Learning to control an articulatory synthesizer by imitating real speech

Ian S. Howard
Sobell Department, Institute of Neurology, UCL, England

Mark A. Huckvale
Phonetics & Linguistics, UCL, England

The goal of our current project is to build a system that can learn to imitate a version of a spoken utterance using an articulatory speech synthesiser. The approach is informed and inspired by knowledge of early infant speech development. Thus we expect our system to reproduce and exploit the utility of infant behaviours such as listening, vocal play, babbling and word imitation. We expect our system to develop a relationship between the sound-making capabilities of its vocal tract and the phonetic/phonological structure of imitated utterances. At the heart of our approach is the learning of an inverse model that relates acoustic and motor representations of speech. The acoustic to auditory mappings uses an auditory filter bank and a self-organizing phase of learning. The inverse model from auditory to vocal tract control parameters is estimated using a babbling phase, in which the vocal tract is essentially driven in a random manner, much like the babbling phase of speech acquisition in infants. The complete system can be used to imitate simple utterances through a direct mapping from sound to control parameters. Our initial results show that this procedure works well for sounds generated by its own voice. Further work is needed to build a phonological control level and achieve better performance with real speech.

1. Introduction

Several different approaches have been adopted in order to get a machine to speak. One approach is to record human speech, chop it up into pieces and then reassemble them in a new desired order. Another approach is to program phonological-to-synthesizer control mapping rules by hand. The approach we take here is to try to discover an acoustic-to-synthesizer control mapping using machine learning techniques. We gain inspiration from infant speech acquisition and in this vain also use an articulator-based synthesiser for our work, to make our system's speech production apparatus more like that of a human. Our work here is obviously only a first step towards producing a useful system, which would also need to learn other associations such as phonological-to-articulatory mapping to be a useful system. Several other authors have made similar investigations. Bailly et al. (1997) modelled the generation of formant trajectories. Guenther (1994, 1995) has also carried out similar work.

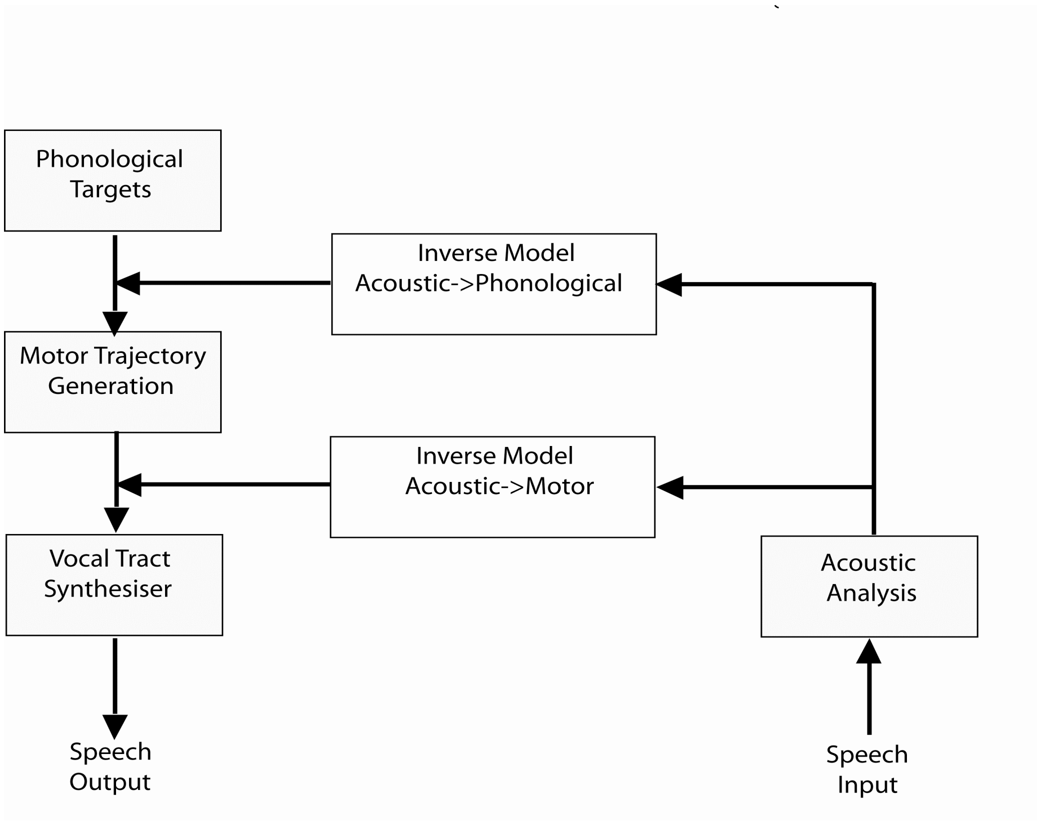
Phonological Targets
Inverse Model Acoustic->Phonological
Motor Trajectory Generation
Inverse Model Acoustic->Motor
Vocal Tract Synthesiser
Acoustic Analysis
Speech Output
Speech Input

Figure 1: Inverse models. Mappings between acoustic, phonological, and articulator representations of speech.

2. Inverse models

Let us consider the speech production and analysis system shown in figure 1. Here we have a system that can both generate artificial speech and also perform a basic acoustic speech analysis.

In this model, the vocal tract synthesiser is controlled by a vector of articulatory parameters which change as a function of time, as specified by a motor trajectory generation stage. It is the task of the latter to move the articulators of the model in such a fashion that the desired speech utterance is generated by the synthesiser.

If we feed back the speech signal generated using this process into the acoustic analysis pathway, our system then has an explicit representation of the motor commands it uses to generate speech, as well as their acoustic consequences. Since we can have access to both the input to our synthesiser, and also its acoustic consequences, we can use this information to define an inverse transformation that will map an acoustic representation of speech back to the motor commands needed to generate it. This inverse transformation is marked on figure 1 as an acoustic-to-motor inverse model. Clearly, for such an inverse model to be useful in practice, it must perform well over a representative range of conditions, corresponding to the kinds of inputs the synthesiser would experience during normal use. It must also account for any time delay between the motor commands and their sensor consequences. In the work described here, this alignment is also performed by the inverse model. A discussion regarding the training of the inverse model is given in the next sections. The concept of inverse models is well established in the field of motor control; see (Wolpert, 1997) for a further discussion of the issues involved.

If our speech production system contained a higher hierarchical level of control, it would also be possible to define an inverse model to a more abstract level of representation. For example, if our motor trajectory generator was controlled by a phonetic input representation, we could define an inverse model pathway mapping between acoustic and phonological representations of speech (also shown in figure 1). In the work described here, we only investigate the low-level acoustic to vocal tract parameter inverse model using an articulator synthesiser based on the work of Maeda (Maeda, 1990) and a simple acoustic analysis based on the JSRU channel vocoder (Holmes, 1982), together with a simple autocorrelation estimate for fundamental frequency. The Maeda parameters are shown in table 1. In our implementation, they are specified at a sampling rate of 8kHz to generate speech signal output at the same rate. This gives the synthesiser an acceptable speech quality without requiring excessive computational resources to run it.

Assuming that we can find the inverse model for our system, it can be used to provide a basic mechanism to control a synthesiser. All we must do is to process input speech with the acoustic analysis and then map this representation to the vocal tract control parameters using the inverse model.

Table 1: Maeda's articulator model parameters.

Parameter	Description
P1	Jaw position
P2	Tongue dorsum position
P3	Tongue dorsum shape
P4	Tongue apex position
P5	Lip height (aperture)
P6	Lip protrusion
P7	Larynx height
P8	Voicing (glottal area)
P9	Fundamental frequency

3. Learning the inverse model by babbling

The main task in the approach we describe here is the estimation of the inverse model that will map between an acoustic representation of speech and the required vocal tract parameters. If we use some kind of motor trajectory generator to vary the vocal tract parameters as a function of time, and then use them to generate synthesised speech (which is then subsequently acoustically analysed), we can create a data set with which to define the input and output relationships of the inverse model. This is shown in figure 2. Training the inverse model then constitutes a classical supervised learning task.

As is the case with regression analysis and pattern recognition in general, we want the inverse model to operate well over a wide range of representative inputs and also to generalize well on previously unseen ones. In order to generate an inverse model that meets these requirements, appropriate training data is required, as well as a suitable pattern recognition/regression technique capable of learning the required transformation. With regard to the generation of suitable training data, the ideal input to the synthesiser would correspond to motor trajectories that resulted in high quality speech output over a wide range of utterances. This input would then sample the vocal tract parameter space in a fashion consistent with its intended use. Training the inverse model on relevant rather than irrelevant data will also result in higher performance, because it will be optimised to implement the transformations that are needed, rather than ones that are not.

At the beginning of the estimation process, the generation of ideal motor trajectories is clearly not possible, because we do not know what they are ***a priori***. We must therefore resort to a method of synthesiser control that will sample its input space in a fashion consistent with our knowledge of speech generation. In the field of control theory, such system identification of a complex system is often carried out using some kind of random input excitation. However, this excitation should be matched to the task in hand. Since we are using an articulatory synthesiser, we know that there are limitations on the maximum rate of change of articulator positions due to biophysical constraints. To sample vocal tract parameter space, we thus adopt an approach that randomly investigates this space, but only does so in a slowly varying fashion consistent with the changes in articulator positions appropriate for the generation of speech. For example, there would be no point in instantaneously moving the jaw position from up to down, because we know that such a change could not occur in a real vocal tract.

We have implemented this slowly varying random signal generation scheme using a Hidden Markov Model (HMM) with output interpolation, as explained below. We describe this as a babble generator, because it generates sounds that have similarities with the babble produced by infants, although the exact form of the sound sequences generated by this approach does not exactly correspond to baby babble. Its task is only to explore the input space of the synthesiser in a way that is useful for the purposes of training the inverse model.

For an inverse transformation to exist, it is necessary for the forward transformation due to the plant (represented here by the synthesizer and acoustic analysis) to be unique. If this is not the case, it will not be possible to find an inverse. That is, if many different vocal tract configurations give rise to the same acoustic output, it will be impossible to know, on the basis of an acoustic measurement, which vocal tract configuration was responsible. One approach that can be adopted in this case is based on distal supervised learning (Jordan & Rumelhart, 1992), which involves first training a simpler forward model and then using this to find the inverse. Although the instantaneous mapping between vocal tract configuration and acoustic output is not in general unique, this problem can also be overcome using a wide acoustic context at the input to the inverse mapping. The latter approach was used here and an acoustic window of 50ms was used as an input to the inverse model.

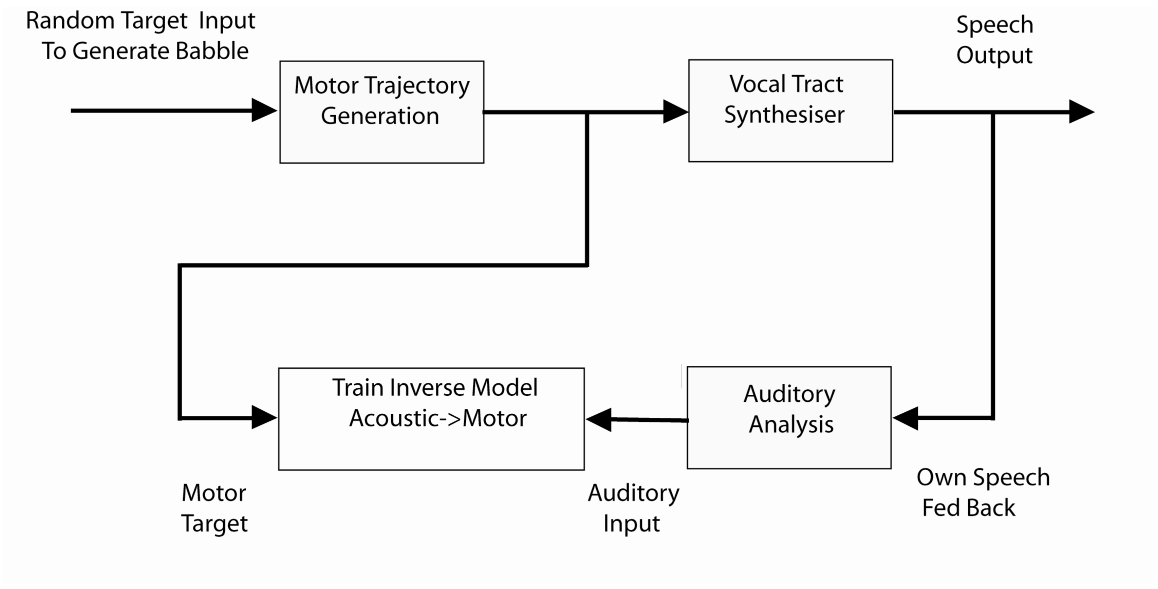
Random Target Input To Generate Babble
Motor Trajectory Generation
Vocal Tract Synthesiser
Speech Output
Train Inverse Model Acoustic->Motor
Auditory Analysis
Motor Target
Auditory Input
Own Speech Fed Back

Figure 2: Learning an inverse model by babbling. Motor command space is sampled using 'random babbling'. Direct training of the inverse model then becomes a classical supervised learning task.

4. Babble generator

The task of the babble generator is to generate sequences of parameters that explore the input space of the synthesiser in a fashion relevant for speech production. The basic idea is to use an HMM to sample phonetically significant regions of synthesiser space and then to interpolate the output to generate a slowly varying time signal vector. By restricting the states of the HMM to phonetics targets (such as vowels and consonant targets) we can incorporate ***a priori*** speech knowledge into the babble generator. Consider the HMM shown in figure 3. Using this model structure, we can sample three points in vocal tract parameter space, represented by the states V1, V2 and V3. Each of these states has associated transition probabilities to the other states and also an associated output parameter vector. We can use this approach to sample part of vowel and consonant space (which forms a subset of the much larger total parameter space) by associating each of the states with the parameters relating to a particular vowel/consonant generated by the articulatory synthesiser. Using a duration parameter in our output vector, we can interpolate between successively sampled vectors to generate continuous trajectories needed to drive the Maeda vocal tract synthesizer.

We have also investigated a more naive scheme by directly sampling randomly from the entire parameter space, rather than from vowel and consonant space. In this case, the output from our babble generator did not sound as much like baby babble. Configurations of the articulators arose that were not relevant in the production of speech. Comparisons of the output speech generated by various babbling schemes can be found online (see section 5).

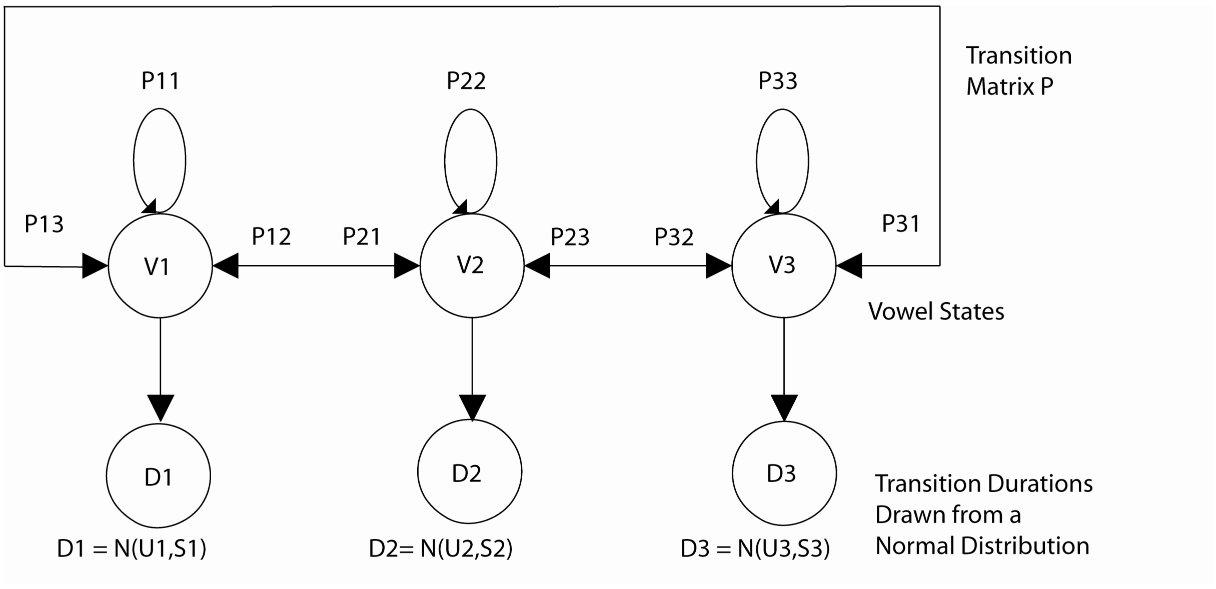

Figure 3: HMM babble generator. Each state represents a possible vocal tract configuration. In this case, only 3 vowels are shown for clarity. The transition matrix determines vowel sequences. The generation matrix specifies related vocal tract parameters which are then interpolated to provide smooth trajectories.

In this initial work, the HMM states were limited to five pure vowels and the consonants /b/ and /g/. By manipulating the transition probabilities between the states it was possible to generate babble either over this entire vowel space and consonant space, or only over a small part of it. An example of the latter was generation of the sequence /bababab/.
To generate smooth parameter trajectories in time at the sampling rate needed by the synthesiser, cosine interpolation was performed on the vectors generated by the state sequences according to the relation

$$\mathbf{VTpar(X) = VTparStart + (Cos(X.PI\ /duration)\ -1)\ .\ (VTparStart - VTparEnd)/2}$$

$$\mathbf{for\ X = 0, 1, 2, \ldots, duration\ samples}$$

Figure 4 shows the parameter trajectory outputs from the babble generator, as well as the corresponding vocoder analysis of the resulting synthesised speech.

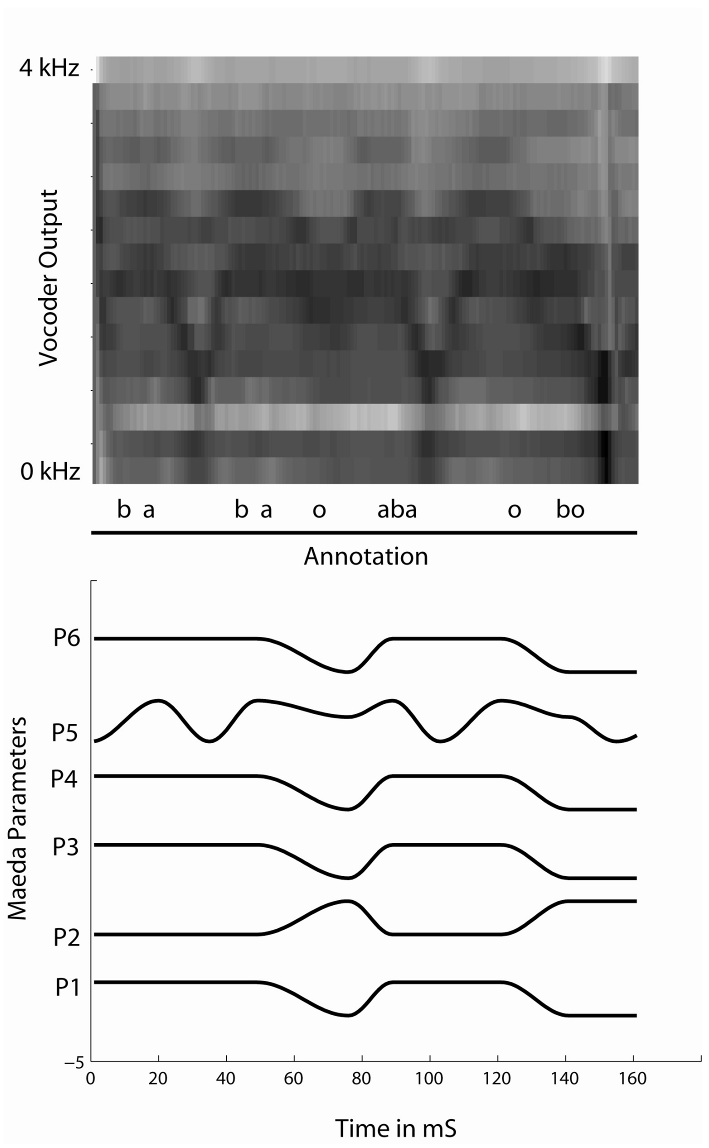

Figure 4: Babble generator output trajectories for parameters 1 to 6 and the corresponding vocoder analysis of the resulting speech signal. In this case the transition matrix was set up to babble over vowel space and also include the consonants /b/ and /g/

5. Experimental details

The babble generator was run on vowel and consonant targets to generate 180 seconds of articulator trajectory data. In our current implementation this data length was limited by computational considerations. This data was then used to drive the Maeda synthesiser and generate a time waveform representation of the output speech signal. The latter was then subjected to acoustic analysis with the channel vocoder, which generated 17 frequency outputs, as well as a fundamental frequency estimate, every 10ms.

For our experiments, a Matlab implementation of the multilayer perceptron (MLP) with linear output units was used to implement the inverse model (Nabney & Bishop, 1995), although in principle many other regression techniques could have been used. Input and output data patterns were normalized by subtracting their mean value and dividing by their standard deviation (after recognition, the inverse procedure was used on the output to reconstruct the estimated data's range). It was trained using back-propagation (Rumelhart et al., 1987). As mentioned previously, the input vector spanned 50ms in time and consisted of 5 adjacent vocoder frames. This time window also provided an automatic means for the inverse model to compensate for any time delay between the acoustic data and the synthesiser control parameters. Since the input to the inverse model spanned 25ms forward and 25ms backwards in time from the current synthesiser control vector, any information in the acoustic data that lay within these limits could be related to the current motor command. If a more time localized representation of the acoustic input had been used (for example a single spectral input frame) it would have been necessary to explicitly account for any time shift (although this could also easily be achieved by using delay lines and optimising the MLP error over delay). The number of hidden units in the MLP was determined by experimentation and a final value of 20 hidden units was used.

The output of the network consisted of 9 linear units, and these were mapped to the 9 Maeda synthesiser parameters. Training the inverse model involved 1000 passes over the data set (which comprised around 18000 different patterns). This value was again arrived at by experimentation and doubling the cycles to 2000 did not significantly improved performance. The output of the inverse model was smoothed using a median filter to remove undesirable spikes. Figure 5 shows the operation of the inverse model on testing data for the jaw position parameter.

Apart from the articulator synthesiser, which was implemented in the C language, all analysis was carried out in Matlab on a PC running under Windows XP. A supplement to this paper is available on the web at www.ianhoward.de/ZASPIL2005.htm and contains .wav files of all the input and output utterances described in this paper.

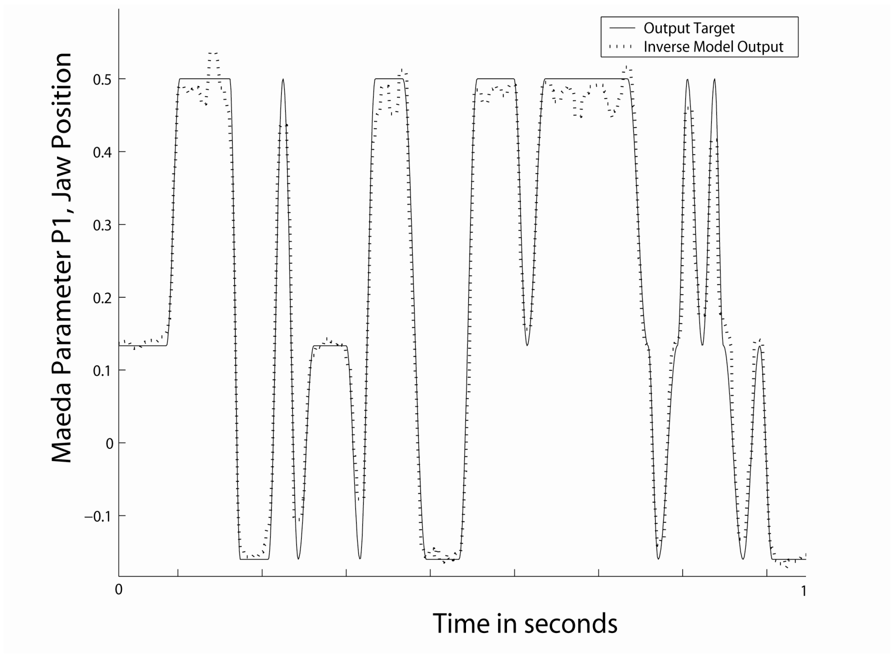

Figure 5: Training target and output generated by the inverse model for jaw parameter P1. A median filter was used to smooth the output.

6. Re-synthesising its own speech

After an inverse model had been trained during a babbling stage, it was clearly interesting to test its performance by re-synthesising some input speech. Evaluations were carried by listening to the speech generated by the system and also by observation of the corresponding wideband spectrograms. At this early stage of the work, this was considered to be the most appropriate form of assessment. In the first instance, we were concerned with whether any useful results could be generated by our system. At some time in the future, more rigorous quantitative evaluations will no doubt become useful. For example, the phonetic analysis of confusion matrices from listening tests on real and re-synthesised utterances could shed light on the deficiencies of the system.

Re-synthesising input speech involved passing an externally generated speech signal (that is, one from another synthesizer or a human subject) through the acoustic analysis and then through the inverse model. This produced a time-varying estimate of the vocal tract control parameters needed to regenerate the original speech utterance. It is clear that there would only be a perfect reconstruction of the input speech if the inverse model were perfect and vocal tract used to generate the speech was identical that of our vocal tract synthesiser. If the two vocal tracts had different physical dimensions, it may not be possible to get an exact reconstruction of the input speech. Such a mismatch arises when children mimic the speech of adults, where there are clearly differences in the relative size of the speech production apparatus. This also manifests itself at an acoustic level. However, at a more abstract phonetic level, similarity between the two can still be achieved. This naturally raises the question of speaker normalization and speech matching criterion. In this initial work we do not focus on these two issues, although they will be investigated in more detail in future work.

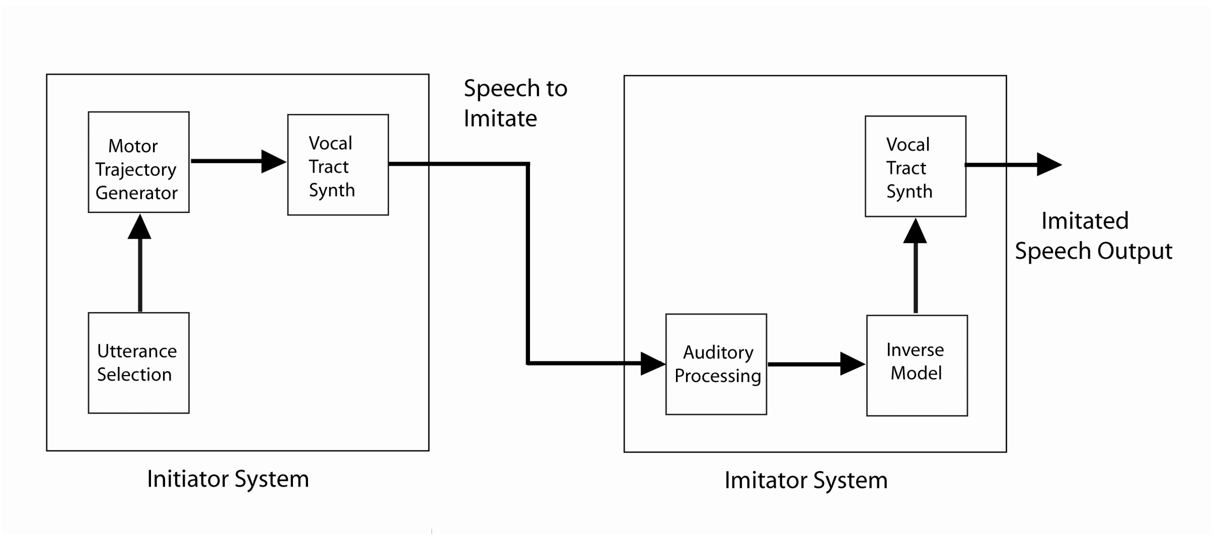

Figure 6: Imitating speech generated by an identical system. This is the simplest case and the issue of speaker normalization does not arise.

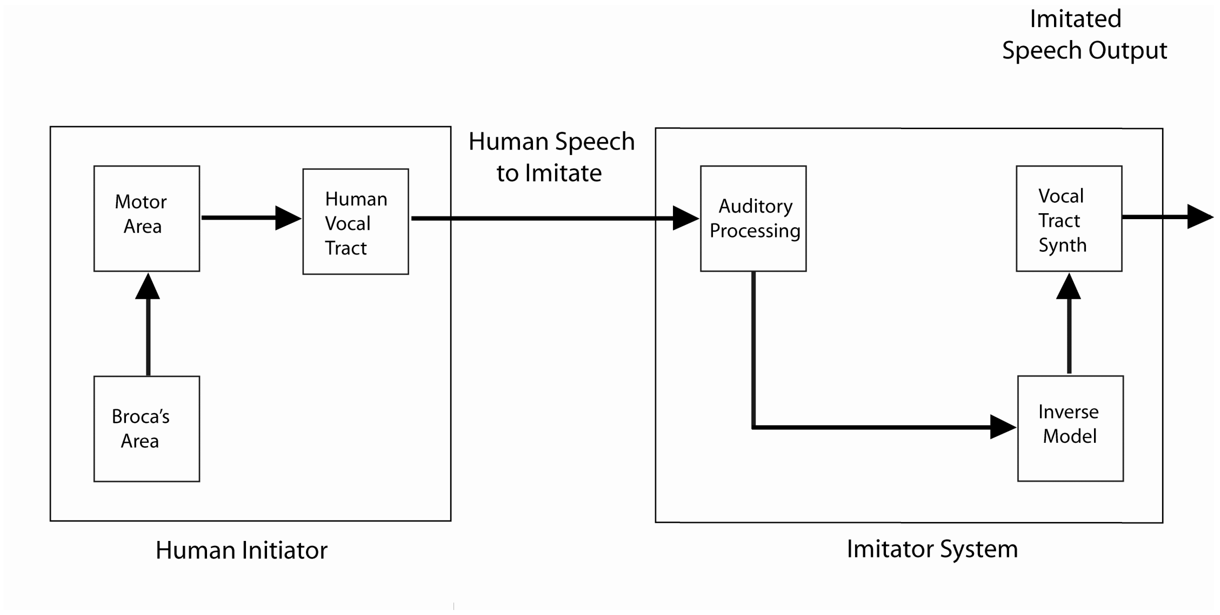

Figure 7: Imitating speech generated by a different (and human) system. In this case speaker normalization is needed.

The simplest task for the inverse mapping between acoustic and vocal tract parameters is the case when both generator and mimic vocal tracts are identical. This is shown in figure 6. We simulated this situation by using the articulator model itself to generate babbled speech utterances (20 seconds of speech was used) and then used these as input to our imitation system. On its own speech, performance was very good (examples on the web), although this could probably still be improved further by using more training data.

7. Re-synthesising speech from human subject

A much more difficult case arises when speech generated by a different vocal tract must be imitated by the system. This is the case when real human speech is used as an input, as shown in figure 7. Simple utterances from one male speaker were used for the evaluation. In this case, the performance was lower, although simple utterances such as /bababababa/ and /bugi bugi bababa/ were still intelligible after re-synthesis (examples on the web). Wideband spectrograms for the input utterance (A) and the synthesiser output (B) are shown in figure 8. It can be seen that the imitated speech has voicing continuing into the silent parts of the utterance. Formant F1 corresponds well to that of the original speech, although F2 tends to be too low during the /i/ vowel section. Formants F3, F4 also appear somewhat too low in their values.

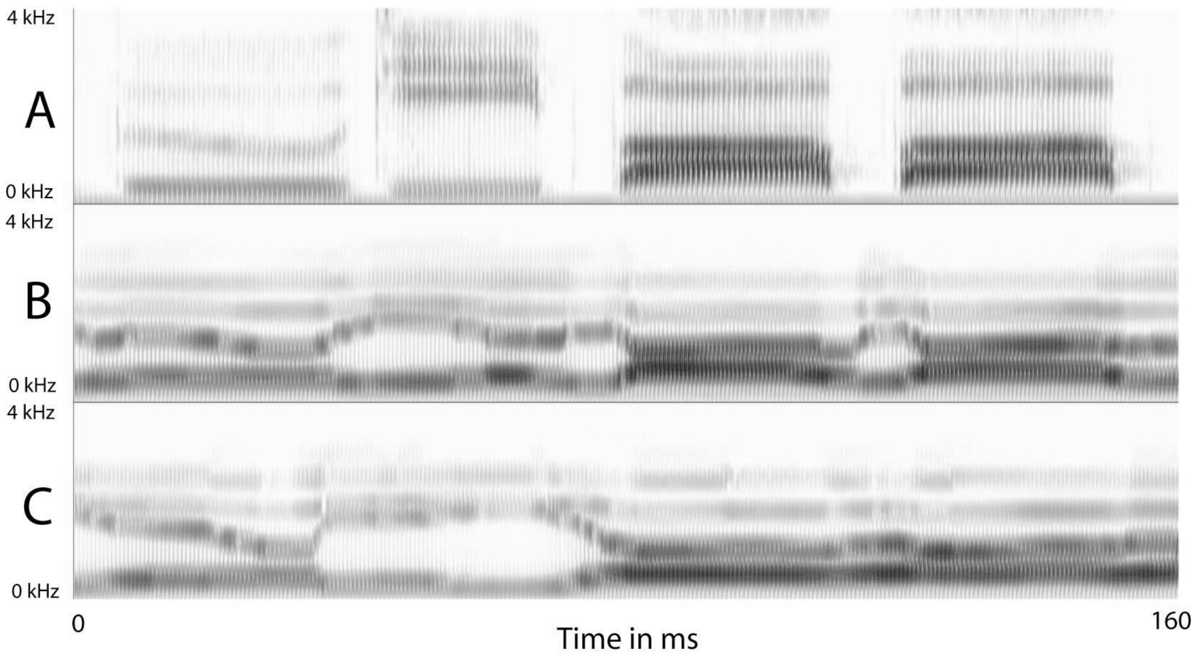

Figure 8: Wideband spectrograms for input utterance /boo gie ba ba/ from male speaker and re-synthesised outputs generated by imitation system. A shows real speech, B re-synthesis using vocoder analysis and C re-synthesis using an additional sparse coding stage.

8. Improved acoustic analysis

In order to improve the performance of the current system, we looked for inspiration from what is known about front end sensory processing in the human nervous system.

It has been known for quite a long time that the lower levels of sensory processing are matched to the statistics of the stimuli that they represent (Barlow, 1962). Recent work in the visual system has been quite successful in modelling receptive fields on neurons in the primary visual cortex by using sparse coding strategies (Olshausen & Field, 1996). Indeed it has been recently shown that such strategies appear relevant in auditory processing (Lewicki, 2002). The interpretation of this processing is still the subject of much debate, but it appears that the early sensory system is involved in both efficient coding as well as the extraction of features in the input modalities that relate to useful aspects of the input. A simple spectral vocoder vector provides quite a general representation of the acoustic input, whereas sparse coding may start to code in terms of features that operate over time and frequency and are specifically relevant for speech. We maintained the vocoder analysis and investigated adding a sparse coding to its output. The input to the latter consisted of 5 adjacent vocoder frames as before. It was implemented in two stages consisting of a whitening filter followed by a sparse filtering stage. The former consisted of linear Sobel edge detection filter and it effectively removed all short-term temporal and spatial correlations in the vocoder data (it effectively differentiated the vocoder data in 2-dimensions). The sparse filter itself was implemented by a 2-dimensional linear filter with as many outputs as inputs (90 in all). Its coefficients were optimized to minimize a cost function that requires the outputs to be both independent and also rarely active. Details of this coding scheme can be found in (Olshausen & Field, 1996). The sparse coding stage also was trained on 360 seconds of speech from one speaker.
After training, the auditory sparse coding stage was run on the babble data and used to train the inverse model as before. Figure 9 shows this modified system. Performance evaluation was once again carried out using real input speech, which was once again run through the inverse model pathway and re-synthesised as before. It was noted that the effect of the sparse coding was to improve performance on transitions (example on the web). Figure 8 C shows a wideband spectrogram of the output.

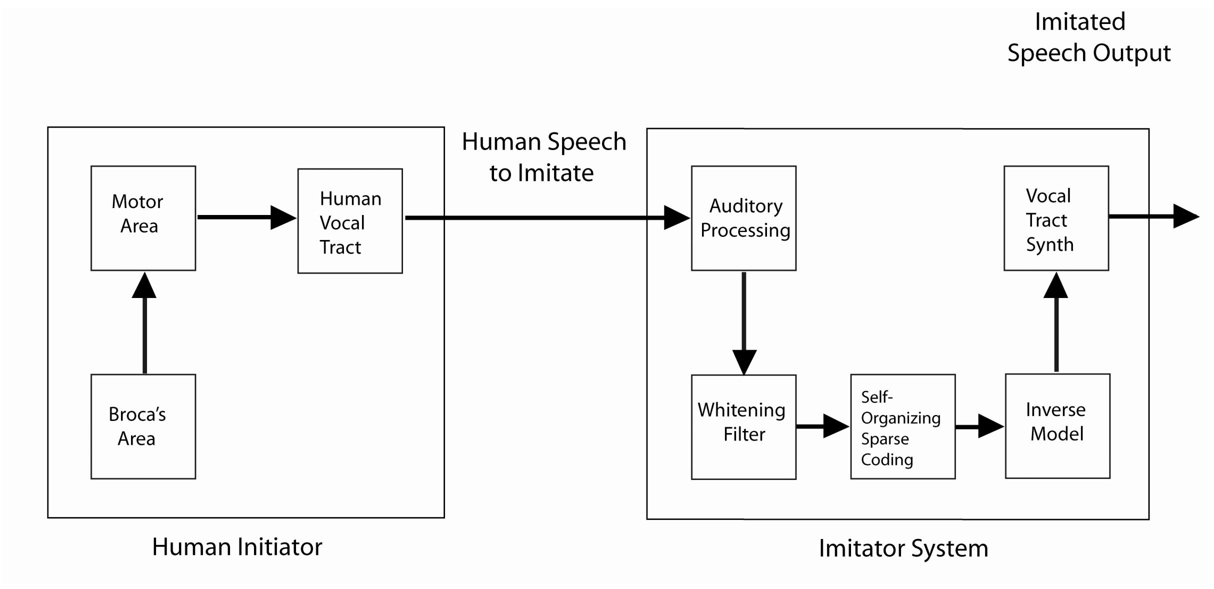

Figure 9: Imitating speech using sparse coding of the auditory representation.

9. Future work

Using our current system architecture, there are several technical improvements that could be made. The feed forward network used to implement the inverse model only makes use of a relatively short time window on the acoustic data. In addition, this network performed a memory-less mapping from input to output and did not take the continuity in the output trajectories into account. That is, it made an estimation of the articulator trajectories without using any prior knowledge regarding their dynamics. A consequence of this was that spikes were sometimes generated in its output signal, and these had to be smoothed out using post-processing. Such issues are elegantly addressed by Kalman filtering techniques, which is also worth of investigation for this kind of task and has been used for similar decoding tasks in many fields (e.g., Wu et al., 2004).

The simple inverse model used here to map between speech and the control of its articulator synthesiser obviously differs quite considerably on what is going on inside an infant's brain during speech acquisition. Firstly, in our scheme, the babble generation is totally separate from the imitation process. In a developing child, it is likely that the mechanisms that cause initial speech production (i.e. cooing and then babbling) are adapted over time to match its linguistic environment (i.e. to the speech to which the infant is exposed) and that these mechanisms develop into ones that are later used for the production of linguistically significant utterances. A more biologically relevant modification to our current scheme would thus be to include an abstract speech generator stage. Initially this would be configured to generate only the simplest of speech sounds, such as those involved in babbling. It should then develop, due to the system's exposure to external speech and feedback of its own speech, to produce more complex speech utterances. As mentioned previously, the best source of internally generated articulator movements to train the inverse model would be those used for the generation of real speech. It would therefore seem advantageous if the training of the inverse model should be an ongoing process (and actually never stop). One may expect its performance to improve as the speech generation process also improved. At the same time there also should be ongoing adaptation sparse coding stages. These issues are currently being investigated and are depicted graphically in figure 10.

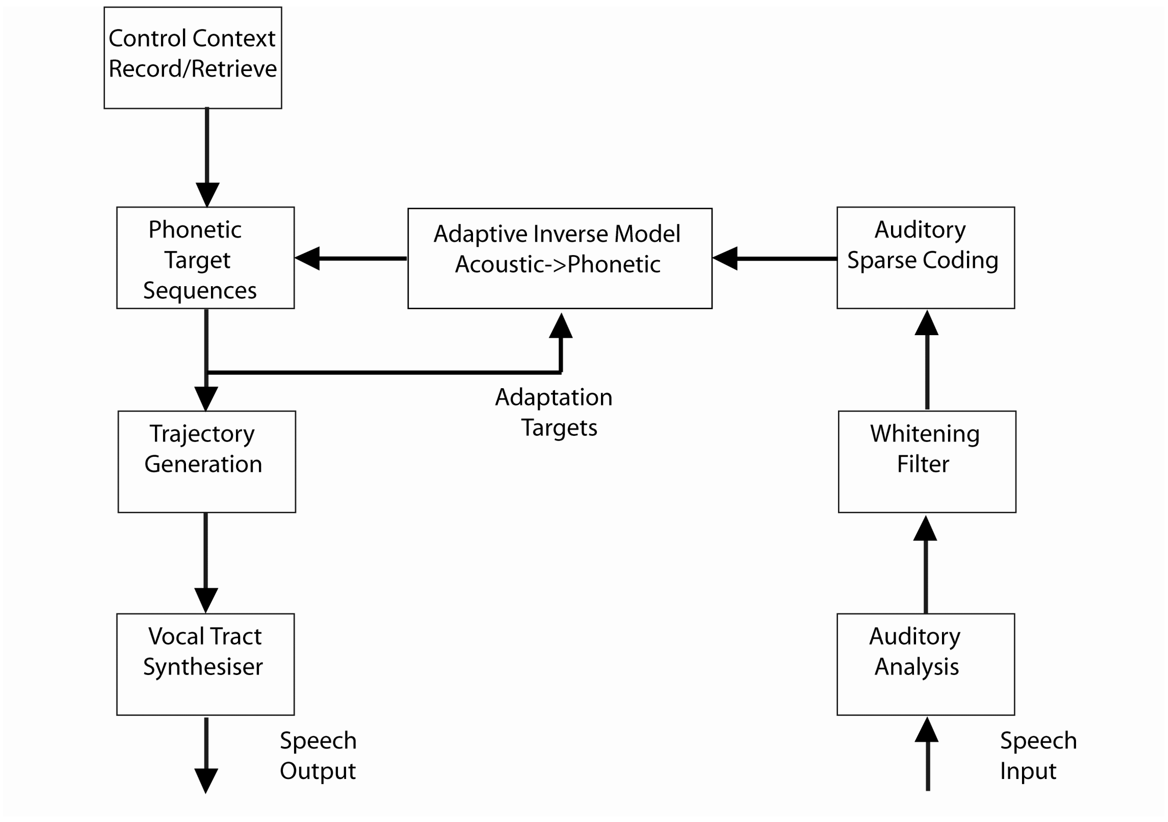

Figure 10: Improved system structure. The same generation process produces babble and then develops into a linguistic speech production mechanism. The inverse model and acoustic features detection are continually adapted and improved, not only during a separate babbling phase.

Acknowledgements

We wish to thank Daniel Wolpert for supporting this work. The implementation of the articulator synthesizer was based on an implementation by Shinji Maeda within the DOS program VTCALCS. We wish to thank Pascal Perrier and an unknown reviewer for commenting on the manuscript.

References

Bailly, G. (1997). Learning to speak. Sensori-motor control of speech movements,'' *Speech Commun.* 22: 251–267.

Barlow H.B. (1961). Possible principles underlying the transformation of sensory messages. In Rosenblith, W. (ed.) *Sensory Communication*. M.I.T. Press, Cambridge MA.

Guenther, F. H. (1994.) A neural-network model of speech acquisition and motor equivalent speech production. *Biol. Cybern.* 72: 43–53.

Guenther, F. H. (1995). 'Speech sound acquisition, coarticulation, and rate effects in a neural-network model of speech production. *Psychol. Rev.* 102: 594–621.

Holmes, J.N. (1980). The JSRU Channel Vocoder. *Proc. IEEE*, 127, Pt. F, 53-60.

Jordan, M.I., and Rumelhart, D.E. (1992). Forward models—Supervised learning with a distal teacher. *Cogn. Sci*. 16:307–354.

Lewicki, M.S. (2002). Efficient coding of natural sounds. *Nature Neurosci.* 5(4):356-363.

Maeda, S. (1990). Compensatory articulation during speech: evidence from the
analysis and synthesis of vocal tract shapes using an articulatory model. In Hardcastle W.J. and A. Marchal (eds.) *Speech production and speech modelling*. Kluwer Academic Publishers, Boston. p.131-149.

Nabney, I. and Bishop, C. (1995). Netlab: Netlab neural network software. http://www.ncrg.aston.ac.uk/netlab/.

Olshausen B.A. and Field D.J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*. 381(6583): 607-9.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning representations by back-propagating errors. *Nature* 323: 533-536.

Wolpert DM. (1997) Computational approaches to motor control. *Trends in Cognitive Sciences*. 1(6): 209-216.

Wu, W., Shaikhouni, A., Donoghue, J. P., and Black, M.J. (2004). Closed-loop neural control of cursor motion using a Kalman filter. *Proc. IEEE Engineering in Medicine and Biology Society*: 4126-4129.