Real-time phase analysis plugin for improving microphone placement in multi-microphone multi-source scenarios

Christian Steinmetz Holcombe Department of Electrical and Computer Engineering, Clemson University Department of Performing Arts, Audio Technology Concentration, Clemson University

Motivation

When using multiple microphones to record or reproduce a source, the relative distance from the source to each microphone introduces a relative time delay. In a stereo configuration this delay affects how the source is localized in the stereo field, and when summed can introduce comb filtering. In the case of the acoustic drum kit, the placement of overhead microphones can negatively affect the quality of sound captured. Since it is desirable to obtain phase coherence for the kick and snare drums [1], there is a unique set of placements that result in acceptable phase coherence for both sources. The goal of this project was to develop a real-time utility to provide insight about the placement of overhead microphones to achieve better phase coherence, where previous techniques are more tedious, less precise, and require additional auditory training.

Theory

In order to provide the audio engineer with information about the relative phase coherence of sound sources from multiple microphones, the sample delay between the sources can be estimated. This was achieved using the Generalized Cross Correlation with Phase Transform (GCC-PHAT) [2]. An adaptation to the Generalized Cross Correlation (GCC), the GCC-PHAT is an established method for calculating the time delay of arrival (TDOA) between signals. This method has been shown to provide robust delay estimation for acoustic sources and is given in the frequency domain as

$$\psi_P[k] = \frac{X_1^*[k] \cdot X_2[k]}{\left| X_1^*[k] \cdot X_2[k] \right|}$$

and in the time domain as

$$\psi_P[n] = \mathscr{F}^{-1}\{\psi_P[k]\}$$

where k is the current frequency bin between 0 and N − 1, * represents the complex conjugate, $X_1[k]$ and $X_2[k]$ are $x_1[n]$ and $x_2[n]$ transformed to the frequency domain, and F^{-1} represents the Inverse Fourier Transform.

The GCC-PHAT estimates the time delay between inputs by calculating the difference in phase between the inputs in the frequency domain. Transforming the result back into the time domain then gives the delay estimation.

It is common when implementing the Discrete Fourier Transform (DFT) to apply a windowing function to the input. Based on previous research of the GCC-PHAT on musical signals, three windowing functions, Hann, Blackman-Harris, and Hamming were included, as they were shown to perform well on percussive signals [3]. The Hann window of length N where n is the current sample is given by

$$w(n) = \frac{1}{2}\left(1 - cos\left(\frac{2\pi n}{N-1}\right)\right)$$

and is applied to each input frame by multiplying each input frame by a window of the same length.

Implementation

After experimentation with the GCC-PHAT in MATLAB to investigate the effects of the Phase Transform on the GCC, windowing functions, and noise removal, the development of the real-time plugin began. The cross platform C++ JUCE framework was used to build a real-time plugin for use in Digital Audio Workstations (DAWs). The prototype GUI can be seen in Figure 1.

The GUI enables user control of the *hop size*, *frame size*, *window type*, *threshold,* and *temperature*. The first three parameters allow the user to modify how the input audio is analyzed by the GCC-PHAT so it can be optimized for different sources. The *threshold* allows the user to omit input audio that is below the set level, since in the MATLAB implementation it was found that frames of near silence, i.e at the noise floor, negatively affected the delay estimation. The *temperature* control allows the user to achieve the most accurate path length estimation by using a model for the speed of sound.

The sample delay estimation is featured prominently in the GUI window. This value is updated after the circular buffer of 4096 samples is analyzed based upon the set user parameters. The mode of the delay estimations produced for the buffer is said to be the sample delay. This has the effect of making the estimation more robust.

This sample delay value is then used to determine the latency in milliseconds as well as the path length difference between the two microphones in centimeters. An accuracy metric is also included that determines the percent of the estimations in the buffer analysis within 1 sample of the sample delay estimation. The greater number of frames that return the same sample delay as the given estimation, the greater the certainty the provided estimation is accurate.

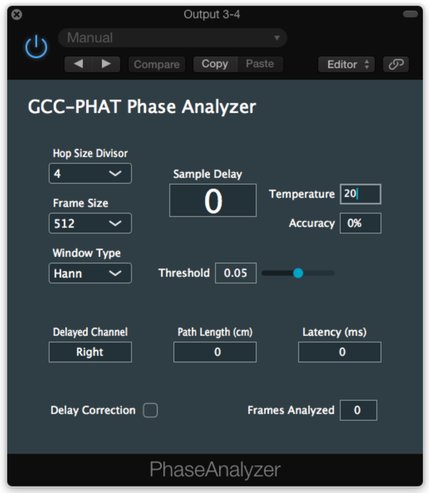

Figure 1
Graphical User Interface shown in Logic Pro X, built with the C++ JUCE framework.

Results

Since the goal of this project was to develop a system to be used by audio engineers on real sources, testing of the plugin was performed by striking acoustic snare and kick drums with overhead microphones placed above the drums in a recording studio setting. The output of the plugin GUI was observed while striking the drums, as the audio engineer would.

To perform the tests, each drum was set with a matched pair of overhead microphones placed above the drum at an equal height. One microphone was then moved upwards from this position in 1 cm intervals up to 30 cm. This setup can be seen in Figure 2. At each interval the drum was struck three times and the sample delay and path length after each strike was recorded. This process was repeated for the kick drum. The temperature of the room was recorded and entered into the plugin in order to achieve the best possible path length difference estimation.

During these tests the AudioUnit (AU) build of the plugin was run in Logic Pro X at 24 bit depth and 44.1 kHz sampling rate. This constituded worse case behavior since a higher sampling rate would provide greater granularity in delay estiamtions. The user parameters were held constant with the *hop size divisor* set to 4, the analysis *frame size* set to 1024 samples, with the Hann *window type*, and a *threshold* of 0.05. The measured *temperature* in the room was 20.2 degrees Celsius.

The distances returned by the delay estimation for each offset distance were averaged and then the actual offset values were subtracted from these estimations to produce residual values. These residuals were then plotted over the offset distance for the kick drum in Figure 3 and for the drum in Figure 4.

For both sources, the estimated path length distance was within 1 cm of the actual path length offset. This held over the 30 cm range tested, within the expected offset difference in most overhead microphone placements.

Figure 2
Two Neumann KM184s placed above the snare drum with an offset of 1 cm between the microphones.

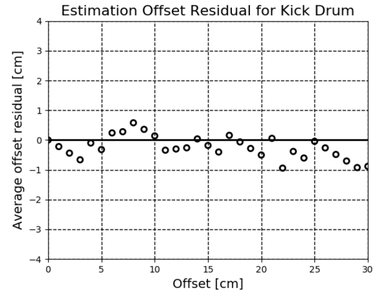

Figure 3
Average residual values over three trials for the kick drum

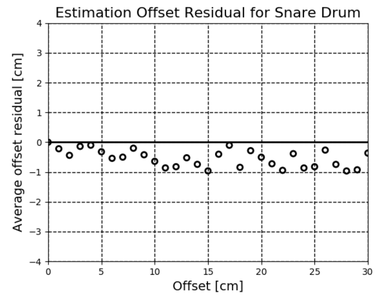

Figure 4
Average residual values over three trials for the snare drum

Summary

Ultimately, the developed real-time plugin is successful in conveying sample accurate delay estimation for a pair of overhead microphones when recording or reproducing a drum kit. By providing the user with detailed information derived from the sample delay estimation, the audio engineer can gain insight about the phase coherence for the kick and snare drums with current microphone placements, and can use this information to improve microphone placement if needed. This plugin reduces the demand placed on the audio engineer to detect phase incoherence, expedites the microphone setup process, and ultimately gives the audio engineer an extra level of assurance in regards to microphone placement before recording or reproduction begins.

References

[1] Owsinski, B., *The Recording Engineer's Handbook*, Artistpro, 2004, ISBN 1932929002.

[2] Knapp, C. and Carter, G., "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4), pp.320–327, 1976.

[3] Clifford, A. and Reiss, J., "Using Delay Estimation to Reduce Comb Filtering of Arbitrary Musical Sources," *J. Audio Eng. Soc*, 61, pp. 917–927, 2013.