Spectrograph COP4520

Rami Jurdi
Zachary Noble
Gregory Freitas
Jayden Bendezu

***Abstract*—This research is in the Digtial Signal Processing field. Where our aims are to improve the performance of realtime signal processing using parallel processing techniques coupled with 1 dimensional Fast Fourier Transforms. It has been done before by other researchers implementing multi-dimensional Fast Fourier Transforms in a multithreadeed context. The purpose of our research is to gain a better understanding of parallel processing techniques and digital signal processing. Thus the main goal is to observe the outcome of implementing the multithreaded Fast Fourier transform algorithm and learn from the state-of-the-art research.**

I. Target Problem

The main problem at hand is to create an application that is able to process signals provided to the program as audio files such as .WAV, .MP3, and .MP4. Then displaying the audio files as a spectrograph of its signals. This problem can be broken into several steps of what we need to acheive.

- A GUI to interact with.
- Accepting audio files via local upload or recording.
- Processing the audio files.
- Processing the signals for audio files.
- Displaying the signals of the audio files on a spectrograph.

A. Approach

Tackling some of the above sub-problems. In order to create a GUI to interact with, we will write the program using the QT C++ GUI framework to simplify taking audio files from disk and decoding the raw data. Through the use of provided libraries of QT the basic functionality of loading files and playing/decoding audio is handled and we use them on a higher level. This will allow us to focus more on the problem of processing signals for the audio files. Then by handling extracting raw data from the audio files using the QAudioDecoder class provided by QT, we intend to use one-dimensional Fast Fourier Transforms to process the signals. (How do we plan to parllelize the FFT is what would be discussed here briefly since will be explained througouly within the algorithms subsection).

B. Plan Outline

(Temporary to keep this in)

1) Setup programming enviornment using Qt C++.
2) Add support using Qt multimedia API's to load and decode audio files from disk.
3) Stream audio data to an output device such as speakers or headphones.
4) Display a waveform of the audio.
5) Implement single-threaded Fast Fourier Transforms.
6) Implement multi-threaded Fast Fourier Transforms.
7) Display the frequency vs. amplitude data calculated from the Fourier transform.
8) Conducting experimental tests comparing multi-threaded implementation vs sequential implementation. Observing any performance boosts if any.

C. Algorithm

Will speak about what type of algorithm we intend to use to parallelize it. Discuss it in depth and even evaluate expected Big O runtime of our implementation.

D. Experimental Results

Here I imagine we can create some test data sets using tables.

Data Set 1			
Algorithm	File Type	File Size	Computation Time
Sequential FFS	foo.WAV	100 kb	N
Parallel FFS	foo.WAV	100 kb	N

Data Set 2			
Algorithm	File Type	File Size	Computation Time
Sequential FFS	foo.WAV	100 kb	N
Parallel FFS	foo.WAV	100 kb	N

Data Set 3			
Algorithm	File Type	File Size	Computation Time
Sequential FFS	foo.WAV	100 kb	N
Parallel FFS	foo.WAV	100 kb	N

II. State-of-the-Art Research

III. Related Work

IV. Our Contributions