Image Processing Toolbox In Matlab

- A collection of functions built on Matlab's computing environment. The functions support
  - import and export images,
  - image display,
  - geometric operations,
  - neighborhood and block operations,
  - filtering and filter design,
  - analysis and enhancement,
  - binary image operations,
  - region of interest operations, and
  - image transform.

Understanding Images in Matlab

- Basic image data type in Matlab is a rectangular matrix, an order set of real picture elements (or pixels).

Resolution: 258×350

350 pixels

258 pixels

258×350 matrix

	1	2	3	…	350
1	0.72	0.82	0.66	…	0.60
2	0.77	0.77	0.77	…	0.60
⋮	⋮	⋮	⋮		⋮
258	0.34	0.28	0.49	…	0.82

Basic Matlab Operations On Images

Resolution: 258×350

load trees_gray;
X(2,3)=*0.77*
size (X) = *258, 350*

Image types

- Matlab supports four basic types of images:
  - Indexed images (typical color images)
  - Intensity images (gray-scale images)
  - Binary images (black and white)
  - RGB images (standard way of representing color data)

Indexed Images

- Typical color images require two matrices, a colormap and an image matrix:

- The colormap is an ordered set of values that represent the colors in the image. The size of the colormap is nx3 for an image containing n colors. Each row of the colormap matrix is a 1-by-3 red, green, blue (RGB) color vector

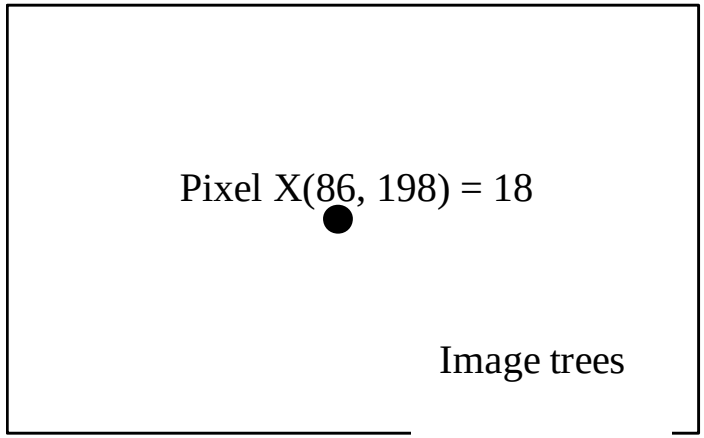
color = [R G B]

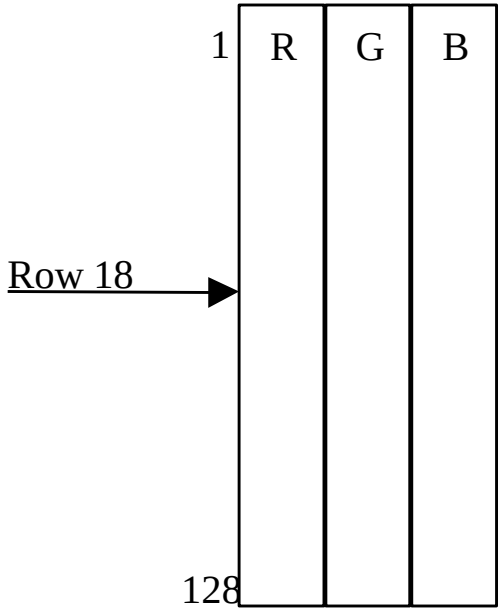
- R, G, B specifies the intensity of the red, green, and blue components of the color. R, G, and B are real scalars that range from 0.0 (black) to 1.0 (full intensity).

- For each image pixel, the image matrix contains a corresponding index in the colormap. When Matlab displays an indexed image, it uses the values in the image matrix to look up the desired color in the colormap.

Indexed Image (2)

```
load trees
% load the image data file
imshow (X, map)
% imshow command displays
 the image on the screen.
size (X) = 258   350
size(map) = 128   3
```



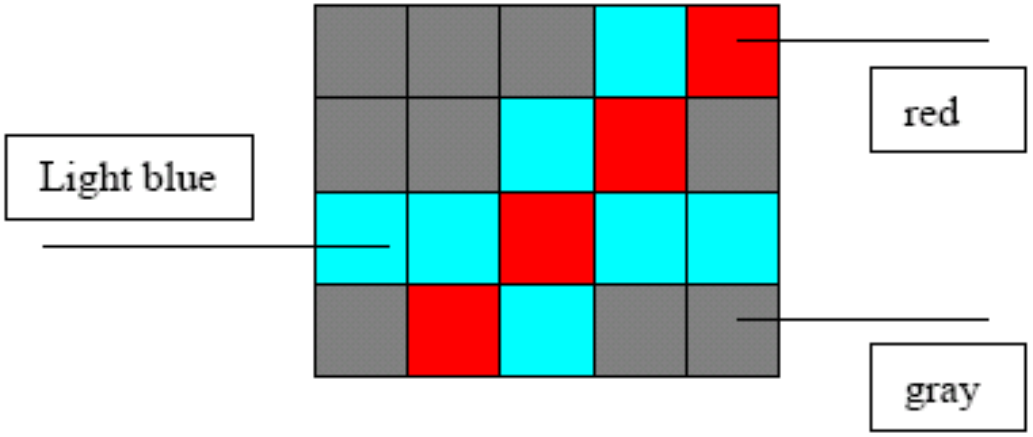


Colormap matrix

map(18,1:3) = 0.1608 0.3529 0.0627

Indexed Image (3)

```
X = [1 1 1 2 3;
     1 1 2 3 1;
     2 2 3 2 2;
     1 3 2 1 1 ];
```




```
map = [0.4, 0.4, 0.4;   % <- gray
       0   , 0.6, 1;    %<- light blue
       1   , 0   ,0];   %<- red

imshow (X, map)
```

Intensity Images

- Store an intensity image as a single matrix, which contains double precision values ranging from 0.0 to 1.0, with each element of the matrix corresponding to an image pixel.

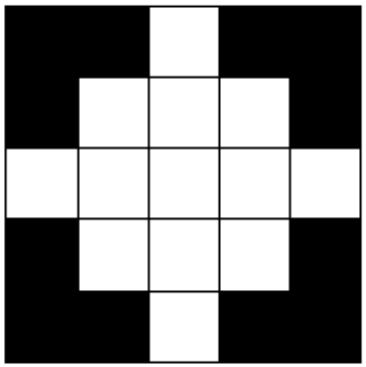
```
load trees
I =  ind2gray(X, map);
% convert indexed image to
gray scale.
imshow (I)
```

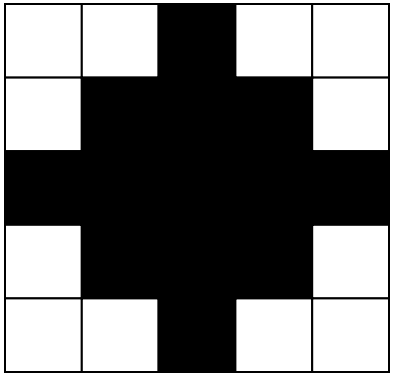
Binary Images

- A binary (black and white) image is a special kind of intensity image. Binary contain only two levels, 0(black) or 1(white).

```
BW =   [0 0 1 0 0
        0 1 1 1 0
        1 1 1 1 1
        0 1 1 1 0
        0 0 1 0 0 ];

imshow(BW)
```





imshow(~BW)

Binary Images(2)

```
load trees
I =  ind2gray(X, map);
BW = edge(I);                % command edge detects change in intensity
imshow(BW)
Imshow(~BW)                  %  ~ invert black to white and vice verse.
```

RGB Images

- In Matlab, the red, green, and blue components of an RGB image reside in three separate intensity matrices
- Each matrix has the same row and column dimensions as the original RGB image.
- The intensities of corresponding pixels from each matrix combine to create the actual pixel color at a given location.

RGB Images (2)

load trees
RGB= ind2rgb(X, map);
imshow(RGB)

```
R = [ .4  .4   0  .4  1
      .4  .4   0  1  .4
       0   0   1  0  0
       .4  1   0  .4  .4 ];
```

```
B = [ .4  .4   1  .4  0
      .4  .4   1  0  .4
       1   1   0  1  1
       .4  0   1  .4  .4 ];
```

```
G = [ .4  .4  .6  .4  0
      .4  .4  .6  0  .4
      .6  .6  0  .6  .6
       .4  0  .6  .4  .4 ];
```

RGB(:,:,1)=R;
RGB(:,:,2)=G;
RGB(:,:,3)=B;
Imshow (RGB)

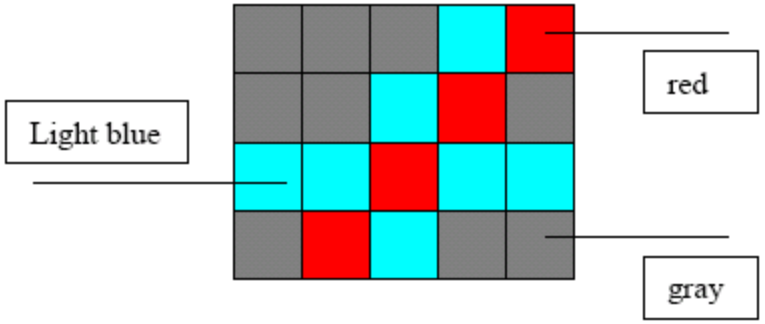
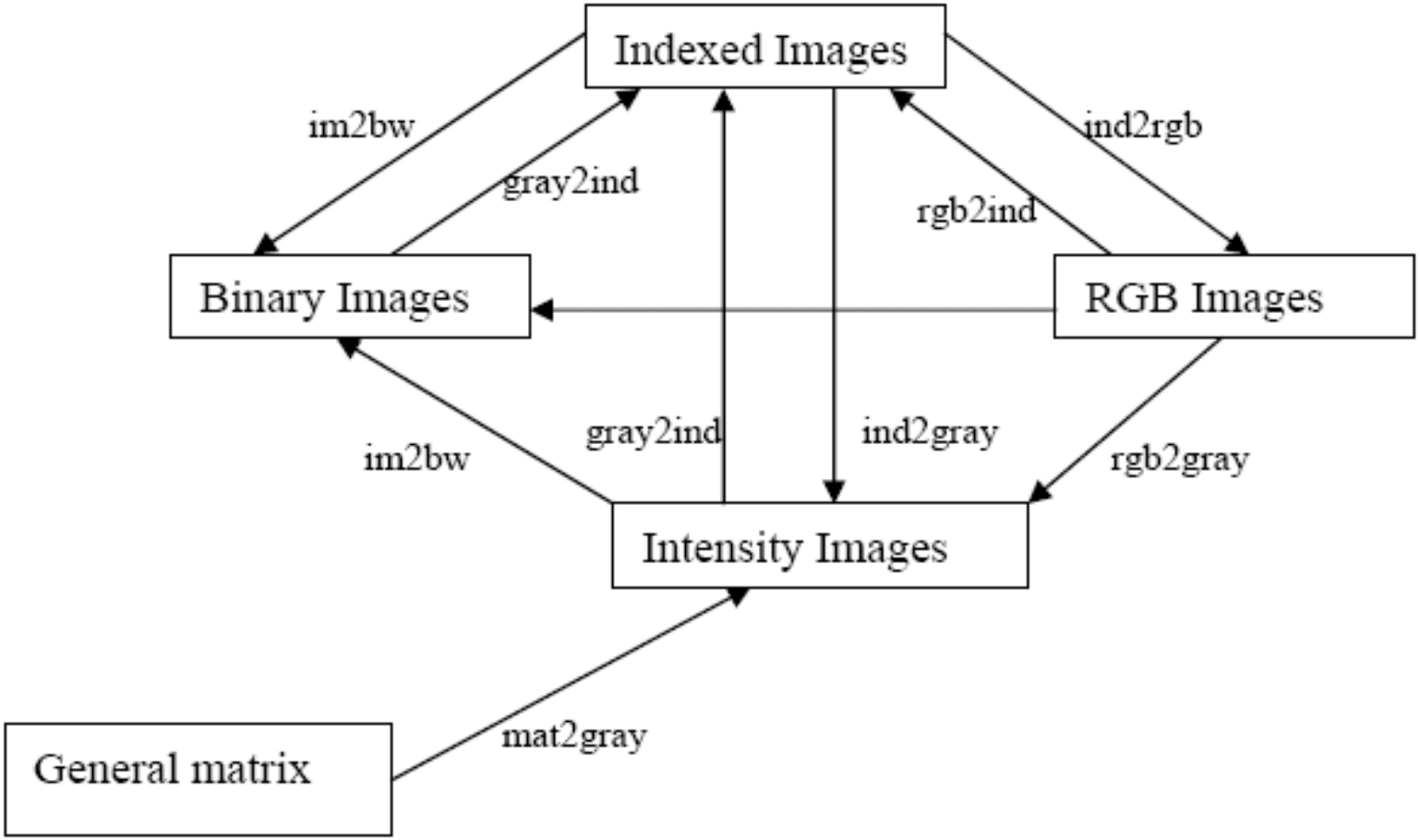


Image Types Converstion



Importing and Exporting Images

- The Matlab *imread* function can read the following graphics file formats
- To write image data from Matlab to a file, use the *imwrite* function

Format Name	Description	Recognized Extensions
TIFF	Tagged Image File Format	`.tif, .tiff`
JPEG	Joint Photographic Experts Group	`.jpg, .jpeg`
GIF	Graphics Interchange Format†	`.gif`
BMP	Windows Bitmap	`.bmp`
PNG	Portable Network Graphics	`.png`
XWD	X Window Dump	`.xwd`

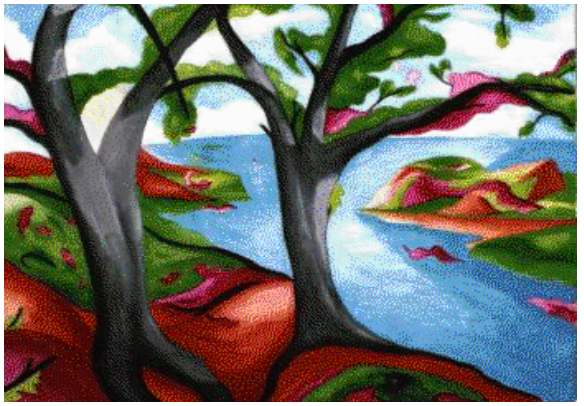
† `GIF` is supported by `imread`, but not by `imwrite`.

Importing and Exporting Images(2)

Image Format	Matlab commands	Description
Graphics Interchange Format (GIF)	[X, map]=gifread('img.gif'); gifwrite(X, map 'filename')	Read/write img.gif from/to disk and return/save as indexed image matrix/colormap. Matlab version 5.0 or higher is no longer supporting the 'gif" format.
Tagged Image File Format (TIFF)	[r,g,b]=tiffread('rgb.tiff'); tiffwrite(X, map, 'test')	Read rgb.tiff from disk and return the component matrices r, g, b. Write an indexed image X with a colormap named map to the file 'test'
MS Windows format (BMP)	[X, map]=bmpread('img.bmp'); bmpwrite(X, map 'filename')	Read/write img.bmp from/to disk and return/save as indexed image matrix/colormap.
Zsoft Paint format (PCX)	[X, map]=pcxread('img.pcx'); pcxwrite(X, map 'filename')	Read/write img.pcx from/to disk and return/save as indexed image matrix/colormap.
Matlab MAT-files	load filename save filename	Load and save Matlab general data files

Importing and Exporting Images(3)

```
[X,map]=imread('trees.tif');
% read the trees in tiff format to the memory (matlab workspace) as index image
imshow(X, map)
% once the image is on the MATLAB workspace, it can be shown on screen.
imwrite(X,map,'test.jpg')
% convert the image to jpg format and save it in a new file 'test.jpg.
% The colormap "map" is included in the imwrite command in order to retain the colormap.
imwrite(X,'testgray.jpg')
% convert the image to jpg format and save it in a new file 'testgray.jpg.
% if the colormap "map" is not included in the imwrite command, the image will be gray.
```

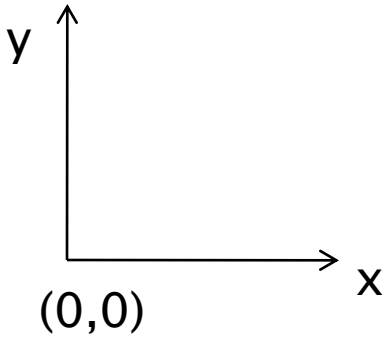


'test.jpg'

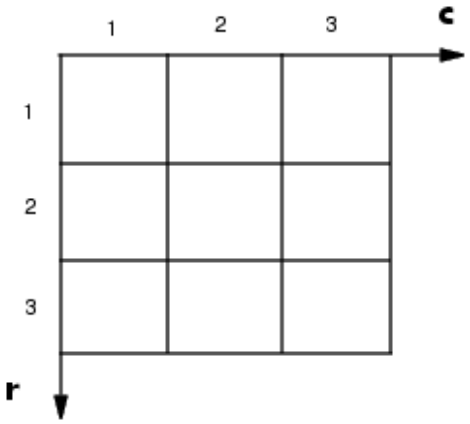
'testgray.jpg'

Coordinate Systems

- **Cartesian Coordinate System** – Original (0, 0) is in the bottom left corner.



- **Matrix Coordinate System** – row-column (i, j) coordinate system. The origin (1, 1) is in the upper left corner. Matrix coordinates are integer and range between 1 and the length of the row or column. The image processing toolbox uses this system for direct image matrix subscripting.



- **Pixel Coordinate System** – Same as Matrix coordinate system, most commonly used in image processing.

Geometric Operations

- Image Resize

The MATLAB command ***imresize*** resizes an image of any type using the specified interpolation method.

Supported interpolation methods include:
'nearest' (default) – nearest neighbor interpolation.
'bilinear' –bilinear interpolation
'bicubic' –bicubic interpolation

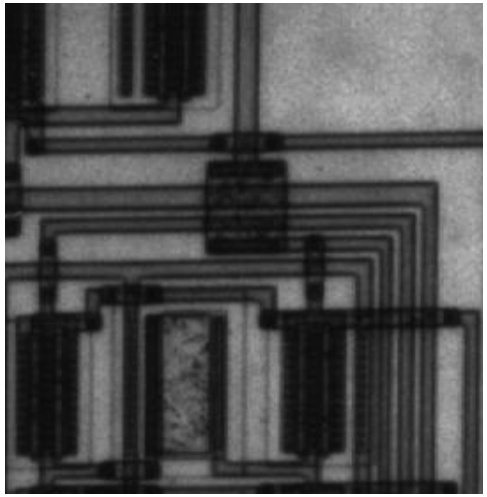
```
[X,map]=imread ('trees.tif');
[Y,map1]=imresize(X,map,2);
% double the size
[Z,map2]=imresize(X,map,[200 400]) ;
%resize the image to 200-by-400
```

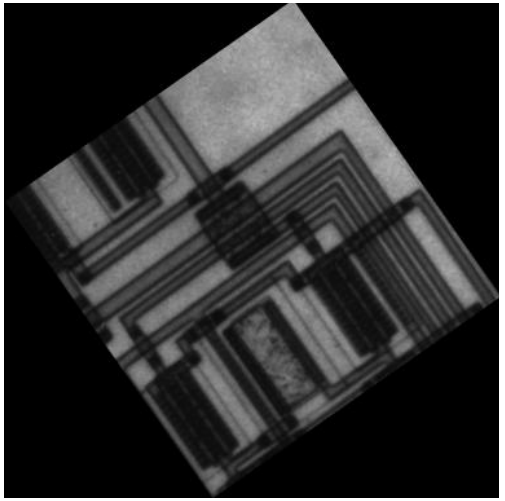
Geometric Operations(2)

- Image Rotation

  The MATLAB command ***imrotate*** rotate an image of any type counterclockwise Negative value rotates the image clockwise.

```
I= imread ('circuit.tif');
J = imrotate(I, 35);
% rotate then 35 degrees
Imshow(I)
figure, imshow(J);
```





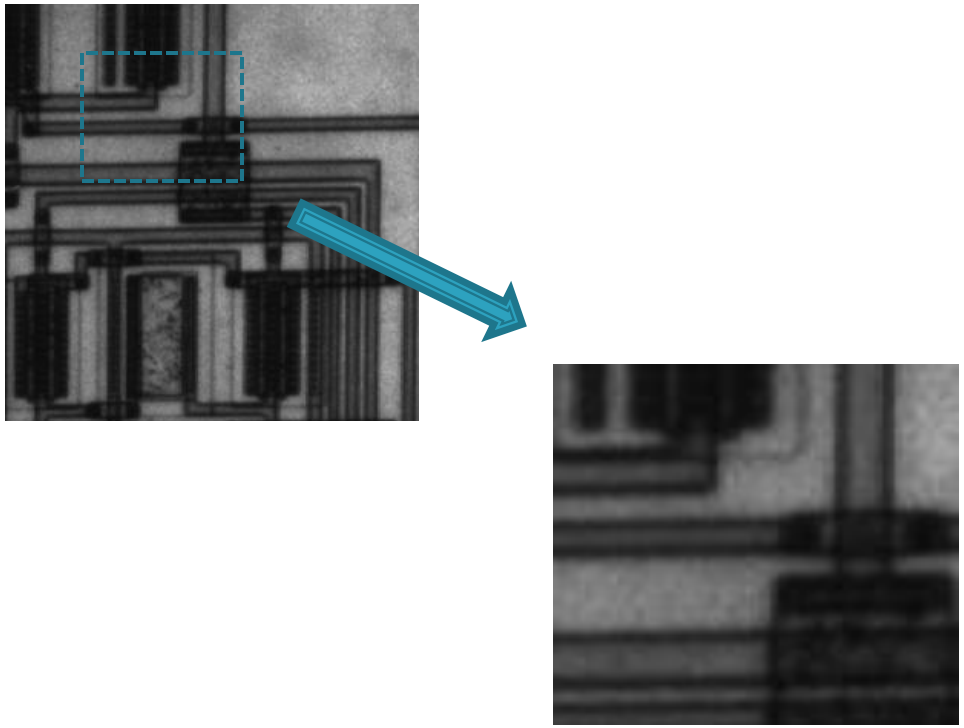
Geometric Operations(3)

- Cropping an Image

  The MATLAB command ***imcrop*** crops an image to a specified rectangle. In the syntaxes below, IMCROP displays the input image and waits for you to specify the crop rectangle with the mouse:

```
I = imread('circuit.tif');
I2 = imcrop(I,[60 40 100 90]);
imshow(I), figure, imshow(I2)
```

vector is in the form [XMIN YMIN WIDTH HEIGHT]; if not specified in the function, Matlab will display the input image and wait for you to specify the crop rectangle with the mouse:

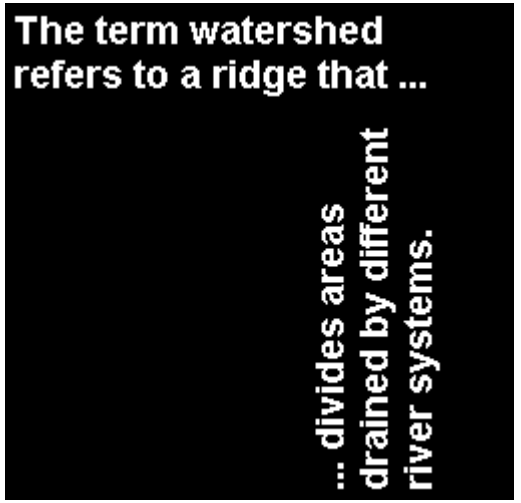


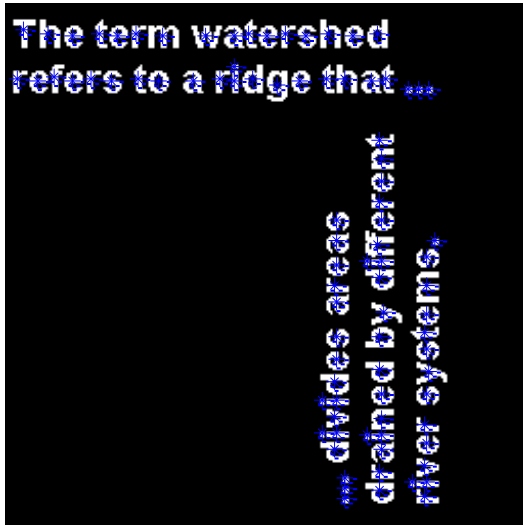
Geometric Operations(4)

- ***bwlabel*** and ***regionprops***

```
bw = imread('text.png');
[L,num] = bwlabel(bw);   %label
s  = regionprops(L, 'centroid');
centroids = cat(1, s.Centroid);
imshow(bw)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
```

Label the connected pixel components in the text.png image, compute their centroids, and superimpose the centroid locations on the image.

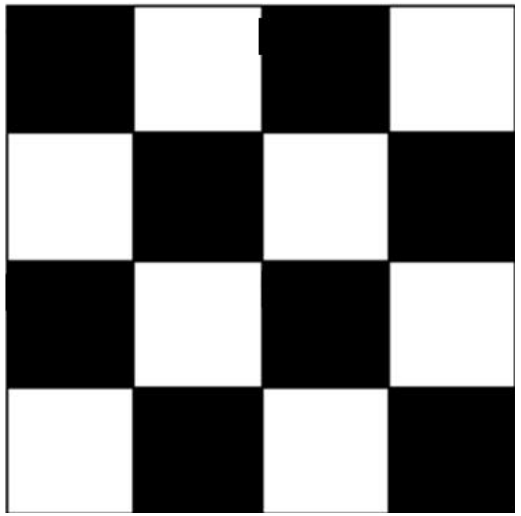




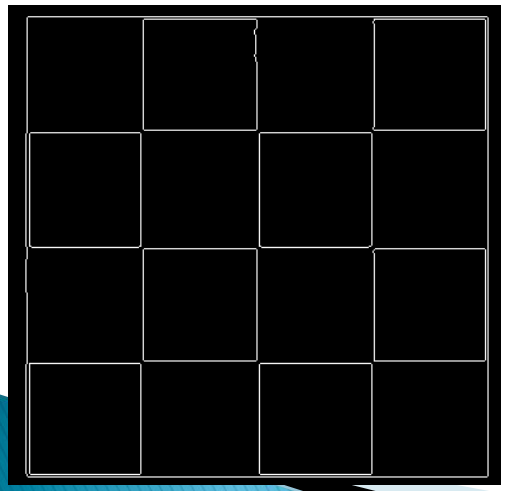
Geometric Operations (5)

- ***edge***

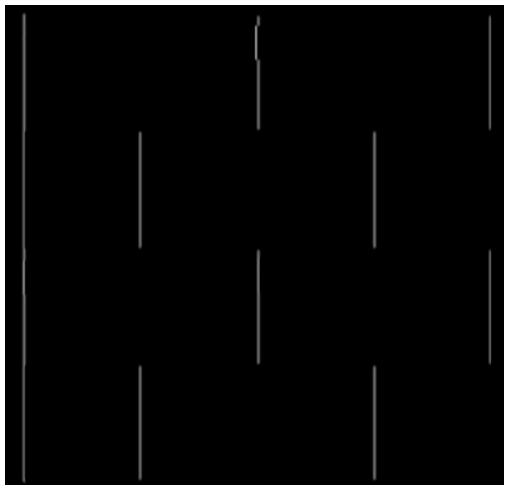
```
I=imread('checker.jpg');
 [BW,thresh,gv,gh]= edge(J,'sobel');
imshow(gv)
imshow(gh)
imshow(BW)
```



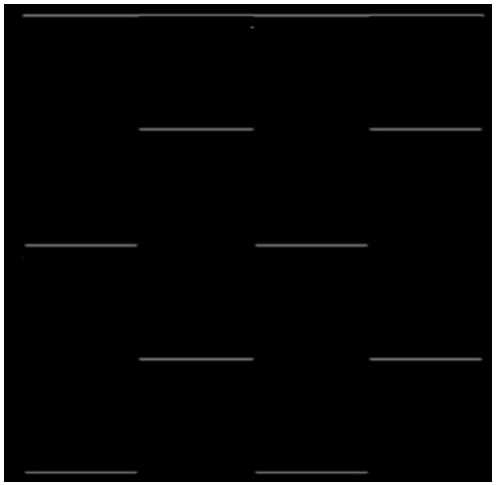
J



BW



gv



gh

Demos in Matlab

- Go to *Matlab->help->Demos->Toolbox->Image Processing* for more functions and their examples.