Deteção de corrosão em superfícies metálicas

Pedro Gonçalves 88859 Pedro Silva 89228

***Resumo* – Detetor de corrosão em superfícies metálicas através da leitura de imagens. Baseia-se no nível RGB de cada píxel para detetar a corrosão. No final apresenta a imagem original, a imagem final com os píxeis de superfícies metálicas que possuem corrosão e um histograma com a mais variada informação sobre a imagem final.**

***Abstract* - Corrosion detector on metallic surfaces by reading images. It is based on the RGB level of each pixel to detect corrosion. At the end it presents the original image, the final image with the pixels of metallic surfaces that have corrosion and a histogram with the most varied information about the final image.**

I. INTRODUÇÃO

Neste relatório, vamos abordar as várias tecnologias que foram desenvolvidas em OpenCV, e que foram implementadas neste projeto, desde a leitura das imagens à deteção de superfícies metálicas com corrosão.

II. SELEÇÃO DE IMAGEM

O nosso programa faz o carregamento de uma imagem a partir do terminal, lendo uma imagem de cada vez. Para testar 2 imagens, será necessário correr o programa 2 vezes. A imagem lida será o nome apresentado no primeiro argumento.

III. PADRÕES BGR

Para a identificação de corrosão, focamo-nos nas suas cores. Como a corrosão normalmente possui tons alaranjados e acastanhados, criámos diversos intervalos BGR (Blue, Green, Red), para identificar os tons previamente mencionados.

```
# Low Red
rust.append([([30, 30, 60], [50, 40, 70])])
rust.append([([25, 30, 70], [40, 45, 80])])
rust.append([([40, 60, 90], [60, 65, 110])])
# Mid Red
rust.append([([70, 57, 115], [83, 90, 130])])
rust.append([([80, 100, 145], [90, 110, 180])])
rust.append([([35, 65, 115], [45, 90, 145])])
rust.append([([20, 45, 140], [40, 65, 165])])
rust.append([([40, 85, 170], [50, 100, 185])])
rust.append([([5, 25, 115], [25, 45, 130])])
# High Red
rust.append([([95, 125, 200], [100, 140, 205])])
rust.append([([115, 150, 220], [120, 160, 225])])
```

Figura 1- Intervalos de BGR para deteção de corrosão

IV. JUNÇÃO DE INTERVALOS BGR

Em cada imagem que é lida, temos que utilizar todos os intervalos que possuímos para detetar todos os tons de corrosão. Isto é feito através de um ciclo, que itera sobre todos os intervalos. Dentro desse ciclo existe um novo ciclo que itera sobre o conteúdo dos tuplos.

```
for i in range(1, len(rust)):
    for (lower, upper) in rust[i]:
        lower = np.array(lower, dtype = "uint8")
        upper = np.array(upper, dtype = "uint8")
        mask = cv2.inRange(img, lower, upper)
        output = cv2.bitwise_and(img, img, mask=mask)
```

Figura 2- Ciclo para criação da nova imagem

Neste ciclo, usando a livraria "*numpy*" de *Python* e a função "*inRange()*" conseguimos então criar uma máscara com os valores pretendidos. Esta máscara é posteriormente

aplicada a cada um dos intervalos, juntando todos os píxeis e criando assim a imagem final.

```
final = cv2.bitwise_or(final, output)
```

Figura 3- Junção do resultado atual ao mais recente

V. ANÁLISE DE IMAGENS

O nosso programa foi testado com várias imagens, as quais possuem ou não superfícies corroídas e foi desenvolvido para passar os testes com imagens sem corrosão, fazendo com que tivéssemos que alterar e criar vários intervalos de modo a tornar o mais fiável possível. Contudo, por vezes não é possível, pois existem objetos do quotidiano com cores e tons muito semelhantes às superfícies metálicas corroídas.

Ao realizar um teste com uma imagem com ferrugem, o nosso programa vai originar uma nova imagem, que apenas terá visíveis os pixeis que possuem ferrugem, todos os outros terão o valor RGB 0 (preto).

Figura 4 – Imagem com corrosão

Figura 5 – Resultado da imagem anterior

Figura 6 – Imagem com corrosão

Figura 7 - Resultado da imagem anterior

Figura 8 – Imagem com corrosão

Figura 9 – Resultado da imagem anterior

Testando o nosso *software* com uma imagem que não possui objetos que se podem confundir com ferrugem, os resultados são bastante satisfatórios.

Figura 10 – Imagem sem corrosão

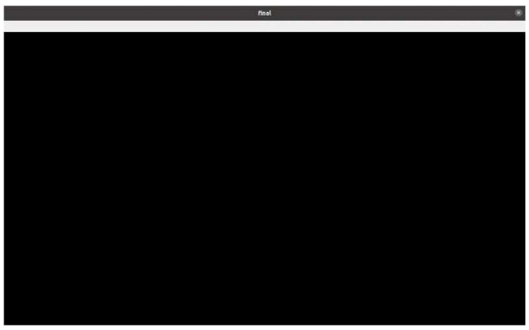

Figura 11 – Resultado da imagem anterior

Podemos observar que o resultado será uma imagem apenas com pixeis com valores RGB 0, o que indica que não há nenhuma superfície corroída na primeira figura.

No entanto, há algumas imagens sem corrosão que segundo o nosso algoritmo, apresentam marcas com corrosão apesar desta não existir.

Figura 12 – Imagem sem corrosão

Figura 13 – Resultado da imagem anterior

Como se pode ver, aparecem alguns pixeis na figura 13 que estão contidos num dos intervalos referidos previamente, mas que não correspondem à corrosão, mas sim a terra. Estes erros são devidos ao facto de a cor de alguns píxeis de terra serem bastante parecidos aos da corrosão de um metal.

VI. HISTOGRAMA

Após a análise do resultado, podemos ainda analisar o histograma obtido. Esse histograma contém um somatório do número de píxeis por valor de RGB. As diferentes cores representam as 3 categoria do RGB.

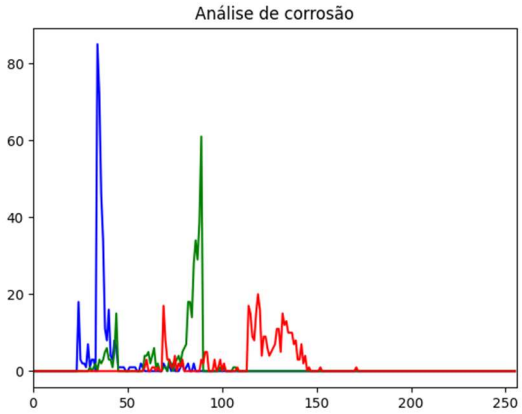

Figura 14 – Histograma da figura 9

Perante este histograma podemos concluir que a imagem possui poucas superfícies corroídas, pois tem um somatório de píxeis reduzido na imagem resultado. Podemos também observar que os azuis da cor da ferrugem são todos dentro da mesma gama (à volta dos 40), os verdes à volta dos 90 e finalmente, os vermelhos são os que se alteram mais, tendo as zonas mais variadas.

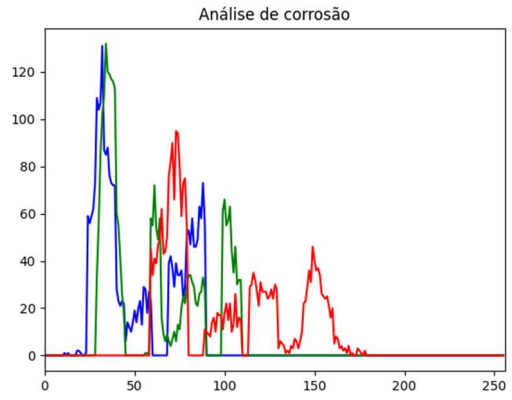

Figura 15 – Histograma da figura 5

Observando agora este histograma podemos reparar que a imagem possui níveis de corrosão mais altos que a anterior, visto que além de ter valores mais altos de azuis, verdes e vermelhos, estão também mais distribuídos, o que aumenta a contagem total de pixeis.

VII. CLASSIFICAÇÃO DA CORROSÃO

Em relação à classificação da corrosão, utilizámos uma função para calcular o número de píxeis que continham ferrugem, e consoante esse número, aplicamos diferentes graus de gravidade da corrosão. Essa mensagem é apresentada no terminal.

```
Número total de pixeis: 677160
Número total de pixeis com ferrugem: 18546
Percentagem de pixeis com ferrugem: 2.7%
Imagem com pouca ou nenhuma corrosao
```

Figura 16 – Classificação da corrosão

VII. ARMAZENAMENTO DO RESULTADO

Também é permitido ao utilizador armazenar a imagem resultante com o nome que pretender, como 2º argumento na chamada do programa (se não colocar nenhum nome, será guardado com o predefinido).

VIII. CONCLUSÃO

Com este projeto desenvolvemos capacidades de análise de imagens através do OpenCV. Foi particularmente desafiante encontrar os intervalos corretos para que o programa não confundisse objetos do quotidiano com corrosão de superfícies metálicas.

IX. REFERÊNCIAS

https://answers.opencv.org/question/178394/detection-of-rust-with-opencv-python/?fbclid=IwAR1Lztp2SBI73kiGvkekMHdWPXO16HbhvNjWfhdgQh7T5gCs1v2mXA8Bp_Q

https://docs.opencv.org/master/d1/db7/tutorial_py_histogram_begins.html