IPS Instituto Politécnico de Setúbal

Escola Superior de Tecnologia de Setúbal

Laboratório 8

Curso Licenciatura em Engenharia Eletrotécnica e Computadores

Autores:

201901691 - Diogo Silva

201900875 - João Beco

Docentes

Professor Pedro Vitoriano

Professor Tito Amaral

03/2023

Índice

Lista de Figuras

1 Introdução

1.1 Descrição do trabalho

A Visão Artificial é um recurso que necessita de câmaras industriais e de sistemas de iluminação para ter a melhor eficiência de processos em ambientes industriais.

Desde sempre foi muito mais produtivo existir um sistema de Visão Artificial, mas, infelizmente só recentemente é que é possível equipar a indústria com esta tecnologia, graças ao avanço do conhecimento digital. Um sistema de Visão Artificial é capaz de realizar inúmeras inspeções a várias peças num segundo de maneira automática e eficiente, com grande precisão, evitando assim, a influência do cansaço e a distração de humano, por exemplo.

Neste trabalho de laboratório, o objetivo é aplicar os conhecimentos previamente adquiridos nos laboratórios anteriores sobre as diversas funções e métodos utilizados no processamento de imagens. Utilizando essas habilidades, a tarefa é desenvolver um algoritmo que possa executar uma função específica relacionada ao processamento de imagens, mas precisamente, Expansão Local de Regiões e o *Clustering*

1.2 Estrutura do trabalho

O presente relatório é constituído por uma introdução, seguido dos exercícios propostos, logo de seguida, a solução dos exercícios propostos, tudo desenvolvido utilizando o programa *Matlab*, e, por fim, uma pequena conclusão.

2 Expansão Local de Regiões

2.1 Exercícios - *Blob Coloring I*

A) Leu-se o ficheiro de imagem "img_M.jpg" e atribuímo à variável Im1.

```
Im1 = imread('img_M.jpg');
imshow(Im1)
```

Figura 1 – img_M.jpg

B) Utilizando a função correspondente, transformamos a imagem numa versão binarizada com o threshold de 0,5.

```
Im1_cinzento=rgb2gray(Im1); % Converter para escala de cinza
threshold = 0.5; % Limiar de 0,5
Im1_bw = imbinarize(Im1_cinzento, threshold); % Aplicar o limiar
para obter a imagem binarizada
imshow(Im1_bw); % Exibir a imagem binarizada
```

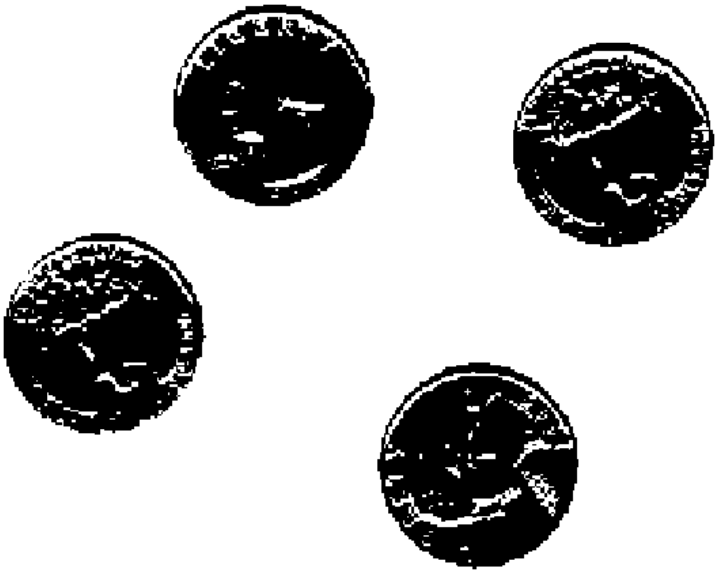

Figura 2 – Binarização da imagem img_M.jpg

C) Utilizámos a função correspondente e aplicamos a transformação de erosão na imagem de forma a maximizar o preenchimento interior das moedas.

```
Elem_estr_im1 = strel("disk",4);
Im1_erode=imerode(Im1_bw, Elem_estr_im1);
imshow(Im1_erode)
```

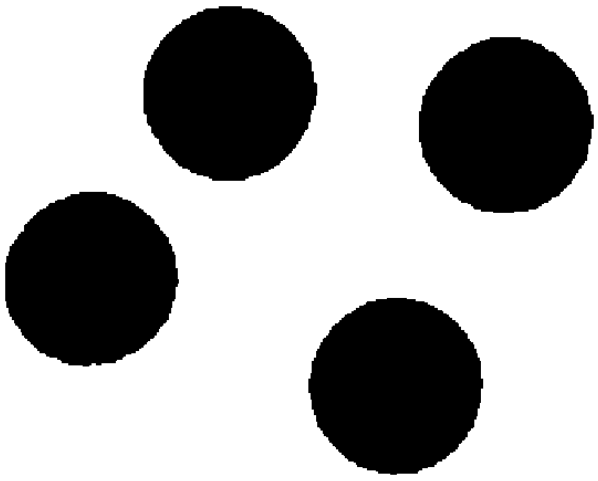

Figura 3 – Erosão de img_M

D) Obtivemos a imagem inversa da imagem atual da seguinte maneira.

```
Im1_erode_inv=imcomplement(Im1_erode);
imshow(Im1_erode_inv)
```

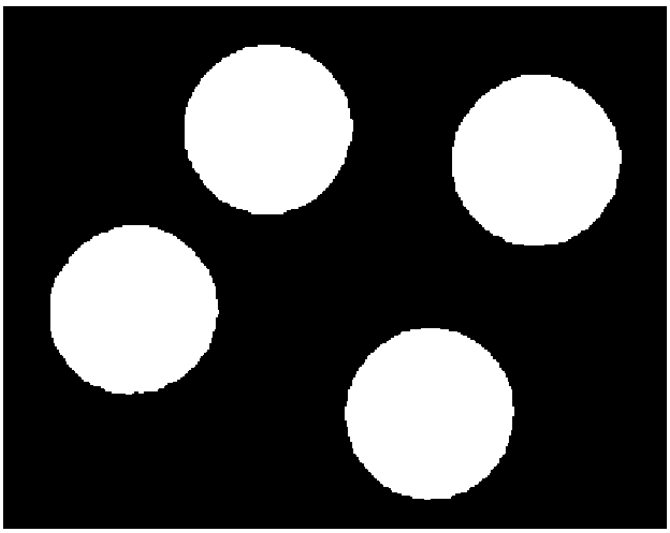

Figura 4 – Inversão de cores da imagem anterior

E) Utilizamos a seguinte função para fazer o processo de *Blob Coloring*, ou seja, atribuímos uma etiqueta de cor a cada pixel que pertence a uma região diferente na imagem.

```
Label1_bw = bwlabel(Im1_erode_inv(:,:,1),4);
im1_blob_coloring_4 = label2rgb(Label1_bw);
imshow(im1_blob_coloring_4);
```

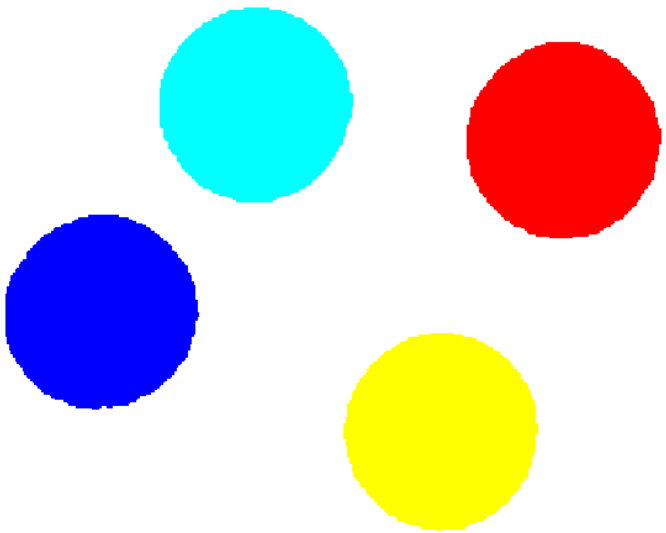

Figura 5 – Cada região contém cor diferente

F) O conjunto de instruções apresentado descreve um processo de manipulação de imagens usando a linha de comando. Inicia-se lendo uma imagem a partir de um ficheiro e visualizando-a. Em seguida, são aplicadas várias transformações na imagem, como binarização, erosão e inversão. O resultado é visualizado em cada etapa. Na etapa de B*lob Coloring*, é atribuída uma etiqueta de cor a cada região diferente da imagem.

G) Com base nesses resultados, é possível propor um método que utilize a segmentação por cor para calcular as áreas das diferentes regiões em pixels. Esse método seria uma forma de análise quantitativa das diferentes regiões identificadas na imagem.

H) Com o mesmo código, testou-se também o ficheiro de imagens pecas1.jpg utilizada no laboratório 3, onde, a imagem original é a seguinte:

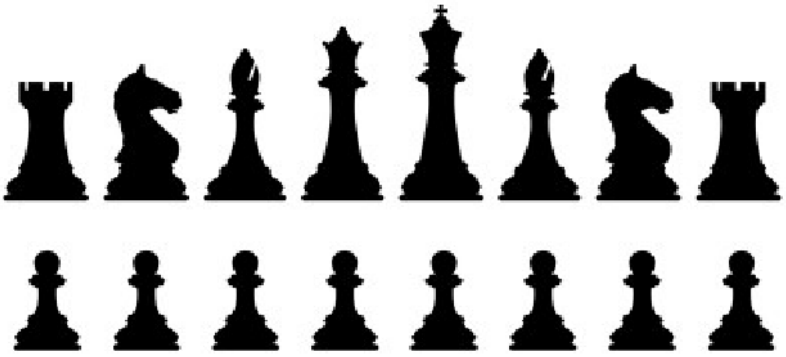

Figura 6 - Imagem pecas1

Após converter a imagem para preto e branco, foi utilizado o Erode de 4 pixeis, tal como aconteceu com a imagem anterior das moedas, mas, neste caso não era necessário, pois, não existem pixels de cores diferentes no interior das peças.

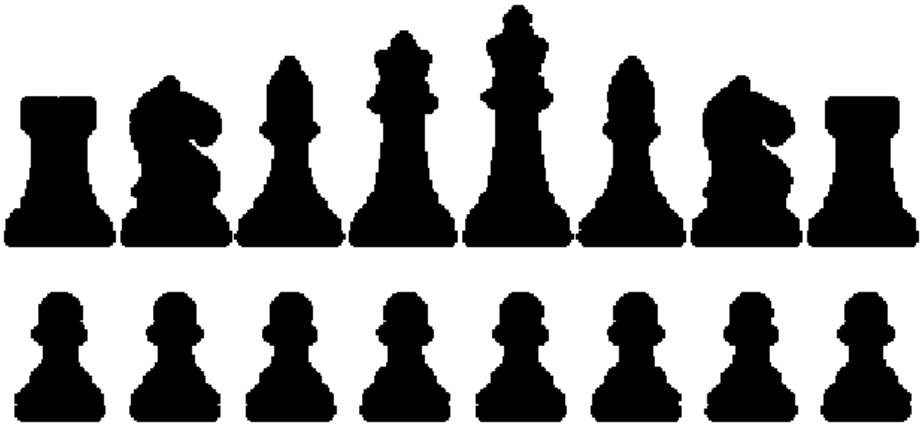

Figura 7 - Erode das peças

Inversão de cores:

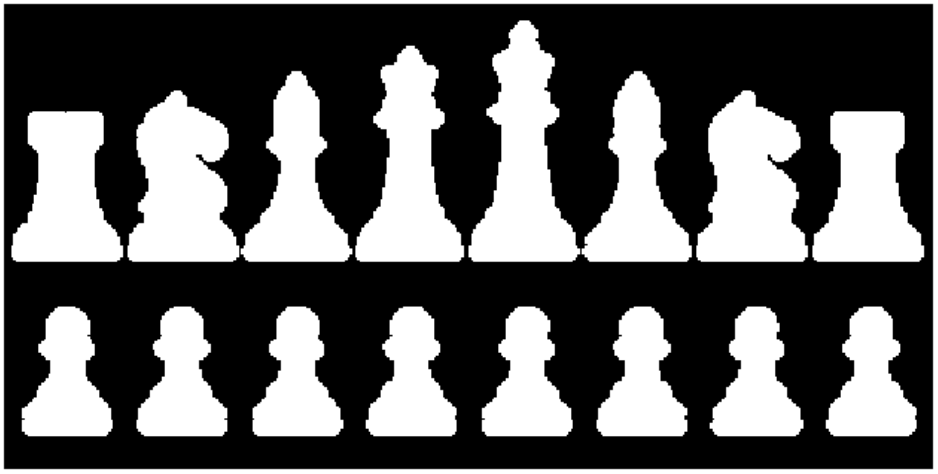

Figura 8 - Inversão de cores das peças

Resultado final:

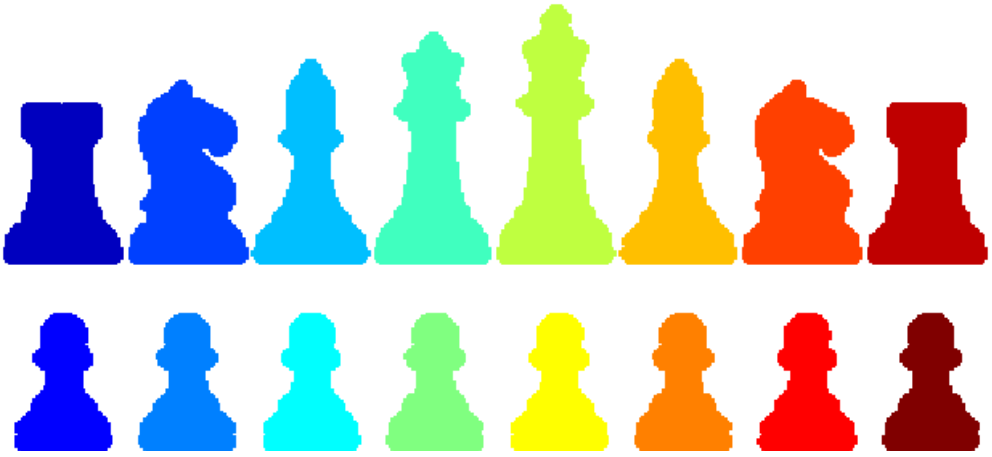

Figura 9 - Resultado final das Peças1.jpg

2.2 Exercícios - *Blob Coloring II*

A) Lemos o ficheiro de imagem "img1.jpg" e atribuiu-se à variável Im2.

```
Im2 = imread('img1.jpg');
imshow(Im2)
```

Figura 10 – Imagem img1.jpg

B) Utilizando a função correspondente, transformamos a imagem numa versão binarizada com o limiar de 0,5, tal com anteriormente:

```
Im2_bw = im2bw(Im2, threshold); % Aplicar o limiar para obter a imagem
binarizada
imshow(Im2_bw); % Exibir a imagem binarizada
```

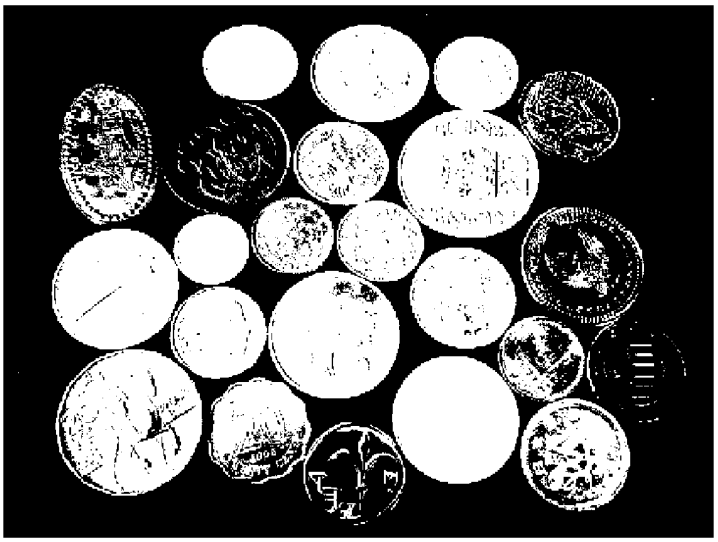

Figura 11 – Binarização de img1.jpg

C) Aplicamos a transformação de dilatação na imagem com o objetivo de preencher ao máximo o interior das moedas. Para isso, utilizamos um elemento estruturante representado por uma matriz de 3x3 preenchida com o valor '1'.

```
matriz3x3=ones(3,3);
Elem_estr_im2 = strel(matriz3x3);
Im2_erode=imdilate(Im2_bw, Elem_estr_im2);
imshow(Im2_erode)
```

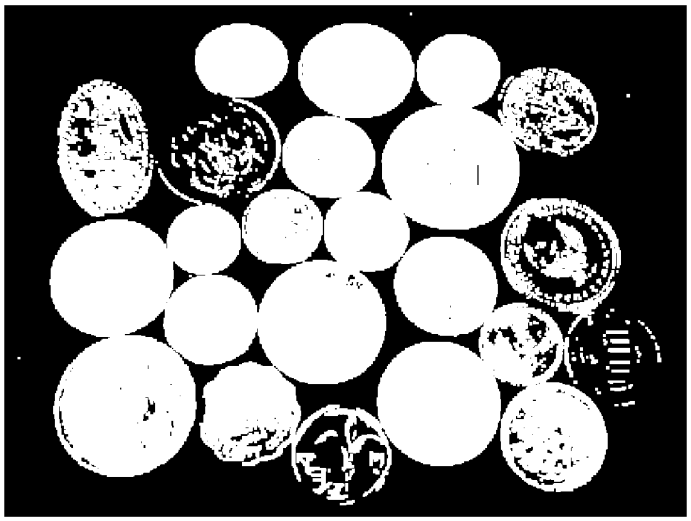

Figura 12 – Dilatação utilizando uma matriz 3x3

D) Utilizamos a seguinte função para fazer o processo de B*lob Coloring*, ou seja, atribuímos uma etiqueta de cor a cada pixel que pertence a uma região diferente na imagem.

```
Label2_4 = bwlabel(Im2_erode(:,:,1),4);
im2_blob_coloring_4 = label2rgb(Label2_4);
imshow(im2_blob_coloring_4);
```

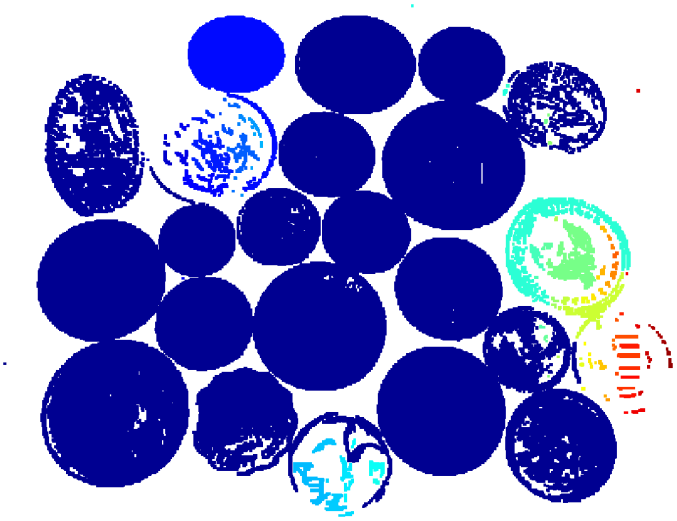

Figura 13 – Cada região contém cor diferente com 4 vizinhos

E) Realizamos novamente o processo de B*lob Coloring*, mas desta vez utilizando 8 vizinhos.

```
Label2_8 = bwlabel(Im2_erode(:,:,1),8);
im2_blob_coloring_8 = label2rgb(Label2_8);
imshow(im2_blob_coloring_8);
```

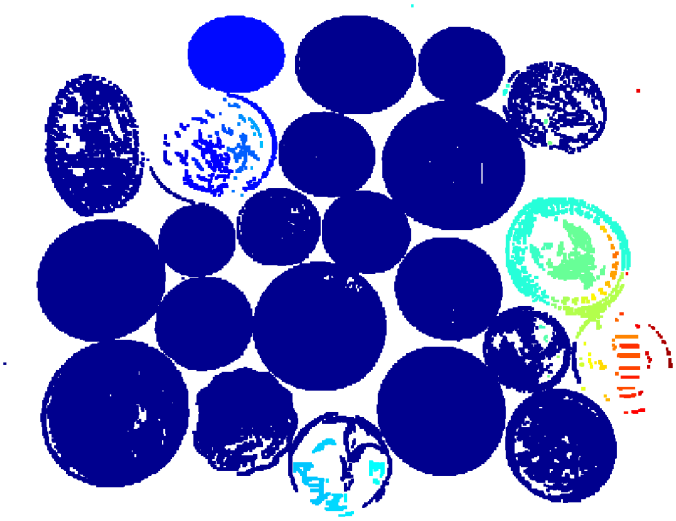

Figura 14 – Cada região contém cor diferente com 8 vizinho

F) Repetiu-se processo a partir da etapa c), mas agora foi utilizado um elemento estruturante representado por uma matriz de 9x9 na transformação morfológica de dilatação.

```
%2.2.F)
matriz9x9=ones(9,9);
Elem_estr_im2_9x9 = strel(matriz9x9);
Im2_dilate9x9=imdilate(Im2_bw, Elem_estr_im2_9x9);
imshow(Im2_dilate9x9)


%2.2.D)
Label2_4 = bwlabel(Im2_dilate9x9(:,:,1),4);
im2_blob_coloring_4 = label2rgb(Label2_4);
imshow(im2_blob_coloring_4);


%2.2.E)
Label2_8 = bwlabel(Im2_dilate9x9(:,:,1),8);
im2_blob_coloring_8 = label2rgb(Label2_8);
imshow(im2_blob_coloring_8);
```

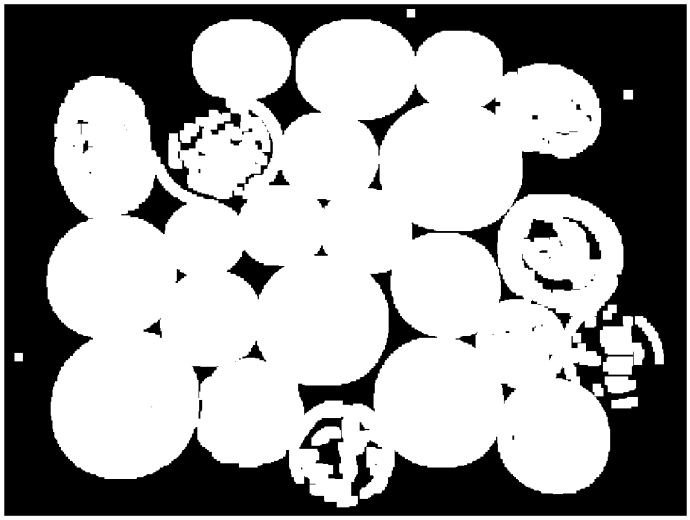

Figura 15 – Dilatação utilizando uma matriz 9x9

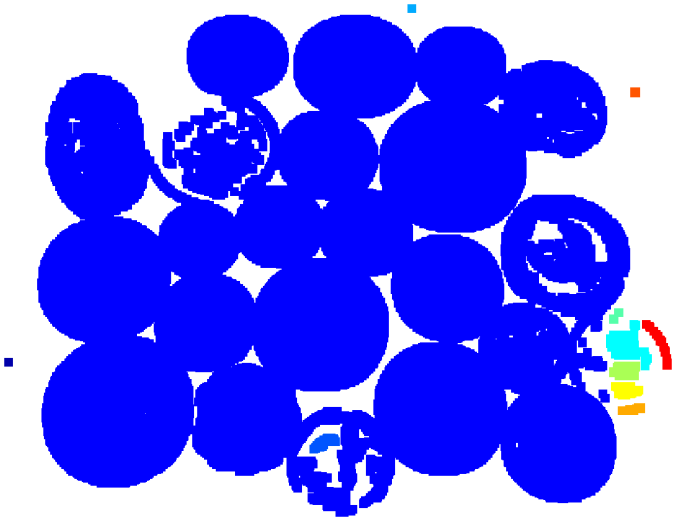

Figura 16 – Cada região contém cor diferente com 4 vizinhos

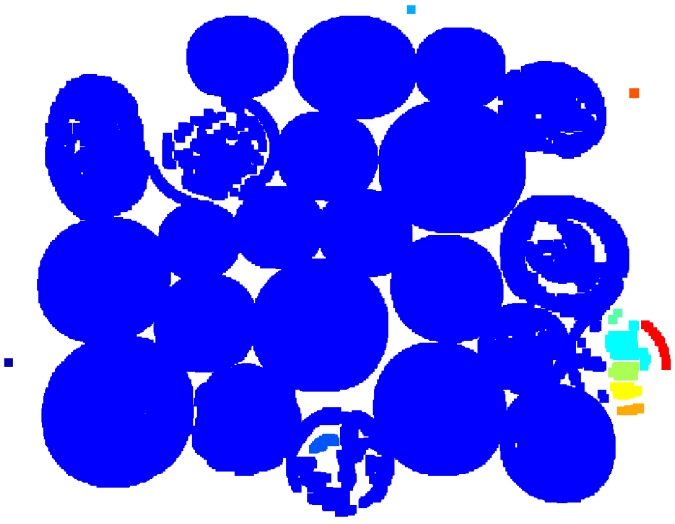

Figura 17 – Cada região contém cor diferente com 8 vizinhos

Comente os resultados, comparando o resultado obtido na etapa d) com o resultado da etapa e), explicando o funcionamento do método e a influência da transformação morfológica ao utilizar dois elementos estruturantes de dimensões diferentes.

Comentário:

O conjunto de instruções descreve um processo de manipulação de imagem usando a linha de comando. Inicia-se lendo uma imagem a partir de um ficheiro e visualizando-a. Em seguida, são aplicadas várias transformações na imagem, como binarização, dilatação e *blob coloring*. O objetivo é preencher o interior das moedas e identificar regiões diferentes na imagem, tal como em 2.1)

Na etapa e), é realizado o B*lob Coloring* com 8 vizinhos, enquanto na etapa f) utiliza-se um elemento estruturante de dimensões maiores (9x9) na dilatação.

Comparando os resultados obtidos nas etapas d) e e), o *Blob Coloring* com 4 vizinhos e 8 vizinhos, respetivamente, pode-se observar diferenças na identificação das regiões. O uso de 8 vizinhos permite uma maior conectividade entre os pixéis, o que pode resultar em regiões maiores. Por outro lado, o uso de 4 vizinhos pode dividir regiões maiores em sub-regiões menores.

Na etapa f), ao utilizar um elemento estruturante de dimensões maiores (9x9) na dilatação, espera-se que o preenchimento das moedas seja ainda mais intenso, resultando em regiões mais uniformes e conectadas.

Em resumo, as transformações morfológicas, como a dilatação, e a escolha do elemento estruturante podem influenciar diretamente nos resultados do processo de *Blob Coloring* e na identificação das regiões na imagem. A dimensão do elemento estruturante afeta a conectividade entre os pixéis e pode resultar em regiões maiores ou menores. A escolha adequada desses parâmetros é essencial para obter resultados desejados na segmentação e análise de regiões numa imagem.

3 Funções .m

3.1 Clustering - Implementação do algoritmo *K-Mean Clustering* (Tons Cinzento):

A) Utilizando um ficheiro .m, desenvolva o código para uma função que aplique o algoritmo de *Clustering K-Means* em imagens em tons de cinza.

B) Na linha de comando, leu-se a imagem armazenada no ficheiro "cameraman.tif" (disponível no ambiente Matlab) e atribuiu-se à variável ImK1.

```
ImK1 = imread('cameraman.tif');
imshow(ImK1); title('Imagem Original');
```

Figura 18 – Imagem original

C) Aplicou-se a função desenvolvida à imagem armazenada em ImK1, escolhendo pelo menos dois valores diferentes para o número de clusters (K), como 3 e 10.

```
%Segmenta a imagem usando k valores de clustering.
[L,Centers] = imsegkmeans(ImK1,3);


% Sobrepoe a imagem etiquetada sobre a imagem 2D
B1 = labeloverlay(ImK1,L);
figure; imshow(B1);
title('Imagem com cluster de valor 3')
```

Figura 19 - Imagem com cluster de valor 3

```
%Novamente o mesmo para K=10
[L,Centers] = imsegkmeans(ImK1,10);


C1 = labeloverlay(ImK1,L);
figure; imshow(C1);
title('Imagem com cluster de valor 10')
```

Figura 20 - Imagem com cluster de valor 10

D) Na linha de comando, leu-se a imagem armazenada no ficheiro "lua1.jpg" e atribuiu-se à variável ImK2.

```
ImK2 = imread('lua1.jpg');
imshow(ImK2); title('Imagem Original');
```

Figura 21 – imagem lua

E) Aplicou-se a função desenvolvida à imagem armazenada em ImK2, escolhendo o número de clusters (K) como 3 e 4.

```
%Segmenta a imagem usando k valores de clustering.
[M,Centers] = imsegkmeans(ImK2,3);


% Sobrepoe a imagem etiquetada sobre a imagem 2D
B2 = labeloverlay(ImK2,M);
figure; imshow(B2);
title('Imagem com cluster de valor 3')
```

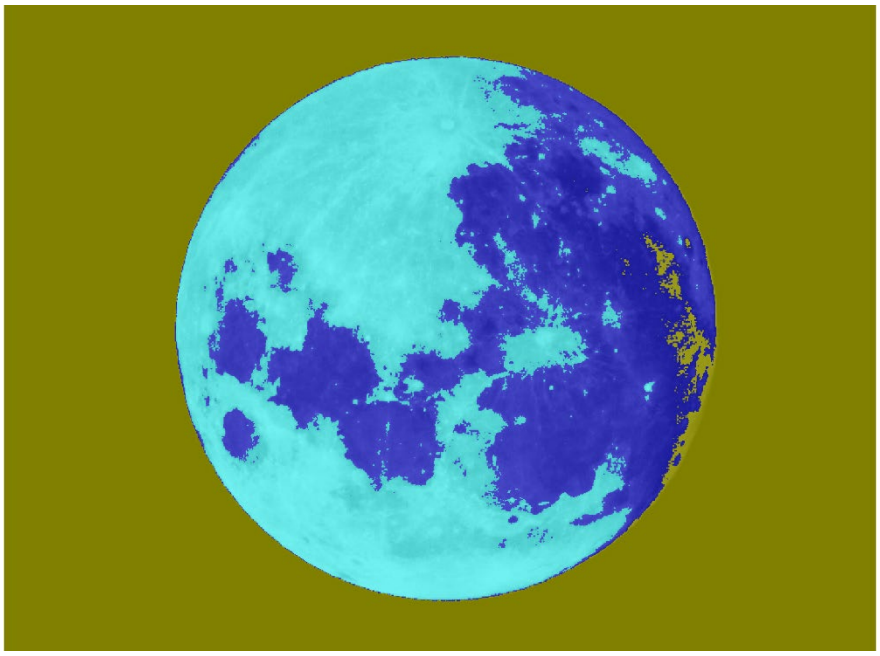

Figura 22 – Imagem Lua com cluster de valor 3

```
%Novamente o mesmo para K=10
[L,Centers] = imsegkmeans(ImK1,10);


C1 = labeloverlay(ImK1,L);
figure; imshow(C1);
title('Imagem com cluster de valor 10')
```

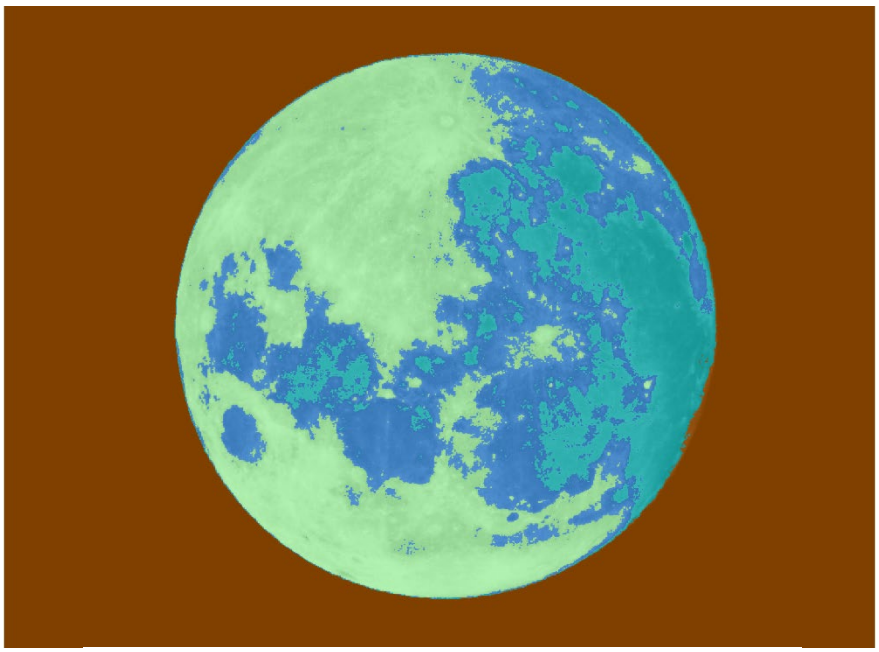

Figura 23 – Imagem Lua com cluster de valor 4

3.2 Clustering - Implementação do algoritmo *K-Mean Clustering* (Cores)

A) Leu-se a imagem armazenada no ficheiro "cores.jpg" e atribuiu-se à variável ImKC1.

```
ImKC1 = imread('cores.jpg');
imshow(ImKC1); title('Imagem Original');
```

Figura 24 - Imagem cores.jpg

F) Aplicou-se a função desenvolvida anteriormente à imagem armazenada em ImKC1, escolhendo dois valores diferentes para o número de clusters (K), como 5 e 10.

```
%Segmenta a imagem usando k valores de clustering.
[L,Centers] = imsegkmeans(ImKC1,5);


% Sobrepoe a imagem etiquetada sobre a imagem 2D
B1 = labeloverlay(ImKC1,L);
figure; imshow(B1);
title('Imagem com cluster de valor 3')




%Novamente o mesmo para K=10
[L,Centers] = imsegkmeans(ImKC1,10);


C1 = labeloverlay(ImKC1,L);
figure; imshow(C1);
title('Imagem com cluster de valor 10')
```

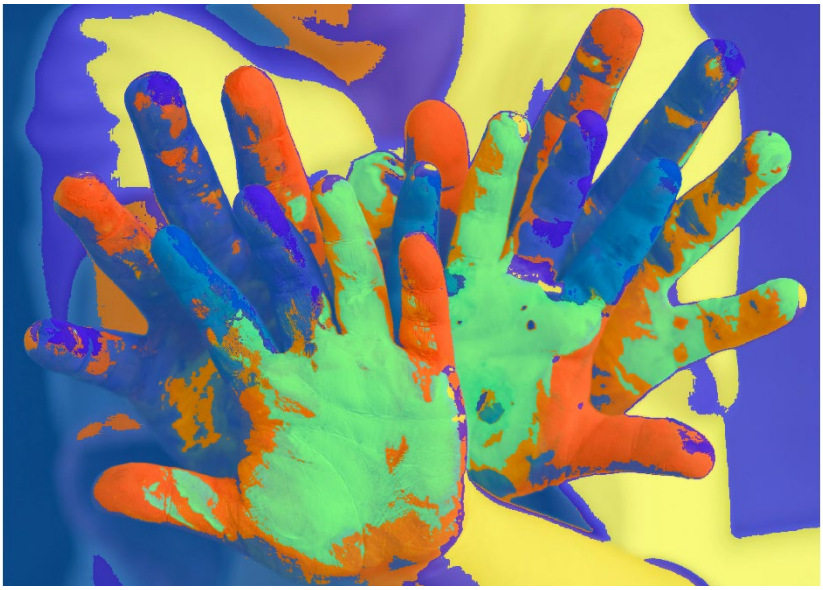

Figura 25 - Imagem cores.jpg com cluster de valor 3

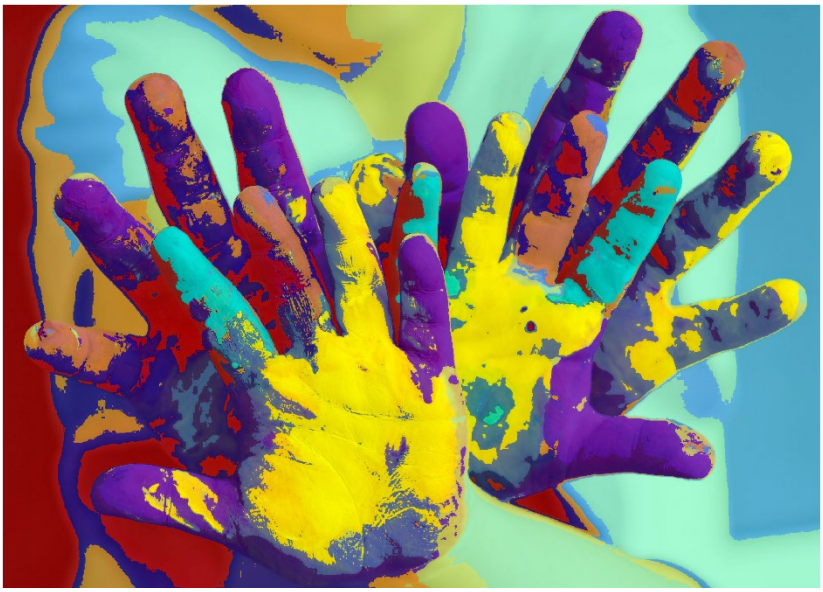

Figura 26 - Imagem cores.jpg com cluster de valor 10

B) Na linha de comando, leu-se a imagem armazenada no ficheiro "img3.jpg" e atribua-a à variável ImKC2.

```
ImKC2 = imread('img3.jpg');
imshow(ImKC2); title('Imagem Original');
```

Figura 27 - Imagem im3.jpg

C) Aplique a função desenvolvida à imagem armazenada em ImKC2, escolhendo diferentes valores para o número de clusters (K), como 2, 4, 6 e 10. Comente os resultados levando em consideração as características da imagem e a função do algoritmo.

```
%Segmenta a imagem usando k valores de clustering.
[M,Centers] = imsegkmeans(ImKC2,2);


% Sobrepoe a imagem etiquetada sobre a imagem 2D
B2 = labeloverlay(ImKC2,M);
figure; imshow(B2);
title('Imagem com cluster de valor 2')



%Novamente o mesmo para K=4
[L,Centers] = imsegkmeans(ImKC2,4);


C2 = labeloverlay(ImKC2,L);
figure; imshow(C2);
title('Imagem com cluster de valor 4')
```

```
%Novamente o mesmo para K=6
[L,Centers] = imsegkmeans(ImKC2,6);


D2 = labeloverlay(ImKC2,L);
figure; imshow(D2);
title('Imagem com cluster de valor 6')




%Novamente o mesmo para K=10
[L,Centers] = imsegkmeans(ImKC2,10);


E2 = labeloverlay(ImKC2,L);
figure; imshow(E2);
title('Imagem com cluster de valor 10')
```

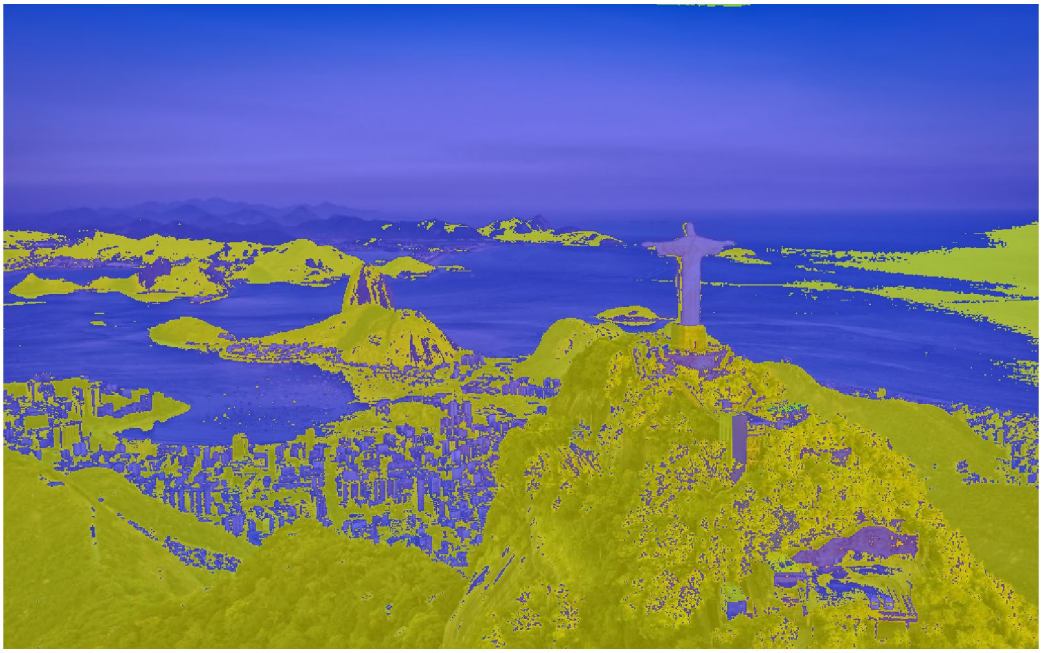

Figura 28 - Imagem im3.jpg Imagem com cluster de valor 2

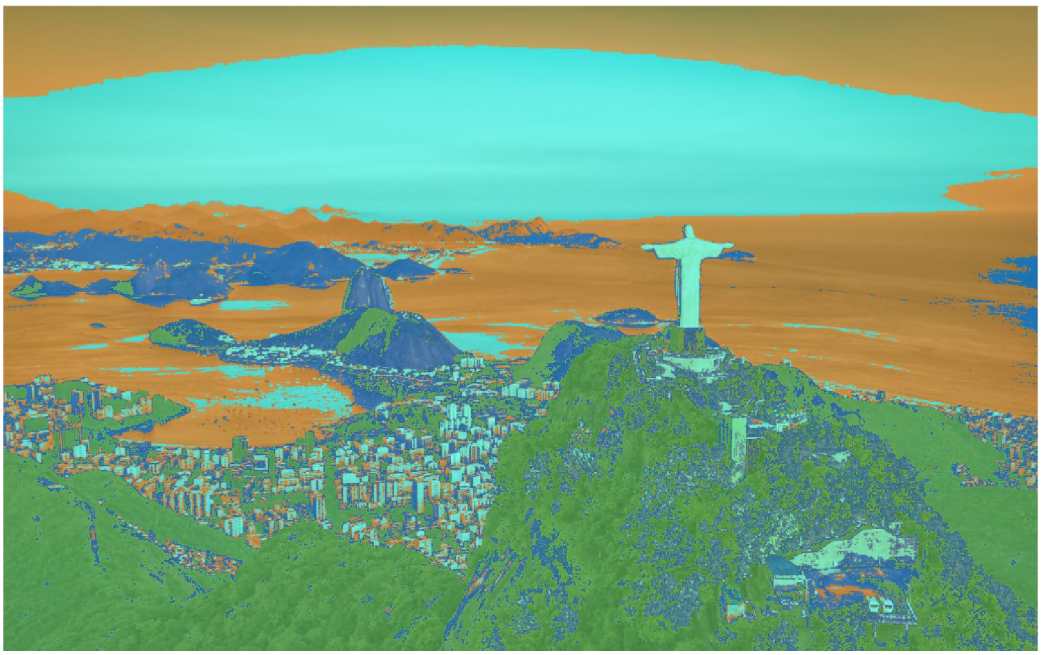

Figura 29 - Imagem im3.jpg Imagem com cluster de valor 4

Figura 30 - Imagem im3.jpg Imagem com cluster de valor 6

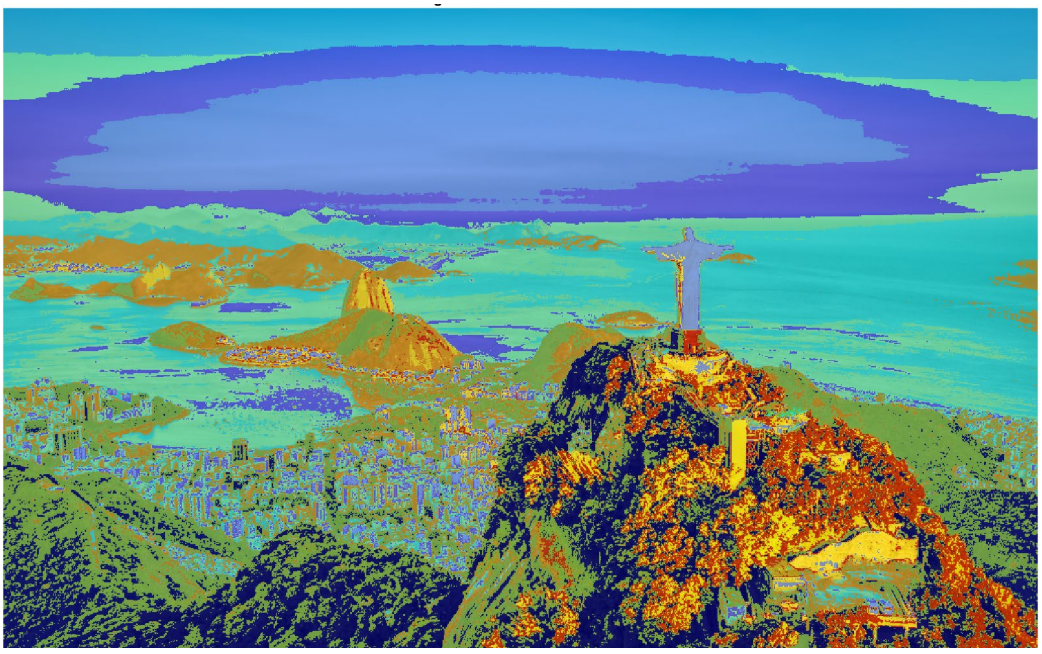

Figura 31 - Imagem im3.jpg Imagem com cluster de valor 10

Comentário:

As instruções fornecidas descrevem a implementação do algoritmo de *Clustering K-Means* para imagens em tons de cinza e a cores. O algoritmo tem como objetivo segmentar as imagens em grupos com base nos valores dos pixéis.

No cenário de *Clustering* em tons de cinza, o algoritmo é aplicado nas imagens "cameraman.tif" e "lua1.jpg" com diferentes valores de K (número de *clusters*). Os resultados são avaliados com base no desempenho do algoritmo e nas características das imagens.

No *clustering* com cores, o algoritmo é aplicado nas imagens "cores.jpg" e "img3.jpg". O número de clusters (K) é escolhido de forma diferente para cada imagem, e os resultados são analisados levando em consideração as características da imagem e o funcionamento do algoritmo.

O algoritmo de *Clustering K-Means* que pode ser usado para segmentar imagens, pois agrupa pixéis semelhantes com base na similaridade das suas características, permitindo a identificação de regiões ou objetos distintos numa imagem. A escolha do número de clusters (K) é fundamental e depende da imagem específica e do nível de segmentação desejado.

Os resultados e a análise do processo de *clustering* fornecem *insights* sobre o quão bem o algoritmo separa diferentes regiões nas imagens com base nos valores dos pixéis. Ao avaliar os resultados e considerar as características da imagem, podemos avaliar a eficácia do algoritmo na segmentação de imagens em tons de cinza e em cores.

4 Conclusões e perspetivas de trabalho futuro

4.1 Conclusão

Para concluir, o objetivo deste trabalho de laboratório foi introduzir os conceitos de processamento de imagem por meio da Expansão Local de Regiões e do *Clustering*. Durante o processo, exploramos técnicas como *Blob Coloring* e a implementação do algoritmo *K-Mean Clustering* em tons de cinza e cores.

O *Blob Coloring* permitiu-nos identificar e delimitar regiões interligadas na imagem, possibilitando a segmentação e análise de diferentes partes da mesma. Essa técnica é útil em várias aplicações, como reconhecimento de objetos, deteção de bordas e análise de texturas.

Por outro lado, a implementação do algoritmo *K-Mean Clustering* proporcionou-nos uma abordagem para agrupar pixéis semelhantes em diferentes tons de cinza ou cores. Isso permite a categorização de elementos na imagem com base nas suas características visuais, possibilitando a identificação de padrões e estruturas complexas.

A combinação dessas técnicas de processamento de imagem abre caminho para uma ampla gama de aplicações práticas. Desde a análise de imagens médicas e reconhecimento de objetos na visão computacional até a segmentação de regiões de interesse em fotografias. Essas abordagens desempenham um papel importante no processamento de imagem e na interpretação de informações visuais.

Ao explorar a Expansão Local de Regiões e o *Clustering*, este trabalho de laboratório permitiu-nos uma introdução significativa no campo do processamento de imagem. Compreendemos a importância da segmentação e análise de regiões conectadas, assim como a capacidade de agrupar elementos similares em tons de cinza e de cores.

Bibliografia

[1] Mathworks, “Matlab”

[2] Microsoft Corporation, “Word,” [Offline]. Disponivel em: www.office.com

[3] Professor Tito Amaral e Professor Pedro Vitoriano, Visão Artificial, disponibilizado no Moodle do IPS, 2022/2023

[4] Mathworks Help Center, consulta de como adicionar parâmetros à função “hist”, disponível em: https://www.mathworks.com/help/matlab/ref/hist.html

[5] Mathworks Help Center, consulta da função “bwlabel”, disponível em: https://www.mathworks.com/help/images/ref/bwlabel.html

[6] Mathworks Help Center, consulta da função “label2rgb”, disponível em: https://www.mathworks.com/help/images/ref/label2rgb.html