

Universidade de Aveiro

# Representação de informação sonora

Ana Almeida

Pedro Alagoa

Relatório no âmbito de Laboratórios de Informática

Maio, 2014  
Aveiro

# Relatório - Representação de informação sonora

Ana Almeida e Pedro Alagoa

Maio 2014, Universidade de Aveiro

# Conteúdo

Lista de Tabelas . . . . .	2
Lista de Figuras . . . . .	3
0.1 Resumo . . . . .	4
<b>1 Introdução</b>	<b>5</b>
<b>2 Preparação</b>	<b>6</b>
2.1 Máquina e sistema operativo utilizados . . . . .	6
2.2 Instalação das aplicações e módulos necessários . . . . .	6
<b>3 Procedimento</b>	<b>7</b>
3.1 Codificação do ficheiro . . . . .	8
3.1.1 MP3 . . . . .	8
3.1.2 AAC . . . . .	8
3.1.3 GSM . . . . .	9
3.2 Ferramenta em Python . . . . .	10
<b>4 Análise de Resultados</b>	<b>11</b>
4.1 Gama Dinâmica . . . . .	11
4.2 RMSD . . . . .	12
4.3 Tamanho do ficheiro . . . . .	12
<b>5 Conclusão</b>	<b>14</b>
<b>Glossário</b>	<b>15</b>
<b>Bibliografia</b>	<b>16</b>
<b>A Comparador</b>	<b>17</b>
A.1 Código . . . . .	17
A.2 Descrição . . . . .	18
<b>B Ferramenta</b>	<b>19</b>

# Lista de Tabelas

4.1	Tamanhos dos ficheiros . . . . .	12
-----	----------------------------------	----

# Lista de Figuras

3.1	Conversão de Stereo para Mono . . . . .	8
3.2	FDK AAC Encoder em execução . . . . .	9
4.1	Gama Dinâmica . . . . .	11
4.2	RMSD . . . . .	12

## 0.1 Resumo

Com o intuito de comparar o resultado da codificação de ficheiros de áudio(MP3, AAC e GSM) e, posteriormente, analisar os seus resultados, foi desenvolvida uma ferramenta utilizando o Python, onde são retirados valores como a gama dinâmica e o desvio entre o ficheiro codificado e o original.

Após a preparação da máquina e da instalação das aplicações e módulos necessários, procedeu-se à codificação do ficheiro modelo. De seguida, desenvolveu-se a ferramenta em Python e esta foi utilizada para retirar os valores pretendidos. Depois procedeu-se à sua análise, sob a forma de gráfico.

Ao analisar os gráficos obtidos, concluiu-se que, no geral, a codificação para MP3 é a mais indicada das 3 codificações, para os valores testados. A única excepção é no caso do GSM, para bitrates muito pequenos. Concluiu-se também que a gama dinâmica não é afectada pela variação de *bitrates* de cada codec.

# Capítulo 1

## Introdução

Este relatório tem como objectivo principal aplicar as funcionalidades da linguagem Python no tratamento de informações de áudio.

Primeiramente, codificou-se um ficheiro de áudio de boa qualidade com codecs e valores de [Bitrate](#) diferentes. Depois, foram convertidos para [WAV](#), para serem processados pelo programa de Python.

O programa em Python servirá como uma ferramenta que compara o resultado da codificação, calculando a gama dinâmica e o desvio entre os valores do ficheiro final e do ficheiro original, que deverá ser obtido através do método [RMSD](#).

Por fim, este relatório conclui com um comentário final à análise dos resultados obtidos.

## Capítulo 2




# Preparação

### 2.1 Máquina e sistema operativo utilizados

A máquina utilizada foi o Toshiba Satellite P50-A-125. O sistema operativo utilizado foi o Windows 8 64 bit.

### 2.2 Instalação das aplicações e módulos necessários

Foi necessário instalar as seguintes aplicações:

- Python v2.7.6 [1] 
- Notepad++ v2.0.3 [2] 
- Audacity v2.0.3 [3] 
- FDK AAC Encoder 0.5.3 [4] (libfdk-aac 3.4.12)

Foi ainda necessário instalar alguns módulos necessários à implementação da ferramenta em Python. Os módulos instalados foram os seguintes:

- NumPy
- SciPy

Como estes módulos não foram desenvolvidos para o sistema operativo em que se realizaram os testes, recorreu-se a *binaries* [5] não oficiais.

O programa de reprodução de áudio utilizado foi o Foobar2000 [6].



## Capítulo 3

# Procedimento

Depois de instalados e configurados os programas, procedeu-se à codificação do ficheiro de áudio em 3 formatos distintos: [MP3](#), [AAC](#) e [GSM](#).

Procurou-se um ficheiro de áudio de alta qualidade e o escolhido foi uma música da autoria de ZUN, que faz parte da BGM do jogo Touhou 6[7]. O ficheiro de áudio utilizado pode ser obtido aqui: [http://wikisend.com/download/703922/th06\\_05.wav](http://wikisend.com/download/703922/th06_05.wav).

O ficheiro utilizado tem as seguintes características:

**Tipo de ficheiro** - WAV

**Tamanho** - 15,1 MB

**Bitrate** - 1411kbps

## 3.1 Codificação do ficheiro

### 3.1.1 MP3

Para codificar o ficheiro com codecs que originam um ficheiro **MP3** utilizou-se o Audacity.

De seguida, como sugerido no enunciado do relatório, utilizou-se apenas um canal de áudio (mono).

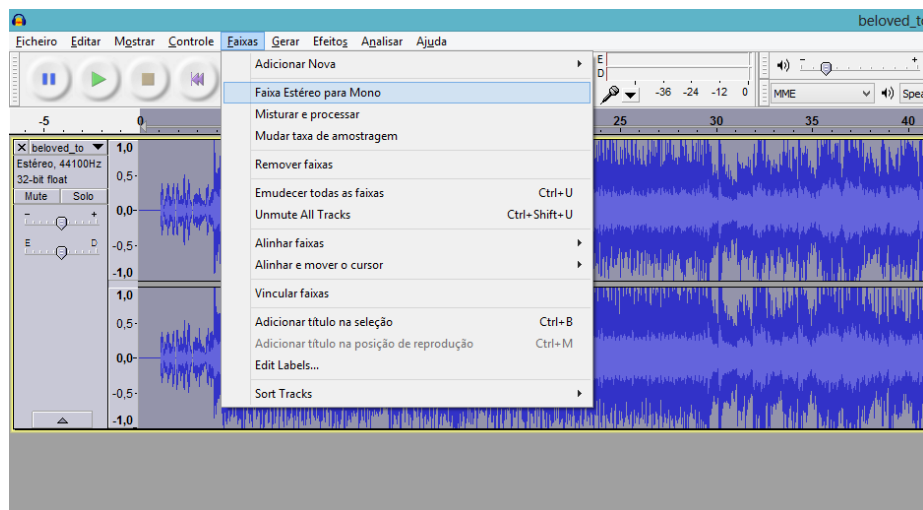


Figura 3.1: Conversão de Stereo para Mono

Depois procedeu-se à codificação do ficheiro, variando-se o valor de **Bitrate**. Ao todo, codificaram-se 6 ficheiros **MP3**, com *bitrates* de:

- 32kbps
- 71kbps
- 96kbps
- 128kbps
- 192kbps
- 320kbps

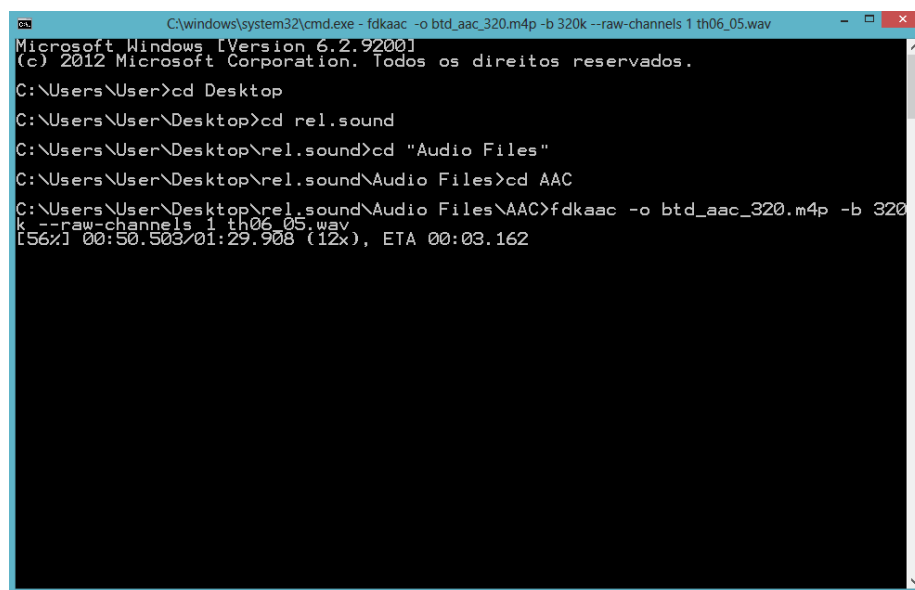
Por fim, estes ficheiros foram convertidos para o formato **WAV**, de modo a serem processados pela ferramenta implementada.

### 3.1.2 AAC

Para codificar o ficheiro com codecs que originam um ficheiro **AAC** utilizou-se a ferramenta FDK AAC Encoder 0.5.3, que utiliza a biblioteca de codecs fdk-aac.

Optou-se por utilizar esta aplicação porque a interface do Audacity não permite definir valores de [Bitrate](#) para os ficheiros [AAC](#) e também porque o torna instável durante a reprodução do ficheiro.

Procedeu-se à codificação do ficheiro, variando-se o valor de [Bitrate](#). Ao todo, codificaram-se 6 ficheiros [AAC](#), com bitrates de iguais aos codificados para o [MP3](#) (ver [Subseção 3.1.1.](#))



```
C:\windows\system32\cmd.exe - fdkaac -o btd_aac_320.m4p -b 320k --raw-channels 1 th06_05.wav
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. Todos os direitos reservados.
C:\Users\User>cd Desktop
C:\Users\User\Desktop>cd rel.sound
C:\Users\User\Desktop\rel.sound>cd "Audio Files"
C:\Users\User\Desktop\rel.sound\Audio Files>cd AAC
C:\Users\User\Desktop\rel.sound\Audio Files\AAC>fdkaac -o btd_aac_320.m4p -b 320k --raw-channels 1 th06_05.wav
[56%] 00:50.503/01:29.908 (12x), ETA 00:03.162
```

Figura 3.2: FDK AAC Encoder em execução

**fdkaac** - Executa o programa

**-o btd\_aac\_320.m4p** - Nome do ficheiro de saída

**-b 320k** - Indica o [Bitrate](#) pretendido

**--raw-channels 1** - Indica o nº de canais a utilizar.

**th06\_05.wav** - Ficheiro de entrada.

De seguida, os ficheiros foram convertidos novamente para [WAV](#), utilizando o Audacity.

### 3.1.3 GSM

A codificação para [GSM](#) só suporta um modo (71kbps), portanto só se obteve um ficheiro desse tipo.

Utilizou-se o Audacity para a conversão do ficheiro para [GSM](#) e, posteriormente, para [WAV](#).

## 3.2 Ferramenta em Python

Para comparar os ficheiros obtidos, desenvolveu-se uma ferramenta em Python que calcula:

- A gama dinâmica.
- O desvio entre os ficheiros codificados e o original, recorrendo a [RMSD](#).

A ferramenta recebe o nome de um ficheiro de áudio como argumento e quebra esse ficheiro em partes, cada uma contendo 1 frame. De seguida, lê o valor de cada parte, em dB, copiando-o para uma lista. Depois calcula valores, tais como o máximo, o mínimo, a gama dinâmica e o [RMSD](#) em relação ao ficheiro original. Por fim, copia esses valores para um ficheiro de texto.

O código pode ser encontrado na íntegra na [Apêndice A](#), onde existe também uma análise mais detalhada do mesmo.

## Capítulo 4

# Análise de Resultados

### 4.1 Gama Dinâmica

A gama dinâmica é o rácio entre o valor máximo e mínimo obtido. No caso dos dB, é uma escala que contém valores negativos. Portanto este rácio funciona da seguinte forma:

**Se é positivo** - Quanto maior for o rácio, maior será a gama dinâmica (e vice-versa).

**Se é negativo** - Quanto menor for o rácio, maior será a gama dinâmica (e vice-versa).

As gamas dinâmicas obtidas foram comparadas em forma de gráfico de barras.

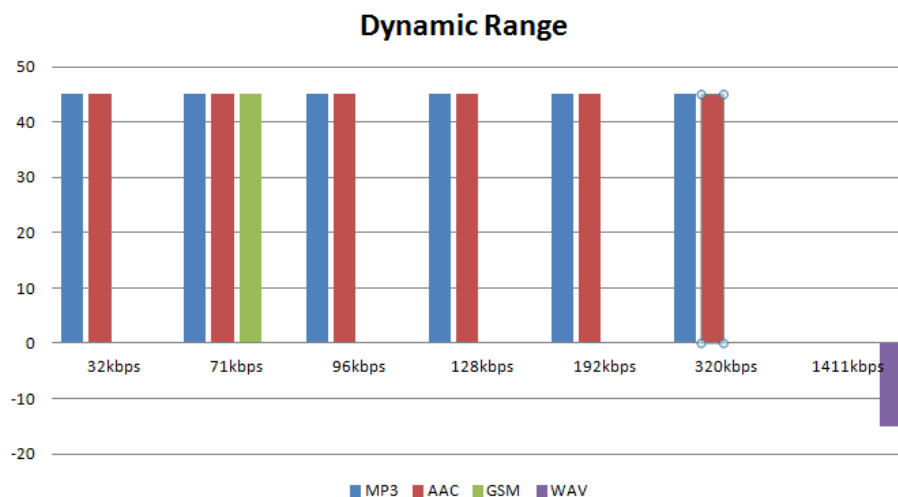


Figura 4.1: Gama Dinâmica

Pode-se observar que, nos ficheiros codificados, a gama dinâmica é igual para todos os ficheiros. Porém, é diferente do ficheiro original, que, como possui

valores de dB negativos, tem uma gama dinâmica superior.

## 4.2 RMSD

A raiz quadrada da média do quadrado das diferenças é um método de calcular a diferença entre valores esperados e valores obtidos.

Quanto menor for o **RMSD**, mais próximo estará o ficheiro do original.

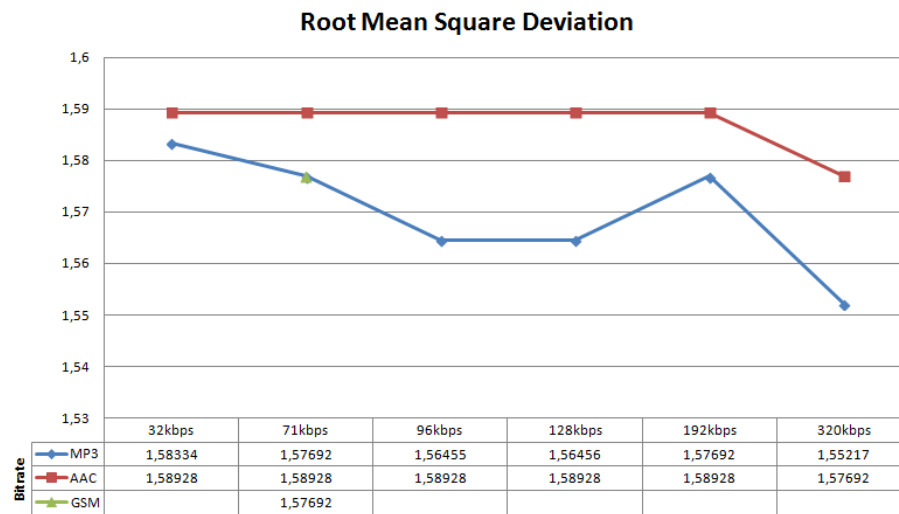


Figura 4.2: RMSD

Ao observar o **Figura 4.2**, verifica-se que os ficheiros **MP3** são mais próximos que o original, no que toca aos valores tidos em conta.

Observa-se ainda que, para *bitrates* baixos, os ficheiros **AAC** não se tornam mais próximos do original, sendo o **RMSD** igual, excepto para o ficheiro codificado a 320kbps.

Verifica-se também que o ficheiro **GSM** é tão próximo do original como o **MP3** com *Bitrate* correspondente.

## 4.3 Tamanho do ficheiro

Foi também verificado o tamanho dos ficheiros **MP3**, **AAC** e **GSM**.

	32kbps	71kbps	96kbps	128kbps	192kbps	320kbps
MP3	352KB	879KB	1055KB	1406KB	2109KB	3515KB
AAC	370KB	798KB	1072KB	1423KB	2126KB	3531KB
GSM	-	787KB	-	-	-	-

Tabela 4.1: Tamanhos dos ficheiros

Podemos verificar na [Tabela 4.1](#) que os ficheiros [MP3](#), no geral, são mais pequenos que os ficheiros [AAC](#).

No caso em que o [GSM](#) está presente, o ficheiro que tem menor tamanho é o [GSM](#).

O ficheiro original(WAV) tem 15,1MB, como já referido no [Capítulo 3](#).

## Capítulo 5

# Conclusão

Comparando os resultados, obtemos as seguintes conclusões:

- A gama dinâmica não é afectada, no que diz respeito à variação de [Bitrate](#).
- Os ficheiros [MP3](#) são mais recomendados que os [AAC](#) para todos os *bitrates*, uma vez que são mais próximos do ficheiro original e ligeiramente mais pequenos.
- Quando se trata de *bitrates* muito pequenos (na ordem dos 71kbps), os ficheiros [GSM](#) são melhores, uma vez que oferecem a mesma qualidade que os [MP3](#), ocupando muito menos espaço.



# Glossário

**AAC** - abreviação de Advanced Audio Coding, um formato de compressão de áudio digital.. [4](#), [7-9](#), [12-14](#)

**Bitrate** - n.º de bits convertidos/processados por unidade de tempo plural. [5](#), [8](#), [9](#), [12](#), [14](#)

**GSM** - formato de áudio, muito utilizado em telemóveis.. [4](#), [7](#), [9](#), [12-14](#)

**MP3** - abreviação de MPEG Layer 3, um formato de compressão de áudio digital.. [4](#), [7-9](#), [12-14](#)

**RMSD** - Root Mean Square Deviation(raiz quadrada da média do quadrado das diferenças. [5](#), [10](#), [12](#)

**WAV** - WAVEform audio file format. [5](#), [8](#), [9](#)

# Bibliografia

- [1] Python. Python Documentation. <http://www.python.org/doc/>. [Acedido em 10 de Maio de 2014].
- [2] Notepad++. Notepad++. <http://notepad-plus-plus.org/>. [Acedido em 10 de Maio de 2014].
- [3] Audacity. Audacity. <http://audacity.sourceforge.net/>. [Acedido em 10 de Maio de 2014].
- [4] Unknown. FDK ACC Encoder. [http://audiophilesoft.ru/load/coders\\_utils/fdkaac/7-1-0-80](http://audiophilesoft.ru/load/coders_utils/fdkaac/7-1-0-80). [Acedido em 10 de Maio de 2014].
- [5] Christoph Gohlke. Python Binaries. [www.lfd.uci.edu/~gohlke/pythonlibs/](http://www.lfd.uci.edu/~gohlke/pythonlibs/). [Acedido em 10 de Maio de 2014].
- [6] Peter Pawlowski. Foobar2000. <http://www.foobar2000.org/>. [Acedido em 10 de Maio de 2014].
- [7] Team Shangai Alice e ZUN. Touhou Official Website. <http://www16.big.or.jp/~zun/top.html>. [Acedido em 10 de Maio de 2014].

# Apêndice A

## Comparador

### A.1 Código

```
1  import wave
2  import sys
3  from scipy.io.wavfile import read
4  import numpy
5  import math
6
7  print "Creating files..."
8  wf=wave.open(sys.argv[1], 'r')
9  samprate, wavdata = read(sys.argv[1])
10
11 print "Chunking..."
12 nframes = wf.getnframes()
13 chunks = numpy.array_split(wavdata, nframes)
14
15 print "Reading data..."
16 dbs = [20*numpy.log10( numpy.sqrt(numpy.mean(chunk**2)) ) for chunk in chunks]
17
18 print "Calculating values..."
19 max = 0;
20 min = 0;
21 i = 0;
22 sum = 0;
23 cont = 0;
24 while i < len(dbs):
25     if dbs[i] > max:
26         if dbs[i] != float("inf"):
27             max = int(dbs[i])
28     if dbs[i] < min:
29         if dbs[i] != float("-inf"):
30             min = int(dbs[i])
31     if dbs[i] != float("-inf") or dbs[i] != float("-inf"):
32         if dbs[i]==dbs[i]:
33             sum += int(dbs[i]*dbs[i])
34             cont += 1;
35     i += 1
```

```

36
37 mrms = numpy.sqrt(sum/cont)
38 if min == 0:
39     min = 1
40 dr = max/min
41 rmsd = numpy.sqrt(((38.8458491991 - mrms) ** 2).mean())
42
43 lines = []
44 lines.append("File name: " + sys.argv[1])
45 lines.append("\n")
46 lines.append("Max: " + str(max) + "\n")
47 lines.append("Min: " + str(min) + "\n")
48 lines.append("Dynamic Range: " + str(dr) + "\n")
49 lines.append("Mean RMS: " + str(mrms) + "\n")
50 lines.append("\n")
51 lines.append("RMSD: " + str(rmsd) + "\n")
52
53 print "Copying to file..."
54 f = open("file.txt", "a")
55 f.writelines(lines)

```

## A.2 Descrição

**1-5** - Importação dos módulos necessários.

**6-9** - Abertura dos ficheiros WAVE com os módulos wave e scipy.

**11-13** - O ficheiro é quebrado em pedaços (1 frame por pedaço)

**14-16** - Os valores de dB dos respectivos frames são retirados para uma lista.

**18-41** - Os valores pretendidos são calculados, percorrendo a lista com os dBs.

**41** - O valor 38.845891991 corresponde ao valor RMS do ficheiro original, que foi obtido através do código presente na [Apêndice B](#)

**42-51** - É criada uma lista de Strings, que vão ser copiadas para um ficheiro de texto.

**53-55** - A lista é copiada para o ficheiro.

## Apêndice B

# Ferramenta

```
1 import wave
2 import sys
3 from scipy.io.wavfile import read
4 import numpy
5 import math
6
7 print "Creating files..."
8 wf=wave.open(sys.argv[1], 'r')
9 samprate, wavdata = read(sys.argv[1])
10
11 print "Chunking..."
12 nframes = wf.getnframes()
13 chunks = numpy.array_split(wavdata, nframes)
14
15 print "Reading data..."
16 dbs = [20*numpy.log10( numpy.sqrt(numpy.mean(chunk**2)) ) for chunk in chunks]
17
18 print "Calculating values..."
19 max = 0;
20 min = 0;
21 i = 0;
22 sum = 0;
23 cont = 0;
24 while i < len(dbs):
25     if dbs[i] > max:
26         if dbs[i] != float("inf"):
27             max = int(dbs[i])
28     if dbs[i] < min:
29         if dbs[i] != float("-inf"):
30             min = int(dbs[i])
31     if dbs[i] != float("-inf") or dbs[i] != float("-inf"):
32         if dbs[i]==dbs[i]:
33             sum += int(dbs[i]*dbs[i])
34             cont += 1;
35     i += 1
36 mrms = numpy.sqrt(sum/cont)
37 if min == 0:
38     min = 1
```

```
39 dr = max/mrms
40
41 lines = []
42 lines.append("File name: " + sys.argv[1])
43 lines.append("\n")
44 lines.append("Max: " + str(max) + "\n")
45 lines.append("Min: " + str(min) + "\n")
46 lines.append("\n")
47 lines.append("Dynamic Range: " + str(dr) + "\n")
48 lines.append("Mean RMS: " + str(mrms) + "\n")
49
50
51 print "Copying to file..."
52 f = open("file.txt", "a")
53 f.writelines(lines)
54
55 print "Done!"
```