Application Walkthrough and Implementation Details

Beaver is a very small .cpp file (Beaver.cpp) which utilizes several additional libraries. Once the application has started, it will notify the user that it has begun to listen for input samples. Behind the scenes, the multithreaded RtAudio library (which can be compiled on Linux, Windows or Mac OS, depending on the compiler flags) is fetching audio data via the Jack Audio Connection Kit Client.

With Beaver, RtAudio is configured to collect 512 samples per buffer, at a 44,100 Hz sample rate, with a mono channel setup, giving an input latency of 11.6 ms. 512 samples at 44,100 Hz is just enough data for MFCC extraction and, in turn, just enough for reasonable sound classification. The RtAudio callback, named “inputCallback”, is called whenever the audio buffer is full.

The Aubio library onset/attack detection algorithm checks the input buffer for any changes in overall energy, especially focusing on high frequency content. Upon registering a change, the method sets an onset detected variable to a non-zero value, meaning a sound has been detected. The buffer containing the detected onset is then passed to the MFCC and low level feature extraction methods.

Once extracted, the features are normalized and added to a feature vector, along with MFCC means and variances. Experimentation was carried out with different combinations of MFCCs and low level features. Thus far, the results are inconclusive apart from substantiating claims that low level features, in general, can sometimes serve to complement the MFCCs, but may also sometimes detract from their usefulness.

The app is hard-coded to expect 20 training samples: 10 for the first instrument and 10 for the second. Sometimes the app initially detects what may be the sound of the 'Enter' key being pressed and attempts to store it as a sample. To avoid any complications, the first sample is simply always ignored. However, if there is no erroneous sample, one extra sample must be inputted to account for the ignored first sample. Once the feature vector for each training sample is obtained, they are saved to a file called mfccs.data, which must be cleared of the temporary data every time the program is restarted.

After the 20th training sample is recorded, the application will automatically begin listening for testing data, displaying the object match type (hard-coded) if classification is successful, or displaying that there was no match, if not. Depending on the resulting object classification, the corresponding audio sample is played. The sample file is streamed from its location, which is currently hard-coded in the program, using SFML (Simple and Fast Multimedia Library).

When the user has completed the testing (or general use) phase, they may press the enter key to exit the program. At which point, the program begins freeing memory and closing files. Currently, if all 20 training samples are not recorded before ceasing program execution, a segmentation fault may be thrown.

Beaver is implemented in C++ with several C and C++ libraries with the Ubuntu Studio 14.10 low latency kernel. These include: RtAudio (Audio I/O), FLANN (Fast Library for Approximate Nearest Neighbors), Aubio (onset detection), Gist (feature extraction), SFML (Simple and Fast Multimedia Library). Also, Cadence was a key tool for configuring the Jack Client, setting sample rates and buffer sizes, and managing Pulse Audio and Alsa to avoid sound

card monopolization. All libraries except Gist and RtAudio were downloaded using Ubuntu's package manager.

The compiler command for Beaver using Jack as the audio driver is as follows:

g++ -Wall -D__UNIX_JACK__ -DUSE_FFTW -D__RTAUDIO_DEBUG__ -IGist-master/src -o beaver beaver.cpp RtAudio.cpp -ljack -lpthread -laubio -lfftw3 -lflann -lsfml-audio libasgist.a

The necessary RtAudio compiler commands and libraries for various operating systems, apart from the additional libraries and flags seen in the above compiler statement, can be found here: https://www.music.mcgill.ca/~gary/rtaudio/compiling.html

Beaver has only been tested thus far on Linux with Jack.