Universidade de Coimbra
Departamento de Engenharia Informática
Tecnologia da Informação

Relatório Trabalho 1

Autores:
Gonçalo Amaral
Miguel Machado
Teresa Salazar

Professor:
Rui Pedro Pinto de Carvalho e Paiva

Outubro, 2016

1 Problema 1

Este problema pedia para:

> Escreva uma rotina em Matlab que dada uma fonte de informação p com um alfabeto A=a1,. . . ,an determine e visualize o histograma de ocorrência dos seus símbolos.

Criou-se a função *displayHistograma(p, A)*, sendo p uma fonte de informação representado por uma matriz, e A o alfabeto, representado por um *cell array*. Posteriormente iremos explicar o porquê de termos escolhido representar o alfabeto desta maneira.

Para cada tipo de fonte decidimos criar o tipo de histograma mais adequado, recorrendo sempre à função *histogramaOcurrencias(p, A)* para gerar o tipo de dados adequados.
Para o caso da fonte introduzida ser uma imagem recorremos à função *imhist()*, que recebe como argumento uma matriz que representa uma imagem em escala de cinzentos, correspondendo 0 a branco e 255 a preto. Este tipo de função já tem em conta que o alfabeto serão os números inteiros de 0 a 255. Desta forma, não é necessário passar o alfabeto como argumento. Neste caso, a função *histogramaOcurrencias(p, A)* não irá fazer quaisquer alterações na fonte.

De seguida, para o caso em que a fonte é composta por caracteres, utilizámos a função *histogram()* que recebe um tipo de objecto específico. Este tipo de objecto do tipo *categorical* será gerado pela função *histogramaOcurrencias(p, A)*. Escolhemos este tipo de objecto pois tem uma particularidade muito curiosa em combinação com um alfabeto do tipo *cell array*. Primeiro criámos um *cell array* com o código ASCII dos caracteres presentes no alfabeto. Recorrendo à função *categorical()* podemos gerar um objecto do tipo *categorical* que associa cada valor ASCII ao correspondente caracter. E convertendo a fonte numa matriz de números, ao a passarmos como argumentos esta fonte convertida, o alfabeto convertido e o alfabeto original na função *histogram()*, iremos obter um histograma apenas com os elementos da fonte convertida, presentes no alfabeto convertido. A cada elemento do alfabeto convertido irá corresponder um elemento do alfabeto original, que será apresentado na label do x.

Finalmente, o último tipo de fonte que é possível receber é uma matriz de números. Neste caso apenas é passada como argumento na função *histogram* uma matriz com os elementos da fonte que estavam presentes no alfabeto.

2 Problema 2

Para este problema foi-nos pedido para:

> Determinar o limite mínimo teórico para o número médio de bits por símbolo

Este limite mínimo teórico equivale a calcular a entropia. Para calcular a entropia usámos a função *entropia(p, A)*, que faz uso da seguinte fórmula:

$$H(A) = -\sum_{i=1}^{k} P(a_i) \log_2 P(a_i) \tag{1}$$

3 Problema 3

Neste problema pede-se para calcular a distribuição estatística e a entropia, "número médio de bits", de diversas fontes de informação (imagens, sons e texto).

Recorreu-se à função *distribuicaoEstatisticaEntropia(p, A)* que chama as funções já criadas nos pontos 1 e 2 de modo a resolver este problema.

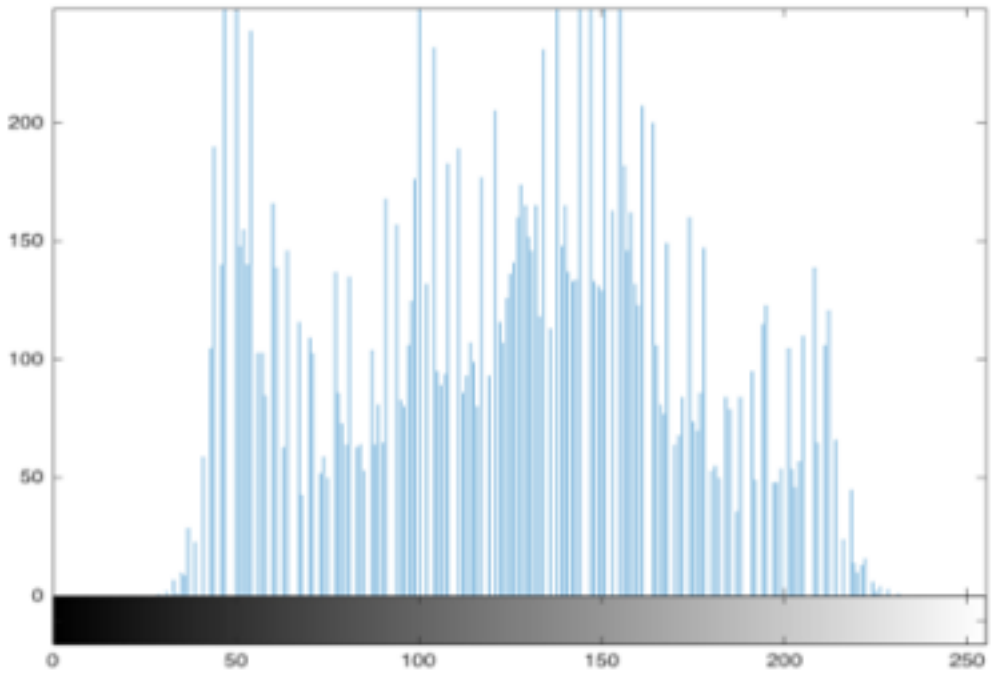

Figure 1: Distribuição estatística de Lena.bmp

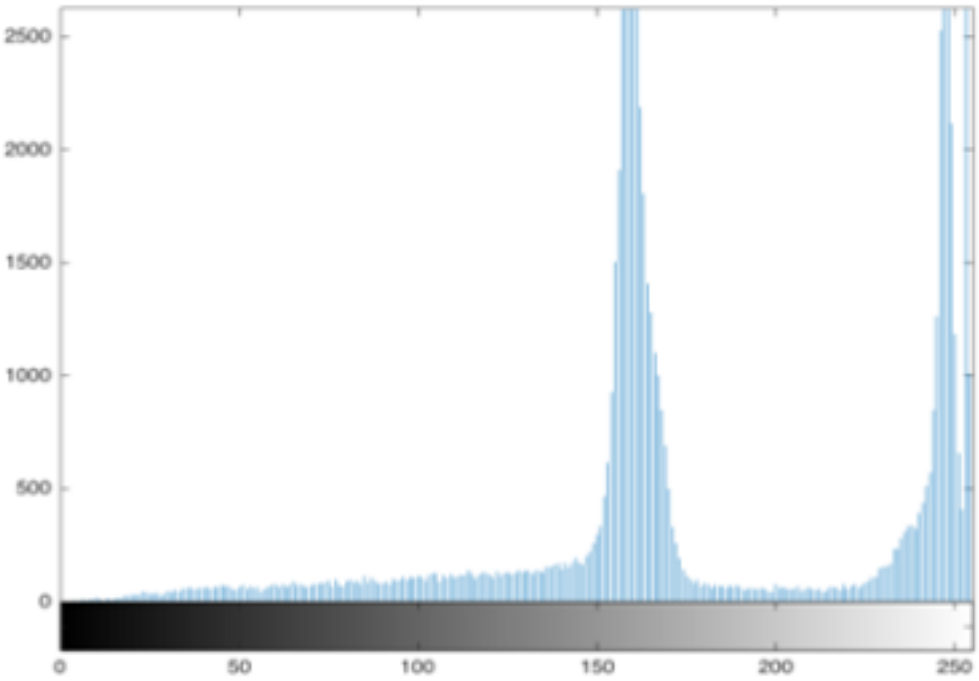

Figure 2: Distribuição estatística de CT1.bmp

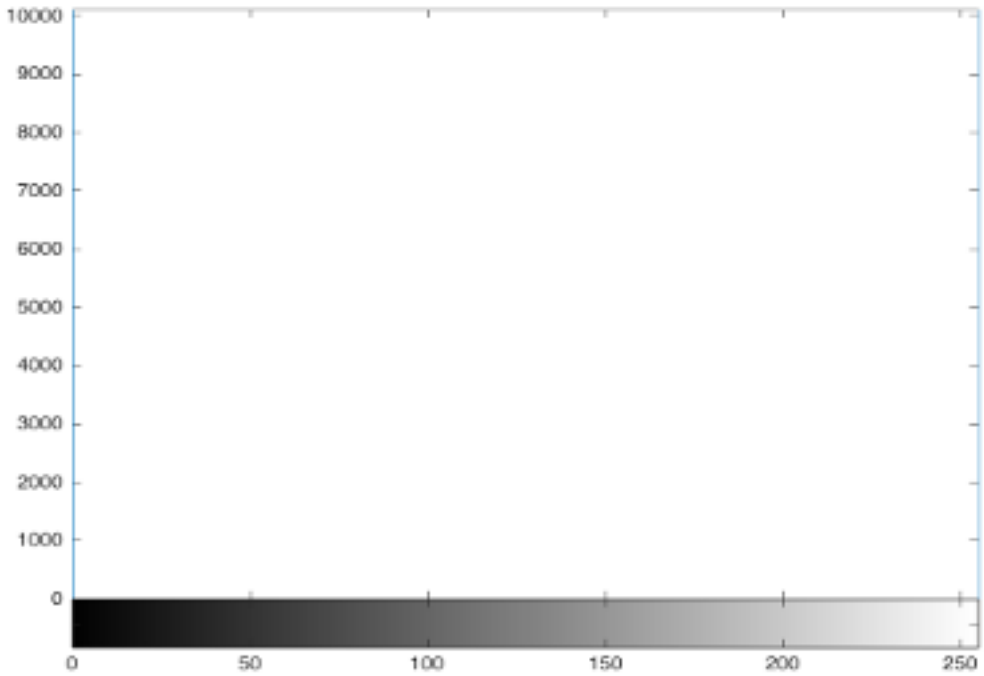

Figure 3: Distribuição estatística de Binaria.bmp

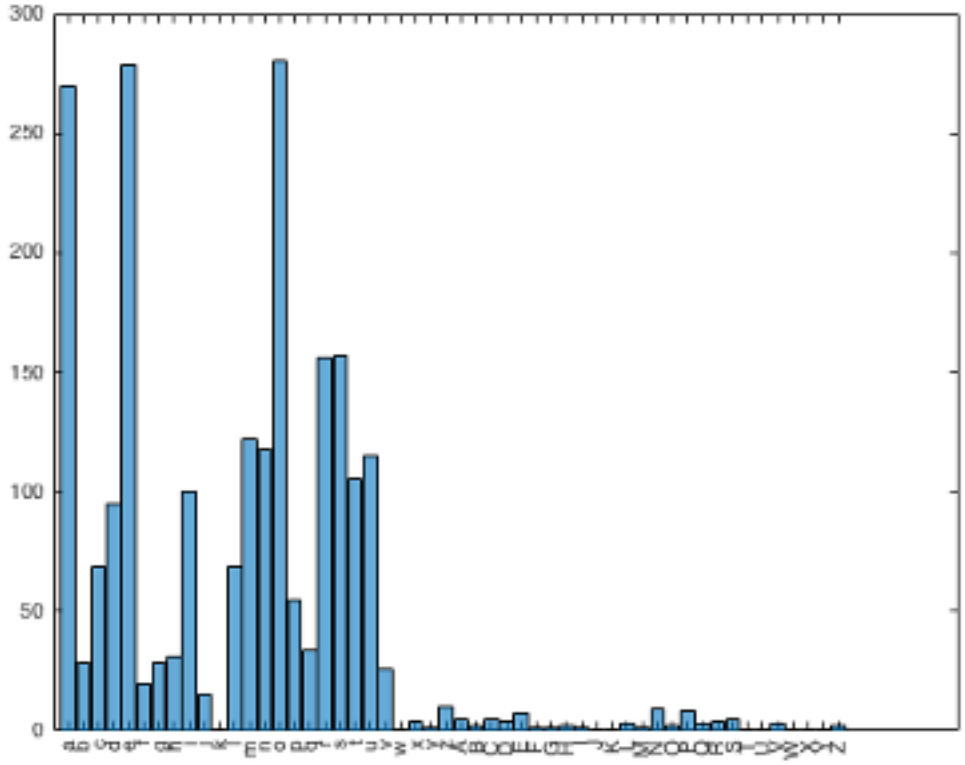

Figure 4: Distribuição estatística de Texto.txt

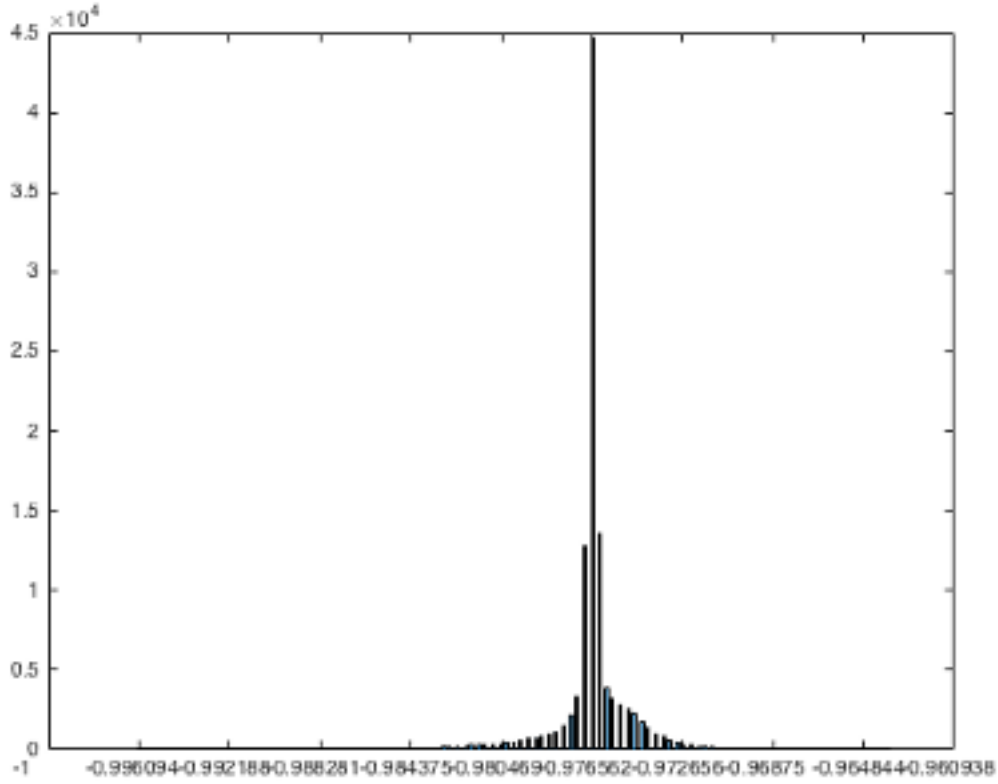

Figure 5: Distribuição estatística de saxriff.wav

Entropia para cada uma das fontes fornecidas:

- Lena.bmp: 6.915336
- CT1.bmp: 5.972234
- Binaria.bmp: 0.9755266
- Texto.txt: 3.445302
- saxriff.wav: 3.535599

Concluindo, a entropia é o limite minímo para o número médio de bits por símbolo, logo é possível comprimir cada uma das fontes de forma não destrutiva, sendo a compressão máxima o valor da entropia.

4 Problema 4

Pediu-se neste problema que:

> Usando as rotinas de codificação de Huffman que são fornecidas, determine o número médio de bits por símbolo para cada uma das fontes de informação usando este código.

As rotinas de Huffman usam a seguinte fórmula para calcular a entropia:

$$l = \sum_{i=1}^{k} p(a_i) l_i \tag{2}$$

Entropia para cada uma das fontes fornecidas:

- Lena.bmp: 6.942505
- CT1.bmp: 6.007546
- binaria.bmp: 1.00000
- texto.txt: 4.022405
- saxriff.wav: 3.561658

Os valores de entropia obtidos são próximos dos valores obtidos no Problema 3 pois:

$$H(S) <= l < H(S) + 1 \tag{3}$$

Conclui-se também que o cada comprimento de cada código é bastante diferente dos outros pois esperam-se códigos menores para os símbolos mais comuns e maiores para os símbolos menos comuns.

Quanto à variância, é possível reduzi-la colocando os símbolos combinados na lista usando a ordem mais elevada possível. Na criação de buffers onde é conveniente que a taxa de enchimento seja aproximadamente constante, isto torna-se útil.

5 Problema 5

Neste problema pede-se para:

> Repita o Problema 3 aplicando agrupamentos de símbolos, isto é, admitindo que cada símbolo é na verdade uma sequência de dois símbolos contíguos.

Para este exerício utilizamos a função *agrupamentoSimbolos(filename)*, que é semelhante à função utilizada no Problema 3. No entanto, aqui o alfabeto será diferente, pois é necessário criar todos os pares de elementos do alfabeto. Para tal, utizámos a função *criarAlfabetoDePares(alf)*. Como agora temos pares de símbolos, as funções criadas no exercício 3 não poderão ser usadas. Para resolver este problema convertemos a nossa fonte de pares e o alfabeto de pares em objectos simples. Para tal usamos a funçao *labelObj* que cria um objecto novo, no qual cada linha da fonte de pares é convertida para a linha onde o mesmo par é encontrado no alfabeto. Após simplificada a fonte, simplificamos o alfabeto de pares, tornando-se cada linha do alfabeto de pares original no número da mesma linha. Exemplo:

> Alfabeto original = [1 2]
> Fonte original = [1 2 2 1 2 1]
>
> Alfabeto de pares = [1 1; 1 2; 2 1; 2 1]

Fonte de pares = [1 2; 2 1; 2 1]

Alfabeto de pares simplificado = [1 2 3 4]
Fonte de pares simplidicada = [2 3 3]

O agrupamento de símbolos permite melhorar o valor da entropia, pois:

$$H(X,Y) = H(X) + H(Y|X) <= H(X) + H(Y)$$

Assim, aumenta a eficiência pois baixa-se o comprimento médio, dado que assim as palavras mais compridas são menos.

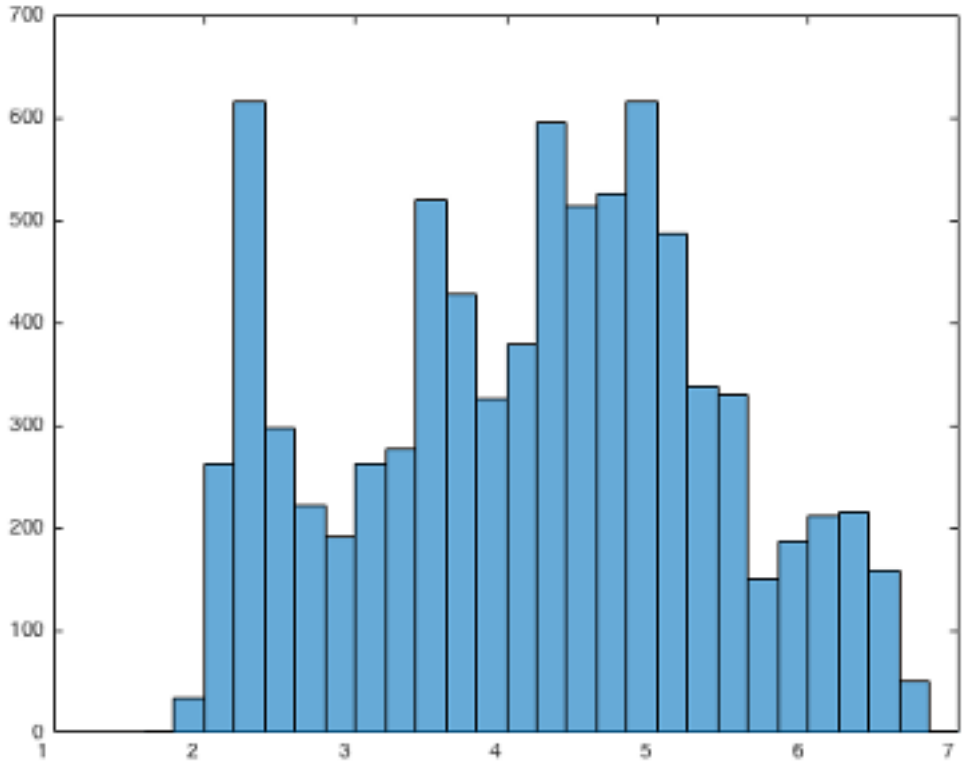

Figure 6: Distribuição estatística de Lena.bmp

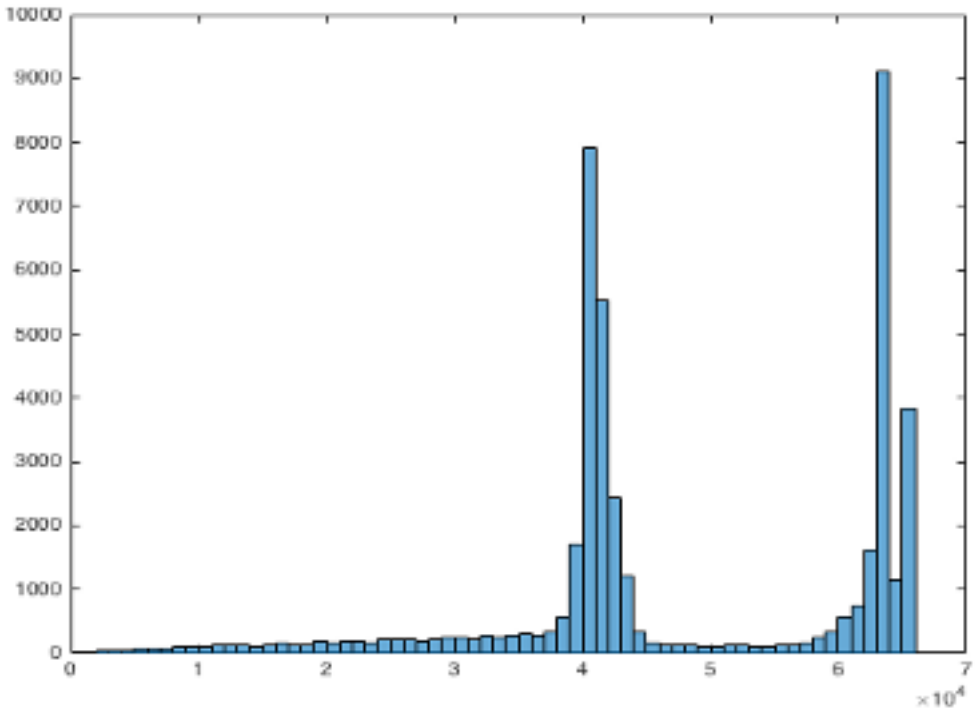

Figure 7: Distribuição estatística de CT1.bmp

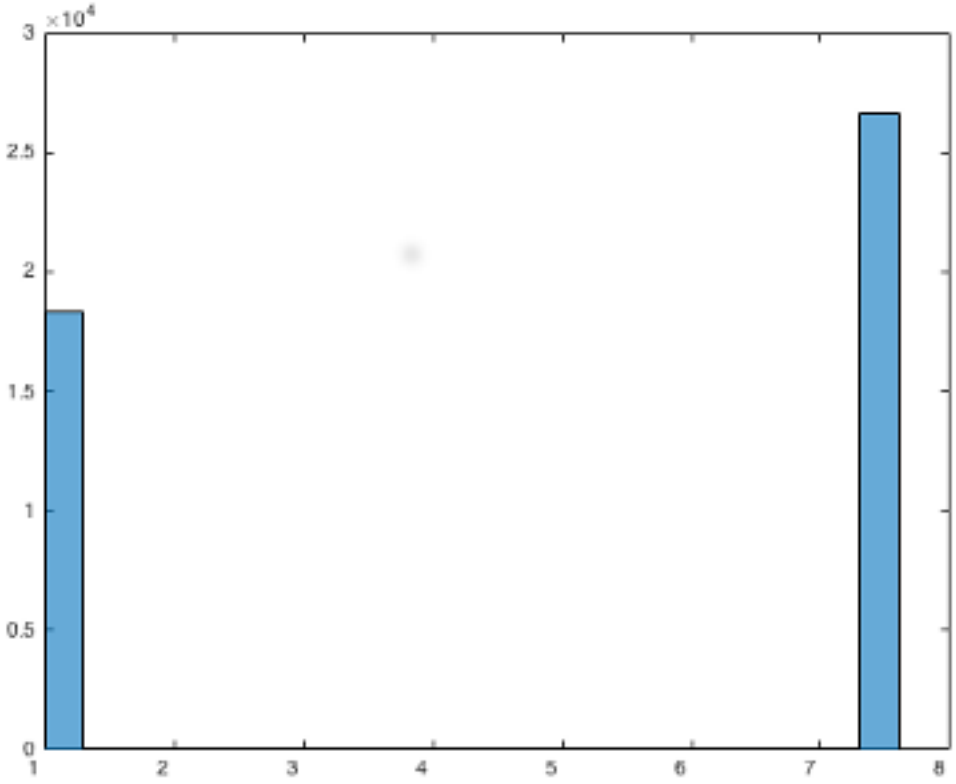

Figure 8: Distribuição estatística de Binaria.bmp

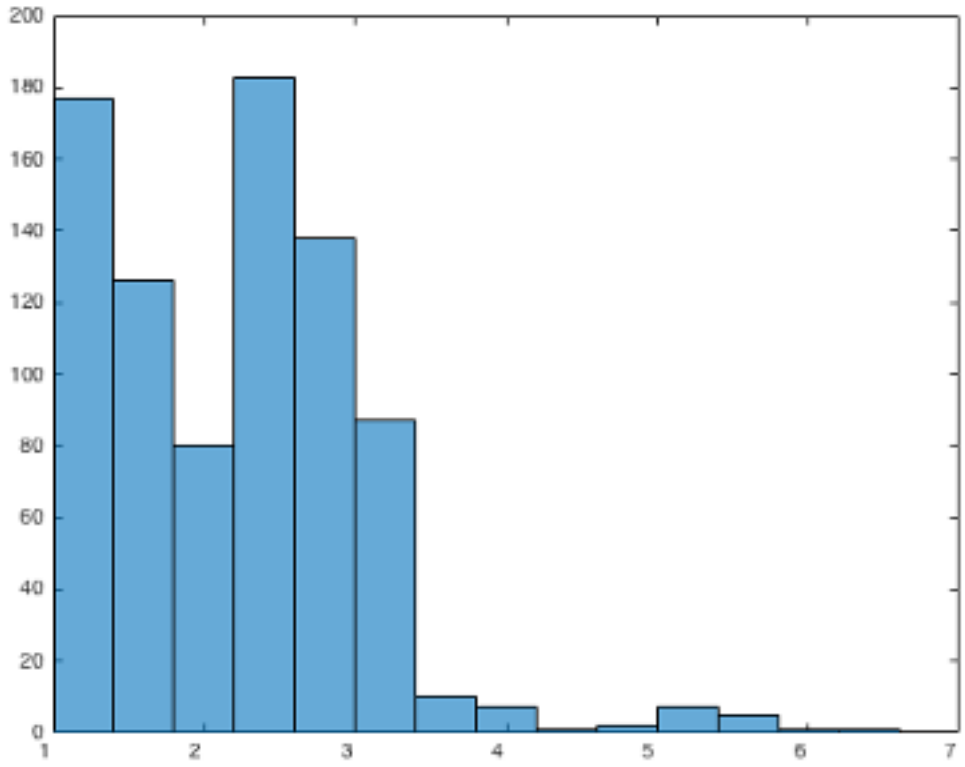

Figure 9: Distribuição estatística de texto.txt

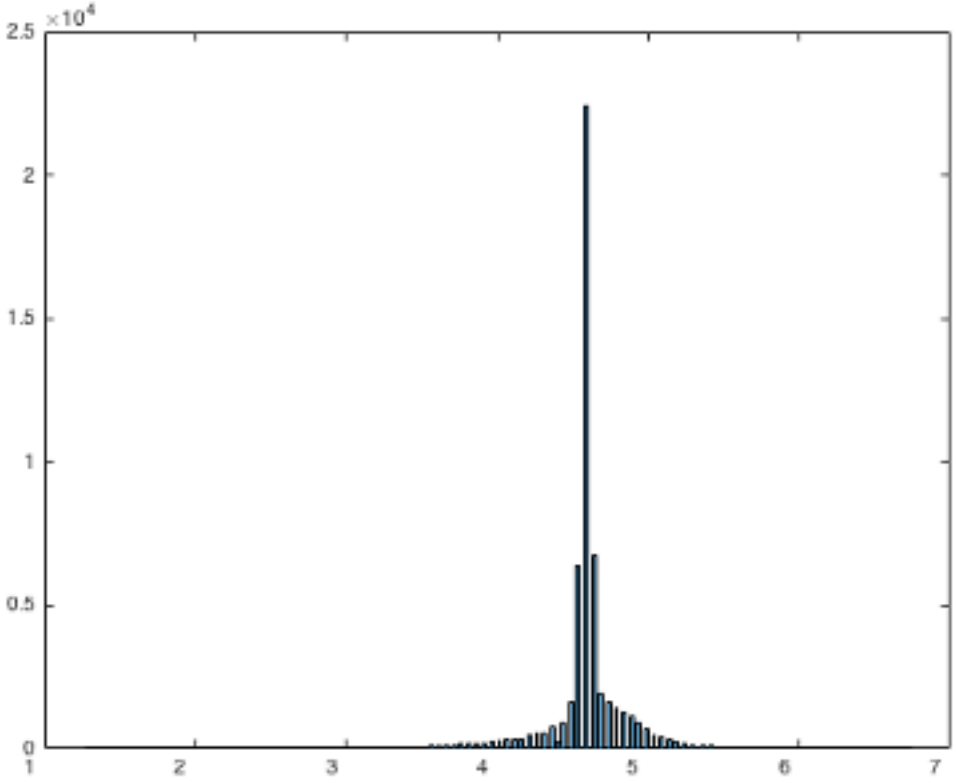

Figure 10: Distribuição estatística de saxriff.wav

Entropia para cada uma das fontes fornecidas:

- Lena.bmp: 4.443608
- CT1.bmp: 4.203238
- Binaria.bmp: 0.975282
- Texto.txt: 2.767432
- saxriff.wav: 3.546061

Concluindo, é sempre preferível estudar várias variáveis em simultâneo (agrupamentos).

6 Problema 6

Pediu-se na primeira alínea problema que:

> Escreva uma rotina em Matlab que, dada a query, o target, um alfabeto A=a1,. . . ,an e o passo, devolva o vector de valores de informação mútua em cada janela.

Assim usámos a função *informacaoMutua(query, target, alf, step)* que percorre a target e que vai chamando a função *calcMutualInf(query, partTarget, alf)* que calcula a informação mútua entre cada parte da target e a query. O output foi o que se esperava:

infoMutua = [2.1219 1.9219 1.6464 2.1710 1.9710 1.7710 2.0464 2.1219 2.3219 2.5219 2.2464 2.2464 2.2464 2.2464 2.4464 2.4464 2.5219 2.7219 2.5219 2.3219 2.3219 2.1219 2.3219 2.3219 2.1219 2.0464 2.0464 2.0464 2.0464 2.0464 2.3219 2.3219 2.0464 2.1219 2.3219 1.8464 1.7710 2.0464 2.0464 2.0464 2.3219].c
Para a alínea b do problema pediu-se para:

> Usando o ficheiro "guitarSolo.wav" como query, determine a variação da informação mútua entre este e os ficheiros "target01 - repeat.wav" e "target02 – repeatNoise.wav". Defina um passo com valor de $\frac{1}{4}$ do comprimento do vector da query (valor arredondado).

Como o target01 e o target02 apenas diferem no facto de o segundo ter ruído, pode-se concluir que os valores de informação mútua do primeiro target sejam superiores.

Para a alínea c do problema pediu-se para:

> Pretende-se agora simular um pequeno simulador de identificação de música. Usando o ficheiro "saxriff.wav" como query e os ficheiros Song*.wav como target: - determine a evolução da informação mútua para cada um dos ficheiros - calcule a informação mútua máxima em cada um deles - e, finalmente, apresente os resultados da pesquisa, seriados por ordem decrescente de informação mútua. Defina um passo com valor de $\frac{1}{4}$ da duração da query.

Os resultados obtidos foram os seguintes:

- Song06.wav: 3.535599
- Song07.wav: 3.535599
- Song05.wav: 0.519376
- Song02.wav: 0.112439
- Song04.wav: 0.092564
- Song03.wav: 0.114060
- Song01.wav: 0.077952

Assim, o ficheiro Song06.wav é o mais semelhante com o ficheiro saxriff.wav.