Speech Synthesis Report

Eric Walker

Problem

Many people suffer from speech impediments or other causes for not being able to speak. Software today is available to allow text-to-speech generation. For example, Microsoft Office has a text-to-speech platform, and Steven Hawking uses a form of nonverbal action to speech software. I wanted to allow others to 'speak' with my voice, so I created a speech synthesis program that allowed users to type what to say and the program would 'speak' it. The input was limited to arpabet symbols to allow more focus on the speech generation to start, but then I found the CMU arpabet pronunciation guide, so anything in that dictionary can be used as input.

Corpora and Development

In order to generate speech from arpabet input, I needed to have a set of audio files for all the sounds that would need to be reproduced. To that end, I recorded myself saying each sound in the arpabet: AO, AA, IY, UW, EH, IH, UH, AH, AX, AE, EY, AY, OW, AW, OY, ER, AXR, EHR, UHR, AOR, AAR, IHR or IYR, P, B, T, D, K, G, CH, JH, F, V, TH, DH, S, Z, SH, ZH, HH, M, EM, N, EN, NG, ENG, L, EL, R, DX, NX, Y, W, and Q. To make each of the sounds appear natural, I combined them into a sentence of sorts to record: Washington father discus pleasure hair bee rude could uh-oh yay my large ear chow'em cure thanks zoo gore boy nettle winter vet. I recorded this with audacity using 16-bit single channel at 16000 Hz to make computation later easier.

In order to test the validity of the model, I would test various combinations of arpabet symbols, mostly taken from the Wikipedia page about the arpabet. In addition, I also tested the program with the above sentence, as that has all the symbols. This was still limited, as not every vowel sound has its stressed and unstressed form, such as IY1 in key and IY0 in very. To this end, I tried to test words that had both kinds of stress for each vowel sound. In development, output wave files are produced, which

can be analyzed visually by spectrograms for differences between a spoken set of words and generated words.

Methods

In order to solve this problem, I took the recorded files of my voice for each arpabet symbol and mushed them together based on user input. That is, I had a wave file for each arpabet symbol, and just appended the files to start. This resulted in some very jerky sound speech. I then looked for better methods to blend the sounds together. First, I looked at the spectrograms of the files appended and of a recording of me saying that particular combination. There was some clear smoothing in my voice that was not present in the appending. It appeared that the major frequencies moved in sine waves from one to another (see Figure 1).

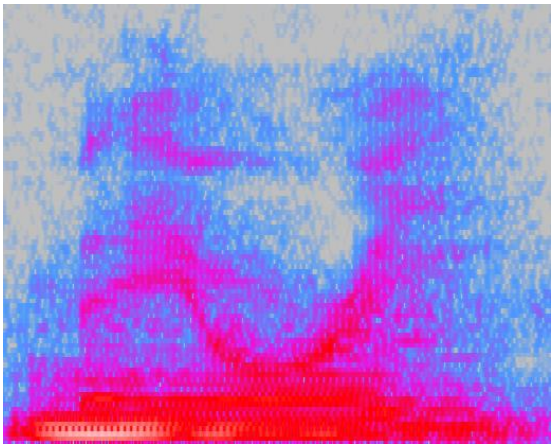

Figure 1: Spectrogram (colored, courtesy of Audacity) of me saying 'Maori'

All of these files were recorded in 16-bit single-channel 16000 Hz wave files.

I looked a lot at Fourier transforms in our previous spectrogram assignment to get an idea of how to combine these sounds together. When sounds are just squished together from the raw files, it sounds very jerky, and in the spectrogram, we can see clear lines between the different source files. My first method was linear smoothing between these: a linear line between the last data point in one file to the first data point in the second file. Although this improved continuity and quality greatly, it was very clear in the spectrogram that it was not how human speech is smoothed. To this end, I found the Fourier transform of the last window of the first file and the first window of the second file, then proceeded to generate intermediate points based on the difference between the major frequencies of each. To calculate the Fourier transform, I used numpy's fast Fourier transform method, which served us well in the spectrogram assignment. I calculated the transform for the last window (25ms or 400 frames) of the

first file, and the first window of the second file, then averaged the transforms. With this, I could take the inverse Fourier transform of to get a transition wave. I then put the middle half of this in between the two files for the final output. This makes it a little jerky, but is better than using the whole thing, as when I first took the Fourier transforms, I used the Hamming window, so the ends appeared to be zero. I didn't divide by the Hamming window because this caused even worse artifacts in the output- very noticeable 'clicks'. Also, I used Hamming instead of Hanning because it doesn't go to 0, but instead .08 on the ends, so it was possible to test this division.

Challenges

Several tasks were difficult in producing the final project. The base task of recording every arpabet sound was more difficult than expected, as sounds tended to blend together, so just getting a small snippet that was a particular sound didn't sound natural in playback, as smoothing was difficult. When attempting to record each sound separately, upon hearing the recording, it didn't sound like natural speech. Thus, stretching words in recording was important to get a nice base file for each sound. That being said, it took multiple tries to get each sound right, as well as some additional recording for emphasis changes. The process of smoothing the gap between sounds was definitely the most difficult part of this project. Getting the Fourier transform to work, as well as optimizing the parts to use was tricky, and It was hard to tell if there was improvement between small changes, such as the size of the window used.

Results

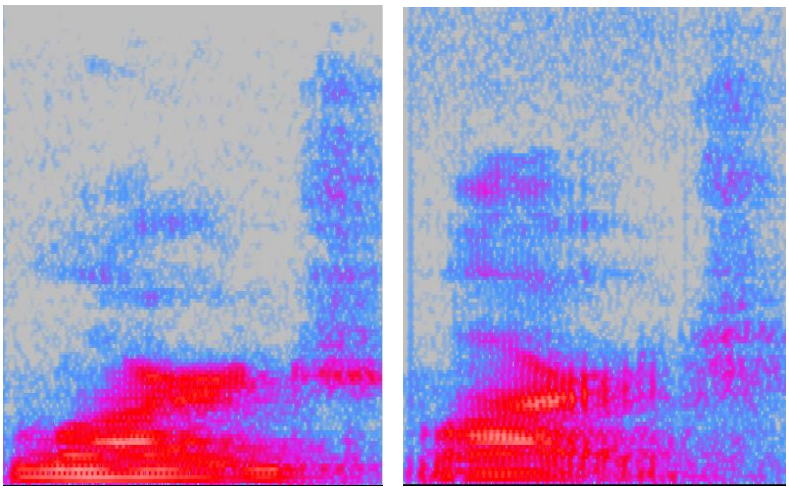

Figure 2: Spectrograms of the sounds that produce the word 'large'. On the left is a recording of me saying the whole word, and on the right is the generated sound file. Note the similarities between the structure of the 'AAR' sound, with the red line going slightly up, and the 'JH' sound of general static on the right of both spectrograms.

Exact results are difficult to come by for this sort of project, but comparisons of spectrograms comes pretty close. In generated wave files, there are vertical lines, generally of light-blue intensity (lower than pink, but higher than light blue) that are the strange artifacts that result from the Fourier transform not exactly matching up with the phase of both the 'before' and 'after' files. Upon general inspection with hearing, the sounds are generally understandable, but some sounds are very hard to replicate, such as the 'W' sound (it's very similar to the 'V' sound). Overall, this works for understandability, but could be heavily improved upon with more time.

Future Work

Though this project was fun, real utility would come out of it with the improvement of the smoothing algorithm and addition of two features: recognizing natural text outside of the CMU pronunciation guide and allowing voice generation. The smoothing algorithm could likely be improved by iterating the Fourier transform multiple times and merging it with the surrounding files. For iterating the Fourier transform, this would include finding the average, but also the ¼, ¾, 1/8, 3/8, 5/8, 7/8… weighted averages. This would allow for smoother transitions between very different windows. In terms of merging the transform with surrounding files, this would look like phasing from the first file to the inverse transform to the next file by taking weighted averaged that sloped towards the next file.

Recognizing natural text is a tricky one for English, as it depends on very complicated, and sometimes contradictory rules. This problem has some pretty good solutions out there, so this would likely involve starting with APIs that do this and trying to improve or modify them to our uses. As stated at the beginning, some people rely on this sort of technology to speak with others. In order to make that more personal, it would be important to allow them to make their own speech patterns. Whether this included taking what sounds they could make and making these into sounds or having a tool that one could try multiple sounds to see if that is the voice they identified with, allowing tweaks here and there. Communication is an essential part of human life, with a heavy emphasis on speech. Allowing more people to comfortably communicate facilitates this process, bringing people closer together.