Chamara k.k.v

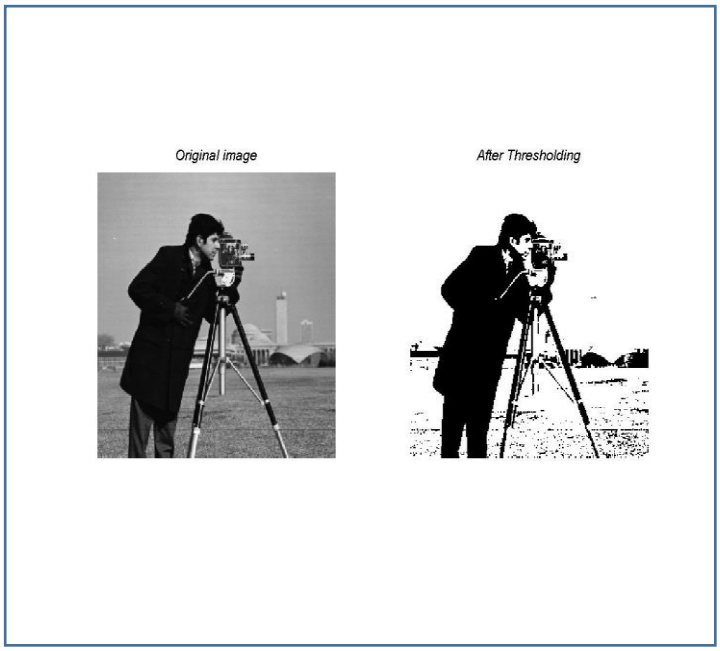
E/14/049

# CO543 – Image Processing Lab 2

## 1). Image arithmetic operations

In thresholding we map the gray scale image matrixes values to the 0 or 256 after comparing the values with the threshold value. Then we can see that the edges are highlighted and we can easily separate the boundaries. It's also depend on the image that you are processing.

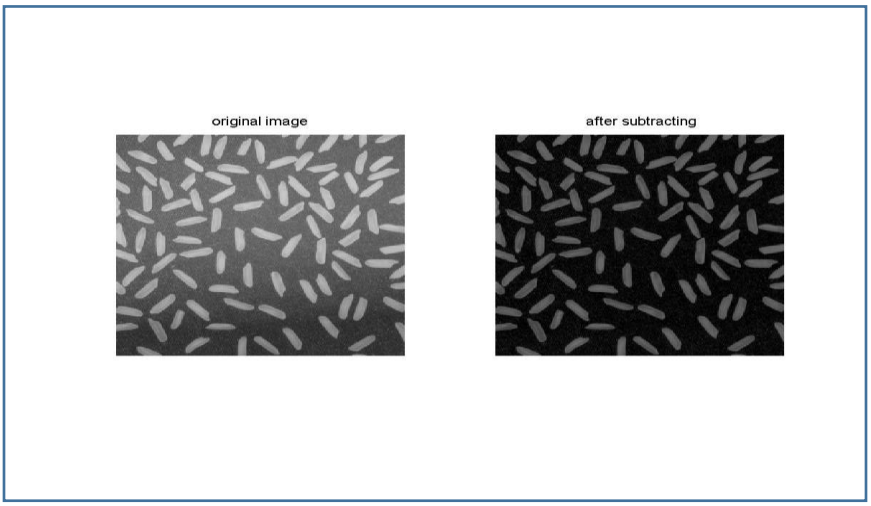
```
prompt = 'What is the Image? ';
imageName =input(prompt,'s');
prompt = 'Enter threshold value? ';
threshold =input(prompt);
img0 =imread(imageName);
p=double(img0);

 for x=1:1:row
    for y=1:1:col
        if(p(x,y)>threshold)
            p(x,y)=255;
        else
            p(x,y)= 0;
        end
    end
 end
subplot(1,2,1),imshow(img0),
title('\itOriginal image')
subplot(1,2,2),imshow(p),
title('\itAfter Thresholding')
```




## 2). Image arithmetic operations

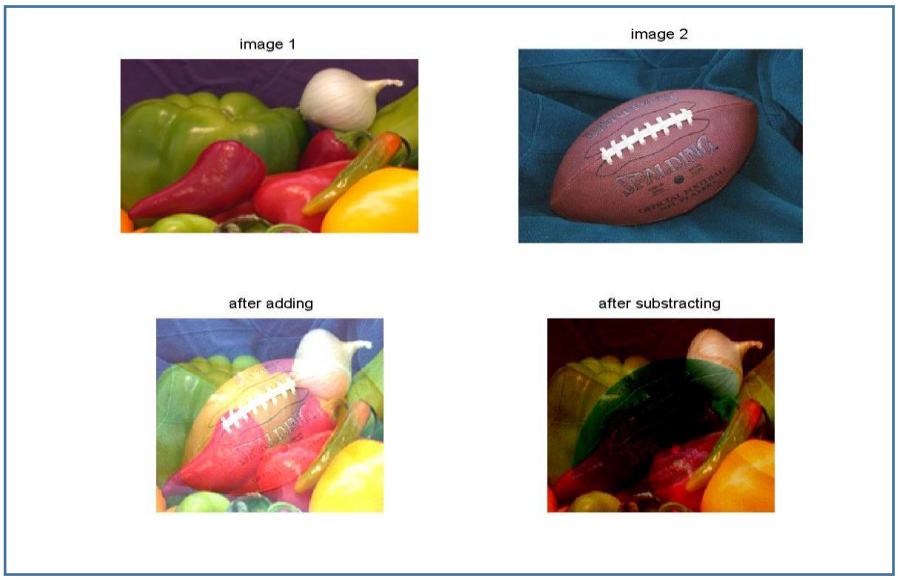
Different images may have different sizes. So in order to use mat lab in build "imadd" function we should pass equal dimensional image matrixes. Since image is an matrix we can add or subtract equal size only. That can be done using "imresize()" function.

```
I = imread('rice.png');
background =
imopen(I,strel('disk',15));
J = imsubtract(I,background);
subplot(1,2,1)
imshow(I)
title('original image')
subplot(1,2,2)
imshow(J)
title(' after subtracting')
```



- The resultant image will have the points where addition of two images respective point values. Image subtraction stuff will give you nice capabilities where you need to clear backgrounds etc.

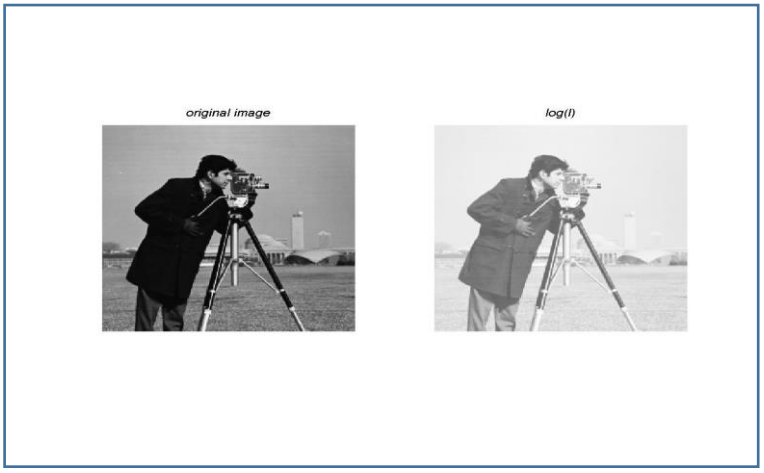
```
img1 =imread('onion.png');
img2 =imread('football.jpg');

if size(img1)==size(img2)
 addImg = imadd(img1,img2);
 subImg =
imsubtract(img1,img2);
else

newIm1 =imresize(img1, [256,
256]);
newIm2 = imresize(img2, [256,
256] );
addImg =
imadd(newIm1,newIm2);
subImg = imsubtract
(newIm1,newIm2);
end
```



## 3) . a). Log transformation

The log transformation can be used to make highly *skewed* distributions less skewed. This can be valuable both for making patterns in the data more interpretable and for helping to meet the assumptions of inferential statistics. For example Fourier spectrum values range of values that range from 0 to 10^6 or higher. Processing such as these numbers nothing to computers but image displays cannot reproduce faithfully such as wide range of values. The net effect is that intensity details can be lost in the display of typical Fourier spectrum. Therefore these cases log transformation will help.

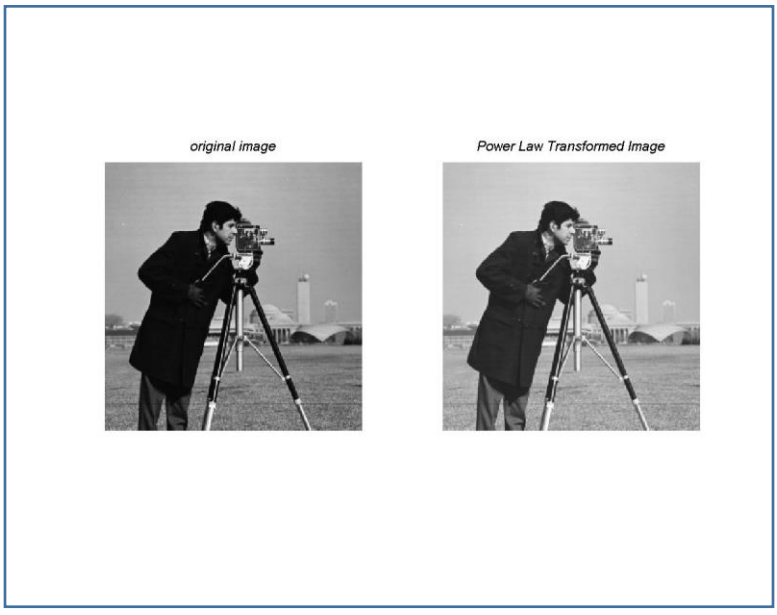
```
L=256;
 img3= imread('cameraman.tif');
c=(L-1)/log(L);

logImg =
uint8(log(double(img3)+1) .* c);
figure(2);
subplot(1, 2, 1); imshow(img3);
title('\itoriginal image');
subplot(1, 2, 2);
imshow(logImg);title('\itlog(I)'
);
```




## b). power transformation

Log transformation can be taken as a subset of power transformation. It give the family of transformation when r value varies. This transformation is called as “gamma” transformation because it uses in gamma corrections. Resultant image will be depend on the r values that we select as I mentioned earlier.

```
img3= imread('cameraman.tif');
r = double(img3)/255;
c = 1;
gamma = 0.6;
s = c*(r).^gamma;
subplot(1, 2, 1); imshow(img3);
title('\itoriginal image');
subplot(1,2,2);
imshow(s);
title('\itPower Law Transformed
Image');
```



## c). Gray level slicing

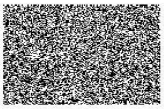
This transformation will help when we want highlight some parts in the image. This is done by display high value for area of interest and low value for the others.

```
img3= imread('cameraman.tif');
j=double(img3);
[row,col,byt]=size(j);
lowThreshold=60;
highThreshold =80;
for x=1:1:row
    for y=1:1:col
        if((j(x,y)>lowThreshold) &&
(j(x,y)<highThreshold))
            j(x,y)=255;
        else
            j(x,y)= img3(x,y);
        end
    end
end
subplot(1, 2, 1);
imshow(img3);
title('\itoriginal image');
subplot(1,2,2),
imshow(s),
title('\itGray level sliced Image');
```



## d). Bit plane slicing

In here image is decompose to bit planes and it is useful to analyzing the relative importance of each bit in an image. This is help is image compression, in which fever than all planes are used to reconstruct the image.

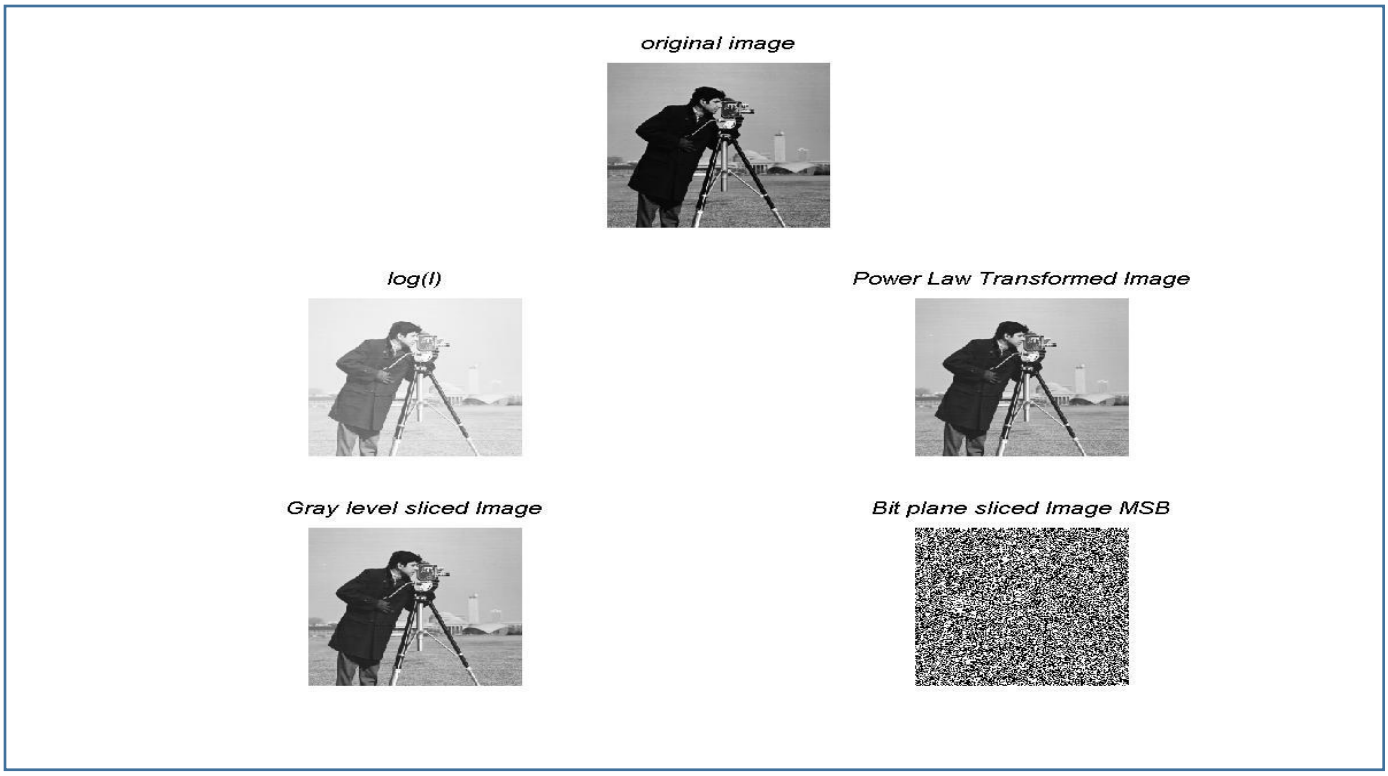










```
img3= imread('cameraman.tif');
[m,n] = size(img3);
% convert the image class from "uint8" to double
b = double(img3);
% convert each pixel into binary using matlab command "de2bi"
c = de2bi(b);
% calling the LSB Bit of each pixel
c1 = c(:,1);c2 = c(:,2);c3 = c(:,3);
c4 = c(:,4);c5 = c(:,5);c6 = c(:,6);
c7 = c(:,7);c8 = c(:,8);

subplot(2, 5, 1);
imshow(img3);
title('\itoriginal image');

r1 = reshape(c1,256,256);
subplot(2, 5, 2),
imshow(r1),
title('\it 1 st Bit plane');

r2 = reshape(c2,256,256);
subplot(2, 5, 3),
imshow(r2),
title('\it 2 nd Bit plane');

r3 = reshape(c3,256,256);
subplot(2, 5, 4),
imshow(r3),
title('\it 3 rd Bit plane');

r4 = reshape(c4,256,256);
subplot(2, 5, 5),
imshow(r4),
title('\it 4 th Bit plane');

r5 = reshape(c5,256,256);
subplot(2, 5, 6),
imshow(r5),
title('\it 5 th st Bit plane');

r6 = reshape(c6,256,256);
subplot(2, 5, 7),
imshow(r6),
title('\it 6 th Bit plane');

r7 = reshape(c7,256,256);
subplot(2, 5, 8),
imshow(r7),
title('\it 7 th Bit plane');

r8 = reshape(c8,256,256);
subplot(2, 5, 9),
imshow(r8),
title('\it 8 th Bit plane');
```




## 4). Masking

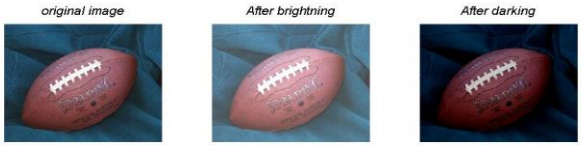
Masking involves setting some of the pixel values in an image to zero, or some other value. Masking can be done in one of two ways: Using an image as a mask. A mask image is simply an image where some of the pixel intensity values are zero or other value, and others are unchanged.

```
maskimg = imread('cameraman.tif');
[rNum, cNum, ~] = size(maskimg);
centerX = ceil(cNum/2);
centerY = ceil(rNum/2);
% Define parameters of the rectangle
windowWidth = 128;
windowHeight = 128;
% Create logical mask
[yy, xx] = ndgrid((1:rNum)-centerY,
(1:cNum)-centerX);
mask = xx < -windowWidth/2 | xx >
windowWidth/2 | ...
    yy < -windowHeight/2 | yy >
windowHeight/2;
% Mask image and show it
maskedImage(mask) = 0;
subplot(1,2,1),imshow(maskimg),
title('\itBit plane sliced Image
MSB');
subplot(1,2,2),imshow(maskedImage),
title('\itAfter masking');
```

## 5). Brightness

We can use arithmetic operation on image to adjust the brightness of an image. When, we add a constant matrix to image matrix brightness will be high, reverse otherwise.

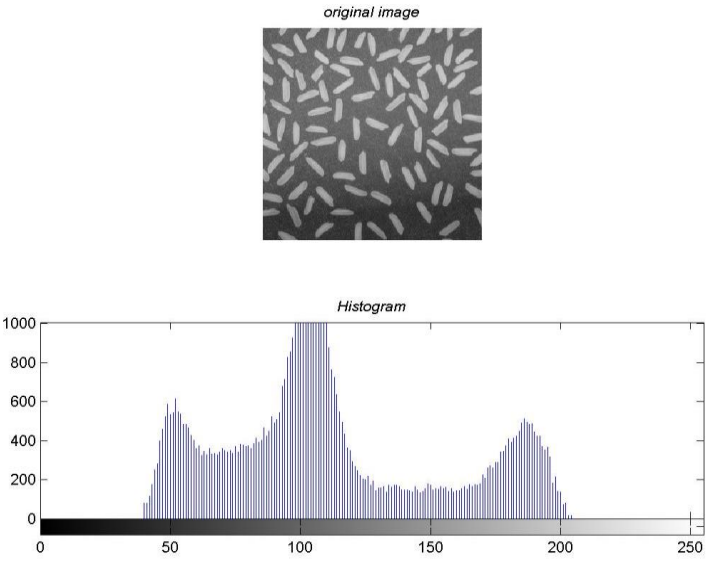
```
img5 =imread('football.jpg');
increase =50;
decrease =50;
inbrImg = img5 + increase ;
debrImg = img5 - decrease ;
figure(4);
subplot(1,3,1),imshow(img5),
title('\itoriginal image');
subplot(1,3,2),imshow(inbrImg),
title('\itAfter brightning');
subplot(1,3,3),imshow(debrImg ),
title('\itAfter darking');
```




## 6). Histogram

An image histogram is a chart that shows the distribution of intensities in an indexed or grayscale image.

```
img6 =imread('rice.png');
figure(5);
subplot(2,1,1),imshow(img6),
title('\itoriginal image');
subplot(2,1,2),imhist(img6),
title('\itHistogram');
```



For RGB Image :

If the input is a multi-column array, hist creates histograms for each column of x and overlays them onto a single plot. hist(x,nbins) sorts x into the number of bins specified by the scalar nbins.

```
I =imread ('pears.png');
 if (size(I, 3) ~= 3)
        error('Input image must be
RGB.')
 end
nBins = 256;
rHist = imhist(I(:,:,1), nBins);
gHist = imhist(I(:,:,2), nBins);
bHist = imhist(I(:,:,3), nBins);
hFig = figure;
subplot(2,1,1);
imshow(I);
title('Input image');
subplot(2,1,2);
h(1) = area(1:nBins, rHist,
'FaceColor', 'r');
hold on;
h(2) = area(1:nBins, gHist,
'FaceColor', 'g');
hold on;
h(3) = area(1:nBins, bHist,
'FaceColor', 'b');
hold on;
title('RGB image histogram');
```