UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Information and Coding (2025/26)

Lab work nº 1 — Due: 19 Oct 2025

Intro

- All programs should be implemented in C++. If you are not familiar with C++, start by taking a look at one of the many available tutorials (see, for example, `https://www.tutorialspoint.com/cplusplus/index.htm`).
- The Standard Template Library (STL) will be also a valuable resource. See, for example, `https://www.tutorialspoint.com/cplusplus/cpp_stl_tutorial.htm`.
- For a comprehensive C++ reference, see `https://en.cppreference.com/`.
- Use a github (or equivalent) repository to manage the developement of your software.

Initial setup

- Install the software `sndfile-example.tar.gz` (available from elearning). It is based on the library `libsndfile`, designed to allow easy reading and writing of many different sound file formats (see `https://libsndfile.github.io/libsndfile/`). It contains a simple program to copy an audio file (in WAV format) into another one, block by block, a small example of a C++ class (`WAVHist`) for outputting the histogram of a WAV audio file, and an example of use of the Discrete Cosine Transform (DCT), that will be used later for compression. Note that the `libsndfile` is written in C, but that there is a C++ wrapper (`sndfile.hh`).

Part I

1. Change the `WAVHist` class in order to also provide:
   - The histogram of the average of the channels (the mono version, i.e., $(L+R)/2$; this is also known as the MID channel, used in coding) and the average difference of the channels (i.e., $(L-R)/2$, which is related to the SIDE channel, $L-R$)[1], when the audio is stereo (i.e., when it contains two channels).[2]

[1] In both cases, use integer division.

[2] It is possible to recover both channels, exactly, using the information in MID and SIDE. Can you find out how?

- Coarser bins, i.e., instead of having an histogram bin for each different sample value, have bins that gather together 2, 4, 8, ..., 2^k values.

Note: For visualizing graphically the histograms, you can either save the histogram data as a text file and use an external application to visualize it or you can extend the functionality of the program in order to graphically display the histogram.

2. Implement a program, named `wav_quant`, to reduce the number of bits used to represent each audio sample (i.e., to perform uniform scalar quantization). Note that the result should be also a WAV file.

3. Implement a program, named `wav_cmp`, that prints, for each channel and for the average of the channels:

   - The average mean squared error between a certain audio file and its original version (also known as the L^2 norm).
   - The maximum per sample absolute error (also known as the L^∞ norm).
   - The signal-to-noise ratio (SNR) of a certain audio file in relation to its original version.

4. Implement a program, named `wav_effects`, that produces some audio effects. These can be, for example,

   - A single echo
   - Multiple echos
   - Amplitude modulation
   - Time-varying delays
   - ...

Part II

5. Install the software `bit_stream.tar.gz` (available from elearning). It contains a C++ class, named `BitStream`, to read and write bits from/to a file. Compile, run the examples and study how it works. Later, you can extend it, for example by implementing other methods that might be necessary or useful.

6. Using the `BitStream` class, implement a codec (encoder and decoder, named `wav_quant_enc` and `wav_quant_dec`), able to pack the result of the reduction of the number of bits by uniform quantization. Therefore, it will be similar to `wav_quant`, but instead of producing a WAV file, the encoder will produce a packed (encoded) representation, whereas the decoder will recover the quantized WAV file.

Part III

7. Implement a lossy codec for mono audio files (i.e., with a single channel), based on the Discrete Cosine Transform (DCT). The audio should be processed on a block by block basis. Each block will have to be converted using the DCT, the coefficients appropriatelly quantized, and the bits written to a file, using the `BitStream` class. The decoder should rely only on the (binary) file produced by the encoder in order to reconstruct (an approximate version of) the audio. Note that the idea is to get good compression without degrading too much the audio.

Part IV

8. Elaborate a concise report, where you describe all the relevant steps and decisions taken in all the items of the work. When appropriate, include also measures of processing time, compression ratios and corresponding errors introduced by the compression process. For this, use several audio files. This report should not be a description of the code implemented. Instead, it should illustrate what can be obtained using the software developed and how it can be obtained.