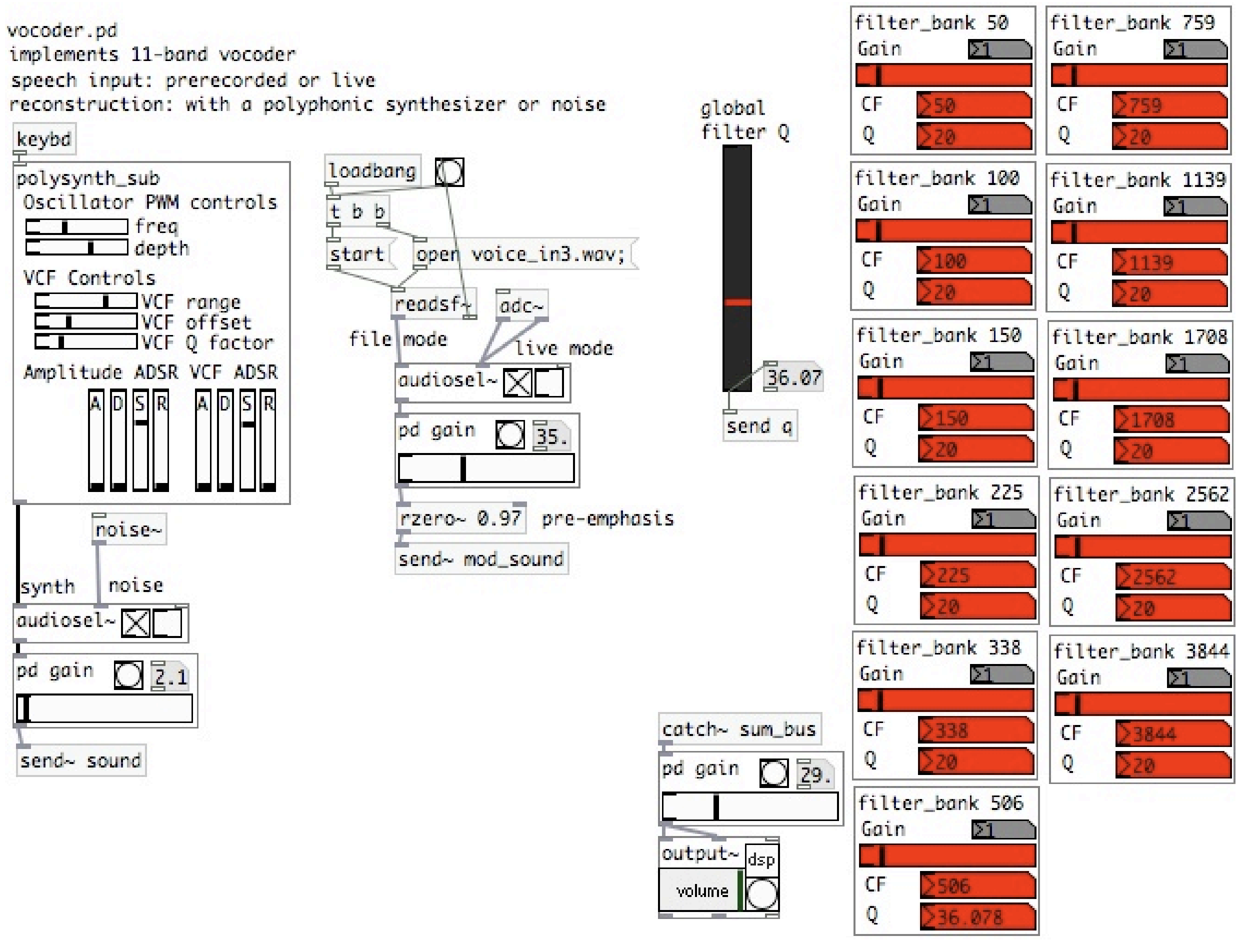
Vocoder

Jonathan Ward



```
vocoder.pd
implements 11-band vocoder
speech input: prerecorded or live
reconstruction: with a polyphonic synthesizer or noise
```

For my final project, I decided to reprise the vocoder project I created for my presentation at the start of the semester. The result of that effort was really just a first pass. It was as much an exercise in how to use Pure Data as it was to actually create a working vocoder. The vocoded speech was only semi-intelligible with the right phrases and enunciation, so there was definitely room for improvement.

The vocoder was originally invented to compress speech. Human speech occupies about 4 kHz of bandwidth. In today's world of multi-megabit per second data transmission speeds, this may not seem like a lot, but it actually is very large for the amount of bandwidth for information conveyed in speech. Humans can speak and comprehend speech at a maximum rate of about 200 words per minute (Wikipedia, *Words per minute*). Interestingly, increasing or decreasing the spoken word rate has little effect on the bandwidth occupied by speech. In contrast, the same 4 kHz of bandwidth can hold 8 kb/s of data. If 8-bit ASCII is used to encode this data into text characters, one can transmit 1200 words per minute, assuming a 5 letter word length. This 60-fold increase demonstrates that there is much room for speech compression.

Apart from being used for their original purpose of speech compression, vocoders are also used as electronic music effects to render human speech onto sounds from musical instruments. For this project, I was more interested in this role as a musical effect than as a tool for compression. To this end, I wanted to create a vocoder in Pure Data using the same techniques available to analog circuit designers to try to emulate the vocoders of the past.

An analog vocoder works by splitting a speech signal into a number of distinct bands and extracting a modulation envelope from each band. In a sense, it treats speech as a set of AM radio stations, each with a different carrier frequency. It extracts the envelope from each station—the information—and ditches the carrier. The extracted envelope from each band is then applied to an identical band of an external instrument's signal. Any frequency content in that band will be amplitude modulated by the envelope of the corresponding speech band. If many bands are used, the resulting output sounds like the instrument is producing speech.

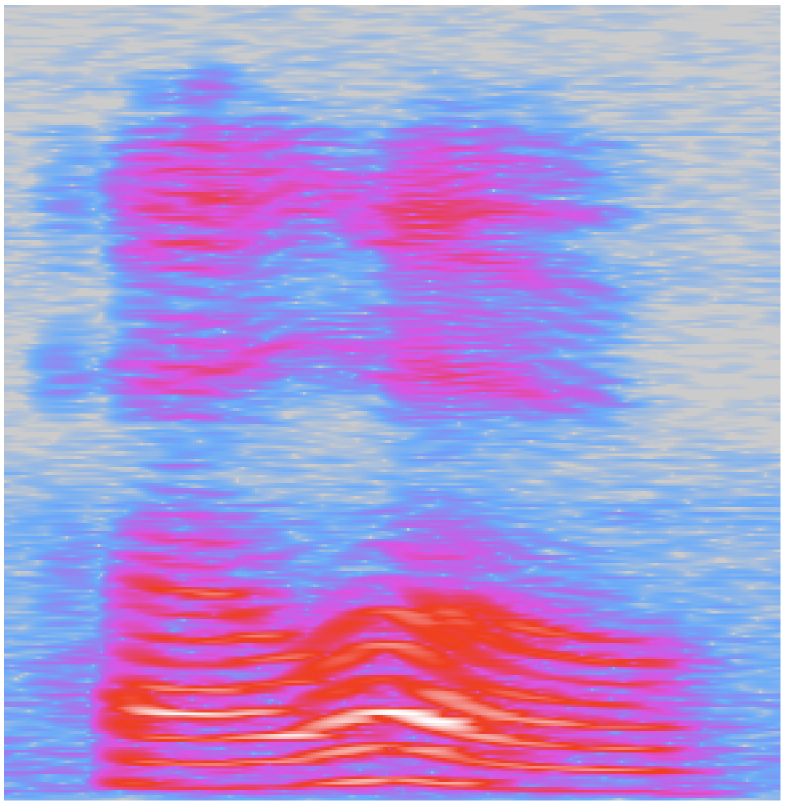
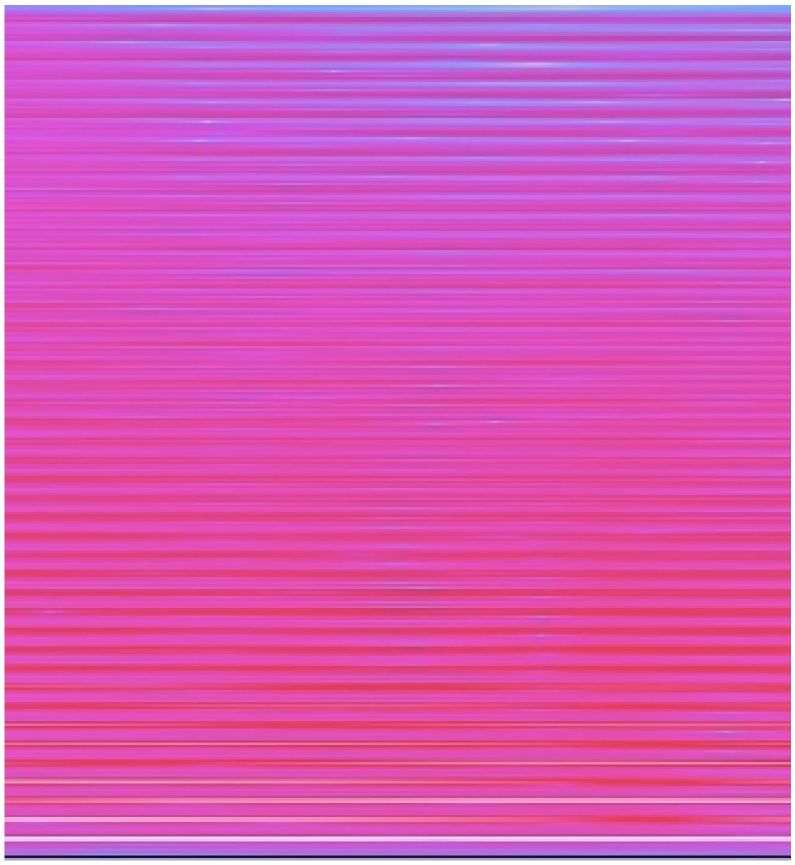
This approach makes sense if one sees a spectrogram of human speech and looks into how speech is produced. To create speech, an organ in the throat called the glottis rhythmically vibrates to create a comb of harmonics. The amplitudes of these harmonics are controlled by a filter formed from the throat and mouth. Different speech sounds are created by changing the filter envelope of the throat and mouth and shifting the fundamental frequency generated by the glottis. By separating the speech signal into many different bands and extracting each band envelope, the vocoder is in a sense extracting the filter passband of the entire throat and mouth.

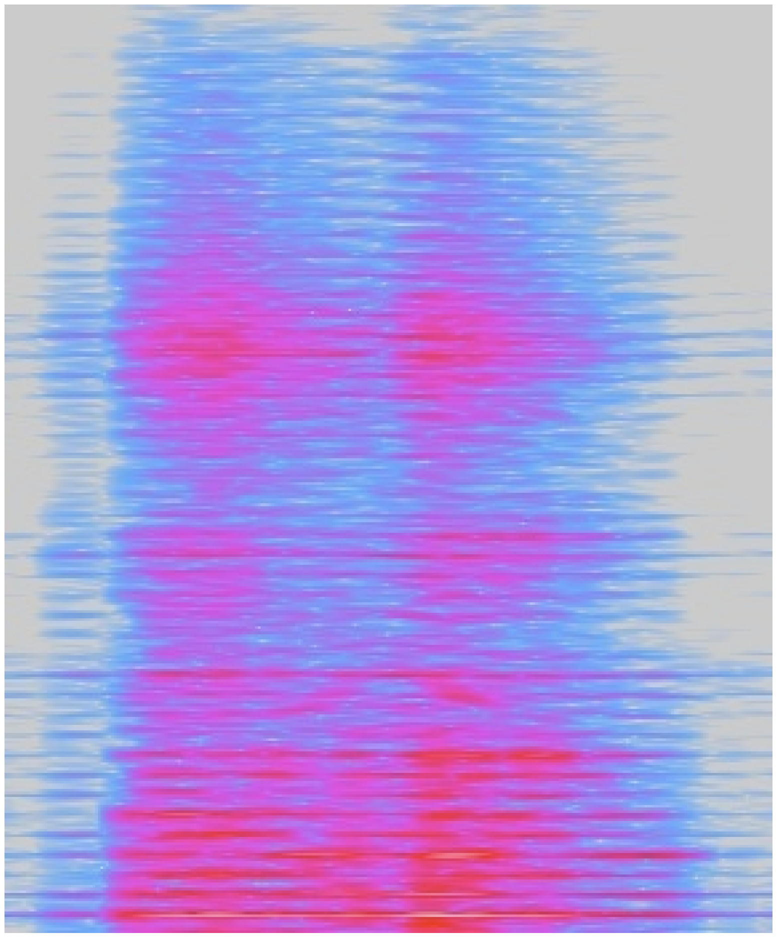
For this project, I created a Pure Data module called filter_bank that implemented the modulation envelope extraction and reconstruction functions. It extracts an envelope from a speech signal band with a configurable center frequency and Q and applies it to an identical band of a reconstructing signal. By instantiating these many of these filter banks with different center frequencies, one is able to reproduce intelligible speech. For ease of use, the filter_bank module's center frequency is set with creation arguments. In addition to being controlled with input boxes, all the filter_bank parameters can be controlled by sending control information through different control pipes. Also, instead of requiring that the user manually wire up each module, filter_bank receives sound input from send~ objects and outputs sound though a throw~ object.

In my setup, I used 11 constant Q filters, so the center frequencies of my bands increased in a logarithmic fashion from 50 Hz to 3844 Hz. This puts more emphasis on the information in the lower bands. This may or may not be the best method. An alternative approach would be to have more bands in the frequency range with the greatest speech content. While further experimentation would help, this setup seemed adequate enough. The reconstruction signal is selectable to either be random noise or the sound output of Prof. Ellis' polyphonic synthesizer.

Originally, I had intended to create a fricative generator to detect the presence of fricatives in speech. Fricatives are hard consonants that have wideband frequency content. Many vocoders have trouble representing these intelligibly. The classical way to solve this problem is to attempt to detect fricatives by looking at the wideband frequency content of the speech. If enough energy is detected across several bands, a brief pulse of random noise outputted to simulate a fricative. While random noise does not sound like an actual fricative by itself, it sounds convincing in context with the rest of the word. After some experimentation, I found that this was unnecessary. Instead, I added a small amount of pre-emphasis to exaggerate the sudden changes in amplitude that fricatives create.

A set of spectrograms appear below to visually represent the operation of the vocoder. The first image shows the spectrogram of a word. One can see the the filtering process of the throat and mouth on the glottis harmonics as well as a shift in the fundamental frequency of those harmonics. The second image shows the regularly spaced harmonics generated using a pulse width modulating synthesizer. The final image shows the vocoded result from the two inputs. The same basic spectral structure of the original word can be seen, but the harmonics are much closer together. This is the result of the closely spaced harmonics of the synthesized sound. A set of sound files is included, along with the Pure Data code, to demonstrate the effect.



References

Words per minute. From http://en.wikipedia.org/wiki/Words_per_minute Accessed on May 7, 2010.

Words per Minute vs Baud Table. From http://homepage.virgin.net/ljmayes.mal/baud/table.htm Accessed on May 7, 2010.