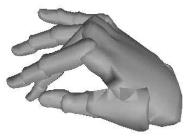
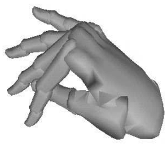
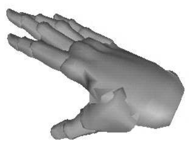
 DH	 F	 H	 L
 M	 N	 R	 S
 SH	 TH	 V	 W
 Z	 ZH		 vowel

Table 4.1: Static Gesture-to-Consonant Mapping for all phonemes. Note, each gesture corresponds to a static *non-stop* consonant phoneme generated by the text-to-speech synthesizer.

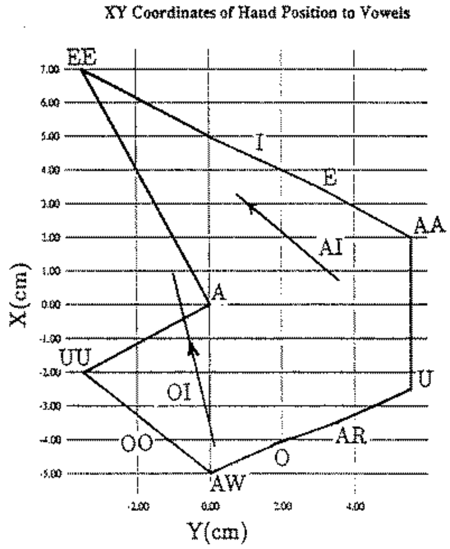


Figure 4.4: Hand-position to Vowel Sound Mapping. The coordinates are specified relative to the origin at the sound A. The position of A relative to the polhemus source (see section 4.4) is specified separately. The X and Y coordinates form a horizontal plane parallel to the floor when the user's is sitting.

Key	Phoneme	Key	Phoneme
A	B	Z	P
S	D	X	T
D	G	C	K
F	J	V	CH
G	NG		

Table 4.2: Keyboard-to-stop Consonant Mapping

- Nasalization is indicated by bending the thumb's interphalangeal joint (IP). The place of articulation is indicated by the amount of thumb circumduction (i.e. how much the thumb is moved into an opposing position).
- With a one-to-one mapping, stop consonants require a very fast, controlled movement of the articulators. This movement is difficult to accomplish. An obvious mapping using contact between thumb and finger to indicate a complete closure is difficult to achieve since the Cyberglove hardware does not detect contact *versus* near contact. Instead, the keyboard is used to produce stop consonants, since the quality of stop consonants is relatively regular and the necessary articulator motion resembles the press of a button. Table 4.2 shows the mapping. Notice that the place of articulation maps horizontally and the voiced/unvoiced distinction maps vertically on the QWERTY keyboard. For each stop consonant, except NG, a keystroke produces a fixed sound with a maximum duration. If the user holds the key down, the stop consonant's formant parameters are linearly interpolated to the formant parameters which are determined by the user's right hand in 5 time steps. For NG though, duration of the keystroke corresponds exactly to the duration of sound produced.
- The height (along scaled z axis) of the hand determines the fundamental frequency of the speech (related to its pitch) according to:

$$F0 = 30(-z + 1.0) + 18 \tag{4.1}$$

F0 is bounded in the integral range [0,63] corresponding to the available parameter values of the speech synthesizer. This range corresponds to a fundamental frequency range of 25 to 417 Hz.

- The depression of the foot pedal determines the overall volume of the speech according to:

$$volume = 2.5 * (footpedal - 0.5) \tag{4.2}$$

The volume is a bounded real value between 0.0 and 1.0 where 0.0 represents complete silence and 1.0 is full volume. The variable $footpedal$ is a real value between 0.0 and 1.0 that is a function of the depression angle of the pedal (0.0 full up, 1.0 = full depression). According to this function, the speech output is silent for about the first half of the pedal's depression, with a quick rise to full volume once the user passes the halfway mark. The volume control also affects the degree of voicing (V). Voicing decreases when volume decreases; for low volumes speech is completely unvoiced. This mimics the rule used in the text-to-speech synthesizer.

The forgoing describes the articulatory mapping between hand-gestures and speech that the user sees. The role of the neural networks is to learn how the user interprets the mapping. Specifically, the user is responsible for attempting to make the gestures for all the sounds of English, and the neural networks must learn to interpret what he does. For some of the sounds and acoustic events the mapping is pre-programmed and does not adapt.

4.3 Qualitative Performance of Glove-TalkII

One subject has been trained extensively to speak with Glove-TalkII. The subject is an accomplished pianist. It was anticipated that his skill in forming finger patterns for playing the piano and his musical training would transfer positively to aid his learning to speak with Glove-TalkII. Currently, after 100 hours of training with Glove-TalkII, his speech is intelligible and somewhat natural-sounding. He still finds it difficult to speak quickly, pronounce polysyllabic words and speak spontaneously. This section describes the process he underwent to learn to speak with Glove-TalkII.

The subject went through 8 learning phases during speech acquisition. These phases are somewhat analogous to stages encountered while learning to play a musical instrument. The phases are:

1. user set-up
2. initial network training
3. individual phoneme formation within simple words and CV/VC pairings
4. word formation and interword pitch control
5. short segment formation with suprasegmental pitch control and singing
6. passage reading
7. fine tuning; movement control and phrasing
8. spontaneous speech

During his training, Glove-TalkII also adapted to incorporate changes required by the subject. Of course, his progression through the stages is not as linear as suggested by the above list. Some aspects of speaking were more difficult than others, so a substantial amount of mixing of the different levels occurred. Practice at the higher levels facilitated perfecting more difficult sounds that were still being practiced at the lower levels. Also, the stages are iterative, that is, at regular intervals the subject returns to lower levels to further refine his speech.

The first phase was initializing the system and familiarizing the subject with the system. The subject's hand parameters were calibrated [9] using the graphical hand and the displayed finger angles (see section 4.5). A scale file was created from recorded hand data (see section 4.5 and appendix B for a typical example of a scale file). The subject was familiarized with putting on the glove and setting up Glove-TalkII.

The second phase was initial network training. The subject familiarized himself with the initial mapping (see section 4.2.2). A complete set of training data was collected from the subject using the procedure described in section 5.4. The first training set consisted of 2830 examples of static consonants for training the consonant network and 3502 examples to train the V/C network. The subject moved his hand according to the initial mapping configuration described in section 4.2.2. These data were used to train Glove-TalkII to map the subject's interpretation of the initial mapping. During data collection, the subject memorized the static hand configuration to static consonant mapping. In addition, he provided hand configurations most suited to his hand that approximated the initial mapping. The simplicity of the data collection procedure and the ease with which the networks train are important for Glove-TalkII to be a useful adaptive interface.

Initially, Glove-TalkII used a V/C network which was designed for novice users. This V/C network (described fully in section 5.1) uses an averaging scheme to reduce the effects of noise associated with a novice user's hand parameters. Normalized radial basis function were used in the consonant and vowel networks (see chapter 5 for specifics of each network).

The third phase involved the subject learning to say individual sounds within consonant–vowel pairs (CV), vowel–consonant pairs (VC), and simple word contexts reliably. This was the first time the subject had spoken with the system. The speech produced was unintelligible at this time; however, individual sounds were recognizable. Most

[9] Due to the robustness of the Cyberglove sensors, calibration is performed infrequently.

importantly, the subject began to recognize different types of phoneme sounds within the sounds he produced. Phoneme recognition helps provide audio feedback for the subject to adapt his motions to produce desired effects. Much of the subject's practice time was spent repeating each of the consonant and vowels sounds, for example "we", "el", "I". After several hours of practice the subject determined which sounds were most difficult to produce. For these difficult sounds, more training data were collected and the networks retrained with the new data added to the original data. This process was repeated several times in an attempt to improve the consonant mapping. As the subject became more proficient at producing static hand configurations (about 5 more hours of practicing) it became clear that some of the difficulty producing individual sounds was that often a mixture of vowels and consonants was being produced. This effect was caused by the V/C network having an output slightly larger than zero for consonant sounds causing the output of the vowel network to mix with the consonant sound. Vowel data and consonant data were collected to retrain the V/C network (the consonant network was also retrained with the new data). The subject found this version of the V/C network enabled him to speak individual sounds very reliably. The only phoneme not completely satisfactory was ZH (as in *pleasure*) which rarely occurs in English.

The subject proceeded to phase 4: word formation and word pitch control. The subject spent time practicing common English words (from [8]). This exercise provided practice forming transitions the subject would be likely to encounter during conversational speech. While practicing these words some of the difficult transitions became evident; for example, the N→L transition in the word "only". For difficult transitions such as this one, the inverse mapping (section 7) was used to assist the subject in finding the correct hand gesture timing to make the word intelligible. One of the phonemes the subject found difficult to say was the R sound in words like "are". Using the inverse mapping and a pseudo-spectrum of the R sound it was observed that this sound is actually a vowel sound with dropping pitch. Thus, the subject experimented with the effects of pitch control on individual words to make them intelligible.

Stages 3 and 4 are less distinct than suggested above. The subject practiced individual words and individual sounds simultaneously. This amalgamation became particularly prominant as the subject became more proficient with individual sounds. Data were collected for improving the consonant sounds during the many hours of practice during phases 3 and 4.

Glove-TalkII was retrained about 10 times during these initial phases, sometimes with more data for particular phonemes and other times with replacement data. For future subjects, good performance of the V/C network must be a key focus in the early stages of learning. Several retraining sessions were probably unnecessary since the phoneme errors were caused by mixtures of vowels and consonants caused by poor vowel/consonant distinctions.

As an interesting aside, the second version of the V/C network still used the time-shifted inputs [10] resulting in occasional stuttering due to slow processing. The single time-step V/C network, trained on the same V/C data, exhibited poor performance at this stage of learning.

Three more significant adjustments were made after the V/C network was performing properly. First, the I position on the vowel mapping was shifted to (5,0) which is midway between EE and E[11]. This modification was necessary because the subject had difficulty saying the I phoneme as in "is" reliably. This phoneme occurs frequently in English causing significant intelligibility problems. This problem was probably due to the I and E sound being placed relatively close to each other on the initial mapping, correspondingly, after the vowel network was trained, the area in the X-Y plane which produces the I sound was relatively small. Second, the subject created another complete training set for every static phoneme sound once the V/C network performed well. The consonant network was trained with this new data set plus the data set used to train the good V/C network. Third, the entire vowel space was compressed by a factor of 0.75 since the subject found that he had to move his hand extensively in X,Y plane to speak. A factor of 0.5 was also tried but was found extreme. Another interesting attempt to provide a better vowel space was to form a radial representation of the static phonemes. Using A as the centre, the remaining 10 vowel phonemes were placed at equidistant positions along a ring 5 cm away from the A. Training data were generated by partitioning the plane into sectors formed by the mid-points between phonemes on the ring, and by specifying phoneme targets for each of the sectors sampled evenly with 60 points out to a radius of 10 cm. Figure 4.5 illustrates the radial vowel space. The subject found that after 15 hours of training on the original vowel space the new vowel space was too different to integrate into his speech quickly. In comparison, shifting the I phoneme was easily integrated. From this observation, it appears that users can adapt relatively quickly to the first mapping, after which it becomes difficult to alter the mapping radically without significant performance penalties.

At this point, Glove-TalkII was relatively stable allowing the subject to produce static phonemes in sequence reliably. The subject could intelligibly say simple words that had been practised. He was proficient at manipulating pitch within a word as well as getting difficult phoneme transitions, especially stop to vowel or stop to non-stop consonants.

The next phase of the subject's learning involved saying short segments. He would combine practiced words into

[10] See section 5.1 for details of the V/C network structures.

[11] See figure 4.4.

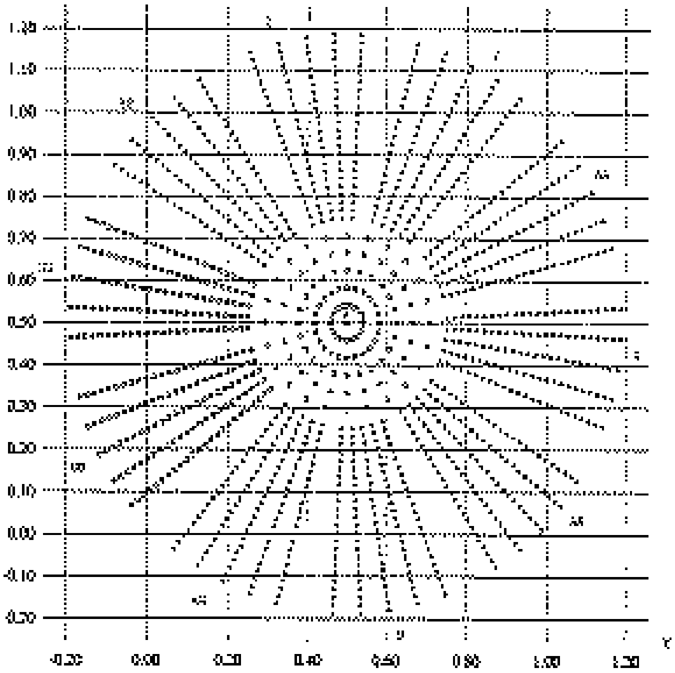


Figure 4.5: Radial vowel space representation. The A sound forms the centre circle. Targets for each of the 10 other cardinal vowels are distributed equally around the circle. Equal sized sectors extending from 5 cm to 10 cm constitute each cardinal vowel target. The cardinal vowel sectors are indicated by the groups of lines projecting away from the centre. The X and Y values are scaled.

meaningful utterances like "Hello, how are you?", the alphabet song ("a-b-c-d-e-f-g... now, I've said my abc, next time won't you sing with me?"), and excerpts from [34] (i.e. "Sam I am." and "I do not like green eggs and ham."). Much practice was required to get the word transitions correct so that they were intelligible. Further, pitch control was practiced over the whole segment to further improve intelligibility. His proficiency with pitch control was such that his version of the alphabet song was actually *sung*. By the end of this phase, the subject could say any simple utterance (1 to 2 syllables) intelligibly after only a few attempts. However, individual words were said fairly slowly at 2.5 to 3 times slower than normal speaking rate[12]. Pronunciation difficulties were mostly in the following areas:

1. producing correct vowel phonemes reliably
2. producing stop consonant clusters
3. timing the R sound precisely

First, it is very important for vowel phonemes to sound correct to achieve proper enunciation of slow speech. With Glove-TalkII, it is difficult to know exactly which vowel will be produced until the foot pedal is depressed since there is poor absolute hand position feedback. Second, timing stop consonants is difficult since the stop phonemes are produced within 10 msec. Small timing errors produce unintelligible stop sounds. Third, if the R sound is sustained for too long, the speech produced sounds muffled and its intelligibility is impaired. Four milliseconds should be a typical duration of the R sound, but this short timing is difficult to achieve since the static gesture required is hard to produce quickly (see figure 4.6). Notice that when making the R sound, the index finger is very bent. To extend the finger requires a fairly large motion which must be made quickly to achieve the necessary transition. One technique to achieve the necessary transition speed is to form the R sound partially instead of completing the finger trajectory. This technique requires a large degree of finger control since the subject's index finger does not oppose the thumb in this case. Another alternative for *some* R sounds is to use one of the R sounding vowel sounds with a drop in pitch as in "ar" in the British pronunciation of "farther". Examples of R's that can be made in this fashion include UR, AR and ER as in "curious", "are", and "curd" respectively. This type of R sound is much easier to produce quickly. The difficulty for the subject is learning to know automatically which way to make the R sound. The subject uses a combination of a pitch drop and a short R burst as a safe alternative for unknown R contexts.

[12] Normal rate is defined by the text-to-speech synthesizer rate for unrestricted text.

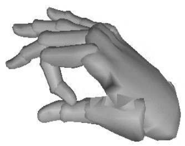


Figure 4.6: Example of hand configuration for the phoneme R.

The subject reached the sixth stage next; learning to read lengthy passages. Some of the passages include [34], [28], and the "Little Miss Muffet" nursery rhyme. As the subject progressed through the stages, the length of speaking time without excess fatigue increased. In the first few stages, one hour of continuous practice was exhausting. By the reading stage he was comfortable enough that 1-2 hours of continuous practice was possible. Reading exercises helped improve intonation control. For example, the children's story "Green Eggs and Ham" [34] has two voices with different intonations i.e.

Would you like them in a house?
Would you like them with a mouse?
I would not like them in a house.
I would not like them with a mouse.

In addition, reading caused improvement in the three most difficult areas for producing intelligible speech: reliably producing vowel sounds, stop consonant clusters and R technique.

Several distinguishing features of the subject's speech were observed in informal listening tests. First, a strong contextual effect occurred. In particular, when a listener hears the subject speak for the first time, she sometimes does not understand a single word; rather perceives a long slurred speech-like utterance. However, once the listener is told what the utterance was and hears the subject say it again, the words become intelligible and distinguishable. Subsequent novel speech also becomes intelligible. This effect is similar to the adaptation people make when listening to speakers with strong accents or speech impairments. For familiar utterances, the subject's speech is very intelligible; for example, counting and saying the alphabet were never misunderstood even by listeners whose first language is not English. Second, the subject speaks slowly. Third, by using appropriate pitch control the subject produces some relatively natural-sounding speech compared to the text-to-speech synthesizer. As shown through interword pitch variation, proper control of pitch improves intelligibility of the subject's speech. Fourth, even with considerable practice some stops (i.e. P, T, K) are still difficult to discriminate in all contexts. While the R sound still sounds a bit muffled, after considerable practice (approximately 50 hours) the AR, ER, and UR sounds are made reliably in appropriate R-contexts, which alleviated the need for the consonant hand configuration for R to be used in these cases.

At this phase in the subject's speech development, the V/C network was retrained using the expert single time-step network[13]. The different architecture went unnoticed by the subject.

The next phase the subject reached while learning to speak is the fine tuning stage. The subject designed exercises to help overcome problem areas and make his speech more easily understood and natural-sounding. The exercises fell into two categories: speed, and phrasing. Speed exercises involved saying individual words and phrases as quickly as possible without compromising intelligibility. The phrases used increased in length from two words up to 5 words as he became more proficient at the exercise. During these exercises the subject tried to minimize his hand motion through vowel space. An interesting artifact of speaking faster is that vowel accuracy is not vital for intelligibility. The speed exercises help overcome the poor R quality as well as improving stop onset and offset timing appropriately. The phrasing exercises involved saying utterances with appropriate pauses controlled by the foot pedal. Pitch control is further refined by synchronizing the phrasing of the speech with the intonation. It is interesting to note that when phrases in a sentence are said quickly in chunks, separated by bringing foot pedal to full upright position (i.e. volume turned off), the speech quality improves greatly. Together, these exercises help the subject speak better.

The final phase is spontaneous speech. Practicing while conversing with someone helped improve the whole spectrum of skills required for intelligible natural speech. Conversations with unaccustomed listeners are particularly useful since they had not adapted to the peculiarities of Glove-TalkII speech and forced the subject to speak well.

After 100 hours of practice the subject progressed from simple barely speech-like noise to intelligible somewhat natural-sounding speech. The subject exhibits two levels of performance, one for rehearsed speech and one for

[13] The V/C network used was a 10-10-1 feed-forward sigmoid activation network. The 10-5-1 network did not achieve a low enough error during training and was discarded.

unrehearsed. Rehearsed speech sounds similar to slow text-to-speech synthesized speech with natural intonation contours. For unrehearsed speech the subject still has difficulty pronouncing polysyllabic words intelligibly, however, with a few tries he can say any utterance found in the English language.

4.4 Hardware and Software Tools

There are five main pieces of hardware used in Glove-TalkII, four for input and one for output. The glove input device is a Cyberglove. This device has 18 strain gauges embedded inside a lightweight glove. The strain gauges respond linearly to bend angle. The device measures the following:

- 10 finger flex angles (The metacarpophalangeal joint and the proximal interphalangeal joint – the outermost joint of the fingers (distal interphalangeal) are not measured),
- 4 abduction angles between the fingers,
- the angle of rotation of the thumb under the palm (circumduction),
- the angle of rotation of the little finger (minimus) under palm (most noticeable when the thumb and pinky are opposing),
- the wrist pitch angle,
- and the wrist yaw angle.

These angles are measured at a frequency of 100Hz. One serial port is needed to communicate with the Cyberglove.

The second input device is a polhemus sensor which measures the X,Y,Z, roll, pitch and yaw of the hand relative to a fixed source. The small sensor is mounted on the back of the Cyberglove on the user's forearm; thus, the six parameters are independent of the user's wrist motion. The device measures the parameters at a frequency of 60Hz and uses another serial port to communicate to the computer.

The third input device is the keyboard. Nine keys are used to produce stop consonants.

The final input device used is a foot pedal. This device has a variable resistance which is an approximately linear function of foot depression. The variable resistance is used in a voltage divider circuit. The variable voltage is sampled by the A/D circuitry included with the computer at a frequency of 8000Hz. The high sampling frequency is not necessary, however, the A/D circuitry included with the computer does not sample at any lower rate. Additionally, several elastic bands have been attached to the base to provide some force feedback and also to return the foot pedal to the fully undepressed position when the user's foot is lifted.

The output device is an Loughborough Sound Images (LSI) parallel formant speech synthesizer. The device requires 16 speech parameters at 100 Hz to operate. The parameters are quantized to 6 bits (integer range [0,63]). The main parameters are these:

- FN - nasal formant frequency; 95 to 410 Hz in increments of 5 Hz
- ALF - low frequency amplitude; logarithmic scale
- F1 - first formant frequency; 115 to 1060 Hz in increments of 15 Hz
- A1 - first formant amplitude; logarithmic scale
- F2 - second formant frequency; 730 to 2620 Hz in increments of 30 Hz
- A2 - second formant amplitude; logarithmic scale
- F3 - third formant frequency; 1510 to 3400 Hz in increments of 30 Hz
- A3 - third formant amplitude; logarithmic scale
- AHF - high frequency amplitude; logarithmic scale
- V - degree of voicing
- F0 - fundamental frequency; 25 to 417 Hz using logarithmic scale
- MS - mark space ratio - controls the ratio of the length of the glottal open period with respect to the fundamental period

The other parameters determine the characteristics of the glottal source function as well as other details of the speech signal. The parameters are sent to the synthesizer using a parallel port. Within Glove-TalkII, only 10 parameters are manipulated by the user (ALF, F1, A1, F2, A2, F3, A3, AHF, V, and F0). These parameters are sufficient to produce high quality speech[14].

The parallel formant synthesizer was chosen as the output device for two main reasons. First, if the synthesizer is sent the "correct" formants for some utterance, it produces sounds nearly identical to the original speech. Thus, if the user produces "good" formants then he produces high quality speech. Unfortunately, the "correct" formants are difficult to get automatically for spoken utterances. Second, text to speech software is available which outputs formant parameters to drive the formant synthesizer [23]. This feature is vital for obtaining training data for the neural networks. For a more complete discussion of different speech synthesis techniques considered refer to [13].

All the software runs on a Silicon Graphics Personal Iris 4D/35. The Xerion Neural Network libraries simulate neural networks and run all the hardware devices [17]. After all the preprocessing and data collection, there is enough computing power remaining in each 10 msec interval to simulate networks with up to 1000 floating point weights. This restriction places significant constraints on the architectures considered.

4.5 Implementation Notes

There are three important implementation details in Glove-TalkII:

- different sampling rates between the polhemus and the Cyberglove
- calibration of Cyberglove data
- scaling the input data

The 18 data values from the Cyberglove come every 10 msec (100Hz). The data from the polhemus device comes every 16.7 msec. These data are used as inputs to the various networks. This discrepancy is dealt with in the following way. Output to the formant synthesizer becomes the reference point. For the synthesizer to work properly, it must receive data every 10 msec. If data is missed the user will hear clicks and plops in the speech. Thus, no delays longer than 10 msec are allowed when sampling the input devices or simulating the neural networks. The program flow follows this pattern:

1. Send current formant values to the synthesizer.
2. Check if data is available from the Cyberglove (which is sending data at 100Hz).
   - Read data if available.
3. Check if data is available from the polhemus (which is sending data at 60Hz)
   - Read data if available.
4. Run the neural networks to determine the next formants. Note, that if no data are available from either input device, data from the previous sample point will be used.
5. Check if a stop consonant key has been pressed; if so, override any formant produced from the networks and produce appropriate formants for the stop consonant.

There are two important points here. First, the input devices are polled because the data appear asynchronously; waiting for data is not possible since output to the formant synthesizer would arrive too late. Second, when input data is not available, previous data is used. This is especially relevant for the polhemus which is operating at 60Hz; data values are often repeated (in fact, 40% of the time data must be reused). A more sophisticated up-sampling algorithm could be used instead of the simple hold operation used here to extrapolate polhemus data; however, the polhemus sampling rate is considerably faster than the fastest changes in the user's hand position which suggests a simple scheme is sufficient.

The data from the Cyberglove are calibrated to reflect finger angles. Since the response of the sensors is linear with respect to bend angle, all that is required is a gain and offset for each parameter. Each gain and offset is set by adjusting them while the user views the angles graphically and numerically. The calibration parameters are modified until the user obtains a reasonable alignment between at least two positions of the fingers and the calibrated output

[14] As mentioned previously, all 10 control parameters are referred to as formant parameters in appropriate contexts throughout the dissertation even though V and F0 are not formants.

values (usually the user does the alignment at 0 and 90 degrees since they are easy to estimate, though the user is free to choose).

All the input values are rescaled to lie between approximately 0 and 1 in their typical range of operation for use in the networks. The networks produce scaled values of the formants between 0 and 1 which are rescaled to the 0 to 63 range required by the formant synthesizer.

4.6 Summary Remarks

The final Glove-TalkII system maps hand gestures to formant parameters. Input to the system consists of 10 hand parameters measured with the Cyberglove, X,Y, and Z position of the hand measured with the polhemus sensor, 9 keys from the keyboard and the depression angle of the foot pedal. The output drives a parallel formant speech synthesizer [31]. Three neural networks learn and implement the mapping from hand gestures to formant parameters. Additionally, fixed mappings are used for some of the phonemes and qualities of the speech.

Glove-TalkII learns to map hand gestures-to-speech from examples based on a predefined gesture vocabulary. This vocabulary provides a starting point for users to generate example gestures.

The initial mapping is loosely based on an articulatory model of speech. An open configuration of the hand corresponds to an unobstructed vocal tract, which in turn generates vowel sounds. Different vowel sounds are produced by movements of the hand in the horizontal X-Y plane corresponding to tongue position. Non-stop consonants are produced by closure of the index and middle fingers or flexing of the thumb, representing constrictions in the vocal tract. Voicing is controlled by the ring finger. Stop consonants are produced by pressing keys on the keyboard. F0 is controlled by hand height and speaking intensity by foot pedal depression.

The user's interpretation of the initial mapping is learned by Glove-TalkII. The V/C network and the consonant network learn the mapping from the user supplied examples during training phases. The vowel network is trained on artificially generated examples. The F0 and volume mappings are implemented by fixed rules.

Two V/C network structures are used at different times within the final Glove-TalkII system. Both are feed-forward networks with sigmoid activation functions. One network uses an averaging scheme at the inputs to reduce the effects of noisy inputs. Unfortunately, it is considerably larger than the other single time-step architecture. The larger network is for novice users. The smaller network can be used as the user becomes more proficient speaking with Glove-TalkII. Both the vowel and consonant networks use normalized radial basis function units in the hidden layer and sigmoid output units.

One subject was trained to use Glove-TalkII. After 100 hours of practice he was able to speak intelligibly while reading and in conversation. His speech is fairly slow (1.5 - 3 times slower than normal speech) and somewhat robotic, sounding similar to speech produced with a text-to-speech synthesizer but with a natural intonation contour. User controlled pitch greatly improves the intelligibility and naturalness of the speech. Intelligible speech while reading novel passages usually requires several attempts, especially with polysyllabic words. Intelligible spontaneous speech is the most difficult to obtain. The more context the listener has, the easier it is for her to adapt to the speaker's speech.

The subject advanced through eight stages of learning beginning with simple words and sounds and progressing to spontaneous speech. These stages are very similar to some of the stages encountered while learning to play a musical instrument. The stages can also be catergorized according to Fitt's three stages of learning [14]: cognitive, associative and autonomous. Using Fitt's levels, stages 1–4 correspond to the cognitive level, stages 5–7 the associative level and stage 8 the autonomous level. One of the key features discovered while the subject was at levels 3 and 4 was that the V/C network must work well for the user to get adequate feedback about which phonemes he produces. In addition to the usual adaptation based on examples, two significant modifications were made to improve the quality of the subject's speech. First, the X,Y coordinate of the I sound was shifted to lie centrally between EE and E. This change expanded the area in the X-Y plane which produces the I sound. Second, the whole vowel space was compressed by a factor of 0.75.

After 100 hours of practice the subject had the majority of difficulty in the following areas:

1. reliable and accurate vowel production
2. proper timing of R sound
3. polysyllabic words
4. obtaining high speaking rate

Chapter 5

Determining Glove-TalkII's Neural Networks

There are three different neural networks used in Glove-TalkII. As discussed in chapter 4, the first network determines whether the user is making a vowel or a consonant sound (the V/C network). The second network maps the user's hand position into vowel sounds (the vowel network). The third network maps the user's hand gesture into consonant sounds (the consonant network). Figure 5.1 illustrates the combination of the three networks used in the final Glove-TalkII system. The following sections describe each of the three network's tasks and the various network architectures investigated to implement them. The networks showing best performance have been selected for Glove-TalkII from the results of analysis performed on each architecture.

Each network requires both training data and testing data. The techniques used to get data influence the overall performance of Glove-TalkII. Further, the ease of training and test data collection will have strong influence on whether the adaptive behaviour of Glove-TalkII is useful in a real-world setting. Section 5.4 describes data collection techniques for each network. The user provides data for the V/C network and the consonant network, whereas the vowel network is provided artificially generated data. Data from these collection procedures are used to train and test the various alternative network architectures for the V/C network, the vowel network and the consonant network.

5.1 The Vowel/Consonant Decision Network (V/C Net)

The vowel/consonant network (V/C network) is responsible for figuring out whether the user wants to say a vowel sound or a consonant sound based on the current configuration of his hand. It uses 10 hand parameters from the Cyberglove to measure hand configuration: 8 flex angles (metacarpophalangeal and metacarpocarpal/trapeziometacarpal joints (knuckle and middle joints) of the thumb, index, middle and ring fingers), thumb abduction angle and thumb rotation angle. The output is a single number representing the probability that the hand configuration indicates a vowel. As described in section 4.2.2, vowels are produced when the user's hand makes an open configuration illustrated in figure 5.2. Whenever the user has a closed hand configuration he produces a consonant; for example, when he produces an "l" sound (see figure 5.3).

The output of the V/C network is used to weight the outputs of the vowel and consonant networks, which then produce a mixture of vowel and consonant formant parameters. The only training data available will include user-produced vowel or consonant sounds, no intermediate data is available[1]. The network should interpolate between the intermediate hand configurations to create a monotonic transition between vowels and consonants. A fairly sharp transition enables the user to move quickly from a vowel sound to a consonant sound in a reliable manner.

The V/C network should also produce reasonable outputs for unexpected hand configurations. For example, if the user hyperextends her hand, the network should continue to produce a vowel sound, since the hand remains in an open configuration. Similarly, if she exaggerates a consonant hand configuration, some type of consonant sound should be produced since the hand forms a virtual constriction. Such configurations are unusual and not expected in the training corpus, but may occur while the user is speaking.

Two different network configurations are considered. The first network is a 10-5-1 network with sigmoid activations. The inputs are the 10 scaled hand angles specified above. The output is the probability of a vowel sound. The important feature of this configuration is that only the current hand configuration influences whether a vowel or consonant is produced. The second network is a 30-15-1 network which uses 3 banks of 10 shift units of the 10

[1] Intermediate training data is difficult to produce since the text-to-speech synthesizer only produces formant data which is categorized as vowels or phonemes.

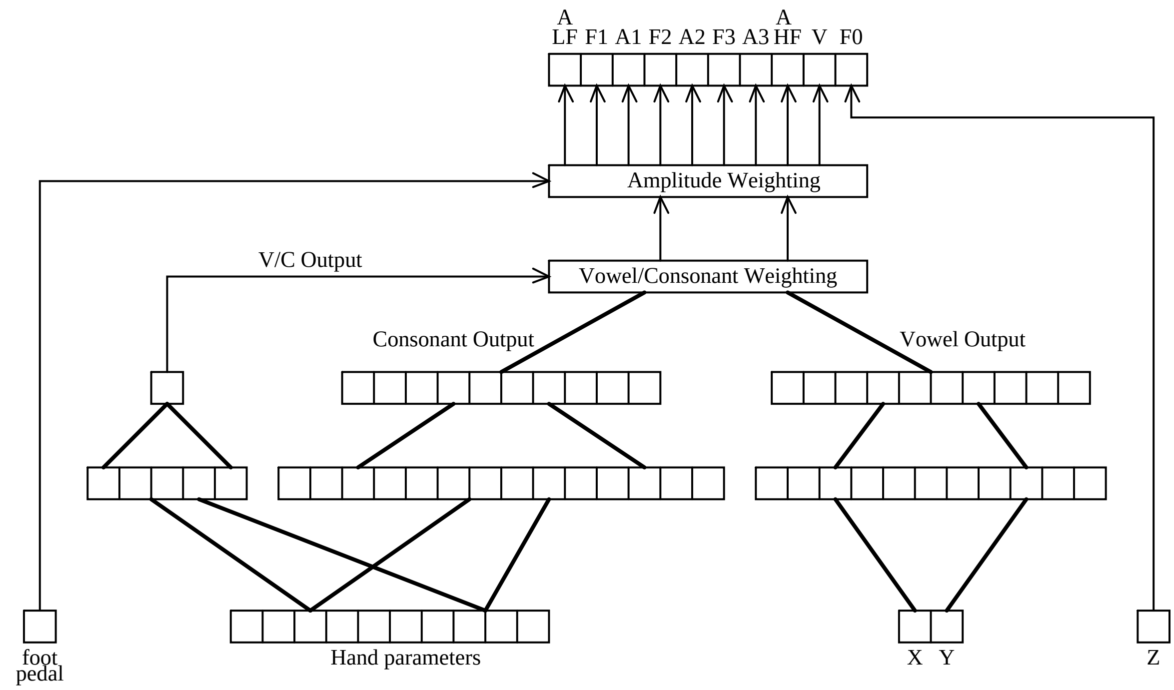


Figure 5.1: Network configurations for the final Glove-TalkII system

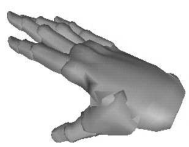


Figure 5.2: Typical hand configuration for a vowel sound.

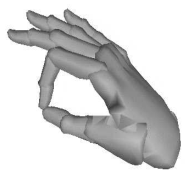


Figure 5.3: Typical hand configuration for a consonant sound (the user is making an L sound).

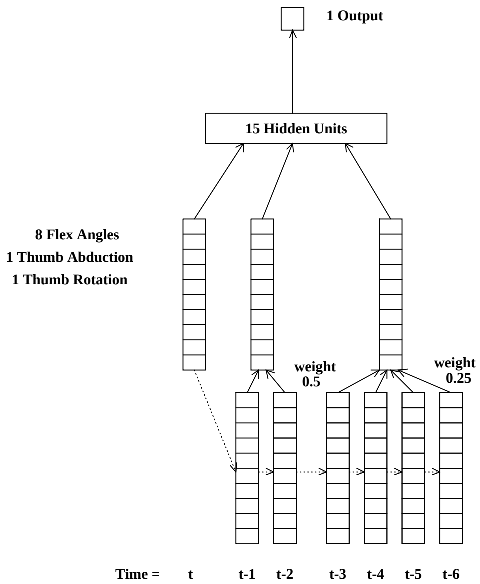


Figure 5.4: Proposed architecture for V/C network. This network uses three banks of shifted values of hand parameters. The first bank uses the current time step of hand parameters. Subsequent banks average previous hand parameter values as shown in the figure.

hand parameters. The current value of the 10 hand parameters (i.e. the values of the hand parameters at time t) constitute the values of for bank of units. The second shift unit bank is the mean of the next two time steps of the hand angles (i.e. average hand parameters from time $t-1$ and $t-2$). The final shift unit bank is the mean of the next four time steps (i.e. from hand parameters averaged at times $t-3$, $t-4$, $t-5$ and $t-6$). Figure 5.4 illustrates this V/C network architecture. A reliable V/C decision may depend on the history of hand configurations, for example, a history of noisy hand data. In such case, if the network only considers the current (noisy) time step of hand data, it may oscillate between a vowel and consonant decision even though the user's hand is relatively still. This is not desirable behaviour perceptually since the speech outputs from the vowel and consonant networks are usually very different, causing the speech to switch back and forth between two very different sounds. Conversely, it is desirable to have the V/C network respond quickly when the user changes from a vowel configuration to a consonant configuration and vice-versa. The averaging scheme used provides a compromise between sensitivity to current hand configurations and averaging over past values to reduce the effects of noise.

A trade-off typically exists with averaging schemes between the relative importance of current hand parameters and past ones. In terms of the vowel/consonant decision, it is important that the user hear any difference immediately during transitions between consonants and vowels so that he can quickly adjust his speech. Clearly, current hand parameters are very important. Conversely, his hand may potentially produce noisy hand parameters (due to shaking for example), thus some averaging seems worthwhile. The compromise made here is to use a total of 7 time steps of hand parameters. The current time step is not averaged at all so that the network has access to the most recent hand configuration. The remaining time steps are averaged as specified above, allowing different amounts of averaging, depending on the relative temporal distance from the current time.

5.1.1 Performance of the V/C Network

The two VC networks are compared perceptually and in terms of mean squared error. The training data consists of 2600 examples of consonant configurations (350 approximants, 1510 fricatives [and aspirant], and 740 nasals) and 700 examples of vowel configurations obtained from an expert user. The test set consists of 1614 examples (1380 consonants and 234 vowels). Section 5.4.2 describes training and testing data collection. Table 5.1 shows the results from both networks.

As is clear from table 5.1, both networks perform very well in terms of mean squared error. Both networks performed well when used with the whole system. During normal speaking neither network made perceptual errors. For the user that these networks were trained for, the decision boundary feels quite sharp, and provides very predictable, quick transitions from vowels to consonants and back. Also, vowel sounds are produced when the user hyperextends his hand. Any unusual consonant configurations which intuitively should be consonants produced consonant-like sounds. As the smaller of the two, network one appears to be the most obvious choice for the final Glove-TalkII system. However, since network two's averaging allows greater noise levels in the user's movements, it should be considered, especially if the user has difficulty making reliable hand gestures. Such difficulties arise when a user first

Net	V/C	epochs	Train mean squared	Test mean squared	weights
1	vowels	47	0.00000	0.00000	61
	consonants		0.00000	0.00000	
2	vowels	61	0.00000	0.00135	481
	consonants		0.00000	0.00000	

Table 5.1: Training and testing results for the two V/C networks. In the training set there were 2600 consonants and 700 vowels. In the test set there were 1380 consonants and 234 vowels.

learns to speak with Glove-TalkII. As the user progresses, network one can be trained on the more robust data and used subsequently.

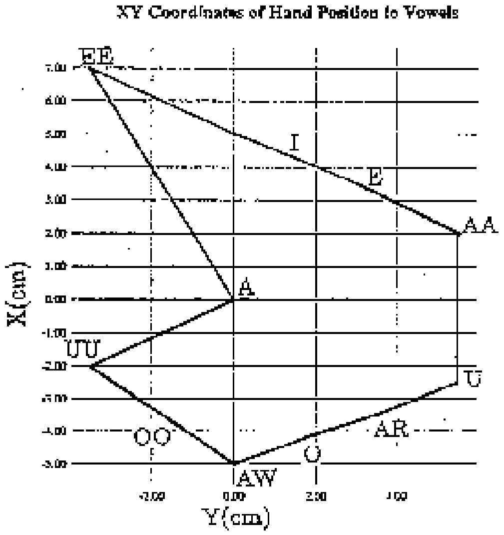


Figure 5.5: Hand-position to Vowel Sound Mapping. The coordinates are specified in centimeters relative to the origin at the sound A. The position of the source is located at the top middle of the graph, 13 cm above the A sound, i.e. the source's coordinates are (13,0) (X is measured vertically). The X and Y coordinates form a horizontal plane parallel to the floor when the user's sits.

5.2 The Vowel Network

In Glove-TalkII, the user's hand position in the x-y plane determines vowel sounds. Eight different neural network architectures are considered in this section to map the x and y values of the hand into the 8 relevant formant parameters (ALF, F1, A1, F2, A2, F3, A3 and AHF; V is fixed at 63 since all vowels are fully voiced). The absolute x and y positions of the hand for each static vowel sound are determined apriori (see figure 5.5). From this predefined mapping, artificial training and test data are generated. Section 5.4.3 describes the initial mapping and how training and test data are generated. For all the networks considered, the x and y values and formant values were roughly scaled between 0 and 1.

Eight different networks are considered:

1. 2-8 linear network
2. 2-8-8 feed forward network with sigmoid activations
3. 2-8-8-11 feed forward network with sigmoid activations and an extra 11 unit softmax layer (see below)
4. eight 2-3-1 feed forward networks with sigmoid activation functions
5. (eight 2-3-1)-11 feed forward network with sigmoid activation functions and an extra softmax layer (see below)
6. 2-11-8 radial basis function network with no normalization and sigmoid output units
7. 2-11-8 normalized radial basis function network and sigmoid output units
8. 2-11-8-8 normalized radial basis function network and sigmoid output units

Preliminary testing showed that the linear network and the feed forward sigmoid activation networks had difficulty separating some of the sounds (this was especially true for similar consonant network architectures, see section 5.3). Therefore, some networks were tried with an extra layer of softmax units [3] (see appendix A) connected to the outputs of the networks. These networks can be thought of as performing "soft" categorization of the formant outputs (in addition to minimizing the formant error) into one of the 11 known vowel categories during training. This process should help the network disambiguate conflicting sounds (in the sum-square error and perceptual domain). For example consider network 3, a 2-8-8-11 network. Figure 5.6 shows it schematically. Notice the formant outputs are fully connected to the 11 softmax units. The softmax units categorize the outputs into one of the 11 target vowels. Note that the softmax units are used only to help train the networks, and only the formant outputs are needed when the system speaks. Requiring the network to categorize the formant outputs causes it to learn to minimize the mean squared error and to maximize its ability to discriminate. Figure 5.7 illustrates how this process works. Here the

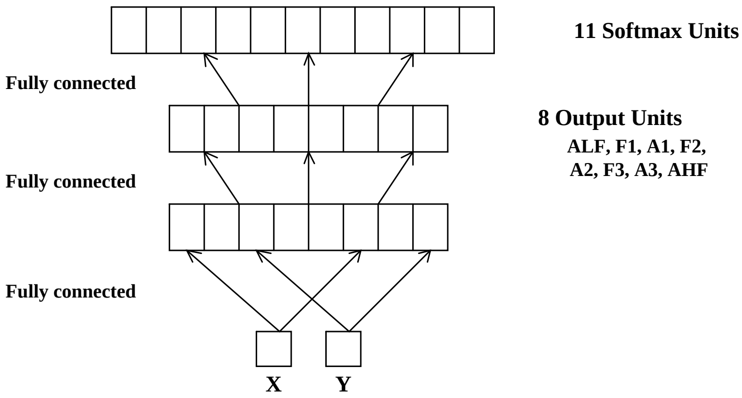


Figure 5.6: Network 3 - the formant output units are fully connected to the layer of softmax units.

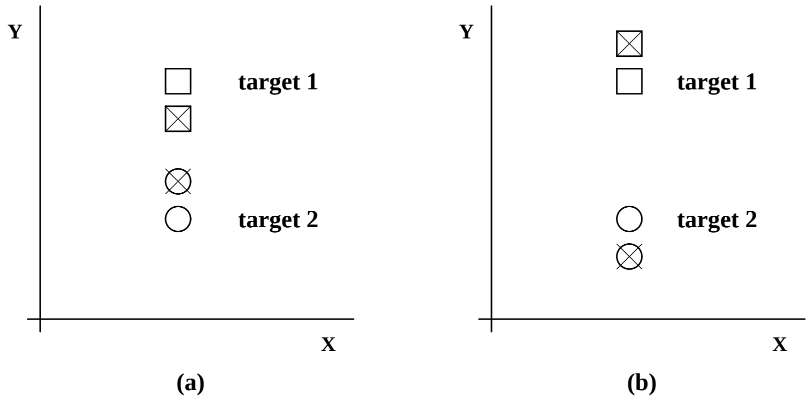


Figure 5.7: Example of more separate equivalent squared error. In figure a the outputs are between the two target values. In figure b, the outputs are outside the target values making them easier to separate; yet, the mean squared error is the same in both cases.

target values are shown by t_1 and t_2. The output of a network for two cases of each class are marked x_1 and x_2. In both cases, the squared error remains the same, but in case b the outputs separate more easily. The analogue in the formant space makes the outputs in case b perceptually more distinct than in case a. The extra layer of softmax units at the outputs should pressure the network to behave as in case b.

The second novel architecture considered uses normalized radial basis function (RBF) units [5]. The radial basis function units used are described by the following equation:

$$o_j = e^{-\frac{\sum(w_{ji}-o_i)^2}{{\sigma_j}^2}} \tag{5.1}$$

where o_j is the (un-normalized) output of the RBF unit, w_{ji} is the weight from unit i to unit j and σ_j is the width of the RBF. The summation is performed over all units connected to unit j. The RBF can be considered as a gaussian bump centred at the incoming weights (w_{ji}'s) with variance ${\sigma_j}^2$ (or width of σ_j). The normalization (as described in appendix A) is this:

$$n_j = \frac{o_j}{\sum_{\omega\in\Omega} o_\omega} \tag{5.2}$$

where n_j is the normalized output of unit j and the summation lies over all the units in the group, Ω, of normalized RBF units. The X and Y values of each of the 11 vowels in the predefined mapping (see figure 4.4 or figure 5.5) determine the centres of the RBFs.

Normalization should provide the required interpolation properties for the task. In particular, the user has a wider dynamic range in the x-y space than the training data. This occurs because the training data was determined from typical values in the x-y plane instead of considering every possible value. Hence, many values in the x-y plane are never seen by the network during training but are seen during running of the network. The network should respond in a reasonable way to these novel observed values. To illustrate, consider the vowel mapping in figure 5.5.

Here the (x,y) coordinates are the abscissa and the ordinate axis. The 11 cardinal vowels are positioned on the page according to the hand position that should produce them. The corresponding cardinal-vowel formants should be produced at these 11 positions. This can be accomplished relatively easily by an RBF network. But, what should be produced at places far away from the cardinal vowels? Consider for example, what sound should be produced if the user's hand is in the upper left corner? One answer to these questions is that once the user's hand extends past the cardinal sound, only the cardinal sound can be heard; that is, if the user passes the EE sound the sound produced continues to sound like an EE. Normalizing the RBF units provides this type of extrapolation. To understand why, imagine 11 RBFs centred at the cardinal vowel sounds with some small but reasonable width. Next, consider a point left and above the RBF centred at the EE sound. The EE-RBF will have a greater response (though the response will likely be very small) than all the other 10 RBF's since it is closest. After normalization this difference magnifies so that only the normalized EE-RBF responds, producing an EE sound.

The next section discusses and compares the training and test results for each network architecture.

5.2.1 Performance of the Vowel Network

The different vowel networks are compared in four different ways. First, the mean squared error is compared for the training set and a test set. Second, a range of X and Y values which span the user's movement space are input to the networks and the F1 and F2 output values are plotted and compared. Third, a perceptual comparison is made by actually attempting to speak with the networks. Fourth, the sizes of the network are considered since significant computing constraints exist when using the system to speak.

Table 5.2 shows the training and test results from the different networks. The training set consists of 100 examples of each vowel sound with random noise added to the x and y centre values. The added noise was picked from the range [-0.025, 0.025] for x and y with uniform distribution (this corresponds to a range of [-0.5,0.5] cm in the x range and [-0.26,0.26] cm in the y range). Fifty examples of each cardinal vowel with different ranges of noise added to the centres provide test results; for example, the test results under the 0.2 column derive from a test set with uniformly distributed noise in the range [-0.1,0.1] added to the x and y coordinates of fifty examples of each cardinal vowel. The number of run weights is the number of weights in each network when the network is used in the final system. For most of the networks there are the same number of weights when the training and running. The only networks where the number is different are the softmax networks since the softmax units are not required once the networks are trained.

Two variations were tried with networks 7 and 8. In both networks, variation 'a' uses fixed RBF widths of $\sqrt{0.025}$ (half of the range of random noise added to the training data). In variation 'b' the widths were nominally set to $\sqrt{0.025}$ and then optimized with the rest of the parameters.

All the networks were trained with conjugate gradient descent and a line search [17]. Weight decay was investigated with some of the networks, but offered no significant improvement in results (and in many cases caused poorer performance). Training stopped after 1000 epochs unless the error appeared to stabilize earlier. With networks 1 and 6b the line search failed [17] after 6 and 126 epochs respectively.

For networks using the extra layer of softmax units (networks 3 and 5), errors from the softmax units had to be weighted with respect to the mean squared error. If the errors were weighted equally, the cross entropy error from the softmax would dominate the mean squared error, causing very poor performance. After a number of different weightings were tried, a relative weighting of 0.01 was found to give good results.

Network 8b is the best network (on the most difficult test set) in terms of mean squared error with an expected formant error of $\pm$ 5%[2]. It is also the largest network to use with 201 run weights. However, if just the mean squared error is considered, all the networks appear to have generalized reasonably well; at least for noise levels up to 0.2. The worst network in terms of generalization is network 3 with an error of 0.0197 which leads to an expected error for each formant of $\pm$ 7%. Network 7a is emphasized in the table since it is the one used in the final version of Glove-TalkII for reasons described below. Further comparison is still possible before actually trying to speak with each network to determine which networks are suitable.

Vowels can be identified approximately by the first two formants, F1 and F2. Thus, one technique for comparing the different networks involves graphing the F1 and F2 outputs with respect to a large range of x and y values. To this end, a test set was created which tesselated the x-y plane in the interval [0,1] on each axis by 400 evenly distributed points. The F1 and F2 outputs of the networks were plotted together on one graph. These graphs show for each network which vowel sound is produced depending on the position of the user's hand. F1 is plotted as the

[2] The expected formant error for the vowel net is found by the following equation:

$$fe = \sqrt{\frac{2 \cdot \text{total error}}{(\#\text{ examples}) \cdot (\#\text{ formants})}}.$$

Net	epochs	Training Error		Testing Mean Squared Error			run	total
		mean	mean cross-				weights	weights
		squared	entropy	0.05	0.1	0.2		
1	6	0.0069		0.0070	0.0087	0.0154	24	24
2	1000	0.0023		0.0025	0.0052	0.0168	96	96
3	1000	0.0027	0.0056	0.0028	0.0060	0.0197	99	99
4	1000	0.0022		0.0024	0.0050	0.0162	104	104
5	1000	0.0027	0.0032	0.0028	0.0057	0.0181	104	104
6a	200	0.0010		0.0013	0.0030	0.0112	118	129
6b	126	0.0010		0.0012	0.0031	0.0124	129	129
7a	150	0.0016		0.0018	0.0038	0.0120	118	129
7b	50	0.0007		0.0008	0.0020	0.0107	129	129
8a	150	0.0019		0.0021	0.0039	0.0112	190	201
8b	150	0.0006		0.0007	0.0019	0.0103	201	201

Table 5.2: Training and test results for 8 different vowel networks. There are 1100 training cases and 550 test examples. The mean cross-entropy error is scaled by 0.01 during training. Three different noise levels are used in the testing data. For networks 7 and 8, case 'a' has the RBF variance fixed at 0.025 and for case 'b' it optimizes from an initial value of 0.025.

radius of the circle and F2 is plotted as its intensity at each of the test points. For example, as shown in figure 5.8, a small, medium intensity dot represents the EE sound while a large, medium intensity dot represents a U sound. The 11 cardinal vowel sounds used to train the networks are shown on the graph as reference points. Remember, the networks were trained on X and Y coordinates of these canonical vowels with added noise ($\pm$ 0.025). The range of X and Y values plotted shows almost the full range of motion the user will ever make while speaking, therefore, the graphs give a good indication of the vowel sounds the user will say.

Consider the graph in figure 5.8 which is produced using network 1. This graphs shows that at each of the reference points the network produces sounds reasonably close to those desired. However, outside the reference points the formants are extrapolated linearly. That is, the changes in each of the parameters continues changing the formant parameters which in turn makes the sounds less vowel-like. For instance, the circles get very small and the intensity decreases for points to the left and above the EE sound. The speech therefore begins at the EE sound but as the user's hand moves to the left and above, it changes to an extreme EE sound, and finally to a non-vowel like sound. Therefore, if the user overshoots the EE sound she will get unintelligible speech using network 1. Networks 1-5 all exhibit this type of behaviour.

The unnormalized RBF network suffers a different problem. The vowel space for network 6a in in figure 5.9 shows that the values inside its reference points are all very reasonable; however, outside the reference points the vowel sound becomes similar to the A sound. This is not surprising since none of the RBF units are active outside the reference range, thus, the RBF layer has no activity and the output is determined by the output units' bias. After the training session the biases produce an A sound like the average of all the vowels.

The normalized RBF networks (networks 7 and 8) are the most interesting. In the cases with fixed RBF widths, the extrapolation appears sensible as shown in figure 5.10 which is obtained from network 7a. Consider the area to the left and above EE. For a large part of the area, the EE sound is produced (though it is slightly closer to the cardinal value using network 8a). For each cardinal vowel a corresponding patch outside the reference area extends to the border. This is desirable behaviour since overshoots will continue to produce the same vowel sounds as the nearest cardinal vowel. The RBF networks which allow their widths to vary also have this desirable quality as shown in figure 5.11 obtained from network 7b. However, it only extends so far. At the extreme values (i.e. the large gray area in the figure) the RBF units underflow, causing the normalization to be ill-defined.

The analysis of the vowel space suggests that either network 7 or 8 would be best. At this point one must try speaking with each of the different networks to determine the best choice. Networks 1, 3, 7a and 8a were all tried. As expected, both networks 1 and 3 caused problems whenever dipthongs like IE and AE were spoken. In these cases, the user often overshot the EE sound and produced a nonsensical sound. Network 7a and 8a sounded very similar. With both networks all vowel sounds were easy to produce. The smaller of the two, network 7a, is chosen as the best network to use in the final version of Glove-TalkII.

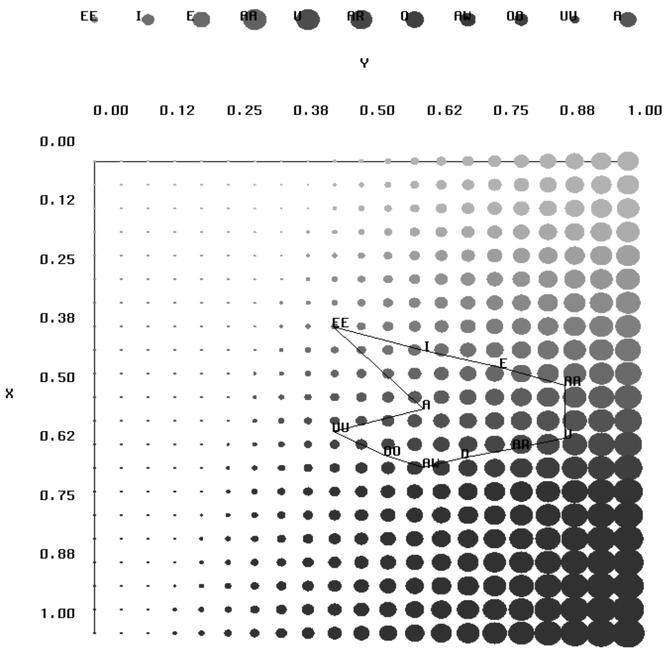


Figure 5.8: Vowel space for network 1.

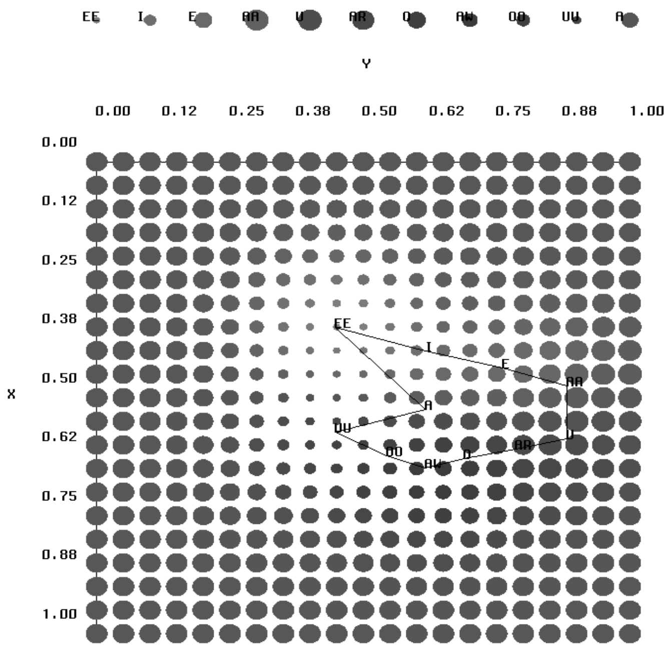


Figure 5.9: Vowel space for network 6a

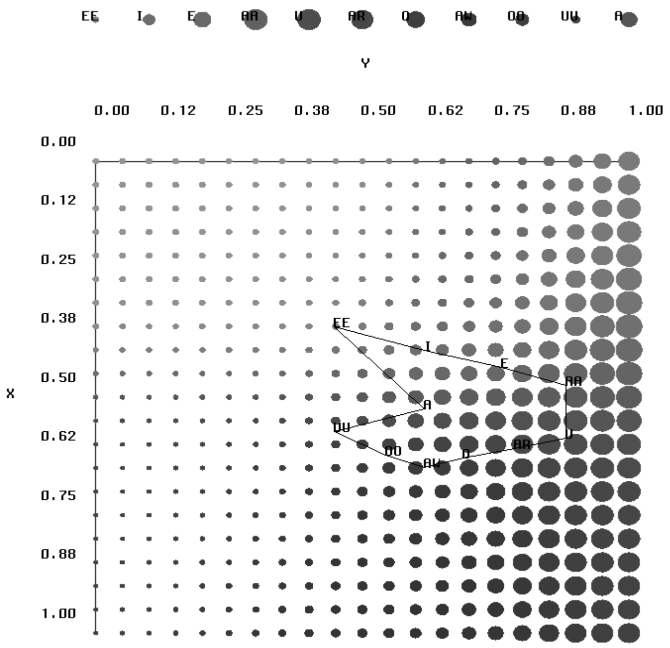


Figure 5.10: Vowel space for network 7a

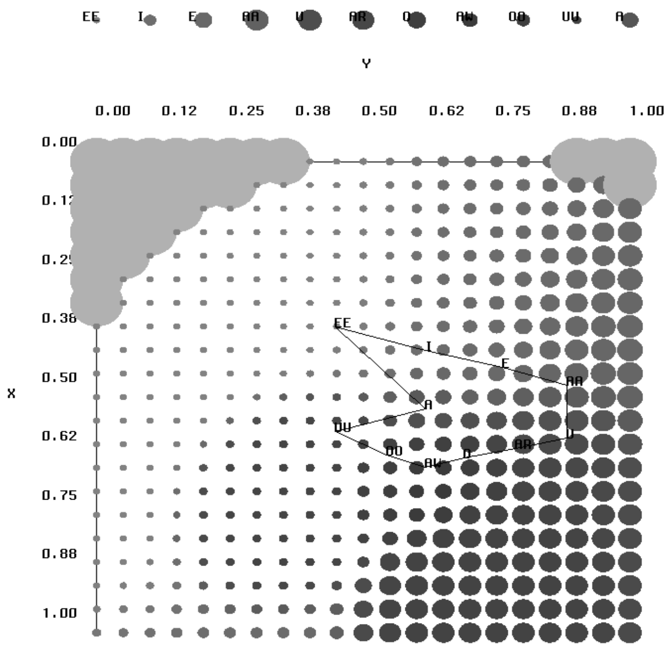


Figure 5.11: Vowel space for network 7b

5.3 The Consonant Network

The consonant network translates hand configurations to consonant sounds. Section 4.2.2 describes the mapping. There are 14 static hand configurations mapped to static formant parameters, corresponding to available phonemes from the text-to-speech synthesizer. The input requires only 10 parameters to measure the hand configuration. Angles measured include the inner and outer flex angles of the thumb, index, middle and ring fingers, the thumb abduction angle and the thumb rotation angle. The output space has 9 formant parameters consisting of ALF, F1, A1, F2, A2, F3, A3, AHF and V. Notice the voicing parameter is required since consonants have different degrees of voicing. This space has considerably more dimensions than the vowel space, thus, training data obtained from the user may not cover the entire space. One feature that makes consonant space different from vowel space is the relatively well-defined ranges of the input parameters.

Several performance criteria are relevant when considering the design of the consonant network. First, the training data from the text-to-speech synthesizer has 14 distinct static phonemes (that are not stop consonants). Each of these phonemes corresponds to a static hand configuration (see the definition of the initial consonant vocabulary in section 4.2.2). The consonant network must map these hand configurations so that each phoneme sounds clear. Second, interpolation between static phonemes should be reasonable. When the user moves his hand between different static phonemes (i.e. from an "l" to an "r") the intermediate sounds produced should not pass through any other phonemes. The hand configurations have been carefully designed to minimize overlap; however, the inductive bias of the network remains responsible for sensibly interpolating the formant space between hand configurations not in the training set. Third, another desirable feature, similar to the vowel space property, is that small errors in achieving targets in formant space should make the sounds more distinctive rather than less distinctive.

Four different consonant network architectures are compared:

1. 10-9 linear network,
2. 10-14-9 feed forward network with sigmoid activations,
3. 10-14-9-14 feed forward network with sigmoid activations and an extra layer of softmax units,
4. 10-14-9 normalized radial basis function (RBF) network with sigmoid output units.

Network 1 and 2 are self-explanatory. Network 3 is a feed forward network with an additional layer of softmax units to assist training (see appendix A). As discussed in section 5.2, softmax units are added to make formant space squared errors sound more distinctive. This becomes particularly important in consonant space since some of the consonants are easily confused (i.e. M and N). In network 3, the softmax units are fully connected to the formant output units. The softmax units are not used during speaking. The normalized RBF network (network 4) is similar to the one used in the vowel network (section 5.2).

The next section compares the performance of each of the networks. The networks are compared using mean squared error, subjective listening tests and size. Unfortunately, consonants depend critically on all 9 formant parameters, so it is difficult to compare the networks graphically as done with the vowel networks.

5.3.1 Performance of the Consonant Network

The four different networks are compared in three different ways. First, the mean squared error is compared on a training set and a test set. Second, speech produced while using each of the four networks gets compared. Third, the size of the networks are compared.

Table 5.3 shows the training and test results from the different networks. The training data consists of 2600 examples. The static consonant phonemes can be broken down into approximants (R, L, and W), fricatives (F, V, S, Z, SH, ZH, TH, DH and the aspirant H) and nasals (M and N). There were 350 approximants (approximately 115 examples of each), 1510 fricatives (170 examples per fricative) and 740 nasals (370 examples per nasal). The 1380 example test set consisted of these:

- 255 approximants
- 960 fricatives
- 165 nasals

The consonants are divided into different categories since it is important to see if the networks perform better on some types of phonemes than others. Section 5.4.1 describes collection of the training and test data.

The number of hidden units in networks 2 and 3 are chosen so that they are comparable in size to network 4. As in the case of the vowel network using softmax units, consonant network 3 weights softmax errors by 0.01. For

Net	consonant	epochs	Mean Squared Training Error	Mean Squared Testing Error	run size
1	approximant	142	0.048	0.105	99
	fricative		0.036	0.097	
	nasal		0.035	0.044	
	total		0.037	0.091	
2	approximant	499	0.004	0.022	289
	fricative		0.004	0.084	
	nasal		0.001	0.002	
	total		0.003	0.063	
3	approximant	700	0.003	0.003	289
	fricative		0.004	0.051	
	nasal		0.001	0.001	
	total		0.003	0.056	
4	approximant	172	0.001	0.001	289
	fricative		0.007	0.014	
	nasal		0.001	0.002	
	total		0.005	0.010	

Table 5.3: Training and test results for 4 different consonant networks. Network 3 used an extra layer of softmax units connected to the outputs with the cross-entropy errors weighted by 0.01; only the sum squared error on the formant outputs are reported.

network 4 the centres of the RBF units were determined by averaging the hand parameter values for each of the 14 consonant phonemes. After experimenting with different values, the RBF widths were fixed at a value of $\sqrt{0.05}$.

All the networks were trained with conjugate gradient descent and a line search. The training stopped when the error did not decrease significantly.

Clearly, network 4 scores best in terms of mean squared error on the test set. The second best network is network 3. Most of the errors of these two networks come from fricatives. This is not surprising since the hand configurations for fricatives are very similar.

Informal listening tests are performed using each network in order to compare them. The main trials consist of trying to produce static consonant phonemes. Network 1 produced very poor speech. Most of the consonant phonemes are difficult to produce if not impossible. With network 2 all the phonemes are easy to produce except for W, M and N. The W phoneme is impossible to make. The M and N phonemes are indistinguishable; sounding something like a cross between M and N most of the time. With network 3 the M and N phonemes are more distinctive as expected, however, the W and SH phonemes are difficult to produce. All the other consonants sound good. Network 4 produces all the consonants well. The only difficulty is that the R and L phonemes are very sensitive to motion of the index finger.

The speech produced with network 4 sounds best. Further, all the networks that produce reasonable speech require the same amount of computation at run time; hence, network 4 is selected for the final version of Glove-TalkII.

5.4 Generating Training Data and Test Data for Glove-TalkII

This section describes the procedures used to collect training and test data for all the networks in Glove-TalkII. The data for the consonant networks and vowel/consonant decision network are obtained from the user. The data for the vowel network are created from a user-defined mapping and does not require data from the user.

Initially, before any data are collected from the Cyberglove, the sensors are calibrated. Two techniques are used: graphical and numerical. Since the sensors have a linear response, the user adjusts a gain and offset for each sensor value. The gain and offset are adjusted so that a graphical image of the user's hand corresponds to her hand. Displaying the flex angle values enables further fine tuning of the calibration parameters. The user adjusts the gains and offsets so that the displayed angles correspond to the angles of her hand. Since the sensors are relatively robust, the calibration can be performed infrequently (i.e. once a year).

Also, before data collection, the typical ranges of motion of each sensor are determined to produce a scale file. The scale file values are used to scale the sensor values to between approximately 0 and 1 for use as inputs to the neural networks. Appendix B shows a typical example of a scale file.

Once the hardware has been calibrated and the ranges have been determined the data collection can begin. The data collection for the consonant network is described first. Data for the V/C network is collected in the same manner as for the consonant network, as described in the next section. The vowel data, created artificially, is described last.

5.4.1 The Consonant Network

The text-to-speech synthesizer produces 14 different static consonant phonemes (not including stop consonants). The initial mapping described in section 4.2.2 provides the static hand configurations for each phoneme. To train the consonant network the user must provide real example hand configurations for each of the 14 different phonemes.

The first step in the data collection procedure is creating target data for each of the 14 consonant phonemes using the text-to-speech synthesizer. Table C.1 in appendix C lists the formant target values for each of the 14 phonemes.

The typical data collection scheme for a single phoneme is as follows:

1. A target consonant plays for 100 msec through the speech synthesizer.
2. The user forms a hand configuration corresponding to the phoneme.
3. The user depresses the foot pedal to begin recording; the start of the recording is indicated by a green square appearing on the monitor.
4. 10-15 time steps of hand data are collected and stored with the corresponding formant targets and phoneme identifier; the end of data collection indicated by the green square turning red.
5. The user chooses whether to save data to a file and whether to redo the current target or move to the next one.

The targets are presented in a fixed order grouped into blocks. The three blocks are approximants (L, R, and W), fricatives (F, V, S, SH, Z, ZH, TH, DH and the aspirant H), and nasals (M and N). All 24 input parameters of the hand are recorded during the training sessions and coupled to the appropriate targets. These are scaled by the minimum and maximum values in the scale file. Notice that left hand parameters are also recorded (with a VPL DataGlove) which may or may not be used by the networks.

Typically, the user performs about 5 trials per consonant before continuing on to the next target in the group. While the training data is being recorded, the user is encouraged to move his hand simulating the motions he might make while saying the sound. The more representative the training data the better the system will work once trained. For the results reported the data collection scheme was repeated at different times until 350 approximants, 1510 fricatives and 700 nasals were collected and scaled for the training data. A disproportionate number of nasals were collected since the consonant network had a difficult time finding suitable decision regions for the M and N phonemes. By providing more training data it was hoped that the networks would produce a better mapping for nasals. For the test data, 255 approximants, 960 fricatives and 165 nasals were collected and scaled.

The use of the foot pedal to begin recording data represents an important development. The foot pedal makes data collection very quick and easy. Before the addition of the foot pedal several methods were tried which proved difficult to use. One technique required the user to move his hand past an imaginary plane (using the polhemus to measure his hand position) to begin the recording. There were two problems with this approach. First, it was difficult to predict exactly where the imaginary plane started, causing mis-starts. Second, the user's hand is always in forward motion when recording data. When the user speaks, his hand moves in many different directions and it is important for the consonant network to learn to allow for the interaction between the user's hand motions and his hand configuration. Another technique required the user to press a key to begin recording. The main problem with this approach was the difficulty synchronizing hand motions with the keystroke. For example, the user first concentrated on pushing the button when ready to record but then did not move his hand much to provide varied hand data for a given phoneme. The foot pedal overcame all these problems. Specifically, the foot pedal provides an effective means to synchronize beginning the data recording and producing hand gestures. Further, it is relatively easy to perform hand gestures while depressing the foot pedal.

A further useful feature gives the user the ability to choose immediately what to do with each example. When the user makes obvious errors she can immediately delete the examples. This feature eliminates much of the data that would cause problems for the learning algorithms. This becomes particularly important for novice users since their error rates are very high. Deleting poor examples after all the data has been generated is cumbersome and error prone.

As the user becomes more advanced he may want to adjust the mapping to better suit his individual needs. The training procedure enables him to do so easily. For example, to do an entire training session of 5 examples for each consonant takes about 30 minutes. Novice users spend considerably more time learning the mapping themselves so that they can reliably make the phonemes needed. There is also a pseudo-spectrogram tool available to generate and play different target phonemes and store them for consonant data collection. The success of an adaptive interface depends critically on the ease with which examples can be collected from the user. In the case of Glove-TalkII, considerable care was taken to provide an easy-to-use procedure with which the user can quickly create more training data.

5.4.2 V/C Network

The V/C network requires hand configurations labeled as consonants or vowels. The hand data collected for the consonant network corresponds exactly to the data needed to indicate that a consonant should be produced. Thus, no further data collection is necessary for consonant-decision data. All consonant data is relabeled with targets specified as 0.0 (indicating a consonant). Unfortunately, for vowels, the hand configurations go unspecified in training data for the vowel network, since only the X and Y coordinates are used for inputs and the data are not collected from the user.

To get vowel targets for data the V/C network the same procedure used for collecting consonant data is used (see section 5.4.1). Instead of producing consonants though, the user must make hand configurations that produce vowels (i.e. she makes an open configuration of her hand according to the mapping defined in section 4.2.2 and illustrated in figure 5.2). Typically, the training session is set up to record 100-200 time steps (1-2 seconds) of data from the user, rather than only the 10-15 time steps recorded during the consonant network training session. During the recording stage, the user moves her hand around as much as possible creating many different hand positions that produce vowels. For example, the user hyperextends her fingers as well as moving towards consonant configurations without actually making a consonant shape to simulate vowel-consonant transitions and vice-versa. Each data vector is labeled with a target of 1.0 indicating a vowel.

700 vowel decision examples were collected in this manner. As stated above, the training data for the consonant network was relabeled providing 2600 examples of consonant decisions.

5.4.3 The Vowel Network

The training and test data for the vowel network is artificially generated based on an initial vowel mapping (see section 4.2.2). Obtaining reliable vowel data from the user for vowels is very difficult since there is poor feedback to provide absolute position of the user's hand relative to the source. This makes it difficult for users to repeat hand positions reliably for vowels as specified in the initial mapping. Absolute positioning is not as much of a problem while speaking since the sounds, once produced, provide feedback indicating the user's location in vowel space.

Each of the 11 cardinal vowel phonemes is coupled with an X-Y coordinate in the initial mapping, defined in table 4.4 in section 4.2.2. Table C.2 in appendix C indicates the target formant parameters for each vowel phoneme. Training (and test) examples are generated from the initial mapping.

For the training data, 100 examples of each vowel are generated providing 1100 examples. Noise was added to the *scaled* X and Y coordinates for each example. The added noise is uniformly distributed in the range -0.025 to 0.025. In terms of unscaled ranges, these correspond to an X range of approximately $\pm$ 0.5 cm and a Y range of $\pm$ 0.26 cm.

Three different test sets were created. Each test set has 50 examples of each vowel for a total of 550 examples. The first test set uses additive uniform noise in the interval $\pm$ 0.025. The second and third test sets use additive uniform noise in the interval $\pm$ 0.05 and $\pm$ 0.1 respectively.

Since data is generated automatically, it is an easy matter to change the initial vowel mapping. To do so requires the user to redefine the vowel mapping to the specifications he wants and then regenerate the training data (and test data). The vowel network then retrains on the new data.

5.5 Summary of Glove-TalkII's Neural Networks

The above analyses illustrates the process of selecting the final network architectures for the Glove-TalkII system. The recommended architecture has two parts for the V/C network. The novice user should use the larger, noise resistant network. As the user begins to make more reliable hand gestures, the smaller, single time-step network becomes sufficient[3]. Clearly, for the vowel and consonant networks the normalized RBF hidden units are preferred.

Training the vowel network on artificial data from the initial mapping provides an easily navigated vowel space. The size of the vowel space can be easily adjusted to suit individual tastes by creating new training data from a user-defined mapping. An easy to use interface helps obtain the training data for the consonant network. To fine tune the consonant mapping the user only has to provide new training data for the phonemes she wants altered. The training data from the consonant network is exactly the data needed to train the V/C network to output a consonant decision. For vowel decision data, the user makes as many different vowel hand configurations as anticipated during speaking.

The main problem with using RBF units for the hidden layer is that if additional sounds/phonemes are required (i.e. to allow for a different language) additional RBF units must be added. This increases the size of the network, and, the whole network has to be retrained. The other network architectures need only be retrained with each new sound/phoneme added to the rest of the training set.

It is important to stress that the mean squared error criterion is only a guide to selecting the best vowel and consonant network. Subjective listening tests must be performed before the final architecture can be decided upon.

[3]Even though, in theory, as long as the computational requirements are not too great, either network should suffice. However, Glove-TalkII currently implements on a multi-user machine, thus computing resources are not always guaranteed, causing stutters in the speech. For this reason, networks should be kept as small as possible.

Chapter 6

Alternate Strategies for Glove-TalkII

Implementing Glove-TalkII basically focuses on developing a technique for mapping gestures to formants. This chapter discusses four alternate techniques which map hand gestures to speech at the formant level. The first section introduces the mapping problem in general. Subsequent sections discuss each of the four methods: the fixed mapping method, the single feed forward network approach, the principal component method, and the constrained auto-encoder method. The final Glove-TalkII system uses a combination of the first two methods along with other techniques described in section 4.

6.1 The General Mapping Problem

In its simplest form, the Glove-TalkII mapping problem can be reduced to the following problem:

Find a mapping between the 35 input parameters [1] and the 10 formant parameters.

Unfortunately, the more interesting expressions of this problem are the most complex. First, the 35 hand parameters are not independent; therefore, only mappings that use the subspace of the 35 dimensional hand space corresponding to the user's hand abilities are useful. Similarly, formant parameters values exist that do not correspond to any speech and the mapping should not allow trajectories in gesture space which should produce speech pass through these non-speech areas. Of course, the mapping from hand space should be exhaustive in the sense that all of formant space which produces speech can be reached. In addition, the mapping should never require users to exceed their maximum hand movement speed to produce certain speech sounds. The danger lies in large formant transitions during burst consonants. A simplistic mapping would require the hand to move too quickly. Furthermore, the mapping should be "natural". A natural mapping would involve two components: first, what the user thinks is a good mapping between gestures and sound produced, and second, when the user has no preconception about what is a good mapping, to produce an easy one to form a cognitive model of. The final Glove-TalkII system version accomplishes the necessary features by learning a carefully designed initial mapping from examples provided by the user.

It should also be possible to find an inverse of the mapping between hand gestures and speech, i.e. given some speech, the mapping should be able to determine a gesture necessary to produce the speech. Using the inverse, the user can see a graphical display of the gesture required for a desired utterance. Without this inverse mapping the user only has the forward model to learn from which means she basically has to move her hand around to discover what noises occur, and then, adjust his or her movements to make these noises sound like speech. As with the final Glove-TalkII system, a cognitive model may exist to assist the user learning the mapping. A description of a technique to invert the final Glove-TalkII system is described in chapter 7.

Designing the adaptive gesture-to-formant mapping is like designing a musical instrument. The sounds desired are already known; what remains is the discovery of how to "play" the instrument to produce the sound/speech with minimal training. Typically, musical instruments go through several iterations before stabilizing to a usable form. Here, the mapping will be learned automatically based on the characteristics of the user's gestures *and* the mapping will be exhaustive.

Finally, the mapping must be computable in real-time. If not, the audio feedback from the user's gesture will be delayed, making an already difficult learning task much harder.

[1] The hand parameters include: 18 right hand parameters from the Cyberglove, 6 3-space parameters from the polhemus device, 10 left hand parameters from the VPL Data-Glove and the 1 depression angle of the foot pedal (also referred to as a hand parameter in this context).

The next sections discuss four alternate techniques to map hand gestures to formant parameters besides the one used in the final Glove-TalkII system. The first method is the fixed mapping method, the second is the single feed-forward network method, the third is the principal component method, and the fourth is the constrained auto-encoder method. Notice that the final Glove-TalkII uses some fixed mappings in addition to feed-forward neural networks.

6.2 Fixed Mapping Method

A simple method for mapping gestures to formants is based upon the designer's intuition. To create this mapping the system designer uses his intuition to determine what the mapping should be. He programs this mapping into the computer. Any preconceived notions the designer has about how the mapping should be are automatically incorporated into the system. Such a system is non-adaptive and once built cannot change automatically to suit the user. On the positive side, the designer has an established cognitive model that can be communicated to the user.

For example, English vowels can be mapped approximately onto a plane with the two dimensions of front tongue position and back tongue position [22]. The formants F1 and F2 determine the vowel sounds to first order. The obvious mapping here would have a two dimensional grid with the grid positions corresponding to tongue positions. These positions have an associated F1 and F2 value. Now, the user only needs to know the correct tongue positions for producing vowels to determine the appropriate hand positions.

Such a method has two main disadvantages: first, it is non-adaptive, which forces the user to learn the mapping *as is*; second, once part of the mapping is fixed it may be impossible to form another part. For these reasons, it becomes very difficult for a designer to make an exhaustive map and at the same time keep a systematic approach to the design. For example, the design above does not easily allow consonants to be produced without interfering with vowel sounds. Fortunately, however, when one part of the mapping becomes fixed, the space of possible alternatives for other parts reduces, making the search easier. The benefits of the fixed mapping method can be separated from its drawbacks by integrating some fixed mapping techniques and some adaptive mappings. For example, the Glove-TalkII system identifies obvious parts of the mapping and fixes them and uses the adaptive techniques in the reduced mapping space to determine the rest. In fact, after preliminary investigations, the foot pedal was added to control overall amplitude of the formants with a fixed relationship. Clearly, care must be taken not to restrict too much mapping space with fixed parameters to allow the adaptive method to play its role in determining the interface.

Glove-TalkII uses several fixed mapping strategies. As mentioned, the foot pedal controls the overall amplitude by using the functional relationship expressed in equation 4.2 in section 4.2.2. Glove-TalkII also uses a fixed mapping between the height of the hand (measured along the Z axis by the polhemus device) and the fundamental frequency (F0), by using the relation in equation 4.1 in section 4.2.2. The vowel space mapping is closely related to the fixed mapping method. The mapping is predefined and not easily adjusted from user based examples. A network learns the predefined mapping from artificial examples. The examples can be altered easily to give a tailored vowel space.

6.3 Feed Forward Model Method

A simple network approach to getting an appropriate mapping uses a feed forward network with input units representing the user's hand parameters and the output units representing the formant parameters simultaneously. This approach still requires a predefined mapping between gestures and speech sounds, but the role of implementing the mapping is left to the network as it trains.

This technique was tried with a single 32-15-9 feed-forward network with sigmoid units. The inputs were X and Y values at time step t and three time steps of the 10 hand parameters used in the V/C network (see section 5.1) averaged in the same manner. The training data were similar to the data used to train all three networks in the final Glove-TalkII system. However, the vowel data collected for the V/C networks were combined with the artificial data generated for the vowel network. The consonant training data had random X and Y values added to them since consonant production should be possible regardless of where the hand is located in the X-Y plane. After training, the network was used in Glove-TalkII. Speech produced with this network is similar to that produced by the final Glove-TalkII system. One major drawback emerged: significant interference between vowel and consonant sounds. For example, when the user wants to make a vowel-consonant-vowel (VCV) transition her normal movement starts at the initial vowel sound, followed by the consonant hand shape, and proceeds to the final vowel sound. When the user makes the consonant sound she moves her hand from the initial vowel sound to the final vowel sound in the X-Y plane. The problem is that consonant quality changes while her hand moves through the X-Y plane. This produces non-speech like artifacts in her speech. The final Glove-TalkII system overcomes this problem by using the V/C decision network to make vowel and consonant sounds independent[2].

[2] This is not to say that in normal speech consonant sounds are unaffected by vowel production and vice-versa (which they are, due

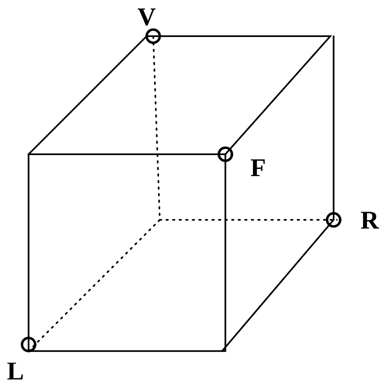


Figure 6.1: Possible structure for four phonemes constructed by an auto-associator network; each phoneme is at the corner of a hypercube with an unimpeded path between each pair of phonemes.

A *brain-state-in-a-box* type neural network architecture (i.e. [1] [30]) could be used instead of a single feed forward neural network with sigmoid units (this idea follows from discussion with M. Mozer (1993)). In this case, the goal of the network would be to construct a hypercube with each of the static phonemes represented as activations on the corners. The key idea is that a path exists between two phonemes that never crosses another path between two other phonemes in the hypercube. Figure 6.1 shows an example structure for four phonemes; L, V, R and F. Notice that an unimpeded path exists between each pair of phonemes.

Once trained, this system would produce the static formant parameters as outputs for each phoneme, depending upon which corner of the hypercube the user's hand shape corresponds to. When the user moves to a new hand configuration, the sound changes to the new static phoneme. The transition would be abrupt. This mapping corresponds exactly to a phoneme level method as described in the spectrum of gesture-to-speech mappings (see section 3.1). One problem gesture-to-phoneme mappings have is that static phonemes concatenated together produce poor quality speech. This point is particularly important to designers of text-to-speech systems who spend considerable effort creating rules to join phonemes together to produce natural sounding speech. Currently, these systems [23] [33] still produce robotic sounding speech.

Several modifications to this technique can be imagined. First, a low pass filter could be used to smooth the transitions between the formants. This would be very similar to the linear interpolation performed between phonemes used in many text-to-speech synthesizers. Second, trajectories between phonemes could be learned which correspond to paths through the hypercube. The main difficulty with this strategy occurs when paths between different phonemes are shared (i.e. V to F may share part of the path for L to F). The transitions between different phonemes may not have the same properties and the average transition may be wrong too. An extreme case of such behaviour could occur if transitions between two phonemes were asymmetric (i.e. F to L is different than L to F). Interestingly, this technique may be very useful for other gesture recognition applications where configuration transitions are not required.

Of course, many other network structures can be devised for gesture-to-speech mapping. This section described only two possibilities.

6.4 Principal Component Method

Another technique for mapping hand gestures to formants uses a mapping between principal components of the two spaces. Representative hand data and synthesizer data determine the principal components of both those spaces. Notice that this technique does not require a pre-defined gesture-to-speech mapping; instead it produces a mapping automatically. The principal components are ordered by the eigenvalues of the correlation matrix from each of the two spaces. The ordering determines the mapping between the two spaces: the n most significant principal components of the hand space are mapped one-to-one to the n most significant principal components of the formant space. To use the mapping the sequence is as follows:

1. Get new vector of hand parameters and scale by means and variances of collected hand data.
2. Project parameters onto first 10 principal components in hand space.
3. Use the 10 magnitudes of the projections as the 10 magnitudes of the principal component vectors of formant space (in order of importance).

co-articulation effects); however, if there is too much interference or unnatural interference the speech produced becomes unintelligible.

4. Reconstruct formant parameters from the vector of magnitudes, matrix of formant principal components, and then scale with the means and variances of the training set.

This can be expressed by the following equations:

$$Q_f \frac{(f - \overline{f})}{{\sigma_f}^2} = Q_h \frac{(h - \overline{h})}{{\sigma_h}^2} \tag{6.1}$$

$$f = \left(Q_f^T Q_h \frac{(h - \overline{h})}{{\sigma_h}^2} \right) {\sigma_f}^2 + \overline{f} \tag{6.2}$$

$$\tag{6.3}$$

where f is the vector of formant parameters sent to the speech synthesizer at time t, determined from h, the vector of hand parameters at time t. Q_h is an orthonormal matrix of the 10 most significant eigenvectors determined from a sample of hand parameters. Q_f is an orthonormal matrix of the 10 eigenvectors from a sample of typical formant parameters. $\overline{f}$, ${\sigma_f}^2$ and $\overline{h}$, ${\sigma_h}^2$ are the means and variances determined from the sample of formant and hand parameters respectively. Notice that both ${\sigma_f}^2$ and ${\sigma_h}^2$ are really diagonal scaling matrices; however, they are indicated as vectors for ease of understanding.

The above mapping procedure assumes a single time step of data. The chosen system uses only the right hand parameters providing 16 principal component vectors[3]. Only 10 principal component vectors exist in formant space; hence, only the first 10 principal components of hand space are used. A single time step will not allow for any mapping of the temporal characteristics of the hand to similar ones in formant space. Rather, the dimensions of high variance in hand space are mapped to dimensions of high variance in formant space, taking no account of whether successive time steps are coordinated or not. In an attempt to avoid this difficulty, in another trial, pairs of successive time steps of data were also used to generate the principal components. In this scenario, there are 32 dimensions for hand space and 20 dimensions for formant space. Further, up to the first 20 principal components can be used in the mapping, depending upon the computational resources available.

Attempts using the single time step principal component method provide some insight. First, a single time step principal component analysis of 60 seconds of hand data (collected from random hand movements using a right handed VPL DataGlove and a polhemus [only 16 parameters]) revealed that the first 10 components account for most of the variance (95.7%). During data collection the user moved his hand in as many different configurations as possible, each without discomfort. Another interesting feature is that the most significant principal component vector of the hand has a large component assigned to the little finger. This suggests another problem with this method. The principal component assigns highest weight to parameters with high variance, but these are not necessarily parameters with a high degree of control. For example, the first principal vector's large component in the direction of the little finger causes the little finger to exert a significant amount of influence over the speech. However, the little finger is not very controllable even though it's value has high variance. In addition, the 10 principal components of formant space (obtained from 132 seconds of text) were relatively equal in magnitude, with the ratio of the largest to smallest being about 550. Speech-like sounds are possible with this mapping, however, it is impossible to coordinate movements to produce intelligible speech.

Since the principal component technique automatically determines a forward mapping, the inverse mapping is invaluable to relate the mapping to the user. The inverse allows the user to display hand gestures graphically for given utterances which she can then try to mimic. Without the inverse mapping the user must discover the hand-to-speech mapping through trial and error. Getting the proper hand parameters from a series of formant parameters is straight forward; just invert the forward linear mapping. The equation is

$$h = \left(Q_h^T Q_f \frac{(f - \overline{f})}{{\sigma_f}^2} \right) {\sigma_h}^2 + \overline{h} \tag{6.4}$$

Using the inverse mapping, hand gestures were determined for the simple utterances "hello". The gestures determined required extremely complex hand movements and configurations. Some of the static configurations were impossible. Additionally, the transitions between gestures were often extreme and impossible to perform in 1/100 second. The two time step principal component method produced a different mapping with similar difficulties with the inverse.

The principal component method relies on representative training data from both the hand gestures and the formant parameter trajectories. As mentioned in sections 6.3 and 6.5, the best way to get representative gesture data remains an open question. Formant tracks can be obtained either from real speech or from text; the former

[3]When this system was tried, only a right-handed VPL Data-Glove and polhemus were available, providing 10 finger angles and 6 3-space coordinates.

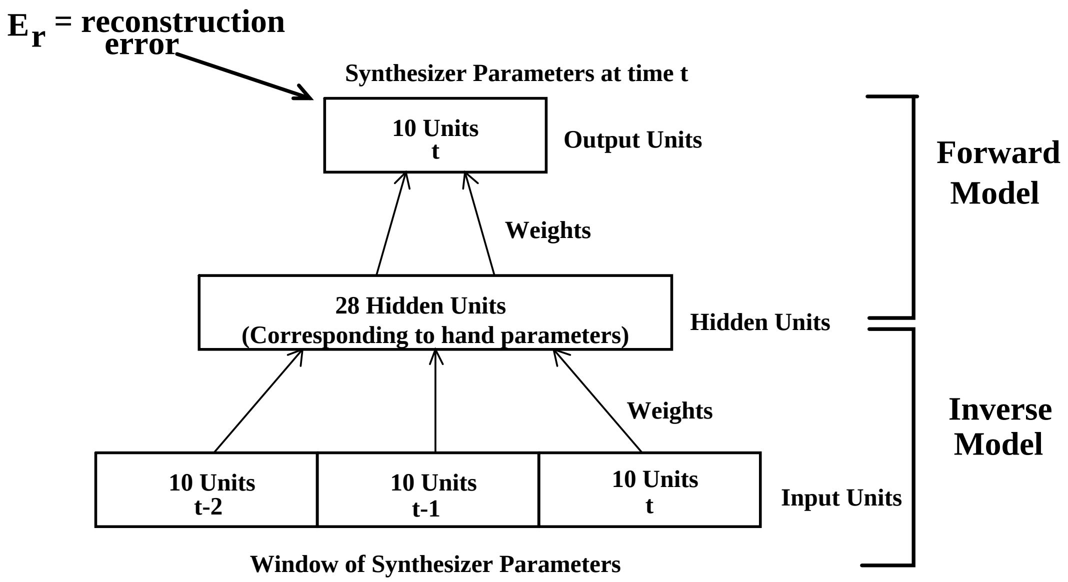


Figure 6.2: Auto-encoder for finding hidden code of synthesizer parameters.

being good sounding but hard to get for significant quantities of speech, and the latter easy to obtain for a diversity of speech, but sounding robotic[4]. Further, there is no guarantee that the principal component mapping produced will be exhaustive since some of the principal components of the gesture space are omitted. To further complicate matters, the mapping may require an impossible hand configuration to produce some necessary speech sound. Finally, no constraint exists which tries to make the gestures "natural" for the user to produce speech.

6.5 Constrained Auto-encoder Method

The mapping between gesture space and formant space is many to one. It is possible to choose a *particular* mapping by using an auto-encoder network [19]. Here the input units are clamped to a temporal window of synthesizer parameters; the hidden units correspond to the gesture parameters and the target outputs are clamped to the current values of the formant parameters. Basically, an inverse model of the mapping forms from the input units and the hidden units, which are in turn used as the inputs to the forward model of the mapping. Figure 6.2 shows a block diagram of such a network scheme; however, neural networks are not essential to implement the forward or inverse models as indicated in the diagram. The key points to this scheme are that both the inverse model and the forward model can be differentiated and that the derivative can be calculated. Neural networks provide a convenient mechanism for this operation. The auto-encoder trains on formant data obtained either from text or real speech. Once the network is trained, the user only has to make gestures corresponding to the activations of the hidden units. The forward model maps them to correct formant values. Similarly, if the user wants to know the trajectory of hand parameters needed for some utterance, the inputs to the inverse model are clamped to the formant parameters of the speech every time step and the model determines the appropriate gesture.

If only the system in figure 6.2 is used, the code produced for hand parameters will have nothing to do with what the user's hand can do. Rather, the coding will be a function of the statistics of the formant data and the initial weight vector. This situation can be avoided. By adding an extra error term (E_h) to capture the constraints of the hand, the auto-encoder becomes able to choose a constrained hidden unit code representative of what the user's hand can do, and at the same time, produce the correct mapping to the formant parameters (see [38] for related work). Figure 6.3 shows the same auto-encoder as in figure 6.2, except an extra error term is injected into the code units to put pressure on the units to behave over time like a user's hand does. What kind of error term captures the constraints of a user's hand? In the following subsections three different possible error functions based on a model of the user's hand are considered to answer this question. The first is based on an analytic model of the hand, while the latter two are based on data obtained from the user. Initial attempts to implement Glove-TalkII were made with the first and third error terms. The second method is computationally complex and has not been investigated.

[4] See section 8.1.1 for more details about automatic formant extraction.

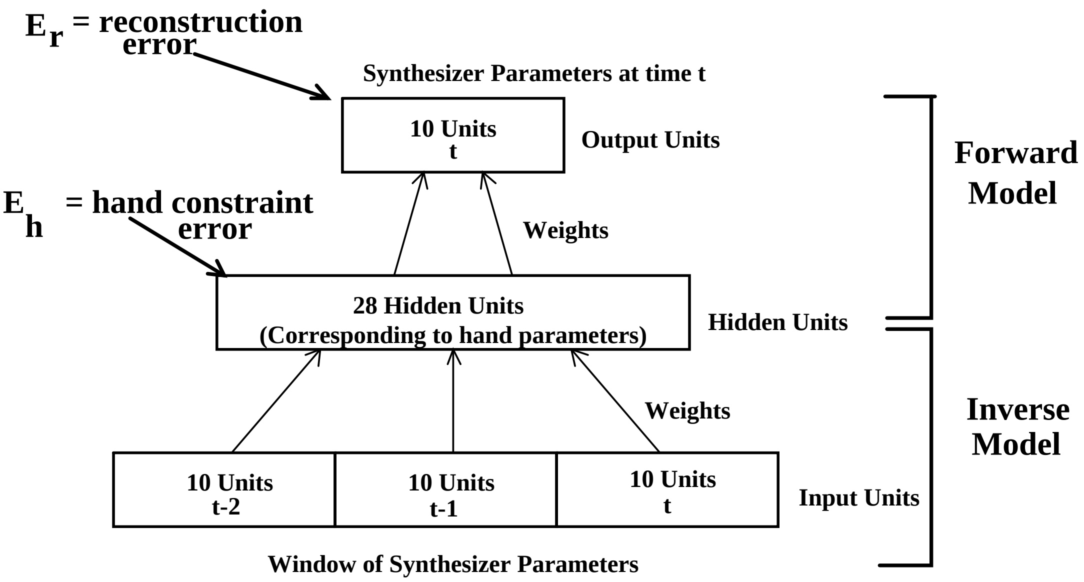


Figure 6.3: Auto-encoder for finding **constrained** hidden code of synthesizer parameters. The constraints are determined from a model of the user's hand.

6.5.1 Getting a Model of the Hand

One very important assumption has been made with regard to modeling the hand using the constrained auto-encoder. The assumption is that some reliable way exists to get data from a user that truly represents the capacity of his hand. It is not sufficient to ask a user to move his hand randomly for some period of time and hope he gives a representative sample of all possible hand trajectories. As any musician can attest, with training and practice our hands can move in ways not previously experienced. While these seemingly odd or different movements are not "natural" immediately, with practice they become automatic – though some motions are harder to make than others and may be relegated only to produce uncommon sounds.

One way to collect example data is to ask the user to make various predetermined trajectories. The trajectories may consist of making fixed target hand shapes in succession. The target hand shapes do not actually have to be attainable with the user's hand, since only the constraints of the hands are of interest; specifically, the critical constraints are discovered precisely when the user cannot reach a particular target. Presumably, several repetitions of this task could capture the variability between target gestures. Coupled to this technique would be a task that asks the user to make random hand movements. This idea relies on the assumption that the user will make the easiest hand movements (for them) during this training phase. These movements become important when the auto-encoder learns the best forward model. It is wise to have the easiest (i.e. most frequent) hand movements mapped to the most frequent combinations fo formant parameters.

The technique used in the final Glove-TalkII system can be used to capture constraints of the user's hand. The user hears an utterance and then makes a gesture which she thinks should represent that sound. This type of data captures the user's cognitive model of the gesture to speech task while providing hand constraints. The information gained can then be used to pressure the activations of the code units to correspond to the user's suggestions during the production of the target utterance. Currently, this type of information has not been used in the auto-encoder method, but it would not be difficult to add an extra term to the hand error term (E_h), which also puts pressure on the activation of the code units directly. For example, the term

$$E_h^{traj} = \frac{1}{2}({h_t}^{\alpha} - h_t^{act})^2 \tag{6.5}$$

would put the necessary pressure on the activations of the hidden units. In this term, h_t^{α} is the value of the hand parameter in the training example at time t, which corresponds to the value the user produced for the formant parameter at time t, and h_t^{act} is the activation of the hidden unit at time t. Of course, some type of temporal alignment of the training data would be required. A dynamic time warping algorithm could be used for this purpose [32]. Notice, the auto-encoder method using this scheme approximates the final Glove-TalkII method since the hand parameters are specified for each target formant trajectory.

Another possibility, once representative hand data is obtained, is to create a model of the hand. The hand model will determine the hand constraint term, (E_h). Three models are discussed in the subsections below.

Crude Prior Model

The first model is one that does not depend on any training data. This model captures the velocity constraints of the hand and fingers. From observations of the hand, it is postulated that hand parameters should change relatively slowly over time. The error term which captures this idea is of the form

$$E_h = \frac{1}{2}(h_t^i - h_{t-1}^i)^2 \tag{6.6}$$

where h_t^i is hidden unit i's activation at time t, and likewise, h_{t-1}^i is that unit's activation at time $t-1$. Basically, this error term penalizes activations that differ widely in successive time intervals. [19] used a similar error term to constrain torques applied to a robot arm with some success; however, training proved difficult (personal communication).

Tests were made using the error function in equation 6.7

$$E_h^* = \frac{Variance(h_t^i - h_{t-1}^i)}{Variance(h_t^i)} \tag{6.7}$$

This function attempts to make adjacent hand parameters (in time) the same, while also pursuing a large dynamic range for each hand parameter. Training the network was very difficult. It was difficult to get the scale of the E_h^* term to match the error term E_r from the outputs. The difficulty arose because although the errors are of the same scale, they are introduced at different parts of the network. Further, this analytic model is only an approximate model of the hand constraints, and does not capture the fact that some movements can be done more quickly than others. As well, there are co-articulation effects between finger movements not represented by this model, allowing for example, the middle finger to make a downward movement while the ring finger makes an upward movement. Such a combination of motions are exceptionally hard to make even at a slow rate.

Pairwise Constraint

Another technique for getting an error function derives from a model of the hand where successive pairs of examples serve as an empirical hand model. In this model, each successive pair of examples defines a mean of a normal distribution. All the normal distributions together determine a probability distribution over pairs of hand parameters. It then becomes possible to maximize the probability of the hand parameters coming from the auto-encoder by adjusting the activities of those units. Symbolically, we want to minimize this error function:

$$E_h^{pc} = -log(P(h_t^{act}, h_{t-1}^{act} | h_t^{training})) \tag{6.8}$$

where the h_t^{act} are the activations of the hidden units at time t, and $h_t^{training}$ are the values of the hand parameters in the training set. Notice pair wise terms are used to capture the interaction between successive hand configurations.

The main problem with this error function is that calculating the error for a single time step of hand parameter codes means iterating through the entire example set. Thus, a single gradient calculation for the formant training set takes $O(MHW)$, where M is the number of formant examples, H is the number of hand parameter examples, and W is the number of weights in the inverse model. Assuming that $O(W)$ gradient calculations are required, the total algorithm takes $O(MHW^2)$. M and H are often as big as W, making the algorithm $O(W^4)$. This is very expensive computationally. For these reasons, this method was not investigated. However, the possibility of this method motivates the next technique.

There are some short cuts for approximating the error E_h^{pc}. For instance, only gaussians with centres close to the new point will have much influence. By keeping track of only the neighbours of pairs of hidden unit activations in the training set, one can reduce the number of computations necessary for computing the error derivatives.

Basically, this model of the hand tries to use the example data to predict whether a new data point is compatible with the training data. If not, the point moves into a region where it becomes feasible. If a cluster of example points exists (suggesting the hand can move freely around that space) the hand code will try to be near the centre of mass. If there is only one point near by, then the code tries to move to that gaussian's mean (i.e. to get very close to that isolated example point). Due to the complexity problem, this is not an elegant solution. Ideally, we want a model that tells whether a particular hand code is likely or unlikely, given the example data. The next section describes a network which attempts to learn such a model.

Network Model of Hand

The final constraint method uses neural networks to determine a model of the hand. This hand model provides an error function to pressure the hidden units of the autoencoder to form "good" codes. With this technique a

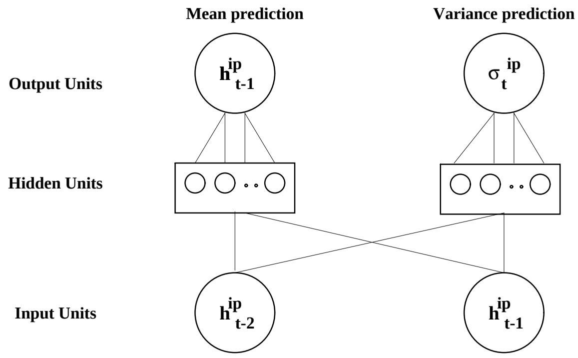


Figure 6.4: A Predictor Network: two coupled networks, one which predicts next sample and another which predicts variance of prediction. Since each of the hand parameters is assumed independent in this discussion, there is one predictor network per hand parameter [6]. A variation of this basic predictor network predicts the logarithm of the standard deviation. This will simplify the derivative of the error function as well as making the quantity dimensionless, and ensuring the variance is positive.

neural network predicts the next value of a hand parameter, based on the previous two values. A coupled network also predicts the variance of the predicted value. Predictions with a small variance indicate that the user's hand is inflexible at this point; therefore, the auto-encoder should make the code very close to the prediction. A high variance prediction, on the other hand, suggests the hand can take on many different values at that point in time; hence, the auto-encoder is free to select any code that works well. Some questions considered with this technique include how to train it, how to integrate it into the auto-encoder network, what is the interaction between learning to predict a value and its variance, and what the interaction is between hand parameters.

To illustrate the network hand model, consider the diagram of the predictor network (two coupled networks) in figure 6.4. The network predicts the next time step of a hand parameter and the variance of that prediction. The model assumes that each hand parameter is independent. On the left of the diagram, a network predicts the next value (h_t^i) of hand parameter (i) from the two previous ones, (h_{t-1}^i) and (h_{t-2}^i). The other network predicts the standard deviation (σ_t^i) of the parameter prediction. Of course, more than two time steps can be used as well as considering context from other hand parameters. In this discussion only two time steps of hand parameter values are used in the predictor network. If necessary, more context can be included to aid prediction, for example, more than two time steps of hand parameter i can be used as past time steps of other hand parameters. Extension to the model should be straight forward. Training the predictor network is interesting in its own right. Discussion follows in the next section.

Suppose the predictor network trains to predict the data correctly and to determine the variance of that prediction. How can the predictions be integrated into the auto-encoder for determining hand codes? Figure 6.5 shows an auto-encoder network with the predictor networks added on. The appropriate error function (E_h) needed to constrain the code units maximizes the probability of generating the next hand parameter (or minimizes the negative log of the probability), given the prediction of that parameter and its variance. Mathematically, for one hidden unit this is

$$minimize\ E_h^i = -log(P(h_t^i|h_t^{ip}, \sigma_t^{ip})) = -log(\frac{1}{\sqrt{2\pi}}) + log(\sigma_t^{ip}) + \frac{(h_t^i - h_t^{ip})^2}{2{\sigma_t^{ip}}^2}, \tag{6.9}$$

where h_t^i is the activation of hidden unit i at time t. The predicted value of the hand parameter i at time t is h_t^{ip} and its standard deviation is σ_t^{ip}, obtained from the prediction network (as a function of the two previous activations of hidden unit i). This function assumes that the distribution of activations are normally distributed with mean and variance determined by the predictor network. Since we assume that hand parameters behave independently, one predictor network is assigned per hand parameter (i.e. each code unit has one predictor network attached to calculate E_h). To do the minimization (using a technique such as steepest descent or conjugate gradient), it is useful to have the derivative of the error with respect to the activities of the units, so that backpropagation [29] can be used to get the gradient of the error with respect to the weights. The derivative can be easily obtained even though the values of h_t^{ip} and σ_t^{ip} are functions of the previous values of the hand parameters. The partial derivative of the error function with respect to the activity of hidden unit i at time t is

$$\frac{\delta E_{h_t}^i}{\delta h_t^i} = \frac{(h_t^i - h_t^{ip})}{{\sigma_t^{ip}}^2}. \tag{6.10}$$

The other derivatives ($\frac{\delta E_{h_t}^i}{\delta h_{t-1}^i}$ and $\frac{\delta E_{h_t}^i}{\delta h_{t-2}^i}$) are simply obtained by backpropagating the error through the prediction

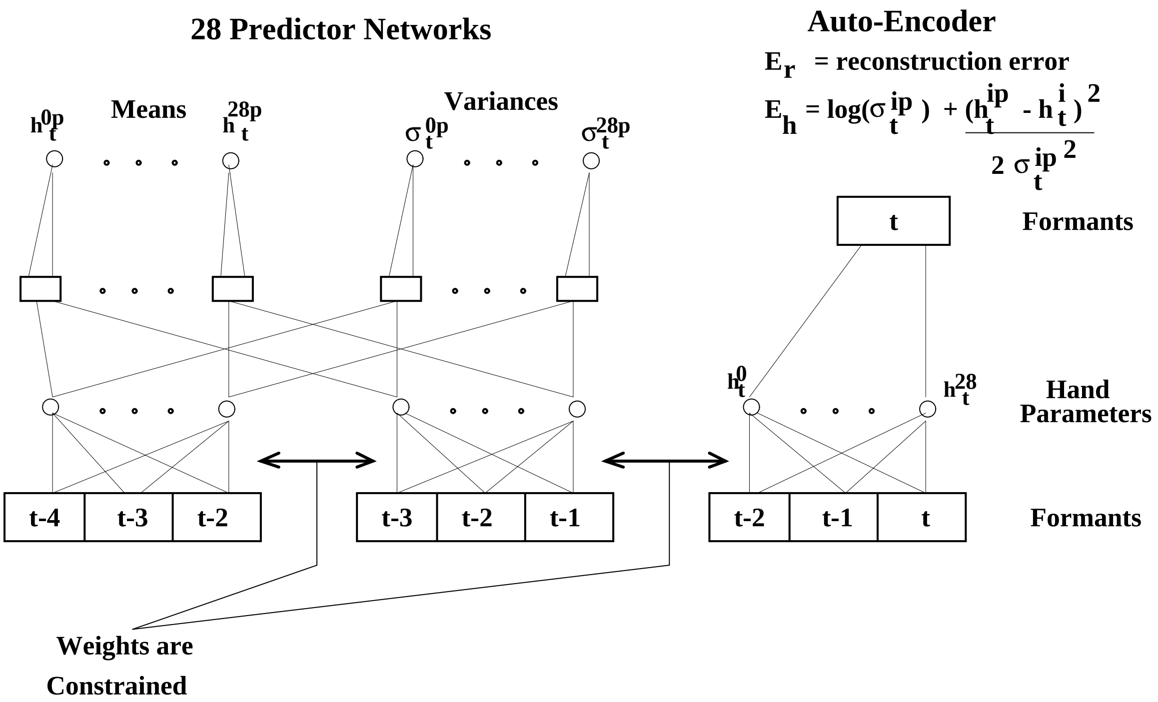


Figure 6.5: Complete constrained auto-encoder: a predictor network for each hand parameter constrains the hidden codes in the auto-encoder. The predictor networks are trained to get a model of the user's hand. The auto-encoder network minimizes the reconstruction error and the hand model error E_h. Notice the derivative of E_h with respect to the weights is a function of the formant parameters at $t-1$ to $t-4$. These derivatives are obtained by backpropagating the hand error (E_h) through the predictor networks and keeping the inverse model weights constrained.

network. As long as the activities and weights for the previous time steps of the auto-encoder are stored, the gradients of the error with respect to the weights for the two time steps back can be added in correctly for the weight update. Being able to back-propagate the error through the predictor network is a great advantage. Remember, the predictor networks are fixed during the training of the auto-encoder. These predictor networks must be trained too. The next section describes some of the details involved with training the predictor networks.

Training the Predictor Networks

Each of the predictor networks is responsible for modeling one hand parameter. Each predictor network (which is actually two coupled neural networks) has as its input two time steps of hand parameter (i) obtained from a training session (see sections 6.3 and 6.5). Each predictor network's output consists of the predicted value of the hand parameter at the next time step and the variance of that prediction. Assuming that the prediction error distributes normally, deriving an appropriate error function to train the network becomes easier. One such function minimizes the negative log probability of generating the actual hand parameter value at time t given the values at $t-1$ and $t-2$. Algebraically this is

$$E_{hm} = -log(E'_{hm}) = log(\sqrt{2\pi}) + log(\sigma_t^i) - \frac{(H_t^i - H_t^{ip})}{2{\sigma_t^{ip}}^2}, \tag{6.11}$$

where E_{hm} is the error of the hand model, H_t^{ip} and ${\sigma_t^{ip}}^2$ are the predicted values of hand parameter i and the variance of the predicted value at time t respectively, H_t^i is the value of the hand parameter at t, and H_{t-s}^i is the value of hand parameter i for times when $s = 1, 2$.

Again the partial derivatives (with respect to the activity of the output units) are useful for getting the gradient of the error function (with respect to the weights) using back-propagation. The partial derivatives are these:

$$\frac{\delta E_{hm}}{\delta H_t^{ip}} = \frac{(H_t^i - H_t^{ip})}{2\sigma_t^{ip}} \tag{6.12}$$

$$\frac{\delta E_{hm}}{\delta \sigma_t^{ip}} = \frac{1}{\sigma_t^{ip}} - \frac{{(H_t^i - H_t^{ip})}^2}{{\sigma_t^{ip}}^3} \tag{6.13}$$

It has also been suggested by [25] that using $log(\sigma_t^{ip})$ as the representation for the standard deviation instead of just σ_t^{ip} may be beneficial because the logarithm makes the output represent a dimensionless quantity.[7] At this point the importance of the scale of the output units for the predictor networks should be stressed. If the training data is not prescaled, then neither the prediction of the output nor the variance have a predetermined range. Therefore, it is wise to use a linear activation function for the output units, to allow the network to scale properly. One subtle point regarding the above discussion: notice the error function E_{hm} leads to a biased estimate for the variance because the maximum likelihood estimate of the noise is not the most probable value of the noise [24]. This factor becomes problematic when the data is sparse. At these points the estimates of σ_t^{ip} tend towards zero. This is probably undesirable. An unbiased estimate value will tend towards infinity, which may be desirable. Unfortunately, getting an unbiased estimate requires more computation. Theoretically, if some measure of the sparseness of the data were available, it would be possible to force the estimates of the variance to be unbiased. Mackay [26] presents a technique for getting such a measure and its application can be found in [24]. Unfortunately, to be useful, his method requires the hessian (with respect to the weights) of the error function. Arguably, a biased estimate of the variance is still useful. The finite number of weights in the network act as regularizer making sparse points have non-zero variance estimates. As well, by placing weight constraints on the network, further regularization will keep the variances non-zero.

Initial trials with the auto-encoder and the predictor networks were unsuccessful. The output variances from the predictor networks were all about the same which did not provide enough pressure to form hand configurations corresponding to what the user's hand is capable of. While this technique was not fruitful initially, more research should be performed to explore fully its potential.

To conclude this section, let us remember the main goal of the predictor network. The auto-encoder chooses one of the many-to-one mappings available between hand parameter space and formant space. By placing constraints on the code units, the mapping selection can reflect what the user's hand can do. If the user has provided a training set of hand gestures that is representative of his range of motion, the hand model networks will capture the constraints of his hand by predicting future hand movements from previous ones. When the prediction is very wrong (as indicated by a high prediction variance), the auto-encoder has a wider choice of the hand parameter value mapped to the current formant parameter. The predictor networks train on the training data provided by the user. The weights are then fixed and the auto-encoder trains on the formant data. The partial derivatives of the hand model are obtained by using the outputs of the predictor networks and the backpropagated errors through the predictor network.

6.5.2 Some Remarks on Alternative Mapping Techniques

Currently, it remains unclear which method best produces the mapping between hand gesture and speech. Investigation of these alternative methods led to the final Glove-TalkII system described in section 4. This investigation was an evolutionary process which built upon the understanding gained from the successes and failures of the alternative methods. The final Glove-TalkII system is really a hybrid of some of the above methods, using the best features from each of the techniques. For example, the mapping is fixed for some of the obvious parameters, such as fundamental frequency and volume, while feed-forward networks are used for the others. Other hybrid solutions are possible. For example, within the auto-encoder paradigm one could use the principal components of the hand parameters to decouple the hand parameter codes, and use the decoupled components as the inputs to the hand model. Many possibilities for implementing a gesture-to-speech device exist and should be investigated; these same techniques may be applicable to other real-world problems.

[7] In this case, $\frac{\delta E_{hm}}{\delta log(\sigma_t^{ip})} = 1 - \frac{(H_t^i - H_t^{ip})^2}{\sigma_t^{ip\,2}}$.

Chapter 7

Inverting Glove-TalkII

When learning to speak with Glove-TalkII a user needs a teacher signal to help learn hand trajectories for particular utterances. What is needed is an inverse mapping of the forward hand-to-gesture mapping. Using the inverse mapping, a target utterance can be mapped to the required hand trajectory (including the foot pedal depression). In the final Glove-TalkII system, an iterative inverse is used to obtain an inverse mapping of target utterances.

The method for inverting the mapping for the final Glove-TalkII system may seem overly complex since the initial mapping is predefined for individual static phonemes. A näive approach might be to concatenate the hand gestures defined for each of the individual phonemes in the target utterance produced with the text-to-speech synthesizer. With this approach, static phonemes are labeled along with their corresponding formant parameter trajectories. The approach is, however, lacking in three important aspects:

1. When speaking, static phonemes serve only as targets that may not be reached in producing utterances.
2. The final Glove-TalkII system learns the initial mapping using examples from the user; therefore, the learned mapping may not correspond directly to the initial mapping.
3. "Good" transitions between phonemes are important for natural-sounding speech.

The predefined mapping does not provide information on how transitions between the static phonemes should occur. In particular, the timing of the transitions is critical for good speech. Additionally, the learned mapping provides a rich space of hand-to-formant parameter mappings which can be exploited to produce good speech. Specifically, there may be combinations of static hand gestures (corresponding to static phonemes) that can produce useful sounds for generating some utterances.

The final Glove-TalkII system defines the forward model from hand gestures to speech. An iterative inverse technique can be used because Glove-TalkII uses neural networks and invertible fixed mappings. The structure of the forward model is shown in figure 7.1. The iterative inverse approach used is as follows:

1. Begin with an initial hand-trajectory estimate for the target utterance.
2. Forward propagate the hand-trajectory and compare with the target utterance.
3. Back propagate the formant parameter error through the forward model to the inputs (i.e. find the change in the output error with respect to the inputs).
4. Change the inputs to lower the error (i.e. optimize the input parameters using the gradient of the error with respect to the inputs)
5. Repeat from step 2 until convergence.

The one-to-many property of the inverse mapping presents difficulties. That is, for each particular utterance a family of hand-gestures exists which can produce it[1]. From the user's perspective, however, only "do-able" hand trajectories are meaningful. Ideally, the hand trajectories determined from the inverse should be as intuitive as possible. Specifically, they should correspond to the initial mapping whenever possible.

One solution to overcome the one-to-many mapping problem constrains the mapping so that only meaningful inverses are selected. Four different constraints are considered below: temporal neighbourhood, initial parameters, V/C decision and fixed volume.

[1] Similar to the ventriloquist effect with speech.

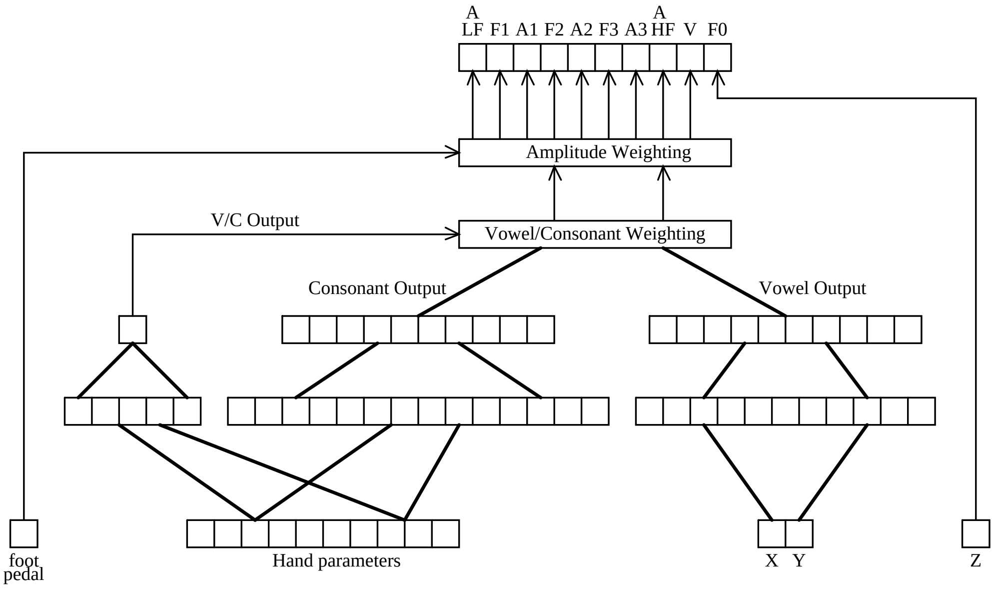


Figure 7.1: The complete forward model from the final Glove-TalkII system maps hand gesture to speech. The iterative inverse of this model finds hand gestures for target utterances. The V/C network is discussed in section 5.1; the consonant network is described in section 5.3, and the vowel network is described in section 5.2. The V/C weighting function weights the outputs of the vowel and consonant networks by the probability of the current hand configuration being a vowel as indicated by the activity of the V/C network. The amplitude weighting function weights the formant amplitudes (ALF, A1, A2, A3, AHF) as a function of the foot pedal depression. The height of the hand (Z) controls the fundamental frequency of the speech (F0).

The first constraint discussed imposes a temporal constraint on the hand parameters. In particular, the squared difference between successive hand parameters is minimized. In equation form, the constraint added to the formant error is this:

$$E_{hp} = 0.5(h_t^i - h_{t-1}^i)^2 \tag{7.1}$$

where h_t^i is the value of hand parameter i at time t. This constraint quantifies the notion that neighbouring hand configurations are similar. This constraint is the same as one of the constraints used in the constrained auto-encoder discussed in section 6.5.

A second constraint considered constrains the network by initializing the inverse with hand parameters that are intuitively close to a good inverse. Two possible initializations are considered: randomly selecting values near a neutral hand configuration; and using an initial guess recorded from the user. In the latter case, the hope is that the network will improve the user's guess where needed, leaving as much unchanged as possible.

Using the initial guess from the user poses some problems of its own. First, the speech produced by the user's hand trajectory must be time aligned with the target utterance. Dynamic time warping is used to do the alignment [32]. The warping is done asymmetrically in that each hand trajectory maps to some formant parameters but not vice versa. The dynamic time warping algorithm uses no slope constraints. Recording the hand trajectory is slightly more complicated compared to data collecting for training the forward mapping. This is because the user's hand must move to perform the desired utterance unlike performing a static gesture. In addition, the user must make the gesture within the time allotted for the target utterance (usually set to 2 times the length of target utterance). Further, currently the user has no audio feedback while attempting to say the utterance. Lack of audio feedback did interfere with the collection of good initial guesses and affected results. Many attempts at providing initial guesses helped circumvent the problem. After data collection, the guesses were played through the forward mapping and the best sounding one was chosen. If proved viable, this time-consuming method is only recommended for particularly difficult utterances.

A third constraint considered restricts the inverse mapping by labeling the formant parameters with a vowel/consonant decision. The phoneme labels produced by the text-to-speech synthesizer are used to do the labeling. By backpropagating the V/C network error, hand configurations can be pressured to adhere to the vowel and consonant order in the target utterance. From previous considerations, this technique of implementing the constraint is not always a good strategy. During transitions between vowels and consonants the inverse must be free to select the timing of the transition to produce the best approximation to the target utterance. For example, reconstruction error should be small during the transitions at the expense of vowel/consonant decision error. Therefore, the V/C error is not included during the latter 40% of each phoneme produced from the text-to-speech mapping. This constraint attempts to maintain the sequence of consonants and vowels heard by the user. Without this constraint, the inverse mapping would be free to choose vowel sounds for consonants and vice versa as long as the formant error remains minimized. While this may produce a better approximation to the target utterance, it may not be easy to learn; the user's model of speech does not usually include vowel approximations to consonants and vice versa.

An obvious extension to this technique would label each of the target formants with the static phonemes produced by the text-to-speech synthesizer. These phoneme targets could be used to pressure the inverse mapping (via the RBF units) to adhere to the initial mapping more closely. While this approach leads to intuitive mappings (i.e. close to the pre-defined mapping), it strays from the goal of an inverse mapping free to find hand trajectories which do not correspond directly to the predefined mapping. Typically, the user tries to produce a target utterance by concatenating hand gestures according to the initial mapping. Only when this attempt does not produce the desired result, or, the user wants to "fine tune" some aspect of her speech, will she use the inverse mapping. Places where speech difficulties exist do not correspond to the intuitive mapping; this suggests that labeling from the text-to-speech synthesizer does not provide an accurate representation. Therefore, the labeling should not be used. The V/C decision error appears to be a compromise between complete freedom to choose the hand configuration and specifying the hand configuration precisely. The V/C decision error says that the hand configuration should correspond to a vowel or a consonant, but does not explicitly say which one.

The fourth and final constraint fixes the scale of the output to 1.0. This encapsulates the notion that the user does not have precise control of volume using the foot pedal. In fact, the foot pedal is typically used as an on/off device, rather than as an analog control. This is probably because very little force feedback comes from the foot pedal[2].

The next section discusses the results of trying variations of the above approaches.

7.1 Results from Inverting the Forward Model

The iterative inverse of three utterances using combinations of the four different constraints will now be discussed. The four constraints are these:

1. temporal constraints
2. initial guess from user or randomly determined
3. V/C decision target
4. fixed volume parameter

The three utterances are "hello", "of the", and "the city". Tables 7.1, 7.2, and 7.3 show the error rates on the three utterances "hello", "of the", and "the city" respectively.

The trajectories over time of the various parts of the network are also used to analyse the various inverse techniques. For each of the attempted variations, the activities of the radial basis functions, the V/C network output and the volume (scale) parameter are plotted at each time step of the utterance. Each radial basis function corresponds roughly to one static phoneme, thus, the trajectory of these units corresponds approximately to the trajectory of the hand based on the predefined mapping. Of course, the correspondence will not be straight-forward where the inverse mapping uses its excess degrees of freedom to produce different mixtures of sounds. This is especially true for vowel sounds. Figures 7.2, 7.3, 7.4, 7.5, 7.6, and 7.7 show activity-over-time for some of the various constraint combinations considered. The activations in these figures are displayed in the following order from top to bottom:

1. overall scale
2. N RBF
3. M RBF
4. H RBF

[2]There are two elastic bands attached to the base of the foot pedal which exert a restore-to-upright force, and provide a small amount of force feedback. An alternative feedback mechanism which may provide better volume control is a uni-directional gas spring with controlled flow. The rate of depression (or gas flow rate) could be used to control the volume. Such a mechanism would simulate breathing via a bellows type apparatus.

5. SH RBF
6. ZH RBF
7. S RBF
8. Z RBF
9. TH RBF
10. DH RBF
11. F RBF
12. V RBF
13. R RBF
14. L RBF
15. W RBF

16. VC output

17. A RBF
18. UU RBF
19. OO RBF
20. AW RBF
21. O RBF
22. AR RBF
23. U RBF
24. AA RBF
25. E RBF
26. I RBF
27. EE RBF

A large black square represents an output of 1.0 and the absence of black within the box represents an output of 0.0. Each column of boxes represents one time step of activations.

The formant parameters from the various inverses are played to further analyse their quality. One must listen to the inverse, since hand trajectories which have a low sum-squared error in formant parameters space do not always produce intelligible speech.

The final technique used to compare the different inverses displays hand trajectories using a graphical image of the hand. The graphical image allows step-by-step display of hand configurations over time. This tool also provides the teaching signal enabling the user to mimic hand gestures required to say desired utterances.

The text-to-speech synthesizer is used to obtain the formant tracks for each of the utterances. It also produces phoneme labels for each of the time steps of formant parameters. These are used to determine the V/C decision boundaries. As explained above, the last 40% of each phoneme is labeled as either a vowel or a consonant so that the inverse can determine the transitions unconstrained.

One subtle point not mentioned above is that stop consonants have a well defined inverse. Each stop consonant corresponds to a key-press by the left hand. In these cases (i.e. the "t" in the utterance "the city"), the iterative inverse should be free to choose the right hand configuration while a stop consonant appears in the outputs. The text-to-speech output labels identify where the stop consonants appear in the output so error can be ignored during these times.

All of the results found in tables 7.1, 7.2 and 7.3 were obtained using the final Glove-talkII system (described in section 4) along with the best networks (described in sections 5.1, 5.2, and 5.3). The 10-5-1 feed forward network is used for the V/C network. Optimization used the conjugate gradient method with a line search. Optimization terminated at 1000 epochs or if the line search failed.

Tables 7.1, 7.2 and 7.3 show that almost all the inverse mappings result in a low total squared error in an absolute sense. Inverses obtained without temporal constraints make up the significant exceptions. The remaining exceptions are variations 3 and 4 on utterance "hello", variation 2 on utterance "of the" and variation 2 on utterance "the city". The errors on these latter cases are relatively small.

Closer inspection of the cases without temporal constraints reveals that the inverse is very sensitive to the V/C network. The main problem occurs when the initial guess causes the V/C network to indicate the wrong sound type, i.e. when it saturates to 1.0 indicating a vowel sound when it should indicate a consonant sound. In these cases, all the backpropagated error from the formant outputs gets sent to the wrong network, since it is gated by the V/C network. For example, in a case where the V/C network outputs a 1.0, any formant parameter errors are gated back through the vowel network only. Because the V/C network does not use the X or Y coordinates to make its decision, there is no pressure to change the vowel decision to a consonant decision. Instead, at that time step the inverse mapping only changes the X and Y coordinates, thereby making the vowel network approximate the

variation	time steps	epochs	$\sum^2$ error	comments
1	56	1000	0.15	no V/C constraint, fixed scale and random initial hand parameters, temporal constraints
2	56	1000	0.21	V/C constraint, fixed scale, random initial hand parameters , temporal constraints
3	56	1000	0.20	no V/C constraint, no fixed scale, random initial hand parameters, temporal constraints
4	56	911	0.12	V/C constraint, no fixed scale, random initial hand parameters, temporal constraints
5	110	1000	3.42	V/C constraint, fixed scale, user determined initial hand parameters, temporal constraints
6	110	1000	2.29	no V/C constraint, no fixed scale, user determined initial hand parameters, temporal constraints
7	56	36	102.25	V/C constraint, fixed scale, random initial hand parameters, no temporal constraints
8	56	345	91.36	no V/C constraint, fixed scale, random initial hand parameters, no temporal constraints

Table 7.1: Results from iterative inverse of “hello” (duration of 0.56 seconds). The comments section describes which constraints were imposed on each of the iterative inverses.

variation	time steps	epochs	$\sum^2$ error	comments
1	18	489	0.08	no V/C constraint, fixed scale and random initial hand parameters, temporal constraints
2	18	526	1.62	V/C constraint, fixed scale, random initial hand parameters , temporal constraints
3	18	923	0.01	no V/C constraint, no fixed scale, random initial hand parameters, temporal constraints
4	18	712	0.07	V/C constraint, no fixed scale, random initial hand parameters, temporal constraints

Table 7.2: Results from iterative inverse of “of the” (duration of 0.18 seconds). The comments section describes which constraints were imposed on each of the iterative inverses.

variation	time steps	epochs	$\sum^2$ error	comments
1	55	1000	0.27	no V/C constraint, fixed scale and random initial hand parameters, temporal constraints
2	55	1000	1.40	V/C constraint, fixed scale, random initial hand parameters , temporal constraints
3	55	1000	0.23	no V/C constraint, no fixed scale, random initial hand parameters, temporal constraints
4	55	1000	0.24	V/C constraint, no fixed scale, random initial hand parameters, temporal constraints

Table 7.3: Results from iterative inverse of “the city” (duration of 0.55 seconds). The comments section describes which constraints were imposed on each of the iterative inverses.

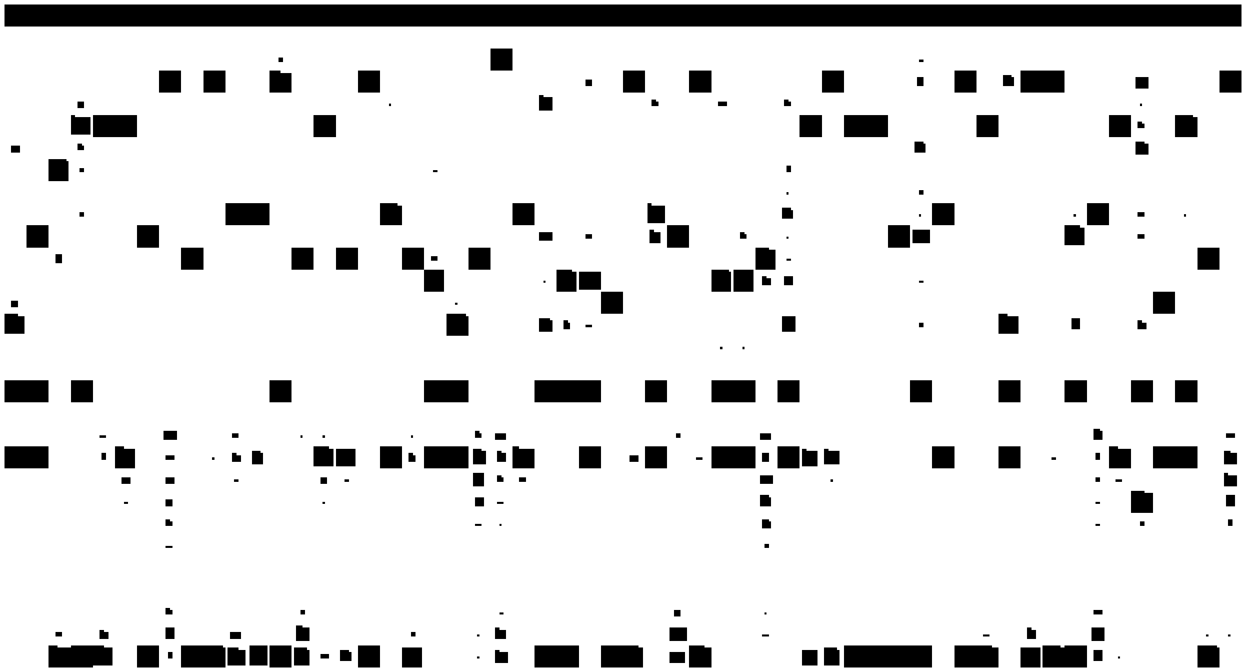


Figure 7.2: Inverse Activations for "hello": Variation 7 – V/C constraint, fixed scale, random initial hand parameters, no temporal constraints

consonant sound to the best of its ability, while completely ignoring the consonant network. This effect can be seen clearly in figure 7.2 which shows the inverse obtained for the utterance "hello" using no temporal constraints. In this figure, each successive time step bears little relation to each other. Further, hand configurations vary widely between each time step, so even if the squared error (or the speech) produced with such a mapping sounds good the hand trajectories are impossible to perform. Temporal constraints were used in all other attempts since the iterative inverse results are so poor without them. Using the V/C constraint cannot overcome the inherent difficulty of omitting temporal constraints, as made clear in table 7.1 and figure 7.2. One possible way to avoid temporal constraints would be to use judiciously chosen initial values for the input to the V/C network (which is also the input to the consonant network). The values should be chosen so as not to saturate the V/C network at the beginning of the optimization. Unfortunately, getting these initial values is not simple since the V/C network is very sensitive to small changes in the input which exasperates the problem.

The sum-squared error suggests that the imposition of temporal constraints remains crucial to finding good inverses. The sum-squared error does not make clear which of the remaining constraints is most useful. In most cases, when the formant parameters obtained from the inverse mappings are played back through the synthesizer, they are generally indistinguishable from the original target utterance. This is true even when the inverse mapping was initialized with data from the user for the utterance. In these cases though, the length of the inverse mapping was twice as long as the target utterance. The resulting speech sounded like the target utterance slowed down by half. The two inverses (not including inverses without temporal constraints) which do not sound exactly like their targets are variations 2 and 4 on the utterance "of the". In variation 2 the speech at the end is noticeably different from the target utterance. The probable cause can be seen in figure 7.3. The final 2 segments of the speech indicate that the V/C network shifted towards a consonant sound but was still quite active. At that point, the speech produced sounds like a mixture of sounds from the vowels and the consonants. In all likelihood, such a mixture will be perceptually non-speech like. Other unsatisfactory speech occurs from the inverse produced using variation 4 from the utterance "of the". The speech is not as poor as with variation 2, however, the beginning portion is noticeably different from the target utterance. Figure 7.4 does not reveal any obvious reason for the problem. Consider just the V/C network output: it starts with a vowel sound for about 5 time steps (according to the vowel outputs, some neutral sound is produced), then produces a consonant sound for 8 time steps (initially a mix of L, V and DH sounds and turning into a solid DH sound), and ends with a vowel sound tapering toward a consonant sound. Oddly, the only perceptual anomaly occurs at the beginning of the utterance. By viewing the speech produced using a pseudo-spectrogram, one can observe that F2 starts slightly higher than in the target utterance and gradually falls to the same value (i.e. in about 3 or 4 time steps). It is unclear why these perceptual anomalies occur; however, they do serve as warnings that the iterative inverse mapping must be used carefully. The final inverse obtained for an utterance must be played through the forward mapping and the synthesizer to make sure that it is perceptually valid.

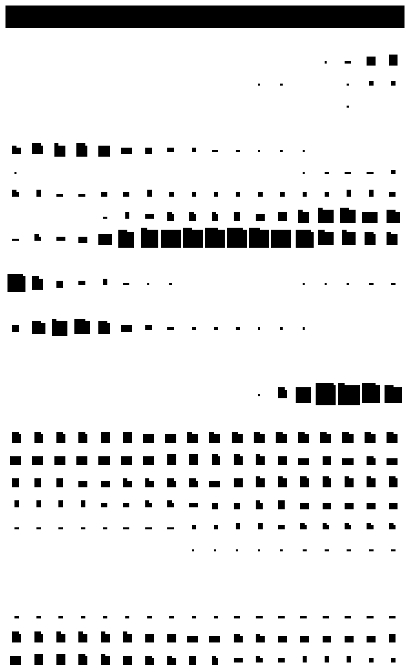


Figure 7.3: Inverse Activations for “of the”: Variation 2 – V/C constraint, fixed scale, random initial hand parameters , temporal constraints

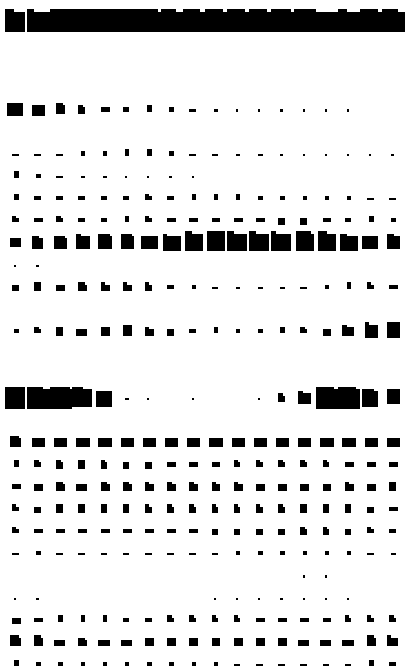


Figure 7.4: Inverse Activations for “of the”: Variation 4 – V/C constraint, no fixed scale, random initial hand parameters, temporal constraints

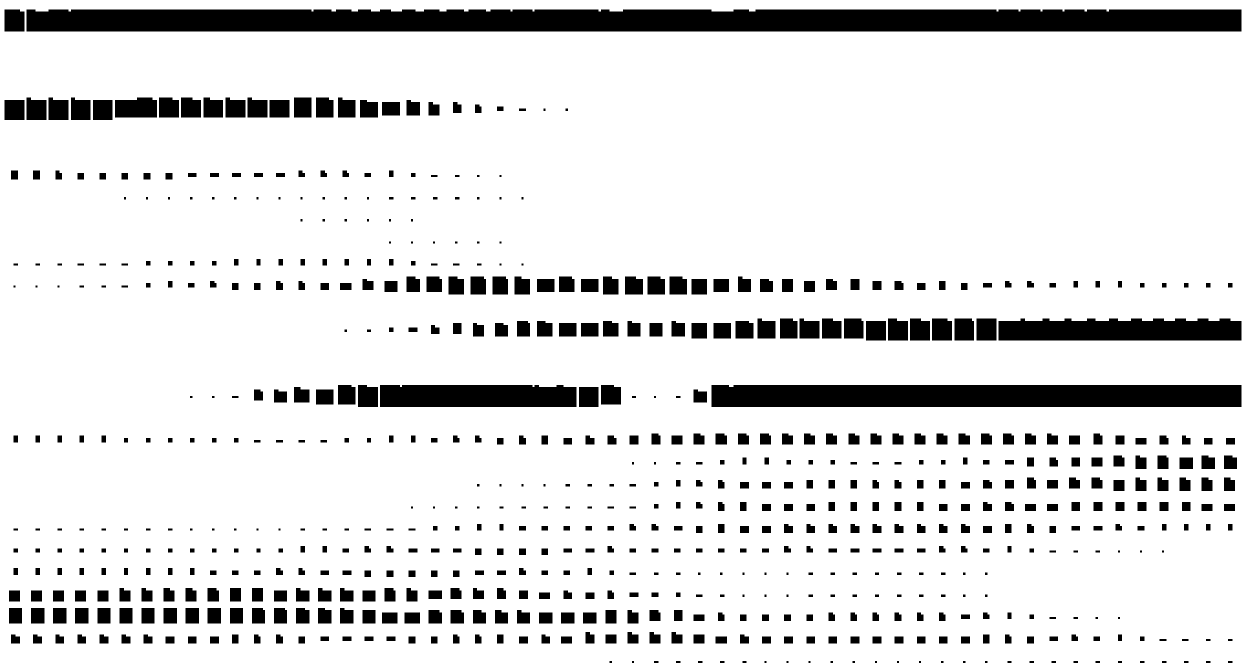


Figure 7.5: Inverse Activations for "hello": Variation 4 – V/C constraint, no fixed scale, random initial hand parameters, temporal constraints

Interestingly, speech obtained from the inverse mapping sounds very good when it is initialized with data from the user. In these cases though, the inverse mapping is twice as long as the target utterance. The resulting speech sounds similar to the target utterance slowed down by half. An interesting effect occurs from the time warping used to align the initial guess with the target utterance. The time warping stretches sounds according to how the user gestures. When the user performs the initial-guess gesture for the target utterance, he specifies the relative timing for each of the sounds. Consider the utterance "hello". In this case the L sound stretches over a period of 20 time steps while the E sound only stretches over 10 time steps. In the target utterance, the L sound accounts for only 4 time steps (out of 55 time steps) while the E sound exists for 10 time steps. Thus, the relative timing of the inverse mapping is specified by the initial mapping provided by the user. This is somewhat problematic, since for some utterances the relative timing of the phonemes may affect the intelligibility of the speech. The inverse mapping must cause the required timing to dictate how the user performs gestures, not the other way around. Using the user's initial guess is less than ideal.

The activity-over-time displays are extremely useful for comparing the various inverses. These displays show which phoneme sound (or combination of sounds) should be produced at each time step. While the user does not expect to maintain exact absolute timing, the relative timing information should be preserved during the utterance. See, for instance, the L sound in "hello" is considerably shorter than the rest of the sounds in the utterance in figure 7.5. Seeing the relative timing information figures prominently in getting correct consonant cluster timings.

Ideally, the inverse should approximate the user's intuitive notion of the correct hand gesture, based on the initial mapping. Then the user would only need to focus on the few differences encountered from the inverse. If the inverse hand gesture differs too much, subtleties will escape the user since she will spend much effort discovering the best general motions. For example, consider the inverse for the utterance "of the", obtained with variation 1 (see figure 7.6). When played through Glove-TalkII, this inverse is almost indistinguishable from the target utterance. Notice however, that it uses only consonant sounds. Thus, the user must figure out which consonant sounds to produce. The middle of the utterance is simple since only one RBF unit is active (the 9^{th} unit corresponding to the DH sound). However, a mixture of consonant sounds exists at the beginning and end to approximate the occurring A. To say the utterance with this inverse, the user must remember this new method of saying A. In comparison, consider the inverse determined using variation 3 in figure 7.7 (which also reproduces the target utterance well). In this case, the user should begin with a vowel sound near the A sound (as expected). She then makes a mixture of the V sound and the DH sound starting at the 7^{th} time step. In two to three more time steps her fingers should move to produce a strong DH sound as indicated by the very active 9^{th} unit. Starting at the 13^{th} time step, she again opens her hand to produce the A vowel sound. Another important observation is that the user should begin by gesturing a V and DH mixture while the first vowel sound is produced. Next, the whole hand should come together at the appropriate time (about the 7^{th} time step), so the user only need to lift her middle finger to get the pure DH sound. The index finger stays in the DH position while opening the hand to produce the ending vowel. The inverse

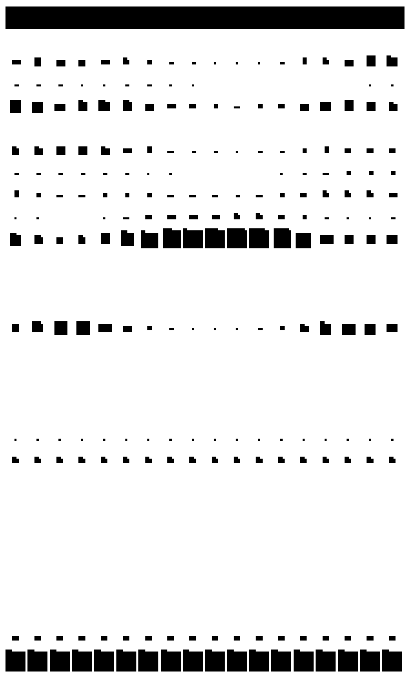


Figure 7.6: Inverse Activations for "of the": Variation 1 – no V/C decision constraint, fixed scale and random initial hand parameters, temporal constraints

mapping provides the essential details of how to position the user's hand during the entire utterance efficiently.

Once an inverse has been decided upon using the sum-squared criteria, the synthesis output, and the activation-over-time display, a graphical hand-display shows the user the necessary hand motions. The graphical hand display is particularly suited for determining how combinations of consonant sounds should be created. These types of sounds are usually unintuitive since the initial mapping does not define them and they are products of what the networks learn. Figure 7.7 gives an example of combinations of consonant sounds. Here, the inverse suggests that a combination of the V and DH sounds can produce the "f" sound in "of the". The activation-over-time display does not indicate what the corresponding hand gesture would look like though. The graphical display shows exactly which hand configuration should be made at each time step.

The graphical hand and activation-over-time display are also useful for determining the best anticipatory behaviours. Since vowels and consonants are independent, one can be prepared while the other is being produced. Temporal constraints use this independence by causing fingers movement for the upcoming consonant sounds during vowels sounds, and vice versa. Anticipatory behaviour is evident in many of the inverse mappings. Consider, for example, the inverse for "hello" in figure 7.5. The hand configuration starts out producing the H sound. In addition, the hand is positioned in X and Y space to produce the appropriate vowel sound (between the E and AA sounds), as soon as the hand opens and causes the V/C network to indicate a vowel. Another example can be found in figure 7.7, which shows the activations from the inverse of "of the". At the beginning of the utterance, the V/C net indicates vowel production. The consonant network indicates that the hand should be producing a mixture of the V and DH sounds, anticipating closure of the hand. Anticipatory behaviour was also noticed during stop production. In this case, hand configurations on either side of the stop consonant dictated the movement of the hand during the stop consonant; anticipating the upcoming sound after the stop finishes. This type of co-articulation obtained from the inverse helps the user speak faster with better quality.

7.2 Inverse Mappings of Alternate Implementations

So far, only the inverse of the final Glove-TalkII system has been described. Chapter 6 describes alternate methods to implement Glove-TalkII (fixed, feed forward, principal component and auto-encoder). Each of these methods has a different technique for inverting the forward mapping.

The inverse of fixed mappings is easy to obtain if the rule is invertible. If it is, just invert the equation specifying the forward mapping. For example, F0 and volume are both mapped with a linear relation specified in equation 4.1

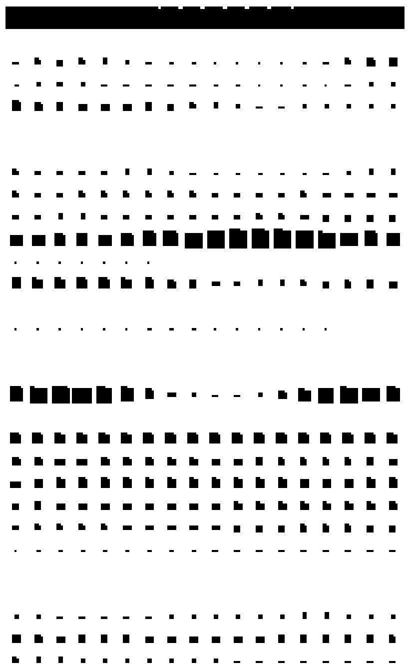


Figure 7.7: Inverse Activations for "of the": Variation 3 – no V/C constraint, no fixed scale, random initial hand parameters, temporal constraints

and equation 4.2 respectively. To invert a target utterance the following relations are used:

$$z = -\left(\frac{(F0-18)}{30}+1\right) \tag{7.2}$$

$$footpedal = \frac{volume}{2.5}+0.5 \tag{7.3}$$

It is possible to create fixed mappings which are not invertible, which necessitates some approximation of the inverse. Fixed mappings are usually indicative that the designer has some cognitive model in mind which specifies the relationship. This cognitive model can be used by the speaker to help determine the inverse.

There are two basic mechanisms to get the inverse mapping from a feed forward network; 1) use an *iterative* inverse or 2) *learn* the inverse. Section 7 describes the iterative inverse. A neural network can be used to learn the inverse relationship of the forward model. Just as with the iterative inverse though, if the forward mapping is many-to-one it becomes necessary to constrain the learning task to pick only meaningful inverses. This technique is used in [19] and relates to the constrained auto-encoder method described in section 6.5.

The other two techniques suggested as alternate implementations of Glove-TalkII provide an inverse *while* they create the forward model. The principal component method's inverse is defined by equation 6.4. The constrained auto-encoder learns both a forward and inverse model of the task simultaneously.

7.3 Summary of Inverse Mapping

There are four important lessons to be learned from the experiments using the iterative inverse:

1. temporal constraints are vital for producing good inverses,
2. the remaining constraints sometimes are useful; however, careful *posthoc* analysis is necessary to determine the quality of the inverse produced,
3. the activation-over-time display is vital for interpreting the inverse,
4. the inverse mapping is particularly suited for determining anticipatory behaviour and mixtures of consonant sounds.

The temporal constraints produce smoothly varying hand gestures with anticipatory behaviour. The other constraints sometimes help produce useful inverses. Unfortunately, it remains unclear when they help and when they do not. A useful procedure for choosing constraints might start with no constraints and then add the others in turn[3]. Combinations of constraints should also be tried.

Starting with an initial guess from the user is fairly time consuming and does not seem to help much. These are the main problems with using the user's initial guess:

1. relative timing is specified by the initial guess,
2. audio feedback is currently unavailable during recording,
3. time warping sometimes may misalign the initial guess,
4. and the V/C decision network is very sensitive to small variations in the input, causing it to saturate easily.

For most speaking requirements, the user's intuition suffices to produce intelligible speech. This is because the final Glove-TalkII system learns the user's interpretation of the initial mapping. The initial mapping provides a good beginning guess for the user to speak any utterance. From his initial guess, a trial and error system, coupled with practice, usually produces intelligible speech. Unfortunately, for particular utterances the user may be unable to find a suitable hand gesture. In these cases the inverse mapping is helpful.

Another subtle point is that the target utterances are currently obtained from the text-to-speech synthesizer. Text-to-speech synthesis is notorious for sounding robotic, hence, the user learns inverses for poor quality speech. Another possibility would use target utterances from real speech. Presumably, only a small set of utterances would need inverses, so efforts to extract formant parameters for the small set of real target utterances for which natural-sounding speech is desired would be worthwhile.

Finally, the level of difficulty in interpreting the inverse mapping suggests that systems which *automatically* generate forward mappings (see sections 6.4 and 6.5) will be difficult to learn to speak with. The obvious way to learn to use these systems is through the inverse mapping, and by watching the graphical hand perform target utterances. Without some model of speech production one observes the user's great difficulty mimicing the graphical hand, since there is too much information displayed. Some way of knowing which details of the inverse are important and which are not must be found. One solution to aid the speaker in this would be to generate a logical language model with which the user can make initial guesses for utterances; this would make it much easier to interpret the finer details of the inverse mapping.

[3] I suggest starting with the constraints in the following order: temporal constraint, V/C constraint, scaling constraint and user supplied initial guess.

Chapter 8

Discussion and Conclusions

Currently, speech produced with the final version of Glove-TalkII is intelligible but not particularly natural-sounding. There are five areas where efforts to improve the quality of speech may be fruitful:

1. making the speech less robotic,
2. improving stop consonants,
3. improving phoneme transitions,
4. better pitch control,
5. better volume control.

Each of these areas is discussed below along with suggestions for improvements.

8.1 Robotic Speech

The first aspect of Glove-TalkII speech perceived by listeners is its robotic quality. The probable cause for this phenomenon is that all the networks are trained with data from a text-to-speech synthesizer. The text-to-speech synthesizer produces robotic sounding speech, so it does not come as a great surprise that Glove-TalkII does also. Originally, the researcher's hope was that the user would be able to modify her hand movements to make the speech sound more natural than that of the basic mapping learned by the networks. This hope was fulfilled to some extent in that some Glove-TalkII speech sounds more "natural" than that of the text-to-speech synthesizer. In particular, user-controlled pitch allows for some very human sounding utterances.

One possible solution would be to train the networks using formant data from real speech. The type of data required are the formant tracks from a large corpus of real speech, so that variations[1] of each phoneme become available. The variations would then be mapped to variations around the predefined static configurations of the user's hand, specified by the initial mapping. To get meaningful variations of the user's hand during the training session, the user should move his hand to say an utterance that contains the allophone. This will capture contextual constraints of the user's hand to determine the appropriate mapping. The main problem with this training regime would be the required large training corpus of labeled formant data from *real* speech. Currently, no automatic means are available to extract formant tracks reliably from real speech. It would be best to create the formants data from the user's own speech if available so that they do not have to mimic a different person's accent.

The next subsection describes a possible scheme to extract formant trajectories from real speech automatically.

8.1.1 Extracting Formant Parameters from Real Speech

A target corpus of formant parameters from real speech is required to improve the quality of Glove-TalkII speech. Unfortunately, an automatic formant extractor is unavailable at the current time. This section proposes a scheme to create one.

The proposed formant extractor creates a network model of the formant-to-speech process (see [31]). The network model allows computation of the change in the synthetic speech signal with respect to the formant parameters. The text-to-formant generator provides an initial guess of the formant parameters for a segment of real speech. Using

[1] These variations are called allophones; for example there are approximately 4 different allophonic variations of the phoneme /l/.

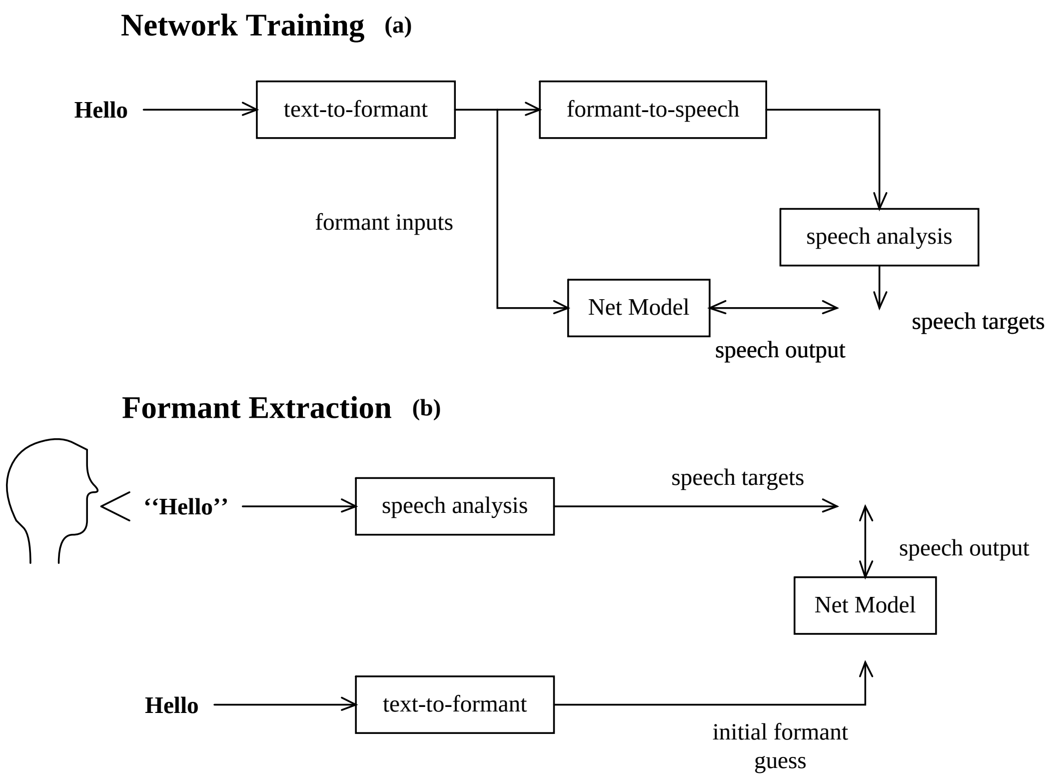


Figure 8.1: A scheme for automatic extraction of formant parameters from real-speech. a) The initial training of the network - the network is trained to map formants to processed speech (basically the same task as the formant synthesizer except the speech wave may be processed). The idea is to have a model of the formant synthesis task so that speech errors can be backpropagated to modify the formant input. b) The formant extraction stage - once the network has been trained the text-to-formant generator is used to make initial guesses for the formants of the real-speech signal. The output errors are backpropagated to adjust the inputs.

the network model and the backpropagated errors, the initial guess from the text-to-formant generator changes until it matches the real speech. This is very similar to the iterative inverse method of obtaining hand parameters (see section 7), except in this case, formant parameters are extracted.

To understand how the automatic formant extractor works, consider the following paradigm (illustrated in figure 8.1). First, the forward model of formants to speech is learned from examples provided by the parallel formant speech synthesizer and the text-to-speech synthesizer. Second, a speaker makes an utterance and the text-to-speech system generates its initial formants. The text can be adjusted to match the real speech as closely as possible. Also, the speaker can adapt her speech to match the text-to-speech output as closely as possible with respect to temporal alignment. Hopefully, a close match can be obtained, so the time alignment will not become problematic. The trained formant-to-speech network adjusts the formant inputs to match closely the real speech. Notice that the representation of the speech output does not necessarily have to be the time domain amplitude waveform. In fact, the time domain speech wave probably is not the best representation to use for the output of the network. For this reason, some speech analysis should be performed. The speech analysis used must be invertible and able to produce high quality speech from the inverse; otherwise, some of the natural-sounding aspects of the real speech signal will be lost, defeating the whole purpose of this digression. Some possible choices for the speech analysis section include fast Fourier transform or Cepstral analysis. Remember, the point of the formant extraction is to generate training utterances for the gesture-to-speech task. For this reason, formant accuracy is not critical for *every* real-speech utterance; rather, only enough utterances are necessary to provide a good ensemble of training cases. Various network architectures need to be considered for the formant extractors. A recurrent network or a time delay neural network seem appropriate since the formant resonators in the parallel formant synthesizer use infinite impulse response (IIR) filters.

8.2 Stop Consonants

In the final version of Glove-TalkII, stop consonants are generated using the keyboard. This is very cumbersome for three reasons: first, the user has a tendency to synchronize left-handed movements with right-handed movements;

second attending to the keyboard is distracting; and third, the user often presses the wrong key, producing the wrong stop consonant. Further, stop consonants have fixed quality and duration (except for the NG sound). Due to the fixed quality of each stop consonant, these sounds are perceptually difficult to distinguish for the listener. Several schemes are available to overcome some of the difficulties.

The first idea involves positioning button devices on the user's left hand[2]. This could be accomplished by mounting nine buttons on a left handed glove. To implement these buttons, two contacts could be placed on the two end segments of each finger (including the thumb), providing approximately 11 usable buttons. The user's left hand is now free to move with the right hand and can still generate stops. Allowing the user's left hand to move with his right hand should help the user synchronize the stop onset and offset times. Further, this scheme should reduce button errors.

Adjusting the quality of the stop consonants poses a difficult problem. In Glove-TalkII, stops are independent of the other vowel and consonant sounds. However, in real speech this is not the case. Providing sensors to allow the user to adjust the quality of the stop consonant probably will not help much since the required sounds are context dependent. Perhaps a rule-based system using the state of the user's right hand during stop production could help improve the quality of the speech. Presumably, the rules necessary would be quite complex given that the rules for stop consonants within the text-to-speech system are fairly complex and still produce poor quality speech.

8.3 Phoneme Transitions

In the final Glove-TalkII system, the user is responsible for managing all temporal aspects of speech. In particular, she must time the phoneme transitions properly. Sometimes, proper timing for specific utterances is very difficult to achieve, due to the constraints on the user's hand. In these cases, a mechanism to assist them will be sought (this idea is similar to the interlocking mechanisms described in [9]). The N to L movement in the word "only" provides an example of a difficult transition. Here the user must sustain the N sound for considerably longer than the L sound. Also, when moving from N to L she must move swiftly and avoid making any superfluous phonemes. There are various possibilities available to assist the user with these transitions.

The most obvious approach begins by identifying all difficult transitions and then creating rules to perform the proper transitions in phoneme space. This involves recognizing each phoneme as the user produces it and predicting which phoneme he intends to make next. For example, in the case of the N to L transition, the N is recognized first; then as the user moves towards the L sound the N to L transition becomes noticeable and a rule can then be invoked to sustain the N sound for a few extra time steps. This will allow the user to reach the L sound in a "natural" length of time. This scheme does not adjust the duration of the complete utterance but only the relative durations of the various phonemes during the utterance. A neural network would probably be useful for this recognition task. In fact, the networks that use an extra layer of softmax units could use them to perform the recognition.

Another approach uses networks to learn the proper transition rules. Instead of training the networks in Glove-TalkII on static hand configurations matched with static phonemes, hand trajectories could be used with phoneme transitions as targets. The easiest method would be to create phoneme pairings occurring in English along with the static phonemes as training examples. The user would then make appropriate hand gestures for the phoneme pairs (and static phoneme). The networks should learn the co-articulation of the user's hand for the given phonemes and adjust the speech accordingly. Of course, the hand gestures would have to be temporally aligned with the dynamic formant parameters. Further, the networks would have to have some mechanism to capture the temporal context of the hand gesture data. A recurrent neural network [36] [30] or a time delay neural network [35] are reasonable first choices to use.

The first method of using fixed rules is the easier of the two schemes. Its main problem regards the poor quality speech produced by phoneme rules, clearly illustrated by any observation of text-to-speech synthesis. The second method has the potential to provide better phoneme transitions, hence, better quality speech. It is, however considerably more difficult to implement. In addition, the scheme depends on the time alignment of the user's hand gestures with the phoneme transitions. In preliminary experiments time warping did not always suffice to provide a good alignment; the data then had to be aligned by hand, which is very time consuming. An advantage of the second scheme is that it can be used with formant data from real speech. As an interesting aside, if a good scheme for mapping hand gestures to phoneme transitions can be found, it will have direct benefits for improving text-to-speech synthesis. Clearly, more research is necessary to discover a good method.

[2] Of course, if the system is configured for a left handed user, the buttons would be on the right hand.

8.4 Pitch Control

Currently the pitch is controlled by a fixed mapping from the height of the hand (along the z axis - see equation 4.1 in section 4.2.2). Pitch control seems to add a significant amount of intelligibility and naturalness to the user's speech when compared with text-to-speech synthesis. In fact, Glove-TalkII speakers produce better quality speech than the text-to-speech synthesizer does mainly by providing a more natural pitch contour to an utterance. Further, the pitch aids the listener by providing semantic context for the utterance.

A trial and error system determines the mapping between the height and pitch. Alternatively, the networks could learn the user's intuitive mapping between hand height and speech pitch. This could be accomplished through psychoacoustic experiments or by learning the mapping with a neural network from user supplied heights mapped to target pitches.

Users seem to adapt quickly to the fixed mapping determined by trial and error. It appears that as long as the mapping contains the complete range of pitch available in a linear relation[3] with the heights of their hands, users can utilize pitch appropriately.

One should consider, however, that the relationship that pitch increases with hand height is important. This seems to coincide with the user's cognitive model of pitch. Further experimentation may yield a matched pitch mapping; however, it may not be worth the additional effort required.

8.5 Volume Control

Speech volume is determined by a fixed mapping between foot pedal depression and formant amplitude (see equation 4.2 in section 4.2.2). Additionally, the foot pedal maps the voicing control, so that at low volumes speech becomes completely unvoiced (this rule is used to obtain a similar effect as with the text-to-speech synthesizer). User control of volume does not appear to have as significant an effect on intelligibility and naturalness as the pitch control does. This observation is confounded by the poor control the user has over volume. Even though the foot pedal allows the user to vary the volume continuously from off to full volume, the user typically only uses the two extremes: i.e. the speech is either on at full volume or off. The main problem appears to be lack of feedback from the foot pedal which would allow the user to exert fine control. One way to improve matters would be to attach a *gas spring* with adjustable flow control to the foot pedal. This device would act like a bellows, simulating breathing. The gas flow rate or angular speed of the foot pedal would be mapped to volume. To use the device, the user would start speaking by pushing hard on the foot pedal until he reaches a maximum depression rate corresponding to a fast onset of speech. As he continues to depress the foot pedal, the user can then slow its depression rate to get a small fluctuation in volume. These small fluctuations are typical of real speech. When the user finishes speaking, the foot pedal can be quickly reset to its upright position ready for the next sentence. This type of foot pedal would allow finer volume control than currently available. Of course, the user can only speak for some maximum length of time before he must take a "breath".

Another possible improvement to volume control would be to learn the user's cognitive model of the mapping between foot depression and volume. Techniques to learn this mapping are the same as described for learning the pitch mapping (section 8.4): use psychoacoustic experiments or train a network using foot pedal/volume training pairs from the user. Whether better volume control enhances the quality of the speech is a question still to be answered.

8.6 Conclusion

Development of Glove-TalkII provides three main contributions in the fields of neural networks, adaptive interfaces and speech prosthesis:

1. provides an AVT gesture-to-speech system with an adaptive interface
2. demonstrates that neural networks can be used to learn complex mappings necessary in adaptive interfaces
3. provides a framework for adaptive interfaces and illustrates its usefulness by placing Glove-TalkII within the framework

The utility of Glove-TalkII as a gesture-to-speech device has been demonstrated by one subject. After 100 hours of training he can carry on an intelligible conversation as well as read out loud. His speech is slower than normal

[3] Linear with respect to the 0 to 63 range of F0 used by the parallel formant synthesizer. This range corresponds to a logarithmic change in frequency.

speech and sounds fairly robotic. Comparable speech is produced by a text-to-speech synthesizer, except Glove-TalkII speech has more natural sounding intonation. The subject progressed through eight stages of development while learning to speak with Glove-TalkII. The 100 hours of training time to speak intelligibly compares favourably with learning times to speak with the Voder [9] which required operators to train for 1 year. Three aspects of speaking are particularly difficult for the subject: fast speaking rate, multisyllabic words and reliable vowel production. In some cases, the subject must say an utterance several times before he is understood.

Glove-TalkII uses three neural networks to implement the mapping between hand gestures and speech. The V/C network gates the output of the vowel network and the consonant network according to whether the user says a vowel or a non-stop consonant. The V/C network and the consonant network are trained from examples obtained from the user during easy-to-use training sessions. The simplicity of the collection procedure is important for a useful adaptive interface. The networks then learn the user's interpretation of a hand-designed gesture vocabulary. The initial gesture vocabulary is loosely based on an articulatory model of the user's hand. The vowel network trains on artificially generated examples determined from the initial mapping. The height of the user's hand and foot pedal depression are mapped to F0 and volume by fixed rules, respectively. Stop consonants are produced on the keyboard.

Other techniques are available to implement Glove-TalkII. Four different techniques have been described in this dissertation. The gated neural network/fixed mapping hybrid used to implement the final Glove-TalkII system evolved from many different experiments with different architectures. The current implementation demonstrates one effective way neural networks can be used to implement an adaptive gesture-to-speech interface. One of the key developments in making Glove-TalkII work occurred when the gesture-to-speech task was sub-divided into several independent tasks. Once identified, the sub-tasks were much easier to implement than the whole task at once. Interesting research remains to be done to compare other neural and non-neural techniques for implementing Glove-TalkII.

An adaptive interface framework was introduced in section 2. Glove-TalkII can be categorized according to the five adaptive interface considerations:

1. characterize user: consonant examples and V/C examples
2. measurements: 10 cyberglove angles, 3 polhemus values, foot pedal and keyboard
3. model altered: domain
4. strategies:
   - discrimination learning for non-consonant phonemes and V/C decision
   - preprogramming for stop consonants, fundamental frequency and volume
   - reprogramming for vowels
5. algorithm: backpropagation through neural networks used to implement adaptation strategy

The final Glove-TalkII system can be placed on the left of centre on the computer adaptation system continuum outlined in figure 2.2 since it uses both supervised and non-adaptive methods to function. Some of the alternate ways considered in section 6 lie at other locations on the continuum.

Several important applications and future research directions present themselves from the Glove-TalkII research. These fall into two distinct categories:

1. using and improving Glove-TalkII
2. applying neural network techniques to other interface applications

Using Glove-TalkII as a speech prosthesis for people with speech impairments presents an appropriate application for Glove-TalkII. To use the current Glove-TalkII system, users must be dextrous and able to hear. These requirements limit the number of candidates who can immediately benefit from Glove-TalkII. Several refinements are necessary to create a practical gesture-to-speech device for these candidates.

To refine Glove-TalkII, several improvements should be considered. First, remember that Glove-TalkII speech is similar in quality to text-to-speech synthesizer output, probably because target formants are derived from there. In section 8.1.1 a scheme to extract formant parameters from real speech automatically was presented. These parameters could be used as formant targets instead of deriving them from the text-to-speech synthesizer. Glove-TalkII could then train on the natural-sounding targets which should improve the quality of Glove-TalkII speech. Automatic formant extraction remains an open research issue.

Second, in some contexts, stop consonants are difficult to distinguish. Future research is necessary to improve these sounds by adjusting the qualities of the stop corresponding to the context of the speech. A major step toward improving stop consonants would use a left handed glove with contact switches instead of the keyboard for stop production.

Third, the hardware currently used to implement Glove-TalkII is both expensive and bulky. Glove-TalkII uses a combined total of about 1000 weights (including the novice V/C network). Output to the parallel formant synthesizer is required every 10 msec, which necessitates an inexpensive machine capable of performing about 200,000 floating point operations per second. The system should operate in real-time. Additionally, the system is not very portable. For practical implementations, a portable, low-cost device should be developed.

An interesting application for Glove-TalkII would involve studying speech production mechanisms. Glove-TalkII provides an artificial vocal tract that can be used to speak different languages with an unlimited vocabulary. Typically, people only have one mechanism which produces speech. This fact makes it difficult to disambiguate speech effects due to perceptual constraints and those due to motor constraints. With Glove-TalkII a user now has two mechanisms by which to speak, enabling psychological studies which try to separate the two effects. For instance, an interesting psychological study would be to determine whether a subject has the same accent with Glove-TalkII as they do with their own vocal-tract produced speech.

The neural network techniques described in this dissertation can be applied to other interface designs. In particular, devices which require "high bandwidth" control could benefit from an adaptive interface implemented with neural networks. Specific devices include: tele-operated robot arms, musical instruments and system control panels. Some of the alternative techniques described in this dissertation create the mapping between the user and the device automatically; see for example, the constrained auto-encoder method. Initial results with the inverse mapping in Glove-TalkII suggest interfaces using these schemes are difficult to use unless the user forms a cognitive model of the task. Further, as expected, once a user becomes familiar with a particular mapping, it is difficult to alter the mapping significantly without a severe performance penalty. Much research remains to establish and quantify the interaction between adaptation by the user and adaptation by the device.

Interest in creating an artificial vocal tract, and specifically a gesture-to-speech device has persisted since the late 1700's. The motivation is two-fold. First, people with speech impairments could learn to speak with such a device. Second, a gesture-to-speech device would advance the understanding of speech production. Such understanding is of vital importance in the fields of speech therapy and speech synthesis. Glove-TalkII provides a successful contribution to this long-standing goal.

Bibliography

[1] J. Anderson and M. C. Mozer. Categorization and selective neurons. In G. E. Hinton and J. A. Anderson, editors, *Parallel Models of Associative Memory.* Erlbaum, Hillsdale, NJ, 1981.

[2] J. S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. In D. S. Touretzky, editor, *Neural Information Processing Systems, Vol. 2*, pages 111–217, San Mateo, CA, 1990. Morgan Kaufmann.

[3] J.S. Bridle. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fougelman-Soulie and J. Herault, editors, *NATO ASI series on systems and computer science.* Springer-Verlag, 1990.

[4] G. Bristow. *Electronic Speech Synthesis.* McGraw-Hill Book Company, London : Granada, 1984.

[5] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[6] D. Browne, P. Totterdell, and M. Norman. *Adaptive User Interfaces.* Academic Press, London, UK, 1990.

[7] D. N. Chin. User modelling in UC, the UNIX consultant. In *CHI'86 Proceedings – Human Factors in Computing Systems*, pages 24–28, 1986.

[8] G. Dewy. *Relativ Frequency of English Speech Sounds.* Harvard University Press, Cambridge, Mass, 1950.

[9] Homer Dudley, R. R. Riesz, and S. S. A. Watkins. A synthetic speaker. *Journal of the Franklin Institute*, 227(6):739–764, June 1939.

[10] R. Erenshteyn, L. Messing, R. Foulds, G. Stern, and S. Galuska. Back propagation neural network for american sign language recognition, March 1994. Unpublished manuscript.

[11] S. Fels and G. Hinton. Glove-Talk: A neural network interface between a data-glove and a speech synthesizer. *IEEE Transaction on Neural Networks*, 4:2–8, 1993.

[12] S. S. Fels. Building adaptive interfaces using neural networks: The Glove-Talk pilot study. Technical Report CRG–TR–90–1, University of Toronto, 1990.

[13] S. S. Fels. Speech synthesis techniques, April 1992. Unpublished manuscript.

[14] P. M. Fitts. Perceptual–motor skill learning. In A. W. Melton, editor, *Categories of human learning.* Academic Press, New York, NY, 1964.

[15] J. L. Flanagan. Voices of men and machines. *Journal of the Acoustical Society of America*, 51:1375–1387, 1972.

[16] A. D. Girson and R. D. Williams. Articulatory-based synthesis for conversational speech. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 873–876, 1990.

[17] Connectionist Research Group. *Xerion Neural Network Simulator Libraries and Man Pages; version 3.183.* University of Toronto, Toronto, ON, CANADA, 1990.

[18] R. D. Jones, Y. C. Lee, S. Qian, C. W. Barnes, K. R. Bisset, G. M. Bruce, G. W. Flake, K. Lee, L. A. Lee, W. C. Mead, M. K. O'Rourke, I. J. Poli, and L. E. Thodes. Nonlinear adaptive networks: A little theory, a few applications. Technical Report LA–UR–91–273, Los Alamos National Laboratory, 1990.

[19] M. I. Jordan. Motor learning and the degrees of freedom problem. In *Attention and Performance*, pages 796–836. Erlbaum, Hillsdale, NJ, 1990.

[20] D. H. Klatt. Software for a cascade/parallel formant synthesizer. *Journal of the Acoustic Society of America*, 67:971–995, 1980.

[21] J. Kramer and L. Leifer. The 'Talking Glove': A speaking aid for nonvocal deaf and deaf-blind individuals. In *Proceedings of RESNA 12th Annual Conference*, pages 471–472, 1989.

[22] P. Ladefoged. *A course in Phonetics (2 ed.)*. Harcourt Brace Javanovich, New York, 1982.

[23] E. Lewis. A 'C' implementation of the JSRU text-to-speech system. Technical report, Computer Science Dept., University of Bristol, 1989.

[24] D. J. MacKay. Inferring an input-dependent noise level. technical note, University of Toronto, 1991.

[25] D. J. MacKay. personal communication, 1991.

[26] D. J. MacKay. Bayesian methods for adaptive models. PhD thesis, California Institute of Technology, 1992.

[27] R. Pausch and R. D. Williams. Tailor: Creating custom user interfaces based on gesture. Technical Report TR–90–06, University of Virginia, 1990.

[28] H. A. Rey. *Curious George*. Houghton Mifflin Company, Boston, 1941.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by back-propagating errors. *Nature*, *323*:533–536, 1986.

[30] D. E. Rumelhart, J. L. McClelland, and the PDP research group. *Parallel distributed processing: Explorations in the microstructure of cognition. Vols. I and II.* MIT Press, Cambridge, MA, 1986.

[31] J. M. Rye and J. N. Holmes. A versatile software parallel-formant speech synthesizer. Technical Report JSRU-RR-1016, Joint Speech Research Unit, Malvern, UK, 1982.

[32] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26:43–49, 1978.

[33] Educational Services. *DECtalk DTC01 Owner's Manual.* Maynard, Mass, 1984.

[34] Dr. Seuss. *Green Eggs and Ham*. Beginner Books, New York, 1960.

[35] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, & Signal Processing*, 37(3):328–339, 1989.

[36] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[37] E. Yair and A. Gersho. The boltzmann perceptron network: A multilayered feed-forward network equivalent to the boltzmann machine. In David Touretzky, editor, *Advances in Neural Information Processing Systems 1 (NIPS*88)*, pages 116–123, San Mateo, 1989. Morgan Kaufman Publishers.

[38] R. S. Zemel. A minimum description length framework for unsupervised learning. PhD Thesis CRG–TR–93–2, University of Toronto, 1993.

Appendix A

Normalized Units and Weights

Abstract

In this appendix, the partial derivatives necessary for backpropagation through networks which use units that are normalized with respect to some group of units are derived. The activity functions of the units and the normalization function are arbitrary. Softmax units at the output and hidden layers are a special case of normalized units and are used to illustrate the derivations. The important point to notice is that the derivative of the error with respect to a normalized unit is a function of the derivative of the error with respect to all of the units in the normalized group. The equations derived can also be applied analogously to normalized weights.

A.1 Normalized Units

Often, when designing neural networks one has occasion to use normalized units (see for example [18]). Typically, the performed normalization divides all the outputs of a group of units by their sum. Softmax units are the most common example of normalized units (see [2] and [37]). Here, the units have an exponential activation function and the outputs are normalized with respect to all the units in the softmax group. Softmax units are typically used as output units to model a probability distribution such that all the activities sum to one. Usually, a cross-entropy error function is used in a 1-of-N classification task, where the correct class' target is set to one and the other classes set to zero. Three different variations of the normalization paradigm are often desired:

- different activation functions for each of the units before normalization is performed; i.e. exponential activation in softmax units or negative exponential activation in the case of radial-basis functions.
- different error functions if the units are output units
- different types of normalization, i.e. dividing the activation of each unit by the sum of the activations of the units in the normalized group, dividing a unit's activation by the L_2 norm of the group of units.

Further, the group may be either at the output layer or a hidden layer. The following discussion describes the necessary equations for implementing normalized units with arbitrary activation functions, arbitrary error functions and arbitrary normalization functions[1]. Further, the well known equations for softmax output units will be derived using this formulation. The equations for softmax units in the hidden layers will also be derived.

Consider the following notation (similar to [29]). First, the activation function for unit j in the normalized group is

$$o_j = f(x_j) \tag{A.1}$$

where o_j is the unnormalized activation of the unit and x_j is the total input of the unit (typically formed from the dot product of weights and outputs of units in the layer below)[2].

For illustration purposes, consider the normalized output of the unit as

$$n_j = \frac{o_j}{\sum\limits_{m=1}^{N} o_m} \tag{A.2}$$

where N is the number of units in the group. Any differentiable function of the group of units' activations suffices.

The implementation for the forward propagation through these normalized units is straight forward. Backpropagation of the error signal through these units is less simple. It is necessary to calculate the term $\frac{\partial E_p}{\partial x_{pj}}$ where E_p is the error on a particular example p, to implement the backpropagation procedure[3]. Complications arise due to error propagated back to one unit in the group will affect the others. Thus, extra care is needed to ensure the correct influence is calculated.

Consider unit j in the group of normalized units on some example p. If we consider the effect of the total input of unit j on the error, the expression is

$$\frac{\partial E}{\partial x_j} = \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E}{\partial n_j} + \sum_{k=1;k\neq j}^{N} \frac{\partial o_j}{\partial x_j}\frac{\partial n_k}{\partial o_j}\frac{\partial E}{\partial n_k} \tag{A.3}$$

[1] "Arbitrary" in the sense that the first derivative is well defined.

[2] In the case of radial basis function the term x_j is calculated using a distance measure like: $x_j = \frac{1}{\sigma_j}\sum(w_{ji} - o_i)^2$. Of course, the outputs from the lower layer could be normalized as well.

[3] From $\frac{\partial E_p}{\partial x_j}$ we can easily get $\frac{\partial E_p}{\partial w_{ji}}$ in the manner described in [29].

This expression is much easier to understand if we only consider the error that is backpropagated to unit j (which is referred to here as E^j) first and see how that error is propagated to all the other units in the group. In this case, we exclude the effects errors distributed from the other units in the group have on unit j. For unit j the equation is

$$\frac{\partial E^j}{\partial x_j} = \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E^j}{\partial n_j} \tag{A.4}$$

The first term from equation A.1 is simply:

$$\frac{\partial o_j}{\partial x_j} = f'(x_j) \tag{A.5}$$

The second term is obtained from equation A.2:

$$\begin{aligned}
\frac{\partial n_j}{\partial o_j} &= \frac{1}{\sum_{m=1}^{N} o_m} + \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} && \text{(A.6)}\\
&= \frac{\sum_{m=1}^{N} o_m - o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} && \text{(A.7)}\\
&= \frac{\sum_{m=1;m\neq j}^{N} o_m}{\left(\sum_{m=1}^{N} o_m\right)^2} && \text{(A.8)}
\end{aligned}$$

The third term has two different forms depending upon whether the unit is an output unit or a hidden unit. The first case shows an error function in terms of n_j and the derivative is calculated. In the latter case, the derivative is calculated by backpropagating the error through the weights from the layer above unit j. Refer to [29] for an example using a Euclidean error function and sigmoid hidden units. Softmax units are used below to illustrate how these equations are calculated in a specific case.

Now consider unit k in the normalized group where k is some other unit besides j. What is the effect on the error by unit k? The equation is derived as follows:

$$\frac{\partial E^j}{\partial x_k} = \frac{\partial o_k}{\partial x_k}\frac{\partial n_j}{\partial o_k}\frac{\partial E^j}{\partial n_j} \tag{A.9}$$

The important point to notice in this equation is that **the change in the error with respect to $\mathbf{x_k}$ is a function of the change in error with respect to $\mathbf{n_j}$**. Remember, we are only considering the error localized at unit j; that is, the error signal propagated back to unit j, which we can see must be distributed as above to all the other units too. Completing equation A.9, the first term is (from equation A.1):

$$\frac{\partial o_k}{\partial x_k} = f'(x_k) \tag{A.10}$$

The second term follows from equation A.2:

$$\frac{\partial n_j}{\partial o_k} = \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} \tag{A.11}$$

The third term is discussed above.

The above equations are general in the sense that they do not differentiate between hidden units and output units. The only change necessary for the different types of units, as described above, is that the partial derivative of the error should be either obtained from the error function or the backpropagated error function. Each unit in the group of normalized units is considered as unit j, one at a time, and its error gets distributed to all the other units. In this way the total expression in equation A.3 is calculated. Using this method, the complexity to calculate the required partial derivates for the group of normalized units is $O(N^2)$, where N is the number of units in the group, since at each unit the error is distributed to the other $N-1$ members in the group. Normally, the number of units in the group is small, therefore, this is not much of a computational penalty. On the positive side, with respect to implementation, the above method requires a single procedure to compute the derivatives for each unit. Further, the form is general enough to work with any type of normalization function.

If we consider the normalization function used in the example above (equation A.2) we can write equation A.3 as

$$\frac{\partial E}{\partial x_j} = f'(x_j)\left[\left[\frac{\sum_{m=1}^{N} o_m - o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\right]\frac{\partial E}{\partial n_j} + \sum_{k=1;k\neq j}\frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k}\right] \tag{A.12}$$

which reduces[4] after simple algebraic manipulation to

$$\frac{\partial E}{\partial x_j} = f'(x_j)\left[\frac{1}{\sum_{m=1}^{N} o_m}\frac{\partial E}{\partial n_j} - o_j\sum_{k=1}^{N}\frac{1}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k}\right] \tag{A.13}$$

In this form the calculation can be done in two stages: first, for the whole group calculate the term

$$\sum_{k=1}^{N}\frac{1}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k} \tag{A.14}$$

which can be done in $O(N)$ calculations. Second, for each unit in the group, calculate the required partial derivative using equation A.13 and the term from equation A.14 calculated for the whole group. This reduces the complexity to $O(N)$ to compute the required partial derivatives (as compared to $O(N)^2$ for the general case). Notice, this reduction depends on the normalization function which means it is not completely general, however, in most cases this type of normalization is used[5].

A.2 Normalized Weights

The above discussion can be directly applied to normalized weights[6]. In particular, consider the normalizing weight equation[7]:

$$\eta_{ij} = \frac{w_{ij}}{\sum_{k=1}^{M} w_{ik}} \tag{A.15}$$

where η_{ij} is the normalized weight connecting unit i to unit j, w_{ij} is the unnormalized weight connecting unit i to unit j, and M is the number of weights being normalized. A typical combining function for unit j is

$$x_j = \sum_{i=1}^{P} \eta_{ij} o_i \tag{A.16}$$

where P is the fan in to unit j and o_i is the output of unit i that connects to unit j. The derivative of the error (E) with respect to each of the weights is analogous to equation A.3. The equation is

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial \eta_{ij}}{\partial w_{ij}}\frac{\partial x_j}{\partial \eta_{ij}}\frac{\partial E}{\partial x_j} + \sum_{k=1;k\neq j}^{M}\frac{\partial \eta_{ik}}{\partial w_{ij}}\frac{\partial x_k}{\partial \eta_{ik}}\frac{\partial E}{\partial x_k} \tag{A.17}$$

For the usual case given in equations A.15 and A.16, equation A.17 simplifies to

$$\frac{\partial E}{\partial w_{ij}} = \left[\frac{\sum_{k=1}^{M} w_{ik} - w_{ij}}{\left(\sum_{k=1}^{M} w_{ik}\right)^2}\right] o_i \frac{\partial E}{\partial x_j} - \sum_{k=1;k\neq j}^{M}\frac{w_{ik}}{\left(\sum_{k=1}^{M} w_{ik}\right)^2} o_i \frac{\partial E}{\partial x_k} \tag{A.18}$$

which may be further simplified as suggested for normalized units. The next sections provide examples of normalized units using softmax units at the output layer and the hidden layer.

[4] This reduction was pointed out by S. Becker and T. Plate, personal communication.
[5] L_2 normalization reduces to a similar form.
[6] Thanks to Peter Dayan for pointing this out, personal communication.
[7] Note that other normalizing function may be used.

A.3 Example: Softmax Units

The above formulation can be used to obtain the necessary equations for backpropagating through softmax units. Consider the following notation for softmax units:

The activation function is

$$o_j = f(x_j) = e^{x_j} \tag{A.19}$$

And the normalized output of unit j is (equation A.2)

$$n_j = \frac{o_j}{\sum_{m=1}^{N} o_m} \tag{A.20}$$

The derivative of the activation function is

$$\frac{\partial o_j}{\partial x_j} = f'(x_j) = e^{x_j} = o_j \tag{A.21}$$

The error function is usually chosen to be the cross entropy function:

$$E = -log(n_j) \tag{A.22}$$

where j is the output unit with target = 1.

The partial derivative of the error function is:

$$\frac{\partial E}{\partial n_j} = \frac{-1}{n_j} \tag{A.23}$$

The partial derivative of the error function with respect to other units besides the target unit is 0, thus, in our notation above $E^j = E$. The partial of the error with respect to n_j is the backpropagated error from preceding layers, if the units are not in the output layer[8].

A.3.1 Softmax Output Units

Consider the equations derived for softmax units that are output units. There are two different cases to discuss; target output and non-target output. First consider the target output unit j (i.e. its target output is 1). If we substitute equations A.21, A.8 and A.23 into A.4 we get the following:

$$\begin{aligned}
\frac{\partial E}{\partial x_j} &= \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E}{\partial n_j} && \text{(A.24)}\\
&= o_j \left[\sum_{m=1;m\neq j}^{N} \frac{o_m}{\left(\sum_{m=1}^{N} o_m\right)^2} \right] \frac{-1}{n_j} && \text{(A.25)}\\
&= o_j \left[\frac{\sum_{m=1}^{N} o_m - o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} \right] \frac{-1}{n_j} && \text{(A.26)}\\
&= \frac{o_j}{\sum_{m=1}^{N} o_m} \left[\frac{\sum_{m=1}^{N} o_m - o_j}{\sum_{m=1}^{N} o_m} \right] \frac{-1}{n_j} && \text{(A.27)}\\
&= n_j - 1 && \text{(A.28)}
\end{aligned}$$

[8] In [29] this backpropagated derivative is the negative of the δ_{pj} term for example p.

Second, consider the non-target output unit k (i.e. its target output is 0). The effect of the error is transmitted to this unit by the effect of the normalization. Remember, the derivative of the error function with respect to the output of non-target units is 0; thus, only the redistribution from the target unit derivative effects the partial derivative of the error with respect to the non-target units' total inputs. The equation is derived from equation A.9 with equations A.21, A.11 and A.23 substituted. Particular attention should be paid to the second term as it differs from the second term in equation A.25:

$$\frac{\partial E}{\partial x_k} = \frac{\partial o_k}{\partial x_k}\frac{\partial n_j}{\partial o_k}\frac{\partial E}{\partial n_j} \tag{A.29}$$

$$= o_k \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{-1}{n_j} \tag{A.30}$$

$$= \frac{o_k}{\sum_{m=1}^{N} o_m}\frac{-o_j}{\sum_{m=1}^{N} o_m}\frac{-1}{n_j} \tag{A.31}$$

$$= n_k \tag{A.32}$$

Clearly, these are the equations for the partial derivatives of the error with respect to the input of a unit for softmax output units, that is: $\frac{\partial E}{\partial x_j} = n_j - target_j$ (compare to [12]).

The next subsection shows that the same equations are used to derive the error derivatives necessary for softmax units that are not output units.

A.3.2 Softmax Hidden Units

The equations for softmax hidden units are basically identical to the ones for softmax output units, except that the third term in equation A.28 is the backpropagated error from the above layer. Briefly, if we consider the j^{th} unit from equation A.4 we get the following:

$$\frac{\partial E^j}{\partial x_j} = \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E^j}{\partial n_j} \tag{A.33}$$

$$= o_j \left[\frac{\sum_{m=1;m\neq j}^{N} o_m}{\left(\sum_{m=1}^{N} o_m\right)^2}\right]\frac{\partial E^j}{\partial n_j} \tag{A.34}$$

$$= n_j(1-n_j)\frac{\partial E^j}{\partial n_j} \tag{A.35}$$

Once we calculate this term we also have to consider how the other units in the group are affected. From equation A.9 we get this:

$$\frac{\partial E^j}{\partial x_k} = \frac{\partial o_k}{\partial x_k}\frac{\partial n_j}{\partial o_k}\frac{\partial E^j}{\partial n_j} \tag{A.36}$$

$$= o_k \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E^k}{\partial n_j} \tag{A.37}$$

$$= -n_k n_j \frac{\partial E^j}{\partial n_j} \tag{A.38}$$

From the view point of implementation, when unit j is encountered its error has to be propagated through itself, using equation A.35 as well as all the other units in the group using equation A.38. Each unit in turn considers itself as unit j, and distributes its error to all the other units in the group, implementing equation A.3 in the process.

Softmax units are particularly pleasant when used with the cross entropy error function at the outputs, since the derivatives simplify down to a local form. Unfortunately, this is not true of the softmax units at the hidden layers.

It is interesting to note that the equations above can be used for training softmax units to learn a probability distribution using a cross-entropy error function. The only necessary modification involves the $\frac{\partial E^j}{\partial n_j}$ term being non-zero for each unit in the group, since there are target values for each of the output units.

A.4 Summary

The equations presented in section A.1 describe the necessary relationships to backpropagate error through normalized units. The important point to remember is that the error backpropagated through a unit j must also be distributed to all the other units. This must be repeated for each unit in the group that has any error backpropagated to it. The equations described are general in the sense that any activation function (with a first derivative) can be used as well as any error function (with a first derivative). Additionally, a similar formulation for normalized weights is derived in section A.2.

A.5 Acknowledgements

Thanks to S. Oore and P. Dayan for helpful discussion.

Appendix B

Typical Example of Scale File

The values in the range file are the typical minimum and maximum values for each parameter. The parameters are in the following order: X, Y, Z, roll, pitch, yaw, interphalangeal angle and metacarpophalangeal angle of the thumb, index, middle, ring and pinkie; abduction angles of the thumb, index, middle, ring and pinkie, thumb rotation, pinkie rotation, wrist pitch and wrist yaw. The bottom values are the minimum and maximum values of all the formant parameters.

```
2.416290 22.607313
-6.393429 3.971875
1.242362 7.072517
-14.703613 17.290649
-31.677179 -6.840190
-13.656143 25.051773
-30.0 90.0
-10.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 90.0
0.0 150.0
-1.200001 13.200001
-0.638309 9.148930
0.416595 34.166622
-22.835785 99.403061
-53.000000 43.000000
-42.719299 45.087723
-11.428608 24.761833
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0
0.0 150.0

0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
0.000000 63.000000
```

Appendix C

Formant Parameter Targets for Static Phonemes

The target formant parameters for each of the static phoneme sounds are listed in tables C.1 (consonants) and C.2 (vowels). The formant synthesizer parameters are between the integer range of 0 and 63. The relation to frequencies and amplitudes are specified in section 4.2.2. The hand configurations corresponding to the consonant sounds are listed in figure 4.1 in section 4.2.2.

The hand coordinates for each static vowel sound are listed in figure 4.4 in section 4.2.2 but are also listed with each vowel sound in table C.2. Notice, the coordinates are relative to the neutral vowel A. The final coordinates are then specified by placing the neutral vowel at an offset from where the polhemus source is situated. For example, when the source is mounted on an armchair in the lab the offset is 10 cm in the X direction and 0 cm in the Y direction.

Additionally, the formant parameters used for stop consonants are listed in tables C.3 and C.4. The formant parameters for each stop consonant vary over time. The buttons which produce each stop consonant are also listed in table C.1. Remember, the NG formant parameters are sustained as long as the user presses the G key. For the other stops, the initial sequence is produced whenever the corresponding key is pressed. If the key is held down the final parameters are smoothly interpolated to the current output of the consonant and vowel networks in 5 time steps.

phoneme	ALF	F1	A1	F2	A2	F3	A3	AHF	V
W	57	7	53	5	52	18	45	48	45
L	54	10	55	7	56	30	54	54	63
R	57	6	60	12	62	6	58	15	63
V	42	7	42	17	46	25	46	52	36
F	15	12	15	17	47	25	47	52	1
DH	42	7	42	16	41	27	43	50	36
TH	15	12	15	16	41	27	43	50	1
Z	42	7	42	24	46	26	46	65	36
S	15	12	15	24	46	26	46	65	1
ZH	42	7	42	25	57	20	63	63	36
SH	15	12	15	25	57	20	63	63	1
H	43	26	49	24	59	30	58	58	1
M	57	10	35	12	45	22	48	15	63
N	57	6	39	22	50	24	50	35	63

Table C.1: Formant parameters for static consonants

phoneme	X	Y	ALF	F1	A1	F2	A2	F3	A3	AHF
EE	-3.5	7	57	6	58	39	53	38	60	58
I	0	5.0	57	12	57	30	57	30	62	60
E	3	3.5	57	19	57	28	62	29	63	61
AA	5.5	2	57	26	57	24	65	28	64	61
U	5.5	-2.5	57	26	57	14	61	30	55	57
AR	3.5	-3.5	57	24	60	11	63	33	56	59
O	1.7	-4.2	57	20	61	7	62	31	50	55
AW	0	-5	57	16	61	5	59	31	47	50
OO	-1.2	-4	57	14	59	7	55	28	52	55
UU	-3.5	-2	57	8	58	10	48	16	43	47
A	0	0	57	18	58	18	59	26	61	65

Table C.2: Formant parameters for static vowels

phoneme	ALF	F1	A1	F2	A2	F3	A3	AHF	V
B	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	48	3	51	12	15	24	15	15	63
	56	3	57	12	58	24	53	53	32
	56	3	57	12	58	24	53	53	32
D	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	56	3	57	25	49	27	46	58	32
	56	3	57	25	49	27	46	58	32
G	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	48	3	56	23	15	20	15	15	63
	56	3	56	23	60	20	58	47	32
	56	3	56	23	60	20	58	47	32
J	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	48	3	51	25	15	27	15	15	63
	45	5	47	25	48	22	55	57	46
	44	6	45	25	51	21	58	60	39
	42	6	45	25	55	20	62	61	36
	43	7	44	27	56	22	63	65	39
	47	7	48	30	55	27	62	63	46
NG	57	30	29	31	45	24	50	15	63
	57	30	29	31	45	24	50	15	63

Table C.3: Formant parameters for voiced stop consonants (start to finish corresponds with top to bottom)

phoneme	ALF	F1	A1	F2	A2	F3	A3	AHF	V
P	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	3	15	12	15	24	15	15	1
	15	1	39	12	53	24	46	53	1
	15	1	39	12	53	24	46	53	1
T	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	40	27	51	58	1
	15	3	15	25	40	27	51	58	1
K	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	15	23	15	20	15	15	1
	15	3	47	23	57	20	54	35	1
	15	3	47	23	57	20	54	35	1
CH	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	15	27	15	15	1
	15	3	15	25	40	27	51	58	1
	15	3	15	25	40	27	51	58	1
	6	8	6	25	48	22	55	57	1
	10	10	10	25	52	20	59	61	1
	13	12	13	25	55	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1
	15	12	15	25	57	20	62	62	1

Table C.4: Formant parameters for voiced stop consonants (start to finish corresponds with top to bottom)