**Image and Video Processing**

**Programming Assignment – Report**

**Week2**

%% Homework 2

%% Problem 1

% Write a Matlab for implementing filtering of a gray scale image. Your

% program should allow you to specify the filter with an arbitrary size

% (but for simplicity, you can assume the filter size is KxL where both K

% and L are odd numbers, and the filter origin is at the center.

inImg = imread('barbara\_gray.bmp');

grayImg=single(inImg);

%my\_filter=[1 2 1; 2 4 2; 1 2 1];

my\_filter = input('Enter the filter as a matrix: ');

%perform convolution

%tmpImg = my\_conv2\_sol(grayImg, my\_filter); %%% here you need to write your own function as my\_conv2

tmpImg = computeConv(grayImg, my\_filter);

%% scale tmpImg so that it ranges in 0 to 255 and is stored in uint8

% function my\_mat2gray(img) returns an image that ranges 0 to 1

filteredImg=uint8(255\*my\_mat2gray(tmpImg));

%% iv) display the original and the filtered images. you can use imshow for displaying an image.

%% make sure to conver all images into uint8 before dispalying them

cf=figure(1);

subplot(1,2,1);

imshow(inImg);

title('Original Image');

subplot(1,2,2);

imshow(filteredImg);

title('Filtered Image with Filter 3');

print(cf, 'HW2\_Q1', '-dtiff');

%computeConv.m

function [ outImg ] = computeConv( img, filt )

%UNTITLED Convolutes the image with the filter matrix

%

outImg = img;

dimImg = size(img);

dimFilt = size(filt);

imgBoundary = (dimFilt - 1)\*0.5

disp('Beginning Convolution')

for i = imgBoundary(1) + 1:dimImg(1) - imgBoundary(1)

for j = imgBoundary(2) + 1:dimImg(2) - imgBoundary(2)

temp = 0;

% for k = -imgBoundary(1):imgBoundary(1)

% for l = -imgBoundary(2):imgBoundary(2)

% temp = temp + img(i-k,j l)\*filt(k+imgBoundary(1)+1,l+imgBoundary(2)+1);

% end

% end

% outImg(i,j) = temp;

outImg(i,j) = sum(sum(img(i-imgBoundary(1):i+imgBoundary(1),j-imgBoundary(2):j+imgBoundary(2)).\*filt));

end

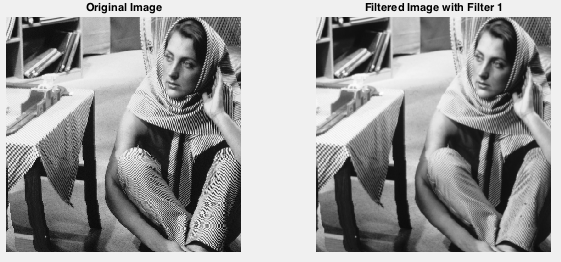
end

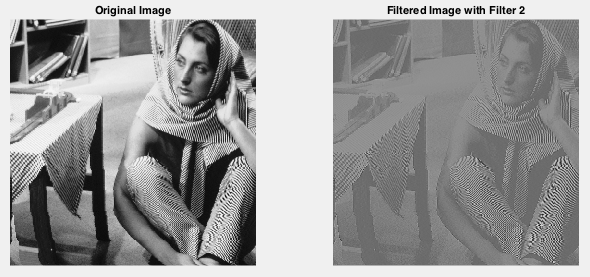
disp('Convolution complete.')

%fin.

end

Images and Output:







%% Homework 2

%% Problem 2

% Write a Matlab to simulate noise removal.

%%

% i) First create a noisy image, by adding zero mean Gaussian random noise

% at two different noise levels (0.01 and 0.1) to your image using

% "imnoise()". You can specify the noise variance in "imnoise( )").

inImg = imread('barbara\_gray.bmp');

%convert the inImg to int or single or double before proceeding!

grayImg=double(inImg);

noiseLevel=0.1;

noisy\_img = imnoise(inImg, 'gaussian', 0, noiseLevel);

%imnoise does not work when inImg is not uint8!

noisy\_img=single(noisy\_img);

%%

% ii) Then apply an averaging filter to the noise added image. For a chosen

% variance of the added noise, you need to try different window sizes (from

% 3x3 to 9x9) to see which one gives you the best trade-off between noise

% removal and blurring. Your program should allow the user to specify the

% window size as an input parameter.

%set filter

filterSize=input('Enter window size (integer): ');

denoising\_filter=ones(filterSize,filterSize)/(filterSize\*filterSize);

tmpImg = computeConv(noisy\_img, denoising\_filter); %%% here you need to write your own function as my\_conv2

%%

% III) Normalize and convert the image to uint8

denoisedImg = 255\*my\_mat2gray(tmpImg);

%% iv) display the original and the filtered images.

cf=figure(2);

subplot(1,3,1);

imshow(inImg);

title('Original Image');

subplot(1,3,2);

imshow(noisy\_img,[]);

title(['Noisy Image with NoiseLevel=' num2str(noiseLevel)]);

%note you need to use [] option to see the full range of noisy image

subplot(1,3,3);

imshow(denoisedImg,[]);

title(['Denoised Image with NoiseLevel=' num2str(noiseLevel) 'FilterSize=' num2str(filterSize)]);

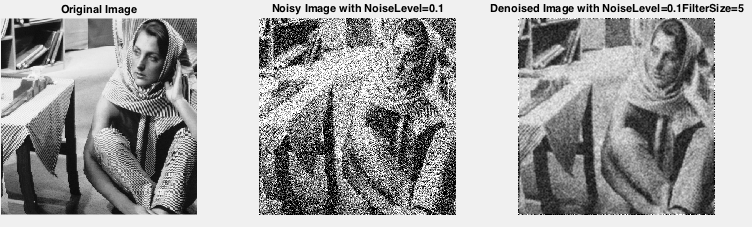
% v) save the filtered images into another file.

print(cf, 'HW2\_Q2', '-dtiff');

%%

Images and Output:





Observations:

Since 0.1 noise is as high as the frequency of the image at a lot of regions (ex – the table cloth and scarf), a lot of such details are lost along with the noise when a filter is applied.