Relatório Trabalho 2 - CUBO RGB

Renato Alberto dos Santos[1], **Rodrigo Norio Tsuhako**[2]

[1]Departamento de Informática - Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

ra96565@uem.br, ra95376@uem.br

Resumo. *O olho humano tem a capacidade de discernir milhares de cores diferentes, este fenômeno faz com que um indivíduo seja capaz de reconhecer padrões mais facilmente em imagens digitais coloridas do que em imagens digitais em tons de cinza. Neste contexto, surge o modelo RGB afim de padronizar a especificação de cores. O conceito do modelo RGB utiliza coordenadas tridimensionais em um cubo, em que cada vértice representa uma cor. O objetivo do presente trabalho é desenvolver o conceito do cubo RGB e analisar as "fatias"geradas. Os resultados obtidos mostram que o programa desenvolvido obteve "fatias"que condizem com o modelo RGB.*

Palavras-chave*: processamento digital de imagens, modelo RGB, cores.*

1. Introdução

O processamento digital de imagens é certamente uma área que vem crescendo muito por possuir aplicações em diversas disciplinas. Uma imagem digital é caracterizada por ter coordenadas espacias finitas e discretas, processar essa imagem significa aplicar técnicas, por meio de um computador, afim de manipular a imagem digital de entrada.

Neste contexto é possível identificar algumas características em relação as cores das imagens digitais. Existem imagens digitais coloridas, em tonalidade de cinza e binárias (preto e branco). As imagens digitais coloridas tem papel fundamental em reconhecimento de padrões, tornando um poderoso descritor das propriedades de um objeto.

Com o objetivo de padronizar a especificação de cores, surgiram diversos modelos. O modelo de cor RGB (Red, Green, Blue) se baseia em um sistema de coordenadas tridimensionais, em que as cores vermelho, verde e azul são primárias e as cores ciano, magenta e amarelo são secundárias. O preto é caracterizado pela ausência de cor e o branco é a presença das três cores primárias.

O presente trabalho tem como objetivo analisar "fatias"do cubo RGB, geradas por meio do programa desenvolvido no presente trabalho.

2. Modelos de Cores

O processamento digital de imagens coloridas surgiu por meio de dois principais fatores: reconhecimento de padrões e visão humana. Em uma imagem digital colorida cada objeto pode possuir cores diferentes, desta forma o reconhecimento de padrões de cores pode simplificar a identificação dos objetos na imagem. A capacidade do olho humano para reconhecer tonalidade de cinza é muito baixa, portanto, analisar imagens coloridas é a opção mais viável aos humanos [Filho and Neto 1999].

O principal objetivo dos modelos de cores é padronizar a especificação das cores, em um modelo amplamente aceito. Os modelos mais utilizados são: RGB (Red, Green, Blue), CMY (Cyan, Magenta, Yellow), CMYK (Cyan, Magenta, Yellow, Black) e HSI (Hue, Saturation, Intensity) [Gonzalez and Woods 2010].

3. Modelo RGB

O modelo RGB é considerado um sistema aditivo de cores, ou seja, a presença das suas cores primárias gera a cor branca. No caso do RGB a presença com máxima intensidade das cores vermelho, verde e azul gera o branco, e a ausência delas gera o preto. Esse modelo é bastante utilizado em equipamentos eletrônicos como televisões, computadores e câmeras digitais [Corrêa et al. 2014].

O RGB é um modelo de cores com base nas dimensões de um cubo, em que possui coordenadas tridimensionais (x,y,z) representadas pelas cores (vermelho, verde, azul). Na representação do cubo, três vértices são cores primárias e três vértices são cores secundárias. O vértice de origem (0,0,0) corresponde a cor preta, e o vértice oposto (1,1,1) é a cor branca. Na linha diagonal entre os vértices preto e branco, estão localizados as tonalidades de cinza [Filho and Neto 1999].

4. Metodologia

Para a execução do trabalho foi utilizado como base o conceito de coordenadas tridimensionais de um cubo RGB que serão apresentados a seguir.

4.1. Hardware e Software

O trabalho foi desenvolvido e testado em uma máquina com as seguintes especificações:

- Processador Intel Core I5-4210, com frequência de 1,7 GHz;
- 8G de memória RAM;
- HD de 1TB;
- Sistema operacional Windows 10 versão 1909.
- Editor de texto Visual Studio Code
- Linguagem Python 3.8.5

4.2. Condução dos testes

Como citado anteriormente, o desenvolvimento do trabalho tomou como base o cubo RGB, onde suas coordenadas tridimensionais seguem o padrão (vermelho, verde, azul).

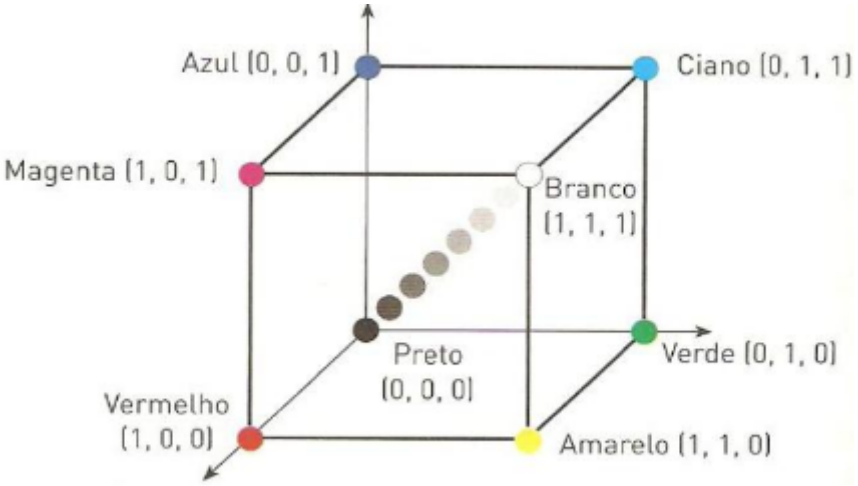

Figura 1. Cubo RGB

O presente trabalho teve como base o cubo RGB apresentado na figura 1. É possível observar que cada vértice representa uma cor, e por questão de normalização os valores possuem intervalo entre 0 e 1. É importante ressaltar que o presente trabalho não normalizou os resultados, portanto, a intensidade das coordenadas estão no intervalo de 0 a 255.

Para padronizar o cubo, cada face foi numerada da seguinte forma:

Número	**Face**
1	Direita
2	Esquerda
3	Frente
4	Trás
5	Baixo
6	Cima

Tabela 1. Representação das Faces

Um dos parâmetros de entrada do programa que o usuário escolhe é a face do cubo, representada pela tabela 1.

Além disso, outro parâmetro que o usuário precisa indicar é a profundidade da ”fatia”referente a face escolhida. O valor da profundidade pertence ao intervalo 0 a 255.

Caso o usuário escolha a face 3 (frente) com profundidade 0. Significa que a ”fatia”é exatamente a face 3, em que seus vértices são representados pelas cores magenta, branco, vermelho e amarelo.

Neste mesmo cenário, se o usuário escolhe profundidade 255, a ”fatia”irá retornar as cores azul, ciano, preto e verde.

Com base na observação anterior, os testes foram conduzidos de forma que foi

possível testar a veracidade das ”fatias”.

5. Resultados

Para cada face foi realizado pelo menos dois testes, afim de confirmar a veracidade dos dados obtidos.

Em relação a face 1:

Figura 2. Direita - 0

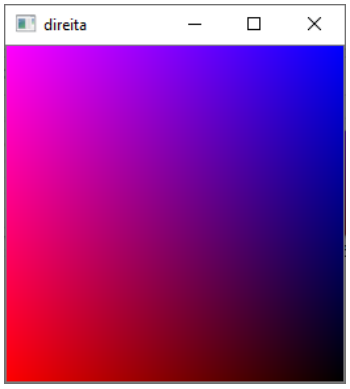

Figura 3. Direita - 255

As figuras 2 e 3 mostram que as ”fatias”obtidas correspondem com o cubo RGB. Em que a figura 2 representa a profundidade 0 e a figura 3 a profundidade 255.

A face 2 resultou nas seguintes ”fatias”:

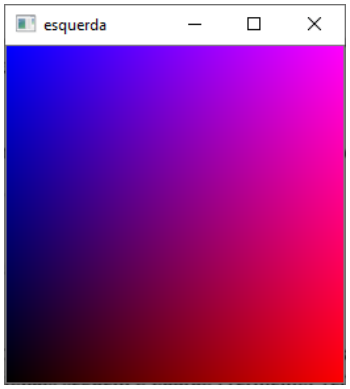

Figura 4. Esquerda - 0

Figura 5. Esquerda - 255

Nas figuras 4 e 5 é possível observar que os resultados são semelhantes aos da face 1, mudando somente o ponto de vista do cubo.

As faces 3 e 4 são retratadas a seguir:

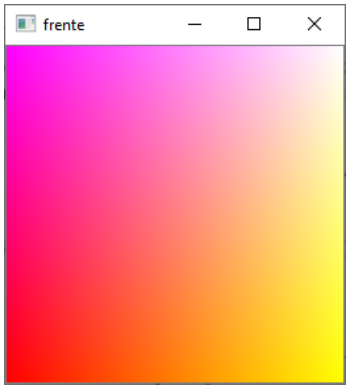

Figura 6. Frente - 0

Figura 7. Frente - 255

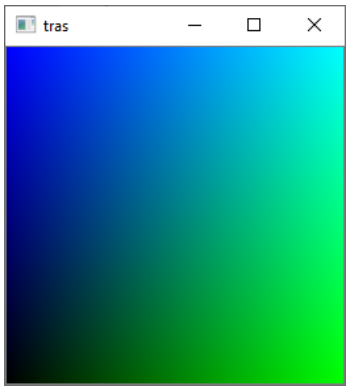

Figura 8. Trás - 0

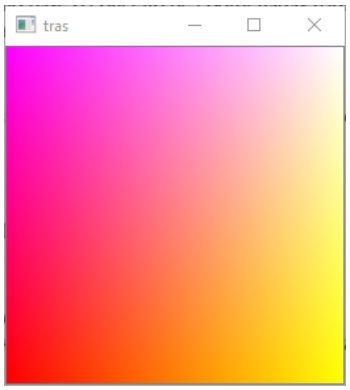

Figura 9. Trás - 255

Da figura 6 até a figura 9 é mostrado a relação de uma face com sua face oposta. É possível observar que por meio da profundidade, uma face pode chegar a sua face oposta usando a profundidade máxima.

As faces 5 e 6 resultaram em:

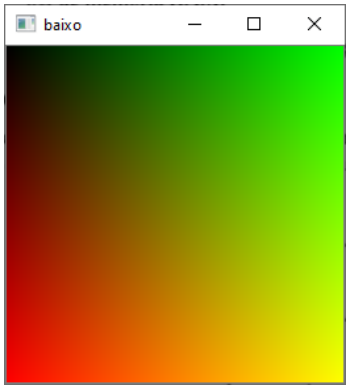

Figura 10. Baixo - 0

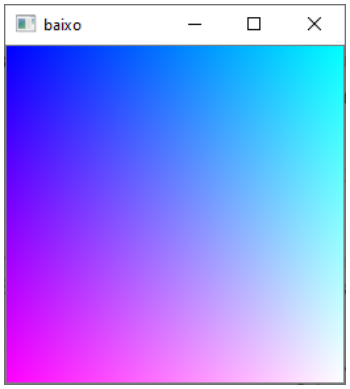

Figura 11. Baixo - 255

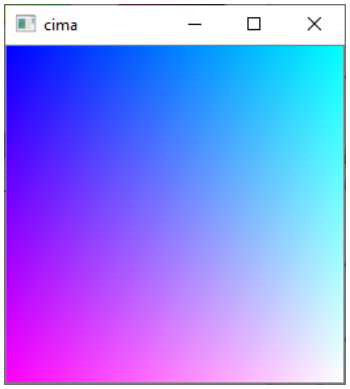

Figura 12. Cima - 0

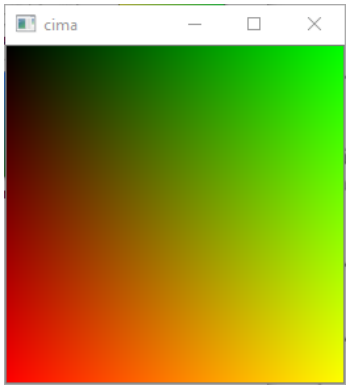

Figura 13. Cima - 255

Da figura 10 até a figura 13 é possível observar que assim como as faces anteriores, a relação entre uma face com sua face oposta, por meio da profundidade, continua.

6. Conclusão

O processamento digital de imagens é uma área que vem crescendo cada vez mais. Dentro desta disciplina, nota-se que o estudo de cores é fundamental para os olhos humanos, já que os humanos conseguem discernir milhares de cores diferentes. Enquanto a capacidade de enxergar tons de cinza é muito baixa. Neste contexto, foi criado o modelo RGB afim de padronizar a especificação de cores.

Desta forma, considera-se que o objetivo deste trabalho foi atendido, uma vez que foi desenvolvido o conceito do cubo RGB e é possível selecionar uma ”fatia”dele.

Com base nos resultados obtidos, é possível concluir que para o conjunto de testes realizados, o programa desenvolvido retornou resultados precisos. Os resultados de todas as 6 faces condizem com o modelo do cubo RGB apresentado no presente trabalho. Além disso, foi possível concluir que uma face com profundidade 255 é igual a sua face oposta com profundidade 0.

7. Implementação

O código a seguir foi desenvolvido em Python 3.8.5 utilizando as bibliotecas numpy e cv2.

```
import numpy as np
from matplotlib import pyplot as plt
import cv2
import time



saida = np.zeros([256, 256,3], dtype=np.uint8)

verde = np.zeros([256, 256], dtype=np.uint8)
verde[::1,::1] = np.linspace(0,255,num=256)
```

```
azul = np.ones([256, 256], dtype=np.uint8)
azul[::1,::1] = np.linspace(0,255,num=256)
azul = azul.T
azul = np.flip(azul)

vermelho = np.zeros([256, 256], dtype=np.uint8)
vermelho[::1,::1] = np.linspace(0,255,num=256)
vermelho= vermelho.T


print("direita = 1, esquerda = 2, frente = 3, tras = 4, baixo = 5, cima

face = input("Escolha a face: ")
fatia = input("Escolha a fatia de 0 a 255: ")
op = int(face)
fatia = int(fatia)


#direita
if op == 1:
    verde[::1,::1] = np.linspace(255-fatia,255-fatia,num=256)
    vermelho = np.zeros([256, 256], dtype=np.uint8)
    vermelho[::1,::1] = np.linspace(255,0,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "direita"

#esquerda
if op == 2:
    verde[::1,::1] = np.linspace(fatia,fatia,num=256)
    vermelho = np.zeros([256, 256], dtype=np.uint8)
    vermelho[::1,::1] = np.linspace(0,255,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "esquerda"

#frente
if op == 3:
    vermelho[::1,::1] = np.linspace(255-fatia,255-fatia,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "frente"

#tras
if op == 4:
```

```
    vermelho[::1,::1] = np.linspace(fatia,fatia,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "tras"

#baixo
if op == 5:
    azul[::1,::1] = np.linspace(fatia,fatia,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "baixo"

#cima
if op == 6:
    azul[::1,::1] = np.linspace(255-fatia,255-fatia,num=256)
    saida[::1,::1] = cv2.merge((azul,verde,vermelho))

    nome = "cima"

cv2.imshow(nome,saida)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Referências

Corrêa, J. H. G., Veja, G. U. S., dos Santos Cunha, N., da Silva, T. G., and Jr, P. D. M. (2014). Um modelo simples e parametrizÁvel para classificaÇÃo de cores no sistema rgb. *IX Congresso Norte Nordeste de Pesquisa e Inovação*.

Filho, O. M. and Neto, H. V. (1999). *Processamento Digital de Imagens*. BrasPort.

Gonzalez, R. and Woods, R. (2010). *Processamento Digital de Imagens*, volume 3. PEARSON.