

Relatório sobre o código de processamento de imagens utilizando a biblioteca OpenCV

Jean Carlo Sanchuki Filho

Introdução

O objetivo deste relatório é descrever o código desenvolvido para o processamento de imagens utilizando a biblioteca OpenCV, com o intuito de explicar suas funcionalidades e seus usos na vida real.

Código

```
• • •

import cv2
import numpy as np

img_path = 'cv1/person.jpg'
img = cv2.imread(img_path)
img4harri = np.copy(img)

#Tratamento de imagem
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
operatedImage = np.float32(gray)

#Threshold
thresh_value = 100
thresh1 = cv2.threshold(gray, thresh_value, 255, cv2.THRESH_BINARY)[1]
thresh2 = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY)[1]

#Harris
dest = cv2.cornerHarris(operatedImage, 2, 5, 0.07)
threshold = 0.01 * dest.max()
corner_mask = dest > threshold
corner_coords = np.transpose(np.nonzero(corner_mask))
radius = 10
color = (0, 255, 0)
thickness = 1
for corner in corner_coords:
    center = (corner[1], corner[0])
    cv2.circle(img4harri, center, radius, color, thickness)

#Sharpen e Blur
kernel_sharpen = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
kernel_blur = np.ones((5,5), np.float32)/25
img_shARPened = cv2.filter2D(img, -1, kernel_sharpen)
img_bLURRED = cv2.filter2D(img, -1, kernel_blur)
```

Descrição do código



Figuras 1 e 2: Imagens originais utilizadas para o processamento

Tratamento de imagem

A imagem é transformada em cinza para que a mesma possa ser processada de forma mais eficiente, pois muitos algoritmos são projetados para trabalhar com imagens em tons de cinza.



Figuras 3 e 4: Figuras 1 e 2 após serem transformadas em tons de cinza

Em seguida, a imagem em tons de cinza é convertida em um array de valores de ponto flutuante usando a função. Isso é necessário porque muitos algoritmos de processamento de imagens exigem que as imagens sejam representadas como matrizes de valores [Figura 5].

```

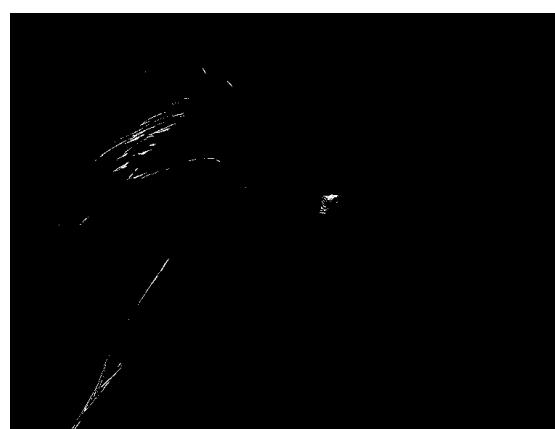
● ● ●

[[83. 83. 83. ... 55. 56. 56.]
 [83. 83. 83. ... 55. 56. 56.]
 [83. 83. 83. ... 56. 57. 57.]
 ...
 [36. 35. 36. ... 67. 67. 67.]
 [35. 35. 35. ... 66. 66. 66.]
 [35. 35. 35. ... 66. 66. 66.]]

```

Figura 5: Exemplo de matriz que representa uma imagem.

Threshold



Figuras 6 e 7: Figura 2 após a aplicação do threshold.

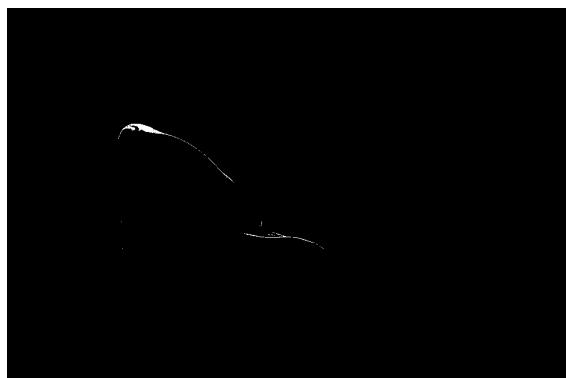


Figura 8 e 9: Figura 3 após a aplicação do threshold.

A função threshold é usada para segmentar uma imagem em regiões de pixels que possuem valores acima ou abaixo de um determinado limite. O limite é definido pelo usuário, e pode ser ajustado de acordo com a necessidade.

Por exemplo, as Figuras 6 e 8 foram processadas com o threshold em 100 e as Figuras 7 e 9 foram processadas com o threshold em 200.

Detecção de curvas

A detecção de curvas é usada para identificar as bordas em uma imagem. O algoritmo de detecção de curvas de Harris é um método comum usado em visão computacional para encontrar características (também chamadas de pontos de interesse) em uma imagem.

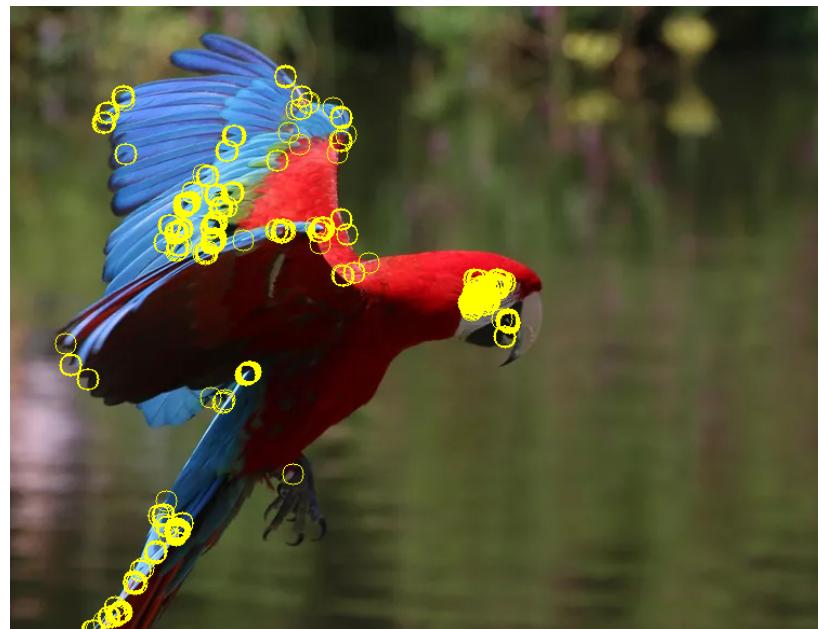


Figura 10: Figura 1 com círculos amarelos em seus pontos de interesse.

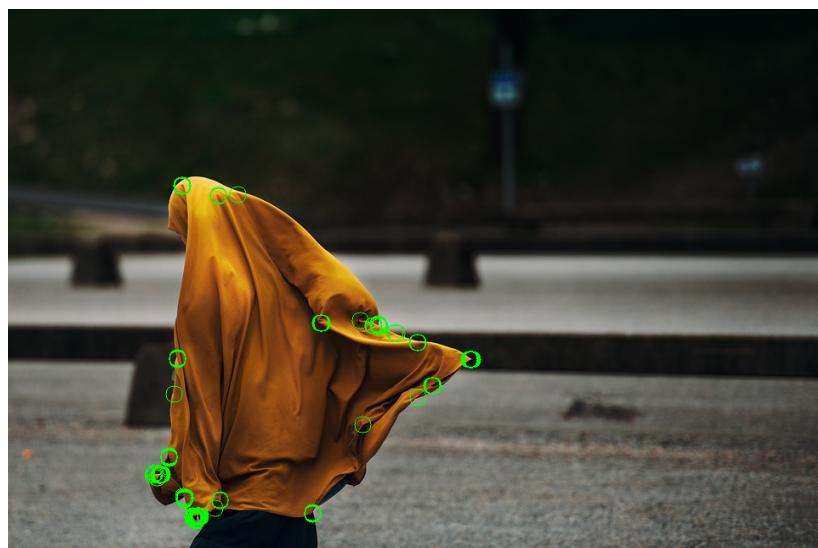


Figura 11: Figura 2 com círculos verdes em seus pontos de interesse.

Aplicação de efeitos

O sharpen é usado para realçar as características da imagem, dando a ela uma aparência mais nítida. Ele é composto por uma matriz 3x3 com valores negativos (-1) em torno da célula central e um valor alto (9) na célula central. Ao aplicar esse kernel em uma imagem, a célula central (que representa o pixel atual) é multiplicada por 9 e os pixels vizinhos são multiplicados por -1. Essa operação faz com que as bordas e os detalhes da imagem sejam realçados, conforme apresentado abaixo [Figura 12] [Figura 13].

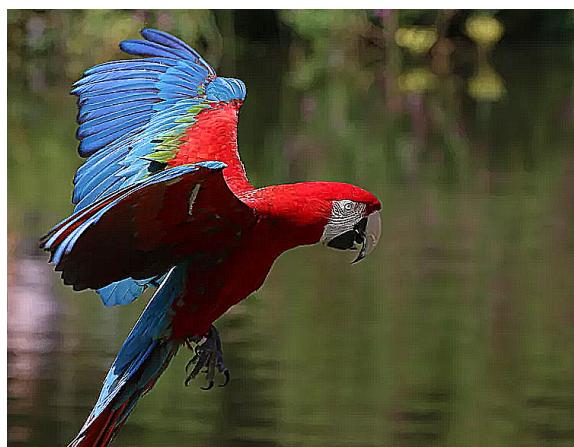


Figura 12 e 13: Figuras 1 e 2 após o tratamento de realce.

O blur é usado para suavizar a imagem, removendo pequenos detalhes e reduzindo o ruído. Ele é composto por uma matriz 5x5 de valores iguais (1/25), que representam a média ponderada dos pixels em torno da célula central. Ao aplicar esse kernel em uma imagem, a célula central (que representa o pixel atual) é multiplicada por 1/25 e os pixels vizinhos também são multiplicados por 1/25. Essa operação faz com que os valores dos pixels sejam "espalhados" ao redor da célula central, suavizando a imagem [Figura 14] [Figura 15].



Figura 14 e 15: Figuras 1 e 2 após a aplicação de blur (desfoco).

Aplicações na vida real

- Detecção de objetos em imagens de vigilância.
- Reconhecimento de objetos.
- Detecção de bordas em imagens médicas, como tomografia e ressonância magnética.
- Reconhecimento de padrões.