ENGENHARIA INFORMÁTICA E MULTIMÉDIA

2022/2023

PROCESSAMENTO DE IMAGENS E VISÃO

Relatório do Trabalho Prático nº1

6 de novembro de 2022

Docente: Pedro Miguel Torres Mendes Jorge

Trabalho realizado por:

António Luís Ferreira, 47500
Tomás Gomes, 48614

Conteúdo

Lista de Figuras

Lista de Tabelas

1 Introdução

No âmbito da Unidade Curricular de Processamento de Imagem e Visão, para o trabalho laboratorial número um, foi proposta a realização de um algoritmo com os objetivos seguintes:

- Desenvolver algoritmo de visão por computador, capaz de contar automaticamente a quantia em dinheiro (moedas), colocado em cima de uma mesa.
- Familiarização com a biblioteca de funções OpenCV (Open Source Computer Vision) para programação de aplicações de visão por computador em tempo real (para linguagem de programação Python)

Para uma melhor contextualização do problema deste trabalho prático, o objetivo consiste em construir um algoritmo que conte uma determinada quantia de dinheiro(moedas de euro), colocadas em cima de uma mesa de superfície homogénea e clara, observada por uma câmara montada num tripé e ajustada de modo a que o plano do sensor seja paralelo ao plano da mesa.

Consistência e robustez são duas particularidades a que teremos de ter atenção visto que o nosso algoritmo terá de ser resistente a algumas perturbações, tais como, a presença de objetos, diferentes moedas no campo de visão, existência de pequenas sombras e o eventual contacto indesejado entre objetos.

Para o desenvolvimento e treino deste algoritmo, foram fornecidas algumas imagens de treino pelo docente que leciona a UC.

2 Construção do algoritmo

A construção deste algoritmo foi realizada em duas fases distintas, a fase de treino, onde extraímos as características intrísecas das imagens e a fase de teste, onde classificámos os resultados obtidos na fase de treino.

Na fase de treino, as imagens foram binarizadas e melhoradas, para uma melhor extração de componentes conexos e características geometricas, tais como, a área, o perímetro e a circularidade. Por fim, após a verificação dos contornos correspondentes a cada moeda, foram guardados esses valores que serviram de base para a construção do nosso modelo de classificação das mesmas.

Na fase de teste, o processo é exatamente o mesmo que o processo realizado na fase de treino em relação à binarização e ao melhoramento das imagens, com exceção do último passo, em que em vez de ser construído um modelo de classificação novo, iremos utilizar o modelo criado na fase de treino para a classificação.

3 Fase de treino

3.1 Binarização

Na binarização das imagens, foram calculados e analisados os histogramas de cada canal de cores da imagem e também da imagem convertida em tons de cinzento. O critério de escolha consistiu basicamente em escolher o canal que tivesse uma melhor separação de classes presentes, neste caso, fundo e objetos da imagem.

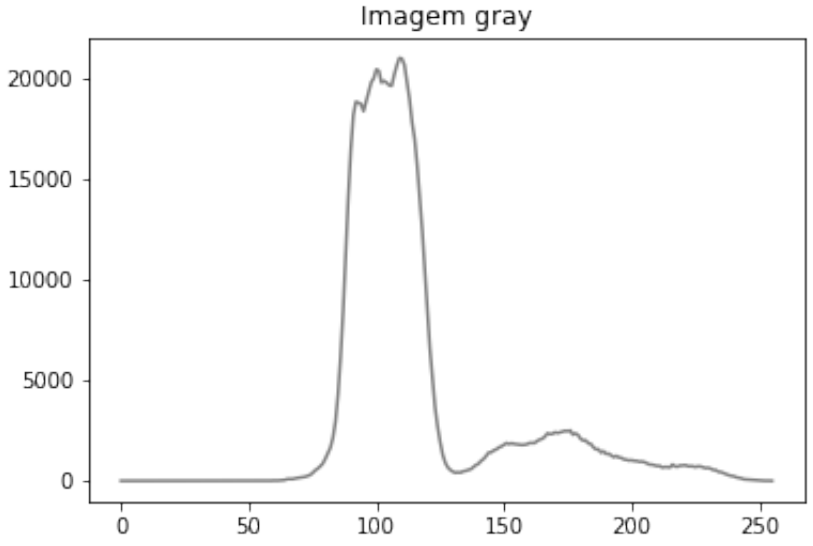

Figura 1: Histograma do plano cinzento da imagem ’P1000697s.jpg’

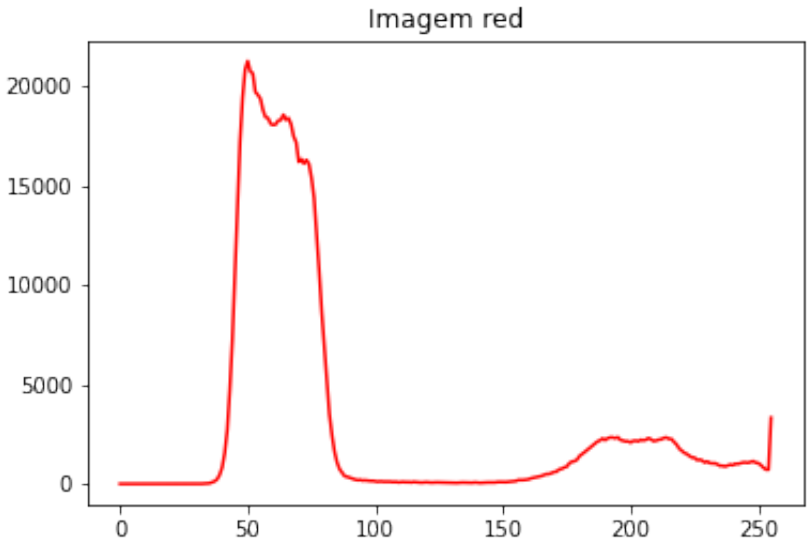

Figura 2: Histograma do plano vermelho da imagem ’P1000697s.jpg’

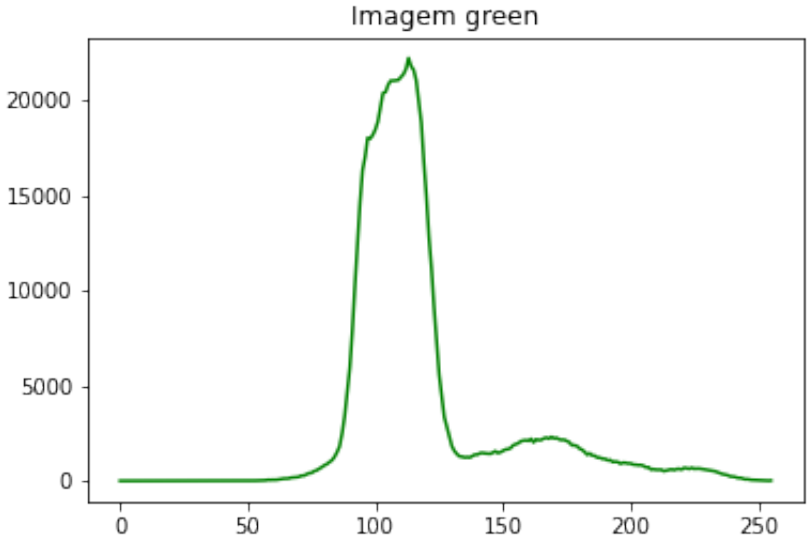

Figura 3: Histograma do plano verde da imagem ’P1000697s.jpg’

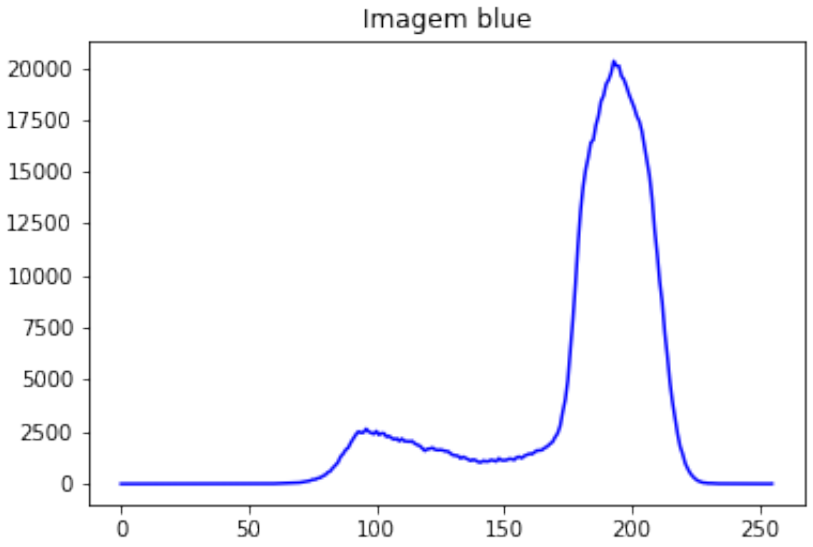

Figura 4: Histograma do plano azul da imagem 'P1000697s.jpg'

Após a análise dos quatro histogramas, podemos claramente concluir que o canal "Red"seria mais vantajoso para realizar a binarização, visto que há uma maior separação entre classes sendo assim mais fácil de cálcular o limiar.

Ao analisar as imagens que representam os 3 canais de cor, e a imagem em níveis de cinzento, é possível confirmar este pressuposto visto que há um maior ”contraste”entre as moedas e o fundo.

Figura 5: Conversão para níveis de cinzento da imagem ’P1000697s.jpg’

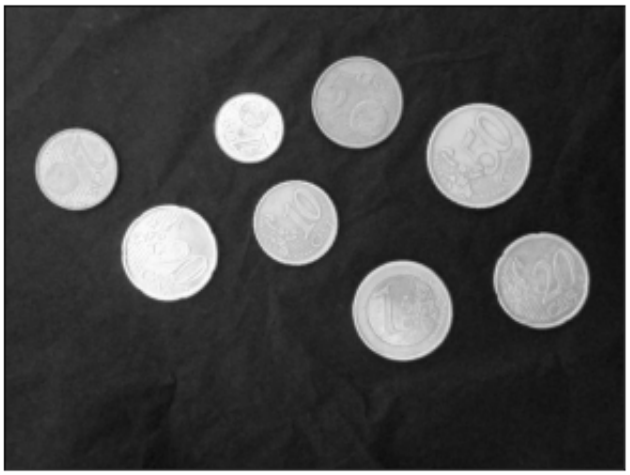

Figura 6: Plano vermelho da imagem ’P1000697s.jpg’

Figura 7: Plano verde da imagem ’P1000697s.jpg’

Figura 8: Plano azul da imagem ’P1000697s.jpg’

Assim, foi utilizado o plano de vermelho para a binarização, obtendo a imagem binária calculando o seu limiar usando o método **cv2.threshold**(red, 0, 255, `cv2.THRESH_OTSU`), sendo os seus parâmetros de entrada:

- red, imagem com plano de vermelho;
- 0, valor mínimo que será ignorado (irrelevante);
- 255, valor máximo a utilizar;
- `cv2.THRESH_OTSU`, expressão usada para ser utilizado o Algoritmo de Otsu para o cálculo do limiar.

3.2 Melhoramento de Imagens

Logo após ser realizada a binarização, foram encontradas algumas falhas no interior das moedas, possivelmente resultantes das pequenas sombras do seu interior.

Para combater estas falhas, foi utilizada a função **cv2.blur**(), que consiste em fazer uma média ponderada dos pixeis do tamanho da janela indicada, neste caso, 8x8, e substituir o seu pixel central por esse valor resultante.

Este procedimento ajudou a remover o ruído interior das moedas e a suavizar alguns contornos.

Nos exemplos seguintes, foram comparadas ambas as imagens resultantes da binarização, primeiro sem filtro blur, e a segunda após a sua aplicação.

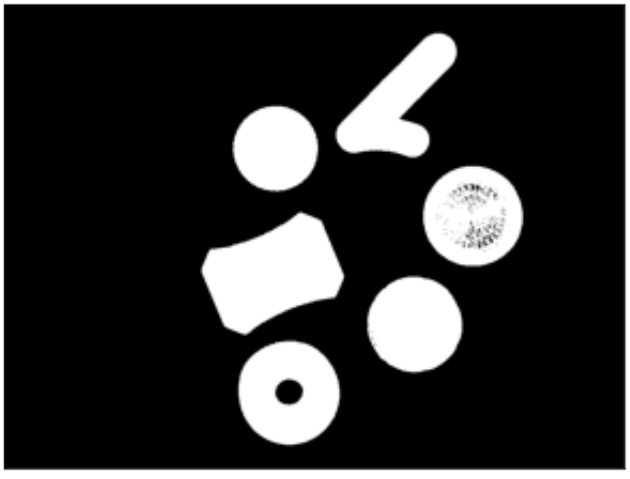

Figura 9: Binarização da imagem ’P1000710s.jpg’ sem filtro blur()

Figura 10: Binarização da imagem 'P1000710s.jpg' com filtro blur()

De seguida, após a binarização, foram utilizados operadores morfológicos de forma a separar objetos que se encontravam em contacto, para uma melhor classificação das moedas no futuro.

É de notar que, para utilizarmos estes operadores morfológicos, é necessária a utilização de elementos estruturantes, estes que são matriz de pixeis que assumem valores apenas de 0 ou 1, estes devem ser utilizados consoante a forma dos objetos da imagem que queremos afetar e, por isso, neste caso, foi utilizada uma elipse de 11x11, implementada da seguinte forma:

cv2.getStructuringElement(`cv2.MORPH_ELLIPSE`, (11, 11), (-1,-1))

- `cv2.MORPH_ELLIPSE`, elemento estruturante;
- (11, 11), dimensão do elemento estruturante;
- (-1,-1), definir posição da âncora do elemento estruturante, neste caso, o seu centro;

Por fim, foram utilizados os operadores morfológicos de erosão e dilatação implementados pelos métodos **cv2.erode()** e **cv2.dilate()** do OpenCV.

A erosão é uma operação que reduz a área onde é aplicada, sendo neste caso usada para separar objetos que, por ventura, estejam em contacto. O valor do pixel de saída é o valor mínimo de todos os pixeis na vizinhança do pixel de entrada. É atribuído o valor máximo (1 ou 255) aos pixeis exteriores.

A dilatação, por outro lado, serve para aumentar a área onde é aplicada, dado que o valor do pixel de saída é o valor máximo de todos os pixeis na vizinhança do pixel de entrada. É atribuído o valor mínimo (0) aos pixeis exteriores.

Para a efetiva separação de algumas moedas que se encontravam encostadas, foram utilizadas quatro erosões seguidas de uma dilatação, como forma a tentar preservar o máximo de área perdida nas erosões, de modo a não juntar de novo as moedas.

3.3 Extracção de componentes conexos

Depois de termos a imagem preparada, o processo seguinte é quantificar diferentes tipos de contornos. Para tal desfeixo foi utilizado dois métodos da bibloteca **OpenCV**, **findContours()** e **drawContours()**.

O método **cv2.findContours()** é implementado da seguinte forma:

- `binary_img.copy()`, cópia da imagem binária para calcular os contornos;
- `cv2.RETR_TREE`, flag que permite obter todos os contornos e a sua hierarquia;
- `cv2.CHAIN_APPROX_NONE`, falg que permite guardar todos os pontos dos contornos.

Contorno	Next Sibling	Previous Sibling	First Child	Parent
Contorno 0	2	-1	1	-1
Contorno 1	-1	-1	-1	0
Contorno 2	3	0	-1	-1
Contorno 3	4	2	-1	-1
Contorno 4	5	3	-1	-1
Contorno 5	6	4	-1	-1
Contorno 6	-1	5	-1	-1

Tabela 1: Matriz de Hierarquia de Contornos da Imagem 'P1000710s.jpg'.

O método **cv2.drawContours()** permite visualizar os contornos que foram obtidos com o método anteriormente explicado(**cv2.findContours()**)

Figura 11: Contornos obtidos da imagem 'P1000710s.jpg'

3.4 Extracção de propriedades

Após obter todos os contornos dos objetos presentes na imagem, o passo seguinte foi analisar cada um deles, com o objetivo de ignorar os contornos que não pertecem às moedas, sendo eles de forma circular ou não.

Para cada contorno calculámos a sua área, usando o método **cv2.counterArea(cnt)**, e o seu respetivo perímetro usando o método **cv2.arcLenght(cnt)**. Com a área e o perímetro de cada objecto da imagem calculámos a sua circularidade, $C = \frac{4*\pi*área}{perimetro^2}$.

Com a Circularidade calculada para destinguir um objecto em forma de círculo dos restantes, optou-se que a Circularidade tenha de ser superior ou igual a 0,8.

O próximo obstáculo deste trabalho foi a destinção das moedas para os restantes objectos que passaram na condição da Circularidade. Para tal, é avaliado a 2º e 3º colunas da Matriz de Hierarquia dos contornos. Em caso de uma dessas colunas tenha um valor diferente de -1, quer dizer que o contorno é Pai ou Filho de outro contorno.

3.5 Modelo de classificação

Para a criação de um modelo de classificação foi inevitável recorrer ao uso de um Método de Classificação Supervisionada.

Este consistiu na análise das áreas de todas as moedas e a partir das mesmas, foram criados vários intervalos de decisão consoante o valor da moeda(classes), como podemos observar na figura seguinte:

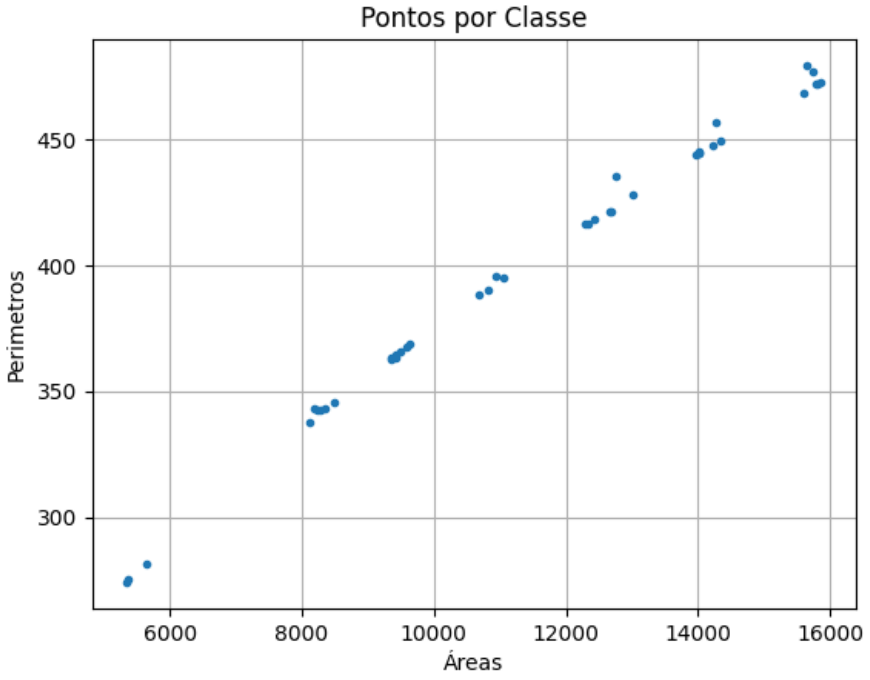

Figura 12: Grafico com a distribuição das Classes, tendo em conta vetores com a área e o perímetro de cada moeda.

Na construção deste modelo, como apenas utilizámos as áreas, os perímetros foram uma componente que não afetava o resultado. Reduzimos a dimensionalidade do modelo na componente principal, como podemos observar na seguinte figura:

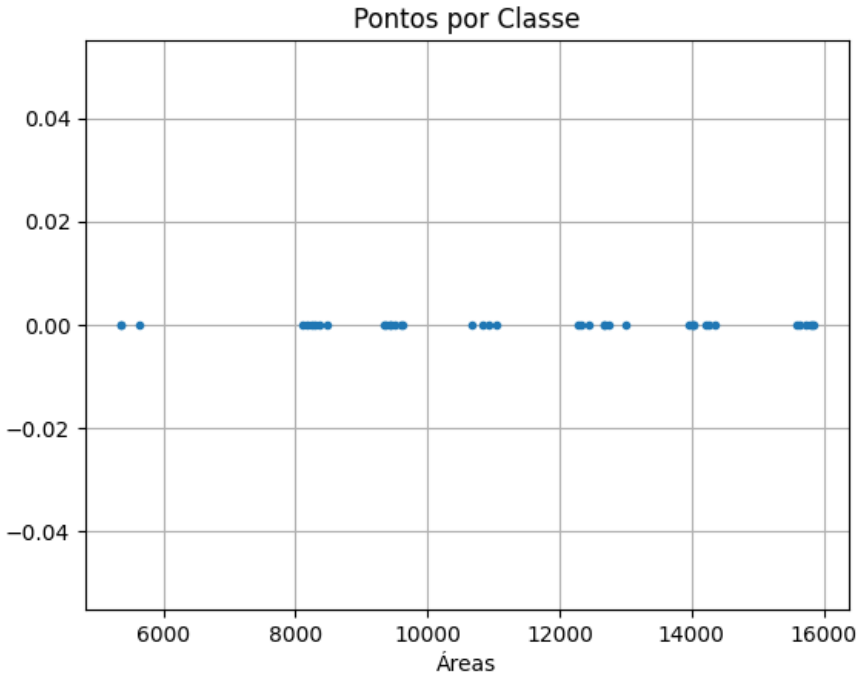

Figura 13: Grafico com a distribuição das Classes, apenas com a área de cada moeda.

Com base nos dados do gráfico acima representado, calculou-se a distância média entre cada classe, e foi estimado onde o intervalo poderia começar, para o caso da moeda de 2 euros.

Assim, resultante deste modelo de classificação, foi originada a seguinte tabela:

Classes	
Moedas	Intervalos de decisão
1 cêntimo	[5300, 5700]
2 cêntimos	[8000, 8500]
5 cêntimos	[10400, 11200]
10 cêntimos	[9000, 9700]
20 cêntimos	[12000, 13100]
50 cêntimos	[15500, 16000]
1 euro	[13200, 14400]
2 euros	[17100, $+\infty$[

Tabela 2: Tabela de intervalos de decisão de cada classe de moedas.

4 Fase Teste

Na Fase de teste foram usados os dois métodos criados na fase anterior.

4.1 Classifica e Quantifica

O método **Classifica** recebe uma área como parâmetro de entrada, e consoante a sua área incrementa o contador da respetiva moeda. O método retorna um tuplo de posições conhecidas com o contador da moeda classificada a um.

Já o método **Quantifica** recebe o tuplo retornado pelo método anterior e atribui o seu valor.

4.2 Resultado

A utilização de ambos estes métodos, originaram as seguintes imagens, com as quantias monetárias respetivas representadas no canto superior esquerdo:

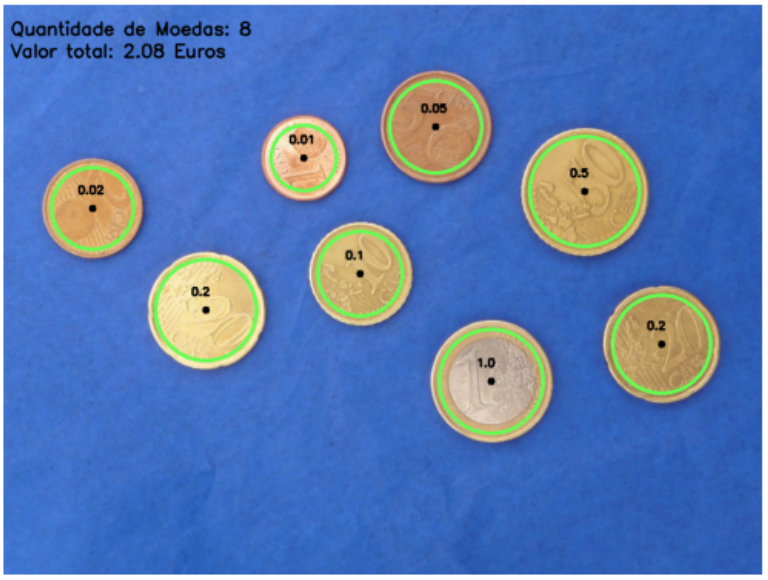

Figura 14: Imagem 'P1000697s.jpg' com a respetiva quantia monetária.

Para uma melhor visualização do contorno da moeda, foi cálculado o centro de cada moeda utilizando o método **cv2.moments()**, que recebe a imagem binarizada e, calcula as coordenadas do centro da moeda que vão ser usadas à posteriori para refazer os respetivos contornos.

Figura 15: Imagem ’P1000710s.jpg’ com a respetiva quantia monetária.

5 Conslusões

Todas as imagens do conjunto de treino foram corretamente classificadas, quer no valor monetário individual de cada moeda, como no número total de moedas por imagem. Foi também implementado um contador que mostra a quantia total de dinheiro sobre o plano da imagem.

Referências

[1] Prof. Arnaldo Abrantes. *Analise de Imagens Binárias.* 2022 https://2223moodle.isel.pt/pluginfile.php/1182403/mod_resource/content/2/Cap%C3%ADtulo%203%20-%20An%C3%A1lise%20Imagens%20Bin%C3%A1rias.pdf

[2] OpenCV. *Open Source Computer Vision.* 2022. https://opencv.org/

[3] MatPlotLib. *MatPlotLib - Visualization with Python.* 2022 https://matplotlib.org/stable/plot_types/index.html

[4] Numpy. *Numpy Reference.* 2022. https://numpy.org/doc/stable/reference/index.html,