

Relatório Laboratorio 7

Thiago Ayres Kimura – RA: 22.221.045-2

Explicação Código (Contorno):

```
contorno.py
1  #pip install opencv-python
2
3  import math
4
5  import numpy as np
6  import cv2
7  import matplotlib.pyplot as plt
8
9  #Importa e converte para RGB
10 img = cv2.imread('./AVIAO_01.jpg')
11 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
12
13
```

Nesta primeira parte do Código importamos as bibliotecas necessárias para o funcionamento do código assim como a imagem, também convertemos a imagem para RGB na linha 11.

```
14
15 #Convertendo para preto e branco (RGB -> Gray Scale -> BW)
16 img_gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
17 a = img_gray.max()
18 _, thresh = cv2.threshold(img_gray, a/2*1.7, a, cv2.THRESH_BINARY_INV)
19
20
21 tamanhoKernel = 5
22 kernel = np.ones((tamanhoKernel, tamanhoKernel), np.uint8)
23 thresh_open = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
24
25 #Filtro de ruído (blurring)
26 img_blur = cv2.blur(img_gray, ksize=(tamanhoKernel, tamanhoKernel))
27
28 # Detecção borda com Canny (sem blurry)
29 edges_gray = cv2.Canny(image=img_gray, threshold1=a/2, threshold2=a/2)
30 # Detecção borda com Canny (com blurry)
31 edges_blur = cv2.Canny(image=img_blur, threshold1=a/2, threshold2=a/2)
32
```

Nesta parte convertemos para preto e branco, assim como definimos o tamanho do kernel. Aplicamos o filtro de ruído na linha 26, e nas linhas 29 e 31 detectamos a borda com Canny sem blurry e com blurry, respectivamente

Figure 1

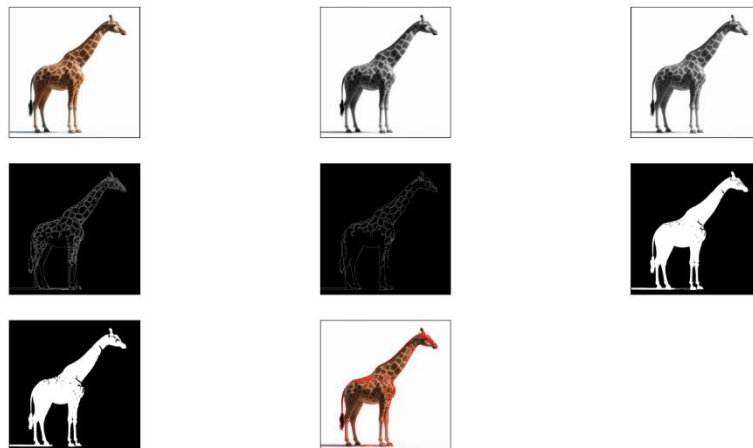


Figure 1

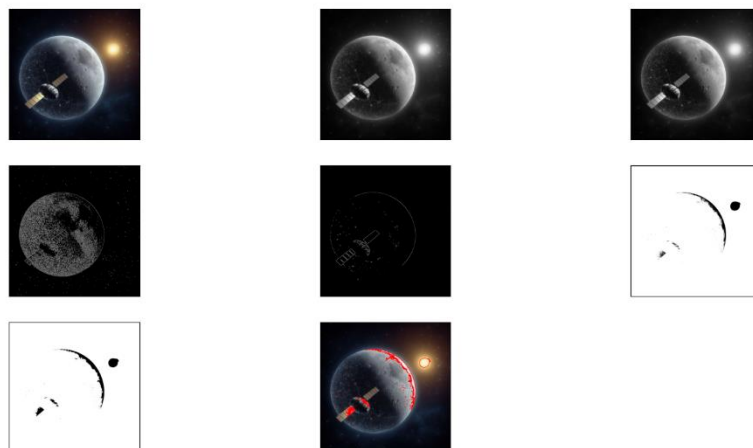
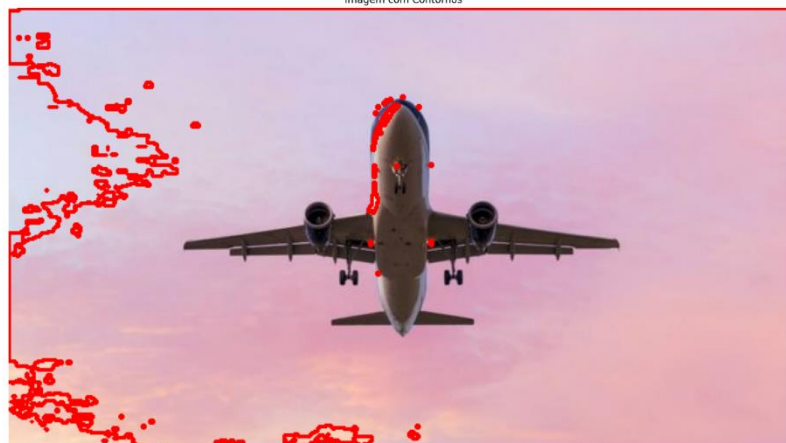


Figure 2

Imagem com Contornos



(x, y) = (514, 544)
(216, 213, 234)

Imagem com Contornos

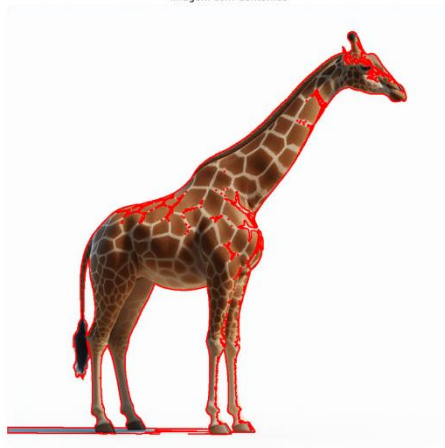
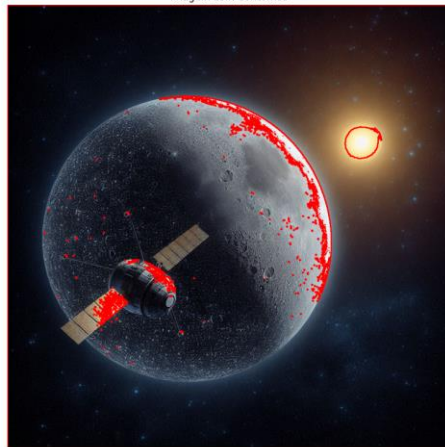


Imagem com Contornos



Explicação Código (Espaço de cores):

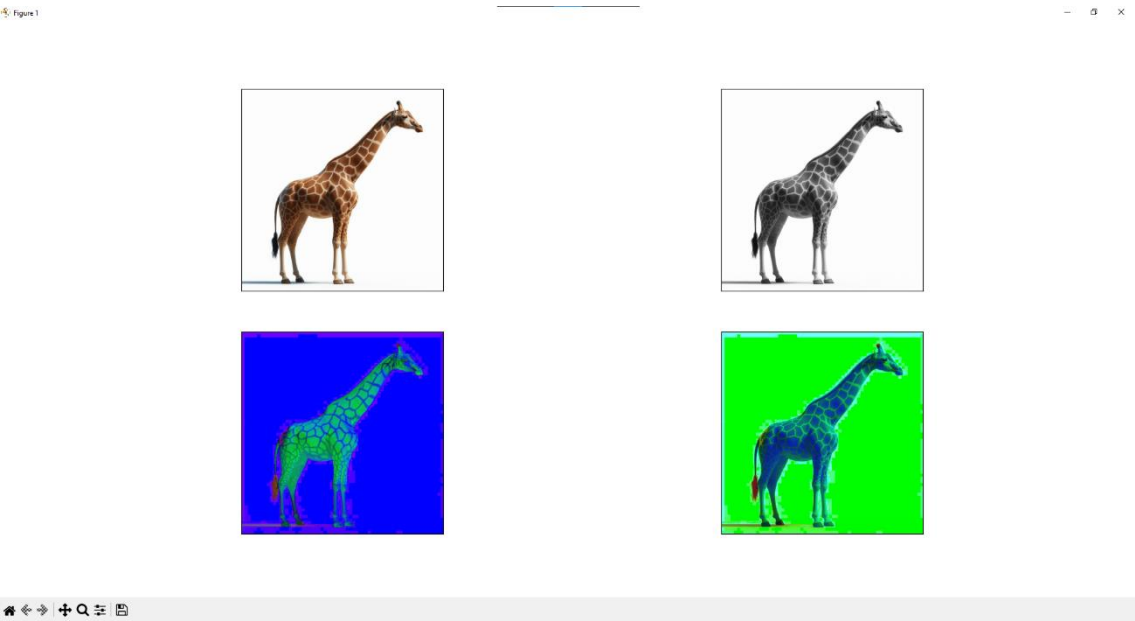
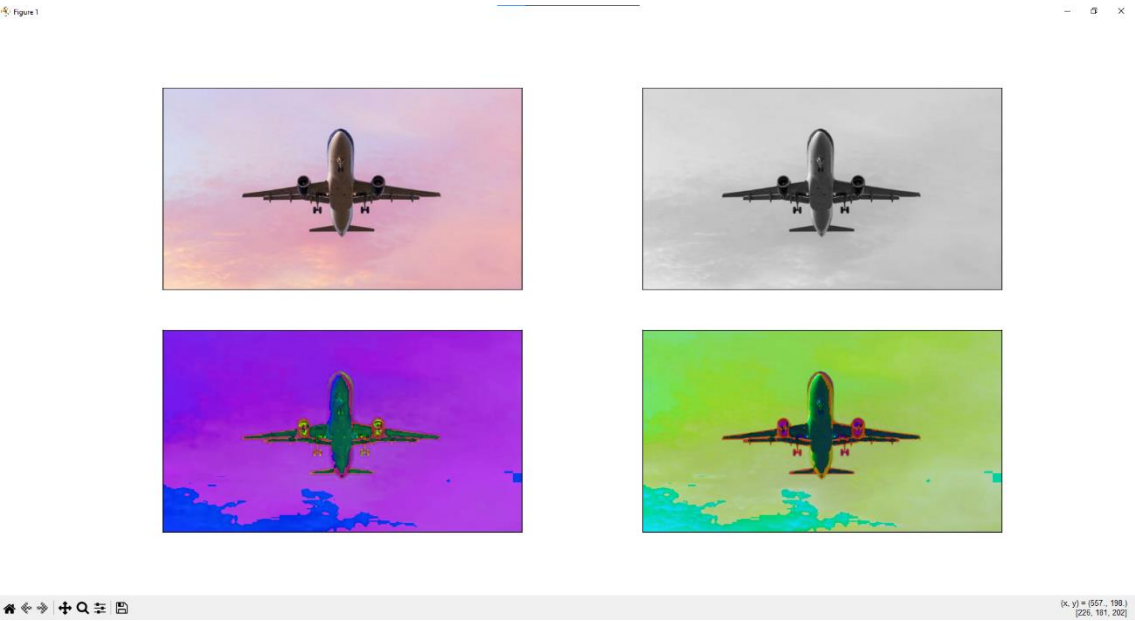
```
espacoCores.py
1  #pip install opencv-python
2
3  import numpy as np
4  import cv2
5  import matplotlib.pyplot as plt
6
7  # Carregans imagem
8  img = cv2.imread('./AVIAO_01.jpg')
9
```

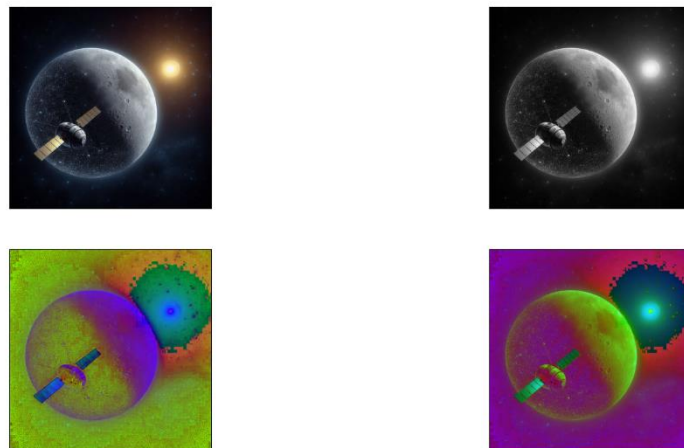
Na primeira parte do código importamos as bibliotecas necessárias para o código funcionar, também carregamos a imagem que iremos aplicar os filtros.

```
10
11  # Convertendo espaço de cores
12  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
13  img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
14  img_hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
15  img_hls = cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
16
17  #plot imagens
18  imagens = [img_rgb, img_gray, img_hsv, img_hls]
19
20  for i in range(4):
21      plt.subplot(2,2,i+1)
22      plt.imshow(imagens[i], 'gray')
23      plt.xticks([], plt.yticks([]))
24  plt.show()
25
26
27
28
29  plt.show()
30
```

Aqui convertemos as imagens para RGB, Cinza e HSV e HLS. Logo depois, plotamos a imagem para visualizarmos o espaço de cores.

Imagens(Espaço de Cores):





Explicação Código (Operador Morfológico):

```

operadorMorfologico.py
1  #pip install opencv-python
2  import numpy as np
3  import math
4  import cv2
5  import matplotlib.pyplot as plt
6
7  #Importa e converte para RGB
8  img = cv2.imread('./AVIAO_01.jpg')
9  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
10
11 #Filtro de ruído (bluring)
12 img_blur = cv2.blur(img,(5,5))

```

No começo do código importamos as bibliotecas necessárias para o código funcionar, importamos também as imagens e convertemos para RGB e por fim, aplicamos o filtro de ruído na imagem.

```

14 #Convertendo para preto e branco (RGB -> Gray Scale -> BW)
15 img_gray = cv2.cvtColor(img_blur, cv2.COLOR_RGB2GRAY)
16 a = img_gray.max()
17 _, thresh = cv2.threshold(img_gray, a/2+100, a,cv2.THRESH_BINARY_INV)
18
19 #preparando o "kernel"
20 kernel = np.ones((12,12), np.uint8)

```

Convertemos a imagem para preto e branco e preparamos o Kernel para tratarmos a imagem.

```

23 #operadores Morfológicos
24 img_dilate = cv2.dilate(thresh, kernel, iterations = 1)
25 img_erode = cv2.erode(thresh, kernel, iterations = 1)
26 img_open = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel)
27 img_close = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
28 img_grad = cv2.morphologyEx(thresh, cv2.MORPH_GRADIENT, kernel)
29 img_tophat = cv2.morphologyEx(thresh, cv2.MORPH_TOPHAT, kernel)
30 img_blackhat = cv2.morphologyEx(thresh, cv2.MORPH_BLACKHAT, kernel)
31
32 # Plot the images
33 imagens = [img, img_blur, img_gray, thresh, img_erode, img_dilate, img_open, img_close, img_grad,
34 |         | img_tophat, img_blackhat]
35
36 formatoX = math.ceil(len(imagens)**.5)
37 if (formatoX**2-len(imagens))>formatoX:
38     formatoY = formatoX-1
39 else:
40     formatoY = formatoX
41
42 for i in range(len(imagens)):
43     plt.subplot(formatoY, formatoX, i + 1)
44     plt.imshow(imagens[i], 'gray')
45     plt.xticks([], plt.yticks([]))
46 plt.show()
47
48

```

Nesta parte, aplicamos os operadores morfológicos e plotamos as imagens.

Imagens (Operador Morfológico):

Figure 1

— a x

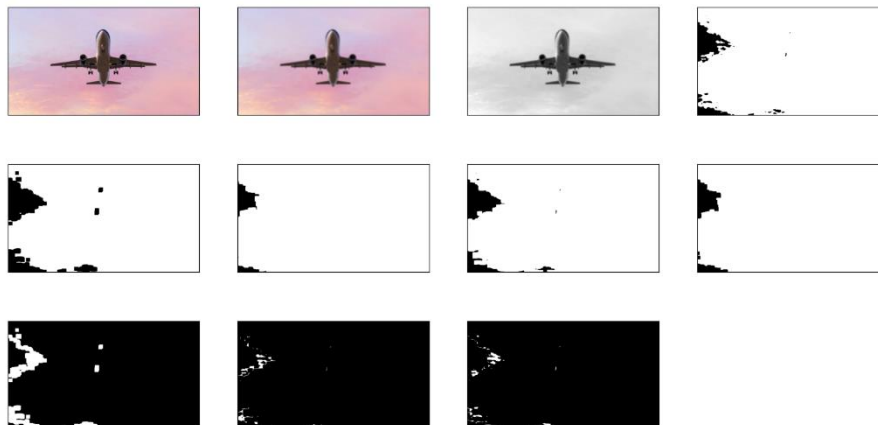


Figure 1

