Primeira Atividade Prática de Processamento Digital de Imagens

Francisco Jair de Oliveira Neres[1]

[1]Departamento de Computação – Universidade Federal do Piaui (UFPI)
Teresina – PI – Brasil

fjair123@gmail.com

1. Introdução

O objetivo dessa atividade consiste em explorar na prática os conceitos básicos aprendidos até o momento na disciplina de Processamento Digital de Imagens. Neste sentido, a parte prática da atividade consistiu em desenvolver um programa que implementasse as seguintes funções de transformação de intensidade de imagem: função logarítmica, função exponencial e função inversa.

Além de Foram implementados modelos de cor (transformação de RGB para HSV, transformação de HSV para RGB, transformação para preto e branco, exibição de canais de cores) e filtros espaciais (filtro da mediana, filtro da média e filtro de média ponderada), algumas transformações exibem canais de cores. Por fim, foram implementadas algumas funções relacionadas a histograma, como: Exibir, Equalizar, Especificar e Comparar. Vale a pena ressaltar que para os filtros da média, mediana e média ponderada foi usado o valor 5x5 para caixa de cálculo.

2. Desenvolvimento

O código pode ser acessado no seguinte link: Github. De qualquer forma, o código implementado pode ser visto abaixo, feito utilizando a linguagem Python, com auxílio das bibliotecas OpenCV, NumPy e Matplotlib.

```
import numpy as np
from os import system
import matplotlib.pyplot as plt
import cv2


img = cv2.imread("img.jpg")
img2 = cv2.imread("img2.jpg")



def show_hist(img):
    h = cv2.calcHist([img], [0], None, [256], [0, 256])
    plt.figure()
    plt.title("Histograma")
```

```
    plt.xlabel("Intensidade")
    plt.ylabel("Número de Pixels")
    plt.plot(h)
    plt.xlim([0, 256])
    plt.show()


def transformation_views(img):
    plt.imshow(img)
    plt.show()
    show_hist(img)


def hist_colors(img):
    chanels = cv2.split(img)
    c = ("b", "g", "r")
    plt.figure()
    plt.title("Histograma Colorido")
    plt.xlabel("Intensidade")
    plt.ylabel("Número de Pixels")
    for (canal, cor) in zip(chanels, c):
        hist = cv2.calcHist([canal], [0], None, [256], [0, 256])
        plt.plot(hist, color=cor)
        plt.xlim([0, 256])
    plt.show()


def hist_equalizer(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    h_eq = cv2.equalizeHist(img)
    show_hist(h_eq)
    show_hist(img)
```

```
def modelsandfilters_view(new_img, img):
    try:
        (h, s, v) = cv2.split(new_img)
        zeros = np.zeros(new_img.shape[:2], dtype="uint8")
        cv2.imshow("Vermelho", cv2.merge([zeros, zeros, h]))
        cv2.imshow("Verde", cv2.merge([zeros, s, zeros]))
        cv2.imshow("Azul", cv2.merge([v, zeros, zeros]))
        cv2.imshow("Transformada", new_img)
        cv2.imshow("Original", img)
        cv2.waitKey(0)
    except:
        cv2.imshow("Transformada", new_img)
        cv2.imshow("Original", img)
        cv2.waitKey(0)


def log_transformation(img):
    num = float(input("Digite o valor a se calculado o log(>0):"))
    c = 255 / (np.log(num))
    log_img = np.array(c * (np.log(img + 1)), dtype=np.uint8)
    transformation_views(log_img)


def inv_transformation(img):
    inv_img = 255 - img
    transformation_views(inv_img)


def expo_transformation(img):
    expo = float(input("Digite o valor do expoente:"))
```

```
        expo_img = np.array(255 * (img / 255) ** expo, dtype="uint8")
        transformation_views(expo_img)


    def rgb_to_hsv(img):
        new_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        modelsandfilters_view(new_img, img)


    def hsv_to_rgb(img):
        new_img = cv2.cvtColor(img, cv2.COLOR_HSV2BGR)
        modelsandfilters_view(new_img, img)


    def rgb_to_gray(img):
        new_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        modelsandfilters_view(new_img, img)


    def median_filter(img):
        new_img = cv2.medianBlur(img, 5)
        modelsandfilters_view(new_img, img)


    def avg_filter(img):
        new_img = cv2.blur(img, (5, 5))
        modelsandfilters_view(new_img, img)


    def avg_pond_filter(img):
        new_img = cv2.GaussianBlur(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY), (5,
5), 0)
```

```
    modelsandfilters_view(new_img, img)


def main_menu():
    op = input(
        "Escolha uma opcao: \n \
                1 - Aplicar Transformações de Intensidade\n \
                2 - Opções relacionadas a Histograma\n \
                3 - Modelos de cor e Filtros Espaciais\n \
                Opção:"
    )
    return op


def  menu_filtersandmodels(img):
    op = input(
        "Escolha uma opcao: \n \
                1 - Transformar para RGB para HSV\n \
                2 - Transformar para HSV para RGB\n \
                3 - Transformar para preto e branco\n \
                4 - filtro da mediana\n \
                5 - filtro da media\n \
                6 - filtro da media ponderada\n \
                Opção:"
    )
    if op == "1":
        rgb_to_hsv(img)
    elif op == "2":
        hsv_to_rgb(img)
    elif op == "3":
        rgb_to_gray(img)
    elif op == "4":
```

```
        median_filter(img)
    elif op == "5":
        avg_filter(img)
    elif op == "6":
        avg_pond_filter(img)


def transfor_menu(img):
    op = input(
        "Escolha uma opcao: \n \
                1 - Aplicar Transformações Logaritma\n \
                2 - Aplicar Transformações Exponencial\n \
                3 - Aplicar Transformações Inversa\n \
                Opção:"
    )
    if op == "1":
        log_transformation(img)
    elif op == "2":
        expo_transformation(img)
    elif op == "3":
        inv_transformation(img)


def hist_menu(img, img2):
    op = input(
        "Escolha uma opcao: \n \
                1 - Exibir\n \
                2 - Equalizar\n \
                3 - Especificar\n \
                4 - Comparar\n \
                Opção:"
    )
```

```
    if op == "1":
        transformation_views(img)
        hist_colors(img2)
    elif op == "2":
        hist_equalizer(img)
        hist_equalizer(img2)
    elif op == "3":
        transfor_menu(img)
    elif op == "4":
        transfor_menu(img)
        transformation_views(img)
        hist_equalizer(img)


op = main_menu()
system("clear")
if op == "1":
    transfor_menu(img)
elif op == "2":
    hist_menu(img, img2)
elif op == "3":
    menu_filtersandmodels(img2)
```

3. Resultados

Após a execução do código, obtemos os seguintes resultados. Logo abaixo podemos observar a Imagem 1 e a Imagem 2 originais usadas nos testes.

Imagem 1. Preto e Branco.

Imagem 2. Colorida.

3.1. Histograma da Imagem 1 e 2.

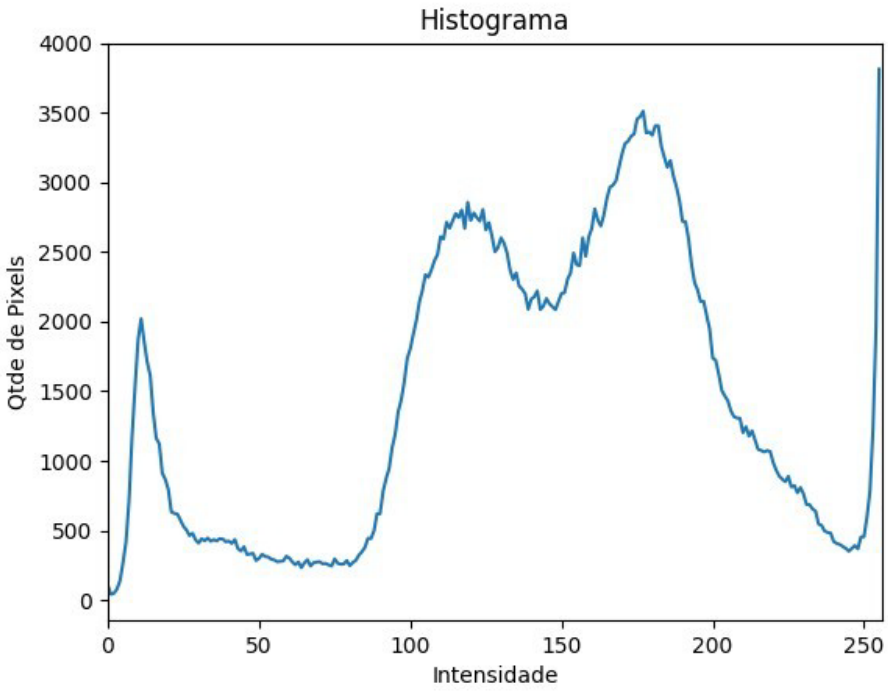


Figura 3. Histograma da Imagem 1

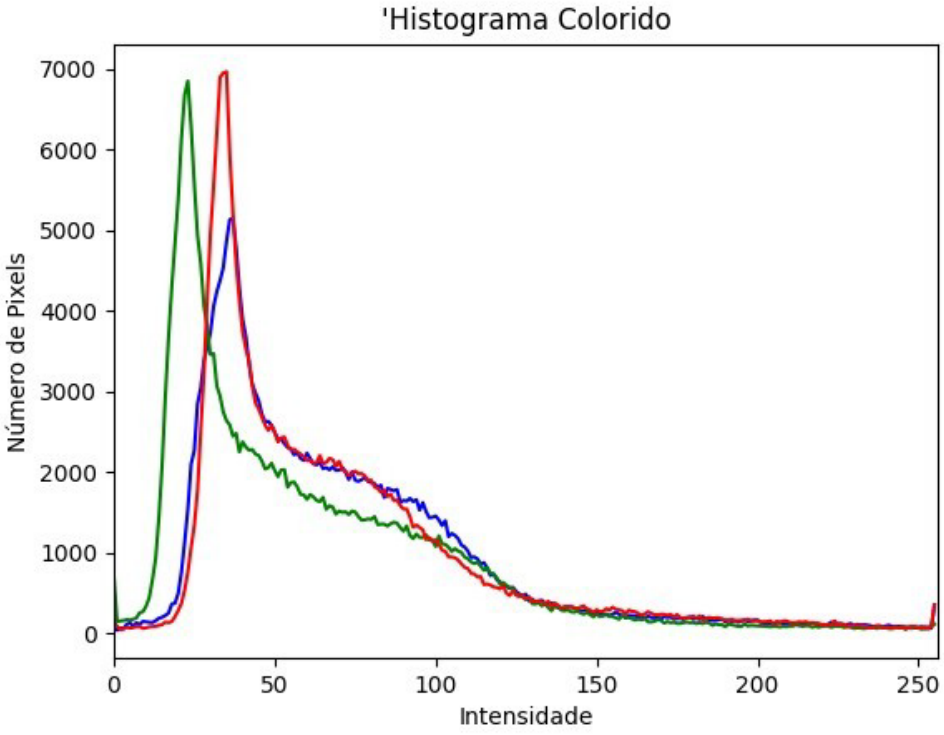


Figura 4. Histograma da Imagem 2.

3.2. Histograma Equalizado

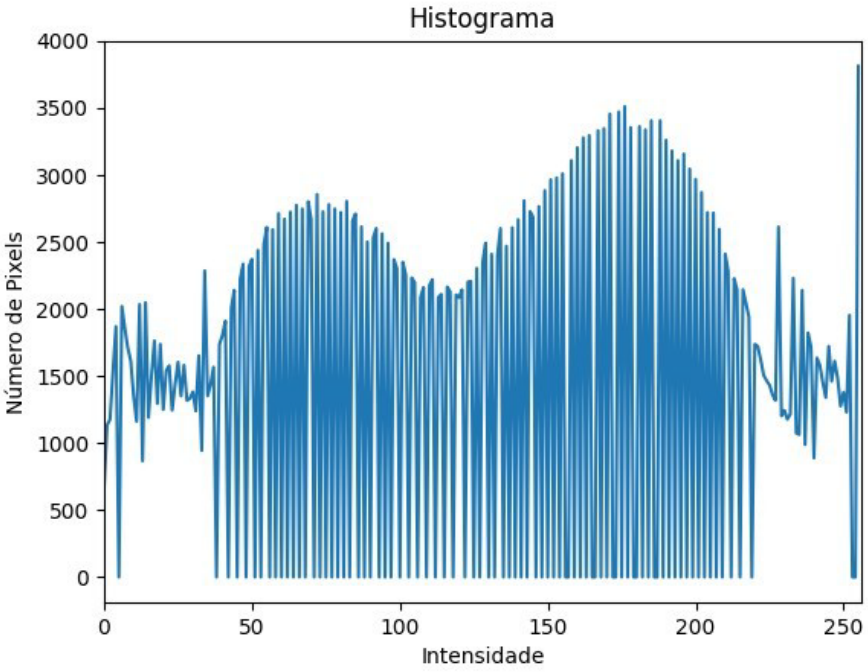


Figura 5. Histograma equalizado da Imagem 1.

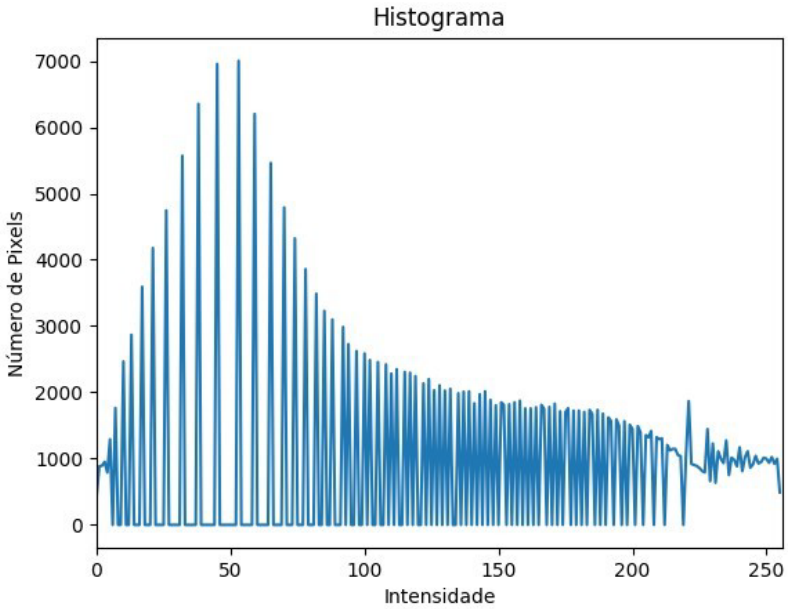


Figura 6. Histograma equalizado da Imagem 2(Transformada para Preto e Branco).

3.3. Transformações

3.3.1. Transformação Logaritma(foi usado o Log(50), esse valor por ser informado na execução do código).

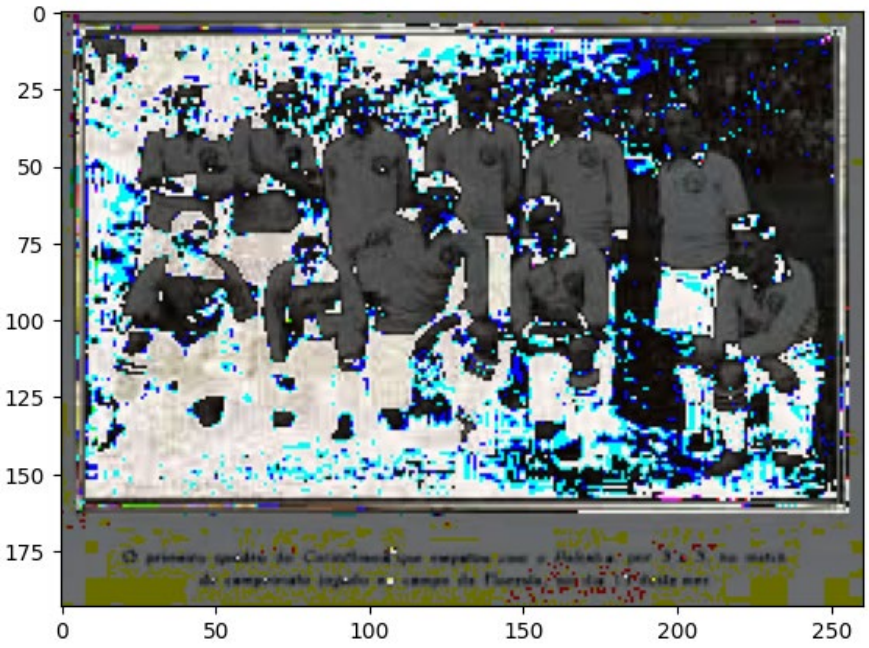


Figura 7. Transformação Logaritma(Log(5)) aplicado na imagem 1.

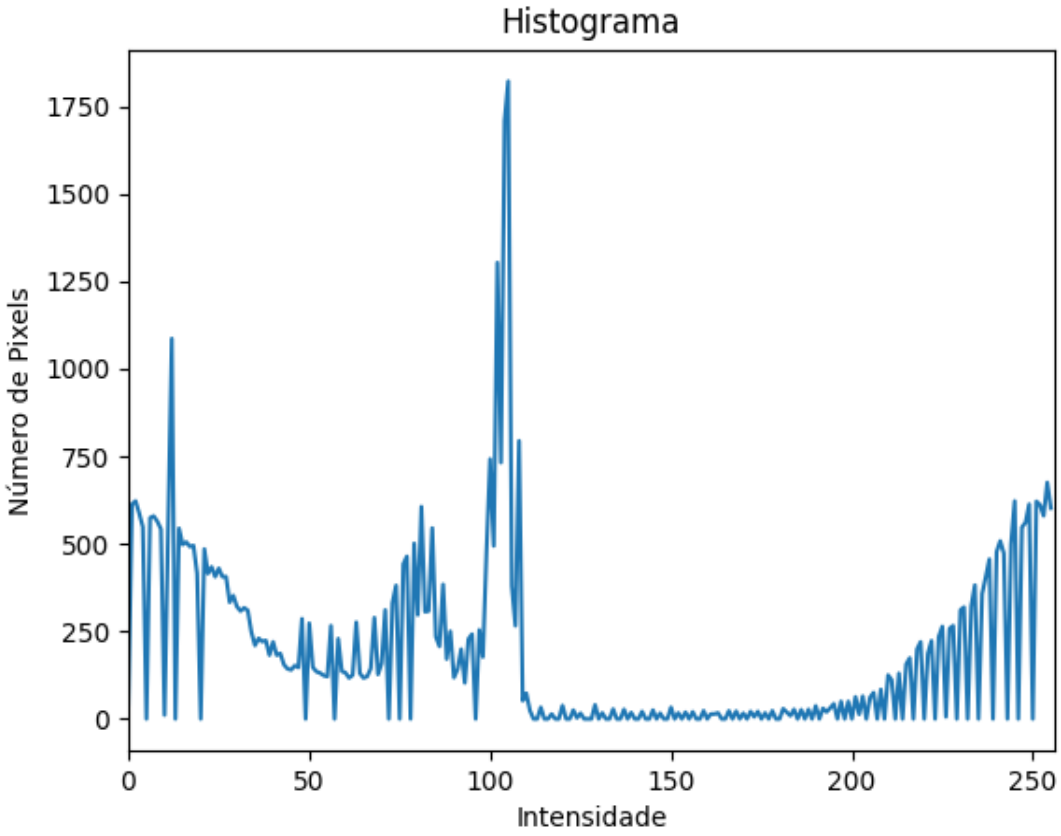


Figura 8. Histograma da Transformação Logaritma(Log(5)) aplicado na imagem 1.

3.3.2. Transformação Exponencial(foi usado expoente 3, esse valor por ser informado na execução do código).

Figura 9. Transformação Exponencial(Expoente 3) aplicado na imagem 1.

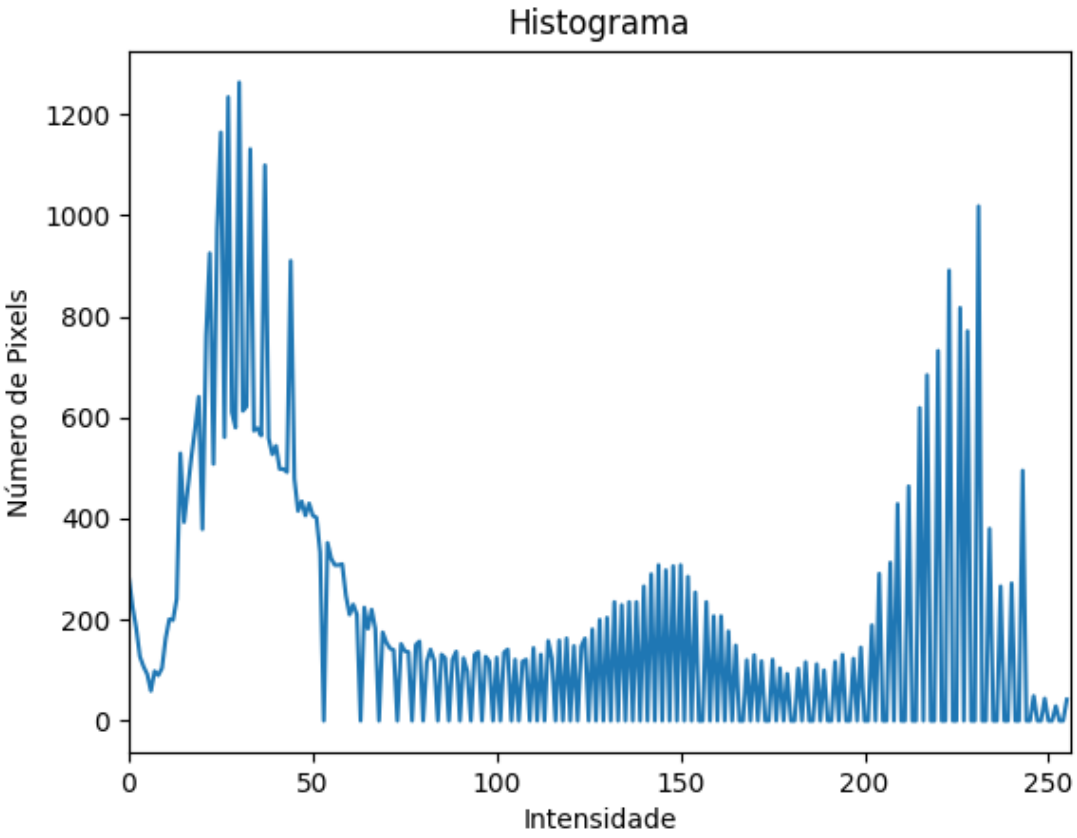


Figura 10. Histograma da Transformação Exponencial(Expoente 3) aplicado na imagem 1.

3.3.3. Transformação Inversa.

Figura 11. Transformação Inversa aplicado na imagem 1.

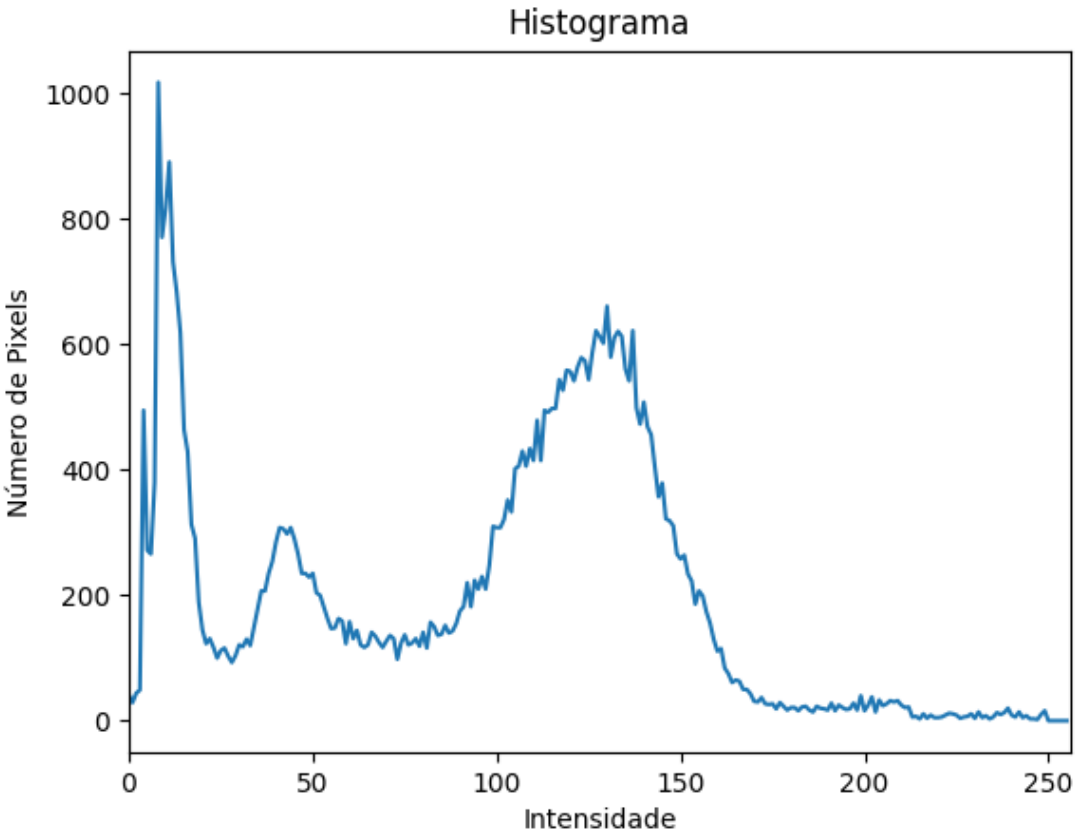


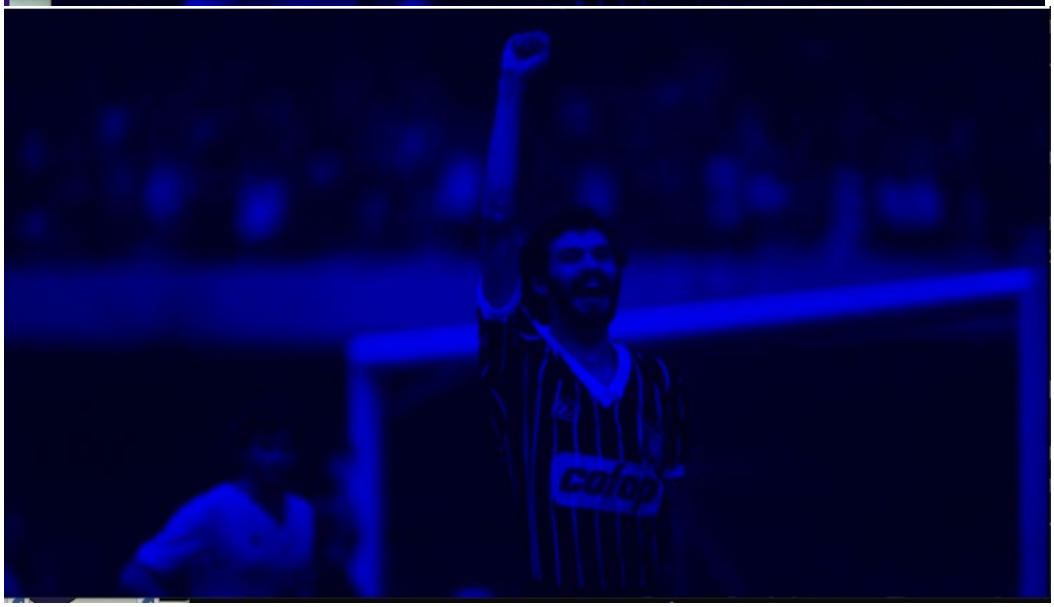
Figura 12. Histograma da Transformação Inversa aplicado na imagem 1.

3.4. Modelos de Cor e Filtragens Espaciais.

3.4.1. Transformação para HSV.

Figura 13. Transformação para HSV aplicado na imagem 2.

Canais de Cor:



3.4.2. Transformação para Preto e Branco.

Figura 14. Transformação para Preto e Branco aplicado na imagem 2.

3.4.3. Filtragem da Mediana

Figura 15. Filtragem da Mediana aplicado na imagem 2.

3.4.4. Filtragem da Média

Figura 16. Filtragem da Média aplicado na imagem 2.

3.4.5 Filtragem da Média Ponderada

Figura 17. Filtragem da Média Ponderada aplicado na imagem 2.

Referências

Log Transformation of an image using python and openCV. Disponível em: <https://www.geeksforgeeks.org/log-transformation-of-an-image-using-python-and-opencv/>. Acesso em: 20 de abril de 2021.

Python intensity transformation operations on images. Disponível em:<https://www.geeksforgeeks.org/python-intensity-transformation-operations-on-images/>. Acesso em: 20 de abril de 2021.

Introdução a Visão Computacional com Python e OpenCV. Disponível em: <https://professor.luzerna.ifc.edu.br/ricardo-antonello/wp-content/uploads/sites 8/2017/02/Livro-Introdu%C3%A7%C3%A3o-a-Vis%C3%A3o-Computacional-com-Python-e-OpenCV-1.pdf>. Acesso em: 01 de maio de 2021.