EP375 Computational Physics

Topic 13

IMAGE PROCESSING

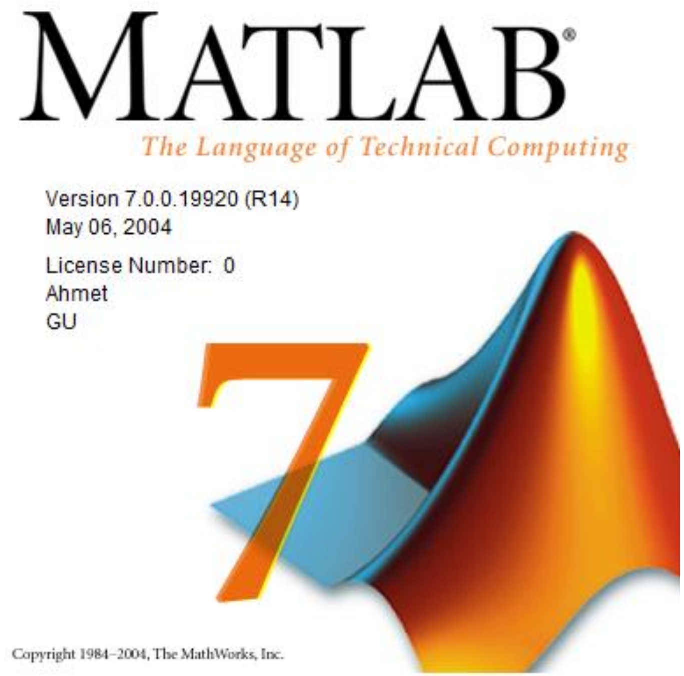
Department of Engineering Physics

University of Gaziantep

Apr 2016

Content

1. **Introduction**
2. **Nature of Image**
3. **Image Types / Colors**
4. **Reading, Writing and Displaying Images**
5. **Example Applications**



Introduction

- We have seen 2D or 3D plots of basic data.

- In this chapter we will discus some of the elementary processes that can be applied to images.

Nature of Image

- An image is a two-dimensional sheet on which the color at any point can have essentially infinite variability.

- 2-D images are M x N array of points usually referred to as picture elements, or pixels, where M and N are the number of rows and columns respectively.

- Each pixel is “painted” by blending variable amounts of the three primary colors: red, green, and blue => RGB.

- The resolution (quality) of a picture is measured by the number of pixels per unit of picture width and height.

- The color resolution is measured by the number of bits in the words containing RGB components.

- Typically, 8 bits (values 0–255) are assigned to each color.

- By combining the three color values, we have 2^{24} combinations of "true colors".

- The human eye can distinguish many more possible combinations!

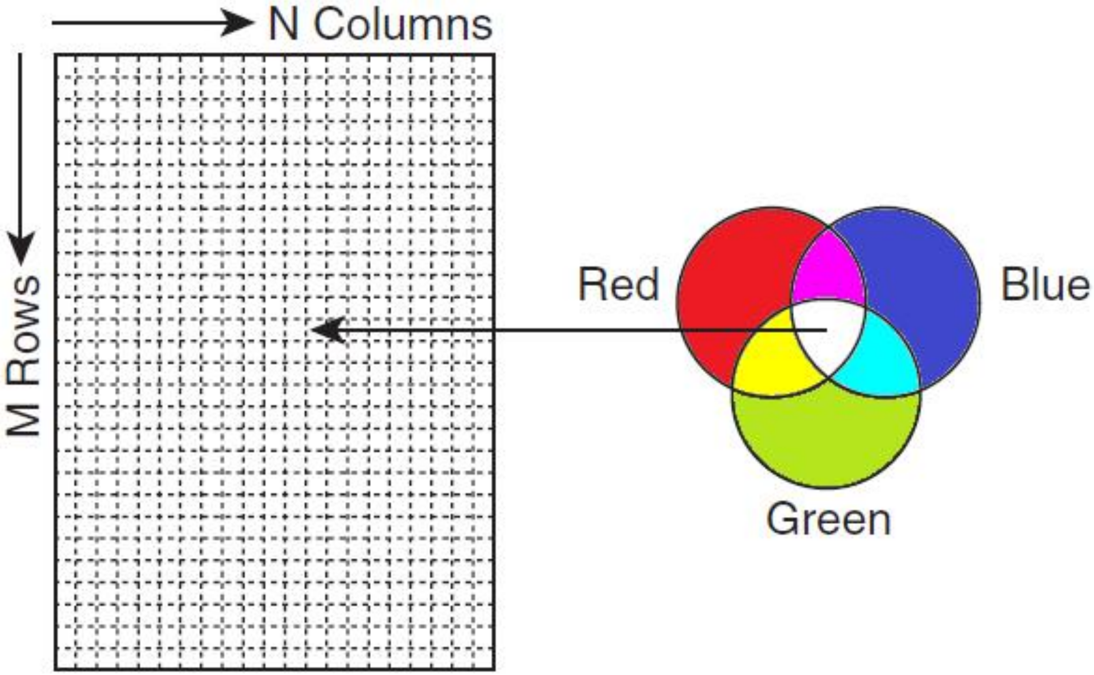


Image Types

- Sources of images are data files captured by cameras, scanners and etc.

- Image files are provided in a wide variety of formats. MATLAB identifies the files in TIFF, PNG, HDF, BMP, JPEG (JPG), GIF, PCX, XWD, CUR, and ICO formats.

True Color Images

stored as an M x N x 3 array

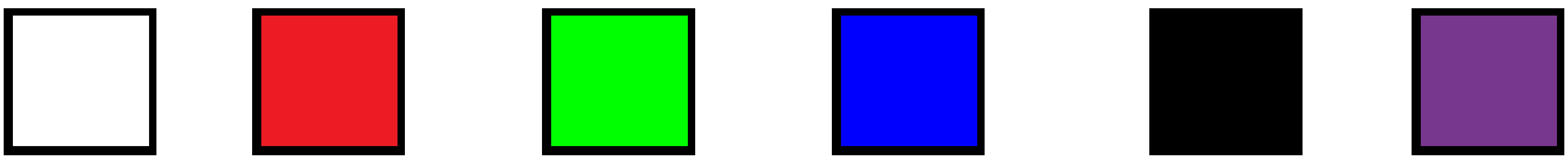
`A(:, :, 3)`

RGB index

1 = Red

2 = Green

3 = Blue



R: 255	R: 255	R: 0	R: 0	R: 0	R: 120
G: 255	G: 0	G: 255	G: 0	G: 0	G: 55
B: 255	B: 0	B: 0	B: 255	B: 0	B: 142

Gray Scale Images

stored the black-to-white intensity value for each pixel as a single value rather than three values.

The value 0 corresponds to black and 255 to white.

Reading, Displaying and Writing Images

```
>> p = imread('myFigure.jpg');
>> imshow(p)
>> imwrite(p, 'new.png', 'png')
```

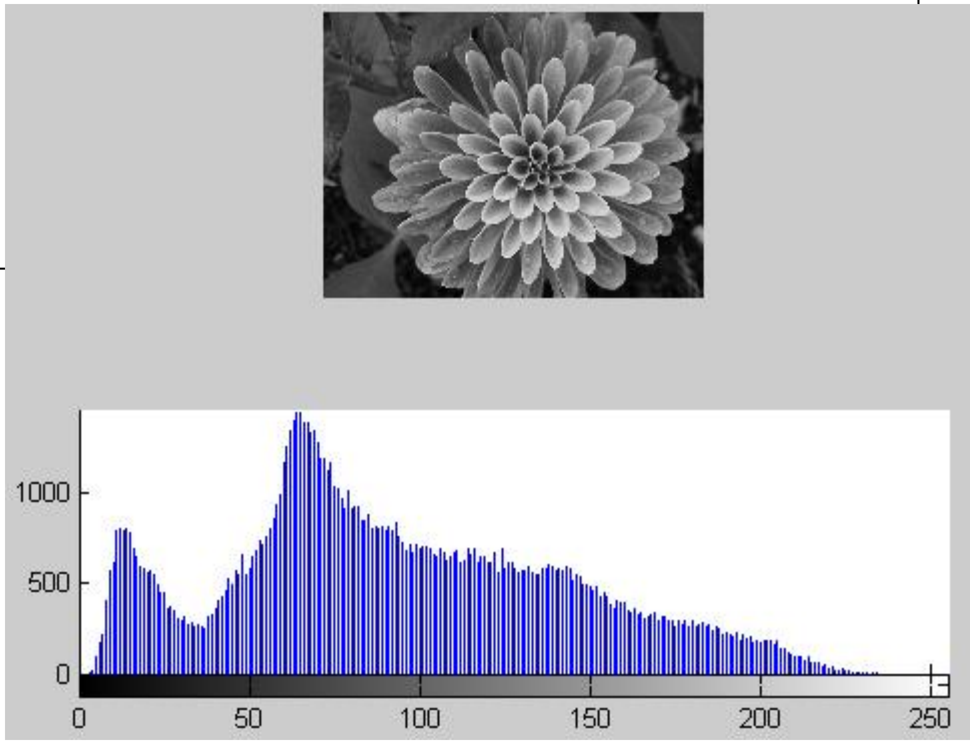
where the result, p, is an M x N x 3 uint8 array of pixel color values.

Basic Image Processing Functions

imread()	open an image file
imshow()	display an image file
size()	size of an image
imresize()	resize an image
rgb2gray()	convert rgb image to grayscale
im2bw()	convert an image to BlackWhite
imhist()	histogram of the image
histeq()	histogram equilization
imwrite()	save image
imcomplement()	comlement of an image
imadd()	add a value to each pixel
imrotate()	rotate an image
imcrop()	crop an image
edge()	edge detection for an image
bwarea()	return area (number of pixels) for a given region

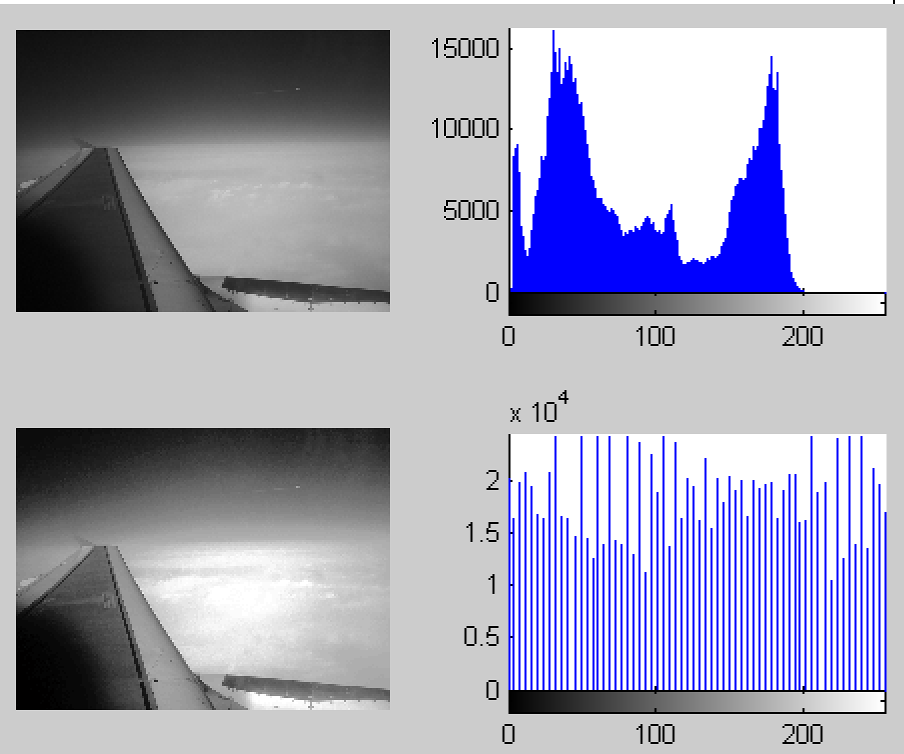
```
% ip1.m
% convert to gray-scale and black & white

A = imread('cicek.jpg');
B = imresize(A,[256,256]);
C = rgb2gray(A);
D = im2bw(A);

subplot(2,2,1); imshow(A)
subplot(2,2,2); imshow(B)
subplot(2,2,3); imshow(C)
subplot(2,2,4); imshow(D)
```

```
% ip2.m
% gray-scale histogram of an image
%
% size(A) = 300   400     3
% size(B) = 300   400

A = imread('cicek.jpg');
B = rgb2gray(A);
disp(size(A))
disp(size(B))
% plot
subplot(2,1,1); imshow(B)
subplot(2,1,2); imhist(B)
```



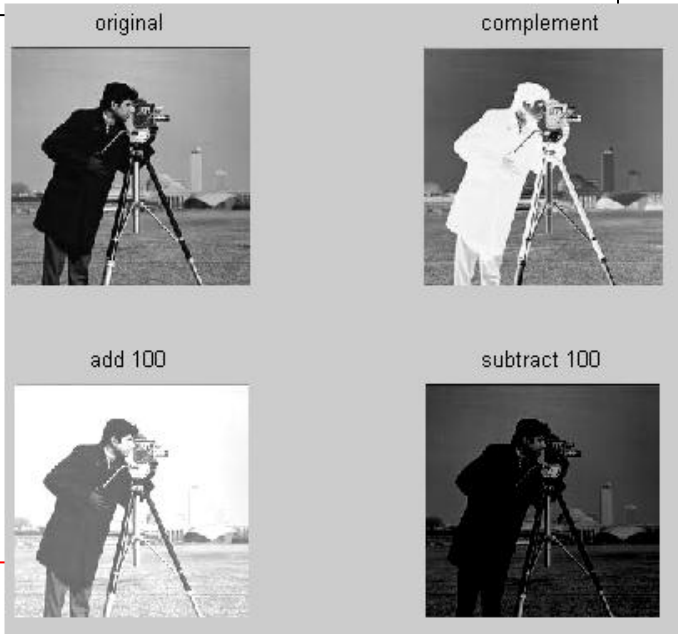
Can you write imhist() function?

```
% ip2.m
% histogram equalization

A = imread('ucak.jpg');
B = rgb2gray(A);
C = histeq(B);

subplot(2,2,1); imshow(B);
subplot(2,2,2); imhist(B);
subplot(2,2,3); imshow(C);
subplot(2,2,4); imhist(C);
```



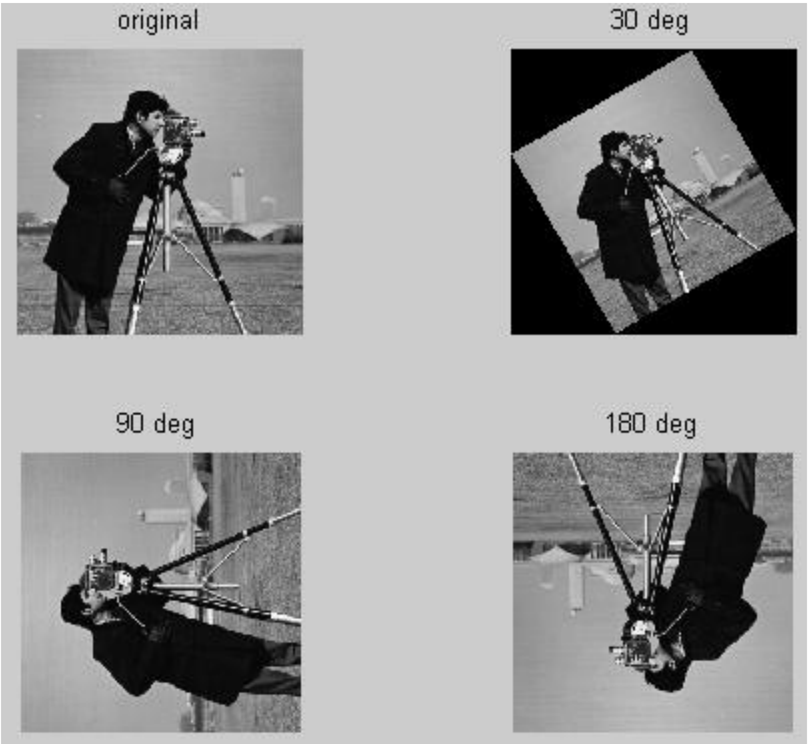
```
% ip3.m
% contrast and negative of an image

A = imread('cameraman.tif');
B = imcomplement(A); % negative of the image
C = imadd(A, 100);   % add 100 to all values
D = imadd(A,-100);   % subtract 100 from all values

% plot
subplot(2,2,1); imshow(A); title('original')
subplot(2,2,2); imshow(B); title('complement')
subplot(2,2,3); imshow(C); title('add 100')
subplot(2,2,4); imshow(D); title('subtract 100')
```

Can you write imcomplement() and imadd() functions?



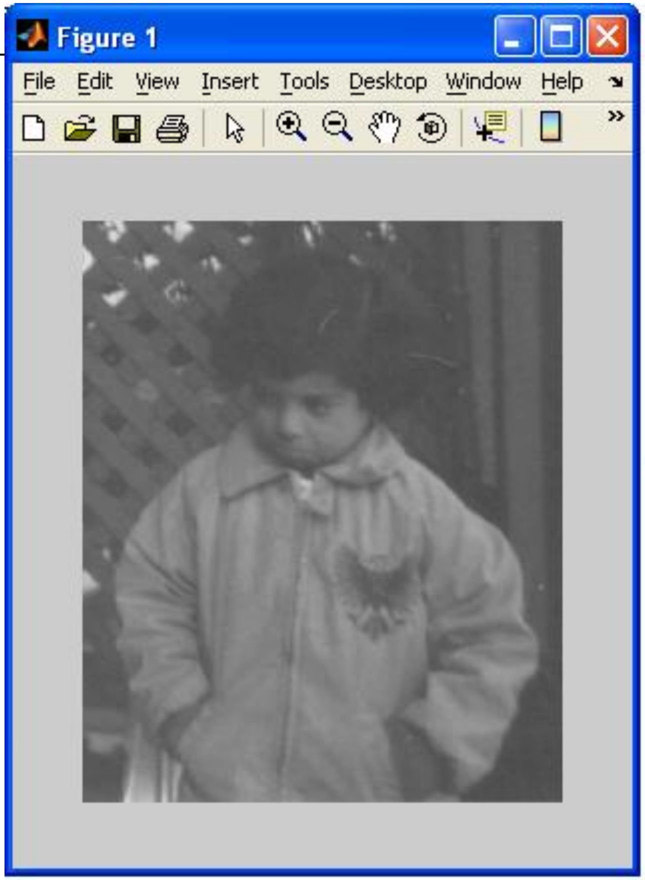
```
% ip4.m
% color components of an image

% copy images
A = imread('cicek.jpg');
B = A;
C = A;
D = A;
% RGB colors
B(:,:,2)=0; B(:,:,3)=0; % keep only red
C(:,:,1)=0; C(:,:,3)=0; % green
D(:,:,1)=0; D(:,:,2)=0; % blue
% plot
subplot(2,2,1); imshow(A)
subplot(2,2,2); imshow(B)
subplot(2,2,3); imshow(C)
subplot(2,2,4); imshow(D)
```

```
% ip5.m
% Rotating an image

A = imread('cameraman.tif');
subplot(2,2,1); imrotate(A,  0); title('original')
subplot(2,2,2); imrotate(A, 30); title('30 deg')
subplot(2,2,3); imrotate(A, 90); title('90 deg')
subplot(2,2,4); imrotate(A,180); title('180 deg')
```



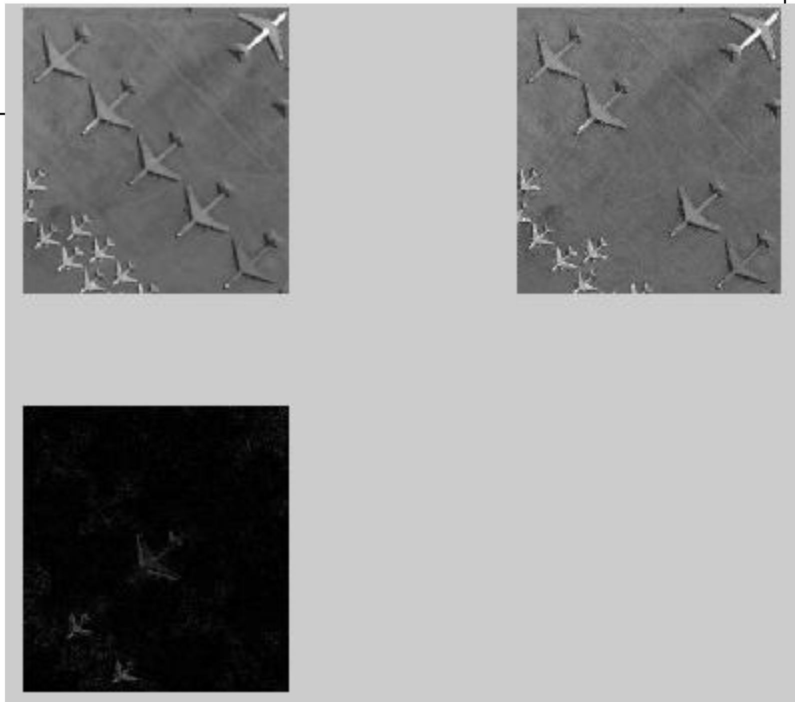
Can you write imrotate() function?

```
% ip6.m
% Cropping an image

A = imread('pout.tif');
B = imcrop(A, [55 10 120 120]);
figure, imshow(A)
figure, imshow(B)
```

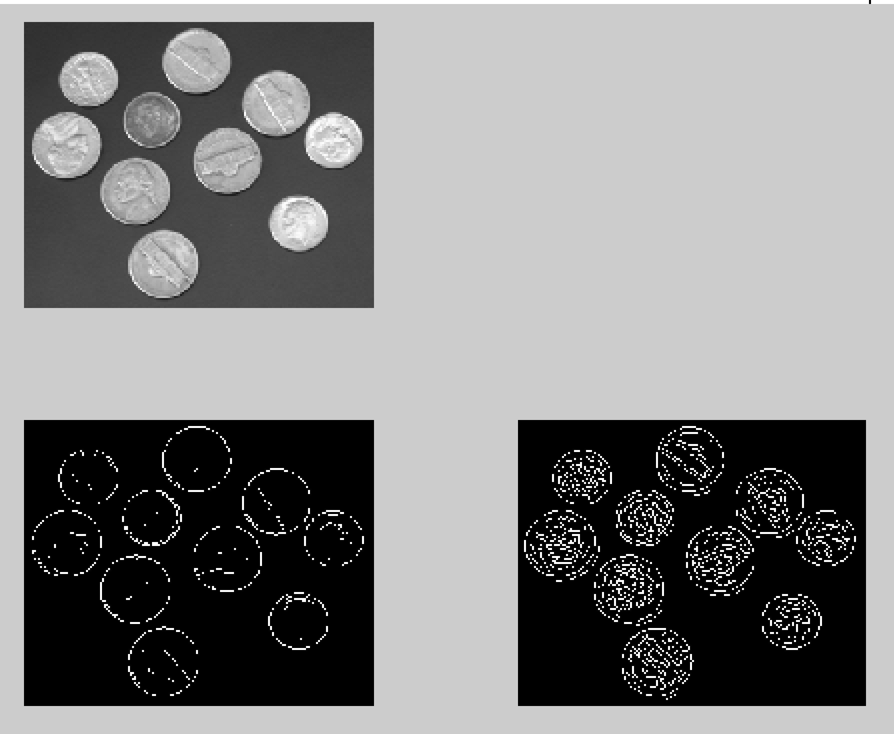


Can you write imcrop() function?

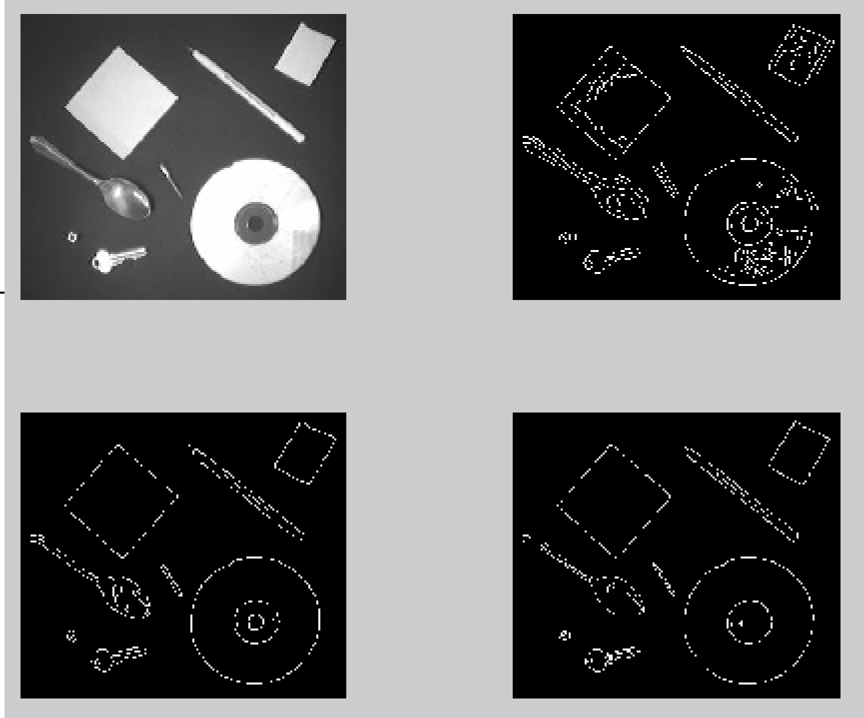
```
% ip7.m
% finding diffreence between two images

clear; clc

i1 = imread('planes1.jpg');
i2 = imread('planes2.jpg');
dif= imabsdiff(i1, i2);

subplot(2,2,1); imshow(i1);
subplot(2,2,2); imshow(i2);
subplot(2,2,3); imshow(dif);
```



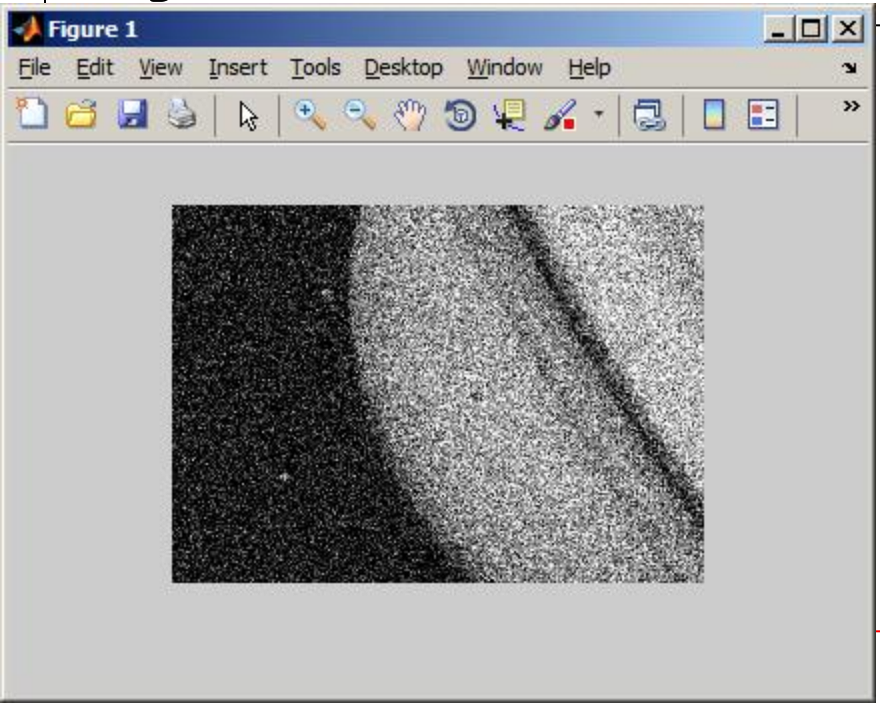
```
% ip8.m
% detect edges
clear; clc

I = imread('coins.png');

BW1 = edge(I,'sobel');
BW2 = edge(I,'canny');

subplot(2,2,1); imshow(I)
subplot(2,2,3); imshow(BW1)
subplot(2,2,4); imshow(BW2)
```

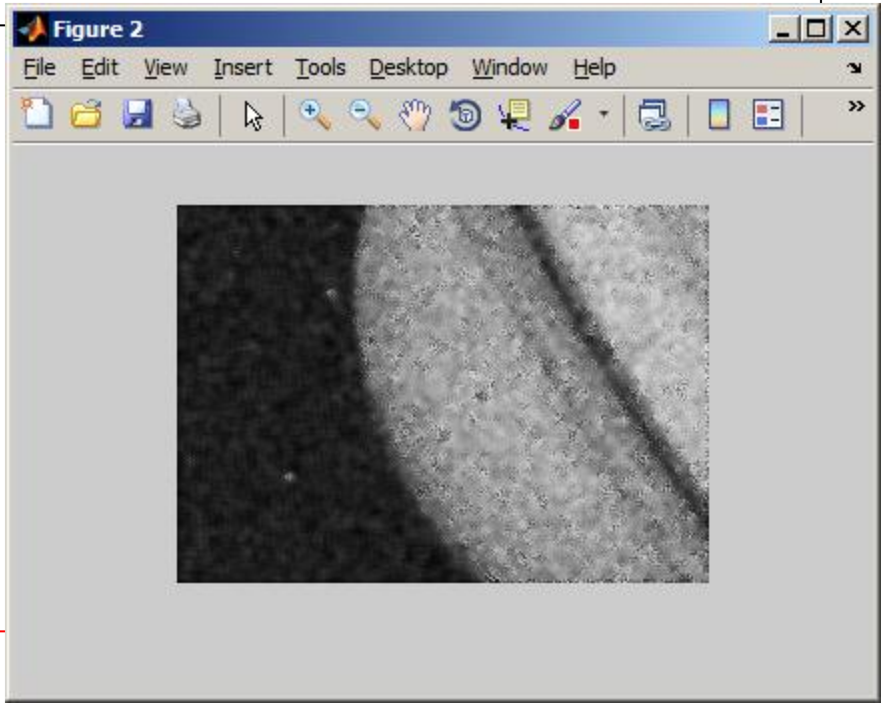


```
% ip9.m
% detect edges
clear; clc

% Read the sample image
A = imread('shapes.jpg');
B = edge(A,'canny');
C = edge(A,'canny', [0.1 0.2], 1);
D = edge(A,'sobel');
% plot
subplot(2,2,1); imshow(A)
subplot(2,2,2); imshow(B);
subplot(2,2,3); imshow(C);
subplot(2,2,4); imshow(D);
```



```
% ip10.m
% noise reduction
RGB = imread('sat_noisy.jpg');
I = rgb2gray(RGB);
J = imnoise(I,'gaussian',0,0.025);

imshow(J)
K = wiener2(J,[5 5]);
figure, imshow(K)
```





```
% ip11.m
%
RGB = imread('sirius.jpg');
I = rgb2gray(RGB);
h = fspecial('disk',20);
I2 = filter2(h,I)/255;
x = I2(1500,:);
figure, imshow(I)
figure, imshow(I2)
figure, plot(x)
```

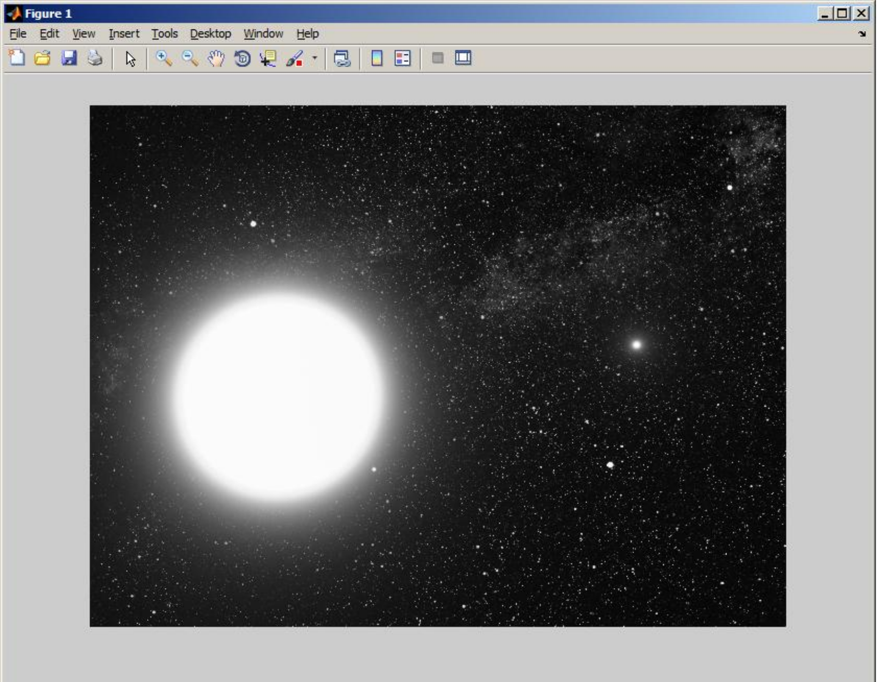


Figure 1
File Edit View Insert Tools Desktop Window Help

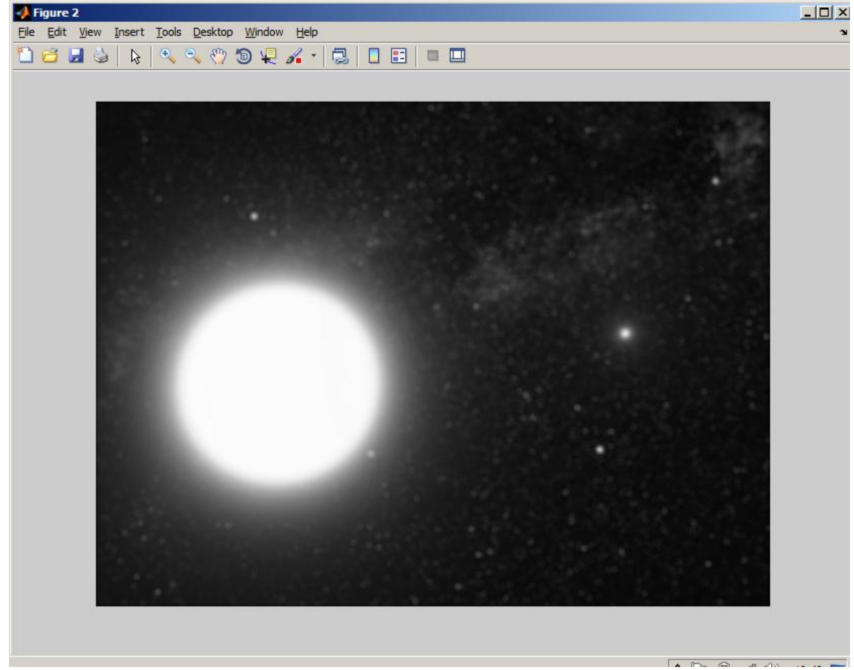


Figure 2
File Edit View Insert Tools Desktop Window Help

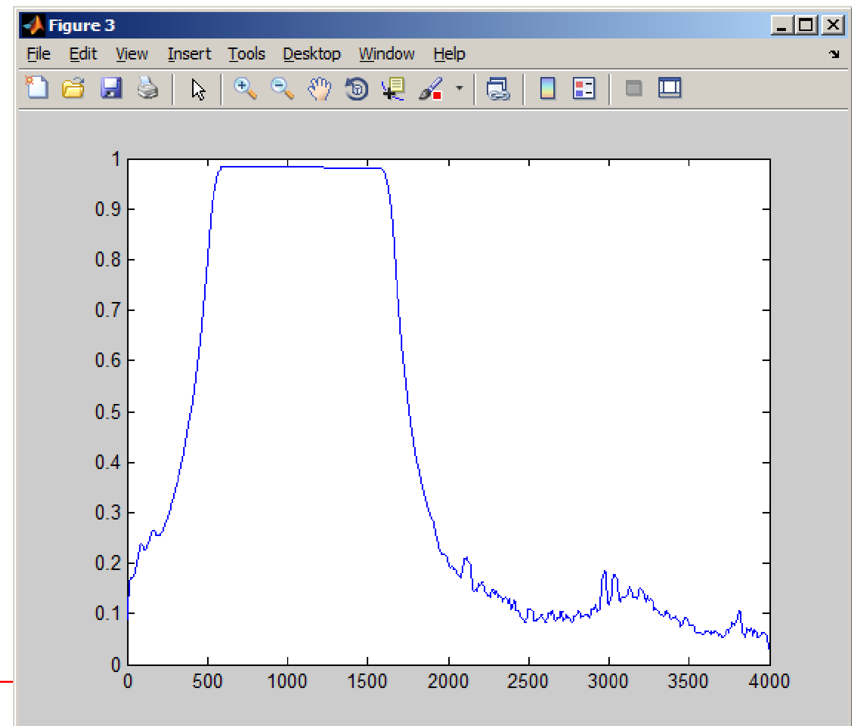
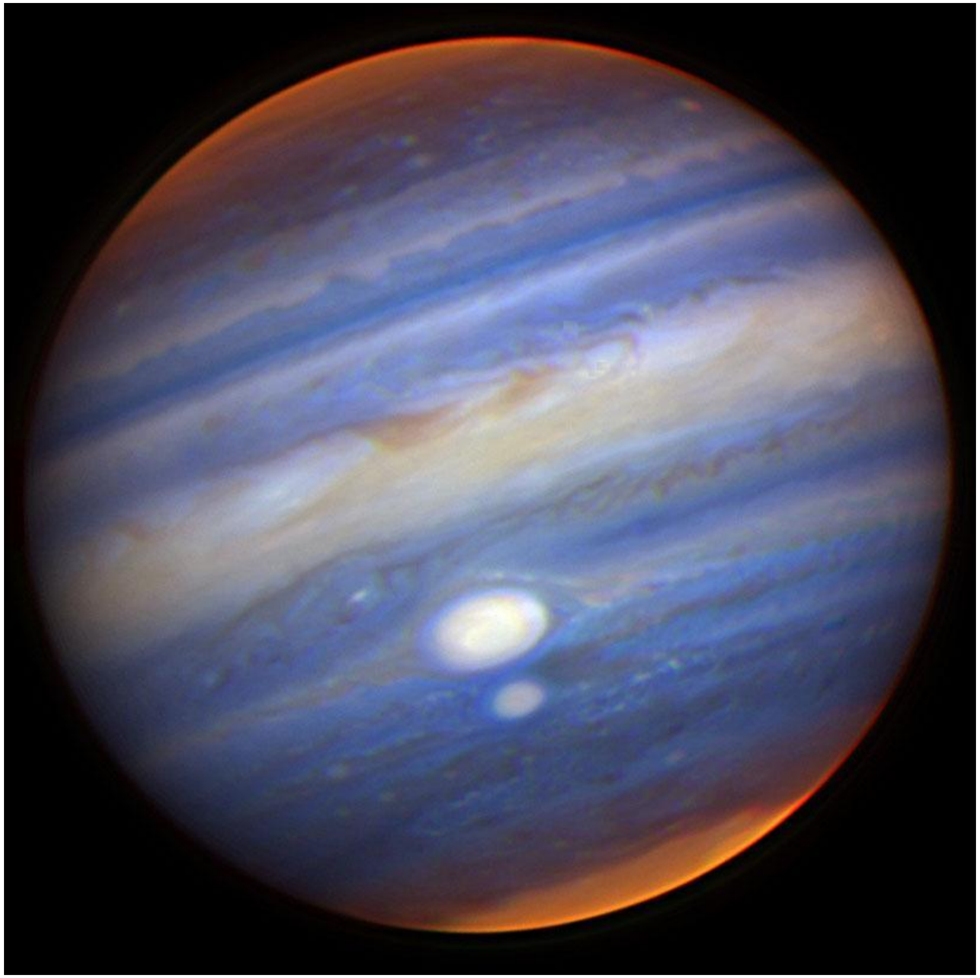


Figure 3
File Edit View Insert Tools Desktop Window Help
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
500
1000
1500
2000
2500
3000
3500
4000

Example

Determine the area of the big white spot on Jupiter as shown below.

Also compare the size of the spot with size of Earth.



Note that appoximate values are:

Radius of Jupiter:

$R_J = 71{,}500$ km

Radius of Earth:

$R_E = 6380$ km

A Complex Example

Wikipedia says:

http://en.wikipedia.org/wiki/Great_Red_Spot#Great_Red_Spot

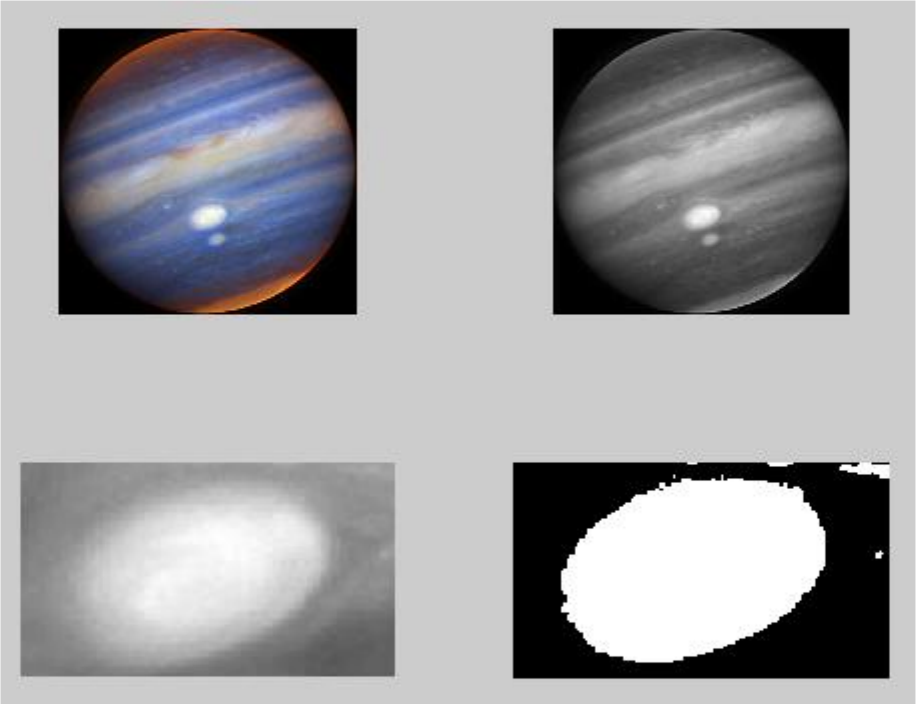
The Great Red Spot is a persistent anticyclonic storm.

The strorm is large enough to be visible through Earth-based telescopes.

The spot is large enough to contain two or three planets the size of Earth.

```matlab
% jupiter.m
clc; clear; figure
A = imread('jupiter2.jpg');
B = rgb2gray(A);
C = imcrop(B, [330 460 140 80]);
D = C > 160;

subplot(2,2,1); imshow(A);
subplot(2,2,2); imshow(B);
subplot(2,2,3); imshow(C);
subplot(2,2,4); imshow(D);

x = size(A(:,:,:));
RJ = 71500;
RE =  6380;
factor = 2*RJ/x(1);
p_area = bwarea(D);
r_area = factor * factor * p_area;

fprintf('Pixel area    = %f\n',p_area);
fprintf('Real  area    = %f km^2\n',r_area);
fprintf('Size of Earth = %f km^2\n',pi*RE^2);
fprintf('Ratio         = %f\n',r_area/(pi*RE^2));
```

```
>> jupiter

Pixel area     = 5299.500000
Real  area     = 187620283.067867 km^2
Size of Earth = 127876644.008780 km^2
Ratio          = 1.467197
```

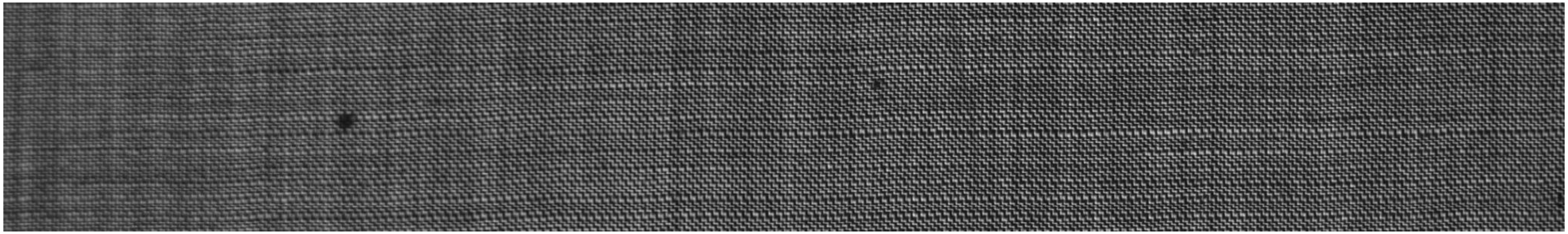


Problem

Using MATLAB image processing tool, measure the angle between the connections for the following figures.

Problem

Using MATLAB image processing tool, find the positions of defects on the fabrics given below.



References:

[1]. Numerical Methods for Engineers, 6th Ed.
S.C. Chapra, Mc Graw Hill (2010)

[2]. http://www.mathworks.com/products/matlab

[3]. Numerical Methods in Engineering with MATLAB,
J. Kiusalaas, Cambridge University Press (2005)

[4]. Essential MATLAB for Engineers and Scientist, 3rd Ed
Hahn B., Valentine D.T. (2007)

[5]. Computational Physics,
Giordano J.N.   Prentice Hall (1997)

[6]. http://en.wikipedia.org/wiki/Infinite_quantum_well

[7]. http://en.wikipedia.org/wiki/Harmonic_oscillator_(quantum)

[8]. http://en.wikipedia.org/wiki/Lenard-Jones_potential