Glove-TalkII—A Neural-Network Interface which Maps Gestures to Parallel Formant Speech Synthesizer Controls

S. Sidney Fels and Geoffrey E. Hinton

Abstract— **Glove-TalkII is a system which translates hand gestures to speech through an adaptive interface. Hand gestures are mapped continuously to ten 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 ContactGlove, a three-space tracker, and a foot pedal), a parallel formant speech synthesizer, and three 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 but with far more natural sounding pitch variations than a text-to-speech synthesizer.**

I. Introduction

ADAPTIVE interfaces are a natural and important class of applications for neural networks. When a person must provide high bandwidth control of a complex physical device, a compatible mapping between the person's movements and the behavior of the device becomes crucial. With many devices, the mapping is fixed and if a poor mapping is used, the device is difficult to control. Using adaptive neural networks, it is possible to build device interfaces where the mapping adapts automatically during a training phase. Such adaptive interfaces would simplify the process of designing a compatible mapping and would also allow the mapping to be tailored to each individual user. The key features of neural networks in the context of adaptive interfaces are the following.

- Neural networks learn input–output functions from examples provided by the user who demonstrates the input that should lead to a specified output. This "extensional" programming requires no computer expertise.
- Adapting the interface to the peculiarities of a new user is simple. The new user has only to create example data to retrain the network.
- Once trained, the networks run very quickly, even on a serial machine. Also, neural networks are inherently suitable for parallel computation.

In this paper, neural networks are used to implement an adaptive interface, called Glove-TalkII, which maps hand gestures to control parameters of a parallel formant speech synthesizer to allow a user to speak with his hands.

There are many different possible schemes for converting hand gestures to speech. The choice of scheme depends on the granularity of the speech that you want to produce. Fig. 1 identifies a spectrum defined by possible divisions of speech based on the duration of the sound for each granularity. What is interesting is that in general, the coarser the division of speech, the smaller the bandwidth necessary for the user. In contrast, where the granularity of speech is on the order of articulatory muscle movements [i.e., the artificial vocal tract (AVT)] high-bandwidth control is necessary for good speech. Devices which implement this model of speech production are like musical instruments which produce speech sounds. The user must control the timing of sounds to produce speech much as a musician plays notes to produce music. The AVT allows unlimited vocabulary, control of pitch, and nonverbal sounds. Glove-TalkII is an adaptive interface that implements an AVT.

Translating gestures to speech using an AVT model has a long history beginning in the late 1700's. Systems developed include a bellows-driven hand-varied resonator tube with auxiliary controls (1790's [13]), a rubber-molded skull with actuators for manipulating tongue and jaw position (1880's [1]), and a keyboard-foot pedal interface controlling a set of linearly spaced bandpass frequency generators called the Voder (1940 [4]). The Voder was demonstrated at the World's Fair in 1939 by operators who had trained continuously for one year to learn to speak with the system. This suggests that the task of speaking with a gestural interface is very difficult and the training times could be significantly decreased with a better interface. Glove-TalkII is implemented with neural networks which allow the system to learn the user's interpretation of an articulatory model of speaking.

Manuscript received August 14, 1994; revised December 31, 1994. This paper originally appeared in the September 1997 issue and is republished here because the authors' corrections were not implemented by the IEEE Transactions/Journals Department in the earlier published version. This work was supported by the Institute for Robotics and Intelligent Systems and NSERC.

S. S. Fels is with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, V6T 1Z4.

G. E. Hinton is with the Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 1A4.

Publisher Item Identifier S 1045-9227(98)01871-2.

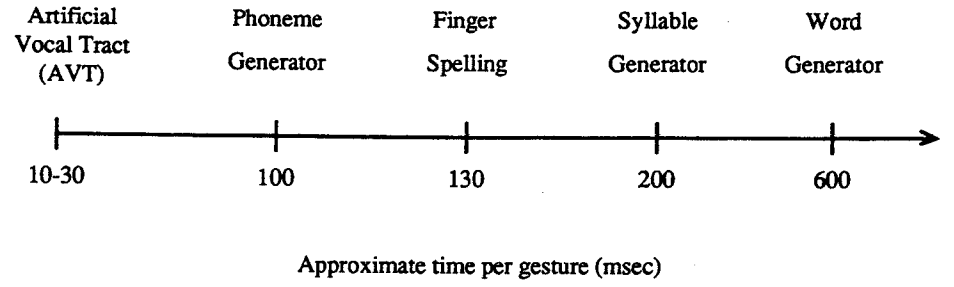

Fig. 1. Spectrum of gesture-to-speech mappings based on the granularity of speech.

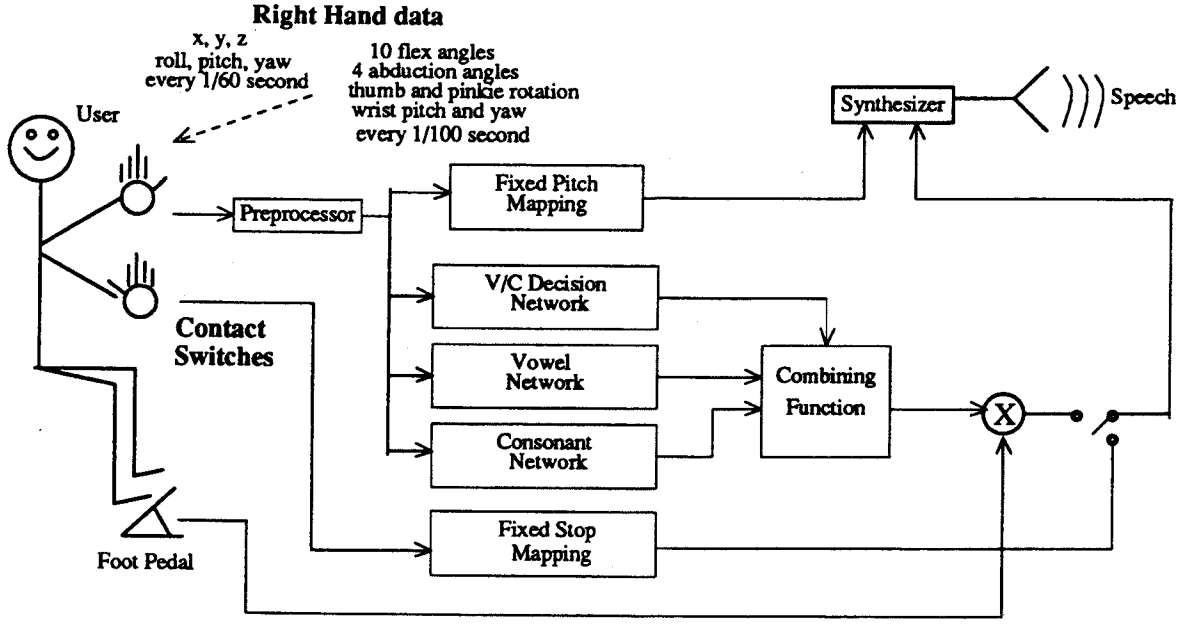

Fig. 2. Block diagram of Glove-TalkII: input from the user is measured by the Cyberglove, polhemus, ContactGlove, and foot pedal, then mapped using neural networks and fixed functions to formant parameters which drive the parallel formant synthesizer [12].

This paper begins with an overview of the whole Glove-TalkII system. Then each neural network is described along with its training and test results. Finally, a qualitative analysis is provided of the speech produced by a single subject after 100 hours of speaking with Glove-TalkII.

II. Overview of Glove-TalkII

The Glove-TalkII system converts hand gestures to speech, based on a gesture-to-formant model. The gesture vocabulary is based on a vocal-articulator model of the hand. By dividing the mapping tasks into independent subtasks, a substantial reduction in network size and training time is possible (see [5] and [6]).

Fig. 2 illustrates the whole Glove-TalkII system. Important features include the input devices, the three neural networks labeled vowel/consonant (V/C), vowel, and consonant, and the speech output device. Input to the system is measured with a Cyberglove, polhemus sensor, keyboard, and foot pedal. The Cyberglove measures 18 angles of the user's hand every 10 ms including two flex angles for each finger (metacarpophalangeal and proximal interphalangeal joints) and abduction angles. The polhemus sensor measures six degrees of freedom of the hand including the X, Y, Z, roll, pitch, and yaw of the user's hand relative to a fixed source. The ContactGlove measures nine contact points between fingers and thumb on the left hand. The foot pedal measures the depression angle of the pedal. These inputs are mapped to speech using three neural networks and other fixed mappings.

The V/C network is trained on data collected from the user to decide whether he wants to produce a vowel or consonant sound. Likewise, the consonant network is trained to produce consonant sounds based on user-generated examples of phoneme sounds defined in an initial gesture of vocabulary. In contrast, the vowel network implements a fixed mapping between hand positions and vowel phonemes defined by the user. Nine contact points on the ContactGlove designate the stop consonants (B, D, G, J, P, K, CH, T, and NG),[1] because the dynamics of such sounds proved too fast to be controlled by the user. The foot pedal provides a volume control by adjusting the speech amplitude and this 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.

[1] Capital letters are used to indicate phonemes available from the text-to-speech synthesizer to differentiate them from phoneme representations like the international phonetic alphabet (IPA).

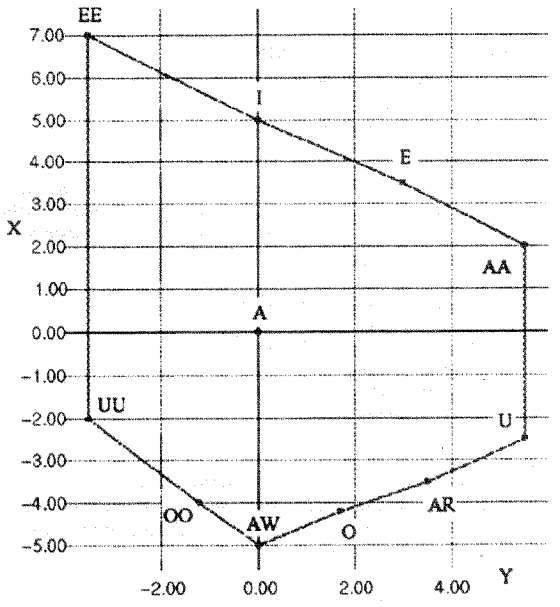

Fig. 3. Hand position to vowel sound mapping. The coordinates are specified relative to the origin at the sound A. The X and Y coordinates form a horizontal plane when the user is sitting. The text-to-speech synthesizer uses 11 cardinal phonemes which are used as targets for the vowel network.

The system sends ten control parameters to a Loughborough Sound Images parallel formant speech synthesizer 100 times/s [12]. The ten parameters are nasal formant amplitude (ALF), first, second, and third formant frequency and amplitude (F1, A1, F2, A2, F3, and A3), high-frequency amplitude (AHF), degree of voicing (V), and the fundamental frequency (F0). Each of the control parameters is quantized to 6 b.

Once trained, Glove-TalkII can be used as follows: to initiate speech, the user forms the hand shape of the first sound he/she intends to produce. The user depresses the foot pedal and the sound comes out of the synthesizer. Vowels and consonants of various qualities are produced in a continuous fashion through the appropriate coordination of hand and foot motions. 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 his/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 *a priori*. The mapping is based on a simplistic articulatory phonetic description of speech [8]. The X,Y coordinates (measured by the polhemus) are mapped to something like tongue position and height[2] producing vowels when the user's hand is in an open configuration (see Fig. 3 for the correspondence and Table I for a typical vowel configuration). Manner and place of articulation for nonstop consonants are determined by opposition of the thumb with the index and middle fingers as described in Table I. The ring finger controls voicing. Only *static* articulatory configurations are used as training points for the neural networks, and the interpolation between them is a result of the learning but is not explicitly trained. Ideally, the transitions should also be learned, but in the text-to-speech formant data we use for training [9] these transitions are poor, and it is very hard to extract accurate formant trajectories from real speech to use for training. The next sections describe the structure and training of each of the three different neural networks.

[2] In reality, the X,Y coordinates map more closely to changes in the first two formants, F1 and F2 of vowels. From the user's perspective, though, the link to tongue movement is useful.

TABLE I
STATIC GESTURE-TO-CONSONANT MAPPING FOR ALL PHONEMES. NOTE, EACH GESTURE CORRESPONDS TO A STATIC NONSTOP CONSONANT PHONEME GENERATED BY THE TEXT-TO-SPEECH SYNTHESIZER

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

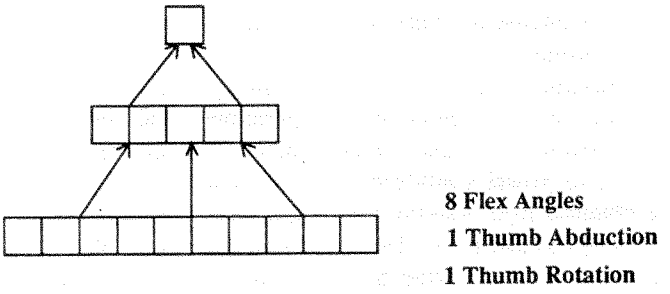

Fig. 4. The vowel/consonant decision network; the ten inputs are the flex angles of the thumb, index, middle, and ring finger, the abduction angle between the thumb and index finger, and the angle of rotation of the thumb (circumduction). The output is the probability that the user intends a vowel.

A. *The Vowel/Consonant (V/C) Network*

The V/C network decides, on the basis of the current configuration of the user's hand, whether to emit a vowel or a consonant sound. For the quantitative results reported here, we used a 10–5–1 feedforward network with sigmoid activations [11] as shown in Fig. 4. The ten inputs are ten scaled hand parameters measured with a Cyberglove: Eight flex angles (metacarpophalangeal and metacarpocarpal/trapeziometacarpal joints of the thumb, index, middle, and ring fingers), thumb abduction angle and thumb rotation angle (circumduction). The output is a single number trained to represent the probability that the hand configuration indicated a vowel.[3] The output of the V/C network is used to determine the mixture of vowel and consonant formant parameters. The training data

[3] The interpretation of the V/C network output representing a probability comes from the fact that the training data targets are interpreted as a probability of a vowel sound.

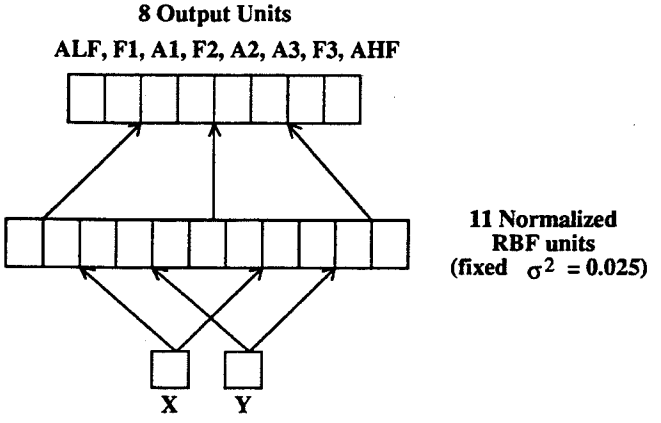

Fig. 5. The vowel network; the input is the X and Y position of the user's hand. The hidden units are RBF units with centers determined from the mapping in Fig. 3 and variances of $\sqrt{0.05}$. The output units are sigmoid units.

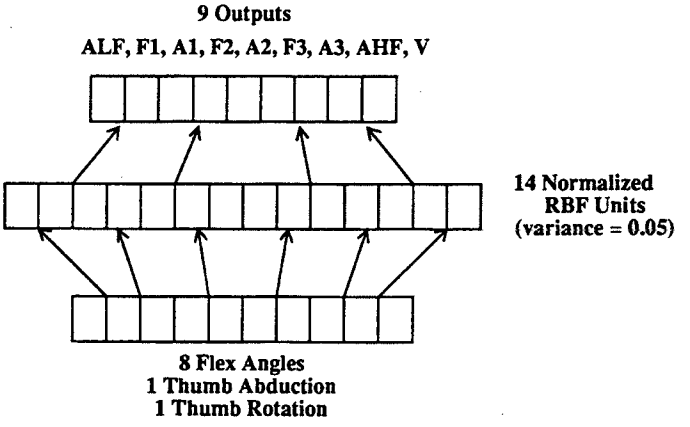

Fig. 6. The consonant network; the same ten hand angles are used for input to the V/C network are used for the consonant network. The hidden units are RBF units with centers determined from the within class average of the hand parameters collected during training. The standard deviations of the RBF units are fixed at $\sqrt{0.05}$. The output units are sigmoid units.

available includes only user-produced vowel or consonant sounds. The network interpolates between hand configurations to create a smooth but fairly rapid transition between vowels and consonants.

For quantitative analysis, typical training data consists of 2600 examples of consonant configurations (350 approximants, 1510 frictives [and aspirant], and 740 nasals) and 700 examples of vowel configurations. The consonant examples were obtained from training data collected for the consonant network by an expert user. The vowel examples were collected from the user by requiring him to move his hand in vowel configurations for a specified amount of time. This procedure was performed in several sessions. The test set consists of 1614 examples (1380 consonants and 234 vowels). After training,[4] the mean squared error on the training and test set was less than 10^{-4}.

During normal speaking the V/C network made no perceptual errors. 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 configurations that would intuitively be expected to produce consonant sounds do indeed produce consonant sounds.

B. The Vowel Network

The vowel network is a 2–11–8 feedforward network as shown in Fig. 5. The 11 hidden units are normalized radial basis functions (RBF's) [3] which are centered to respond to one of the 11 cardinal vowels. The outputs are sigmoid units representing eight synthesizer control parameters (ALF, F1, A1, F2, A2, F3, A3, AHF). The RBF used is

$$o_j = e^{\frac{-\sum(w_{ji}-o_i)^2}{\sigma_j^2}} \tag{1}$$

where o_j is the (unnormalized) output of the RBF unit, w_{ji} is the weight from unit i to unit j, o_i is the output of input unit i, and θ_j^2 is the variance of the RBF. The normalization used is

$$n_j = \frac{o_j}{\sum_{\omega\in\Omega} o_\omega} \tag{2}$$

where n_j is the normalized output of unit j and the summation is over all the units in the group Ω of normalized RBF units. The derivatives for the normalized units needed for gradient-based learning are derived in the Appendix. The centers of the RBF units are fixed according to the X and Y valued of each of the 11 vowels in the predefined mapping (see Fig. 3). The variances of the 11 RBF's were determined empirically and σ^2 is set to 0.025.

The weights from the RBF units to the output units are trained. For the training data, 100 identical examples of each vowel are generated from their corresponding X and Y positions in the user-defined mapping, providing 1100 examples. Noise is then 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 ± 0.5 cm and a Y range of ± 0.26 cm.

Three different test sets were created. Each test set had 50 examples of each vowel for a total of 550 examples. The first test set used additive uniform noise in the interval ± 0.025. The second and third test sets used additive uniform noise in the interval ± 0.05 and ± 0.1, respectively.

The MSE on the training set was 0.0016. The MSE on the additive noise test sets (noise = ± 0.025, 0.05, and 0.01) was 0.0018, 0.0038, and 0.0120, which corresponds to expected errors of 1.1%, 3.1%, and 5.5% in the formant parameters, respectively. This network performs well perceptually. The key feature is the normalization of the RBF units. Often, when speaking, the user will overshoot cardinal vowel positions (especially when producing dipthongs) and all the RBF units will be quite suppressed. However, the normalization magnifies any slight difference between the activities of the units, so the sound produced will be dominated by the cardinal vowel corresponding to the one whose center is closest in hand space.

[4]The V/C network, the vowel network, and the consonant network are trained using a conjugate gradient descent optimization technique combined with a line search.

C. The Consonant Network

The consonant network discussed here for qualitative analysis is a 10–14–9 feedforward network shown in Fig. 6. The 14 hidden units are normalized RBF units. Each RBF is centered at a hand configuration determined from training data collected from the user corresponding to one of 14 static consonant phonemes. The target consonants are created with a text-to-speech synthesizer. Table I defines the initial mapping for each of the 14 consonants. The nine sigmoid output units represent nine control parameters of the formant synthesizer (ALF, F1, A1, F2, A2, F3, A3, AHF, V). The voicing parameter is required since consonant sounds have different degrees of voicing. The inputs are the same as for the V/C decision network: ten hand parameters from the Cyberglove.

Training and test data for the consonant network is obtained from the user. Target data is created for each of the 14 consonant sounds using the text-to-speech synthesizer. The scheme to collect data for a single consonant data is as follows.

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

Using this procedure 350 approximants, 1510 fricatives, and 700 nasals were collected and scaled for the training data. The hand data were averaged for each consonant sound to form the RBF centers. For the test data, 255 approximants, 960 fricatives, and 165 nasals were collected and scaled. The RBF standard deviations were determined empirically and set to $\sqrt{0.05}$.

The MSE on the training set was 0.005 and on the testing set was 0.01, corresponding to expected errors of 3.3% and 4.7% in the formant parameters, respectively. Of course, a low (or high) MSE does not necessarily mean that the speech produced will be of high quality or even intelligible. The MSE does serve as a guide, though, to determine whether the network is going to work reasonably, after which perceptual testing must be performed. Listening to the output of the network reveals that each sound is produced reasonably well when the user's hand is held in a fixed position. The only difficulty is that the "R" and "L" sounds are very sensitive to motion of the index finger.

III. Qualitative Performance of Glove-TalkII

One subject, who is an accomplished pianist, has been trained extensively to speak with Glove-TalkII. We expected his preexisting skill in forming finger patterns and his musical training would help him learn to speak with Glove-TalkII. After 100 hours of training, his speech with Glove-TalkII is intelligible and somewhat natural-sounding. He still finds it difficult to speak quickly, pronounce polysyllabic words, and speak spontaneously. While learning to speak with Glove-TalkII, the user progressed through eight distinct stages:

1) familiarize user with system;
2) initial training of the V/C and consonant networks with user-supplied training data;
3) individual phoneme formation within simple words and CV/VC pairings;
4) word formation and interword pitch control;
5) short segment formation with suprasegmantal pitch control and singing;
6) passage reading;
7) fine tuning: movement control and phrasing;
8) spontaneous speech.

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 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.

During his training, Glove-TalkII also adapted to suit changes required by the subject. Initially, good performance of the V/C network is critical for the user to learn to speak. If the V/C network performs poorly, the user hears a mixture of vowel and consonant sounds, making it difficult to adjust his hand configurations to say different utterances. For this reason, it is important to have the user comfortable with the initial mapping so that training data collected leads to the V/C network performing well. In the 100 h of practice, Glove-TalkII was retrained about 15 times.[5] Four significant changes were made for the new user from the original system analyzed in Sections II-A, II-B, and II-C. First, the "NG" sound was added to the nonstop consonant list by adding an additional hand shape. For the "NG" sound the user touches his pinkie to his thumb on his *right* hand. To accommodate this change, the consonant and V/C network had two inputs added to represent the two flex angles of the pinkie. Also, the consonant network has an extra hidden unit for the "ng" sound. Second, the consonant network was trained to allow the RBF centers to change. This was done by first training the hidden-to-output weights until little improvement was seen. Then, both the input-to-hidden weights (i.e., the RBF centers) and the hidden-to-output weights were allowed to adapt. This noticeably improved performance for the user. Third, the vowel mapping was altered so that the "I" was moved closer to the "EE" sound and the entire mapping was reduced to 75% of its size. Fourth, for this subject, the V/C network needed was a 12–10–1 feedforward sigmoid unit network. It is anticipated this network will be sufficient for most users. Understanding the interaction between the user's adaptation and Glove-TalkII's adaptation remains an interesting research pursuit.

[5] Far fewer retrainings would be required for future users.

IV. SUMMARY

The initial mapping for Glove-TalkII 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 a horizontal X-Y plane that corresponds to movements of the first two formants which are roughly related to tongue position. Consonants other than stops are produced by closing the index, middle, or ring fingers or flexing the thumb, representing constrictions in the vocal tract. Stop consonants are produced by contact switches worn on the left hand. F0 is controlled by hand height and speaking intensity by foot pedal depression.

Glove-TalkII learns the user's interpretation of this initial mapping. The V/C network and the consonant network learn the mapping from examples generated by the user during phases of training. The vowel network is trained on examples computed from the user-defined mapping between hand position and vowels. The F0 and volume mappings are nonadaptive. In many interface applications it is necessary to map from a user's gesture space to control a complex device. The methods used to build Glove-TalkII can be applied to these interfaces.

One subject was trained to use Glove-TalkII. After 100 hours of practice he is able to speak intelligibly. His speech is fairly slow (1.5 to three times slower than normal speech). It sounds similar to speech produced with a text-to-speech synthesizer but has a far more natural intonation contour which greatly improves the intelligibility and expressiveness of the speech. Reading novel passages intelligibly usually requires several attempts, especially with polysyllabic words. Intelligible spontaneous speech such as found in conversation is possible but difficult. It is anticipated that training on formant trajectories estimated from real speech will significantly improve the quality of the user's speech. Currently, it is an open research issue to extract accurate formant trajectories automatically [10].

Glove-TalkII could be used by speech-impaired people. To make it useful for this community it must be made portable and inexpensive. The current implementation requires a machine capable of approximately 200 000 floating point operations/s that has three serial ports and a parallel port. Laptop computers are already available to fit these requirements. The glove and foot pedal devices are tethered to the machine, which limits mobility of the user. To solve this limitation it is necessary for the input devices to have a system of wireless transmission of the data to the host computer or to mount the host computer on the user. Finally, the foot pedal would be cumbersome in a completely portable system, but it should be possible to design an alternative method of controlling the single volume parameter. Once these technical issues are resolved, Glove-TalkII could provide a portable, inexpensive, adaptive, artificial vocal tract device to assist speech-impaired people.

APPENDIX
DERIVATION OF GRADIENTS FOR NORMALIZED UNITS

Often when designing neural networks, one has occasion to use normalized units (see, for example, [7]). 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 [4]). 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' targets 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 RBF's;
- 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.[6] The well-known equations for softmax output units can be derived from this more general formulation. The equations for softmax units in the hidden layers will also be derived.

Consider the following notation (similar to [11]). First, the activation function for unit j in the normalized group is

$$o_j = f(x_j) \tag{3}$$

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).[7] For illustration purposes, consider the normalized output of the unit as

$$n_j = \frac{o_j}{\sum_{m=1}^{N} o_m} \tag{4}$$

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

The implementation of the forward propagation through these normalized units is straightforward. 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.[8]

Complications arise because error propagated back to one unit in the group will affect the others.

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

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

[8] From $\frac{\partial E_p}{\partial x_j}$ we can easily get $\frac{\partial E_p}{\partial w_{ji}}$ in the manner described in [11].

Consider the 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

$$\frac{\partial E}{\partial x_j} = \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E}{\partial n_j} + \sum_{k=1;k\neq j}^{N} \frac{\partial o_j}{\partial x_j}\frac{\partial n_k}{\partial o_j}\frac{\partial E}{\partial n_k}. \quad (5)$$

This expression is much easier to understand if we 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

$$\frac{\partial E^j}{\partial x_j} = \frac{\partial o_j}{\partial x_j}\frac{\partial n_j}{\partial o_j}\frac{\partial E^j}{\partial n_j}. \quad (6)$$

The first term from (3) is simply

$$\frac{\partial o_j}{\partial x_j} = f'(x_j). \quad (7)$$

The second term is obtained from (4)

$$\frac{\partial n_j}{\partial o_j} = \frac{1}{\sum_{m=1}^{N} o_m} + \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} \quad (8)$$

$$= \frac{\sum_{m=1}^{N} o_m - o_j}{\left(\sum_{m=1}^{N} o_m\right)^2} \quad (9)$$

$$= \frac{\sum_{m=1;m\neq j}^{N} o_m}{\left(\sum_{m=1}^{N} o_m\right)^2}. \quad (10)$$

The third term has two different forms depending on 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 [11] 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 when k is some other unit besides j. What is the effect on the error by unit k? The equation is derived as follows:

$$\frac{\partial E^j}{\partial x_k} = \frac{\partial o_k}{\partial x_k}\frac{\partial n_j}{\partial o_k}\frac{\partial E^j}{\partial n_j}. \quad (11)$$

The important point to notice in this equation is that *the change in the error with respect to x_k is a function of the change in error with respect to 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 (11), the first term is [from (3)]

$$\frac{\partial o_k}{\partial x_k} = f'(x_k). \quad (12)$$

The second term follows from (4)

$$\frac{\partial n_j}{\partial o_k} = \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}. \quad (13)$$

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 (or both). 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 (5) is calculated. Using this method, the complexity of calculating the required partial derivatives 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 (4) we can write (5) as

$$\frac{\partial E}{\partial x_j} = f'(x_j)\left[\left[\frac{\sum_{m=1}^{N} o_m - o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\right]\frac{\partial E}{\partial n_j} + \sum_{k=1,k\neq j}^{N} \frac{-o_j}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k}\right] \quad (14)$$

which reduces[9] after simple algebraic manipulation to

$$\frac{\partial E}{\partial x_j} = f'(x_j)\left[\frac{1}{\sum_{m=1}^{N} o_m}\frac{\partial E}{\partial n_j} - o_j \sum_{k=1}^{N} \frac{1}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k}\right]. \quad (15)$$

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

$$\sum_{k=1}^{N} \frac{1}{\left(\sum_{m=1}^{N} o_m\right)^2}\frac{\partial E}{\partial n_k} \quad (16)$$

which can be done in $O(N)$ calculations Second, for each unit in the group, calculate the required partial derivative using (15) and the term from (16) 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 many cases this type of normalization is used.[10]

[9] This reduction was pointed out by S. Becker and T. Plate, personal communication.

[10] L_2 normalization reduces to a similar form.

ACKNOWLEDGMENT

The authors thank J. Bridle, D. van Camp, P. Dayan, S. Oore, M. Revow, and M. Ruicci for their contributions.

REFERENCES

[1] A. G. Bell, "Making a talking machine," *Beinn Bhreagh Rec.*, pp. 61–72, Nov. 1909.

[2] J. S. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Neural Information Processing Systems*, vol. 2, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990.

[3] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, pp. 321–355, 1988.

[4] H. Dudley, R. R. Riesz, and S. S. A. Watkins, "A synthetic speaker," *J. Franklin Inst.*, vol. 227, no. 6, pp. 739–764, June 1939.

[5] S. S. Fels and G. E. Hinton, "Glove-Talk: A neural-network interface between a data-glove and a speech synthesizer," *IEEE Trans. Neural Networks*, vol. 4, pp. 2–8, Jan. 1993.

[6] S. S. Fels, "Building adaptive interfaces using neural networks: The Glove-Talk pilot study," Univ. Toronto, Canada, Tech. Rep. CRG-TR-90–1, 1990.

[7] 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," Los Alamos National Lab., NM, Tech. Rep. LA-UR-91–273, 1990.

[8] P. Ladefoged, *A Course in Phonetics*, 2nd ed. New York: Harcourt Brace Javanovich, 1982.

[9] E. Lewis, "A 'C' implementation of the JSRU text-to-speech system," Comput. Sci. Dept., Univ. Bristol, U.K., Tech. Rep., 1989.

[10] A. Lowry, M. C. Hall, and P. M. Hughes, "Iterative parameter optimization techniques for parallel-formant encoding of speech," in *Proc. European Conf. Circuit Theory and Design*, 1989, pp. 537–541.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by backpropagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[12] J. M. Rye and J. N. Holmes, "A versatile software parallel-formant speech synthesizer," Joint Speech Res. Unit, Malvern, U.K., Tech. Rep. JSRU-RR-1016, 1982.

[13] W. R. von Kempelen, *Mechanismus der menschlichen Sprache nebst Beschreibung einer sprechenden Maschine*, with an introduction by H. E. Brekle and W. Wild. Stuttgart, Germany: Stuttgart-Bad Cannstatt F. Frommann, 1970.

[14] E. Yair and A. Gersho, "The Boltzmann perceptron network: A multi-layered feedforward network equivalent to the Boltzmann machine," in *Advances in Neural Information Processing Systems I (NIPS'88)*. San Mateo, CA: Morgan Kaufmann, 1989.

S. Sidney Fels received the B.A.Sc. degree in electrical engineering from the University of Waterloo, Ont., Canada, in 1988 and the M.Sc. and Ph.D. degrees in computer science from the University of Toronto, Canada, in 1990 and 1994.

His is an Assistant Professor in the Department of Electrical and Computer Engineering at the University of British Columbia, Vancouver. His current research interests include intelligent agents, gesture-to-music, and interactive art.

Geoffrey E. Hinton received the B.A. degree in psychology from the University of Cambridge, U.K., in 1970, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, U.K., in 1978.

He is a Professor in the Departments of Computer Science and Psychology at the University of Toronto, Canada, and a Fellow of the Canadian Institute for Advanced Research. His research interests include methods of using connectionist networks for learning, memory, perception, symbol processing, and motor control.