Project report
Sonotone

Julian Hattinguais - Germain Lecorps - Manuel Voutat

April 5, 2016

Contents

1 Introduction

Being partially deaf costs a lot of money. Once people have been diagnosed, they are given a hearing aid which has to be set up regularly by a professional and that is what costs a lot of money.
The tool we developed is made to be a viable alternative to those regular settings via a simple application that enables patients to set up their hearing aid themselves.
This tool should come as the cheap solution for patients that cannot afford the price of a professional on a regular basis.

2 Specifications

The goal of this project was to develop a software that allows a user to to set his hearing aid up by himself using his audiogram.

Seeing those conditions, we have to able to provide different features.

- Choose the frequencies to be modified
- Process real time audio
- Apply the needed filters in real time

The software will provide two different modes to be used.

- Basic mode : Choose the frequencies to modify and the gain to apply depending on the audiogram
- Advanced mode : Possibility to modify every frequencies and to choose the filter that will be applied to set up the hearing aid as pleased

The software has to be easy to use as it will be mostly used by senior who are not familiar with computers and software. Moreover, the goal is to provide an alternative to the cost of seeing a professional to set up the hearing aid. It is then important to be able to set up the hearing aid without the help of an IT specialist.

3 Technologies and tools used

The project was develop entirely in python using different tools and modules developed for python.

3.1 NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- *a powerful N-dimensional array object*
- *sophisticated (broadcasting) functions*
- *tools for integrating C/C++ and Fortran code*
- *useful linear algebra, Fourier transform, and random number capabilities*

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Source : http://www.numpy.org/

From NumPy, we have imported several mathematical functions to be able to create our own filters such as *low_pass*, *high_pass* or *peaking* filters.
As stated before, NumPy also provides a powerful N-dimensional array object which comes to be very helpful to implement Z-Transforms and Fourier Transforms.

3.2 SciPy

SciPy (pronounced “Sigh Pie”) is an open source Python library used by scientists, analysts, and engineers doing scientific computing and technical computing.
SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Source : https://en.wikipedia.org/wiki/SciPy

SciPy provided us with many useful already implemented filters. We used those filters (such as *lfilter*) to create our own filters and doing so, process the signal for our hearing aid device.
Scipy allows us to use directly implemented Fast Fourier Transform as well as many others scientific functions that we used to process audio files and live audio.
We also used a SciPy module allowing us to read and use a wave file. We needed that step to be able to do live audio processing afterward.

3.3 Matplotlib

matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code.

Source : http://matplotlib.org/

These few lines make it quite easy to understand what is the purpose of matplotlib. We used it to plot the result of our filters and to make sure what we were doing was actually accurate.

3.4 Pyaudio

PyAudio provides Python bindings for PortAudio v19, the cross-platform audio I/O library. With PyAudio, you can easily use Python to play and record audio streams on a variety of platforms (e.g., GNU/Linux, Microsoft Windows, and Mac OS X).

Source : https://pypi.python.org/pypi/PyAudio

To capture live audio a do live audio processing, we had to use pyaudio. We encountered a few problems with that module but we will come back on those later on.

3.5 Tkinter

Tkinter is the default graphic interface given by python. We decided to use it because it was providing everything we needed to make our design with simple tools.

4 Work done

There are three main modules operational.

4.1 Filters

Using NumPy and SciPy, we have implemented 8 different filters enabling us to adjust an audio signal as pleased.

- *low-pass* : A low-pass filter is a filter that passes signals with a frequency lower than a certain cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency
- *high-pass* : A high-pass filter is a filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency
- *band-pass* : A band-pass filter is a filter that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range
- *Notch* : Opposite of a band-filter
- *all-pass* : An all-pass filter is a signal processing filter that passes all frequencies equally in gain, but changes the phase relationship among various frequencies
- *peaking* : band-pass filter that amplifies the frequencies inside the band
- *low-shelf* : low-pass filter that enables to cut or amplify the kept frequencies
- *high-shelf* : high-pass filter that enables to cut or amplify the kept frequencies

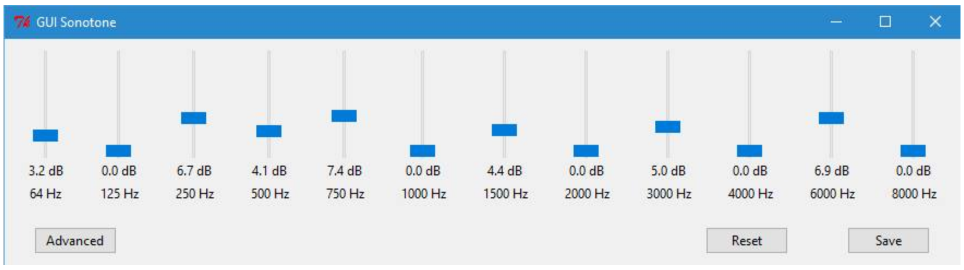
4.2 Graphic equalizer

To test those filters, we have created a graphic equalizer that works with wave files. It basically is an equalizer in which we modify the wanted frequencies from an audio file with the wanted filters. The equalizer applies those filters on the audio and creates a modified wave file.
We can visualize both spectrum afterward and compare them to be sure that the filters do the correct things.

4.3 Live equalizer

The final goal of the project was to create a live equalizer, which is actually going to be used inside the hearing aid device.
We used the work we had previously done on the graphic equalizer to get

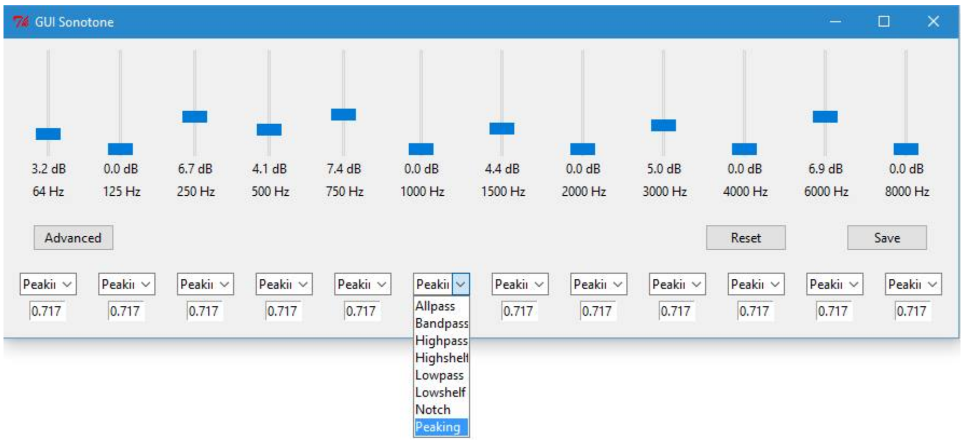
our live equalizer working. It parses an xml configuration that has been set previously by the user depending (or not) of his audiogram. Once the configuration is loaded, the software applies the needed frequencies to the signal and the output can be heard live by the user.



3.2 dB	0.0 dB	6.7 dB	4.1 dB	7.4 dB	0.0 dB	4.4 dB	0.0 dB	5.0 dB	0.0 dB	6.9 dB	0.0 dB
64 Hz	125 Hz	250 Hz	500 Hz	750 Hz	1000 Hz	1500 Hz	2000 Hz	3000 Hz	4000 Hz	6000 Hz	8000 Hz

Figure 1: Basic mode

It is only possible to choose the frequencies to increase with the basic mode. A peaking filter will be applied.



3.2 dB	0.0 dB	6.7 dB	4.1 dB	7.4 dB	0.0 dB	4.4 dB	0.0 dB	5.0 dB	0.0 dB	6.9 dB	0.0 dB
64 Hz	125 Hz	250 Hz	500 Hz	750 Hz	1000 Hz	1500 Hz	2000 Hz	3000 Hz	4000 Hz	6000 Hz	8000 Hz
Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii	Peakii
0.717	0.717	0.717	0.717	0.717		0.717	0.717	0.717	0.717	0.717	0.717

Figure 2: Advanced mode

With the advanced mode, users can not only choose the frequencies to modify, but also the type of filter to be applied and the width of the filter.

5 Evolution

In the future, we thought of a few things that could be either added or improved.

1. The user interface could be improved in the future. A live overview of the filters applied could be added to improve the user experience. With such a feature, the user would be able to visualize how the signal is actually modified by the filters he decided to apply.

2. Every filters could be improved in the future. As for now, our filters are not as good as professional filters. They are efficient enough for our live equalizer but we still observe saturation and interference on some frequencies. That could be fixed by creating some new filters to apply after our current filters and hence, it is a possible evolution.

3. A feature that apply the correct filters directly from an audiogram could be added as well. The raw audiogram would be given to the software and the correct filters would be applied automatically to create the "perfect" correction without having to touch yourself the frequencies and filters.

6 Problems encountered

1. Signal processing theory : That probably was our main issue during this project. To implement our filters and apply them properly on different signal to obtain a coherent result, we needed a good understanding of signal processing.
   As we were following the signal processing course at the same time of the project, we did not have the needed knowledge as fast as we wanted to. That held us back for a long part and once we had all the needed elements, the project started to develop much faster.

2. In the initial requirements of the project, we were to use Kivy for the user interface. We took some time to try to understand how it worked but it happened to be too complicated to use for what we needed. Therefore, we decided to use Tkinter instead which is easier to get familiar with and good enough for what we planned to do for the user interface.

3. We faced some serious problems with the sound acquisition from a microphone for the live equalizer. To digitize the data from the microphone, we use a callback function from pyaudio. That function digitize the data on a C basis (as a char string) and has to be converted in integer to be processed in python. Once those data have been processed,

it has to be converted back to C data in order to get an output signal. It took some time to understand how the data had to laid out in order to get a correct output signal.

4. When sound is captured with the microphone, there are interference and saturation independent to the filters we apply to the signal. However, we are not quite sure of where the problem is coming from. It could be coming from the microphone, from the python modules or even from our filters. Unfortunately, we have not been able to determine the source of the problem and it remains unsolved for the moment.

7 Conclusion

We chose this project out of all the others because it was an innovative project that could led to an actual prototype that would be used for patients. Even though we encountered a few issue during the development of the project, we managed to steer the project toward the expected outcome that was an functional software to be implemented in hearing aid devices. This project is upgradable as some features could be improved or added.
In the end, during this project, we have acquired a good overall knowledge of signal processing and we learnt python that was a new language for us.