Work in Progress: A Simple Plug-in Implementation of Extreme Audio Stretching

Ethan Laptew*	*Yufeng Lu (advisor)
ECE Department, Bradley University	ECE Department, Bradley University
elaptew@bradley.edu	Ylu2@bradley.edu

Abstract

Music is ubiquitous and one of the most popular multimedia formats in our lives. Significant research efforts have been made in music signal processing. It becomes challenging for music educators to understand and apply these developed techniques. This study aims to develop a simple audio plugin for extreme audio stretching for music production. An audio effect algorithm, PaulStretch, is implemented and evaluated. It could be used as an educational tool to explore music signal processing applications.

Keywords: Music Production Extreme Audio Stretching PaulStretch

I. Introduction

In sound design and music production, time stretching and audio warping has been playing an essential role in producing high-quality music tracks. There are three main methods for stretching audio. The simplest one is to change an audio clip's playback speed, which affects the tempo and pitch. Another method is time scale modification (TSM). This is used to speed up or slow down audio without affecting the frequency content. J. Driedger and M. Muller reviewed fundamental TSM methods in [1]. In general, the procedure of TSM includes three processing pipelines: signal decomposition, frame relocation adaption, and signal reconstruction. Both parametric and non-parametric methods are applied to analyze and decompose signals [2-6]. A repetitive window breaks audio into monophonic and polyphonic, where sinusoids are repeated to match the desired length. Monophonic favors time resolution, while Polyphonic favors frequency resolution. The problem with this technique is that you get long stagnant tones that jolt suddenly to new frequencies. Even visually, audio programs struggle to display stretched audio. This method is fine for stretching audio a little to moderate amount. Finally, the third method, audio slicing, cuts the signal up into small frames with blank or empty spaces in between. This has an unpleasant result as jittery, broken, and unpleasant tones are obtained. Significant efforts are also in synchronizing and smoothing signals [7 – 10].

All of these are just a small portion of research efforts in music signal processing. There is an application gap between music signal processing and music education. It becomes

challenging for music educators to understand and apply these techniques. As such, many audio plugins have been developed so that users can play and try out different techniques interactively. PaulStretch is an audio effect algorithm, that is useful for pads, textures, choirs, and orchestras. It is a simple plugin wonderful for beginners to comprehend. The screenshot of PaulXStretch open-source plugin by Sonosaures [11] is shown in Figure 1 below. It is very complicated both mechanically and visually. RealStretch is another example. It is meant for live performance, much like a guitar pedal, and works by pushing the play head back every few seconds [12].

Figure 1. Control Panel of PaulXStretch Plugin [11]

The goal of this study is to create a simple audio plugin (.VST/.Component) that can stretch audio to extreme lengths, without changing pitch. Specifically, an audio effect algorithm, the PaulStretch algorithm, is implemented. it aims to develop a simple plug-in implementation of extreme audio stretching, PaulStetech.Q, for music production, not live performance. Simplicity and multi-platform support are the major features to be sought after.

II. The PaulStretch Algorithm

PaulStretch is an audio effect algorithm created by Paul Nasca for extreme stretching in 2006. PaulStretch has two main parameters. The first is Stretch Factor - How much the audio will be stretched. This has no limit and may process until the computer runs out of memory or battery. The second is Time Resolution - This breaks up the sample into windows, which affects the frequency and the time resolution of the resulting sounds. Smaller windows have better time resolution while larger windows have better frequency resolution.

The mathematical process of PaulStretch is described as follows: Based on user input parameters such as stretch factor and time resolution, an original audio file (.wav or .mp3) is progressively stretched through windowing data into overlapped segments. Then the Fast Fourier Transform (FFT) is applied to each windowed frame. The original phase information is discarded, then randomized phase information is inserted to create enjoyable textures, instead of stagnant tones. The inverse FFT of each window is computed and added back together, overlapping but spaced out how the user specified.

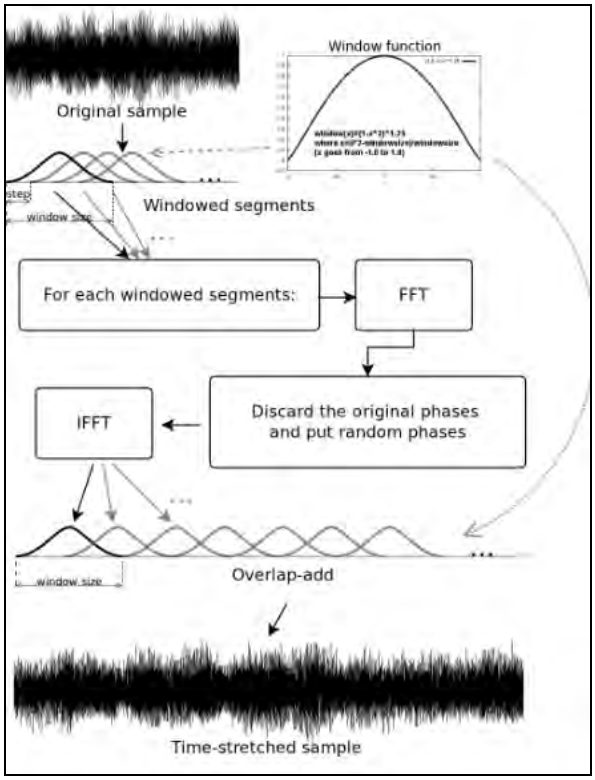

Figure 2. Paulstretch algorithm [13]

III. Progress and Results

Since MATLAB is one of the standard programming languages used in engineering programs, it is chosen as the first platform to implement the plugin. MATLAB app designer can be used to create professional apps through an easy graphical user interface and an integrated editor. The developed MATLAB app is shown in Figure 3.

MATLAB app designer eases up the programming-heavy area of app development for those who do not have professional software development experience. However, the user interface within MATLAB app designer is very restrictive. Pre-defined codes for buttons and class declarations are not fully accessible to be edited. Passing information between buttons and back-end operations becomes extremely challenging. In addition, a professional license for MATLAB compiler is needed. Without this license, one can only share the project file, not export it as a standalone application. The drawback of coding capabilities in the app developer, along with the compiler is not fully available. It decreases the availability of the plugin to students and consumers. Therefore, a popular cross-platform game engine by Unity Technologies is evaluated.

Figure 3. PaulStretch MATLAB GUI

Unity has been used to create a wide variety of software including video games, medical apps, virtual reality experiences, mobile apps, and more. The developed Unity app is shown in Figure 4.

Figure 4. PaulStretch Unity GUI

The challenges with Unity derives from its limited support of manipulating user input audio files. Editing audio files is not straightforward for programmers. Developers may spend more time working with user-defined packages and filer converters. One of the most important design features is to test the algorithm for music production with simplicity and easiness.

As a result, JUCE, an open-source cross-function application framework is adopted as the platform for the plugin implementation. It allows users to create audio plugins with a VST framework. From the framework, it generates a C++ project in Xcode. Figure 5 shows a proposed graphic user interface (GUI) of PaulStretch developed using Aseprite. The developed PaulStretch plugin (PaulStretch.Q) supports multiple platforms. As a demonstration, Logic Pro, an audio workstation and MIDI sequencer software application from Apple Inc. is used to test the plugin. Figure 6 shows the PaulStretch JUCE GUI in Logic Pro X.

Figure 5. PaulStretch Aseprite GUI

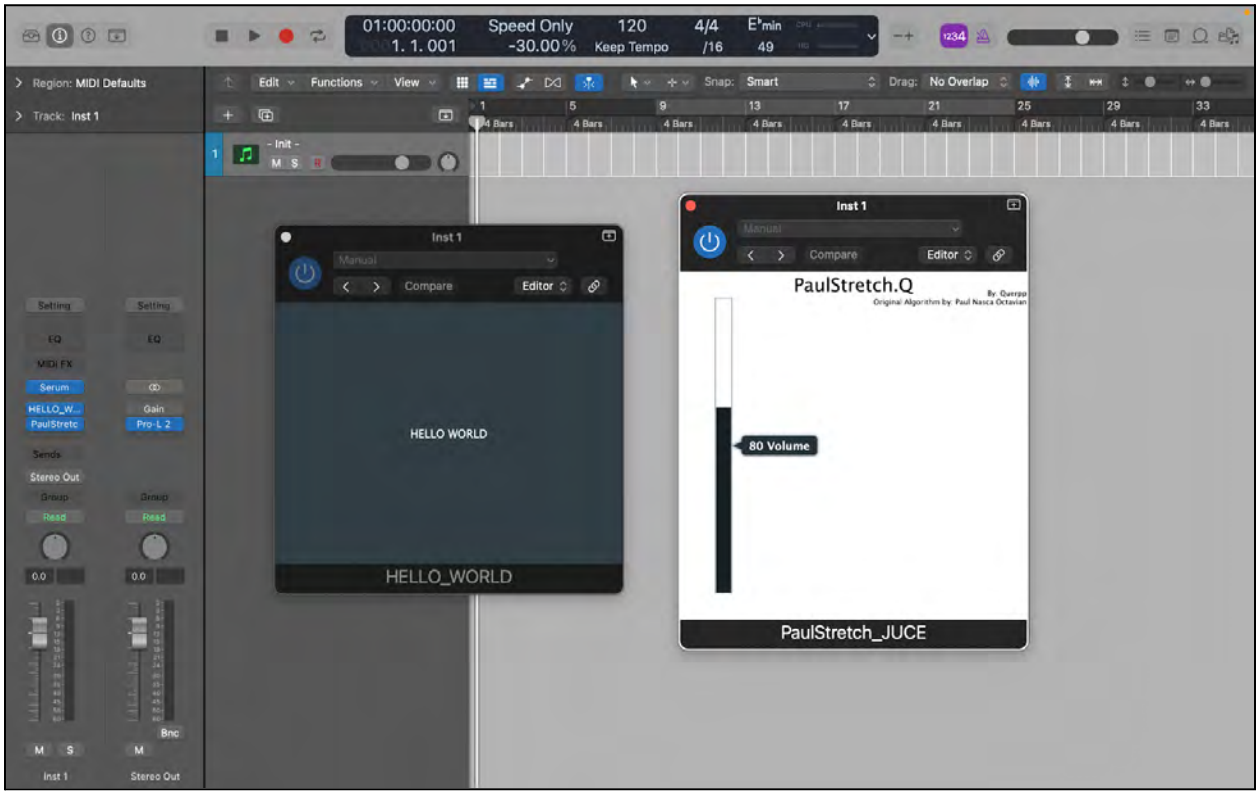

Figure 6. PaulStretch JUCE GUI in Logic Pro X (12.2023)

As a summary, Table 1 compares the language, application, and file types of different platforms for the audio plugin implementation.

Table 1. Comparative Table of development platforms.

	MATLAB	Unity	JUCE / C++
Language	MatLab Script	C#	C++
Application	Engineering Simulations	Video Games and Software	Audio Plugins
Files	Project File	Stand Alone App	VST/Component

IV. Conclusion

In this project, a simple plug-in implementation of extreme audio stretching, PaulStretch.Q, is proposed. The PaulStretch algorithm is implemented on JUCE, an open-source cross-function platform in C++. It could be an educational tool for music production or sound design. Engineering students may use it to explore their interest in audio signal processing applications as well.

References

[1] Jonathan Driedger, and Meinard Muller, “A review of time scale modification of music signals”, Applied Science, vol. 6, Issue 2, 2016.

[2] Pierre Hanna, Myriam Desainte Catherine, “ Time scale modification of noises using a spectral and statistical model”, the proceedings of IEEE Acoustics, Speech and Signal Processing, 2003.

[3] Scott N Levine, and Julius O Smith, “A compact and malleable sines+transients+noise model for sound”, Modern acoustic and signal processing, Springer, New York, 2007.

[4] Leonardo Fierro, Vesa Valimaki, “Sitrano: A Matlab App for Sines-Transients-Noise Decomposition of Audio Signals”, *the proceedings of 24th International Conference on Digital Audio Effects (DAFx)*, pp.73-80, 2021.

[5] E. Moulines and J. Laroche, “Non-parametric techniques for pitch-scale and time-scale modification of speech,” Speech Communication, vol. 16, no. 2, pp. 175–205, 1995.

[6] J. B. Allen and L. R. Rabiner, “A unified approach to short-time Fourier analysis and synthesis”, the proceedings of the IEEE, vol. 65, no. 11, pp. 1558–1564, 1977.

[7] T. Roberts and K. K. Paliwal, “Time-Scale Modification Using Fuzzy Epoch-

Synchronous Overlap-Add (FESOLA) ”, the proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics *(WASPAA)*, pp. 31-34, New Paltz, NY, USA, 2019. doi: 10.1109/WASPAA.2019.8937258.

[8] E. Moulines and F. Charpentier, “Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones”, Speech Communication, vol. 9, no. 5, pp. 453 – 467, 1990.

[9] J. Allen, “Short-term spectral analysis, synthesis, and modification by discrete Fourier transform,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 25, no. 3, pp. 235–238, 1977.

[10] S. Roucos and A. Wilgus, “High quality time-scale modification for speech,” in Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 10, April 1985, pp. 493–496

[11] PaulXStretch, https://sonosaurus.com/paulxstretch/, Sonosaurus Inc.

[12] RealStretch, https://www.mathworks.com/matlabcentral/fileexchange/79487-realstretch

[13] Paul Nasca, Paulstretch algorithm, http://www.paulnasca.com/algorithms-created-by-me#TOC-PaulStretch-extreme-sound-stretching-algorithm