COMPRESSION OF LINE SPECTRAL FREQUENCY PARAMETERS WITH ASYNCHRONOUS INTERPOLATION

Rachel Moldover and Alexander Kain

Center for Spoken Language Understanding
Department of Science & Engineering
Oregon Health & Science University
Portland, OR, USA

ABSTRACT

TTS systems require a trade-off between size and speech quality. A larger acoustic inventory allows synthesis of speech that sounds more natural. The Asynchronous Interpolation Model improves the quality to size ratio, allowing better compression of large acoustic inventories, as well as better quality speech from a small system. At maximum compression, our method represents most phonemes by a single frame of data. Coarticulation effects are specified as context-specific non-linear interpolation functions. Dividing the speech features into multiple data streams allows asynchronous interpolation. In this study, AIM was applied to LSF parameters. Varying the number of streams allows for variable amount of compression. We used three different objective measures to investigate the effect of number and partitioning of streams. The first few weight functions (and the last one) seem to offer the most error reduction. Partitions separating the first 6 LSFs score well with all three measures.

***Index Terms*—** speech synthesis, temporal decomposition, compression, acoustic inventory, TTS

1. INTRODUCTION

Computer synthesized speech, often known as "text to speech," (TTS) has many applications. TTS allows blind people to listen to their e-mail, or portable car navigation systems to give real-time verbal directions to the driver. It is possible to produce speech that is intelligible using a small amount of memory. For example, Yamaguchi et. al [1] wrote a rule-based module for speech synthesis which used only 275 kB for both parameters and code. However, the speech from such synthesizers is of low quality; it sounds noisy and robotic. Today, the most natural sounding TTS is created by concatenative unit selection synthesis. A larger acoustic inventory allows synthesis of speech that sounds more natural.

How can a speech database be compressed with minimal loss of quality after resynthesis? Compression of speech is done constantly by the telephone networks. Many different compression / decompression (codec) algorithms have been developed for this application. But, because telephone speech is not phonetically labeled, these codecs cannot take advantage of phoneme-specific knowledge. The Asynchronous Interpolation Model (AIM) makes use of this phonetic knowledge to achieve a greater ratio of quality to database size [2].

Rachel Moldover passed away in December 2008.

2. METHODS

2.1. General AIM methods

An acoustic inventory that has been compressed by AIM has three main components: basis vectors, weight functions, and partitions. Roughly speaking, the basis vectors are *target* feature values for a speech event (such as a phone). For example, in this study, each basis vector consisted of 18 line spectrum frequency parameters (LSFs). The weight functions are context-specific interpolation functions. (A simple weight function could be a straight line, or a sigmoid). To allow for asynchronous transitions, the partitions define which weight function applies to which features. (For example, the first 2 LSFs might be associated with one weight function, and the remaining 16 with another.) The AIM is similar in many ways to temporal decomposition [3, 4]. However, in the AIM compression of an acoustic inventory, a single basis vector may be used in multiple locations, allowing further compression. In addition, the AIM use of partitions allows us to efficiently model asynchronous transitions and provides more options for variable compression / quality.

The AIM is motivated by the way in which speech events (such as phonemes) transition into one another. It is well known that phoneme boundaries are indistinct. It is not as well recognized that there may be multiple *boundaries*. Not all speech features transition at the same time or rate. Consider the diphone /i: v/ in which the movement of the formants begins long before the onset of frication. In a sense there are two boundaries; one boundary at which formant movement begins and another at which frication begins.

AIM compression may be summarized in five steps:

1. Convert the speech signal to interpolable features.
2. Choose locations of basis vectors.
3. Choose partitions to define each feature stream
4. Calculate weight functions.
5. Merge basis vectors as desired.

AIM compression must be applied to speech features that are interpolable with respect to time. Linear predictive coefficients (LPC) cannot be used, but formant frequencies and LSF values can be interpolated. (Raw speech waveforms are not interpolable at the time scale which is of interest to us.) In this study, LSF features were used.

After this initial compression to LSF values, each item in the acoustic inventory is further compressed to basis vectors, weight functions, and partitions. Given a speech waveform, let the short-term speech signal $\mathbf{x}$ at frame m be equal to a synthesis operation $\mathcal{S}$

on feature vector $\mathbf{f}[m]$, with the features partitioned into N streams $\mathbf{s}_n[m]$

$$\mathbf{x}[m] = \mathcal{S}(\mathbf{f}[m]) = \mathcal{S}(\mathbf{s}_1[m], \ldots, \mathbf{s}_N[m]) \quad (1)$$

where different streams can represent different types of feature trajectories (one stream could represent formants and another could represent voice-source). Or, different streams can represent partitions of the same type of feature (one stream could represent the odd LSFs and another could represent the even LSFs).

For the sake of simplicity, we will consider an AIM in which a feature value only depends on the basis vectors directly to the left and right (in time). The weight function will be represented by a value for each frame. A single feature value may then be calculated as follows:

$$\mathbf{s}_n[m] = (1 - w_n^{u_l \to u_r}[m]) \cdot \mathbf{b}_s^{u_l} + w_n^{u_l \to u_r}[m] \cdot \mathbf{b}_s^{u_r} \quad (2)$$

u_l and u_r are acoustic events to the left and right of frame m. $w_n^{u_l \to u_r}[m]$ are given from the frame associated with event u_l to the frame associated with u_r. In later implementations of AIM, the weight functions may be represented in a different form.

2.2. AIM applied to LSF speech model

2.2.1. Speech to LSFs

Previously, AIM has been successfully applied to a model of speech which separates the signal into formants and voice-source [5]. In this work, AIM was applied to an LSF model of speech. We began with a sentence from the DARPA TIMIT acoustic-phonetic continuous American English speech corpus [6]. We used speech recorded at the Center for Spoken Language Understanding (CSLU) which has been labeled phonetically as well as with pitch marks. The speech was downsampled from 22050 to 16000 Hz. Eighteen LSFs were calculated from overlapping, windowed 25-ms frames.

2.2.2. LSFs to AIM

We used AIM compression to convert the LSFs into local basis vectors and weight values. Basis vector values were typically extracted from the midpoints of phonemes; however, since basis vectors represent single acoustic events, some phonemes needed to contain several basis vectors. Specifically, diphthongs contained two separate basis vectors for the two different targets (/aI/: "aI1", "aI2"), voiced plosives contained two basis vectors for closure and burst (/b/: "bc", "b"), and unvoiced plosives contained three basis vectors for closure, burst, and aspiration (/t/: "tc", "tb", "th"). Finally, we represent affricates as a combination of other basis vectors (/tS/: "tc", "tb", "S").

Weight functions were represented by values stored for every frame of speech. Weight values were calculated for each frame to minimize the mean squared error in the feature values.

2.2.3. AIM to speech

The synthesized feature values (LSFs) were calculated from the weight values and partition information. Copy synthesis was used to recreate the original sentence.

2.3. Ways of measuring error

Error was calculated in three different ways. The error was calculated as the root-mean-square (RMS) error between the original and the synthesized LSF vectors.

We did this with raw LSFs and then with LSFs weighted by a perceptually-motivated warping function (mel-scale). Finally, the log spectral distance (LSD) was also calculated.

2.4. Varying number of streams

As a study of compression versus quality we varied the number of streams used to represent a speech sample and looked at the reconstruction error.

One of the challenges in optimizing AIM compression is to decide which features transition between phonemes most synchronously and therefore should be modeled by the same stream. A feature vector with d dimensions can be represented by anywhere from one to d streams (d streams would be equivalent to no AIM compression). For any number of streams between 2 and $d-1$, there are many ways to associate features with each stream or *partition* them. For example, suppose we have 18 features and 2 streams. We could represent feature 1 by the first stream and represent features 2 through 18 by the second stream. Or, we could represent the odd-numbered features by the first stream and the even-numbered features by the second stream.

2.4.1. Continuous partition assumption

The total number of ways that the features can be partitioned into streams is given by the Bell number [7].

The Bell number for 18 features is prohibitively large. (e.g. the Bell number for 13 is 27,644,437) Therefore, for this experiment, we made an assumption that synchrony is local in frequency. Therefore, we only considered partitions composed of adjacent features. For example, suppose we had only three features, numbered sequentially. We would consider the following partitions: ([1] and [2,3]) or ([1,2] and [3]). We would not consider ([1,3] and [2]), since features 1 and 3 are not adjacent.

Within these constraints, we tried all possible partitions for every number of streams from 1 to 18 (131,072 in total). For each possible set of partitions, we calculated the error. We saved the set of partitions with the lowest error value for each number of streams. Then the lowest error values were graphed for each of the three objective measures.

2.4.2. Random feature values

As one step towards understanding optimal AIM of speech data, in addition to applying AIM to features extracted from speech, we also created random feature values. At first random features with a flat distribution were tested. Then, we created random features with the same mean values and variances as LSFs from a speech sample. These random features were also compressed with AIM, partitioned in various ways, and the best results saved.

3. SOLUTION TO A THEORETICAL PROBLEM

In order to better interpret our experimental results, we considered a theoretical problem with an exact solution. Suppose there are only two basis vectors, between which there are an infinite number of frames. Let the first basis vector consist of all zeros and let the second consist of all ones. Let the feature values be random numbers with a flat distribution between zero and one. Now, instead of a fixed

W	Flat Distribution, RMS Error	Speech Distribution, RMS Error	Speech Distribution, Weighted RMS Error
2	1–17,18	1–9,10–18	1–2,3–18
3	1–13,14,15–18	1–9,10,11–18	1,2,3–18

Table 1. Best partitions in random features. W represents the number of weights.

W	unweighted MSE	weighted MSE	LSD
2	1–16,17–18	1–2,3–18	1,2–18
3	1–6,7–16,17–18	1,2–4,5–18	1,2–6,7–18
4	1–6,7–12,13,14–18	1,2,3–6,7–18	1,2–6,7–12,13–18
5	1–6,7–12,13,14–16,17–18	1,2,3,4–6,7–18	1,2,3–6,7–12,13–18

Table 2. Best partitions of LSFs. W represents the number of weights.

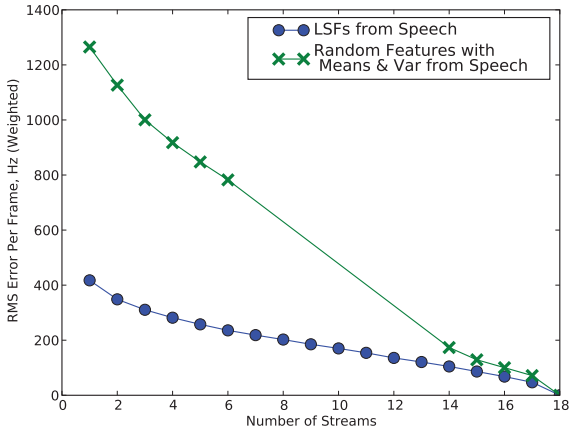


Figure 1. Random features versus speech features.

number of feature dimensions and a variable number of weight functions, consider a single weight function which is used to represent a variable number of feature dimensions.

Let x be the number of features per frame. The mean error per frame (totaled over all features) is given by $(x-1)/12$, even when $x = 1$ (in which case the error is zero). Suppose we have n features represented by a single weight function. Now we decide to add a second weight function. What is the best way to partition the n features? If we choose to represent p by the first weight function and $n-p$ by the other, then the error will be $(p-1)/12 + (n-p-1)/12 = (n-2)/12$, regardless of what number we choose for p! If we choose to use the first weight function to represent a single feature, this chosen feature will be perfectly modeled (with an error of zero). The rest of the $n-1$ features will be represented by the second weight function for an total error of $(n-2)/12$. Therefore, it makes no difference how we partition the features. The error is reduced by $1/12$ each time we add a weight function.

4. RESULTS

4.1. Partitions

Table 1 shows a selection of the best partitions found for random features. The data was based on 491 frames, and partition choice in the first column is mostly affected by random variation. The second column shows random features with means and variances taken from speech. The means and variances taken from the speech data favor a partition between nine and ten. LSF values with high variance are more likely to be modeled separately (as are LSFs which vary independently). The LSF values in the mid-range (9 through 12) tend to have the highest variance. Column 3 shows the effects of a perceptually weighted error measurement; the lowest LSF values are partitioned separately.

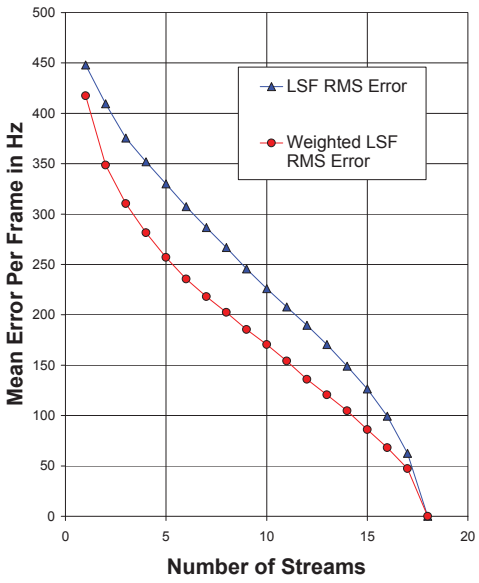
Table 2 shows the best partitions found for actual speech data using each of the three error calculations. The weighted MSE and LSD place much more importance on correct modeling of the first two LSFs as expected. With all methods, it is common to have a partition boundary between LSFs 6 and 7. This makes sense as we know that the first three pairs of LSFs often track the first three formants.

4.2. Comparison and analysis of error curves

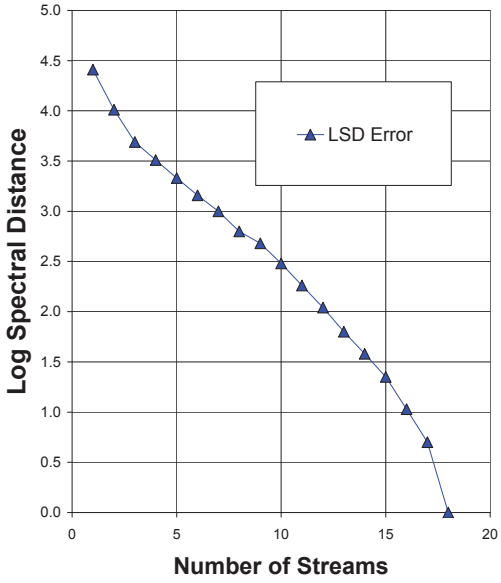
Figure 1 shows two sets of data. The lower trace shows the best results after compression of a TIMIT sentence. The upper trace shows the best results for compression of random features. The random feature values were given the same mean and variance values as the LSFs from the TIMIT sentence. Each data point represents the lowest weighted LSF error that was obtained after trying all possible contiguous partitions. Errors are measured between the LSF-coded waveform and the LSF-AIM-coded waveform, thus using as many streams as there are parameters results in zero error.

One can see that the error in the random data is not quite linear, as it would be for features values with a constant mean and variance. There is a distinctive shape due to the means and variance from speech and due to the weighted error measure. Comparing the random data to the speech data in Figure 1 shows that the LSFs are well correlated. The speech data can be represented by one weight function with a much lower error rate than random data. This confirms that LSFs are a reasonable candidate for AIM compression.

Figure 2(a) shows the root-mean-square (RMS) error in LSFs per frame. Raw error and weighted error are both shown. Figure 2(b) shows the error calculated as the log spectral distance between synthesized signals. These results give some idea of the trade-off between compression and quality. The similarity between these curves demonstrates that this trade-off is somewhat independent of



(a) RMS error: raw and weighted.



(b) Log spectral distance (LSD).

Figure 2. Error curves.

the method chosen for measurement of error. The sharper drops at the beginning and ends of these curves means that the largest gains in quality may occur with the first few weights (And the last few; but using 18 weight is equivalent to just using LSF compression).

5. FUTURE WORK

There are many, many ways to improve and expand the AIM. We need to find an error function that best reflects listener opinion for the type of distortions introduced by AIM compression. This would then potentially drive changes in the function used to calculate weight functions from features. AIM must be compared to other TTS methods with similar size and quality. We would like to consider varying the number of streams to depend on the particular diphone transition. We must investigate how to optimize basis vector location and number of basis vectors. In this study, the number of basis vectors (and locations) *per partition* was constant, but varying these might improve quality as well. Finally, the best means of quantization of all parameters should also be considered.

6. ACKNOWLEDGMENTS

This work was supported by NSF Grant 0713617. The views in this paper do not necessarily reflect those of the NSF.

7. REFERENCES

[1] M. Yamaguchi and J.-P. Hosom, "Development of a Rule-Based Speech Synthesizer Module for Embedded Use," *IEICE Trans Fund Elec, Com CS*, vol. 76, no. 11, pp. 1990–1998, 1993.

[2] A. Kain and J. van Santen, "Unit-Selection Text-to-Speech Synthesis Using an Asynchronous Interpolation Model," *Proceedings of 6th ISCA Workshop on Speech Synthesis*, Aug. 2008.

[3] S. Ghaemmaghami, M. Deriche, and B. Boashash, "Comparative study of different parameters for temporal decomposition based speech coding," in *ICASSP*, 1997.

[4] B. Atal, "Efficient coding for LPC parameters by temporal decomposition," in *ICASSP*, 1983, pp. 81–84.

[5] A. Kain and J. van Santen, "Compression of acoustic inventories using asynchronous interpolation," in *IEEE Workshop on Speech Synthesis*, 2002, pp. 83–86.

[6] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1," *NASA STI/Recon Technical Report N*, vol. 93, Feb. 1993.

[7] E.T. Bell, "Exponential numbers," *Amer. Math. Monthly*, vol. 41, no. 41, pp. 1–419, 1934.