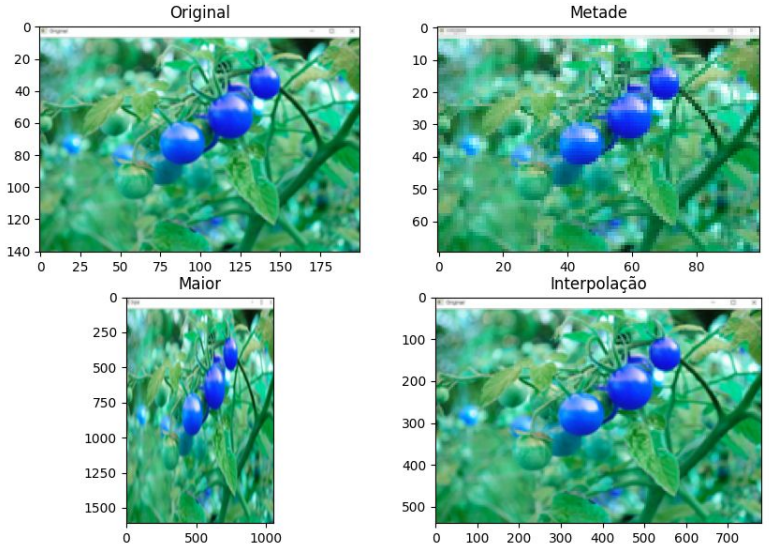
Introdução à Visão Computacional com OpenCV e Python

Processamento de Imagem e vídeo

Ruy de Oliveira

Mudando o tamanho de imagens

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

path = r'.\imagens\frutas.png'
img = cv2.imread(path)

metade = cv2.resize(img, (0, 0), fx = 0.5, fy = 0.5)
maior = cv2.resize(img, (1050, 1610))
interpolada= cv2.resize(img, (780, 540),
            interpolation = cv2.INTER_NEAREST)

Titles =["Original", "Metade", "Maior",
"Interpolação"]
images =[img, metade, maior, interpolada]
count = 4
for i in range(count):
    plt.subplot(2, 2, i + 1)
    plt.title(Titles[i])
    plt.imshow(images[i])

plt.show()
```

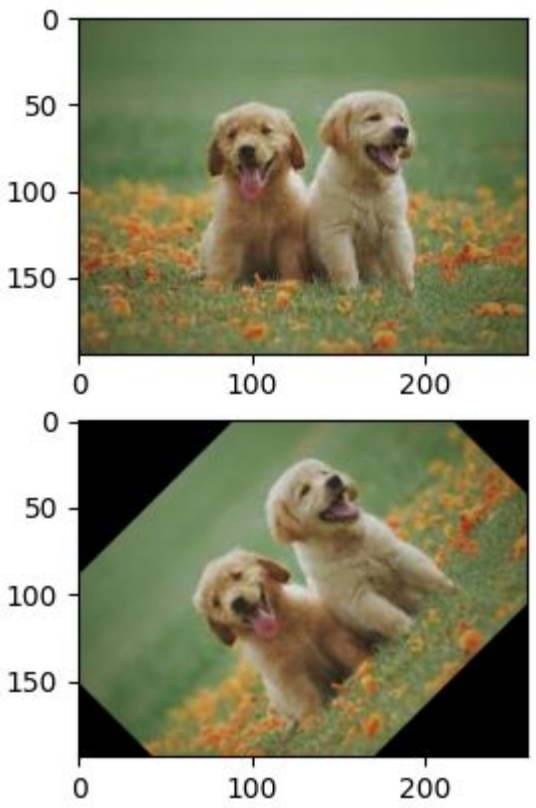


Técnicas de interpolação para mudança de tamanho

cv2.INTER_AREA: para diminuir tamanho da imagem
cv2.INTER_CUBIC: Lento, porém mais eficiente
cv2.INTER_LINEAR: para fazer zoom. Esta é a técnica de interpolação padrão no OpenCV

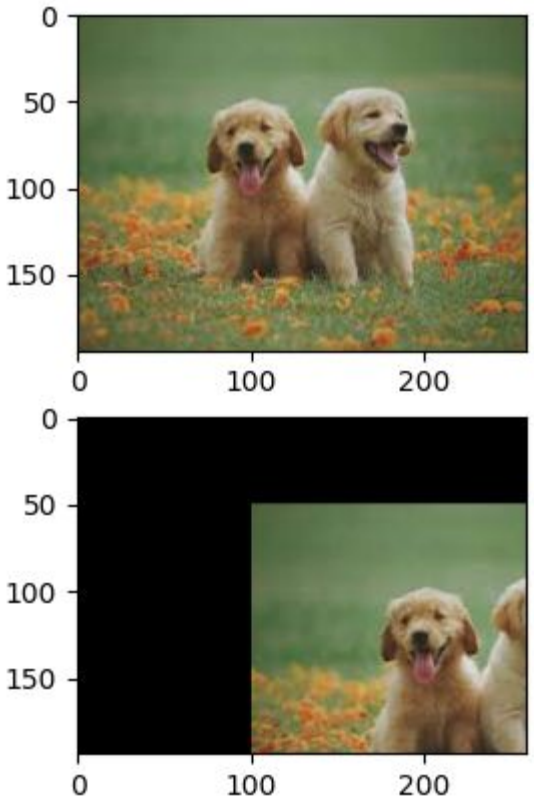
Ex9 - Rotacionando uma imagem

```python
# Rotaciona uma imagem

# importa bibliotecas
import cv2
import numpy as np
import matplotlib.pyplot as plt

path = r'.\imagens\dogs.jpg'

try:
    # leitura da imagem
    img = cv2.imread(path)

    # Shape of image in terms of pixels.
    (linhas, colunas) = img.shape[:2]

    # getRotationMatrix2D cria a matriz necessária à transformação
    # parametros para centralzar rotacionar 45 graus sem mudanca de tamanho (fator de escala igual a 1)
    M = cv2.getRotationMatrix2D((colunas / 2, linhas / 2), 45, 1)
    #res = cv2.warpAffine(img, M, (colunas, linhas))
    print(linhas, colunas, linhas/2,colunas/2)
    res = cv2.warpAffine(img, M, (colunas, linhas))
    #converte para RGB e plota numa unica imagem
    RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    RGB_img_rot = cv2.cvtColor(res, cv2.COLOR_BGR2RGB)
    plt.subplot(2, 1, 1)
    plt.imshow(RGB_img)
    plt.subplot(2, 1, 2)
    plt.imshow(RGB_img_rot)

    # plota as duas imagens
    plt.show()

    # salva a imagem resultante no disco
    cv2.imwrite('.\output_images\img_rotacionada.jpg', res)
except IOError:
    print ('Erro na leitura do arquivo !!!')
```



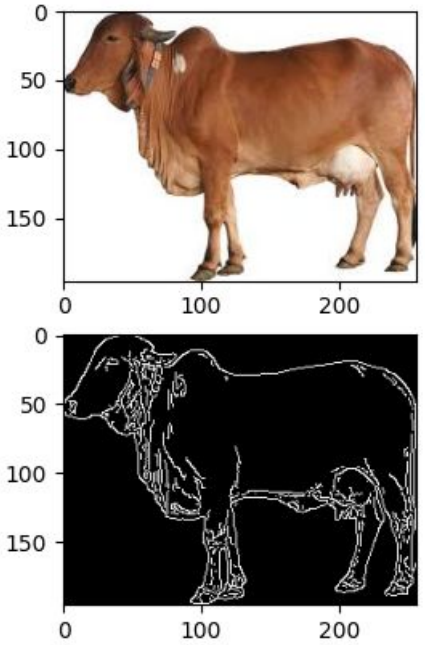
Ex10-Deslocando uma imagem

```python
# Desloca uma imagem

# importa bibliotecas
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Cria a matriz de deslocamento (translacao).
# Se o desolccamento e de (x, y), entao a matriz seria
# M = [1 0 x]
#     [0 1 y]
# Vamos deslocar aqui de (100, 50).
M = np.float32([[1, 0, 100], [0, 1, 50]])

try:
    # leitura da imagem
    img = cv2.imread('.\imagens\dogs.jpg')

    # dimensoes da imagem em termos de pixels.
    (linhas, colunas) = img.shape[:2]

    # a o metodo warpAffine realiza o desolocamento com base na matriz de deslocamento
    res = cv2.warpAffine(img, M, (colunas, linhas))

    #converte para RGB e plota numa unica imagem
    RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    RGB_img_rot = cv2.cvtColor(res, cv2.COLOR_BGR2RGB)
    plt.subplot(2, 1, 1)
    plt.imshow(RGB_img)
    plt.subplot(2, 1, 2)
    plt.imshow(RGB_img_rot)

    # plota as duas imagens
    plt.show()
    # salva a imagem resultante no disco
    cv2.imwrite('.\output_images\img_deslocada.jpg', res)
except IOError:
    print ('Erro na leitura do arquivo !!!')
```



Ex11-Detecção de bordas

```python
# Detecta bordas numa imagem

# importa bibliotecas
import cv2
import numpy as np
import matplotlib.pyplot as plt

try:
    # leitura da imagem
    img = cv2.imread('.\imagens\cow1.jpg')

    # Canny edge detection.
    #t_lower = 50  # limite inferior de sensibilidade
    #t_upper = 150  # limite superior de sensibilidade
    img_bordas = cv2.Canny(img, 100, 200)

    #converte para RGB e plota numa unica imagem
    RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    RGB_img_rot = cv2.cvtColor(img_bordas, cv2.COLOR_BGR2RGB)
    plt.subplot(2, 1, 1)
    plt.imshow(RGB_img)
    plt.subplot(2, 1, 2)
    plt.imshow(RGB_img_rot)

    # plota as duas imagens
    plt.show()
    # salva a imagem resultante no disco
    cv2.imwrite('.\output_images\img_bordas.jpg', res)
except IOError:
    print ('Erro na leitura do arquivo !!!')
```



Características importantes

- **Detecção de bordas acentuadas**
- **Essencial em reconhecimento de imagem e detecção de objetos**
- **Há vários algoritmos, aqui está sendo usado o algoritmo Canny Edge Detection**

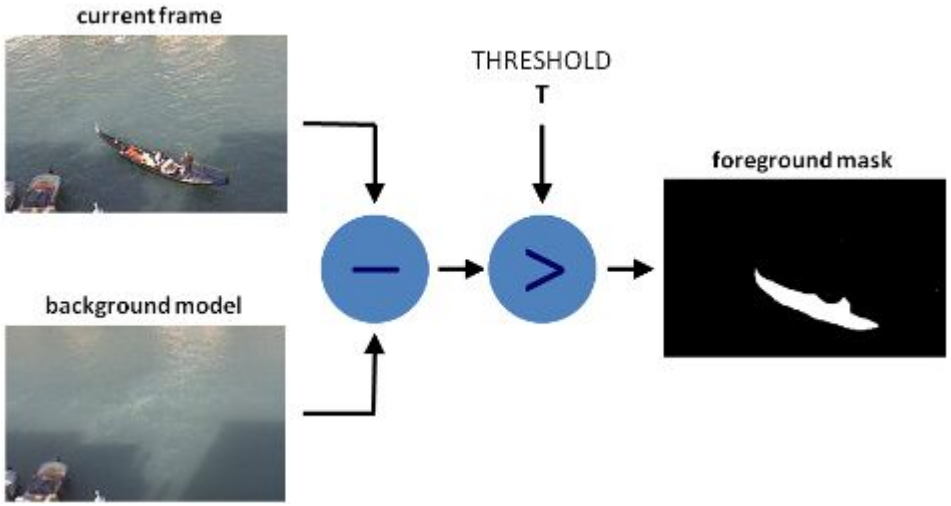
Ex12- Subtração de plano de fundo (background subtraction)

Detecta movimento na imagem

Usado na segurança, rastreamento de pedestres, contagem de visitantes, número de veículos no tráfego, etc

O plano de fundo (background) estático é eliminado

Imagem binária (máscara de primeiro plano) provê informação sobre objetos em movimento na imagem



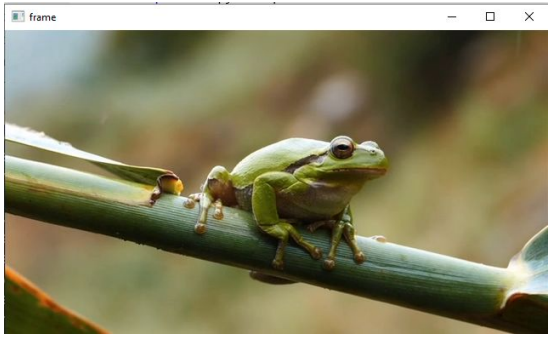
Algoritmos de subração de plano de fundo

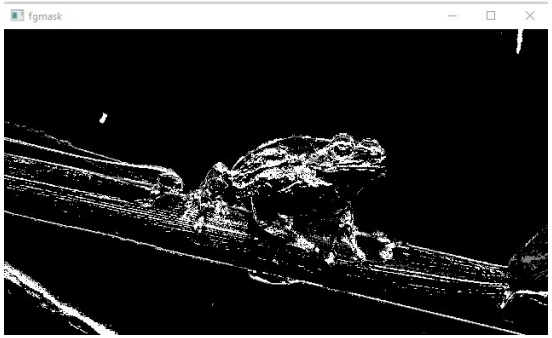
BackgroundSubtractorMOG: básico
BackgroundSubtractorMOG2: mais estável sob variação de luminosidade e sombra
Geometric multigrid: usa métodos estatísticos e algoritmo de segmentação baysiano por pixel

Ex12- Subtração de plano de fundo (background subtraction)

Detecta movimento na imagem

```python
# importa bibliotecas
import cv2
import numpy as np
import matplotlib.pyplot as plt

#le video
cap = cv2.VideoCapture('.\\videos\\frog.mp4')
#gera o modelo de subtração
fgbg = cv2.createBackgroundSubtractorMOG2()

while(1):
    # ret recebe true se a leitura estiver correta, ou false caso contrario
    # frame recebe um quadro do video (matriz), baseado na taxa de quadros por segundos sendo usada
    ret, frame = cap.read()

    #aplica o modelo de subtração na sequencia de quadros do video
    #fmask é o resultado denominado "mascara de primeiro plano"
    fgmask = fgbg.apply(frame)

    # plota os dois quadro sucessivos (duas imagens)
    cv2.imshow('fgmask', fgmask)
    cv2.imshow('frame',frame )

    # espera 30 milisegundos entre cada quadro
    # a operacao lógica E com o valor 0xFF visa garantir que apenas
    # os 8 bits menos significativos do codigo ascii da tecla lida
    # por waikey, seja considerado
    # o programa termina quando o video chega ao fim ou
    # a tecla "esc" e pressionada
    k = cv2.waitKey(30) & 0xff
    if k == 27:  #esc tem ascii código 27
        break
# libera o espaco em memoria usado pelo video e pela janela de impressao na tela
cap.release()
cv2.destroyAllWindows()
```

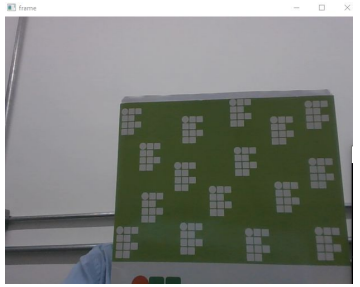


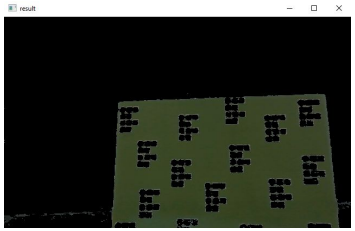


Máscara de primeiro plano

Ex13- Filtro de cor

```python
# importa bibliotecas
import cv2
import numpy as np

#captura video da Webcam
cap = cv2.VideoCapture(0)

while(1):
    # ret recebe true se a leitura estiver correta, ou false caso contrario
    # frame recebe um quadro do video (matriz), baseado na taxa de quadros por segundos sendo usada
    _ , frame = cap.read()

    # converte padrao de cor BGR para HSV - e mais adequado para essas aplicacoes
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # limites da cor verde no padrao HSV - hsl(hue, saturation, lightness)
    # intensidade (0 a 360 graus), saturacao (0 a 100%), brilho (0 a 100%)
    verde_inf = np.array([25, 50, 70])
    verde_sup = np.array([100, 255, 255])

    # preparo da mascara para sobreposicao
    masc = cv2.inRange(hsv, verde_inf, verde_sup)

    # a regiao preta na mascara tem valor 0, de modo que quando multiplicada
    #  logicamente (via E logico) com a imagem original remove as regioes nao verdes
    result = cv2.bitwise_and(frame, frame, mask = masc)

   # plota: imagem original, mascara e imagem resultante
    cv2.imshow('frame', frame)
    cv2.imshow('mask', masc)
    cv2.imshow('result', result)

    k = cv2.waitKey(1) & 0xff
    if k == 27:  #esc tem ascii código 27
        break
# libera o espaco em memoria usado pelo video e pela janela de impressao na tela
cap.release()
cv2.destroyAllWindows()
```

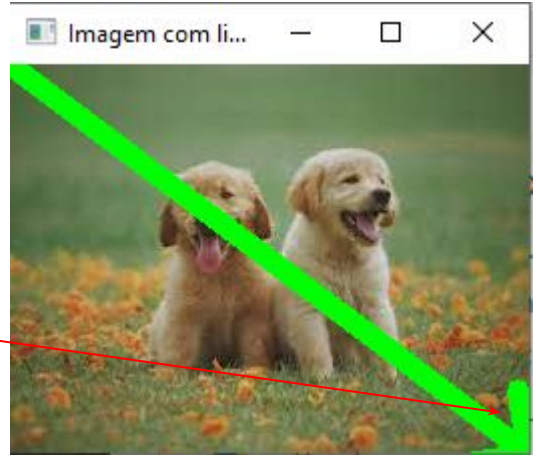




Os limiares inferior e superior definem a qualidade da filtragem

Ex14- Criando linha em imagem

```python
# Cria linha em imagem

# importa bibliotecas
import cv2
import numpy as np

#Le figura
img = cv2.imread('.\imagens\dogs.jpg')
print(img.shape)

# coordenada do ponto inicial, aqui (0, 0), canto superior esq. da imagem
ponto_inicial = (0, 0)

#coordenada do ponto final, aqui (258, 194), canto inferior dir. da imagem
ponto_final = (258, 194)

# cor verde em BGR
cor = (0, 255, 0)

# espessura da linha de 9 px
espessura = 9

# Usando o metodo cv2.line() para criar um linha diagonal com espessura de 9 px
cv2.line(img, ponto_inicial, ponto_final, cor, espessura)

#cv2.arrowedLine(img, ponto_inicial, ponto_final, cor, espessura)

# plota a imagem na tela
cv2.imshow('Imagem com linha', img)

#tecla "esc" tem ascii código 27
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```



Ex15- Criando círculo em imagem

```python
# Cria circulo em imagem

# importa bibliotecas
import cv2
import numpy as np

#Le figura
img = cv2.imread('.\imagens\dogs.jpg')
print(img.shape)

# coordenada do centro do círculo
coordinadas_centro = (130, 150)

# raio do círculo
raio = 20

# cor vermelha em BGR
cor = (0, 0, 255)

# espessura da linha de 2 px
espessura = 5

# o metodo cv2.circle() cria um círculo com linha borda azul de espessura 5 px
image2 = cv2.circle(img, coordinadas_centro, raio, cor, espessura)

# plota a imagem na tela
cv2.imshow('Imagem com circulo', img)

#tecla "esc" tem ascii código 27
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

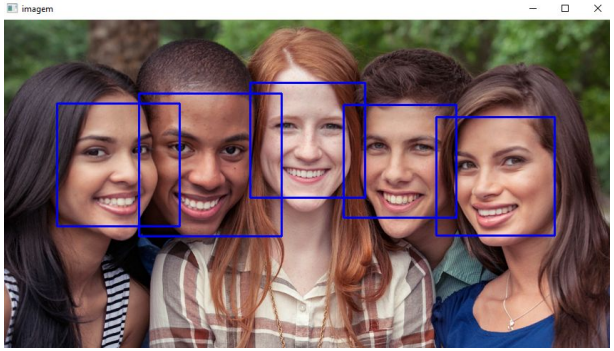
Ex16- Inserindo texto em imagem

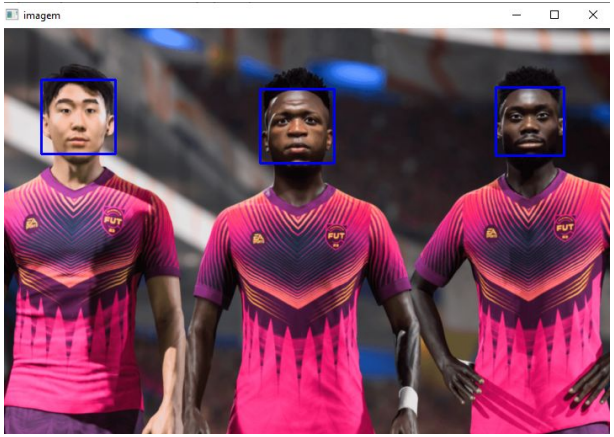
```
3   # importa bibliotecas
4   import cv2
5   import numpy as np
6
7   #Le figura
8   img = cv2.imread('.\imagens\dogs.jpg')
9   print(img.shape)
10
11  # fonte
12  fonte = cv2.FONT_HERSHEY_SIMPLEX
13
14  # org
15  org = (50, 35)
16
17  # escala da fonte
18  escala_fonte = 0.7
19
20  # cor azul em BGR
21  cor = (255, 0, 0)
22
23  # espessura da linha de 2 px
24  espessura = 5
25
26  # o metodo cv2.putText() insere texto na imagem
27  image_result = cv2.putText(img, 'OpenCV no IFMT', org, fonte,
28                             escala_fonte, cor, espessura, cv2.LINE_AA)
29
30  # plota a imagem na tela
31  cv2.imshow('Imagem com circulo', img)
32
33  #tecla "esc" tem ascii código 27
34  if cv2.waitKey(0) & 0xff == 27:
35      cv2.destroyAllWindows()
36
```

ExXX- Detecção de face

```python
# Detecta faces

# importa bibliotecas
import cv2
import numpy as np
import os

# carrega o algoritmo Cascade para detecao de face e atribui o modelo a variavel face_cascade
face_cascade = cv2.CascadeClassifier(os.path.join(cv2.data.haarcascades, 'haarcascade_frontalface_default.xml'))

# le figura
img = cv2.imread('.\\imagens\\faces2.png')
print(img.shape)

# converte em escala de cinza (grayscale)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# detecta as facess
faces = face_cascade.detectMultiScale(gray, 1.1, 4)

# insere um retangulo ao redor das faces detectadas
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
# Plota resultado na tela
cv2.imshow('imagem', img)

#tecla "esc" tem ascii código 27
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```

Além de detectar, insere um retângulo (bounding box) ao redor das faces detectadas

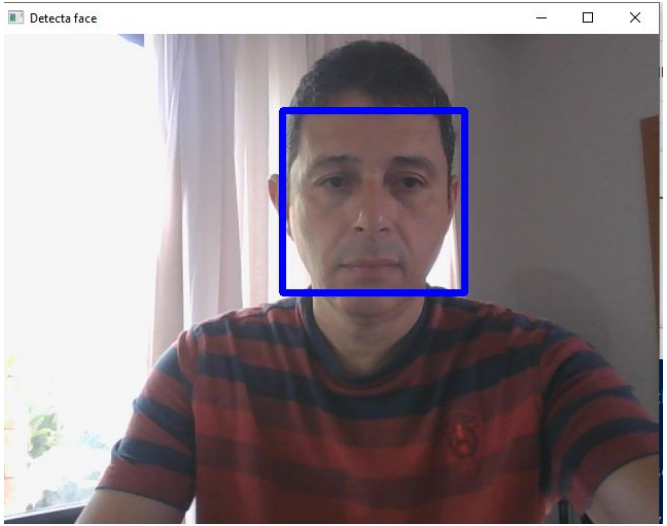




ExXX- Detecção de face em vídeo

```python
# Detecta faces em video

#importa biblioteca
import cv2

# captura video da camera
video = cv2.VideoCapture(0)

# enquanto o video estiver rodando repete o loop
while video.isOpened():
    ret, quadro = video.read()
    # se nao houver retorno em ret, indica fim do video
    if not ret:
        break
    # converte em escala de cinza (grayscale) para processamento mais rapido
    img = cv2.cvtColor(quadro, cv2.COLOR_BGR2GRAY)

    # carrega o algoritmo Cascade para detecao de face e o atribui a variavel classificador_face
    classificador_face = cv2.CascadeClassifier(
        f"{cv2.data.haarcascades}haarcascade_frontalface_default.xml")

    # faz a detecao das objetos (faces) e os insere na matriz definida pelo variavel objetos_detectados
    objetos_detectados = classificador_face.detectMultiScale(
        img, minSize=(20, 20)) # minSize indica o tamanho minimo de are a ser detectada

    # plota um retangulo azul em cada face detectada, se houver
    if len(objetos_detectados) != 0:
        for (x, y, altura, largura) in objetos_detectados:
            cv2.rectangle(
                quadro, (x, y), ((x + altura), (y + largura)), (255, 0, 0), 5)

    # mostra o video com as faces detectadas
    cv2.imshow('Detecta face', quadro)

    if cv2.waitKey(1) == 27:
        break

video.release()
cv2.destroyAllWindows()
```

Além de detectar, insere um retângulo (bounding box) ao redor das faces detectadas

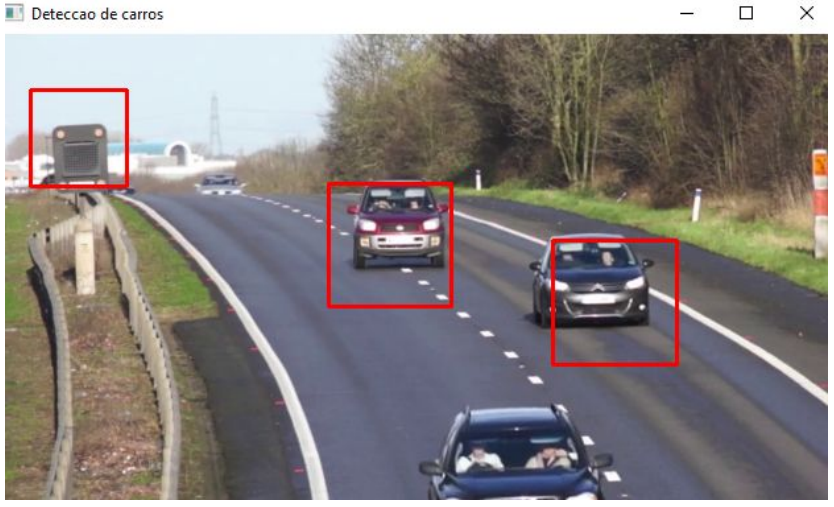


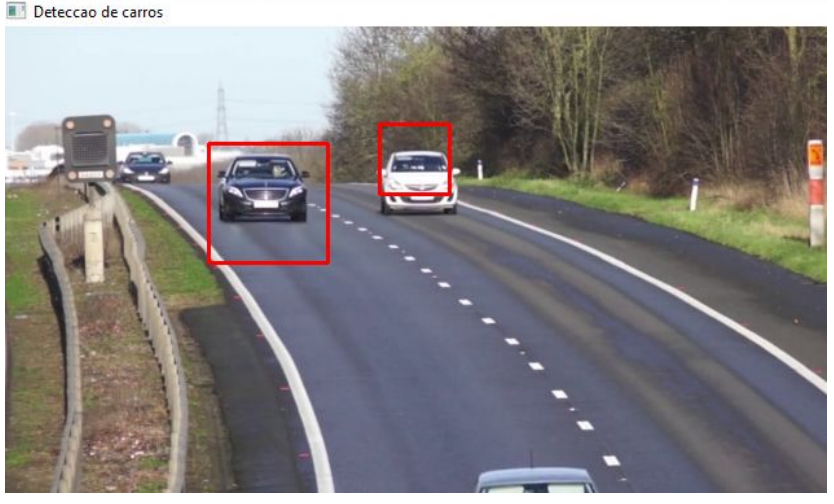
ExXX- Detecção de olhos em vídeo

```python
# Detecta olho em video

#importa biblioteca
import cv2

# captura video da camera
video = cv2.VideoCapture(0)

# enquanto o video estiver rodando repete o loop
while video.isOpened():
    ret, quadro = video.read()
    # se nao houver retorno em ret, indica fim do video
    if not ret:
        break
    # converte em escala de cinza (grayscale) para processamento mais rapido
    img = cv2.cvtColor(quadro, cv2.COLOR_BGR2GRAY)

    # carrega o algoritmo Cascade para detecao olho e o atribui a variavel classificador_face
    classificador_olho = cv2.CascadeClassifier(
        f"{cv2.data.haarcascades}haarcascade_eye.xml")

    # faz a detecao das objetos (olhos) e os insere na matriz definida pelo variavel objetos_detectados
    olhos_detectados = classificador_olho.detectMultiScale(
        img, minSize=(20, 20)) # minSize indica o tamanho minimo de area a ser detectada

    # plota um retangulo azul em cada oho detectado, se houver
    if len(olhos_detectados) != 0:
        for (x, y, altura, largura) in olhos_detectados:
            cv2.rectangle(
                quadro, (x, y), ((x + altura), (y + largura)), (255, 0, 0), 5)

    # mostra o video com as olhos detectados
    cv2.imshow('Detecta face', quadro)

    if cv2.waitKey(1) == 27:
        break

video.release()
cv2.destroyAllWindows()
```

ExXX- Detecção de carro

```python
# Detecta carro

# importa as biblioteca
import cv2

# # carrega video de arquivo
cap = cv2.VideoCapture('.\\videos\\cars.mp4')

# arquivo XML com descrição de caracteristicas do objeto que se quer detectar
classificador_carro = cv2.CascadeClassifier('.\XML\\cars.xml')

# repete ate o final do video
while True:
    # le quadros do video
    ret, quadro = cap.read()

    # converte em escala de cinza (grayscale) para processamento mais rapido
    img_cinza = cv2.cvtColor(quadro, cv2.COLOR_BGR2GRAY)

    # Detecta carros de diferentes tamanhos na imagem cinza
    carros = classificador_carro.detectMultiScale(img_cinza, 1.1, 1)

    # plota um retangulo azul em cada carro detectado, se houver
    for (x,y,w,h) in carros:
        cv2.rectangle(quadro,(x,y),(x+w,y+h),(0,0,255),2)

    # mostra video com carros detetados
    cv2.imshow('Deteccao de carros', quadro)

    if cv2.waitKey(33) == 27:
        break

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```





Referências

- https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- https://www.v7labs.com/blog/computer-vision-applications
- https://www.geeksforgeeks.org/opencv-python-tutorial/
- https://github.com/Aman-Preet-Singh-Gulati/Vehicle-count-detect/blob/main/Vehicle_Detect_Count.ipynb
- https://stackabuse.com/object-detection-with-opencv-python-using-a-haar-cascade-classifier/
- https://www.analyticsvidhya.com/blog/2022/04/object-detection-using-haar-cascade-opencv/

Considerações Finais

- OpenCv dispõe de muito mais funcionalidades
- Mais técnicas de Inteligência Artificial estão sendo incorporadas ao OpenCV
- Haar-Cascade é relativamente básico na detecção de objetos
- Yolo é mais avançado e eficiente em muitos cenários