# Spatial Filtering

李东晓

lidx@zju.edu.cn



浙江大学信息与电子工程学院
College of Information Science & Electronic Engineering, Zhejiang University

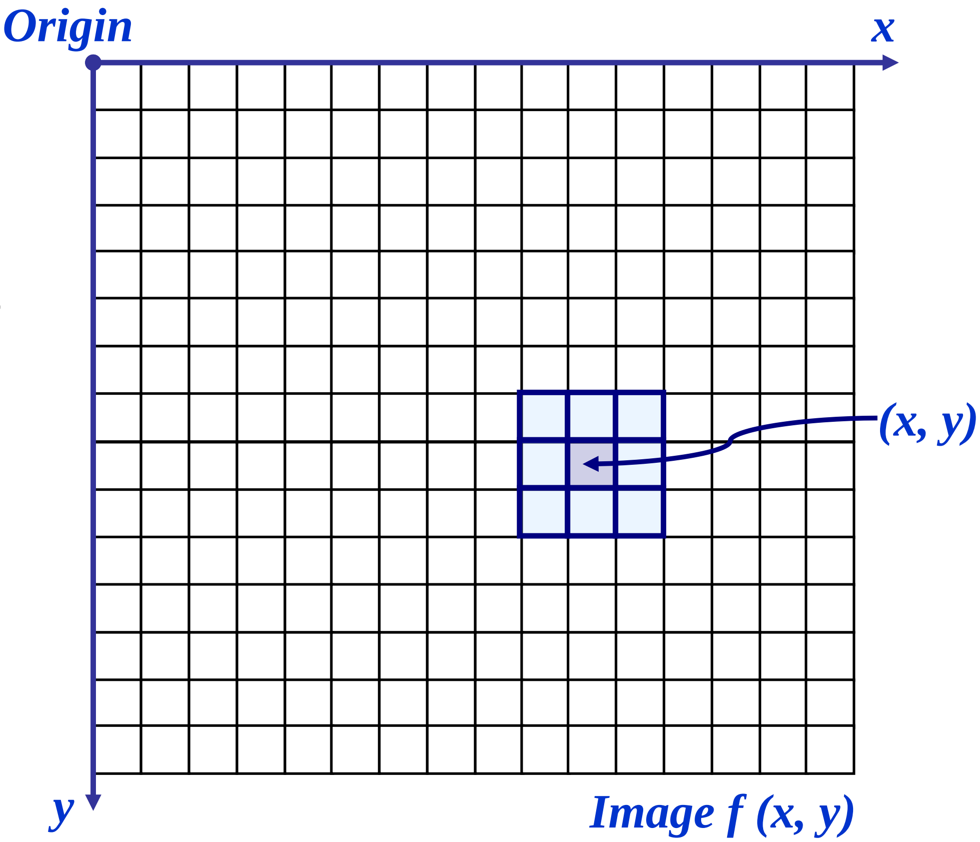
# Contents

# Basic Spatial Domain Image Enhancement

Most spatial domain enhancement operations can be reduced to the form

$g(x, y) = T[f(x, y)]$

where $f(x, y)$ is the input image, $g(x, y)$ is the processed image and $T$ is some operator defined over some neighbourhood of $(x, y)$

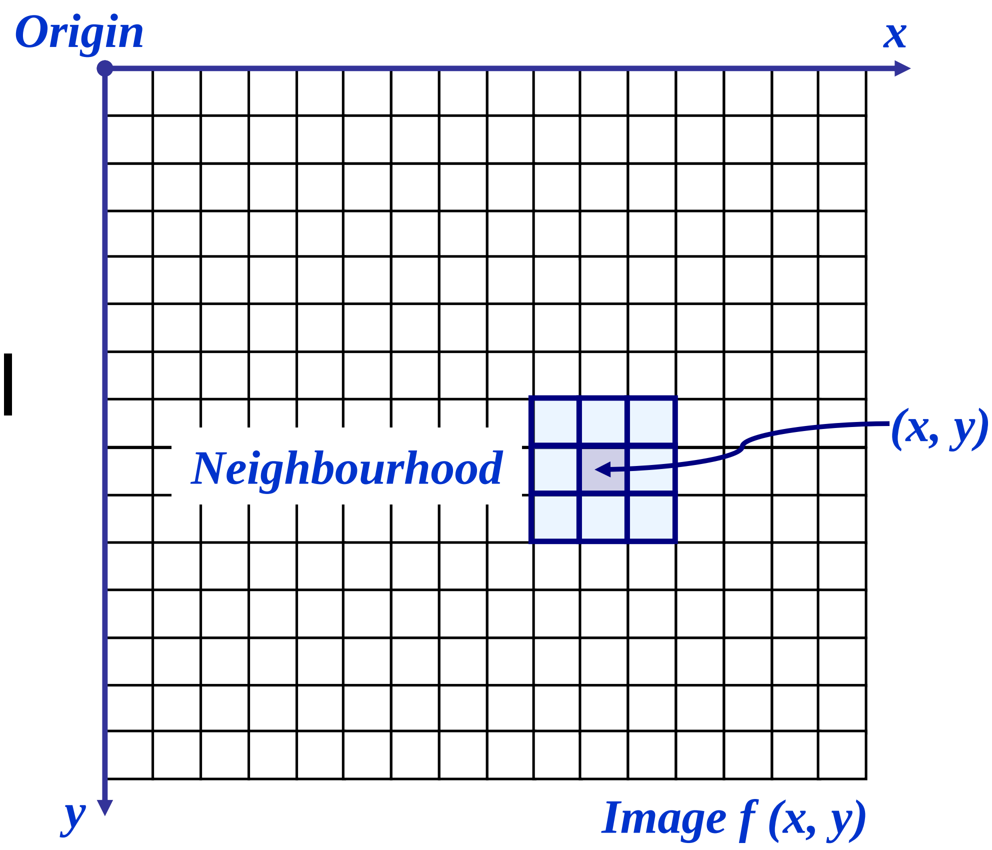


# Neighbourhood Operations

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



# Simple Neighbourhood Operations

Some simple neighbourhood operations include:

- **Min:** Set the pixel value to the minimum in the neighbourhood
- **Max:** Set the pixel value to the maximum in the neighbourhood
- **Median:** The median value of a set of numbers is the midpoint value in that set (e.g. from the set [1, 7, 15, 18, 24] 15 is the median). Sometimes the median works better than the average

# Simple Neighbourhood Operations

*Original Image*

x

| | | | | | |
|---|---|---|---|---|---|
| 123 | 127 | 128 | 119 | 115 | 130 |
| 140 | 145 | 148 | 153 | 167 | 172 |
| 133 | 154 | 183 | 192 | 194 | 191 |
| 194 | 199 | 207 | 210 | 198 | 195 |
| 164 | 170 | 175 | 162 | 173 | 151 |

y

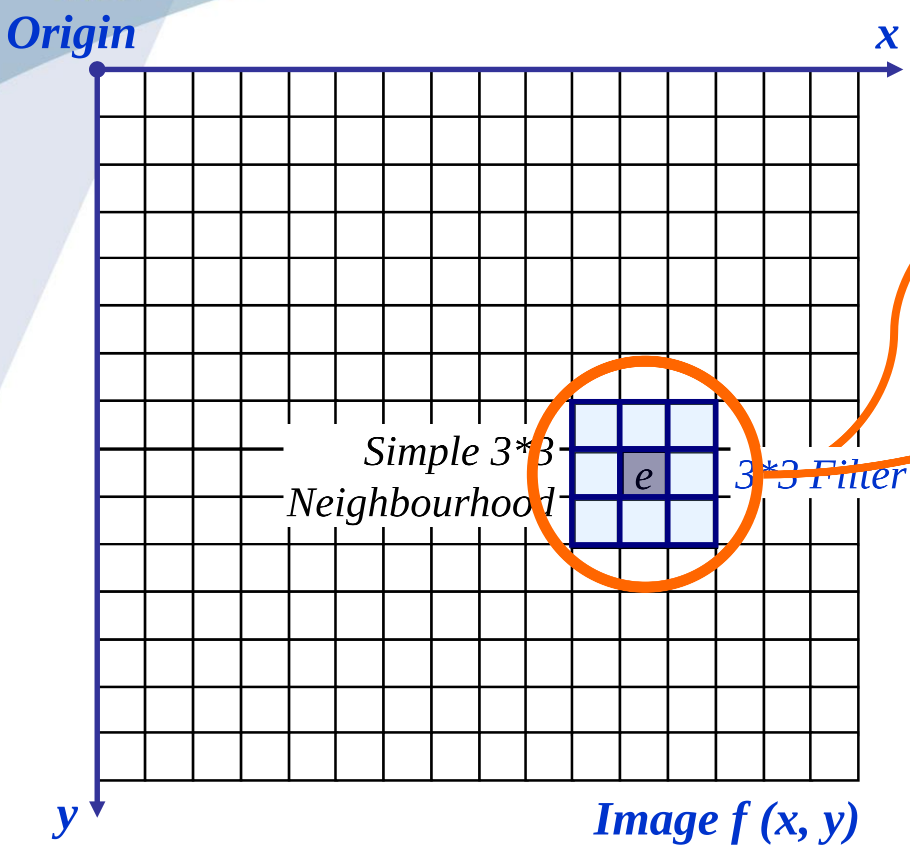
*Enhanced Image*

x

| | | | | | |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

y

# The Spatial Filtering Process




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

**Original Image Pixels**

*

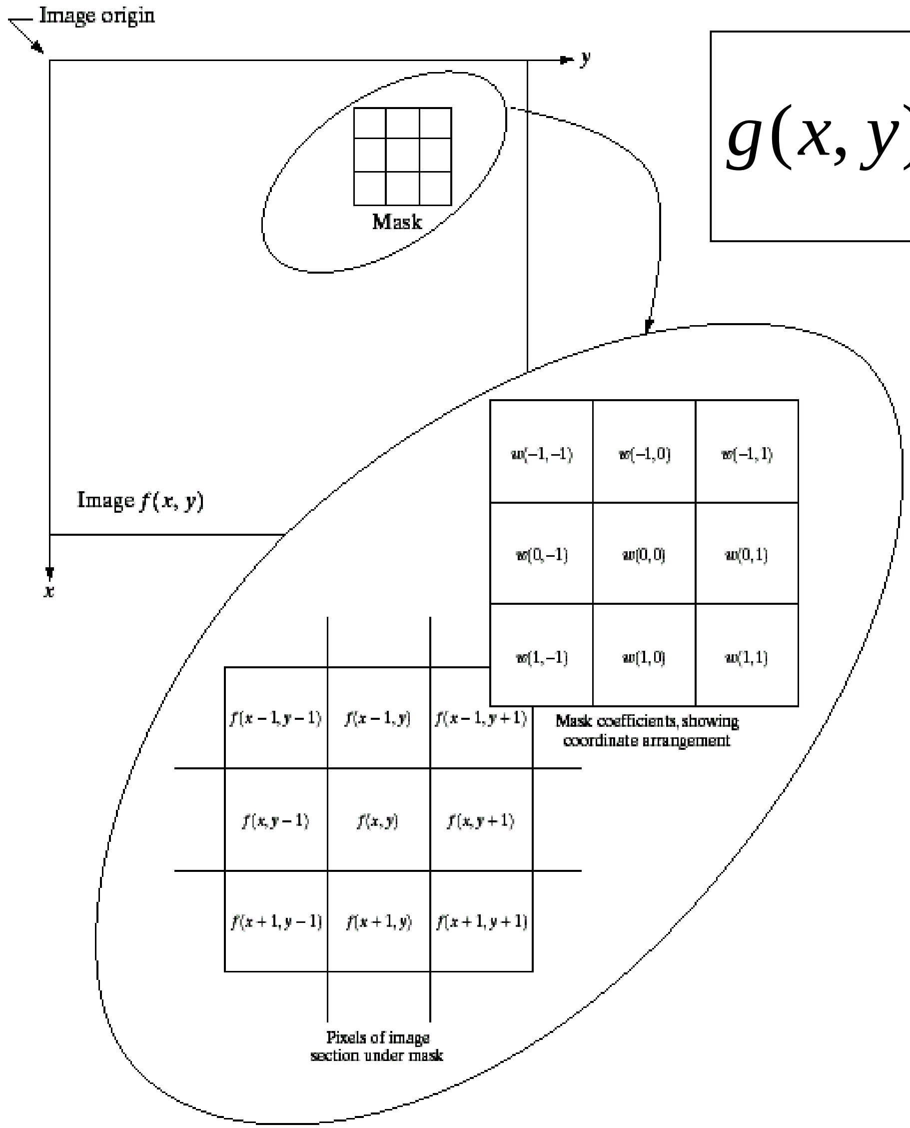
| | | |
|---|---|---|
| *r* | *s* | *t* |
| *u* | *v* | *w* |
| *x* | *y* | *z* |

**Filter**

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

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

# Spatial Filtering: Equation Form




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

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

# Smoothing Spatial Filters

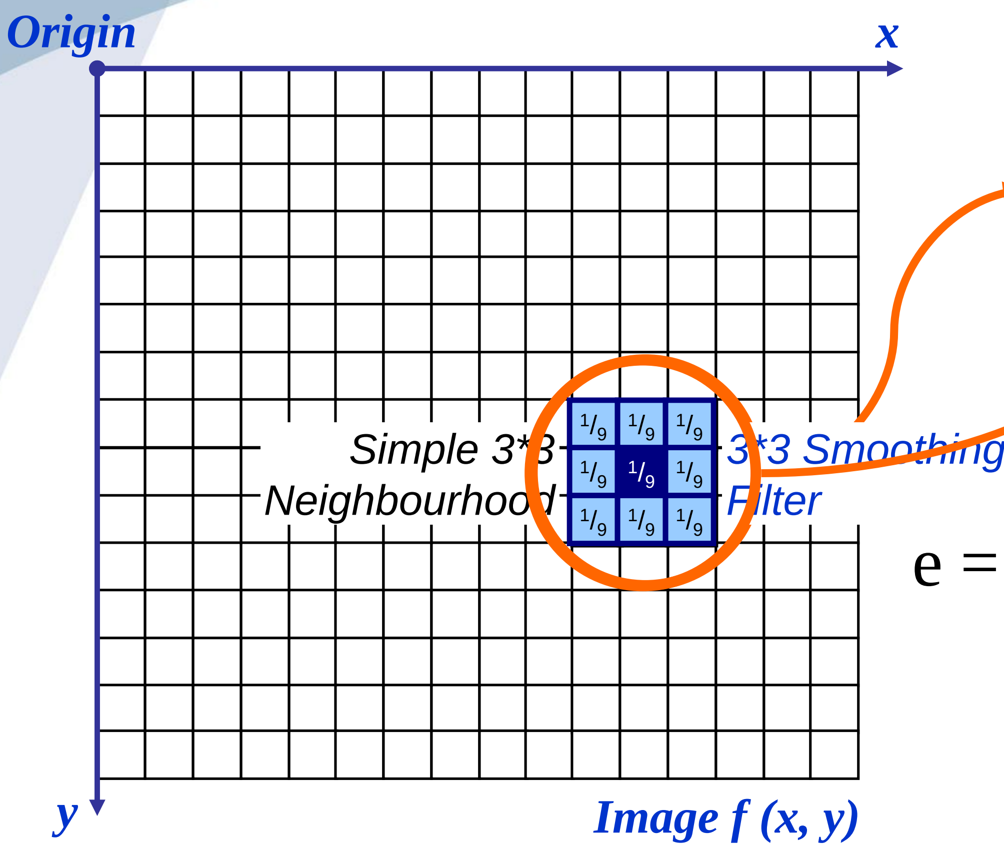
One of the simplest spatial filtering operations we can perform is a smoothing operation

- 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

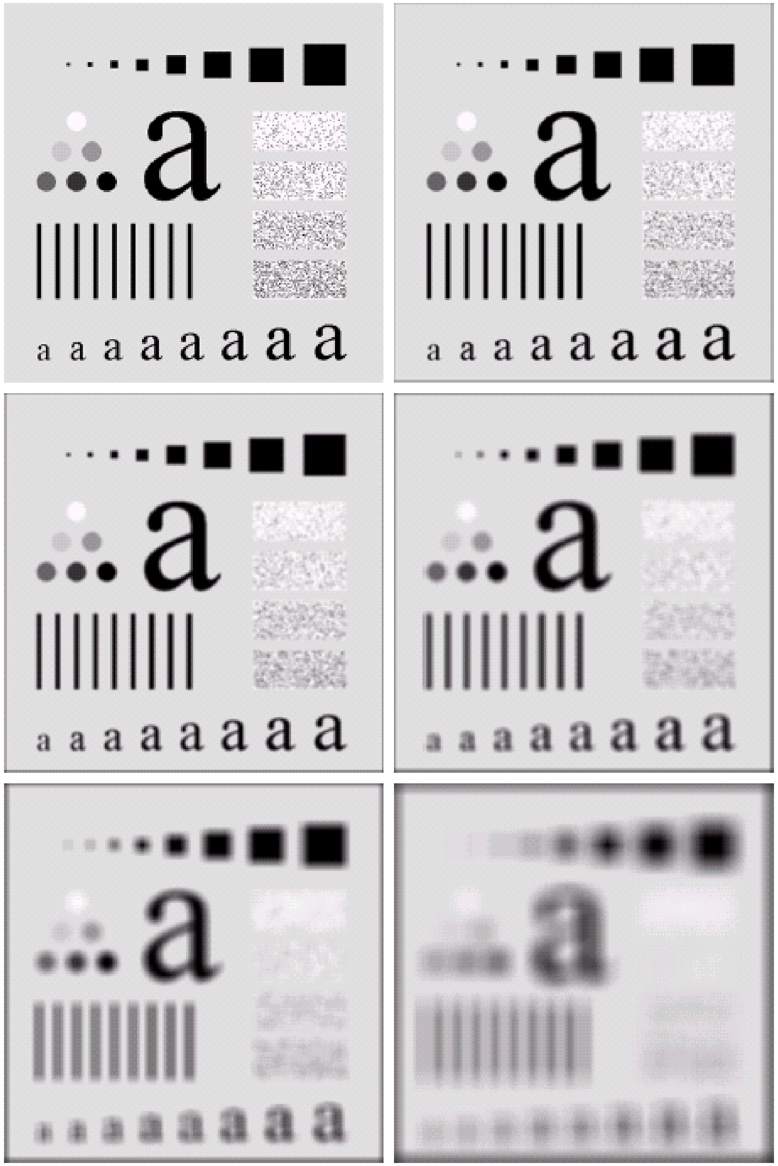
# Image Smoothing Example

The image at the top left is an original image of size 500*500 pixels

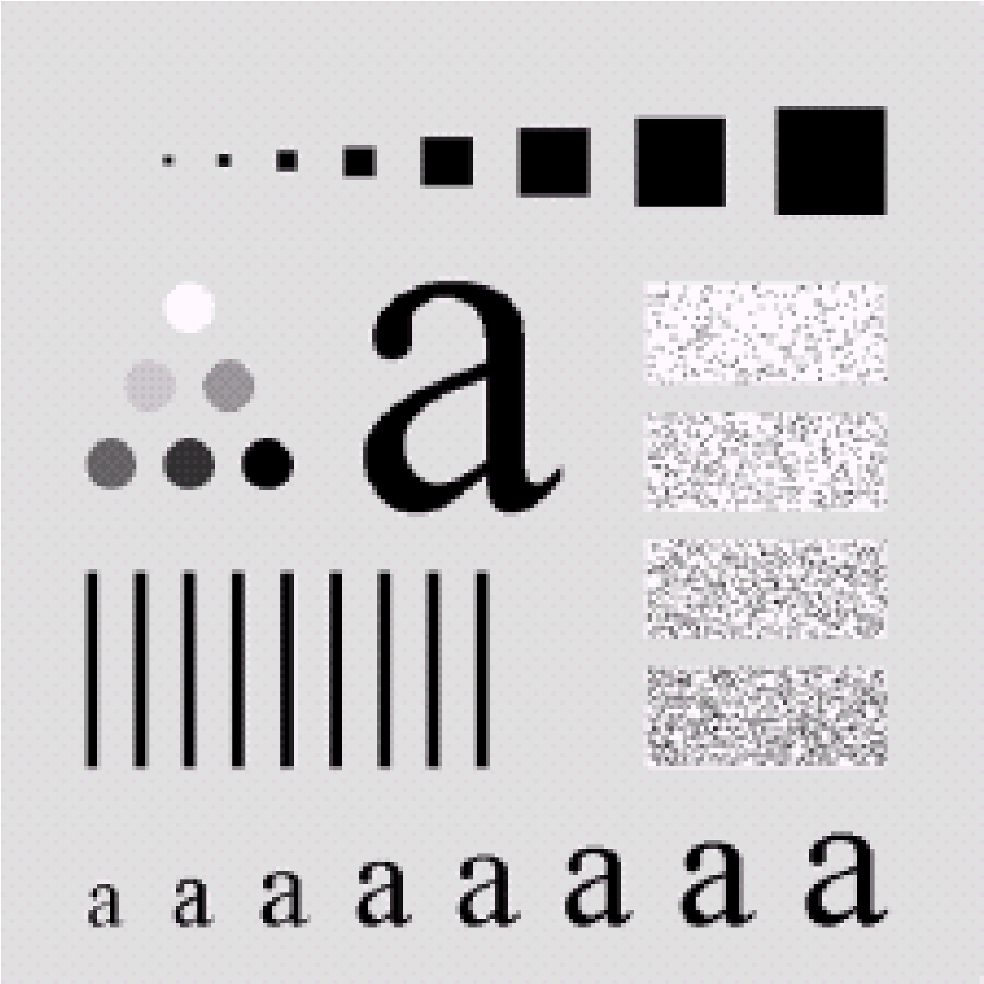
The subsequent images show the image after filtering with an averaging filter of increasing sizes：

3, 5, 9, 15 and 35

Notice how detail begins to disappear

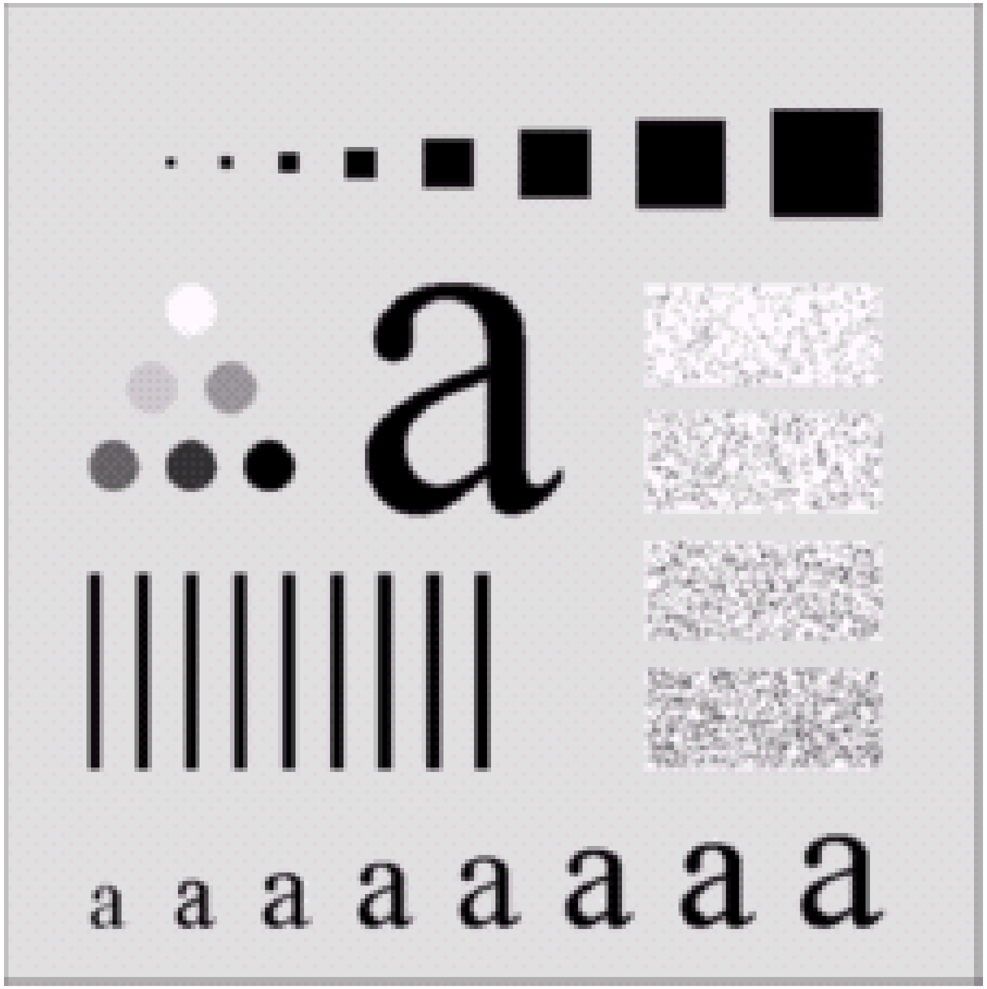


# Image Smoothing Example



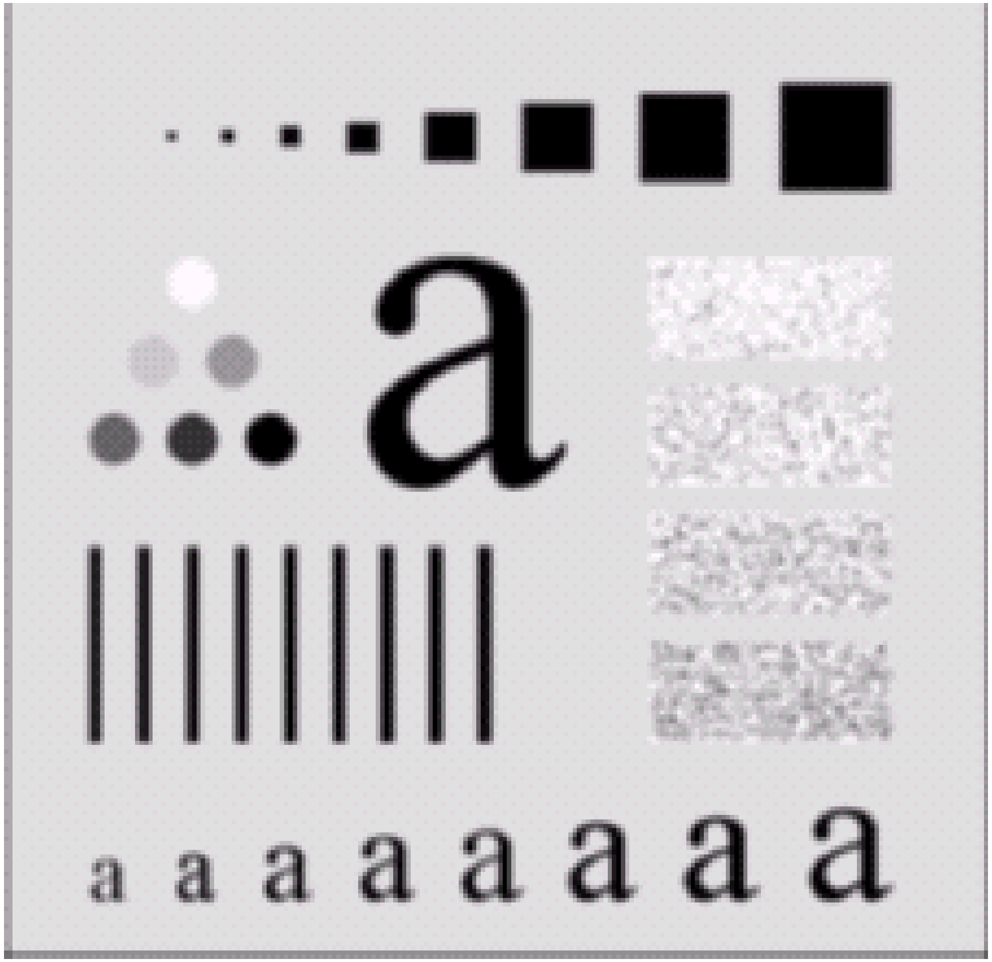
Original

# Image Smoothing Example



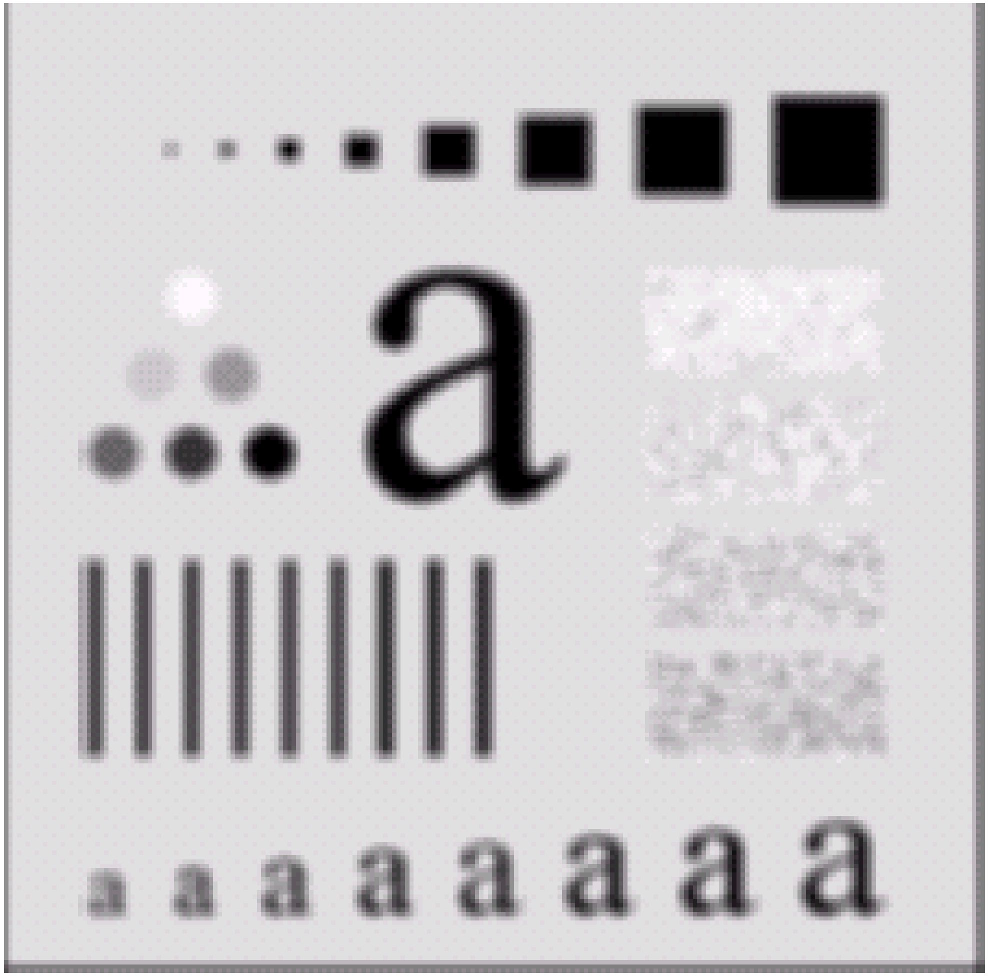
3x3 Filter

# Image Smoothing Example



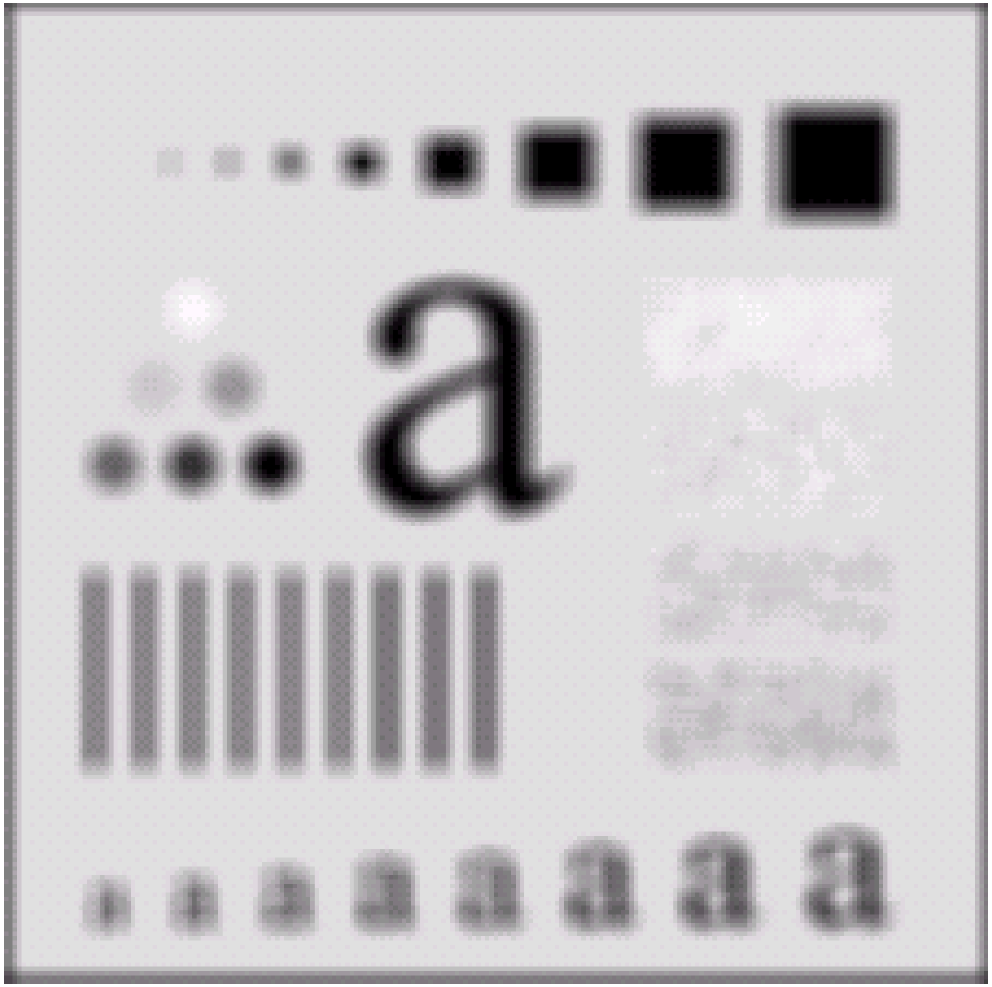
5x5 Filter

# Image Smoothing Example



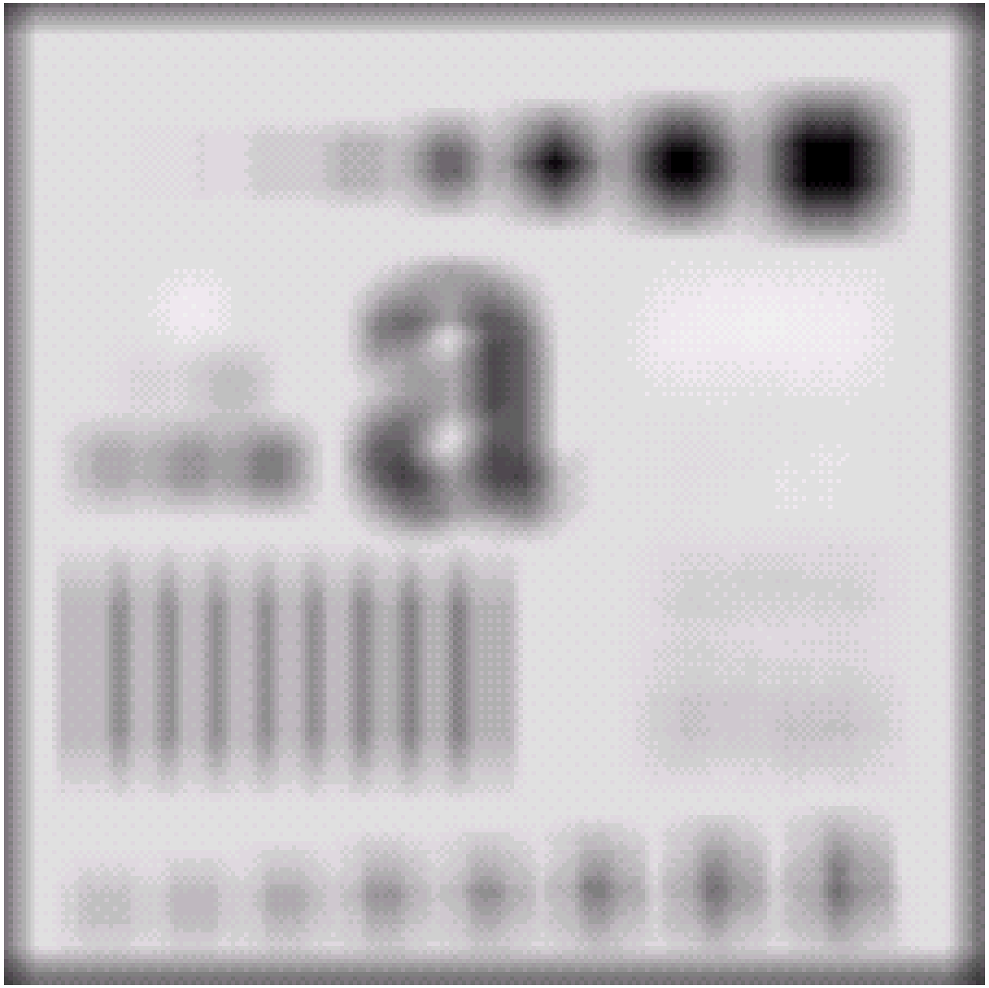
9x9 Filter

# Image Smoothing Example



15x15
Filter

# Image Smoothing Example



35x35
Filter

# Weighted Smoothing Filters

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

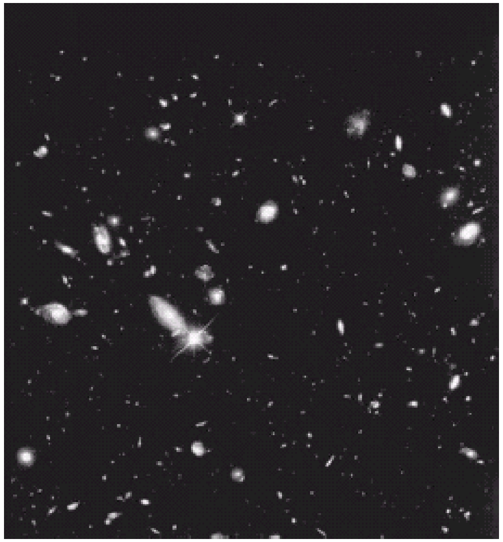
- Pixels closer to the central pixel are more important
- Often referred to as a *weighted averaging*

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

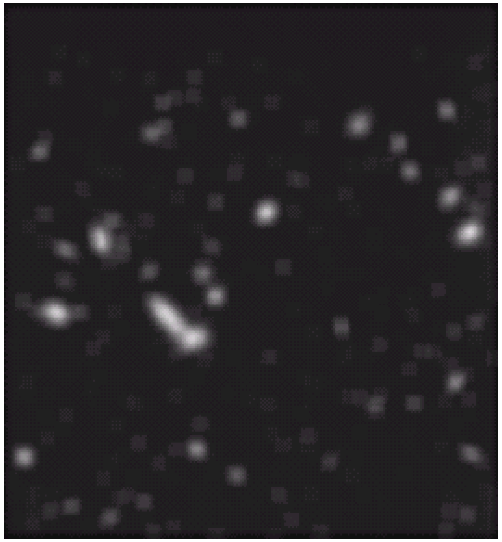
Weighted averaging filter

# Another Smoothing Example

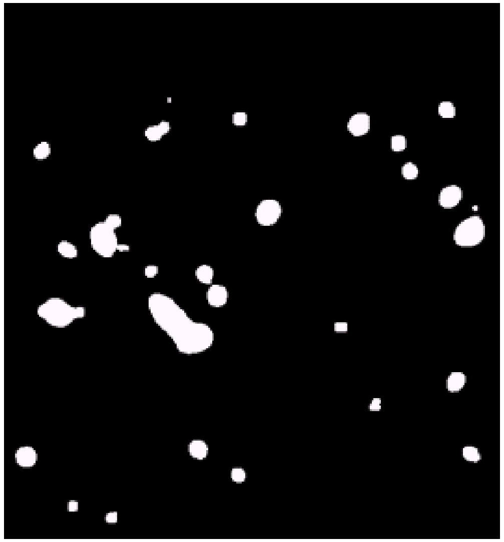
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding



Original Image

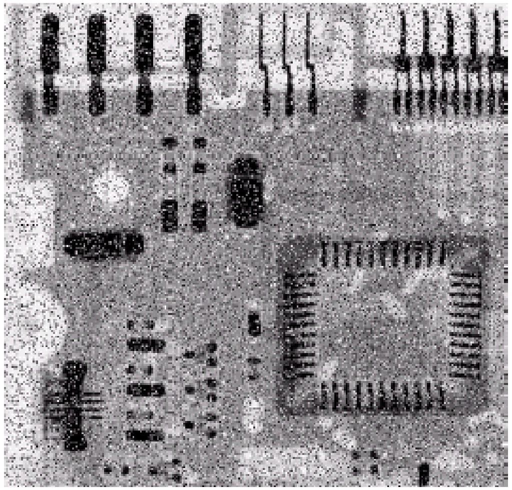


Smoothed Image



Thresholded Image

# Averaging Filter Vs. Median Filter



Original Image With Noise

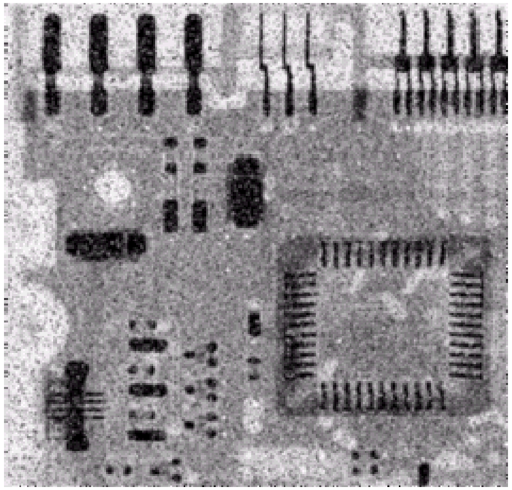


Image After Averaging Filter

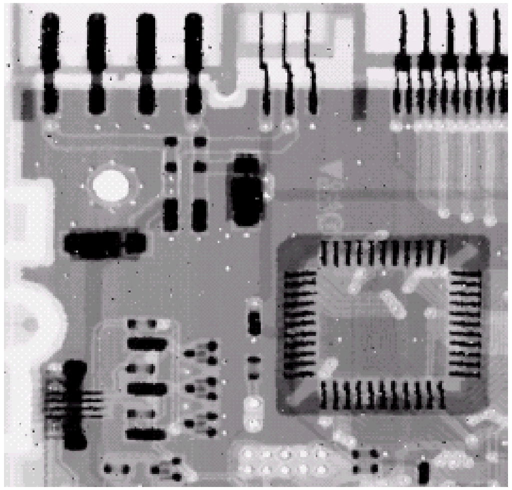
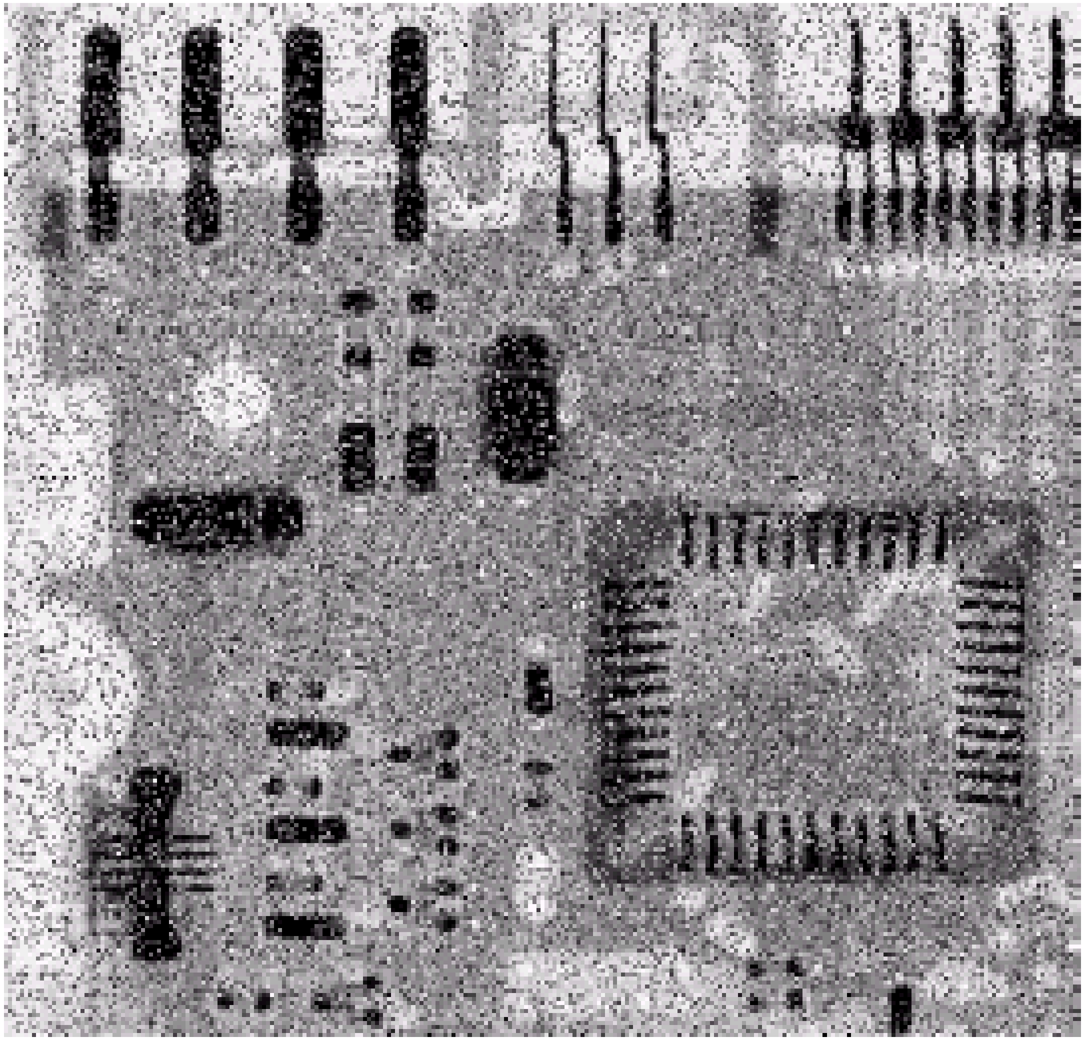


Image After Median Filter

Filtering is often used to remove noise from images

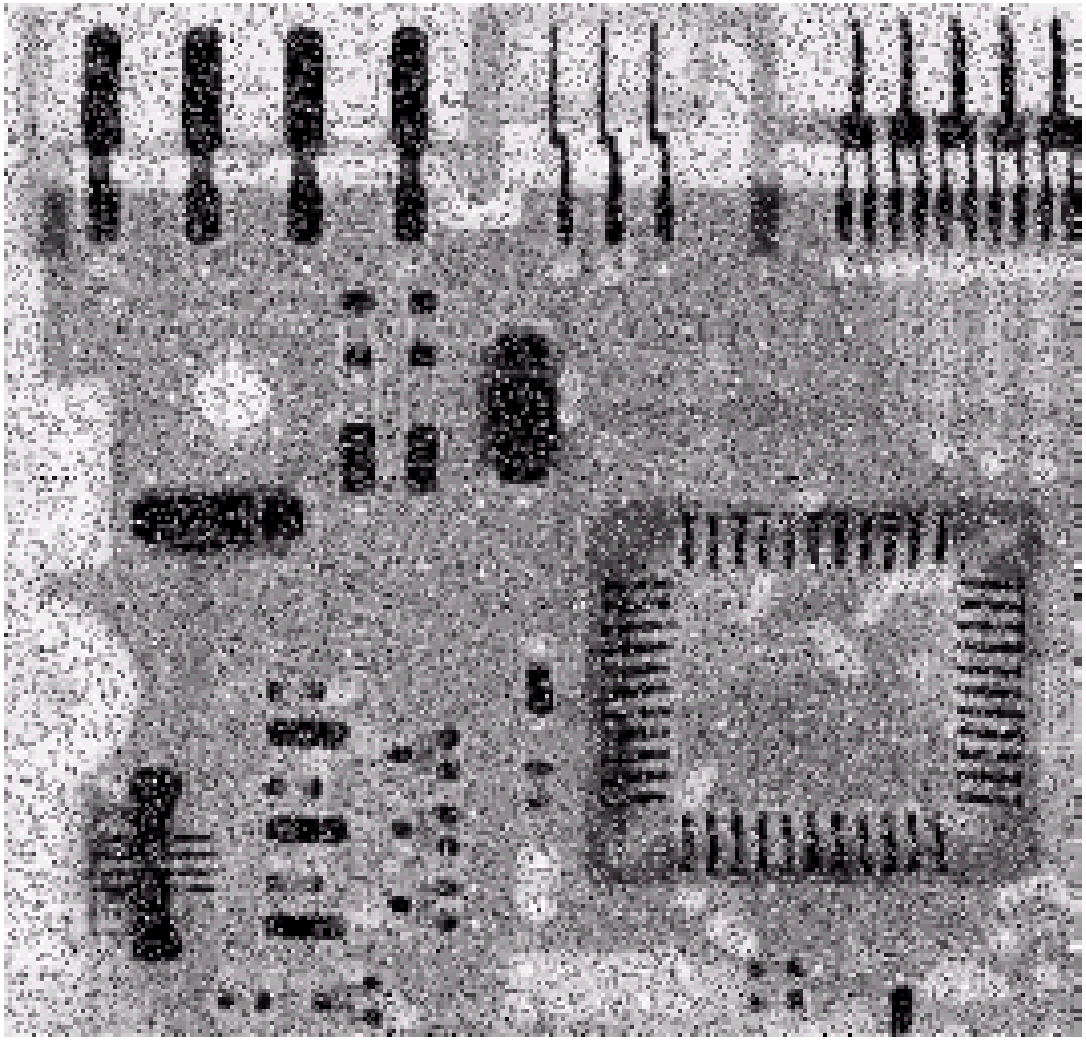
Sometimes a median filter works better than an averaging filter

# Averaging Filter Vs. Median Filter



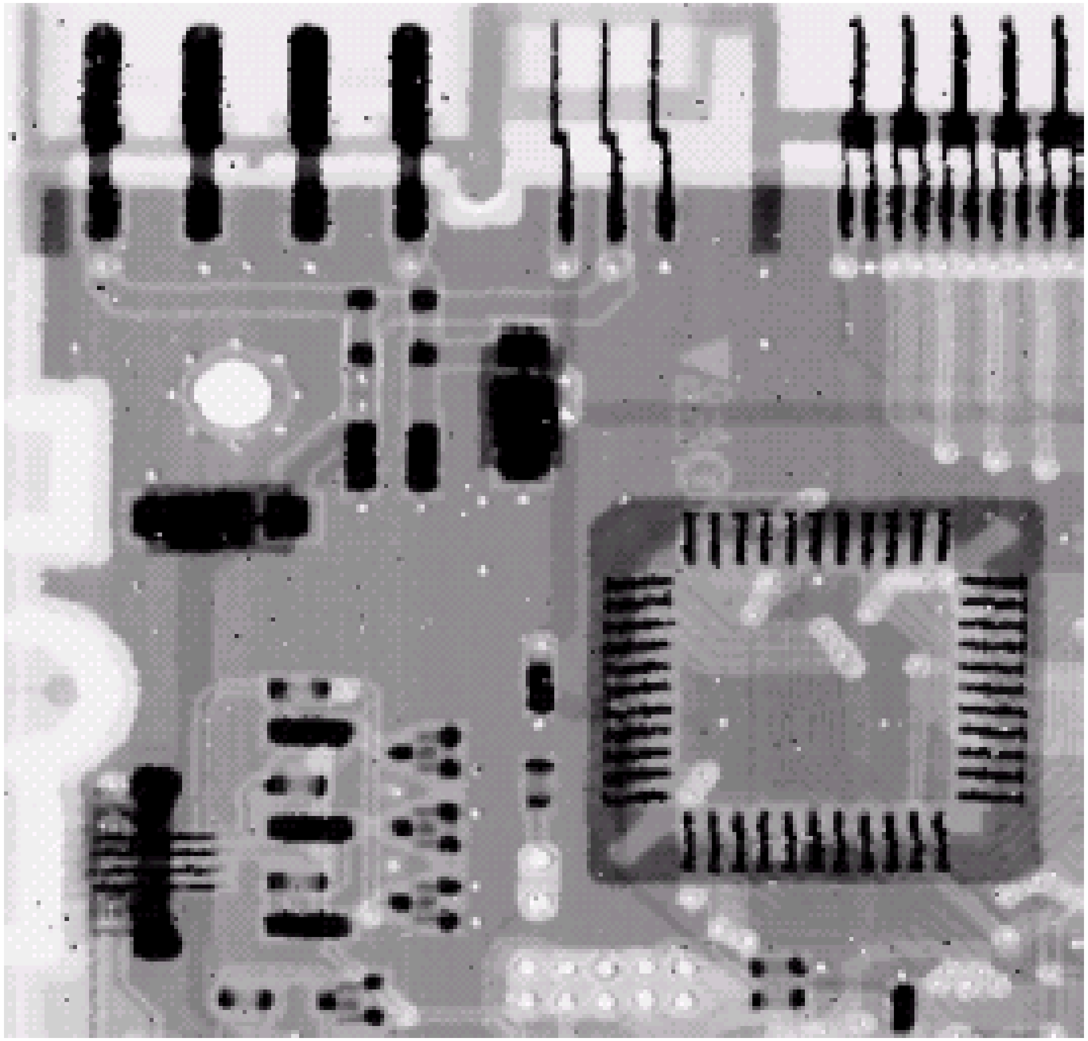
Original

# Averaging Filter Vs. Median Filter



Averaging Filter

# Averaging Filter Vs. Median Filter



Median Filter

# Simple Neighbourhood Operations

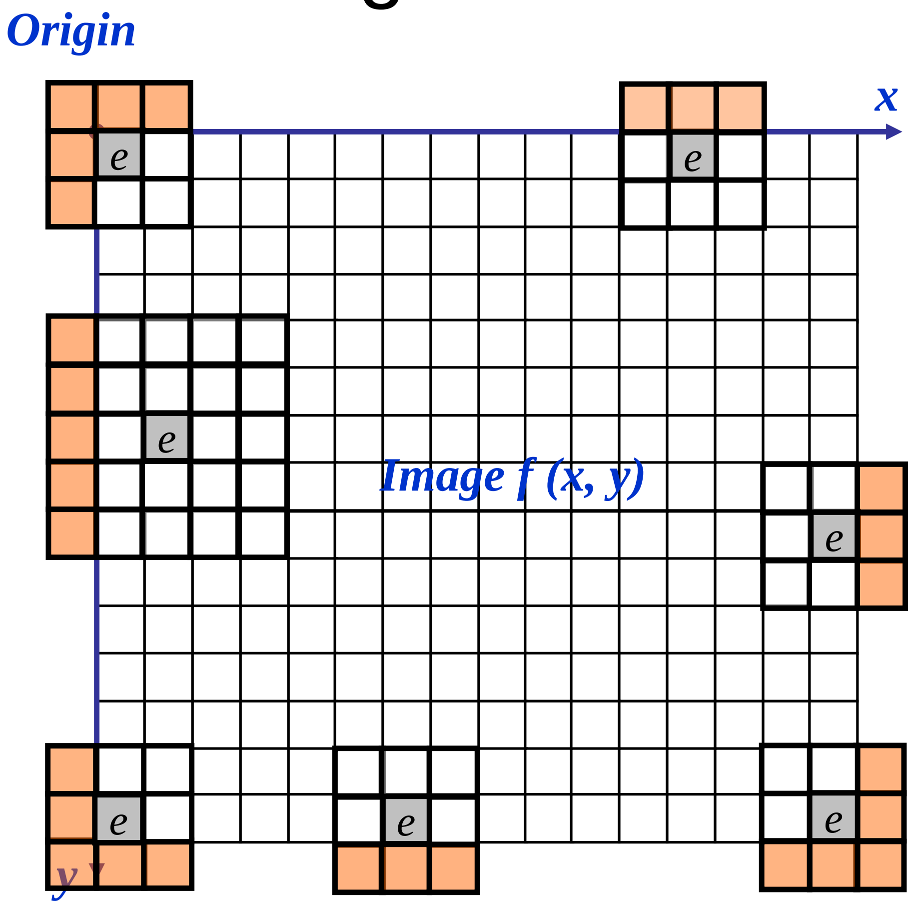
x

| | | | | | |
|---|---|---|---|---|---|
| 123 | 127 | 128 | 119 | 115 | 130 |
| 140 | 145 | 148 | 153 | 167 | 172 |
| 133 | 154 | 183 | 192 | 194 | 191 |
| 194 | 199 | 207 | 210 | 198 | 195 |
| 164 | 170 | 175 | 162 | 173 | 151 |

y

# Strange Things Happen at The Borders!

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

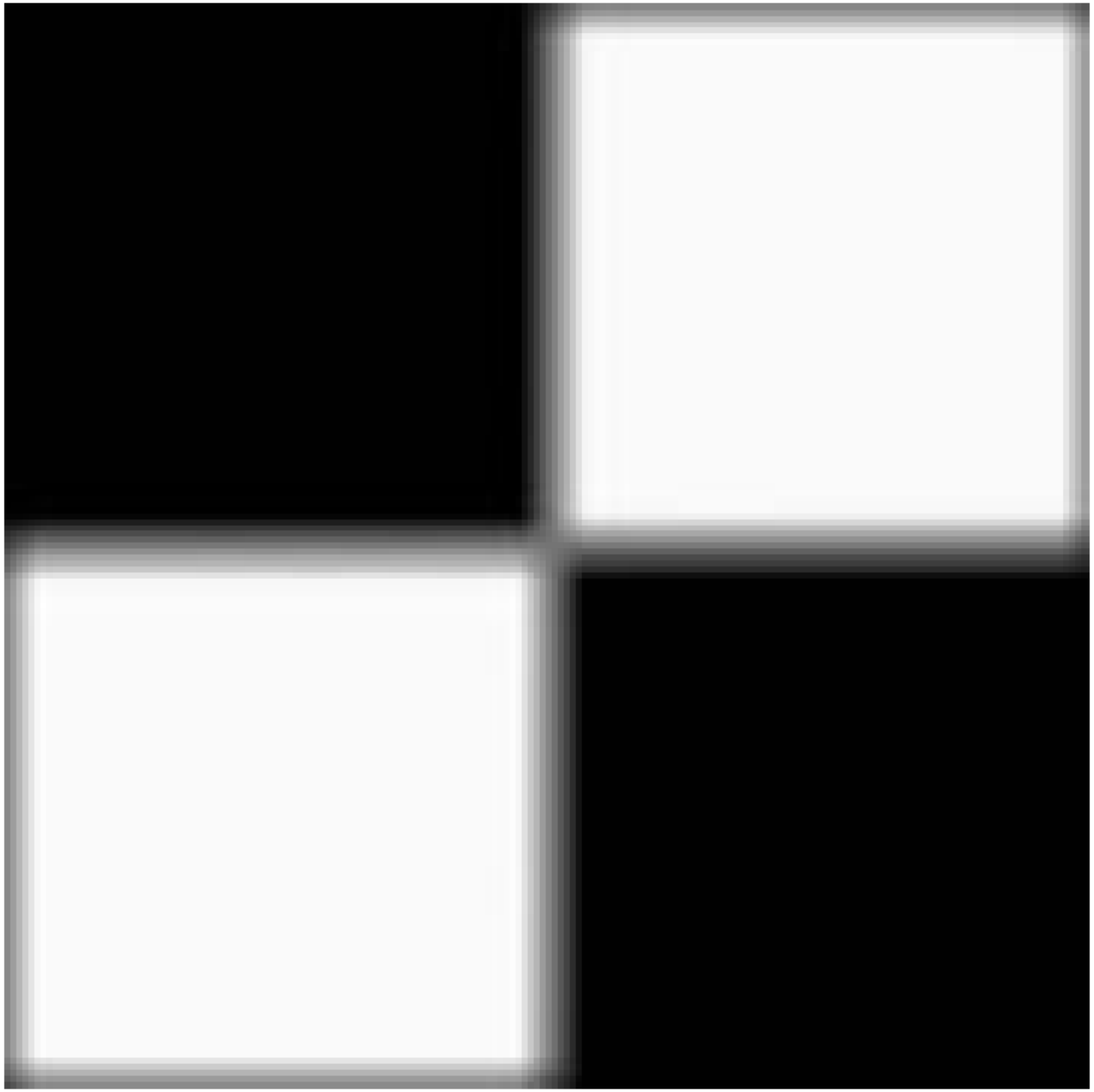


# Strange Things Happen at The Borders!

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

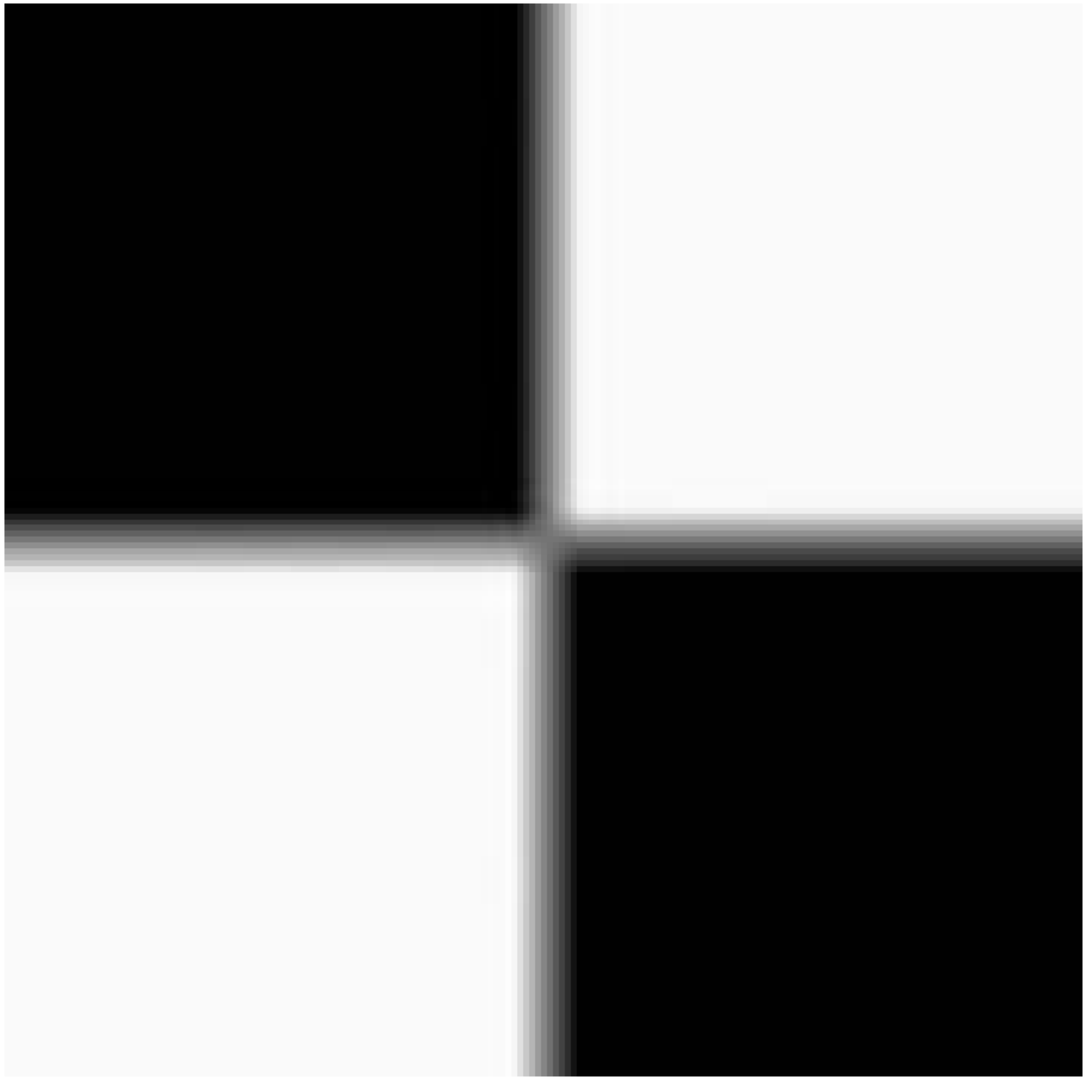
- Omit missing pixels
  - Only works with some filters
  - Can add extra code and slow down processing
- padarray(A, padsize, padval, direction)
  - padsize
  - padval: 0 (default), ‘circular’, ‘replicate’, ‘symmetric’
  - direction: ‘both’(default), ‘post’, ‘pre’

# Strange Things Happen at The Borders!



Zero Padding

# Strange Things Happen at The Borders!

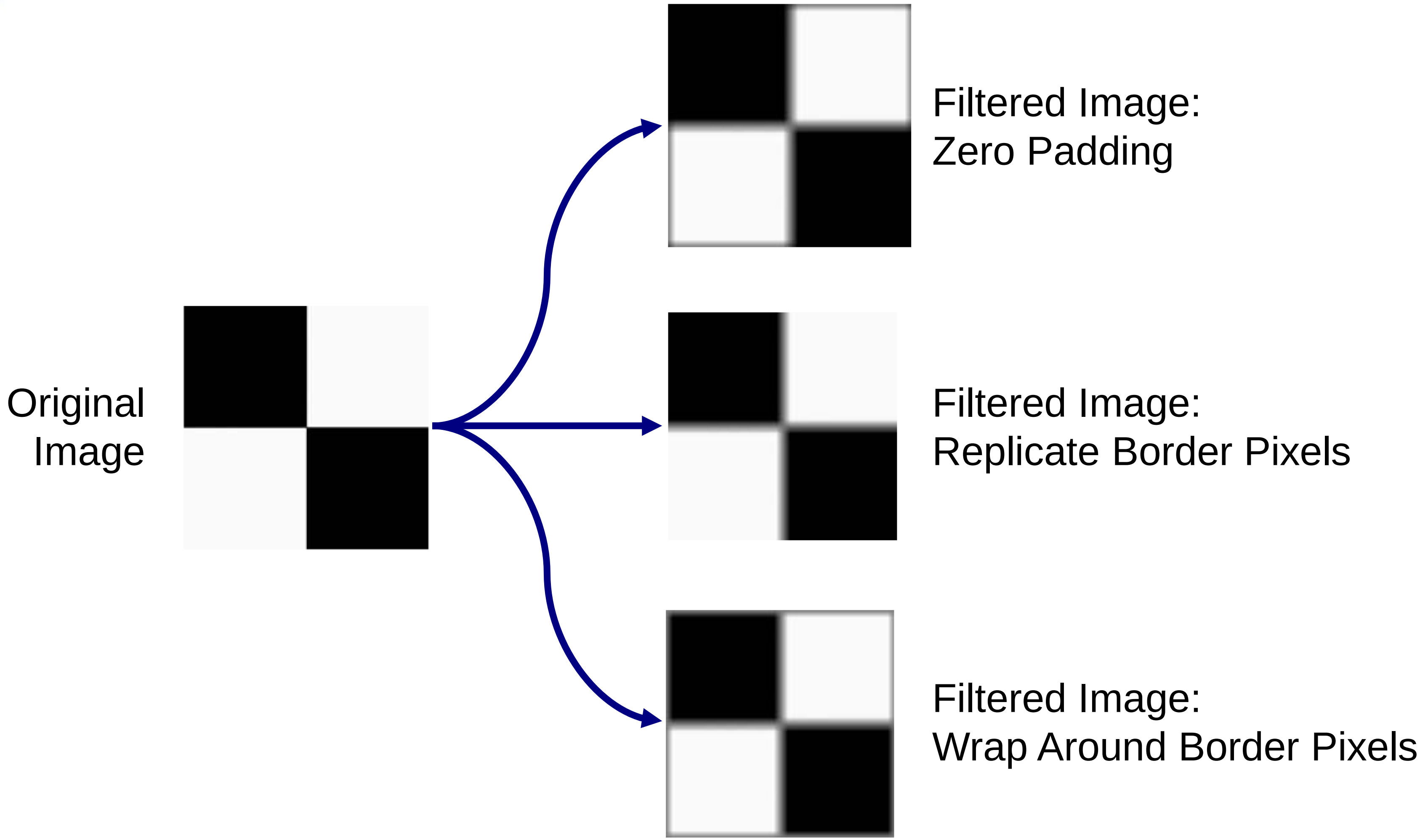


Replicate Border Pixels

# Strange Things Happen at The Borders!

Wrap Around Border Pixels

# Strange Things Happen at The Borders!



# 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

# Correlation & Convolution

- Correlation

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

- Convolution

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

For symmetric filters it makes no difference

# 3.6 Sharpening Spatial Filtering

- Sharpening filters
  - 1st derivative filters
  - 2nd derivative filters

# Sharpening Spatial Filters

Previously we have looked at smoothing filters which remove fine detail

*Sharpening spatial filters* seek to highlight fine detail

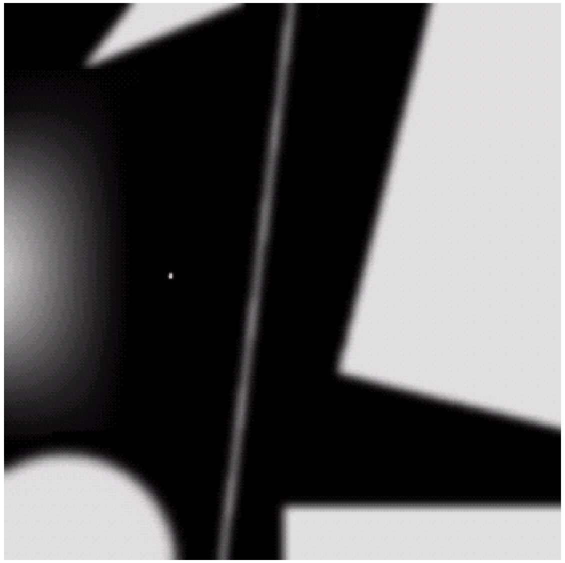
- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

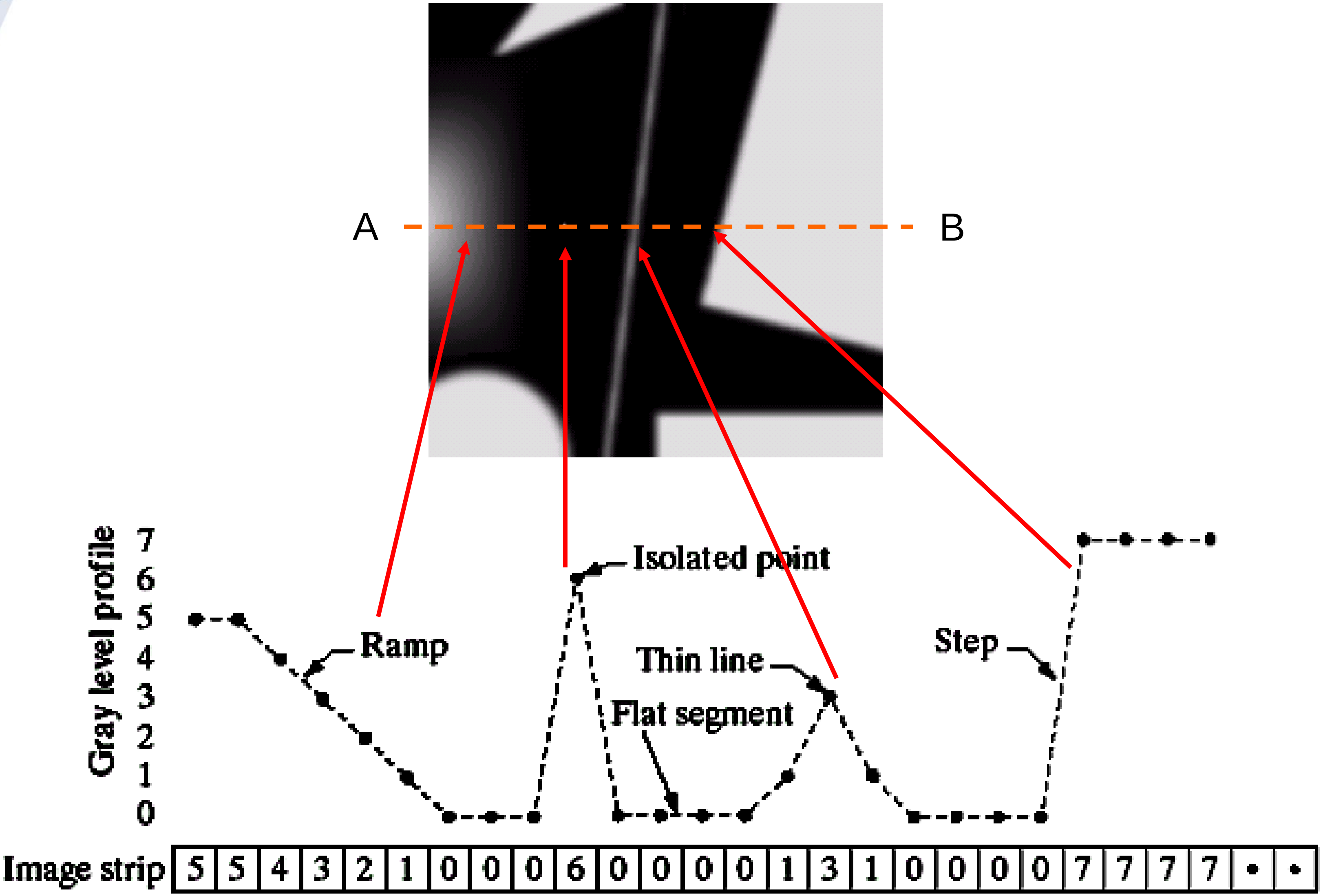
# Spatial Differentiation

Differentiation measures the *rate of change* of a function

Let's consider a simple 1 dimensional example



# Spatial Differentiation



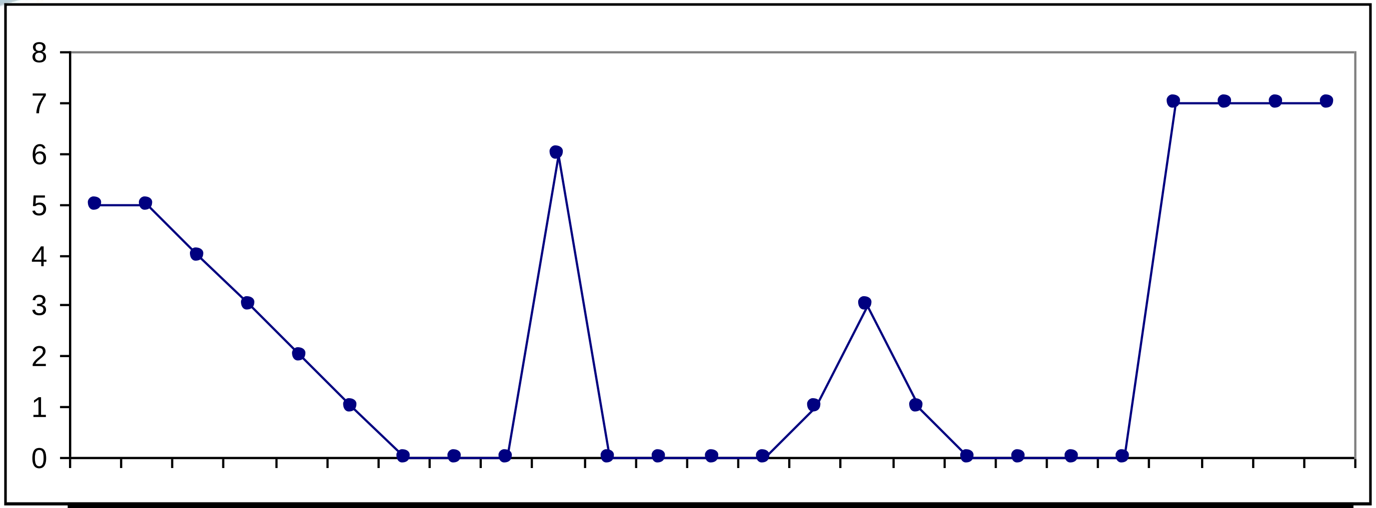
# 1st Derivative

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

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

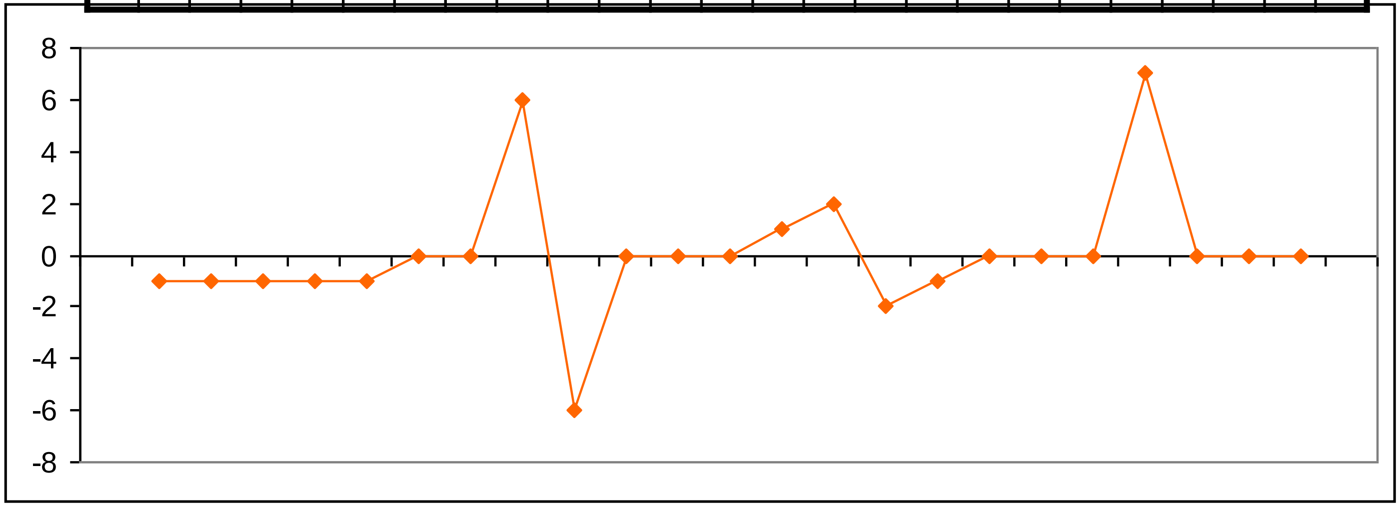
It's just the difference between subsequent values and measures the rate of change of the function

# 1st Derivative (cont…)




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



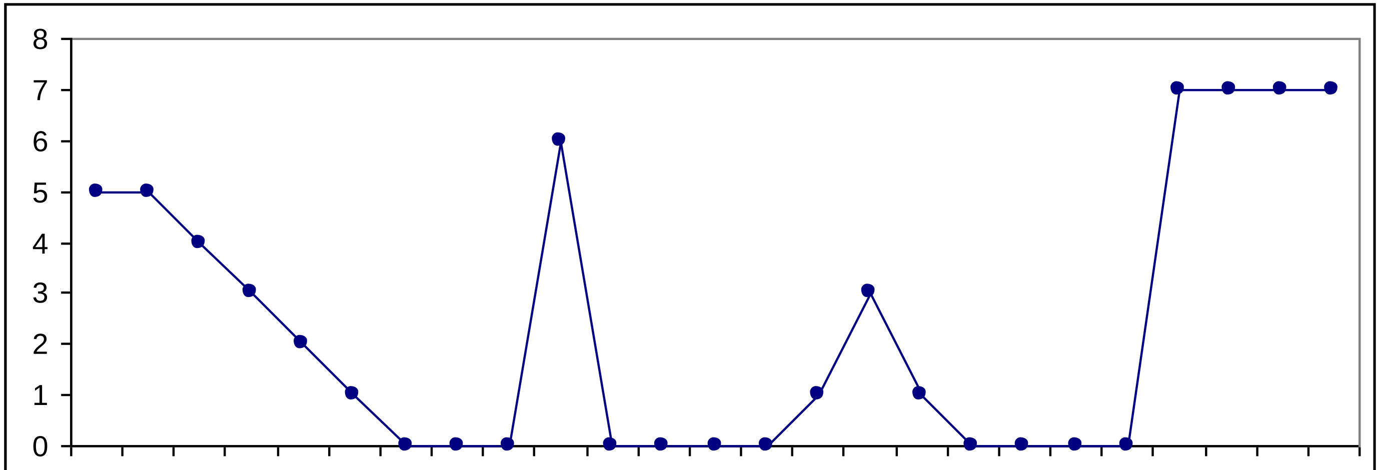
# $2^{nd}$ Derivative

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

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

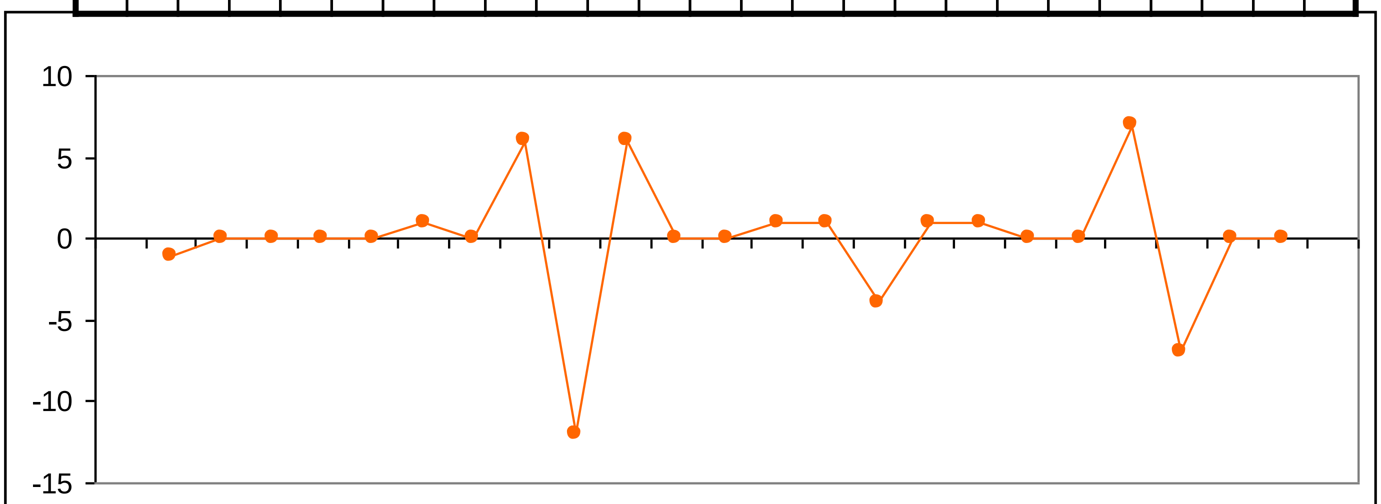
Simply takes into account the values both before and after the current value

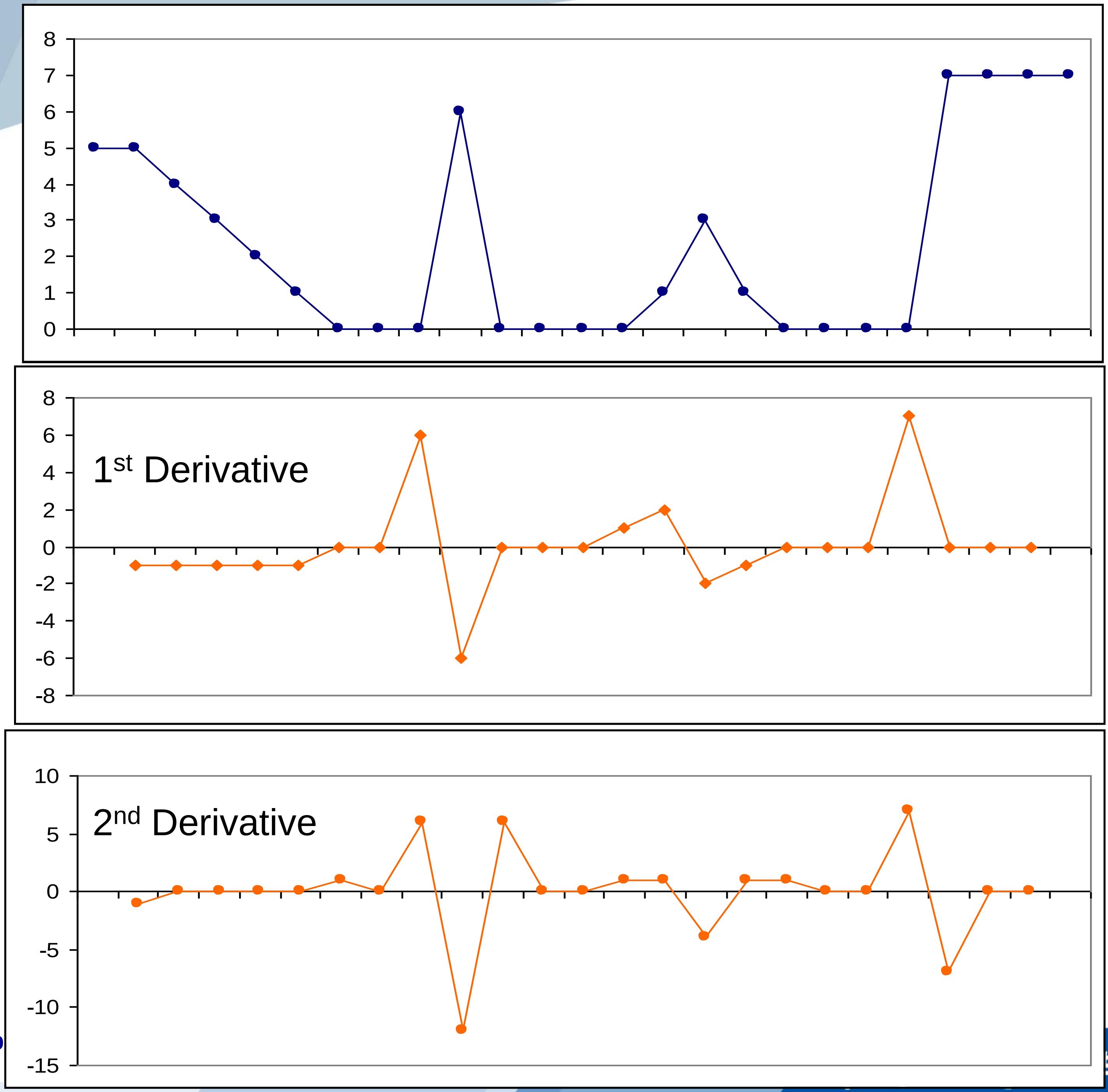
# 2nd Derivative (cont…)




| | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | |






L04 Sp

# Using 2nd Derivatives For Image Enhancement

The 2$^{nd}$ derivative is more useful for image enhancement than the 1$^{st}$ derivative

- Stronger response to fine detail
- Simpler implementation
- We will come back to the 1$^{st}$ 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 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 (cont…)

So, the Laplacian can be given as follows:

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

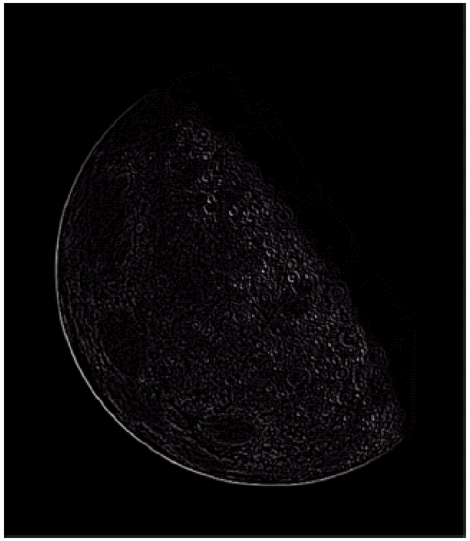
We can easily build a filter based on this

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

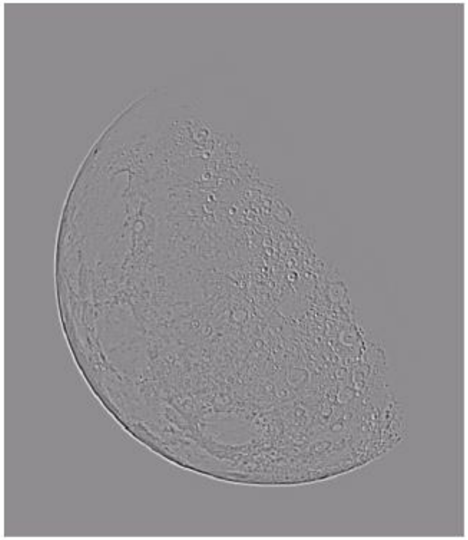
# The Laplacian (cont…)

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

Original Image



Laplacian Filtered Image

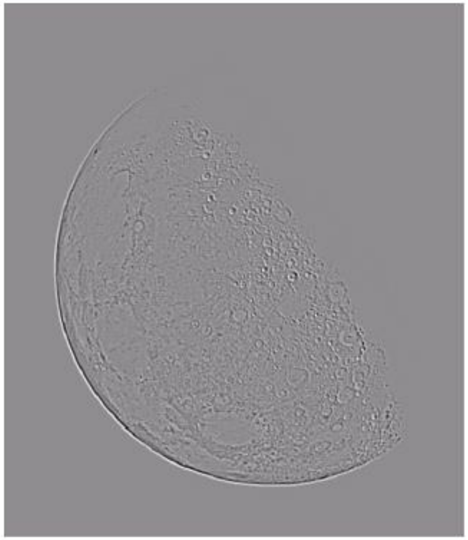


Laplacian Filtered Image Scaled for Display

# But that is Not Very Enhanced!

The result of a Laplacian filtering is NOT an enhanced image

Subtract the Laplacian result from the original image to generate our final sharpened enhanced image



Laplacian
Filtered Image
Scaled for Display

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

# 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

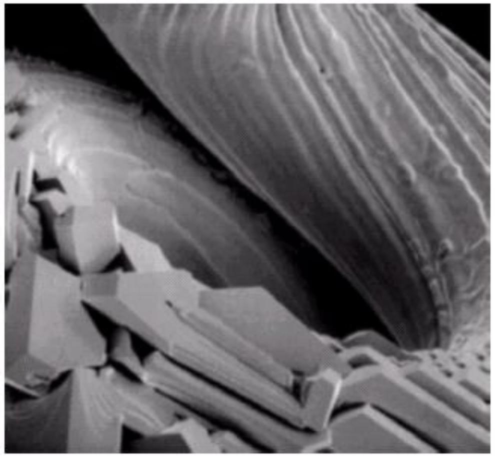
# Simplified Image Enhancement

The entire enhancement can be combined into a single filtering operation

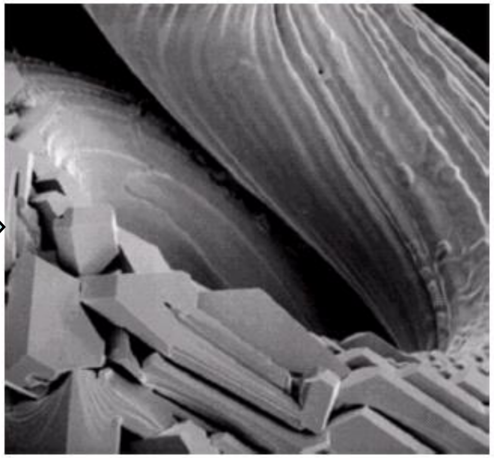
$$
\begin{aligned}
g(x,y) &= f(x,y) - \nabla^2 f \\
&= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\qquad + f(x,y+1) + f(x,y-1) \\
&\qquad - 4f(x,y)] \\
&= 5f(x,y) - f(x+1,y) - f(x-1,y) \\
&\qquad - f(x,y+1) - f(x,y-1)
\end{aligned}
$$

# Simplified Image Enhancement (cont…)

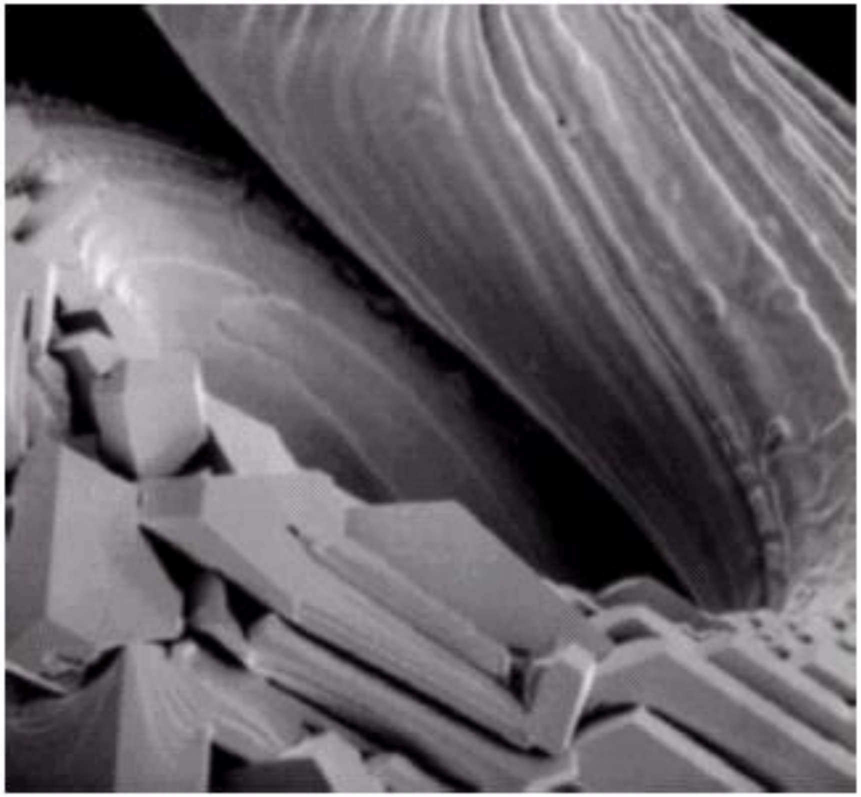
This gives us a new filter which does the whole job for us in one step

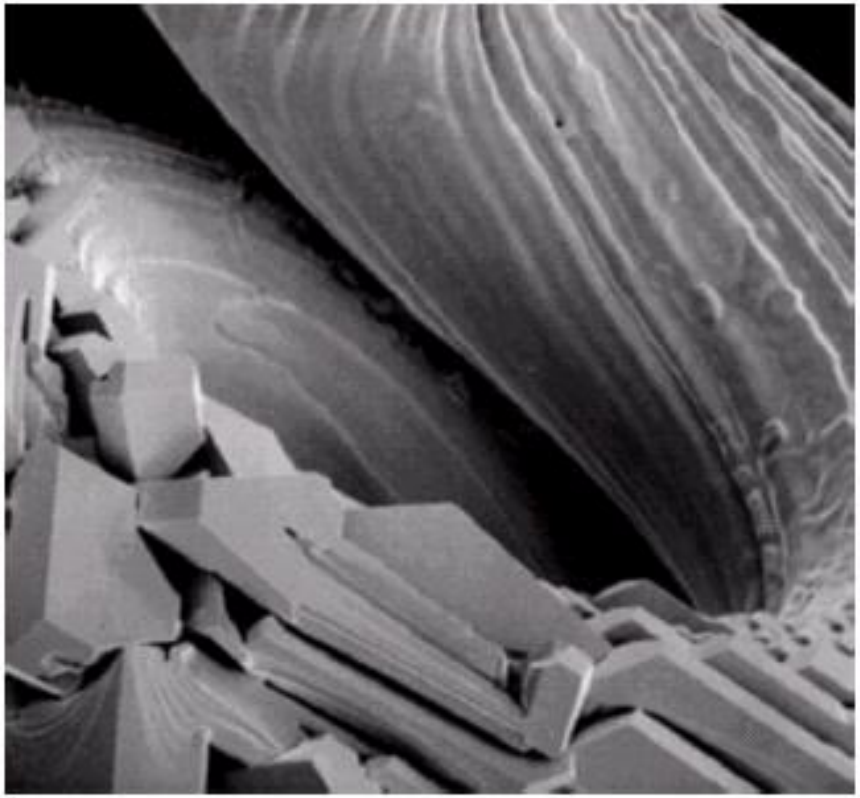


| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |



# Simplified Image Enhancement (cont…)





# Variants On The Simple Laplacian

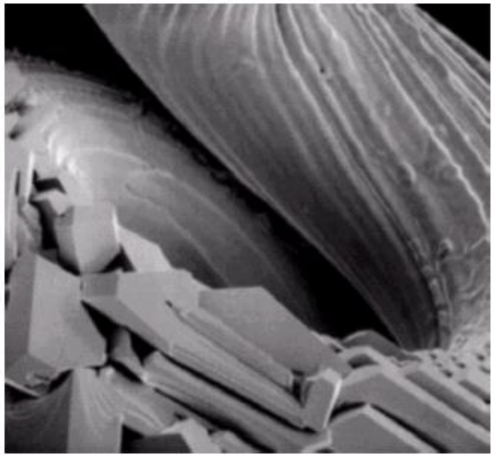
There are lots of slightly different versions of the Laplacian that can be used:

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

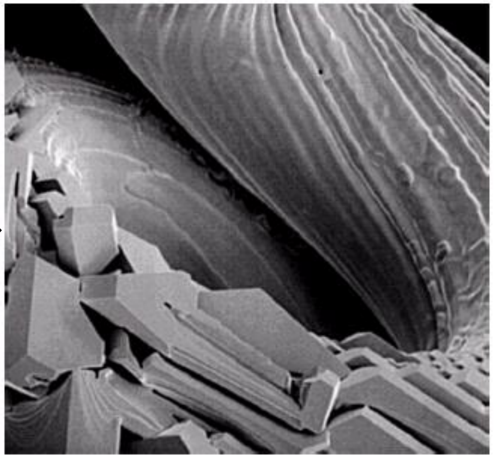
Simple Laplacian

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

Variant of Laplacian



| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |



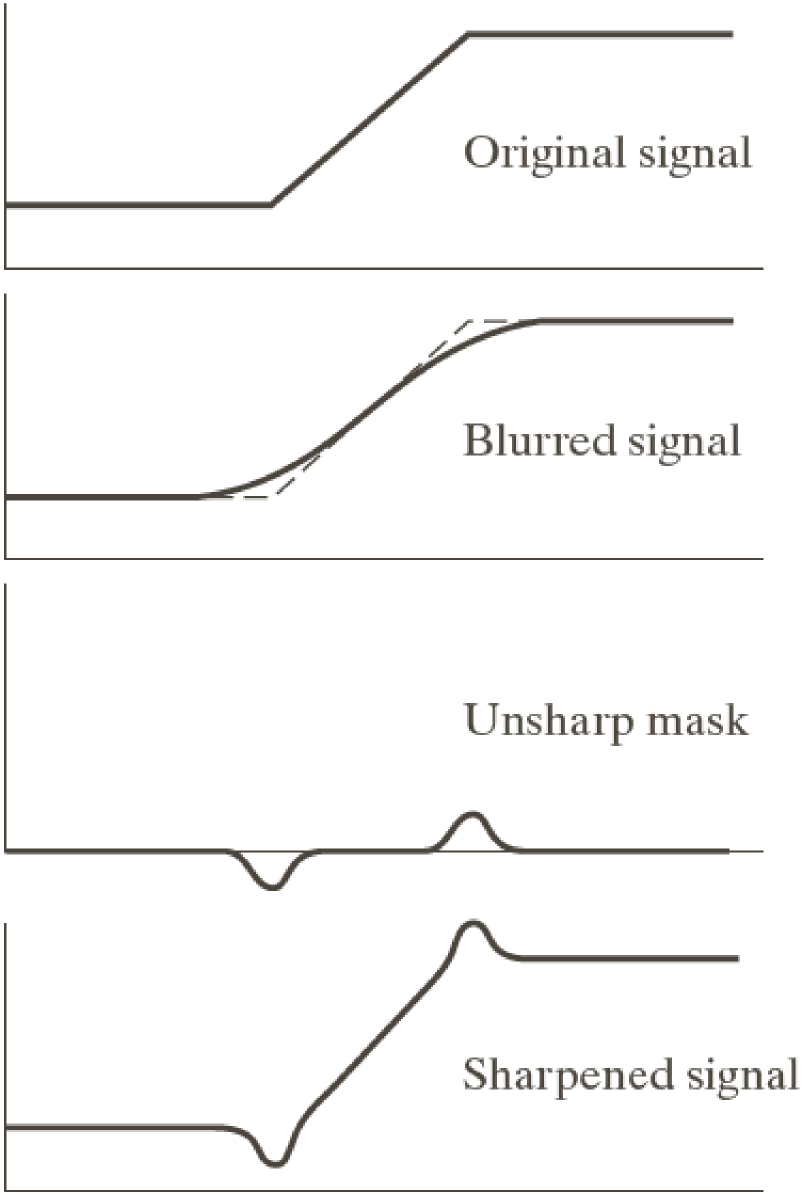
# Unsharp Masking

1. Blur the original image $\bar{f}(x, y)$
2. Obtain mask: $g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$
3. Add to the original:

$$g(x, y) = f(x, y) + k * g_{\text{mask}}(x, y) \qquad (k \geq 0)$$

- Unsharp masking $k = 1$
- highboost filtering $k > 1$

# Unsharp Masking



# Unsharp Masking


Original image


Blurred with a Gaussian filter


Mask


Result of using unsharp masking result


Result of using highboost filtering result

# 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:

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

For practical reasons this can be simplified as:

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

# 1st Derivative Filtering (cont…)

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

$$|\nabla f| \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)|$$
$$+ |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

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 |

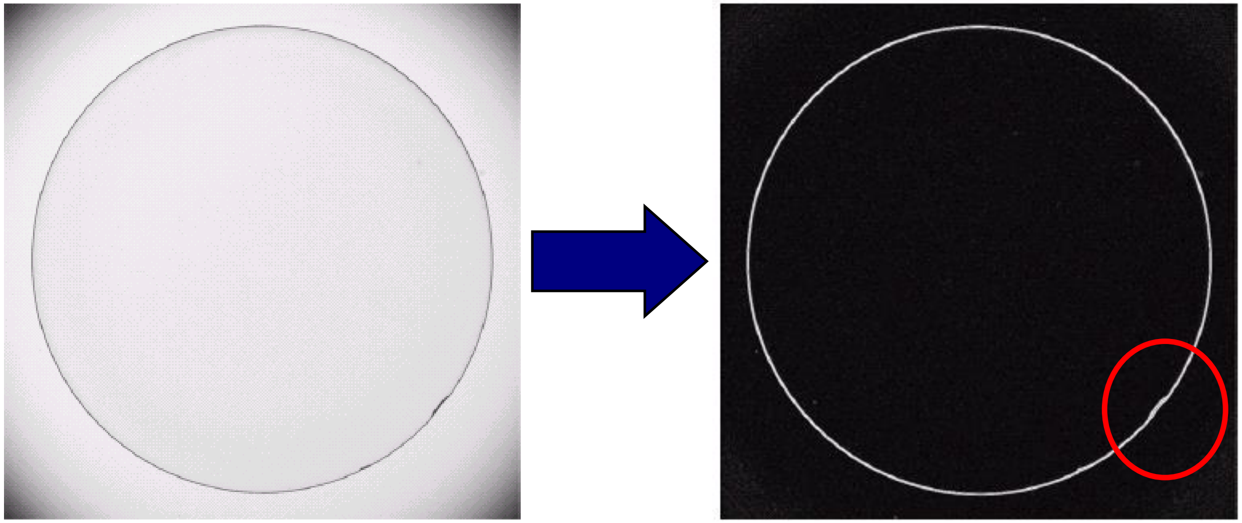
| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

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

# Roberts cross gradient Operators

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

# 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:

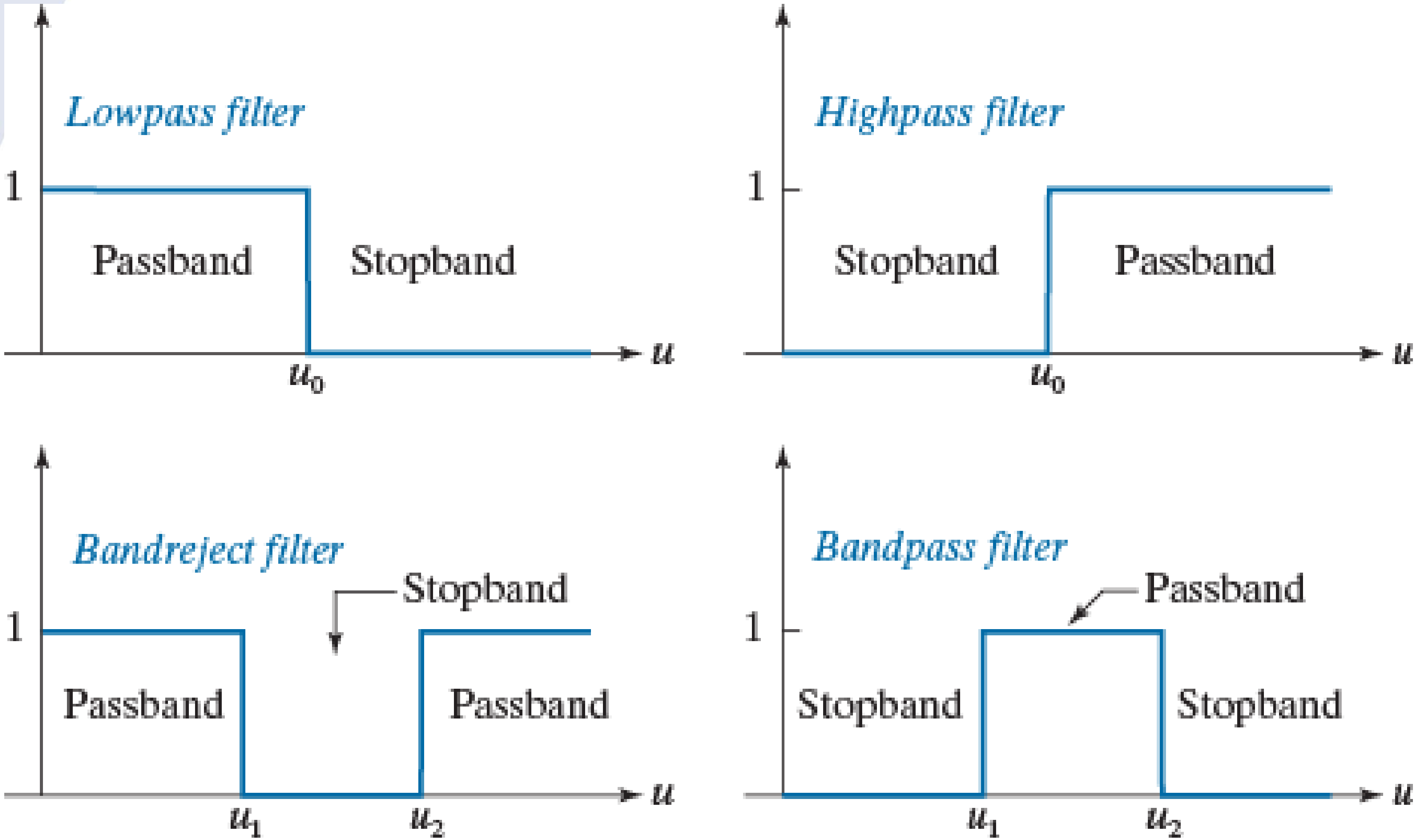
– 1st order derivatives generally produce thicker edges

– 2nd order derivatives have a stronger response to fine detail e.g. thin lines

– 1st order derivatives have stronger response to grey level step

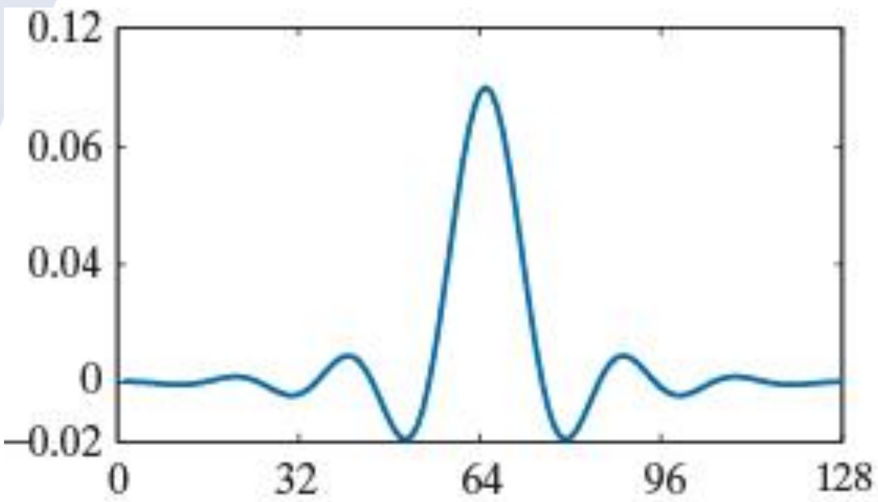
– 2nd order derivatives produce a double response at step changes in grey level

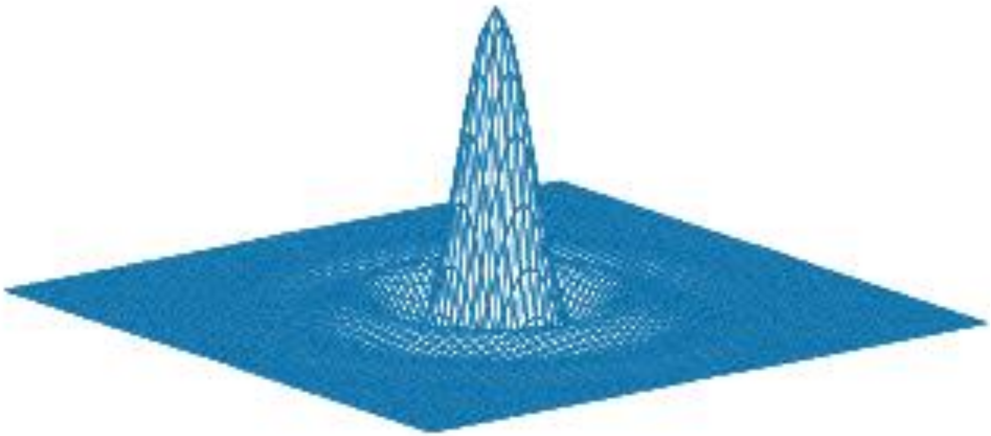
# Highpass, Bandreject, and Bandpass Filters from Lowpass Filters



# Highpass, Bandreject, and Bandpass Filters from Lowpass Filters

| Filter type | Spatial kernel in terms of lowpass kernel, *lp* |
|---|---|
| Lowpass | $lp(x,y)$ |
| Highpass | $hp(x,y) = \delta(x,y) - lp(x,y)$ |
| Bandreject | $br(x,y) = lp_1(x,y) + hp_2(x,y)$ $= lp_1(x,y) + \left[\delta(x,y) - lp_2(x,y)\right]$ |
| Bandpass | $bp(x,y) = \delta(x,y) - br(x,y)$ $= \delta(x,y) - \left[lp_1(x,y) + \left[\delta(x,y) - lp_2(x,y)\right]\right]$ |



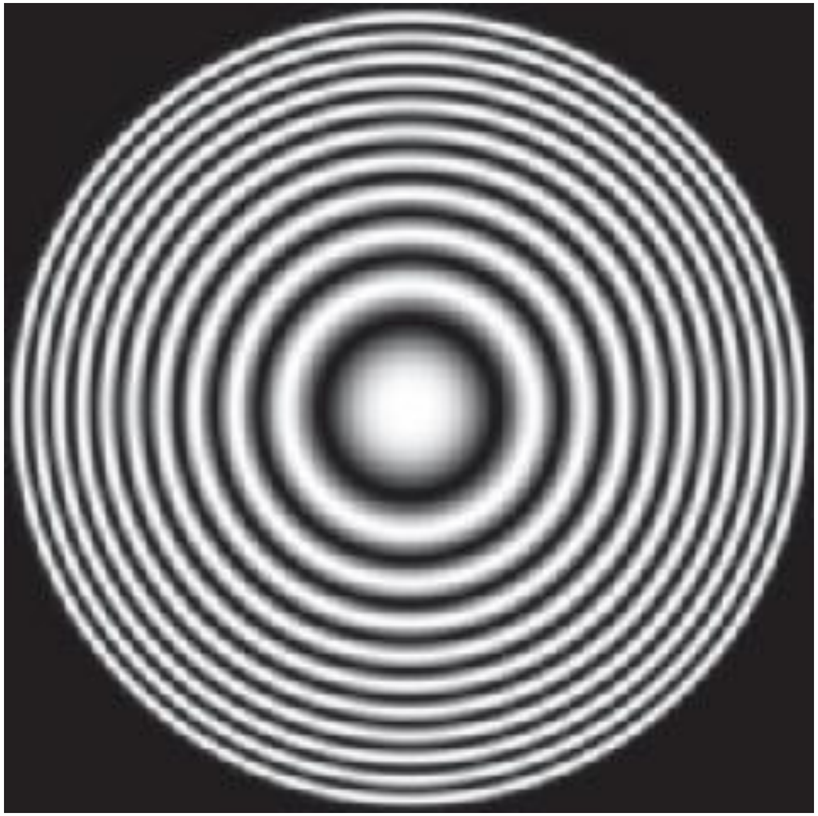
Rotate

1-D spatial lowpass filter ⟹ 2-D spatial lowpass filter

# Zone Plate Image (同心圆反射板)

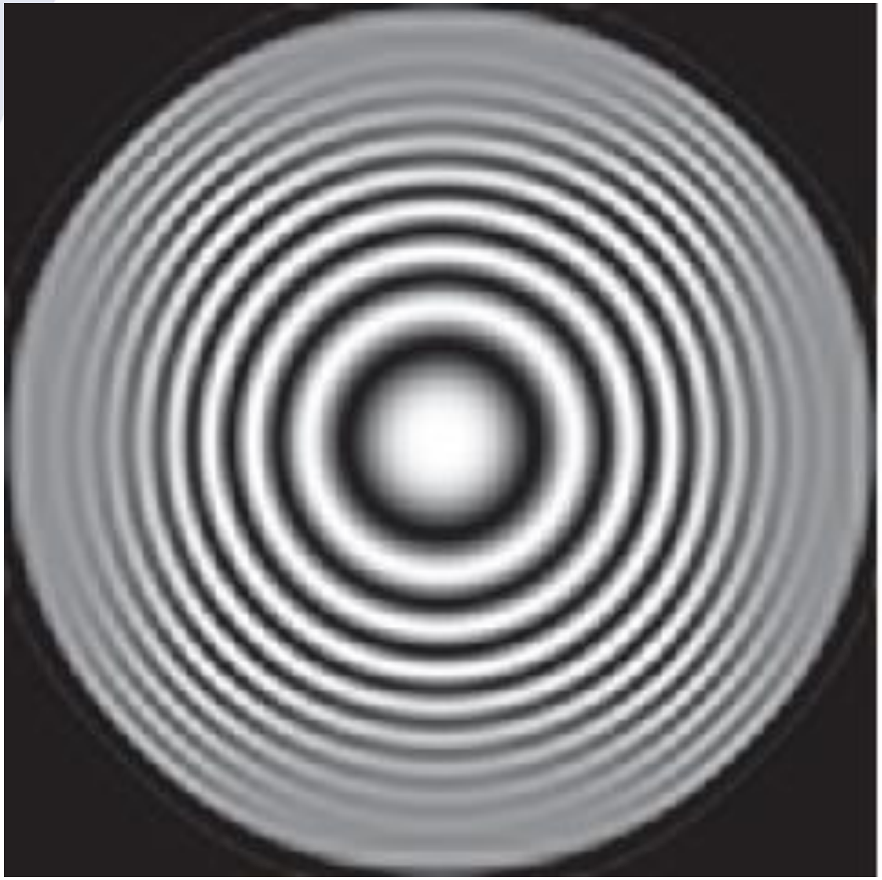
- Test image for filtering
  - [-8.2,8.2], step 0.0275
  - 597 x 597 pixels

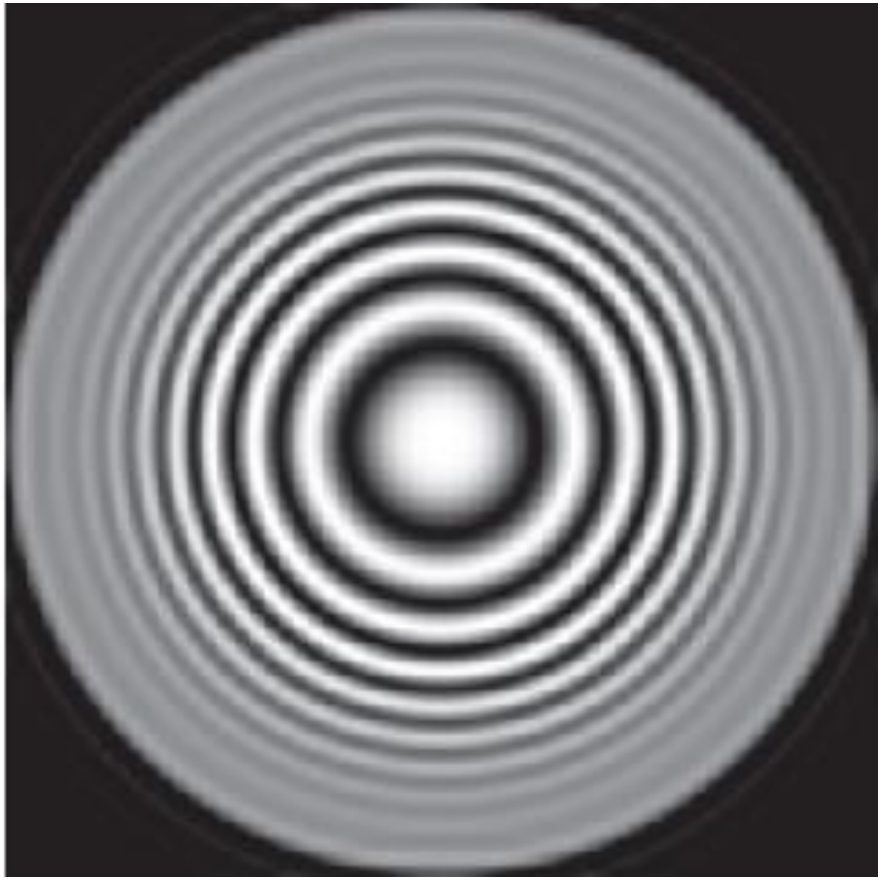
$$z(x,y)=\frac{1}{2}\left[1+\cos\left(x^2+y^2\right)\right]$$



- Spatial frequency increases with distance from the center

# Result of Filtering the Zone Plate Image

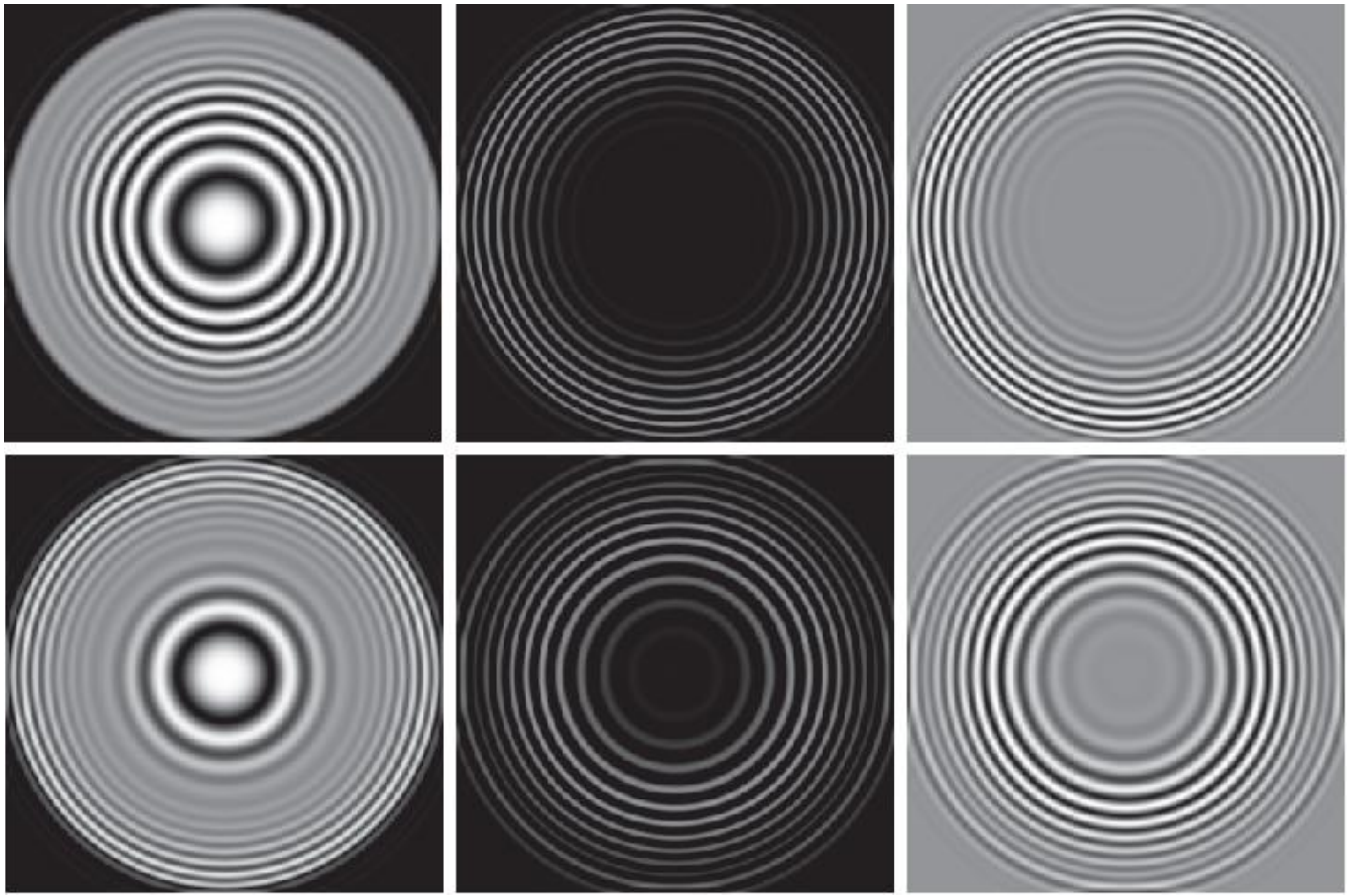




Using a Separable 2-D Filter Using an Isotropic 2-D Filter

# More Filtering Results

Lowpass Filtered Highpass Filtered Scaled for Display



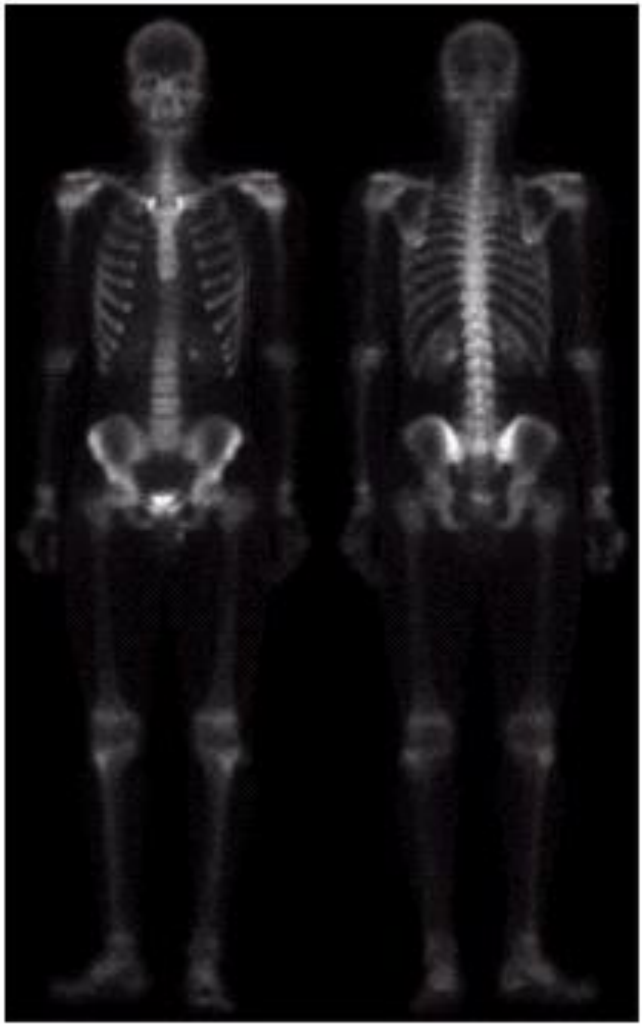
Bandreject Filtered Bandpass Filtered Scaled for Display

# Combining Spatial Enhancement Methods

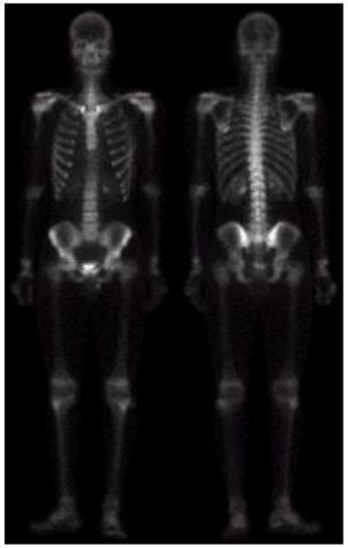
Successful image enhancement is typically not achieved using a single operation

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

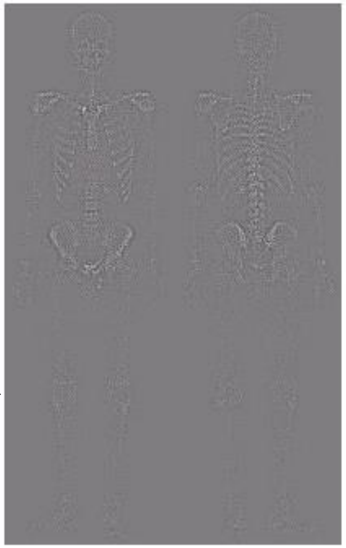


# Combining Spatial Enhancement Methods (cont…)



(a)

Laplacian filter of bone scan (a)



(b)

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

Edge enhanced
Noise also enhanced!

(c)

Sobel filter of bone scan (a)

(d)

(c) Edge enhanced
Noise also enhanced!

# Combining Spatial Enhancement Methods (cont…)

**(e)**

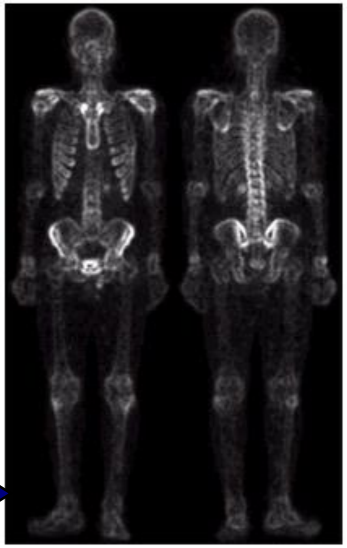
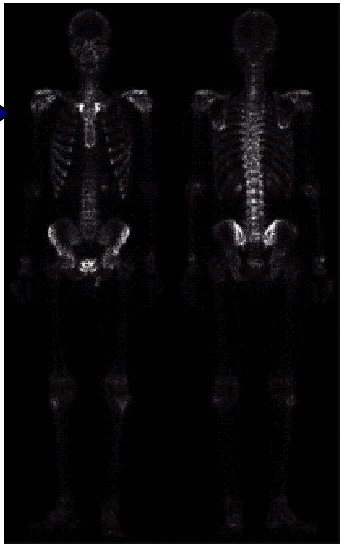


Image (d) smoothed with a 5*5 averaging filter

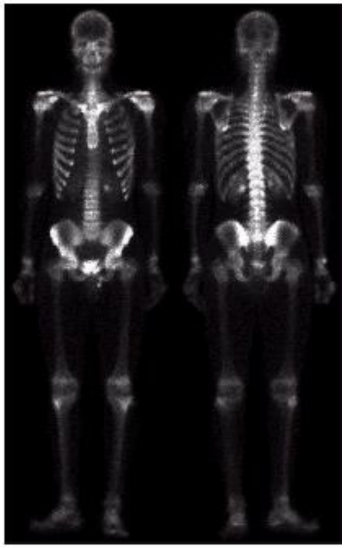
The product of (c) and (e) which will be used as a **mask**

**(f)**



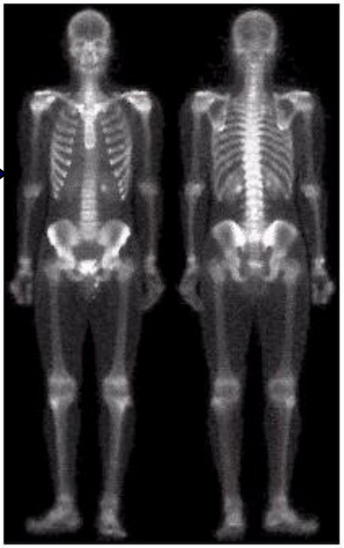
Sharpened image which is sum of (a) and (f)

**(g)**

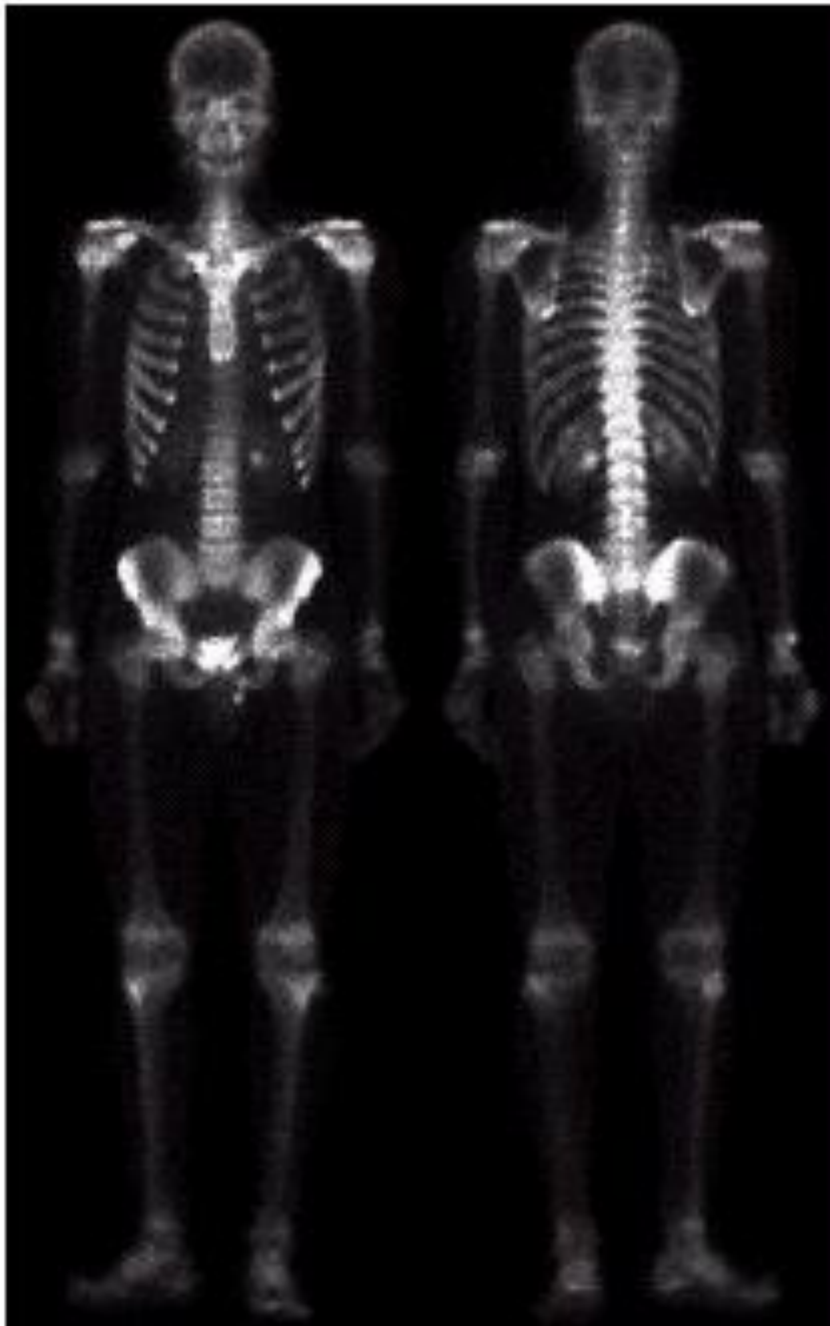


Result of applying a power-law trans. to (g)

**(h)**



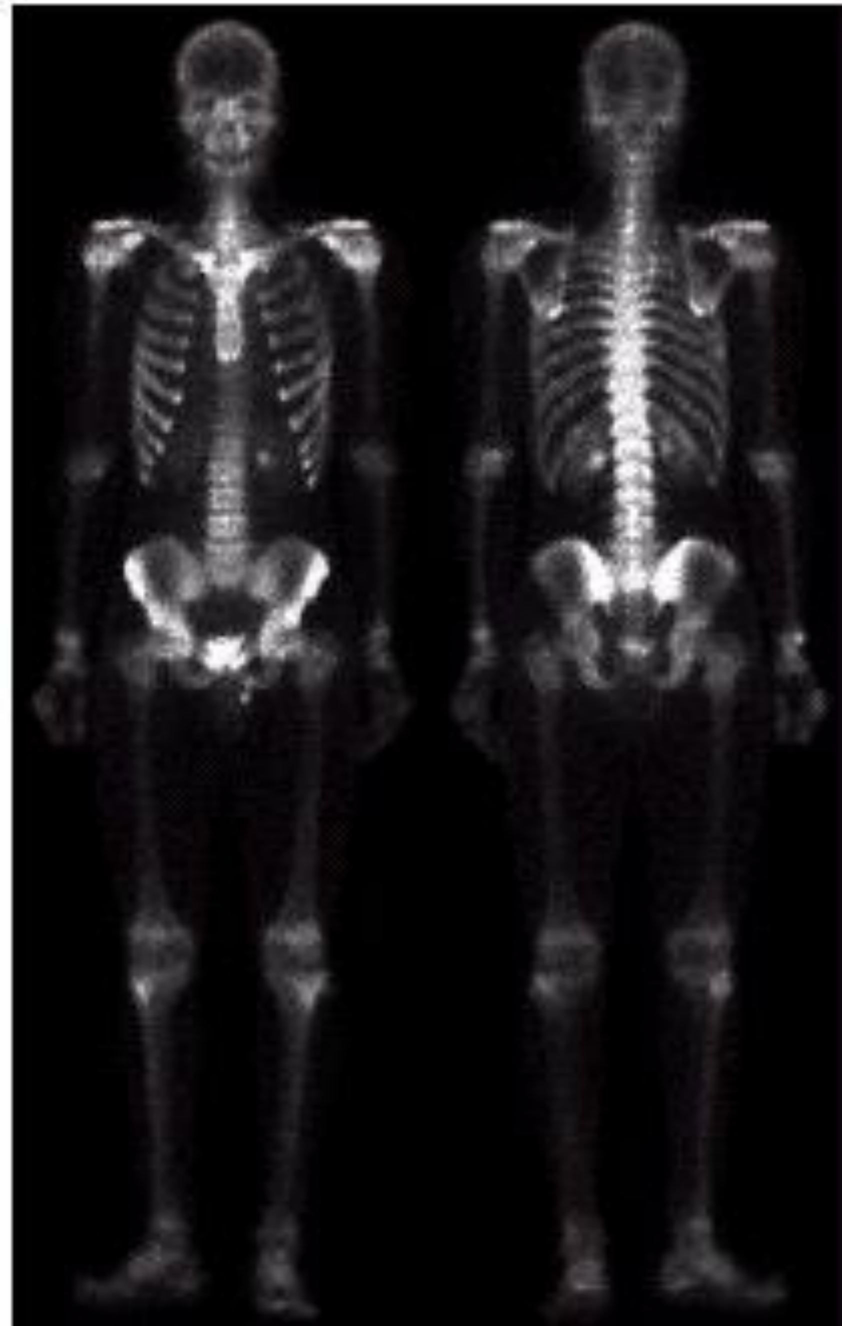
Edge enhanced
Noise NOT enhanced



(g) Edge enhanced
Noise NOT enhanced

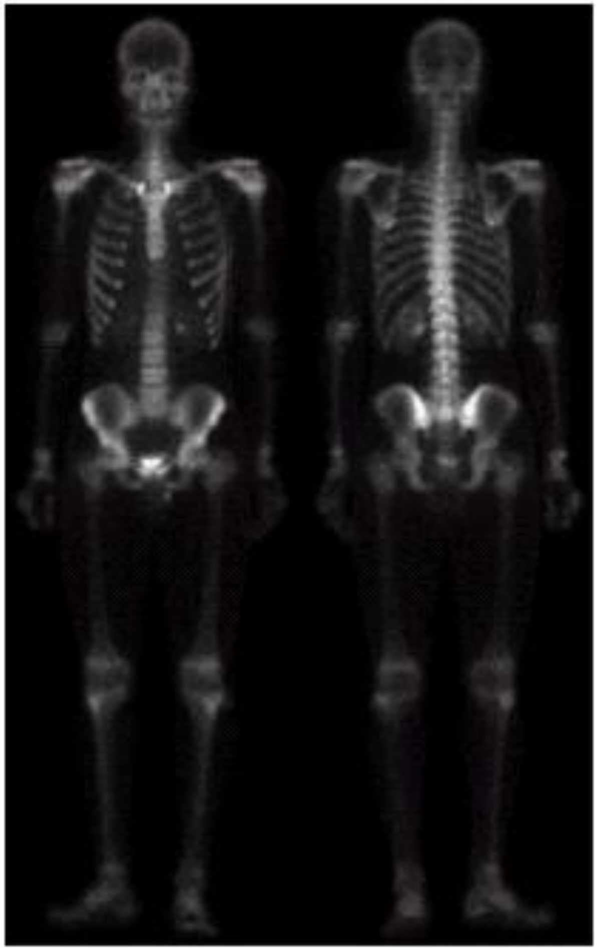
(c)

(g)



# Combining Spatial Enhancement Methods (cont…)

Compare the original and final images



# Assignments

课后作业题目请对照参考第4版英文原版

- 3.26（设图像大小为*N*x*N*，*N*>>8；滤波时图像边界外填充0）
- 3.34
- 3.36

# Assignments

- 选做1个编程作业：Proj03-xx

（参见Laboratory Projects_DIP3E.pdf）

DDL: 2周后，课前

- 递交1份实验报告，命名“学号姓名_prj1.pdf”，内容提纲包括：

1. 实验任务： 描述本次实验的任务，即所选择的Proj03-xx题目
2. 算法设计：理论上描述所设计的算法
3. 代码实现：描述编程环境，给出自己编写的核心代码
4. 实验结果：描述具体的实验过程，给出每个小实验的输入数据、算法参数和实验结果，并对结果做简要的讨论
5. 总结：简要总结本次实验的技术内容，以及心得体会。