PRÁTICA 1.

Atenção! Os quadros indicados com a letra **T** referem-se a Treinamento. Devem ser implementados mas não fazem parte do relatório da prática. Os quadros indicados coma letra **E** são os exercícios a serem entregues como relatório da prática.

1) Entrando com Matrizes no MatLab.

Matrizes podem ser inseridas no MatLab das seguintes maneiras:

a) Diretamente através de uma lista explícita de elementos.
b) Carregadas através de arquivos externos.
c) Geradas através de funções internas.
d) Criadas através de funções do usuário.

Convenções:

a) Separar os elementos de uma linha com espaços em branco ou vírgulas.
b) Usar ponto e vírgula (;) ou "Enter" para indicar o fim de cada linha.
c) Envolver a lista de elementos entre colchetes [ ] .
d) Cada linha deve ter sempre o mesmo número de elementos.

T_1: Digitar as Matrizes diretamente nos seguintes formatos e verificar os resultados.

```
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

B = [ 16  3  2  4
   20 30  4  4
    5 6  7  8]

C = [ 20,30 40;
   50,40,10
   80 20 15]

D = [ 1,2,3,4;5 6 7 8
   9 9 9 9
   8,8,8,8;
   4 5,9 4]
```

2) Classes de Dados no MatLab.

Classes de Dados:
A tabela da Figura 1 mostra as classes de dados suportadas pelo MatLab.
As coordenadas dos pixels nas imagens são da classe ‘inteiro’.
As oito primeiras classes mostradas na tabela são chamadas de numéricas.
Os valores dos pixels nas imagens podem ser de qualquer classe numérica.
As operações numéricas no MatLab são realizadas com a classe ‘double’

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

Figura 1- Classe dos Dados no MatLab

Conversão entre Classes de Dados:

Sintaxe:
B = nome_da _classe(A)

T_2: Converter as classes de dados das Matrizes digitadas anteriormente.

Usar o comando *whos* para ver a classe de dados em cada caso:

```
whos
AI = uint8(A); BI = uint16(B); CI = uint32(C); DS = single(D);
whos
```

Verificar as classes de dados e o número de bytes de cada matriz antes e depois da conversão.

3) Indexando Matrizes.

O MatLab suporta um grande número de esquemas de indexação de Matrizes que facilita a manipulação dos dados. A indexação permite acessar um ou mais elementos diretamente da matriz.

T_3: Verificar o resultado de cada linha abaixo, tentando prever qual será.

Para selecionar um determinado elemento da matriz deve-se utilizar:
A(i,j) sendo i número da linha e j o número da coluna com i,j = 1, ...n
Ex: Elemento da linha 2 e coluna 3:
A(2,3)

Usa-se dois pontos (:) para significar "**todos**" ou "**até**".
Ex: Todos os elementos da terceira coluna:
A3 = A(:,3)

Ex: **Todos** os elementos da segunda linha:
A2 = A(2,:)

Ex: **Todos** os elementos da linha 1 **até** a linha 2 e da coluna 1 **até** a coluna 3:
A4 = A(1:2, 1:3)

Ex: Fazer todos os elementos da coluna 4 iguais a zero:
A5 = A
A5 (:, 4) = 0

Ex: O elemento da última linha e última coluna:
A(end,end)

Ex: O elemento da última linha e duas colunas antes da última:
A(end, end-2)

Ex: Os elementos da linha 2 até a última e da última coluna até a primeira com passo de –2:
A(2:end, end:-2:1)

Ex: Transformar uma matriz em um vetor coluna alinhando pelas colunas:
V1 = A(:)

Ex: Gerar o vetor transposto:
V2 = V1.'

T_4: Indexar uma matriz através de outra matriz.

Vetores podem ser usados para indexar elementos de uma matriz.
A notação A([a b], [c d]) indexa os elementos da matriz A da seguinte maneira:

- (linha a, coluna c) (linha a, coluna d)
- (linha b, coluna c) (linha b, coluna d)

Ex:
A6 = A([1 3], [2 3])

Uma matriz da classe logical pode ser usada para indexar os elementos em uma outra matriz onde o valor lógico é um.
Ex:
AL = logical([1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1])
A(AL)

Obs: Na verdade, O MatLab armazena uma matriz na forma de um vetor alinhado por suas colunas, da primeira até a última. Assim, a indexação pode ser realizada de maneira uni-dimensional, do primeiro ao último elemento da matriz, na seqüência de colunas.

Ex: Indexar o elemento A(3,3) é o mesmo que indexar o 11º. elemento do vetor:
A(3,3)
A(11)

Logo, pode-se obter a soma de todos os elementos da matriz usando:
S1 = sum(A(:))

Ou:
S2 = sum(sum(A))

4) Tipos de Imagens no MatLab:

O Toolbox de Processamento de Imagens do MatLab trabalha com quatro tipos de Imagens:

- Imagens de Intensidades
- Imagens Binárias
- Imagens Indexadas
- Imagens RGB

Imagens como Matrizes:

O Toolbox de Processamento de Imagens do MatLab considera as imagens como matrizes, adotando a convenção dada na Figura 2b.

(Obs: A Figura 2a denota a convenção adotada para Processamento de Imagens em geral).

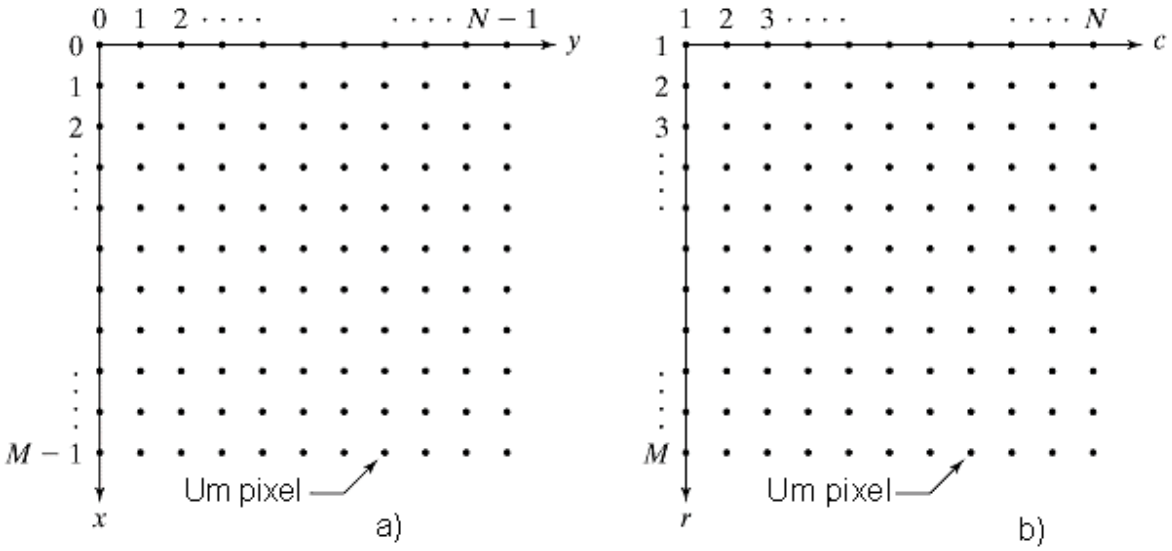


Figura 2 - a) Convenção para Processamento de Imagens. b) Convenção do MatLab

Imagens de Intensidades:

É uma Matriz de dados cujos valores representam as intensidades em cada ponto. Se os elementos de intensidade forem da classe uint8 seus valores estarão no intervalo [0, 255]. Se forem da classe uint16 seus valores variarão no intervalo [0, 65535]. Se os elementos forem da classe double, seus valores por convenção estarão no intervalo [0,1].

Imagens Binárias:

É um arranjo lógico de zeros e uns onde os dados são da classe logical.

Imagens Indexadas:

Requerem duas Matrizes. Conforme mostra a Figura 3, a matriz (X) contém um valor numérico que é um índice para uma segunda matriz (map) formada pelas quantidades de cores R(Red – Vermelho) G(Green – Verde) B(Blue – Azul) para o pixel correspondente.

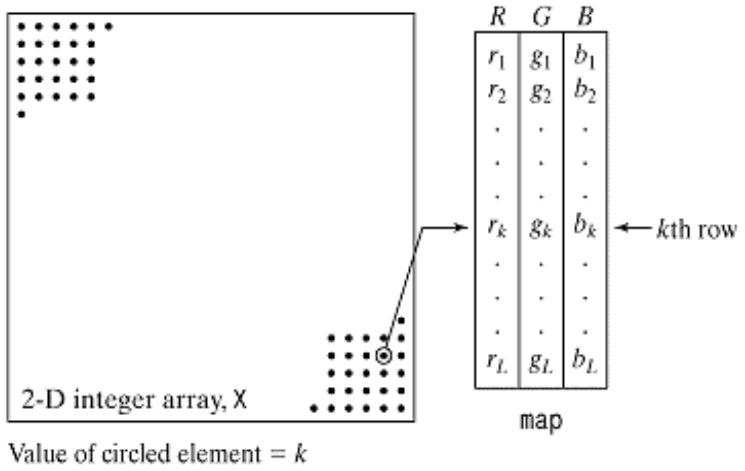


Figura 3 - Imagem Indexada

Imagens RGB:
São compostas por três matrizes separadas para cada cor R, G e B, normalizadas no intervalo [0,1].

5) Convertendo uma Matriz para uma Imagem de Intensidades.

Sintaxe:
I = mat2gray(A,[amin amax])
I = mat2gray(A)

Descrição:
I = mat2gray(A,[amin amax])
Converte a Matriz A para a Imagem de Intensidades I. A Matriz A deve ser double.
A Matriz I conterá valores entre 0 (preto) e 1 (branco).
Os Parâmetros amin e amax são os valores na Matriz A que corresponderão a 0 e 1 na Imagem I.

I = mat2gray(A)
Estabelece os valores de amin e amax como o mínimo e máximo dos valores da Matriz A.

T_5: Converter as Matrizes A, B, C, e D digitadas anteriormente para Imagens de Intensidades e observar os resultados.

IA = mat2gray(A)
IB = mat2gray(B, [2,30])
IC = mat2gray(C, [10,40])
ID = mat2gray(D, [0,10])

6) Mostrando uma Imagem de Intensidades.

Sintaxe:
imshow(I,n)
imshow(I,[low high])

Descrição:
imshow(I,n)
Mostra a Imagem de Intensidades I com n níveis discretos de cinza Se o valor de n for omitido, ***imshow*** usa 256 níveis de cinza em sistemas de 24-bits, ou 64 níveis de cinza em outros sistemas.

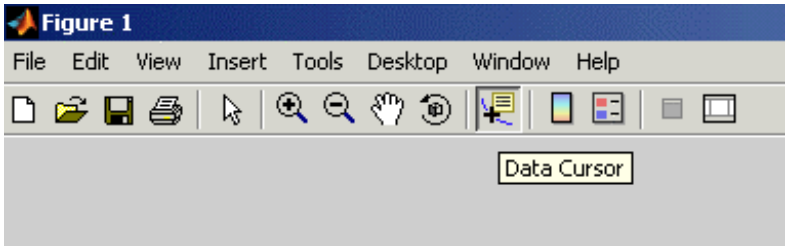
imshow(I,[low high])
Mostra a Imagem I em nível de cinza, especificando os limites dos valores de branco e preto. O parâmetro low (e qualquer valor menor do que ele) corresponde ao preto; o parâmetro high (e qualquer valor maior do que ele) é mostrado como branco. Valores intermediários são mostrados em escala de cinza, usando o número padrão de níveis. Se usado uma matriz vazia entre colchetes([]) para os parâmetros [low high], a função ***imshow*** usa [min(I(:)) max(I(:))]; ou seja, o menor valor em I é mostrado como preto e o maior valor em I é mostrado como branco.

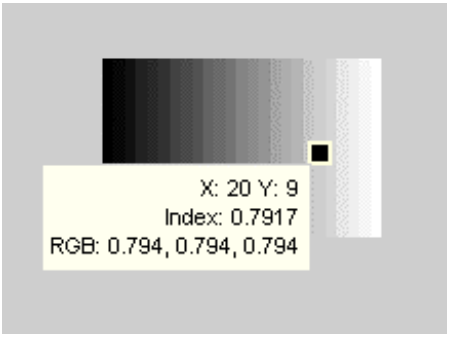
T_6: Mostrar a imagem dada pela Matriz E em escala de cinza e observar os resultados.

```
E = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
     1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25]

IE = mat2gray(E)

imshow(IE)
```

T_7: Verificar os valores de cada pixel diretamente na imagem.

Com o ícone "Data Cursor", conforme mostra a Figura abaixo, posicionar sobre a Imagem em escala de Cinza e verificar os valores dos níveis. Observe que X refere-se às colunas e Y refere-se às linhas.





E_1: Repetir o experimento anterior, mas limitando a imagem na sua escala de cinza. Observar os resultados.

imshow(IE,4)
imshow(IE,[0.3,0.7])
imshow(IE,[])

a) Comentar os resultados mostrados em cada caso.

Converter a Imagem de Intensidades IE para a classe *uint8:* (A Figura 4 mostra as funções para conversão entre as classes de dados de Imagens)
F = im2uint8(IE);
Mostrar a imagem F e responder:

b) Qual a classe dos dados dos pixels da Imagem IE?
c) Qual o intervalo dos valores?
d) Qual o intervalo a intensidade dos pixels da imagem F?

Name	Converts Input to:	Valid Input Image Data Classes
im2uint8	uint8	logical, uint8, uint16, and double
im2uint16	uint16	logical, uint8, uint16, and double
mat2gray	double (in range [0, 1])	double
im2double	double	logical, uint8, uint16, and double
im2bw	logical	uint8, uint16, and double

Figura 4 - Funções de conversão entre classes de dados de imagens

7) Convertendo uma Imagem de Intensidades para uma Imagem Indexada.

Sintaxe:
[X,map] = gray2ind(I,n)

Descrição:
gray2ind coloca em escala e arredonda uma Imagem de Intensidades I produzindo uma Imagem Indexada equivalente.
[X,map] = gray2ind(I,n)
converte a Imagem de Intensidades I para uma Imagem Indexada X, com um mapa de n cores, indexada pelo parâmetro map. Se n for omitido, o default é 64. O valor de n deve ser um inteiro entre 1 e 65536.

E_2: Converter a Imagem de Intensidades anterior em uma Imagem Indexada. Utilizar o "Data Cursor" para verificar os resultados. Responder os itens de a) até f).

[X,map] = gray2ind(IE,25)
imshow(X,map)

a) Qual a diferença entre uma Imagem de Intensidades e uma Imagem Indexada?
b) Porque a Imagem Indexada convertida continua em escala de cinza?

[X,map] = gray2ind(IE)
imshow(X,map)

c) Qual a diferença entre a Imagem Indexada anterior e esta?
d) O que aconteceu com a visualização da Imagem? Porque?
e) Qual a função do parâmetro map?

[X,map] = gray2ind(IE,255)
imshow(X)

f) Qual a diferença entre as três visualizações das Imagens Indexadas?

8) Convertendo uma Imagem de Intensidades para uma Imagem Binária.

Sintaxe:
G = im2bw(f,T)

Descrição:
Gera uma Imagem Binária através de Limiarização ("Thresholding"). Os valores serão zero para intensidades menores que T e um para os outros pixels. O valor especificado para T deve estar no intervalo [0,1] independente da classe dos dados de entrada. A Imagem Binária de saída será da classe logical. (Se T for omitido será considerado T=0.5)

T_8: Converter a Imagem de Intensidades IE para uma imagem Binária.

BE = im2bw(IE);
imshow(BE)
whos IE, whos BE

Variar o valor de T e verificar o resultado.

E_3: Converter a Matriz F abaixo:

a) Para Imagem de Intensidade com valores no intervalo [0, 255].
b) Para Imagem de Intensidade com valores no intervalo [0, 1].
c) Para Imagem Binária com Limiar de 30%.

F = [-0.5 -0.5 -0.2 0.2 0.9 1.2 1.5 ; -0.8 0.7 0.72 0.2 0.9 1.2 1.5; -0.5 0.5 0.2 0.2 0.9 1.2 1.5; -0.5 0.45 0.2 0.2 0.9 1.2 1.5; -0.5 -0.5 0.42 0.2 0.49 1.2 1.5]

d) Qual a diferença entre as duas conversões abaixo?

- *G1 = mat2gray(F)*
- *G2 = im2double(im2uint8(F))*

9) Lendo Imagens de Arquivos.

Imagens podem ser lidas pelo MatLab através da função *imread:*

Sintaxe:
F = imread('arquivo.tipo')

A Figura 5 mostra os tipos de arquivos de imagens suportados pelo Toolbox de Processamento de Imagens do MatLab e as variantes de cada um.

Format	Full Name	Variants
'bmp'	Windows Bitmap (BMP)	1-bit, 4-bit, 8-bit, 16-bit, 24-bit, and 32-bit uncompressed images and 4-bit and 8-bit run-length encoded (RLE) images
'cur'	Windows Cursor resources (CUR)	1-bit, 4-bit, and 8-bit uncompressed images
'gif'	Graphics Interchange Format (GIF)	1-bit to 8-bit images
'hdf'	Hierarchical Data Format (HDF)	8-bit raster image data sets, with or without an associated colormap, and 24-bit raster image data sets
'ico'	Windows Icon resources (ICO)	1-bit, 4-bit, and 8-bit uncompressed images
'jpg' or 'jpeg'	Joint Photographic Experts Group (JPEG)	Any baseline JPEG image or JPEG image with some commonly used extensions, including: **Image Type** **Bitdepth** **Compression** grayscale 8- or 12-bit lossy grayscale 8-, 12-, or 16-bit lossless RGB 24- and 36-bit lossy or lossless
'pbm'	Portable Bitmap (PBM)	1-bit images using either raw (binary) or ASCII (plain) encoding
'pcx'	Windows Paintbrush (PCX)	1-bit, 8-bit, and 24-bit images
'pgm'	Portable Graymap (PGM)	ASCII (plain) encoding with arbitrary color depth, or raw (binary) encoding with up to 16 bits per gray value
'png'	Portable Network Graphics (PNG)	1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit indexed images; and 24-bit and 48-bit RGB images
'pnm'	Portable Anymap (PNM)	PNM is not a file format itself. It is a common name for any of the other three members of the Portable Bitmap family of image formats: Portable Bitmap (PBM), Portable Graymap (PGM) and Portable Pixel Map (PPM).
'ppm'	Portable Pixmap (PPM)	ASCII (plain) encoding with arbitrary color depth or raw (binary) encoding with up to 16 bits per color component
'ras'	Sun Raster (RAS)	1-bit bitmap, 8-bit indexed, 24-bit true color and 32-bit true color with alpha data
'tif' or 'tiff'	Tagged Image File Format (TIFF)	Any baseline image, including 1-bit, 8-bit, and 24-bit uncompressed images; 1-bit, 8-bit, and 24-bit images with packbits compression; 1-bit images with CCITT compression; and 16-bit grayscale, 16-bit indexed, and 48-bit RGB images
'xwd'	X Windows Dump (XWD)	1-bit and 8-bit ZPixmaps, XYBitmaps, and 1-bit XYPixmaps

Figura 5 - Tipos de arquivos suportados

T_9: Ler e mostrar imagens verificando informações sobre elas.

(Obs: Se o arquivo não estiver no diretório de trabalho é preciso informar o caminho, ou salvar a imagem no diretório de trabalho)

f = imread('chestxray_gray.jpg');

(Obs: o ponto e vírgula (;) impede que os valores sejam mostrados na Janela de Comando do MatLab)

Verificar o número de linhas e colunas da imagem:

size(f)

Atribuir os valores das linhas e colunas às variáveis M e N:

[M,N] = size(f)

Mostrar as informações da Matriz (Imagem):

whos f

Mostrar a Imagem:

imshow(f)

Ler uma segunda imagem de arquivo e mostrar as duas imagens lidas:

g = imread('rose_gray.tif');

whos g

imshow(f), figure, imshow(g)

Verificar o valor dos níveis nas coordenadas e a distância entre dois pixels:

pixval

(Obs: clicar com o botão direito do mouse sobre um pixel da figura, manter seguro e posicionar sobre outro pixel qualquer. O valor obtido é a distância entre os dois pixels).

10) Gravando Imagens em Arquivos.

Sintaxe:
imwrite(f, 'arquivo.tipo')
Descrição:
A imagem *f* é salva em disco, no *arquivo.tipo*.
Se a função não contiver o caminho, o arquivo é salvo no diretório de trabalho.

Uma sintaxe da função ***imwrite***, aplicável somente a imagens JPEG, é:
imwrite(f, 'arquivo.jpg', 'quality', q)

onde ***q*** é um inteiro entre 0 e 100. Quanto menor o valor de ***q*** maior a degradação devido à compressão JPEG.

E_4: Salvar Imagens JPEG em arquivos e calcular a Taxa de Compressão.

```
h = imread('bubbles.jpg');
imshow(h)
imfinfo bubbles.jpg
```

O Número de Bytes da imagem original é computado multiplicando-se Width x Height x BitDepth e dividindo-se o resultado por 8.
A Taxa de Compressão da imagem é obtida dividindo-se o Número de Bytes da imagem original pelo parâmetro FileSize.

a) Salvar a imagem *bubbles.jpg* com o nome de *bubbles5.jpg* , com q=5.
b) Ler a imagem *bubbles5.jpg* e gerar a informação sobre ela através da função *imfinfo bubbles5.jpg*
c) Mostrar as duas imagens, calcular a Taxa de Compressão de cada uma delas e concluir sobre o tamanho dos arquivos e sobre a qualidade de cada imagem.

Uma Variável de Estrutura pode ser usada para armazenar as informações da função *imfinfo* e assim calcular a Taxa de Compressão. Cada campo de informação é associado à Variável de Estrutura através de um ponto (.). Por exemplo, se a Variável de Estrutura for K, as informações de altura e largura da imagem serão K.Heigth e K.Width . Assim, a Taxa de Compressão da imagem pode ser calculada por:

```
K = imfinfo('bubbles.jpg');
image_bytes = K.Width*K.Height*K.BitDepth/8;
compress_ratio = image_bytes/K.FileSize
```

Uma sintaxe da função ***imwrite***, aplicável somente a imagens TIF, é:
imwrite(f, 'arquivo.tif', 'compression', 'parameter', 'resolution', [colres rowres])

onde:
'parameter' pode ter um dos seguintes valores:

- 'none' → indicando não compressão.
- 'packbits' → indicando compressão do tipo packbits (default para imagens não binárias).
- 'ccitt' → indicando compressão tipo ccitt (default para imagens binárias).

O array (1x2) ***[colres rowres]*** contém dois inteiros que fornecem a resolução das colunas e das linhas em dpi (dots-per-inch). Os valores defaults são [72 72]. Por exemplo, se as dimensões da imagem são em polegadas, ***colres*** é o número de pixels (ou pontos) por polegadas (dpi) na direção vertical, e ***rowres*** é a dpi na direção horizontal.

E_5: Salvar Imagens TIF em arquivos, alterando a resolução espacial.

O arquivo *circuit.jpg* é uma imagem em nível de cinza de 8 bits, 72 dpi de resolução espacial, tamanho de 450 x 450 píxels em uma dimensão de 6,25 x 6,25 polegadas. (res = 450/6,25 = 72 dpi)

Ler a imagem e verificar suas características:
f = imread('circuit.jpg');
imfinfo circuit.jpg

Salvar a imagem no formato TIF, sem compressão, reduzindo sua dimensão para 2,5 x 2,5 polegadas e mantendo seu tamanho em 450 x 450 píxels. Logo, a nova resolução espacial deverá ser: res2 = 450/2,5 = 180 dpi
Verificar suas características.

imwrite(f, 'circuit.tif', 'compression', 'none', 'resolution', [180 180])
imfinfo circuit.tif

Mostrar as duas imagens e responder:

a) O que aconteceu com as dimensões das imagens?
b) Explicar o que é Resolução Espacial da imagem.
c) Qual o tamanho dos dois arquivos em bytes?

11) Indexando Imagens.

Como imagens são matrizes, os esquemas de indexação de matrizes podem ser usados diretamente nas imagens.

T_10: Alterar imagens através da indexação.

O arquivo *rose_gray.tif* é uma imagem em nível de cinza de 8 bits, classe uint8, tamanho de 263 x 264 píxels. Digitar os comandos e verificar o que cada esquema de indexação faz..

```
f = imread('rose_gray.tif');
imshow(f)
whos f

fp = f(end:-1:1,:);
fl = f(:,end:-1:1);
imshow(f), figure, imshow(fp), figure, imshow(fl)

fc = f(65:198, 65:198);
fs = f(1:2:end, 1:2:end);
imshow(f), figure, imshow(fc), figure, imshow(fs)

plot(f(132,:))
```

12) Arranjos Padrões.

O MatLab pode gerar arranjos padrões que são úteis em diversas aplicações:

- zeros(M,N) → matriz de zeros da classe double.
- ones(M,N) → matriz de uns da classe double.
- true(M,N) → matriz de uns da classe logical.
- false(M,N) → matriz de zeros da classe logical.
- magic(M) → quadrado mágico MxM. A soma dos elementos ao longo de qualquer linha, qualquer coluna ou da diagonal principal é sempre a mesma.
- rand(M,N) → matriz de números aleatórios uniformemente distribuídos no intervalo [0,1].
- randn(M,N) → matriz de números aleatórios normalmente distribuídos (Gaussiana) com média 0 e variância 1.

T_11: Verificar os seguintes arranjos padrões gerados pelo MatLab.

```
Z = zeros(5,5)
whos Z

U = ones(3,3)
whos U

M = magic(4)
whos M
sum(M(:,1)), sum(M(3,:))

R1 = rand(4,4)
R2 = randn(4,4)
```