DIGITAL IMAGE PROCESSING

IN

MATLAB

Presented By :

Dr. V. Masilamani

IIITDM Kancheepuram, Chennai

Email : masila@iiitdm.ac.in

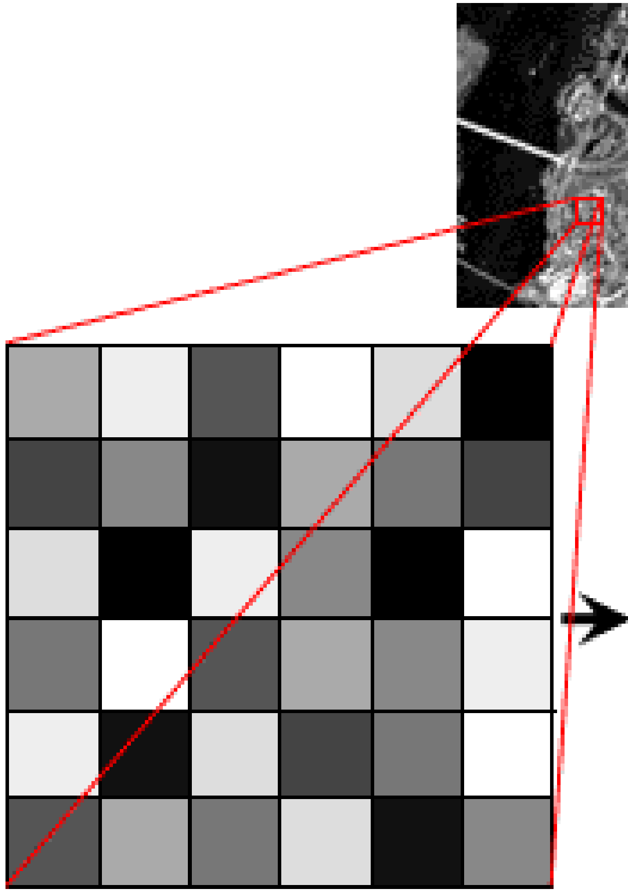
Introduction to Image Processing

- **Grayscale digital image :** Matrix of integers from 0 to L-1
- **Color digital image :** Three matrices of integers from 0 to L-1 each
- **Digital image processing** means applying sequence of operations on digital images using digital computer.
- **Major Applications :**
  - Improvement of pictorial information for better human perception
  - For autonomous machine applications
  - Efficient storage and transmission.

Introduction to Image Processing

- A digital image will be obtained from analog image by sampling and quantization.
- Sampling:
  - Representation of an image by 2-D matrix
- Quantization:
  - Each matrix element represented by one of the finite set of discrete values.

Introduction to Image Processing



170	238	85	255	221	0
68	136	17	170	119	68
221	0	238	136	0	255
119	255	85	170	136	238
238	17	221	68	119	255
85	170	119	221	17	136

Figure 1. Example of a digital image

Introduction to Image Processing

- A pixel is the smallest unit of a digital image or graphic that can be displayed and represented on a digital display device.
- Pixels are combined to form a complete image or video.
- Major type of digital images
  - Binary image: pixel values of a binary image will be either 0 or 1.
  - Grayscale image: pixels values will be depends on the number of bits used to represent one pixel. In general 8-bit grayscale representation is used, where the pixel values will fall into [0, 255].
  - Color image : color images will be represented using 3 different primary color planes, say R, G, and B. In general each pixel will be represented using 24 bits.

Introduction to Image Processing

(a) Binary image

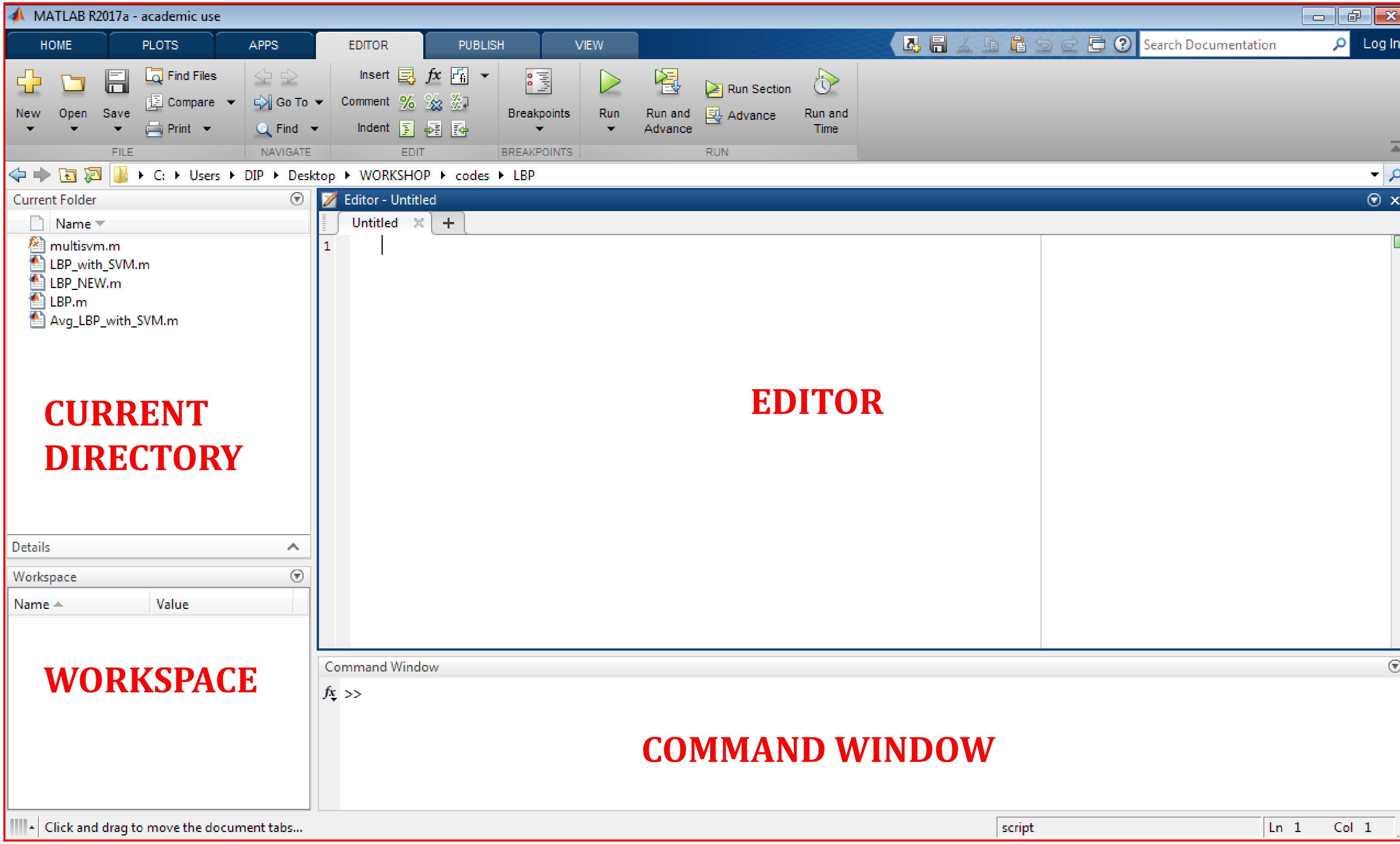
(b) Grayscale image

(c) Color image

Introduction to Matlab

- Matlab stands for Matrix Laboratory
- MATLAB features a family of application-specific solutions called toolboxes
- Most of the digital image processing solutions are available in image processing toolbox (IPT) and computer vision toolbox(CVT) of Matlab.
- IPT and CVT in Matlab provides operations like
  - Image enhancement
  - Noise reduction
  - Geometric transformations (rotation, scaling, etc.)
  - Image segmentation

Introduction to Matlab



Matlab inetrface

Introduction to Matlab

- Matlab stands for Matrix Laboratory
- All data is represented in the form of matrix.
- MATLAB features a family of application-specific solutions called toolboxes
- Most of the digital image processing solutions are available in image processing toolbox of Matlab.
- Image processing toolbox in Matlab provides operations like
    - Image enhancement
    - Noise reduction
    - Geometric transformations (rotation, scaling, etc.)
    - Image segmentation

Matrix Initilialization

- Matrix representation in matlab

```
>> a = [ 1 2 3 4 5; 2 3 4 5 6; 3 4 5 6 7; 4 5 6 7 8]
```

- To initialize a matrix of size 5X10 with all 0’s.

```
>> a=zeros(5, 10)
```

- To initialize a matrix of size 5X10 with all 1’s.

```
>> a=ones(5, 10)
```

- To initialize a matrix of size 5X5 with the integer 150

```
>> a=ones(5, 5).*150
```

- To linearize a matrix

```
>> [r c]=size(a);
>> b=reshape(a, [ 1 (r*c)]);
```

Loop Statements in Matlab

- **for**

Eg:-

```
sum=0;
for i=1:10
        sum=sum+i;        % Finding sum of first 10 natural numbers
end
```

- **while**

Eg:-

```
fact=1;
i=1;
while i<5
        fact=fact*i;        % Finding factorial 5
 i=i+1;
end
```

Conditional Statements in Matlab

if

Check given number is odd or even

Example 1:

```
x=input('Enter a number \n');
if mod(x,2)==0
        disp("Even No");
else
        disp("Odd No");
end
```

Check given number is negative or positive

Example 2:

```
x=input('Enter a number\n');
if x>0
        disp("Positive")
elseif x==0
        disp("Zero")
else
        disp("Negative");
end
```

Basic Statistics

- Mean
  - for 1-D case M = mean(A)
  - for 2-D case M = mean2(B)
- Median
  - M = median(A)
- Mode
  - M = mode(A)
- Variance
  - V = var(A)
- Standard Deviation
  - Column-wise std- y= std(A,0)
  - Row-wise std - y=std(A,1)
- Skewness
  - Column-wise skewness- y= skewness(A,0)
  - Row-wise skewness - y= skewness(A,1)
- Kurtosis
  - k = kurtosis(X)

Displaying an Image in Matlab

- **imshow**
  - syntax : imshow(filename/matrix_name)
  - example : imshow('cameraman.tif');
    imshow(A);

- **figure**
  - figure creates new figure window to open multiple images at a time
  - example : figure; imshow(A); title('First image');
    figure; imshow(B); title('Second image');

Displaying an Image in Matlab

- **imread**
  - syntax : Variable_Name=imread(filename);
  - example : A=imread('cameraman.tif');
    imshow(A);

Writing an Image in Matlab

- **imwrite**
  - syntax : imwrite(variable_name, filename, format)
  - example :

```
I=uint8(ones([100 100]).*150);
imwrite(I, 'writeimage.tif','tif');
R=imread('writeimage.tif');
imshow(R);
```

Image Formats in Matlab

BMP	Windows Bitmap
GIF	Graphics Interchange Format
HDF	Hierarchical Data Format
JPEG JPG	Joint Photographic Experts Group
JP2 JPF JPX J2C J2K	JPEG 2000
PBM	Portable Bitmap
PCX	Paintbrush
PGM	Portable Graymap
PNG	Portable Network Graphics
PNM	Portable Any Map
PPM	Portable Pixmap
RAS	Sun™ Raster
TIFF TIF	Tagged Image File Format
XWD	X Window Dump
CUR	Windows Cursor resources

Geometric Transformation of Images in Matlab

imresize

- syntax : imresize(variable_name, Scale)
- example :

```
I=imread('cameraman.tif');
B=imresize(I,0.5);
C=imresize(I,2);
figure;imshow(I);
figure;imshow(B);
figure;imshow(C);
```

(a) Original image

(b) scaled down

(b) scaled up

Geometric Transformation of Images in Matlab

- **imresize**
  - syntax : imresize(variable_name, Size)
  - example :

```
I=imread('cameraman.tif');
B=imresize(I, [200 200]);
figure;imshow(I);
figure;imshow(B);
```

(a) Original image

(b) Resized image

Geometric Transformation of Images in Matlab

imresize

- The interpolation techniques need to consider during scaling operation can also mention during usage of imresize function. Three interpolation techniques are available.
  - o nearest
  - o bilinear
  - o bicubic

- syntax : imresize(variable_name, size, method)
- example :

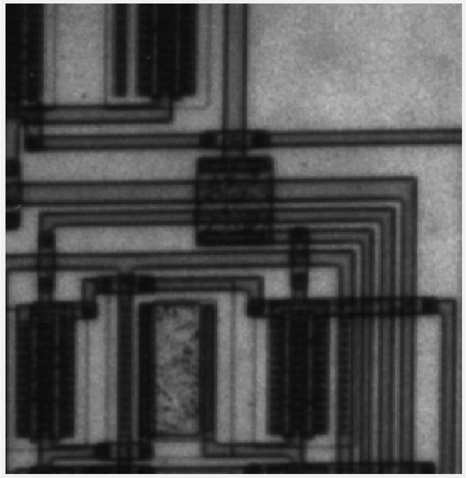
```
I=imread('cameraman.tif');
B=imresize(I, [200 200], ‘nearest’);
figure;imshow(B);
```

Geometric Transformation of Images in Matlab

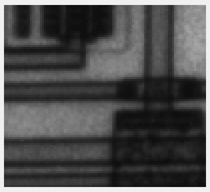
imcrop

- We can crop unwanted regions from image using imcrop function
- Syntax : imcrop(variable_name, rectangle_postion)
- Example :

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



(a) Original image



(b) Cropped image

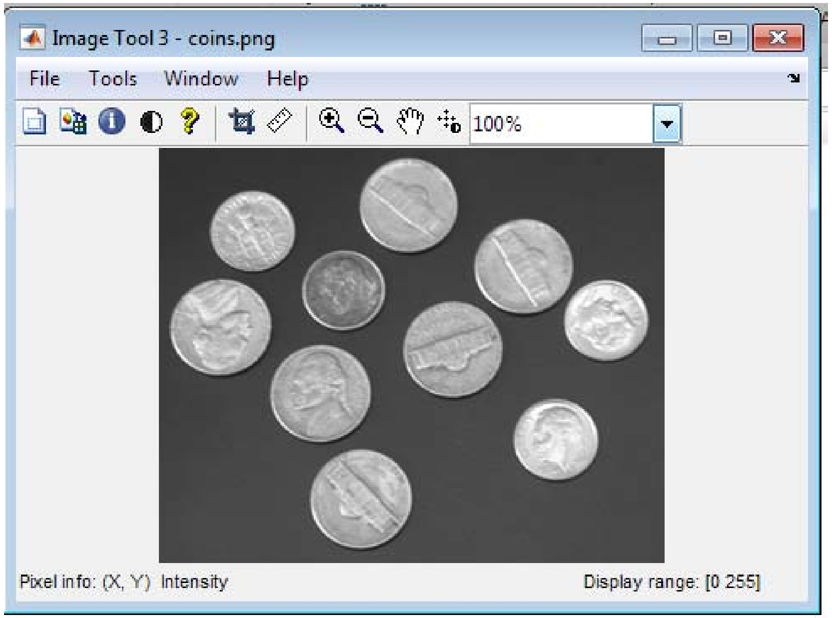
Display Images Using imtool

imtool

- imtool function can also used to display an image, where we can see the range of pixel values in the image, pixel value at each location, etc.
- syntax : imtool(variable_name);

  or

  imtool(file_name);
- example : imtool('coins.png');



Color Image Processing

- We can extract each color component of a given RGB image

Example :

```
I=imread('D:\images\peppers.jpg');
R=I(:,:,1);
G=I(:,:,2);
B=I(:,:,3);
figure; imshow(R);title('R Color Plane');
figure; imshow(G);title('G Color Plane');
figure; imshow(B);title('B Color Plane');
```

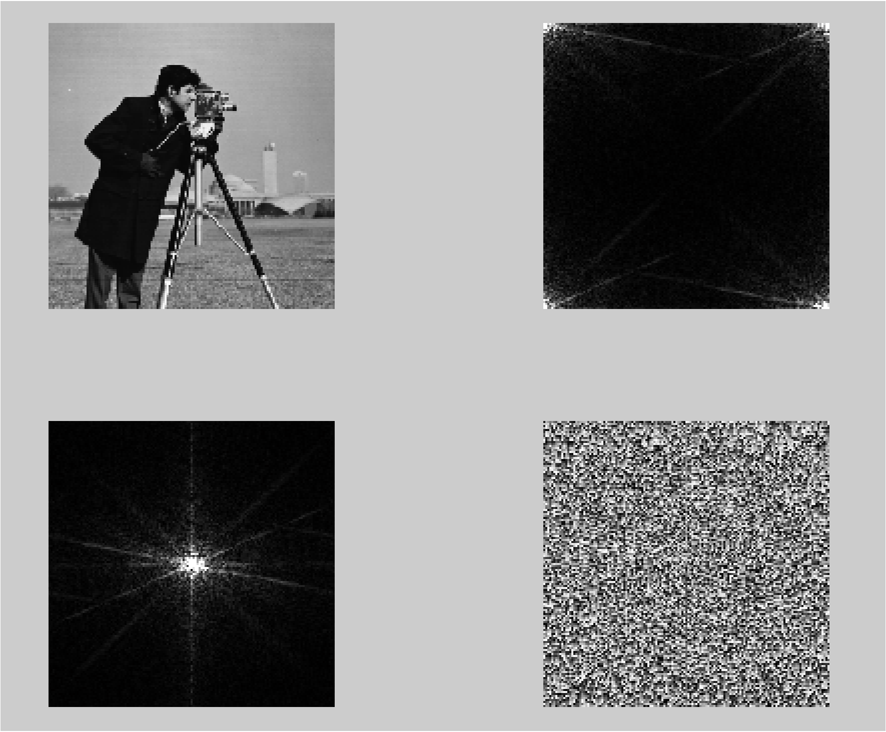
- To convert and RGB image to grayscale use rgb2gray and to convert grayscale image to binary use im2bw

```
I=imread('D:\images\peppers.jpg');
G=rgb2gray(I);
B=im2bw(G);
figure; imshow(G);title('Grayscale Version');
figure; imshow(B);title('Binary Image');
```

Frequency Domain

- fft2

```
Im  = imread('cameraman.tif');
F = fft2(double(Im));
figure;
subplot(2,2,1);
imshow(Im);
subplot(2,2,2);
imshow(abs(F),[24 100000]);
subplot(2,2,3);
imshow(abs(fftshift(F)),[24 100000]);
subplot(2,2,4);
imshow(angle(fftshift(F)),[-pi pi]);
```



Frequency Domain

- Swapping of phase and magnitude two images

```
I1 = imresize(imread('cameraman.tif'),[512 512]);
I2 = imresize(imread('pout.tif'),[512 512]);
F1 = fftshift(fft2(I1));
F2 = fftshift(fft2(I2));
R1 = abs(ifft2(ifftshift(abs(F1).*exp(j*angle(F2)))));
R2 = abs(ifft2(ifftshift(abs(F2).*exp(j*angle(F1)))));
figure;
subplot(2,2,1); imshow(I1);
subplot(2,2,2); imshow(I2);
subplot(2,2,3); imshow(R1,[]);
subplot(2,2,4); imshow(R2,[]);
```

Frequency Domain

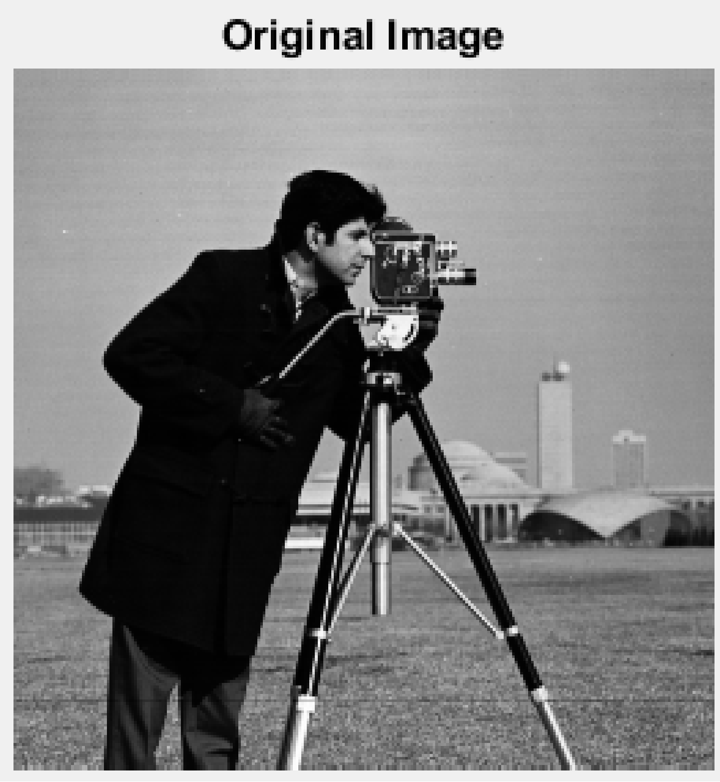
- Swapping of phase and magnitude two images

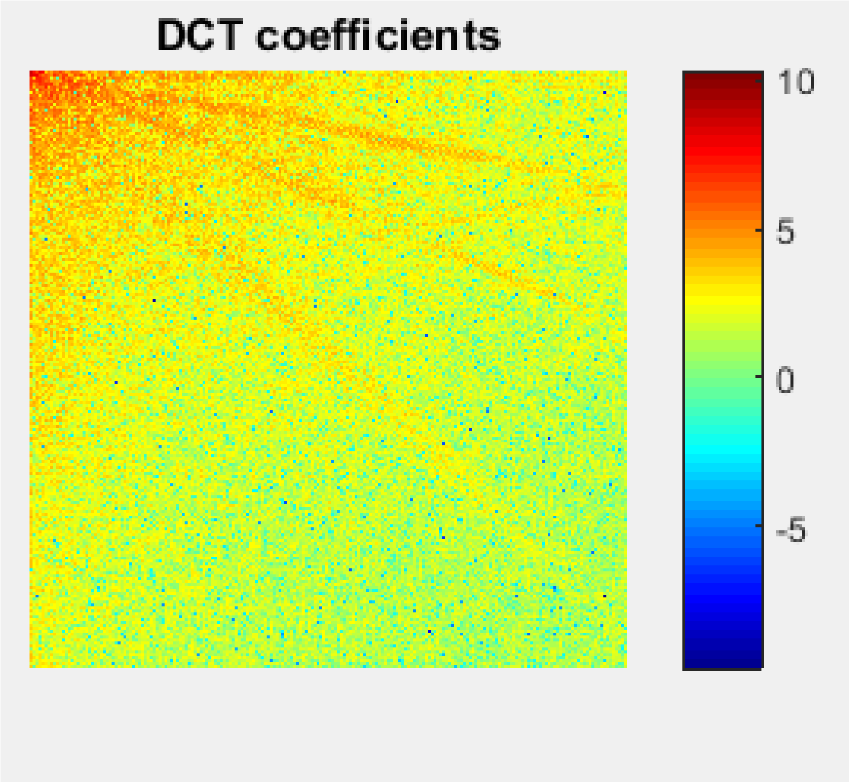
Discrete Cosine Transform

- B = dct2(A) returns the two-dimensional discrete cosine transform of A. The matrix B is the same size as A and contains the discrete cosine transform coefficients.
- Example

```
I = imread('cameraman.tif');
figure;imshow(I);title('Original Image');
J = dct2(I);
figure; imshow(log(abs(J)),[]); title('DCT coefficients');
colormap(gca, jet(64));        % gca handle for image shown in the last figure
colorbar;
```

Discrete Cosine Transform





Singular Value Decomposition (SVD)

- **svd**

Example :

A=[5, 6,2;4,7,2;5,5,6;6,4,8;]
[U S V]=svd(A);

```
U =

   -0.4351    0.4753    0.7442   -0.1757
   -0.4370    0.5976   -0.5438    0.3953
   -0.5255   -0.2088   -0.3501   -0.7468
   -0.5862   -0.6111    0.1668    0.5052
```

```
S =

   17.5073         0         0
         0    5.3428         0
         0         0    0.9735
         0         0         0
```

```
V =

   -0.5751    0.0105    0.8180
   -0.6078    0.6638   -0.4358
   -0.5476   -0.7479   -0.3753
```

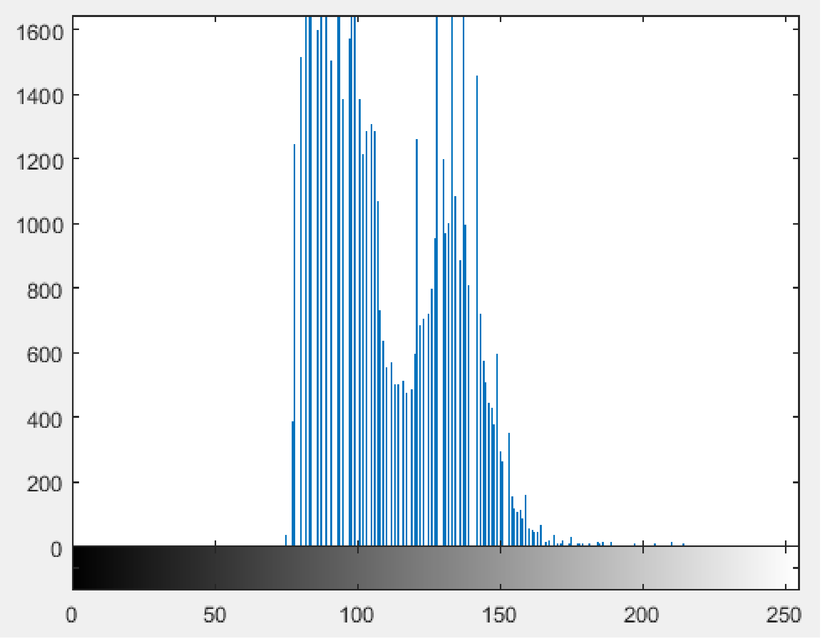
Histogram Processing

- To find histogram of an image use **imhist()**

Example :

```
I=imread('pout.tif');
figure; imshow(I);
figure;  imhist(I);
```

Image



Histogram

Image Enhancement

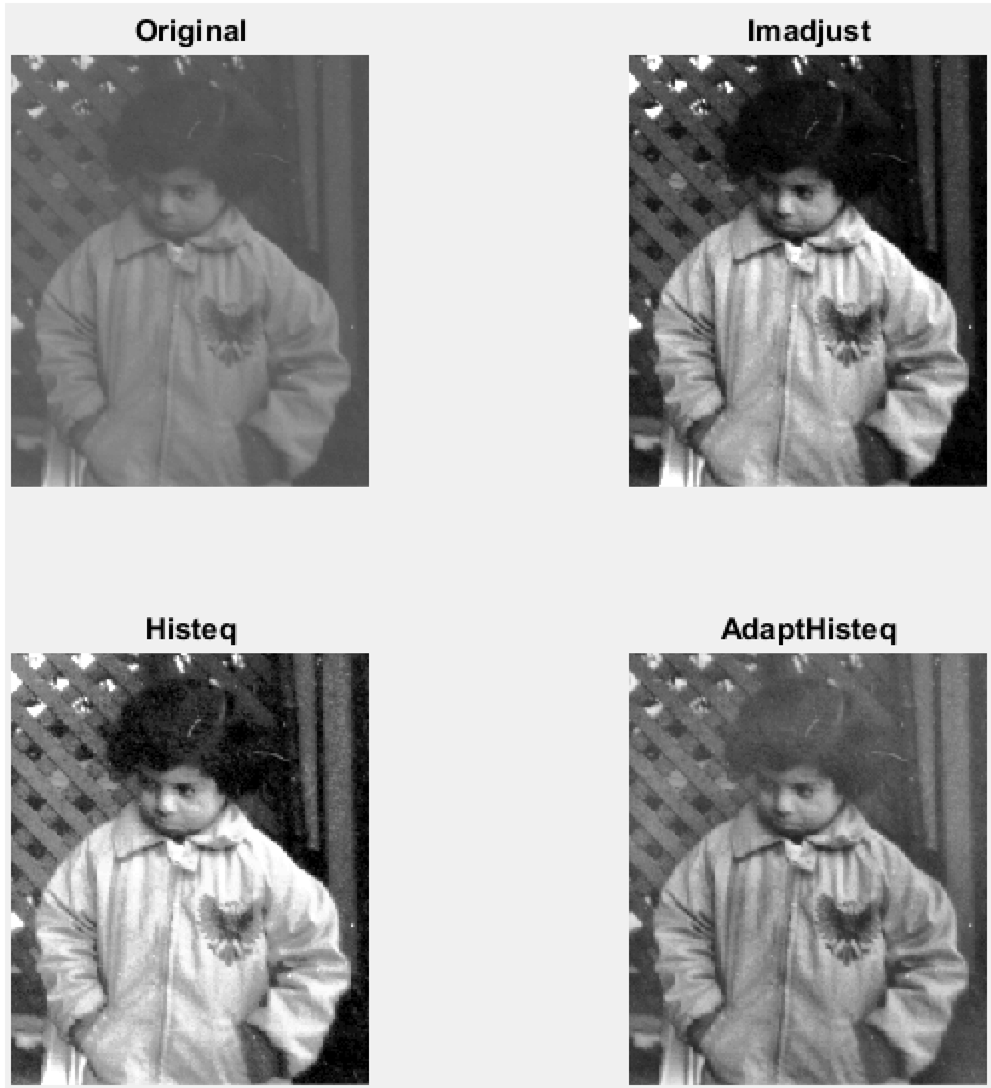
- **imadjust** increases the contrast of the image by mapping the values of the input intensity image to new values such that, by default, 1% of the data is saturated at low and high intensities of the input data.
- **histeq** performs histogram equalization. It enhances the contrast of images by transforming the values in an intensity image so that the histogram of the output image approximately matches a specified histogram (uniform distribution by default).
- **adapthisteq** performs contrast-limited adaptive histogram equalization. Unlike histeq, it operates on small data regions (tiles) rather than the entire image. Each tile's contrast is enhanced so that the histogram of each output region approximately matches the specified histogram (uniform distribution by default). The contrast enhancement can be limited in order to avoid amplifying the noise which might be present in the image.

Image Enhancement

- **imadjust, histeq, adapthisteq**

```
pout=imread('pout.tif');
pout_imadjust=imadjust(pout);
pout_histeq=histeq(pout);
pout_adapthisteq=adapthisteq(pout);
figure;
subplot(2,2,1);imshow(pout);title('Original');
subplot(2,2,2);imshow(pout_imadjust);title('Imadjust');
subplot(2,2,3);imshow(pout_histeq);title('Histeq');
subplot(2,2,4);imshow(pout_adapthisteq);title('AdaptHisteq');
```

Image Enhancement



Addition of Noise

- **imnoise :** Adding noise to image.

J = imnoise(I,TYPE,PARAMETERS) Add noise of a given TYPE to the intensity image I. TYPE is a string that can have one of these values:

- 'gaussian' Gaussian white noise with constant mean and variance
- 'localvar' Zero-mean Gaussian white noise with an intensity-dependent variance
- 'poisson' Poisson noise
- 'salt & pepper' "On and Off" pixels
- 'speckle' Multiplicative noise

Addition of Noise

- **salt & pepper**

```
I=imread('cameraman.tif');
N=imnoise(I,'salt & pepper', 0.01);
figure;
subplot(1,2,1);imshow(I);title('Original image');
subplot(1,2,2);imshow(N);title('Salt & Pepper Noise');
```

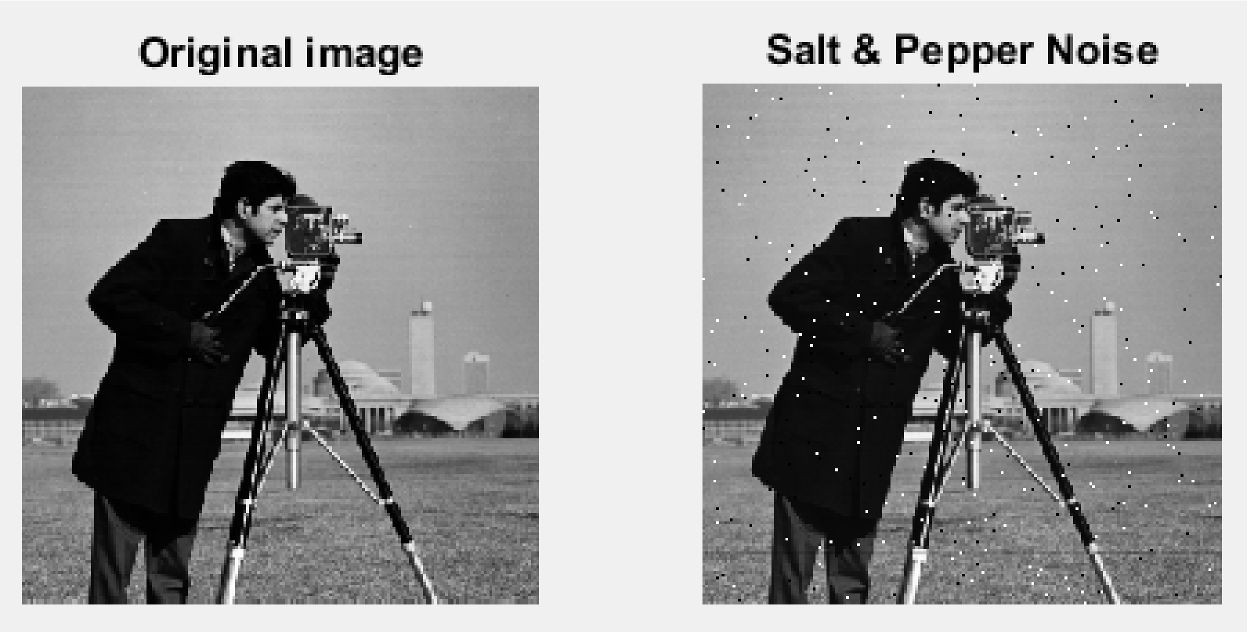


Image Filtering

- **medfilt2 :** for median filtering

```
I=imread('cameraman.tif');
N=imnoise(I,'salt & pepper', 0.01);
F=medfilt2(N, [5 5]);
figure;
subplot(1,3,1);imshow(I,[]);title('Original image');
subplot(1,3,2);imshow(N,[]);title('Salt & Pepper Noise');
subplot(1,3,3);imshow(F,[]);title('Filtered Image');
```

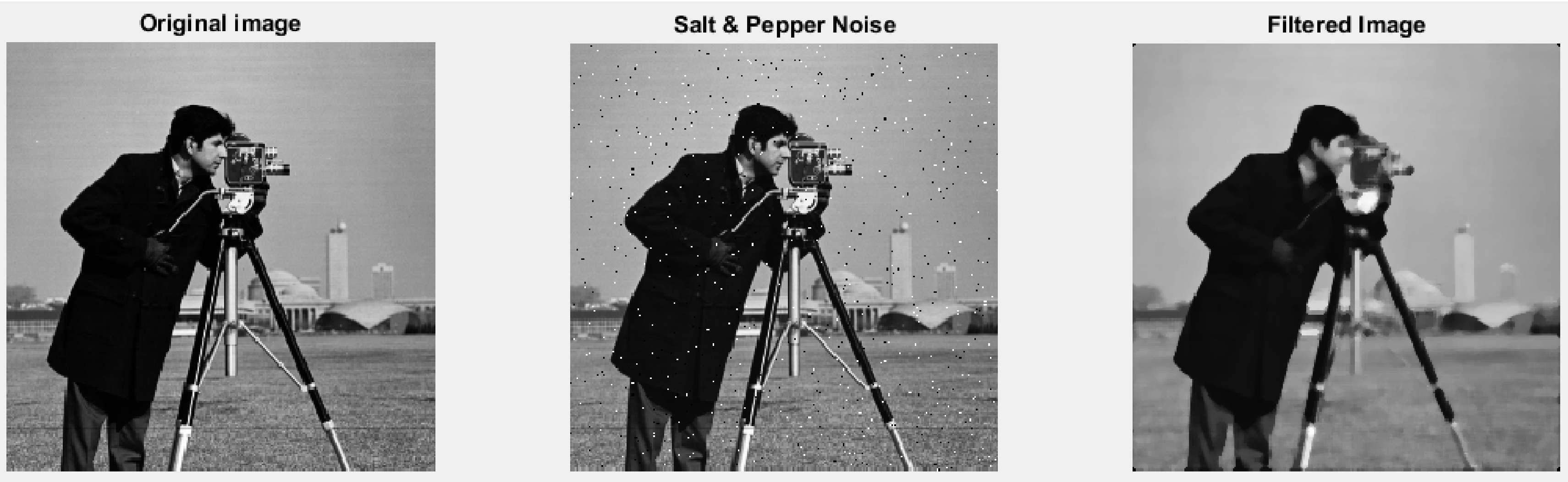


Image Filtering

- **wiener2 :** for wiener filtering

```
I=imread('cameraman.tif');
N=imnoise(I,'gaussian',0,0.025);
F=wiener2(N, [5 5]);
figure;
subplot(1,3,1);imshow(I,[]);title('Original Image');
subplot(1,3,2);imshow(N,[]);title('Gaussian Noise');
subplot(1,3,3);imshow(F,[]);title('Filtered Image');
```

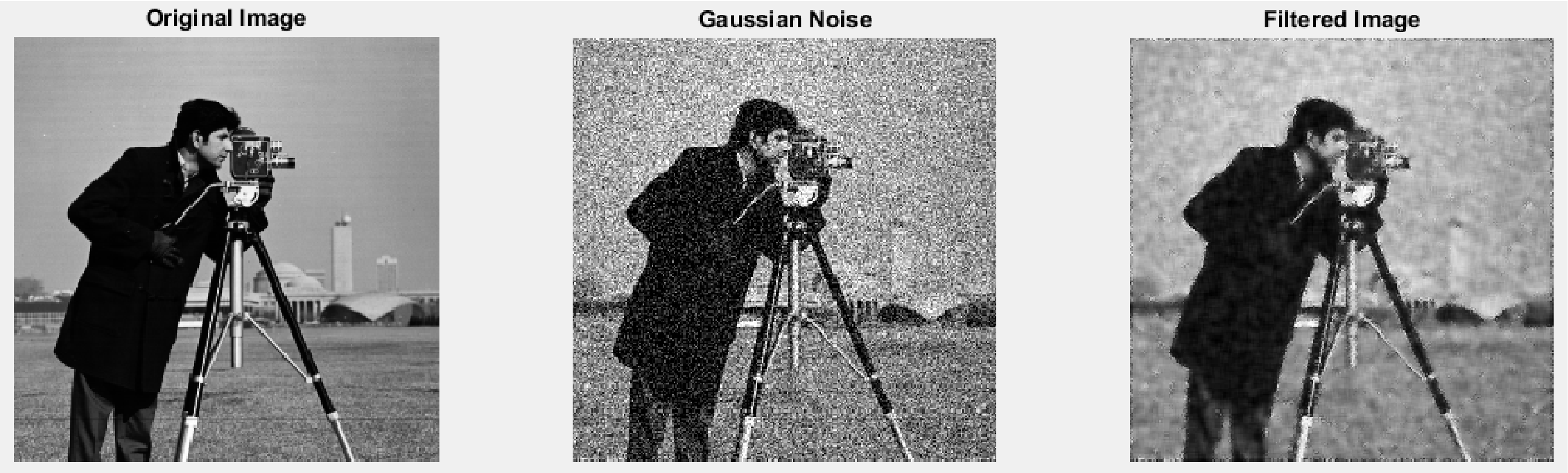


Image Filtering

- **fspecial :** To define new filter
  - syntax : fspecial(Type, Size)
  - Types :

'average'	:	averaging filter
'disk'	:	circular averaging filter
'gaussian'	:	Gaussian lowpass filter
'laplacian'	:	filter approximating the 2-D Laplacian operator
'log'	:	Laplacian of Gaussian filter
'motion'	:	motion filter
'prewitt'	:	Prewitt horizontal edge-emphasizing filter
'sobel'	:	Sobel horizontal edge-emphasizing filter

Image Filtering

- **fspecial :** To define new filter
  - Example 1 : **H=fspecial('average',[3 3]);**

This will create a filter H as follows :

$$H = \begin{matrix} 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \end{matrix}$$

- Example : **G=fspecial('gaussian',[5,5]);**

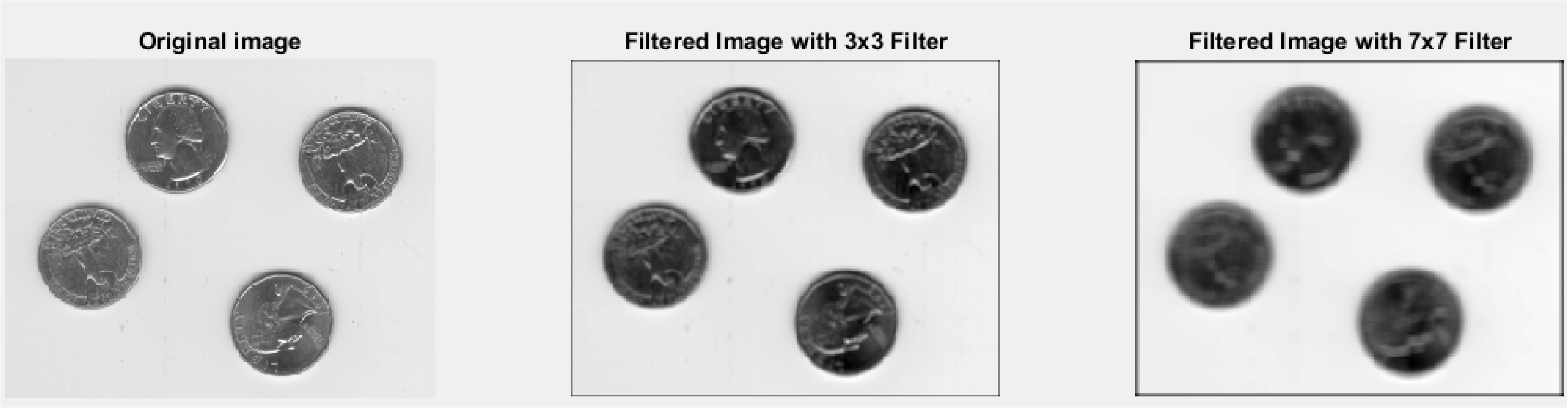
0.0000	0.0000	0.0002	0.0000	0.0000
0.0000	0.0113	0.0837	0.0113	0.0000
0.0002	0.0837	0.6187	0.0837	0.0002
0.0000	0.0113	0.0837	0.0113	0.0000
0.0000	0.0000	0.0002	0.0000	0.000

Image Filtering

- **imfilter :** To perform filtering operation with a specific filter

Example 1:

```
I = imread('eight.tif');
H1=fspecial('average', [3 3]);
H2=fspecial('average', [7 7]);
F1=imfilter(I, H1);
F2=imfilter(I, H2);
figure;
subplot(1,3,1); imshow(I); title('Original image');
subplot(1,3,2);imshow(F1,[]);title('Filtered Image with 3x3 Filter');
subplot(1,3,3);imshow(F2,[]);title('Filtered Image with 7x7 Filter');
```



Edge Detection

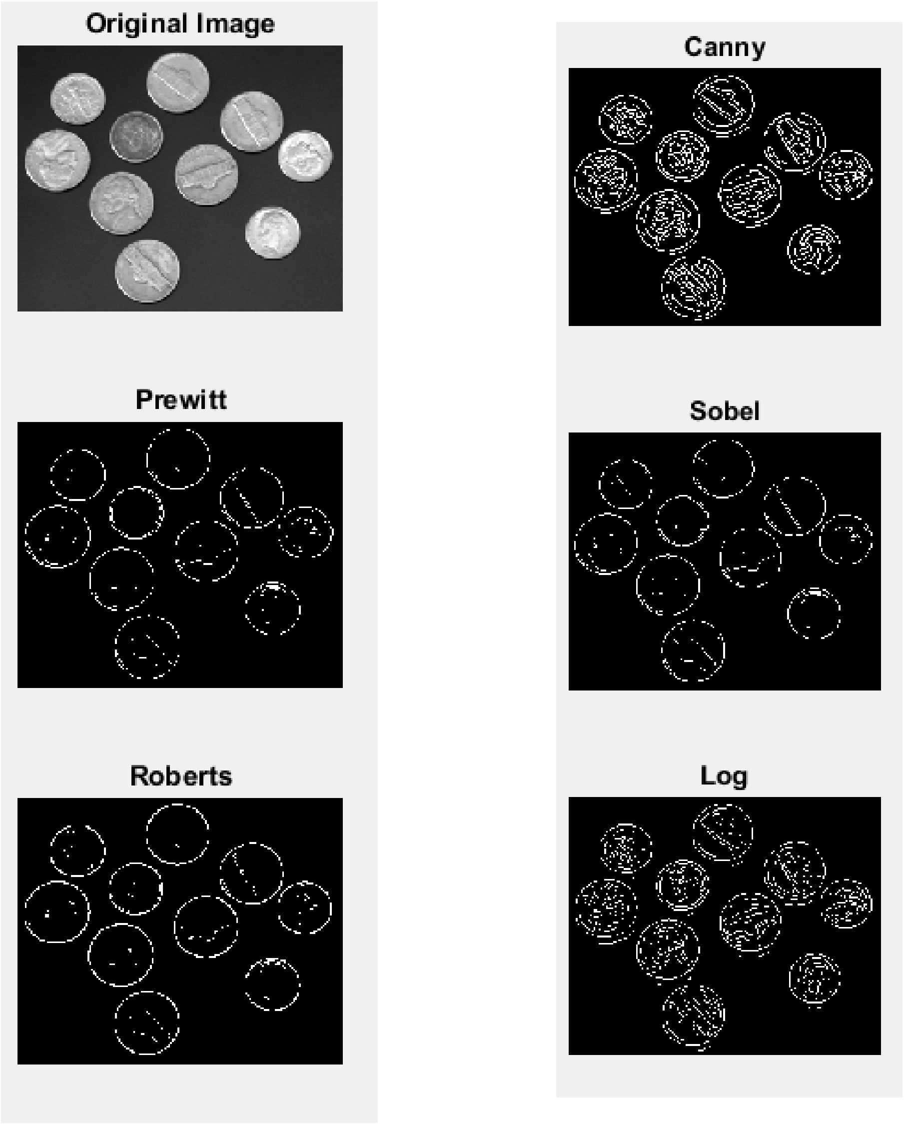
- **Edge detection** is an image processing technique for finding the boundaries of objects within images.
- It works by detecting discontinuities in brightness.
- Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision.

- **edge** Function finds edges in intensity image.
  - edge takes an image I as input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

Edge Detection

```
I=imread('coins.png');
E1=edge(I, 'canny');
E2=edge(I, 'prewitt');
E3=edge(I, 'sobel');
E4=edge(I, 'roberts');
E5=edge(I, 'log');
subplot(3,2,1);imshow(I,[]);title('Original Image');
subplot(3,2,2);imshow(E1,[]);title('Canny');
subplot(3,2,3);imshow(E2,[]);title('Prewitt');
subplot(3,2,4);imshow(E3,[]);title('Sobel');
subplot(3,2,5);imshow(E4,[]);title('Roberts');
subplot(3,2,6);imshow(E5,[]);title('Log');
```

Edge Detection



Morphological Operations

- Morphology means study of structure
- Finding structures, completing incomplete structure to be addressed
- Morphological operation will involve a small image called structuring element.

Morphological Operations

- **imdilate** and **imerode** are two commonly used morphologcal operators
- **IM2 = imerode(IM,SE)** erodes image IM with structuring element SE.
- The argument SE will be obtained by **strel** function.

```
se1 = strel('square',11)     % 11-by-11 square
se2 = strel('line',10,45)    % line, length 10, angle 45 degrees
se3 = strel('disk',15)       % disk, radius 15
se4 = strel('ball',15,5)     % ball, radius 15, height 5
```

Morphological Operations

Imerode and imdilate

```
I1=rgb2gray(imread('D:\images\morph1.bmp'));
I=im2bw(I1);
se=strel('square',10);
I2=imdilate(I,se);
I3=imerode(I2,se);
figure;
subplot(1,3,1);imshow(I);title('Step 1 : Original Image');
subplot(1,3,2);imshow(I2);title('Step 2 : Dilated Image');
subplot(1,3,3);imshow(I3);title('Step 3 : Eroded Image');
```

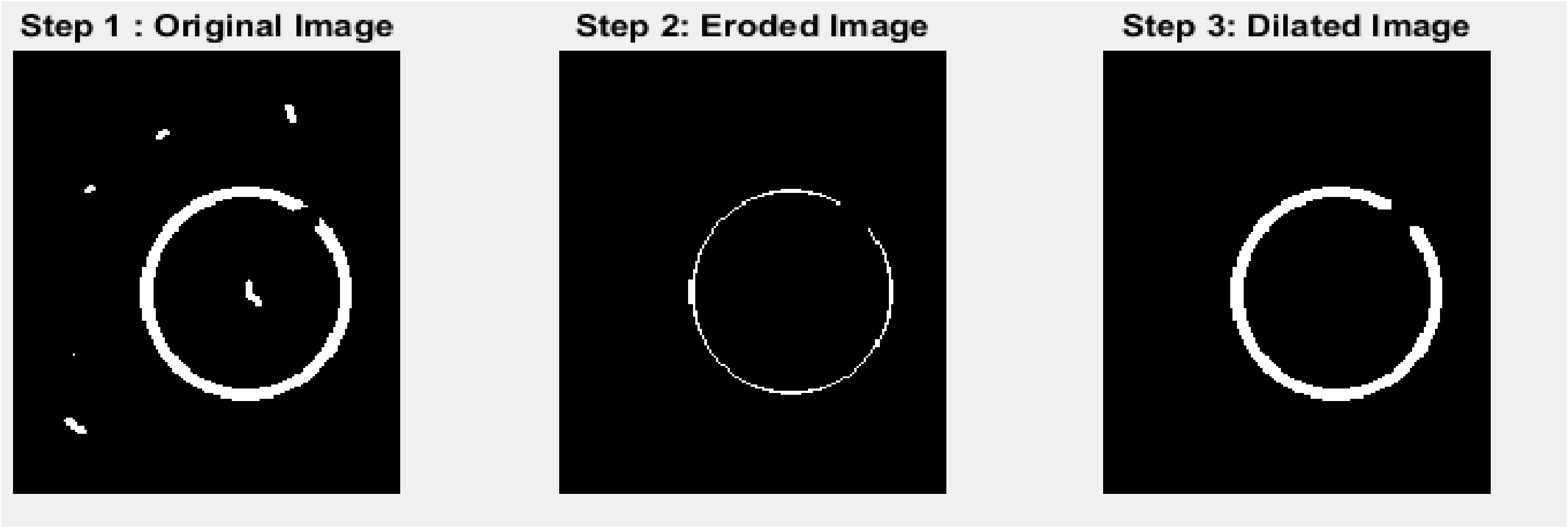
Morphological Operations

Imerode and imdilate

```
I1=rgb2gray(imread('D:\images\morph.bmp'));
I=im2bw(I1);
se=strel('square',4);
I2=imerode(I,se);
I3=imdilate(I2,se);
figure;
subplot(1,3,1);imshow(I);title('Step 1 : Original Image');
subplot(1,3,2);imshow(I2);title('Step 2: Eroded Image');
subplot(1,3,3);imshow(I3);title('Step 3: Dilated Image');
```

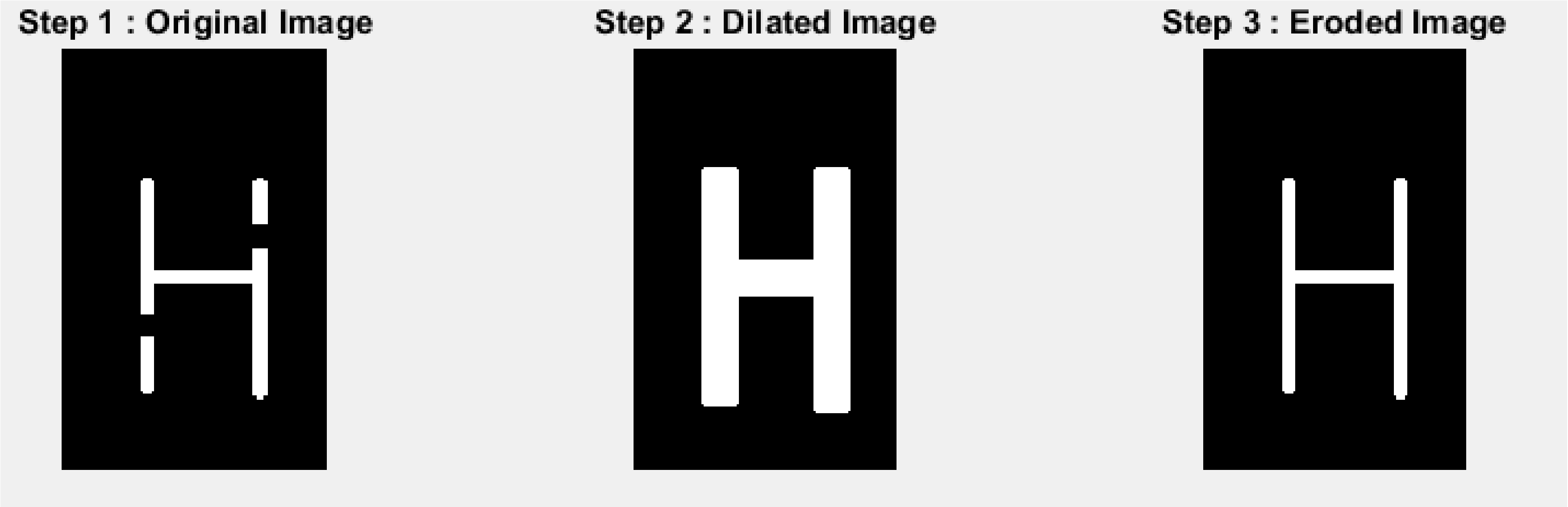
Morphological Operations

- **imerode and imdilate**



Morphological Operations

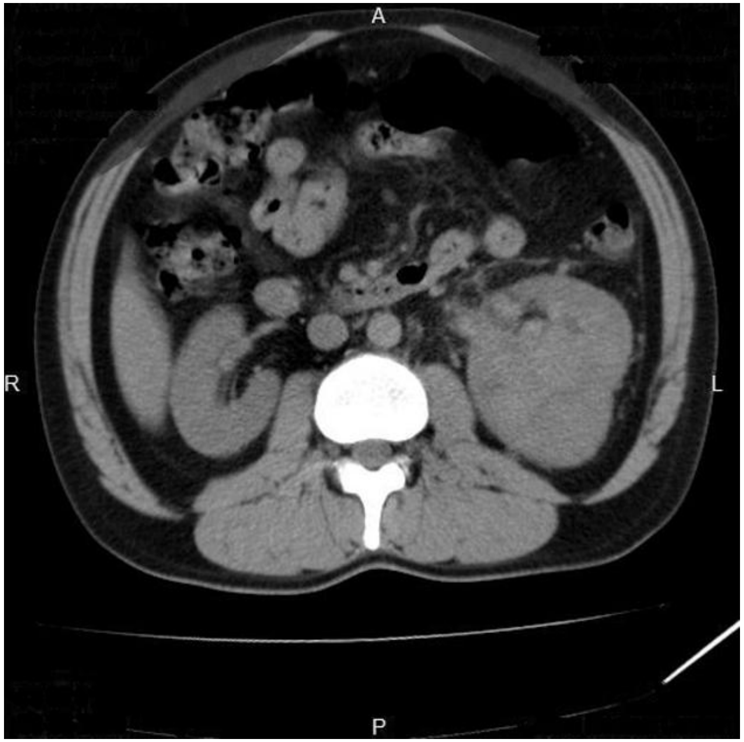
- **imerode and imdilate**



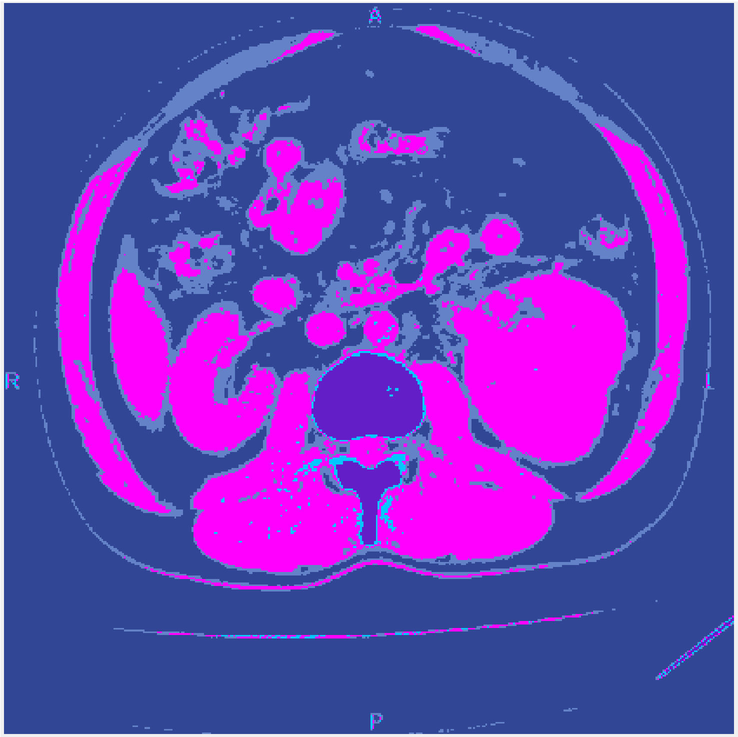
Gray to RGB Conversion for Better Visualization

```
I=imresize(imread('D:\images\brain.tif'),[512 512]);
R=uint8(zeros(512,512));
G=uint8(zeros(512,512));
B=uint8(zeros(512,512));
[R1 C1]=size(I);
for i=1:R1
   for j=1: C1
      if I(i,j)>230
         R(i,j)=100;        G(i,j)=30;         B(i,j)=200;
      elseif  I(i,j)>150
         R(i,j)=0;         G(i,j)=200;        B(i,j)=255;
     elseif  I(i,j)>100
         R(i,j)=255;        G(i,j)=0;         B(i,j)=255;
      elseif  I(i,j)>50
         R(i,j)=100;        G(i,j)=130;         B(i,j)=200;
      else
         R(i,j)=50;        G(i,j)=70;         B(i,j)=150;
      end
   end
end
ColorImage = cat(3,R,G,B);
figure;imshow(ColorImage);
```

Gray to RGB Conversion for Better Visualization



Original grascale image



Corresponding color image

Frame Extraction From Video

```
filename = 'rhinos.avi';
mov = VideoReader(filename);
opFolder = fullfile(cd, 'snaps');
if ~exist(opFolder, 'dir')
   mkdir(opFolder);
end
numFrames = mov.NumberOfFrames;
numFramesWritten = 0;
for t = 1 : 5: numFrames
currFrame = read(mov, t);
 opBaseFileName = sprintf('%d.png', t);
opFullFileName = fullfile(opFolder, opBaseFileName);
imwrite(currFrame, opFullFileName, 'png');   progIndication = sprintf('Wrote
frame %d of %d.', t, numFrames);
disp(progIndication);
numFramesWritten = numFramesWritten + 1;
end
progIndication = sprintf('Wrote %d frames to folder "%s"',numFramesWritten,
opFolder);
disp(progIndication);
```

Frame Extraction From Video






Detecting a Cell Using Image Segmentation

Step 1: Read Image

Step 2: Detect Entire Cell

Step 3: Dilate the Image

Step 4: Fill Interior Gaps

Step 5: Remove Connected Objects on Border

Step 6: Smoothen the Object

Detecting a Cell Using Image Segmentation

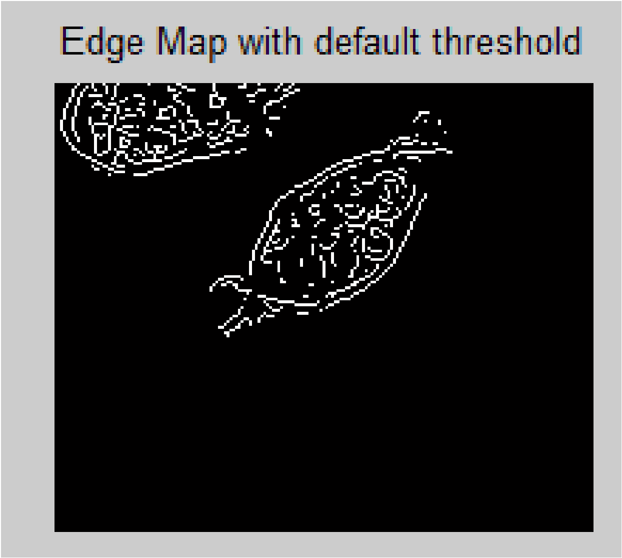
Step 1: Read Image

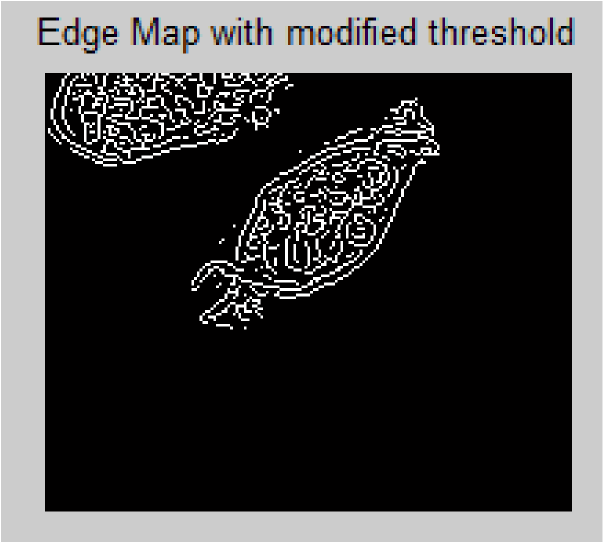
```
I = imread('cell.tif');
figure, imshow(I), title('original image');
```

Detecting a Cell Using Image Segmentation

Step 2: Detect Entire Cell

```
[BW, threshold] = edge(I, 'sobel');
fudgeFactor = .5;
BWs = edge(I,'sobel', threshold * fudgeFactor);
figure, imshow(BW), title('Edge Map with default threshold');
figure, imshow(BWs), title('Edge Map with modified threshold');
```

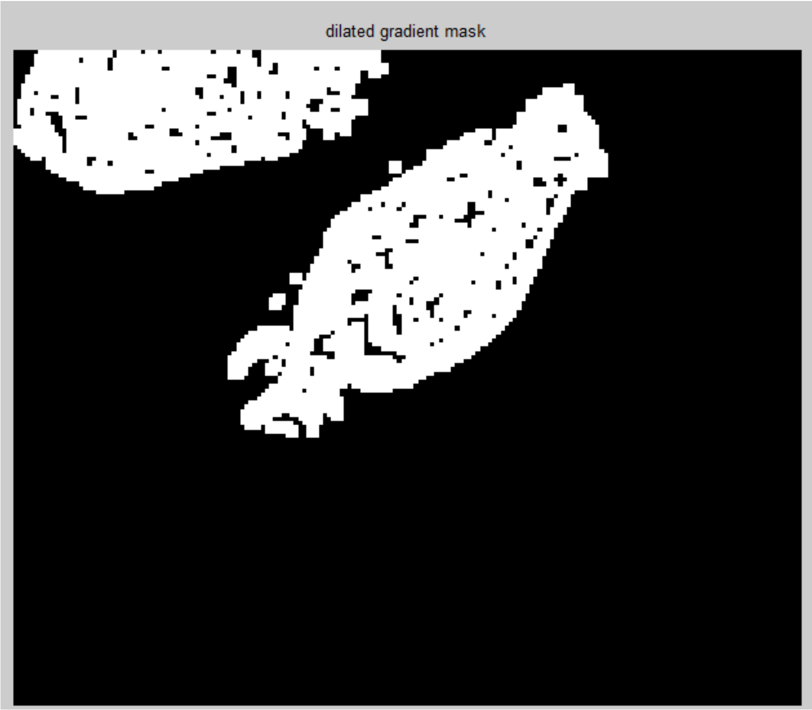




Detecting a Cell Using Image Segmentation

Step 3: Dilate the Image

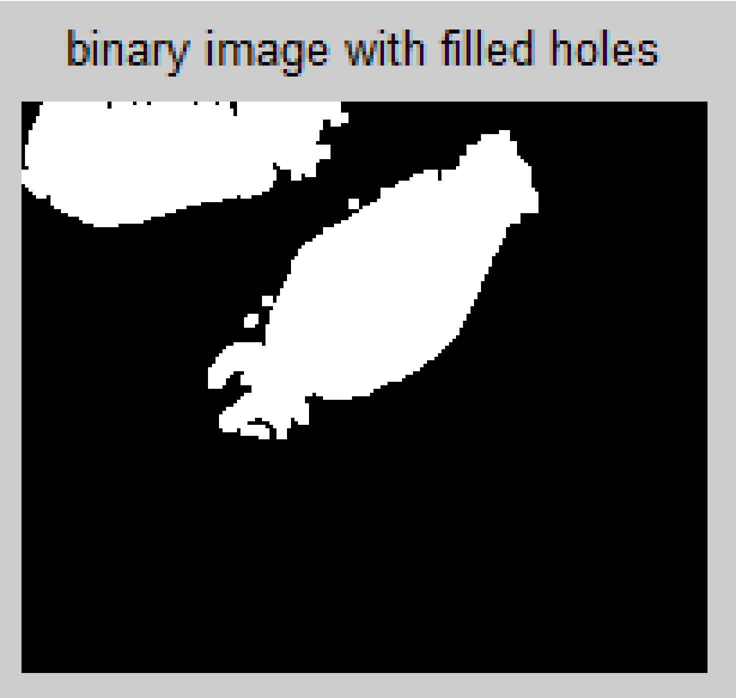
```
se=strel('square',3);
BWsdil = imdilate(BWs, se);
figure, imshow(BWsdil), title('dilated gradient mask');
```



Detecting a Cell Using Image Segmentation

Step 4: Fill Interior Gaps

```
BWdfill = imfill(BWsdil, 'holes');
figure, imshow(BWdfill); title('binary image with filled holes');
```



Detecting a Cell Using Image Segmentation

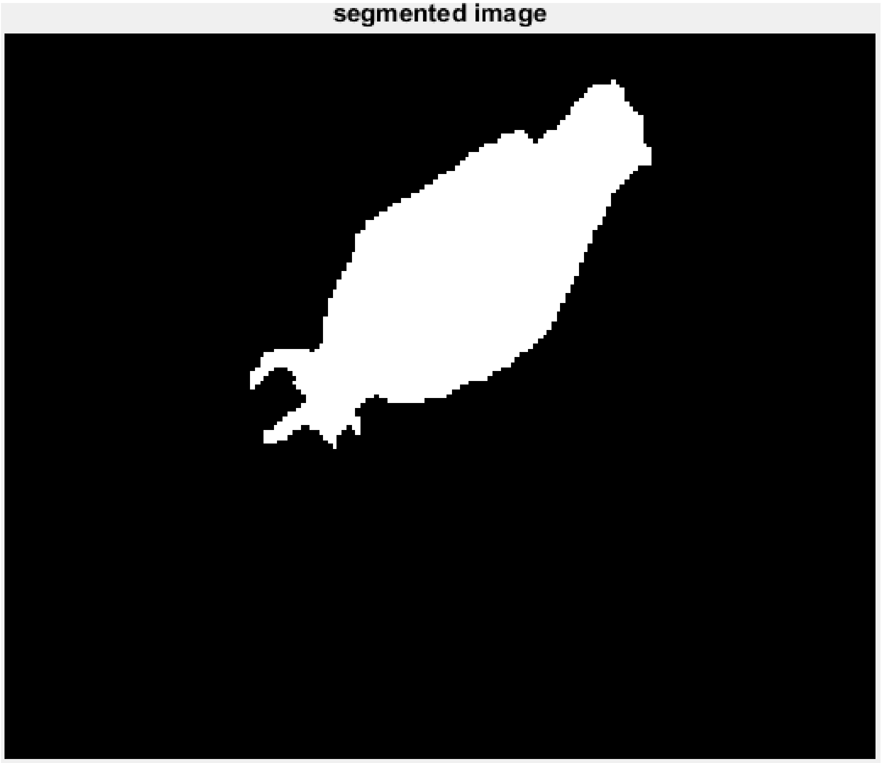
Step 5: Remove Connected Objects on Border

```
BWnobord = imclearborder(BWdfill, 4);
figure, imshow(BWnobord), title('cleared border image');
```

Detecting a Cell Using Image Segmentation

Step 6: Smoothen the Object

```
seD = strel('disk',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
figure, imshow(BWfinal), title('segmented image');
```



Automatically Detect and Recognize Text in Natural Images

Step 1: Detect Candidate Text Regions Using MSER

Step 2: Remove Non-Text Regions Based On Basic Geometric Properties

Step 3: Remove Non-Text Regions Based On Stroke Width Variation

Step 4: Merge Text Regions For Final Detection Result

Step 5: Recognize Detected Text Using OCR

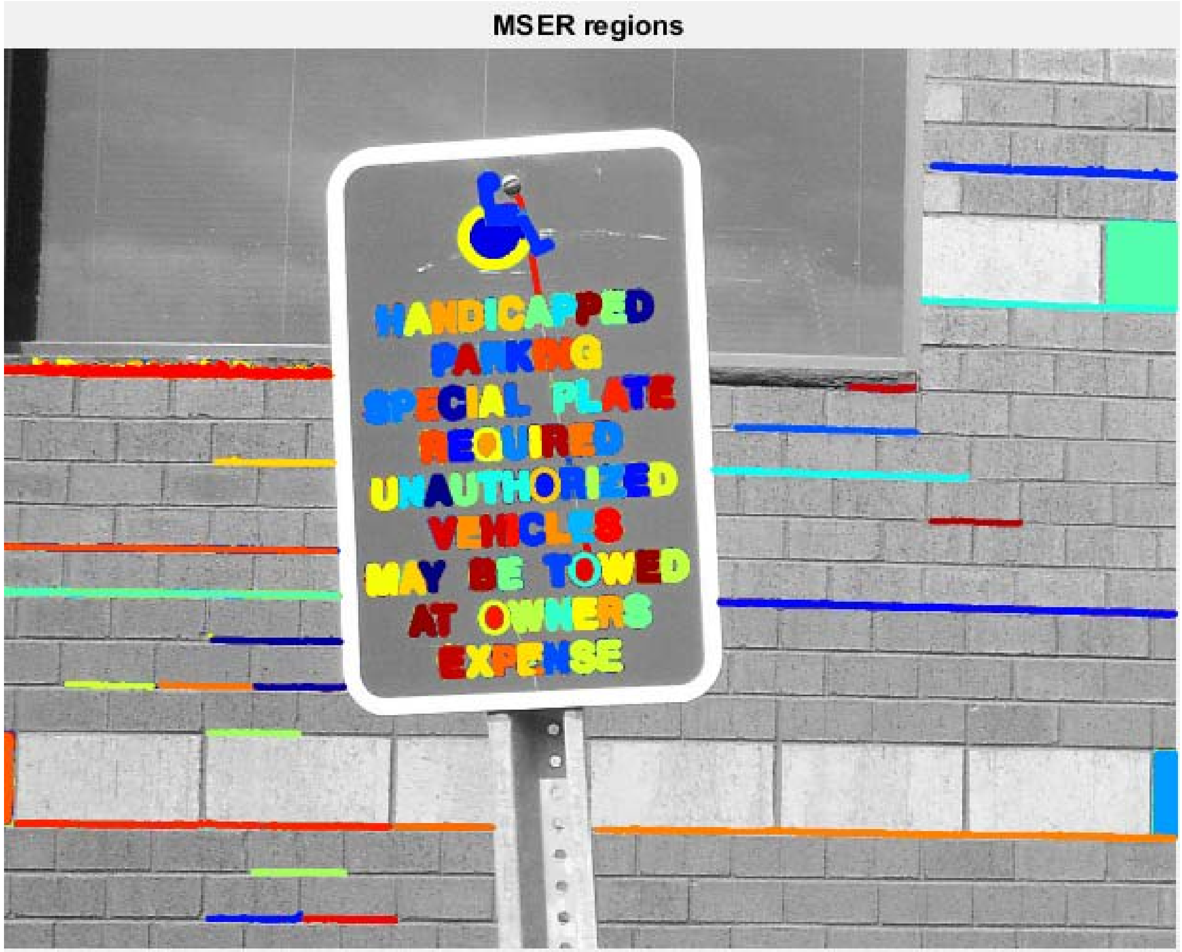
Automatically Detect and Recognize Text in Natural Images

Step 1: Detect Candidate Text Regions Using MSER

```
colorImage = imread('handicapSign.jpg');
I = rgb2gray(colorImage);
% Detect MSER regions.
[mserRegions, mserConnComp] = detectMSERFeatures(I, ...
   'RegionAreaRange',[200 8000],'ThresholdDelta',4);   % ThresholdDelta
Threshold of the stability value
figure
imshow(I)
hold on
plot(mserRegions, 'showPixelList', true,'showEllipses',false)
title('MSER regions')
```

Automatically Detect and Recognize Text in Natural Images

Step 1: Detect Candidate Text Regions Using MSER



Automatically Detect and Recognize Text in Natural Images

Step 2: Remove Non-Text Regions Based On Basic Geometric Properties

```
% Use regionprops to measure MSER properties
mserStats = regionprops(mserConnComp, 'BoundingBox', 'Eccentricity', ...
    'Solidity', 'Extent', 'Euler', 'Image');
% Extent – The ratio between number of pixels in the bounding box and in the region
% Euler- number of objects in the region minus the number of holes in those objects
%Image – binary image corresponds to pixel index list of the connected component
% Solidity - the proportion of the pixels in the convex hull that are also in the region
 % Compute the aspect ratio using bounding box data.
bbox = vertcat(mserStats.BoundingBox);
w = bbox(:,3);
h = bbox(:,4);
aspectRatio = w./h;
```

Step 2: Remove Non-Text Regions Based On Basic Geometric Properties

```
% Threshold the data to determine which regions to remove. These thresholds
% may need to be tuned for other images.
filterIdx = aspectRatio' > 3;
filterIdx = filterIdx | [mserStats.Eccentricity] > .995 ;
filterIdx = filterIdx | [mserStats.Solidity] < .3;
filterIdx = filterIdx | [mserStats.Extent] < 0.2 | [mserStats.Extent] > 0.9;
filterIdx = filterIdx | [mserStats.EulerNumber] < -4;
```

Automatically Detect and Recognize Text in Natural Images

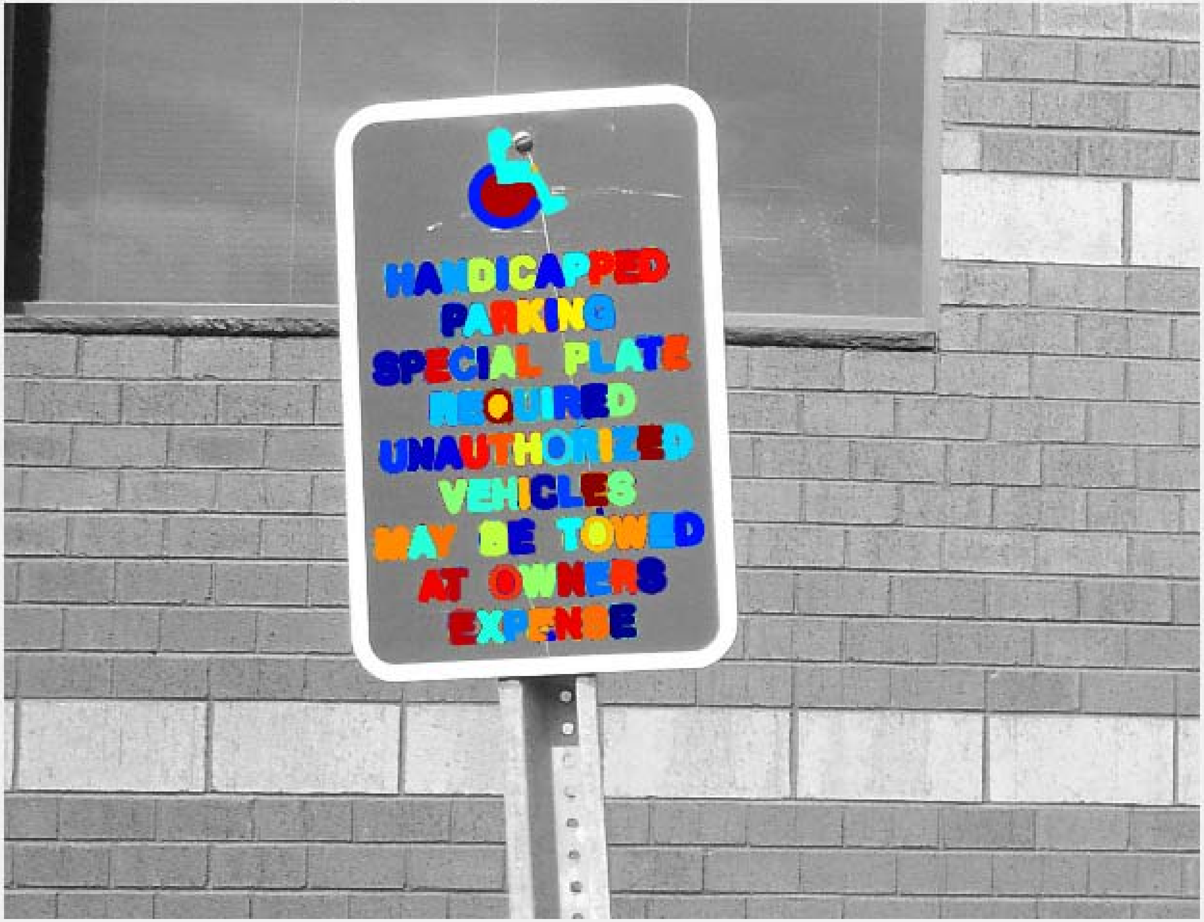
Step 2: Remove Non-Text Regions Based On Basic Geometric Properties

```
% Remove regions
mserStats(filterIdx) = [];
mserRegions(filterIdx) = [];

% Show remaining regions
figure
imshow(I)
hold on
plot(mserRegions, 'showPixelList', true,'showEllipses',false)
title('After Removing Non-Text Regions Based On Geometric Properties')
hold off
```

Automatically Detect and Recognize Text in Natural Images

Step 2: Remove Non-Text Regions Based On Basic Geometric Properties

After Removing Non-Text Regions Based On Geometric Properties



Step 3: Remove Non-Text Regions Based On Stroke Width Variation

```
% Get a binary image of the a region, and pad it to avoid boundary effects
% during the stroke width computation.
regionImage = mserStats(6).Image;
 regionImage = padarray(regionImage, [1 1]);
% Compute the stroke width image.
 distanceImage = bwdist(~regionImage);
skeletonImage = bwmorph(regionImage, 'thin', inf);
strokeWidthImage = distanceImage;
strokeWidthImage(~skeletonImage) = 0; % Show the region image
alongside the stroke width image.
figure; subplot(1,2,1);
imagesc(regionImage);
title('Region Image');
subplot(1,2,2);
 imagesc(strokeWidthImage); title('Stroke Width Image')
```

Step 3: Remove Non-Text Regions Based On Stroke Width Variation

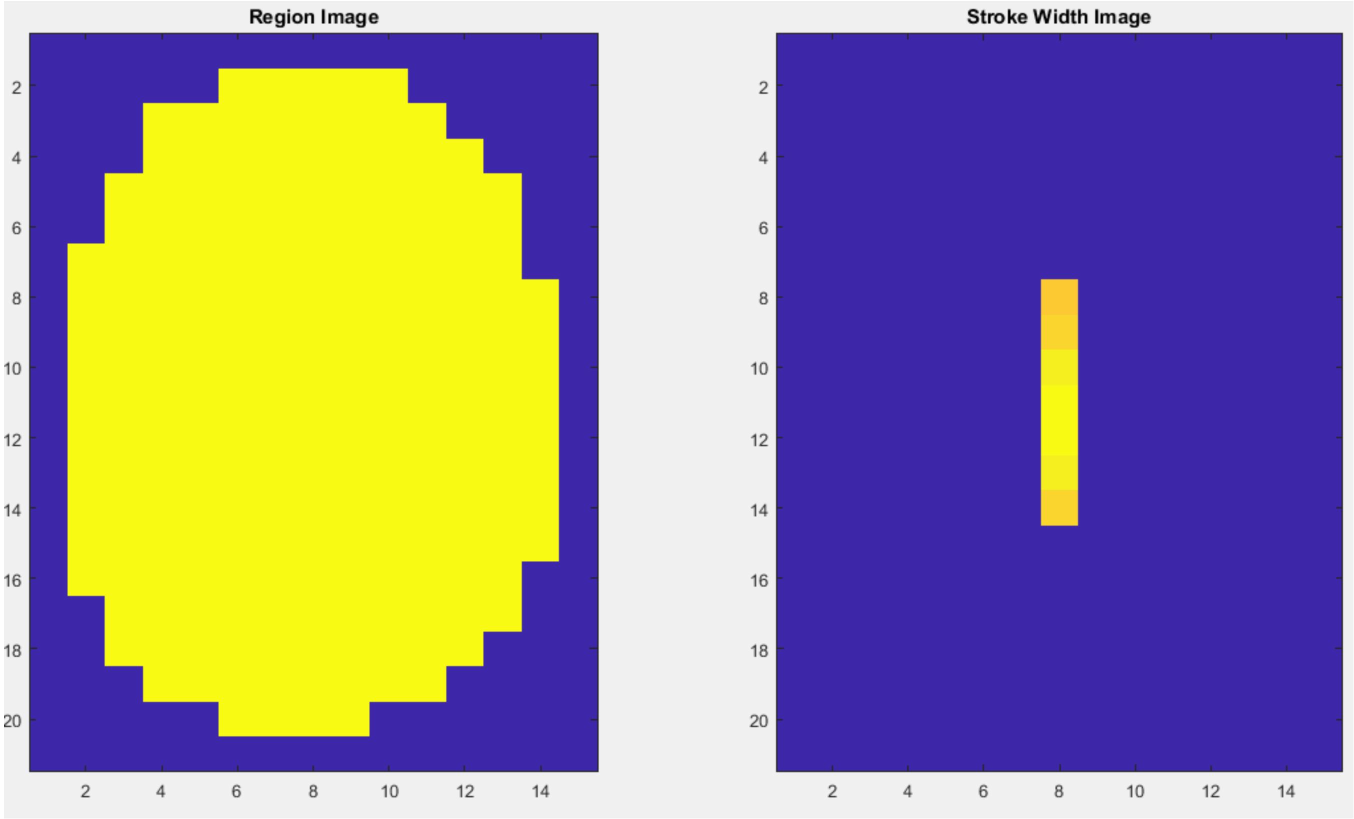
```
% Compute the stroke width variation metric
strokeWidthValues = distanceImage(skeletonImage);
strokeWidthMetric = std(strokeWidthValues)/mean(strokeWidthValues);

% Threshold the stroke width variation metric
strokeWidthThreshold = 0.4;
strokeWidthFilterIdx = strokeWidthMetric > strokeWidthThreshold;

% Threshold the stroke width variation metric
strokeWidthThreshold = 0.4;
strokeWidthFilterIdx = strokeWidthMetric > strokeWidthThreshold;
```

Automatically Detect and Recognize Text in Natural Images

Step 3: Remove Non-Text Regions Based On Stroke Width Variation

```
% Process the remaining regions
for j = 1:numel(mserStats)
   regionImage = mserStats(j).Image;
   regionImage = padarray(regionImage, [1 1], 0);
   distanceImage = bwdist(~regionImage);
   skeletonImage = bwmorph(regionImage, 'thin', inf);
   strokeWidthValues = distanceImage(skeletonImage);
   strokeWidthMetric = std(strokeWidthValues)/mean(strokeWidthValues);
   strokeWidthFilterIdx(j) = strokeWidthMetric > strokeWidthThreshold;
end
% Remove regions based on the stroke width variation
mserRegions(strokeWidthFilterIdx) = [];
mserStats(strokeWidthFilterIdx) = [];
% Show remaining regions
Figure; imshow(I); hold on
plot(mserRegions, 'showPixelList', true,'showEllipses',false)
title('After Removing Non-Text Regions Based On Stroke Width Variation')
hold off
```

Step 3: Remove Non-Text Regions Based On Stroke Width Variation



Automatically Detect and Recognize Text in Natural Images

Step 3: Remove Non-Text Regions Based On Stroke Width Variation

Automatically Detect and Recognize Text in Natural Images

Step 4: Merge Text Regions For Final Detection Result

```
% Get bounding boxes for all the regions
bboxes = vertcat(mserStats.BoundingBox);

% Convert from the [x y width height] bounding box format to the [xmin ymin
% xmax ymax] format for convenience.
xmin = bboxes(:,1);
ymin = bboxes(:,2);
xmax = xmin + bboxes(:,3) - 1;
ymax = ymin + bboxes(:,4) - 1;

% Expand the bounding boxes by a small amount.
expansionAmount = 0.02;
xmin = (1-expansionAmount) * xmin;
ymin = (1-expansionAmount) * ymin;
xmax = (1+expansionAmount) * xmax;
ymax = (1+expansionAmount) * ymax;

% Clip the bounding boxes to be within the image bounds
xmin = max(xmin, 1);
ymin = max(ymin, 1);
```

Automatically Detect and Recognize Text in Natural Images

Step 4: Merge Text Regions For Final Detection Result

```
% Clip the bounding boxes to be within the image bounds
xmin = max(xmin, 1);
ymin = max(ymin, 1);
xmax = min(xmax, size(I,2));
ymax = min(ymax, size(I,1));

% Show the expanded bounding boxes
expandedBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
IExpandedBBoxes =
insertShape(colorImage,'Rectangle',expandedBBoxes,'LineWidth',3);

figure
imshow(IExpandedBBoxes)
title('Expanded Bounding Boxes Text')
% Compute the overlap ratio
overlapRatio = bboxOverlapRatio(expandedBBoxes, expandedBBoxes);

% Set the overlap ratio between a bounding box and itself to zero to
% simplify the graph representation.
n = size(overlapRatio,1);
```

Automatically Detect and Recognize Text in Natural Images

Step 4: Merge Text Regions For Final Detection Result

```
% Set the overlap ratio between a bounding box and itself to zero to
% simplify the graph representation.
n = size(overlapRatio,1);
overlapRatio(1:n+1:n^2) = 0;
% Create the graph
g = graph(overlapRatio);
% Find the connected text regions within the graph
componentIndices = conncomp(g);
% Merge the boxes based on the minimum and maximum dimensions.
xmin = accumarray(componentIndices', xmin, [], @min);
ymin = accumarray(componentIndices', ymin, [], @min);
xmax = accumarray(componentIndices', xmax, [], @max);
ymax = accumarray(componentIndices', ymax, [], @max);
% Compose the merged bounding boxes using the [x y width height] format.
textBBoxes = [xmin ymin xmax-xmin+1 ymax-ymin+1];
```

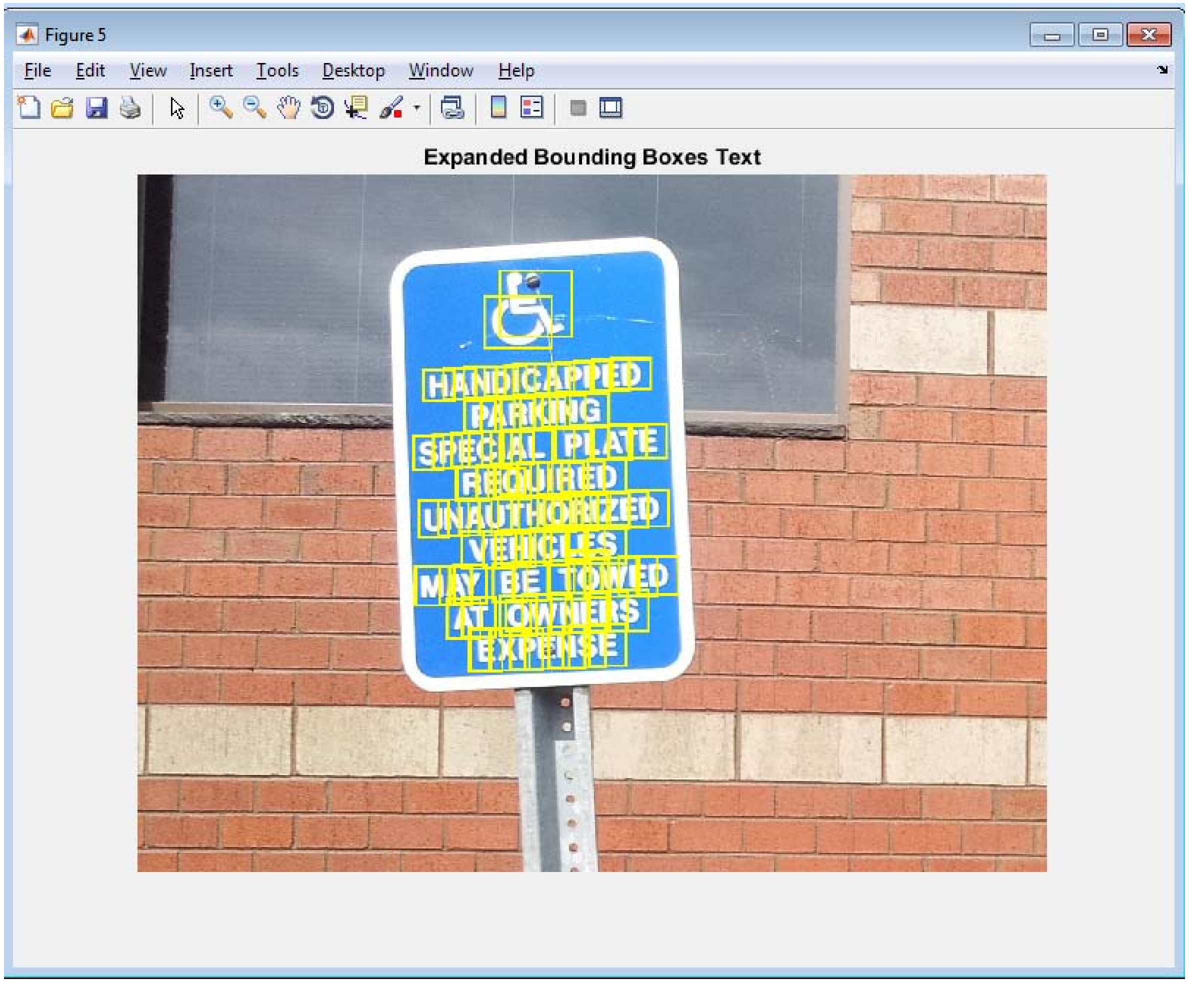
Automatically Detect and Recognize Text in Natural Images

Step 4: Merge Text Regions For Final Detection Result

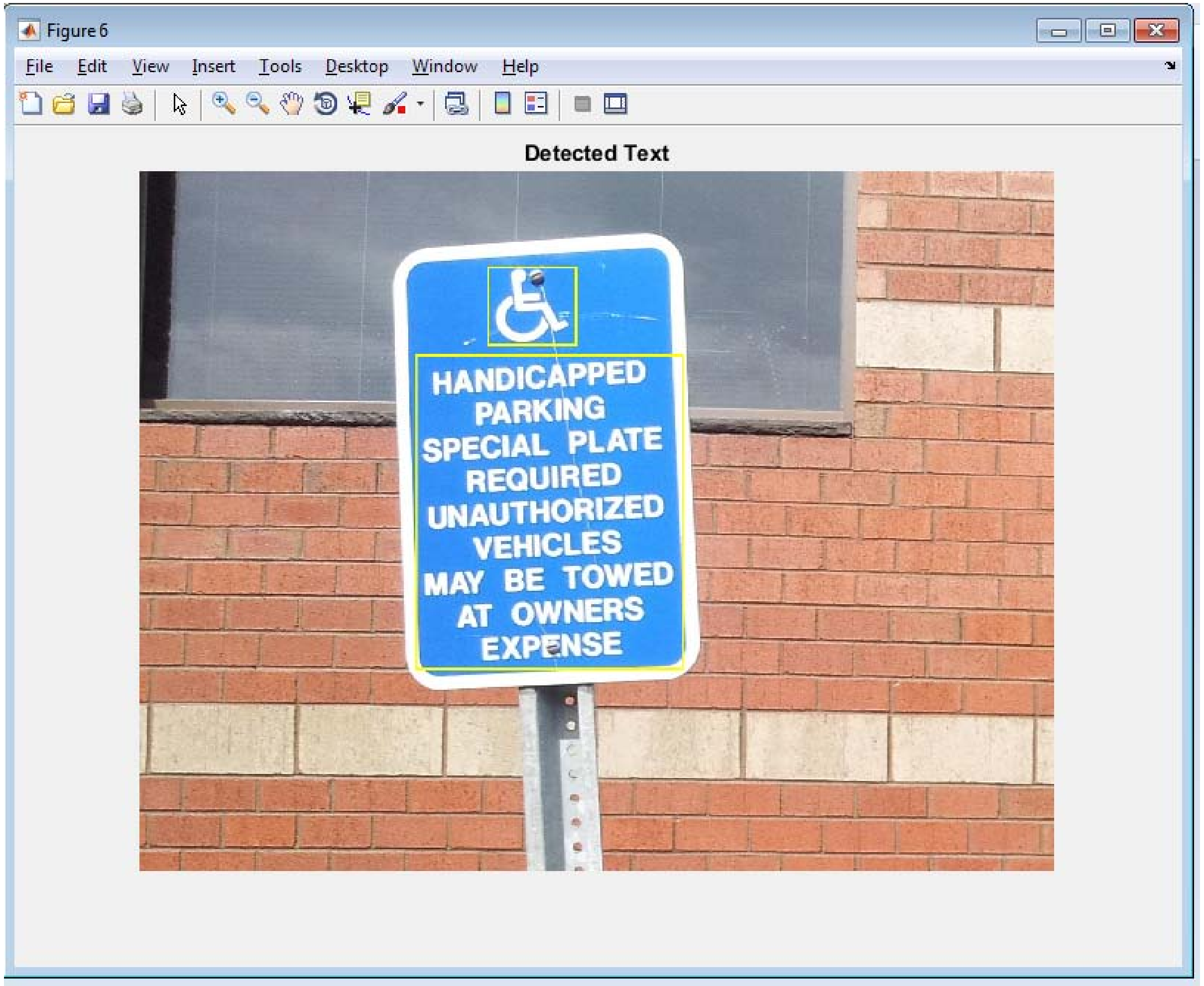
```
% Remove bounding boxes that only contain one text region
numRegionsInGroup = histcounts(componentIndices);
textBBoxes(numRegionsInGroup == 1, :) = [];

% Show the final text detection result.
ITextRegion = insertShape(colorImage, 'Rectangle', textBBoxes,'LineWidth',3);

figure
imshow(ITextRegion)
title('Detected Text')
```

Automatically Detect and Recognize Text in Natural Images

Step 4: Merge Text Regions For Final Detection Result



Step 4: Merge Text Regions For Final Detection Result



Automatically Detect and Recognize Text in Natural Images

Step 5: Recognize Detected Text Using OCR

```
ocrtxt = ocr(I, textBBoxes);
[ocrtxt.Text]
```

THANK YOU…