Project 1: Point and Neighborhood Processing

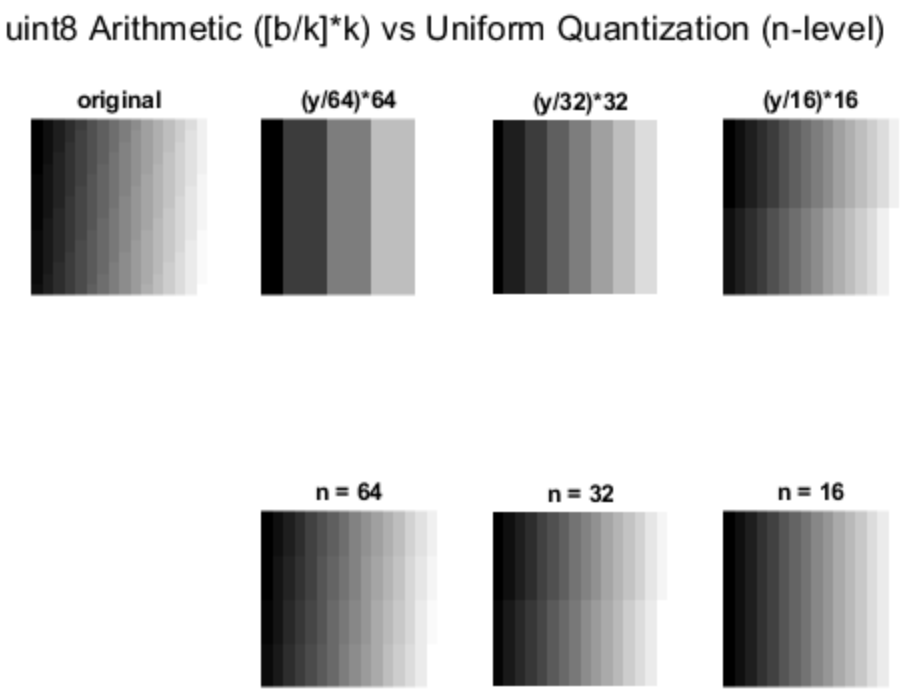
Table of Contents

Name: Socheath Sok ID: 014470701 Class: EE 483, T/Th 2:00 - 3:15 pm

Ch 4 Problem 2 & 3

Create an uint8 image from a 0 to 255 vector. Apply y = (b/k)*k on an image where k = [64 32 16]. Why are the results not the same as the original?

```
clc;clear;close all;

x = 0:255;                          % uint vector from 0 to 255
r = reshape(x, [16 16]);            % reshape x to 16x16
y = uint8(r);                       % convert r to uint8

figure;
subplot(2,4,1);imshow(y);title('original');

% uint8 arithmetic
subplot(2,4,2);imshow((y/64)*64);title('(y/64)*64');    % k = [64 32
 16]; equation (im/k)*k
subplot(2,4,3);imshow((y/32)*32);title('(y/32)*32');
subplot(2,4,4);imshow((y/16)*16);title('(y/16)*16');

% Uniformly Quantized Images
subplot(2,4,6);imshow(ind2gray(grayslice(y,64), gray(64)));title('n =
 64');
subplot(2,4,7);imshow(ind2gray(grayslice(y,32), gray(32)));title('n =
 32');
subplot(2,4,8);imshow(ind2gray(grayslice(y,16), gray(16)));title('n =
 16');

sgtitle('uint8 Arithmetic ([b/k]*k) vs Uniform Quantization (n-
level)')
```



From the subplot, it is clear that each column in the image represents a grayscale value and applying uint8 arithmetic using differnt values of k created output images with different number of grayscales from the original. This is because, unlike the uniform quantization process where the grayscale range (0 to 255) is divided into equal portions with each made up of 64 pixel intensities and mapped to an integer value (0 to n-1), the number of grayscales for uint8 arithmetic is calculated differently. With the command (b/k)*k, the maximum graycale output is determine by the equation 255/v+1. Additionally, the result is rounded after performing uint8 division to remove all fractional part, so it makes sense that there are less grayscales in the output images. As a result, applying uniform quatization procedure to an image will produce a similar image because each value of n is made up of 64 output grayscales. With regard to the uint8 arithmetic, the images generated depends on the value of k and for k = [64 32 16], the output grayscales are [5 9 17].

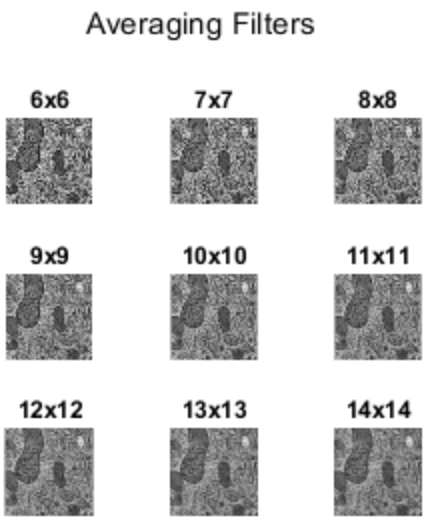
Ch 5 Problem 6,7,8

Apply averaging filters on a detailed image to determine the smallest sized filter which the fined detail cannot be seen. Apply Gaussian flters with given parameters until fine details disappear. Compare the results

Testing Averaging Filters

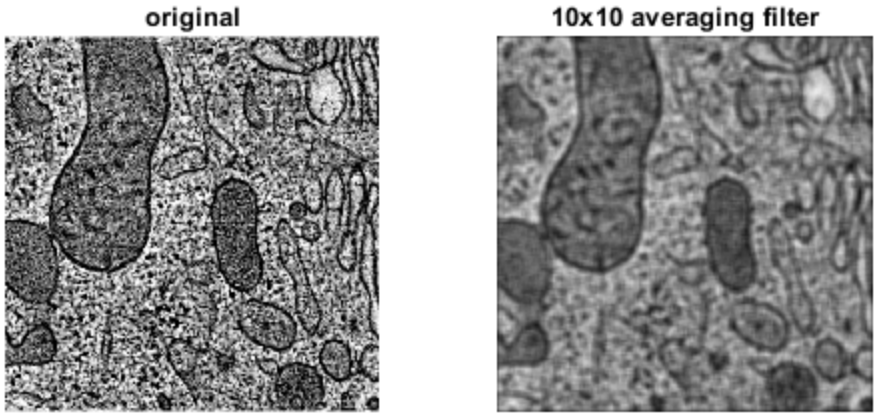
```
clc;clear;close all;
imshow('cell1.jpg'); impixelinfo;
x = imread('cell1.jpg');

% Applying averaging filers from 6x6 to 14x14
I = (6:14);
for k = 1: length(I)
```

```
    z = I(k);
    f1 = fspecial('average',[z,z]);
    figure(1);
    subplot(3,3,k)
    imshow(imfilter(x,f1));
    title(strcat(num2str(I(k)),'x',strcat(num2str(I(k)))));
    sgtitle('Averaging Filters')
end
```



side-by-side comparison

```
figure(2);
x = imread('cell1.jpg');
subplot(1,2,1);imshow(x); title('original');
subplot(1,2,2); imshow(imfilter(x,fspecial('average',[10,10])));
title('10x10 averaging filter');
```

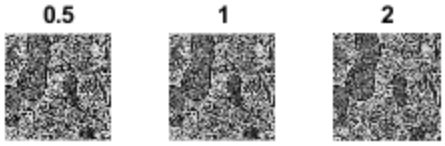


Testing all Gaussian filters parameters

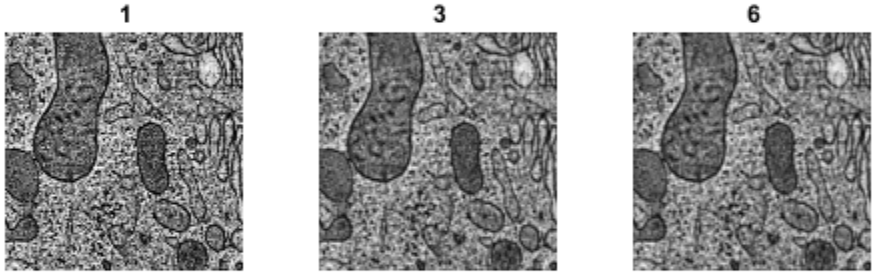
```
clc;clear;close all;
imshow('cell1.jpg'); impixelinfo;
x = imread('cell1.jpg');

f3 = [0.5 1 2]; f7 = [1 3 6];
f11 = [1 4 8]; f21 =  [1 5 10];
for k = 1:length(f3)
    a = f3(k); b = f7(k); c = f11(k); d = f21(k);
    f_3 = fspecial('gaussian',[3,3],a);
    f_7 = fspecial('gaussian',[7,7],b);
    f_11 = fspecial('gaussian',[11,11],c);
    f_21 = fspecial('gaussian',[21,21],d);
    figure(1); sgtitle('3x3 Gaussian Filters')
    subplot(1,3,k);
 imshow(imfilter(x,f_3));title(strcat(num2str(a)));
    figure(2);sgtitle('7x7 Gaussian Filters')
    subplot(1,3,k);
 imshow(imfilter(x,f_7));title(strcat(num2str(b)));
    figure(3);sgtitle('11x11 Gaussian Filters')
    subplot(1,3,k);
 imshow(imfilter(x,f_7));title(strcat(num2str(c)));
    figure(4);sgtitle('21x21 Gaussian Filters')
    subplot(1,3,k);
 imshow(imfilter(x,f_7));title(strcat(num2str(d)));
```

```
end
```

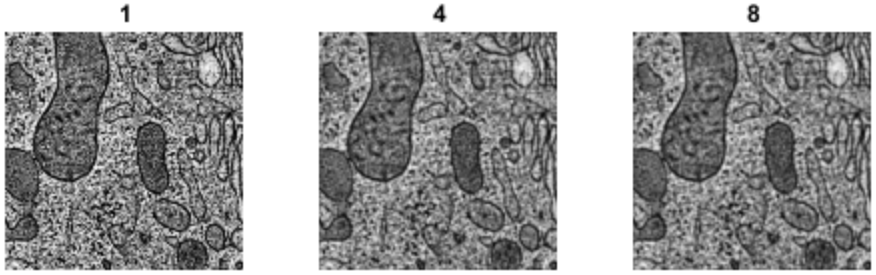
3x3 Gaussian Filters



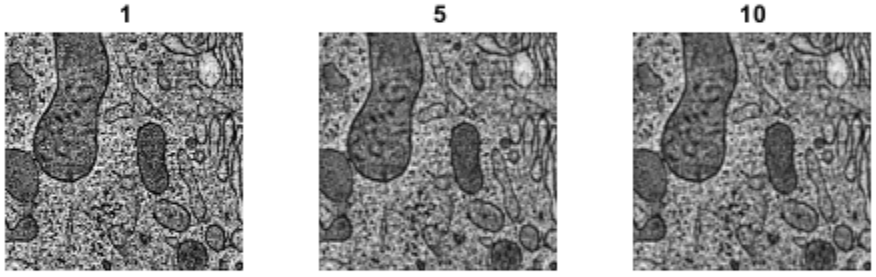
7x7 Gaussian Filters



11x11 Gaussian Filters

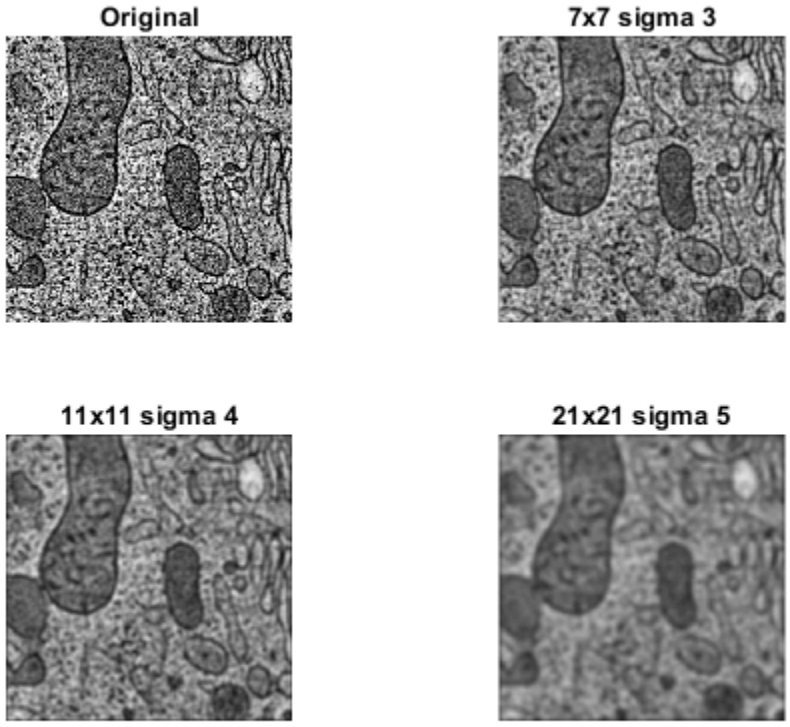


21x21 Gaussian Filters



Side-by-side comparisons

```
x = imread('cell1.jpg');
a = (fspecial('gaussian',[7,7],3));
b = (fspecial('gaussian',[11,11],4));
c = (fspecial('gaussian',[21,21],5));
figure;
subplot(2,2,1); imshow(x); title('Original')
subplot(2,2,2); imshow(imfilter(x,a)); title('7x7 sigma 3')
subplot(2,2,3); imshow(imfilter(x,b)); title('11x11 sigma 4')
subplot(2,2,4); imshow(imfilter(x,c)); title('21x21 sigma 5')
```



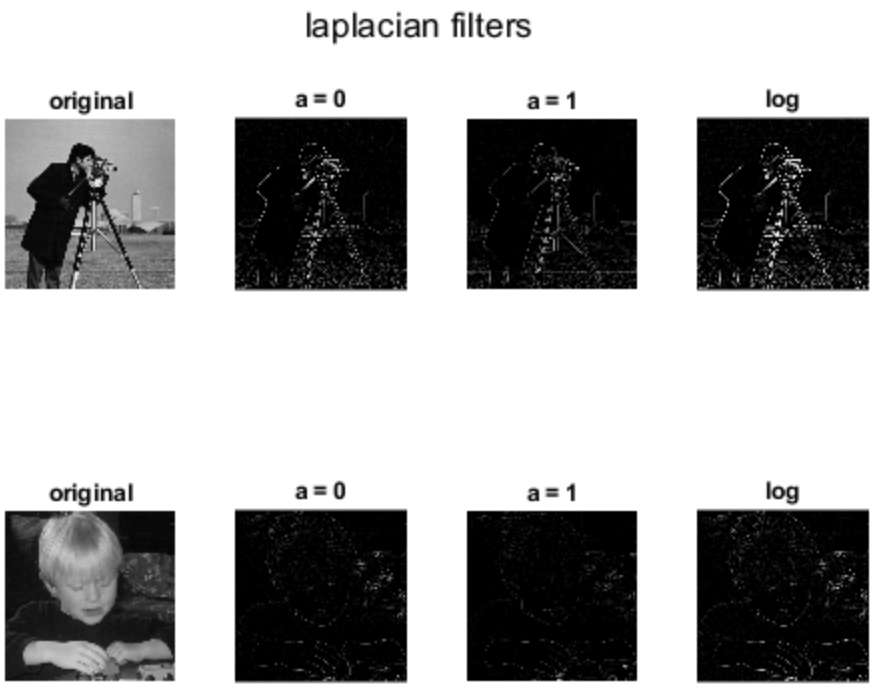
For the averaging fitler, fine details cannot be seen with 10x10 averaging filter. With the given 3x3 Gaussian filter parameters, fine details can be seen in all three images. As for 7x7, 11x11, and 21x21 fitlers, fine details cannot be seen starting at sigma 3, 4 and 5, respectively. For easy viewing, side-by-side comparisons between the original and smallest desired filter for both types are set up under each test. In terms of observation, with the fine details image selected, both averaging and gaussian filters are used to reduce the amount of noise and details. As the filter size for averaging filter increases, the image becomes more and more blurry. However, this blurry effect is a lot less noticeable with the guassian filter because instead of taking the mean of the neighbors around each pixel, this filter blur the image using the guassian function. Futhermore, the area of blur after the gaussian filter seem to be proportional to the standard deviation.

Ch 5 Problem 10

Apply different Laplacian filters to two images to determine which produce the best edge image.

```
clc;clear;close all;
```

```
c = imread('cameraman.png');
r = imread('engineer.png');
f0 = fspecial('laplacian',0);f1 = fspecial('laplacian',1);          %
 3x3 laplacian filter wit h different alpha values
figure(1); sgtitle('laplacian filters');
subplot(2,4,1);imshow(c); title('original');
subplot(2,4,2);imshow(imfilter(c,f0)); title('a = 0');
                  % alpha min
subplot(2,4,3);imshow(imfilter(c,f1)); title('a = 1');
                  % alpha max
subplot(2,4,4);imshow(imfilter(c,fspecial('log',[3,3])));
 title('log');                  % laplacian of gaussian
subplot(2,4,5);imshow(r); title('original');
subplot(2,4,6);imshow(imfilter(r,f0)); title('a = 0');
subplot(2,4,7);imshow(imfilter(r,f1)); title('a = 1');
subplot(2,4,8);imshow(imfilter(r,fspecial('log',[3,3])));
 title('log');
```



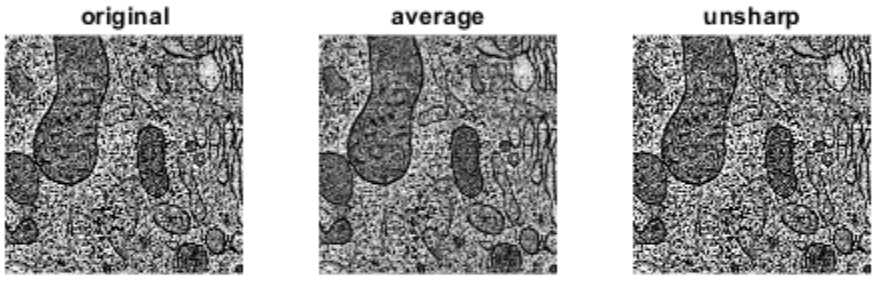
In the cameraman image, the edges can easily be picked up by the laplacian filter at both the minimum and maximum alpha values. However, when the same filters are applied to the engineer image, the edges are harder to pick up, and the results show that the alpha value does affect the quality of edge detection with a lower value (a = 0) performing better than a higher one (a = 1). Even though a laplacian filter can be used by itself to detect the edges, the quality of detection can often be improved with additional image processing techniques. In this two cases, gaussian filters can be used to smooth out the images first before applying the laplacian filters. Looking at the images side-by-side, the laplacian of gaussian (log) filter is clearly

better in detecting the edges. In the cases where values are outside the range [0 255], scaling operation in the form on a linear transformation is needed to force all values back into the displayable range.

Ch 5 Problem 19

Apply an unsharp mask filter after a 3x3 averaging filter. Can this be used to reverse the blurring effects?

```
clc;clear;close all;
x = imread('cell1.jpg');
f1 = fspecial('average',[3,3]);
f2 = fspecial('unsharp');                % default a = 0.2
y = imfilter(x,f1);
subplot(1,3,1); imshow(x);  title('original');
subplot(1,3,2); imshow(y);  title('average');
subplot(1,3,3); imshow(imfilter(y,f2));title('unsharp');
```



From the result, the sharpness of an image from the unsharp filter seem to be better than both the original and the one generated with the blurring effect. However, by applying a 3x3 averaging filter, some information such as noise and minor details is removed permanently to smooth out or blur the image. As such, with the 3x3 unsharp mask filter, the image along with the remaining information is simply sharpened to counteract the blurring filter. Moreover, the quality of sharpness depends on the alpha value with lower one being sharper, but regardless of the parameter, the lost details cannot not be recovered.

Published with MATLAB® R2019b