

# Current Status of the IBM Trainable Speech Synthesis System

R. E. Donovan<sup>1</sup>, A. Ittycheriah<sup>1</sup>, M. Franz<sup>1</sup>, B. Ramabhadran<sup>1</sup>, E. Eide<sup>1</sup>,  
M. Viswanathan<sup>1</sup>, R. Bakis<sup>1</sup>, W. Hamza<sup>1</sup>, M. Picheny<sup>1</sup>, P. Gleason<sup>2</sup>, T. Rutherford<sup>2</sup>,  
P. Cox<sup>3</sup>, D. Green<sup>3</sup>, E. Janke<sup>3</sup>, S. Revelin<sup>4</sup>, C. Waast<sup>4</sup>, B. Zeller<sup>5</sup>, C. Guenther<sup>5</sup>, J. Kunzmann<sup>5</sup>

<sup>1</sup> IBM T.J.Watson Research Center, PO Box 218, Yorktown Heights, New York, 10598, USA

<sup>2</sup> IBM, 8051 Congress Ave., Suite 2091, Boca Raton, Florida, 33487, USA

<sup>3</sup> Mail Point 094, IBM UK Ltd, Hursley Park, Winchester, Hampshire, SO212JN, UK

<sup>4</sup> IBM France, Tour Descartes - 2 av Gambetta, La Defense 5-92400 Courbevoie, France

<sup>5</sup> IBM Heidelberg, Vangerowstrasse 18, D-69115 Heidelberg, Germany

## Abstract

This paper describes the current status of the IBM Trainable Speech Synthesis System. The system is a state-of-the-art, trainable, unit-selection based concatenative speech synthesiser. The system uses hidden Markov models (HMMs) to provide a phonetic transcription and HMM state alignment of a database of single-speaker continuous-speech training data. The runtime synthesiser uses the HMM state sized segments that result as its basic synthesis units. It determines which segments to concatenate to produce a target sentence using decision trees built from the training data and a dynamic programming search to optimise a perceptually motivated cost function. The synthesiser can operate both in general domain Text-to-Speech mode, and in *Phrase Splicing* mode to provide higher quality synthesis in limited domains. Systems have been built in at least 10 different languages and over 70 voices.

## 1. Introduction

The IBM Trainable Speech Synthesis System is a state-of-the-art, trainable, unit-selection based concatenative speech synthesis system. This paper describes the current status of the system, which was previously introduced in [1]. The system uses hidden Markov model (HMM) state-sized segments as its basic synthesis units and decision trees in its segment search. The system is based on the work described in [2], [3], and also has similarities with that described in [4] and [5]. Unlike [2], [3] the current system combines the decision tree approach with a dynamic programming search similar to that used in [6], [7].

This paper is structured as follows. Section 2 describes the methods used to prepare a dataset for use with the synthesiser. Section 3 describes the runtime operation of the synthesiser in general domain Text-to-Speech (TTS) mode. Section 4 describes the runtime operation of the system in limited domain *Phrase Splicing* mode. Finally, Section 5 describes the results to date with the synthesiser.

## 2. Dataset Preparation

For each new language the first step in dataset preparation is to prepare the script to be read during recording. The script is currently prepared by running a greedy algorithm over about 100,000 newswire sentences. On each pass the greedy algo-

rithm computes a score for every sentence not yet selected based on the diphones within it and the wider contexts in which they occur. The wider context comprises the diphone, the preceding phone and any word boundaries between the three phones. The score is heavily weighted in favour of diphones, in previously unseen wider contexts, which are under-represented in the script selected to date. The best scoring sentence is added to the script and the process repeated. The result is a training script in which the most diverse sentences are at the top, and in which every additional sentence brings as much new variety in phonetic context as possible.

The top 1400-2000 sentences of the training script are recorded, providing about 2.5 to 3.5 hours of training speech. The recordings are made using a high quality flat frequency response cardioid microphone in a sound-proofed room. Recordings are usually made at a 22kHz or 44kHz sampling rate; the speech is down-sampled during system building to the rate required in synthesis. The speech recording is made in stereo with a laryngograph signal recording, [8], which is processed to determine the moments of glottal closure in the voiced sections of the speech.

The speech data is initially coded into 12 dimensional mel frequency cepstral coefficients (MFCCs) plus log energy and the first and second time differentials of these parameters using 25ms frames at a uniform 10ms frame rate. The TTS system front-end is used to prepare a pronunciation dictionary for the words in the script. A set of speaker independent HMMs is used to obtain an initial phonetic transcription of the speech, with the HMMs inserting silences between words where appropriate and deciding between alternative pronunciations where they exist in the dictionary. A post processing heuristic then removes silences which have been mistakenly aligned through the closures of words beginning or ending with a plosive, [2]. This initial alignment is used to train a set of speaker-dependent cross-word decision-tree state-clustered HMMs, [9]. These HMMs are mostly 3 state left-to-right models, except for the voiced plosive models which align better with 2 states. The data is recoded using 25ms pitch synchronous frames through regions of voiced speech, and 6ms frames at a 3ms frame rate through unvoiced speech, using the scaling method described in [2] to ensure that cepstra from the different sized frames are comparable. The recoding gives better resolution through regions of unvoiced speech which is especially helpful in segmenting plo-

sives due to the short timescales and rapid transitions involved. After further training the HMMs are used to provide another alignment, and the model building process repeated using the recorded data. After more training the final models are used to provide an HMM state-level alignment of the training data.

The acoustic decision trees used in synthesis are built from the final HMM alignment. A separate decision tree is built for each unclustered HMM state using the standard maximum likelihood tree growing procedure used in most modern speech recognition systems, [9]. The splits are made using broad class context questions applied to the immediate phonetic and word boundary context only, with the likelihood computations based on the MFCC, energy and time differential parameters described above. The criteria used to stop tree growth are to insist on a minimum of 20 speech segments per leaf and a minimum increase in log-likelihood per split, with the latter being set so as to result in approximately 5000 leaves.

Energy prediction decision trees are also built from the final HMM alignment. Again, a separate tree is built for each unclustered HMM state using the same tree growing procedure. In this case the context considered includes the two preceding phones, the two following phones, and the word boundaries between them. The likelihood computations are based on the log peak energy value in each segment, defined as the maximum energy found in any 2ms window within the segment. The trees are built to have approximately 3000 leaves. In prediction the median energy value in each leaf is used as the predicted value.

Perceptually modified cepstral vectors and spectral continuity cost decision trees are also computed from the data and the final HMM alignment, as described in [10]. These components are used to determine spectral continuity costs between segment endpoints during the runtime search.

Different phones have consistent effects on natural  $F_0$  contours, resulting in what is known as microprosody. By modeling these effects the system can produce more natural  $F_0$  contours in synthesis. Segmental pitch deltas are therefore computed from the final HMM alignment and the processed laryngograph signal. For each segment in the database a pitch delta is computed as the  $F_0$  at the end of the segment divided by the  $F_0$  at the beginning of the segment. The geometric mean of these deltas is computed for each unclustered HMM state and recorded for later use in pitch contour generation.

In order to reduce the runtime system size and improve runtime speed some of the training data is discarded from the runtime dataset. This *pre-selection* is performed by keeping only the first 25 occurrences (or less where necessary) of each leaf in the training data. This method tends to provide a variety of prosodic and acoustic realisations of each leaf while also enabling the runtime selection of longer units due to the large number of contiguous segments which are typically retained. More complex pre-selection algorithms have been investigated, [11], but have not yet been shown to outperform the simple method just described and so are not currently used.

Finally, a proprietary speech compression algorithm can be used to compress the waveforms in the pre-selected dataset. The compression algorithm achieves factor 7 compression at any sample rate for almost no noticeable degradation in speech quality. The algorithm was designed to be as fast as possible during decompression and to compress each frame independently to allow random access to the speech waveforms during synthesis.

Use of the compression algorithm is optional. Its use reduces dataset sizes by factors of 2.3 for 8kHz systems, 2.5 for 11kHz systems, and larger factors for higher sampling rate systems.

### 3. Runtime Synthesis

During synthesis text processing, text to phone conversion, phrase boundary placement, duration prediction and  $F_0$  prediction are performed by an independent rule-based front-end. The result of this processing is passed, one phrase at a time, to the back-end which generates the synthetic speech.

The first stage in back-end processing is the conversion of the specified phone sequence into a target acoustic leaf sequence, by dropping the sequence of contexts implied by the phones down the acoustic trees. Next, target energies are determined for each state using the energy decision trees. Target durations for each state are obtained as the median duration of the corresponding acoustic leaf scaled such that the sum of the state durations in each phone is equal to the phone duration specified by the front-end. Target  $F_0$  values for the end of each state are obtained by linearly interpolating between the points in the  $F_0$  contour specified by the front-end given the state target durations, and adding the segmental pitch deltas described in Section 2 in a perturbation-relaxation fashion to the contour.

With the target acoustic leaf sequence and target state prosody values established the synthesiser proceeds to the segment search. The aim of the search is to select the sequence of segments which best produces the target sentence. The search is based on dynamic programming (d.p.), with pruning applied in the forward pass to limit the number of parallel segment paths under consideration. Each step of the forward pass begins by determining which acoustic tree leaves can supply segments for consideration for use in the current state. The target leaf supplies its segments with zero cost. It has been found helpful to allow segments from other leaves to be used when there is a poor match between the target prosody and the inherent prosody of the segments in the target leaf. Therefore, the target leaf's sibling supplies its segments with cost  $T$  and other leaves descended from the target leaf's grandparent node supply their segments with cost  $2T$ . Segments descended from more distant relatives are not available due to a  $3.3T$  pruning cost applied throughout the forward pass. In practice segments are used from leaves other than the target leaf in only about 1.5% of synthesis states.

All the segments available are then costed for the difference between their inherent prosody and the target prosody. Cost curves are used which are designed through trial-and-error to reflect the amount of audible degradation introduced by modifying the segment's inherent prosody by different amounts. Thus, reducing duration is free, while increasing duration is not, and is more expensive for unvoiced speech than voiced speech. The cheapest cost to evaluate (fundamental frequency) is applied first. In order to reduce the computational burden, any segment whose total cost (leaf + fundamental frequency) exceeds the pruning cost is skipped when evaluating the energy modification costs, and so on. Duration and energy costs are capped at  $T$ , which is the cost corresponding to the approximate limit of acceptable signal processing modification. Since modifying these segments further would introduce a signal processing artifact into the synthetic speech the system chooses to produce the

segments at this limit and not produce the target prosody. When this occurs additional costs are invoked to cost the extent of this shortfall.

A small cost reward is given to any segment which was contiguous in the original recordings with any of the segments present in the d.p. array of the previous state. Continuity is costed properly in the d.p. below; this reward is included at this stage to encourage the selection of contiguous segments for the d.p. stage.

The five lowest cost segments under consideration are then entered into the d.p. array of the current state. The continuity reward added above is subtracted again; just because a segment was contiguous with a segment in the previous state's d.p. array doesn't mean the d.p. path will link the two segments. The standard d.p. maximisation computation is then performed to compute the best predecessor for each segment in the current state's d.p. array, using the cost to date of the segments in the previous state's d.p. array and the spectral continuity cost. The continuity cost is computed as in [10] with a small extra reward given to contiguous segments. Finally the cost to date of each segment in the current state's d.p. array is computed and stored.

After each stage in the d.p. forward pass the synthesiser looks back down the partial paths to determine if they converge at any point in the past. If it is found that the paths do converge then a traceback is computed from the point of convergence back in time to any previous point of convergence or the start of the phrase if this is the first convergence in the phrase. The traceback determines which segment will be used to produce each state in synthesis. The new section for which segments have been determined is then sent immediately to the signal processing routines to be turned into speech. This streaming, and the subsequent streaming of the speech waveform as it is generated, minimises the delay in speech being heard following a synthesis request.

The signal processing routines concatenate the selected segments and modify them to have the target prosody, or the capped prosodic values if applicable. The modification algorithm used is currently unpublished, but is similar in concept to the frequency domain (FD) PSOLA algorithm described in [12].

## 4. Phrase Splicing

In addition to the general domain TTS mode of operation described above, the IBM synthesiser can also be operated in *Phrase Splicing* mode to provide extremely high quality synthesis in limited domains. In phrase splicing, a core synthesis system is built as described above, and a set of application specific *splice files* are recorded in the same voice. The splice files cover all the key phrases appearing in the synthesis domain, with the phrases recorded in appropriate prosodic contexts. The phrase splicing system enables phrases from the splice files to be seamlessly spliced together with phrases from other splice files and with unseen words (*variables*) synthesised by the core system.

Introduced in [13] the phrase splicing algorithms have been modified slightly to be compatible with the search streaming described in Section 3. In the current implementation phrase splicing exists essentially as an alternative front-end. Input text is fed to the rule-based text normalisation routines used in TTS and the normalised text then used to search a splice file dictionary prepared as part of the build. The splice file dictionary is a

hash table built from the splice files which relates all normalised text substrings occurring in the splice files to the state alignment data (including phone identity and prosody) for the corresponding waveform segments. A phone sequence and target prosody are assembled from the data associated with the longest substrings of the synthesis normalised text that can be found in the splice file dictionary. When words are in the synthesis text which are not present in the splice files a phone sequence is obtained for the word from the TTS rule-based front-end. In this case target prosody is obtained from the energy prediction trees described in Section 2, from similar duration prediction trees, and by linearly interpolating the pitch contour between the ends of the phrases on either side (or a default value at sentence ends). The pitch deltas described in Section 2 are added to the interpolated region. Finally, the target pitch contour undergoes discontinuity smoothing, in which the natural variation of pitch over contiguous segments is not smoothed, to remove any sudden jumps at splice points.

With the phone sequence and target prosody determined the synthesiser operates exactly as described in Section 3, except that the synthesis inventory is augmented with the splice file segments from the splice files used to construct the synthesis phone sequence, and that these segments always make it into the d.p. array. The result is that, in regions far from boundaries, the synthetic speech is produced almost exactly as it was in the splice file recordings. At boundaries between splice files however, the system automatically splices in segments from appropriate contexts from the core synthesiser ensuring spectral and prosodic continuity at the join. At variables the system switches seamlessly from splice file speech to core system speech and back again, again ensuring spectral and prosodic continuity.

## 5. Results

Although synthesis datasets for a completely new language have been prepared in as little as one week, it typically takes a few months of work to get good performance; recording environments and equipment need to be obtained, pronunciations used during training often need to be corrected by hand, and invariably other minor problems arise or mistakes are made. Subsequent voices in the same language however can typically be built in 3 or 4 days from the commencement of recording. To date, a total of over 70 voices have been built in at least 10 different languages. US English, UK English, French and German systems will be demonstrated at this workshop.

The runtime engine is multi-threaded and shares dataset information between threads synthesising the same voice. Memory usage is approximately 40MB per voice at an 8kHz sampling rate for compressed systems. Uncompressed systems have an image size of approximately 90MB, but run 34% faster than compressed systems. At the time of writing absolute system speed is still under review.

Figure 1 shows a wideband spectrogram of the sentence fragment "...the IBM..." taken from the sentence "Welcome to the IBM Research Text-to-Speech demonstration page." synthesised in a female voice at an 11kHz sampling rate. The vertical lines show the state boundaries and the "chk" labels the boundaries of the chunks of speech concatenated to construct the speech. In typical synthesis concatenation occurs on approximately 70% of state boundaries.

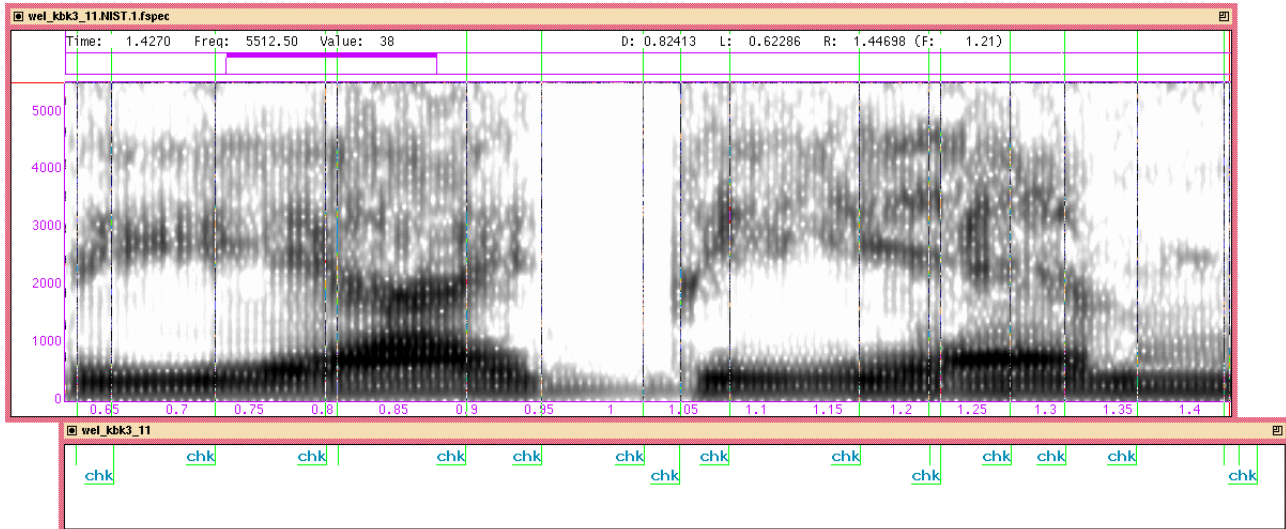


Figure 1: Wideband spectrogram of the sentence fragment “...the IBM...” synthesised in a female voice at an 11kHz sampling rate. State boundaries (vertical lines) and concatenation boundaries (chk labels) are shown.

Phrase splicing systems have been built in a number of languages for numerous voices in several different domains. In US English domains have included mutual fund trading, financial news, and airline reservations. A demonstration of part of the mutual fund trading system is available by calling +1 972 402 5963 and saying “mutual fund”.

## 6. Conclusion

The IBM Trainable Speech Synthesiser is a state-of-the-art unit-selection based concatenative speech synthesiser. It enables both high quality general domain Text-to-Speech synthesis and very high quality limited domain synthesis to be performed in multiple languages. New voices can be built in existing languages in 3 or 4 days.

## 7. Acknowledgments

Thanks to the large number of people who have recorded training databases, and to all the IBMers who have performed listening tests over the last five years.

## 8. References

- [1] Donovan, R.E., and Eide, E.M. (1998) The IBM Trainable Speech Synthesis System, *Proc. ICSLP'98, Sydney*.
- [2] Donovan, R.E. (1996) *Trainable Speech Synthesis*, PhD. Thesis, Cambridge University Engineering Department.<sup>1</sup>
- [3] Donovan, R.E. & Woodland, P.C. (1999) A Hidden Markov Model Based Trainable Speech Synthesiser, *Computer Speech and Language*, Vol. 13, No. 3, pp. 223–242.
- [4] 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*.
- [5] Black, A., and Taylor, P. (1997) Automatically Clustering Similar Units for Unit Selection in Speech Synthesis, *Proc. Eurospeech'97, Rhodes*.
- [6] Black, A.W., and Campbell, N. (1995) Optimising Selection of Units from Speech Databases for Concatenative Synthesis, *Proc. Eurospeech'95, Madrid*.
- [7] Hunt, A., and Black, A. (1996) Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database, *Proc. ICASSP'96, Atlanta*.
- [8] Fourcin, A.J., and Abberton, E. (1971) First Applications of a New Laryngograph, *Medical & Biological Illustration*, Vol. 21, No. 3, pp. 172–182.
- [9] 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*.
- [10] Donovan, R.E. (2001) A New Distance Measure for Costing Spectral Discontinuities in Concatenative Speech Synthesisers, *Proc. 4th ISCA Tutorial and Research Workshop on Speech Synthesis, Atholl Palace Hotel, Scotland*.
- [11] Donovan, R.E. (2000) Segment Pre-selection in Decision-Tree Based Speech Synthesis Systems, *Proc. ICASSP 2000, Istanbul*.
- [12] Moulines, E., and Charpentier, F. (1990) Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones, *Speech Communication*, 9, pp. 453–467.
- [13] Donovan, R.E., Franz, M., Sorensen, J.S. & Roukos, S. (1999) Phrase Splicing and Variable Substitution using the IBM Trainable Speech Synthesis System, *Proc. ICASSP '99, Phoenix*.

<sup>1</sup>Available by anonymous ftp to svr-ftp.eng.cam.ac.uk