

Instituto Superior de Engenharia de Lisboa

Codificação de Sinais Multimédia

2º Semestre de 2017/2018

O ficheiro com o relatório do trabalho (em formato PDF) e os ficheiros de código implementado devem ser encapsulados num ficheiro ZIP com o número do grupo e submetidos no Moodle até 15 de Abril.

Este Trabalho explora os conceitos de compressão de dados sem perdas baseados na teoria da informação.

1. Elabore uma função ("gera_huffman") que gere uma tabela com o código binário para cada símbolo de um dado conjunto, usando o método de Huffman. Esta função deve ter como parâmetros de entrada um conjunto de símbolos e as suas probabilidades (ou em alternativa pode usar o número de ocorrências de cada símbolo, dado pelo seu histograma). Também pode em alternativa gerar não uma tabela mas outra estrutura de dados com os códigos pretendidos.
2. Elabore uma função ("codifica") que dada uma mensagem (sequência de símbolos) e a tabela do ponto anterior, retorne uma sequência de bits com a mensagem codificada.
3. Elabore uma função ("descodifica") que dada uma sequência de bits (mensagem codificada) e a tabela do ponto 1, retorne uma sequência de símbolos (mensagem descodificada).
4. Elabore uma função ("escrever") que dada uma sequência de bits (mensagem codificada) e o nome do ficheiro, escreva a sequência de bits para o ficheiro.
5. Elabore uma função ("ler") que dado o nome do ficheiro, leia uma sequência de bits (mensagem codificada) contida no ficheiro.
6. Teste as funções elaboradas usando para o efeito vários ficheiros com diferentes tipos de média:

IMAGEM: Use para o efeito a imagem "Lena.tif" em tons de cinzento.

TEXTO: Use os ficheiros "ubuntu_server_guide.pdf" e "ubuntu_server_guide.txt".

ÁUDIO: Use o ficheiro "HenryMancini-PinkPanther30s.mp3".

MIDI: Use o ficheiro "HenryMancini-PinkPanther.mid".

ECG: Eletrocardiograma - use o ficheiro "ecg.txt".

- a) Gere o código usando a função realizada no ponto 1. Meça o tempo que demora a função.
 - b) Meça a entropia e o número médio de bits por símbolo. Calcule a eficiência.
 - c) Faça a codificação da mensagem contida no ficheiro (usando a função realizada no ponto 2). Meça o tempo que a função demora a fazer a codificação.
 - d) Grave um ficheiro com a mensagem codificada, usando a função realizada no ponto 4. Veja o tamanho do ficheiro.
 - e) Leia do ficheiro o conjunto de bits, usando a função realizada no ponto 5.
 - f) Faça a descodificação da mensagem (usando a função realizada no ponto 3.) Meça o tempo que a função demora a fazer a descodificação.
 - g) Compare a mensagem descodificada com a original e verifique que são iguais (erro nulo).
7. O relatório deve conter uma descrição breve das funções realizadas e uma tabela com todos os resultados extraídos.

Exemplo para o ficheiro com imagem

```
from time import time
from os import path
import cv2
import matplotlib.pyplot as plt

# Lê a imagem em níveis de cinzento
```

```

x = cv2.imread("Lena.tiff",cv2.IMREAD_GRAYSCALE)

# Converte a imagem (matriz) numa sequência de números (array)
xi = x.ravel()

# Calcula o histograma
h, bins, patches = plt.hist(xi,256,[0,256])

# Gera o código de Huffman
t0 = time()
tabela_codigo = gera_huffman(np.arange(0,256),h)
t1 = time()
print "time:", t1-t0

# Codifica e grava ficheiro
seq_bit0 = codifica(xi,tabela_codigo)
escrever(seq_bit0, filename)
t2 = time()
print "time:", t2-t1

# Lê ficheiro e descodifica
seq_bit1 = ler(filename)
yi = descodifica(seq_bit1,tabela_codigo)
t3 = time()
print "time:", t3-t2

size_ini = path.getsize("filename original image")
size_end = path.getsize("filename compressed")
print "taxa: ", 1.* size_ini / size_end

plt.show()
cv2.waitKey(0)
plt.close("all")
cv2.destroyAllWindows()

```