# Codificação de Áudio e Vídeo
# 2º Trabalho

Joel Pinheiro (65151)

Luis Assunção (42967)

***Resumo* – O objectivo deste segundo trabalho foi desenvolver um algoritmo com capacidade de codificar e descodificar ficheiros de áudio, nomeadamente WAV, de forma lossy e lossless. Para tal utilizamos um método de codificação preditiva, que tomando partido das relações entre os dados nos permite prever qual o valor seguinte.**

***Palavras Chave* – Lossless, Lossy, WAV, Previsão, Golomb.**

## I. INTRODUÇÃO

Neste segundo trabalho temos como objetivo desenvolver algoritmos de compressão e descompressão para ficheiros WAV, baseados no método de codificação preditiva, quer Lossless ou Lossy.

Para tal, usámos a libraria libsndfile (http://www.mega-nerd.com/libsndfile/) que nos auxilia para lermos um ficheiro WAV para um array de inteiros. Aplicamos um algoritmo de previsão a esse mesmo array por forma de reduzir a entropia entre as amostras, que de seguida passamos ao codificador de golomb para compressão. Uma vez que a entropia gerada pelo nosso algoritmo preditivo é "baixa", iremos obter taxas de compressão "eficientes". Para descomprimir basta-nos efetuar estes passos por ordem inversa.

## II. CODIFICAÇÃO LOSSLESS

No que diz respeito a codificação Lossless, desenvolvemos dois algoritmos de previsão distintos.

Passamos a demonstrar o primeiro:

### *Previsão emaranhada*

Designámos "emaranhada" pela forma como construímos o array de inteiros que vamos passar ao algoritmo de previsão.

Um ficheiro WAV é composto por dois canais, que vamos designar de L para esquerdo, e R para direito. Assumindo que cada amostra é composta por um tuplo (L,R), e que um dado ficheiro tem 100 amostras, então construímos um array com 100 * 2 posições, segundo o padrão L1,R1,L2,R2,L3,R3,L4,R4, etc.

### *A. Preditor de 1º nível*

$$e_n = f_n - f_{n-2}$$

No nosso algoritmo de predição de primeiro nível é usado o array de amostras emaranhado e apura os residuais usando a diferença do próprio valor da amostra com o valor da amostra da anterior da anterior. Este algoritmo explora apenas a redundância que existe em amostras perto umas das outras.

### B. *Preditor de 2º nível*

$$e_n = f_n - \widehat{f_n}$$
$$\widehat{f_n} = (f_{n-1} + f_{n-2}) \times \frac{1}{2}$$

O nosso algoritmo efetua a subtração do valor atual pela média dos dois valores anteriores. Esta média, devido a natureza do nosso array, vai ter sempre em consideração os valores dos canais direito e esquerdo, assumindo que não haverá muita variação entre os canais esquerdo e direito, e para os valores futuros também. Este algoritmo é adequando teoricamente para voz, ou sons que variem pouco, produzindo uma entropia muito baixa. Para amostras cujas amplitudes variem muito em cursos espaços de tempo, existem algoritmos preditivos mais adequados que este.

## III. CODIFICAÇÃO LOSSY

Para implementar um algoritmo Lossy, apenas aplicámos a seguinte equação:

$$\widetilde{e_n} = \left\lfloor \frac{e_n}{k} \right\rfloor$$

Para efetuar a operação inversa na descodificação, basta aplicar a seguinte equação:

$$e_n = \widetilde{e_n} \times k$$

Devido ao arredondamento por defeito, não vamos conseguir recuperar o valor original, daí a designação Lossy. O ficheiro resultante não será igual ao original, e pode mesmo ser completamente diferente do ponto de vista da sua utilização, dependendo do quantilizador, k. No entanto devido a natureza do algoritmo de codificação de Golomb, vamos poder representar estes valores mais pequenos com menos bits, comprimindo assim ainda mais o ficheiro resultante.

## IV. CLASSES IMPLEMENTADAS

```
#ifndef TRABALHO2_LOSSLESSAUDIOCODEC_H
#define TRABALHO2_LOSSLESSAUDIOCODEC_H

#include "./Golomb/Golomb.h"


class LosslessAudioCodec {
public:
    LosslessAudioCodec(char* inputFile, char*
outputFile, int m, int predictor_level);
    ~LosslessAudioCodec();
    void Encode();
    void Decode();
    int *entangledArray;
    int *entangledDecodedArray;
    int entangledArraysSize;
protected:
private:
    GolombEncoder* gEncoder;
    GolombDecoder* gDecoder;
    char* inputFile;
    char* outputFile;
    int m;
    int predictor_level;
};
```

```
#ifndef TRABALHO2_LOSSYAUDIOCODEC_H
#define TRABALHO2_LOSSYAUDIOCODEC_H

#include "./Golomb/Golomb.h"

class LossyAudioCodec {
public:
    LossyAudioCodec(char* inputFile, char*
outputFile, int m, int q, int predictor_level);
    ~LossyAudioCodec();
    void Encode();
    void Decode();
    int *entangledArray;
    int *entangledDecodedArray;
    int entangledArraysSize;
protected:
private:
    GolombEncoder* gEncoder;
    GolombDecoder* gDecoder;
    char* inputFile;
    char* outputFile;
    int m;
    int q;
    int predictor_level;
};
```

## IV. EXECUÇÃO

Nesta secção mostraremos a forma como o nosso projecto se encontra estruturado e como executá-lo:

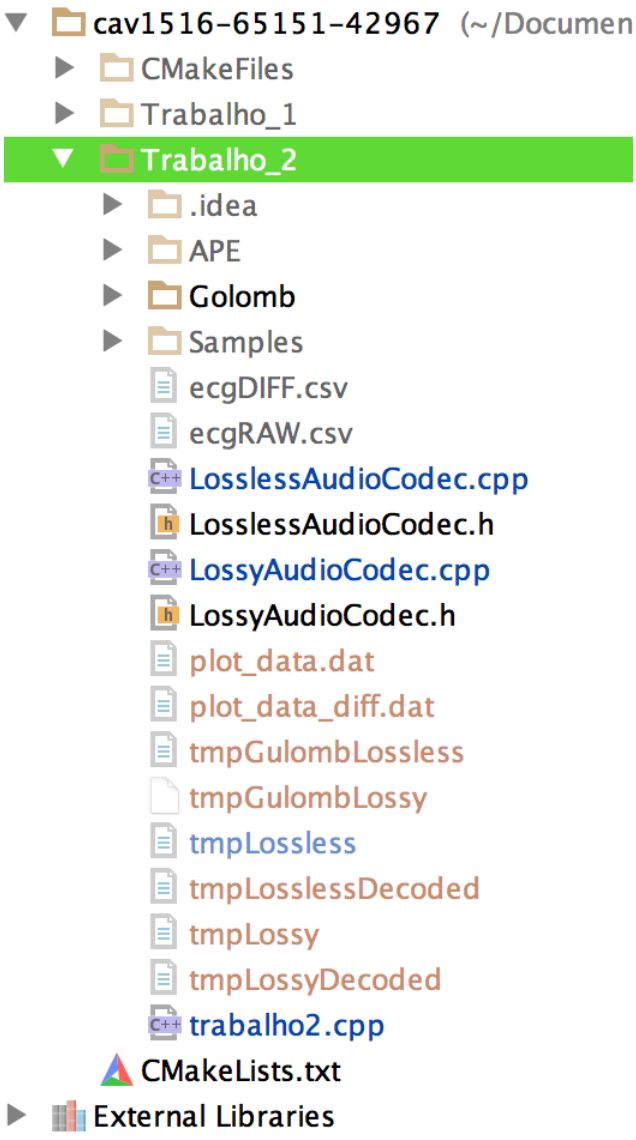


*Figura 1 : Estrutura do projecto*

Este projecto é composto por dois subprojectos:

- O formato APE, residente na pasta APE tem que ser compilado recorrendo ao comando CMAKE manualmente ou através da linha de comandos. Baseados nos exemplos fornecidos pelos autores do formato APE;
- Correr o CMAKE dentro da pasta do Trabalho_2 para compilar os nossos codecs Lossy e Lossless.

### *A. Codec Lossless e Lossy desenvolvido*

Para compilar o programa pode correr o cmake e de seguida fazer make:

```
$ cmake CMakeLists.txt
$ make
```

De seguida, para correr o programa faça:

```
$ ./TRABALHO2 [wav original] [wav lossless] [wav lossy] [M] [Q] [PREDICTION_LEVEL]
```

Onde M é o valor usado para a compressão Gulomb e Q o quantizador do modelo Lossy. Exemplo:

```
$ ./TRABALHO2 ./Trabalho_2/Samples/sample07.wav ./Trabalho_2/Samples/Lossless.wav ./Trabalho_2/Samples/Lossy.wav 480 50 1
```

### B. *APE Lossless Codec*

Para compilar o programa pode correr o cmake e de seguida fazer make:

```
$ cmake CMakeLists.txt
$ make
```

De seguida, para correr o programa faça:

```
Usage: [Input File] [Output File] [Mode]

Modes:
    Compress (fast): '-c1000'
    Compress (normal): '-c2000'
    Compress (high): '-c3000'
    Compress (extra high): '-c4000'
    Compress (insane): '-c5000'
    Decompress: '-d'
    Verify: '-v'
    Convert: '-nXXXX'

Examples:
    Compress: mac "Metallica - One.wav" "Metallica - One.ape" -c2000
    Decompress: mac "Metallica - One.ape" "Metallica - One.wav" -d
    Verify: mac "Metallica - One.ape" –v
```

Exemplo:

```
./mac sample02.wav x.ape -c2000
```

## IV. ANÁLISE DOS RESULTADOS EXPERIMENTAIS

As seguintes cores correspondem a:

- Verde: Distribuição original
- Rosa: Distribuição codificado no modo Lossless
- Azul: Distribuição codificada no modo Lossy

### A. *Sample 02*

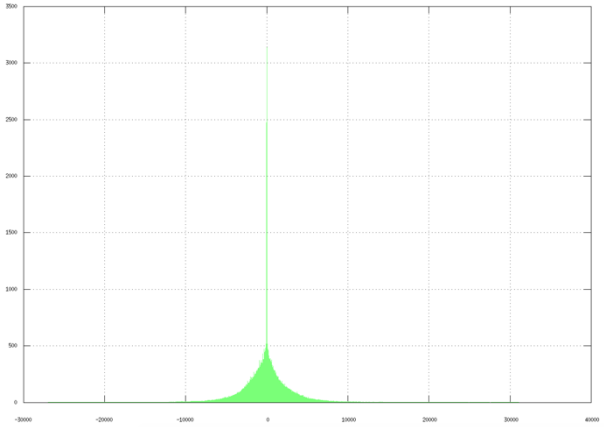

*Figura 2: Sample Original*

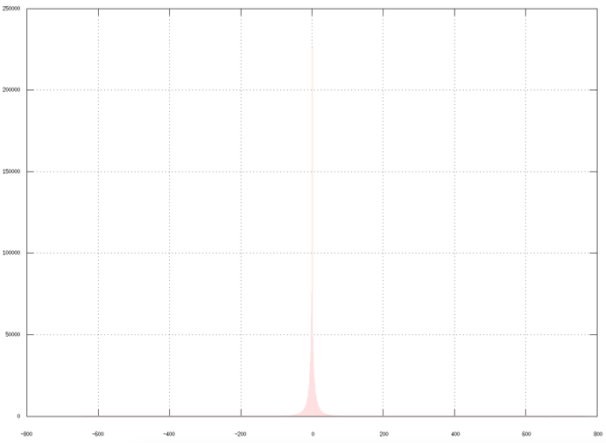

*Figura 3 : Sample Lossless*

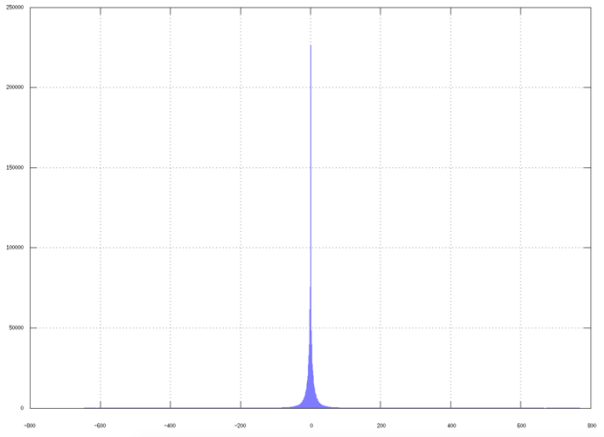

*Figura 4 : Sample Lossy*

```
LosslessAudioCodec Begin:
Frames (Samples): 647388
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e
descodificação do Codec
Lossless: 3952 milisegundos.
Frames (Samples): 647388
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tamanho do ficheiro Gulomb
Lossless codificado e: 1882425
Lossless PSNR Lossless: inf

LossyAudioCodec Begin:
Tempo de codificação e
descodificação do Codec Lossy:
3958 milisegundos.
Tamanho do ficheiro Gulomb Lossy
codificado e: 1602285
Lossy PSNR Lossy: -20.5462
```

*Resultado 1 : Operação de codificação e descodificação do sample 02 (Predictor 1)*

```
LosslessAudioCodec Begin:
Frames (Samples): 647388
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e
descodificação do Codec
Lossless: 4282 milisegundos.
Tamanho do ficheiro Gulomb
Lossless codificado e: 2146390
Lossless PSNR Lossless: inf

LossyAudioCodec Begin:
Frames (Samples): 647388
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e
descodificação do Codec Lossy:
3777 milisegundos.
Tamanho do ficheiro Gulomb Lossy
codificado e: 1657856
Lossy PSNR Lossy: -3.80201
```

*Figura 5 : Operação de codificação e descodificação do sample 02 (Preditor 2)*

| | Level 1 Predictor | | Level 2 Predictor | | APE |
|---|---|---|---|---|---|
| | Lossless | Lossy | Lossless | Lossy | Lossless |
| Tamanho | 1882425 | 1602285 | 2146390 | 1657856 | 1580708 |
| Tempo | 3952 | 3958 | 4282 | 3777 | - |
| PSNR | inf | -20.5462 | inf | -3.80201 | inf |

*Tabela 1 : Comparação de Resultados Sample 02*

Nota: Tamanho original do ficheiro é de 2589596 bytes.

### *B. Sample 03*

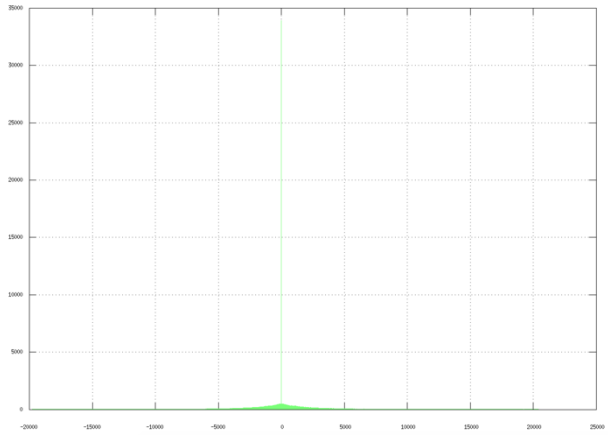

*Figura 6 : Sample Original*

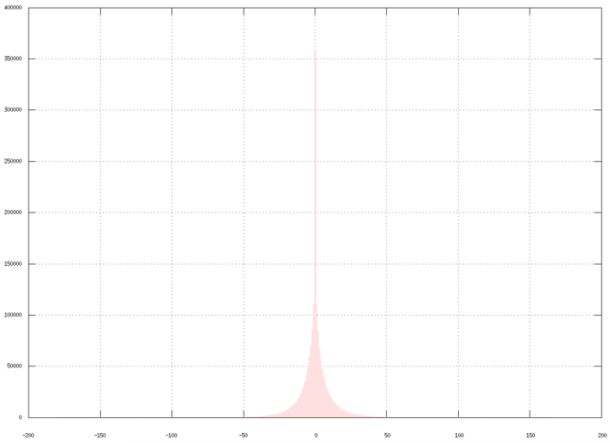

*Figura 7 : Sample Lossless*

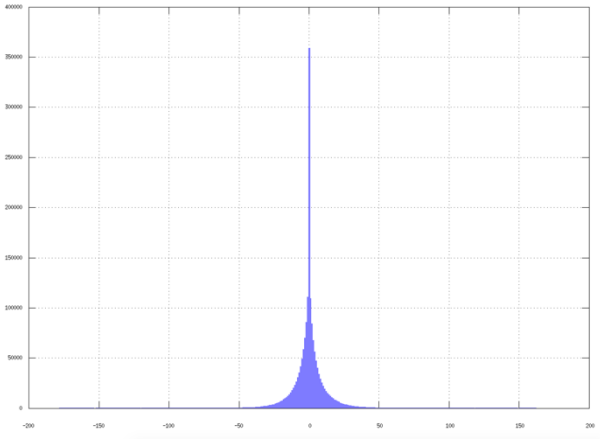

*Figura 8 : Sample Lossy*

```
LosslessAudioCodec Begin:
Frames (Samples): 882588
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e descodificação do Codec Lossless: 5188 milisegundos.
Tamanho do ficheiro Gulomb Lossless codificado e: 2467142
Lossless PSNR Lossless: inf

LossyAudioCodec Begin:
Frames (Samples): 882588
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e descodificação do Codec Lossy: 4613 milisegundos.
Tamanho do ficheiro Gulomb Lossy codificado e: 2166474
Lossy PSNR Lossy: -30.8372
```

*Figura 9 : Operação de codificação e descodificação do sample 03 (Preditor 1)*

```
LosslessAudioCodec Begin:
Frames (Samples): 882588
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e descodificação do Codec Lossless: 5348 milisegundos.
Tamanho do ficheiro Gulomb Lossless codificado e: 2617722
Lossless PSNR Lossless: inf

LossyAudioCodec Begin:
Frames (Samples): 882588
Samplerate: 44100
Channels: 2
Format: 65538
Sections: 1
Tempo de codificação e descodificação do Codec Lossy: 4691 milisegundos.
Tamanho do ficheiro Gulomb Lossy codificado e: 2209761
Lossy PSNR Lossy: -4.02251
```

*Figura 10 : Operação de codificação e descodificação do sample 03 (Preditor 2)*

## IV. RESULTADOS OBTIDOS

| | Predictor Level 1 | | | | Predictor Level 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Lossless | | Lossy | | Lossless | | Lossy | | |
| | Tempo | Tamanho | Tempo | Tamanho | Tempo | Tamanho | Tempo | Tamanho | Tamanho Original |
| Sample 01 | 4395 | 3812596 | 4261 | 3502842 | 4397 | 3945835 | 4003 | 3524392 | 5176796 |
| Sample 02 | 2079 | 1882425 | 1974 | 1720184 | 2271 | 2146390 | 2036 | 1787261 | 2589596 |
| Sample 03 | 3001 | 2467142 | 2994 | 2321819 | 3088 | 2617722 | 2798 | 2378484 | 3530396 |
| Sample 04 | 1969 | 1722521 | 1875 | 1582814 | 2172 | 2210756 | 1962 | 1670146 | 2354396 |
| Sample 05 | 2860 | 2470576 | 2677 | 2344370 | 3121 | 2875894 | 2960 | 2467995 | 3647996 |
| Sample 06 | 3267 | 2809633 | 3025 | 2664262 | 3415 | 3040233 | 3142 | 2838957 | 4235996 |
| Sample 07 | 3346 | 3364984 | 3138 | 2650213 | 3626 | 4113025 | 3222 | 2791641 | 3765596 |

## V. NOTAS FINAIS

Tento em conta os resultados obtidos anteriormente, podemos concluir que:

- Os cálculos dos valores de PSNR estão de facto incorretos, valores negativos

implicam que o ficheiro resultante é na prática apenas ruido, e tal não se verifica.

- Como seria de esperar, as taxas de compressão dos nossos algoritmos Lossy são superiores que as do Lossless, no entanto não se comparam a taxa de compressão do formato APE, que apesar de ser Lossless, supera largamente os nossos algoritmos.
- Reparámos que o nosso algoritmo de segunda ordem Lossless tem uma taxa de compressão inferior ao Lossless de primeira ordem, portanto o Lossless de primeira ordem é preferível ao de Segunda ordem.
- O algoritmo Lossy de segunda ordem tem uma taxa de compressão equiparável ao Lossy de primeira ordem, no entanto o Lossy de primeira ordem apresenta artefactos, na forma se sons agudos na ausencia de dados, enquanto que o Lossy de ordem dois não apresenta qualquer deformação do audio aparente. Torna-se evidente que para Lossless o gru um é preferivel, e para Lossy grau 2 é preferível.s

## VI. BIBLIOGRAGIA

1. http://www.mega-nerd.com/libsndfile/