Detecção de Faces Usando Classificadores Haar (Parte 1)

ES235 - Aula 17
João Marcelo Teixeira
Willams Costa

Introdução

- Método bastante eficiente proposto por Paul Viola e Michael Jones no trabalho “Rapid Object Detection using a Boosted Cascade of Simple Features” em 2001
- Utiliza aprendizagem de máquina
- Função cascata é treinada a partir de várias imagens “positivas” e “negativas”

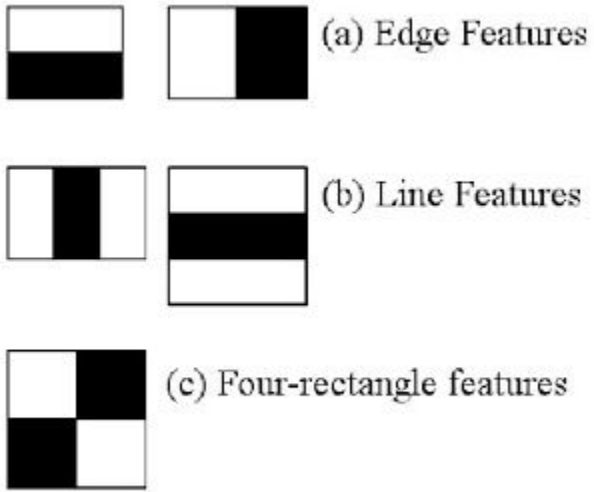
Haar Features

- Imagens positivas
  - Contêm faces

- Imagens negativas
  - Não contêm faces

Haar Features

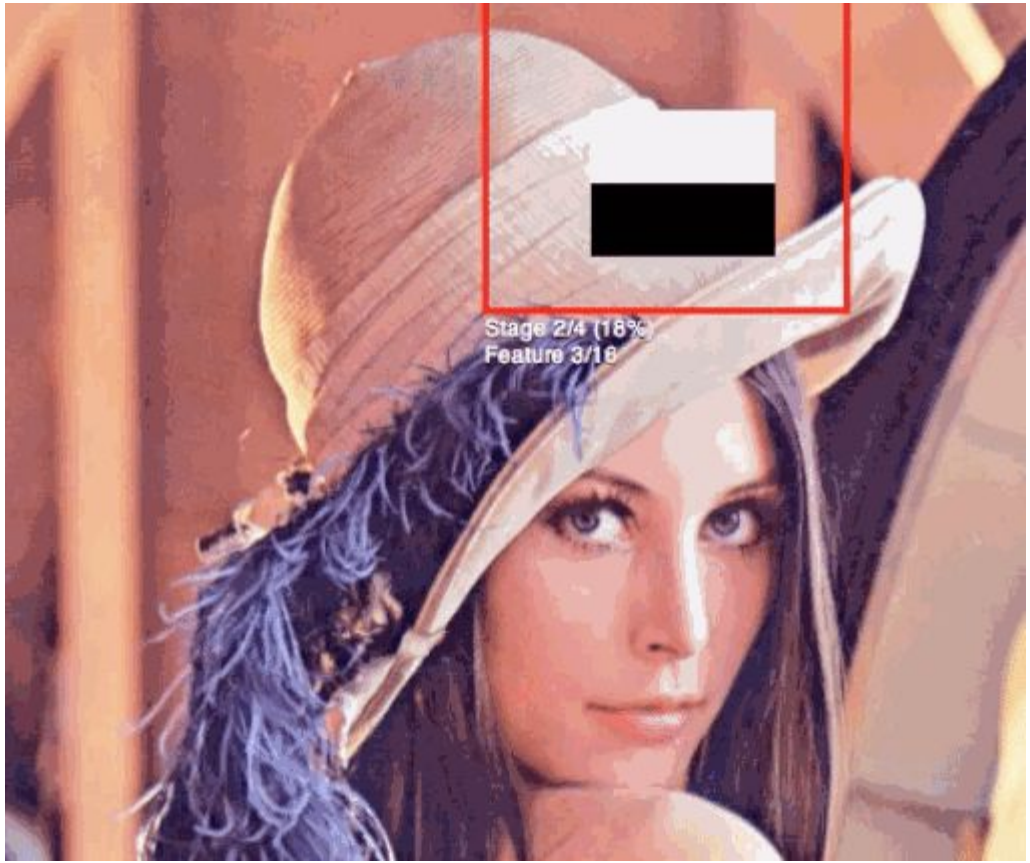
- Cada feature extraída apresenta um valor calculado a partir da soma dos valores dos pixels abaixo da região branca menos a soma dos valores dos pixels abaixo da região preta



Haar Features

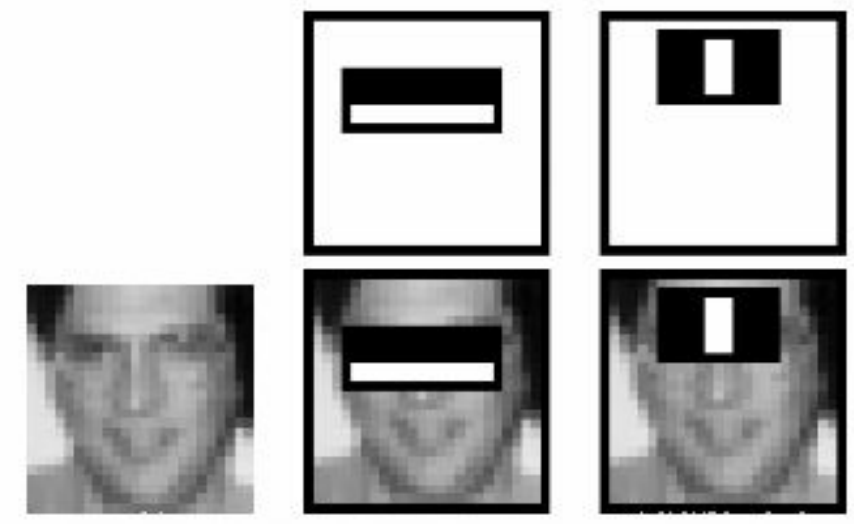
- Problema: milhares de features por imagem!

- Alguém lembra das “imagens integrais”?



Cascata de Classificadores

- Adaboost é utilizado para reduzir a quantidade de features necessárias (consegue reduzir de 160k para 6k features, por exemplo)



Cascata de Classificadores

- Na maioria das imagens as regiões não possuem faces
- A ideia então é descartar regiões sem faces com testes simples e aumentar a quantidade de testes em regiões com maior probabilidade de possuírem faces
- Cinco primeiros estágios (trabalho original) com 1, 10, 25, 25 e 50 features

Usando o OpenCV

-

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Usando o OpenCV

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv2.imshow('img',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Usando o OpenCV

Bases já treinadas disponíveis:

- haarcascade_eye.xml
- haarcascade_eye_tree_eyeglasses.xml
- haarcascade_frontalcatface.xml
- haarcascade_frontalcatface_extende...
- haarcascade_frontalface_alt.xml
- haarcascade_frontalface_alt2.xml
- haarcascade_frontalface_alt_tree.xml
- haarcascade_frontalface_default.xml
- haarcascade_fullbody.xml
- haarcascade_lefteye_2splits.xml
- haarcascade_licence_plate_rus_16st...
- haarcascade_lowerbody.xml
- haarcascade_profileface.xml
- haarcascade_righteye_2splits.xml
- haarcascade_russian_plate_number....
- haarcascade_smile.xml
- haarcascade_upperbody.xml

Treinamento (criando a própria base)

Passo 1: Separar amostras negativas

Directory structure:

```
/img
  img1.jpg
  img2.jpg
bg.txt
```

File bg.txt:

```
img/img1.jpg
img/img2.jpg
```

Treinamento (criando a própria base)

Passo 2: Separar amostras positivas

- opencv_createsamples
- opencv_annotation

Directory structure:

```
/img
  img1.jpg
  img2.jpg
info.dat
```

File info.dat:

```
img/img1.jpg  1  140 100 45 45
img/img2.jpg  2  100 200 50 50   50 30 25 25
```

Treinamento (criando a própria base)

Passo 3: Executar o treinamento

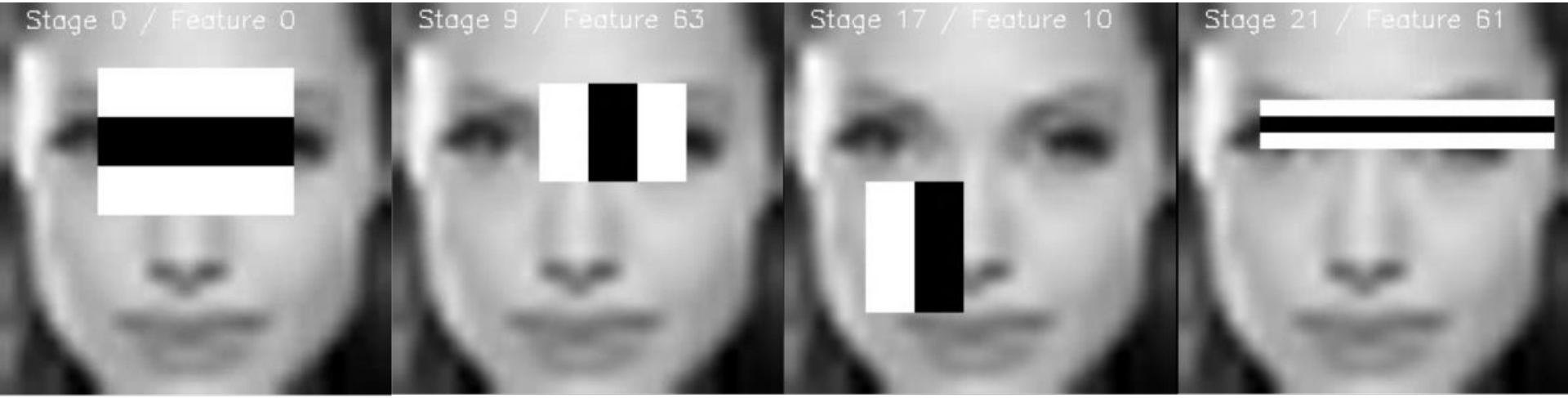
- opencv_traincascade

  (várias opções de argumentos, verificar documentação e exemplos!)

Treinamento (criando a própria base)

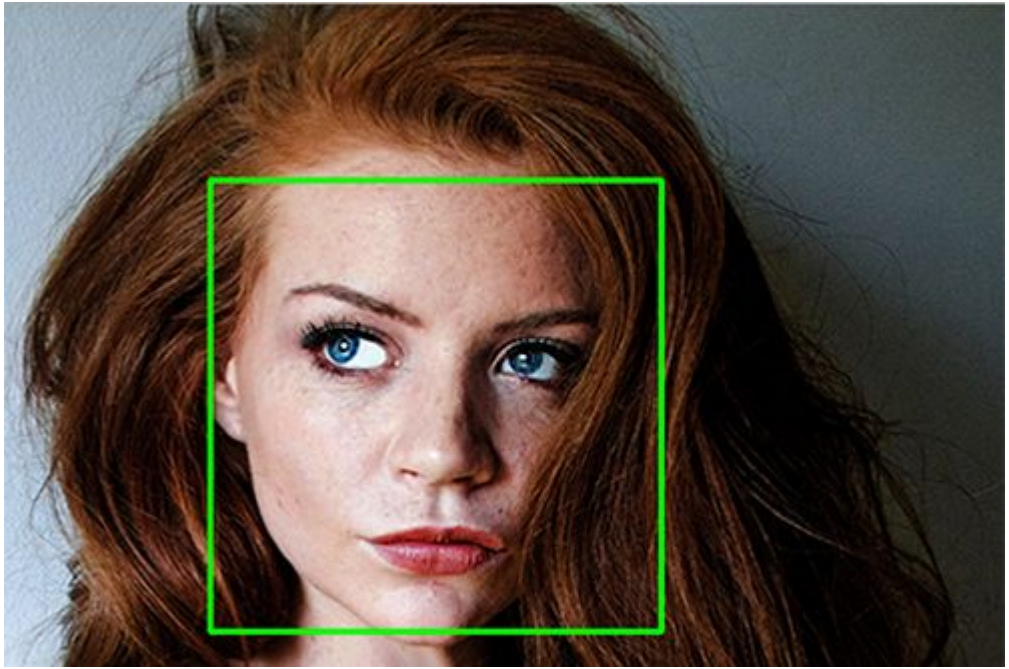
Passo 4: Visualizar classificadores (opcional)

- opencv_visualisation



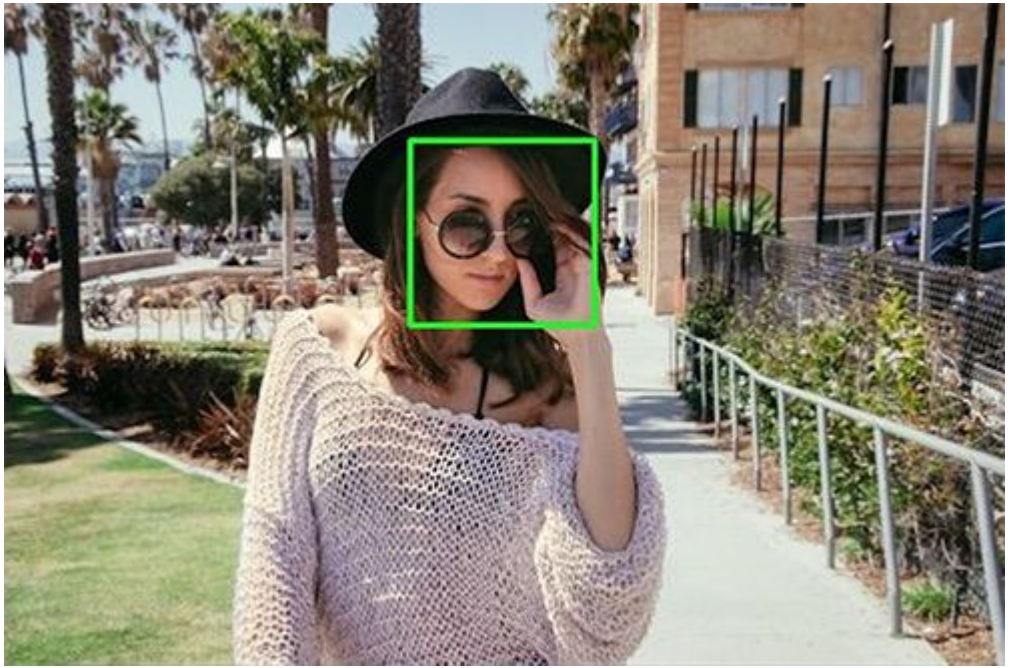
Exemplos

- scaleFactor=1.4,
- minNeighbors=1,
- minSize=(30, 30)



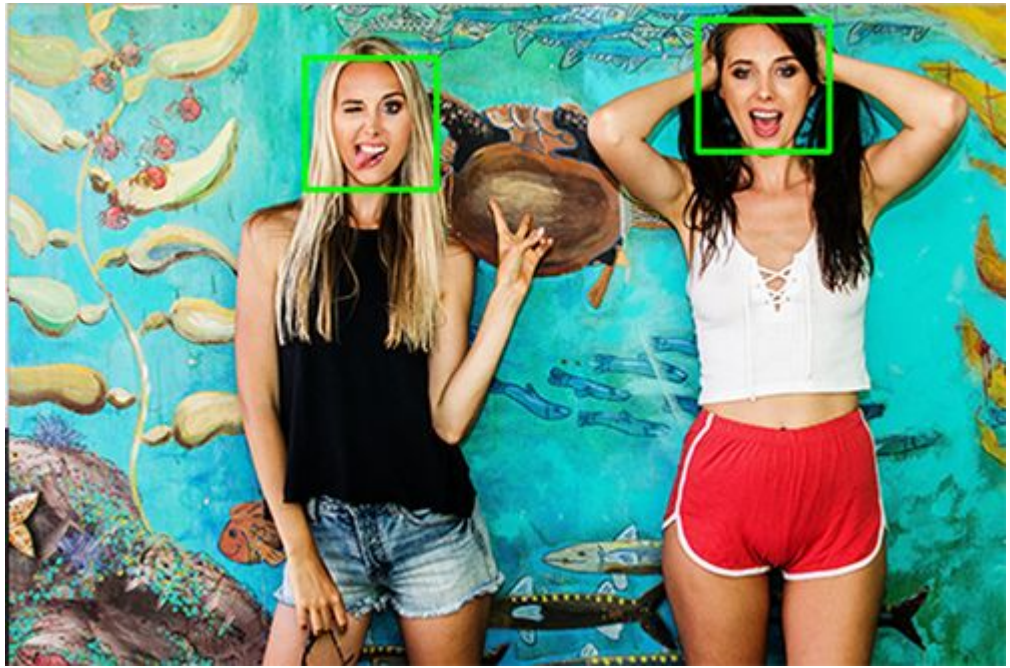
Exemplos

- scaleFactor=1.4,
- minNeighbors=1,
- minSize=(30, 30)



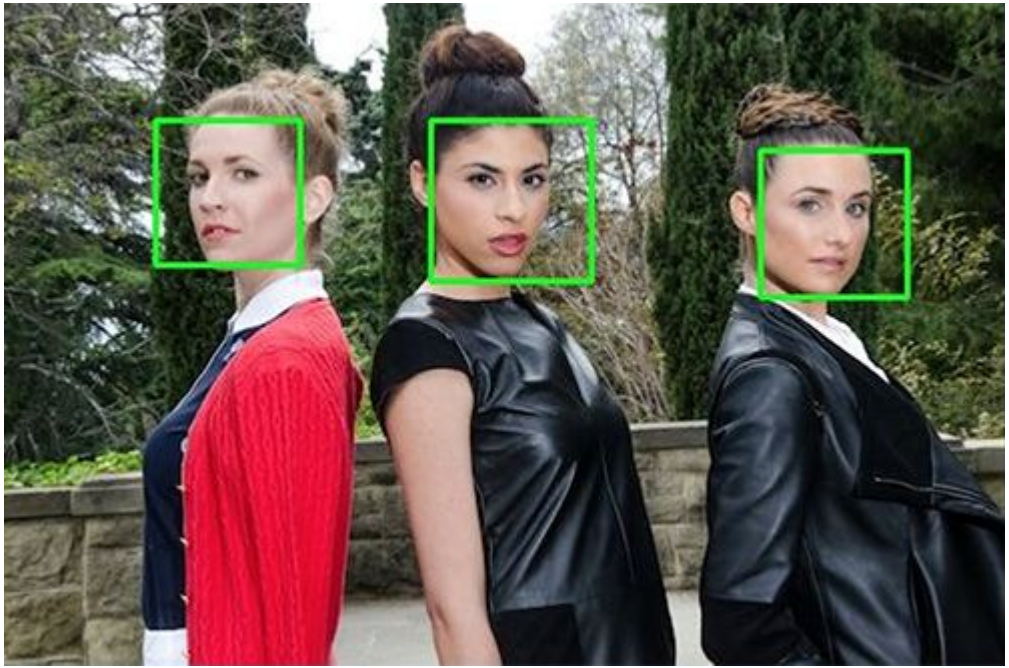
Exemplos

- scaleFactor=1.4,
- minNeighbors=1,
- minSize=(10,10)



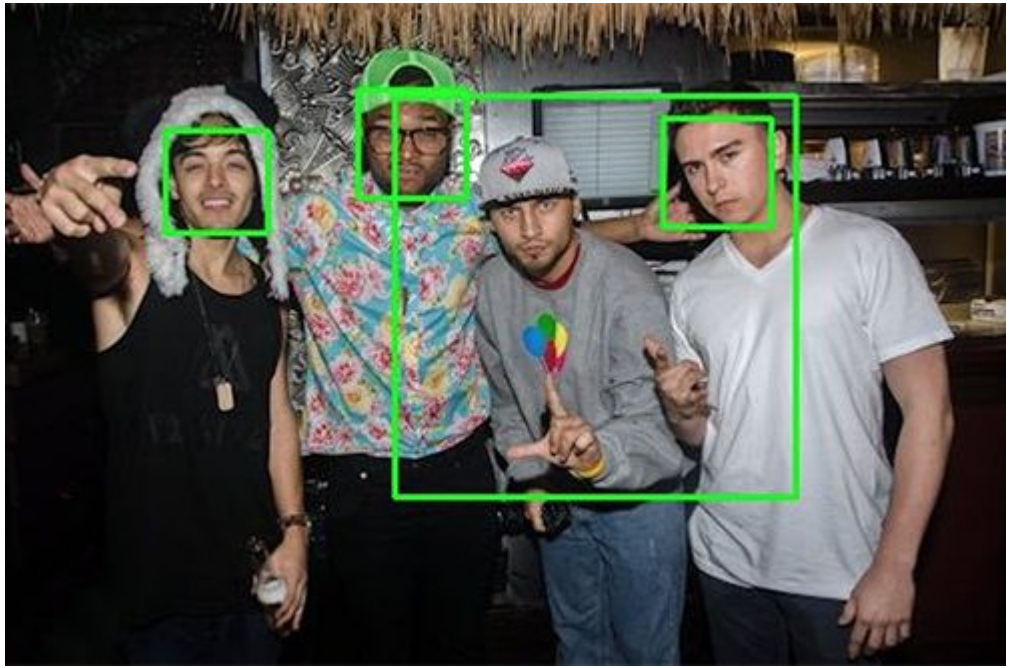
Exemplos

- scaleFactor=1.5,
- minNeighbors=2,
- minSize=(30, 30)



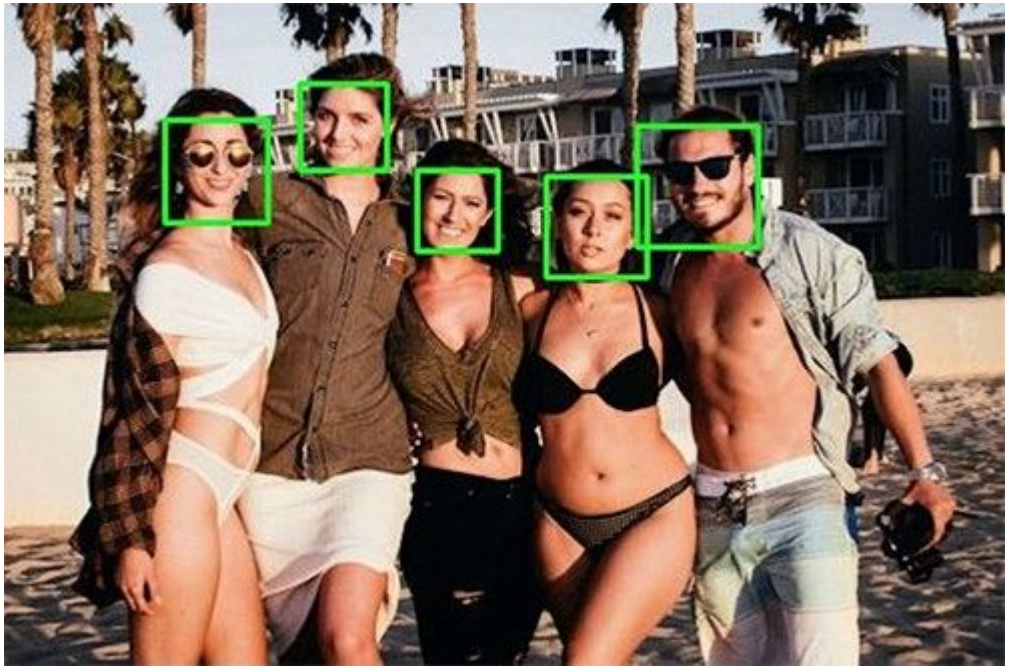
Exemplos

- scaleFactor=1.3,
- minNeighbors=2,
- minSize=(30, 30)



Exemplos

- scaleFactor=1.6,
- minNeighbors=2,
- minSize=(20,20)

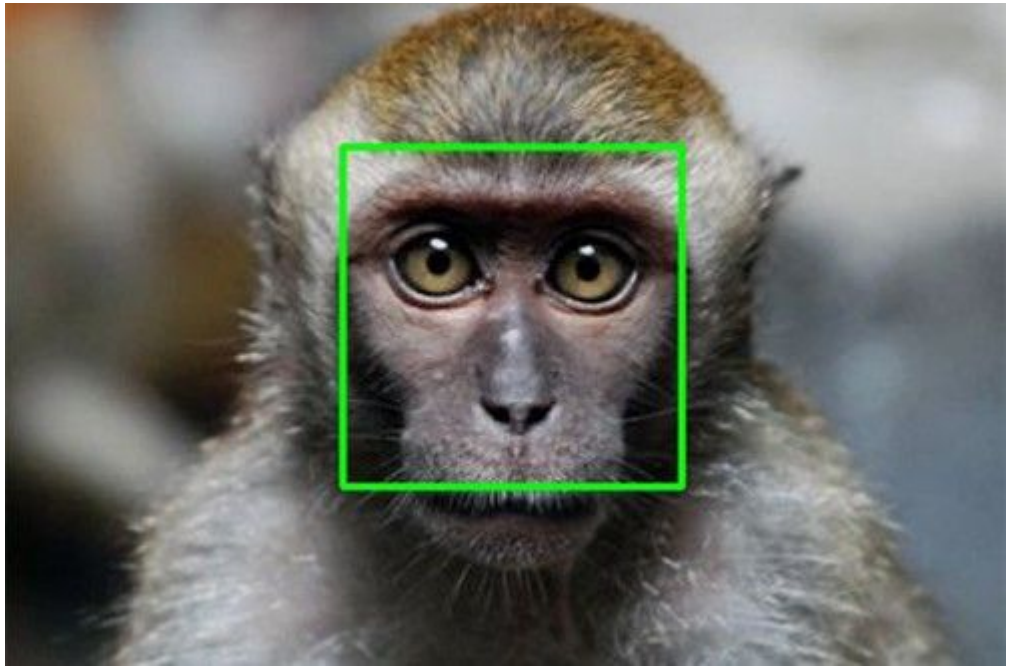


Exemplos

- scaleFactor=1.1,
- minNeighbors=5,
- minSize=(1, 1)

Exemplos

- scaleFactor=1.1,
- minNeighbors=1,
- minSize=(30, 30)



REFERÊNCIAS

Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html

https://www.datacamp.com/community/tutorials/face-detection-python-opencv

https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177

https://www.superdatascience.com/blogs/opencv-face-detection

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf