Colin Keenan

ECE 5470

2-17-2020

Homework 3

1. Histogram Equalization

**Compute Histogram of an image**

clear

fprintf("Question 1, Part b:")

image = imread('Fig3-1.jpg');

hist = zeros(256,1);

for i = 1:size(image,1)

for j = 1:size(image,2)

r = image(i,j);

hist(r+1) = hist(r+1) + 1;

end

end

figure();

subplot(1,2,1)

imshow(image);

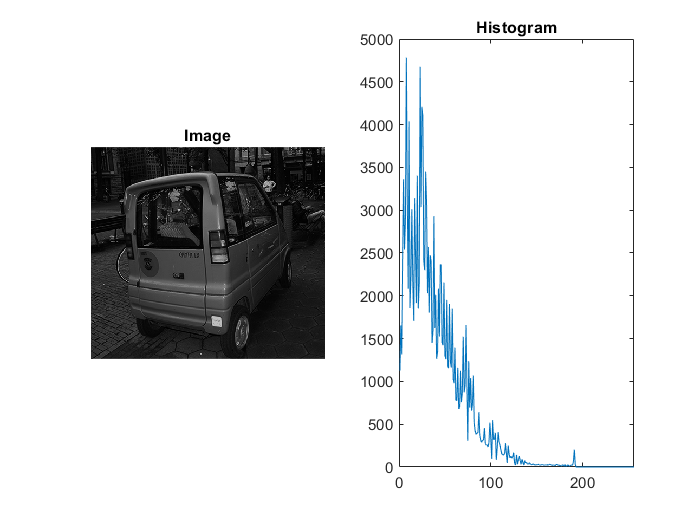
title("Image");

subplot(1,2,2)

plot(hist);

xlim([0 256]);

title("Histogram");



Explanation: The code above looks at each pixel and determines the level of intensity. Then in an array spanning the values of intensities, in the case of 8-bit depth, 256 values from 0-255, but since MATLAB starts at index 1, and the value at the index of that intensity is incremented. This clearly shows that the image out of the box is heavily favored towards the dark end of the spectrum.

**Use Histogram Equalization on image Fig3-1.jpg**

clear

fprintf("Question 1, Part c,d:")

file = 'Fig3-1.jpg';

image = imread(file);

hist = histogram(image);

prob = hist ./ (size(image,1) \* size(image,2));

info = imfinfo(file);

bits = info.BitDepth;

levels = 2^bits;

image\_mod = zeros(size(image));

for i = 1:size(image,1)

for j = 1:size(image,2)

level = image(i,j);

prob\_sum = summation(prob,1,level);

new\_level = (levels-1) \* prob\_sum;

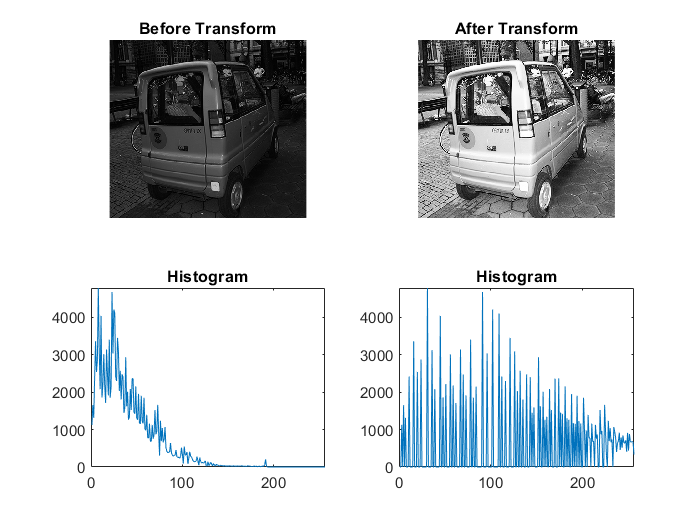
image\_mod(i,j) = round(new\_level) / 255;

end

end

hist\_mod = histogram(image\_mod .\* 255);

figure();

subplot(2,2,1)

imshow(image);

title("Before Transform");

subplot(2,2,2)

imshow(image\_mod);

title("After Transform");

subplot(2,2,3)

plot(hist);

xlim([0 256]);

title("Histogram");

subplot(2,2,4)

plot(hist\_mod);

xlim([0 256]);

title("Histogram");

Explanation: This code takes the histogram of the image and attempts to evenly split each value across the entire spectrum 0-255, in the case of 8-bit depth. This is done by getting the number of times a certain intensity is used and generating a probability function of those intensities. Then the new level is generated by multiplying the possible levels minus 1 by the cumulative probability of that level starting at level 0 and then rounding since each level is discrete. As you can see in the output image, it is much brighter and a lot of the darker details are clear now, such as the cobblestone road, and the sidewalk.

2. Smoothing Spatial Filtering

**Smoothing Spatial Filtering with masks of 3x3 and 5x5 with coefficients of 1**

clear

fprintf("Question 2, Part a,b,c:")

file = 'donation.tif';

image = imread(file);

image\_double = im2double(image);

image\_mod\_3x3 = zeros(size(image\_double));

image\_mod\_5x5 = zeros(size(image\_double));

mask\_3x3 = [[1 1 1]

[1 1 1]

[1 1 1]];

weight\_3x3 = sum(sum(mask\_3x3));

for i = 1+1 : size(image\_double,1)-1

for j = 1+1 : size(image\_double,2)-1

frame\_3x3 = [[image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1)]

[image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1)]

[image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1)]];

masked\_3x3 = frame\_3x3 .\* mask\_3x3;

new\_level = sum(sum(masked\_3x3)) / weight\_3x3;

image\_mod\_3x3(i,j) = new\_level;

end

end

mask\_5x5 = [[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]];

weight\_5x5 = sum(sum(mask\_5x5));

for i = 1+2 : size(image\_double,1)-2

for j = 1+2 : size(image\_double,2)-2

frame\_5x5 = [[image\_double(i-2,j-2) image\_double(i-2,j-1) image\_double(i-2,j+0) image\_double(i-2,j+1) image\_double(i-2,j+2)]

[image\_double(i-1,j-2) image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1) image\_double(i-1,j+2)]

[image\_double(i+0,j-2) image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1) image\_double(i+0,j+2)]

[image\_double(i+1,j-2) image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1) image\_double(i+1,j+2)]

[image\_double(i+2,j-2) image\_double(i+2,j-1) image\_double(i+2,j+0) image\_double(i+2,j+1) image\_double(i+2,j+2)]];

masked\_5x5 = frame\_5x5 .\* mask\_5x5;

new\_level = sum(sum(masked\_5x5)) / weight\_5x5;

image\_mod\_5x5(i,j) = new\_level;

end

end

figure();

subplot(1,3,1)

imshow(image);

title("Before Transform");

subplot(1,3,2)

imshow(image\_mod\_3x3);

title("After 3x3 Smooth Filter");

subplot(1,3,3)

imshow(image\_mod\_5x5);

title("After 5x5 Smooth Filter");



Explanation: This code provides basic smoothing with a 3x3 and 5x5 masks filled with equal weights of 0. In effect for each new pixel, all 8 pixels around it, along with the original are averaged and that average is the new value. The 3x3 mask appears to be the winner in this instance due to smoothing out some of the grain, but also does not take away a great deal of detail from the image.

**Perform Spatial Filtering and thresholding on Fig3-2b.tif**

clear

fprintf("Question 2, Part d:")

file = 'Fig3-2b.tif';

image = imread(file);

image\_double = im2double(image);

% Spatial Filtering

image\_mod\_3x3 = zeros(size(image\_double));

mask\_3x3 = [[1 1 1]

[1 1 1]

[1 1 1]];

weight\_3x3 = sum(sum(mask\_3x3));

for i = 1+1 : size(image\_double,1)-1

for j = 1+1 : size(image\_double,2)-1

frame\_3x3 = [[image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1)]

[image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1)]

[image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1)]];

masked\_3x3 = frame\_3x3 .\* mask\_3x3;

new\_level = sum(sum(masked\_3x3)) / weight\_3x3;

image\_mod\_3x3(i,j) = new\_level;

end

end

mask\_5x5 = [[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]

[1 1 1 1 1]];

weight\_5x5 = sum(sum(mask\_5x5));

for i = 1+2 : size(image\_double,1)-2

for j = 1+2 : size(image\_double,2)-2

frame\_5x5 = [[image\_double(i-2,j-2) image\_double(i-2,j-1) image\_double(i-2,j+0) image\_double(i-2,j+1) image\_double(i-2,j+2)]

[image\_double(i-1,j-2) image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1) image\_double(i-1,j+2)]

[image\_double(i+0,j-2) image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1) image\_double(i+0,j+2)]

[image\_double(i+1,j-2) image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1) image\_double(i+1,j+2)]

[image\_double(i+2,j-2) image\_double(i+2,j-1) image\_double(i+2,j+0) image\_double(i+2,j+1) image\_double(i+2,j+2)]];

masked\_5x5 = frame\_5x5 .\* mask\_5x5;

new\_level = sum(sum(masked\_5x5)) / weight\_5x5;

image\_mod\_5x5(i,j) = new\_level;

end

end

% Thresholding

upper\_range = 255/255;

lower\_range = 200/255;

transform = 1:256;

for x = 1:256

if((x >= lower\_range\*255) && (x <= upper\_range\*255))

transform(x) = 255;

else

transform(x) = 0;

end

end

image\_mod\_thresh\_3x3 = zeros(size(image\_mod\_3x3));

for i = 1:size(image\_mod\_3x3,1)

for j = 1:size(image\_mod\_3x3,2)

r = image\_mod\_3x3(i,j);

if((r >= lower\_range) && (r <= upper\_range))

image\_mod\_thresh\_3x3(i,j) = 1;

else

image\_mod\_thresh\_3x3(i,j) = 0;

end

end

end

image\_mod\_thresh\_5x5 = zeros(size(image\_mod\_5x5));

for i = 1:size(image\_mod\_5x5,1)

for j = 1:size(image\_mod\_5x5,2)

r = image\_mod\_5x5(i,j);

if((r >= lower\_range) && (r <= upper\_range))

image\_mod\_thresh\_5x5(i,j) = 1;

else

image\_mod\_thresh\_5x5(i,j) = 0;

end

end

end

figure();

subplot(3,2,1)

imshow(image);

title("Before Transform");

subplot(3,2,2)

plot(transform);

xlim([0,255]);

ylim([0,255]);

title("Plot for Thresholding on Image");

subplot(3,2,3)

imshow(image\_mod\_3x3);

title("After 3x3 Smooth Filter");

subplot(3,2,4)

imshow(image\_mod\_thresh\_3x3);

title("After 3x3 Smooth Filter and Thresholding");

subplot(3,2,5)

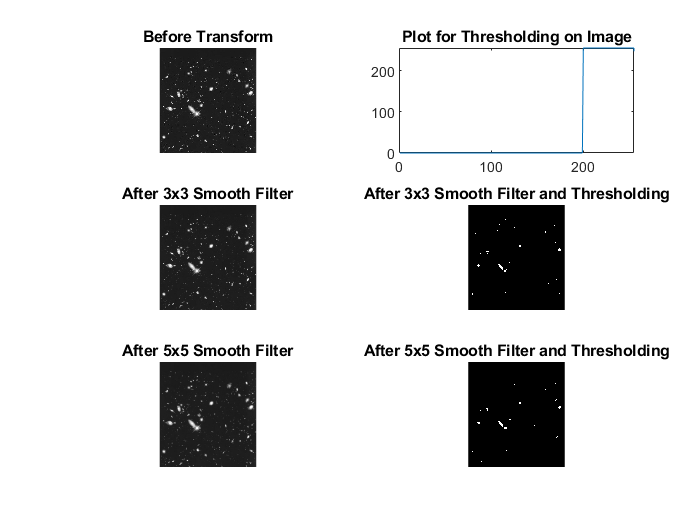
imshow(image\_mod\_5x5);

title("After 5x5 Smooth Filter");

subplot(3,2,6)

imshow(image\_mod\_thresh\_5x5);

title("After 5x5 Smooth Filter and Thresholding");



Explanation: This code applies a smoothing filter in both 5x5 and 3x3 variants to a outer space picture with the goal of eliminating small celestial objects while keeping the large ones intact. Then a threshold filter is applie to the image to clearly define the differences between large celestial bodies and small bodies along with noise.

3. Order-Statistics Filters

**3x3 Median, Max, and Min Filters to image Fig3-3.tif**

clear

fprintf("Question 3, Part a,b:")

file = 'Fig3-3.tif';

image = imread(file);

image\_double = im2double(image);

image\_mod\_median = zeros(size(image\_double));

image\_mod\_max = zeros(size(image\_double));

image\_mod\_min = zeros(size(image\_double));

% Median, Max, and Min Filters

for i = 1+1 : size(image\_double,1)-1

for j = 1+1 : size(image\_double,2)-1

frame\_3x3 = [[image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1)]

[image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1)]

[image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1)]];

median\_level = median(median(frame\_3x3));

max\_level = max(max(frame\_3x3));

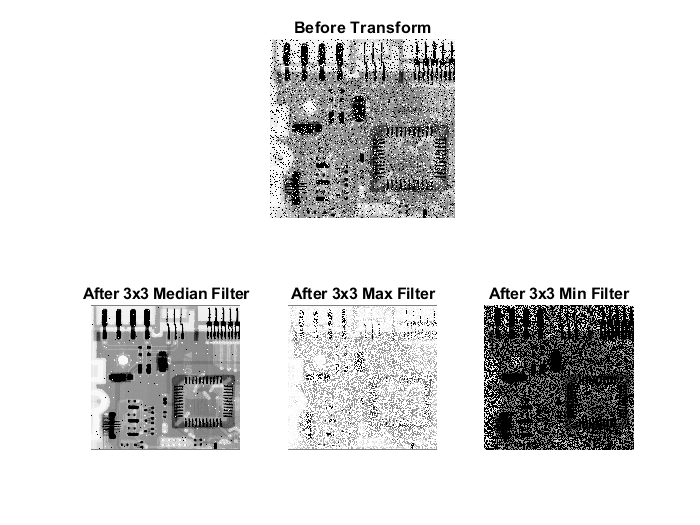
min\_level = min(min(frame\_3x3));

image\_mod\_median(i,j) = median\_level;

image\_mod\_max(i,j) = max\_level;

image\_mod\_min(i,j) = min\_level;

end

end

figure();

subplot(2,3,[1 2 3])

imshow(image);

title("Before Transform");

subplot(2,3,4)

imshow(image\_mod\_median);

title("After 3x3 Median Filter");

subplot(2,3,5)

imshow(image\_mod\_max);

title("After 3x3 Max Filter");

subplot(2,3,6)

imshow(image\_mod\_min);

title("After 3x3 Min Filter");

Explanation: This code applies median, max, and min filters in the 3x3 mask variant, where for every pixel, it gets the median, max, or min pixel intensity of it and its surrounding 8 pixels and that is the new value for the pixel. The median filter is the clear winner in this case as it is the clearest without modification of the original image by arithmetic means, due to the fact that if the majority of the pixels in the mask are pixels related to the original image, then regardless of any noise, that original image's intensity will remain. The max filter highlights the "pepper" noise, blacker intensity noise, and can be used to subtract from the main image to delete some of the noise. The min filter on the other hand highlights the "salt" noise, whiter intensity noise, and again can be used in conjunction with the original image to eliminate some noise.

**Apply 5x5 media, max, and min filters and see changes**

clear

fprintf("Question 3, Part c:")

file = 'Fig3-3.tif';

image = imread(file);

image\_double = im2double(image);

image\_mod\_median = zeros(size(image\_double));

image\_mod\_max = zeros(size(image\_double));

image\_mod\_min = zeros(size(image\_double));

% Median, Max, and Min Filters

for i = 1+2 : size(image\_double,1)-2

for j = 1+2 : size(image\_double,2)-2

frame\_5x5 = [[image\_double(i-2,j-2) image\_double(i-2,j-1) image\_double(i-2,j+0) image\_double(i-2,j+1) image\_double(i-2,j+2)]

[image\_double(i-1,j-2) image\_double(i-1,j-1) image\_double(i-1,j+0) image\_double(i-1,j+1) image\_double(i-1,j+2)]

[image\_double(i+0,j-2) image\_double(i+0,j-1) image\_double(i+0,j+0) image\_double(i+0,j+1) image\_double(i+0,j+2)]

[image\_double(i+1,j-2) image\_double(i+1,j-1) image\_double(i+1,j+0) image\_double(i+1,j+1) image\_double(i+1,j+2)]

[image\_double(i+2,j-2) image\_double(i+2,j-1) image\_double(i+2,j+0) image\_double(i+2,j+1) image\_double(i+2,j+2)]];

median\_level = median(median(frame\_5x5));

max\_level = max(max(frame\_5x5));

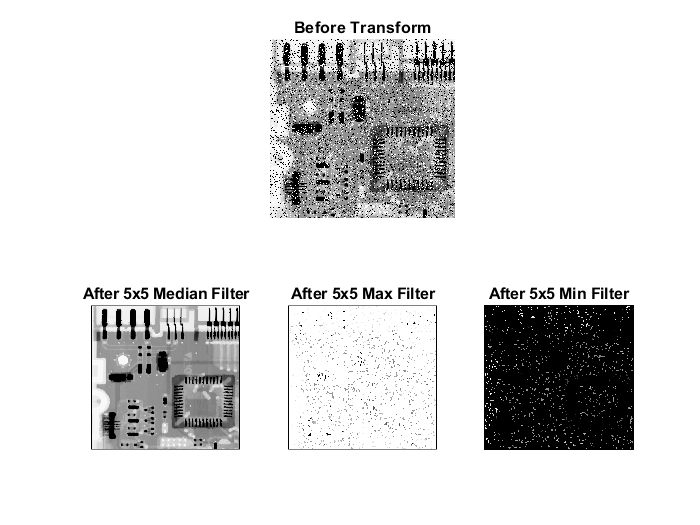
min\_level = min(min(frame\_5x5));

image\_mod\_median(i,j) = median\_level;

image\_mod\_max(i,j) = max\_level;

image\_mod\_min(i,j) = min\_level;

end

end

figure();

subplot(2,3,[1 2 3])

imshow(image);

title("Before Transform");

subplot(2,3,4)

imshow(image\_mod\_median);

title("After 5x5 Median Filter");

subplot(2,3,5)

imshow(image\_mod\_max);

title("After 5x5 Max Filter");

subplot(2,3,6)

imshow(image\_mod\_min);

title("After 5x5 Min Filter");

Explanation: This code applies the same statistical filters as shown previously, but with a mask size of 5x5, which in effect gives the new pixel more data to compare, which depending on the image is good or bad. In this case it is good, especially in the case of the median filter which results in almost no noise left in the image, but finer details of certain traces are lost.

**APPENDIX**

function hist = histogram(image)

% Histogram generates an array containing the occurrences at each intensity

% value of a given image input; image must be 8 bit with integer intensities

% from 0-256

hist = zeros(256,1);

for i = 1:size(image,1)

for j = 1:size(image,2)

r = image(i,j);

hist(r+1) = hist(r+1) + 1;

end

end

end

function sum = summation(array, from, to)

%Summation calculates the summation from 'from' to 'to' in array 'array'

sum = 0;

for i = from:to+1

sum = sum + array(i);

end

end