AN INTERACTIVE SIGNAL PROCESSING EDUCATIONAL SOFTWARE MODULE FOR OUTREACH PROGRAM

M. Powers, A. K. Ziarani, and D. M. McNamara

Department of Electrical and Computer Engineering
Clarkson University
Potsdam, NY 13699, USA

ABSTRACT

This work presents a freely downloadable software module designed for the outreach program conducted at Clarkson University. The purpose of the module is to introduce students in grades 5-8 to the field of signal processing. The primary goal of this program is to generate interest in the signal processing field. The second purpose of the outreach program is to have undergraduates design, program, and utilize the developed program in workshops with elementary and junior high students students. There are three main components of the software: use of a spectrogram, application of several filters, and modification of the pitch and timescale of the sample. In any of the components, the user can record a five second sample from a microphone attached to the computer. The user can then use the selected component to modify the signal, and listen to how the sound changed through an attached speaker. Each component has a short introduction and instructions on its use. The program can be set up in the classroom at a workstation where students can interact with it.

Index Terms— Signal Processing, Education

1. INTRODUCTION

Computers have become a part of every day life. As people have more interaction with computers their use and possibilities expand into new territory. It seems that each generation is continually becoming more proficient with computers and are using them in a wide array of educational endeavors. Many educational programs have already been successfully implemented, ranging from the memorization and mastering of vocabulary words to the drilling of basic mathematical skills. This software is no different. With this in mind, the goal was to create a simple to use program to show some basic signal processing.

Others have created such program for outreach such as seen in [3] and [2] which targeted high school students. Our goal was to target an even younger crowd by making a simple to use ”fun” program that conveys some basic signal processing. Our secondary goal was to give undergraduate students experience in implementing some of what they learn in their signal processing classes into this program. In addition to professional experience gained by this, they also gained the experience of presenting and interfacing with the surrounding community.

When the program is initiated, a main screen opens giving the user the option to navigate to three different main components: Spectrogram, Filter Zone, and Warping Place. In each component, the user can record a five second sample from an external microphone. The user can then play the signal back, analyze the signal (in the case of the spectrogram), modify the signal with several filters (Filter Zone), or transform the pitch or timescale of the signal (Warping Place). Each component acts as a separate program; a new sound must be recorded upon entering, and is discarded upon exiting. In the components Filter Zone and Warping Place, many changes can be made to the signal, with the most recent change having the ability to be undone. At any time, the signal can be played back on external speakers. Through manipulating the audio signal and listening to the effects, the user can get an idea of what each tool does and its use. It is hoped that the software will spark an interest in the students which will encourage further study of signal processing or electrical engineering.

The software introduced in this paper is publicly available on the website of the Signal Processing Laboratory at Clarkson University (www.clarkson.edu/∼spl/outreach.html) for free distribution to the interested audience.

2. THE MATLAB-BASED GUI PROGRAM

The software model was developed in MATLAB, with help from the “guide” function. Upon execution, a main screen opens, welcoming the user with four places to navigate: the spectrogram, filter zone, warping place, and to exit. Beside each of the three component choice buttons are descriptions informing the user of what each component entails, and an example of something that could be done with it. See Fig. 1

This work was supported by U.S. NSF Award BES-0447705 and NIH Award 1R15DC007385-01A1.

for the layout of the main screen window.

2.1. Recording and Playing Samples

All three components share the common method of recording and playback of the audio sample. To begin, the user clicks on the "Record New Sound" button. If a previous sound had been recorded, a warning box would appear asking the user if they wished to discard the old signal and record a new one. A message box then comes up to let the user know that recording has been started for five seconds. After five seconds pass, the box auto-closes and a second message box pops up to inform the user to wait while the data is being stored.

At this point, the user can click on the "Play Sound" button in order to listen to what exactly has been recorded. The user also has other options, depending on the component choice of Spectrogram, Filter Zone, or Warping Place, sections 2.2, 2.3, 2.4.

To record, the MATLAB function "record" is used to create an audio recorder object. The audio data is then stored as a vector using "getaudiodata". The sampling rate is also stored, which is 44.1 kHz by default but can be modified in Warping Place.

Fig. 1. The Main Screen to the program.

2.2. The Spectrogram Component

Upon opening, the spectrogram component greets the user with an opening message,

> Welcome to Spectrogram! A spectrogram uses colors to represent the different frequencies over time of a sound sample. Time is on the X-axis, and frequency is on the Y-axis. The magnitude is represented by different colors, with red being low, and purple high. You can essentially see how the pitch of your voice changes over time. As a challenge, see if you can make a stripe going up over time. How would you make it go down?

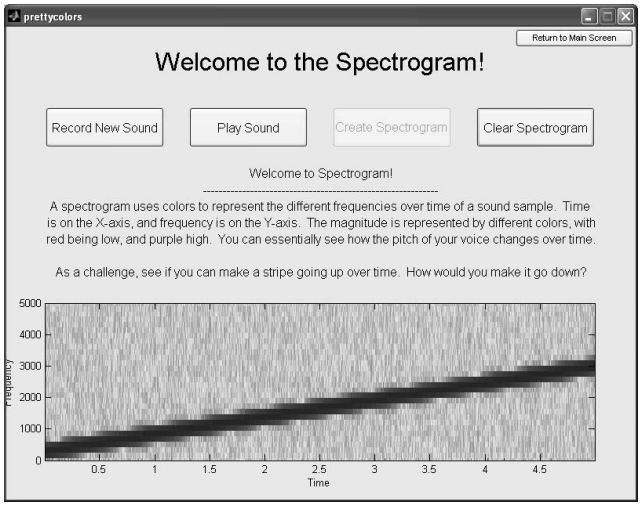

Fig. 2. Spectrogram component. The plot is the solution to the first challenge.

The greeting both gives a basic description of what the fundamentals of the spectrogram, but also gives the user two challenges. There is a simplification that a spectrogram uses colors, whereas it could be displayed in any number of ways; using colors is just simplest to describe and display. The solution to the challenge is simply to emulate the basic chirp signal with your voice. See Fig. 2 for both an example of the solution to the initial challenge and the layout of the Spectrogram component window. To reverse the slope of the stripe, simply emulate a chirp of decreasing frequency.

After the user has recorded a sample (section 2.1), the sound can either be played back, or the spectrogram can be created. Once the spectrogram is created, the user can either re-record a sample, play the sample, clear the graph, or exit to the main screen. The old signal is discarded upon re-recording or exiting.

The spectrogram is computed using the MATLAB function "specgram". The signal and sampling frequency are passed into the function, while the NFFT, WINDOW, and NOVERLAP variables of the function are all left to default. For most short samples of a human voice, the spectrogram with these settings will provide a clear view of what frequency components are present at what times.

2.3. Filter Zone

Upon opening, Filter Zone greets the user with a welcome message and a brief description of filters, "designed to pass through some frequencies, while muffling others." The work-

Fig. 3. Filter Zone component. A bandpass filter with a center frequency of 380 Hz has been selected.

shop has the option of implementing the four main types of filters: lowpass, highpass, bandpass, and bandstop. Selection of which type of filter to use is made in the "filter type" box on the left side of the window. When each type of filter is selected, the description on the right side of the window changes to inform the user on what type of filter is selected, what effect that filter will have on the signal, and what that filter could be used for. There is a slider at the bottom of the window to select the cutoff or center frequency of the filter. The slider is on a log-scale range from 100 to 10000 Hz, defaulting to 1000 Hz when the workshop is initiated. See Fig. 3 for the layout of the Filter Zone window.

After the user has recorded a sample (section 2.1), the sound can either be played back, or if a filter has been selected, the filter can be applied. If a filter has already been applied, the user can also undo the most recently applied filter. Many filters can be applied to the same sample. After a filter has been applied or undone, the user is informed at the completion, and is prompted to play the new sound.

The filters are applied using the function "f_firideal"; part of the FDSP toolbox [**?**]. This toolbox is available for download at (http://people.clarkson.edu/∼schillin/fdsp/). All filters are on the order of 800 with a Hamming window type. In the case of the bandpass and bandstop filters, the bandwidth of the filter is determined as 80% of the center frequency.

2.4. Warping Place

Upon opening, Warping Place greets the user with a welcome message and tells the user "here you can warp the pitch or speed of your voice, or even warp both simultaneously." The workshop has the option of implementing three types of warping: pitch and time warp, pitch warp, and time warp. Selection of which type of warp to use is made in the "warping type" box on the left side of the window. When each type of warp is selected, the description on the right side of the window changes to inform the user on what type of warp is selected, what effect that warp will have on the signal, and an example of how you could use that warp. There is a slider at the bottom of the window to select the pitch/time factor. The slider is on a log-scale range from 0.5 to 2, defaulting to 1 when the workshop is initiated. See Fig. 4 for the layout of the Warping Place window.

Fig. 4. Warping Zone component. A pitch/time warp with a factor of 0.68 has been selected.

After the user has recorded a sample (section 2.1), the sound can either be played back, or if a warp has been selected, the warp can be applied. If a warp has already been applied, the user can also undo the most recently applied warp. Many warps can be applied to the same sample. After a warp has been applied or undone, the user is informed at the completion, and is prompted to play the new sound.

The three warps all work differently. The pitch/time warp is the simplest, merely scaling the sampling frequency accordingly.

The time warp is somewhat more complex. The signal is first divided into 500 samples. Depending on if the time factor is below or above 1.00, each sample is either extended or compressed. In the case of the extension, the new sample is formed by the adding a piece of the old sample onto the end of itself. In the case of the compression, the new sample is formed by truncating some of the old sample. These 500 samples are then combined into a new signal.

The pitch warp is the most complex, first implementing the pitch/time warp algorithm to change both pitch and time rate, and then implementing the time warp algorithm to revert the time rate back, leaving only the pitch shifted.

3. RESULTS

This software module was presented to students (fourth and fifth graders) at the Boys & Girls Club in Massena, NY, Fig. 5. A brief presentation was given, and then the students were allowed to experiment with their own phrases. The hands on experience was able to keep their attention while they learned about signal processing. Overall, the students were very excited to interact with the software. The outreach program also worked with local high school science clubs. In addition to experiments with the developed software, a more in depth talk was given focusing on engineering specifically on electrical engineering and signal processing as a career path.

Fig. 5. Presentation of software to students.

4. CONCLUSIONS

Further work can be done to improve the software. The first being to compile this program so that it is stand alone and does not need MATLAB^{TM} to run. A save/load option could be implemented to allow the instructor to load pre-made sounds to fit the lesson plan, such as a chirp signal. A convert option could be written to allow the user to load a sound in .wav format. A shortcoming of the time warp algorithm is that no window is used when the 500 samples are compressed or extended. This allows for discontinuities in the signal which can be heard as a clicking sound. With an improved time warp algorithm, this could be avoided, which would improve the algorithm and also the pitch warp algorithm, which uses the time warp.

5. REFERENCES

[1] R. J. Schilling and S. L. Harris, *Fundamentals of Digital Signal Processing Using MATLAB,* Toronto, Canada: Thomson, 2005.

[2] M. G. Morrow, T. B. Welch, and C. H. G. Wright, "An inexpensive software tool for teaching real-time DSP," *Proceedings of the 1st IEEE DSP in Education Workshop*, Hunt, Texas, October 15-18, 2000.

[3] A. Spanias, T. Thrassyvoulou, C. Panayiotou, and C. Song, "Using J-DSP to Introduce Communications and Multimedia Technologies to High Schools," *33rd ASEE/IEEE Frontiers in Education Conference (FIE-03)*, Boulder, Colorado, November 5-8, 2003.