UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA

Information and Coding (2023/24)

Lab work n$^{\underline{o}}$ 1 — Due: 29 Oct 2023

Intro

- All programs should be implemented in C++. If you are not familiar with C++, start by taking a look of one of 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., $(L+R)/2$, also known as the MID channel) and the difference of the channels (i.e., $(L-R)/2$, also known as the SIDE channel)[1], when the audio is stereo (i.e., when it contains two channels).
   - 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.

[1] In both cases, use integer division.

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_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.

3. Implement a C++ class and associated program, `wav_quant`, to reduce the number of bits used to represent each audio sample (i.e., to perform uniform scalar quantization).

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. Develop a C++ class, denoted `BitStream`, to read and write bits from/to a file. This class should have, at least, methods to write one bit, read one bit, write N bits and read N bits (with $0 \leq N \leq 64$), and read and write strings. You can implement other methods that you think might be necessary or useful. Note that this code should be as efficient as possible, because it will be extensively used by the codecs. Also, note that the bits have to be packed in bytes, from the most to the least significant bit. For example, consider that a certain file contains only two bytes, with hexadecimal values A7 and C0. Then, when asked to return the first 11 bits from this file, the `BitStream` class will have to return 10100111110.

6. Using the `BitStream` class, implement a program (encoder and decoder), where the encoder should be able to convert a text file containing only 0s and 1s into the binary equivalent (each byte of the binary file should represent eight of the bits in the text file). The decoder should implement the opposite operation. The main goal of these two programs is to test the `BitStream` class developed before, although they might also be used in the future for debugging the output of other codecs.

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 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.