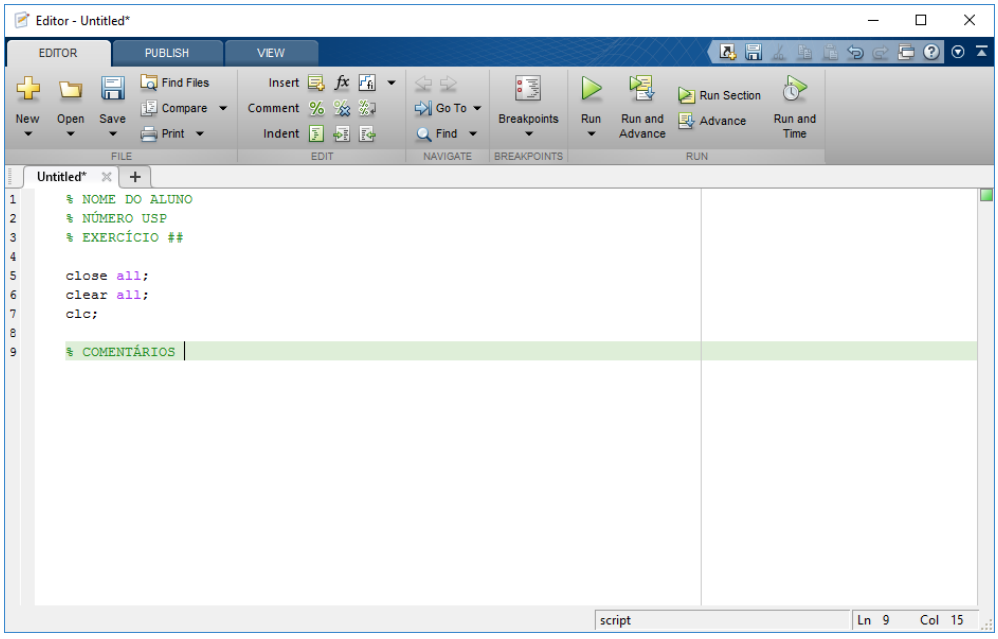
SEL5886 –Visão Computacional
Prof. Dr. Marcelo Andrade da Costa Vieira

Prática 3 – Processamento de Imagens Coloridas

Instruções:

- Essa prática consiste de 12 Exercícios (E_1 a E_12).
- Além dos exercícios a prática também tem 6 Treinamentos (T_1 a T_6).
- Os Treinamentos NÃO precisam ser enviados! Apenas os Exercícios.
- Alguns exercícios precisam de Funções adicionais, que estão disponíveis no site da disciplina para *download*.
- Deve ser gerado um arquivo no editor do Matlab (extensão *.m) para cada exercício pedido.
- Deve-se colocar comentários nos programas desenvolvidos.
- As perguntas devem ser respondidas também como comentários no arquivo.
- Deve-se tornar o diretório onde estão as figuras e os arquivos *.m como um diretório padrão do Matlab.
- Depois de terminado os exercícios, todos os arquivos *.m devem ser comprimidos em um único arquivo e enviado ao professor pelo site de *upload* da IRIS até a data máxima de entrega.

- Utilizar o padrão mostrado na Figura abaixo para seus arquivos *.m:
    - Colocar um cabeçalho contendo seu nome, número USP e o número do exercício correspondente (E1, E2, E3...);
    - Iniciar todos os exercícios com os 3 comandos mostrados na Figura abaixo, que servem para limpar as variáveis e as figuras abertas, além de limpar a tela de comando do Matlab;
    - Colocar comentários nas linhas de programa.



PRÁTICA 3

1) Imagens RGB.

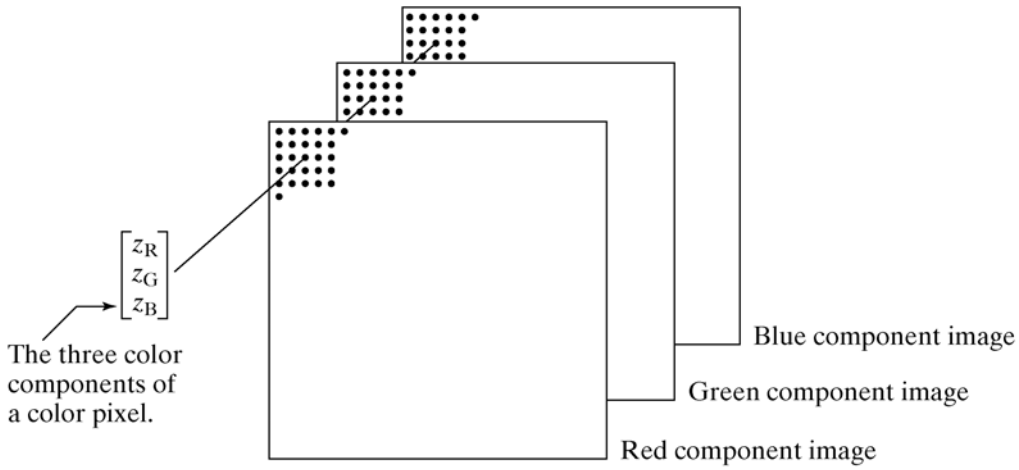


Figura 1 - Píxels de uma Imagem RGB

T_1: Decompor a imagem Flor.bmp em suas três componentes RGB.

```
f = imread('Flor.bmp');
fR = f(:, :, 1);
fG = f(:, :, 2);
fB = f(:, :, 3);
subplot(2,3,1);imshow(f)
subplot(2,3,2);imshow(fR)
subplot(2,3,3);imshow(fG)
subplot(2,3,4);imshow(fB)
```

T_2: Gerar a imagem RGB a partir de suas três componentes.

```
g = cat(3, fR, fG, fB);
subplot(2,3,1);imshow(fR)
subplot(2,3,2);imshow(fG)
subplot(2,3,3);imshow(fB)
subplot(2,3,4);imshow(g)
```

2) Imagens Indexadas.

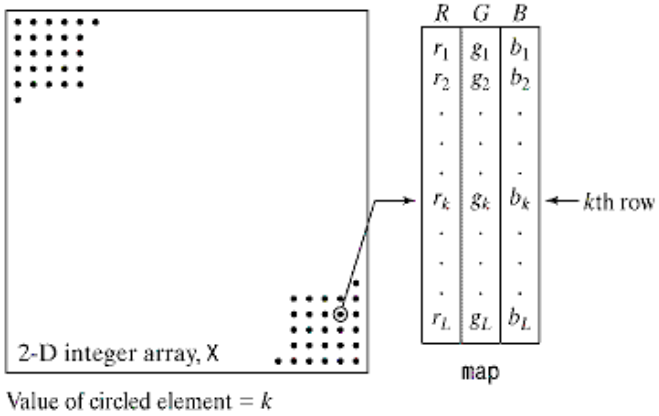


Figura 2 Imagens Indexadas

Uma Imagem Indexada no MATLAB tem dois componentes:

a) Uma Matriz de Dados Inteiros (***X***), e
b) Uma Matriz de cor (***map***)

A Matriz ***map*** é um arranjo de m x 3 elementos de classe ***double*** contendo valores em ponto-flutuante no intervalo [0,1].
O comprimento ***m*** do ***map*** é igual ao número de cores definido.
Cada linha da Matriz ***map*** especifica o valor de R,G,B de uma cor na Matriz ***X***. Ou seja, um Pixel de cor em ***X*** é um ponteiro para sua cor RGB em ***map***.

T_3: Converter uma imagem RGB em uma imagem Indexada.

```
RGB = imread('peppers.png');
[X,map] = rgb2ind(RGB,256);
figure
imshow(X,map)
```

Para alterar o número de cores de uma imagem indexada pode-se usar a função ***imapprox*** que tem a seguinte sintaxe:

[Y, newmap] = imapprox(X, map, n)

onde ***n*** é o número de cores da nova imagem.

E_1: Reduzir e aumentar o número de cores da imagem indexada X mostrando e discutindo os resultados.

Para se especificar um mapa de cores pode-se usar a seguinte sintaxe:

map(k, :) = [r(k) g(k) b(k)]

onde ***[r(k) g(k) b(k)]*** são os valores RGB que especificam uma linha de um mapa de cor. O mapa deve ser completado variando-se ***k.***

E_2: Gerar um mapa de cores para a imagem X e obter a imagem equivalente.

Long name	Short name	RGB values
Black	k	[0 0 0]
Blue	b	[0 0 1]
Green	g	[0 1 0]
Cyan	c	[0 1 1]
Red	r	[1 0 0]
Magenta	m	[1 0 1]
Yellow	y	[1 1 0]
White	w	[1 1 1]

Figura 3 -= Valores RGB para as cores básicas

A Figura 3 mostra a lista de valores RGB para as cores básicas.

T_4: Alterar o fundo (Background) da imagem X para verde. Qualquer dos comandos abaixo pode ser usado:

whitebg(g')
whitebg('green')
whitebg([0 1 0])

O MatLab possui diversos mapas de cores pré-definidos. A Figura 4 mostra alista de colormaps disponíveis no MatLab. O número de cores pode ser especificado incluindo-se o número entre parêntesis.

E_3: Alterar, usando a Figura 4, o mapa de cores da imagem X variando-se o número de cores. Comentar sobre o colormap gray. Concluir a respeito.

Name	Description
`autumn`	Varies smoothly from red, through orange, to yellow.
`bone`	A gray-scale colormap with a higher value for the blue component. This colormap is useful for adding an "electronic" look to gray-scale images.
`colorcube`	Contains as many regularly spaced colors in RGB color space as possible, while attempting to provide more steps of gray, pure red, pure green, and pure blue.
`cool`	Consists of colors that are shades of cyan and magenta. It varies smoothly from cyan to magenta.
`copper`	Varies smoothly from black to bright copper.
`flag`	Consists of the colors red, white, blue, and black. This colormap completely changes color with each index increment.
`gray`	Returns a linear gray-scale colormap.
`hot`	Varies smoothly from black, through shades of red, orange, and yellow, to white.
`hsv`	Varies the hue component of the hue-saturation-value color model. The colors begin with red, pass through yellow, green, cyan, blue, magenta, and return to red. The colormap is particularly appropriate for displaying periodic functions.
`jet`	Ranges from blue to red, and passes through the colors cyan, yellow, and orange.
`lines`	Produces a colormap of colors specified by the `ColorOrder` property and a shade of gray. Consult online help regarding function `ColorOrder`.
`pink`	Contains pastel shades of pink. The pink colormap provides sepia tone colorization of grayscale photographs.
`prism`	Repeats the six colors red, orange, yellow, green, blue, and violet.
`spring`	Consists of colors that are shades of magenta and yellow.
`summer`	Consists of colors that are shades of green and yellow.
`white`	This is an all white monochrome colormap.
`winter`	Consists of colors that are shades of blue and green.

Figura 4 - Mapa de cores (colormap) pré-definidos

3) Manipulando-se Imagens RGB e Indexadas.

Function	Purpose
`dither`	Creates an indexed image from an RGB image by dithering.
`grayslice`	Creates an indexed image from a gray-scale intensity image by multilevel thresholding.
`gray2ind`	Creates an indexed image from a gray-scale intensity image.
`ind2gray`	Creates a gray-scale intensity image from an indexed image.
`rgb2ind`	Creates an indexed image from an RGB image.
`ind2rgb`	Creates an RGB image from an indexed image.
`rgb2gray`	Creates a gray-scale image from an RGB image.

Figura 5 - Funções de conversão entre imagens

T_5: Converter a imagem indexada X em escala de cinza e em RGB

```
GR = ind2gray(X,map);
RGB2 = ind2rgb(X,map);
figure, imshow(GR)
figure, imshow(RGB2)
```

E_4: Explicar o que faz a função dither (renderização) em imagens coloridas e em imagens em escala de cinza.

```
[X1, map1]= rgb2ind(RGB, 16, 'nodither');
[X2, map2]= rgb2ind(RGB, 16, 'dither');
G1 = dither(GR);
figure;imshow(RGB)
figure;imshow(X1,map1)
figure;imshow(X2,map2)
figure;imshow(G1)
```

4) Conversão de RGB para outros espaços de cores.

RGB para o NTSC:
A função ***rgb2ntsc*** realiza esta conversão e a função ***ntsc2rgb*** faz a re-conversão.

T_6: Converter a imagem RGB em NTSC e gerar separadamente as componentes de Luminância, Matiz e Saturação.

```
YIQ = rgb2ntsc(RGB);
subplot(2,2,1);imshow(YIQ)
subplot(2,2,2);imshow(YIQ(:,:,1))
subplot(2,2,3);imshow(YIQ(:,:,2))
subplot(2,2,4);imshow(YIQ(:,:,3))
```

RGB para YCbCr:
A função ***rgb2ycbcr*** realiza esta conversão e a função ***ycbcr2rgb*** faz a re-conversão.

E_5: Realizar a conversão para este espaço de cor conforme T_6 e explicar.

RGB para HSV:
A função ***rgb2hsv*** realiza esta conversão e a função ***hsv2rgb*** faz a re-conversão

E_6: Realizar a conversão para este espaço de cor conforme T_6 e explicar.

RGB para CMY:
A função ***imcomplement*** realiza a conversão entre os dois espaços de cores.

E_7: Realizar a conversão para este espaço de cor conforme T_6 e explicar.

RGB para HSI:
O Matlab não possui esta função implementada em seu Toolbox. Para fazer esta conversão copiar o arquivo ***rgb2hsi.p*** e ***hsi2rgb.p*** para o diretório work .

E_8: Realizar a conversão para este espaço de cor conforme T_6 e explicar.

5) Filtragem espacial de Imagens coloridas.

Para aplicar um filtro em uma imagem RGB deve-se:

1) Extrair as três componentes.
   R = RGB(:,:,1); G = RGB(:,:,2); B = RGB(:,:,3)
2) Filtrar cada componente individualmente
   RF = imfilter(R,w); GF = imfilter(G,w); BF = imfilter(B,w);
3) Reconstruir a imagem filtrada em RGB
   RGBF = cat(3,RF,GF,BF)

Ou pode-se filtrar diretamente a imagem RGB como se esta fosse em escala de cinza.
RGBF = imfilter(RGB,w)

Suavização (Passa Baixa):

E_9: Filtrar a imagem RGB através de um Filtro da Média de 25 x 25.
1) **Filtrando separadamente as componentes RGB.**
2) **Filtrando a imagem RGB sem separar as componentes**

Avaliar os resultados.

E_10: a) Converter a imagem RGB para HSI, separar as componentes, filtrar a componente de Intensidade (I) com o mesmo filtro da média de E_9, recompor a imagem HSI e re-converter para RGB.
b) Filtrar as 3 componentes (H S e I) separadamente e re-converter para RGB.
c) Filtrar a imagem HSI sem separar as componentes.
O que se pode concluir?

Realce(Passa Alta):

E_11: Filtrar a imagem RGB através de um Filtro Laplaciano, processando separadamente as componentes RGB.
Avaliar os resultados.

6) Trabalhando diretamente no espaço Vetorial RGB.

Detecção de bordas em Imagens RGB. Copiar o arquivo ***colorgrad.p*** para o diretório work.

A função colorgrad implementa o gradiente em imagens RGB com a seguinte sintaxe:

[VG, A, PPG] = colorgrad(f, T)

onde:

f é a Imagem RGB
T é um Threshold opcional no intervalo [0 1] – o default é zero.
VG é o Vetor Gradiente RGB (Magnitude)
A é o ângulo do vetor gradiente em radianos
PPG é o gradiente formado através da detecção de bordas nas componentes individuais. O detector utilizado é o de Sobel.

E_12: Aplicar o detector de bordas Gradiente no espaço RGB, conforme a função *colorgrad* e verificar a diferença entre o uso do detector aplicado nas componentes individuais, através de subtração das imagens resultantes.