Project 1
Audio Processing in Time Domain

Instructor: Charbel Azzi
SYDE 252 - Linear Systems and Signals
Fall 2022
University of Waterloo

Due: 11:59 PM, Friday, November 11, 2022

Learning Outcomes

You will follow the instructions, described in this document, and will perform some investigations. You will apply the knowledge you will acquire in this course to analyse and enhance audio signals. The objectives of the first project is to: 1) apply digital filtering systems using Matlab to shape the properties of an input audio signal, and 2) devise algorithms to analyse an audio signal. This is an easy and fun project but do not expect that everything you need will be given to you in the lectures or the textbook. This project was streamlined to cover graduate attributes Knowledge Base, Problem Analysis, Investigation and Design to name a few.

Project Description

You will be provided with some audio signals (sound files). You will process and analyse each one of those signals in three phases:

1) Read the sound file as the input signal to the system. This as a pre-processing step to help you setup for the next two phases.

2) Process the input sound signal to design a noise free output sound signal. This will be done through a low pass filtering system which is applied to the input signal to remove the noise resulting an noise-free output sound signal. The output sound signal will be different from the input sound signal.

3) Apply the signal processing concepts we have learned in this course to analyse the input signal by creating simple algorithms that can help us extract relevant information from the input signal (e.g. count the number of drum beats, detect silent regions, etc).

This document assumes that you will use Matlab, so all instructions are given in Matlab Syntax. However, you could use other languages if you wish.

Part 1: Preparations

You are provided with the following 3 sound clips.

- Birds.wav
- Speech.wav
- Drum.wav

Create a Matlab code that will set you up to solve the major two phases on this project:

1) Create a script or function to read **each** one of these sound files into Matlab and find their sampling rate. (*Hint:* You can read the Matlab help pages/documentation about the function `audioread1`.)

2) Use size to check whether the input sound is stereo (2 channels/columns) or mono (1 channel/column). Use if: If stereo, add the two columns to make it single channel (or a 1-column array)

3) Play each sound in Matlab

4) Write each of the sounds to a new file

5) Plot each of the sound waveform of one of your sound files as a function of the sample number

6) Play with the sampling rate of each of the input signals. Determine the original sampling rate, then downsample or upsample it to 16KHz (`resample` function).

Suggestion: You could put all the above steps in a function and then apply the same processing routine to several sound files by calling your function on each signal separately.

Part 2: Filtering

Design three low pass filters and low pass each of the 3 sound clips with the appropriate filtering settings. The three filters to be used are:

- Mean filter or moving average filter, essentially take an average around the input signal $x[n]$ of interest and output the averaged signal $y[n]$. It is defined as the following difference equation:

$$y[n] = \frac{1}{L}\sum_{k=0}^{L-1} x[n-k], \tag{1}$$

  where L is the window size. For example a mean filter of window size 3 is called a 3-point mean filter or average filter $y[n] = \frac{1}{3}(x[n] + x[n-1] + x[n-2])$ and calculates the average over the current and previous two samples of the input signal. It keeps sliding over all the samples of the input.

- Weighted average filter, give higher weight or importance to some samples over the others of the input signal $x[n]$ of interest and output weighted averaged signal $y[n]$. It is defined as:

$$y[n] = \sum_{k=0}^{L-1} b_k x[n-k], \tag{2}$$

where L is the window size and b_k is the filter weight of every sample. For example a weighted average filter of window size 3 is $y[n] = \frac{1}{4}x[n] + \frac{1}{2}x[n-1] + \frac{1}{4}x[n-2])$ where the weights $b_k = [b_0, b_1, b_2] = \frac{[1,2,1]}{4}$. The total sum of the weights should be equal to 1. Note that the sample $x[n-1]$ was given higher importance then the other two samples. The choice of the weight coefficient is usually made using statistical distributions such as the normal or Gaussian distribution. **For this project you are only asked to use a Gaussian filter as the weighted average filter**.

- Median filter which replaces the value of a sample by the median value of the filter. It can be written as:

$$y[n] = median(x[n], x[n-1], x[n-2], ..., x[n-k]), \text{for k = 0, 1, 2, ..., L} \tag{3}$$

where L is the window size. For example a median filter of window size 3 is called a 3-point median filter $y[n] = median(x[n] + x[n-1] + x[n-2])$ and takes the median value of the current and previous two samples of the input signal and replaces the current sample value with that median. It keeps sliding over all the samples of the input.

Hint: Matlab has built in functions for filtering such as `filter` or `conv` that you can use to easily create a Mean and Median filter after choosing the window size and setting up the filter. For a Gaussian filter, you will need to additionally use the Gaussian/normal distribution (many built in functions in Matlab such as `gausswin` to create set up the input to the `filter` function (do not forget to average it out).

The role of low pass filtering is to remove noise. You are asked to do apply the 3 filters described above on each of the given 3 sound clips to perform the following tasks:

1) Design and implement a Mean, Gaussian, and Median filter in Matlab (or any other language) as described above. You need to explain your methodology in the report (i.e. the math and concepts used and NOT the code). Use the equations I provided to define your designed filter. For the Gaussian filter, you will need to explain what is the appropriate weight equation b_k.

2) Apply each filter on every one of the given sound clips.

3) Tune the window size of the various filter and determine the best one which was sufficient for removing noise. Tabulate your results.

4) Which filter and window size worked the best? Provide a full analysis.

Do not apply these on the 3 input signals. Answer the following two tasks as general questions:

1) Describe the moving average filter as an appropriate impulse response function in the discrete and continuous domain.

2) Using Eq.(1) how would you define a high pass filter? provide only one mathematical definition and explain what it does in one example as I described each filter above.

Part 3: Analyzing Signals

In the time domain, devise three techniques/algorithms to do the following tasks:

1) Determine the number of syllables in the speech clip.

2) Determine the beats per minute in the drum clip. When you play a drum, there is a sound of clicks that corresponds to the beat (called metronome). This can usually vary from 40 to 210 beats per minute. Beats per minute estimation can be complex and requires software to detect it accurately. I only expect you to come out with a small and easy algorithm that only focuses on the drum hits as the beats or any combination you want. There is no single solution here just make sure you explain your algorithm and determine its advantages and limitations while keeping it simple.

3) Detect the silent regions in the birds clip. You need to show at least one plot where your algorithm can detect these regions inside the clip.

Hint: Think about detecting peaks in a signal, and using one of the filters you designed above to start with.

Deliverables & Grading Scheme

You are asked to tackle the above phases and their tasks and submit a technical report with all your findings. The grading scheme and associated marking is indicated in the following list:

1) Part 1 [10pts]: the mandatory deliverables are 1) Your Matlab file (.m file) (or in other languages), 2) include the plots of step 5 in your results section of the report (check report template and full deliverables on LEARN) , and 3) your `.wav` files from step 6.

2) Part 2 [35 marks]: the mandatory deliverables are 1) Your Matlab files (.m file) (or in other languages), 2) include all the plots in your results section of the report (check report template and full deliverables on LEARN) and make sure to show your results by properly labelled annotated graphs and supporting texts and tables (for example for different window sizes), 3) your methodology and concepts/math used should go in the Methodology section. Your report should tackle all the above tasks (you can create

subsections if needed). Correctness, all tasks tackled and referenced with graphs and references.

3) Part 3 [30 marks]: the mandatory deliverables are 1) Your Matlab files (.m file) (or in other languages), 2) include all the plots in your results section of the report (check report template and full deliverables on LEARN) and make sure to show your results by properly labelled annotated graphs and supporting texts, 3) your algorithms used should go in the Methodology section. Your report should tackle all the above tasks (you can create subsections if needed). Correctness, all tasks tackled and referenced with graphs and references.

4) Matlab code [10 marks]: the code should be commented and outputs the signals for each phase and task that you report. Code must be included in the appendix along with a soft copy of the original code running.

5) Technical Report [10 marks]: introduction, methodology, results and analysis, conclusion and recommendation.

6) Graphs and tables [5 marks]: properly labelled, self communicating, well annotated and clearly communicate findings and are supported by a description in the text.

Please submit **one zipped file** that includes your final report, the generated sound clips, and the code as all your Matlab files.