Intelligent Signal Processing Coursework for midterm

Introduction

The midterm coursework for Intelligent Signal Processing consists of a series of four individual exercises. These exercises cover the first five topics of the course:

- Digitising, representing, and storing audio signals
- Editing and processing digital audio
- Frequency domain representations
- Extracting features from audio signals
- Speech recognition.

The exercises are strongly based on the subjects covered during the course, but also invite the student for further investigation.

It is recommended that the students carefully read all the sections of this document, both to ensure a good understanding of the coursework exercises, in addition to knowing what to submit.

Exercise 1

Description

The goal of this exercise is to create a web-based audio application using p5.js and its library p5.sound that processes a pre-recorded sound file, sending the processed audio signal to the computer's speakers or audio output. Optionally, the user could also record the processed audio signal as a digital audio file on the computer's drive.

The application should include the following effects: *low-pass filter*, *waveshaper distortion*, *dynamic compressor*, *reverb* and *master volume*.

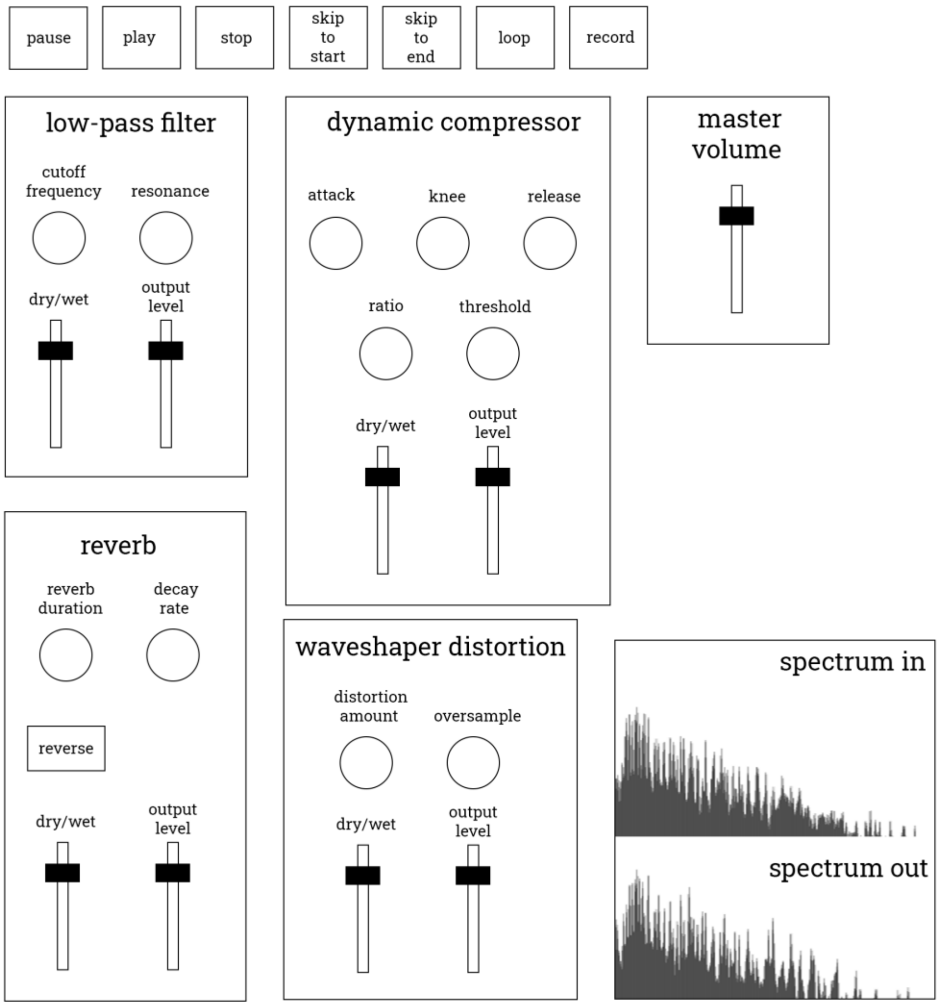

Figure 1. Schema of the GUI of the application. [Replace the pic. Add my Template]

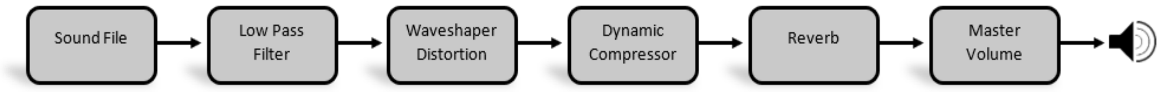

Figure 2. Internal signal flow of the application.

Regarding the pre-recorded sound file, you should record in Audacity these two lines from the poem *If—* by Rudyard Kipling:

If you can dream – and not make dreams your master;
If you can think – and not make thoughts your aim;

The audio must be recorded at an optimal recording level without clipping.

The recording must also be edited in Audacity, in order to remove possible silences at the beginning and end of the file. Finally, the recording must be normalised and saved as a WAV format, at 48 kHz and 24-bits.

The functionality of the application should meet the following requirements:

1. The application should include the playback controls and effects controls shown in Image 1.
2. Internally, the effects must be connected in a chain, as shown in Image 2.
3. The application should include a Record button that allows the user to start/stop recording the processed signal as a WAV file.
4. The application must display both the spectrum of the original sound and the spectrum of the processed sound.

Ideas for further development:

1. Enhance the filter effect by adding a *type selector* that allows the user to select between a *low-pass*, *high-pass* or *band-pass* filter.
2. Allow the user to select between the live microphone input and the pre-recorded audio file as the audio source for the application.
3. Configure a *delay* audio effect and add this to the audio chain before the *dynamic compressor*.

List of deliverables

For Exercise 1, you should submit in a ZIP file:

- [A Jupyter notebook] The source code of the application *exercise 1*.
- A screencast recording demonstrating that the application meets all the requirements and shows implementation of the further developments (maximum length of two minutes).
- A link to the application running in a web page using the Coursera static web page function.
- A written report in PDF format, approximately 500 words. This report must include:
  - A brief description of the processes of audio recording, editing, processing and saving in Audacity. This section must include at least two screenshots of Audacity showing both the original recorded voice, and the recorded voice after editing and normalising it.
  - A brief description of the main characteristics of each effect and how they have been programmed.
  - A brief analysis of the application discussing how the *low-pass filter* and the *master volume* effects affect the sound's spectrum. This should also be illustrated through screenshots.
  - A brief description of the further development implemented.

Marking criteria

	Done?	Marks
The screencast recording has a maximum length of two minutes, and it demonstrates that the application meets all the requirements and shows the further developments implemented.		1
The sound file has been satisfactorily recorded, edited, processed and saved.		1

The application includes the requested playback controls, and these have been satisfactorily implemented. In particular, the Record button allows the user to record the processed audio signal in WAV format.		1
The effects have been connected in a chain. The chain is functioning properly, and the user can listen to the processed audio signal.		1
The filters have been correctly configured, and include the requested controls.		1
The written report includes a brief description of the processes of audio recording, editing, processing and saving in Audacity.		1
The written report includes a brief description of the main characteristics of each effect and how they have been programmed.		1
The written report includes a brief analysis of the application discussing how the low-pass filter and the master volume effects affect the sound's spectrum.		1
The application includes further development.		2
Total		10

Exercise 2

Description

A famous DJ has contacted you to develop an interactive web-based application for visualising his music during its concerts. The application must be based on p5.js, p5.speech and the JavaScript audio feature extraction library Meyda.

<u>Task 1</u>

First, to evaluate your skills, you are asked to perform the following task. The DJ sends you three sounds (Ex2_sound1.wav, Ex2_sound2.wav and Ex2_sound3.wav) and you have to select Meyda audio features that could help represent these sounds visually in an appropriate manner. For example, if the 'brightness' of one of the sounds radically changes over time, to select an audio feature that measures the brightness of this sound could be a good choice from a perspective of producing visual impact.

To perform Task 1, you have to fill in the following table. You have to select three Meyda audio features for each sound and justify your selections.

	Meyda audio features	**Justification**
Sound 1		
Sound 2		

Sound 3		

Task 2

The second task consists of creating the aforementioned web-based application for audio visualisation. The application (*exercise 2*) will use the song Kalte_Ohren_(_Remix_).mp3 (*) as an audio source.

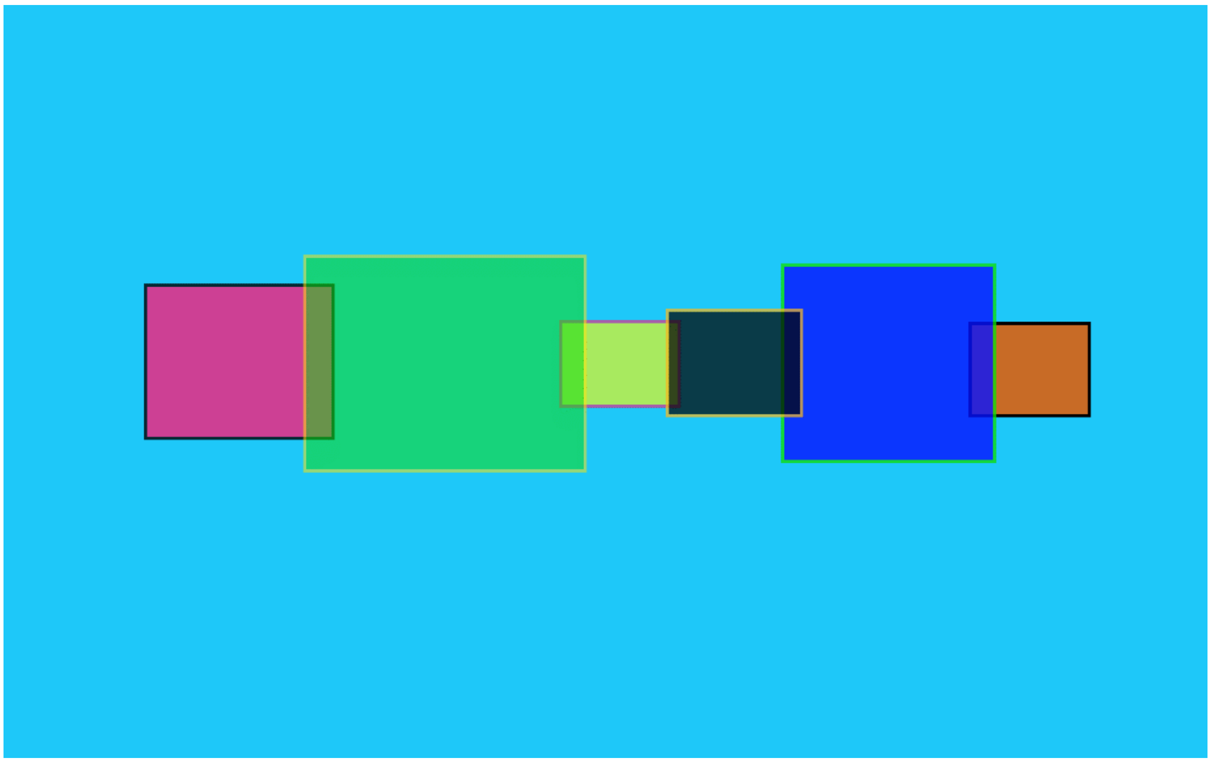

Figure 3. Idea for the audio visualisation application.

You could use the image of Figure 3 as an inspiration. The visual variables could include:

1. Number of rectangles.
2. Rectangle size.
3. Rectangle fill colour.
4. Rectangle border size.
5. Rectangle border colour.
6. Rectangle fill colour opacity.
7. Rectangle border opacity.
8. Rectangle rotation.
9. Background colour.

You have the full freedom to choose which audio features to use and how to map them to the visual variables.

Ideas for further development:

1. The application could include a voice control system, implemented with p5.speech, that could recognise voice commands such as:
    a. Black, White, Red, Blue, Green: to change the background colour to one of these colours.
    b. Square, Triangle, Circle, Pentagon: to change the shape of the generated figures to one of these shapes.

List of deliverables

For Exercise 2, you should submit in a ZIP file:

- The source code of the application *exercise 2*.
- A screencast recording demonstrating that the applications meet all the requirements and showing the further developments implemented (maximum length of two minutes).
- A link to the application running in a web page using the Coursera static web page function.
- A written report in PDF format, approximately 500 words. This report must include:
    - The table of Task 1.
    - A description and justification of the audio features and mapping implemented in Task 2 from a perspective of visual impact.
    - A brief description of the further development implemented.

Marking criteria

	Done?	Marks
The screencast recording has a maximum length of two minutes, and it demonstrates that the application meets all the requirements and shows the further developments implemented.		1
The written report includes the table of Task 1 and the selected audio features as well as the justifications are appropriated.		2
In the application *exercise 2,* the audio feature extraction and the mapping between visual variables and audio features has been correctly configured.		2
The application *exercise 2* visualises the song in an appropriate manner.		2
The written report includes a description and justification of the audio features and mapping implemented in Task 2 from a perspective of visual impact.		1
The application includes further development.		2
Total		10

(*) *Kalte Ohren (Remix)* by Dysfunction_AL (c) copyright 2019 Licensed under a Creative Commons Attribution (3.0) license. http://dig.ccmixter.org/files/destinazione_altrove/59536 Ft: Starfrosch, Kara Square

Exercise 3

Description

Audio steganography is the *art* of hiding data in an audio signal in an imperceptible manner.

Let us assume that you work at the UK Security Service (MI5) and, as an expert in audio steganography, you have to perform the following tasks:

Task 1

The MI5 intercepts an audio file (Ex3_sound1.wav). You are suspicious that it contains secret data embedded in the file by the LSB audio steganography method, a common method of audio steganography consisting of hiding the *secret* data into the least significant bit (LSB) of an audio file.

The first task of Exercise 3 is to create an application in Python (*exercise 3.1*) that must extract and read the hidden message embedded in Ex3_sound1.wav.

Task 2

The second task consists of analysing a group of suspicious audio files (Ex3_sound2.wav, Ex3_sound3.wav and Ex3_sound4.wav), determining which one contains a four-number secret code. In this case, the spy has used a more sophisticated method. It seems he has used amplitude modulation to 'move' the secret code to an ultrasonic range of frequencies, then mixing this code with that of the suspicious audio files.

To solve this task, you must create an application in Python (*exercise 3.2*) that should detect which one of the audio files seems to contain suspicious data in an ultrasonic range of frequencies, and should be able to play the secret code in an audible range of frequencies.

Task 3

The third task consists of creating a more sophisticated algorithm for embedding hidden messages based on the LSB audio steganography method (*exercise 3.3*). You will create an application in Python and use the audio file Ex3_sound5.wav to embed the secret message 'Father Christmas does not exist'. The application must include an algorithm that performs the opposite operation (i.e. an algorithm able to extract the hidden message embedded in Ex3_sound5.wav).

You could, for example, to distribute the hidden message through non-consecutive audio samples using a random pattern, use more than one least significant bits to hide the secret data, etc.

List of deliverables

For Exercise 3, you should submit in a ZIP file:

- The source code of the applications *exercise 3.1*, *exercise 3.2*, and *exercise 3.3*.
- For each application, a link to the application running in a Jupyter notebook link in Coursera.

Marking criteria

	Done?	Marks

The Jupyter notebook for *exercise 3.1* correctly extracts the hidden message embedded in Ex3_sound1.wav.		2
The Jupyter notebook for *exercise 3.1* includes markdown cells describing the code in detail.		1
The application *exercise 3.2* correctly detects which one of the audio files includes the secret code.		1
The application *exercise 3.2* is able to play, in an intelligible way, the secret code hidden in one of the audio files.		2
The Jupyter notebook for *exercise 3.2* includes markdown cells describing the code in detail.		1
The application *exercise 3.3* embeds the required message in Ex3_sound5.wav using your own system based on the LSB audio steganography method.		1
The application *exercise 3.3* includes an algorithm able to extract the hidden message embedded in Ex3_sound5.wav.		1
The Jupyter notebook for *exercise 3.3* includes markdown cells describing the code in detail.		1
Total		10

Exercise 4

Description

A software development company has contacted you to create a speech recognition system to integrate in a Python project they are developing. In particular, the project consists of an airport virtual assistant.

You have to build a prototype of the application (*exercise 4*) that should meet the following requirements:

1. The application must be written in Python.
2. Your client prefers to host the ASR software package in the application to avoid slow-down or interruptions to the system in the event of issues with the internet connection. In particular, they want you to base the speech recognition system on the Mozilla DeepSpeech software package.
3. The application must be capable of language-selection, at the very least compatible with the following languages: English, Italian and Spanish (see document Ex4_models.pdf).
4. The airport virtual assistant will be installed in an environment that can be extremely noisy. So, the speech recognition system should be configured to be able to handle this situation. Your client gives you freedom for implementing any solution (for example, to configure in python a gain/amplification, low pass filter, or some other audio filter to improve the error rate).
5. The company has prepared a set of audio files with which you can evaluate the system. For this evaluation, you will test how well it recognises several phrases in each language. You

also have to record and evaluate two short sentences (your_sentence1.wav and your_sentence2.wav). Feel free to prepare your own sentences.

The output of your work should be a table with the following information, where WER (*) is the word error rate for the phrase (see Ex4_audio_files.zip):

Language	**File**	**WER**
English	suitcase.wav	0%
Spanish	maleta.wav	25%
English	your_sentence1.wav	0%
English	your_sentence2.wav	20%
…	…	…

6. In this step of the project, you have to build a prototype, so you have to focus on the functionality of the application rather than in its visual design.

Ideas for further development:

1. To evaluate several automatic speech recognition (ASR) systems and to produce a report with the results of the evaluation. You should make your recommendation for which ASR system the company should use, or if you do not think any of them are suitable, you should say that. Justify your finding with data. (Important: for this task, evaluate only the English language).

   For maximum marks, you should evaluate at least a couple of ASR systems. We recommend that you use the offline ones shown in the course as they do not have limits on the number of requests. You can use cloud APIs if you wish, but be careful with the number of requests you make.

List of deliverables

For Exercise 4, you should submit in a ZIP file:

- The source code of the application *exercise 4* (Note: you do not have to include the acoustic models used).
- The source code used to perform the evaluation of ASR systems (further development).
- The audio files your_sentence1.wav and your_sentence2.wav.
- A screencast recording demonstrating that the application meets all the requirements and shows implementation of the further developments (maximum length of three minutes).
- A written report in PDF format, approximately 700 words. This report must include:
  - A brief description of the software used and how it has been configured.
  - A brief analysis of the solution applied to the issue of the noisy environment.
  - An analysis of the result of the test based on the set of audio files provided by the client and your own recordings.
  - A brief analysis of the evaluation of several ASR libraries (further developments).

Marking criteria

	Done?	Marks
The screencast recordings have a maximum length of three minutes in total, and they demonstrate both the test result and the implementation of further developments.		1
The written report includes all the analysis and descriptions specified in the List of deliverables. The submission also includes the audio files your_sentence1.wav and your_sentence2.wav.		2
The application satisfactorily passes the test in English (WER < 25%) based on the set of audio files.		2
The application satisfactorily passes the test in Italian (WER < 35%) based on the set of audio files.		1
The application satisfactorily passes the test in Spanish (WER < 35%) based on the set of audio files.		1
The application includes solutions that attenuate the issue of the noisy environment.		1
The application includes further development.		2
Total		10

(*) Word error rate (WER) can be computed as:

$$WER = \frac{S + D + I}{N} * 100$$

where

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- N is the number of words in the sentence