Vy Bui

Homework 2

ENGR 652

Vy Bui

ENGR652

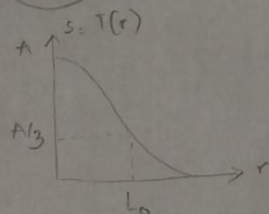
Homework 2

3.1 Let denote:

- g : the desired function
- f : the original image

To normalize the intensities within the range $[0, L-1]$. First we need to derive a function g in the range $[0,1]$ by $g' = \frac{f - f_{min}}{max(f - f_{min})}$. Next, we multiply g' by $L-1$, this yields : $g = \frac{f - f_{min}}{max(f - f_{min})} \cdot (L-1)$ which gives the values in range $[0, L-1]$.

3.2 a) General form : $s = T(r) = Ae^{-kr^2}$



Based on the figure : $\rightarrow \quad Ae^{-kr^2} = \frac{A}{3}$

$$\Leftrightarrow \quad e^{-kr^2} = \frac{1}{3}$$

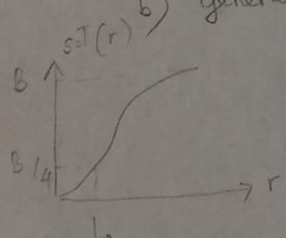
$$\Leftrightarrow \quad -kr^2 = \ln\frac{1}{3}$$

$$\Leftrightarrow \quad -kL_0^2 = \ln\frac{1}{3}$$

$$\Leftrightarrow \quad k = -\frac{\ln\frac{1}{3}}{L_0^2}$$

Therefore, the transformation function is : $s = T(r) = Ae^{\left(\frac{\ln\frac{1}{3}}{L_0^2}\right)r^2}$

b) General form : $s = T(r) = B\left(1 - e^{-kr^2}\right)$



Based on the figure : $\rightarrow \quad B\left(1 - e^{-kr^2}\right) = \frac{B}{4}$

$$\Leftrightarrow \quad 1 - e^{-kr^2} = \frac{1}{4}$$

$$\Leftrightarrow \quad e^{-kr^2} = 1 - \frac{1}{4} = 0.75$$

$$\Leftrightarrow \quad -kr^2 = \ln 0.75$$

$$\Leftrightarrow \quad k = -\frac{\ln 0.75}{r^2} = -\frac{\ln 0.75}{L_0^2}$$

Therefore, the transformation function is : $s = T(r) = B\left(1 - e^{\left(\frac{\ln 0.75}{L_0^2}\right)r^2}\right)$

c) General form : $s = T(r) = (D - C)\left(1 - e^{-kr^2}\right) + C$

(3.6) Unlike its continuous counterpart, discrete histogram equalization cannot be proved in general that it results in a flat histogram because a histogram is an approximation to a PDF, and no new allowed intensity levels are created in the process in order to redistribute the pixel intensities.

(3.18) a) In a $n \times n$ filter mask, there are n^2 points. n is odd, the median is found by sorting the values then the value at the position $\frac{n+1}{2}$. There are $\frac{n^2-1}{2}$ values is less than or equal to median value and $\frac{n^2-1}{2}$ values is greater or equal to median value. When the isolated cluster (has area $A_{cluster}$) is less than $\frac{1}{2} A_{filter} = \frac{1}{2} n^2$

$\rightarrow A_{cluster} \leq \frac{n^2-1}{2}$

b) Conditions of n that make one / more of clusters cease to be isolated as in (a) is that $A_{cluster} \leq \frac{n^2-1}{2}$, aslo means n is odd. In the cas n is even, median is computed by the average of the values in positions $\frac{n}{2}$ & $\frac{n}{2}+1$.

(3.23) The averaging filter & Laplacian filter are linear operations so the result doesn't change if the order of these operations were reversed.

(3.25) A significant improvement in sharpness of 3.38 (e) over 3.38(d) because using the Laplacian filter w/ -8 in the center provides additional differentiation (sharpening) in the diagonal directions

$$(1)\ \nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

$$(2)\ \nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 8f(x,y)$$

The Laplacian mask w/ -4 (1) in the center performs a differential operation in the horizontal & vertical directions. The Laplacian mask w/ -8 (2) in the center performs a differential operation in the horizontal, vertical & diagonal directions which means that it will detect the change in 3 directions compare to 2 directions as (1). Thus, (2) produces shaper result.

(3.27) Gaussian filter: $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$.

$$g_{mask}(x,y) = f(x,y) - G(x,y)$$

$c = 1$

$G(x, y)$

0.003	0.0133	0.0219	0.0133	0.003
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.003	0.0133	0.0219	0.0133	0.003

3.28 Subtracting the laplacian from an image :

$$f(x,y) - \nabla^2 f(x,y) = f(x,y) - \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)\right]$$

$$= 6f(x,y) - \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)\right]$$

$$= 5\left\{\frac{6}{5} f(x,y) - \frac{1}{5}\left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)\right]\right\}$$

$$= 5\left\{\frac{6}{5} f(x,y) - \underbrace{\bar{f}(x,y)}_{\text{average of } f(x,y)}\right\}$$

We can write : $f(x-y) - \nabla^2 f(x,y) \sim \underbrace{f(x,y) - \bar{f}(x,y)}_{\text{definition of unsharp masking}}$

Therefore, we can conclude that substract the laplacian from an image is proportional to unsharp masking.

MATLAB Exercise 1

```
function [ out ] = imlin( in, G, B)
out = (in - min(in(:)))*255/(max(in(:)) - min(in(:)));
out = G * out + B;
end
```

MATLAB Exercise 2

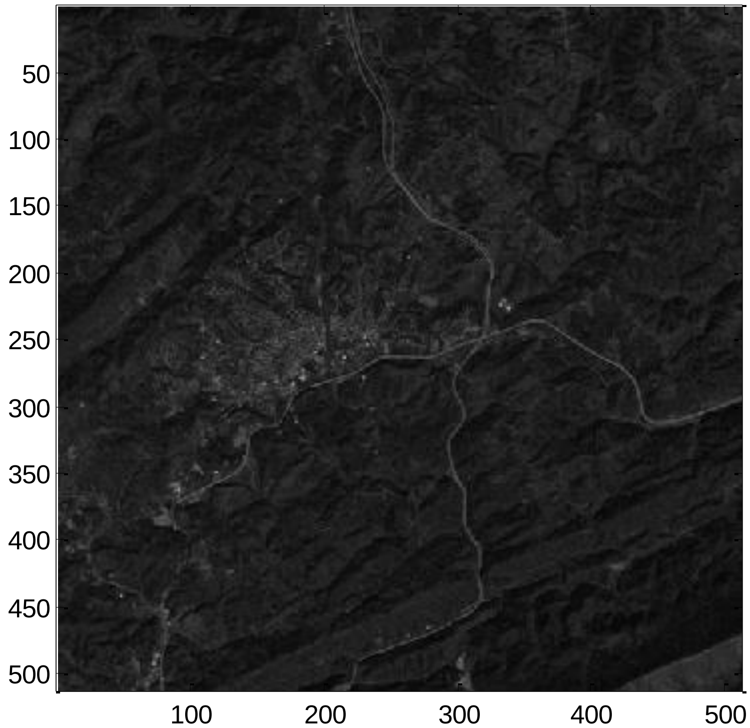
```
function [ out ] = imlin2( in, a, b)
out = (in - min(in(:)))*(b - a)/((max(in(:)) - min(in(:))) + a;
end
```

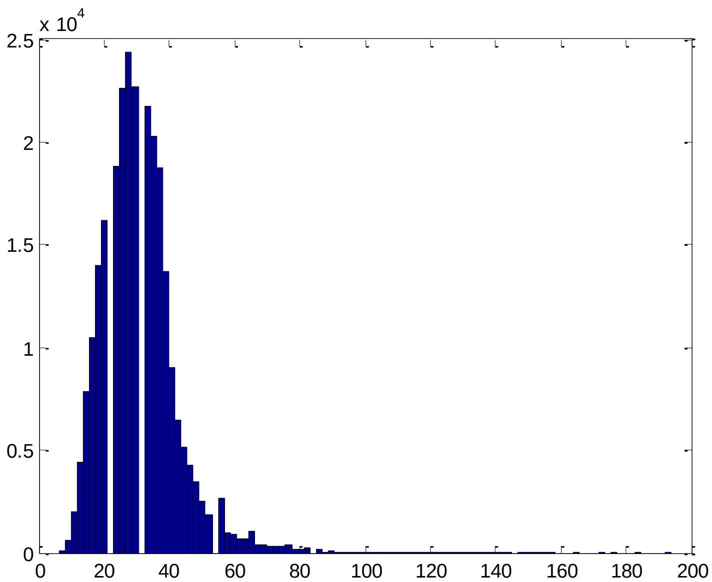
MATLAB Exercise 3

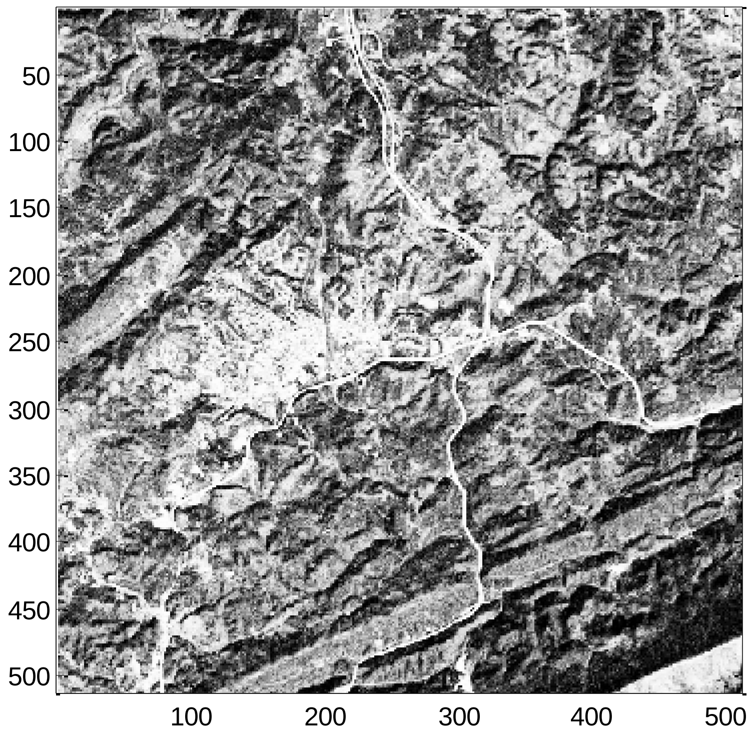
```
function [out,F]=myhisteq(in,L);
%
% [out,F]=myhisteq(in,L);
%
% My histogram equalization function
% Takes an L level input image 'in' (must be integers)
% and maps it to an L Level output
% with an approximately flat histogram.
%
% out       equalized image (L levels)
% F         mapping function
% in        input image (L levels, integer values)
% L         # levels in input image
im_size = size(in);
row = im_size(1);
col = im_size(2);
total_pixels = row*col;
j = 1;
s = 0;
out = in;
for rk = 0:L-1;
    nk(j) = length(find(in == rk));
    pr(j) = nk(j)/total_pixels;
    s = s + (L-1)*pr(j);
    sk(j) = s;
    out(find(in == rk)) = sk(j);
    j = j + 1;
end
F = sk;
end
```

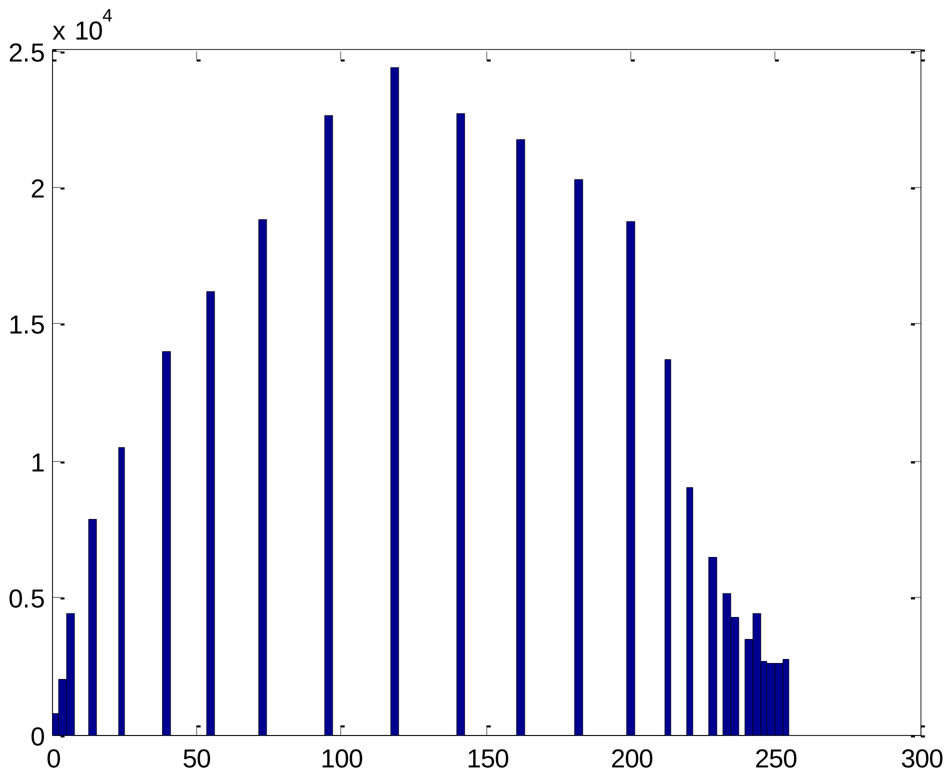
```
load wva
test = wva(:,:,1);
hist(test(:),100); figure;
image(test); colormap(gray(256)); axis image

test2=myhisteq(test,256);
figure; hist(test2(:),100); figure;
image(test2); colormap(gray(256)); axis image
```


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500


x 10^4
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140
160
180
200


50
100
150
200
250
300
350
400
450
500
100
200
300
400
500

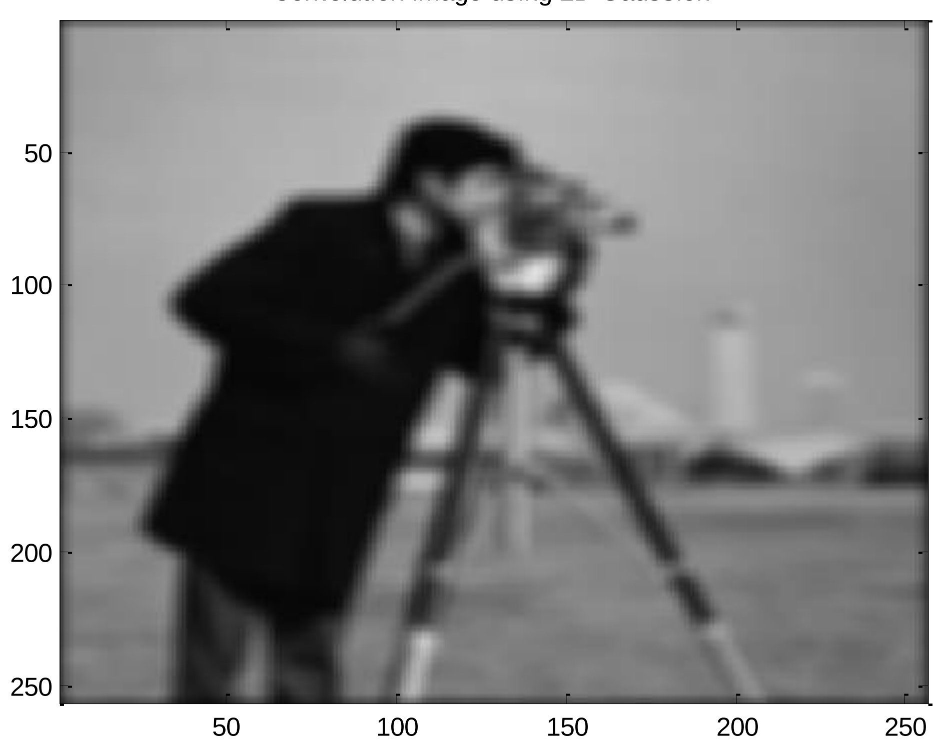

x 10^4
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
300

MATLAB Exercise 4

```
clear
clc
%% Convolution image using 2D Gaussion
x = double(imread('cameraman.tif'));
figure
image(x)
colormap(gray(256));
title('Original Image');
% Gauss_filter = fspecial('gaussian', [5 5], 3);
hx = fspecial('gaussian',[1,5],3) ;
hy = fspecial('gaussian',[5,1],3) ;
hx2=repmat(hx,[5,1]);
hy2=repmat(hy,[1,5]);
h=hx2.*hy2;
tic
conv2D = conv2(x,h,'same');
toc
figure
image(conv2D);
colormap(gray(256));
title('Convolution image using 2D Gaussion');
%% Convolution image using 1D Gaussion
row_Gauss = fspecial('gaussian',[1,5],3);
col_Gauss = fspecial('gaussian',[5,1],3) ;
[row col] = size(x);
im_row = reshape(x',1,row*col);
tic
row_conv = conv(im_row,row_Gauss,'same');
toc
row_conv = reshape(row_conv, col, row);
row_conv = row_conv';
im_col = (row_conv(:))';
tic
col_conv = conv(im_col,col_Gauss','same');
toc
col_conv = reshape(col_conv, row, col);
figure
image(col_conv);
colormap(gray(256));
title('Convolution image using seperated Gaussian vector filter');
%% Compare conv 1D and 2D
diff = conv2D - col_conv;
figure
imshowpair(conv2D,col_conv,'diff');
```

Original Image

Convolution image using 2D Gaussion



Convolution image using seperated Gaussian vector filter

Filter Size	Time 2D	Time 1D
2x2	0.000748 seconds	0.001262 + 0.001134 seconds
5x5	0.001052 seconds	0.001336 + 0.001302 seconds
9x9	0.002443 seconds	0.000735 + 0.000977 seconds

The difference between these two techniques is compared in Line 41-44. It’s observed that there’s only different at the edges of the two images.