PHRASE SPLICING AND VARIABLE SUBSTITUTION USING THE IBM TRAINABLE SPEECH SYNTHESIS SYSTEM

R.E. Donovan, M. Franz, J.S. Sorensen, S. Roukos

IBM T.J. Watson Research Center
PO Box 218, Yorktown Heights, NY, 10598, USA
red@watson.ibm.com {franzm, sorenj, roukos}@us.ibm.com

ABSTRACT

This paper describes a phrase splicing and variable substitution system which offers an intermediate form of automated speech production lying in-between the extremes of recorded utterance playback and full Text-to-Speech synthesis. The system incorporates a trainable speech synthesiser and an application specific set of pre-recorded phrases. The text to be synthesised is converted to a phone sequence using phone sequences present in the pre-recorded phrases wherever possible, and a pronunciation dictionary elsewhere. The synthesis inventory of the synthesiser is augmented with the synthesis information associated with the pre-recorded phrases used to construct the phone sequence. The synthesiser then performs a dynamic programming search over the augmented inventory to select a segment sequence to produce the output speech. The system enables the seamless splicing of pre-recorded phrases both with other phrases and with synthetic speech. It enables very high quality speech to be produced automatically within a limited domain.

1. INTRODUCTION

The phrase splicing and variable substitution system described in this paper offers an intermediate form of automated speech production which lies in-between the extremes of recorded utterance playback and full Text-to-Speech (TTS) synthesis. It enables very high quality speech, superior to that generated by TTS systems, though still inferior to pure recordings, to be generated in applications in which speech production is required within a limited domain. The system enables the seamless splicing of pre-recorded phrases both with other pre-recorded phrases and with synthetic speech; using the method described phrase splicing and variable substitution become essentially the same thing.

The system was developed initially for use in the IBM Mutual Fund Interactive Voice Response System. In this application the system is used principally for splicing together pre-recorded phrases. It is used to combine recordings of a few thousand mutual fund names, a few hundred numbers, and a few hundred carrier phrases, to synthesise any of the billions of possible sentences requested by the host system. In addition, words or phrases missing from the recordings can be synthesised and seamlessly spliced with the pre-recorded phrases.

A pure variable substitution system might involve a set of carrier phrases containing variables. A simple example is "The telephone number you require is XXXX", a problem which has been satisfactorily addressed before by recording digits in all possible immediate digit contexts. However, for more general variables, such as names for example, this may not be possible. In these cases the system described in this paper offers an attractive solution.

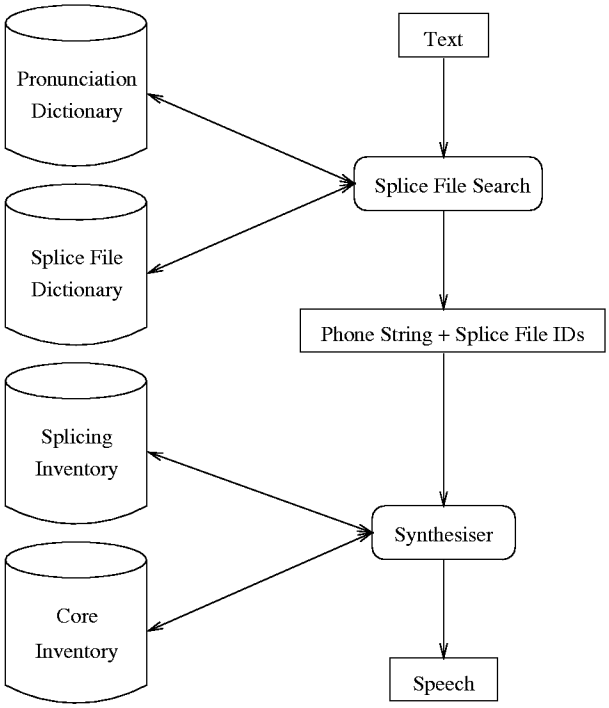

Figure 1: Splicing System Overview.

2. SYSTEM OVERVIEW

Figure 1 shows an overview of the phrase splicing and variable substitution system. The system is based upon the trainable speech synthesis system described in [3], henceforth termed the *core system*. In addition, processed recordings of the phrases to be spliced are also required, in the form of the splice file dictionary and splice file inventory.

During synthesis the text to be synthesised is supplied by a host system. A search algorithm is used in conjunction with the splice file dictionary and a pronunciation dictionary (used to supply the pronunciations of variables or unknown words) to determine the phone sequence to be synthesised given the text. The core

system's synthesis inventory is temporarily augmented with the inventories associated with the splice files used to construct the phone sequence. Finally, the synthesiser selects, modifies, and concatenates segments from the augmented inventory to generate the output speech.

The remainder of this paper is structured as follows. Section 3 describes the system construction details. Sections 4 and 5 describe the synthesis time operation of the system. Section 6 explains why the system works as it does, and Section 7 presents some results. Finally, Section 8 draws the conclusions of the current work, and Section 9 briefly discusses possible future work.

3. SYSTEM CONSTRUCTION

In order to perform phrase splicing or variable substitution it is first necessary to train the core system on the chosen speaker. This involves recording 45-60 minutes of speech together with the corresponding laryngograph signal. The speech data is used to train a set of decision-tree state-clustered hidden Markov models (HMMs), [1]. The HMMs are used to segment the training database into decision tree leaves. Synthesis information, such as segment duration, energy, pitch, endpoint spectral vectors, and the location of the moments of glottal closure through regions of voiced speech (obtained from the laryngograph data), is determined for each of these segments. Separate sets of trees are built from duration and energy data to enable the prediction of duration and energy during synthesis. Further details of the core system training procedure can be found in [2], [3].

The phrases to be spliced together or joined to variables must also be recorded in the chosen voice, together with a laryngograph signal. The splicing process does not alter the prosody of these phrases, and therefore it is important to ensure that they are recorded in prosodic contexts similar to those in which they will be used in synthesis. These files are processed using the HMMs in exactly the same way as the speech used to construct the core system. This processing yields a set of files, henceforth known as *splice files*, associated with each of these phrases. One of the splice files, termed the *lex file*, contains information about the words and phones in the phrase and their time alignment to the speech waveform. The other splice files contain synthesis information about the phrase identical to that described above for the core system; one of these files contains the speech waveform.

A *Splice File Dictionary* is constructed from the lex files to contain every word sequence of every length present in the splice files, together with the phone sequence aligned against those words. Silences occurring between the words in each entry are retained in the corresponding phone sequence definition. As an example, the splice file dictionary entries generated from the sentence "You have ten dollars only." are shown in Figure 2.

4. TEXT TO PHONE CONVERSION

During synthesis the text to be synthesised is converted into a phone sequence completely automatically. This is performed using a search algorithm, the splice file dictionary and a pronunciation dictionary. The process creates the phone string from phone sequences seen in the splice files where possible, and a pronunciation dictionary where not. This is desirable, compared to constructing it solely from a pronunciation dictionary, for two reasons:

- If the synthesis phone sequence adheres to splice file phone sequences (including silences) over large regions, then large fragments of splice file speech can be used in synthesis. This results in fewer joins and hence higher quality output speech.
- Pronunciation ambiguities (eg. the word "the" can be /DH AX/ or /DH IY/) can be resolved if such words are available in splice files in the appropriate word context.

You have ten dollars only	Y UW HH AE V X T EH N D AO L ER Z OW N L IY
You have ten dollars	Y UW HH AE V X T EH N D AO L ER Z
You have ten	Y UW HH AE V X T EH N
You have	Y UW HH AE V
You	Y UW
have ten dollars only	HH AE V X T EH N D AO L ER Z OW N L IY
have ten dollars	HH AE V X T EH N D AO L ER Z
have ten	HH AE V X T EH N
have	HH AE V
ten dollars only	T EH N D AO L ER Z OW N L IY
ten dollars	T EH N D AO L ER Z
ten	T EH N
dollars only	D AO L ER Z OW N L IY
dollars	D AO L ER Z
only	OW N L IY

Figure 2: Splice file dictionary entries generated from the sentence "You have ten dollars only". /X/ is the silence phone.

In the current system the search is performed using a left to right greedy algorithm. Initially the N word string to be synthesised is looked up in the splice file dictionary. If it is present, then the corresponding phone string is retrieved. If not, then the word string comprising the first $N-1$ words is looked up. This continues until either some word string is found or only the first word remains and this is not present in the splice file dictionary, in which case this word is looked up in the pronunciation dictionary. Having established the phone sequence for this first word (or word string), the process repeats for the remaining words in the sentence until the complete phone sequence is established. The identities of all the splice files used to construct the phone sequence are noted for use in synthesis.

5. SYNTHESIS

The use of the core system to perform TTS synthesis is described in detail in [3], but will be summarised here to aid understanding of the modifications made in the phrase splicing and variable substitution system.

In the TTS system the text to be synthesised is converted to a phone string by dictionary lookup, with the selection between alternatives for words with multiple pronunciations being made manually. The decision trees are used to convert the phone sequence into an acoustic, duration, and energy leaf for each feneme[1] 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. Pitch tracks are predicted using a separate trainable model. The next stage of synthesis is to perform a dynamic programming (DP) search over all the waveform segments aligned to each acoustic leaf in training, to determine the segment sequence to use in synthesis. The optimal set of segments is that which most accurately produces the required sentence after TD-PSOLA, [4],

[1] 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.

has been applied to modify the segments to have the requested duration, energy and pitch values. The cost function used in the DP algorithm therefore reflects the ability of TD-PSOLA to perform modifications without introducing perceptual degradation. Algorithms embedded within the DP can modify the requested acoustic leaf identities, energies and durations to ensure high quality synthetic speech. Once the segment sequence has been determined, energy discontinuity smoothing is applied. Finally, the selected segment sequence is concatenated and modified to match the requested duration, energy and pitch values using an implementation of the TD-PSOLA algorithm.

In the splicing system the first stage of synthesis is to augment the core system's segment inventory with the segments contained in the splice files identified by the splice file dictionary search described in Section 4. The segments and their related synthesis information is temporarily added to the same structures in memory used to hold the core inventory. The splice file segments are then available to the synthesis algorithms in exactly the same way as the core inventory segments. The only difference is that the new segments are marked as splice file segments, which enables them to be treated slightly differently by the synthesis algorithms.

The second stage of synthesis proceeds exactly the same as the phone-to-speech part of the TTS procedure described above, except that:

- The predicted pitch is simply the speaker's mean pitch value.
- During the DP search
  - Splice file segments are not costed against the predicted duration, energy or pitch values.
  - Pitch discontinuity costing is applied.
- After segment selection
  - The requested duration and energy of each splice file segment are set to the duration and energy of the segment selected.
  - The requested pitch of every segment is set to the pitch of the segment selected.
  - Pitch discontinuity smoothing is applied.

The pitch discontinuity between two segments is defined as the pitch on the current segment minus the pitch of the segment following the previous segment in the training database or splice file in which it occurred. There is therefore no discontinuity between segments which were adjacent in training or a splice file, and so these pitch variations are neither costed nor smoothed. In addition, pitch discontinuity costing and smoothing is not applied across silences in the speech longer than a predetermined threshold duration; these silences are assumed to be intonational phrase boundaries at which pitch resets are allowed. Pitch discontinuity smoothing operates as follows. The discontinuities at each segment boundary in the synthetic sentence are used to compute a cumulative discontinuity curve, being the running total of these discontinuities from left to right across the sentence. This cumulative curve is then low pass filtered. The difference between the filtered and unfiltered curves is computed, and these differences used to modify the requested segment pitch values.

6. DISCUSSION

The use of a separate splice file inventory, and the process of temporarily augmenting the core inventory with sentences selected from it, is preferred over simply ensuring that large fragments of sentences to be synthesised are present in the core system training data. This is because (i) it enables a very large number of splice file sentences to be available to the system without an explosion in the complexity of the synthesis DP search, and (ii) it enables splice file segments to be treated differently from core segments by the synthesis algorithms to ensure better quality output speech.

To understand why the modifications described in Section 5 result in high quality spliced or variable substituted speech, consider the behaviour of splice file speech under the DP cost function. Splice file segments are not costed against predicted duration, energy or pitch values. Also, the pitch continuity, spectral continuity and energy continuity costs between segments adjacent in splice files are, by definition, zero. Therefore, using a sequence of segments which were adjacent in a splice file has zero cost, except at the end points where the sequence must join something else. During synthesis therefore, deep within a region in which the synthesis phone sequence matches a splice file phone sequence, large fragments of splice file speech can be used at no cost.

Consider the point in the synthesis phone sequence where two splice file phone sequences are joined. Simply butting together the corresponding splice file waveforms would result in zero cost for duration, energy and pitch, right up to the join from both directions. However, continuity costs at the join could be very high, since no care has been taken to ensure continuity. The DP therefore automatically backs off from such joins, and splices in segments from the core inventory to provide a spectrally and prosodically smoother path between the two splice files. These core segments *are* costed against predicted duration, energy, and pitch, which splice file segments are not, but can nevertheless result in a lower total cost.

Pitch discontinuity costing is applied to discourage the use of segments with widely differing pitches next to each other in synthesis. In addition, after segment selection, the pitch contour implied by the selected segment pitches undergoes discontinuity smoothing in an attempt to remove any serious discontinuities which still occur. Since there is no pitch discontinuity between segments which were adjacent in a splice file, deep within splice file regions the smoothing has no effect and the pitch contour is unaltered. Obtaining the pitch contour through synthetic regions using this method might at first seem dangerous, but in fact it works surprisingly well; indeed it is possible to generate pitch contours for whole sentences in TTS mode using this method, again with surprisingly good results.

The result of the above is that deep within splice file regions, far from boundaries, the output speech is produced almost exactly as it was in the original recordings. At boundary regions between splice files, segments from the core inventory are inserted to provide a join which is spectrally and prosodically smooth. Words whose phone sequence was obtained from the pronunciation dictionary, for which splice files do not exist, are synthesised purely from segments from the core inventory, with the above algorithms again enforcing spectral and prosodic smoothness with the surrounding splice file speech.

7. RESULTS

Figure 3 shows the speech waveform and wideband spectrogram of the spliced sentence "You have twenty thousand dollars in cash.". Vertical lines show the underlying decision tree leaf structure, and "seg" labels show the boundaries of the actual speech fragments used to synthesise the sentence. The sentence was con-

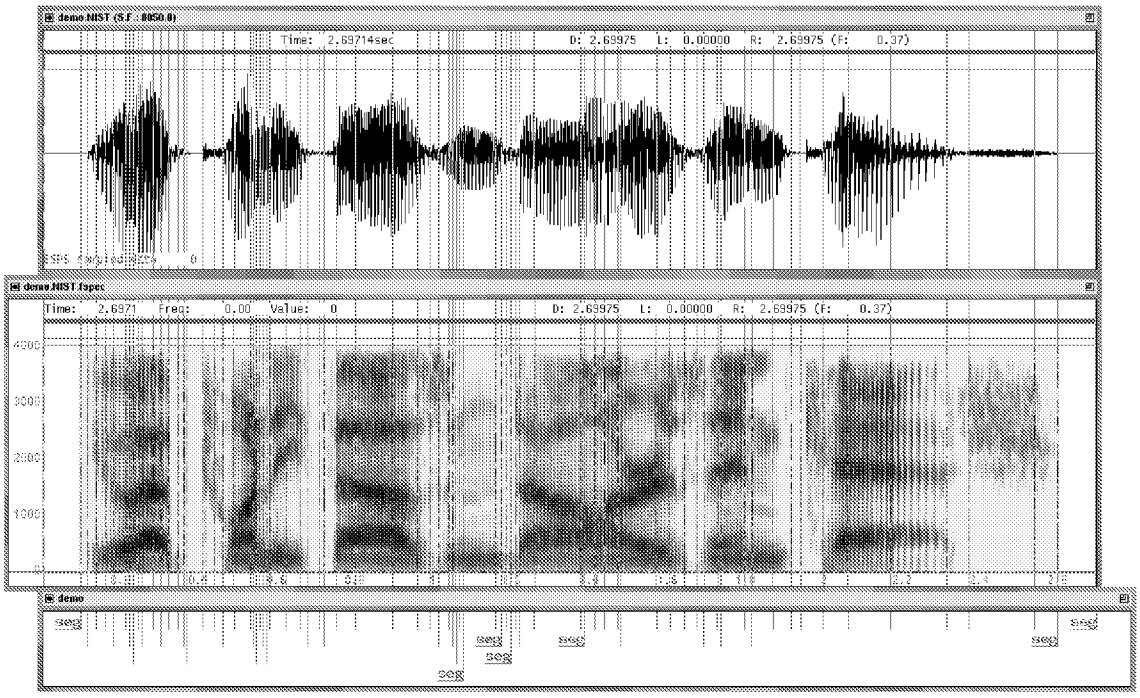

Figure 3: The spliced sentence "You have twenty thousand dollars in cash". Vertical lines show the underlying decision tree leaf structure, and "seg" labels show the boundaries of the actual speech fragments used to synthesise the sentence.

structed by splicing together the two sentences "You have twenty thousand one hundred dollars." and "You have ten dollars in cash.". As can be seen from the locations of the seg labels, the pieces "You have twenty thousan-" and "-ollers in cash" have been synthesised using large fragments from the splice files, with the "-nd do-" region being constructed from three fragments from the core inventory.[2] When performing variable substitution, the situation is essentially the same, except that the region constructed from the core inventory may be one or more words long.

The speech can be heard to be of extremely high quality. The use of large fragments from appropriate prosodic contexts means that the sentence prosody is extremely good, much better than can currently be expected using TTS synthesis. In addition, the use of large fragments results in a small number of joins, minimising distortion due to concatenation discontinuities.

8. CONCLUSION

The use of the dynamic programming algorithm with the modifications described in Section 5 enables the seamless splicing of pre-recorded speech both with other pre-recorded speech and with synthetic speech, to give very high quality output speech. The use of the splice file dictionary and related search algorithm enables, within a limited domain, a host system to request and obtain very high quality synthetic sentences constructed from the appropriate pre-recorded phrases where possible, and synthetic speech where not.

[2] It is possible, though unlikely, that fragments from other regions of the splice files are used to fill the boundary region.

9. FUTURE WORK

Future work will be directed at improving the quality of purely synthetic regions and at reducing the image size of the runtime system. In addition, future effort may be directed at improving the splice file search described in Section 4. Possible improvements to this algorithm include:

- Using phone or word context information during the search to maximise the overlap between splice files at join regions.
- Using a dynamic programming search instead of the greedy algorithm.

10. 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] Donovan, R.E. (1996) *Trainable Speech Synthesis*, PhD. Thesis, Cambridge University Engineering Department.[3]

[3] Donovan, R.E., and Eide, E.M. (1998) The IBM Trainable Speech Synthesis System, *Proc. ICSLP'98, Sydney.*

[4] Moulines, E., and Charpentier, F. (1990) Pitch-Synchronous Waveform Processing Techniques for Text-to-Speech Synthesis Using Diphones, *Speech Communication*, 9, pp. 453–467.

[3] Available by anonymous ftp to 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