Phase Vocoder Web Application using WebCL, HTML 5 and Javascript

Colter McQuay
University of Victoria
Victoria, B.C.
cjam@uvic.ca

Abstract

This project will explore the capabilities and possibilities of creating web applications for Music Information Retrieval tasks using newly created web standards such as HTML 5 and Web CL. The advantage of creating web applications is that they are accessible, extremely robust, platform independent and easy to maintain. Since application upgrades involve updating code on the server, clients don't have to wait to update their software; they are always using the latest version of the application. In this project I will be exploring the aforementioned technology by creating a web version of a Phase Vocoder using HTML 5, JavaScript and Web CL. I will attempt to create both a CPU based and GPU based version of the Phase Vocoder to allow for performance comparisons between the two implementations.

Phase Vocoders are extremely useful tools in audio signal processing. They allow for fast and efficient time-frequency processing of audio signals by taking the Fast Fourier Transform (FFT) of windowed segments of the audio signal. Many audio effects can be implemented using the Phase Vocoder by modifying the phase or frequency data associated with the windowed segments of the audio signal. This paper describes the implementation and functionality of a Phase Vocoder written using Matlab. The Phase Vocoder itself will be thoroughly explained as well as some of the effects that were achieved with it.[1]

Keywords: GPU, Phase Vocoder, MIR, Web Applications.

Introduction & Motivation

The purpose of this project is to develop a cross platform, cross browser web based Phase Vocoder. The application will be designed using open source libraries and new standards such as HTML 5, Web CL and JavaScript to create a Phase Vocoder which will offload its processing tasks to a capable processor to increase performance. Creating the Phase Vocoder as a web application, will provide the power of the Phase Vocoder concept to anyone with a capable web browser and an internet connection.

HTML 5

The world is becoming an increasingly connected place. More and more content is being put onto the Internet and into "the cloud", allowing access to it from anywhere over the Internet. These improvements to the Internet infrastructure have been accompanied by huge advances in processor technology. Computers are becoming more powerful and more connected than ever before. In order to keep up with these advances, developers are creating applications that are more powerful, more robust and more capable than ever. Despite all of the advances in technology, the web application layer has been limited by the original HTML 4.01 specification which was published December 24, 1999 [1]. Therefore, the web application developers have been stuck trying to develop robust and interactive web applications using standards developed for computers that we used 12 years ago. There are plug-ins such as flash that have been created to bridge this gap; however, some devices support these plug-ins and some devices don't. This makes development of these cross-platform applications difficult from the device support perspective. HTML 5 is the new HTML specification being developed to bridge the gap between the technology that we are and will be using, and the capabilities of the web applications that can be developed. The following excerpt is the abstract from the HTML 5 working draft [2].

[1] The author of this paper assumes that the audience has basic knowledge of the Fast Fourier Transform and other basic signal processing operations and terminology as well as some understanding about web application development and the technologies used within this domain.

"This specification defines the 5th major revision of the core language of the World Wide Web: the Hypertext Mark-up Language (HTML). In this version, new features are introduced to help Web application authors, new elements are introduced based on research into prevailing authoring practices, and special attention has been given to defining clear conformance criteria for user agents in an effort to improve interoperability."

OpenCL & Web CL

In today's technological research and development cycle, developers and engineers are locked in a push-pull relationship. As engineers create new technology and hardware that is more capable, developers create applications that demand more capable hardware and technology. Multi-core processors are now standard in every modern computer and mobile devices and have improved performance drastically; however, there are still limitations to these processors. In recent years, advances have been made to the dedicated Graphics Processing Units (GPU) to expose the refined computational hardware available on these cards. The power of these GPU's have been made available to the programmer through application programming interfaces (API) provided by the GPU manufacturers.

OpenCL, which stands for Open Computing Language, is a recently created open standard for facilitating cross-platform parallel computations on modern processing units (including but not restricted to GPU's). OpenCL has been adopted by many computer chip manufacturers and can drastically improve the performance of applications by offloading certain computations to more refined hardware which can execute these computations in parallel. This standard spawned the Web CL standard which was created to define JavaScript bindings to the OpenCL standard. This standard will allow developers to create web applications which can harness the power of OpenCL and the performance increase associated with running computations on processing units that are refined at a hardware level.

Time Stretching with Phase Vocoder

The Phase Vocoder (see Figure 1 for block diagram) is an extremely interesting audio processing tool that manipulates audio by taking overlapping, windowed blocks of the signal (see Figure 2) and calculating the frequency-domain representation. The frequency-domain data (i.e. Phase and Magnitude) of the signal is then manipulated to produce a desired effect (i.e. setting phase to zero would make a sample of a person talking sound like a robot). The frequency-domain data is then used to synthesize a new audio signal by converting each block of frequency-domain data back into the time-domain. These new blocks of time-domain signal are subsequently overlapped and added back together to create the new signal. One of the amazing effects that the Phase Vocoder can produce is time stretching audio while maintaining the pitch of the audio. This is done by changing the spacing (hop size) of the time-domain blocks in the synthesis stage relative to the analysis stage to effectively stretch or compress the new audio signal. The amount that the clip is stretched depends on the original analysis block, and hop sizes versus the synthesis block, and hop sizes. No additional processing is necessary in between the analysis and synthesis stage in order to time-stretch an audio signal.

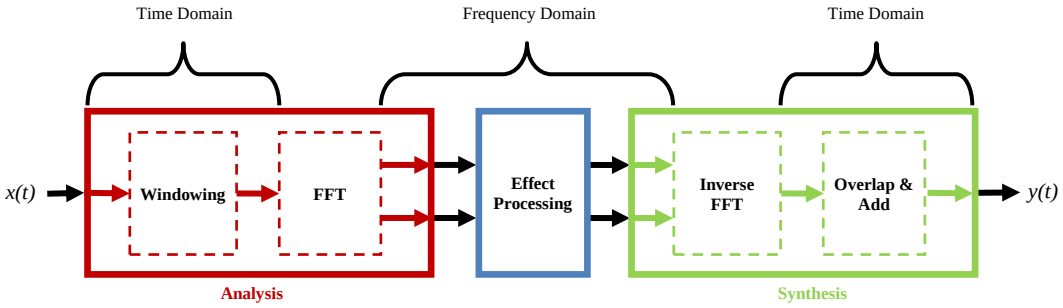

Figure 1 - Phase Vocoder Design

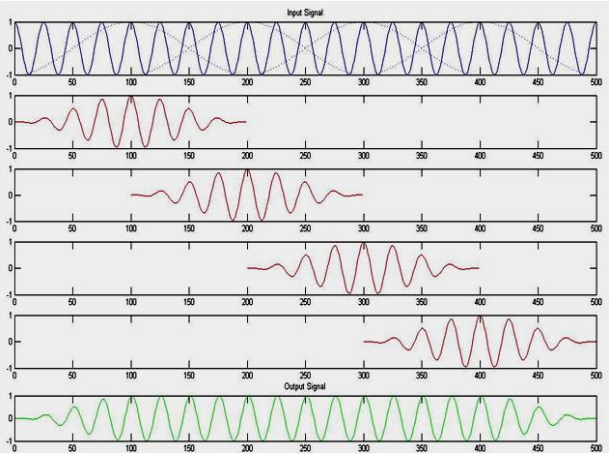

Figure 2 - Phase Vocoder windowing process

Related Work

I have past research and experience related to this project that I feel will help me immensely in fulfilling the specifications described in the Problem Formulation section below. In a previous class, I have implemented a Phase Vocoder (FFT to IFFT) and many of the effects that it is capable of producing in Matlab. This project gave me a solid understanding and basis for both implementing the Phase Vocoder and understanding where tasks can be offloaded onto the GPU.

I have also done three semesters of research in the area of Computational Electromagnetics. During this time I was in charge of writing and designing software which would run electromagnetic simulations on the GPU using Open CL. This experience will provide me knowledge necessary for porting the current serial implementation of the Phase Vocoder into a parallel implementation to be run on the GPU.

Problem Formulation

I would like to use the Phase Vocoder concept to provide a web based application that will allow users to time-stretch an audio clip to match the tempo of another audio clip. I would like to ideally mimic the time warping functional provided in Ableton Live [3] within the web browser. Ideally, my goal is to provide two implementations of the Phase Vocoder. The first implementation will use strictly JavaScript within the browser to provide support for users that do not have Web CL capabilities. The second implementation will use JavaScript in conjunction with Web CL to provide a higher performance version of the same web application.

Description

To complete this project I plan on using the newly supported HTML 5 standards, JavaScript, Web CL in conjunction with Mozilla's Audio Data API [4]. I will be developing this application strictly for Mozilla Firefox since the Web CL extension created by Nokia [5] is only available for Firefox. In addition to using these standards and technologies, I will also be using some JavaScript libraries such as JQuery and Mozilla's Audio Data API as well as using some Fast Fourier Transform implementations written for JavaScript [6] and OpenCL [7] for this project.

Time-Line

The timeline for this project is as follows:

1. Develop a simple application to display and visualize the waveform and frequency domain information of a particular audio file. *(October 18th)*
2. Develop a JavaScript-based Phase Vocoder implementation which will allow time stretching of an entire audio clip by a certain factor. *(October 24th)*
3. Implement and test a Web CL-based FFT algorithm. *(October 28th)*
4. Develop a Web CL-based Phase Vocoder implementation which will provide the same functionality as the implementation in step 1 above. *(November 4th)*
5. Develop functionality for both implementations that will allow the audio clip to be stretched by different factors throughout different parts of the clip. The new functionality should also allow for different block sizes to be specified for each part of the clip to be stretched. *(November 10th)*
6. Develop a user interface similar to that of Ableton Live's [3], which will allow the user to see the clip's wave form and allow the user to insert markers which will dictate where and how the audio clip will be stretched and modified. *(November 15th)*

Key Results

The following are the key results that I would like to achieve with this project.

1. Functional web based Phase Vocoder with both a JavaScript and Web CL implementation to allow support for all client browsers.
2. Phase Vocoder implementation that will allow time-stretching capabilities to match songs with various tempos with each other. (Similar to Ableton Live's warping functionality [3])
3. Both versions of the application should provide the same results given the same signal and same input parameters.

Summary

This project will by no means be a trivial task. Mozilla's Audio Data API is still in the experimental stages and is by no means a standard in the industry. Even if the final project is to the level described in the Key Results section, this implementation is based on Mozilla's browser specific implementation of an Audio Data API which is not yet standardized and thus will limit the functioning of this application to Mozilla's Firefox. The issue of interoperability also rears itself in the use of Web CL which is currently only available as an extension of Firefox. The project; however, will be a proof of concept of what may be possible in web browsers as the HTML5 and Web CL standards mature in modern browsers.

References

[1] World Wide Web Consortium. (1999, December) World Wide Web Consortium. [Online]. http://www.w3.org/TR/html401/

[2] World Wide Web Consortium. (2011, May) HTML5. [Online]. http://www.w3.org/TR/html5/

[3] Ableton. (2011) Ableton - The Fastest Way To Warp A Track In Ableton Live. [Online]. http://www.ableton.com/pages/the_bridge/tour/lesson/the_fastest_way_to_warp_a_track

[4] David Humphrey et al. (2011, October) Audio Data API - MozillaWiki. [Online]. https://wiki.mozilla.org/Audio_Data_API

[5] Nokia Research. (2011, October) WebCL. [Online]. http://webcl.nokiaresearch.com/index.html

[6] Mozilla Wiki. (2010, May) Audio Data API JS Library - MozillaWiki. [Online]. https://wiki.mozilla.org/Audio_Data_API_JS_Library

[7] Eric BainVille. (2011, June) BEALTO. [Online]. http://www.bealto.com/