Uso de paletas e pseudocores

Felipe Diniz Tomás - RA:110752

I. Introdução

Originalmente uma imagem preto-e-branca pode conter detalhes que muitas vezes passam despercebidos. O contraste que a cor proporciana em sua variedade de tons torna mais simples a percepção de detalhes. A técnica pela qual uma imagem monocromática é traduzida em cores é denominada pseudo-cor, isso é feito através do mapeamento em cada um dos níveis de cinza de uma imagem em preto e branco em uma cor atribuída, ou seja, os canais responsáveis pelas cores recebem uma atribuição de paletas de cores arbitrárias, modificando a forma como a imagem é composta. Essa imagem colorida, quando exibida, pode facilitar a identificação de certas características para o observador, além de auxiliar no estudo de como a imagem digital é elaborada.

Os mapeamentos são computacionalmente simples e rápidos. Isso torna a pseudo-cor uma técnica atraente para uso em sistemas de processamento de imagem digital projetados para uso no modo interativo. Vários mapas de cores podem fornecer efeitos de aprimoramento de contraste, efeitos de contorno ou mapeamento de nível de cinza (representando áreas de um determinado nível de cinza) [1].

Quando tratamos do modelo de cor RGB, sendo um sistema aditivo de cores, cada cor é composta por seus componentes espectrais primários: vermelho, azul e verde. A aplicação da paleta modificará cada espectro primário variando sua escala, a cor de um pixel é criada ao ser determinada pela soma de seu RGB, criando uma cor autêntica para determinado tipo de lugar.

II. Metodologia

As execuções realizadas foram feitas no seguinte ambiente computacional:

- A IDE utilizada foi o Spyder 5.1.5[1]
- Sistema operacional Windowns 10.
- Python versão 3.10[2]
- Numpy versão 1.21.3 [3].
- Matplotlib versão 3.4.3 [4].
- OpenCv versão 4.5.4-dev [5].

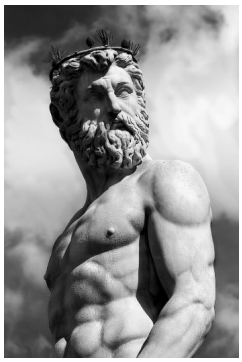
A. Casos de testes

Para testar a aplicação de paletas, foram utilizados 4 imagens em escala de cinza: Gioconda.jpg, Lady-with-an-Ermine.jpg, Sculpture.jpg e Toucan.png.

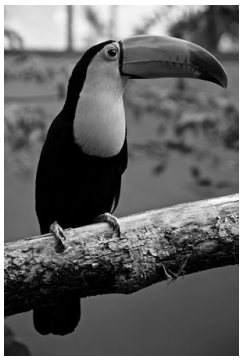
[1]https://www.spyder-ide.org/
[2]https://www.python.org/
[3]https://numpy.org/
[4]https://matplotlib.org/
[5]https://opencv.org/

(a) Gioconda.jpg

(b) Lady-with-an-Ermine.jpg



(c) Sculpture.jpg



(d) Toucan.png

Fig. 1: Casos de testes

III. Código-fonte

O código-fonte, mesmo que simples, foi aplicado principalmente os conceitos de indexação vistos em salas de aula através dos exemplos práticos, além de um estudo do funcionamento de cada ferramenta.

```
def select_img():
    print("=========================================")
    print("Digite o número correspondente a imagem que deseja carregar:\n")
    filenames = next(walk(os.path.dirname(__file__) + "/"), (None, None, []))[2]
    files = [ file for file in filenames if file.endswith((".jpg", ".png", ".tiff")) ]

    if len(files) == 0:
        print("Não existe arquivos de imagem no diretório raiz.")
        sys.exit()

    for i in range(0, len(files)):
        print('{{}} {}'.format(i, files[i]))
    index = input()

    while not index.isdigit() or int(index) >= len(files) or int(index) < 0:
        print("\nEsta opção não existe, digite novamente.")
        index = input()

    imagem = cv2.imread(os.path.dirname(__file__) + "/" + files[int(index)], cv2.IMREAD_GRAYSCALE

    print("\nQual paletas de cores deseja utilizar:")
    paleta = input()

    while paleta not in plt.colormaps():
        print("\nEsta paleta não existe.\n"
              "caso queira visualizar as paletas disponíveis digite 1, \n"
              "caso contrário digite o nome da paleta.")
        paleta = input()

        if (paleta == "1"):
            print("Paletas disponiveis: \n")
            print(plt.colormaps())
            print("\nQual paletas de cores deseaja utilizar:")
            paleta = input()


    return imagem, paleta, files[int(index)]
```

Fig. 2: Função Seleção de imagem.

Na figura 2, podemos ver a função responsável por ler as imagens disponíveis no diretório raiz ao código-fonte e selecionar qual será aberta. Além disso nela é escolhido a paleta de cores pelo usuário.

```
51    # Aplica a paleta de cor na imagem
52
53    def set_palette (original_img, paleta):
54
55        paleta = paleta(np.arange(0, 256)) * 255
56
57        tr = paleta[::, 2]
58        tg = paleta[::, 1]
59        tb = paleta[::, 0]
60
61        canais = [np.uint8(tr[original_img]), np.uint8(tg[original_img]),
62                  np.uint8(tb[original_img])]
63
64        img_colorized = np.dstack(canais)
65
66        return img_colorized
```

Fig. 3: Função que aplica a paleta.

Já a figura 3 mostar a função que aplicará a paleta nos canais RGB da imagem em escala de cinza, combinando cada canal através da função do numpy dstack.

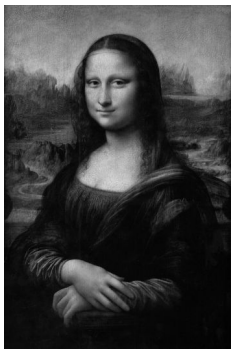
Por fim, a main será responsável por mostrar e salvar as imagens com a biblioteca Opencv, como pode ser visto na figura 4.

```
def select_img():
    print("=======================================")
    print("Digite o número correspondente a imagem que deseja carregar:\n")
    filenames = next(walk(os.path.dirname(__file__) + "/"), (None, None, []))[2]
    files = [ file for file in filenames if file.endswith((".jpg", ".png", ".tiff")) ]

    if len(files) == 0:
        print("Não existe arquivos de imagem no diretório raiz.")
        sys.exit()

    for i in range(0, len(files)):
        print('{{}} {}'.format(i, files[i]))
    index = input()

    while not index.isdigit() or int(index) >= len(files) or int(index) < 0:
        print("\nEsta opção não existe, digite novamente.")
        index = input()

    imagem = cv2.imread(os.path.dirname(__file__) + "/" + files[int(index)], cv2.IMREAD_GRAYSCALE

    print("\nQual paletas de cores deseja utilizar:")
    paleta = input()

    while paleta not in plt.colormaps():
        print("\nEsta paleta não existe.\n"
              "Caso queira visualizar as paletas disponíveis digite 1, \n"
              "caso contrário digite o nome da paleta.")
        paleta = input()

        if (paleta == "1"):
            print("Paletas disponiveis: \n")
            print(plt.colormaps())
            print("\nQual paletas de cores deseaja utilizar:")
            paleta = input()


    return imagem, paleta, files[int(index)]
```

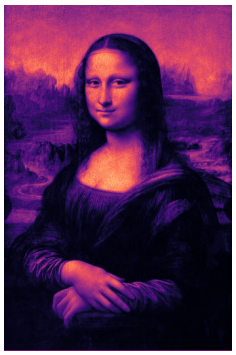
Fig. 4: Função Main.

IV. Imagens resultantes

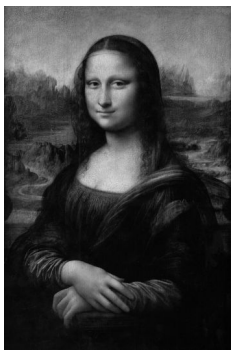
Os resultados foram gerados utilizando diferentes paletas de cores, as figuras a seguir mostram essa experimentação com as paletas da biblioteca matplotlib.



(a) Gioconda.jpg em escala de cinza.



(b) Gioconda.jpg com paleta magma.



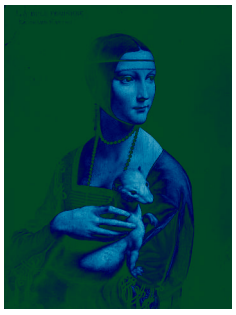
(c) Gioconda.jpg em escala de cinza.

(d) Gioconda.jpg com paleta twilight.

(e) ady-with-an-Ermine.jpg em escala de cinza.

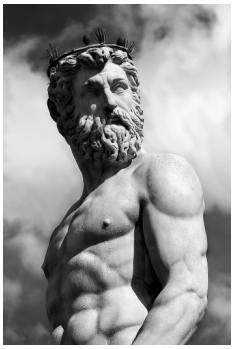
(f) ady-with-an-Ermine.jpg com paleta jet.

(g) ady-with-an-Ermine.jpg em escala de cinza.



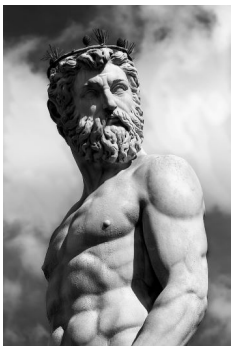
(h) ady-with-an-Ermine.jpg com paleta ocean.

Fig. 5: Casos de testes

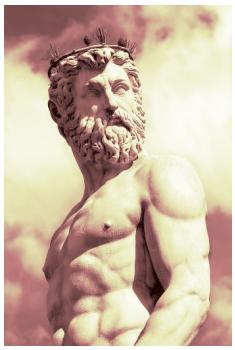


(a) Sculpture.jpg em escala de cinza.

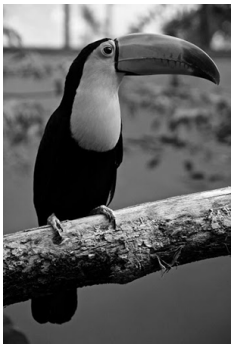
(b) Sculpture.jpg com paleta inferno.



(c) Sculpture.jpg em escala de cinza.

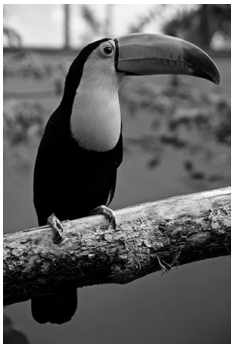


(d) Sculpture.jpg com paleta pink.



(e) Toucan.png em escala de cinza.

(f) Toucan.png com paleta turbo.

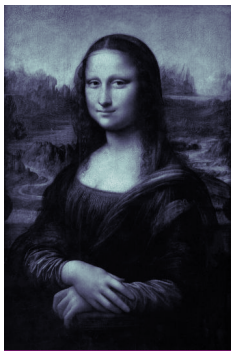


(g) Toucan.png em escala de cinza.

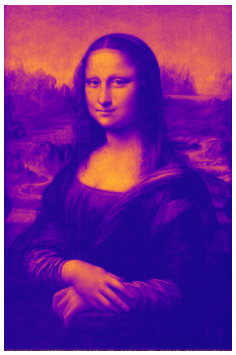
(h) Toucan.png com paleta rainbow.

Fig. 6: Casos de testes

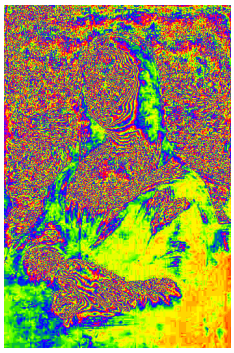
Para fins de ilustração e comparação, também foi escolhido diversas paletas aplicadas para a mesma imagem Gioconda.jpg.



(a) Gioconda.jpg com paleta bone



(b) Gioconda.jpg com paleta plasma



(c) Gioconda.jpg com paleta prims.

(d) Gioconda.jpg com paleta cool.

Fig. 7: Diferentes paletas aplicadas a Gioconda.jpg

REFERENCES

[1] C. H. Radewan, “Digital Image Processing With Pseudo-Color,” in *Acquisition and Analysis of Pictorial Data* (G. A. Michael, ed.), vol. 0048, pp. 50 – 56, International Society for Optics and Photonics, SPIE, 1975.