**Digital Image Processing – Homework #3**

**Juan Silva**

**ECE 595**

**Sept. 25, 2019**

1. Use the Porsche image to perform the following operations:   
     
   a. Smoothing with a 3x3 linear filter   
   b. Median filtering with a 3x3 support set   
   c. Unsharp masking with a 3x3 linear filter, k = 1.75.   
   d. Unsharp masking with a 3x3 median filter, k = 1.75.

Print the original and the output images, and comment about the effect of the filters.

**hw4\_q1.m:**

clear, clc, close all

format short, format compact

%===|Question 1|====%

%(a)

%Open image & create imate

porsche = imread('Porsche.tiff');

%subplot(3, 2, 1);

title('No Filter');

imshow(porsche);

figure

%Apply smoothing filter

temp1 = fspecial('average', [3 3]);

smooth = imfilter(porsche, temp1, 'replicate');

%subplot(3, 2, 2);

title('Smoothing Filter');

imshow(smooth);

figure

%(b)

%Apply median filter

median = medfilt2(porsche, [3 3]);

%subplot(3, 2, 3);

title('Median Filter');

imshow(median);

%(c)

%Unsharp masking with 3x3 linear filter (k = 1.75)

k = 1.75;

figure

%mask = orignal - filter

%unsharp = original + k \* mask

linear\_mask = porsche - smooth;

unsharp1 = porsche + k \* linear\_mask;

%subplot(3, 2, 4);

title('Unsharp, Linear, k = 1.75');

imshow(unsharp1);

%(d)

%Unsharp masking with 3x3 median filter (k = 1.75)

figure

%mask = orignal - filter

%unsharp = original + k \* mask

median\_mask = porsche - median;

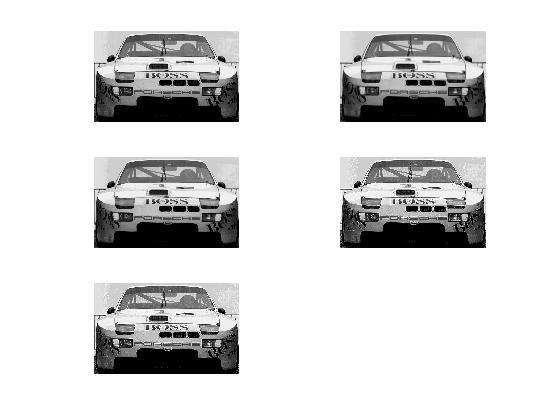
unsharp2 = porsche + k \* median\_mask;

%subplot(3, 2, 5);

title('Unsharp, Median, k = 1.75');

imshow(unsharp1);

**Results**



1. Write a Matlab program to generate an image composed of two concentric circles as shown below. The inner circle should have a radius of 50 pixels and a mean value of 192. The outer circle should have a radius of 100 pixels and a mean value of 128. The background should have a mean value of 64. Add uniform random noise to each pixel in the range ‐16 … +16 (see Matlab’s rand function). Run your program and save the image in “tif” format.

**hw4\_q2.m:**

clear, clc, close all

format short, format compact

%===|Question 2|====%

%Generate an image composed of two circles

%Inner circle has a radius of 50 pixels with mean of 192

%Outer circle has a radius of 100 pixels with mean of 64

%Circle Radius

R1 = 100; %inner circle

R2 = 50; %outer circle

%Image array

N = 400;

I = 64\*(ones(N,N));

%generate

for row = I : N

for col = 1 : N

if(row - N/2)^2 + (col - N/2)^2 < R2^2

I(row,col) = 192;

elseif(row - N/2)^2 + (col - N/2)^2 < R1^2

I(row,col) = 128;

end

end

end

%Generate noise: r = a + (b-a).\*rand(100,1)

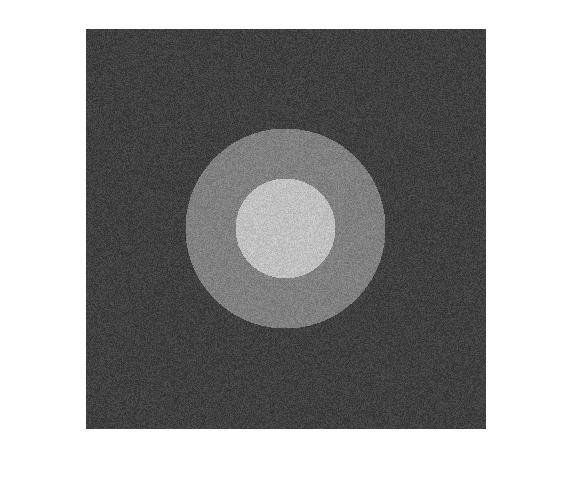
I = I + (16 - (-16)) \* (rand(N) - 0.5);

imshow(I, [0 255]);

%Write to a file

imwrite(uint8(I), 'circles.tif');

**Results**



1. Develop a program to resize the example image “cameraman.tif” from its original size of 256x256 to an enlarged size of 400x400, using bilinear interpolation. Do not use Matlab high level functions. Apply your program to the image and include the output image in your solution.

**hw4\_q3.m:**

clear, clc, close all

format short, format compact

%===|Question 3|====%

%Read image

cameraman = imread('cameraman.tiff');

[M1,N1] = size(cameraman);

%Assign new size

M2 = 400;

N2 = 400;

resize = zeros(M2,N2);

%Determine ratios

x1 = (N2 / N1);

y1 = (M2 / M1);

%

for y = 1 : M2

for x = 1 : N2

%find single pixel on original image

v = x / x1;

w = y / y1;

%have to round to find lowest values

low\_v = floor(v);

low\_w = floor(w);

%Ensures not out of bounds

if low\_v < 1

low\_v = 1;

end

if low\_w < 1

low\_w = 1;

end

if low\_v > N1 - 1

low\_v = N1 - 1;

end

if low\_w > M1 - 1

low\_w = M1 - 1;

end

%offset from center

x0 = v - low\_v;

y0 = w - low\_w;

%Bilinear Interpolation

resize(x,y) = cameraman(low\_w, low\_v) \* (1 - x0) \* (1-y0) + ...

cameraman(low\_w, low\_v + 1) \* x0 \* (1 - y0) + ...

cameraman(low\_w + 1, low\_v) \* (1 - x0) \* y0 + ...

cameraman(low\_w + 1, low\_v + 1) \* x0 \* y0;

end

end

imshow(resize, []);

**Results**



1. For each of the following proposed filter that will only be applied to pixels in the corrupted columns (every 5th column). Pixels not in the corrupted columns are identical to the original image and will be left unchanged. Rank the approaches in terms of what you think the mean squared error (MSE) would be between the filtered image and the original. Ignore boundary effects such as zero‐padding at the edges. Give an explanation for your ranking.

*SKIPPED.*

1. Write a Matlab program which takes as input arguments a binary image and a percentage p of pixels to be randomly flipped to generate salt and pepper noise. Flipping a pixel means that if it is white it will be made black, and if it is black it will be made white. Use your routine to generate 3 different versions of the binary eyechart: one with low levels of noise (p = 1%), one with a medium level of noise (p = 5%), and one with lots of noise (p = 20%).

clear, clc, close all

format short, format compact

%===|Question 5|====%

**hw4\_q5.m:**

im = imread('Porsche.tiff');

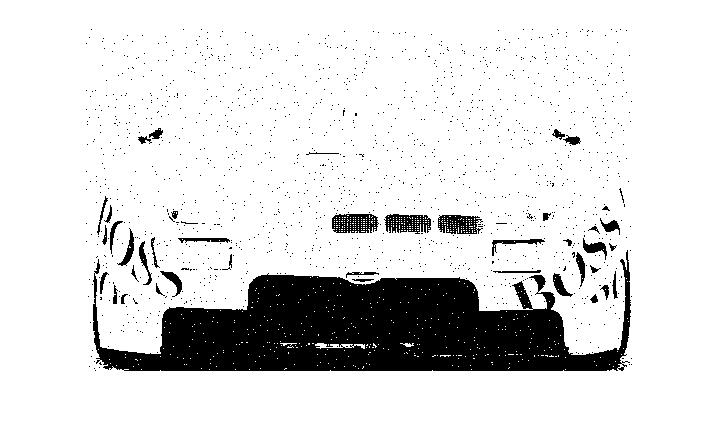
p = input('Enter value for p: ');

%Generates noise based on value of p

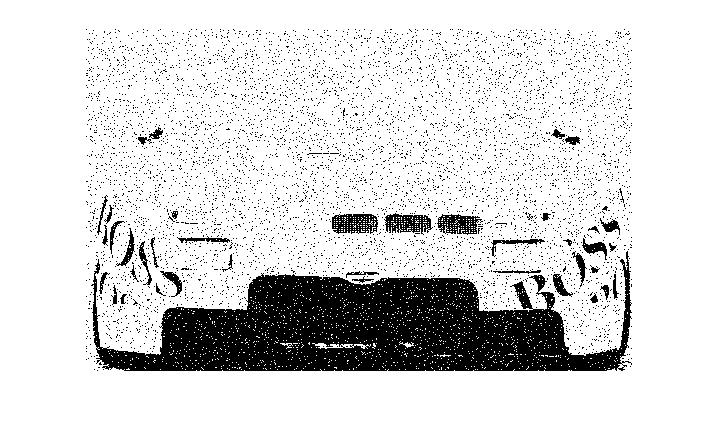
out = xor(im, rand(size(im)) < p / 100);

imshow(out);

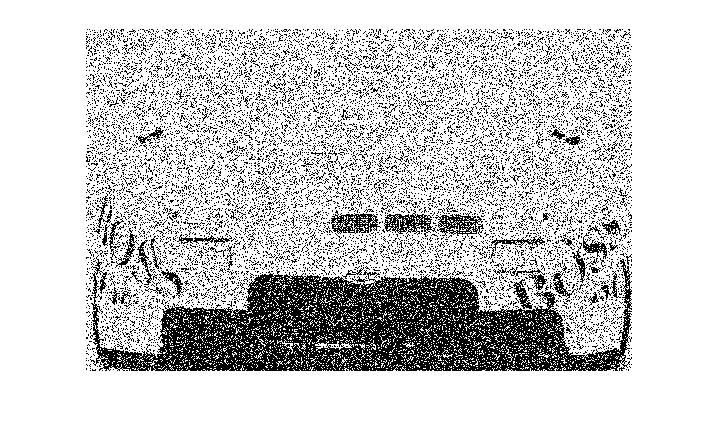
**Results**



p = 1%



p = 5%



p = 20%

1. Show that subtracting the Laplacian from an image is proportional to unsharp masking. Use the definition for the Laplacian given in Eq. (3.6-6).

