

# THE IBM TRAINABLE SPEECH SYNTHESIS SYSTEM

*R.E. Donovan & E.M. Eide*

IBM T.J. Watson Research Center  
PO Box 218, Yorktown Heights, NY, 10598, USA  
red@watson.ibm.com eide@watson.ibm.com

## ABSTRACT

The speech synthesis system described in this paper uses a set of speaker-dependent decision-tree state-clustered hidden Markov models to automatically generate a leaf level segmentation of a large single-speaker continuous-read-speech database. During synthesis, the phone sequence to be synthesised is converted to an acoustic leaf sequence by descending the HMM decision trees. Duration, energy and pitch values are predicted using separate trainable models. To determine the segment sequence to concatenate, a dynamic programming (d.p.) search is performed over all the waveform segments aligned to each leaf in training. The d.p. attempts to ensure that the selected segments join each other spectrally, and have durations, energies and pitches such that the amount of degradation introduced by the subsequent use of TD-PSOLA is minimised. Algorithms embedded within the d.p. can alter the required acoustic leaf sequence, duration and energy values to ensure high quality synthetic speech. The selected segments are concatenated and modified to have the required prosodic values using the TD-PSOLA algorithm. The d.p. results in the system effectively selecting variable length units, based upon its leaf level framework.

## 1. INTRODUCTION

Through the late 1980s and early 1990s the speech segments most frequently used in concatenative speech synthesis systems (in European languages) were diphones, and subsequently diphones augmented with longer polyphone units. In the last few years research has increasingly focused on the automatic selection and segmentation of speech units for use in concatenative speech synthesis systems using trainable statistical models, [2], [5], [6], [7]. The attraction of this approach over the more traditional manual selection and segmentation approaches are that trainable approaches offer the possibility of selecting a set of units which is optimal in some useful sense through training on much larger amounts of speech data than can sensibly be analysed by hand, which should lead to superior synthetic speech. In addition, the full automation of the selection and segmentation process, and the use of simple continuous speech databases rather than isolated nonsense words, makes the process of generating new voices, and potentially new languages, much easier and much faster.

The system described in this paper is an extension of the work described in [3], [4]. The principal new features are the use of a dynamic programming search over all available segments during synthesis, the use of additional sets of

decision trees to predict duration and energy parameters, and the inclusion of a pitch prediction algorithm. The rest of this paper is structured as follows. Section 2 describes the methods used to construct a synthesis system in a given voice, Section 3 describes the runtime synthesis process, and Section 4 describes the dynamic programming search in detail. Results are presented in Section 5, and likely future work briefly discussed in Section 6.

## 2. SYSTEM CONSTRUCTION

The system construction details are essentially the same as those in [4], and therefore only important differences will be mentioned here. The current system is trained on 45 minutes of speech and clustered to give approximately 2000 acoustic leaves. The variable rate Mel frequency cepstral coding used in [4] is replaced with a pitch synchronous coding using 25ms frames through regions of voiced speech, with 6ms frames at a uniform 3ms or 6ms frame rate through regions of unvoiced speech. Plosives are represented by 2-state models, but the burst is not optional as it was in [4]. Lexical stress clustering is not currently used, and certain segmentation cleanups described in [4] are not implemented. The tree building process uses the algorithms described in [1], which are essentially the same as those used in [4], but which will be described here to aid understanding of Section 4.3.

A binary decision tree is constructed for each feneme<sup>1</sup> as follows. All the data aligned to a feneme is used to construct a single Gaussian in the root node of the tree. A list of questions about the phonetic context of the data is used to suggest splits of the data into two child nodes. The question which results in the maximum gain in the log-likelihood of the data fitting Gaussians constructed in the child nodes compared to the Gaussian in the parent node is selected to split the parent node. This process continues at each node of the tree until one of two stopping criteria is met. These are when a minimum gain in log-likelihood cannot be obtained or when a minimum number of segments in both child nodes cannot be obtained, where a segment is all contiguous frames in the training database with the same feneme label. The second stopping criteria differs from that used in [1], in which a minimum number of frames is enforced, because a minimum number of segments is required for subsequent segment selection algorithms. Also, node merging is not permitted in order to maintain the one parent structure necessary for the Backing Off algorithm described in Section 4.3.

<sup>1</sup>A feneme is a term used to describe an individual HMM model position. eg. the model for /AA/ comprises three fenemes AA\_1, AA\_2, and AA\_3.

The acoustic (HMM) decision trees are built asking questions about only immediate phonetic context. While asking questions about more distant contexts may give slightly more accurate acoustic models it can result in being in a leaf in synthesis from which no segments are available which concatenate smoothly with neighbouring segments, for reasons similar to those described in Section 4.3. Separate sets of decision trees are built to cluster duration and energy data. Since the above concern does not apply to these trees they are currently built using 5 phones of phonetic context information in each direction, though to date the effectiveness of this increased context, or indeed the precise values of the stopping criteria have not been investigated.

### 3. RUNTIME SYNTHESIS

**Parameter Prediction.** During synthesis the words to be synthesised are converted to a phone sequence by dictionary lookup, with the selection between alternatives for words with multiple pronunciations being performed manually. The decision trees are used to convert the phone sequence into an acoustic, duration, and energy leaf for each feneme in the sequence. The median training values in the duration and energy leaves are used as the predicted duration and energy values for each feneme. The acoustic leaf sequence, duration and energy values just described are termed the *requested* parameters from hereon. Pitch tracks are also predicted using a separate trainable model not described in this paper.

**Dynamic Programming.** The next stage of synthesis is to perform a dynamic programming (d.p.) search over all the waveform segments aligned to each acoustic leaf in training, to determine the segment sequence to use in synthesis. The d.p. algorithm, and related algorithms which can modify the requested acoustic leaf identities, energies and durations, are described in detail in Section 4.

**Energy Discontinuity Smoothing.** Once the segment sequence has been determined, energy discontinuity smoothing is applied. This is necessary because the decision tree energy prediction method predicts each feneme's energy independently, and does not ensure any degree of energy continuity between successive fenemes. Note that it is energy *discontinuity* smoothing (the discontinuity between two segments is defined as the difference between the energy (per sample) of the second segment minus the energy (per sample) of the segment in the training data following the first segment), not energy smoothing; changes in energy of several orders of magnitude do occur between successive fenemes in real human speech, and these changes must not be smoothed away.

**TD-PSOLA.** Finally, the selected segment sequence is concatenated and modified to match the required duration, energy and pitch values using an implementation of the TD-PSOLA algorithm, [8]. The implementation is essentially the same as that described in [4], except that the Hanning windows used are set to the smaller of twice the synthesis pitch period or twice the original pitch period.

### 4. DYNAMIC PROGRAMMING

The dynamic programming (d.p.) search attempts to select the optimal set of segments from those available in the acoustic decision tree leaves to synthesise the requested

acoustic leaf sequence with the requested duration, energy and pitch values. The optimal set of segments is that which most accurately produces the required sentence after TD-PSOLA has been applied to modify the segments to have the requested characteristics. The cost function used in the d.p. algorithm, described in Section 4.1, therefore reflects the ability of TD-PSOLA to perform modifications without introducing perceptual degradation. Two additional algorithms, described in Sections 4.2 and 4.3, enable the d.p. to modify the requested parameters where necessary to ensure high quality synthetic speech.

#### 4.1. The Cost Function

**Continuity Cost.** The strongest cost in the d.p. cost function is the spectral continuity cost applied between successive segments. This cost is calculated for the boundary between two segments A and B by comparing a spectral vector calculated from the start of segment B to a spectral vector calculated from the start of the segment following segment A in the training database. The continuity cost between two segments which were adjacent in the training data is therefore zero. The vectors used are 24 dimensional Mel binned log FFT vectors. The cost is computed by comparing the loudest regions of the two vectors after scaling them to have the same energy; energy continuity is costed separately. This method has been found to work better than using a simple Euclidean distance between cepstral vectors.

The effect of the strong spectral continuity cost together with the feature that segments which were adjacent in the training database have a continuity cost of zero is to encourage the d.p. algorithm to select sequences of segments which were originally adjacent wherever possible. The result is that the system ends up effectively selecting and concatenating variable length units, based upon its leaf level framework.

**Duration Cost.** The TD-PSOLA algorithm introduces essentially no artifacts when reducing durations, and therefore duration reduction is not costed. Duration increases using the TD-PSOLA algorithm however can cause serious artifacts in the synthetic speech due to the over repetition of voiced pitch pulses, or the introduction of artificial periodicity into regions of unvoiced speech. The duration stretching costs are therefore based on the expected number of repetitions of the Hanning windows used in the TD-PSOLA algorithm.

**Pitch Cost.** There are two aspects to pitch modification degradation using TD-PSOLA. The first is related to the number of times individual pitch pulses are repeated in the synthetic speech, and this is costed by the duration costs just described. The other cost is due to the fact that pitch periods cannot really be considered as isolated events, as assumed by the TD-PSOLA algorithm; each pulse inevitably carries information about the pitch environment in which it was produced, which may be inappropriate for the synthesis environment. The degradation introduced into the synthetic speech is more severe the larger the attempted pitch modification factor, and so this aspect is costed using curves which apply increasing costs to larger modifications.

**Energy Cost.** Energy modification using TD-PSOLA involves simply scaling the waveform. Scaling down is free

under the cost function since it does not introduce serious artifacts. Scaling up, particularly scaling quiet sounds to have high energies, can introduce artifacts however, and it is therefore costed accordingly.

## 4.2. Cost Capping/Post Selection Modification

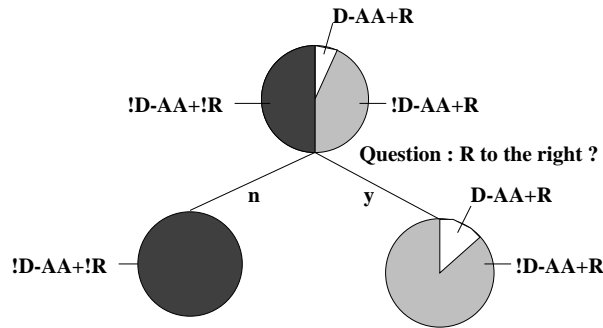
During synthesis, simply using the costs described above results in the selection of good segment sequences most of the time. However, for some segments in which one or more costs becomes very large the procedure breaks down. To illustrate the problem, imagine a feneme for which the predicted duration was 12 Hanning windows long, and yet every segment available was only 1-3 Hanning windows long. This would result in poor synthetic speech for two reasons. Firstly, whichever segment is chosen the synthetic speech will contain a duration artifact. Secondly, given the cost curves being used, the duration costs will be so much cheaper for the 3-Hanning-window segment(s) than the 1 or 2 Hanning-window segment(s), that a 3-Hanning-window segment will probably be chosen almost irrespective of how well it scores on every other cost measure. It was to cope with cases like this that the cost capping/post selection modification scheme was introduced.

Under the cost capping scheme, every cost except continuity is capped during the d.p. at the value which corresponds to the approximate limit of acceptable signal processing modification. After the segments have been selected, the post-selection modification stage involves changing (generally reducing) the requested characteristics to the values corresponding to the capping cost. In the above example, if the limit of acceptable duration modification was to repeat every Hanning window twice, then if a 2-Hanning-window segment were selected it would be costed for duration doubling, and ultimately produced for 4 Hanning windows in the synthetic speech. Thus the requested characteristics can be modified in the light of the segments available to ensure good quality synthetic speech. The mechanism is typically invoked only a few times per sentence.

## 4.3. Backing Off

The decision trees used in the system enable the rapid identification of a sub-set of the segments available for synthesis with hopefully the most appropriate phonetic contexts. However, in practice the decision trees do occasionally make mistakes, leading to the identification of inappropriate segments in some contexts. To understand why, consider the following example.

Imagine that the tree fragment shown in Figure 1 exists, in which the question "R to the right?" was determined to give the biggest gain in log-likelihood. Now imagine that in synthesis the context /D-AA+!R/ is to be synthesised. The tree fragment in Figure 1 will place this context in the /!D-AA+!R/ node, in which there is unfortunately no /D-AA/ speech available. Now, if the /D/ has a much bigger influence on the /AA/ speech than the presence or absence of the following /R/ then this is a problem. It would be preferable to descend to the other node where /D-AA/ speech is available, which would be more appropriate despite it's /+R/ context. In short, it is possible to descend to leaves which do not contain the most appropriate speech for the context specified. The most audible



**Figure 1:** A parent node and two leaves from a hypothetical /AA/ tree. Shading indicates the fraction of the data at each node with the context shown, in which ! indicates logical NOT, and - and + are used to separate the preceding and following contexts respectively from the central phone.

result of this type of problem is formant discontinuities in the synthetic speech, since the speech available from the inappropriate leaf is unlikely to concatenate smoothly with its neighbours.

The solution to this problem adopted in the current system has been termed *Backing Off*. When backing off is enabled the continuity costs computed between all the segments in the current leaf and all the segments in the next leaf during the d.p. forward pass are compared to some threshold. If it is determined that there are *no* segments in the current leaf which concatenate smoothly (i.e. cost below the threshold) with any segments in the next leaf, then both leaves are backed off up their respective decision trees to their parent nodes. The continuity computations are then repeated using the set of segments at each parent node formed by pooling all the segments in all the leaves descended from that parent. This process is repeated until either some segment pair costs less than the threshold, or the root node in both trees is reached. By determining the leaf sequence implied by the selected segment sequence, and comparing this to the original leaf sequence, it has been determined that in most cases backing off does change the leaf sequence (it is possible that after the backing off process the selected segments still come from the original leaves). The process has been seen (in spectrograms), and heard, to remove formant discontinuities from the synthetic speech, and is typically invoked only a few times per sentence.

If there are *no* segments with a concatenation cost lower than the threshold then there *will* be a continuity problem, which hopefully backing off will solve. However, it may be the case that even when there are one or more pairs of concatenable segments available these cannot be used because they do not join to the rest of the sequence. Ideally then, the system would operate with multiple passes of the entire dynamic programming process, backing off to optimise sequence continuity rather than pair continuity. However, this approach is probably too computationally intensive for a practical system.

Finally, note that the backing off mechanism could also be used to correct the leaf sequences used in decision tree based speech recognition systems.

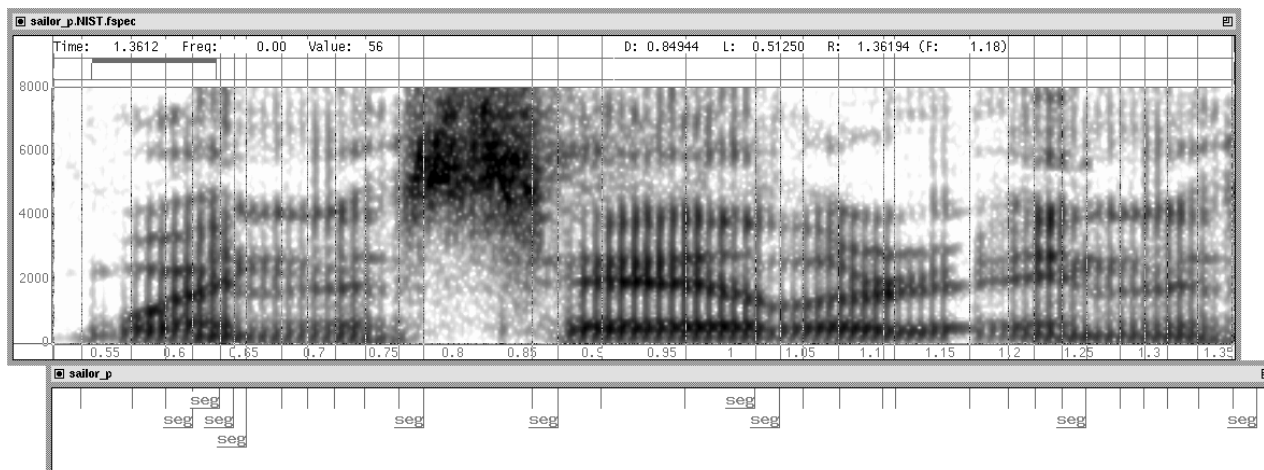


Figure 2: A wideband spectrogram of the synthetic sentence fragment “When a sailor in a...”. Leaf boundaries (vertical lines), and segment boundaries (seg labels) are shown.

## 5. RESULTS

Figure 2 shows a wideband spectrogram of the synthetic sentence fragment “When a sailor in a...”, taken from the sentence “When a sailor in a small craft faces the might of the vast Atlantic Ocean today, he takes the same risks as generations took before him.”. The underlying leaf boundary locations are shown with vertical lines, and the boundaries of the segments actually concatenated to construct the speech indicated by the “seg” labels. This sentence should be available on the conference CD-ROM.

The sentence in Figure 2 has an average segment length of approximately 2.4 leaves. This is a typical figure for a sentence from a different text to the training data containing many unseen words. Of the words in this sentence “sailor”, “faces”, “might”, “vast”, “same”, “risks”, “generations” and “took” are not present in the training data, with “craft” and “him” only being present as parts of larger words. For sentences with vocabularies more similar to the training vocabulary the average segment length can rise to 3.0 leaves or much higher.

The synthetic speech produced by the current system has considerably better formant continuity than that produced by the system described in [4]. In other respects, such as segmental intelligibility and duration and energy prediction, the current system is perhaps inferior, though this is thought to be principally due to algorithms implemented in [4] not yet implemented in the current system. The current system’s runtime image is approximately 150MB, and it runs about an order of magnitude slower than real time on a 133MHz Power PC.

## 6. FUTURE WORK

In addition to quality improvements in all areas, future work will be concerned with the reduction of the system’s runtime image size, both through *pre-selecting* some optimal sub-set of the segments in each leaf to ship with the synthesiser, and through compressing this sub-set. This pre-selection will also greatly increase the runtime speed of the system.

## 7. ACKNOWLEDGMENTS

Thanks to Lalit Bahl and Jeff Kunitz for recording speech databases.

## 8. REFERENCES

1. Bahl, L.R., deSouza, P.V., Gopalakrishnan, P.S., and Picheny, M.A. (1993) Context Dependent Vector Quantization for Continuous Speech Recognition, *Proc. ICASSP'93, Minneapolis*, Vol. 2. pp. 632–635.
2. Black, A.W., and Campbell, N. (1995) Optimising Selection of Units from Speech Databases for Concatenative Synthesis, *Proc. Eurospeech'95, Madrid*, pp. 581–584.
3. Donovan, R.E. and Woodland, P.C. (1995) Improvements In An HMM-Based Speech Synthesiser, *Proc. Eurospeech'95, Madrid*, pp. 573–576.
4. Donovan, R.E. (1996) *Trainable Speech Synthesis*, PhD. Thesis, Cambridge University Engineering Department.<sup>2</sup>
5. Hauptmann A.G. (1993) SpeakEZ: A First Experiment In Concatenation Synthesis From A Large Corpus, *Proc. Eurospeech'93, Berlin*, pp. 1701–1704.
6. Huang, X., Acero, A., Adcock, J., Hon, H-W., Goldsmith, J., Liu, J., and Plumpe, M. (1996) Whistler: A Trainable Text-to-Speech System, *Proc. ICSLP'96, Philadelphia*, pp. 2387–2390.
7. Itoh, K., Nakajima, S., and Hirokawa, T. (1994) A New Waveform Speech Synthesis Approach Based on the COC Speech Spectrum, *Proc. ICASSP'94, Adelaide*, Vol. 1, pp. 577–580.
8. Moulines, E., and Charpentier, F. (1990) Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones, *Speech Communication*, 9, pp. 453–467.

<sup>2</sup>Available by anonymous ftp to [svr-ftp.eng.cam.ac.uk](http://svr-ftp.eng.cam.ac.uk), or via the World Wide Web at [http://svr-www.eng.cam.ac.uk/People/Ex\\_Students/red/Personal.html](http://svr-www.eng.cam.ac.uk/People/Ex_Students/red/Personal.html)