# Teoria da Informação

# Trabalho Prático nº 1

# Entropia, Redundância e Informação Mútua

# Introdução

**Período de execução**: 4 aulas práticas laboratoriais.

**Constituição dos Grupos:** 3 elementos da mesma turma PL.

**Formato de Entrega**:
Submissão de um ficheiro zip com o código fonte e relatório no inforestudante.

**Prazo de Entrega**:
**1 de Novembro, sexta-feira, 23h59**

**Esforço extra aulas previsto**: 15h/aluno

**Objectivo**: Pretende-se que o aluno adquira sensibilidade para as questões fundamentais de teoria de informação, em particular informação, redundância, entropia e informação mútua.

# Trabalho Prático

**1.** Escreva uma rotina em Matlab que dada uma fonte de informação *P* com um alfabeto $A=\{a_1,\ldots,a_n\}$ determine e visualize o **histograma de ocorrência dos seus símbolos**.

**2.** Escreva o código que dada uma fonte de informação *P* com um alfabeto $A=\{a_1,\ldots,a_n\}$ determine o **limite mínimo teórico para o número médio de bits por símbolo**.

**3.** Usando as rotinas desenvolvidas nas alíneas 1) e 2) determine a distribuição estatística e o número médio de bits por símbolo para as seguintes fontes:

- landscape.bmp
- MRI.bmp
- MRIbin.bmp
- soundMono.wav
    - Nota: alfabeto de som no intervalo [-1, 1[ (limite superior aberto); se o intervalo entre amostras sucessivas for 'd', o alfabeto será {-1, -1+d, -1+2d, ..., 1-d}. 'd' depende da dimensão do alfabeto (256 se 8 bits de quantização, 65536 se 16 bits, ...)
- lyrics.txt (nesta fonte considere somente os símbolos regulares do alfabeto, ignorando acentos e símbolos de pontuação)

**Nota: a chamada das rotinas indicadas neste ponto e nos seguintes deve ser feita num script, por exemplo, main.m**

✍ Apresente os resultados.
✍ Analise e comente os resultados.

📖 Será possível comprimir cada uma das fontes de forma não destrutiva? Se Sim, qual a compressão máxima que se consegue alcançar? Justifique.

**Notas:**

- A leitura de ficheiros de texto deverá ser efetuada com recurso às funções ***fopen*** e ***fscanf*** (sintaxe idêntica à da linguagem C – consultar a ajuda do Matlab em caso de dúvida) ou através da função ***fileread***
    - Poderá também utlizar as funções ***double*** e ***num2str*** (conversão de string para número e vice-versa)

- As rotinas ***imRead*** e ***audioRead*** permitem, respetivamente, efetuar a leitura de ficheiros de imagem (bmp, gif, jpeg, etc.) e wav.
- A função ***imfinfo*** dá informação sobre uma dada imagem (e.g., dimensão, número de bits por pixel, etc.)
- A função ***audioinfo*** dá informação sobre um dado áudio (e.g., dimensão, número de bits por sample, interprete, etc.)
- As suas sintaxes são as que seguidamente se apresentam:

- ***Y = imread(filename, format)*** ou ***Y = imread(filename)***

  ***filename*** – string com o nome do ficheiro que contém a imagem a ler
  ***format*** – string com a identificação do formato do ficheiro (não especificado → inferido pela análise do ficheiro)
  ***Y*** - matriz de pixels (bidimensional para uma imagem de níveis de cinzento ou indexada e tridimensional para uma imagem RGB)

  Exemplo: A = imread('imagem.bmp')

  Para visualizar uma imagem aberta poderá usar a função ***'imshow(x)'***

  Exemplo: imshow(A)

- ***info = imfinfo(filename)***

  ***filename*** – string com o nome do ficheiro que contém a imagem cuja informação se pretende obter
  ***info*** – estrutura de dados com informação sobre a imagem

  Exemplo:

```
info = imfinfo('imagem.bmp');
disp(info.Width)
```

- ***[Y, fs] = audioread(filename)***

  ***filename*** - String com o nome do ficheiro que contém o clip de áudio a ler
  ***Y*** - Amostras (amplitude) do áudio; caso o áudio seja estéreo, Y conterá dois vectores de valores (amostras)
  ***fs -*** Frequência de amostragem

  Exemplo: [Y, fs] = audioread('audio.wav')

Para reproduzir um som em matlab poderá usar um player '**audioplayer*(y, fs)'***

Exemplo: ap = audioplayer(Y, fs); ap.play();

- ***info = audioinfo(filename)***

  ***filename*** – string com o nome do ficheiro que contém o som cuja informação se pretende obter
  ***info*** – estrutura de dados com informação sobre o som

  Exemplo:
  ```
  info = audioinfo('audio.wav');
  disp(info.BitsPerSample)
  ```

**4.** 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.

✍ Analise e comente os resultados.
✍ Analise e comente a variância dos comprimentos dos códigos resultantes.
🕮 Será possível reduzir-se a variância? Se sim, como pode ser feito em que circunstância será útil?

**Nota:**
A rotina *hufflen* determina o número de bits do código Huffman necessários à codificação de um conjunto de símbolos com uma dada frequência de ocorrência. A sua sintaxe é a seguinte:

***HLen= hufflen(freqOcurr)***

**freqOcurr** = [f(1),f(2),…,f(n)] – vector com a frequência de ocorrência dos *n* símbolos em que *f*(*i*) corresponde à frequência de ocorrência do símbolo número *i* do alfabeto.

**HLen** = [len(1),len(2),…,len(n)] – vector com o número de bits em que *len*(*i*) representa o número de bits necessários à codificação do símbolo número i.

Exemplo:
*hufflen([1,0,4,2,0,1]) => ans = [3,0,1,2,0,3]*
*hufflen([10,40,20,10]) => ans = [3,1,2,3]*

**5.** Repita a alínea 3) aplicando agrupamentos de símbolos, isto é, admitindo que cada símbolo é na verdade uma sequência de dois símbolos contíguos.

✍🕮 Analise e comente os resultados.

**6.** Em muitas situações reais, é necessário procurar-se uma imagem conhecida no meio de conjunto de imagens maior. Por exemplo, sistemas de condução automática têm que identificar peões que atravessam a estrada e travar o carro automaticamente.

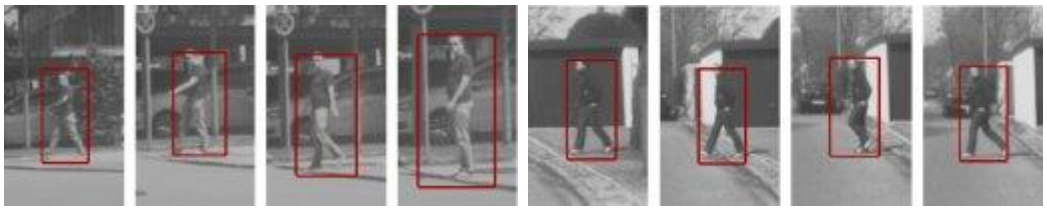
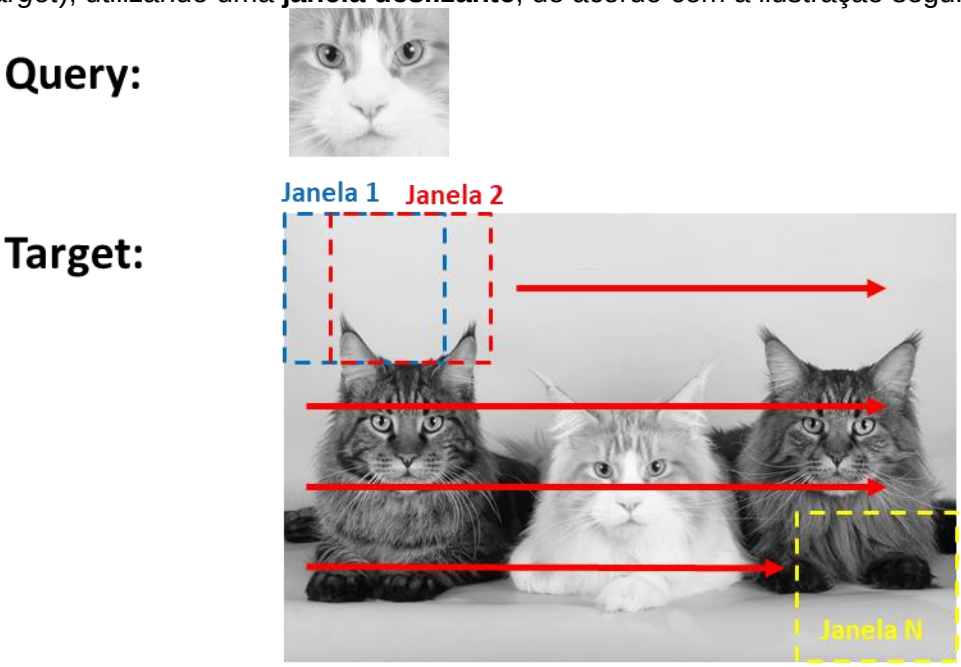


Figura 1 – Exemplo de segmentação e identificação automática de peões (adaptado de *Daimler Pedestrian Path Prediction Benchmark Dataset*)

Estes sistemas têm que se adaptar automaticamente a imagens de qualidade reduzida devido a fatores externos (como nevoeiro, fumos de outros veículos, etc). Em qualquer dos casos, é possível aplicar um conjunto vasto de técnicas. Neste ponto, será calculada a **informação mútua** entre uma imagem a pesquisar (*query*) e a imagem onde pesquisar (*target*), utilizando uma **janela deslizante**, de acordo com a ilustração seguinte:




Deste modo, a query será comparada com secções diferentes do target (correspondentes às janelas ilustradas). Por outras palavras, a query "deslizará" sobre o target, sendo calculada a informação mútua em cada uma das janelas. O intervalo entre janelas consecutivas é designado por **passo**. O mesmo passo deve ser considerado horizontalmente e verticalmente.

a) Escreva uma rotina em Matlab que, dada a query, o target, um alfabeto $A=\{a_1,\ldots,a_n\}$ e o passo, devolva a matriz de valores de informação mútua em cada janela.

***Simulação:***

*Dados:*

query = [ 2 6 ;
4 9 ]; (*matriz 2x2*)
target = [ 6 8 9 7 ;
2 4 9 9 ;
4 9 1 4 ]; (*matriz 3x4*)
alfabeto = 0 : 9;
step = 1

*Resultados a obter:*

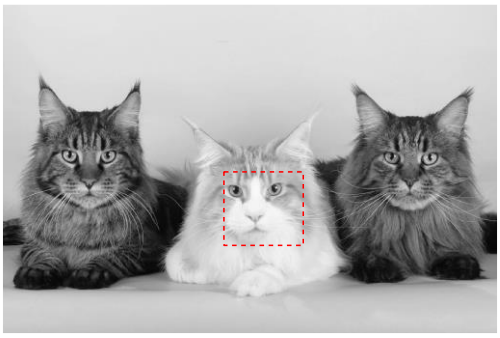
infoMutua = [ 2.0000 1.5000 0.8113 ;
1.5000 1.5000 1.5000] (*matriz 2x3*)

b) Usando o ficheiro "query" como query, determine a variação da informação mútua entre este e os ficheiros "target_original.bmp", "target_noise.bmp", "target_inverted.bmp" e "target_lightning_contrast.bmp". Defina um passo com valor de 15 pixeis.

b.i) Calcule o valor máximo de informação mútua da query com cada imagem de target, e as coordenadas 2d da janela em que esse valor é máximo.

✍ Apresente estes valores no relatório.

b.ii) Usando essas coordenadas, desenhe um retângulo sobre cada imagem target na localização da janela com maior valor de informação mútua.



**Nota:** para desenhar o retângulo, utilize a função ***rectangle*** fornecendo as coordenadas do canto superior esquerdo do retângulo, largura e altura do mesmo no parâmetro 'Position'. Exemplo:

```
rectangle('Position', [x, y, width, height], 'LineWidth', 2, 'LineStyle', '--', 'EdgeColor', 'r');
```

✍ Analise e comente os resultados. Foi possível localizar a imagem nas várias distorções do target? Qual o efeito na informação mútua obtida?

c) Pretende-se agora simular um pequeno simulador de identificação de faces de gatos. Usando o ficheiro “query.bmp” como query e os ficheiros “target*.bmp” como target (mantenha o passo de 15 pixeis da alínea anterior):
   - calcule a informação mútua máxima para cada target
   - verifique se corresponde à face do gato na imagem
   - e, finalmente, apresente os resultados da pesquisa, seriados por ordem decrescente de informação mútua. O sistema consegue encontrar o animal correto?

✍ Apresente os resultados (informação mútua em cada caso).
✍ Analise e comente os resultados.