Teoria da Informação

UNIVERSIDADE DE COIMBRA

Entropia, Redundância e Informação Mútua

Dário Filipe Torres Félix - 2018275530

Jonas Vieira Rua - 2016218077

José Marcelo Marques da Cruz - 2017255151

Introdução

O presente trabalho tem como objetivo adquirir competências nos campos da Teoria de Informação, em específico, informação, redundância, entropia e informação mútua. O código respetivo a cada um dos pontos do trabalho foi desenvolvido na linguagem MATLAB.

Notas: A leitura dos ficheiros é feita ao iniciar o programa, sendo a informação retirada usada depois nos diferentes exercícios, como mostra a **Figura 1**:

```
%%  LANDSCAPE
landscape = imread("data/landscape.bmp");
infoLandscape = imfinfo("data/landscape.bmp");
agLandscape = agrupaSimbolos(landscape, infoLandscape.BitDepth);
%%  MRI
mri = imread("data/MRI.bmp");
infoMri = imfinfo("data/MRI.bmp");
agMri = agrupaSimbolos(mri, infoMri.BitDepth);
%%  MRI_BIN
mri_bin = imread("data/MRIbin.bmp");
infoMri_bin = imfinfo("data/MRIbin.bmp");
agMri_bin = agrupaSimbolos(mri_bin, infoMri_bin.BitDepth);
%%  SOUNDMONO
soundMono = audioread("data/soundMono.wav");
infoSoundMono = audioinfo("data/soundMono.wav");
agSoundMono = agrupaSimbolos(soundMono, infoSoundMono.BitsPerSample);
%%  LYRICS
lyrics = fileread("data/lyrics.txt");
lyrics = lyrics(isstrprop(lyrics, "alpha"));
lyrics = double(lyrics);
agLyrics = agrupaSimbolos(lyrics,7);
```

Figura 1

Não foi incluído código para 1, sendo que foi usada a função de histograma do MATLAB.

Ponto 3:

Pegando na funções elaborada no ponto **2**, e com recurso à função **histogram** do MATLAB, espera-se o cálculo do número médio de bits por símbolo e respetiva distribuição estatística das seguintes fontes:

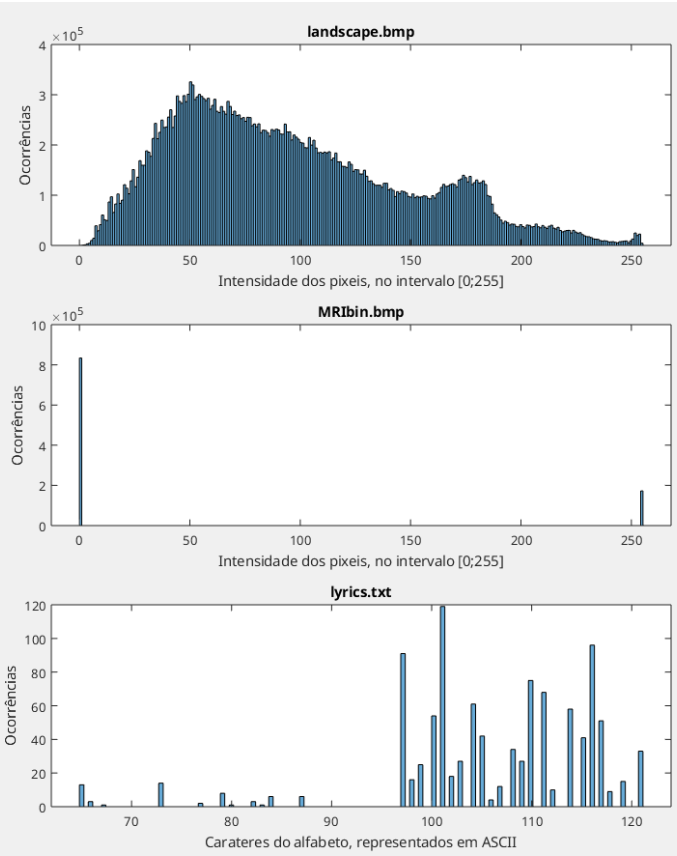
- landscape.bmp;
- MRI.bmp;
- MRIbin.bmp;
- soundMono.wav;
- lyrics.txt.

Esta é feita recorrendo à **calcEntropia**, que recebe uma fonte, "**fonte**" e a quantidade de elementos constituintes do alfabeto da mesma, "**bins**". Este número é variável, dependente da fonte (imagem, som, texto).

Os resultados obtidos são os presentes nas **Figuras 2** e **3**:

```
Limite mínimo teórico da média bits/símbolo:
        landscape.bmp: 7.60691
        MRI.bmp: 6.86054
        MRIbin.bmp: 0.66108
        soundMono.wav: 4.06573
        lyrics.txt: 4.41071
>>
```

Figura 2



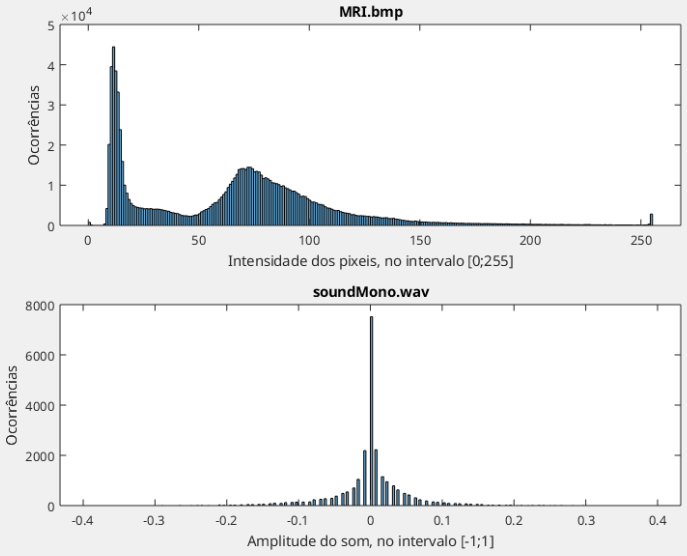


Figura 3

A entropia é tanto maior, quanto mais próximos forem os valores das probabilidades de ocorrência de cada símbolo do alfabeto de dada fonte, atingindo o valor máximo quando os símbolos são equiprováveis.

Através da análise dos valores obtidos, e dos histogramas apresentados, facilmente se verifica esta condição, tendo, por exemplo, em relação às imagens, maior entropia o gráfico de **landscape.bmp** (valores mais distribuídos, maior entropia), e menor o gráfico de **MRIbin.bmp** (apenas dois valores, 0 e 255, entropia baixíssima).

"**Será possível comprimir cada uma das fontes de forma não destrutiva? Se sim, qual a compressão máxima que se consegue alcançar?**"

R: A compressão máxima possível é limitada inferiormente pelo valor da entropia, como diz o Teorema de Shannon. Assim sendo, é possível comprimir as fontes de forma não destrutiva (sendo, neste caso, os valores apresentados na **Figura 2**).

Ponto 4:

Usando, agora, rotinas de codificação de Huffman, pretende-se determinar o número médio de bits por símbolo para cada uma das fontes referidas no ponto **3**. Isto é feito através da chamada da função **compHuffman**, que recebe uma fonte de informação, "**fonte**" e o número de elementos no alfabeto da mesma, "**bins**".

```
function [l, v] = compHuffman(fonte, bins)
    nSimbolos = size(fonte, 1) * size(fonte, 2);

    t = histcounts(fonte,bins);
    t = t(t~=0); % remove as entradas dos símbolos do alfabeto que não existem na fonte
    t = t/nSimbolos;

    l = hufflen(t);
    v = var(l, t); % calcula a variância do comprimento dos códigos

    l = l*t';
end
```

Figura 4

Foram obtidos os seguintes valores:

```
Média de bits/símbolo usando códigos de Huffman:
        landscape.bmp: 7.6293
                Variância: 0.751616
        MRI.bmp: 6.891
                Variância: 2.19308
        MRIbin.bmp: 1
                Variância: 0
        soundMono.wav: 4.11071
                Variância: 4.33351
        lyrics.txt: 4.44349
                Variância: 1.08206
>>
```

Figura 5

Sabemos que os códigos de Huffman são limitados inferiormente pela entropia, e superiormente pela entropia + 1. Olhando para os valores da figura 5, verifica-se que os resultados obedecem à condição descrita.

"**Será possível reduzir-se a variância? Se sim, como pode ser feito e em que circunstância será útil?**"

R: É possível reduzir a variância, exceto em MRIbin.bmp(uma vez que a variância é 0). Para obter o valor mínimo, aquando da junção dos símbolos de menor probabilidade, dá-se precedência à junção dos símbolos mais simples, antes dos mais complexos, caso ambos tenham a mesma probabilidade. Ou seja, os símbolos combinados são introduzidos na árvore o mais próximo possível da raiz da mesma. É útil para contornar congestionamentos de rede numa transmissão de dados, uma vez que reduz a capacidade de transmissão da rede necessária para transmitir o maior símbolo, uma vez que este é o que determina a largura de banda mínima necessária para que toda a fonte seja transmitida sem interrupções.

Ponto 5:

Em 5, apenas é repetido o que foi realizado no ponto **3**, com a diferença que os símbolos são agrupados 2 a 2 antes de realizar cálculos. Este passo é feito pegando na fonte, seja ela uma coluna, linha, ou matriz, e transformando numa matriz unidimensional **n** x **1** (matriz coluna. No caso de a fonte ser uma coluna, não se altera nada), recorrendo ao comando simples: **fonte = fonte (:)**. Os símbolos são agrupados apenas depois deste passo, recorrendo ao método **agrupaSimbolos**, que recebe uma fonte de informação, "**fonte**", e o número de bits para codificar cada símbolo da mesma, "**bitsPsimbolo**".

```
function f = agrupaSimbolos(fonte, bitsPsimbolo)
    fonte = double(fonte(:));
    f = zeros(ceil(length(fonte)/2), 1);

    if(rem(length(fonte),2) ~= 0)
        fonte = [fonte;0];
    end

    for i = 1:length(f)
        f(i) = fonte(2*i-1)*(2^bitsPsimbolo) + fonte(2*i);
    end
end
```

Figura 6

Após os cálculos, os valores obtidos são os ilustrados na **Figura 7**, e os histogramas estão presentes na **Figura 8**:

```
Limite mínimo teórico da média bits/símbolo (com agrupamentos 2 a 2):
        landscape.bmp: 6.27727
        MRI.bmp: 5.22693
        MRIbin.bmp: 0.400694
        soundMono.wav: 3.3108
        lyrics.txt: 3.65218
>>
```

Figura 7

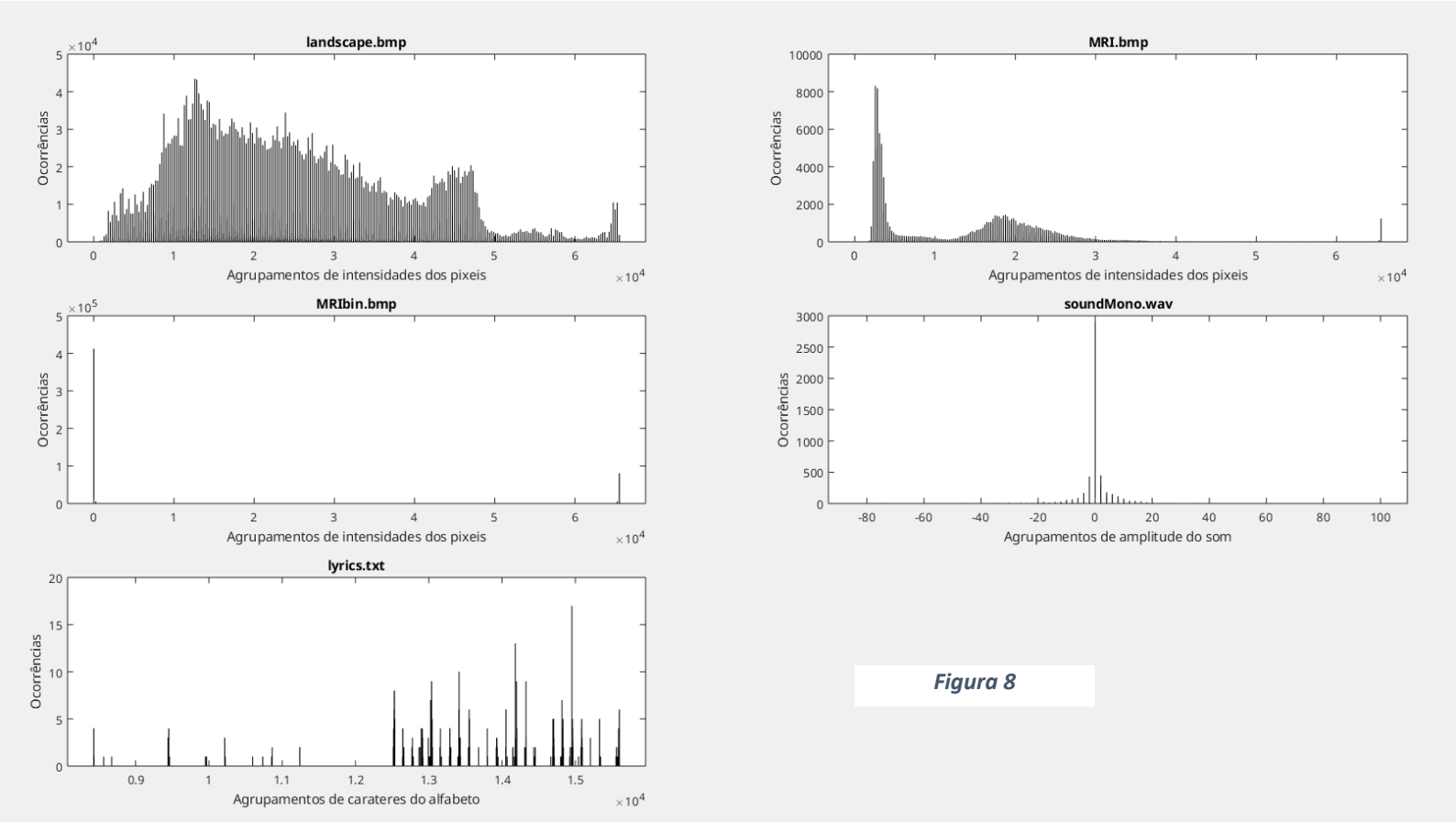


Figura 8

Como esperado, os valores obtidos para a entropia são menores do que no ponto **3**, agrupando os símbolos. Através da análise dos histogramas, verifica-se que, devido aos agrupamentos, há uma maior discrepância nos valores de ocorrência dos "novos" símbolos. As probabilidades destes afastam-se da distribuição uniforme, fazendo, portanto, reduzir a entropia. Isto acontece porque, geralmente, ao saber o contexto (símbolo anterior), as probabilidades alteram-se, pois sabemos que dado símbolo tem maior/menor probabilidade de aparecer antecedido de outro, o que causa a diminuição da incerteza.

Ponto 6:

- a) Como pedido na primeira alínea, foi criada uma rotina **infoMutua,** que recebe uma imagem a pesquisar, "**query**", uma imagem onde pesquisar, "**target**", o número de bits por símbolo, "**bitsPerSymbol**", o intervalo entre janelas sucessivas, "step", e devolve a matriz de valores de informação mútua em cada janela de pesquisa.

- b.i) Usando a função criada na alínea anterior, foram calculados os valores máximos de informação mútua entre a query e os diferentes targets, e as coordenadas da janela onde ocorre o mesmo, obtendo os seguintes resultados:

```
Informação mútua máxima nos 4 primeiros target:
        target_original.bmp: 1.35003, em (x,y): (421, 316)
        target_noise.bmp: 1.19635, em (x,y): (421, 316)
        target_inverted.bmp: 1.35003, em (x,y): (421, 316)
        target_lightning_contrast.bmp: 1.22402, em (x,y): (421, 316)
>>
```

Figura 9

- b.ii) Imagem obtida:

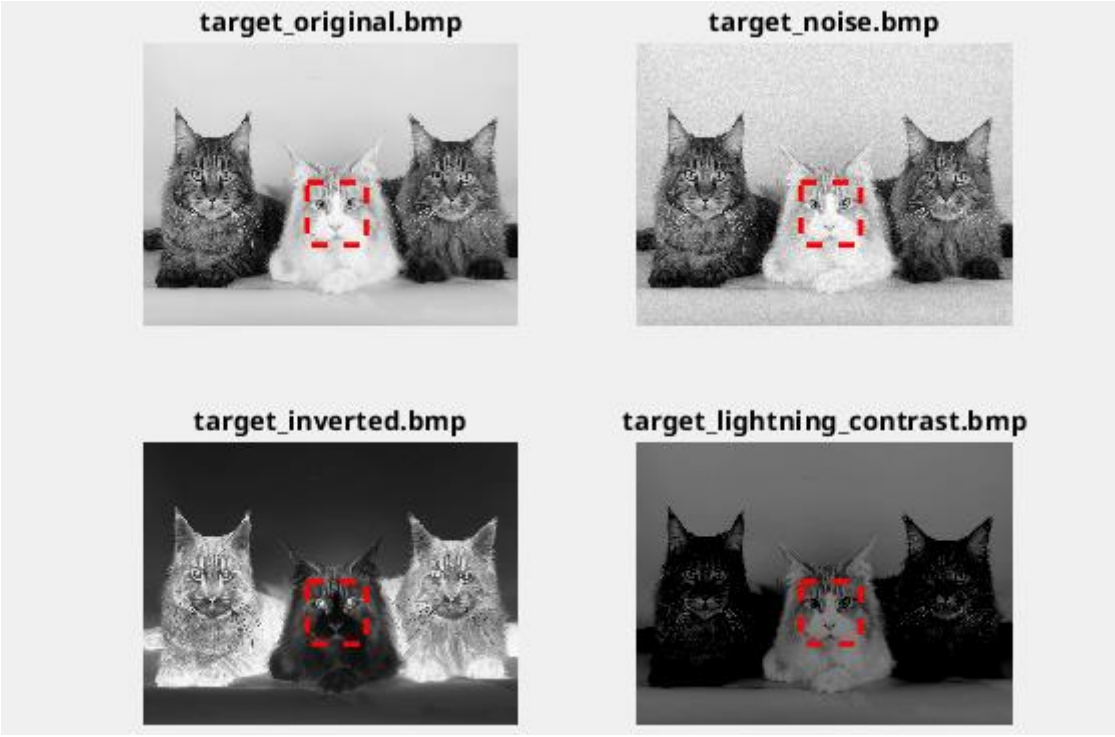


Figura 10

Como facilmente se verifica, em todos os targets foi possível reconhecer e localizar a query, encontrando a face do gato.

Pegando no **target_original.bmp** como referência, verifica-se que:

1. Em **target_inverted.bmp**, quando comparada com a informação mútua em **target_original.bmp,** a informação mútua mantém-se para **target_inverted.bmp**. As alterações causadas à intensidade/cor de cada píxel são uniformes, isto é, todos os píxeis têm o seu valor alterado para (255 – [valor original]); isto mantém a mesma dependência estatística que tínhamos em **target_original.bmp**, exceto que, se antes tínhamos que, para um píxel, de valor *i,* em **query,** havia uma probabilidade *x* de o píxel correspondente na imagem **target_original.bmp** ter valor *j*, agora temos que, para o mesmo píxel em **query,** há a mesma probabilidade *x* de o píxel correspondente em **target_inverted.bmp** tenha o valor 255 – *j*. No entanto isto em nada afeta o cálculo de entropia, portanto, também não irá alterar os valores de informação mútua, pois continuamos a ter os mesmos valores de probabilidade, apenas associados à "cor inversa".

2. Em **target_lightning_contrast.bmp** temos a segunda informação mútua mais alta. Neste caso as alterações feitas aos valores dos píxeis não são uniformes, pois alguns píxeis sofrem um aumento de intensidade enquanto outros vêm uma diminuição da sua intensidade, e outros podem ainda manter a sua intensidade original. Desta forma a dependência estatística entra **query** e **target_lightning_contrast.bmp** é diminuída, o que leva, consequentemente, à diminuição da informação mútua.

3. Em **target_noise.bmp**, à semelhança do que acontece em **target_lightning_contrast.bmp**, as alterações sofridas pelos valores dos píxeis não são semelhantes para todos. No entanto, a introdução de ruído na imagem causa alterações aleatórias, diminuíndo de forma mais drástica a dependência estatística entre a query e o target, levando a uma diminuição maior da informação mútua.

- c) Ao contrário do exercício anterior, as janelas com informação mútua para as várias imagens target não se obtêm todas nas mesmas coordenas, pois estas não são variações de uma mesma imagem. No entanto as janelas com maior informação mútua coincidem com as faces dos gatos nas várias imagens, permitindo-nos localizar as mesmas dentro de uma imagem maior.

```
Informação mútua no target*.bmp, por ordem decrescente:
        target1.bmp: 1.70079, em (x,y): (121, 196)
        target2.bmp: 1.20068, em (x,y): (76, 151)
        target3.bmp: 1.17056, em (x,y): (166, 91)
        target4.bmp: 1.11233, em (x,y): (256, 196)
>>
```

Figura 11

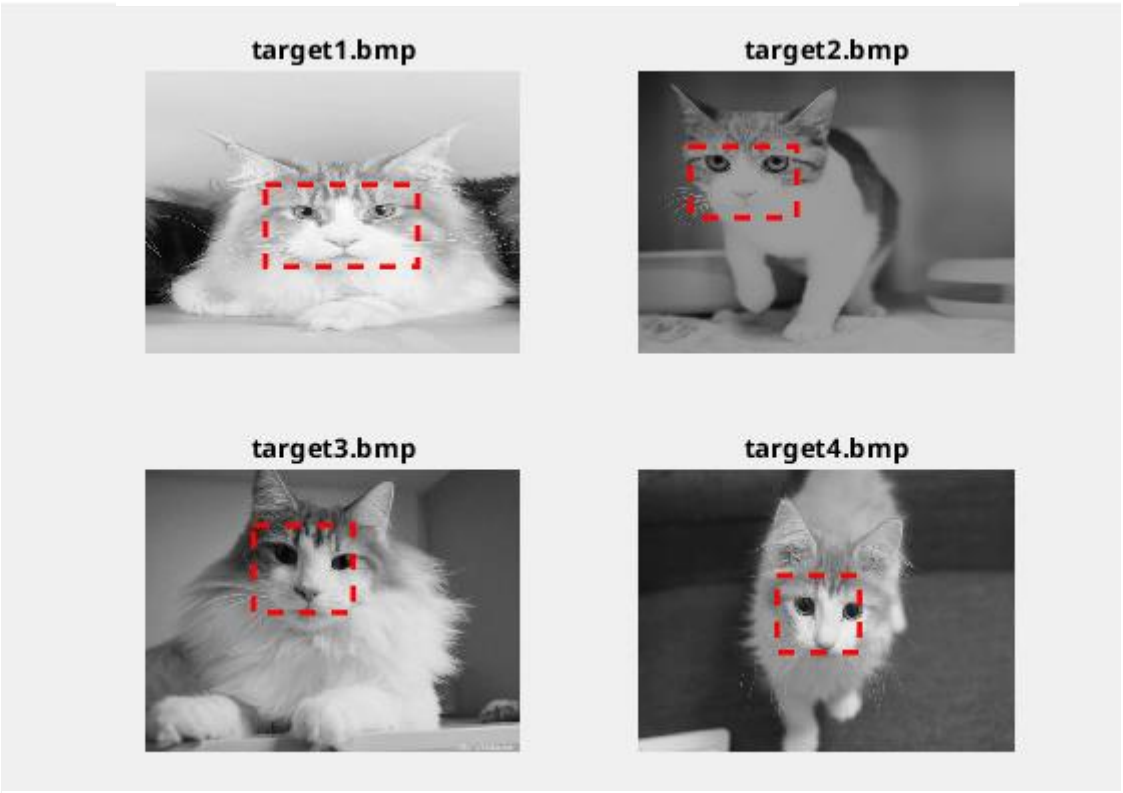


Figura 12