EE 4211 Computer Vision

Lecture 2B: Image enhancement (Spatial)

Semester B, 2021-2022

Spatial Domain Topics

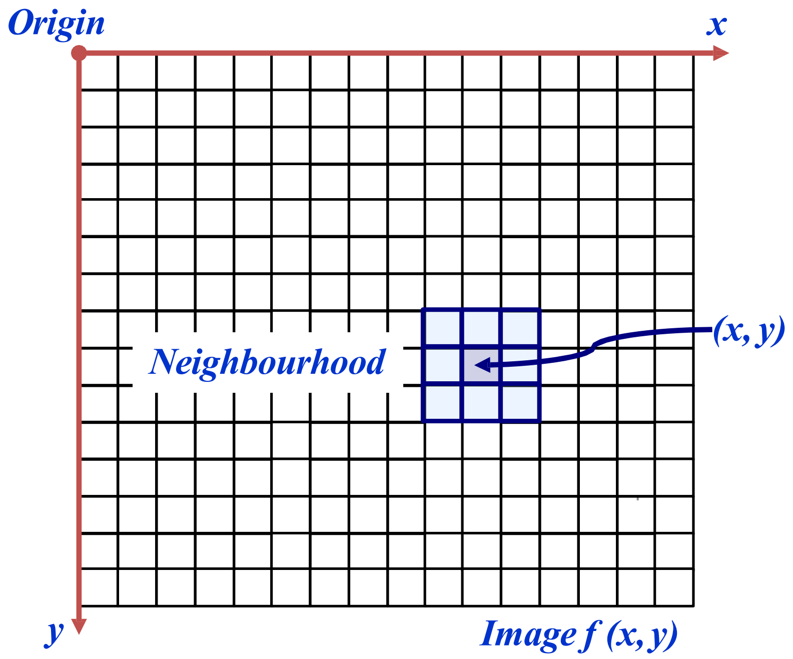
- Point processing – Gray values change without any knowledge of its surroundings (Part I)
  - Log, power-law, piecewise linear
  - Histogram Equalization
- Neighborhood processing (filtering) – Gray values change depending on the gray values in a small neighborhood of pixels around the given pixel (Part II)
  - Smoothing filters
  - Median filters
  - Sharpening

Spatial Filtering

- Basics of Spatial Filtering
- Smoothing Spatial Filters
  - Averaging filters, Order-Statistics filters
- Sharpening Spatial Filters
  - Laplacian filters, Sobel filter
- Combining Spatial Enhancement Methods

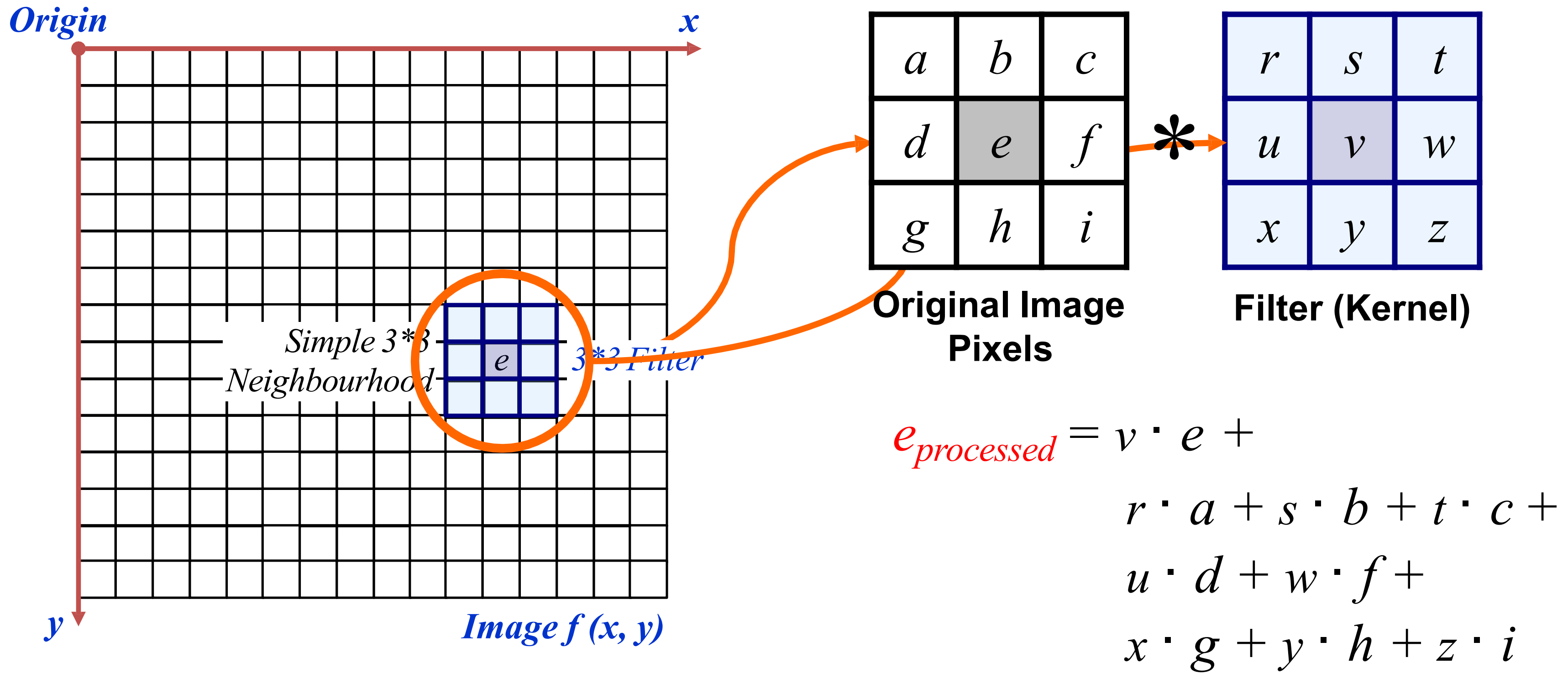
Linear Spatial Filtering

- g(x,y) = T[f(x,y)]
  - f(x,y): input image
  - g(x,y): output image
  - T: an operator on f defined over some neighborhood of (x,y)
- A spatial filter consists of
  - a neighborhood, and
  - a predefined operation



The Spatial Filtering Process

- Repeated for every pixel in the original image to generate the filtered image

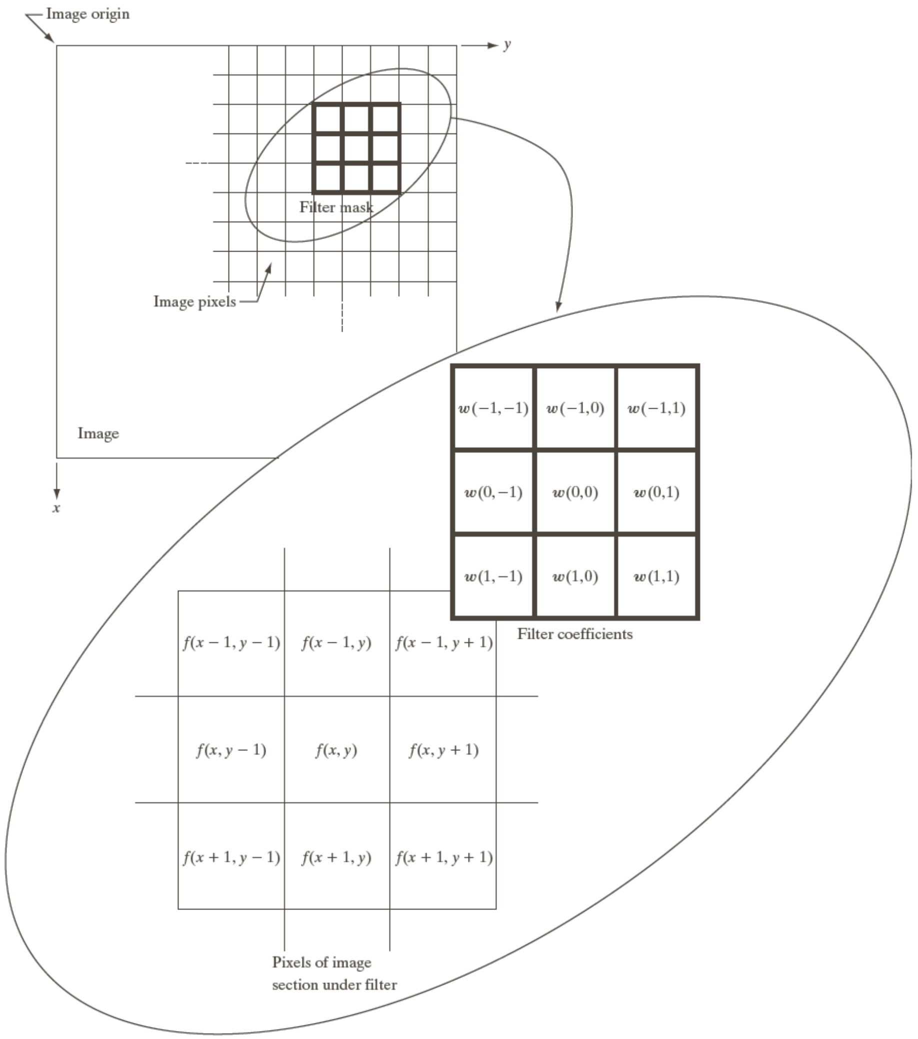


$$e_{processed} = v \cdot e + r \cdot a + s \cdot b + t \cdot c + u \cdot d + w \cdot f + x \cdot g + y \cdot h + z \cdot i$$

Spatial Filtering: Equation Form

- Filtering can be given in equation form by

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$



Spatial Filtering

- Basics of Spatial Filtering
- Smoothing Spatial Filters
  - Averaging filters, Order-Statistics filters
- Sharpening Spatial Filters
  - Laplacian filters, High-boost filters
- Combining Spatial Enhancement Methods

Smoothing Spatial Filters

- Smoothing filters are used for blurring and for noise reduction
- Blurring is used in preprocessing steps, such as
  - Removal of small details from an image prior to object extraction
  - Bridging of small gaps in lines or curves
- Noise Reduction can be accomplished by blurring with linear or non-linear filters

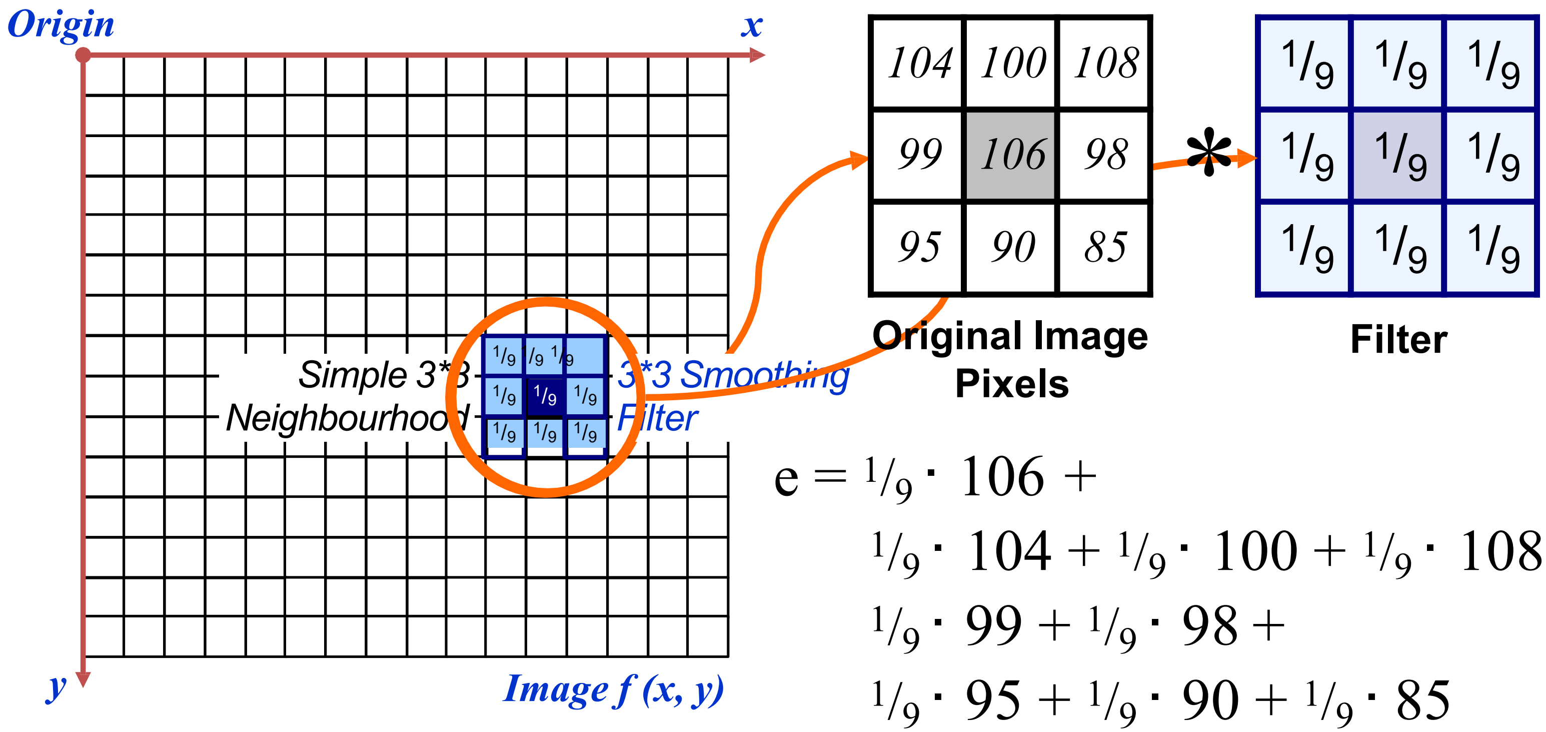
Spatial Smoothing Linear Filters

- Smoothing : One of the simplest spatial filtering operations
- Replace each pixel by the average of pixels in a square window surrounding this pixel
  - Especially useful in removing noise from images
  - Also useful for highlighting gross information

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

Simple Averaging Filter

Average Filtering Process



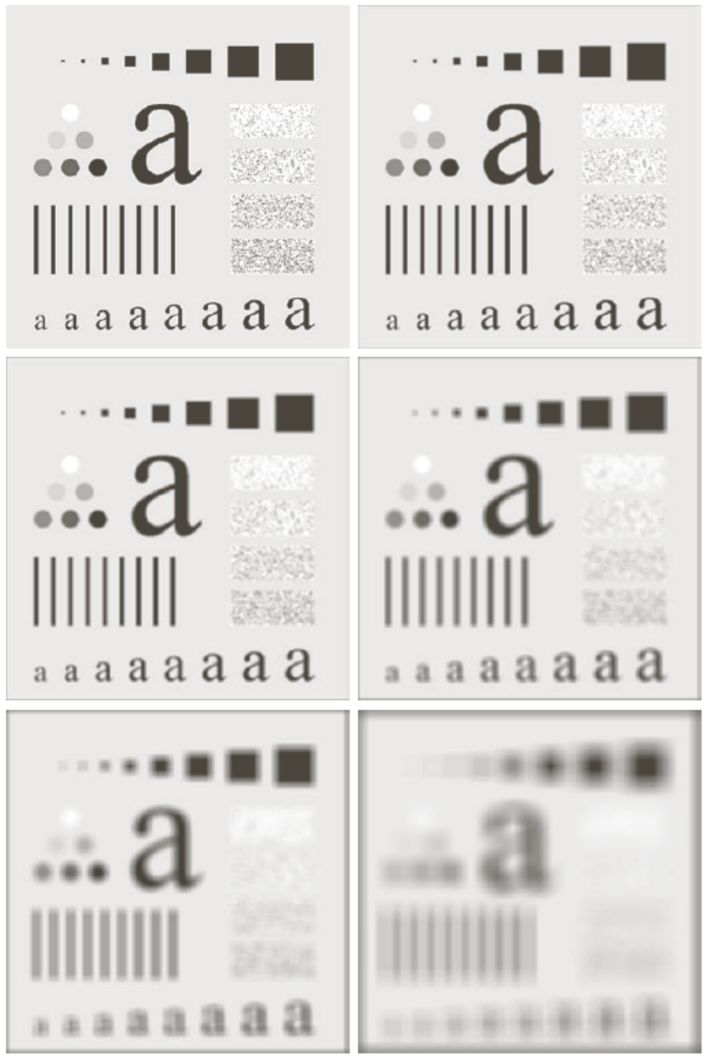
104	100	108
99	106	98
95	90	85

$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$
$^1/_9$	$^1/_9$	$^1/_9$

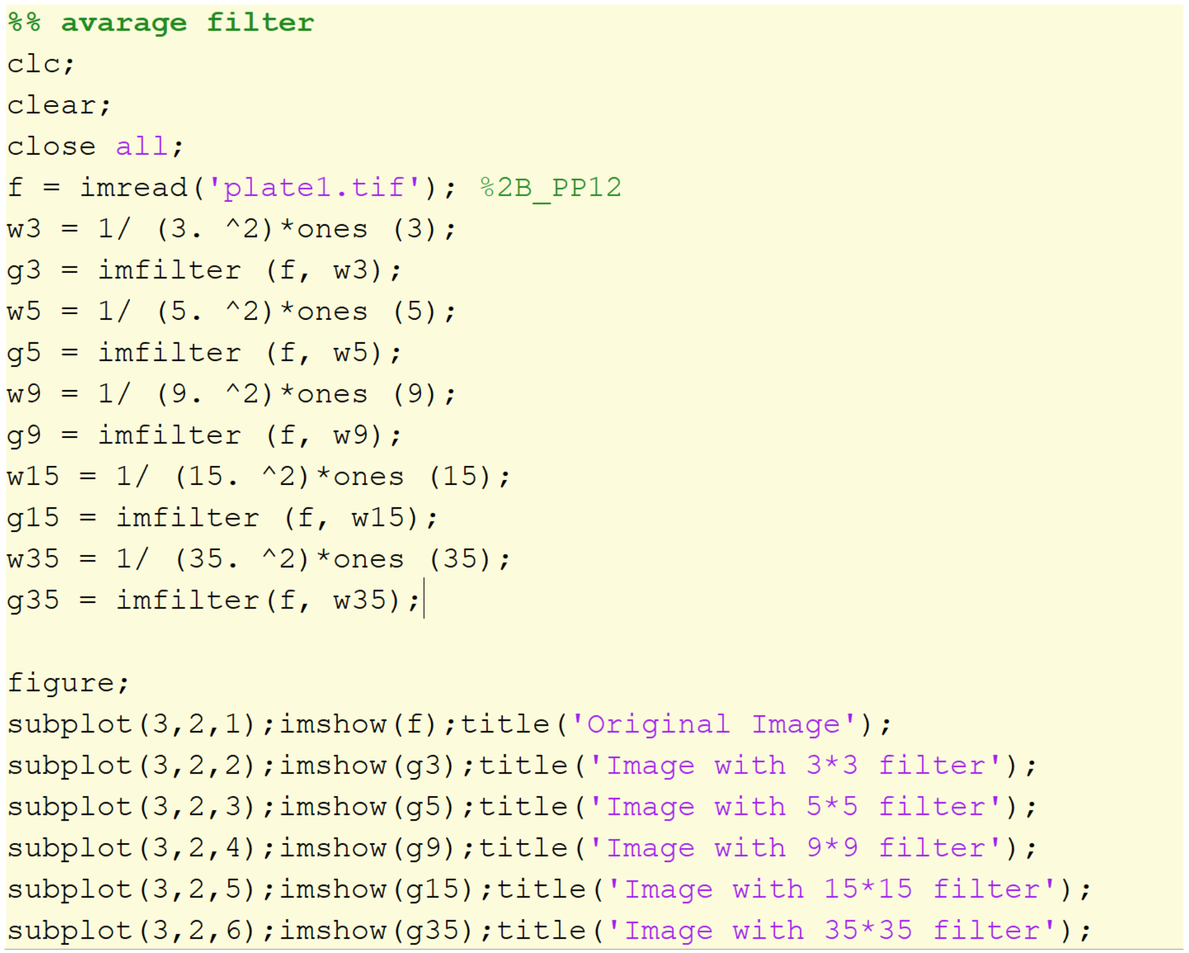
$$\begin{aligned} e = {} & {}^1/_9 \cdot 106 + \\ & {}^1/_9 \cdot 104 + {}^1/_9 \cdot 100 + {}^1/_9 \cdot 108 + \\ & {}^1/_9 \cdot 99 + {}^1/_9 \cdot 98 + \\ & {}^1/_9 \cdot 95 + {}^1/_9 \cdot 90 + {}^1/_9 \cdot 85 \end{aligned}$$

Examples

- Original image size 500x500 pixels
- Results of smoothing with averaging filter masks of size n=3, 5, 9, 15, 35, respectively



Examples



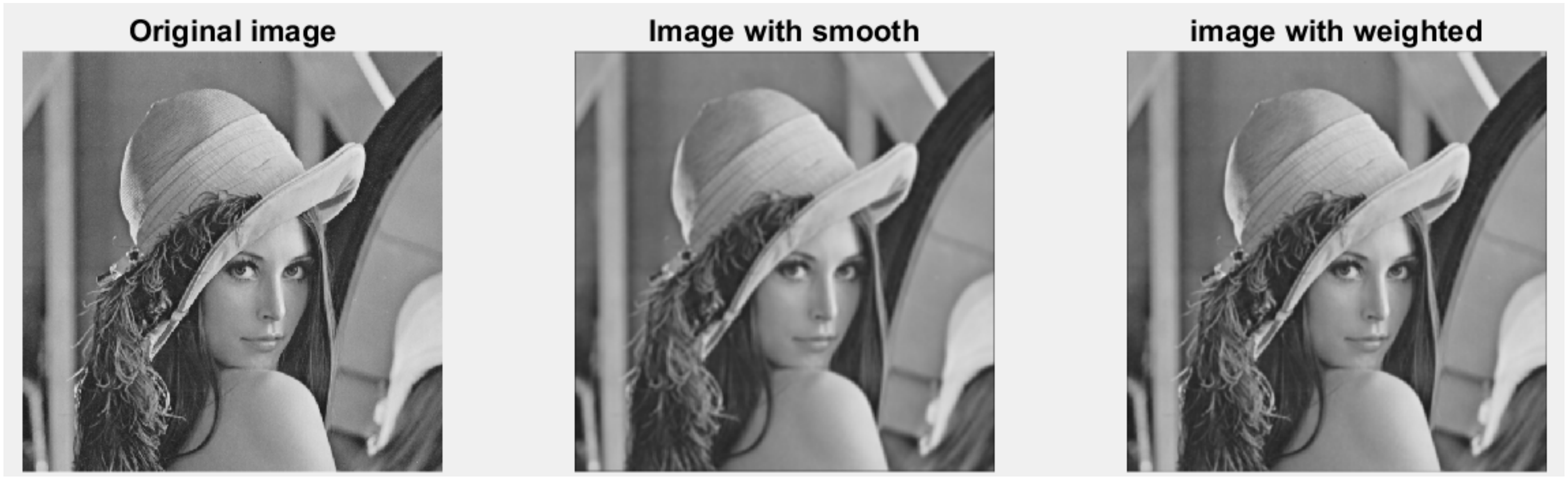
```matlab
%% avarage filter
clc;
clear;
close all;
f = imread('plate1.tif'); %2B_PP12
w3 = 1/ (3. ^2)*ones (3);
g3 = imfilter (f, w3);
w5 = 1/ (5. ^2)*ones (5);
g5 = imfilter (f, w5);
w9 = 1/ (9. ^2)*ones (9);
g9 = imfilter (f, w9);
w15 = 1/ (15. ^2)*ones (15);
g15 = imfilter (f, w15);
w35 = 1/ (35. ^2)*ones (35);
g35 = imfilter(f, w35);

figure;
subplot(3,2,1);imshow(f);title('Original Image');
subplot(3,2,2);imshow(g3);title('Image with 3*3 filter');
subplot(3,2,3);imshow(g5);title('Image with 5*5 filter');
subplot(3,2,4);imshow(g9);title('Image with 9*9 filter');
subplot(3,2,5);imshow(g15);title('Image with 15*15 filter');
subplot(3,2,6);imshow(g35);title('Image with 35*35 filter');
```

Weighted Smoothing Filters

- Instead of averaging all the pixel values in the window, this filter gives the closer-by pixels higher weighting, and far-away pixels lower weighting.
- Reduce value of coefficients as a function of increasing distance from the origin
- An attempt to reduce blurring in the smoothing process

$^1/_{16}$	$^2/_{16}$	$^1/_{16}$
$^2/_{16}$	$^4/_{16}$	$^2/_{16}$
$^1/_{16}$	$^2/_{16}$	$^1/_{16}$

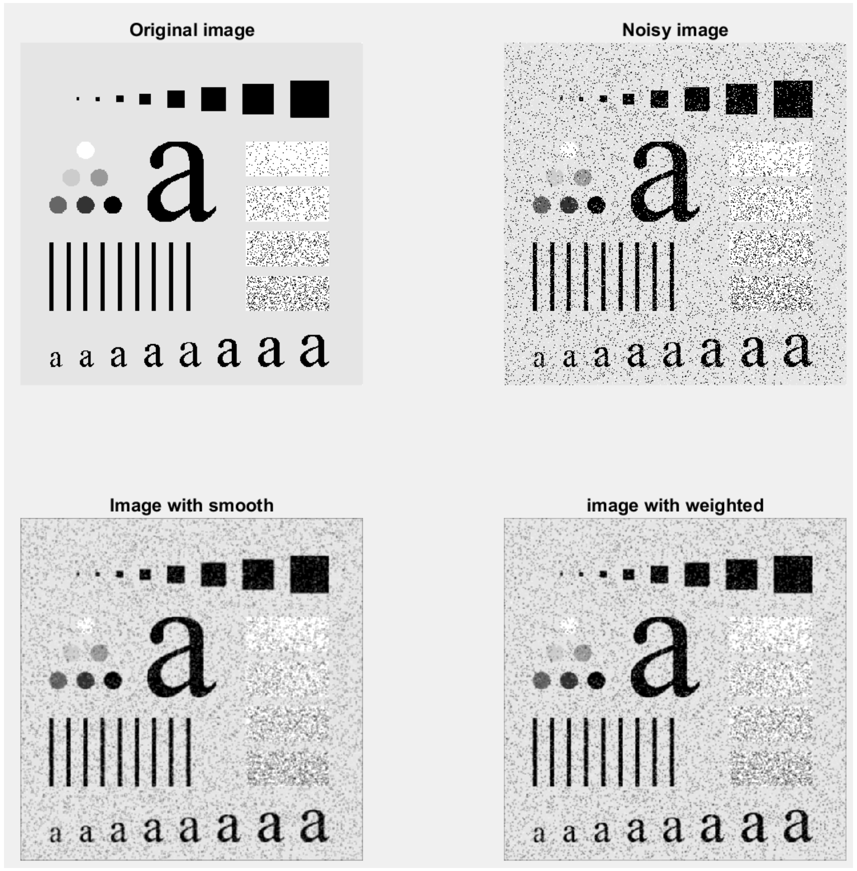
Examples to blur

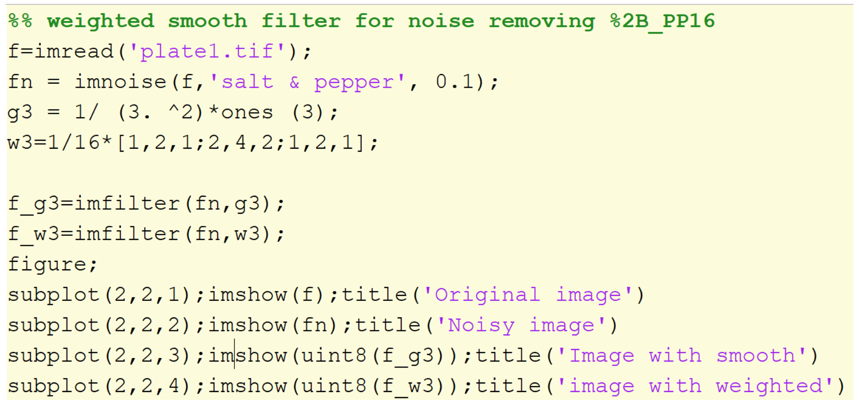


```
%% weighted smooth filter for smooth %2B_PP15
f=imread('lena.bmp');
g5 = 1/ (5. ^2)*ones (5);
w5=1/25*[0,0,1,0,0;0,2,2,2,0;1,2,5,2,1;0,2,2,2,0;0,0,1,0,0];

f_g5=imfilter(f,g5);
f_w5=imfilter(f,w5);
figure;
subplot(1,3,1);imshow(f);title('Original image')
subplot(1,3,2);imshow(uint8(f_g5));title('Image with smooth')
subplot(1,3,3);imshow(uint8(f_w5));title('image with weighted')
```

Examples to remove noise

- By smoothing the original image, we get rid of lots of the finer detail which leaves only the gross features for thresholding





```
%% weighted smooth filter for noise removing %2B_PP16
f=imread('plate1.tif');
fn = imnoise(f,'salt & pepper', 0.1);
g3 = 1/ (3. ^2)*ones (3);
w3=1/16*[1,2,1;2,4,2;1,2,1];

f_g3=imfilter(fn,g3);
f_w3=imfilter(fn,w3);
figure;
subplot(2,2,1);imshow(f);title('Original image')
subplot(2,2,2);imshow(fn);title('Noisy image')
subplot(2,2,3);imshow(uint8(f_g3));title('Image with smooth')
subplot(2,2,4);imshow(uint8(f_w3));title('image with weighted')
```

Order-Statistics Filters

- Non-linear filters
- Response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter
- Example:
  - median filter, max filter, min filter

Median Filters

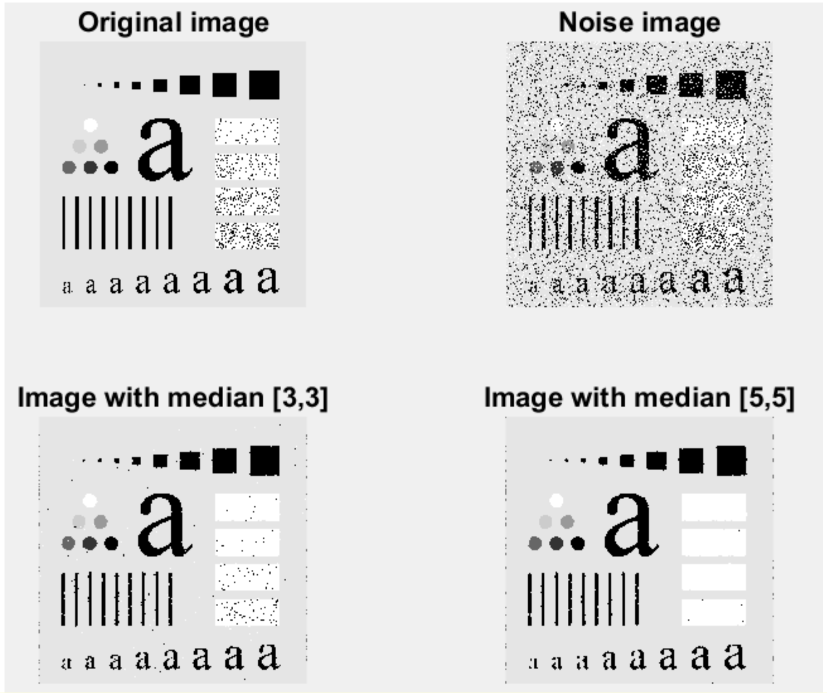
- Obtained by **sorting** all pixels in the analysis window in increasing or decreasing order of amplitudes and **picking the middle value** if the number of pixels is odd, or the average of the two values in middle if the number of pixels is even.

$$g(x,y) = median\{f(x-n, y-m), (n,m) \in N\}$$

- Popularly used for certain types of random noise (impulse noise, salt and pepper noise)
  - Excellent noise-reduction capabilities
  - Less blurring effect that linear smoothing filters of similar size

2D Median Filtering Example

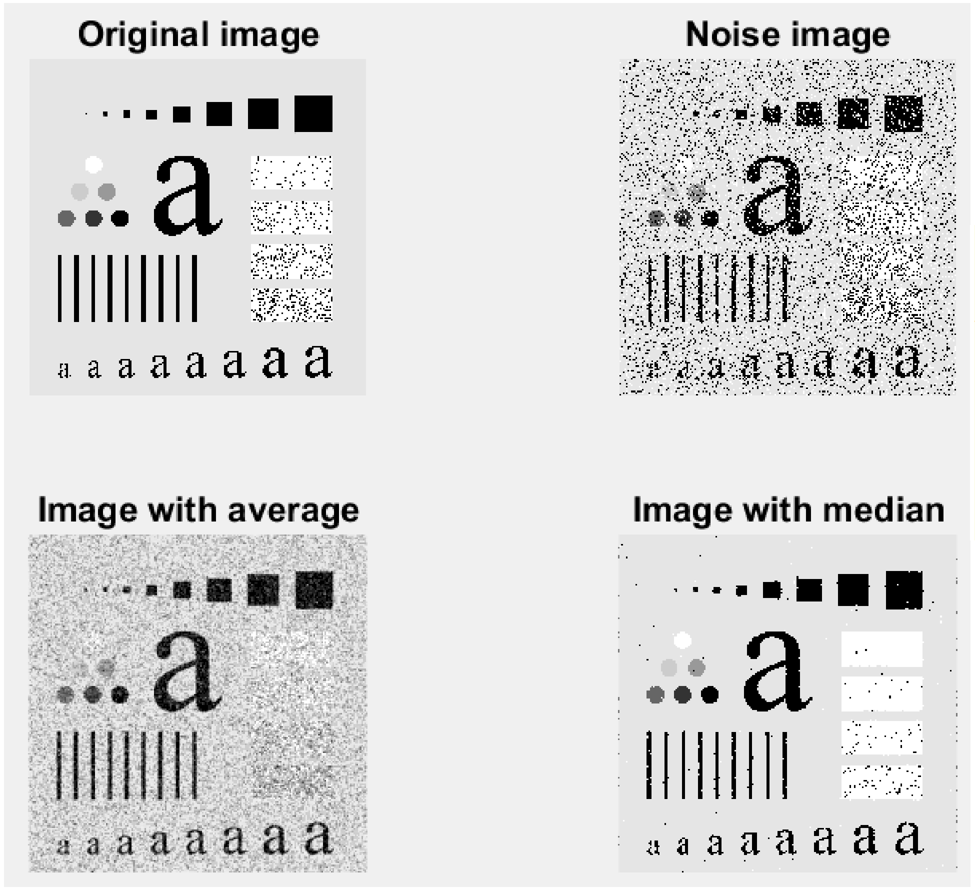
- Filtering is often used to remove noise from images

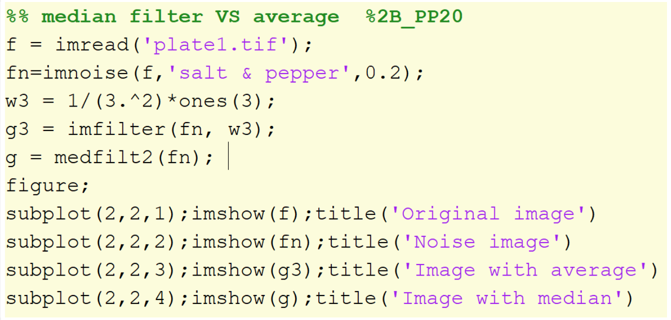


```
%% median filter %2B_PP19
f = imread('plate1.tif');
fn=imnoise(f,'salt & pepper',0.2);
g3 = medfilt2(fn,[3,3]);
g5 = medfilt2(fn,[5,5]);
figure
subplot(2,2,1);imshow(f);title('Original image')
subplot(2,2,2);imshow(fn);title('Noise image')
subplot(2,2,3);imshow(g3);title('Image with median [3,3]')
subplot(2,2,4);imshow(g5);title('Image with median [5,5]')
```

Averaging Filter vs. Median Filter Example

- Filtering is often used to remove noise from images
- Sometimes a median filter works better than an averaging filter





```
%% median filter VS average  %2B_PP20
f = imread('plate1.tif');
fn=imnoise(f,'salt & pepper',0.2);
w3 = 1/(3.^2)*ones(3);
g3 = imfilter(fn, w3);
g = medfilt2(fn);
figure;
subplot(2,2,1);imshow(f);title('Original image')
subplot(2,2,2);imshow(fn);title('Noise image')
subplot(2,2,3);imshow(g3);title('Image with average')
subplot(2,2,4);imshow(g);title('Image with median')
```

Spatial Filtering

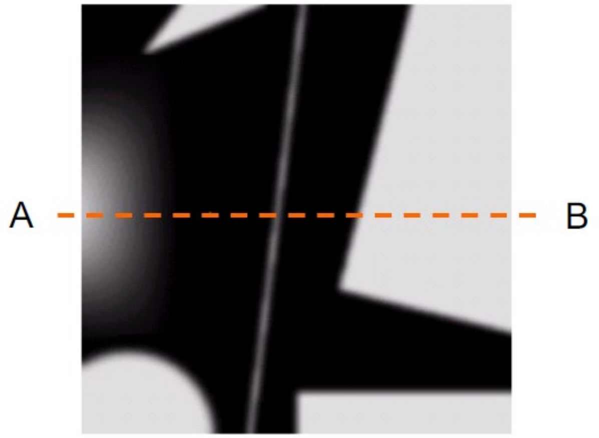
- Basics of Spatial Filtering
- Smoothing Spatial Filters
  - Averaging filters, Order-Statistics filters
- Sharpening Spatial Filters
  - Laplacian filters, Sobel filter
- Combining Spatial Enhancement Methods

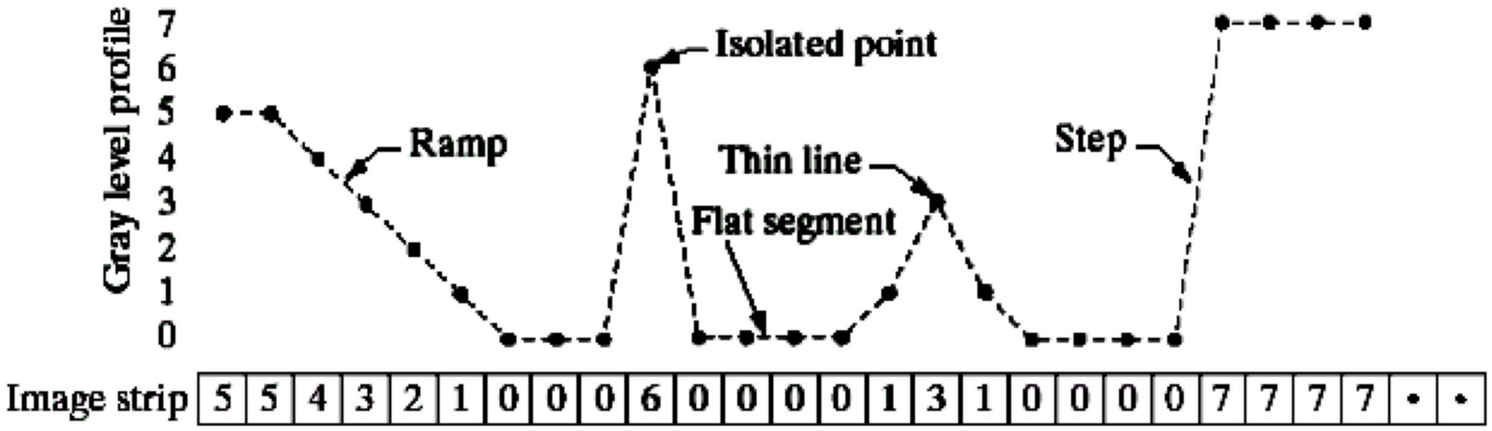
Sharpening Spatial Filters

- Smoothing filters remove fine detail
- Sharpening spatial filters seek to highlight fine detail
  - Remove blurring from images
  - Highlight edges
- Sharpening filters are based on spatial differentiation

Spatial Differentiation

- Differentiation measures the rate of change of a function
- Let's consider a simple 1 dimensional example



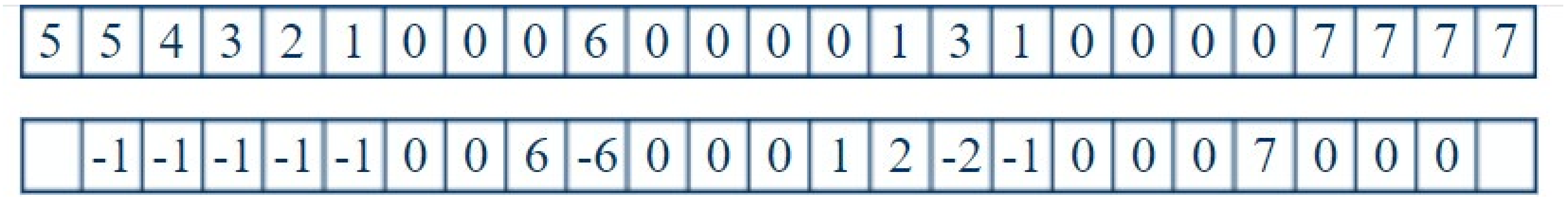


1st Derivative

- The formula for the 1st derivative of a function is as follows:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

- It's just the difference between subsequent values and measures the rate of change of the function



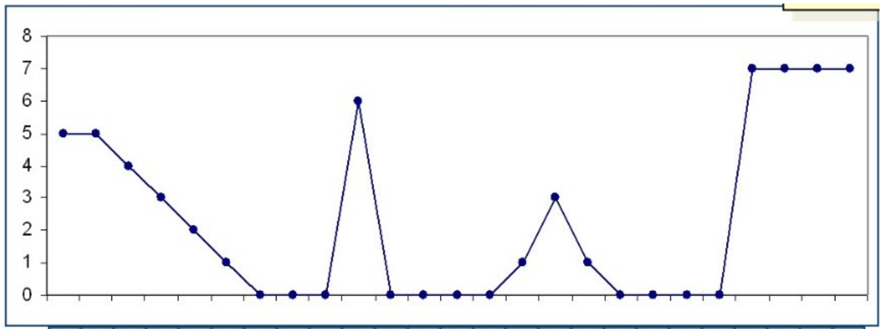
5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7
	-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0	

2nd Derivative

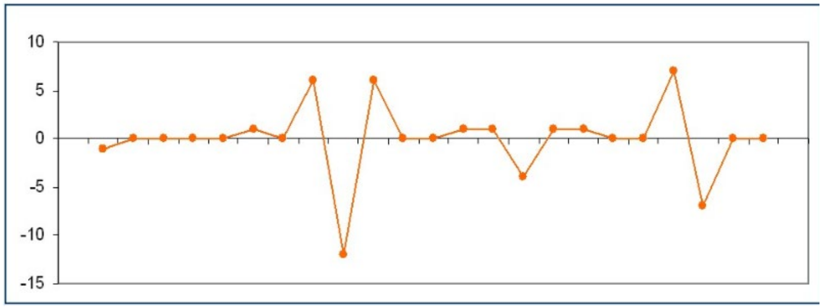
- The derivatives of a function is as follows:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

- Simply takes into account the values both before and after the current value



5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7
	-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0	
	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	



Using Second Derivatives for Image Enhancement

- The 2nd derivative is more useful for image enhancement than the 1st derivative
  - Stronger response to fine detail
  - Simple implementation
- The first sharpening filter we will look at is the Laplacian
  - Isotropic
  - One of the simplest sharpening filters

The Laplacian

- The Laplacian is defined as follows:

$$\nabla^2 f = \frac{\partial^2 f}{\partial^2 x} + \frac{\partial^2 f}{\partial^2 y}$$

- where the partial 2nd order derivative in the *x* direction is defined as follows:

$$\frac{\partial^2 f}{\partial^2 x} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

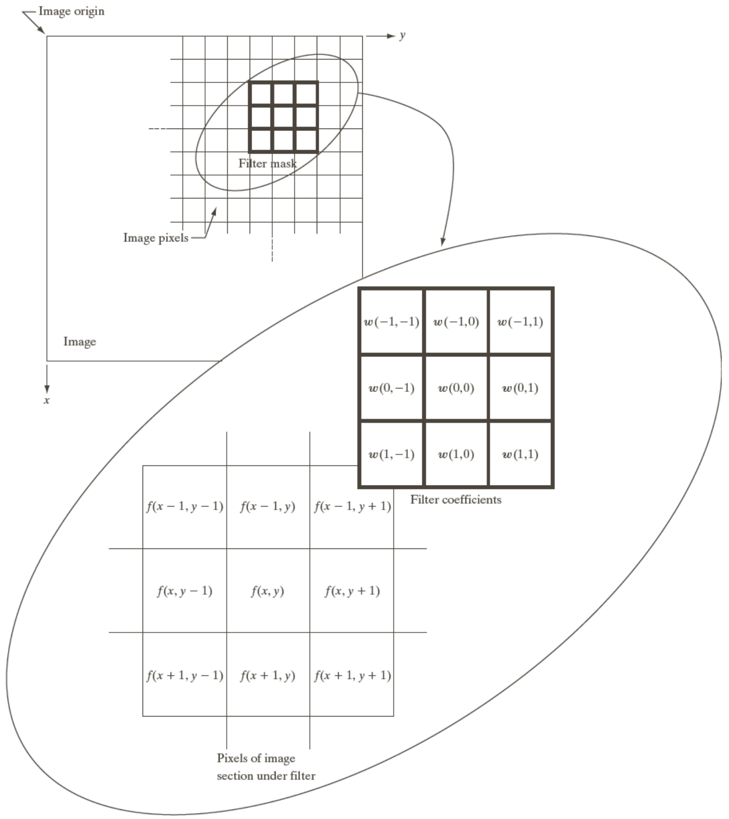
- and in the *y* direction as follows:

$$\frac{\partial^2 f}{\partial^2 y} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

The Laplacian Operator

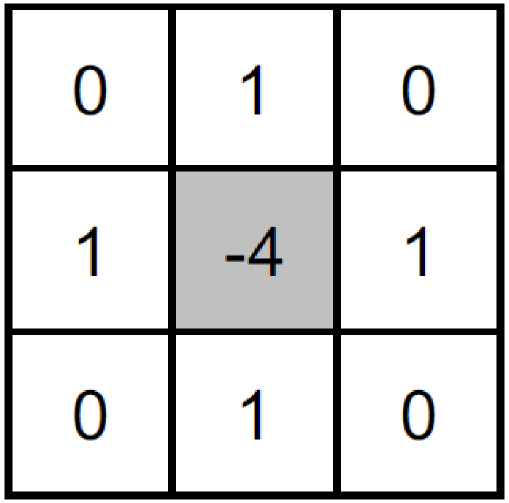
- So, the Laplacian can be given as follows:
  - $$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$
- We can easily build a filter

0	1	0
1	-4	1
0	1	0

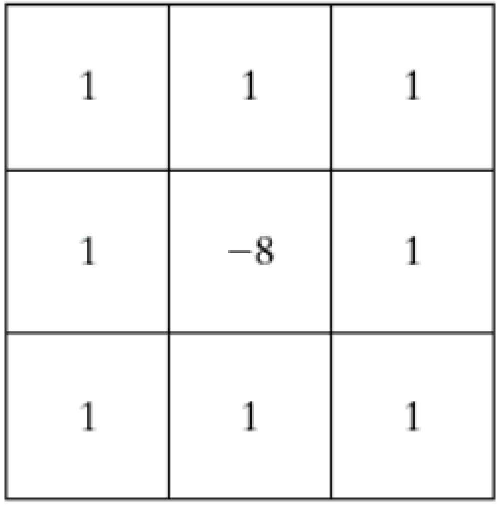


Laplacian Mask

- This Laplacian mask is implemented differently by incorporating the diagonal directions. The center value is now -8.



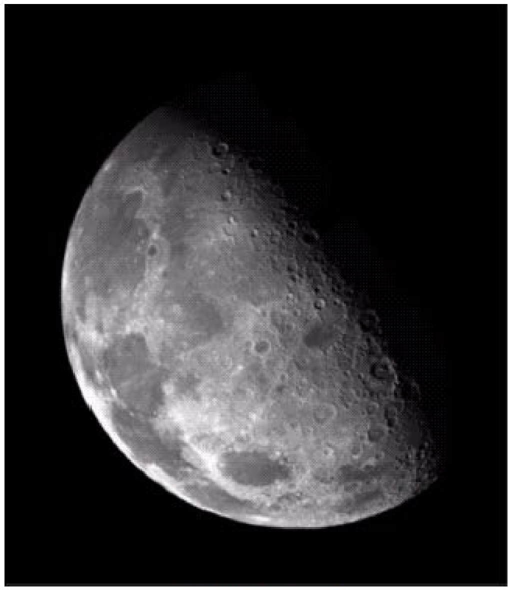
0	1	0
1	-4	1
0	1	0

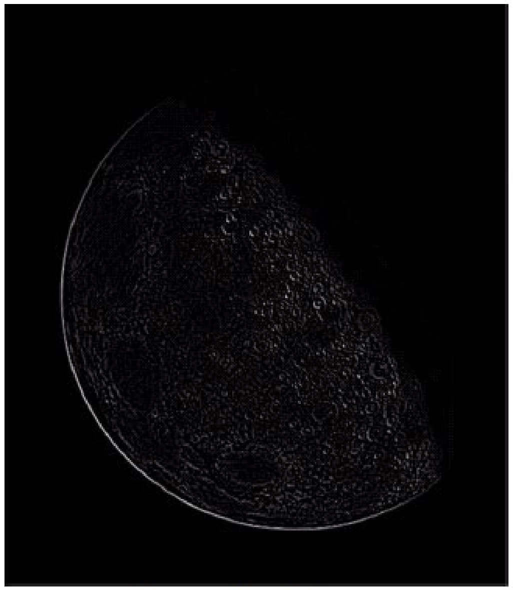


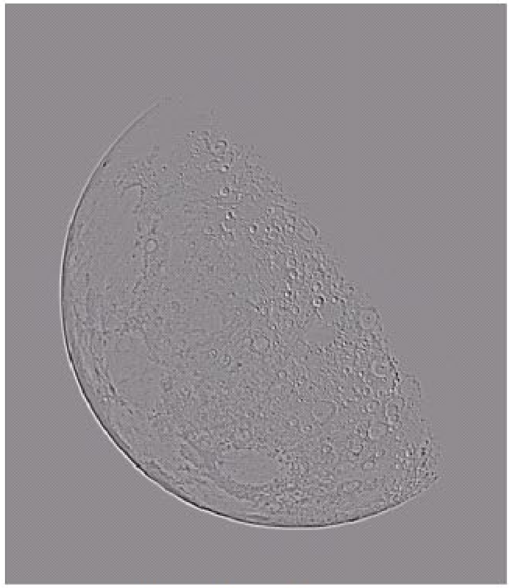
1	1	1
1	−8	1
1	1	1

The Laplacian Filter Example

- Applying the Laplacian filter to an image we get a new image that highlights edges and other discontinuities


Original Image

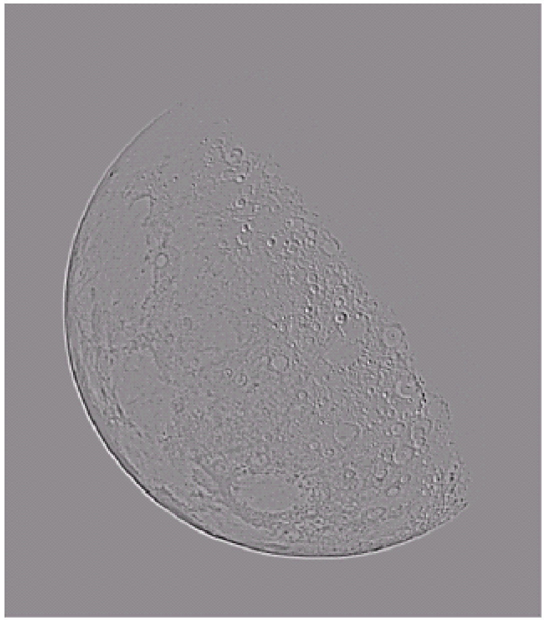

Laplacian Filtered Image


Laplacian Filtered Image Scaled for Display

But That Is Not Very Enhanced!

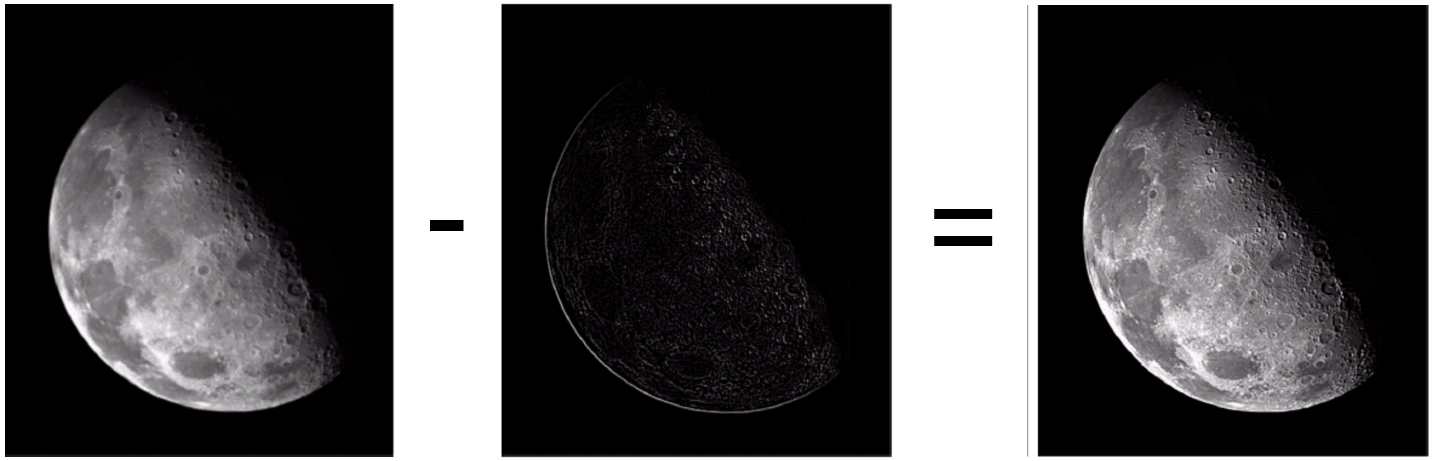
- The result of a Laplacian filtering is not an enhanced image
- We have to do more work in order to get our final image
- Subtract the Laplacian result from the original image to generate our final sharpened enhanced image

$$g(x, y) = f(x, y) - \nabla^2 f$$


Laplacian Filtered Image
Scaled for Display

Laplacian Image Enhancement

- In the final sharpened image, edges and fine detail are much more obvious



Original Image

Laplacian Filtered Image

Sharpened Image

Laplacian Image Enhancement

```
%% laplacian  %2B_PP32
f1 = imread('moon.tif');
w4 = fspecial('laplacian', 0);
g1 = imfilter(f1, w4);
f2 = im2double(f1);
g2 = imfilter(f2, w4);
g3 = imsubtract(f2,g2);
g4 = imadd(f2,g2);
figure;
subplot(2,2,1);imshow(f1);
subplot(2,2,2);imshow(g1, [ ]);
subplot(2,2,3);imshow(g2, [ ]);
subplot(2,2,4);imshow(g3);
figure;
subplot(2,1,1);imshow(g3);
subplot(2,1,2);imshow(g4);
```

Simplified Image Enhancement

- The entire enhancement can be combined into a single filtering operation

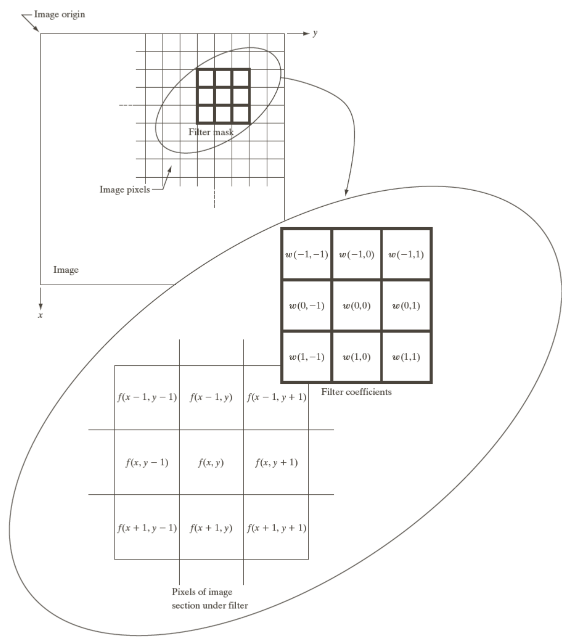
$$g(x,y) = f(x,y) - \nabla^2 f$$

$$= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1)$$

$$+ f(x,y-1) - 4f(x,y)$$

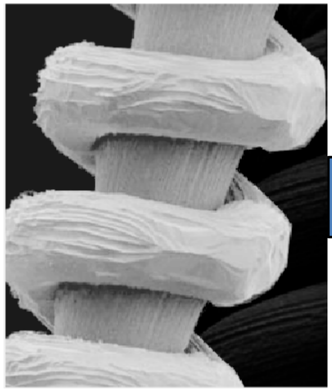
$$= 5f(x,y) - f(x+1,y) - f(x-1,y)$$

$$- f(x,y+1) - f(x,y-1)$$

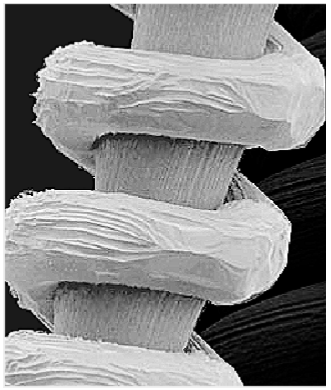


Simplified Image Enhancement

- This gives us a new filter which does the whole job for us in one step



0	-1	0
-1	5	-1
0	-1	0



Variants On The Simple Laplacian

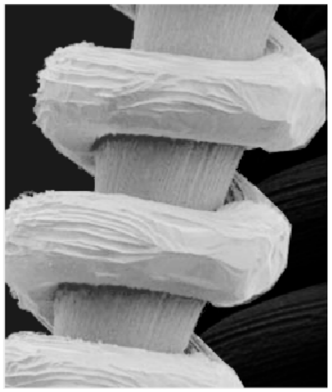
- There are lots of slightly different versions of the Laplacian that can be used:

0	1	0
1	-4	1
0	1	0

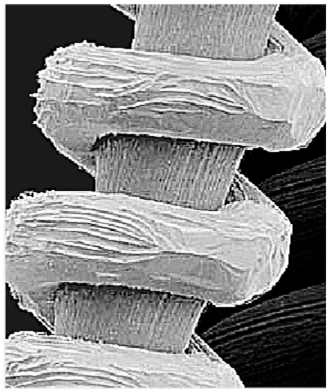
Simple Laplacian

1	1	1
1	-8	1
1	1	1

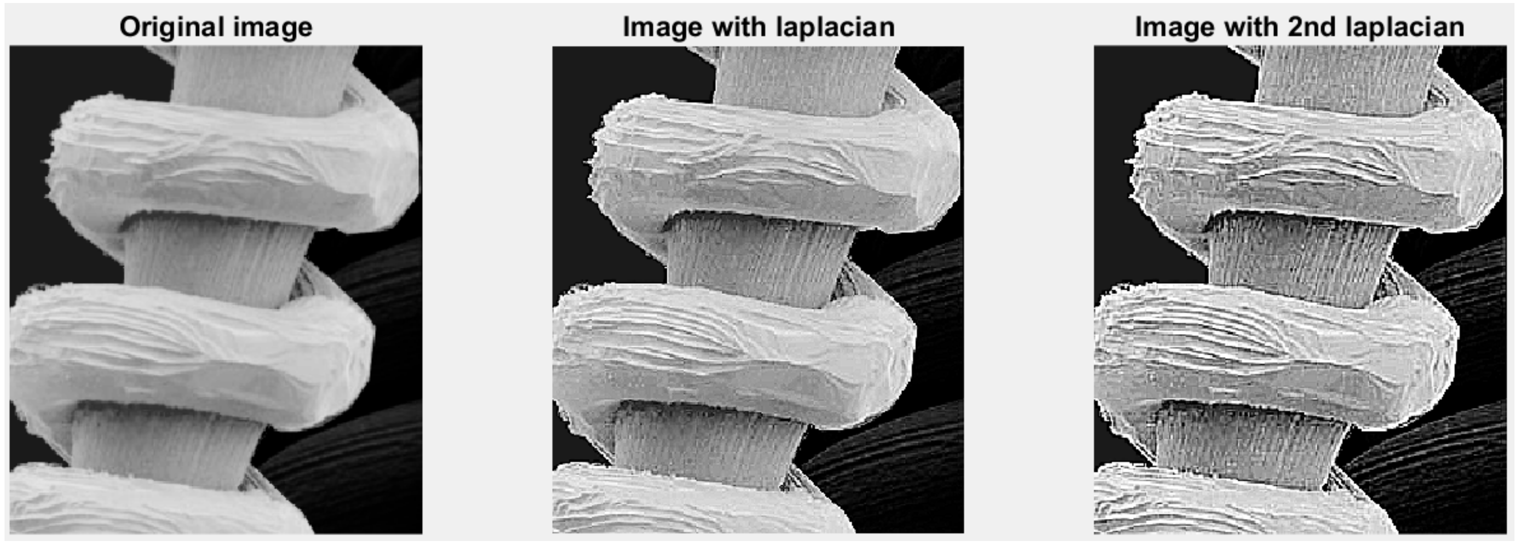
Variant of Laplacian



-1	-1	-1
-1	9	-1
-1	-1	-1



Comparison of Two Laplacians



```
%% simplifed laplacian  %2B_PP37
% Laplacian simplication

f1 = imread ('edge.tif');
w5 = [0 -1 0; -1 5 -1; 0 -1 0];
g1 = imfilter (f1, w5);
w9 = [-1 -1 -1; -1 9 -1; -1 -1 -1];
g2 = imfilter (f1, w9);
figure;
subplot(1,3,1);imshow(f1);title('Original image')
subplot(1,3,2);imshow(g1);title('Image with laplacian')
subplot(1,3,3);imshow(g2);title('Image with 2nd laplacian')
```

First Derivatives: Gradient Operator

- First derivatives are implemented using the magnitude of the gradient

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f = mag(\nabla \mathrm{f}) = [G_x^2 + G_y^2]^{1/2}$$

$$= \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

Approximation:

$$\nabla f \approx |G_x| + |G_y|$$

Gradient Mask

- On the basis of a first-order derivative of a 2-D function f(x,y), the simplest approximation of the gradient mask: 2x2

$$G_x = (z_8 - z_5) \quad \text{and} \quad G_y = (z_6 - z_5)$$

$$\nabla f = [G_x^2 + G_y^2]^{1/2} = [(z_8 - z_5)^2 + (z_6 - z_5)^2]^{1/2}$$

$$\nabla f \approx |z_8 - z_5| + |z_6 - z_5|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Gradient Mask

- Roberts 2x2 cross-gradient operators [1965]

$$G_x = (z_9 - z_5) \quad \text{and} \quad G_y = (z_8 - z_6)$$

$$\nabla f = [G_x^2 + G_y^2]^{1/2} = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2}$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Gradient Mask

- Sobel operators, 3x3

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

$$\nabla f \approx |G_x| + |G_y|$$

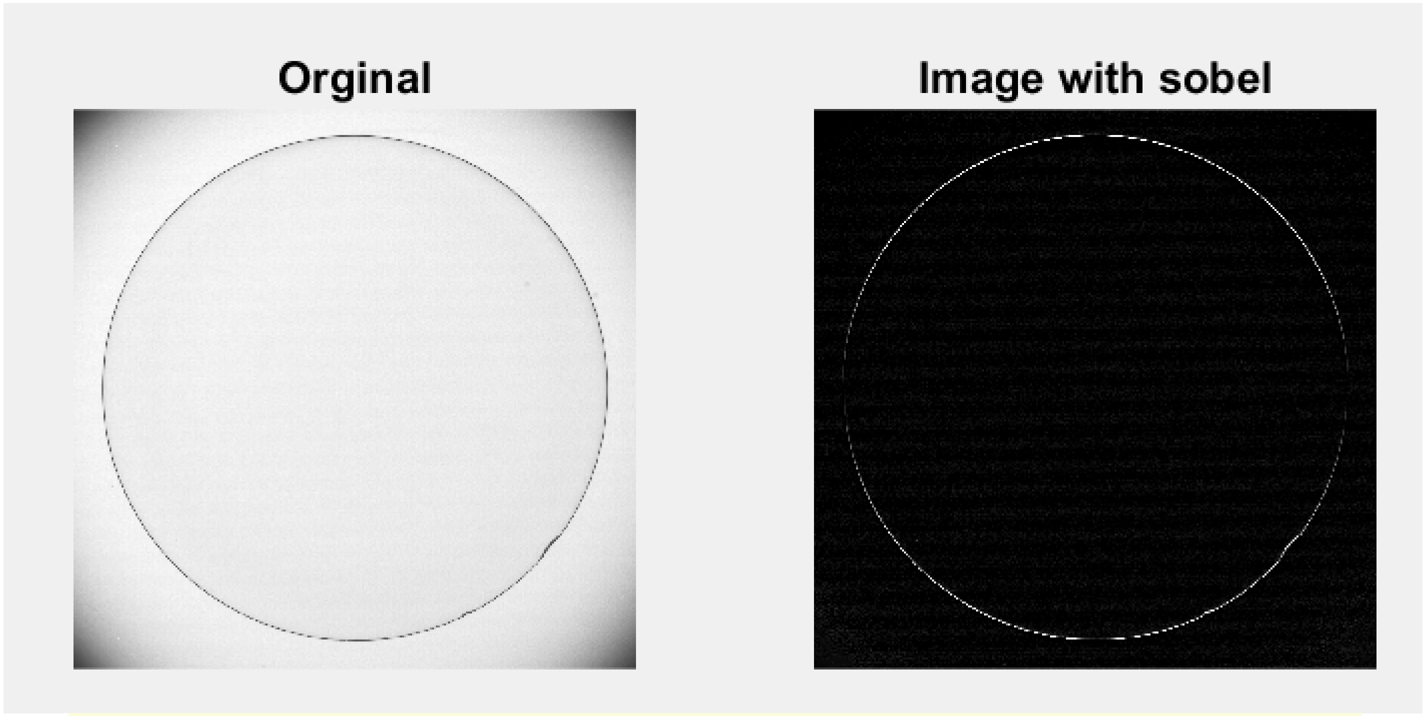
z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

The weight value 2 is to achieve smoothing by giving more important to the center point

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Example



```
%% sobel transform  %2B_PP42
f1 = imread('circle.tif');
w = fspecial('sobel');
g1 = imfilter(f1, w);
figure;
subplot(1,2,1);imshow(f1); title('Orginal');
subplot(1,2,2);imshow(g1);title('Image with sobel');
```