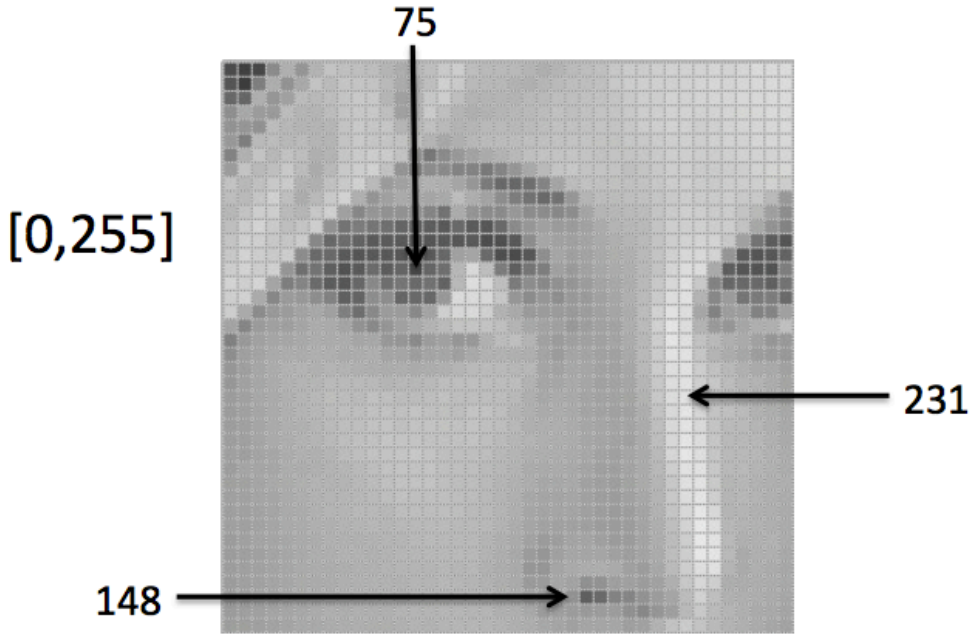
Imagem, Pixels, Modelos de Cor e Operações

Visão Computacional - UFMA

Imagem Digital

- Uma imagem contém uma quantidade discreta de elementos chamados **pixels**
- Cada pixel possui um valor
  - intensidade no caso de tons de cinza



Representação de Imagens

- São normalmente discretas, representadas numa matriz regular de valores inteiros quantizados

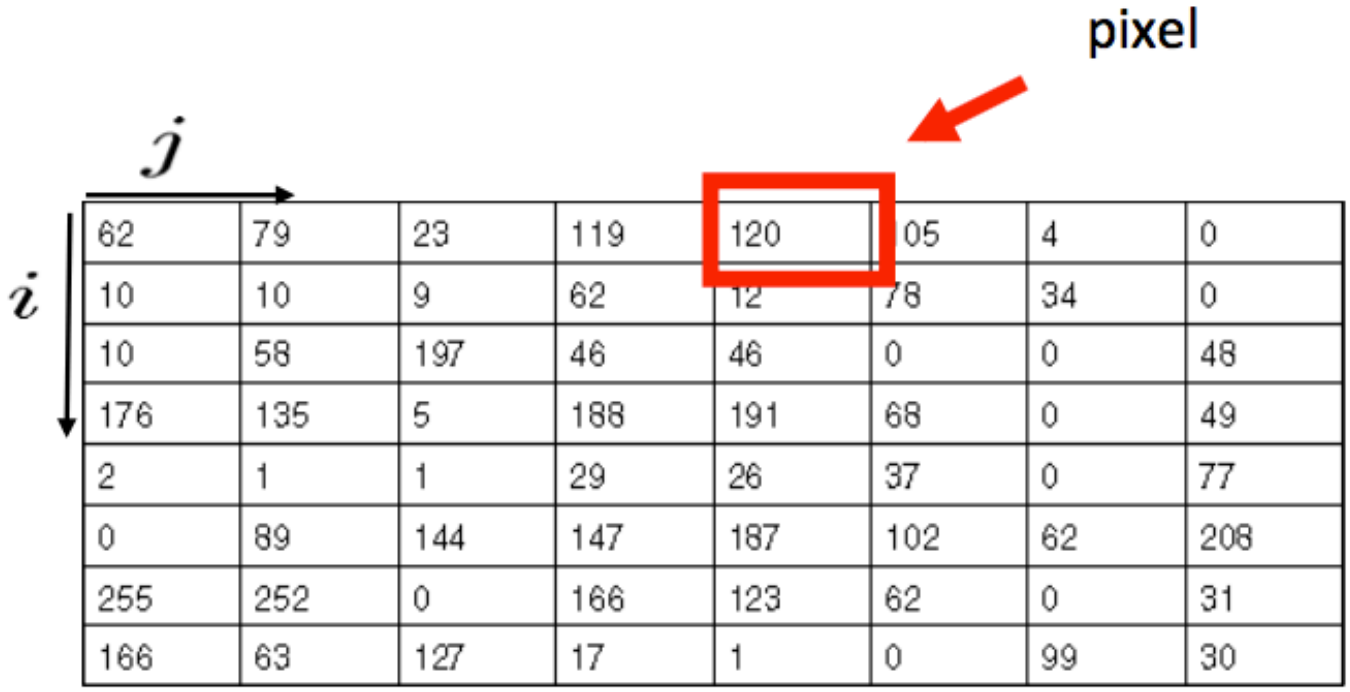


Imagem como função

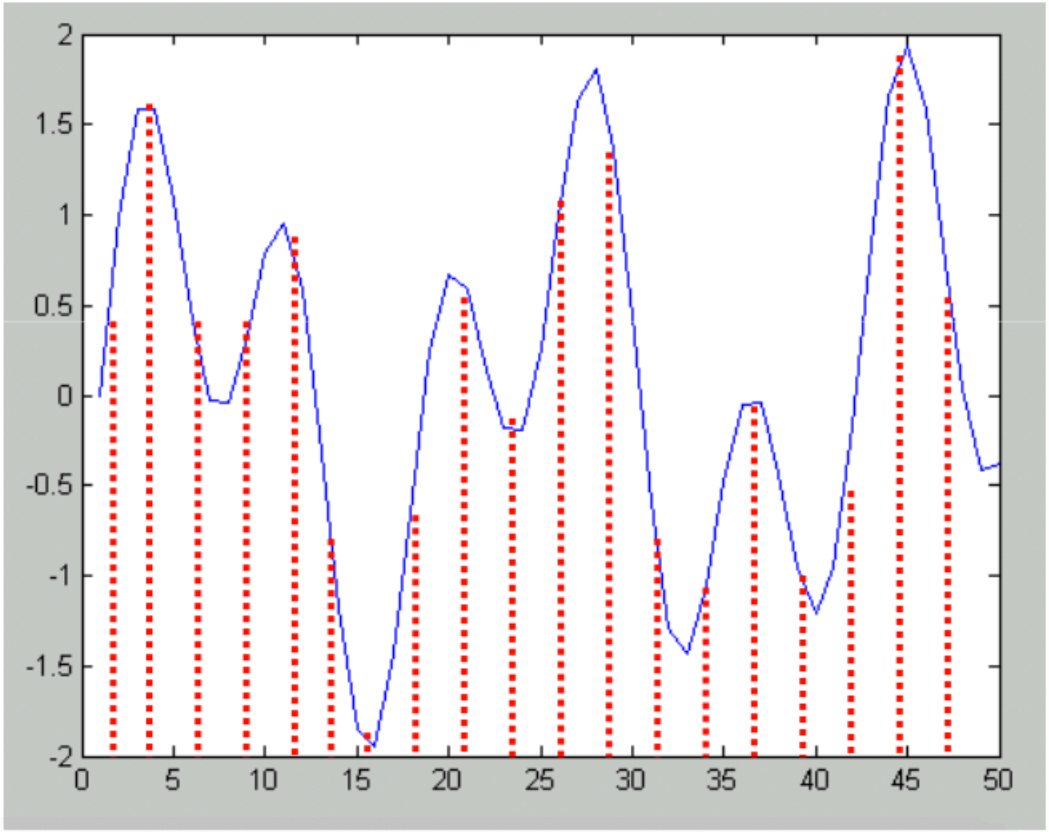
- Teoricamente, imagens são funções de R^2 a R^M do brilho refletido de uma cena

  f(x,y) fornece o valor de intensidade numa posição **(x,y)**

- O processo envolve as etapas de:
  - Amostragem
  - Quantização

Amostragem

- Amostra a função 2D contínua para um conjunto de elementos discretos (pontos da imagem)
- Sensor



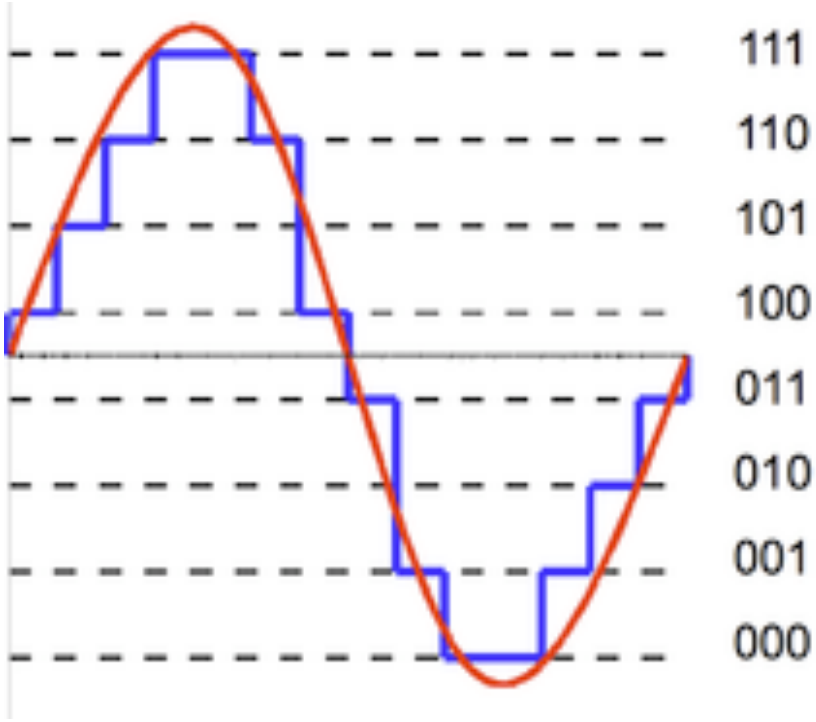
Amostragem

- Possui relação com a quantidade de nitidez que seja necessário na imagem que será adquirida

Do topo esquerdo para baixo direita: 256x192,128x96, 64x48, 32x24

Quantização

- Define a dimensão de cor dos pontos da imagem
  - Sinal em 3 bits



Quantização

- Define a quantidade de cores presentes em um ponto da imagem.
- Normalmente representada em bits

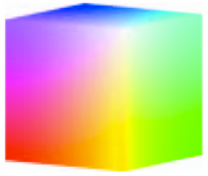
Exemplo de quantização uniforme

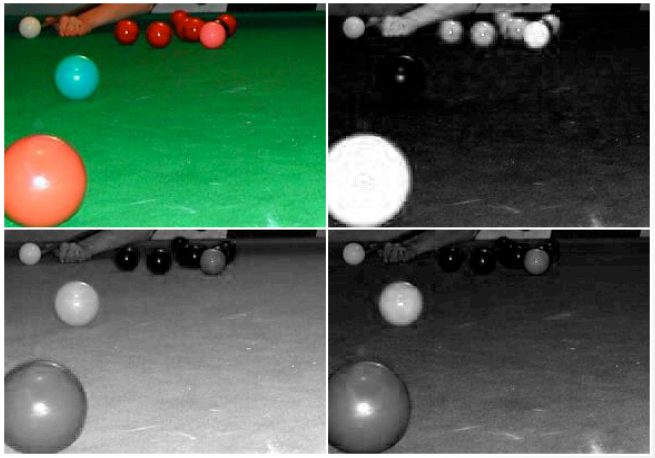
Coloridas x Tons de Cinxa

- Tons de Cinza = apenas Luminancia (intensidade de luz - escala de cinza)
  - Representação Simples
  - Humanos podem entender
- Coloridas (Luminancia + Crominancia)
  - Múltiplos canais (normalmente 3)
  - Processamento mais complexo

Imagens RGB

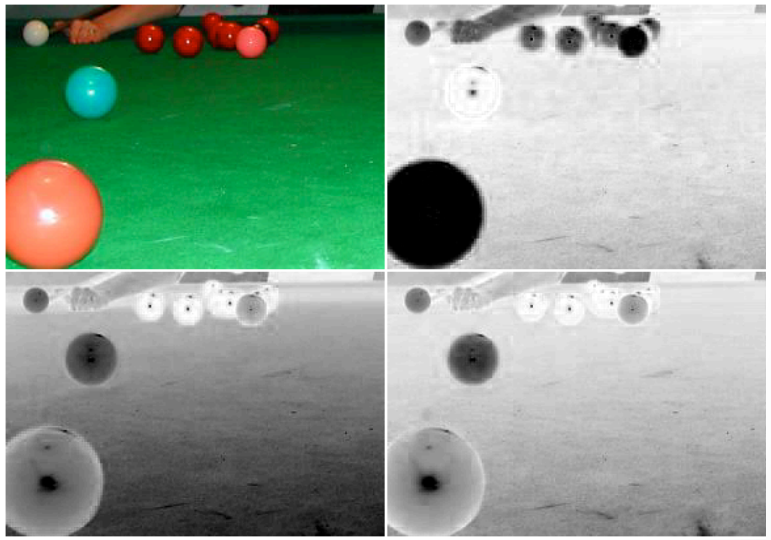
- Red-Green-Blue
  - Mais comum
  - Canais correspondem aproximadamente às seguintes frequências
    - R 700nn
    - G 546nn
    - B 436nn
  - Quando combinados:





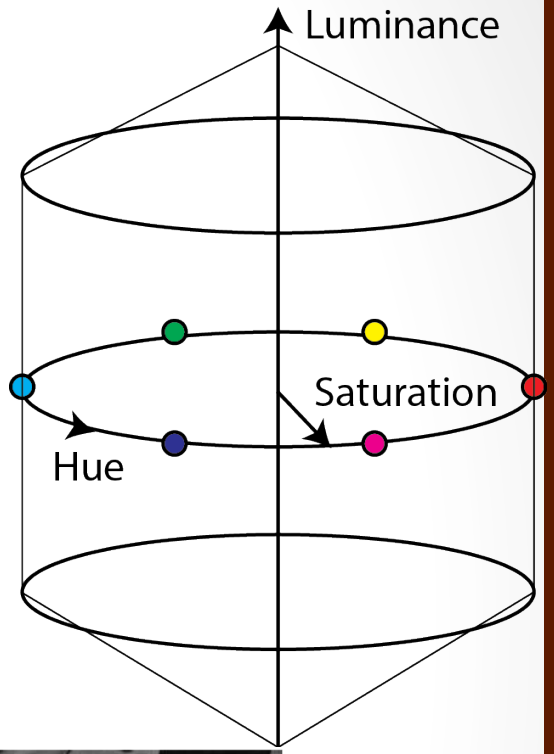
Imagens CMY

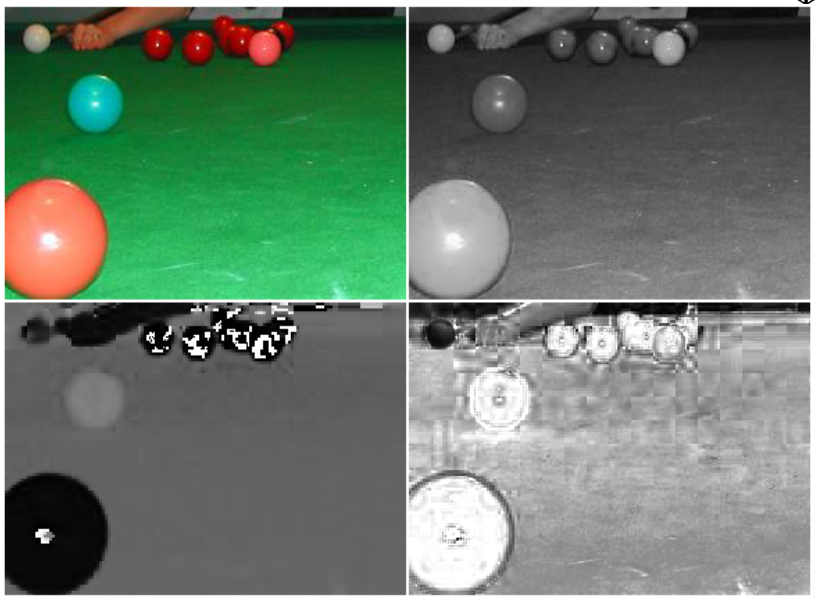
- Cyan-Magenta-Yellow
  - Cores secundárias baseadas na distancia do Branco
  - Toma como base o sistema subtrativo
    - C = 255 – R
    - M = 255 – G
    - Y = 255 – B
  - Usado em impressoras



Imagens HSL

- Hue-Saturation-Luminance
  - “tom-saturação-luminancia”
- Separa o que é Cor do que é Luminancia num sistema cilindrico
  - Hue 0..360
  - Luminancia de 0..1
  - Saturation de 0..1

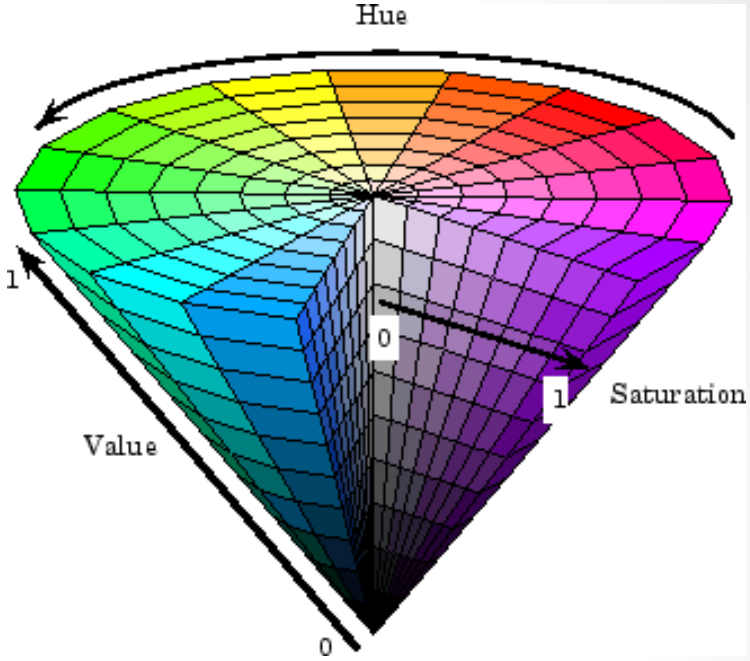


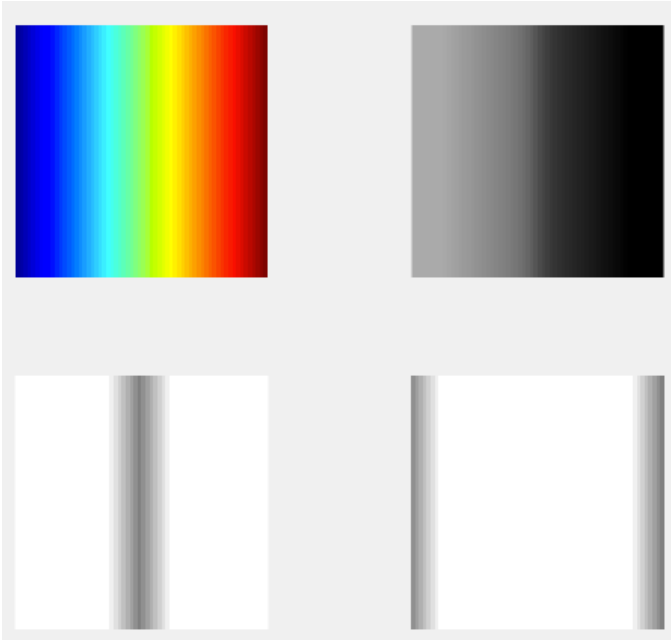


Imagens HSV

- Hue-Saturation-Value
  - Matiz-saturação-intensidade
- Utilizado como paleta de cores

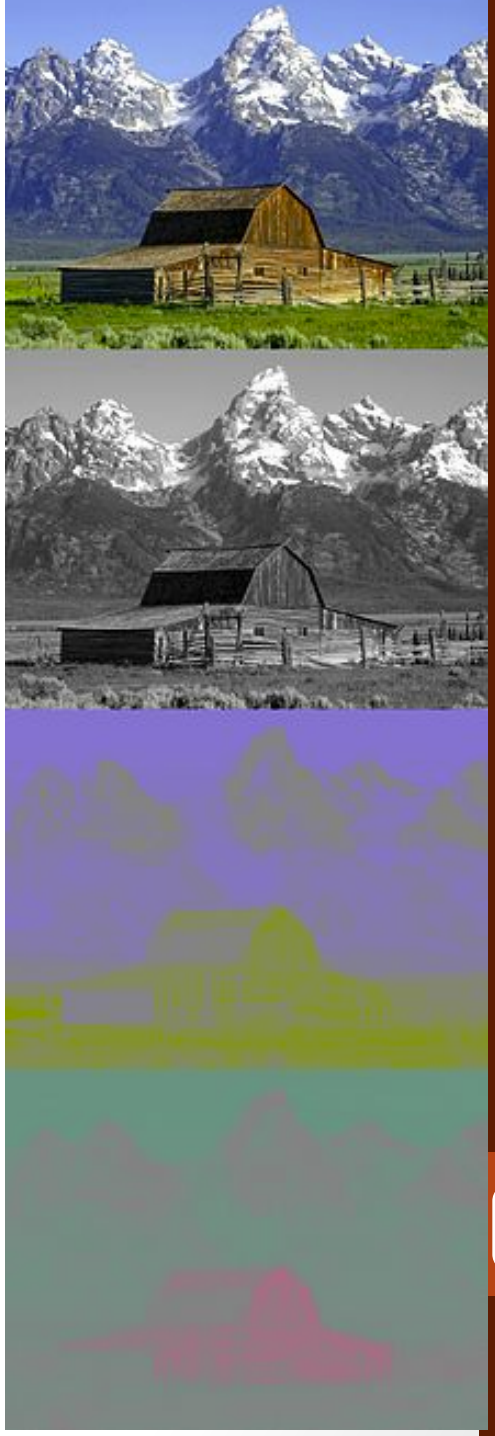
- Hue (cor em si) 0..1
- Value (ou brilho) 0..1
- Saturation (pureza) 0..1





Imagens YCrCb

- Dedicado ao video analógico
- Y = Luminancia
- Cr = Crominancia em vermelho em relação a um valor de referência
- Cb = Crominancia em azul em relação a um valor de referência

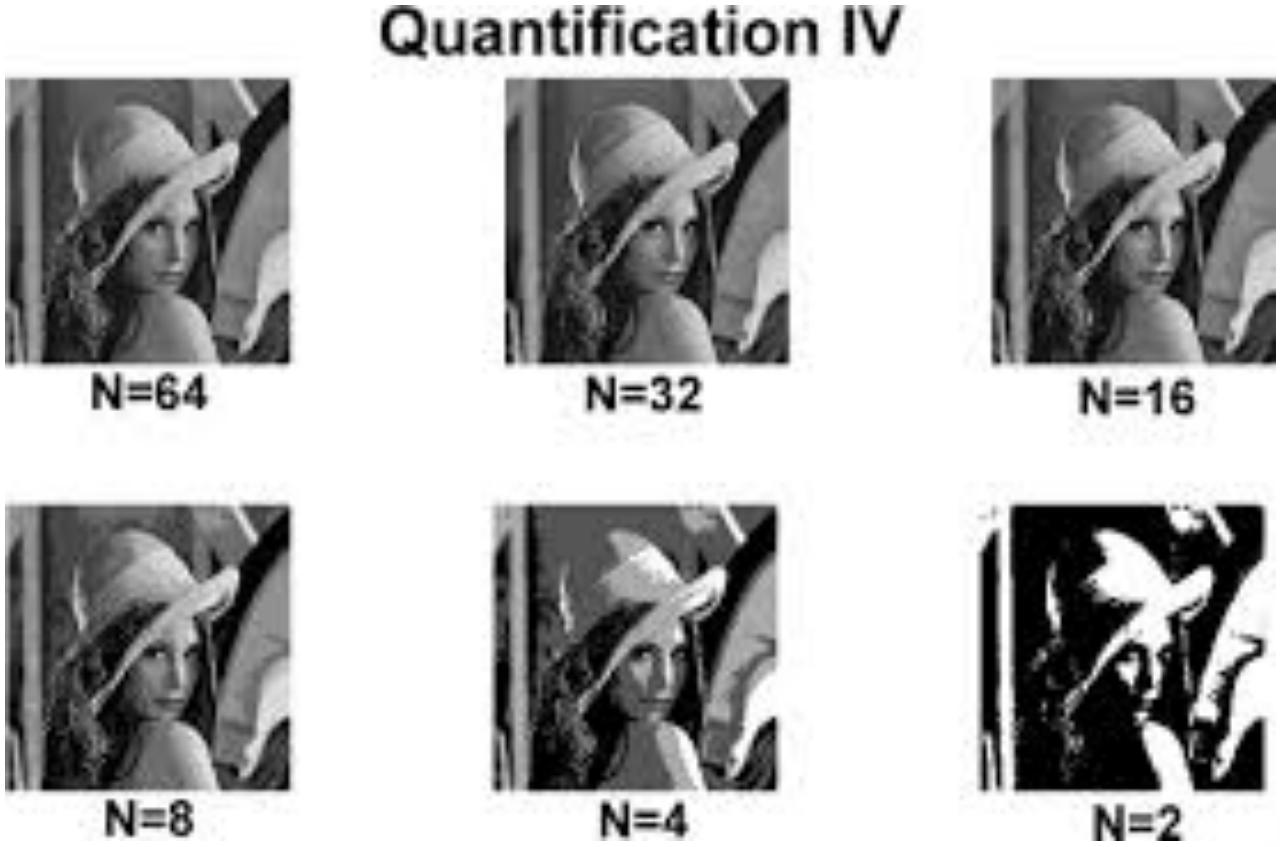


Outros espaços de cores

- CIE XYZ
- CIE XYZ
- CIE L*u*v*
- CIE L*a*b*
- Bayer

Aplicação

- Quantização Uniforme



Cheque o exemplo implementado no material da disciplina

Aplicação

- Quantização Dithering
  - Um dos algoritmos: Floyd–Steinberg
  - Acessa cada pixel, aplica a escala e propaga o erro conforme vizinhança abaixo

$$\begin{bmatrix} & & * & \frac{7}{16} & \cdots \\ \cdots & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} & \cdots \end{bmatrix}$$

Cheque o exemplo implementado no material da disciplina

Aplicação

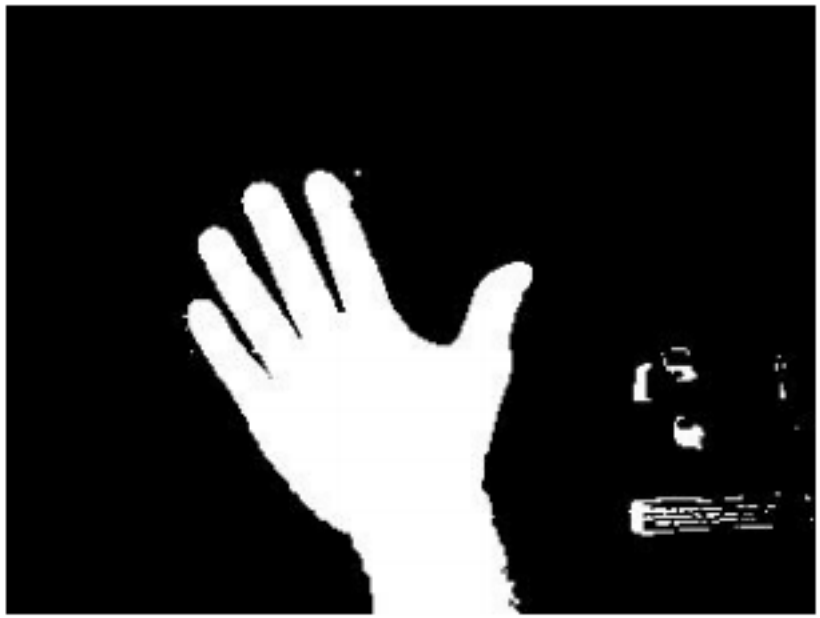
- Detecção de pele com HSL

(S >= 0.2) AND (0.5 < L/S <3.0) AND (H <= 28º OR H >= 330º)

Cheque o exemplo implementado no material da disciplina

Aplicação

- Segmentação de pele com YCbCr e tracking de mão
- $85 \leq Cb \leq 135$ $135 \leq Cr \leq 180$ $Y \geq 80$



TOWARDS A BIOMETRIC PURPOSE IMAGE FILTER ACCORDING TO SKIN DETECTION

Aplicação

- Detecção de Olhos Vermelhos com HSL

(L >= 0.25) AND
(S >= 0.4) AND
(0.5 < L/S <1.5) AND
(H <= 14º OR H >= 324º)

Operações básicas

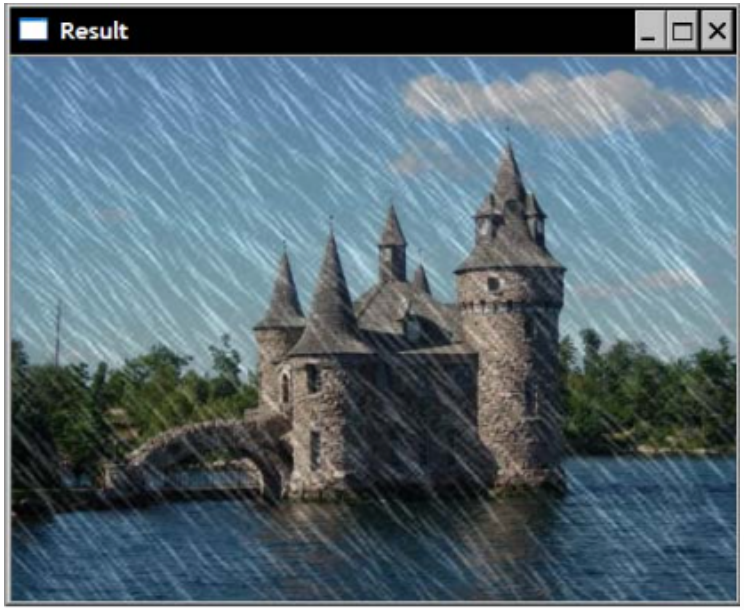
- Leitura do disco
  - Mat image= **imread**("img.jpg”)
- Imagem zerada
  - Mat imagem (240,320,CV_8U)
    - CV_8U -> 1 canal, unsigned
    - CV_8U3C -> 3 canais, unsigned
    - CV_16S -> inteiro 16 bits, 1 canal
- Copiar referencia
  - result**.copyTo**(image3)
- Clone
  - copia = image.clone()
- Salvar
  - **imwrite**("output.jpg", result);

Operações básicas

- **addWeighted**(image1,0.7,image2,0.9,0.,result)

+

=



Outras
add(img1, img2, result, mask*)
add(img1, Scalar(k), result)
Não obrigatório

Operações Básicas

- Aritméticas
  - absdiff
  - subtract
  - multiply
  - divide
- BitWise
  - bitwise_and
  - bitwise_or
  - bitwise_xor
  - bitwise_not
- Operações
  - sqrt
  - pow
  - abs
  - cuberoot
  - exp
  - log

Operações básicas

- No opencv 2.0, overload de operadores:
  - result= 0.7*image1+0.9*image2;
- Bitwise
  - &, |, ^, ~
- Comparações
  - <, <=, ==,!=, >, >=;
- Matemáticas
  - img.inv() → inversa
  - img.t() → transposta
  - img.determinant() → determinante
  - vetor.norm() → norma do vetor
  - vetor.cross(v2) → produto vetorial
  - vetor.dot(v2) → produto escalar

Operações básicas

- Split/Merge de canais da imagem

```
std::vector<cv::Mat> planes;
split(image1,planes);

....

merge(planes,result) ;
```

Operações básicas

- Definindo regiões de interesse
- Útil quando se precisa processar apenas um pedaço da imagem
- Como?

```
Mat imageROI;
imageROI= image(Rect(385,270,logo.cols,logo.rows));
addWeighted(imageROI,1.0,logo,0.3,0.,imageROI);
```

Parta para as práticas!

- Faça os exercícios propostos
  - Instalar opencv
  - Configurar um projeto para o opencv
  - Usar qualquer programa demo para testar, por exemplo, um que abra uma imagem qualquer

  - Faça os exercícios propostos pelo professor para se acostumar com o opencv (Estão no site)