

**UNIVERSIDADE ESTADUAL DE CAMPINAS**

**MO443 – Trabalho 4**

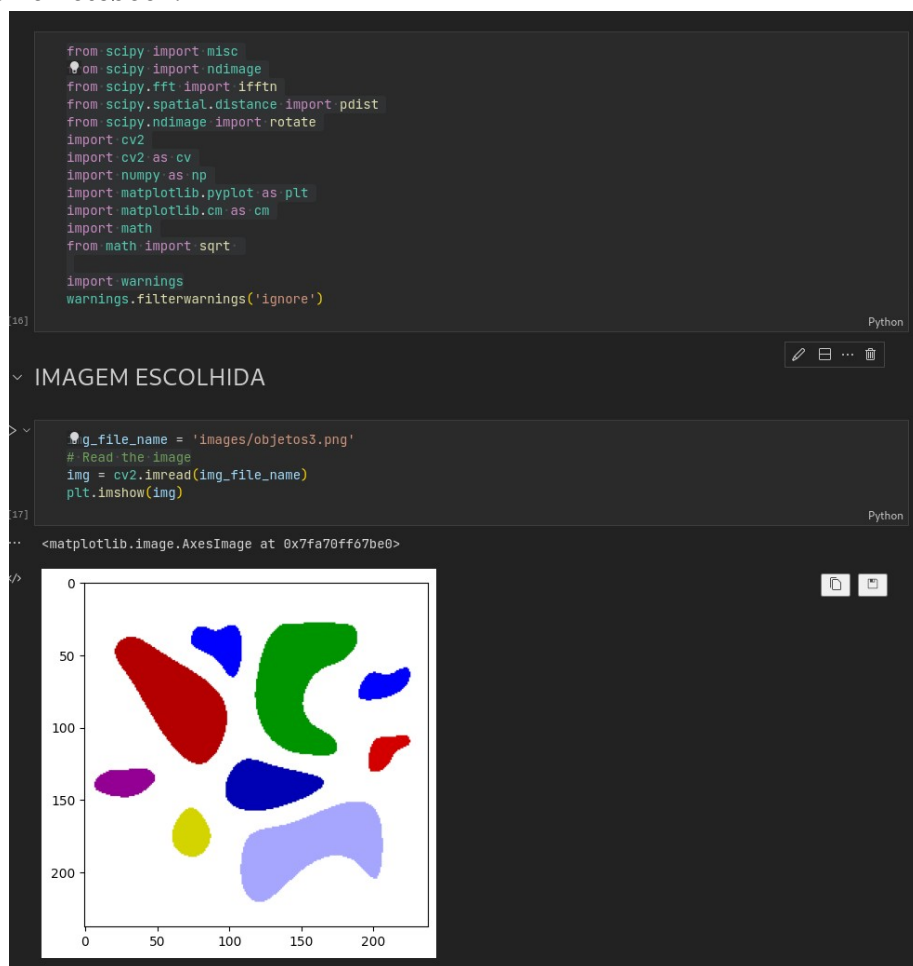
**Nicolas Guilherme Silva Moliterno - 232339**

## 1 – Introdução:

Este trabalho foi desenvolvido utilizando a plataforma de prototipação e experimentação [Jupyter](#). A intenção foi permitir que o trabalho fosse desenvolvido de forma gradual, já que as células são executadas uma por vez e os resultados são obtidos de forma incremental. Estou enviando, junto do pdf do notebook, o notebook original. A intenção é que a imagem de input pode ser alterada com bastante facilidade e basta rodar as células em sequência para obter os resultados discutidos.

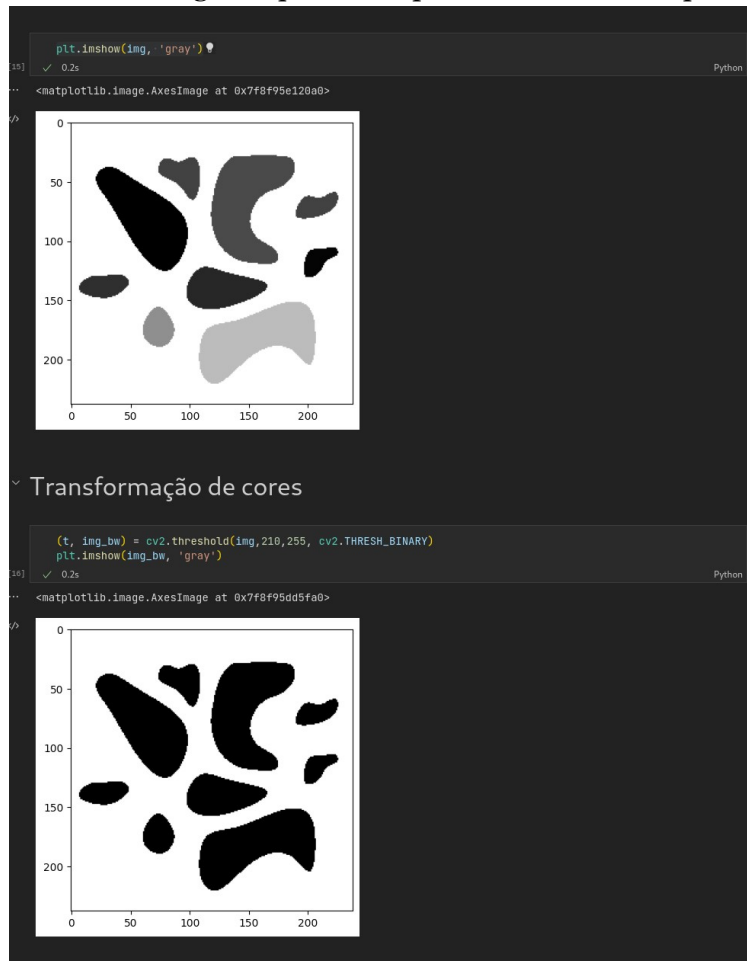
## 2 – Imagem escolhida:

O valor de `img_file_name` pode ser modificado para o caminho de outras imagens válidas no notebook.



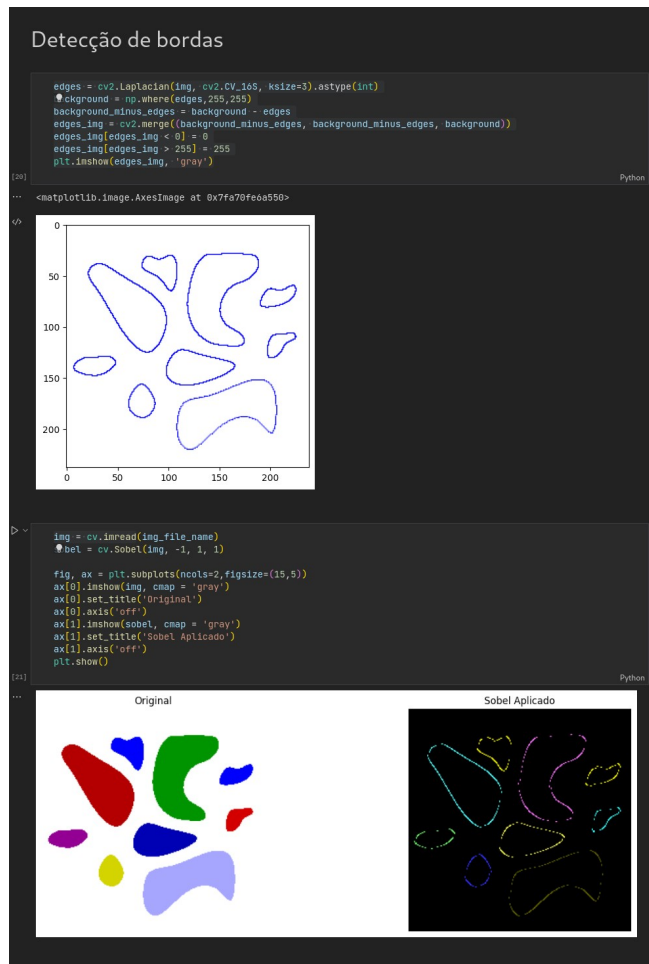
## 3 – Transformação de cores:

A imagem já foi lida em grayscale, então temos somente que ajustar a faixa de preto e criar uma imagem equivalente porém somente com preto e branco.



#### 4 – Bordas:

A detecção das bordas é feita usando o método pronto da Laplaciana, implementado pelo *OpenCV*.



## 5 – Propriedades:

Calculo da excentricidade:

$$excentricidade = \frac{\sqrt{\left(\frac{eixomenor}{2}\right)^2 + \left(\frac{eixomaior}{2}\right)^2}}{\frac{eixomenor}{2}}$$

## Extração de propriedades dos objetos

```
cs = cv2.findContours(img_bw, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)[0]
☛ = np.delete(cs, 0)
print('numero de regiões', len(cs))
```

[22]

Python

... numero de regiões 9

```
areas = np.zeros((len(cs)))
# salvar as áreas para usar no histograma
```

[23]

Python

## Calculo da excentricidade

```
def calc_ecc(c):
    ☛ (eM, em) = cv2.fitEllipse(c)[1]
    ... (aux1, aux2) = (em / 2, eM / 2) #em = eixo menor / eM = eixo maior
    ... # aux1 = em/2 ; aux2 = eM/2
    ... # ecc = sqrt(aux1*aux1 - aux2*aux2) / aux1
    ... ecc = round(np.sqrt(pow(aux1,2) - pow(aux2,2))/aux1,2)
    ... return ecc
```

[24]

Python

```
# Print das propriedades
☛ for i, c in enumerate(cs):
    mms = cv2.moments(c)
    cx, cy = (int(round(mms['m10']/mms['m00'])), int(round(mms['m01']/mms['m00'])))
    area = cv2.contourArea(c)
    areas[i] = area
    p = cv2.arcLength(c, True)
    ecc = calc_ecc(c)
    s = area/cv2.contourArea(cv2.convexHull(c))
    print('região {}>2g): área: {:.<10.2f} perímetro: {:.<10.2f} excentricidade: {:.<4.2f} solidez: {:.<4.2f
```

[25]

Python

```
... região 0: área: 716.50      perímetro: 101.98      excentricidade: 0.63 solidez: 0.98
região 1: área: 4867.00      perímetro: 311.08      excentricidade: 0.88 solidez: 0.78
região 2: área: 688.50      perímetro: 108.67      excentricidade: 0.88 solidez: 0.97
região 3: área: 1761.50      perímetro: 179.78      excentricidade: 0.88 solidez: 0.97
região 4: área: 478.00      perímetro: 94.43      excentricidade: 0.88 solidez: 0.93
região 5: área: 584.00      perímetro: 104.91      excentricidade: 0.87 solidez: 0.91
região 6: área: 3698.50      perímetro: 265.12      excentricidade: 0.91 solidez: 0.98
região 7: área: 843.50      perímetro: 125.64      excentricidade: 0.75 solidez: 0.90
região 8: área: 4107.00      perímetro: 319.42      excentricidade: 0.74 solidez: 0.75
```

+ Código

+ Markdown

📄 ... 🗑

## 6 – Histograma:

## Histograma

```
small = len(areas[areas < 1500])
medium = len(areas) - (len(areas[areas < 1500]) + len(areas[areas >= 3000]))
big = len(areas[areas >= 3000])
colors = ['blue']
labels = [small, medium, big]
plt.hist(x=areas, bins=[0, 1500, 3000, 4500], color = colors, label = labels, rwidth=.9)

result = """
    numero de regiões pequenas: {} \n
    numero de regiões medias: {} \n
    numero de regiões grandes: {}
"""
print(result.format(small, medium, big))

plt.title('Area\n', fontweight = 'bold')
plt.xlabel("Histograma de areas dos objetos")
plt.ylabel("Numero de Objetos")
plt.show()
```

Python

numero de regiões pequenas: 5

numero de regiões medias: 1

numero de regiões grandes: 3

