Relatório Visão Por Computador
Aula Prática 1

Luís Silva, nmec 88888
luisfgbs@ua.pt
Mestrado em Robótica e Sistemas Inteligentes
Universidade de Aveiro

I. Introdução

Este relatório foi redigido no âmbito da UC Visão Por Computador, lecionada pelo professor Paulo Dias e visa encapsular o método de resolução dos exercícios e resultados obtidos

Novembro 2021

II. OpenCv

A. First Example

Neste exercício recebemos o ficheiro Aula_01_ex_01.py e pede-se que se note a forma como se abre uma imagem usando OpenCv.

1) Leitura de Imagem: O path da imagem é recebido como argumento do programa e depois esse mesmo path é utilizado em conjunto com o comando cv2.imread(path_to_image, flag) para fazer a leitura da imagem. O parâmetro flag indica a forma como a leitura da imagem deve ser feita, pode assumir os valores indicados na Fig.1

2) Dimensões da imagem: As dimensões da imagem podem ser obtidas empregando o seguinte código image.shape, onde image é o handle da imagem lida anteriormente. De notar que, neste caso, como a imagem é a cores, serão devolvidos 3 valores, altura, largura e e canais de cor

3) Visualização da imagem: Para que visualize uma imagem, basta que à função cv.imshow(), sejam passados 2 argumentos. O primeiro deve ser uma string representando o nome a ser atibuído à janela e o segundo deve ser o handle de imagem lida anteriormente

B. Direct pixel manipulation

Neste exercício pretende-se que exploremos a manipulação direta do valor dos píxeis da imagem
Pede-se que se crie uma cópia da imagem e que nesta se alterem para 0 todos os píxeis que tenham um valor inferior a 128 e que, após as tranformações se visualize em simultâneo as duas imagens.

Enumerator
IMREAD_UNCHANGED Python: cv.IMREAD_UNCHANGED
IMREAD_GRAYSCALE Python: cv.IMREAD_GRAYSCALE
IMREAD_COLOR Python: cv.IMREAD_COLOR
IMREAD_ANYDEPTH Python: cv.IMREAD_ANYDEPTH
IMREAD_ANYCOLOR Python: cv.IMREAD_ANYCOLOR
IMREAD_LOAD_GDAL Python: cv.IMREAD_LOAD_GDAL
IMREAD_REDUCED_GRAYSCALE_2 Python: cv.IMREAD_REDUCED_GRAYSCALE_2
IMREAD_REDUCED_COLOR_2 Python: cv.IMREAD_REDUCED_COLOR_2
IMREAD_REDUCED_GRAYSCALE_4 Python: cv.IMREAD_REDUCED_GRAYSCALE_4
IMREAD_REDUCED_COLOR_4 Python: cv.IMREAD_REDUCED_COLOR_4
IMREAD_REDUCED_GRAYSCALE_8 Python: cv.IMREAD_REDUCED_GRAYSCALE_8
IMREAD_REDUCED_COLOR_8 Python: cv.IMREAD_REDUCED_COLOR_8
IMREAD_IGNORE_ORIENTATION Python: cv.IMREAD_IGNORE_ORIENTATION

Fig. 1. Lista de flags disponíveis

1) Leitura da imagem e dimensões: O path da imagem continua a ser recebido como argumento mas desta vez, para facilitar a manipulação, utilizo no cv2.imread() a flag cv2.IMREAD_GRAYSCALE para que cada píxel da imagem seja representa apenas por um valor

2) Manipulação dos píxeis: O acesso, comparação e alteração do valor dos píxeis é feito iterativamente da seguinte forma

```
for i in range(height):
    for j in range(width):
        if img_copy[i,j]<128:
            img_copy[i,j]=0
```

Fig. 2. Processo de acesso, comparação e alteração dos valores dos píxeis

3) Resultados: De notar que ao empregar o cv2.imshow(), para se criar duas janelas, os nomes atribuídos às mesmas devem ser distintos, caso contrário, apenas uma janela será

criada e apenas uma das imagens será visualizada. O resultado da manipulação é o seguinte e são bem perceptíveis as zonas onde os valores foram alterados

```
# Show the image
cv2.imshow( "Original", image )
cv2.imshow( "Tampered", img_copy )
```

Fig. 3. Código para visualização da imagem original e manipulada em simultâneo

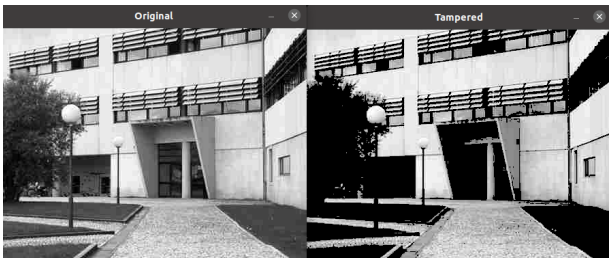

Fig. 4. Visualização da imagem original e manipulada em simultâneo

C. Image subtraction

Neste exercício pede-se que, recebendo como argumentos duas imagens (Deti.bmp e Deti.jpg) se proceda à subtração de uma pela outra.

1) Subtract: cv2.subtract(image1, image2) calcula a diferença, elemento a elemento, de 2 arrays. Obtemos assim, no final, uma nova matriz que representa a diferença entre as duas imagens

2) Resultados: Vemos que o resultado de cv.subtract(image2, image1) é igual ao de cv.subtract(image2, image1). Como as imagens são muito próximas e devido também às dimensões da imagem no relatório, os resultados ficam um pouco menos perceptíveis

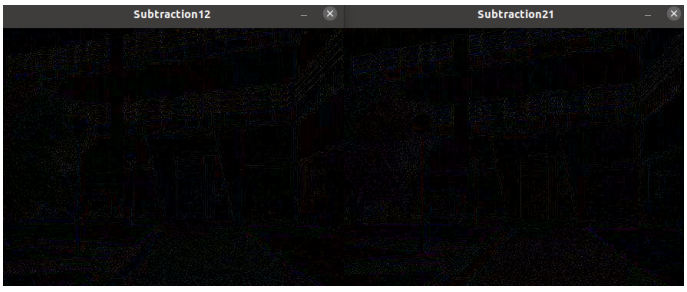

Fig. 5. Visualização diferença elemento a elemento entre imagens

D. Mouse Callback e conversão de cor

Neste execício pede-se que se permita ao utilizador que de forma interativa, selecione com o rato um pixel que deverá ser usado como centro de um círculo e que, de seguida, se desenhe o círculo na imagem
Aproveitei também este script para a resolução também do exercício 6 que pede apenas que se faça a conversão de colorspaces de uma imagem para GRAYSCALE depois de lida.

1) Leitura da imagem: Para receber os paths das imagens sobre as quais queremos trabalhar, comecei a utilizar o argparser

```
parser = argparse.ArgumentParser(description='Typing Test Option Parser')
parser.add_argument('-i', '--image', type=str, required=True,help='Full path to image file.')
args = parser.parse_args()
```

Fig. 6. Definição argparser

2) Conversão entre colorspaces: Para realizar a conversão desejada para grayscale usamos cv2.cvtColor(cv2.COLOR_BGR2GRAY)

```
# Read the image
image = cv2.imread( args.image, cv2.IMREAD_COLOR )
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) # Convert to gray
```

Fig. 7. Conversão de cor

3) Definição do Callback: A função cv2.setMouseCallback(window, handler) define um callback na janela window que será processado por handler. Para evitar variáveis globais emprega-se o partial que envia ao handler como parâmetros, neste caso, a imagem sobre a qual se quer desenhar o círculo e a janela onde se deve fazer o display dessa imagem

```
cv2.setMouseCallback(window_name, partial(onMouse, w_name=window_name, img=image))
```

Fig. 8. Definição Callback

4) Definição do Handler: Recebe do callback os parâmetros x, y, event, flags, params, sendo que x e y indicam a posição do pixel que associado ao evento.
De seguida, se o evento for um clique com o botão esquero do rato, então usando as coordenadas do pixel, desenhamos sobre a imagem um círculo de raio 10, cor vermelha e centro em (x,y)
Finalmente mostra-se a imagem final na janela correta

5) Resultados:

```python
def onMouse(event, x, y, flags, params, w_name, img):
    if event == cv2.EVENT_LBUTTONDOWN:
        cv2.circle(img, (x,y), 10, color)
        cv2.imshow(w_name, img)
```

Fig. 9. Processo de desenho do círculo

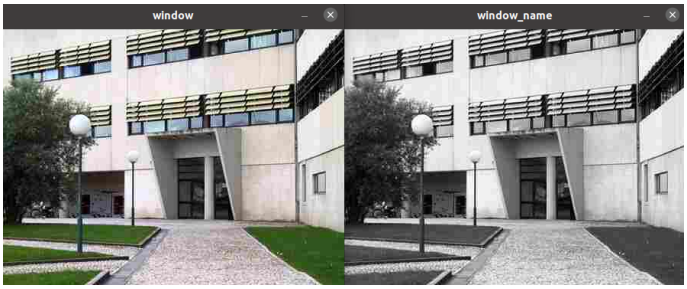

Fig. 10. Imagem original e imagem convertida para grayscale

Fig. 11. Visualização de círculos sobre a imagem