UNIVERSIDADE PAULISTA

CIÊNCIAS DA COMPUTAÇÃO

Processamento de Imagens
e
Visão Computacional

Prof. César C. Xavier

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

ROTEIRO

Representação de Cores no Espaço

- Introdução
- Cores no Espaço RGB
  - Segmentação Canais de Cores
  - Práticas Python
- Cores no Espaço HSV
  - Segmentação Canais de Cores
  - Práticas Python
- Aplicação do Espaço HSV
  - Práticas Python

Prof. César C. Xavier

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Introdução

- Foi estudado diferentes tipos de formato de imagens.
- Existem diferentes modelos matemáticos de representar as cores em imagens digitais:
  - RGB (*Red, Green* e *Blue*); e
  - HSV (*Hue, Saturation* e *Value)*.
- Compreender os diferentes espaços de cores facilita o desenvolvimento de sistemas voltados para Visão Computacional.
- Desempenho na segmentação de objetos depende do espaço de cor utilizado.

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Introdução

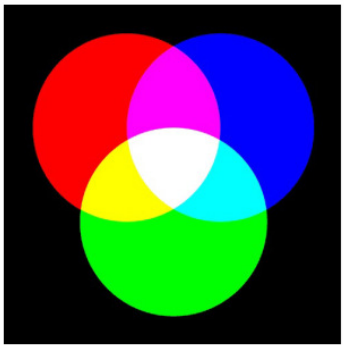
- Imagens binárias e em tons de cinza são mais fáceis de representar.
- Imagens com um único canal:
  - Binárias: 2 bits; e
  - Tons de Cinza: 8 bits.
- Imagens RGB: três canais, um para cada cor de cada pixel da imagem.

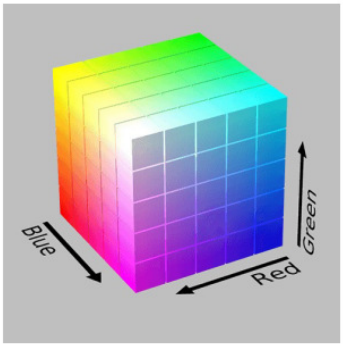
Processamento de Imagens e Visão Computacional

Cores no Espaço RGB

- Largamente utilizado:
  - Armazenamento de cor; e
  - Exibição em monitores.

- Considera as três cores primárias Vermelho (*red* - R), Verde (*green* - G) e Azul (*blue* - B).





Processamento de Imagens e Visão Computacional

Cores no Espaço RGB

- Segmentação Canais RGB

R

G

B

Processamento de Imagens e Visão Computacional

Cores no Espaço RGB

- Segmentação Canais RGB – OpenCV

```
import cv2

#   Carregando   imagem   RGB e   segmentando  canais

imagem = cv2.imread("frutas.png")
azul, verde, vermelho = cv2.split(imagem)

 #  Exibindo imagens  dos  canais   separados

cv2.imshow("Canal R",     vermelho)
cv2.imshow("Canal G",     verde)
cv2.imshow("Canal B",     azul)

#   Salvando imagens  dos  canais   separados

cv2.imwrite("frutas-canal-vermelho.jpeg", vermelho)
cv2.imwrite("frutas-canal-verde.jpeg",    verde)
cv2.imwrite("frutas-canal-azul.jpeg", azul)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


Processamento de Imagens e Visão Computacional

Cores no Espaço RGB

- Segmentação Canais RGB – OpenCV
  - É possível, a partir dos canais RGB, recombiná-los em uma única imagem colorida.

```
import cv2

#   Carregando   imagem   RGB e   segmentando  canais

imagem = cv2.imread("frutas.png")
azul, verde, vermelho = cv2.split(imagem)


# Combinando os três canais em uma única imagem

imagem_combinada = cv2.merge((azul, verde, vermelho))
cv2.imshow("Imagem combinada", imagem_combinada)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Cores no Espaço RGB

- Convertendo imagens RGB p/ Tons Cinza – OpenCV
  - Imagens em tons de cinza permitem maior velocidade de processamento para a obtenção de regiões de interesse, bordas, manchas e junções.

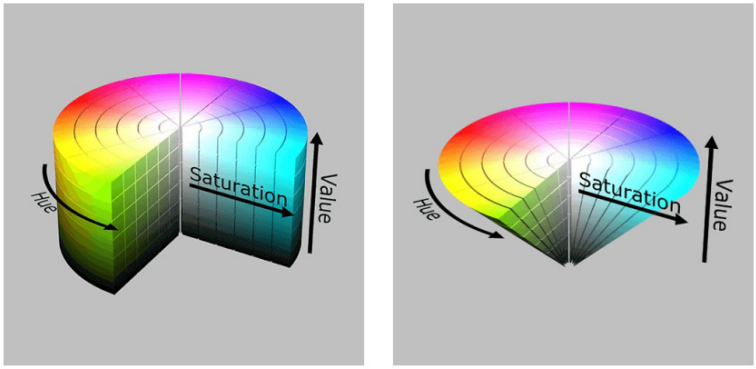
```
import cv2

#   Carregando   imagem   RGB e   segmentando  canais

imagem = cv2.imread("frutas.png")

# Convertendo para tons de cinza

imagem_cinza = cv2.cvtColor(imagem, cv2.COLOR_RGB2GRAY)

cv2.imshow("Imagem Tons de Cinza", imagem_cinza)

cv2.waitKey(0)
cv2.destroyAllWindows()
```


UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Cores no Espaço HSV

- Largamente utilizado:
  - Sistemas de Visão Computacional; e
  - Processamento de Imagens.
- Considera as três canais: a matiz (*hue* – *H*), a saturação (*saturation* – *S*) e o valor (*value* - *V*)[1].



[1]Obs.: ou HSB onde B, corresponde ao brilho/intensidade (*brightness*).

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Cores no Espaço HSV

- Matiz

  - Representa a tonalidade da cor e permite diferenciar o azul do vermelho e do verde.
  - Figura: (b) cores originais; (a) com -80% de matiz; e (c) com +80% de matiz.

(a)

(b)

(c)

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Cores no Espaço HSV

- Saturação

  - Representa a intensidade da cor. Maior a saturação, mais pura é a cor.
  - Figura: (b) cores originais; (a) com -80% de saturação; e (c) com +80% de saturação.

(a)

(b)

(c)

Processamento de Imagens e Visão Computacional

Cores no Espaço HSV

- Valor (ou Intensidade)

  - Representa ao brilho da cor, à luminosidade ou escala de claridade. Maior a intensidade, mais próximo ao branco e, caso contrário, mais próximo do preto.
  - Figura: (b) cores originais; (a) com -80% de luminosidade; e (c) com +80% de luminosidade.

(a) (b) (c)

Processamento de Imagens e Visão Computacional

Segmentação de Cores no Espaço HSV

- Extração dos canais de matiz, saturação e intensidade - OpenCV

```
import cv2

# Carregando imagem RGB
imagem = cv2.imread("frutas.png")

# Convertendo RGB para HSV
imagem_hsv = cv2.cvtColor(imagem, cv2.COLOR_BGR2HSV)

# Ssegmentando canais H, S e V
matiz, saturacao, intensidade = cv2.split(imagem_hsv)

 #Exibindo imagens  dos  canais   separados

cv2.imshow("Canal H",    matiz)
cv2.imshow("Canal S",    saturacao)
cv2.imshow("Canal V", intensidade)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Segmentação de Cores no Espaço HSV

- Recombinando os canais de matiz, saturação e intensidade - OpenCV

```
import cv2

# Carregando imagem RGB
imagem = cv2.imread("frutas.png")
imagem_hsv = cv2.cvtColor(imagem, cv2.COLOR_BGR2HSV)

matiz, saturacao, intensidade = cv2.split(imagem_hsv)

cv2.imshow("Canal H",   matiz)
cv2.imshow("Canal S",   saturacao)
cv2.imshow("Canal V", intensidade)

imagem_recomb = cv2.merge((matiz, saturacao, intensidade))

cv2.imshow("Imagem Recombinada", imagem_recomb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


UNIP UNIVERSIDADE PAULISTA
Processamento de Imagens e Visão Computacional

Segmentação de Cores no Espaço HSV

- Modificando os canais - OpenCV

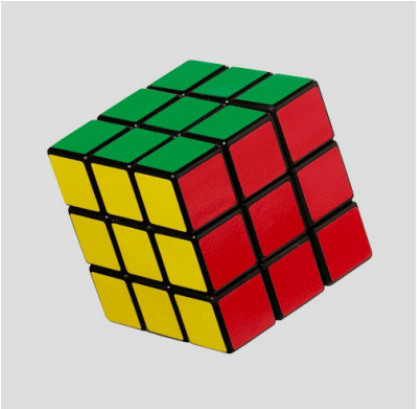
```
import cv2

CTE = 10

# Carregando imagem RGB
imagem = cv2.imread("frutas.png")

# Convertendo RGB para HSV
imagem_hsv = cv2.cvtColor(imagem, cv2.COLOR_BGR2HSV)

# Segmentando canais H, S e V
matiz, saturacao, intensidade = cv2.split(imagem_hsv)

saturacao_plus = saturacao + CTE
saturacao_minus = saturacao - CTE

imagem_plus =cv2.merge((matiz, saturacao_plus, intensidade))
imagem_minus=cv2.merge((matiz, saturacao_minus, intensidade))

imagem_plus_rgb = cv2.cvtColor(imagem_plus, cv2.COLOR_HSV2BGR)
imagem_minus_rgb = cv2.cvtColor(imagem_minus, cv2.COLOR_HSV2BGR)

# Exibindo imagem alterada
cv2.imshow("Imagem Plus", imagem_plus_rgb)
cv2.imshow("Imagem Minus", imagem_minus_rgb)
cv2.imshow("Imagem Original", imagem)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

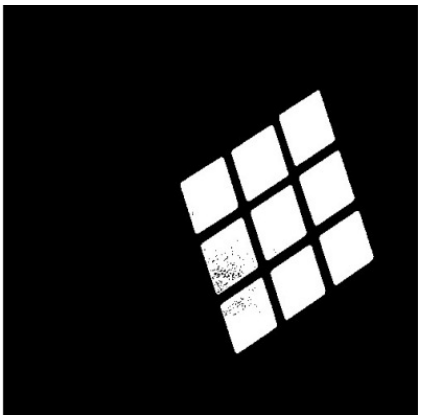
UNIP UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Cores no Espaço HSV

- Aplicações
  - Segmentação de objetos coloridos para:
    - Contagem;
    - Rastreamento





UNIP UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Segmentação de Cores no Espaço HSV

- Segmentação Cores - OpenCV

```
import cv2
import numpy as np

imagemRGB = cv2.imread("cubo_magico.jpg")
imagemHSV = cv2.cvtColor(imagemRGB, cv2.COLOR_BGR2HSV)

#vermelho
#tomClaro= np.array([80, 110, 110])
#tomEscuro = np.array([200, 255, 255])

# amarelo
#tomClaro= np.array([10, 100, 100])
#tomEscuro = np.array([50, 255, 255])

# verde
tomClaro = np.array([40, 100, 100])
tomEscuro = np.array([80, 255, 255])

imgSegmentada=cv2.inRange(imagemHSV, tomClaro, tomEscuro)
cv2.imshow("Segmentada", imgSegmentada)
cv2.imshow("Original", imagemRGB)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Prof. César C. Xavier

UNIVERSIDADE PAULISTA

Processamento de Imagens e Visão Computacional

Segmentação de Cores no Espaço HSV

- Exercício #2: faça programa em Python, utilizando OpenCV, que modifique o canal de tons de cinza de uma imagem colorida.

Prof. César C. Xavier