Speech Synthesis by Concatenating Maximally Fitting Phones

Vesa Siivola
43036b
Project work for S-89.155 Postgraduate Course in Audio Signal Processing, Sound synthesis

February 14, 2003

Abstract

This paper presents a method for choosing phones directly from the source waveform so that natural sounding speech can be synthesized. The method relies on finding a phone with maximally similar context with the phoneme to be synthesized from the source transcription. This allows the method to find as long passage as possible from the source text, resulting in natural sounding synthesized speech. The method tries to keep the synthesized power, speaking rate and pitch consistent.

Contents

1 Introduction

Traditionally, speech synthesizers have been constructed by teaching a few prototype phonemes (or triphones) and using signal processing techniques to transform these prototypes to required pitch and duration. In this paper, an opposite approach is taken. The synthesizer chooses a best fitting phone directly from the source data and uses the phone without any signal transformations. The chosen phone should have maximally similar context compared to the phone to be produced. It should also have consistent duration, power and pitch when compared to previously produced phonemes. With these simple criteria, a surprisingly natural sounding speech can be produced. The database for the synthesizer should be big enough and the phoneme segmentations have to be adequate to get good results.

2 Data and preprocessing

For data, a book read out loud for blind people was used. There was a single speaker, who read 13 hours of clear Finnish text. Transcription of the book was also available.

The transcription was aligned with the audio by a simple HMM based phonetic speech recognizer [Siivola, 1999]. Calculating a full Viterbi search for the 13 hour recording is impossible so the modified algorithm used simple pruning and sliding window to make the task computationally feasible. A more detailed description of an early version of the algorithm can be found in [Siivola, 2001]. The resolution of the alignment was 8 ms. The accuracy of the alignment was not verified since there was no reference alignment to compare to. For this speech synthesizer, the alignment was good enough. The transcription contained symbols for all Finnish phonemes, both long and short versions. There was also two kind of silences, long and short.

The pitch and power estimates were calculated for voiced sounds. For pitch estimation, a cepstrum based estimator from Speech Filing System software was used [Huckvale, 2000].

3 Finding the best phone

The search for the best phone takes into account the following factors:

- The context of the phoneme: give higher score when the phoneme to be synthesized has as similar left and right contexts as the phoneme found in the data set (see Figure 1).

- Adjacency in source database: if the previously chosen phone and the current phone are adjacent in the database, the current phone should fit well

sentence to be synthesized	...pa rhAn_tul oksen_takAmi...
database	...tu rhAn_tul tuAn tamperELe...

Figure 1: *The context search. The synthesizer is about to synthesize the phoneme /t/. The search has found phone /t/ in the database. For this phone, the matching left context is 5 phonemes and right context 2 phonemes (shaded).*

to synthesized speech. Increase the score for the current phone.

- Penalize the phone, if the length of the phone differs significantly from the previous phoneme of similar type (short vowel, long vowel, short consonant, long consonant, short silence, long silence).
- Penalize a voiced phone, if the pitch differs significantly from the pitch of the previous voiced phoneme.
- Penalize a voiced phone, if the power differ significantly from the power of the previous voiced phoneme.

The exact form of the score function is

$$\begin{aligned} S(x) \quad = \quad & \alpha_1 \cdot \min(lc(x), rc(x)) + \alpha_2 \cdot \max(lc(x), rc(x)) \\ & +\alpha_3 \cdot adjacent(x) + \alpha_4 \cdot adjacentvowel(x) \\ & -\alpha_5 \cdot lengthpenalty(x) - \alpha_6 \cdot pitchpenalty(x) - \alpha_7 \cdot powerpenalty(x) \end{aligned}$$

where

$lc(x)$ returns, how many preceding similar phonemes the required phoneme shares with the database phone.

$rc(x)$ returns, how many following similar phonemes the required phoneme shares with the database phone.

$adjacent(x)$ returns 1, if the current phone is adjacent to the previously synthesized phone in the source database.

$adjacentvowel(x)$ returns 1, if the current phone and the previously synthesized phone are adjacent vowels in the source database. This is done for keeping the diphthongs whole.

$lengthpenalty(x)$ finds the previous synthesized phoneme, that has the same class as x and returns the difference of their length (≥ 0).

pitchpenalty(x) returns 0 if phoneme x is not voiced. Otherwise, it finds the previous synthesized voiced phoneme and returns the difference of the pitches (≥ 0).

powerpenalty(x) returns 0 if phoneme x is not voiced. Otherwise, it finds the previous synthesized voiced phoneme and returns the logarithm of difference of the powers (≥ 0).

The weight parameters a_i were chosen by hand so that finding the longest similar context was the most important feature. The form of the score function allows efficient pruning of the search for the best phone: If the score for the current phone is lower than the best score after calculating the context scoring, the phone cannot be the best (the rest of the terms reduce the score). This allows us to skip the pitch and power calculations. This is important, if the pitch and power calculations should be done on-line.

4 Some notes on the implementation

The user interface has a simple phonetic converter, so that Finnish text can be entered at the command prompt and this will be converted to phonetic symbols used in the transcription. Any punctuation is converted to a long silence.

When the waveforms are concatenated, the end of the previous phone and start of the new phone are windowed with a slope (see Figure 2). This eliminates the discontinuities at the borders. If the transition was phase synchronized for voiced phonemes, the quality of transitions could still be improved. This is not implemented in this work.

The synthesizer was written in C++ and is about 1 350 lines (not including the speech recognizer and pitch tracker), of which 800 lines were written specifically for this project. The rest of the code was borrowed from other projects and included mostly functions for reading and writing audio files and transcriptions. The code is tested to work on Linux, Irix and Tru64 operating systems. It should compile on any UNIX-like platform.

5 Discussion

Unfortunately, an objective evaluation of the system was not performed due to lack of resources. Personally, I think that the synthesizer sounds mostly surprisingly natural, but the quality of output is not consistent. Some example samples of synthesized speech can be found at my home page [Siivola, 2003], so the reader is invited to make his own quality assessment. Phase synchronizing the phoneme boundaries would probably improve the quality slightly.

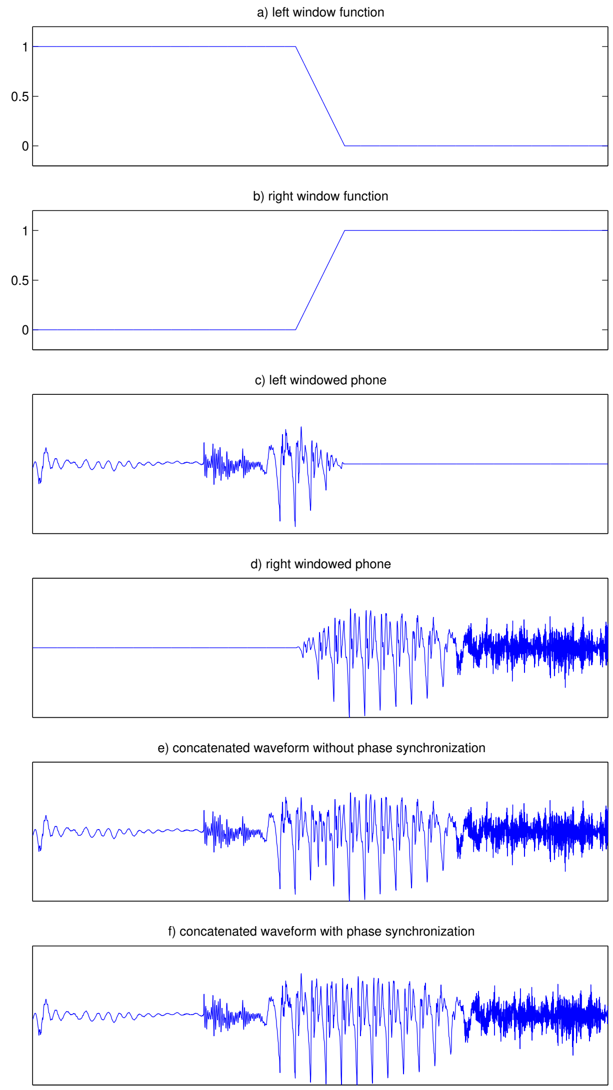

Figure 2: *a) and b) show the windowing functions at the concatenation point. c) and d) show the windowed waveforms. In e), the effects of phase mismatch can be seen. e) can be compared with f), where the phase has been synchronized.*

Generally, a speech synthesis system sounds monotonic without intricate rule-set to model the prosody of a sentence. The proposed system implicitly makes its own prosody. This is a good thing as far as no exact control of the prosody is required and some quirks are tolerated. For tighter control, the algorithm for picking the best phone should be modified.

The current system makes it relatively easy to change speaking voice: Find a new database with transcription, align the transcription, run the pitch tracker and power calculator on the waveform and you are ready to synthesize. The algorithm is fairly easy to implement and no hand tuning is required except for the α_i weight coefficients. It seems that the performance is not very sensitive to the α_i-parameters, so setting them should be easy.

6 Conclusions

A different method for speech synthesis was presented. It is easy to implement and requires almost no hand tuning. The quality of the output speech is good, if occasional quirks are tolerated. The method does prosody modeling implicitly by finding the best possible phones according to the cost function. The method does not give the user a very tight control of the synthesis parameters like prosody or rhythm and timing. For consistently well performing synthesizer, signal processing approaches for normalizing the power and pitch would be required.

References

[Huckvale, 2000] Huckvale, M. (2000). Sfs user manual. http://www.phon.ucl.ac.uk/resource/sfs/help/index.htm.

[Siivola, 1999] Siivola, V. (1999). An adaptive method to achieve speaker independence in a speech recognition system. Master's thesis, Helsinki University of Technology.

[Siivola, 2001] Siivola, V. (2001). Puheäänityksen jako foneemeihin käsikirjoituksen perusteella. http://www.cis.hut.fi/vsiivola/index.html.

[Siivola, 2003] Siivola, V. (2003). http://www.cis.hut.fi/vsiivola/syntsa. Speech synthesis samples.