Audio and Video Coding Lab work nº1 Universidade de Aveiro

Miguel Maia, *Universidade de Aveiro, 76434*
Silvério Pereira, *Universidade de Aveiro, 76505*
Joao Amaral, *Universidade de Aveiro, 76460*

***Abstract*—Report documenting the development of the first practical guide.**

I. Description of the exercises developed within this practical guide

II. Exercise 2

In this exercise the wavhist.h file was changed in order to show the mono version of the audio file. The usage of the wavhist binary is as follows:

```
wavhist <input file> <channel>
```

To get the mono version the channel must be specified as -1. The mono histogram for the audio sample given is show in Figure 1

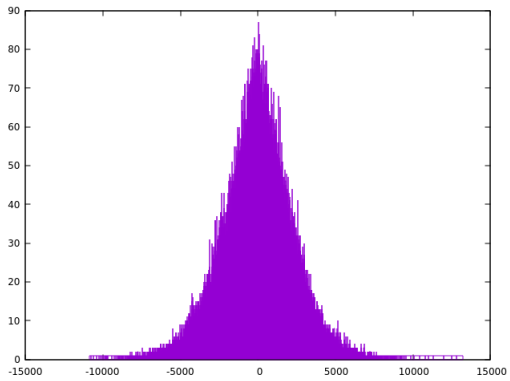

Fig. 1: Mono histogram

III. Exercise 3

A. Exercise description

In this exercise, a sound file is compressed using uniform scalar quantization. The encoding case is used as follows:

```
wavquant -e|--encode <input file>
<output file> <quantsize>
```

in which the quantsize is the number of bits we want to use for each sample. To achieve this, a BinaryStream class was created for I/O operations at the bit level. The outputfile contains the quantized data, as well as the following metadata structure:

```
struct
{
        int format;
        int channels;
        int samplerate;
        short quantsize;
} SF_METADATA;
```

which is used in the decoding process. For decoding the following command is used:

```
wavquant -d|--decode <input file>
<output file>
```

it reads the SF_METADATA structure and the quantized data to output a sound file.

B. Performance measures

The user system time used by the program was 0.070s and was the same for the encoding and decoding process. With 8 bit quantization we achieve a compression rate of 5.27, with an SNR of 28 and a max sample error of 255.

IV. Exercise 4

A. What is the SNR?

The objective of this exercise is to calculate and print the signal-to-noise ratio (SNR) of a certain audio file in relation to the original file. Signal-to-noise ratio (SNR) is a measure to compare the power of a desired signal to the level of background noise. This means that it is the ratio between the power of a signal (meaningful information) and the power of background noise (unwanted signal).

$$\mathrm{SNR} = \frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}},$$

Fig. 2: Image extracted from wikipedia.

Signals are often expressed using the logarithmic decibel scale. Based upon the definition of decibel, signal and noise may be expressed in decibels (dB) as

$$P_{\text{signal,dB}} = 10\log_{10}(P_{\text{signal}})$$

and

$$P_{\text{noise,dB}} = 10\log_{10}(P_{\text{noise}}).$$

Fig. 3: Image extracted from wikipedia.

B. How to calculate the SNR?

To calculate the SNR, it is necessary to include as arguments of the program, the original wav file and secondly the transformed audio file:

```
wavcmp <original file> <noise file>
```

First of all, it is necessary to read all the samples from either the original file and the noise file. Then, the samples from the original file are square raised and its sum is calculated and saved into an array, for each channel. The same process is done to the noise file samples, but instead of being just square raised, the samples from the original file are subtracted by the samples from the noise file, and the sum of this subtraction is saved into an array. Then, to calculate the SNR, a division for each sample is made:

```
for(unsigned int i=0; i<samples.size(); i++){ // calculate the sum
    size_t ch_idx = n++ % SF_METADATA.channels; // channel index
    sum_signal[ch_idx] += pow ( samples[i] , 2); // signal sum
    sum_error[ch_idx]  += pow ( samples[i] - samplesNoise[i], 2);
    currSnr = 10*log10(sum_signal[ch_idx] / sum_error[ch_idx]);
    if(currSnr > psnr)
        psnr = currSnr;
    error = samples[i] - samplesNoise[i];
    if(error > max_sample_error)
        max_sample_error = error;

}
```

Fig. 4: SNR calculation

In order to obtain the PSNR, which is the maximum value of the SNR, all values of SNR are compared when the biggest one is found, it means the PSNR has been found. Also, the max sample error is calculated, subtracting the current sample from the original file by the current sample from the noise file. When calculating the max sample error for a new sample, its result is compared with the previous maximum sample error and in case the current one is bigger, then it means that it is so far the maximum sample error.

V. Exercise 5

A. Exercise description

The wavcb binary is used as follows:

```
wavcb <input file> <codebook name>
<codebook size> <block size>
<overlap factor> <threshold>
<max iterations>
```

For this we created an LBG class with the following methods:

- `updateTrainingVectors()`, in which we accept samples and add them to the training vectors used by the LBG algorithm, it also accepts an overlap factor.
- `chooseInitialTrainingVectors()`,where we choose the initial centroids, our strategy for initialization is to choose N of our current training vectors at random, where $N = codebooksize$.
- `clusterAssign()` is in charge of assigning a centroid to each training vector, it assigns each training vector to it's closest centroid and uses euclidean distance as the distance measure.
- `moveCentroids()` uses the assignments made by `clusterAssign()` to move the centroids to the average of their assignment, if there were no assignments to centroid there's no change in it's location.
- `calculateDistortion()` also uses the assignments to calculate the distortion of the solution.
- `runLBG()` accepts the threshold and maximum iteration parameters and runs the LBG algorithm, as was implemented by us, the LBG algorithm is described as follows
  1) $M = number\,of\,training\,vectors$
     $N = codebook\,size$
  2) classify M training vectors into N clusters according to distance, using `clusterAssign()`
  3) update cluster centers (centroids) by averaging the training vectors assigned to them, using `moveCentroids()`
  4) use `calculateDistortion()` to calculate distortion
  5) if the difference between the current and last distortions is less than threshold or max iterations have passed return calculated codebook
  6) else repeat
- `writeCodebook()` to save codebook to disk

In the actual program, the order of operations is

1) `updateTrainingVectors()`, note that we use the full input audio file for training
2) `chooseInitialTrainingVectors()`
3) `runLBG()`
4) `writeCodebook()`

B. Performance description

With a codebook size of 128, a block size of 16 and a max iteration of 100 we use 3m51.202s of user time.

SNR	C1:-9.4552 C2:-5.3159	C1:-9.4552 C2:-5.3159
Time	3m36,881s	3m36,881s
Codebook size	2	4
Block size	16	16
Overlap Factor	2	2
Threshold	100	100
Max Iterations	100	100

VI. Exercise 6

A. Exercise description

The wavvq binary can be used in the following ways:

```
wavvq -e|--encode <input file>
<output file> <codebook>
```

where we only have to specify a previously calculated codebook,

```
wavvq -e|--encode <input file> <output file>
<codebook name> <codebook size>
<block size> <overlap factor>
<threshold> <max iterations>
```

where we specify all the parameters, like in wavcb, both the codebook and the quantized file are saved to disk.

```
wavvq -d|--decode <input file>
<output file> <codebook>
```

used for decoding a quantized file with a given codebook

The codebook calculation, if needed is the same as described in V-A so we'll skip that. To encode an audio file, given an already calculated codebook we need to sample the audio file into vectors of a specified size, check the closest indexes to those vectors and save those. This is equivalent to updating out training vectors with an overlap factor of 0 and running our `calculateDistortion()` method, so we do exactly that to calculate the indices, the quantized file is also saved with some metadata, similar to the one described in III but with some more parameters, namely element size and codebook size

B. Performance description

Using the codebook calculated in V-A we have an user time of 0m2.200s and a compression rate of 35.98. for decoding we use 0m0.025s of user time, however the generated audio file had too much noise so our SNR was -3 and our max sample error was 41833.

VII. Distribution of work

Silvério Pereira - 40%
Miguel Maia - 30%
João Amaral - 30%