HMM-BASED SMOOTHING FOR CONCATENATIVE SPEECH SYNTHESIS

M. Plumpe, A. Acero, H. Hon, X. Huang

Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA

ABSTRACT

This paper will focus on our recent efforts to further improve the acoustic quality of the Whistler Text-to-Speech engine. We have developed an advanced smoothing system that a small pilot study indicates significantly improves quality. We represent speech as being composed of a number of frames, where each frame can be synthesized from a parameter vector. Each frame is represented by a state in an HMM, where the output distribution of each state is a Gaussian random vector consisting of $\mathbf{x}$ and $\Delta\mathbf{x}$. The set of vectors that maximizes the HMM probability is the representation of the smoothed speech output. This technique follows our traditional goal of developing methods whose parameters are automatically learned from data with minimal human intervention. The general framework is demonstrated to be robust by maintaining improved quality with a significant reduction in data.

1. INTRODUCTION

In contrast to most Text-To-Speech (TTS) systems (including both formant and concatenative synthesizers) that requires human experts to hand-craft and fine-tune the synthesis units (or unit parameters) [2][3], Whistler uses an automatic procedure to configure and generate the synthesis units directly from any recording database [7][8]. Whistler is a concatenative synthesizer, using units that are segmented by Microsoft's speech recognition engine Whisper [6]. Whistler uses decision-tree clustered phone-based units. Each unit is a cluster of phones, whose phonetic contexts and other characteristics such as stress are used to traverse automatically trained decision trees for finding the cluster [5]. At synthesis time, Whistler loads the appropriate units from the database, synthesizes the units from the stored residuals and filters, and concatenates them.

One of the most significant problems with concatenative synthesizers is spectral mismatch at concatenation point. This is normally caused by not having the identical context for the units, intra-speaker variation, or errors in segmentation. Our selection process clearly contributes to errors, due to the fact that no manual correction is performed on the segmentation. Common solutions to reduce this problem include manual re-selection of units, manual adjustment of unit boundaries, spectral smoothing, and the use of longer units to reduce the number of concatenations. Following our desire to have a system that is trained automatically, we always avoid manual re-selection of units and manual adjustment of unit boundaries in favor of automatic approaches. We have also avoided traditional spectral smoothing techniques due to the decreased naturalness of the resulting synthesized speech.

The smoothing technique discussed in this paper uses a rigorous probabilistic framework backed up with automatically generated statistics to smooth the spectrum of the vocal tract filter across unit boundaries. The statistics help ensure that smoothing only occurs when necessary and to the extent required to achieve a transition similar to transitions seen in training. Compared to traditional spectral smoothing techniques that try to reduce distortion uniformly, our smoothing technique will smooth the spectrum according to what was observed in the junctions of real speech during training.

One relevant approach is described in [12]. Their approach differs from ours mainly in several aspects. First, they use a Mel log spectral approximation filter to synthesize the speech, rather than using a filter/residual arrangement. The primary difference is that we retain a filter calculated from an actual speech utterance for the mean of the Gaussian, along with its corresponding residual, for this reason we describe our system as a smoothing system instead of a synthesis system. Next, they model duration by HMM as well, which complicates the search for optimal parameters. Finally, our framework includes a multiple instance approach. Another multiple instance approach to synthesis without any waveform processing is described in [9].

This paper is organized as follows. In Section 2 we discuss the probabilistic framework and statistics of the smoothing technique, as well as a multiple instance framework. In section 3 we discuss the practical effects of the technique and experimental evaluations of speech output. Finally in section 4 we summarize our major findings and outline our future work.

2. SMOOTHING

The motivation of our approach is to develop a technique and statistics that will enable the synthesizer to take two units whose boundaries have a perceptually significant spectral mismatch and modify these units such that the spectral mismatch is reduced or eliminated.

2.1 Probabilistic Framework

A given segment of input speech X can be decomposed into N *frames*. For voiced speech, a frame is a set of P samples where

P is the local pitch period, while for unvoiced speech P can be set to a nominal length of say 10ms. From here on, we will assume a parametric synthesizer that can synthesize a frame given a vector $\mathbf{x}_i$ at a time instant t_i, or an utterance given the vectors X at time instants T. Each frame is represented by a state in an HMM, with each state's output distribution modeled as a Gaussian random vector consisting of $\mathbf{x}$ and $\Delta\mathbf{x}$, with a diagonal covariance matrix. As HMMs are normally viewed from a recognition perspective rather than a synthesis perspective, we will now further develop the framework.

Because of the variability present in each speech segment, we assume a probabilistic model for X

$$P(X) = \prod_{i=0}^{N-1} N(\mathbf{x}_i \mid \boldsymbol{\mu}_i, \Sigma_i) \cdot \prod_{i=1}^{N-1} N(\mathbf{x}_i - \mathbf{x}_{i-1} \mid \Delta\boldsymbol{\mu}_i, \Delta\Sigma_i) \quad (1)$$

where we have assumed that the vector $\mathbf{x}_i$ follows a Gaussian distribution, and that all vectors are independent of each other. Since we know adjacent vectors are highly correlated, we can capture this correlation by adding a Gaussian component for the delta information, and assuming they are also independent. This assumption has been successfully used in speech recognition, and we use it here as well because of its simplicity.

Given the probabilistic model described above, we can find the maximum likelihood estimate X_{ML} that maximizes (1) by taking the derivative and equating to 0. If we further assume a diagonal covariance matrix, maximizing (1) is equivalent to minimizing

$$E = \sum_{i=0}^{N-1} \frac{(x_i - \mu_i)^2}{\sigma_i^2} + \sum_{i=1}^{N-1} \frac{(x_i - x_{i-1} - \Delta\mu_i)^2}{\Delta\sigma_i^2} \quad (2)$$

where x_i refers to any component of the vector $\mathbf{x}_i$ for simplicity. Equation (2) reflects a compromise between keeping the static vector parameters close to the mean and maintaining the differential information. A generalization of (2) that allows for more control of the relative importance of the static and the delta information can be obtained by introducing a constant D.

$$E = \sum_{i=0}^{N-1} \frac{(x_i - \mu_i)^2}{\sigma_i^2} + D\sum_{i=1}^{N-1} \frac{(x_i - x_{i-1} - \Delta\mu_i)^2}{\Delta\sigma_i^2} \quad (3)$$

Minimizing (3) leads to a tridiagonal set of linear equations in x_i that can be solved very efficiently [14].

The usefulness of this approach arises when the speech phrase X is a concatenation of different units. In this case, the delta constraint will decrease the spectral discontinuities at the unit boundaries. Instead of calculating the mean, we choose an actual speech utterance for μ, in order to retain the naturalness inherent in concatenative synthesizers. Synthesis is accomplished by finding the sequence of vectors X that maximize the HMM likelihood. This is accomplished by solving the tridiagonal set of linear equations that result from the minimization of (3). Note that a separate set of tridiagonal equations must be solved for each component of the vector $\mathbf{x}_i$.

Due to the prosody modification required in speech synthesis, the number of frames in a synthesized unit will be in general different than that of our model (1). This requires a mapping between N distributions in our model and M frames in the target segment in speech synthesis. We can define a linear mapping with time that repeats or skips states of the HMM, corresponding to duplicating or removing distributions to accommodate a different number of frames.

2.2 Selection of Parameters

Our current implementation uses a source-filter representation [1] of the speech as $\mathbf{x}$, so we will describe it here as such. The features of the vectors $\mathbf{x}$ include the filter coefficients, the residual samples, and amplitude information. We have found that we can improve quality without smoothing the residual, and as our initial attempts at smoothing the residual were unsuccessful, we will temporarily ignore its presence in the $\mathbf{x}$ vector. We have found smoothing the RMS energy to increase quality, though there are many other possible approaches we have not compared this against.

If the filter portion of the parameter vector $\mathbf{x}$ represented the formant frequencies and bandwidths of a formant synthesizer, minimizing (3) can be viewed as implementing an automatically trained formant synthesizer. Due to the difficulty in accurately estimating formants, we do not use formants at this time. Instead, any of the standard source-filter representations could be used for $\mathbf{x}$, such as LPC reflection coefficients, line spectral frequencies [10] or cepstral coefficients. For the experiments in this paper, we choose to use line spectral frequencies, also referred to as line spectral pairs, or LSP's, because of their stability properties and the fact that each LSP only influences a local region of the spectrum. This characteristic makes independent smoothing of the LSP coefficients a reasonable approach to smoothing the spectrum. This provides motivation for the use of a diagonal covariance matrix as described in section 2.1.

The primary drawback of the LSP coefficients is that the bandwidth of a pole is determined by the proximity of two coefficients. The averaging and smoothing of LSP coefficients will have a tendency to produce LSP's that are more evenly spaced, resulting in formants with increased bandwidths. This increase in bandwidths results in quality degradation. Retaining actual speech data for μ helps reduce this effect.

In smoothing the filter coefficients, we are making the implicit assumption that the filter and excitation are independent, which they are not. If the excitation is not fairly white, the formants in the modified filter will conflict with the formants remaining in the residual, producing muddled speech. We found that, at a 22kHz sampling rate, a 14^{th} order filter was insufficient, and increased the filter to a cascade of 14^{th} and 8^{th} order filters. Alternatively, a source model and smoothed source parameters might eliminate this problem and allow for large modifications.

2.3 Parameter Estimation

To estimate the smoothing parameters of each unit, we need to first define the number of states N and their time instants T. To do this we select the unit with highest probability from the segmentation HMM that lies in a neighborhood of the mean pitch, duration, and amplitude (discarding outliers) [7], and call

this the *golden unit*, whose parameters will be used as the static mean μ. We set N and T to be the number of states the time instants for the states of the golden unit.

We can estimate the means and variances in (1) for each unit u, from all the instances of that unit u_j present in a training database by taking the sample means and variances, which are the maximum likelihood estimates given the model in (1). The unit instances need to be aligned prior to computing the means and variances. From the HMM framework, viterbi alignment minimizing (2) provides this alignment. We calculate the mean and delta-mean from the golden unit, and set the variances to unity. We have found that iteratively aligning based on the calculated statistics does not significantly impact quality, likely due to retaining the same mean across iterations.

2.4 Data Reduction

In practice we can reduce the number of states by tying some of the states together. This parameter sharing allows us to reduce the amount of storage required for the synthesizer, as well as the need for a larger training database. The motivation behind the approach discussed here is that the problem we are attempting to solve occurs at unit boundaries, so the most important statistics are those at the unit boundaries.

We first train as described in Section 2.3, then discarded the statistics for all states except the first and last of each unit. A simpler but likely nearly identical approach would be to simply calculate the sample mean and variance for the first and last frame of each unit, skipping the alignment step. The static mean is retained for all states, as this is simply the information present in the golden unit. Statistics for remaining states were generated at the runtime using the following heuristics, designed to limit filter modifications to states near unit boundaries:

- The variance (σ^2) is halved for each state away from a boundary. This tends to limit filter changes to several states around the boundary.
- The Delta Variance ($\Delta\sigma^2$) is linearly interpolated from the first to the last state.
- The Delta Mean ($\Delta\mu$) is the average of the statistic used for the previous delta mean and the current actual delta mean. In this way, the delta mean comes closer to the actual delta mean each state away from the boundary.

An informal study similar to the one described in section 3 showed almost no perceptual difference between the full smoothing system and the data reduced smoothing system. Approximately one out of every twenty sentences was found to have a spectral discontinuity that was satisfactorily eliminated using the full data smoothing system that is still perceptible in the data reduced smoothing system.

2.5 Multiple Instances

In order to have a scalable system, we wish to have a multiple instance database and synthesis technique. At synthesis time, the synthesizer will compare all instances of the required synthesis units, selecting the instances that will result in the best overall speech quality. By increasing the number of instances in the database, we can make a trade-off between speech quality and database size and CPU utilization.

A multiple instance system fits elegantly into the smoothing framework. We would like to partition the space represented by each decision-tree clustered phone-based unit into multiple regions, so that each region can be more closely modeled by its own mean and actual speech data that represents the mean. One of the primary shortcomings of the smoothing system is that large modification of the units results in less natural sounding speech. By having multiple instances, the amount of modification will be reduced, resulting in higher quality speech.

2.5.1 Training

Attempting to perform clustering based on all features in a unit (on the order of 200 features) is unlikely to be successful without an extremely large amount of input data. To solve this problem, we cluster the data using only the first and last frame of each unit instance. This is a reasonable approach since the goal is to reduce mismatch and unit boundaries. Clustering is performed using a k-means type algorithm, using (2) with variances set to unity as the distance measure to iteratively identify the cluster mean.

As in the single instance smoothing system, choosing the instance to represent the cluster is a key task, as this instance will be used in synthesis. It is thus selected from a subset of all instances that survive outlier pruning and are high in HMM probability. The instance from this set closest to the mean of each cluster is chosen to represent that cluster.

For each cluster, the instances assigned to that cluster are aligned against the instance selected to represent that cluster as in the single instance system. The statistics $\Delta\mu_i$ are calculated for each cluster as before. The variance statistics σ_i, and $\Delta\sigma_i$ are calculated for all clusters using a grand variance scheme, since the number of instances available for training is limited.

2.5.2 Synthesis

Several techniques could be used to identify which cluster to synthesize. One possibility is to perform a simple viterbi type search, with the cost being the spectral mismatch at the boundary. If we assume that degradation occurs in smoothing due to filter modification, (3) with $D=0$ measures this degradation. Performing a viterbi search while smoothing, with this degradation measure as the cost, will select the clusters that result in the least degradation due to filter modification. This is the technique that we use, although we have found that it is important to weight the distance at the boundaries more heavily, as this is still the key point of degradation.

3. EXPERIMENTAL EVALUATION

The training database used contains over 7,000 sentences from a single speaker, giving approximately 100,000 individual phone units. All instances of each generalized triphone were aligned as discussed in the previous section. The means and variances were calculated for each frame. At synthesis time,

(3) is minimized over all frames between pauses in the speech output. We empirically found 4 to be a good value for the parameter *D*. The calculated filter is used with the residual from the golden unit to synthesize the speech. The speech synthesized for evaluation used transplanted prosody from sentences not included in the training database.

The single instance smoothing system was found to significantly decrease spectral discontinuities at concatenations where discontinuities should not occur, while leaving large spectral jumps where they belong. See Figure 1 for an example of the input and output for the first LSP coefficient. The system does have drawbacks, for example regions that should have highly variable filter coefficients, such as fricatives, will have the filter smoothed, although we have found this to be a minimal problem in practice. A greater problem is a fairly constant buzziness, which we believe to be introduced through mismatch in the filter and residual.

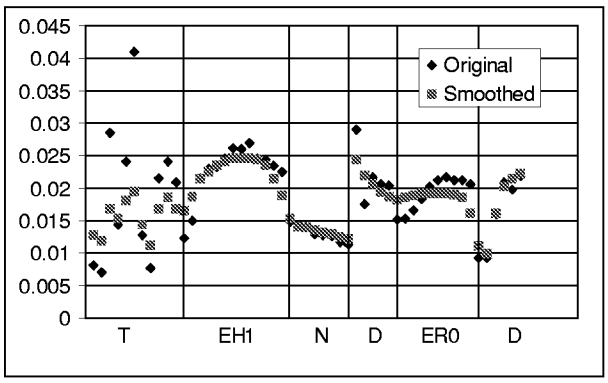

Figure 1. Plot of the original and smoothed first LSP coefficient for each from for the word "tendered." Smoothing leaves appropriate mismatches such as at the /N/ to /D/ boundary, but eliminates mismatch where it doesn't belong, such as at the /D/ to /ER0/ boundary.

In a small informal test consisting of five naïve listeners performing a preference evaluation of ten sentences with and without smoothing, the smoothing approach came out as the clear winner in terms of speech quality, as shown in Table 1.

System	**Percent Preferred**
Smoothed	58
Unsmoothed	20
No Preference	22

Table 1. Results of an ABX type evaluation comparing the smoothed and unsmoothed synthetic speech.

4. SUMMARY

A robust, automatically trained smoothing system was presented in this paper. It is designed to reduce spectral discontinuities in a concatenative speech synthesizer. Through the use of static and dynamic statistics, the spectrum is modified only when needed and only in an amount appropriate for each particular concatenation. Informal perceptual studies indicate that the algorithm is a significant improvement.

There are several potential areas for future study. One of the key drawbacks of the current implementation is that it tends to result in increased formant bandwidths. This could be solved by adding a third term to the error equation that constrains the delta between LSP coefficients during a given frame. Another approach would be to use formant frequencies and bandwidths as the features to be smoothed. Our initial experiments at using the same technique to smooth the Fourier transform of the residual were unsuccessful. Further effort at smoothing the residual, perhaps through a parametric model such as the LF model should result in the ability to make larger changes to the spectrum, as the residual could be changed accordingly.

5. REFERENCES

[1] Acero A.. "Source-Filter Models for Time-Scale and Pitch-Scale Modification of Speech". *IEEE Proc. ICASSP*. Seattle, May 1998, pages 881-884.

[2] Allen J., Hunnicutt S., and Klatt D. *From text to speech: the MITalk system*. MIT Press, Cambridge, MA, 1987.

[3] Bailly G. and Benoit C., editors. *Talking Machines: Theories, Models, and Designs*. Elsevier Science, 1992.

[4] Donovan R.E. and Woodland P.C. "Improvements in an HMM-Based Speech Synthesizer". *Proc. Eurospeech Conference*, Madrid, Spain, 1995, pages 573-576.

[5] Hon H., Acero A., Huang X., Liu J., and Plumpe M.. "Automatic Generation of Synthesis Units from Trainable Text-To-Speech Systems". *IEEE Proc. ICASSP*. Seattle, May 1998, pages 293-206.

[6] Huang X., Acero A., Alleva F., Hwang M.Y., Jiang L. and Mahajan M. "Microsoft Windows Highly Intelligent Speech Recognizer: Whisper". *IEEE Proc. ICASSP*. Detroit, May 1995.

[7] Huang X., Acero A., Adcock J., Hon H., Goldsmith J., Liu J., and Plumpe M. "Whistler: A Trainable Text-to-Speech System". *proc. ICSLP*. Philadelphia, Oct, 1996.

[8] Huang X., Acero A., Hon H., Ju Y., Liu J., Meredith S., Plumpe M., "Recent Improvements on Microsoft's Trainable Text-To-Speech System - Whistler". *IEEE Proc. ICASSP*. Munich, Germany, April, 1997, pages 959-962.

[9] Hunt A.J. and Black A. W. "Unit Selection in a Concatenative Speech Synthesis System Using a Large Speech Database", *IEEE Proc. ICASSP*. Atlanta, May 1996, pages 373-376.

[10] Itakura F. "Line Spectrum Representation of Linear Predictive Coefficients of Speech Signals", *J. Acoust. Soc. Am*, 57, 535(a), 1975.

[11] Klatt D. "Review of text-to-speech conversion for English". *Journal of the Acoustical Society of America*, 82(3):737-793, 1987.

[12] Masuko, Takashi, et al. "Speech Synthesis Using HMMs with Dynamic Features". *IEEE Proc. ICASSP*. Atlanta, 1996, pages 389-392.

[13] Microsoft Research's Speech Technology Group web page: http://www.research.microsoft.com/research/srg/.

[14] William H. Press, et al. *Numerical Recipes in C*, Cambridge University Press, 1992, pages 50, 51.