

Minicurso de OpenCV

Lucas Pascotti Valem

lucaspascottivalem@gmail.com

Universidade Estadual Paulista (UNESP)
Rio Claro, São Paulo, Brazil

24 de Outubro de 2019

O que é?

- O nome é a abreviação de *Open Source Computer Vision*;
- Disponibiliza diversas funções para **Visão Computacional**:
 - Processamento de imagens (detecção, *tracking*, reconhecimento facial, entre outros).
- Biblioteca de programação, de **código aberto**;
- Disponível em diversas linguagens de programação (C++, Python, Ruby, Java, etc.);



Setup

Neste minicurso, iremos utilizar o OpenCV 4 na linguagem Python 3.6 ou superior. Lista de comandos para instalação (no Ubuntu):

- `sudo apt-get install python3`
- `mkdir minicurso`
- `cd minicurso`
- `python3 -m venv env`
- `source env/bin/activate`
- `pip install opencv_python`

Carregando uma Imagem

Carregando uma imagem e exibindo na tela.

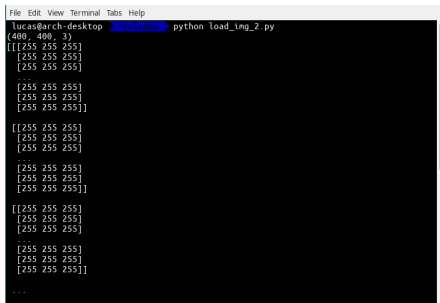
```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("seccomp.jpg")
5 cv2.imshow("image", img)
6 cv2.waitKey()
```



Representação da Imagem

A imagem é representada como uma matriz de pixels (BGR).

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("seccomp.jpg")
5 print(img.shape)
6 print(img)
```

A terminal window with a black background and white text. The title bar shows 'File Edit View Terminal Tabs Help'. The prompt is 'lucas@arch-desktop ~'. The command 'python load_img_2.py' has been executed. The output shows the image shape as '(400, 400, 3)' followed by a grid of BGR pixel values. The first row is '[[255 255 255]', the second is '[255 255 255]', and the third is '[255 255 255]'. This pattern repeats for several rows, with some rows having additional closing brackets, suggesting a truncated or multi-line output. The terminal ends with '...'.

Alterando Pixels

Pintando o pixel (i, j) de vermelho:

```
img[i, j] = [0, 0, 255]
```

Pintando a coluna c de branco:

```
img[:, c] = [255, 255, 255]
```

Pintando a linha l de preto:

```
img[l, :] = [0, 0, 0]
```

Iterando sobre os Pixels

Pintando a imagem de preto:

```
1 height, width, channels = img.shape
2 for h in range(height):
3     for w in range(width):
4         # Pixel passa a ser preto
5         img[h, w] = [0, 0, 0]
```

Salvando uma Imagem

Para salvar uma imagem nova, basta usar a função abaixo. Diferentes tipos de extensões são suportadas.

```
cv2.imwrite("new_img.jpg", img)
```


Criando uma nova imagem

Criando uma imagem com todos os pixels pretos:

```
import cv2
import numpy as np

# Create a black image
height = 100
width = 200
img = np.zeros((height, width), dtype=np.uint8)

cv2.imshow("image", img)
cv2.waitKey()
```

Alterando Resolução

A resolução (dimensões) de uma imagem podem ser alterada pela seguinte função:

```
original_img = cv2.resize(original_img, (200, 200))
```

Desenhando um Círculo

```
1 import cv2
2 import numpy as np
3
4 img = np.zeros((100, 100), dtype=np.uint8)
5
6 # Circle properties
7 center_coordinates = (50, 50)
8 radius = 20
9 color = (255, 0, 0)
10 thickness = -1
11
12 # Draw a circle
13 image = cv2.circle(img, center_coordinates,
14                     radius, color, thickness)
```



Desenhando um Retângulo

```
1 # Rectangle properties
2 x = 20 # top-left vertex
3 y = 25 # top-left vertex
4 width = 50
5 height = 60
6 color = (255, 0, 0)
7 thickness = -1
8
9 # Draw rectangle
10 cv2.rectangle(img,
11                (x, y),
12                (x+width, y+height),
13                color, thickness)
```

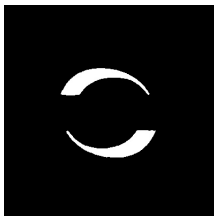


Como podemos pintar o logo da Seccomp de preto?



Segmentação por Cor (Máscara)

```
1 # Read
2 img = cv2.imread("seccomp.jpg")
3
4 # Compute segmentation mask
5 mask = cv2.inRange(img,
6                     (0, 0, 130),
7                     (70, 70, 255))
8
9 # Show it
10 cv2.imshow("mask", mask)
11 cv2.waitKey()
12 cv2.imwrite("mask_seccomp.jpg", mask)
```



Segmentação por Cor (Máscara)

```
1 # Paint regions marked in the mask
2 pts = np.where(mask)
3 pts = zip(*pts)
4 for x, y in list(pts):
5     img[x, y] = [0, 0, 0]
6
7 cv2.imshow("new_logo", img)
8 cv2.waitKey()
9 cv2.imwrite("painted_seccomp.jpg", img)
```



Sensoriamento Remoto

Dada a imagem abaixo, implemente um código que faz uma segmentação aproximada (pela cor) da área de floresta e da área desmatada.



Conversão de espaços de cor

É possível realizar a conversão para diferentes espaços de cor, alguns exemplos:

```
1 img = cv2.imread("seccomp.jpg")
2 img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
3 img = cv2.cvtColor(img, cv2.COLOR_RGB2LAB)
4 img = cv2.cvtColor(img, cv2.COLOR_LAB2HSV)
5 img = cv2.cvtColor(img, cv2.COLOR_HSV2GRAY)
```

Alterando o Brilho

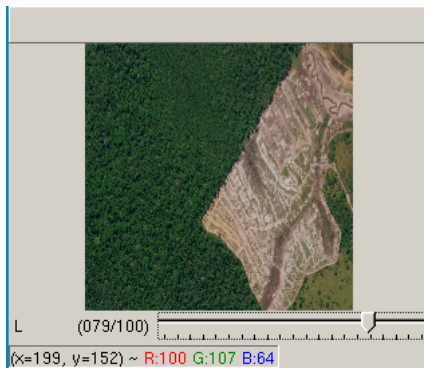
O brilho pode ser alterado pelo espaço de cores LAB, modificando a componente L (0-255).

```
1 import cv2
2
3 img = cv2.imread("amazon.jpg")
4 img = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
5 img[:, :, 0] = img[:, :, 0] * 0.2
6 img = cv2.cvtColor(img, cv2.COLOR_LAB2BGR)
7 cv2.imshow("image", img)
8 cv2.waitKey()
9 cv2.imwrite("darker_amazon.jpg", img)
```



Usando Trackbars

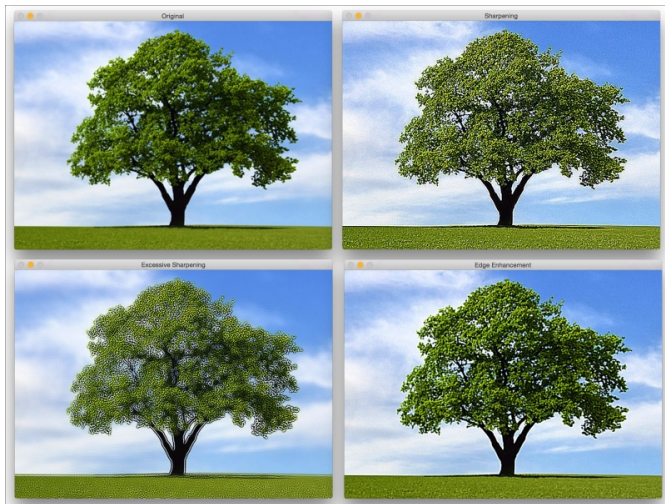
Trackbars facilitam a alteração dinâmica de valores parâmetros!



Usando Trackbars

```
1 import cv2
2 import numpy as np
3
4 def nothing(x):
5     pass
6
7 cv2.namedWindow("image")
8 cv2.createTrackbar("L", "image", 0, 100, nothing)
9
10 original_img = cv2.imread("amazon.jpg")
11 original_img = cv2.cvtColor(original_img, cv2.COLOR_BGR2LAB)
12 original_img = cv2.resize(original_img, (200, 200))
13
14 while True:
15     img = original_img.copy()
16     factor = cv2.getTrackbarPos("L", "image")/100
17     img[:, :, 0] = img[:, :, 0]*factor
18
19     img = cv2.cvtColor(img, cv2.COLOR_LAB2BGR)
20     cv2.imshow("image", img)
21
22     k = cv2.waitKey(1) & 0xFF
23     if k == ord('q'):
24         break
25
26 cv2.destroyAllWindows()
```

Aplicando Filtros



Filtros podem ser aplicados pela definição de kernels de convolução.

Exemplos:

$$\text{Identidade} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

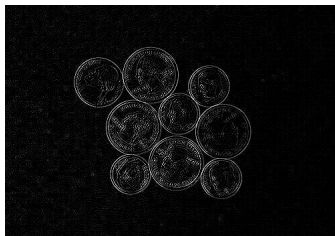
$$\text{Detecção de contorno} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\text{Realce de Contorno} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$\text{Gaussiano 3x3 (borra a imagem)} = \frac{1}{256} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Aplicando Filtros

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('coins.jpg')
5
6 kernel = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])
7 dst = cv2.filter2D(img, -1, kernel)
8
9 cv2.imshow('new_image', dst)
10 cv2.waitKey()
11 cv2.imwrite('coins_shape.jpg', dst)
```



Aplicando Thresholding

```
1 import cv2
2
3 img = cv2.imread("coins.jpg", 0)
4 _, thresh = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
5 cv2.imwrite("thresh_img.png", thresh)
```



Concatenando Imagens

```
1 import cv2
2 import numpy as np
3
4 im1 = np.full((100, 100), 100, dtype=np.uint8)
5 im2 = np.full((100, 100), 0, dtype=np.uint8)
6
7 im_h = np.hstack([im1, im2])
8 im_v = np.vstack([im1, im2])
9
10 cv2.imwrite('concat_horizontal.png', im_h)
11 cv2.imwrite('concat_vertical.png', im_v)
```



Recortando Imagens

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("mario.jpg")
5 img = img[200:500, 200:-200]
6 cv2.imwrite("mario_cropped.jpg", img)
```

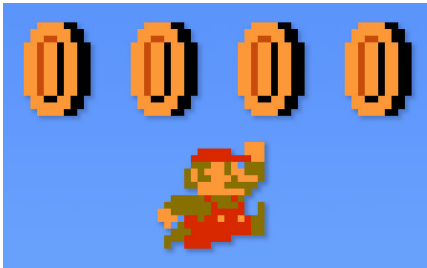


Imagem Original



Imagem Recortada

Identificando Padrões

A identificação de padrões, nesse caso *template matching* pode ser feita a partir de uma imagem de procura e uma imagem principal.

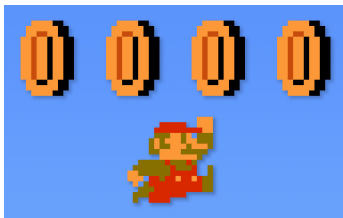


Imagem Principal



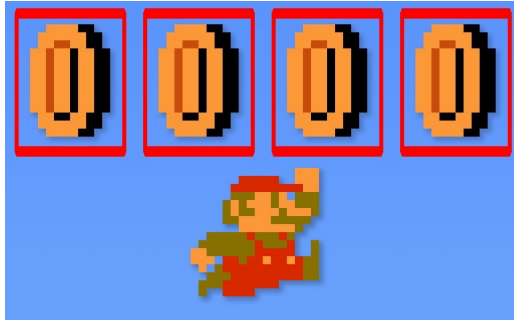
Padrão (*Template*)

Identificando Padrões

Detecta o padrão e desenha um retângulo vermelho em cada detecção:

```
1 import cv2
2 import numpy as np
3
4 main_image = cv2.imread("mario.jpg")
5 gray_image = cv2.cvtColor(main_image, cv2.COLOR_BGR2GRAY)
6
7 template = cv2.imread("pattern.jpg", 0)
8 w, h = template.shape[::-1]
9
10 match = cv2.matchTemplate(gray_image, template, cv2.TM_CCOEFF_NORMED)
11 threshold = 0.8
12 position = np.where(match >= threshold)
13 for point in zip(*position[::-1]):
14     cv2.rectangle(main_image,
15                   point,
16                   (point[0] + w, point[1] + h),
17                   (0, 0, 255), 0)
18 cv2.imshow("Template found", main_image)
19 cv2.waitKey(0)
```

Identificando Padrões



Carregando e Exibindo Vídeos

O OpenCV permite a visualização de vídeos armazenados em memória. Cada frame pode ser manipulado como uma imagem.

```
1 import cv2
2
3 cap = cv2.VideoCapture("video_path.mp4")
4
5 while True:
6     ret, frame = cap.read()
7
8     cv2.imshow("video", frame)
9
10    k = cv2.waitKey(1) & 0xFF
11    if k == ord('q'):
12        break
13
14 cv2.destroyAllWindows()
15 cap.release()
```

Carregando a Webcam

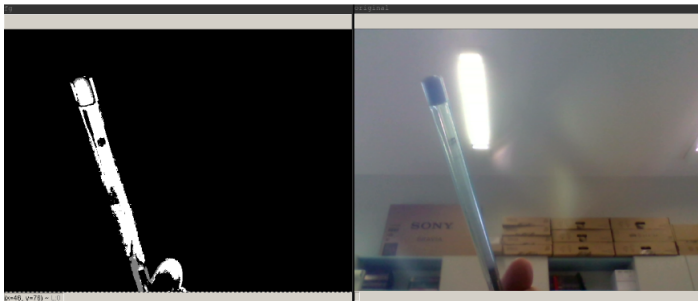
Para visualizar a webcam, basta colocar o ID da webcam na função `cv2.VideoCapture` ao invés do caminho do vídeo:

```
1  import cv2
2
3  cap = cv2.VideoCapture(0)
4
5  while True:
6      ret, frame = cap.read()
7
8      cv2.imshow("video", frame)
9
10     k = cv2.waitKey(1) & 0xFF
11     if k == ord('q'):
12         break
13
14 cv2.destroyAllWindows()
15 cap.release()
```

Subtração de Background

Subtratores de background podem ser usados com a finalidade **identificar diferenças entre imagens**.

Um dos mais populares é o MOG, que pode ser instanciado e utilizado em conjunto com a imagem da webcam, por exemplo.



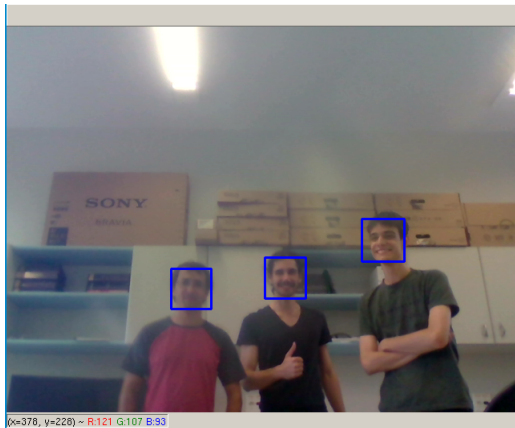
Subtração de Background

Subtração de Background

```
1 import cv2
2 import numpy as np
3
4 cap = cv2.VideoCapture(0)
5 fgbg = cv2.createBackgroundSubtractorMOG2()
6
7 while True:
8     _, frame = cap.read()
9
10    fgmask = fgbg.apply(frame, learningRate=-1)
11
12    cv2.imshow("video", frame)
13    cv2.imshow("mask", fgmask)
14
15    k = cv2.waitKey(30) & 0xff
16    if k == ord('q'):
17        break
18
19 cap.release()
20 cv2.destroyAllWindows()
```

Detecção de Faces

A detecção de faces pode ser realizada carregando os pesos de treinamento de um classificador.



Detecção de Faces

```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4
5 face_cascade = cv2.CascadeClassifier('haarcascade_frontalface.xml')
6
7 while (True):
8     _, frame = cap.read()
9
10    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
11    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
12    for (x, y, w, h) in faces:
13        cv2.rectangle(frame,
14                       (x, y),
15                       (x + w, y + h),
16                       (255, 0, 0), 2)
17
18    cv2.imshow('frame', frame)
19
20    key = cv2.waitKey(1)
21    if key == 27:
22        break
23
24 cap.release()
25 cv2.destroyAllWindows()
```

Obrigado!

Questões?

