
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
**CAMPUS SEDE**
**CAPÍTULO ESTUDANTIL IEEE RAS UFCG**
**PROCESSO SELETIVO DE INTEGRANTES DO CAPÍTULO ESTUDANTIL IEEE RAS UFCG**

**PRIMEIRA ETAPA**
**MISSÕES INTRODUTÓRIAS**
**MISSÃO 1.0**

# CONHECENDO O OpenCV

**CANDIDATA: Mércia Regina da Silva**

**CAMPINA GRANDE, 2024.**

**SUMÁRIO**

## 1. INTRODUÇÂO

**Visão Computacional – OpenCV -** a visão computacional procura modelar e replicar a visão humana usando software e hardware. Tentando simular a visão natural, a visão computacional tem embasamento cientifico decorrente de estudos de algoritmos que buscam analisar e compreender conceitos da visão humana nas áreas de Biologia, Óptica e Matemática (álgebra linear, geometria e estatística).

OpenCV (Open Source Computer Vision) é uma biblioteca de programação, de código aberto e inicialmente desenvolvida pela Intel com o objetivo de tornar a visão computacional mais acessível a desenvolvedores.

Uma outra biblioteca utilizada na execução desta missão foi a **Numpy** denominado dessa forma devido a abreviação de **Numerical Python** (**Python Numérico**), é uma biblioteca de código aberto destinada a realizar operações em arrays multidimensionais, amigavelmente denominada como ndarray nesta biblioteca.

Nesta Missão 2.0 com os estudos básicos de conceitos sobre uma biblioteca de visão computacional e numérica foram realizadas as manipulações e as operações dentro de matrizes (arrays) seguindo o padrão de cores RGB e no decorrer das execuções este padrão de cores foram modificadas durante a manipulação da imagem.

## 2. OBJETIVOS

✓Um sistema de visão computacional para manipular uma imagem em tempo real utilizando as bibliotecas OpenCV com a visão computacional e a Numpy com a manipulação de valores de cores RGB dentro de arrays (matrizes).

**Material utilizado:**

✓Ubuntu;

✓Python 3;

✓Jupyter Notebook - é uma aplicação web de código aberto que permite criar e compartilhar documentos interativos chamados de “cadernos” (notebooks);

✓Biblioteca cv2;

✓Biblioteca Numpy;

✓Imagem com a extensão JPE;

## 3. METODOLOGIA

3.1 Foi realizada a leitura e execução dos exemplos do Capítulo 1 e 2 da Apostila Introdução a Visão Computacional com Python e OpenCV, segue os códigos copiados do material:

```
# Importação das bibliotecas
import cv2

# Leitura da imagem com a função imread()
imagem = cv2.imread('entrada.jpg')

print('Largura em pixels: ', end='')
print(imagem.shape[1]) #largura da imagem
print('Altura em pixels: ', end='')
print(imagem.shape[0]) #altura da imagem
print('Qtde de canais: ', end='')
print(imagem.shape[2])

# Mostra a imagem com a função imshow
cv2.imshow("Nome da janela", imagem)
cv2.waitKey(0) #espera pressionar qualquer tecla

# Salvar a imagem no disco com função imwrite()
cv2.imwrite("saida.jpg", imagem)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
(b, g, r) = imagem[0, 0] #veja que a ordem BGR e não RGB

print('O pixel (0, 0) tem as seguintes cores:')
print('Vermelho:', r, 'Verde:', g, 'Azul:', b)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0]):
  for x in range(0, imagem.shape[1]):
    imagem[y, x] = (255,0,0)
cv2.imshow("Imagem modificada", imagem)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0]): #percorre linhas
  for x in range(0, imagem.shape[1]): #percorre colunas
    imagem[y, x] = (x%256,y%256,x%256)
cv2.imshow("Imagem modificada", imagem)
cv2.waitKey(0)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0]): #percorre linhas
  for x in range(0, imagem.shape[1]): #percorre colunas
    imagem[y, x] = (x%256,y%256,x%256)
cv2.imshow("Imagem modificada", imagem)
cv2.waitKey(0)
```

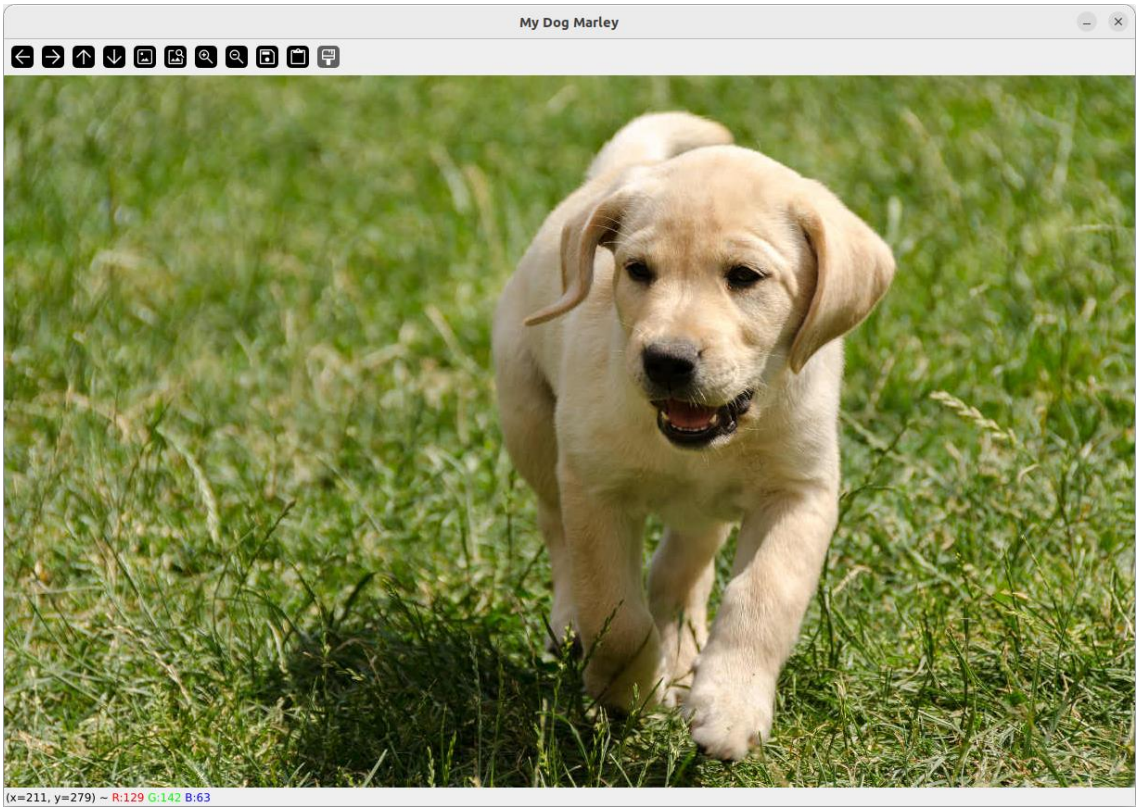
```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0], 1): #percorre as linhas
  for x in range(0, imagem.shape[1], 1): #percorre as colunas
    imagem[y, x] = (0,(x*y)%256,0)
cv2.imshow("Imagem modificada", imagem)
cv2.waitKey(0)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0], 10): #percorre linhas
  for x in range(0, imagem.shape[1], 10): #percorre colunas
    imagem[y:y+5, x: x+5] = (0,255,255)
cv2.imshow("Imagem modificada", imagem)
cv2.waitKey(0)
```

**3.2 O código foi executado no programa Jupyter Notebook com seus devidos comentários em cada execução:**

3.2.1 – A imagem utilizada com formato JPE foi manipulada a partir de sua imagem original seguindo os padrões de cores RGB (R-red, F-green, B-blue).

A função *imread* (realizada a leitura da imagem enviada), essa imagem é lida na em entrada pela função *print* com o comando *imagem.shape*[0] temos um *array*. E sua saída seria executada com a leitura da imagem com a função *cv2.imwrite* desta forma:

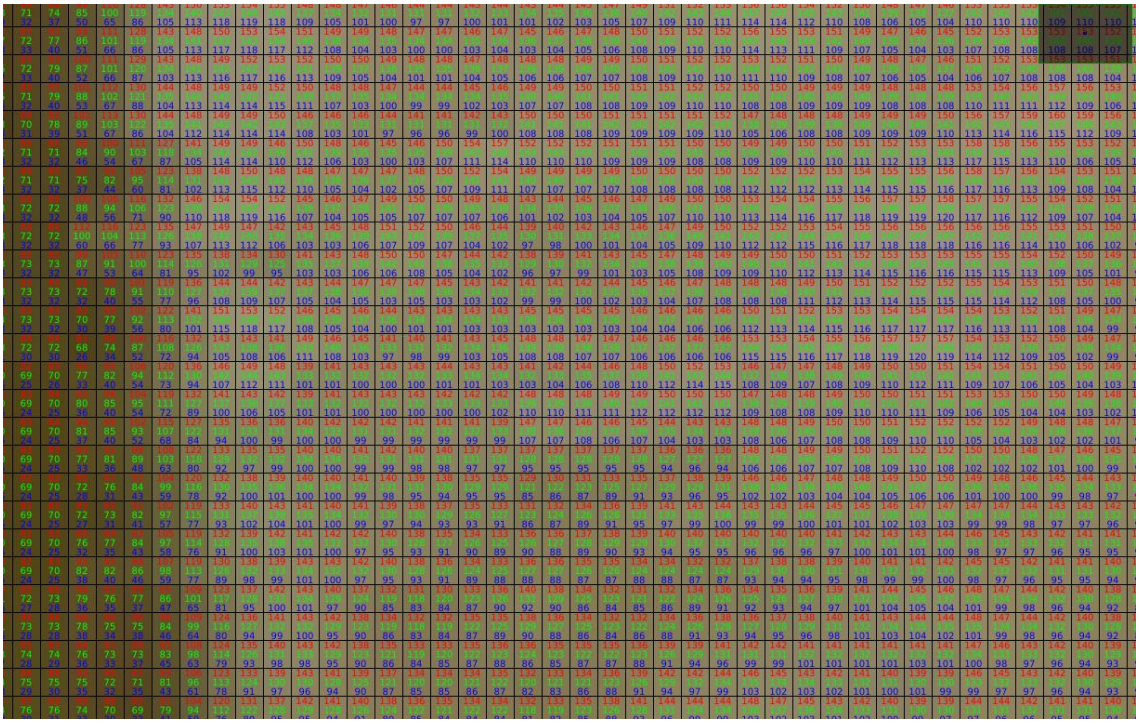



```
# Importação das bibliotecas
import cv2

# Leitura da imagem com a função imread()
imagem = cv2.imread('entrada.jpg')

print('Largura em pixels: ', end='')
print(imagem.shape[1]) #largura da imagem
print('Altura em pixels: ', end='')
print(imagem.shape[0]) #altura da imagem
print('Qtde de canais: ', end='')
print(imagem.shape[2])

# Mostra a imagem com a função imshow
cv2.imshow("Nome da janela", imagem)
cv2.waitKey(0) #espera pressionar qualquer tecla

# Salvar a imagem no disco com função imwrite()
cv2.imwrite("saida.jpg", imagem)
```

**3.2.2 No capítulo 2 do material teórico utilizado foi feito o estudo do Sistema de Coordenadas e Manipulação de Pixels.**

Que através do sistema de coordenadas é possível alterar individualmente cada pixel ou ler a informação individual do pixel, usando a função *imread*, neste caso a imagem é definida como uma matriz **Numpy** que é armazenada em memória através da variável imagem, o código de retorno na tupla foi (b, g, r) os respectivos valores das cores do pixel superior mais a esquerda.

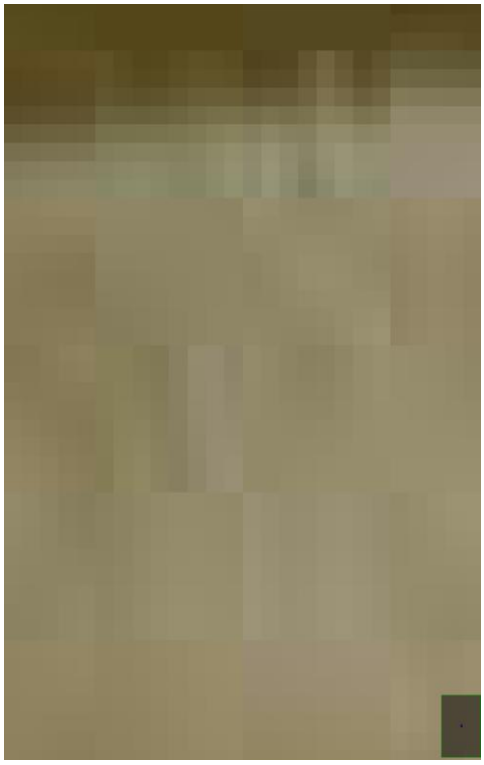
O método nos retorna uma sequência BGR conforme a imagem abaixo:



```
import cv2
imagem = cv2.imread('ponte.jpg')
(b, g, r) = imagem[0, 0] #veja que a ordem BGR e não RGB

print('O pixel (0, 0) tem as seguintes cores:')
print('Vermelho:', r, 'Verde:', g, 'Azul:', b)
```

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0]):
  for x in range(0, imagem.shape[1]):
    imagem[y, x] = (255,0,0)
cv2.imshow("Imagem modificada", imagem)
```

**3.2.3 A alteração nas componentes das cores da imagem conforme as coordenadas de linha e coluna geram a imagem abaixo:**



As variáveis de linha e coluna para serem as componentes de cor, onde as variáveis componentes da cor devem assumir o valor entre 0 e 255 foi utilizado a operação resto da divisão por 256 para manter o resultado entre 0 e 255.

```
import cv2
imagem = cv2.imread('ponte.jpg')
for y in range(0, imagem.shape[0]): #percorre linhas
  for x in range(0, imagem.shape[1]): #percorre colunas
    imagem[y, x] = (x%256,y%256,x%256)
cv2.imshow("Imagem modificada", imagem)
cv2.waitKey(0)
```

## 4. REFERÊNCIA

Apostila **“Introdução a Visão Computacional com Python e OpenCV”**, Versão 0.8 – Não corrigida, Capítulos 1 e 2, páginas 6 a 11.
Site: www.antonello.com.br