

▼ Adquirindo a imagem

```

1 #Aquisição das imagens
2 !wget -O "Neymar.jpg" "https://www1.pictures.zimbio.com/gi/Brazil+v+Paraguay+2018+FIFA+World+Cup+Russ
3 !wget -O "Ronaldo.jpg" "https://riotimes-11af1.kxcdn.com/wp-content/uploads/2019/09/Ronaldo-Gau
4 !wget -O "lewandowski.jpg" "https://th.bing.com/th/id/OIP.qADGrCjNyPTg-fi6BLM56QHaJo?pid=Api&w=718&h=

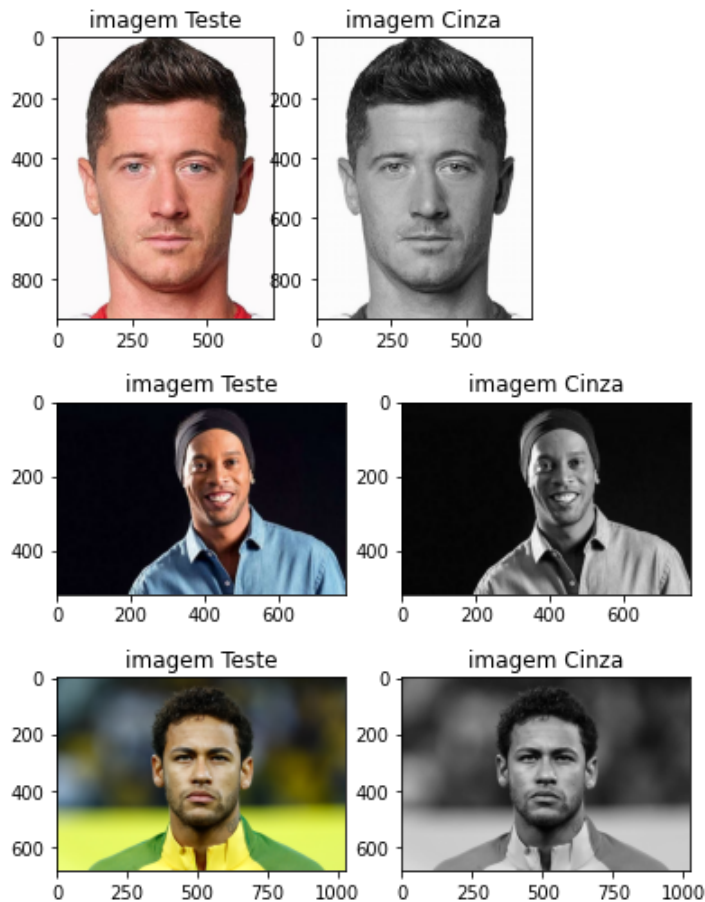
```

▼ Teste de exibição da imagem

```

1 #Bibliotecas Importadas
2
3 import cv2 as cv
4 import numpy as np
5 from matplotlib import pyplot as plt
6
7 #Leitura da imagem e criação de uma cópia em tons de cinza
8 #Para nos ajudar durante a detecção
9
10 img1 = cv.imread('lewandowski.jpg')
11 img2 = cv.imread('Ronaldo.jpg')
12 img3 = cv.imread('Neymar.jpg')
13
14
15 img1 = cv.cvtColor(img1,cv.COLOR_BGR2RGB)
16 gray = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
17
18 img2 = cv.cvtColor(img2,cv.COLOR_BGR2RGB)
19 gray2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
20
21 img3 = cv.cvtColor(img3,cv.COLOR_BGR2RGB)
22 gray3 = cv.cvtColor(img3, cv.COLOR_BGR2GRAY)
23
24
25 #Exibição da imagem
26 plt.figure(figsize=(15,12))
27 plt.subplot(261), plt.imshow(img1), plt.title('imagem Teste')
28 plt.subplot(262), plt.imshow(gray, cmap = 'gray'), plt.title('imagem Cinza')
29
30 plt.figure(figsize=(20,10))
31 plt.subplot(261), plt.imshow(img2), plt.title('imagem Teste')
32 plt.subplot(262), plt.imshow(gray2, cmap = 'gray'), plt.title('imagem Cinza')
33
34 plt.figure(figsize=(20,10))
35 plt.subplot(261), plt.imshow(img3), plt.title('imagem Teste')
36 plt.subplot(262), plt.imshow(gray3, cmap = 'gray'), plt.title('imagem Cinza')
37
38
39 plt.show()
40

```



▼ Procedimento para detecção de faces

```

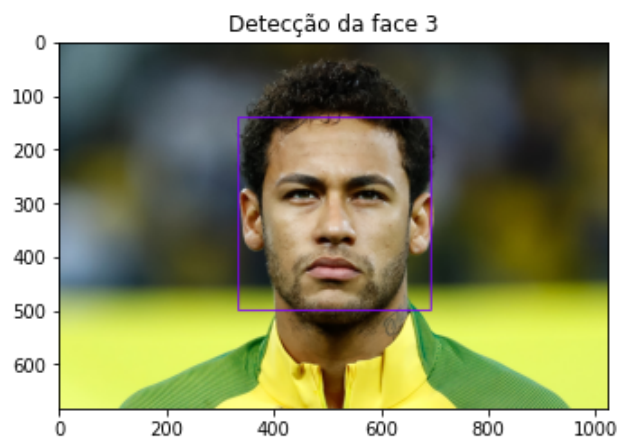
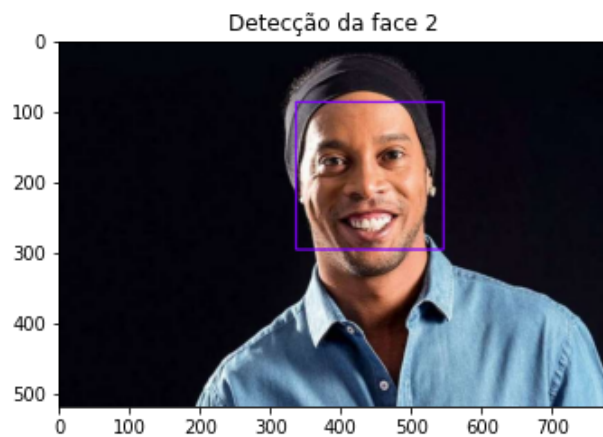
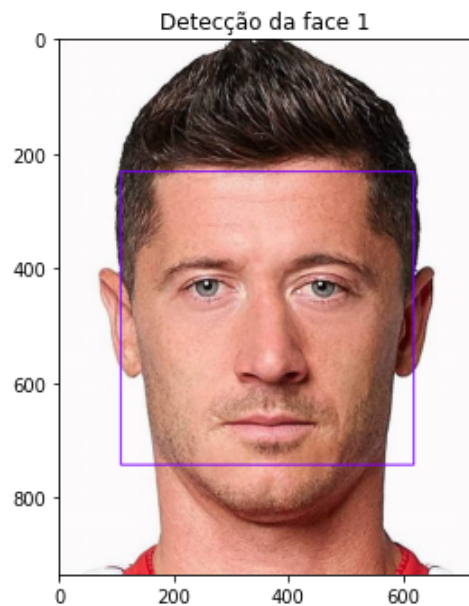
1 #Definindo onde salvar o XML da detecção
2 face_classifier = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_frontalface_default.xml')
3
4 #Retorna a face detectada(Region of interest) como uma tupla
5 def detectandoFace(cinza):
6     faces = face_classifier.detectMultiScale(cinza, 1.3, 5)
7     return faces
8
9 # #Se não detectar faces,retorna que não encontrou
10 # if faces is ():
11 #     print("Nenhuma face encontrada")
12
13 #Impressão da imagem 1 com o overlay de detecção
14 for (x,y,w,h) in detectandoFace(gray):
15     cv.rectangle(img1, (x,y), (x+w,y+h), (127,0,255), 2)
16     plt.figure(figsize=(12,12))
17     plt.imshow(cv.cvtColor(img1, cv.COLOR_BGR2RGB))
18     plt.subplot(221), plt.imshow(img1), plt.title('Detecção da face 1')
19     plt.show()
20
21 for (x,y,w,h) in detectandoFace(gray2):
22     cv.rectangle(img2, (x,y), (x+w,y+h), (127,0,255), 2)
23     plt.figure(figsize=(12,12))
24     plt.imshow(cv.cvtColor(img2, cv.COLOR_BGR2RGB))
25     plt.subplot(221), plt.imshow(img2), plt.title('Detecção da face 2')

```

```

26 plt.show()
27
28 for (x,y,w,h) in detectandoFace(gray3):
29     cv.rectangle(img3, (x,y), (x+w,y+h), (127,0,255), 2)
30     plt.figure(figsize=(12,12))
31     plt.imshow(cv.cvtColor(img3, cv.COLOR_BGR2RGB))
32     plt.subplot(221), plt.imshow(img3), plt.title('Detecção da face 3')
33     plt.show()
34

```



▼ Procedimento adicionando a detecção dos olhos a de face

```
1 # Definindo onde salvar o XML da detecção
2 face_classifier = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_frontalface_default.xml')
3 eye_classifier = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_eye.xml')
4
5 #Retorna a face detectada(Region of interest) como uma tupla
6 # faces = face_classifier.detectMultiScale(gray, 1.3, 5)
7 def detectandoFace(cinza):
8     faces = face_classifier.detectMultiScale(cinza, 1.3, 5)
9     return faces
10
11 #Se não detectar faces,retorna que não encontrou
12 # if faces is ():
13 #     print("Nenhuma face encontrada.")
14
15
16 #Impressão da imagem com o overlay de detecção
17 for (x,y,w,h) in detectandoFace(gray):
18     cv.rectangle(img1,(x,y),(x+w,y+h),(127,0,255),2)
19
20     #Detecção dos olhos(Region of Interest)
21     roi_gray = gray[y:y+h, x:x+w]
22     roi_color = img1[y:y+h, x:x+w]
23     eyes = eye_classifier.detectMultiScale(roi_gray)
24
25     for (ex,ey,ew,eh) in eyes :
26         cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,255,0),2)
27
28     #Exibição da imagem com os devidos overlays
29     plt.figure(figsize=(12,12))
30     plt.imshow(cv.cvtColor(img1, cv.COLOR_BGR2RGB))
31     plt.subplot(221), plt.imshow(img1), plt.title('Detecção dos olhos')
32     plt.show()
33
```

Detecção dos olhos

▼ Procedimento de exibição de imagem

200 |  |

```

1 def exhibe_imagem(img_x, gray_x):
2     #Definindo onde salvar o XML da detecção
3     face_classifier = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_frontalface_default.xml')
4     eye_classifier = cv.CascadeClassifier(cv.data.harcascades + 'haarcascade_eye.xml')
5
6     #Retorna a face detectada(Region of interest) como uma tupla
7     faces = face_classifier.detectMultiScale(gray_x, 1.3, 5)
8
9
10    #Se não detectar faces,retorna que não encontrou
11    if faces is ():
12        print("Nenhuma face encontrada.")
13
14
15    #Impressão da imagem com o overlay de detecção
16    for (x,y,w,h) in faces:
17        cv.rectangle(img_x,(x,y),(x+w,y+h),(127,0,255),2)
18
19    #Detecção dos olhos(Region of Interest)
20    roi_gray = gray_x[y:y+h, x:x+w]
21    roi_color = img_x[y:y+h, x:x+w]
22    eyes = eye_classifier.detectMultiScale(roi_gray)
23
24    for (ex,ey,ew,eh) in eyes :
25        cv.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(255,255,0),2)
26
27
28    #Exibição da imagem com os devidos overlays
29    plt.figure(figsize=(12,12))
30    plt.imshow(cv.cvtColor(img_x, cv.COLOR_BGR2RGB))
31    plt.subplot(221), plt.imshow(img_x), plt.title('Detecção dos olhos')
32    plt.show()
33
34 exhibe_imagem(img1, gray)
35 exhibe_imagem(img2, gray2)
36 exhibe_imagem(img3, gray3)

```

