Speech Compression Documentation
Summer 2019 Computational Science REU
Colleen Olson
Dr. Bill Robertson, Austin Wassenberg, Alex Kaszynski
July 15th, 2019

How to Use FinalSpeechCompression.py

The purpose of this program is to decrease the amount of data necessary to be sent between "devices" while still keeping the file as coherent as it will allow. The frequency choice has the largest effect on the quality of the audio file.

Give the program a minute or two to boot up. You will see a black screen.

Inputting a .wav file:

Every input that the program takes needs to be in a specific form. It's case-sensitive, spaces will throw it off, etc.

Your .wav file needs to be recorded in mono, NOT stereo. It needs to have a sampling rate associated with it. Recording in Audacity works if you change it to mono. The sampling rate is usually 8,000 or 44,100 samples/second.

The .wav file needs to be in the same directory (folder) that the .exe is in.

If your .wav file is named Example.wav, type in **'Example.wav'**.

White Noise:

Introducing white noise into the audio file helps prevent the sounds from being too pitchy, and gives it a grumbly-like speech in an attempt to make it sound more natural. If you choose to introduce white noise, you will still have all of the frequency choices available.

A simple **'y'** or **'n'** input will do.

If you say 'y' to adding white noise, it will ask you how much sine you would like to add. Input a number, ex. '0' or '0.1'. Adding 0 sine will make it just composed of frequency-banded white noise. Adding in too much sine will defeat the purpose of introducing the white noise.

0.1 is a good default ratio.

FrameTime:

The program works by breaking the sound file up into small windows and analyzing those windows independently. A good frameTime is 0.01 seconds. Too long, such as 0.1 seconds, will produce choppy output. Too short, such as 0.005, and speech sounds will disappear.

Simply type **'0.01'**.

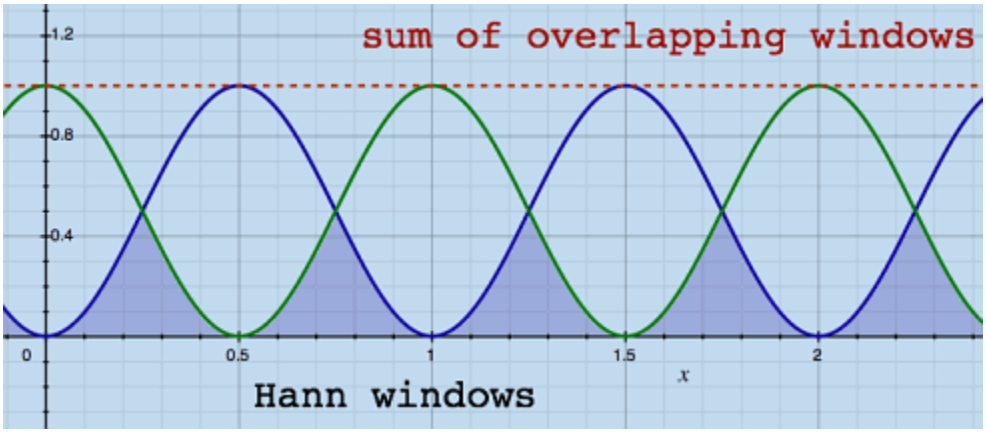
Naming your file:

The edited sound file will be saved to the same directory that the other files are saved to, and will be a .wav file also. Enter a name, such as ‘name’, and the program will automatically add the .wav suffix.

Overlapping Windows:

The purpose of having overlapping windows is to help smooth the sound, but it also doubles the amount of data, lessening the compression rate. Sometimes there is barely a quality difference between overlapping and non-overlapping windows.

Input **‘y’** or **‘n’** for overlapping windows.



http://www.katjaas.nl/pitchshift/pitchshift.html

Window Shape:

A filter is applied to each window to prevent sharp, spiky edges. The filter is usually bell curve like to decrease the amplitude at the edges. If you chose overlapping windows, the decreased edges would add back together to become relatively level again.

The traditionally accepted window is the Hann window. Type **‘Hann’**.

Other window options are ‘square’, which essentially is no filter, and ‘extended_Hann’, which is the same curve as the Hann window but will add up to 1 when overlapped. There is barely a difference between those two options.

Frequency Options:

There are many frequency options. For an explanation of the individual options, refer to Dr. Robertson’s separate documentation.

Options:

- rob_one
- rob_two
- rob_three
- vowel_formants_one
- vowel_formants_two
- evenly_spaced_ftest

- **evenly_spaced_fcent** (use this as the default option)

If you choose either of the evenly spaced frequencies, you will be asked for a starting and ending frequency as well as a number of frequencies.

Default: **100** to **4000** Hz, with **20** frequencies.

If you also chose white noise, 50-8000 Hz with 50 frequencies is better.

Saving Data:

If you want to see an accurate compression value, you'll need to save the data matrix as a binary text file. Input **'y'** or **'n'** if you would like to save it.

If yes, type a name for the file. You need to add .txt to the end. Ex. **data.txt**

The purpose of saving data to a binary text file is to compare the size of that file to the original audio. This produces an accurate percentage for compression. The data matrix is the only thing that would be sent between two devices. To compare, go into the directory and check the properties section and data size for both the original audio file and the binary file.

Check your directory for the new audio file and the binary data file if you chose that.

Interpreting the printed statements at the end of the program:

The program will print 3 things: the number of elements in the data matrix, the number of elements in the original audio file, and then a ratio of those values. This number does not necessarily accurately represent the size of the files depending on their format, simply the number of elements.

It prints 'Percentage 0.04' but it is not 0.04%, it is 4%.