

Aula 06: Processamento de Imagens

1. Introdução

Imagens podem ser entendidas como informações que, ao invés de serem descritas no tempo, são descritas no espaço.

Desse modo, uma imagem digital é um conjunto de informações formado por amostras cujo valor corresponde a uma cor de uma determinada palheta.

Estas amostras são os chamados *pixels* que, quando colocados suficientemente próximos uns dos outros, formam uma imagem contínua quando vista pelos olhos humanos.

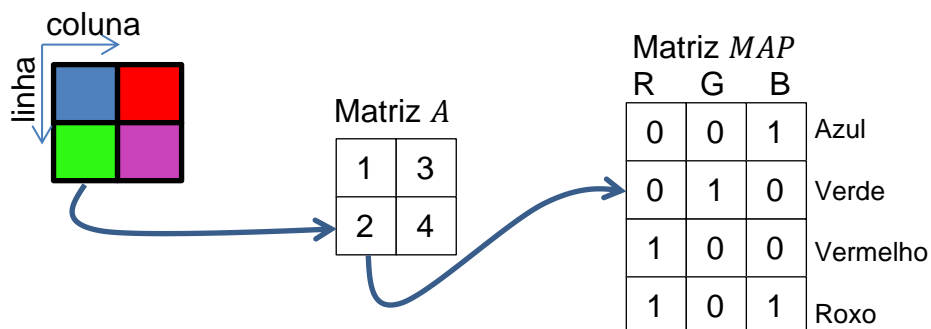
A posição de cada *pixel* em uma imagem é dada pela indexação da respectiva amostra em uma matriz, através de dois números inteiros ($m \times n$) que representam a linha e a coluna de cada matriz.

2. Formatos de arquivos de imagens

Existem diversos formatos de imagens digitais, mas na aula de hoje usaremos apenas dois: O formato *.mat*, utilizado para o armazenamento de qualquer tipo de dado no MATLAB, e o famoso *.jpg*.

O formato *.mat* é utilizado para explorar o modo como as imagens são tratadas através de matrizes. Neste tipo de arquivo, o armazenamento da imagem é feito através de uma matriz $A(M \times N)$, onde cada elemento representa um pixel, e uma matriz $MAP(C \times 3)$, onde cada linha corresponde a uma das C diferentes cores que compõem a imagem, e cada coluna corresponde às intensidades das cores vermelho, verde e azul (RGB) que formam cada uma destas cores.

Como exemplo, vamos utilizar a imagem formada por 4 pixels apresentada abaixo. Para esta imagem, a matriz A tem dimensão (2×2) , ou seja, duas linhas e duas colunas e a matriz MAP tem dimensão (4×3) , uma vez que a imagem é formada por apenas 4 cores.

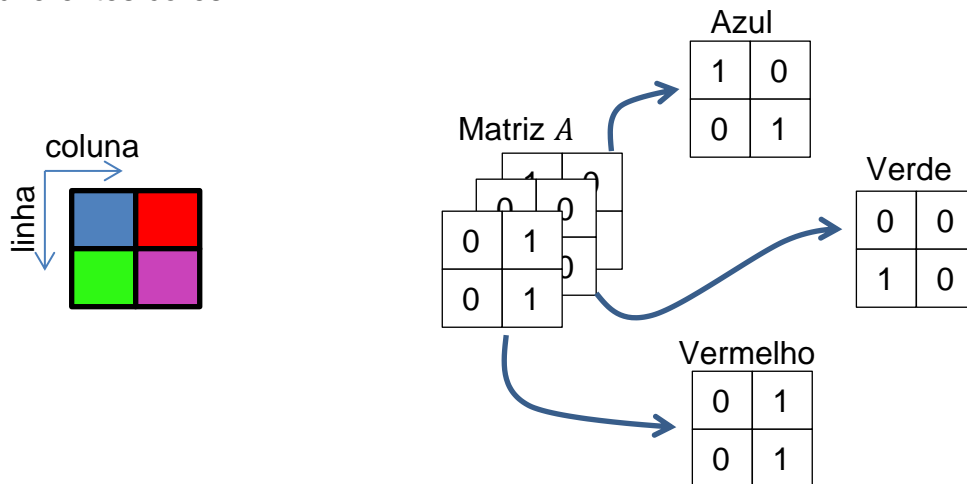


Assim, o pixel da linha 2, coluna 1, por exemplo, é definido com a atribuição do valor 2 na posição $A(2,1)$. Este valor representa a linha em que a cor verde foi definida na matriz MAP , através de um 0 na coluna correspondente ao vermelho (R), de um 1 na coluna correspondente ao verde (G) e de outro 0 na coluna correspondente ao azul (B).

Para carregar uma imagem do tipo *.mat* no MATLAB, utiliza-se o comando *load nome_da_imagem*.

O formato *.jpg* é um dos formatos mais utilizados para codificação de imagens, não só no MATLAB mas em todos os outros programas de tratamento de imagens. Neste tipo de arquivo, a palheta de cores não é utilizada, e em vez disso, os dados da imagem são armazenados em apenas uma matriz *A* de dimensão $(M \times N \times 3)$ onde cada nível da matriz contém informações sobre as cores vermelho, verde e azul que compõe cada um dos $M \times N$ pixels.

Como exemplo, vamos usar a mesma imagem composta por quatro pixels de diferentes cores:



Usemos agora o pixel da posição (2,2) de cor roxa. Assim como visto no exemplo anterior, a cor roxa é composta pelas cores vermelha e azul. Por este motivo, $A(2,2,1) = 1$, $A(2,2,2) = 0$ e $A(2,2,3) = 1$.

O comando $x = \text{imread}(\text{nome_da_imagem}, 'jpg')$ é o comando utilizado para carregar uma imagem neste formato para o MATLAB e o comando $\text{imwrite}(A, \text{nome_do_arquivo}, 'jpg')$ é o comando que grava a matriz *A* no arquivo.

3. Atividades*

- I. O arquivo **Aula06_ex01.mat** fornece a imagem de um palhaço definida nos formatos de uma imagem *.mat*. Crie um programa chamado de **Aula06_ex01.m** que:
 - a) Carregue esta imagem no workspace digitando **load Aula06_ex01** e verifique que duas variáveis foram criadas: *map*, que contém as informações de cores e intensidades e *X*, que contém as informações sobre os pixels e suas posições.

Com base nestas variáveis, responda: Quantos pixels compõem esta imagem e quantas são suas possíveis cores?

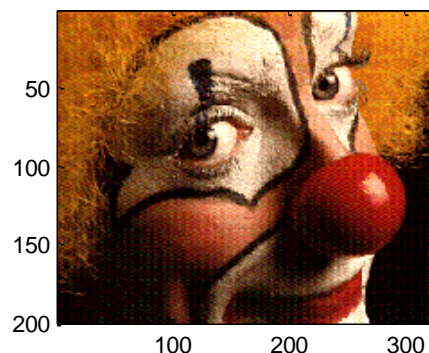
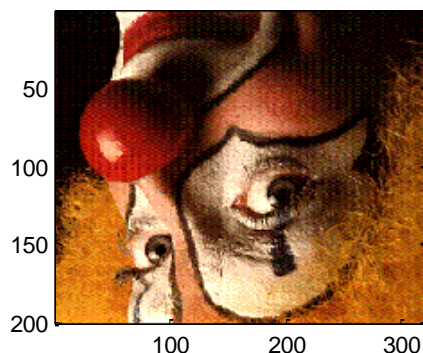
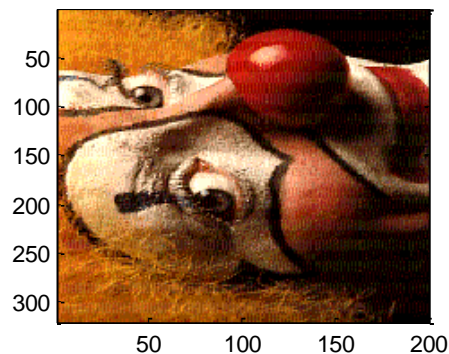
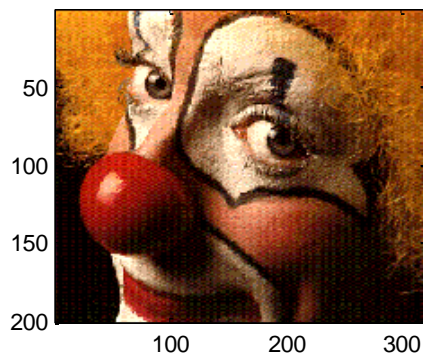
*Atividades baseadas na apostila do Laboratório de Comunicações Digitais -março/2007- Márcio Eisencraft e Marco Antônio Assis.

- b) Digite `image(X)` para ver a imagem e verifique que apesar um palhaço, a imagem criada apresenta cores de aspecto não-naturais. Isso ocorre porque ao digitar o comando `image`, o MATLAB utiliza uma palheta de cores padrão. Assim, para utilizar a palheta correta, deve-se utilizar, em conjunto com `image`, o comando `colormap(map)`.
- c) Para aumentar o brilho da imagem, pode-se utilizar o comando `brighten(β)`, onde $-1 < \beta < 1$. Digite `help brighten` para ter mais informações sobre este comando e verifique seus efeitos sobre o palhaço.
- d) Um zoom em uma imagem nada mais é do que a seleção de uma determinada parte da imagem. Por exemplo, para dar um zoom no olho esquerdo do palhaço podemos utilizar a seguinte sequência de comandos:

```
Xolho=X(50:100, 150:225);
image(Xolho);
colormap(map)
```

Obtenha, agora, uma ampliação do nariz do palhaço.

- e) Assim como a ampliação (ou zoom), outras operações matriciais podem ser utilizadas para trabalhar com imagens. Por exemplo, deitar a imagem significa transpor a matriz e inverter a ordem das linhas da matriz, colocando a última linha em primeiro e a primeira linha em último, significa colocar a imagem de ponta-cabeça. Assim, crie uma sequência de comandos que forneça as imagens, cada uma na sua posição:



- II. O arquivo **Aula06_ex02.jpg** fornece a imagem de um moinho de vento. Execute a sequência de comandos apresentada na figura abaixo e interprete o resultado.

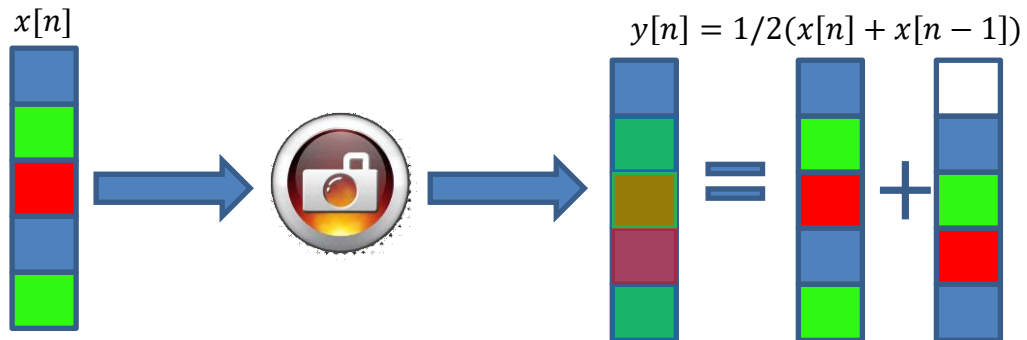
```
X=imread('Aula06_ex02.jpg');
Xred=X(:,:,1);
Xgreen=X(:,:,2);
Xblue=X(:,:,3);

subplot(221)
image(X)
subplot(222)
image(Xred)
subplot(223)
image(Xgreen)
subplot(224)
image(Xblue)
```

- III. O borrão em uma imagem é causado pelo movimento do objeto em relação à câmera fotográfica no instante em que é feita a aquisição da imagem. Este movimento faz com que os pixels se embaralhem com outros pixels de modo que cada um deles contenha informações dos N pixels anteriores.

Para entender este processo, vamos supor que cada coluna da imagem original seja chamada de $x[n]$, que cada coluna da imagem armazenada pela câmera seja $y[n]$ é que o ato de fazer a fotografia seja um sistema que tem $x[n]$ como entrada e $y[n]$ como saída.

Como exemplo, vamos supor uma imagem formada por cinco pixels em coluna, onde o borrão acontece somente entre dois pixels.



Assim de um modo mais geral, podemos dizer que o borrão, gerado pela sobreposição dos N últimos pixels pode ser modelado como

$$y[n] = \frac{1}{N} \sum_{k=n-N+1}^n x[k] = \frac{1}{N} (x[n] + x[n-1] + \dots + x[n-N+1])$$

Que nada mais é do que um sistema de médias móveis. Este sistema pode ser implementado através do loop *for*, como vimos nas aulas anteriores, mas também através do comando *filter*.

O comando *filter* é utilizado para verificar a resposta $y[n]$ de um sistema devido a uma entrada $x[n]$. Como exemplo, utilizemos a seguinte equação,

$$y[n] + 0,9 \cdot y[n - 2] = 0,3 \cdot x[n] + 0,6 \cdot x[n - 1] + 0,5x[n - 2]$$

cuja resposta impulsiva é obtida através da seguinte sequência de comandos:

```
n=0:10;
x=impulso(n);
b=[0.3 0.6 0.5];
a=[1 0 0.9];
y=filter(b,a,x);
```

Voltando ao caso da imagem, para borrá-la **verticalmente**, considerando a sobreposição dos últimos 25 pixels, temos a seguinte equação:

$$y[n] = \frac{1}{25} \cdot (x[n] + x[n - 1] + \dots x[n - 23] + x[n - 25])$$

e, conseqüentemente a seguinte sequência de comandos:

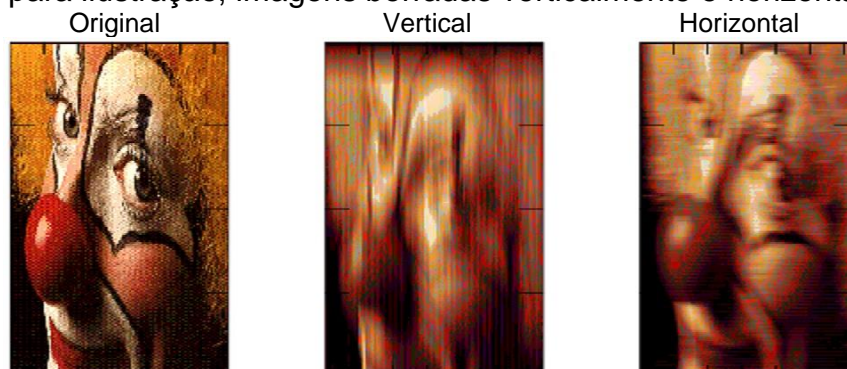
```
load Aula06_ex01
N=25;
subplot(211);
Yvert=filter(ones(1,N)/N,1,X);
image(Yvert);
colormap(map);
```

Do mesmo modo, para recuperá-la, utilizamos o filtro inverso fazendo:

```
Xrec=filter(1,ones(1,N)/N, Yvert);
subplot(212);
image(Xrec);
colormap(map);
```

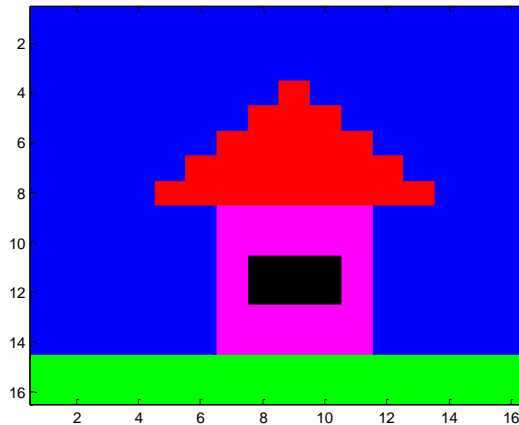
Com base nestes exemplos, faça uma sequência de programa capaz de simular um borrão **horizontal** e depois recuperá-lo.

Apenas para ilustração, imagens borradas verticalmente e horizontalmente:



Exercícios de fixação

- I. A partir da criação das matrizes, $A(16 \times 16)$ e $map(5 \times 3)$, desenvolva uma sequência de comandos capaz de criar a figura abaixo. Salve-a em um arquivo *Aula06_fix01.mat*.



- II. Gere novamente a imagem da casinha, porém, desta vez, crie apenas uma matriz $A(16 \times 16 \times 3)$ e salve-a em um arquivo *Aula06_fix02.jpg*
- III. O arquivo *Aula06_fix03.mat* apresenta a foto de uma placa tirada com o carro em movimento.
- a) (Assis e Eisenkraft, março de 2007) Usando as técnicas aprendidas em sala de aula, carregue esta imagem e tente identificar o que está escrito na placa. (não é necessário obter a placa original, basta reconhecer o que está escrito).
- b) Acesse o site www.mathworks.com/help e tente recuperar a placa original através das dicas apresentadas.

