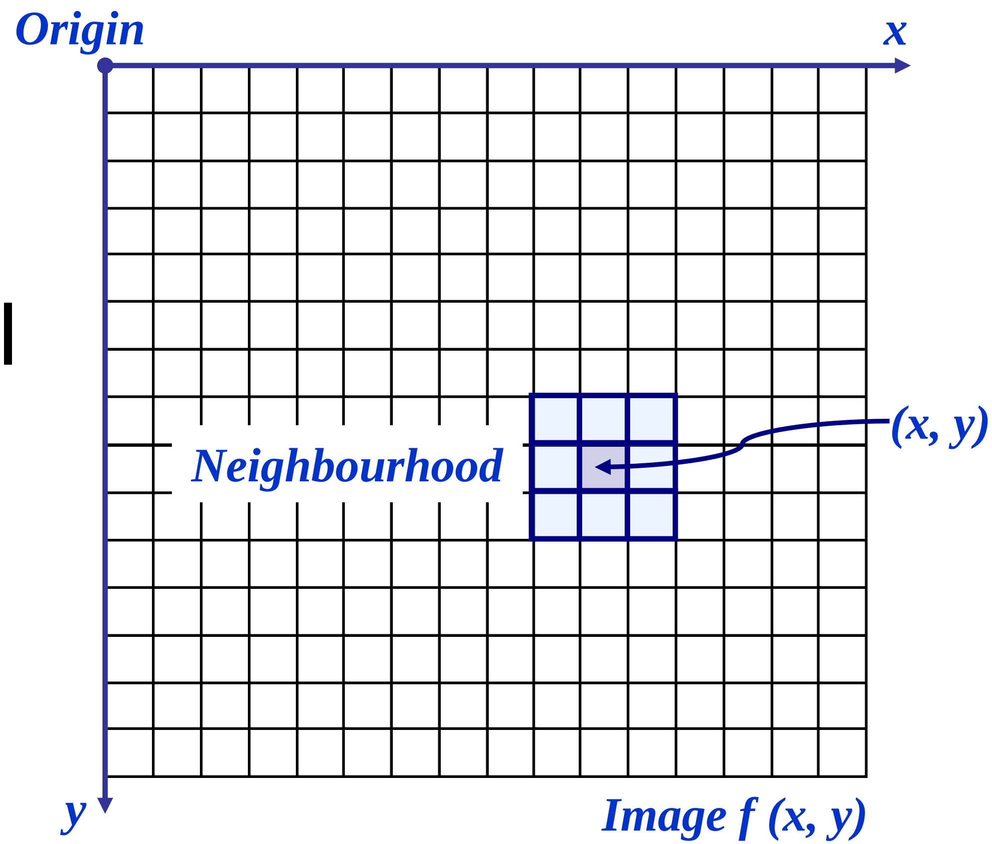
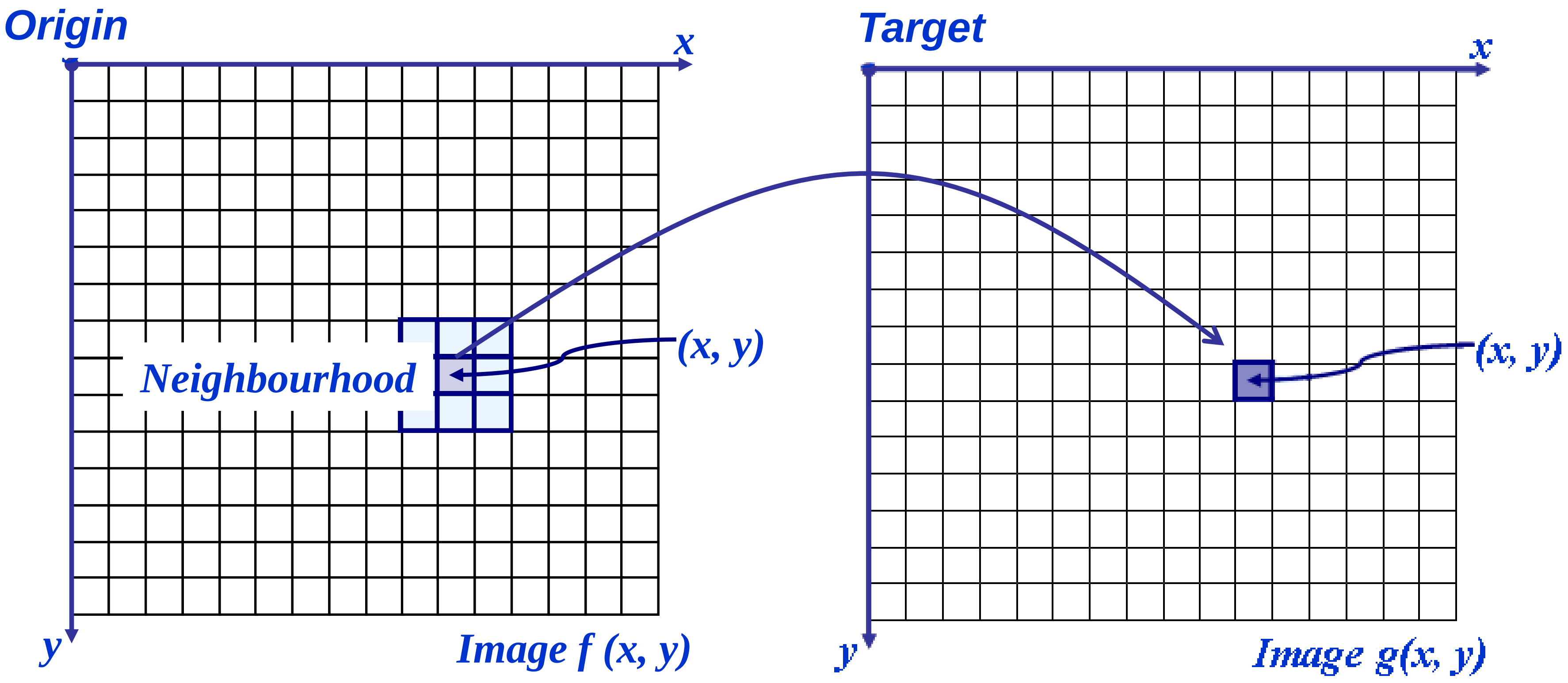
Neighbourhood Operations

- Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations
- Neighbourhoods are mostly a rectangle around a central pixel
- Any size rectangle and any shape filter are possible



Neighbourhood Operations

For each pixel in the origin image, the outcome is written on the same location at the target image.

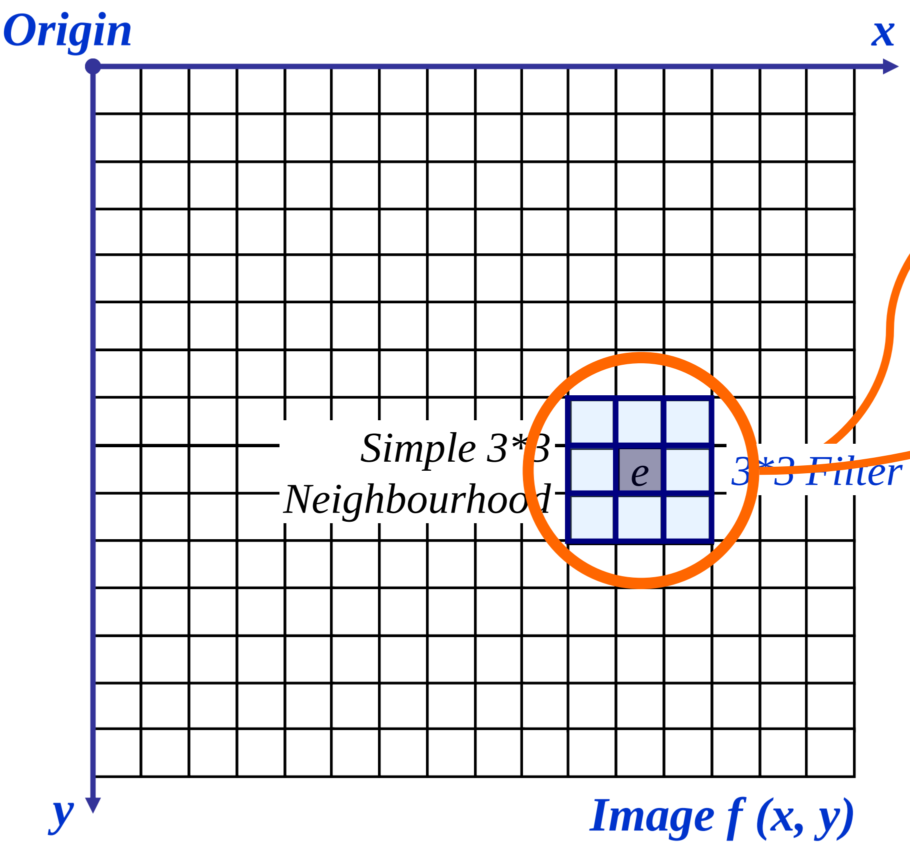


Simple Neighbourhood Operations

Simple neighbourhood operations example:

- **Min:** Set the pixel value to the minimum in the neighbourhood
- **Max:** Set the pixel value to the maximum in the neighbourhood

The Spatial Filtering Process



a	*b*	*c*
d	*e*	*f*
g	*h*	*i*

Original Image Pixels

$*$

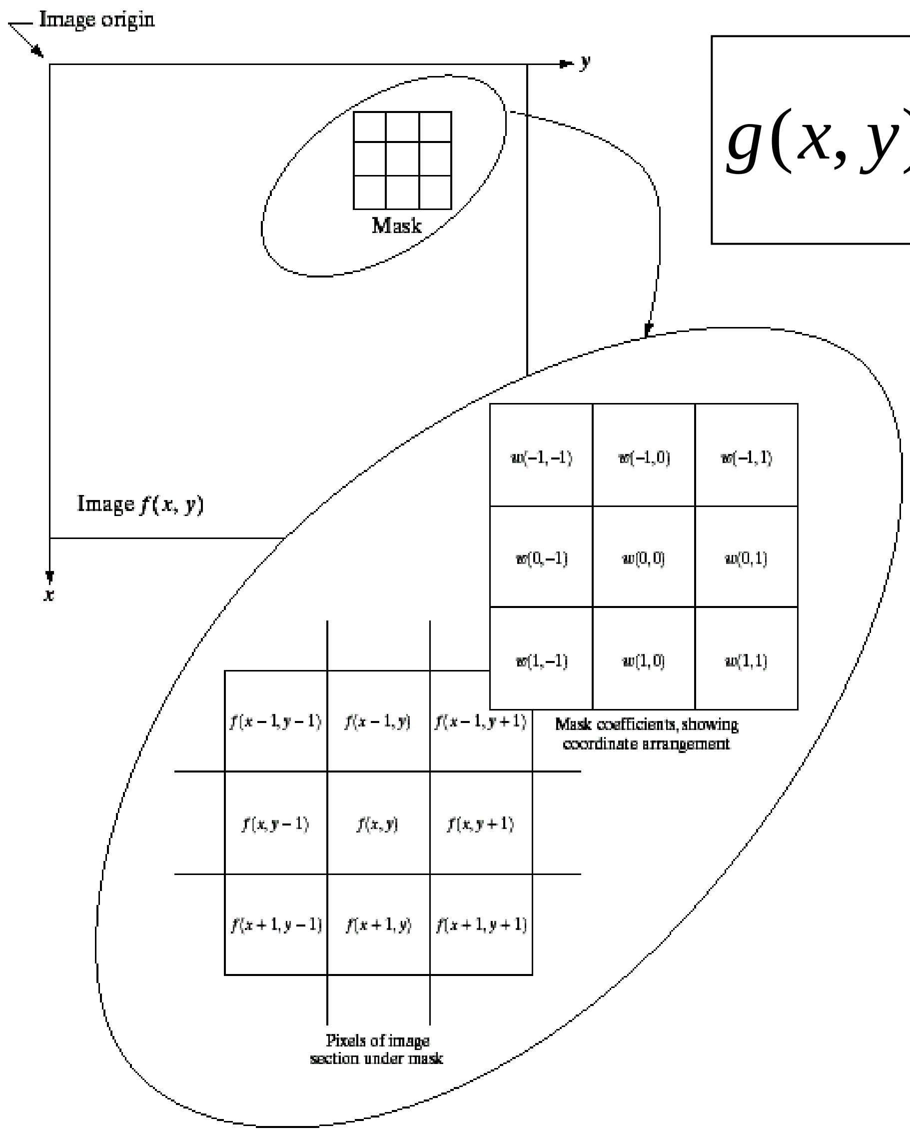
j	*k*	*l*
m	*n*	*o*
p	*q*	*r*

Filter (*w*)

$$\begin{aligned} e_{processed} = {} & n*e + \\ & j*a + k*b + l*c + \\ & m*d + o*f + \\ & p*g + q*h + r*i \end{aligned}$$

The above is repeated for every pixel in the original image to generate the filtered image

Spatial Filtering: Equation Form



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

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

Smoothing Spatial Filters

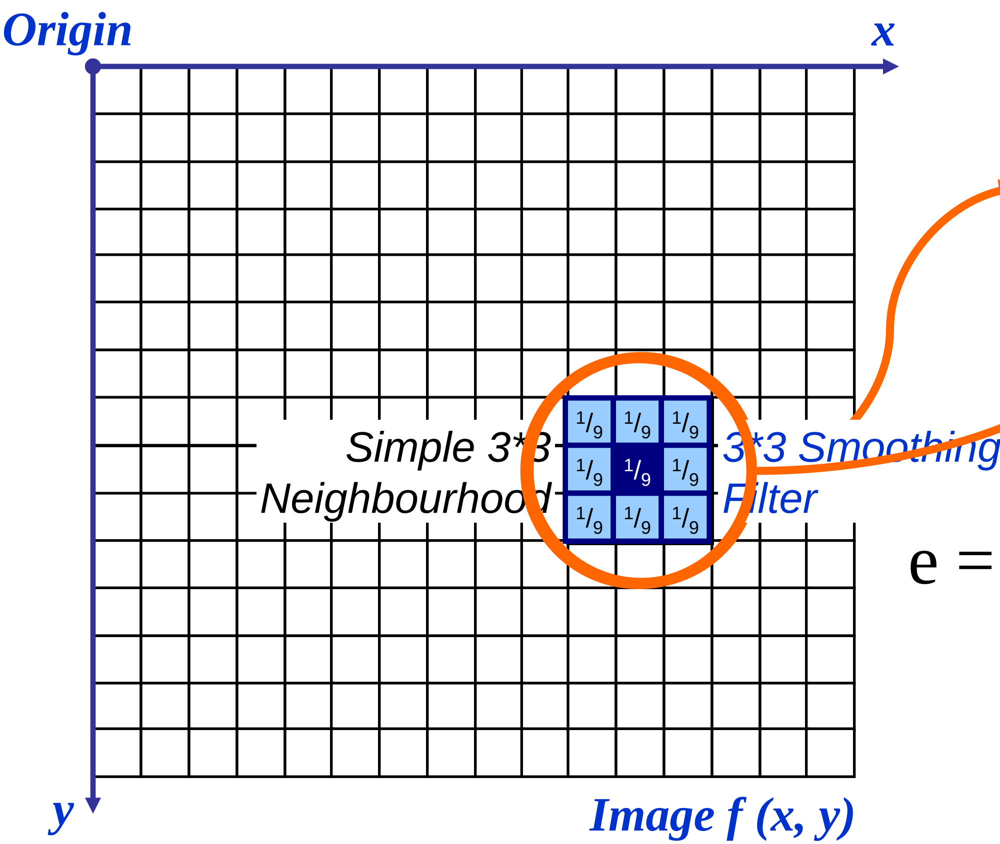
One of the simplest spatial filtering operations we can perform is a smoothing operation

- Simply average all of the pixels in a neighbourhood around a central value
- Especially useful in removing noise from images
- Also useful for highlighting gross detail

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

Simple averaging filter

Smoothing Spatial Filtering



104	100	108
99	106	98
95	90	85

Original Image Pixels

*

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

Filter

$$
\begin{aligned}
e = {} & {}^1/_9*106 + \\
& {}^1/_9*104 + {}^1/_9*100 + {}^1/_9*108 + \\
& {}^1/_9*99 + {}^1/_9*98 + \\
& {}^1/_9*95 + {}^1/_9*90 + {}^1/_9*85 \\
= {} & 98.3333
\end{aligned}
$$

The above is repeated for every pixel in the original image to generate the smoothed image

Image Smoothing Example

- The image at the top left is an original image of size 500*500 pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes
  - 3, 5, 9, 15 and 35
- Notice how detail begins to disappear

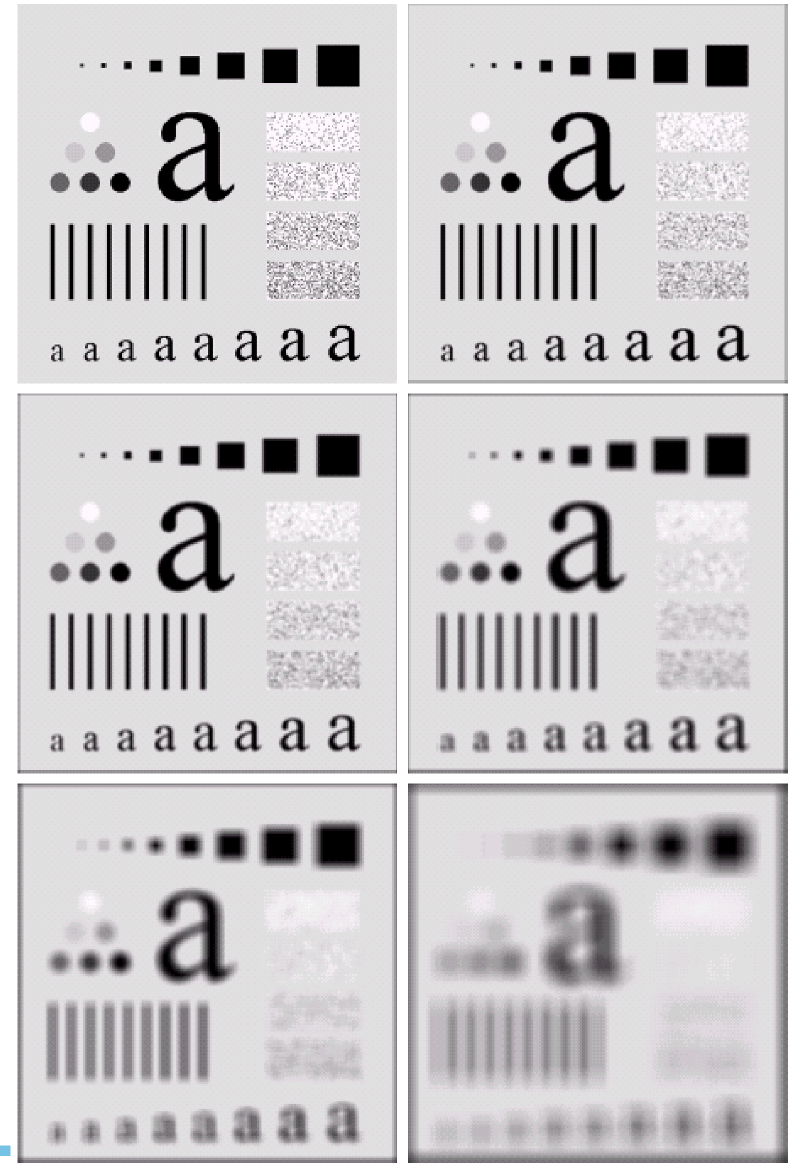


Image Smoothing Example

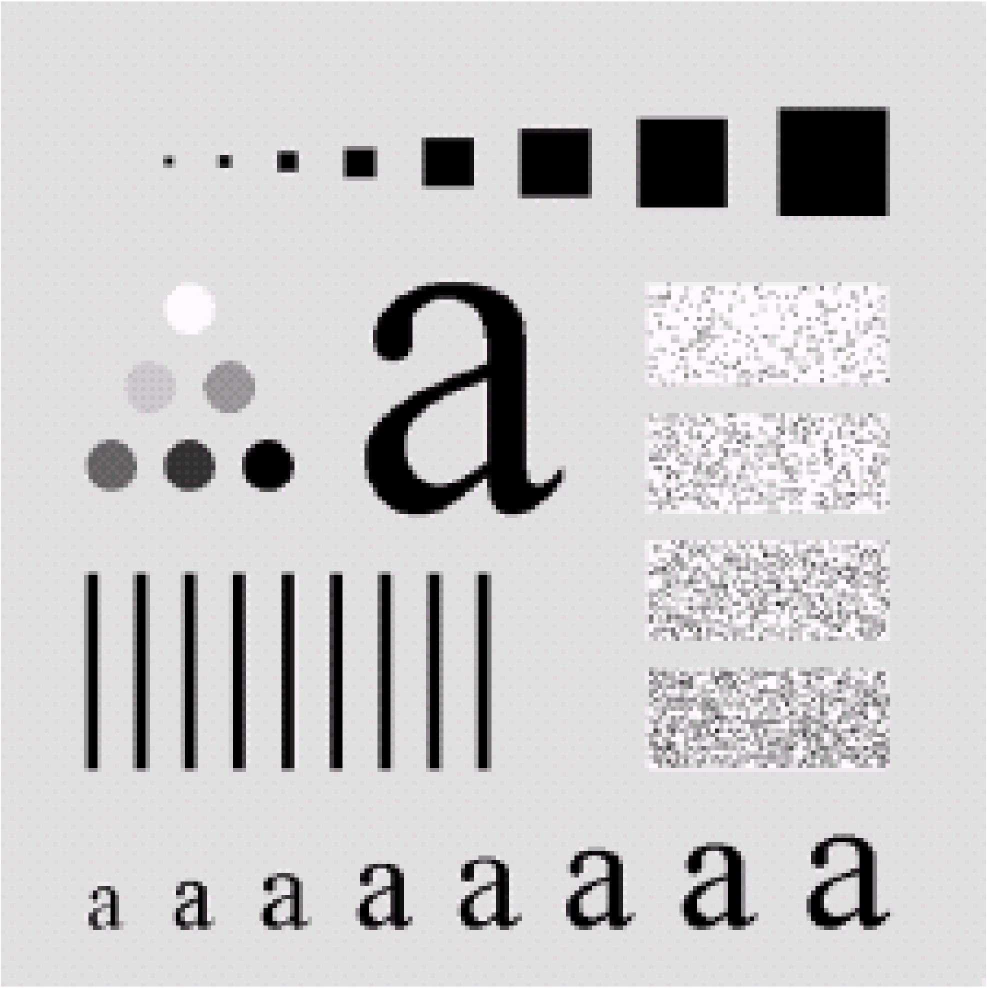


Image Smoothing Example

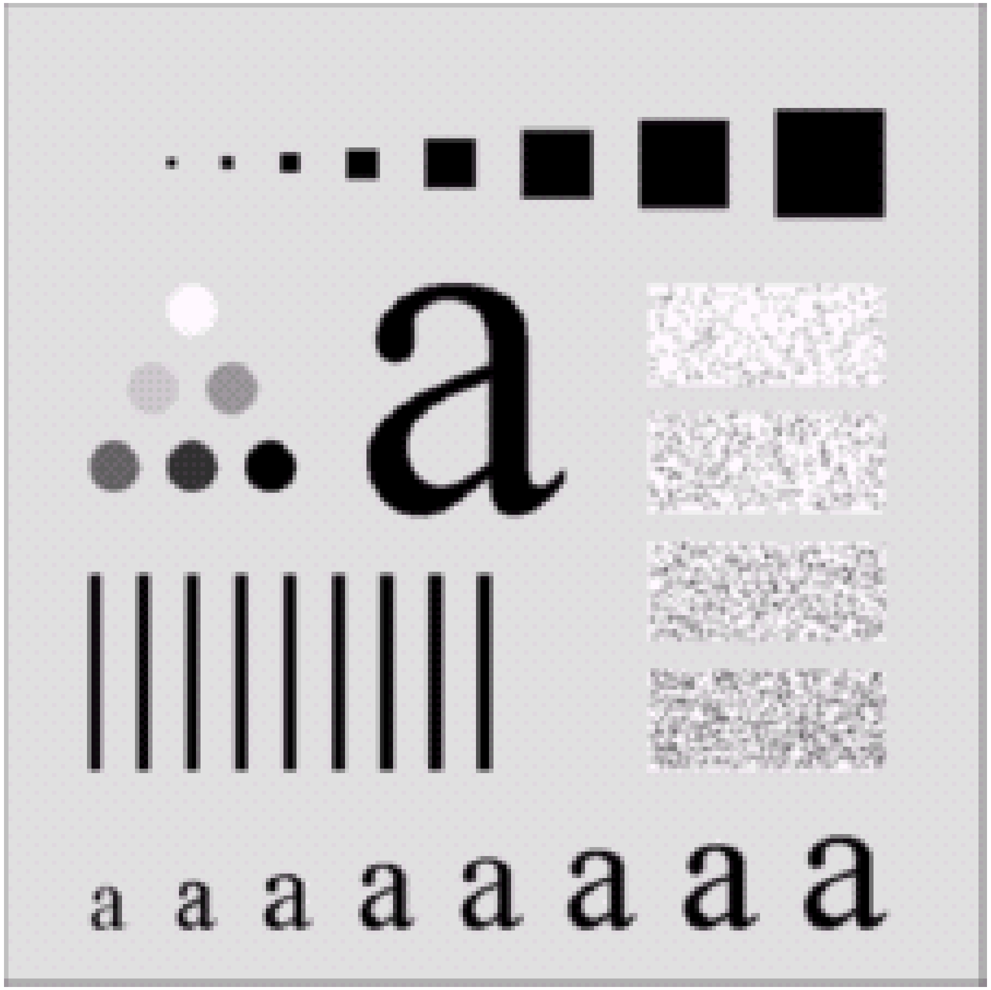


Image Smoothing Example

Image Smoothing Example

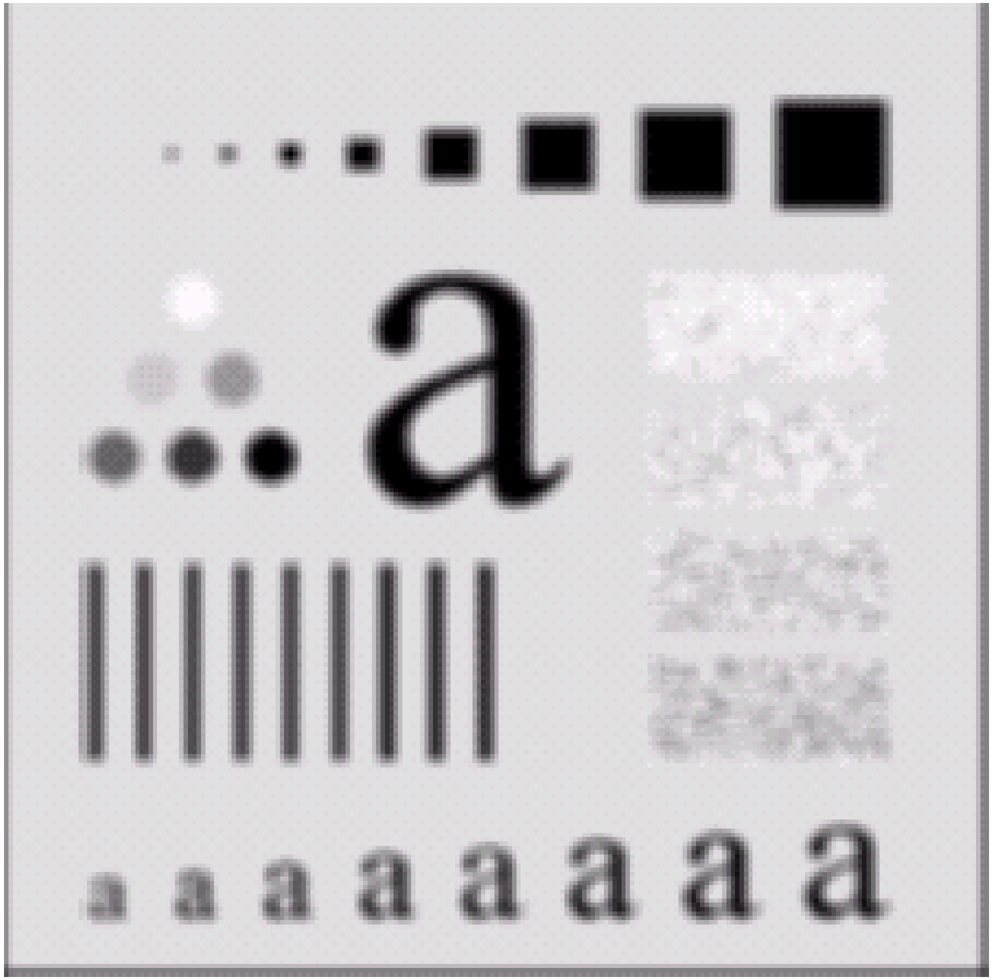


Image Smoothing Example

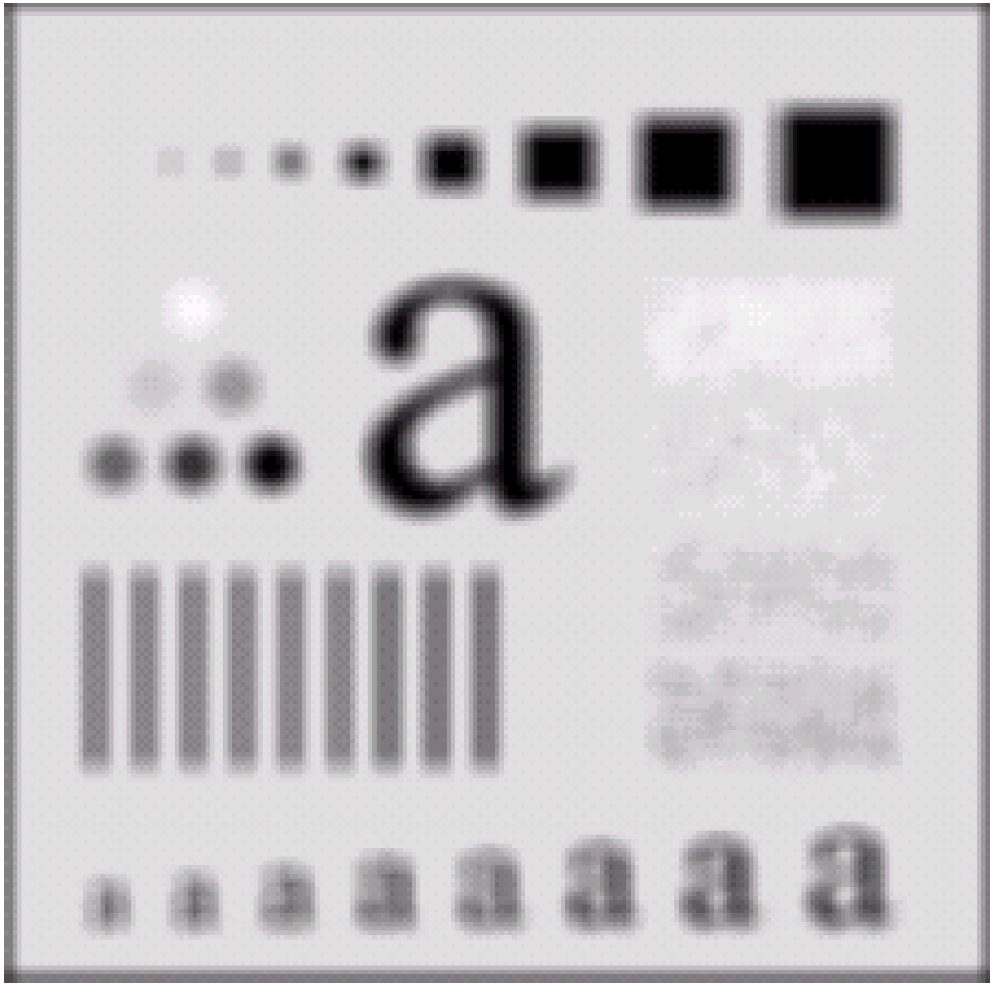
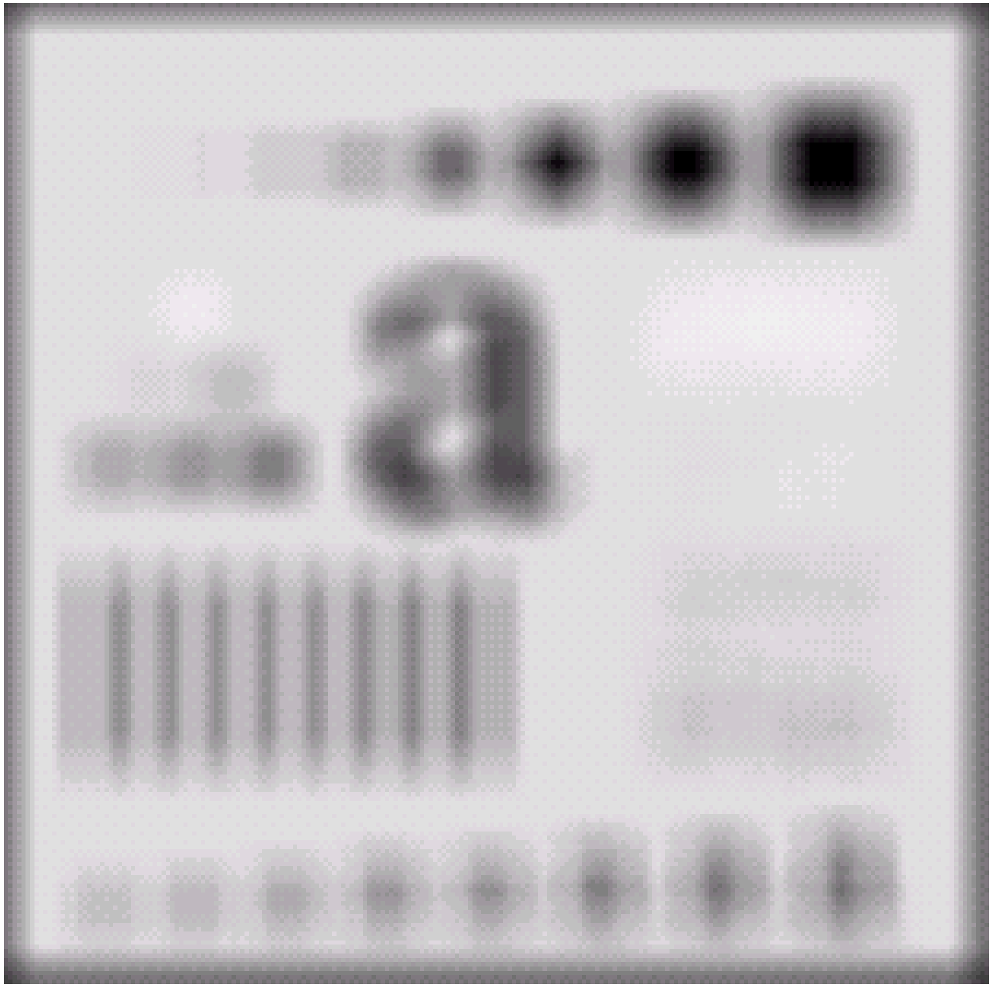


Image Smoothing Example



Weighted Smoothing Filters

More effective smoothing filters can be generated by allowing different pixels in the neighbourhood different weights in the averaging function

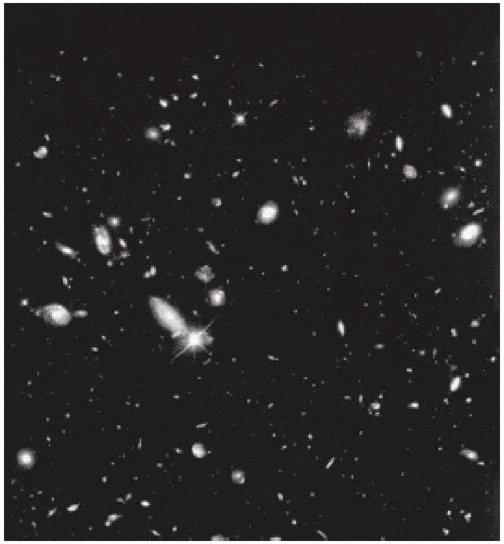
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

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

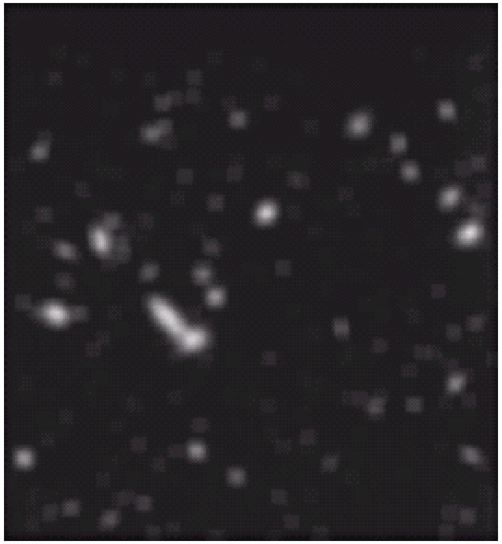
Weighted averaging filter

Another Smoothing Example

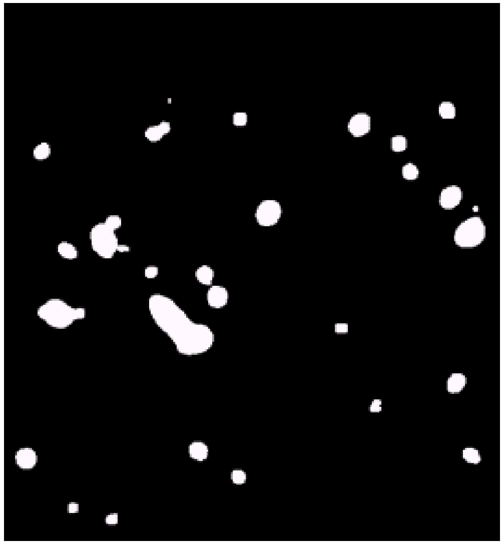
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image

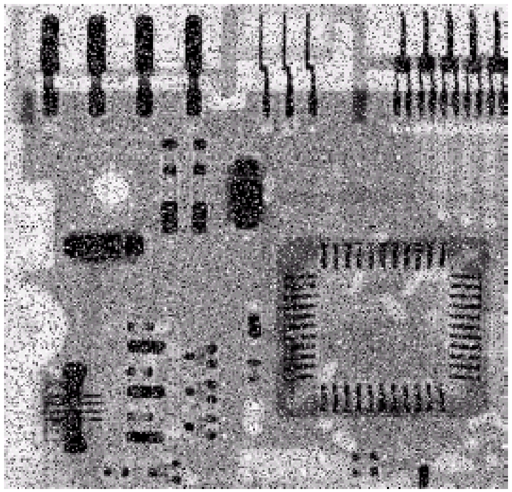


Smoothed Image



Thresholded Image

Averaging Filter Vs. Median Filter Example



Original Image With Noise

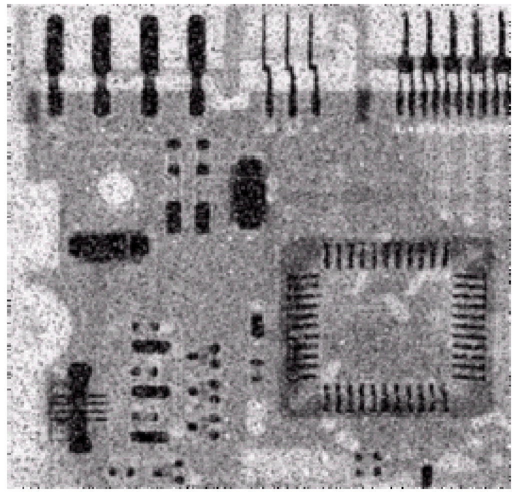


Image After Averaging Filter

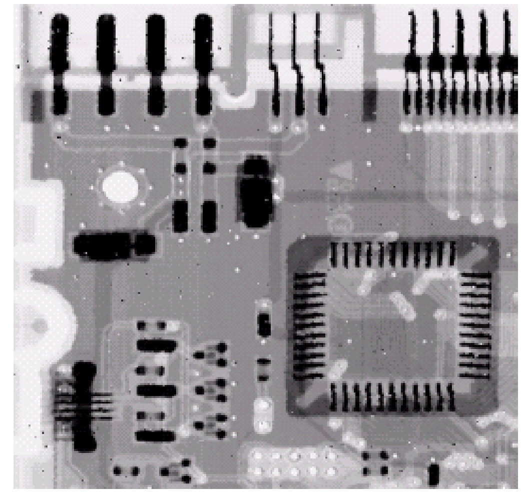


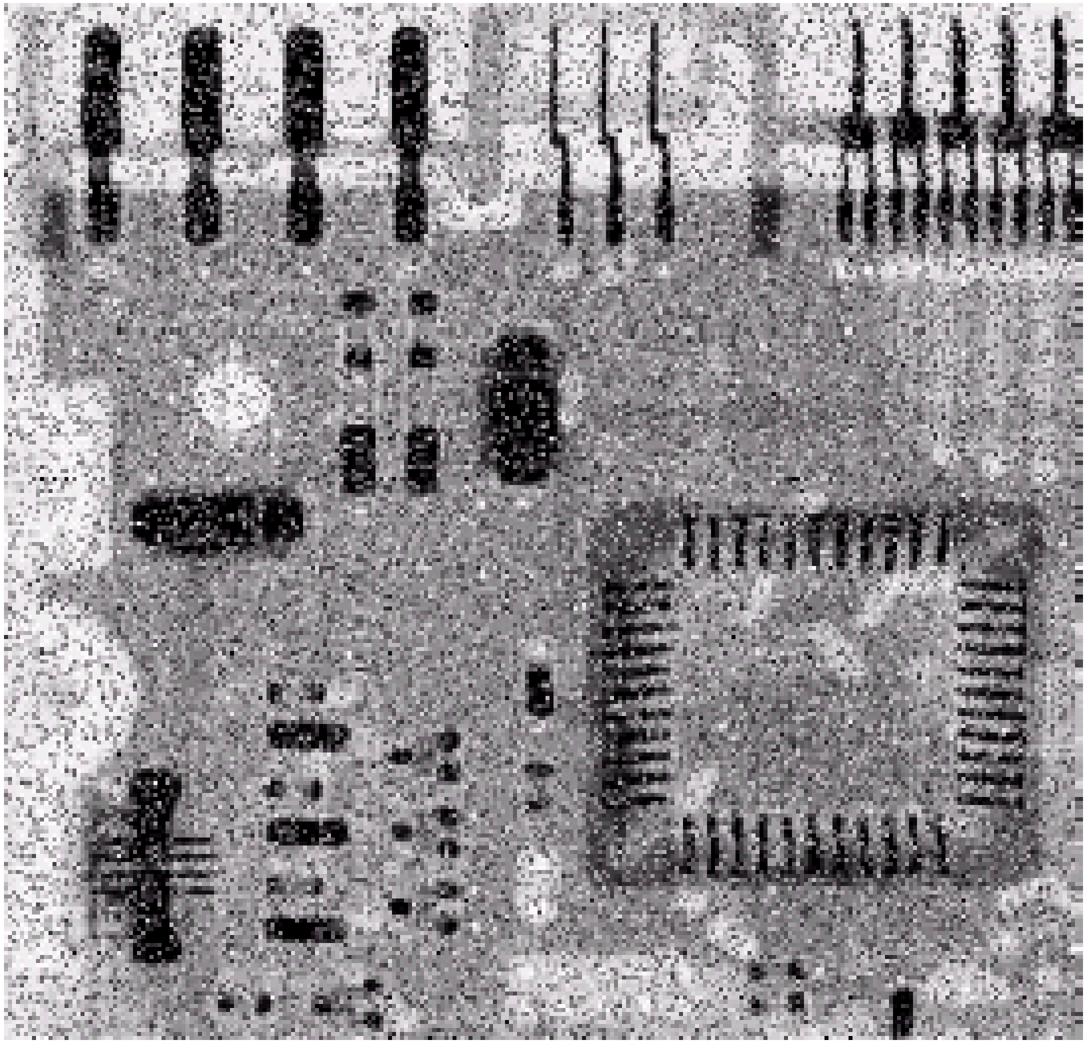
Image After Median Filter

Filtering is often used to remove noise from images

Sometimes a median filter works better than an averaging filter

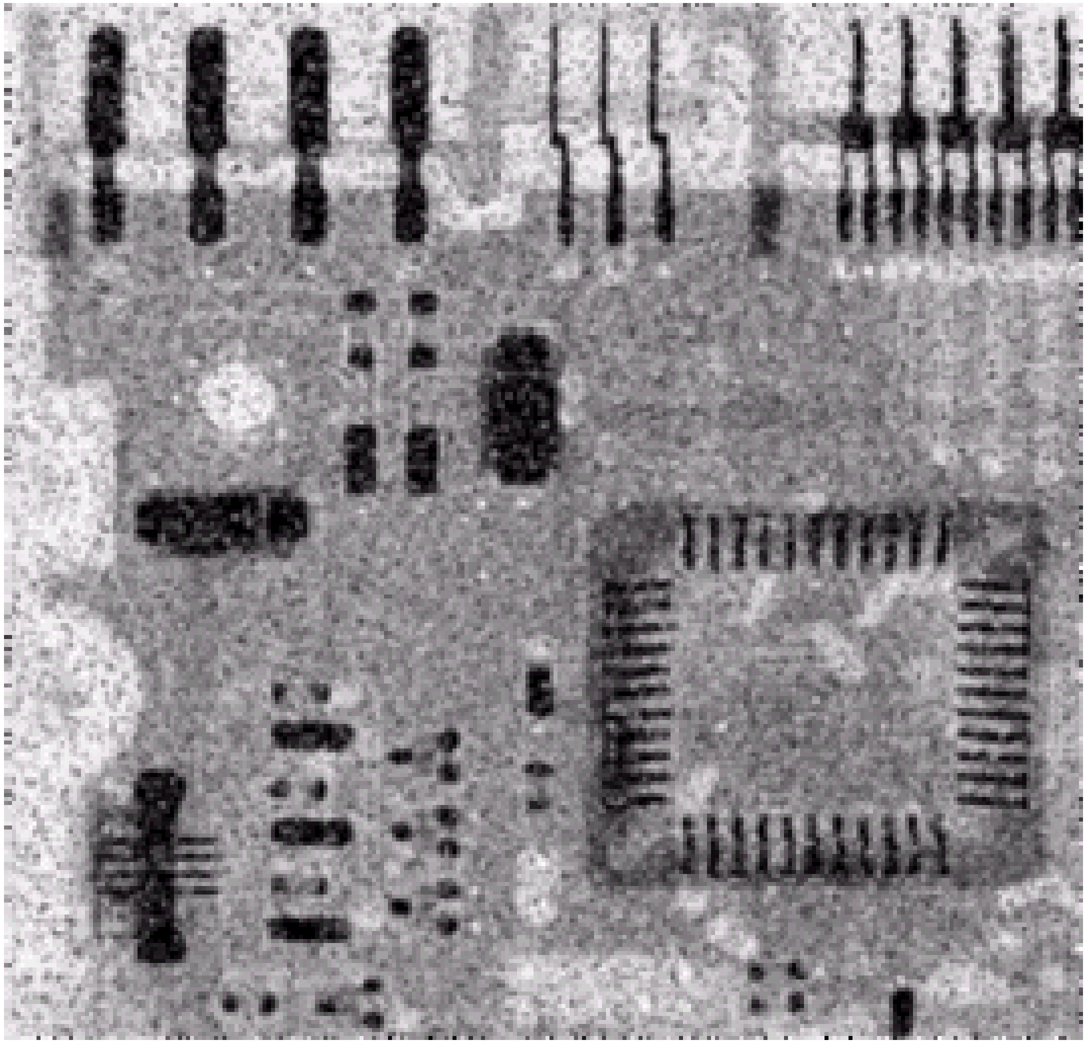
Averaging Filter Vs. Median Filter Example

Original



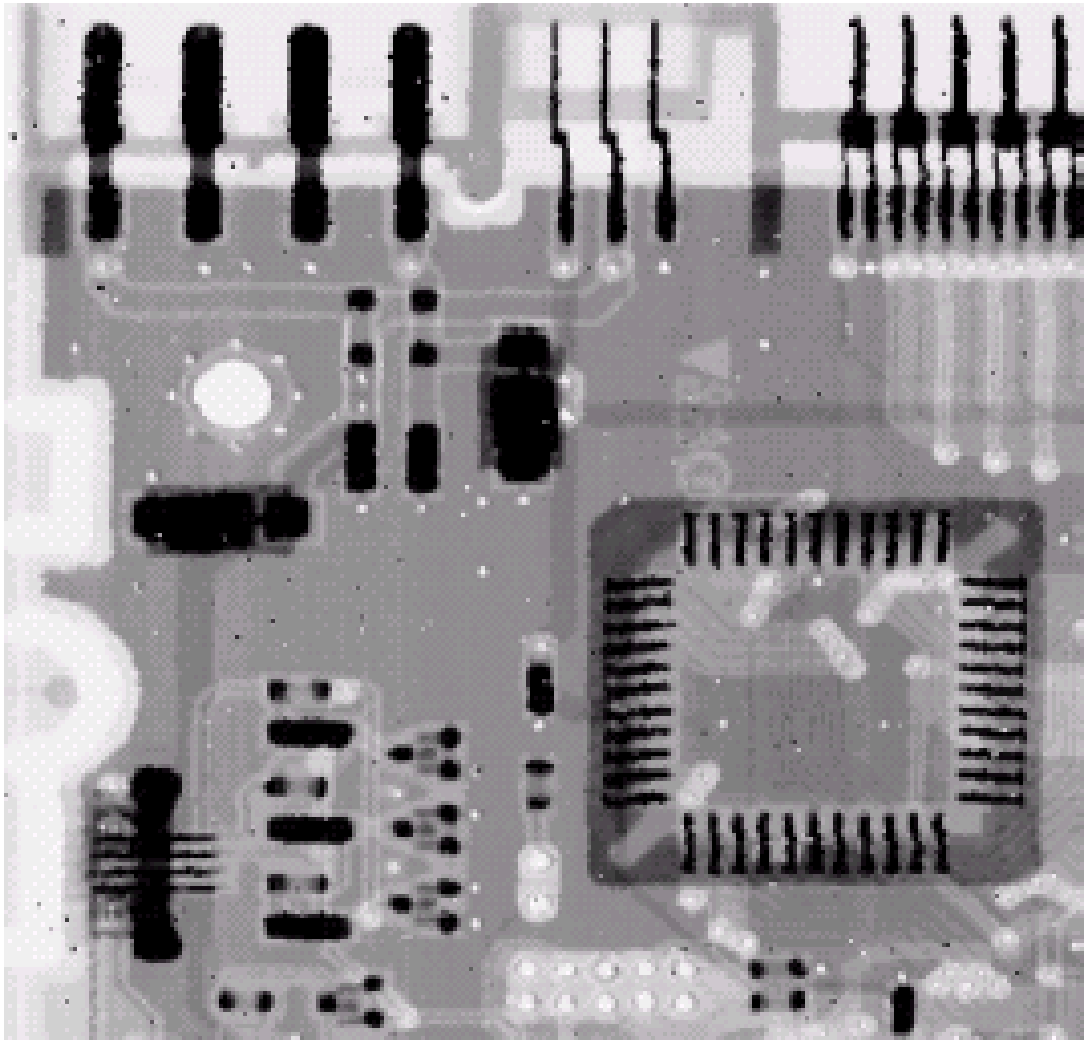
Averaging Filter Vs. Median Filter Example

Averaging Filter



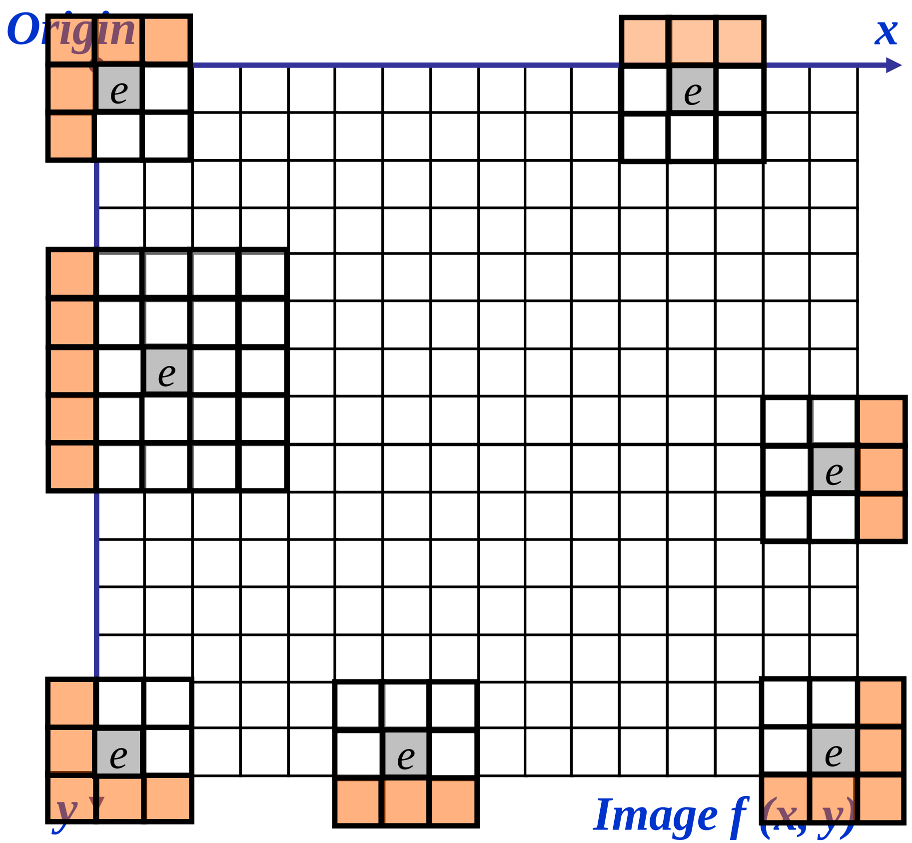
Averaging Filter Vs. Median Filter Example

Median Filter



Strange Things Happen At The Edges!

At the edges of an image we are missing pixels to form a neighbourhood



Strange Things Happen At The Edges! (cont…)

There are a few approaches to dealing with missing edge pixels:

- Omit missing pixels
  - Only works with some filters
  - Can add extra code and slow down processing
- Pad the image
  - Typically with either all white or all black pixels
- Replicate border pixels
- Truncate the image

Correlation & Convolution

The filtering we have been talking about so far is referred to as *correlation* with the filter itself referred to as the *correlation kernel*

Convolution is a similar operation, with just one subtle difference

a	b	c
d	e	e
f	g	h

Original Image Pixels

$*$

r	s	t
u	v	w
x	y	z

Filter

$$
\begin{aligned}
e_{processed} = {} & v*e + \\
& z*a + y*b + x*c + \\
& w*d + u*e + \\
& t*f + s*g + r*h
\end{aligned}
$$

For symmetric filters it makes no difference

Sharpening Spatial Filters

Previously we have looked at smoothing filters which remove fine detail

Sharpening spatial filters seek to highlight fine detail

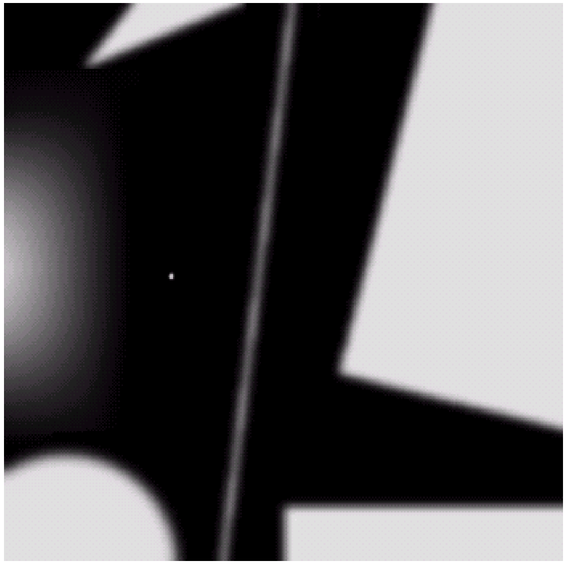
- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

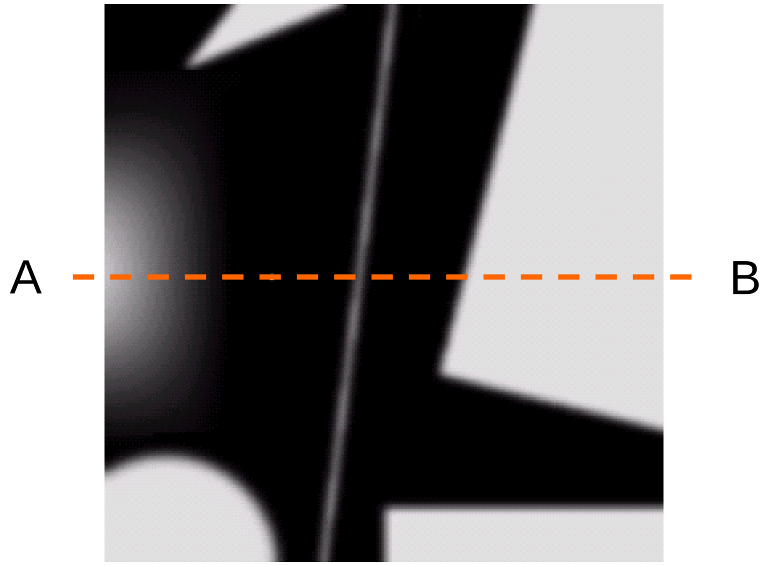
Spatial Differentiation

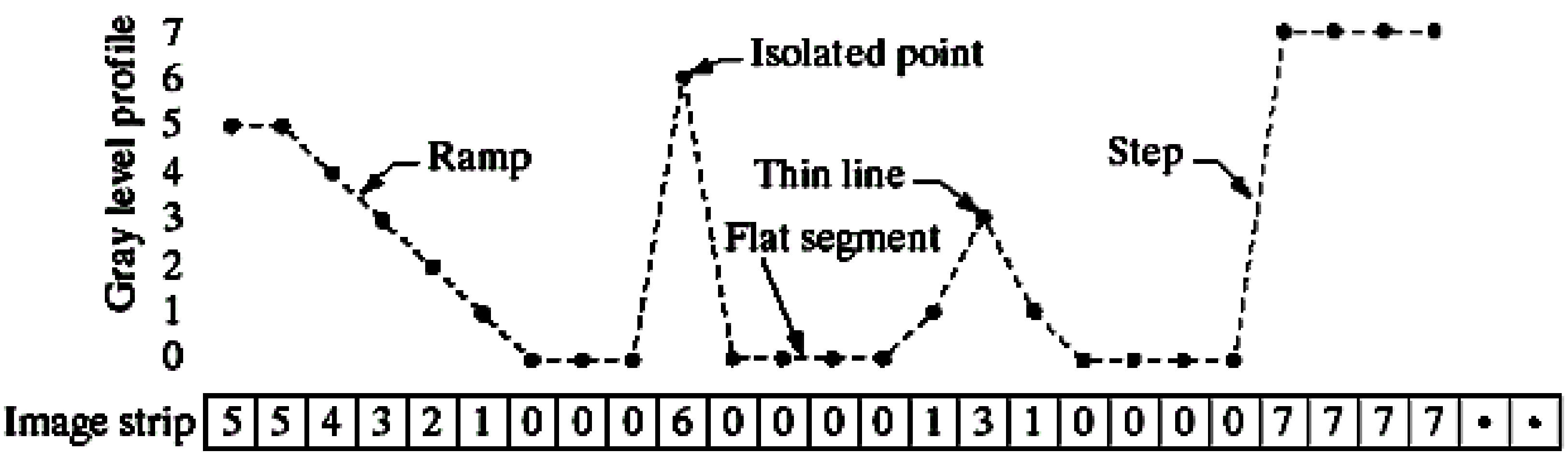
Differentiation measures the *rate of change* of a function

Let's consider a simple 1 dimensional example



Spatial Differentiation





1st Derivative

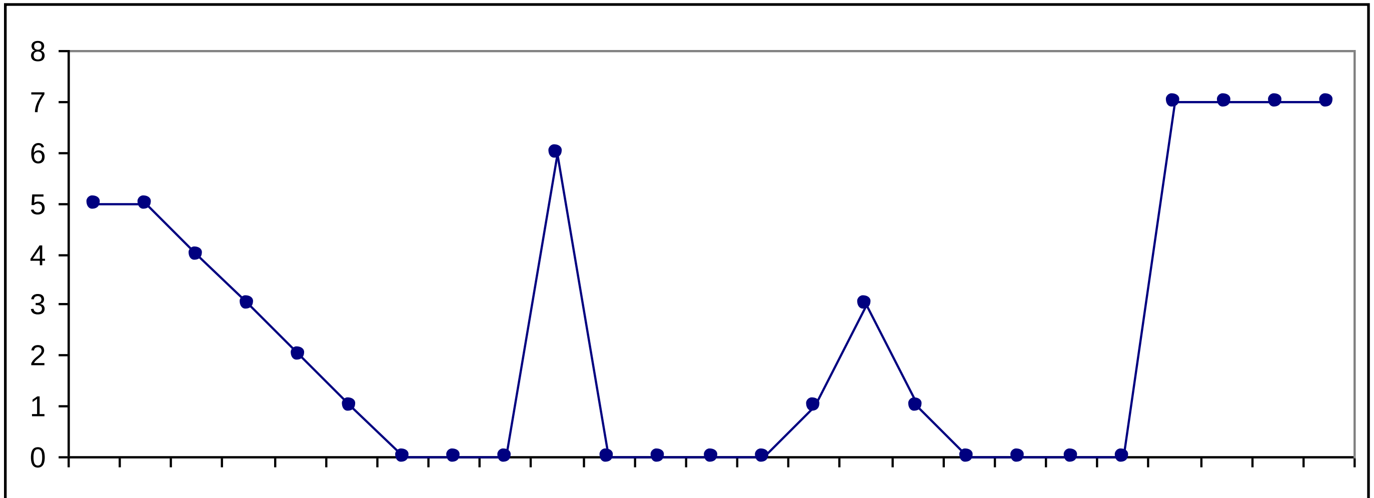
The formula for the 1st derivative of a function is as follows:

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

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

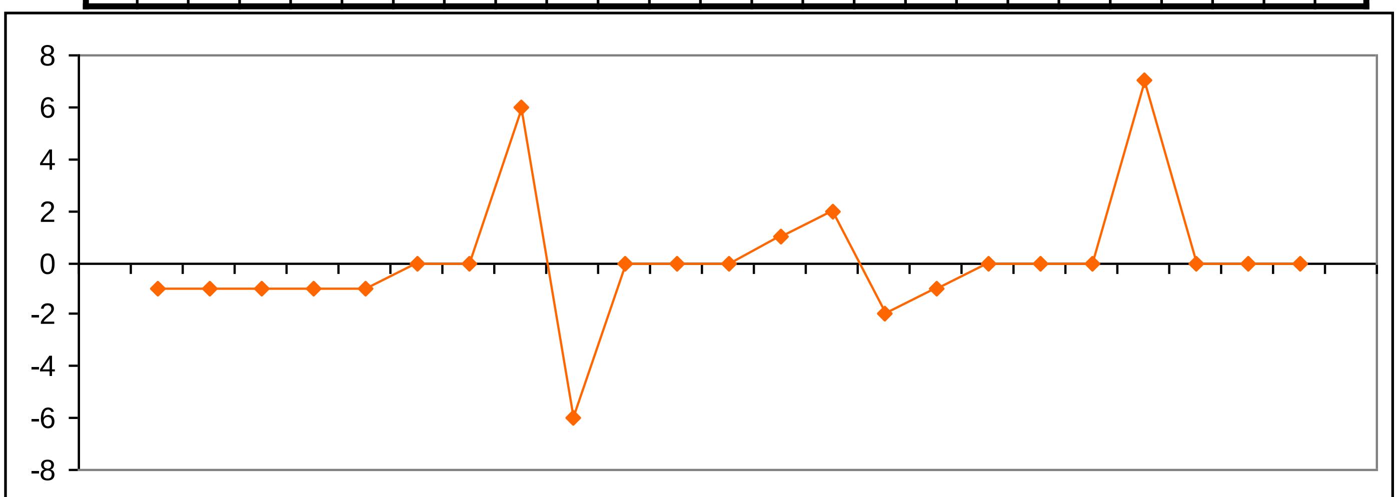
1st Derivative (cont…)

f(x)



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
	0	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0	

f'(x)



2^{nd} Derivative

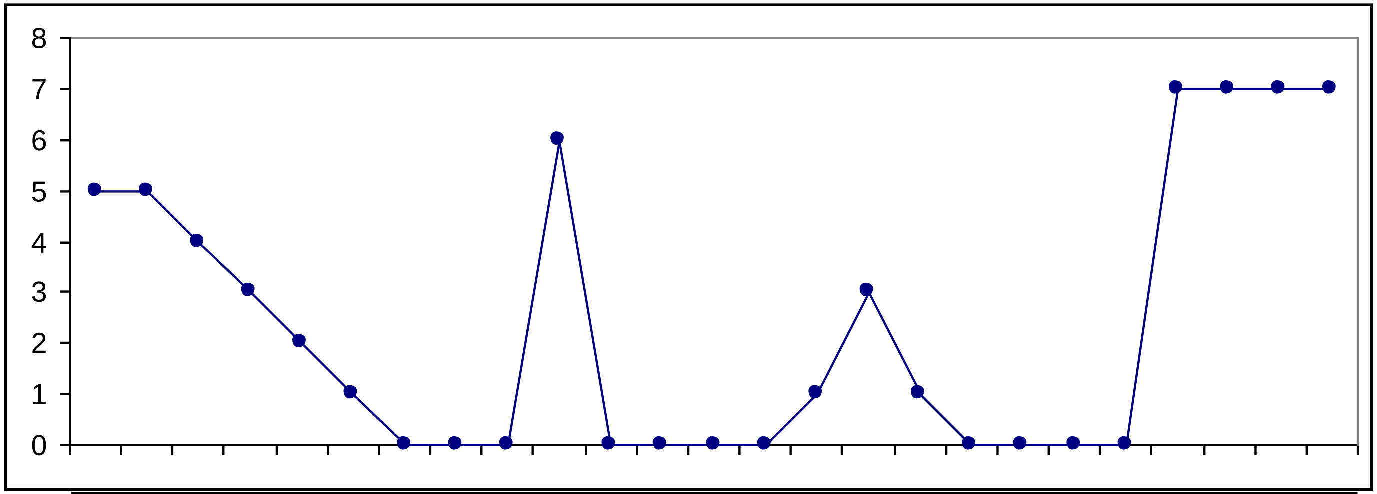
The formula for the 2^{nd} derivative of a function is as follows:

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

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

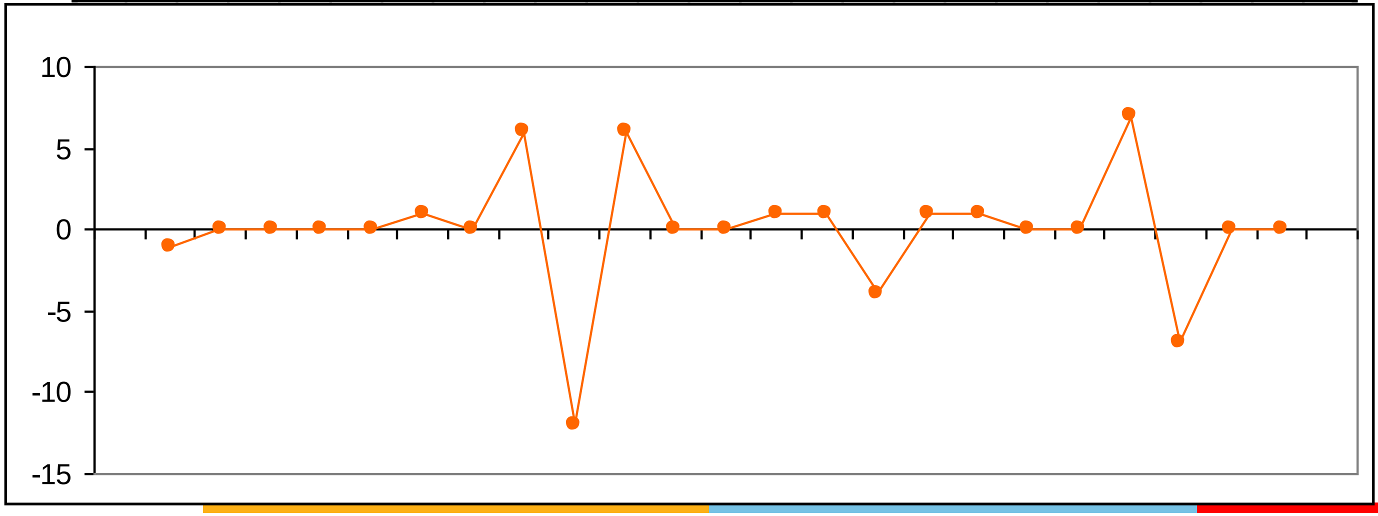
2^{nd} Derivative (cont…)

f(x)

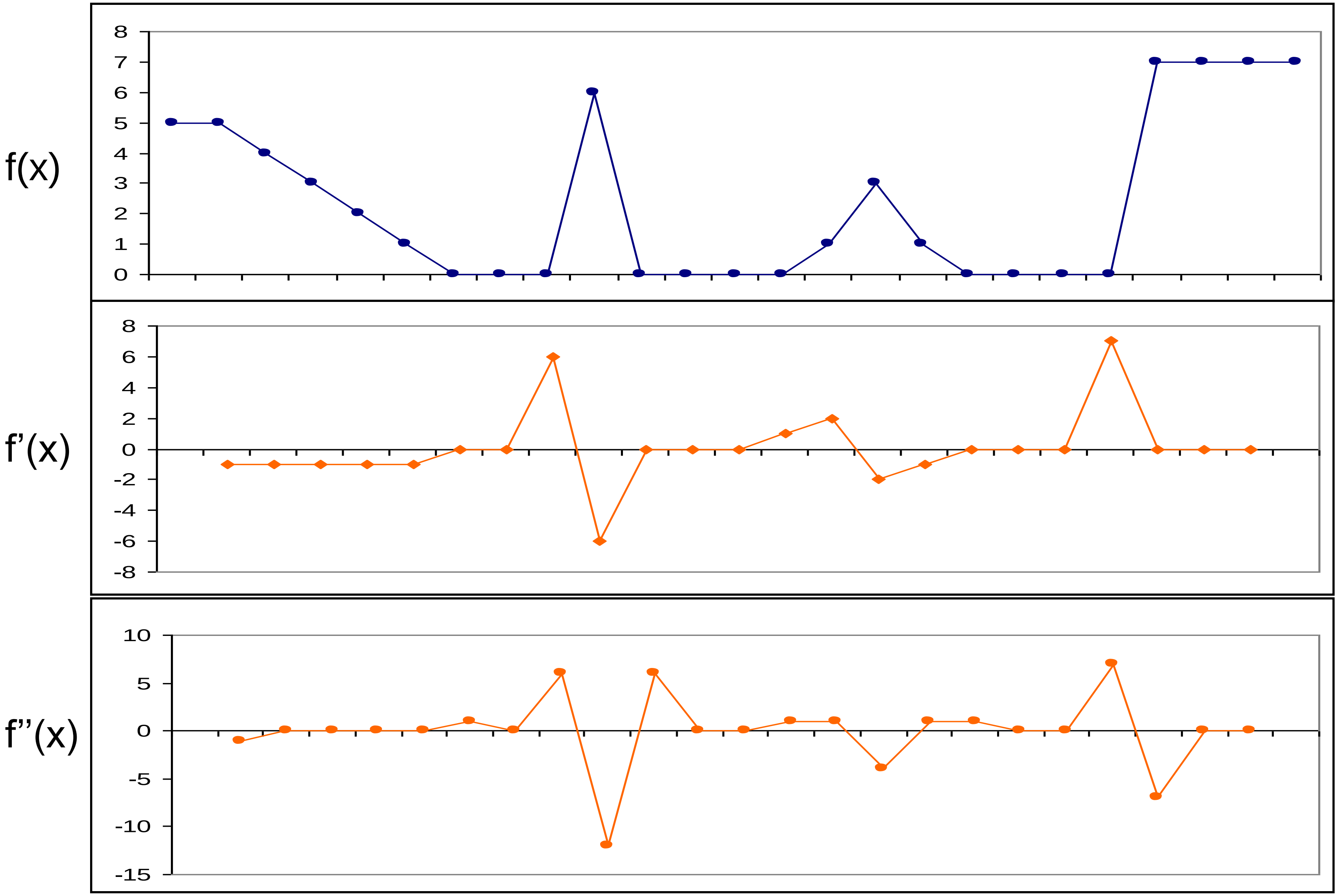


5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7
	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	

f''(x)



1st and 2nd Derivative



Using Second Derivatives For Image Enhancement

- The 2nd derivative is more useful for image enhancement than the 1st derivative
  - Stronger response to fine detail
  - Simpler implementation
  - We will come back to the 1st order derivative later on
- The first sharpening filter we will look at is the *Laplacian*
  - Isotropic
  - One of the simplest sharpening filters
  - We will look at a digital implementation

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 1st 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 (cont…)

So, the Laplacian can be given as follows:

$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

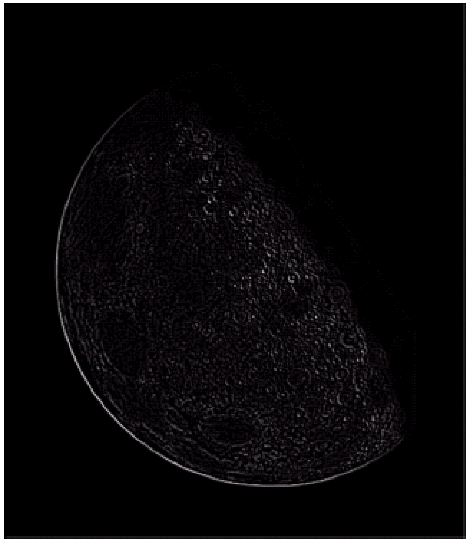
We can easily build a filter based on this

0	1	0
1	-4	1
0	1	0

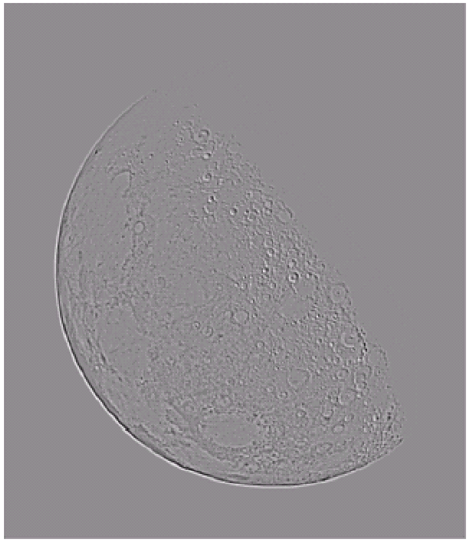
The Laplacian (cont…)

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

Original Image



Laplacian Filtered Image

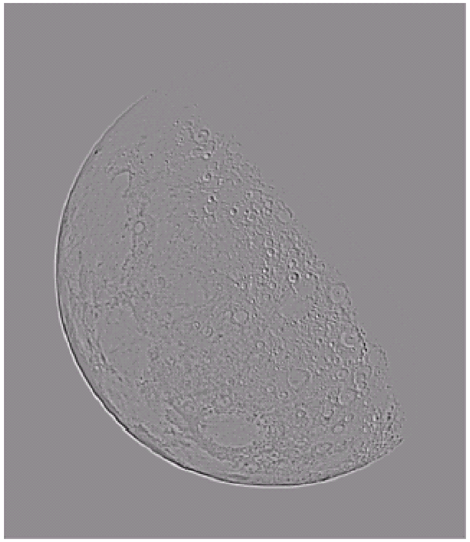


Laplacian Filtered Image Scaled for Display

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

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

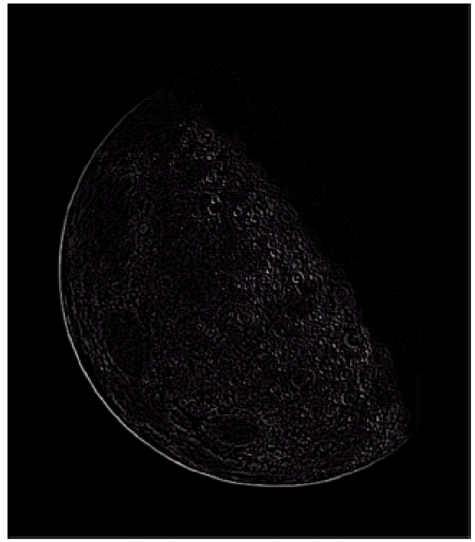


Laplacian Filtered Image Scaled for Display

Laplacian Image Enhancement

Original Image

\-



Laplacian Filtered Image

=

Sharpened Image

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

Laplacian Image Enhancement

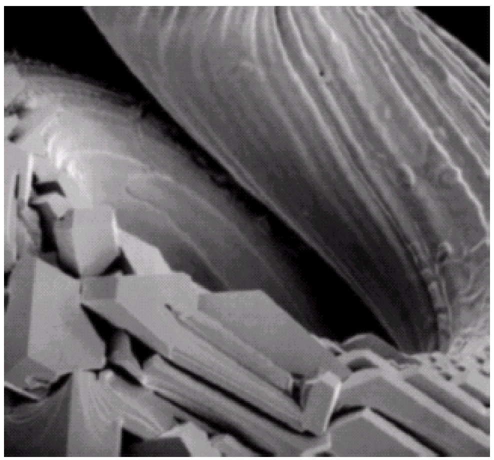
Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

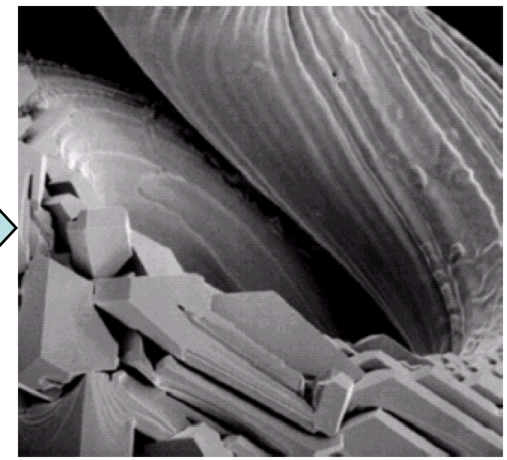
$$
\begin{aligned}
g(x, y) &= f(x, y) - \nabla^2 f \\
&= f(x, y) - [f(x+1, y) + f(x-1, y) \\
&\qquad + f(x, y+1) + f(x, y-1) \\
&\qquad - 4f(x, y)] \\
&= 5f(x, y) - f(x+1, y) - f(x-1, y) \\
&\qquad - f(x, y+1) - f(x, y-1)
\end{aligned}
$$

Simplified Image Enhancement (cont…)

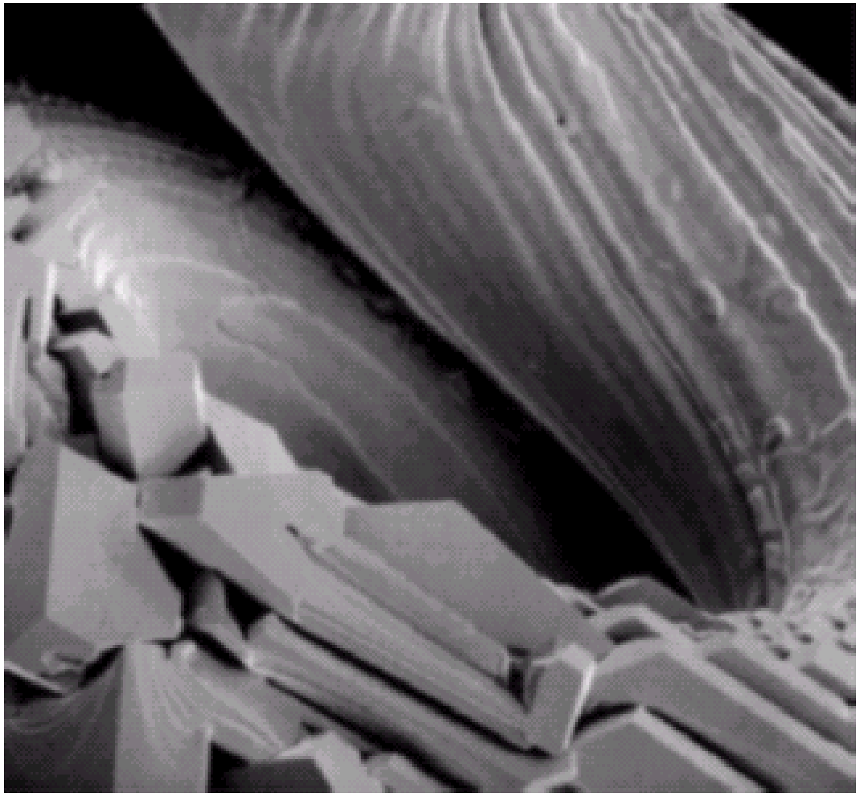
This gives us a new filter which does the whole job for us in one step

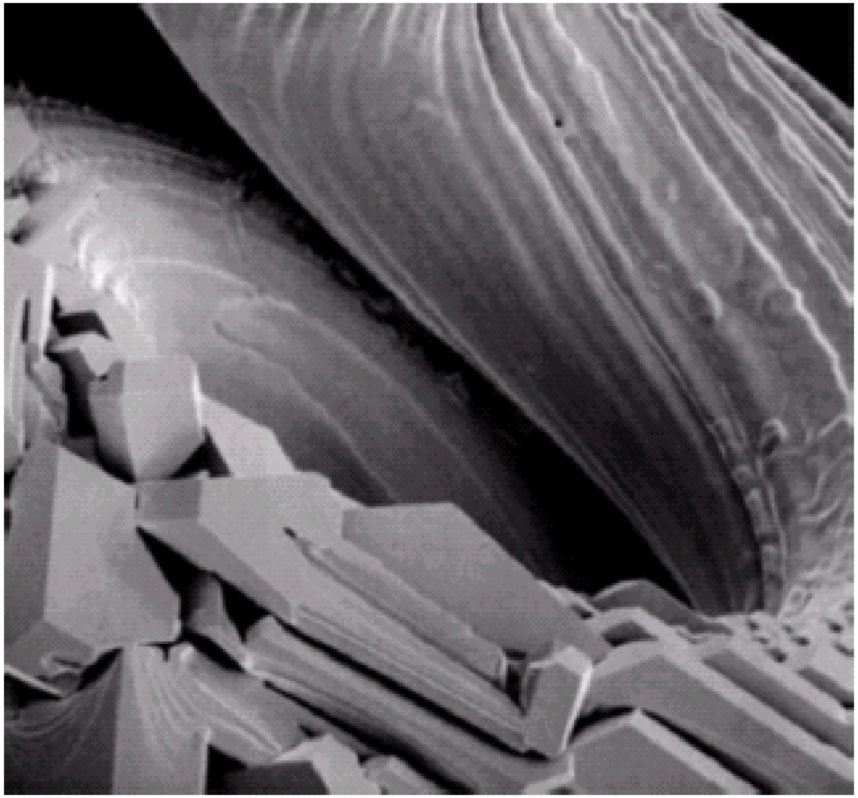


0	-1	0
-1	5	-1
0	-1	0



Simplified Image Enhancement (cont…)





Variants On The Simple Laplacian

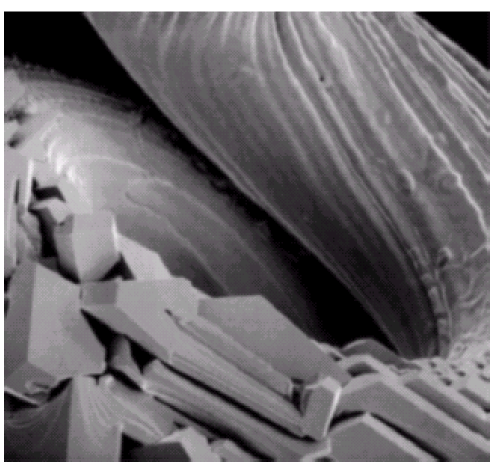
There are lots of slightly different versions of the Laplacian that can be used:

0	1	0
1	-4	1
0	1	0

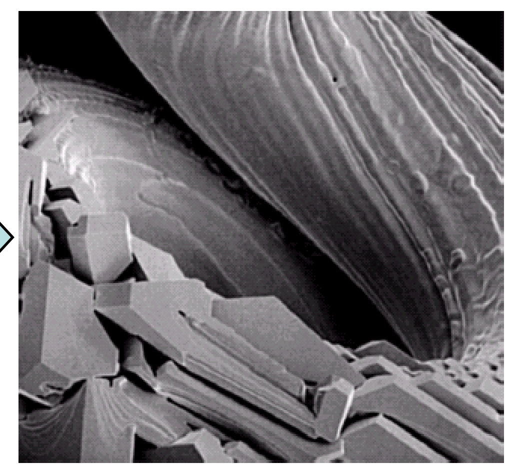
Simple Laplacian

1	1	1
1	-8	1
1	1	1

Variant of Laplacian



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



Unsharp Mask & Highboost Filtering

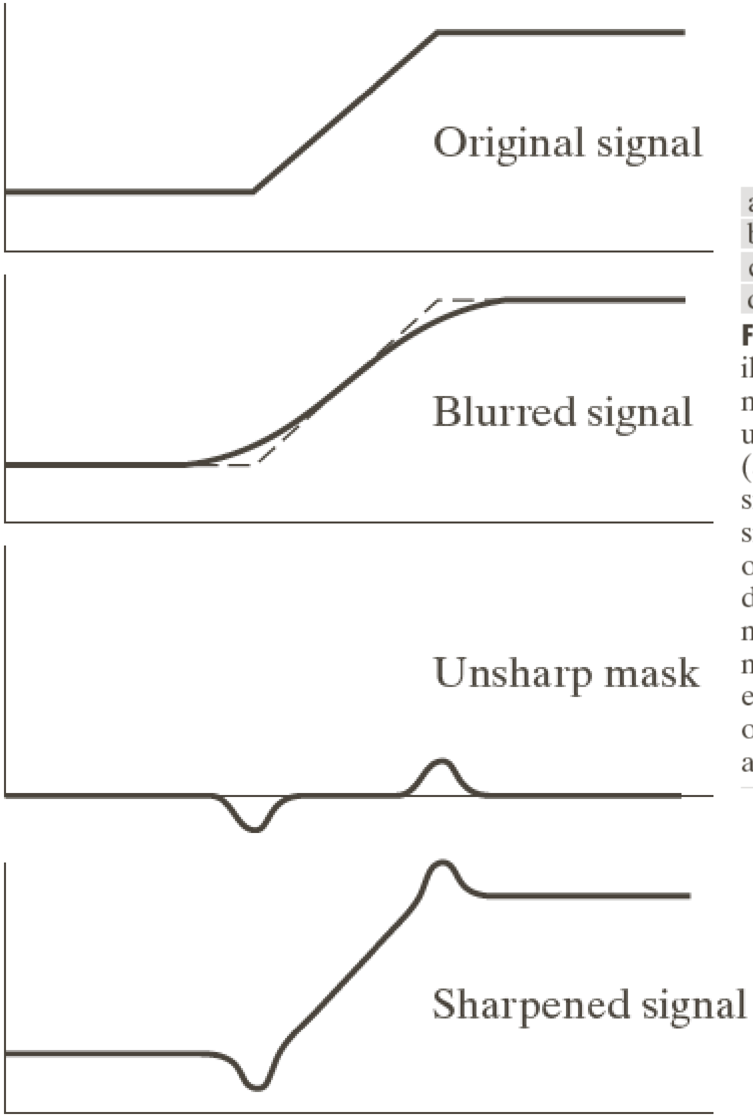
Using sequence of linear spatial filters in order to get Sharpening effect.

-Blur

- Subtract from original image

- add resulting mask to original image

Highboost Filtering



a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

a
b
c
d
e

FIGURE 3.40 (a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.

1st Derivative Filtering

Implementing 1st derivative filters is difficult in practice

For a function $f(x, y)$ the gradient of f at coordinates (x, y) is given as the column vector:

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

1st Derivative Filtering (cont…)

The magnitude of this vector is given by:

$$\begin{aligned}\nabla f &= mag(\nabla \mathrm{f}) \\ &= \left[G_x^2 + G_y^2\right]^{1/2} \\ &= \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}\end{aligned}$$

For practical reasons this can be simplified as:

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

1st Derivative Filtering (cont…)

There is some debate as to how best to calculate these gradients but we will use:

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right|$$

$$+ \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

which is based on these coordinates

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Sobel Operators

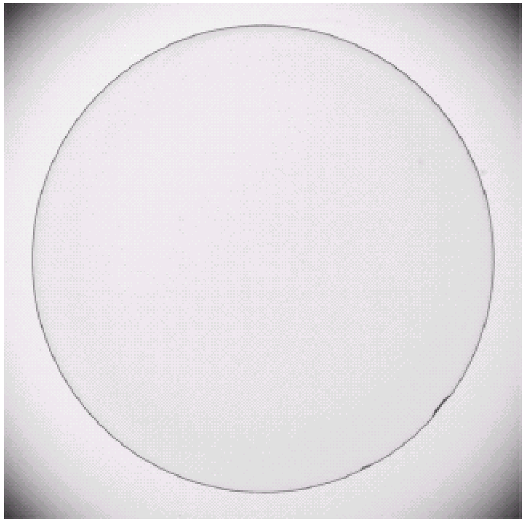
Based on the previous equations we can derive the *Sobel Operators*

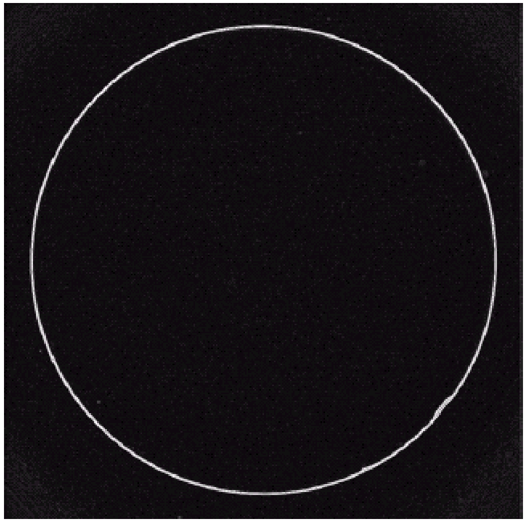
-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

To filter an image it is filtered using both operators the results of which are added together

Sobel Example





An image of a contact lens which is enhanced in order to make defects (at four and five o'clock in the image) more obvious

Sobel filters are typically used for edge detection

1st & 2nd Derivatives

Comparing the 1st and 2nd derivatives we can conclude the following:

- 1st order derivatives generally produce thicker edges
- 2nd order derivatives have a stronger response to fine detail e.g. thin lines
- 1st order derivatives have stronger response to grey level step
- 2nd order derivatives produce a double response at step changes in grey level

Combining Spatial Enhancement Methods

- Successful image enhancement is typically not achieved using a single operation
- Rather we combine a range of techniques in order to achieve a final result
- This example will focus on enhancing the bone scan to the right

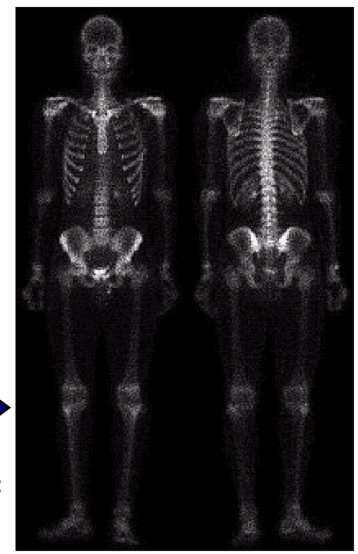
Combining Spatial Enhancement Methods (cont…)

(a)

Laplacian filter of bone scan (a)

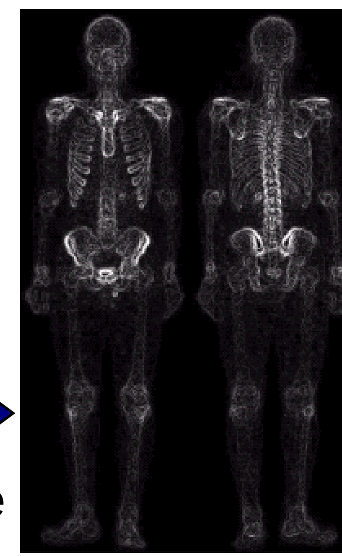
(b)

Sharpened version of bone scan achieved by subtracting (a) and (b)



(c)

Sobel filter of bone scan (a)



(d)

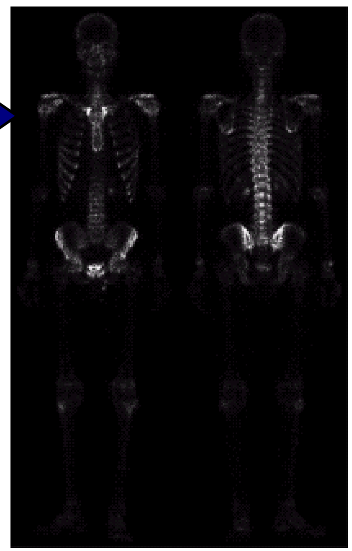
Combining Spatial Enhancement Methods (cont…)

(e)

Image (d) smoothed with a 5*5 averaging filter

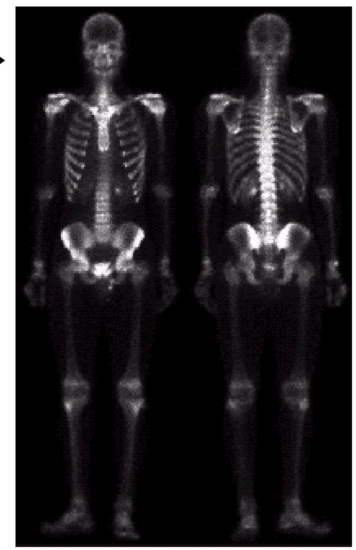
The product of (c) and (e) which will be used as a mask

(f)



Sharpened image which is sum of (a) and (f)

(g)



Result of applying a power-law trans. to (g)

(h)

Combining Spatial Enhancement Methods (cont…)

Compare the original and final images