Computação Gráfica

Processamento de Imagens
(Aula 2)

Leonardo Medeiros

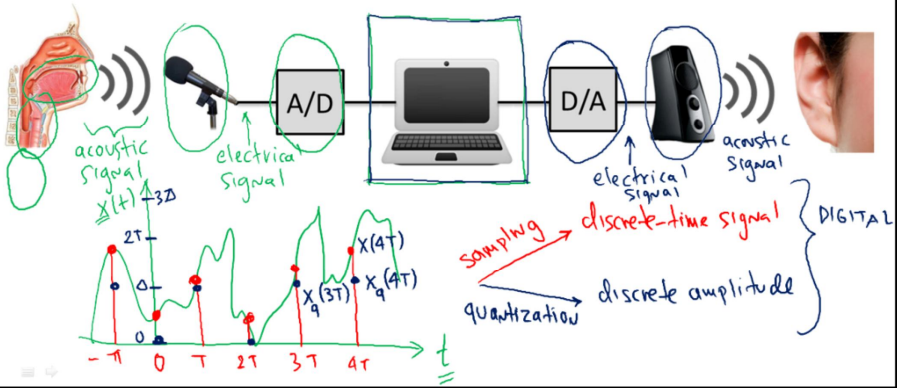
Instituto Federal de Alagoas

22 de Abril de 2017

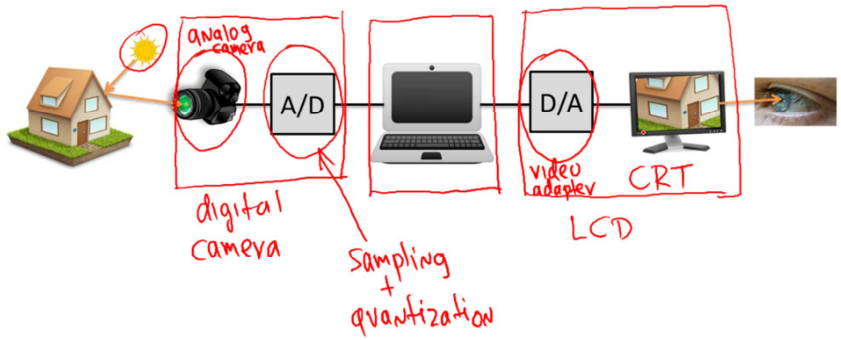
Roteiro

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Analog vs Digital Signals



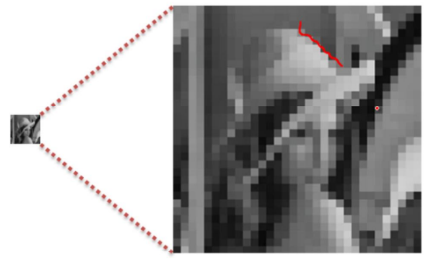
Analog vs Digital Signals



Sampling

$2^8 = 256$

pixel ← picture element

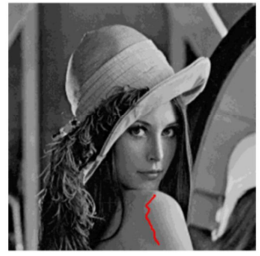


zero-order hold

Quantization

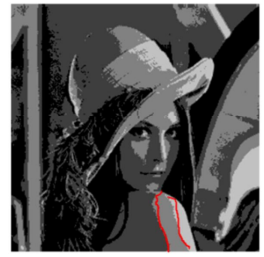
8 bits per pixel

$2^8 = 256$



4 bits per pixel

$2^4 = 16$



2 bits per pixel

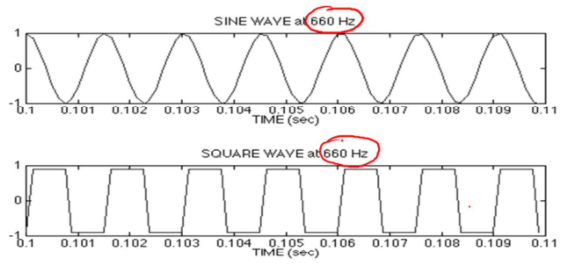
$2^2 = 4$

Images and Videos

- 1D: tones, speech, audio, biomedical, remote sensing, etc
- 2D: text, grayscale, color, multispectral, hyperspectral images, etc
- 3D: video, 3D volume, etc
- MD: video of a volume, etc

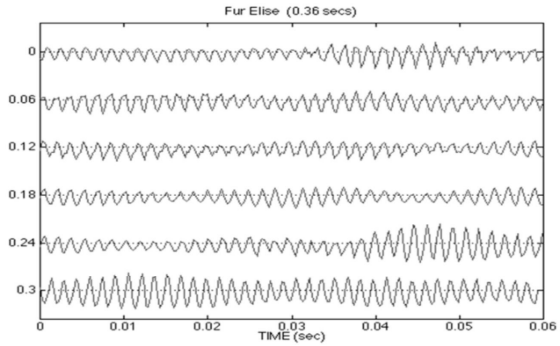
$s(t)$ $s(n)$ $s(x,y)$ $s(x,y,t)$ $s(x,y,z)$ $s(x,y,z,t)$

1D Signals: Tones



J. H. McClellan, R. W. Schafer, and M. A. Yoder, "DSP First: A Multimedia Approach," Prentice Hall, 1998.

1D Signal: Piano Piece



J. H. McClellan, R. W. Schafer, and M. A. Yoder, "DSP First: A Multimedia Approach," Prentice Hall, 1998.

Images

This is a Coursera Course on the "Fundamentals of Digital Image and Video Processing"

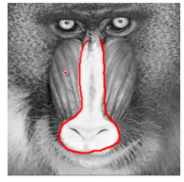
1 bit per pixel

8 bits per pixel

24 bits per pixel

RGB Color Image

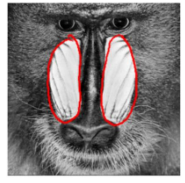
R



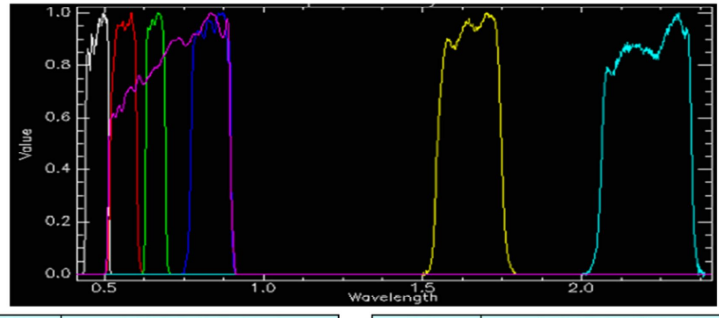
$2^8 = 256$

G

B



LANDSAT ETM+ Spectral

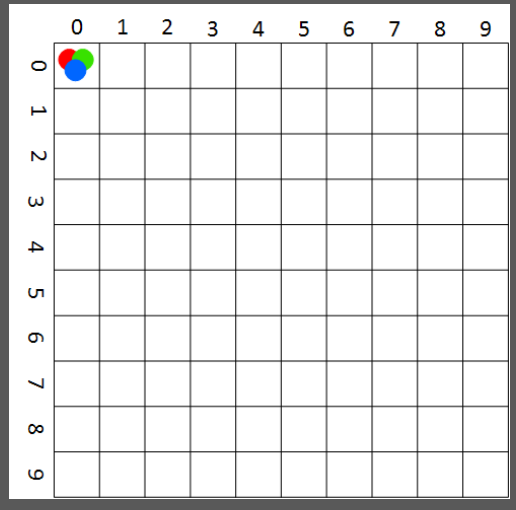


Color	LANDSAT ETM+ band	
White	1 (0.45 µm to 0.515 µm)	B
Red	2 (0.525 µm to 0.605 µm)	G
Green	3 (0.63 µm to 0.69 µm)	R
Blue	4 (0.75 µm to 0.9 µm)	

Color	LANDSAT ETM+ band	
Yellow	5 (1.55 µm to 1.75 µm)	
Not shown	6 (10.4 µm to 12.5 µm)	→ 60m
Cyan	7 (2.08 µm to 2.35 µm)	
Magenta	Pan (0.51 µm to 0.9 µm)	→ 15m

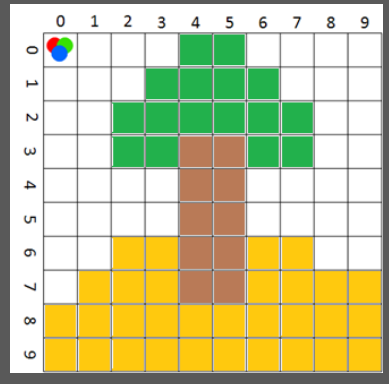
How do Computers store images?

- OpenCV uses RBG color space by default.



Each pixel coordinate (x, y) contains 3 values ranging for intensities of 0 to 255 (8-bit).

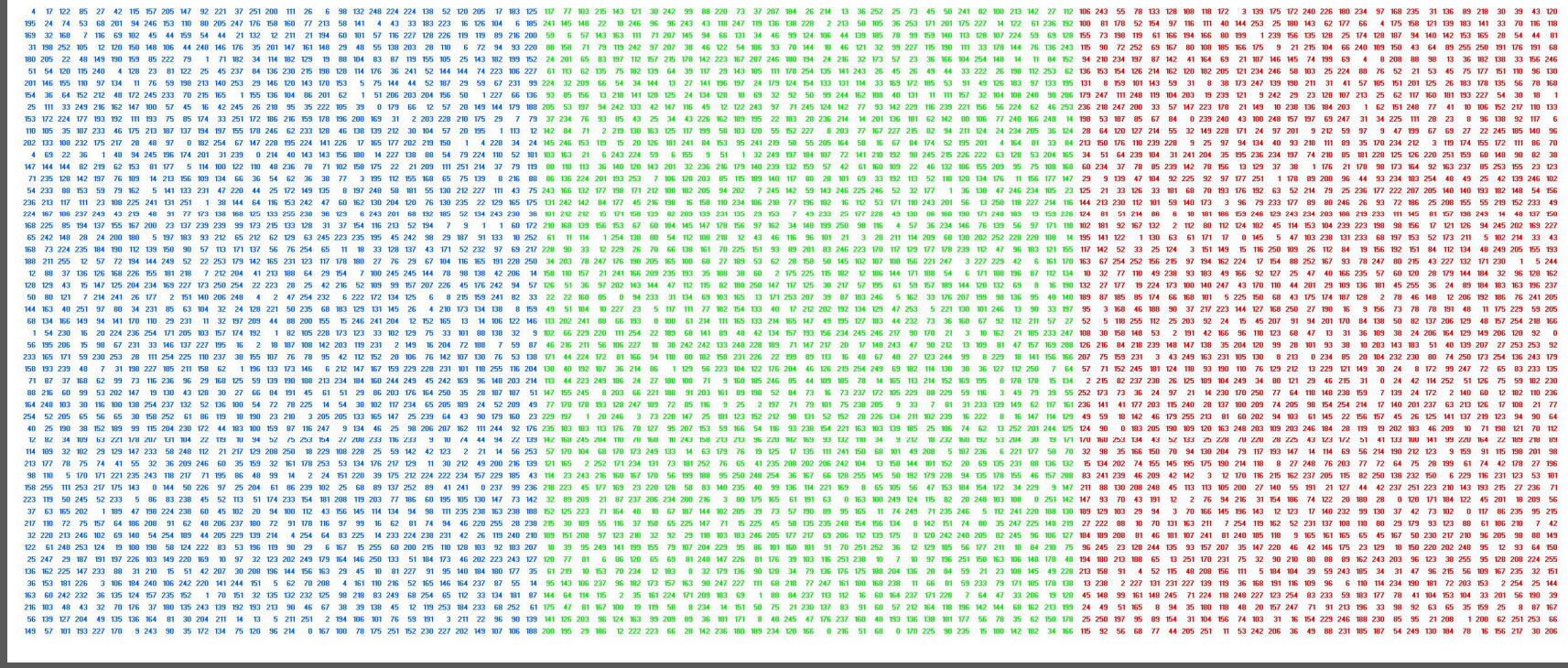
- Red
- Green
- Blue



Mixing different intensities of each color gives us the full color spectrum, example Yellow:

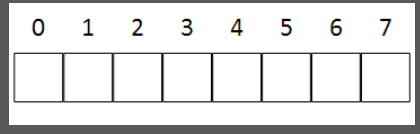
- Red – 255
- Green – 255
- Blue - 0

How does it look when stored on a computer?

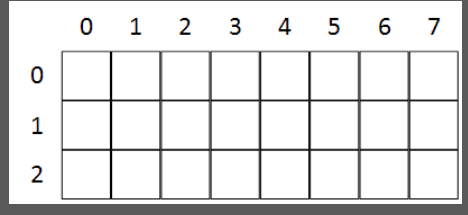


Images are stored in multi-dimensional arrays

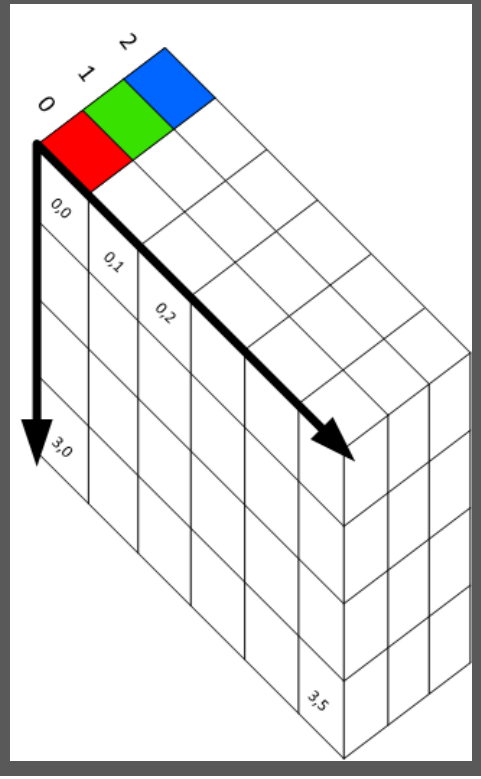
- Think of an array as table. A 1-dimensional arrays looks like this:



- A 2-dimensional array looks like this:

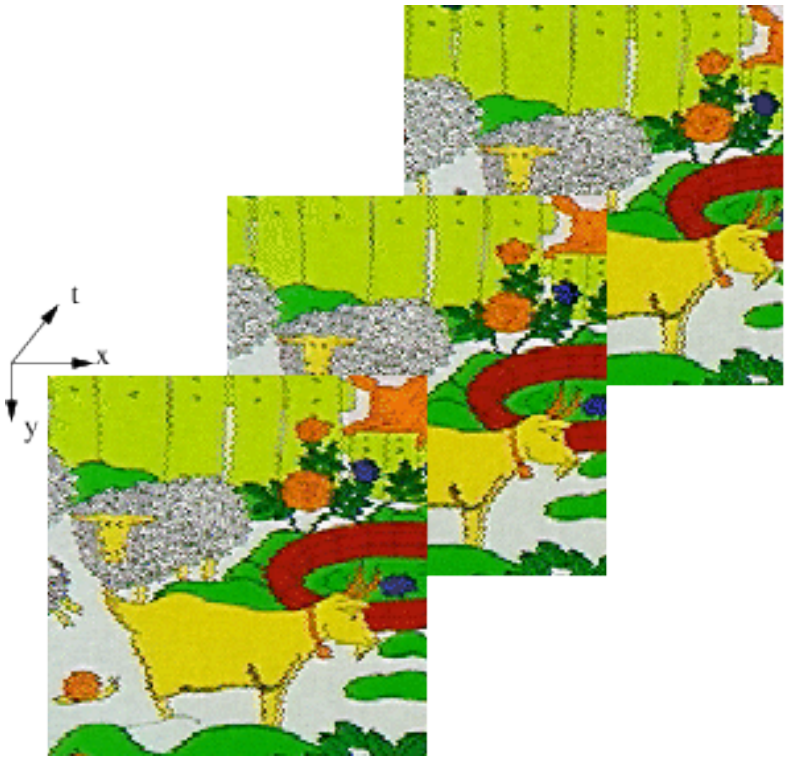


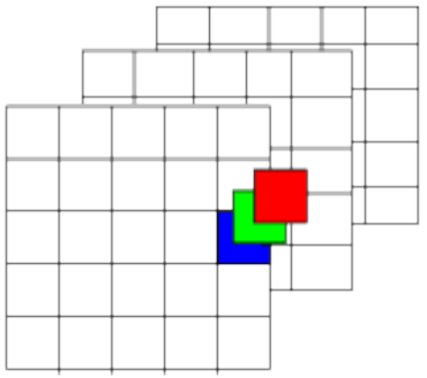
- 3-dimensional array



Formação de uma imagem colorida

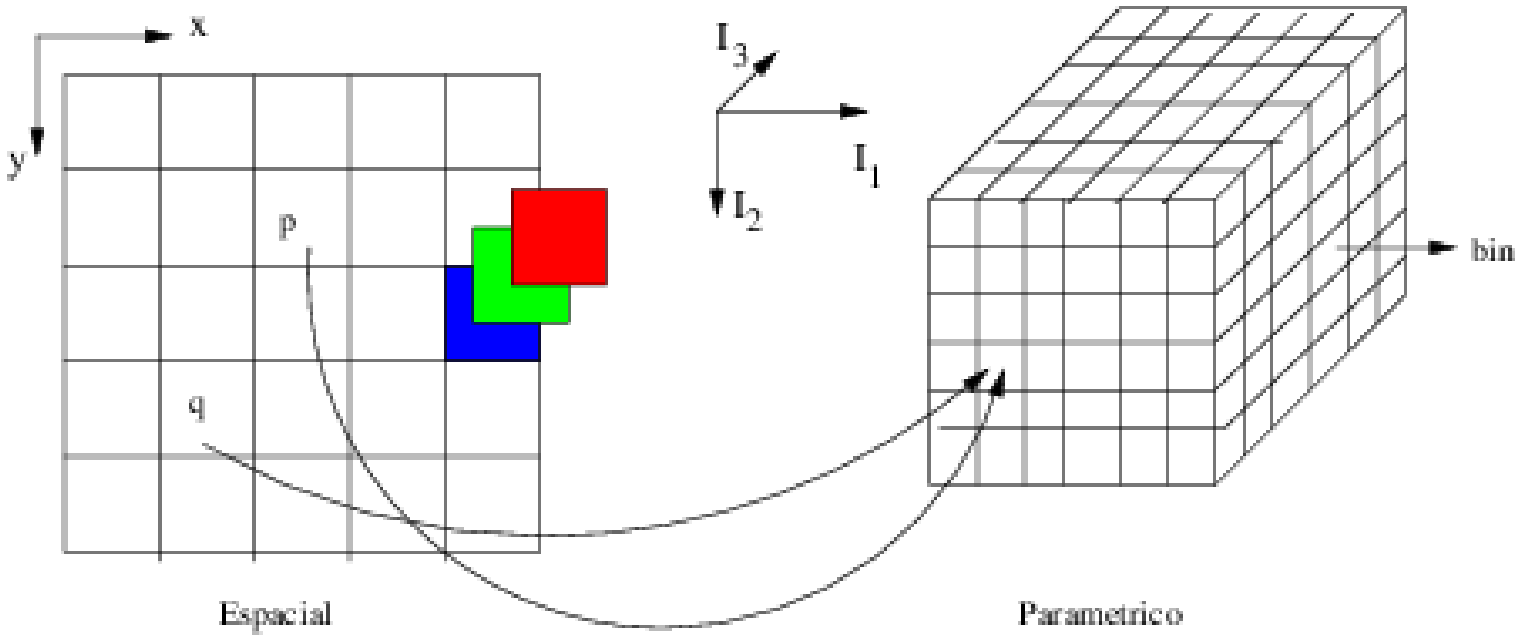
No caso de imagens coloridas (fotografia, vídeo), cada pixel p terá associado um vetor $\vec{I}(p) = (I_1(p), I_2(p), I_3(p))$ com as medidas de reflexão de luz nos comprimentos de onda do vermelho, verde, e azul, respectivamente. A amostragem da função $f(x, y, t)$ ao longo do tempo gera spels $p = (x, y, t)$ (*space elements*).





Espaços imagem e paramétrico

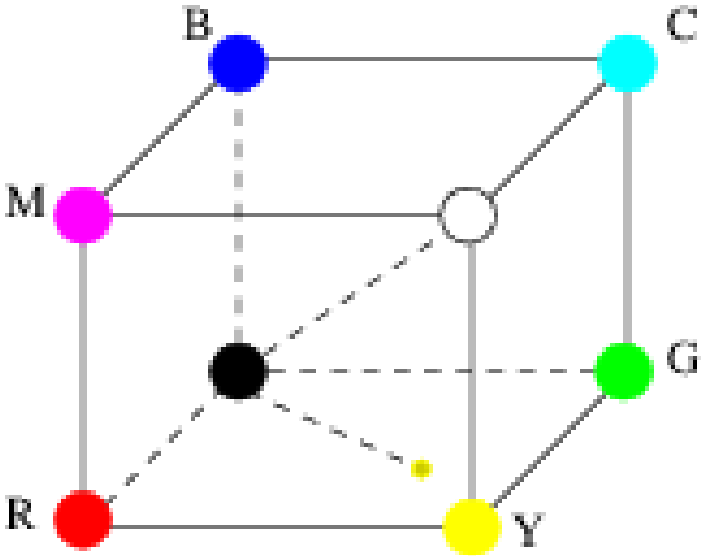
Os três componentes de cor de um pixel são vistos como características (atributos, parâmetros), mapeando cada pixel em um ponto do espaço paramétrico correspondente.



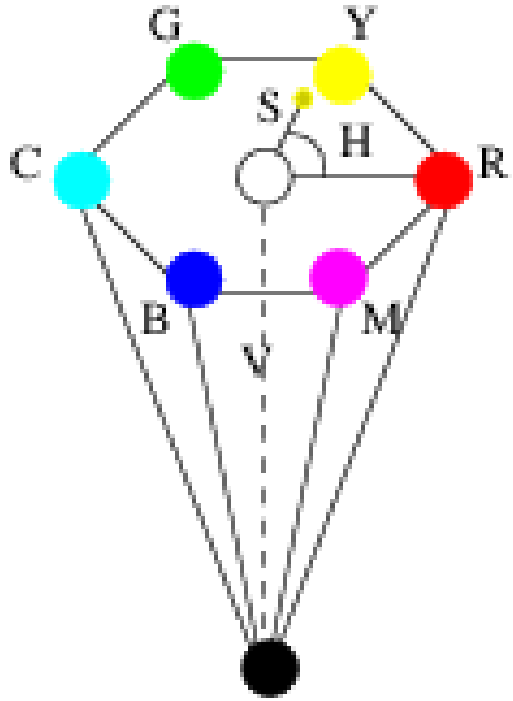
* Para b=24bits (8 bits por componente de cor), o número de bins (cores) é $2^{24} - 1$.

Espaços de cor

- Uma imagem colorida pode ser armazenada em diferentes espaços de cor (RGB, YCbCr, CMY, HSV, Lab,etc.).
- A conversão da imagem de um espaço de cor para outro é uma operação matemática pixel a pixel (e.g., multiplicação matricial envolvendo $\vec{I}(p)$).



RGB and CMY

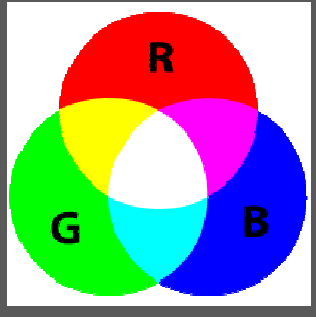


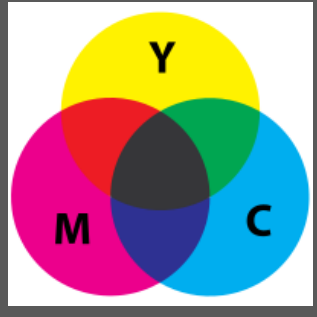
HSV

Color Spaces

Ever heard of the terms RGB, HSV or CMYK?

These are color spaces, which is simply a way to represent color.



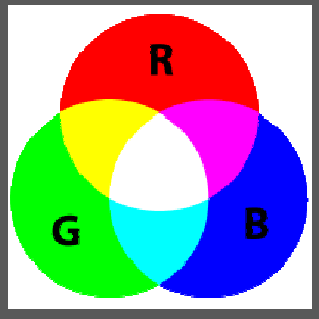


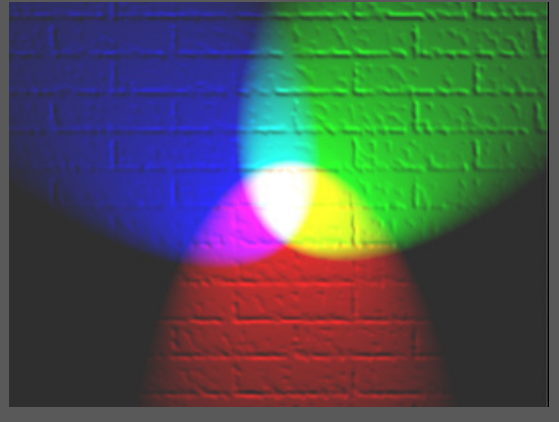
RGB….wait BGR Color Space

OpenCV's default color space is RGB.

RGB is an additive color model that generates colors by combining blue, green and red and different intensities/brightness. In OpenCV we 8-bit color depths

- Red
- Green
- Blue





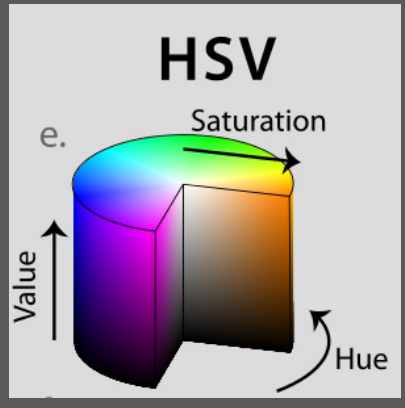
However, OpenCV actually stores color in the BGR format.

COOL FACT - We use BGR order on computers due to how unsigned 32-bit integers are stored in memory, it still ends up being stored as RGB. The integer representing a color e.g. 0x00BBGGRR will be stored as 0xRRGGBB00.

HSV Color Space

HSV (Hue, Saturation & Value/Brightness) is a color space that attempts to represent colors the way humans perceive it. It stores color information in a cylindrical representation of RGB color points.

Hue – Color Value (0 – 179)
Saturation – Vibrancy of color (0-255)
Value – Brightness or intensity (0-255)



It's useful in computer vision for color segmentation. In RGB, filtering specific colors isn't easy, however, HSV makes it much easier to set color ranges to filter specific colors as we perceive them.

Visit these links to learn more:

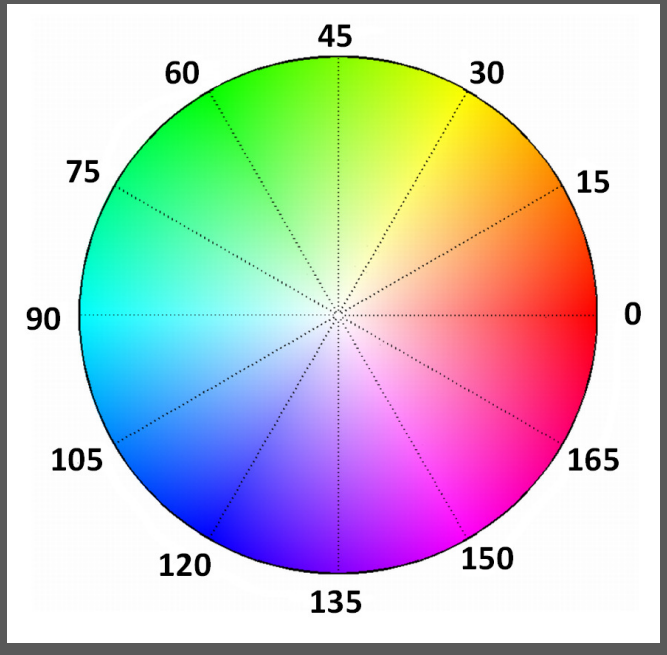
- wikipedia.org/wiki/HSL_and_HSV
- http://coecsl.ece.illinois.edu/ge423/spring05/group8/FinalProject/HSV_writeup.pdf

Color Filtering

The Hue (Hue color range, goes from 0 to 180, not 360) and is mapped differently than standard

Color Range Filters:

- Red – 165 to 15
- Green – 45 to 75
- Blue – 90 to 120



Baixar Imagem no SIGAA

Antes de prosseguirmos, baixem a imagem 'babuino.jpg' do SIGAA e coloquem n subdiretório (**img** superior ao seu código fonte.

Abrindo e Exibindo a Imagem

```
import cv2
import numpy as np

image = cv2.imread('../img/babuino.jpg')
cv2.imshow('Figura', image)
cv2.waitKey()
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Valor BGR e Suas Camadas

```
import cv2
import numpy as np

image = cv2.imread('../img/babuino.jpg')

# BGR Values for the first 0,0 pixel
B, G, R = image[0, 0]
print B, G, R

#BGR Layers
print image.shape
```

image.shape

A forma (shape) da uma imagem é acessado por **image.shape**. Que retorna uma tupla com número de **linhas, colunas e canais** (camadas) se a imagem for colorida.

Brincando com a Imagem

Abra a imagem no Paint e identifique uma região vermelha. Identifique a coluna e linha e depois extraia os valores BGR desta região. Utizando o Paint:

1. Click na ferramenta Seleção de Cores;
2. Selecione a Cor identificada;
3. Click em Editar Cores.

ATENÇÃO: O Paint inverte a ordem para Linha, Coluna.
Como resultado irá obter os valores para: Vermelho, Verde, Azul. Além de Matriz, Saturação e Iluminação (A serem estudados no futuro).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Brincando com a Imagem

Selecionei o ponto 98, 180 (linha, coluna) e recuperei os seguintes valores BGR (Azul = 196, Verde = 162 e Vermelho = 239).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Brincando com a Imagem

```
import cv2
import numpy as np

image = cv2.imread('../img/babuino.jpg')

# BGR Values for the first 0,0 pixel
B, G, R = image[98, 180]
print B, G, R

#BGR Layers
print image.shape
```

Resultado

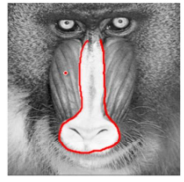
```
>>> 196 162 239
>>> (266L, 400L, 3L)
```

Baixar Imagem no SIGAA

Antes de prosseguirmos, baixem a imagem 'babuino2.jpg' do SIGAA e coloquem n subdiretório (**img** superior ao seu código fonte.

RGB Color Image

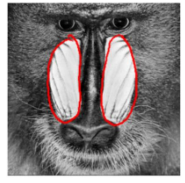
R



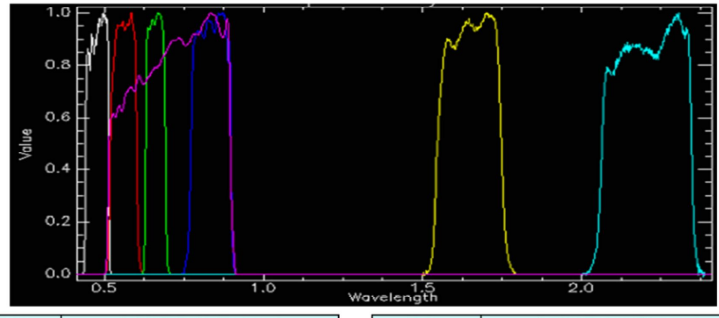
$2^8 = 256$

G

B



LANDSAT ETM+ Spectral



Color	LANDSAT ETM+ band	
White	1 (0.45 µm to 0.515 µm)	B
Red	2 (0.525 µm to 0.605 µm)	G
Green	3 (0.63 µm to 0.69 µm)	R
Blue	4 (0.75 µm to 0.9 µm)	

Color	LANDSAT ETM+ band	
Yellow	5 (1.55 µm to 1.75 µm)	
Not shown	6 (10.4 µm to 12.5 µm)	→ 60m
Cyan	7 (2.08 µm to 2.35 µm)	
Magenta	Pan (0.51 µm to 0.9 µm)	→ 15m

Exibindo os 3 Canais (BGR) Separados

```
import cv2
import numpy as np
image = cv2.imread('../img/babuino2.jpg')

# OpenCV's 'split' function splites the image
#into each color index
B, G, R = cv2.split(image)

cv2.imshow("Original", image)
cv2.imshow("Red", R)
cv2.imshow("Green", G)
cv2.imshow("Blue", B)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Mesclandos os 3 Canais (BGR) Separados

```
import cv2
import numpy as np
image = cv2.imread('../img/babuino2.jpg')

# OpenCV's 'split' function splites the image into each
    color index
B, G, R = cv2.split(image)

# Let's re-make the original image,
merged = cv2.merge([B, G, R])
cv2.imshow("Merged", merged)

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

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

Ampliando o Sinal de Azul

```
import cv2
import numpy as np
image = cv2.imread('../img/babuino2.jpg')

# OpenCV's 'split' function splites the
#image into each color index
B, G, R = cv2.split(image)

# Let's amplify the blue color
merged = cv2.merge([B+100, G, R])
cv2.imshow("Merged with Blue Amplified", merged)

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

Contraprova

Usando o Gimp ou Photshop faca a mesma operacão.

Bibliografia

- Solomon C., Breckon T. **Fundamentals of Digital Image Processing.. A Practical Approach with Examples in Matlab** (Wiley-Blackwell, 2011)(ISBN 9780470844724)(en)(355s)
- A. Conci, E. Azevedo e F.R. Leta - **Computação Gráfica: volume 2 (Processamento e Análise de Imagens Digitais)**, Campus/Elsevier. 2008 - ISBN 85-352-1253-3.
- Open CV Tutorials

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA ALAGOAS

DÚVIDAS ?


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
ALAGOAS