ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Trabalho Prático 2

Processamento de Imagem e Biometria

2022

Índice

1) Exercício 1

a)

Bird – Realização de transformações independentes (Vermelho, Verde e Azul na escala de cinza da imagem inicial e obter o resultado da combinação de todas as transformações combinadas.

CT - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Face_themogram - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Finger – Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Flowers - Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Goldhill - Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Iris - Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

MR - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

PET - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Squares – Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Thyroid - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

XRay - Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

b)

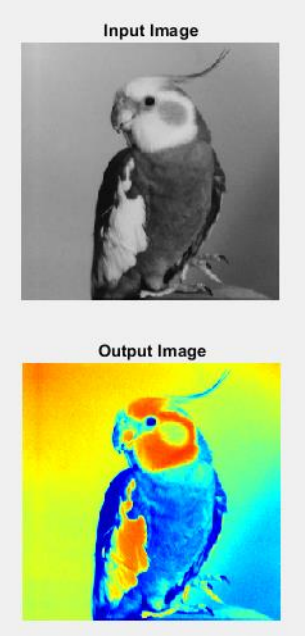
Bird:

Realização de transformações independentes (Vermelho, Verde e Azul na escala de cinza da imagem inicial e obter o resultado da combinação de todas as transformações combinadas.

Código:

```
%PRE-ALLOCATE A MATRIX
Output = zeros([size(I,1) size(I,2) 3]);

%Define a colormap
map = colormap(jet(256));

%Assign the columns to 1-D RED,GREEN and BLUE
Red = map(:,1);
Green = map(:,2);
Blue = map(:,3);

%MAP THE COLORS BASED ON THE INTENSITY OF THE IMAGE
Output(:,:,1) = Red(I);
Output(:,:,2) = Green(I);
Output(:,:,3) = Blue(I);

Output = im2uint8(Output);
```

Resultado:



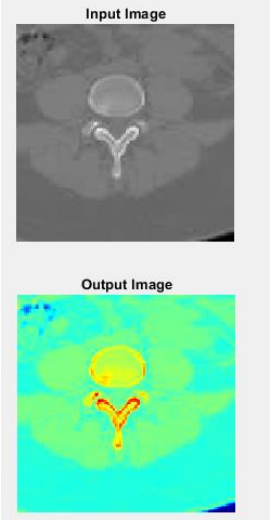
CT:

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:



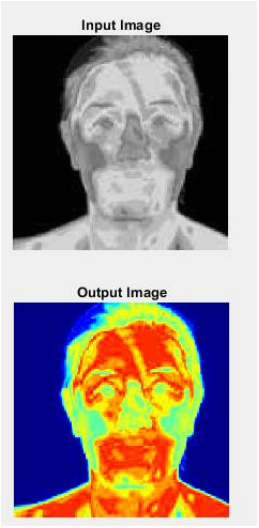
Face_themogram

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:



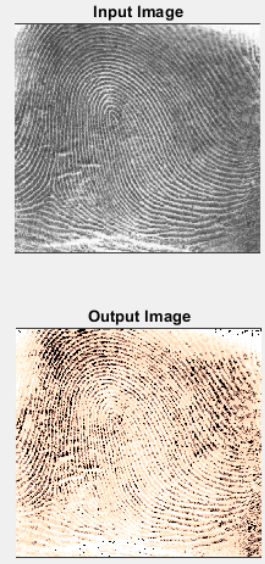
Finger

Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Código:

```
[row,col]=size(A);
for i=1:1:row
    for j=1:1:col
        if (A(i,j)>= 0) && (A(i,j)< 90)
            red(i,j)=0;
            green(i,j)=0;
            blue(i,j)=0;
        elseif (A(i,j)>= 90) && (A(i,j)< 108)
            red(i,j)=168;
            green(i,j)=120;
            blue(i,j)=106;
        elseif (A(i,j)>= 108) && (A(i,j)< 144)
            red(i,j)=255;
            green(i,j)=222;
            blue(i,j)=190;
        elseif (A(i,j)>= 144) && (A(i,j)< 198)
            red(i,j)=255;
            green(i,j)=245;
            blue(i,j)=225;
        elseif (A(i,j)>= 198) && (A(i,j)< 255)
            red(i,j)=255;
            green(i,j)=255;
            blue(i,j)=255;
        end
    end
end
pseudo_image=cat(3,red,green,blue);
pseudo_image=pseudo_image/255;%convert from 0-255 to 0-1
```

Resultado:



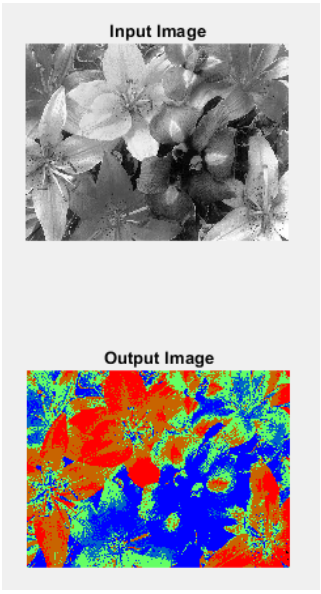
Flowers

Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Código:

```
[row,col]=size(I);
for i=1:1:row
    for j=1:1:col
        if (I(i,j)>= 0) && (I(i,j)< 90)
            red(i,j)=0;
            green(i,j)=100;
            blue(i,j)=200;
        elseif (I(i,j)>= 90) && (I(i,j)< 108)
            red(i,j)=0;
            green(i,j)=150;
            blue(i,j)=150;
        elseif (I(i,j)>= 108) && (I(i,j)< 144)
            red(i,j)=100;
            green(i,j)=100;
            blue(i,j)=100;
        elseif (I(i,j)>= 144) && (I(i,j)< 198)
            red(i,j)=150;
            green(i,j)=150;
            blue(i,j)=0;
        elseif (I(i,j)>= 198) && (I(i,j)< 255)
            red(i,j)=200;
            green(i,j)=100;
            blue(i,j)=0;
        end
    end
end
pseudo_image=cat(3,red,green,blue);
pseudo_image=pseudo_image/255;%convert from 0-255 to 0-1
```

Resultado:



Goldhill

Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Código:

```
[row,col]=size(A);
for i=1:1:row
    for j=1:1:col
        if (A(i,j)>= 0) && (A(i,j)< 90)
            red(i,j)=0;
            green(i,j)=0;
            blue(i,j)=0;
        elseif (A(i,j)>= 90) && (A(i,j)< 108)
            red(i,j)=168;
            green(i,j)=120;
            blue(i,j)=106;
        elseif (A(i,j)>= 108) && (A(i,j)< 144)
            red(i,j)=255;
            green(i,j)=222;
            blue(i,j)=190;
        elseif (A(i,j)>= 144) && (A(i,j)< 198)
            red(i,j)=255;
            green(i,j)=245;
            blue(i,j)=225;
        elseif (A(i,j)>= 198) && (A(i,j)< 255)
            red(i,j)=255;
            green(i,j)=255;
            blue(i,j)=255;
        end
    end
end
pseudo_image=cat(3,red,green,blue);
pseudo_image=pseudo_image/255;%convert from 0-255 to 0-1
```

Resultado:

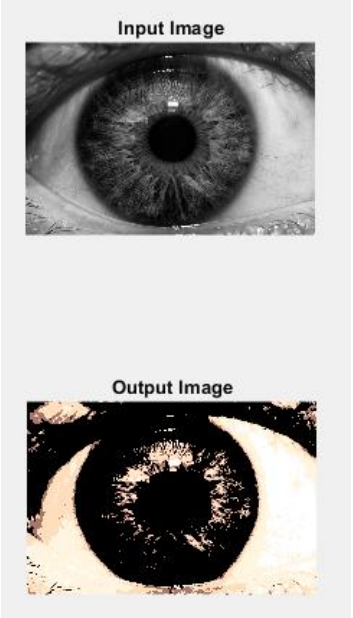
Iris

Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Código:

```
[row,col]=size(A);
for i=1:1:row
    for j=1:1:col
        if (A(i,j)>= 0) && (A(i,j)< 90)
            red(i,j)=0;
            green(i,j)=0;
            blue(i,j)=0;
        elseif (A(i,j)>= 90) && (A(i,j)< 108)
            red(i,j)=168;
            green(i,j)=120;
            blue(i,j)=106;
        elseif (A(i,j)>= 108) && (A(i,j)< 144)
            red(i,j)=255;
            green(i,j)=222;
            blue(i,j)=190;
        elseif (A(i,j)>= 144) && (A(i,j)< 198)
            red(i,j)=255;
            green(i,j)=245;
            blue(i,j)=225;
        elseif (A(i,j)>= 198) && (A(i,j)< 255)
            red(i,j)=255;
            green(i,j)=255;
            blue(i,j)=255;
        end
    end
end
pseudo_image=cat(3,red,green,blue);
pseudo_image=pseudo_image/255;%convert from 0-255 to 0-1
```

Resultado:



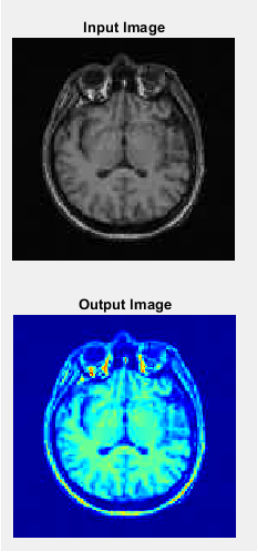
MR

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:



PET

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:

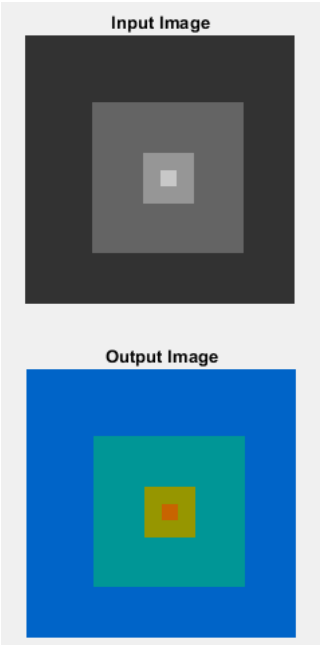
Squares

Manipulação manual dos valores da cor correspondente em RGB na escala de cinza da imagem inicial.

Código:

```
[row,col]=size(I);
for i=1:1:row
    for j=1:1:col
        if (I(i,j)>= 0) && (I(i,j)< 90)
            red(i,j)=0;
            green(i,j)=100;
            blue(i,j)=200;
        elseif (I(i,j)>= 90) && (I(i,j)< 108)
            red(i,j)=0;
            green(i,j)=150;
            blue(i,j)=150;
        elseif (I(i,j)>= 108) && (I(i,j)< 144)
            red(i,j)=100;
            green(i,j)=100;
            blue(i,j)=100;
        elseif (I(i,j)>= 144) && (I(i,j)< 198)
            red(i,j)=150;
            green(i,j)=150;
            blue(i,j)=0;
        elseif (I(i,j)>= 198) && (I(i,j)< 255)
            red(i,j)=200;
            green(i,j)=100;
            blue(i,j)=0;
        end
    end
end
pseudo_image=cat(3,red,green,blue);
pseudo image=pseudo image/255;%convert from 0-255 to 0-1
```

Resultado:



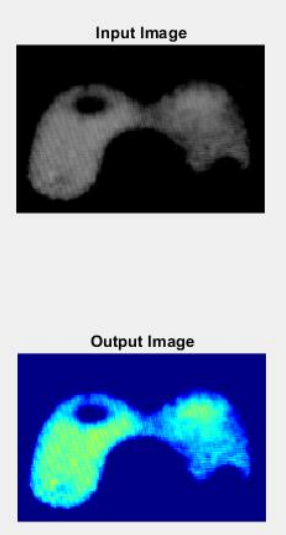
Thyroid

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:



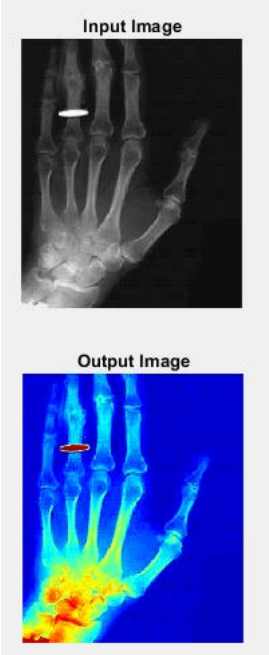
XRay

Conversão da imagem e do seu mapa de cores correspondente para RGB (ind2rgb).

Código:

```
rgbImage = ind2rgb(I, colormap);
```

Resultado:



2) Exercício 2

a)

Face1 – Blue plane corrupted by Salt & Pepper Noise

Face2 – Complement image

Face3 – Decrease brigthness

Face4 – Color Balancing – Heavy in Blue

Lena1 – Increase intensity

Lena2 – Color Balancing – Heavy in Blue

Lena3 – Decrease intensity

Lena4 – Red plane corrupted by Salt & Pepper Noise

Monarch1 – Color Balancing – Heavy in Blue

Monarch2 – Color Balancing – Heavy in Magenta

Monarch3 – Blurred

Monarch4 – Negative version

b)

Face1 – Blue plane corrupted by Salt & Pepper Noise

Solução:

```
1 - [noisyR,noisyG,noisyB] = imsplit(face1.png);
2 - net = denoisingNetwork("dncnn");
3 - denoisedR = denoiseImage(noisyR,net);
    denoisedG = denoiseImage(noisyG,net);
    denoisedB = denoiseImage(noisyB,net);
4 - denoisedRGB = cat(3,denoisedR,denoisedG,denoisedB);
5 - imshow(denoisedRGB);
```

Face2 – Complement image

Solução: rgb = imcomplement (‘face2.png’);

imshow(rgb);

Face3 – Decreased brightness

Solução: I = imread(face3.png);

I1 = I+50 ;imshow(I1);

Face4 – Color slicing

Solução: 1 - [noisyR,noisyG,noisyB] = imsplit(face1.png);
2 - net = denoisingNetwork("dncnn");
3 - denoisedR = denoiseImage(noisyR,net);
denoisedG = denoiseImage(noisyG,net);
denoisedB = denoiseImage(noisyB,net);
4 - denoisedRGB = cat(3,denoisedR,denoisedG,denoisedB);
5 - imshow(denoisedRGB);

Lena1 – Increased intensity

Solução: I = imread(lena1.png);

I1 = I - 50 ;

imshow(I1);

Lena2 – Color Balancing – Heavy in Blue

Solução: J = histeq(lena2.png);

imshow(J);

Lena3 – Decreased intensity

Solução: I = imread(lena3.png);

I1 = I+50;

imshow(I1);

Lena4 – Red plane corrupted by Salt & Pepper Noise

Solução: **1 -** [noisyR,noisyG,noisyB] = imsplit(face1.png);

2 - net = denoisingNetwork("dncnn");
3 - denoisedR = denoiseImage(noisyR,net);
denoisedG = denoiseImage(noisyG,net);
denoisedB = denoiseImage(noisyB,net);
4 - denoisedRGB = cat(3,denoisedR,denoisedG,denoisedB);
5 - imshow(denoisedRGB);

Monarch1 – Color Balancing – Heavy in Blue

Solução: J = histeq(monarch1.png);

imshow(J);

Monarch2 – Color Balancing – Heavy in Magenta

Solução: J = histeq(monarch2.png);

imshow(J);

Monarch3 – Blurred

Solução: Recovered = deconvwnr(monarch3.png, psf,0);

Imshow(Recovered);

Monarch4 – Negative version

Solução: I = imread(monarch4.png);

I2 = 255 – I1;

3)

a) e b)

(i)

Suporte de imagens coloridas, em todas as operações desenvolvidas no primeiro trabalho prático e as indicadas neste trabalho prático.

Para premitir o suporte para este tipo de imagem podemos fazer uma verificação do tipo da imagem introduzida e caso esta seja uma imagem colorida podemos sempre convertela para uma imgem a preto e branco atravez da função (rgb2gray).

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

(ii) Inserção de marca de água com texto genérico, numa localização (x, y) indicada pelo utilizador.

Para adicionar a marca de água foi escolhido o texto 'PIB MARK' e a localização foi a de coordenadas (0,0). Em relação ao processo de adição, este dividiu-se nas seguintes partes:

1. Definição dos parâmetros do texto a ser adicionado;

```
Transparency = 0.6;
fontColor = [1,1,1];
position = [0,0];
```

2. Formação da imagem do texto a ser adicionado.

```
mask = zeros(size(I),'like',I);
outimg = insertText(mask,position,'PIB Mark', ...
    'BoxOpacity',0,...
    'FontSize',40,...
    'TextColor', 'white');
```

3. Passagem da imagem criada para uma imagem binaria.

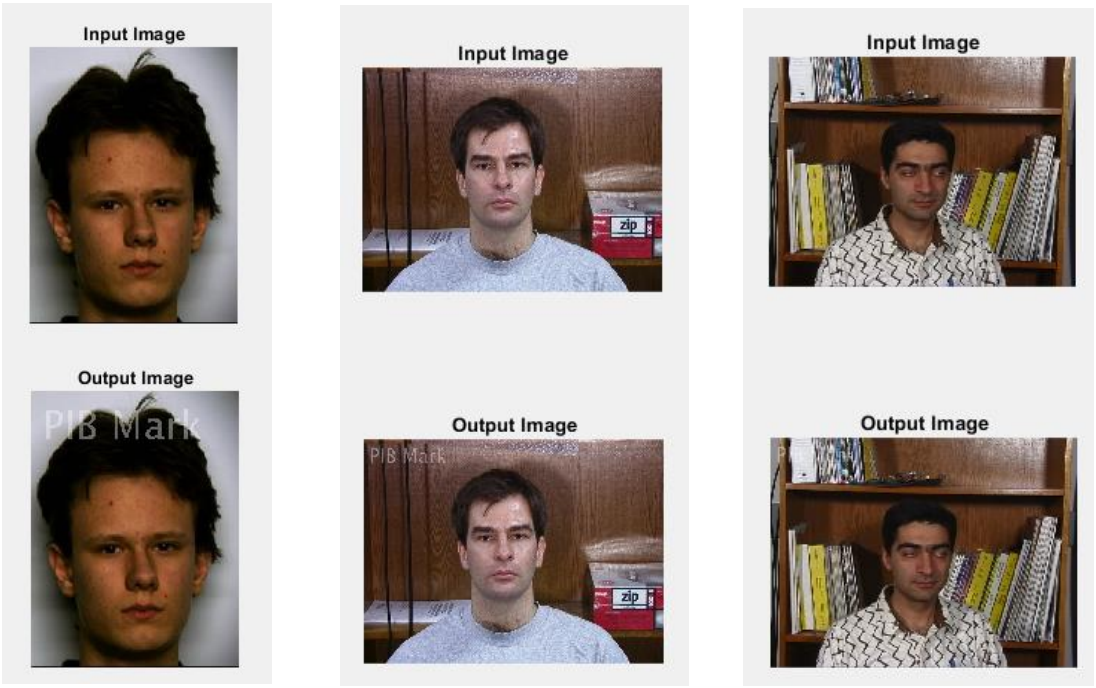
```
bwMask = imbinarize(rgb2gray(outimg));
```

4. Sobreposição da imagem criada na imagem original.

```
finalImg = labeloverlay(I,bwMask,...
    'Transparency',Transparency,...
    'Colormap',fontColor);
```

5. Mostrar imagem resultante.

Resultados:



(iii) Elaboração de um efeito visual (“filtro") à sua escolha, tal como a colocação de óculos, flores, borboletas ou um chapéu sobre a imagem do utilizador.

Para elaboração deste efeito foi utilizado uma imagem de uns oculos para adicionar à face presente em cada imagem. Para tal os procedimentos efetuados foram os seguintes:

1º Deteção da face na imagem.

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

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

2º Recorte da face da imagem original.

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

3º Ajuste de tamanho da imagem dos oculos com a imagem Recortada.

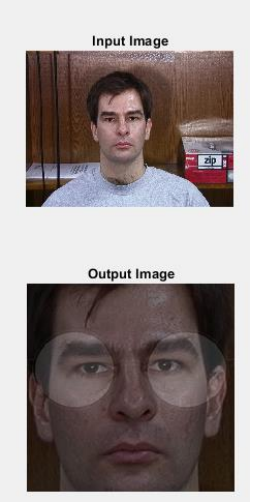
```
[rows, cols, chans] = size(J);
IG = imresize(IG,[rows cols]);
```

4º Sobrepossição das duas imagens.

```
C = imfuse(J,IG,'blend','Scaling','joint');
```

5º Mostrar imagem resultante.

Resultados obtidos:



(iv) Reconhecimento facial, através de ações de autenticação e identificação.

Processo de reconhecimento facial:

1º Deteção da face na imagem;

```
faceDetector = vision.CascadeObjectDetector;

bbox = step(faceDetector, I);
```

2º Criação de um retângulo para adicionar na face encontrada;

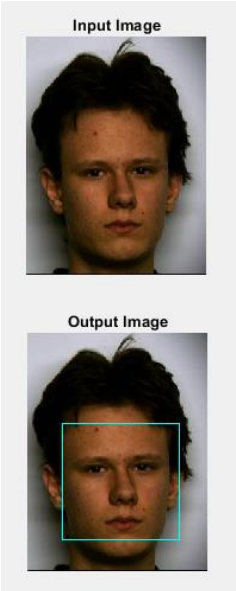
```
shapeInserter = vision.ShapeInserter('BorderColor','Custom','CustomBorderColor',[0 255 255]);
```

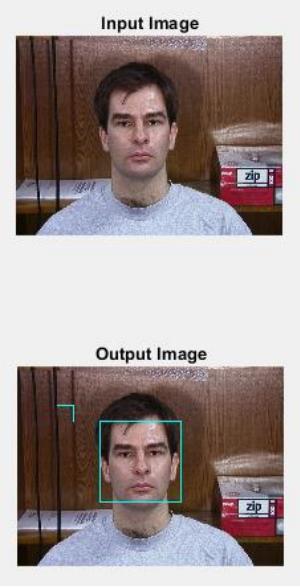
3º Adição dos retângulos nas faces encontradas;

```
I_faces = step(shapeInserter, I, int32(bbox));
```

4º Mostrar resultados;

Resultados:





4.

a) e b)

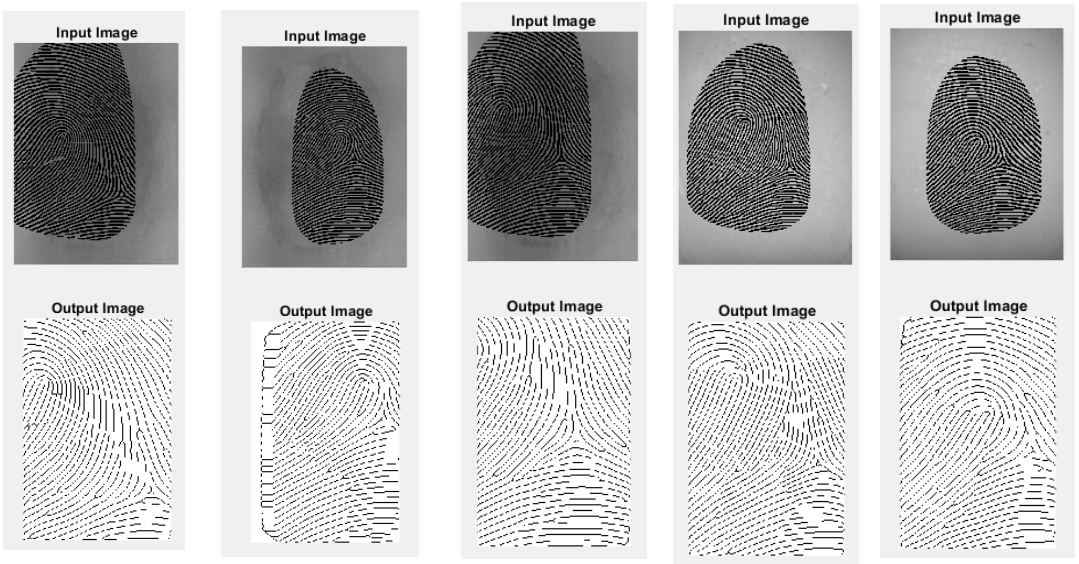
Considerando a aplicação desenvolvida no Exercício 8 do primeiro trabalho prático, para imagens de impressão digital foi desenvolvida uma nova aplicação com o objetivo de detetar as minúcias.

Em primeiro lugar foi necessário utilizar as imagens já processadas pela aplicação desenvolvida no Exercício 8 do primeiro trabalho prático e abrir na nova aplicação para assim dar inicio a um novo processamento.

De Seguida foi necessário fazer um ampliamento da imagem para que o resultado final seja mais percetível de se observar.

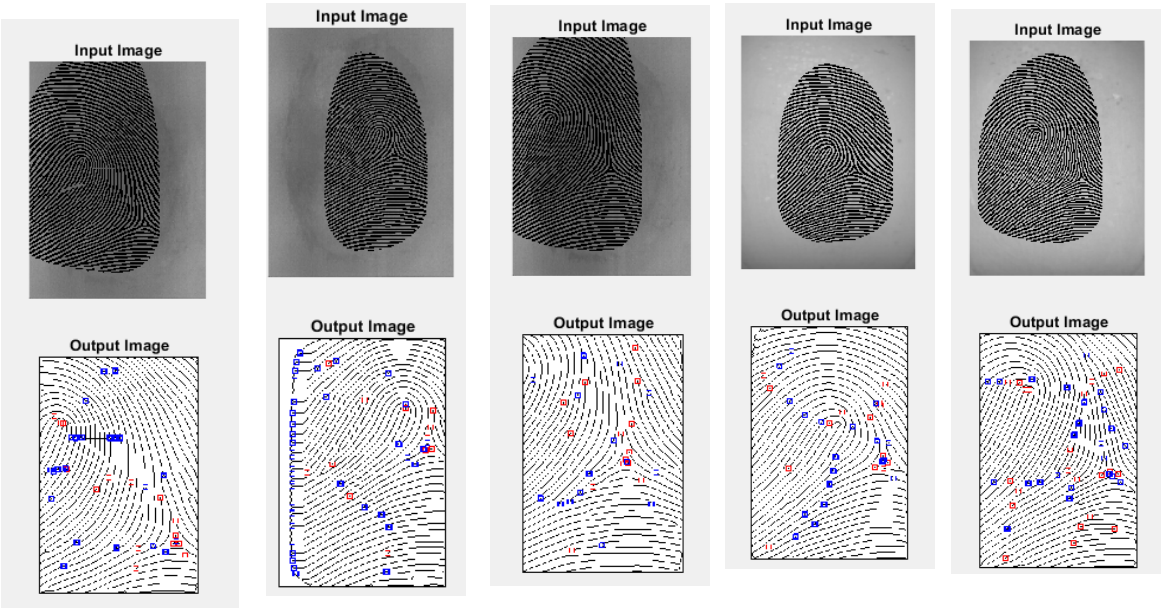
Em terceiro lugar foi feita uma operação morfológica de ‘Thining’ sobre a imagem deixando a imagem preparada para o processamento de minúcias.

Exemplos da operação:



Por último foram processadas as linhas resultantes da operação de ‘Thining’ e foram assim identificadas as minúcias podendo assim formar os padrões para mais tarde serem armazenados e depois utilizados no processo de comparação para autenticar utilizadores.

Resultados:



Qualidade dos resultados:

Em conclusão, os resultados com as impressões digitas do tipo ótico, no geral foram melhores do que as do tipo captativo.

Também podemos verificar que ocorreram algumas falhas na deteção de algumas minúcias, no entanto também podemos ter em conta que o processamento feito no exercício 8 do primeiro trabalho prático pode ter influenciado o resultado final deste exercício.