ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Trabalho Prático 1

Processamento de Imagem e Biometria

2022

Índice

1) Exercício 1

Exercício 1 - Guia 1

a)

Matlab1:

read_image.m : Histograma

read_image_color.m : conversão de cores para níveis de cinzento

read_image_color_v2.m :

Figura 1: conversão de cores para níveis de cinzento;

Figura 2: troca das componentes R e B:

Figura 3: mostrar a imagem rgb como imagem de níveis de cinzento;

test_images.m : conversão de cores para níveis de cinzento e transformação para imagem binária

Matlab2:

arithmetic_operations.m:

Figura 1: adição dos pixéis das imagens

Figura 2: subtração dos pixéis das imagens

Figura 3: multiplicação dos pixéis das imagens

Figura 4: adição de uma constante aos pixéis das imagens

Figura 5: multiplicação por uma constante aos pixéis das imagens

logical_operations.m:

Figura 1: junção de duas imagens I1 e I2 (AND)

Figura 2: uma imagem ou outra I1 ou I2 (OR)

Figura 3: conversão inversa dos valores das duas imagens I1 xor I2 (XOR)

Figura 4: inverso do valor da imagem I1 e I2 NOT

optical_ilusion.m:

conversão de cores para níveis de cinzento

Matlab4:

image_lut.m:

Figura 1: inverso da imagem (NOT)

Figura 2: conversão para uma imagem de 3 níveis

Figura 3: conversão para uma imagem binária

image_negative.m:

inverso da imagem (NOT)

image_negative_biniarization.m:

conversão para níveis de cinzento e para imagem binária

Matlab5:

hist_processing.m:

Figura 1: ajuste de intensidades da imagem

Figura 3: ajuste de intensidades da imagem

image_contrast.m:

Figura 1 e 2: ajuste de contraste nas imagens

image_hs.m:

ajuste de intensidade de uma imagem com base dos valores de intensidade de outra imagem

image_operations.m:

Figura 1: imagem negativa

Figura 2: soma de constante

Figura 3: subtração de constante

Figura 4: amplificação de imagem

Figura 5: junção de duas imagens I1 e I2 (AND)

Figura 6: Ajuste de intensidade

Figura 7: Ajuste de contraste

b)

Código:

```
function read_image()
% Fechar todas as janelas de figuras.
close all;
% Limpar a consola.
clc

% Ler a imagem a partir do ficheiro.
I = imread('camera.gif');
%I = imread('bird.gif');
%I = imread('squares.gif');

% Obter as dimensões (resolução da imagem).
[M, N] = size(I);
[pixelCounts, greyLevels] = imhist(I);
numberOfPixels = sum(pixelCounts);
% Imprimir mensagem com as dimensões e resolução da imagem.
fprintf('Image with resolution %d x %d = %d pixels\n', M, N, numberOfPixels);
fprintf ('(i) Numero de Pixeis distintos: ');
fprintf ('%d ',pixelCounts);

H = entropy(I)

% Lançar nova janela de figura e mostrar a imagem em níveis de cinzento
% e o respetivo histograma.
figure(1);
subplot(121); imshow(I); colorbar; title(' Image ' );
subplot(122); imhist(I); title( sprintf(' Histogram. H=%.2f\n',H) );
impixelinfo;
% Escrever a imagem em níveis de cinzento como ficheiro PNG.
imwrite( I, 'out.png' );

% Calcular a energia da imagem.
E = sum(sum( I.^2 ));

% Calcular a potência da imagem.
P = E / (M*N);

% Calcular o valor médio da imagem.
m = sum(sum( I )) / (M*N);

C = max(max(I)) - min(min(I));
mi  = min(min(I));
mx = max(max(I));

[maxOc,ind] = max(pixelCounts);


fprintf('(ii) O pixel que ocorre o maior número de vezes é de intensidade %d e occore %d vezes.\n',ind-1,maxOc );
fprintf('(iii) Pixel com maior valor de intensidade: %d\n', mx );
fprintf('(iv) Valor Médio: %.2f, Contraste: %d, Entropia: %.2f\n',m, C, H);
```

c)

camera.gif

```
Image with resolution 256 x 256 = 65536 pixels
(i) Numero de Pixeis distintos: 0 0 0 0 0 0 0 4 423 1477 1259 1175 1456 1529 1685 1338 968 471 261 187 169 140 107 109 104 96 99 114
H =

    7.0097

(ii) O pixel que ocorre o maior número de vezes é de intensidade 14 e occore 1685 vezes.
(iii) Pixel com maior valor de intensidade: 253
(iv) Valor Médio: 118.72, Contraste: 246, Entropia: 7.01
```

bird.gif

```
Image with resolution 256 x 256 = 65536 pixels
(i) Numero de Pixeis distintos: 0 0 0 0 0 0 0 0 0 0 0 10 22 0 87 196 282 0 259 206 0 186 186 160 0 150 121 0 145 137 121 0 131 144 0
H =

    6.7744

(ii) O pixel que ocorre o maior número de vezes é de intensidade 147 e occore 2049 vezes.
(iii) Pixel com maior valor de intensidade: 212
(iv) Valor Médio: 125.39, Contraste: 201, Entropia: 6.77
```

squares.gif

```
Image with resolution 256 x 256 = 65536 pixels
(i) Numero de Pixeis distintos: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 4
H =

    1.0774

(ii) O pixel que ocorre o maior número de vezes é de intensidade 50 e occore 44800 vezes.
(iii) Pixel com maior valor de intensidade: 200
(iv) Valor Médio: 67.77, Contraste: 150, Entropia: 1.08
```

2) Exercício 2

Exercício 2 - Guia 1

3) Exercício 3

Exercício 3 - Guia 2

a)

Ajuste de brilho:

```java
private byte saturate(double val) {
    int iVal = (int) Math.round(val);
    iVal = iVal > 255 ? 255 : (iVal < 0 ? 0 : iVal);
    return (byte) iVal;
}

public Mat brightness(Mat myImage, Mat Result) {
    int beta = 0;

    try (Scanner scanner = new Scanner(System.in)) {
        System.out.print("* Enter the beta value [0-100]: ");
        beta = scanner.nextInt();
    }
    byte[] imageData = new byte[(int) (myImage.total()*myImage.channels())];
    myImage.get(0, 0, imageData);
    byte[] newImageData = new byte[(int) (Result.total()*Result.channels())];
    for (int y = 0; y < myImage.rows(); y++) {
        for (int x = 0; x < myImage.cols(); x++) {
            for (int c = 0; c < myImage.channels(); c++) {
                double pixelValue = imageData[(y * myImage.cols() + x) * myImage.channels() + c];
                /// Java byte range is [-128, 127]
                pixelValue = pixelValue < 0 ? pixelValue + 256 : pixelValue;
                newImageData[(y * myImage.cols() + x) * myImage.channels() + c]
                        = saturate(pixelValue + beta);
            }
        }
    }
    Result.put(0, 0, newImageData);

    return Result;
}
```

Resultado:

Ajuste de Contraste:

```
private byte saturate(double val) {
    int iVal = (int) Math.round(val);
    iVal = iVal > 255 ? 255 : (iVal < 0 ? 0 : iVal);
    return (byte) iVal;
}
 public Mat contrast(Mat myImage, Mat Result) {
    double alpha = 1.0;
    try (Scanner scanner = new Scanner(System.in)) {
        System.out.print("* Enter the alpha value [1.0-3.0]: ");
        alpha = scanner.nextDouble();
    }
    byte[] imageData = new byte[(int) (myImage.total()*myImage.channels())];
    myImage.get(0, 0, imageData);
    byte[] newImageData = new byte[(int) (Result.total()*Result.channels())];
    for (int y = 0; y < myImage.rows(); y++) {
        for (int x = 0; x < myImage.cols(); x++) {
            for (int c = 0; c < myImage.channels(); c++) {
                double pixelValue = imageData[(y * myImage.cols() + x) * myImage.channels() + c];
                /// Java byte range is [-128, 127]
                pixelValue = pixelValue < 0 ? pixelValue + 256 : pixelValue;
                newImageData[(y * myImage.cols() + x) * myImage.channels() + c]
                        = saturate(alpha * pixelValue);
            }
        }
    }
    Result.put(0, 0, newImageData);

    return Result;
}
```

Resultado:

4) Exercício 4

Exercício 4 – Guia 2

a)

Função:

```
private byte saturate(double val) {
    int iVal = (int) Math.round(val);
    iVal = iVal > 255 ? 255 : (iVal < 0 ? 0 : iVal);
    return (byte) iVal;
}
    //! [basic-linear-transform-output]
    Mat newImage = Mat.zeros(image.size(), image.type());
    //! [basic-linear-transform-output]

    //! [basic-linear-transform-parameters]
    double alpha = 1.0; /*< Simple contrast control */
    int beta = 0;       /*< Simple brightness control */

    alpha = 2;
    beta = -100;

    byte[] imageData = new byte[(int) (image.total()*image.channels())];
    image.get(0, 0, imageData);
    byte[] newImageData = new byte[(int) (newImage.total()*newImage.channels())];
    for (int y = 0; y < image.rows(); y++) {
        for (int x = 0; x < image.cols(); x++) {
            for (int c = 0; c < image.channels(); c++) {
                double pixelValue = imageData[(y * image.cols() + x) * image.channels() + c];
                /// Java byte range is [-128, 127]
                pixelValue = pixelValue < 0 ? pixelValue + 256 : pixelValue;
                newImageData[(y * image.cols() + x) * image.channels() + c]
                        = saturate(alpha * pixelValue + beta);
            }
        }
    }
    newImage.put(0, 0, newImageData);
    //! [basic-linear-transform-operation]

    //! [basic-linear-transform-display]
    /// Show stuff
    HighGui.imshow("Original Image", image);
    HighGui.imshow("imajust", newImage);
```

b)

Resultado:

5) Exercício 5

Exercício 5 – Guia 3

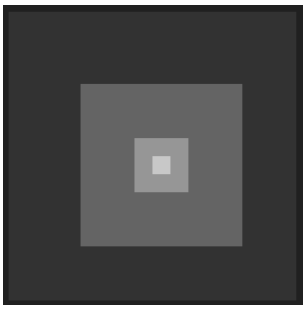
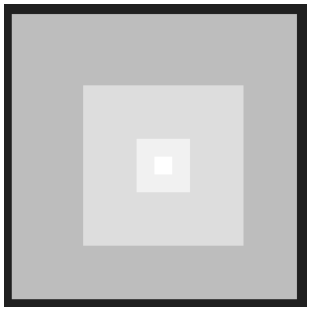
a)

Função:

```
import cv2 as cv
import numpy as np

# Open the Image
img = cv.imread('C:\\Users\\duart\\Desktop\\ISEL\\PIB\\MatLab\\out.png')

# Apply intencity transformation.
c = 255/(np.log(1 + np.max(img)))
log_transformed = c * np.log(1 + img)

# Specify the data type.
log_transformed = np.array(log_transformed, dtype = np.uint8)

# Save the output.
cv.imwrite('intensity_transformation.jpg', log_transformed)
```

b)

Resultado:

Original	Transformada
	
	
	

6) Exercício 6

Exercício 6 Guia 3

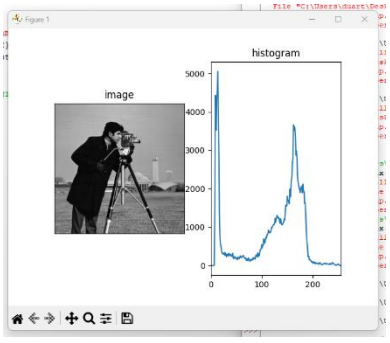
a)

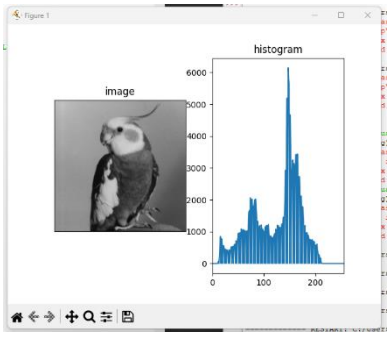
Função:

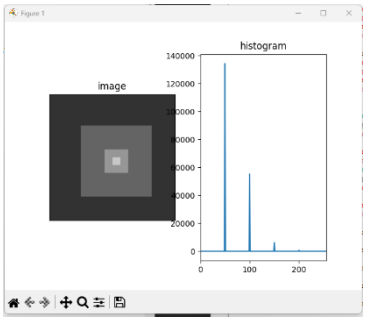
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

# imagem
img = cv2.imread('C:\\Users\\duart\\Desktop\\ISEL\\PIB\\Python\\camera.png')

#plot da imagem
plt.subplot(1,2,1)
plt.imshow(img,cmap='gray')
plt.title('image')
plt.xticks([])
plt.yticks([])

#plot do histograma da imagem
plt.subplot(1,2,2)
hist,bin = np.histogram(img.ravel(),256,[0,255])
plt.xlim([0,255])
plt.plot(hist)
plt.title('histogram')

plt.show()
```

b)

Resultados:







7) Exercício 7

(i) Ocultação de Identidade

Procedimentos:

1º Input da imagem;

2º Histograma da imagem de entrada;

3º Verificar se é uma imagem rgb ou em níveis de cinzento;

4º Se for RGB transformar em Grayscale;

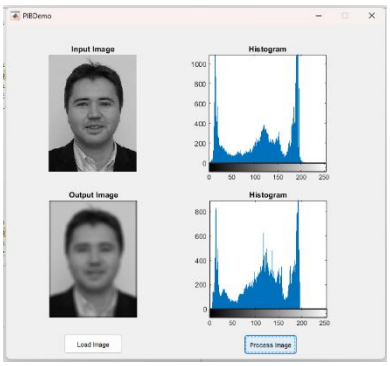
```
if size(I,3)==3
I = rgb2gray(I);
end
```

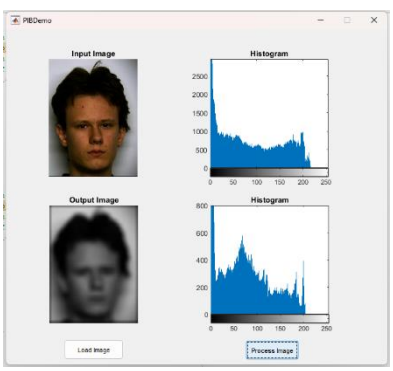
5º Aplicar um filtro gaussiano causando um efeito de desfoque na imagem;

```
H = fspecial('gaussian',8,10);
out = imfilter(I,H);
```

6º Mostrar a imagem ajustada e o seu histograma.

Exemplos:





(ii)Correção Tonal

Procedimentos:

1º Input da imagem;

2º Histograma da imagem de entrada;

3º Verificar se é uma imagem rgb ou em níveis de cinzento;

4º Se for RGB transformar em Grayscale;

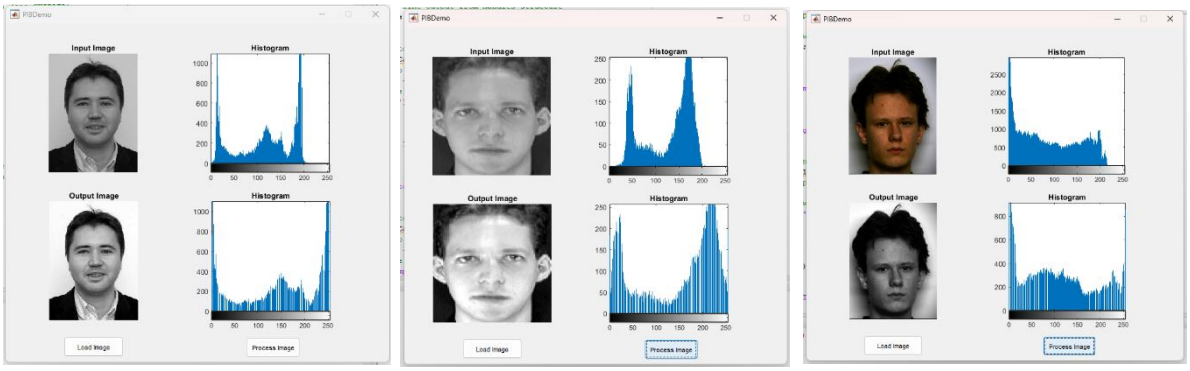
```
if size(I,3)==3
I = rgb2gray(I);
end
```

5º Ajuste de imagem a traves da função iamadjust do matlab;

```
I_imadjust = imadjust(I);
```

6º Mostrar a imagem ajustada e o seu histograma.

Exemplos:



(iii) Deteção de Pele

Procedimentos:

1º Input da imagem;

2º Histograma da imagem de entrada;

3º Verificar se é uma imagem rgb ou em níveis de cinzento;

4º Se for RGB transformar em Grayscale;

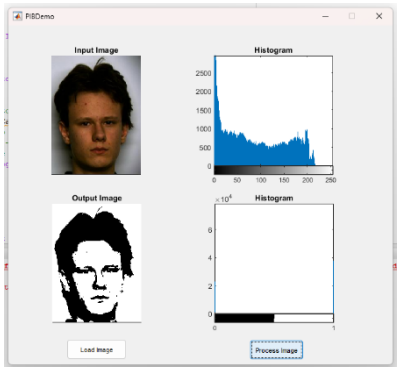
```
if size(I,3)==3
I = rgb2gray(I);
end
```

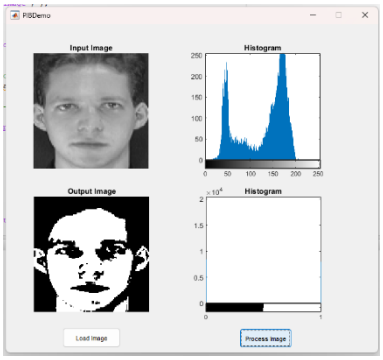
5º Transformação da imagem para imagem binária;

```
BW = im2bw(I,0.3);
```

6º Mostrar a imagem ajustada e o seu histograma;

Exemplos:





(iv) Deteção de Contornos

Procedimentos:

1º Input da imagem;

2º Histograma da imagem de entrada;

3º Verificar se é uma imagem rgb ou em níveis de cinzento;

4º Se for RGB transformar em Grayscale;

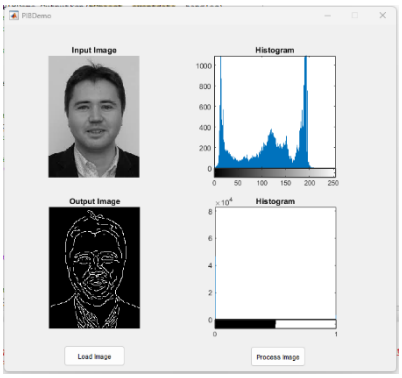
```
if size(I,3)==3
I = rgb2gray(I);
end
```

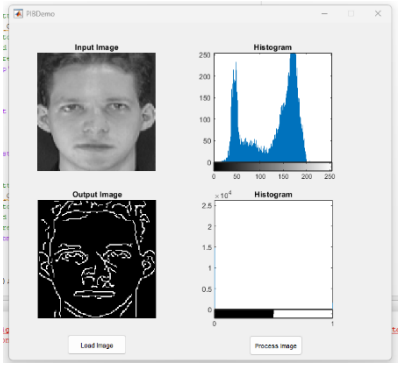
5º Fazer a deteção de contornos da imagem através da função edge(canny) do matlab;

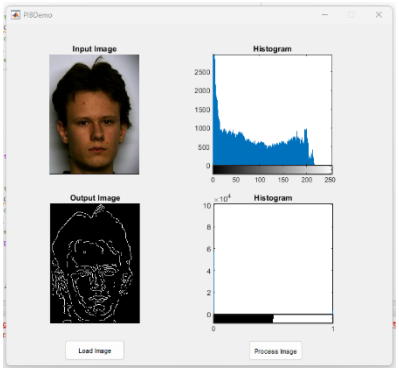
```
BW2 = edge(I,'canny');
axes(handles.axes3);
imshow(BW2);
```

6º Mostra a imagem e o seu histograma;

Exemplos:







(v) Isolamento da Face (Corte de Imagem)

Procedimentos:

1º Input da imagem;

2º Histograma da imagem de entrada;

3º Detetar a face presente na imagem;

```
FaceDetect = vision.CascadeObjectDetector;
FaceDetect.MergeThreshold = 7 ;
```

4º Detetar o local da face presente na imagem;

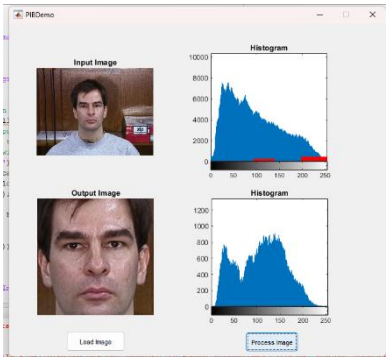
```
BB = step(FaceDetect, I);
```

5º Recortar a face presente na imagem;

```
for i = 1 : size(BB, 1)
  J = imcrop(I, BB(i, :));
end
```

6º Mostra a imagem recortada e o seu histograma;

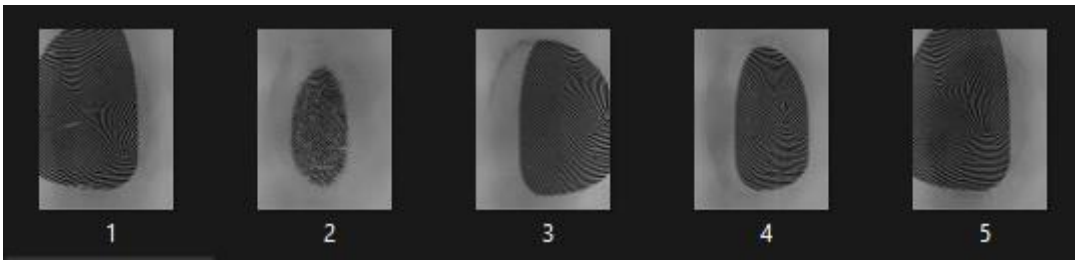
Exemplos:



8) Exercício 8

a)

Imagens geradas do tipo captativo:



Imagens geradas do tipo ótico:

b)

Procedimento:

1º Aplicar um ajuste de contraste na imagem original;

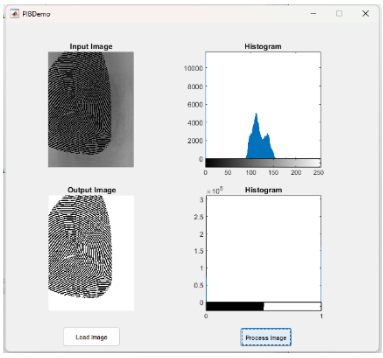
```
IB = imadjust(I);
```

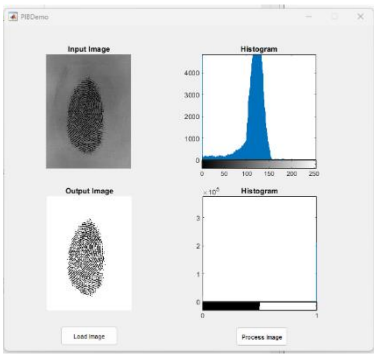
2º Gerar uma imagem binária a partir da nova imagem de contraste ajustado;

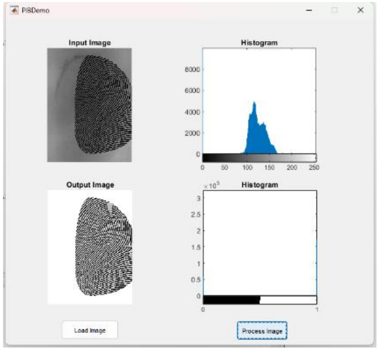
```
BW = im2bw(IB,0.4);
```

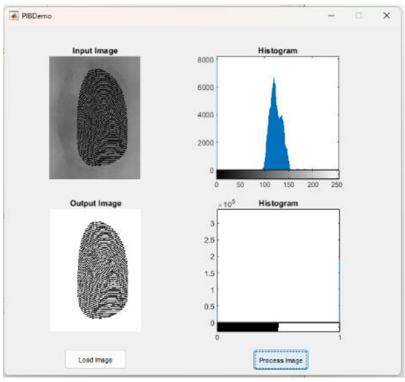
Exemplos:

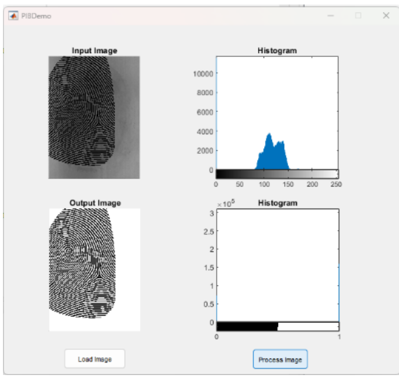
Imagens do tipo captativo:



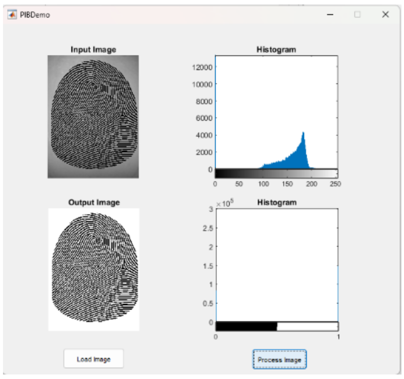


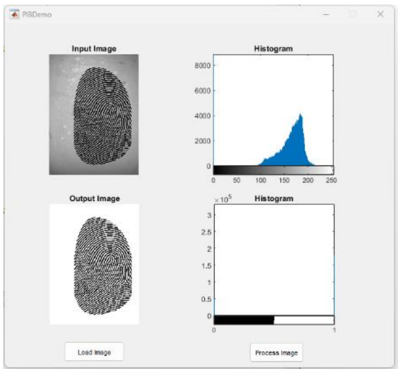


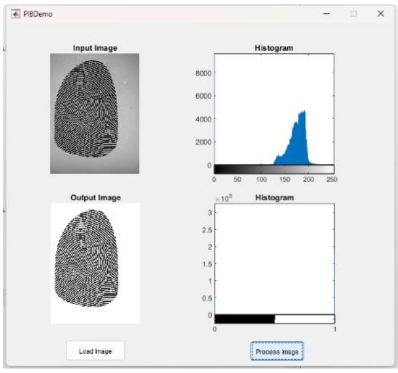


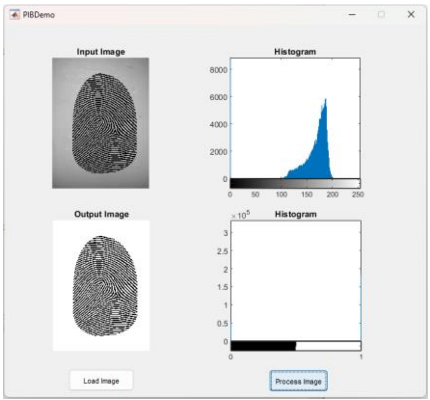


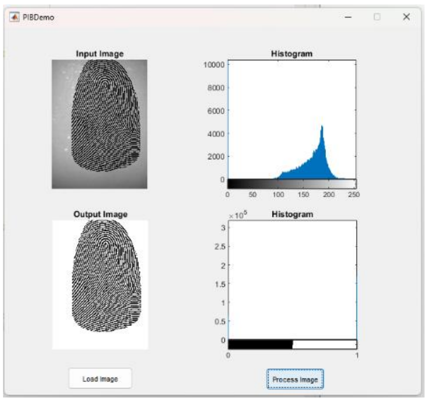
Imagens do tipo ótico:











9) Exercício 9

Adaptive Histogram Equalization (AHE):

- É uma técnica de processamento de imagem digital usada para melhorar o contraste das imagens. Comparado com a Histogram equalization é diferente tendo em conta que a Adaptive Histogram Equalization calcula vários histogramas, cada um correspondendo a diferentes secções da imagem, e usa-as para redistribuir os valores de intensidade da imagem. Sendo então utilizado para melhorar contrastes e contornos de bordas em cada região da imagem. Tendo como desvantagem o facto de amplificar o ruído em regiões relativamente homogéneas da imagem.

Contrast Limited Adaptive Histogram Equalization (CLAHE):

- Para prevenir o problema de amplificação de ruído existe a Contrast Limited Adaptive Histogram Equalization (CLAHE) que é uma variante da AHE, na qual a amplificação de ruído é limitada, com o propósito de reduzir a mesma.

Canny edge detection:

- Canny edge detection é um detetor de bordas/contornos, que faz a extração de informação de diferentes objetos de visão e reduz acentuadamente a quantidade de dados a serem processados.

- Critérios para a aplicação deste método:

  - Deteção de erro com taxa de erro baixa, significa que a deteção deve conseguir obter o maior número de bordas demonstradas na imagem possível eficazmente.
  - Uma borda na imagem deve ser marcada apenas uma vez, e se possível o ruido da imagem não deverá criar novas bordas.