Digital Image Processing

Image Enhancement
Spatial Filtering

From:

Digital Image Processing, Chapter 3

Refael C. Gonzalez & Richard E. Woods

Contents

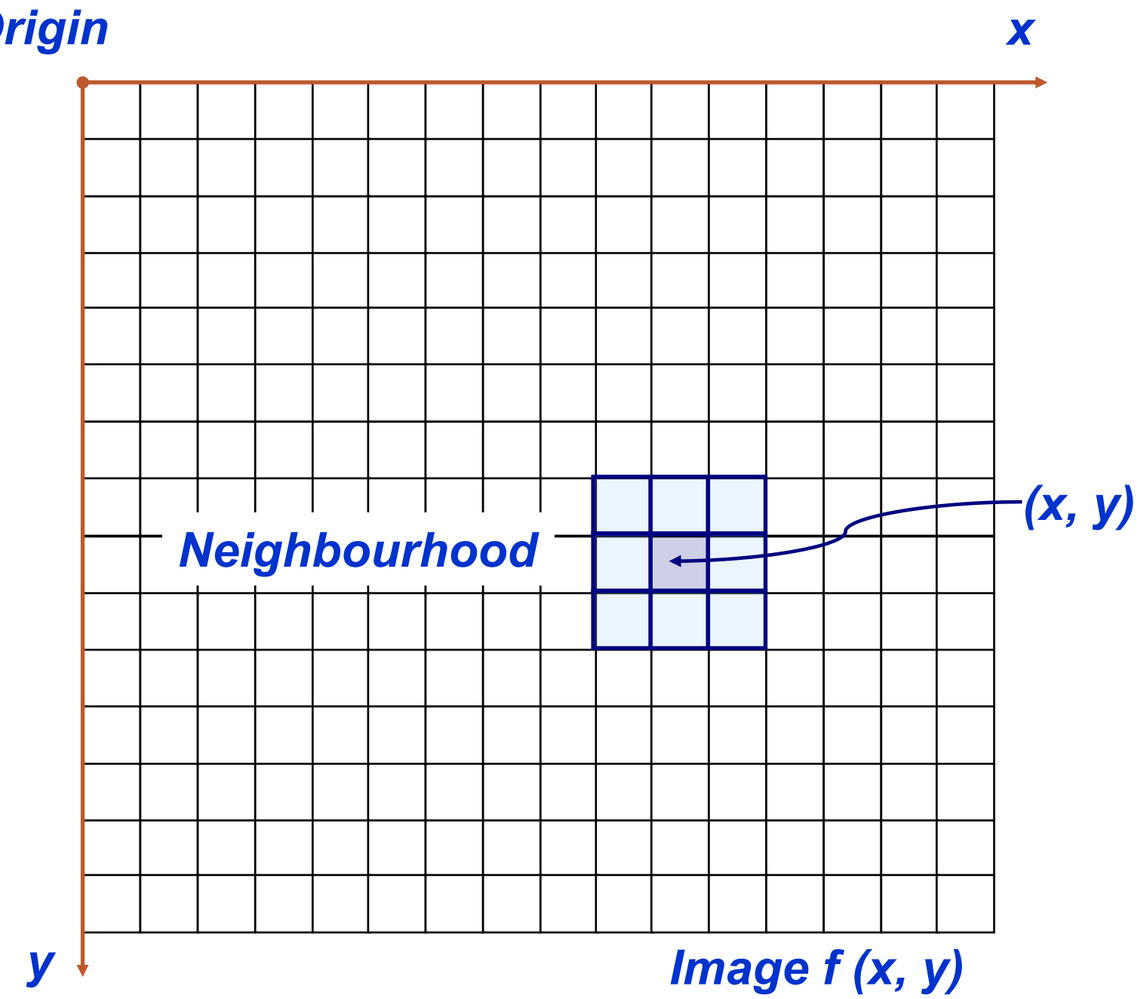
- What is spatial filtering?
- Smoothing Spatial filters.
- Sharpening Spatial Filters.

Neighbourhood Operations

Neighbourhood operations simply operate on a larger neighbourhood of pixels than point operations

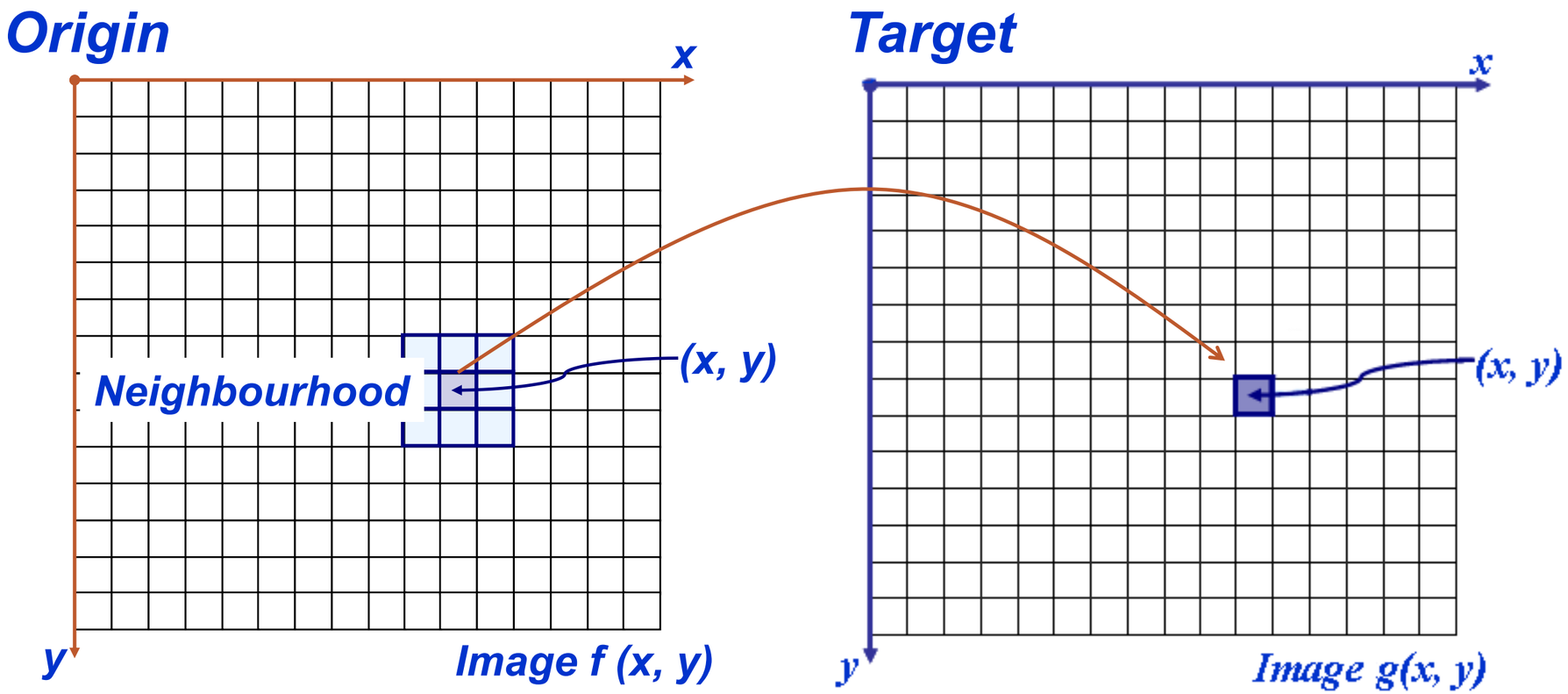
Neighbourhoods are mostly a rectangle around a central pixel

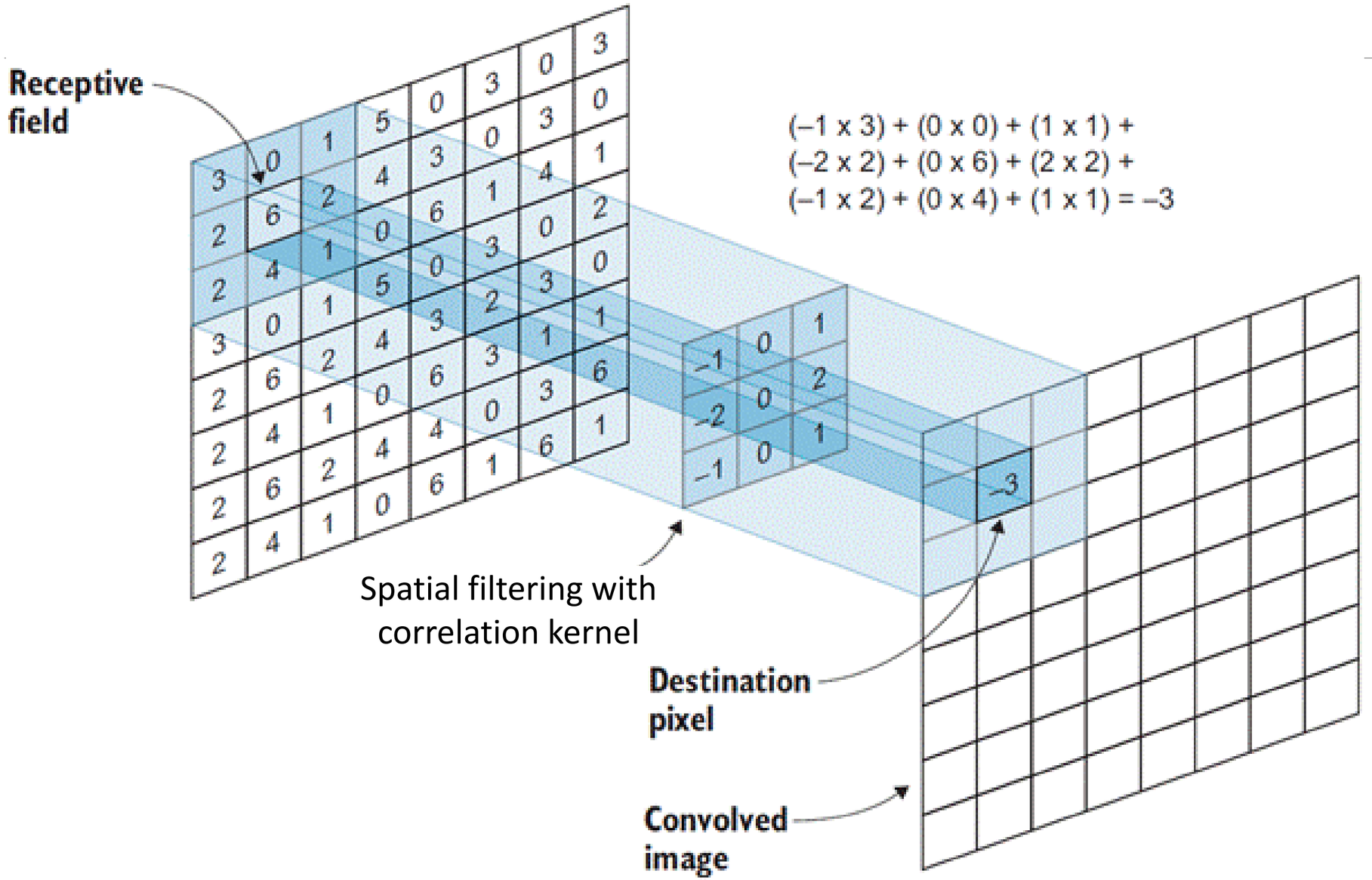
Any size rectangle and any shape filter are possible



Neighbourhood Operations

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



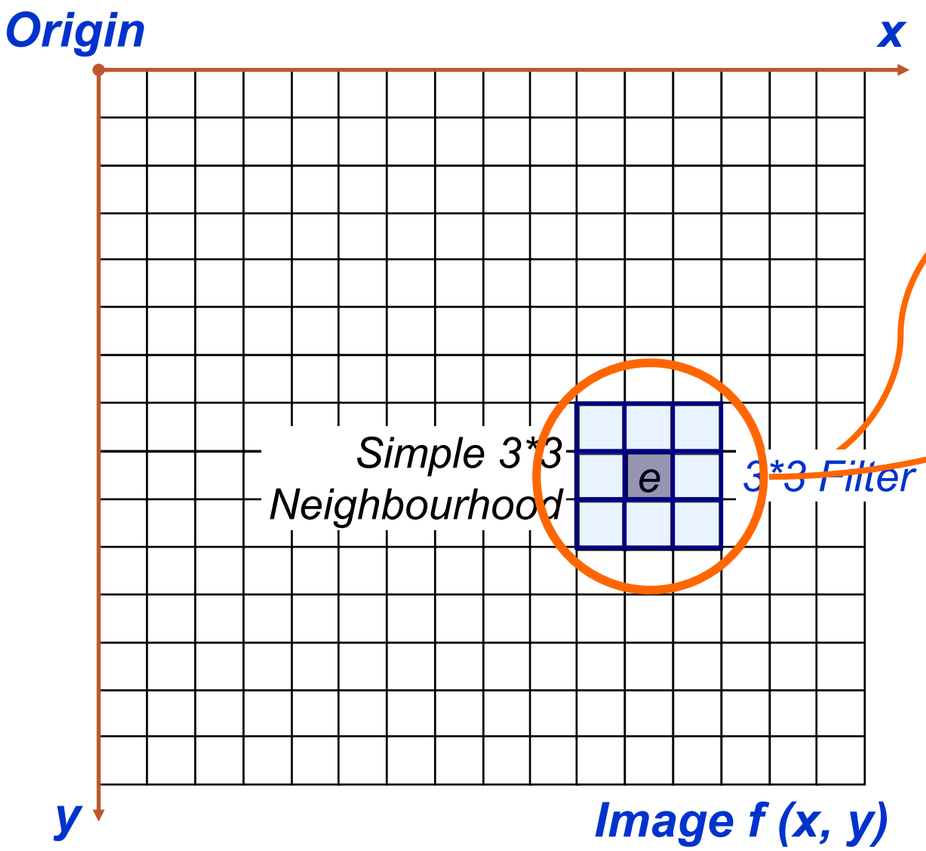

Receptive field
(–1 x 3) + (0 x 0) + (1 x 1) +
(–2 x 2) + (0 x 6) + (2 x 2) +
(–1 x 2) + (0 x 4) + (1 x 1) = –3
Spatial filtering with correlation kernel
Destination pixel
Convolved image

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
- **Try this.**

The Spatial Filtering Process



a	b	c
d	e	f

Original Image Pixels

o

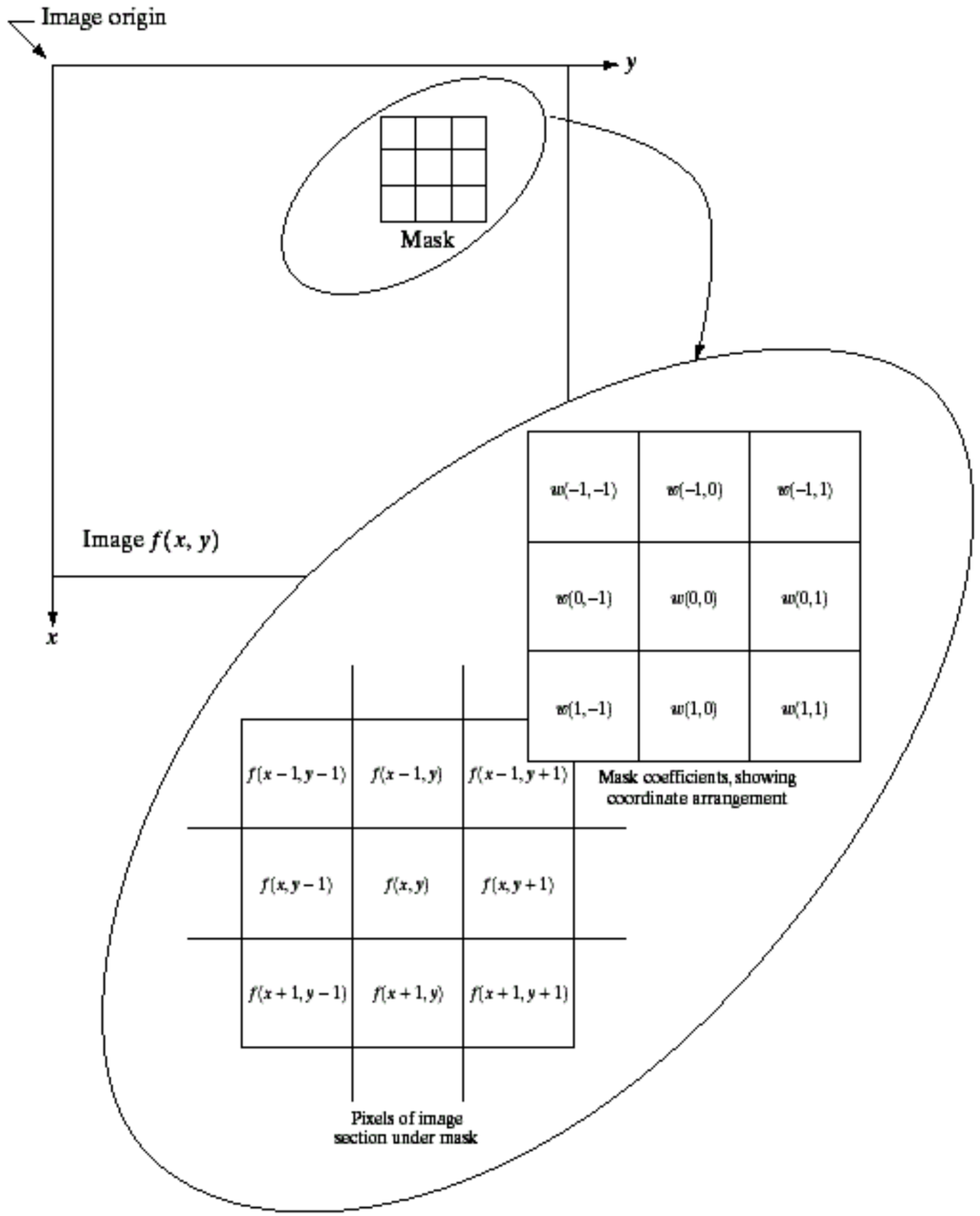
j	k	l
m	n	o
p	q	r

Filter (*w*)

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

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

Spatial Filtering: Equation Form



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

a=b=1

Filtering can be given in equation form as shown above

Notations are based on the image shown to the left

Smoothing Spatial Filters

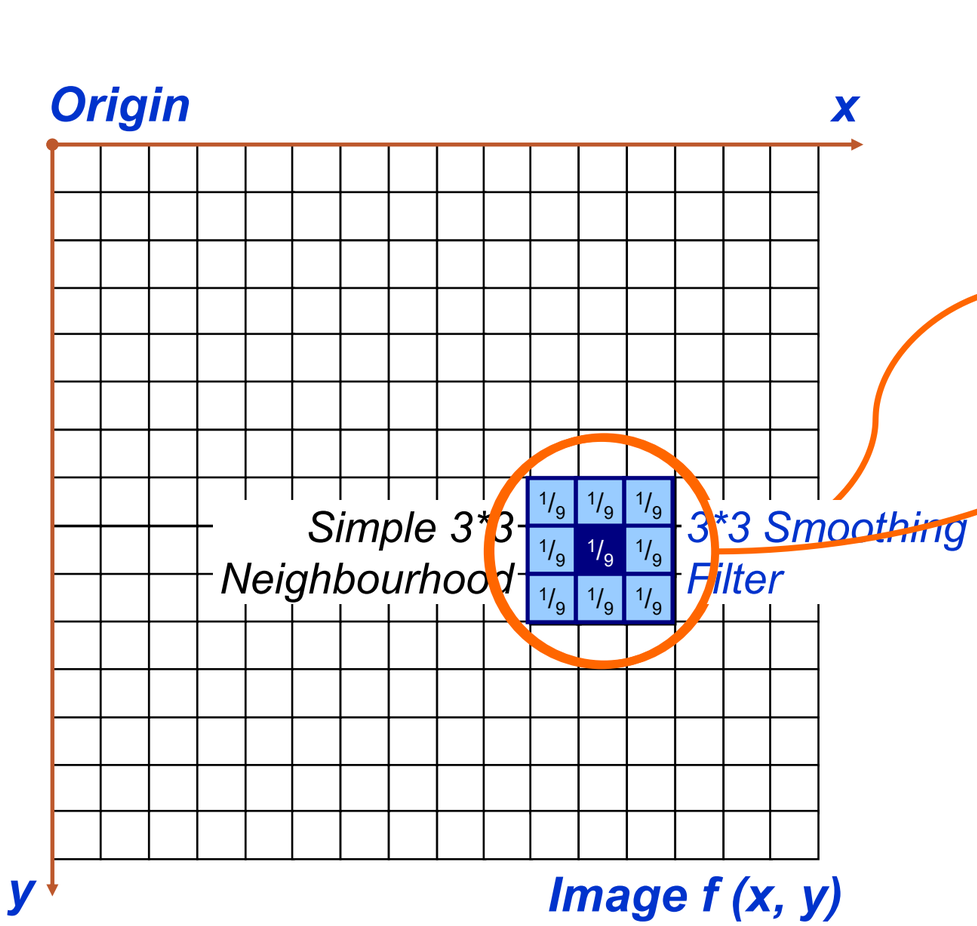
One of the simplest spatial filtering operations we can perform is a smoothing operation

- 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



Original Image Pixels

104	100	108
99	106	98
95	90	85

Filter

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

Result

	e	

$$e = 1/9*106 + 1/9*104 + 1/9*100 + 1/9*108 + 1/9*99 + 1/9*98 + 1/9*95 + 1/9*90 + 1/9*85$$

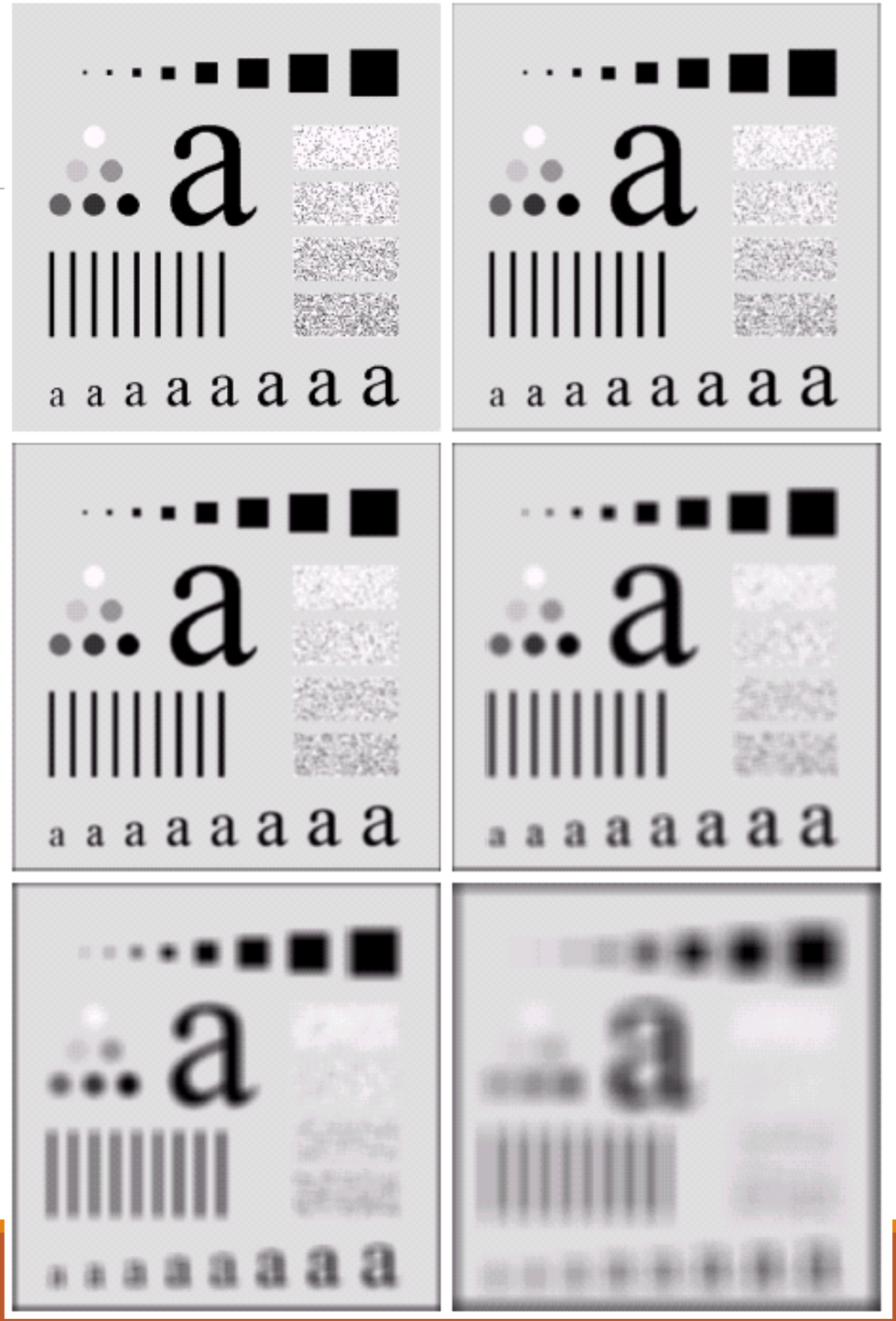
$$= 98.3333$$

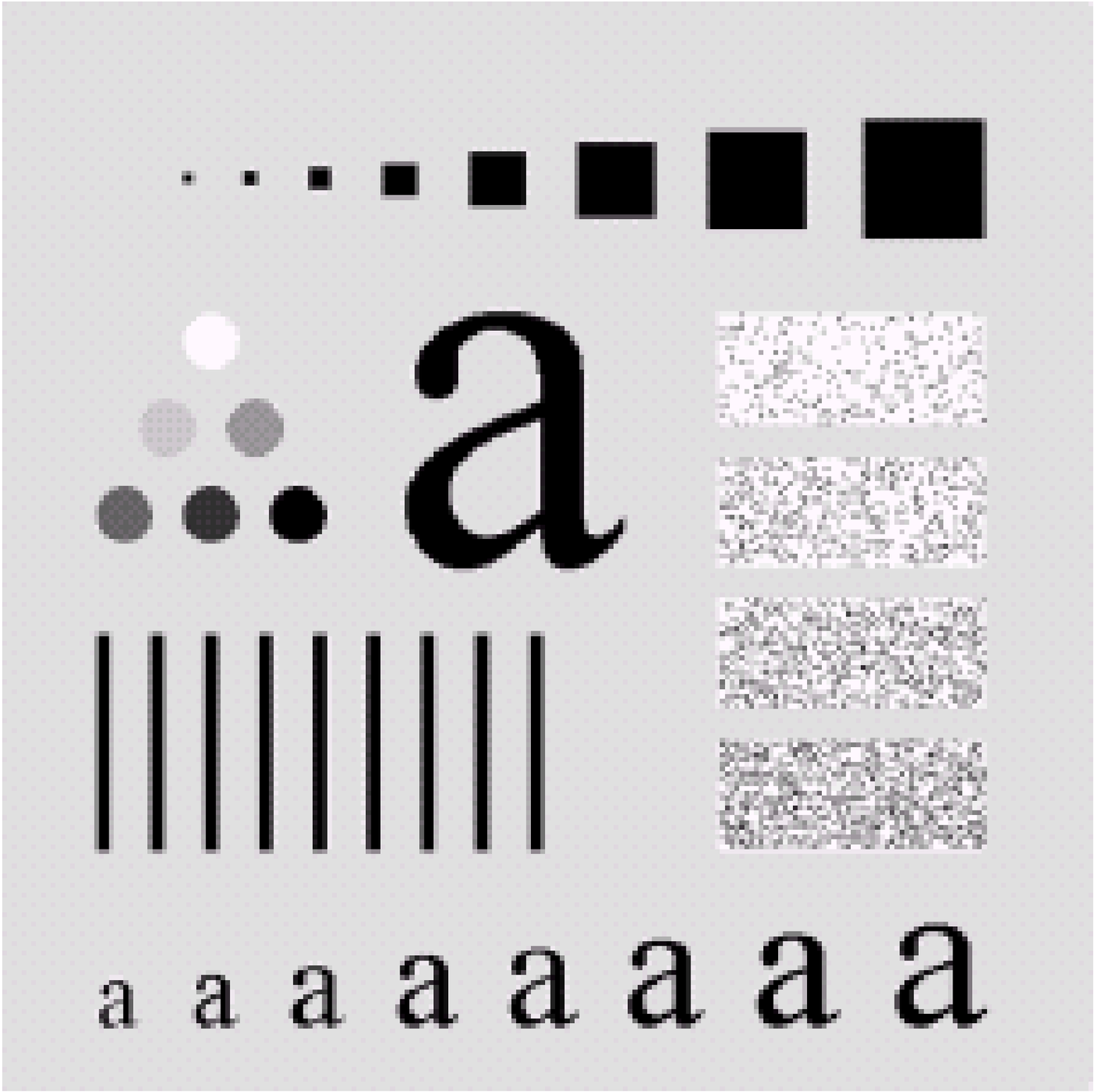
$$=(104+100+108+99+106+98+95+90+85)/9$$

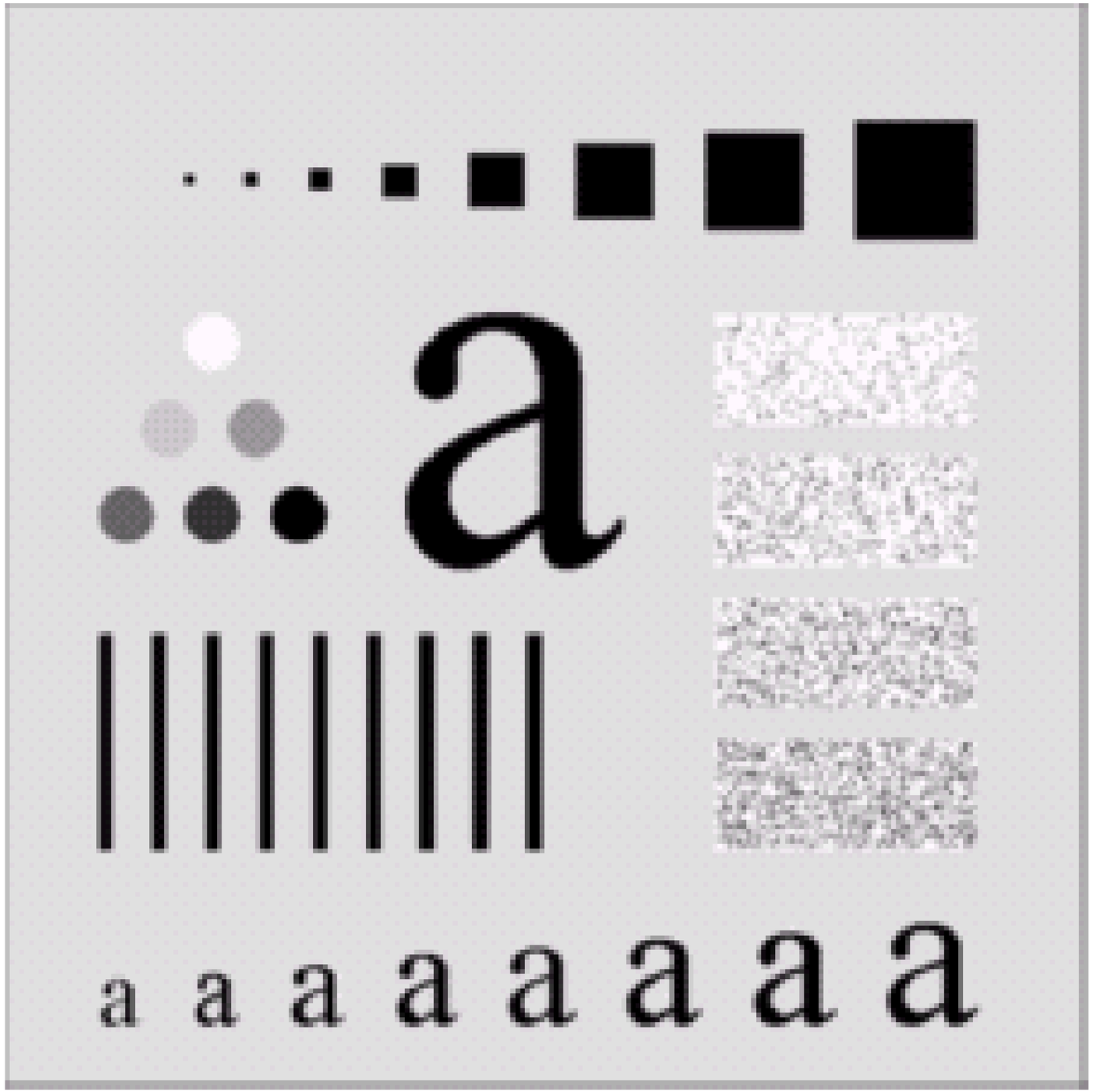
The above is repeated for every pixel in the original image to generate the smoothed image

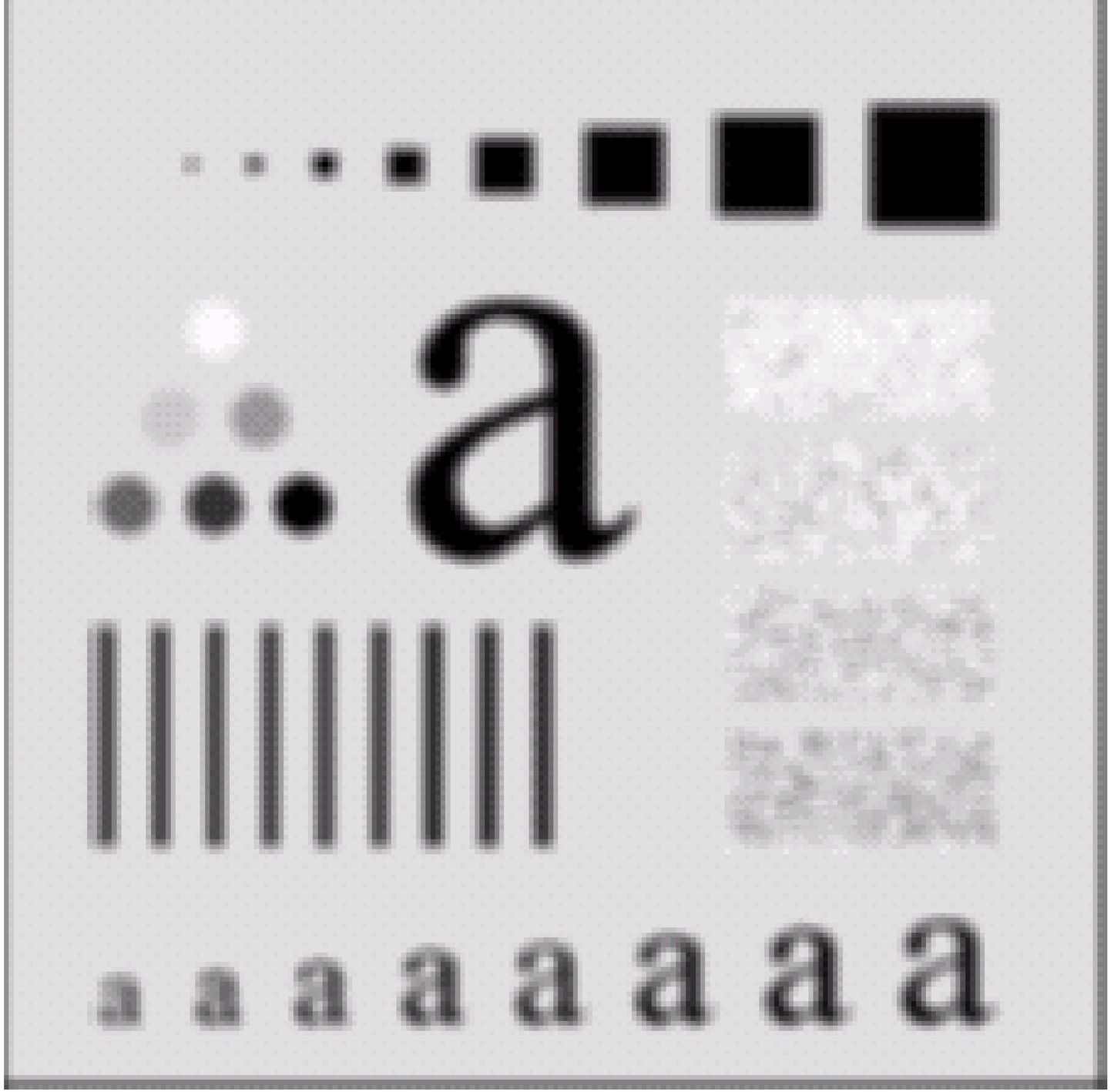
Image Smoothing Example

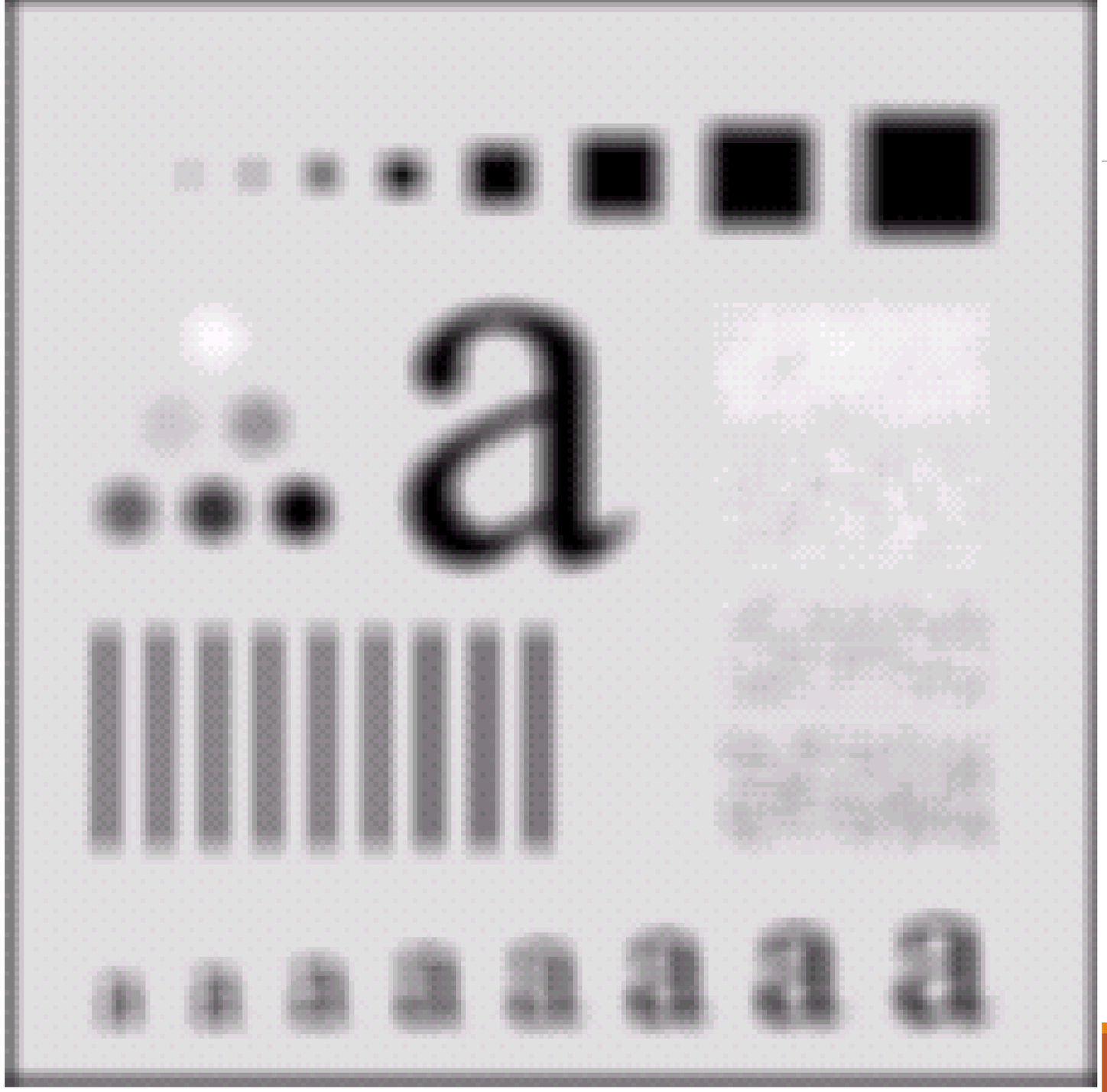
- The image at the top left is an original image of size 500*500 pixels
- The subsequent images show the image after filtering with an averaging filter of increasing sizes
  - 3, 5, 9, 15 and 35
- Notice how detail begins to disappear
- **Try that yourself**

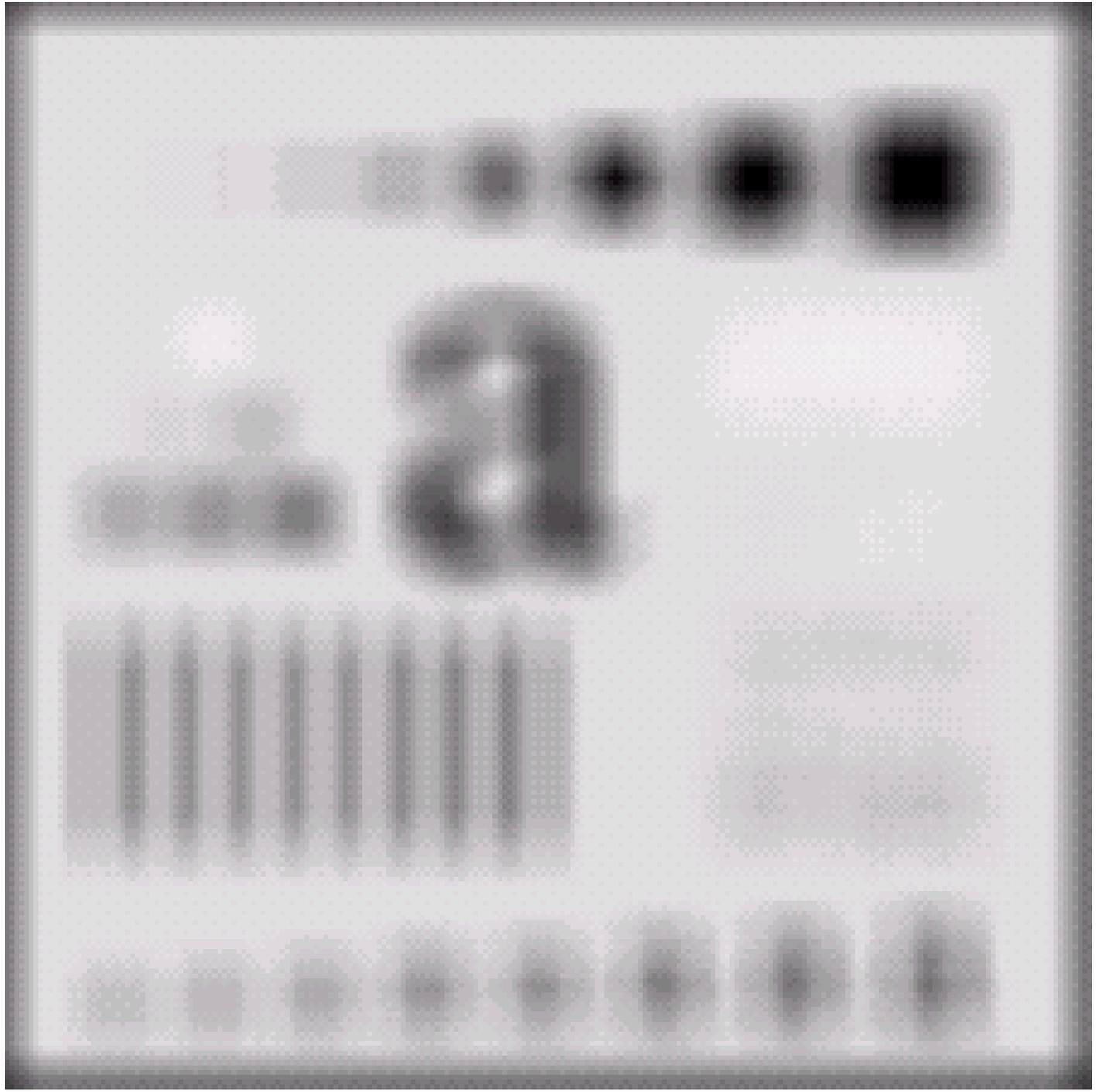


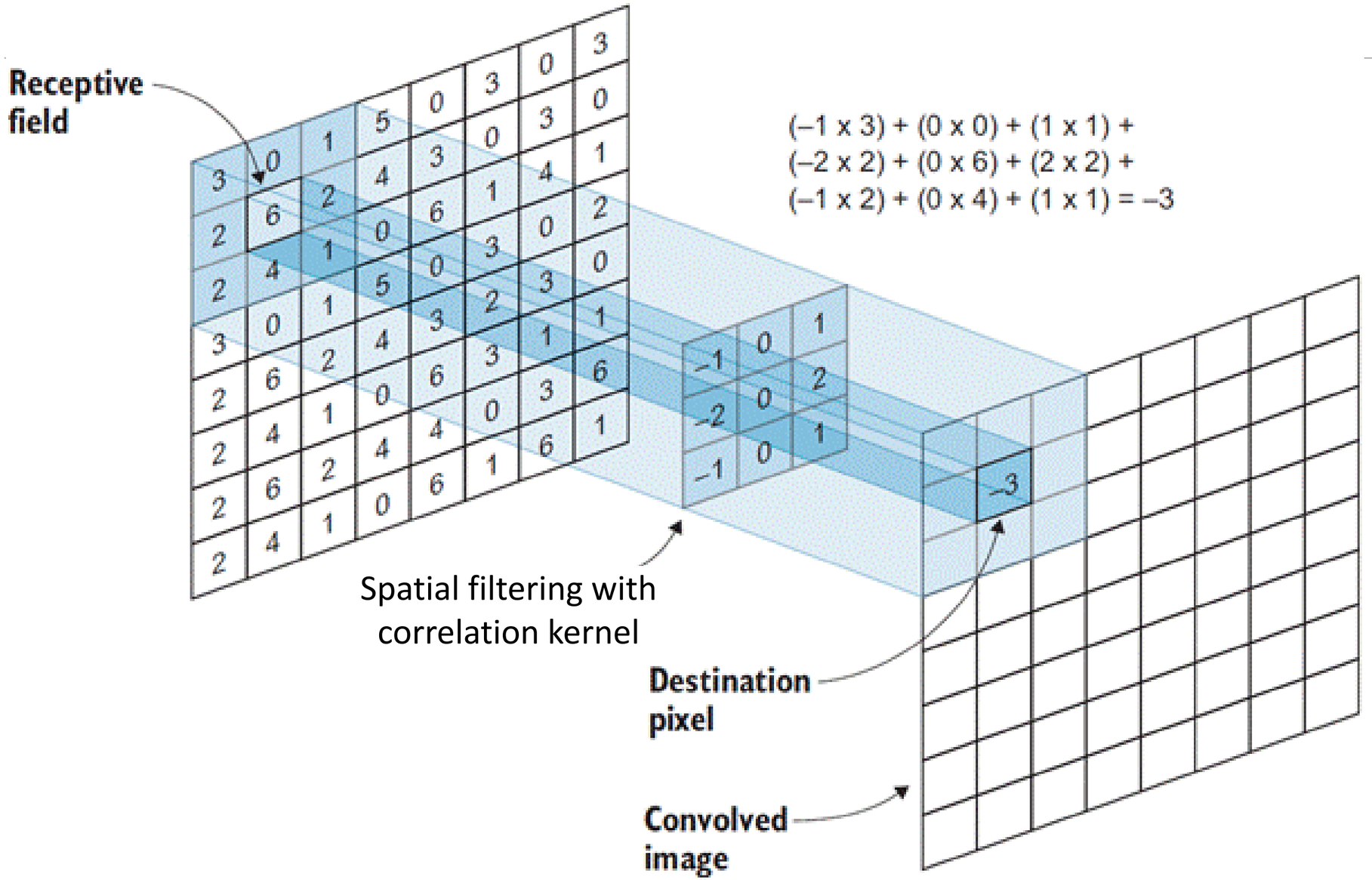

a
a a a a a a a a


a
a a a a a a a a



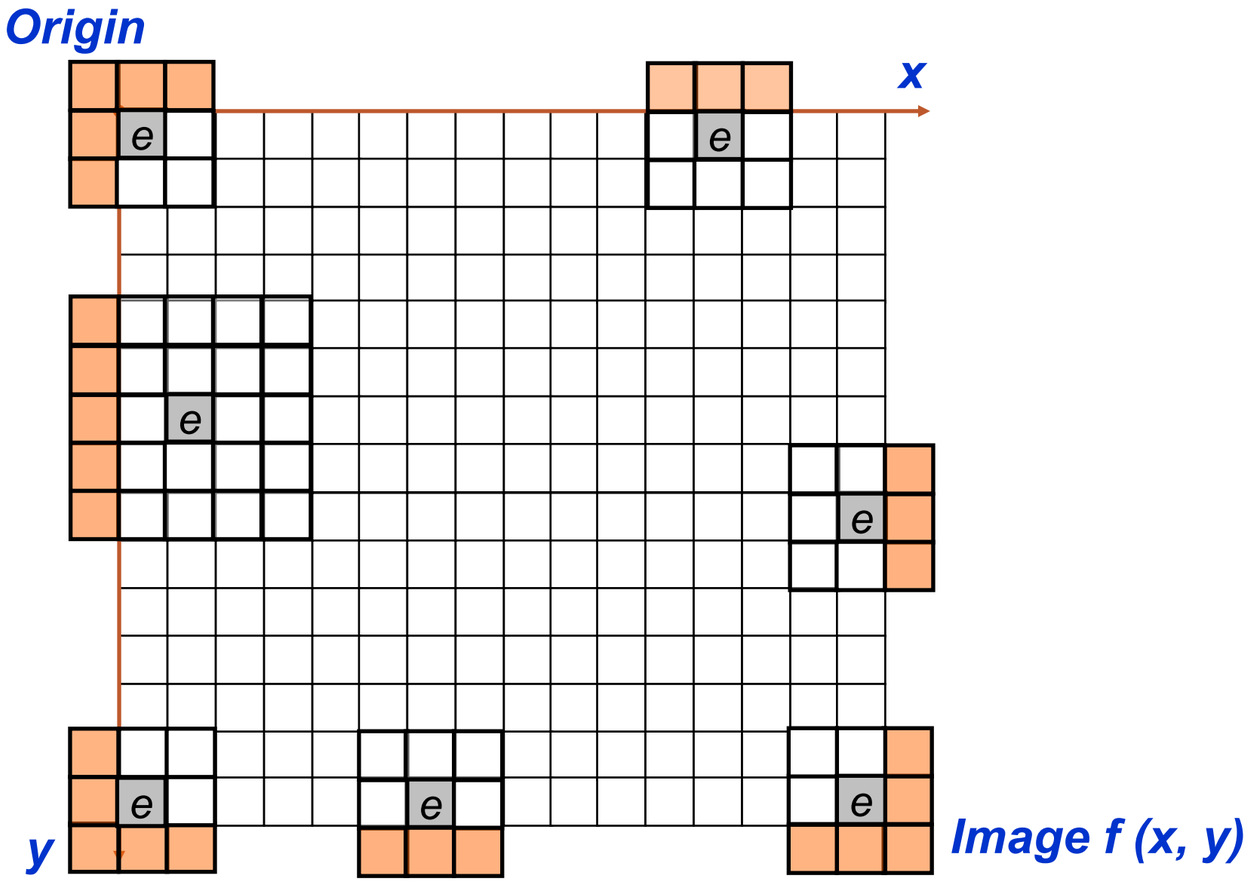





Receptive field
(–1 x 3) + (0 x 0) + (1 x 1) +
(–2 x 2) + (0 x 6) + (2 x 2) +
(–1 x 2) + (0 x 4) + (1 x 1) = –3
Spatial filtering with correlation kernel
Destination pixel
Convolved image

Strange Things Happen At The Edges!

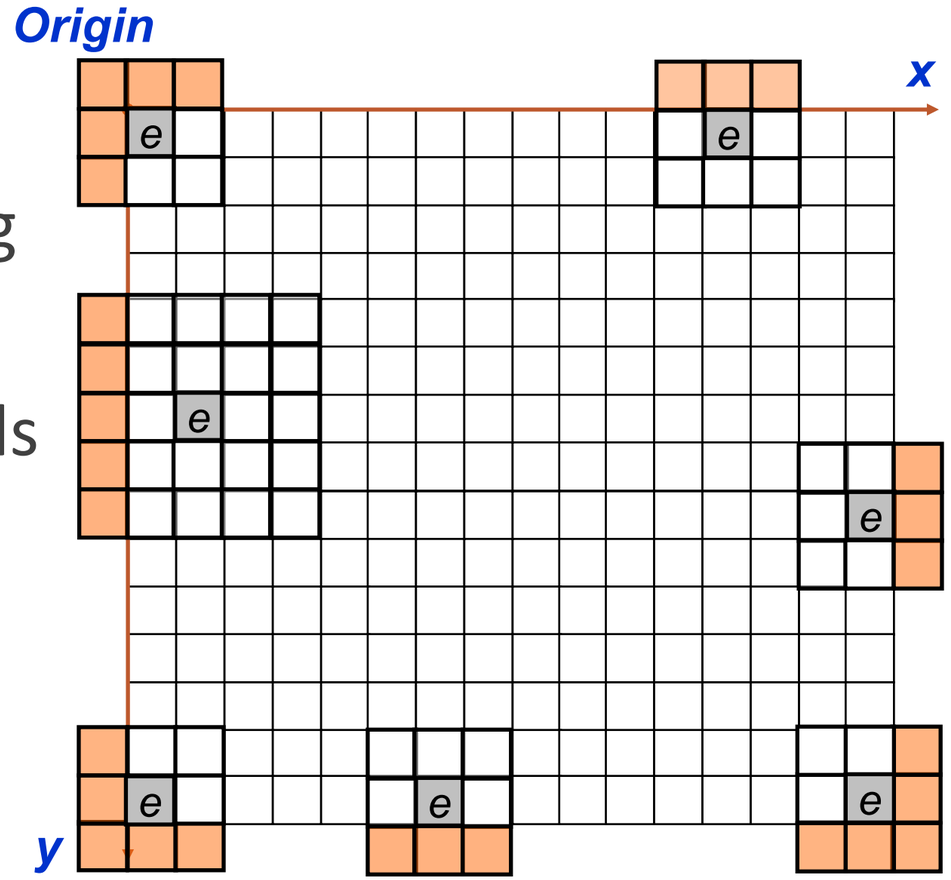
At the edges of an image we are missing pixels to form a neighbourhood



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

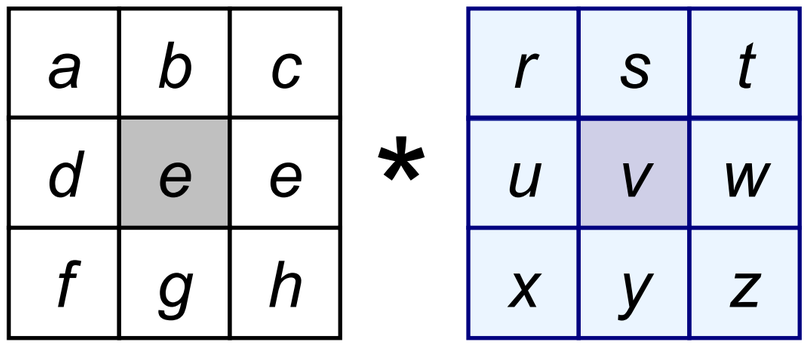


Correlation & Convolution

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

For symmetric filters it makes no difference



Original Image Pixels **Filter**

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

The key difference between the two is that convolution is associative. That is, if F and G are filters, then F*(G*I) = (F*G)*I. The nice thing about this is that filters can be precomputed, and we only have to convolve one filter with our image.

Weighted Smoothing Filters

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

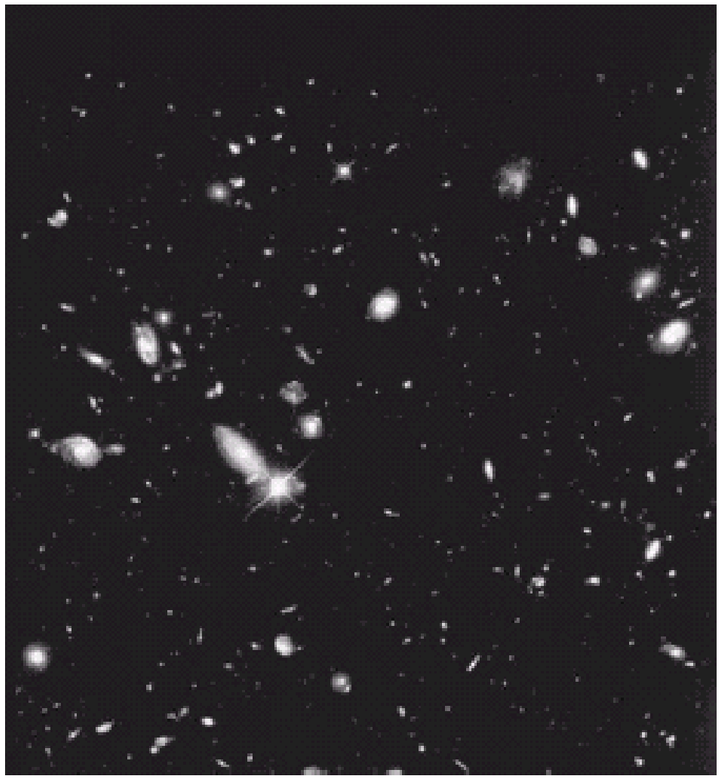
- Pixels closer to the central pixel are more important
- Often referred to as a weighted averaging

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

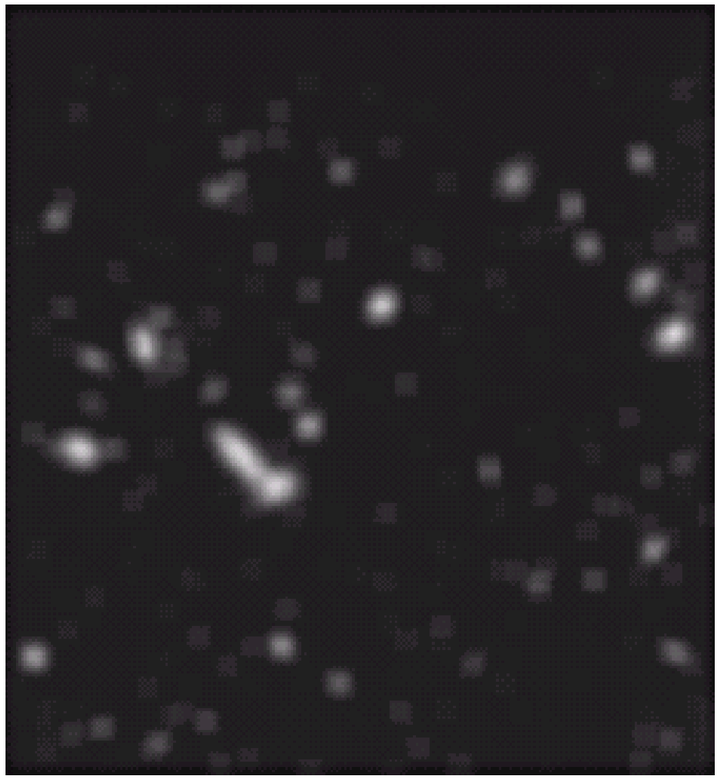
Weighted averaging filter

Another Smoothing Example

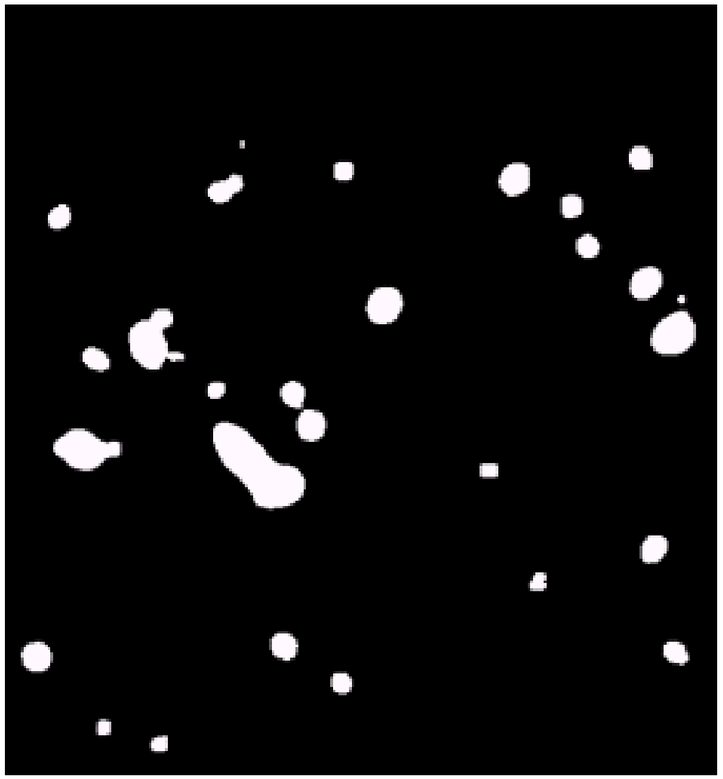
By smoothing the original image we get rid of lots of the finer detail which leaves only the gross features for thresholding.



Original Image



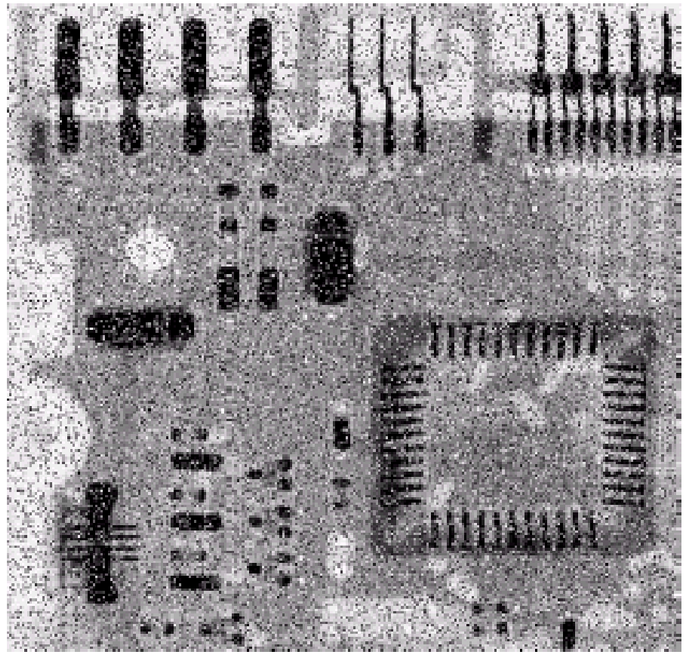
Smoothed Image



Thresholded Image

* Image taken from Hubble Space Telescope

Averaging Filter Vs. Median Filter Example



Original Image With Noise

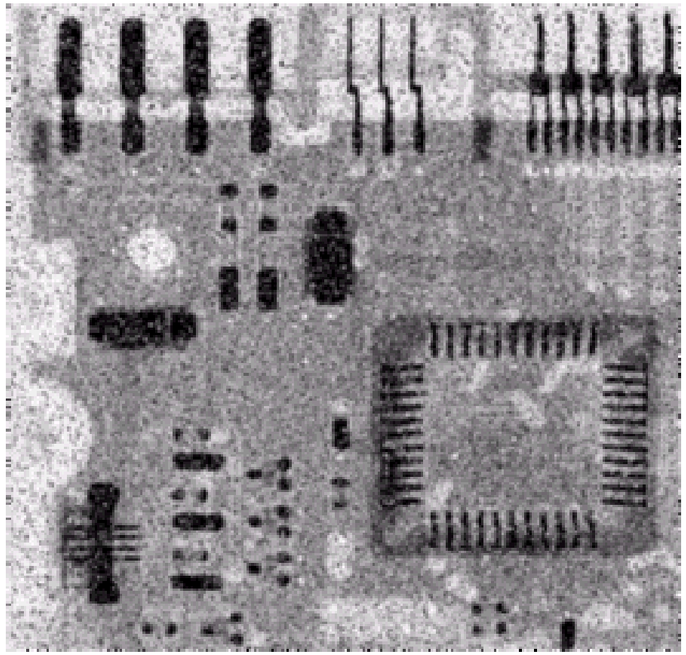


Image After Averaging Filter

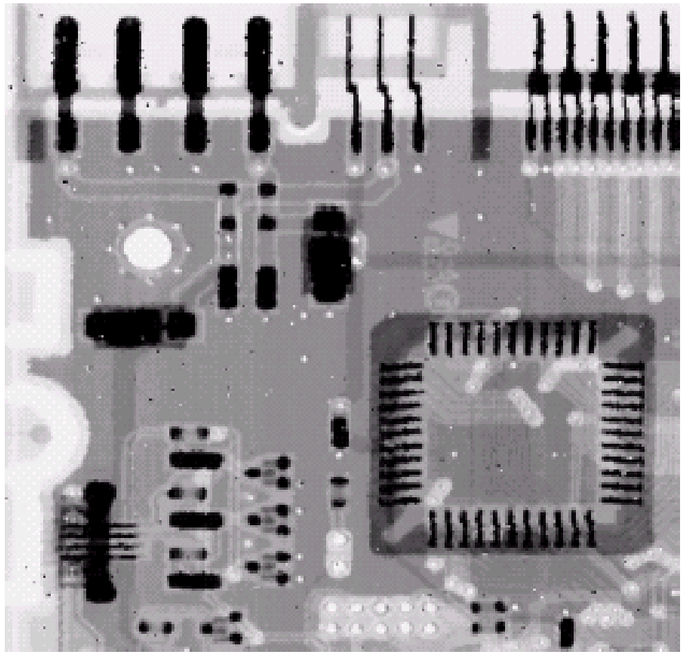


Image After Median Filter

Median[2 2 80] = 2

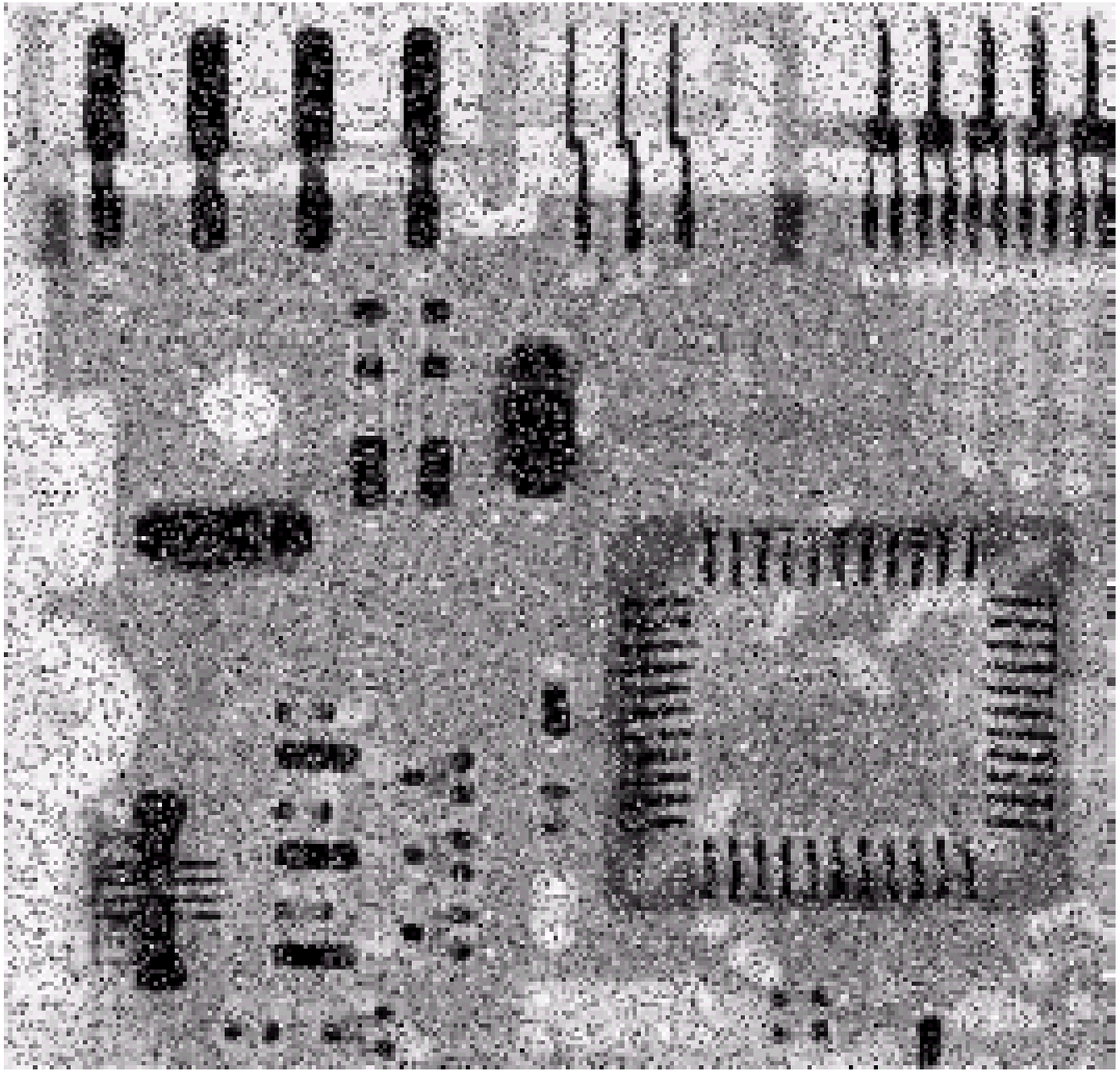
Median[2 80 6] = Median[2 6 80] = 6

Median[80 6 3] = Median[3 6 80] = 6

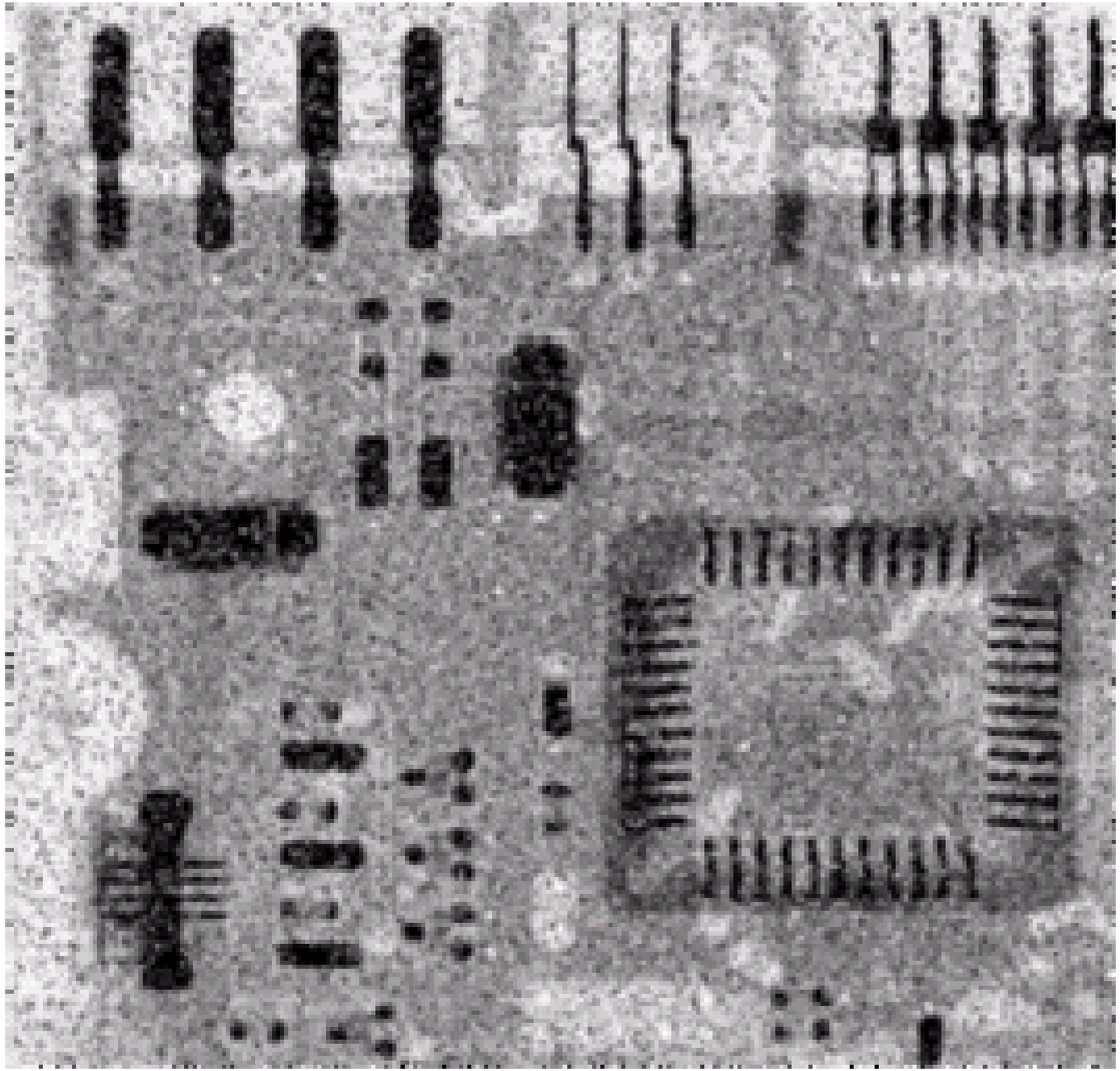
Median[6 3 3] = Median[3 3 6] = 3

- Filtering is often used to remove noise from images
- Sometimes a median filter works better than an averaging filter
- The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the **middle number** of neighbouring entries.

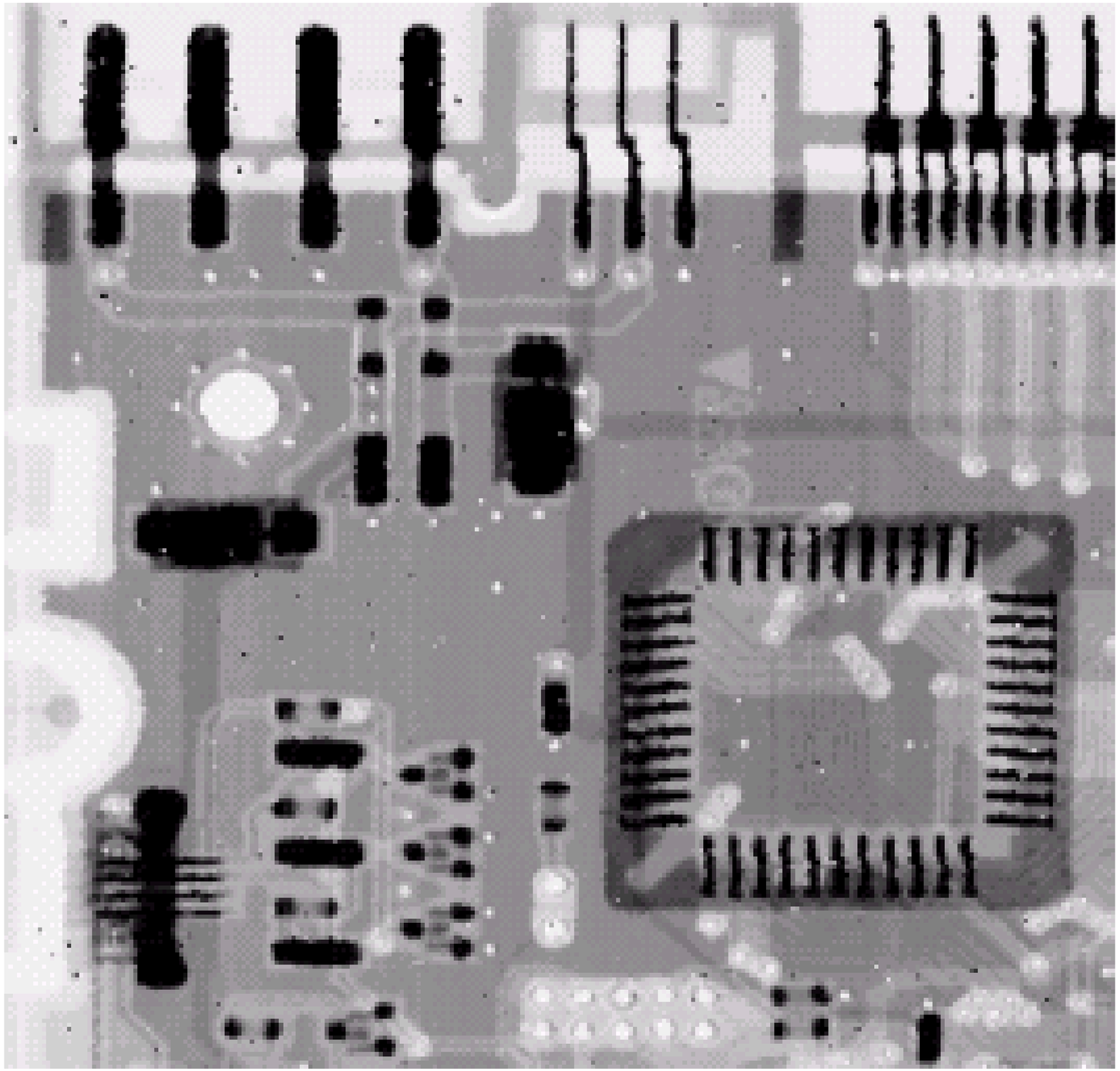
Original



Averaging Filter



Median
Filter



Sharpening Spatial Filters

Previously we have looked at smoothing filters which remove fine detail

Sharpening spatial filters seek to highlight fine detail

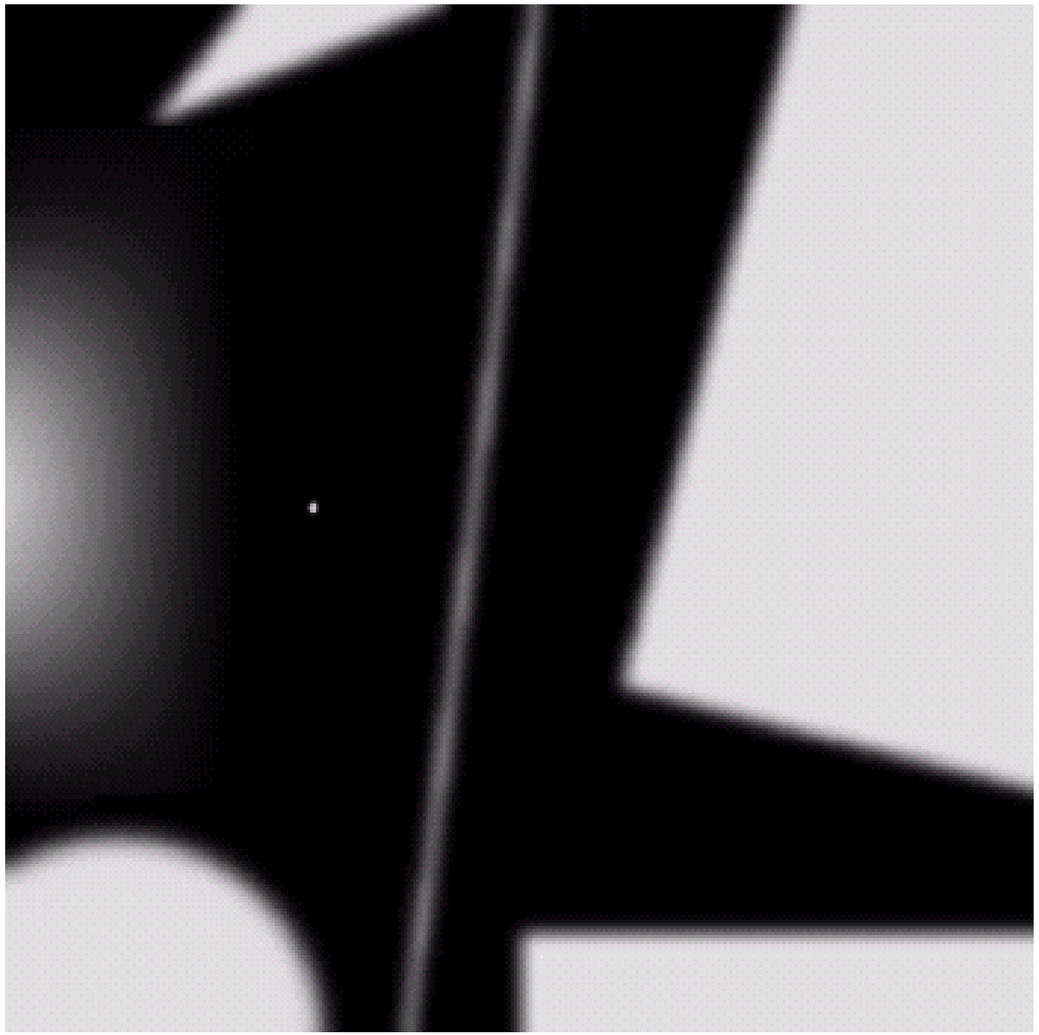
- Remove blurriness from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*

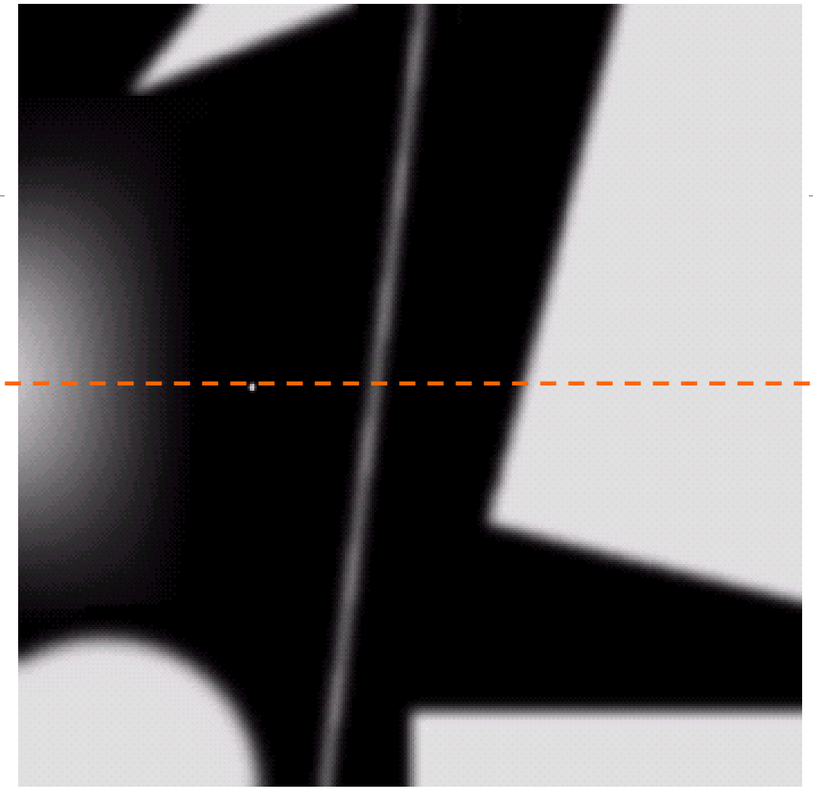
Spatial Differentiation

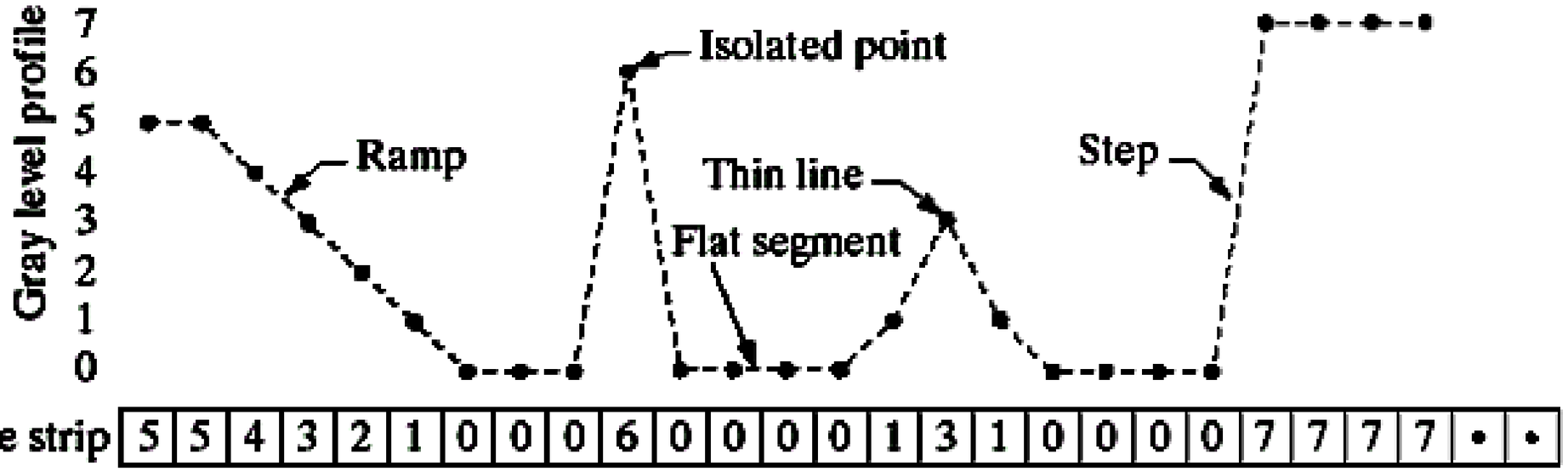
Differentiation measures the *rate of change* of a function

Let’s consider a simple 1 dimensional example



Spatial Differentiation





1st Derivative

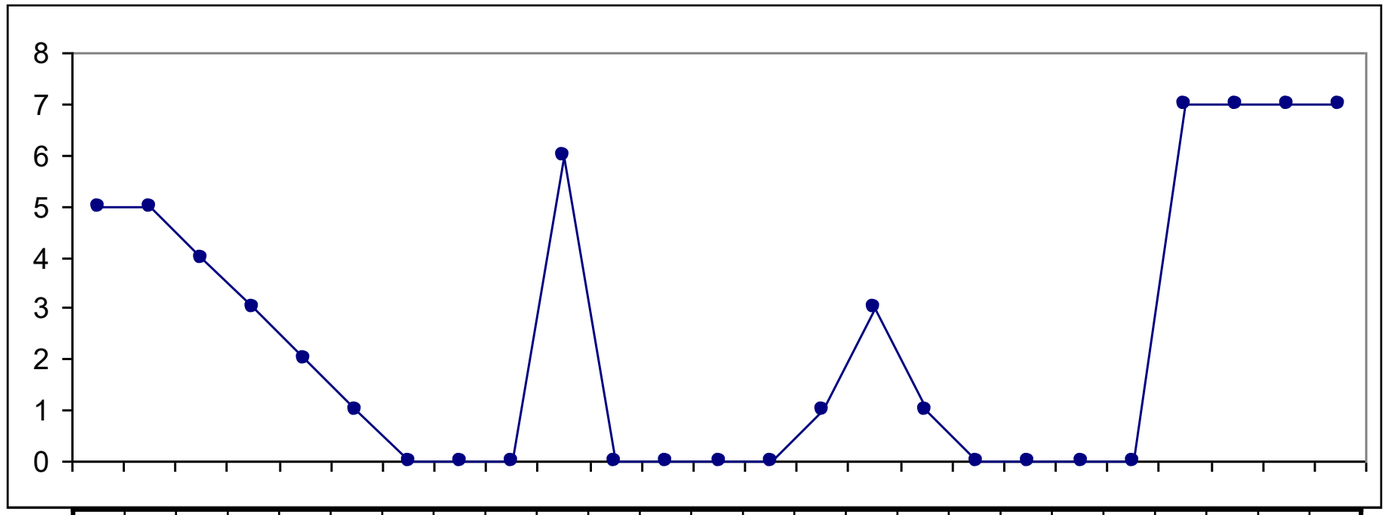
The formula for the 1st derivative of a function is as follows:

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

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

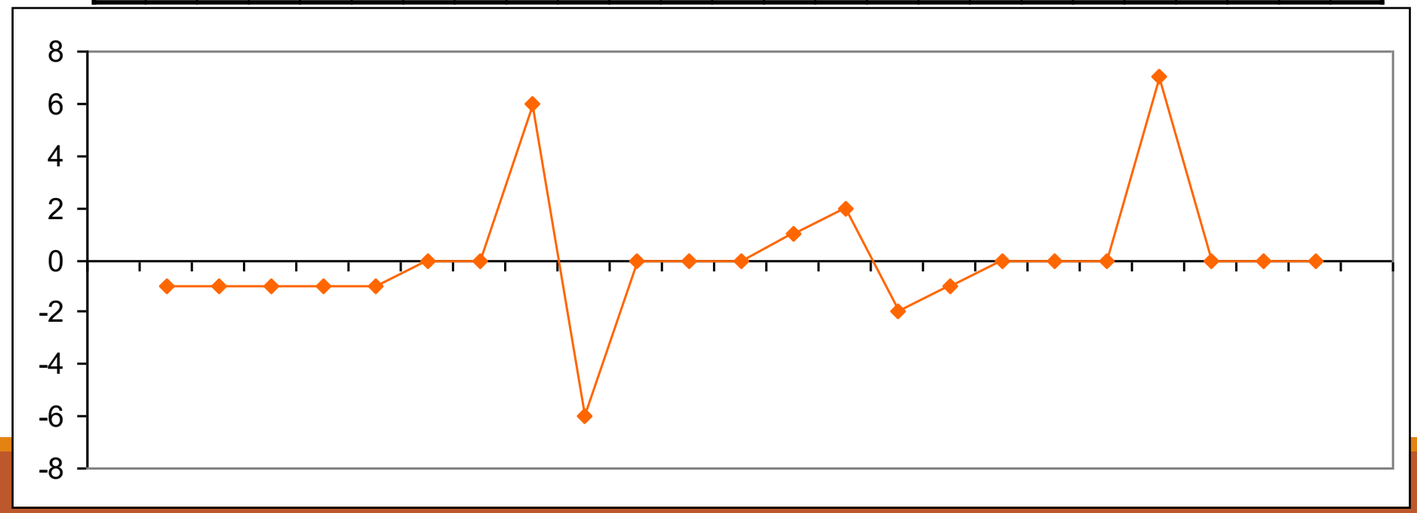
1st Derivative (cont…)

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

f'(x)



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

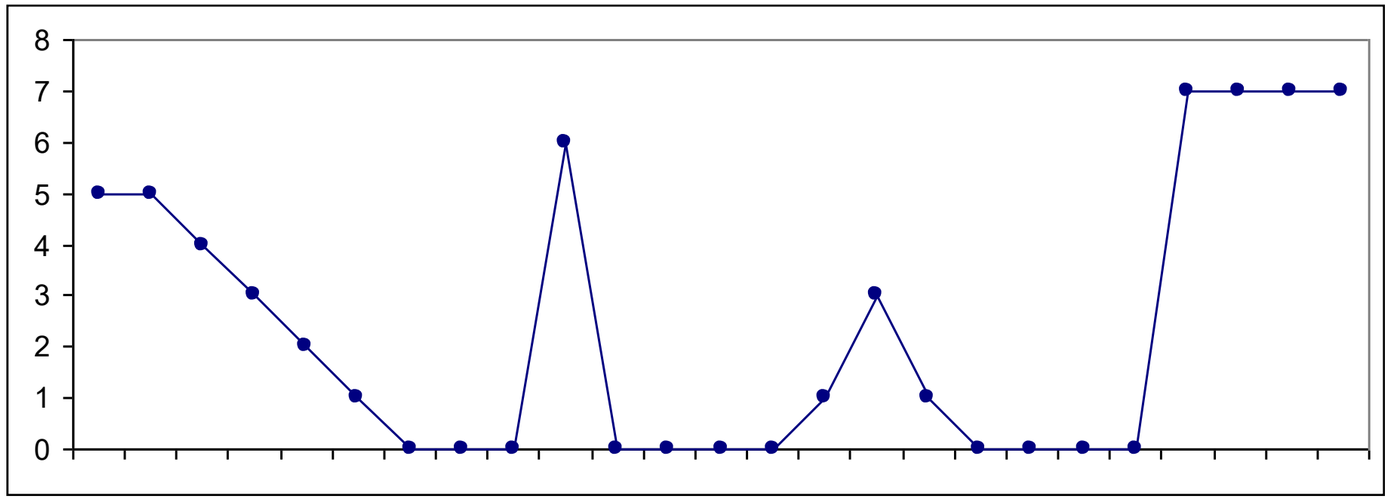
2nd Derivative

- The formula for the 2nd derivative of a function is as follows:

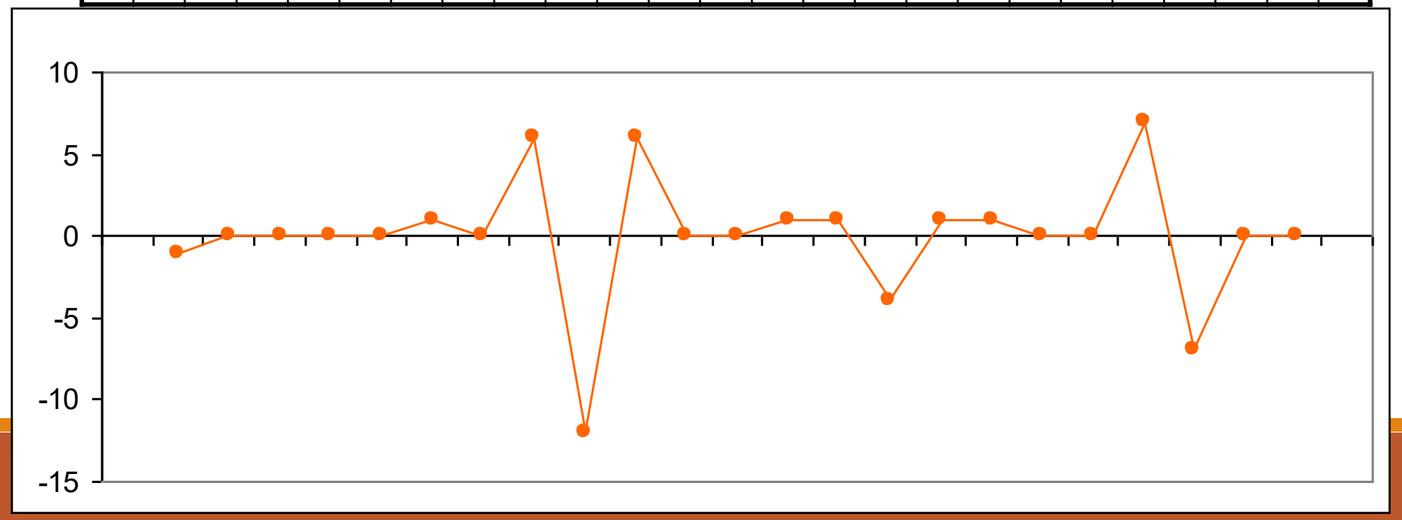
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

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

2nd Derivative (cont…)



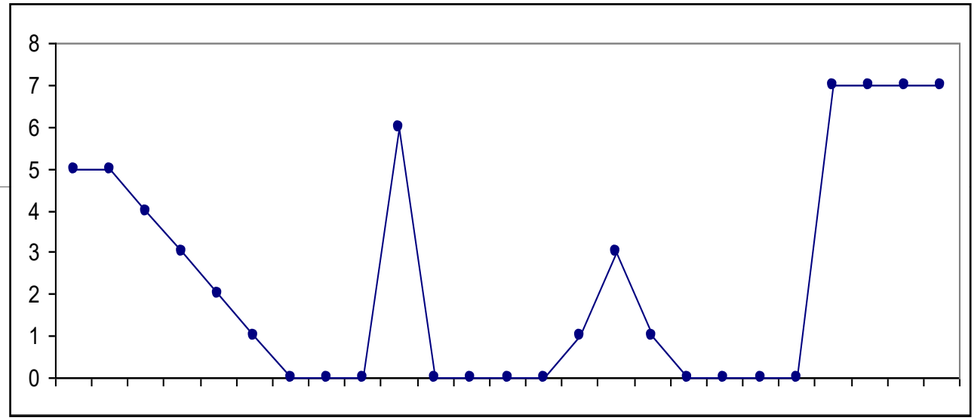
f(x)	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7
f'(x)	0	-1	-1	-1	-1	-1	0	0	6	-6	0	0	0	1	2	-2	-1	0	0	0	7	0	0	0	
f''(x)		-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	



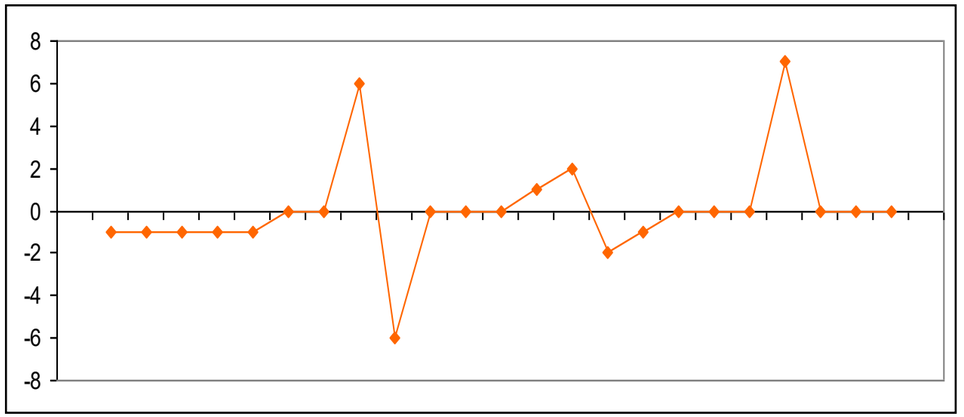
$$\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$$

1st and 2nd Derivative

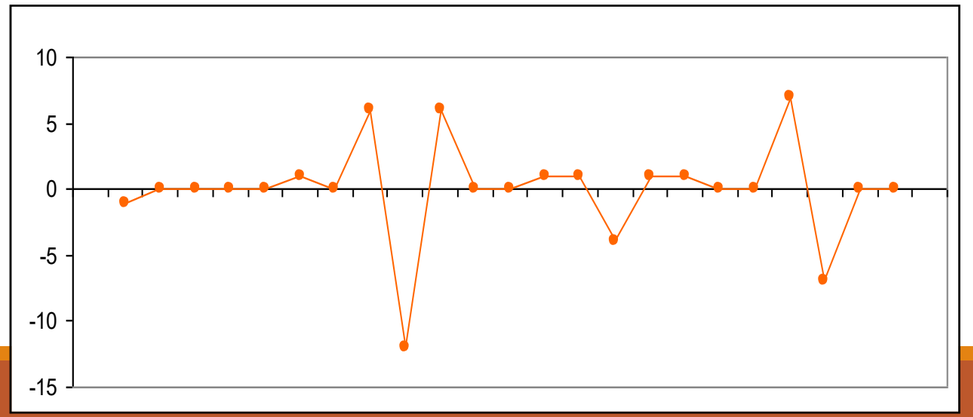
f(x)



f’(x)



f’’(x)



Using Second Derivatives For Image Enhancement

- The 2nd derivative is more useful for image enhancement than the 1st derivative
  - Stronger response to fine detail

- The first sharpening filter we will look at is the *Laplacian*
  - Isotropic (Identical in all directions)
  - One of the simplest sharpening filters
  - We will look at a digital implementation

The Laplacian

- The Laplacian is defined as follows:

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

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

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

- and in the *y* direction as follows:

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

The Laplacian (cont…)

- So, the Laplacian can be given as follows:

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

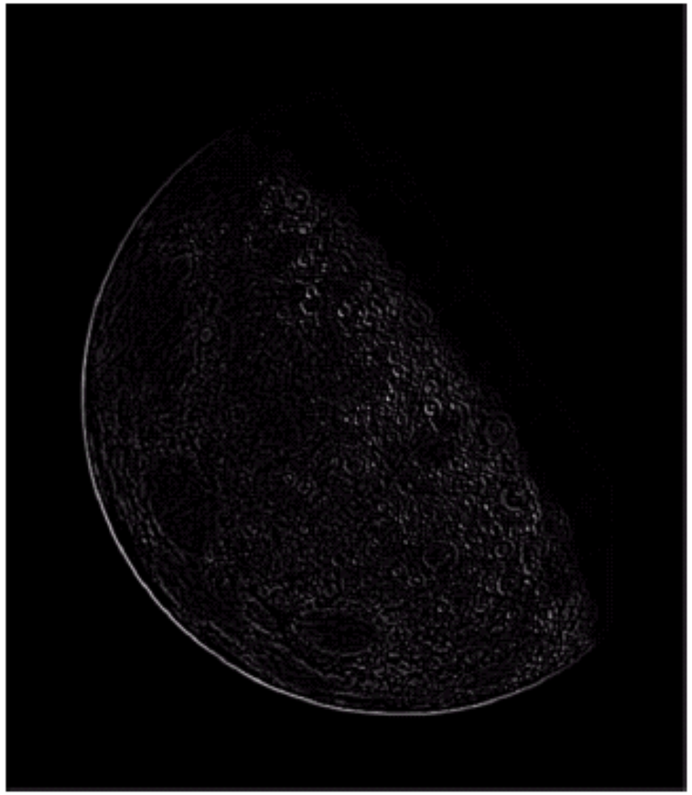
- We can easily build a filter based on this

0	1	0
1	-4	1
0	1	0

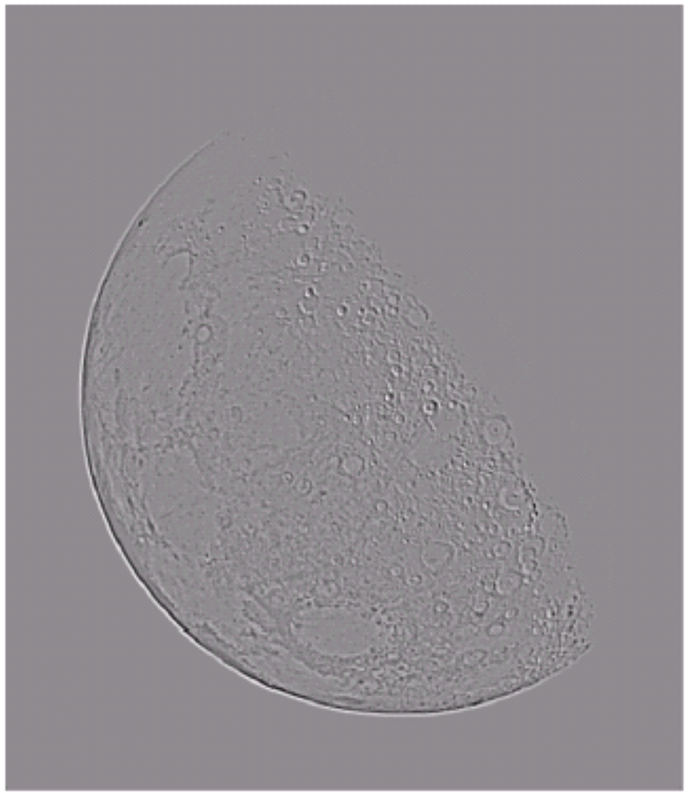
The Laplacian (cont…)

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

Original Image



Laplacian Filtered Image



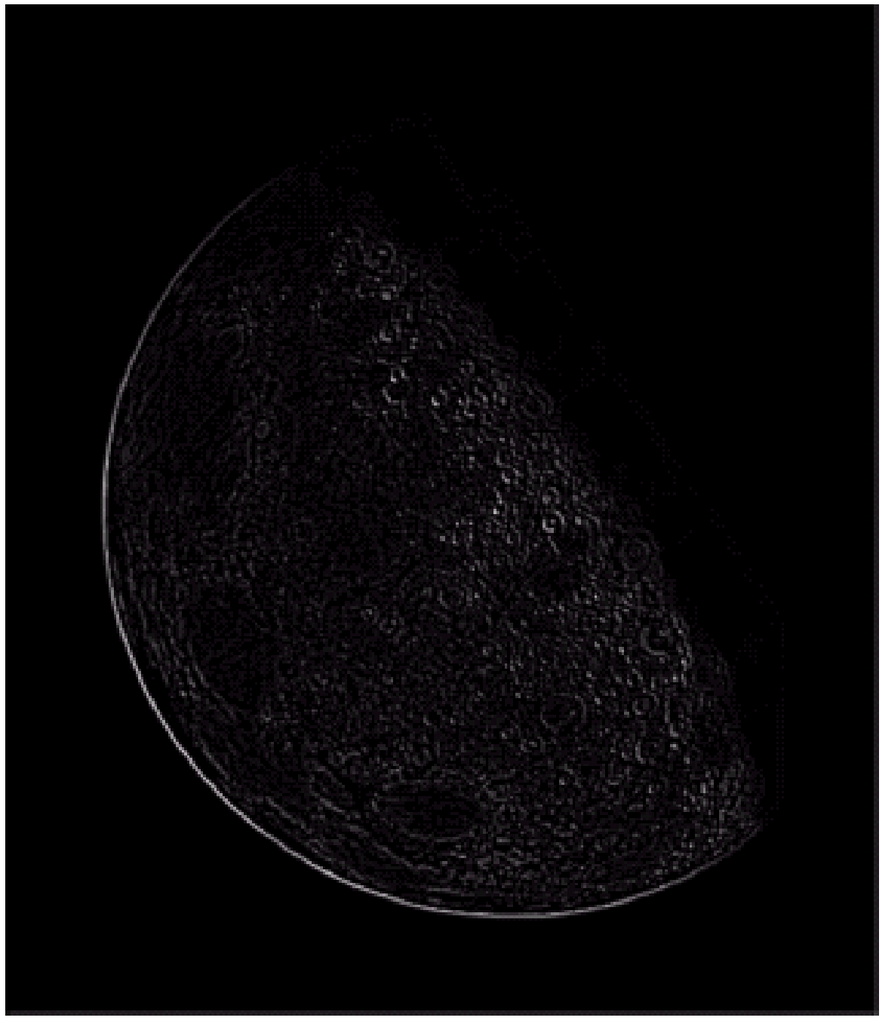
Laplacian Filtered Image Scaled for Display

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

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

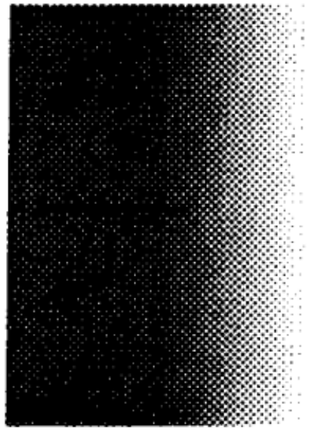


Laplacian Image Enhancement

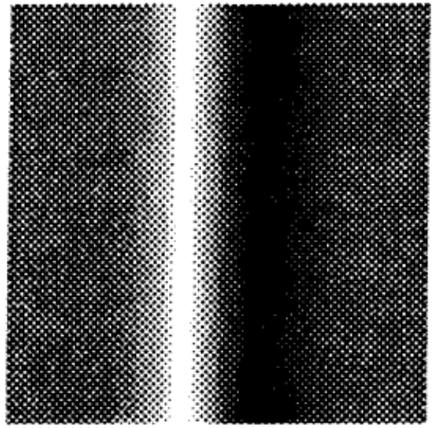
In the final sharpened image edges and fine detail are much more obvious

Laplacian Image Enhancement

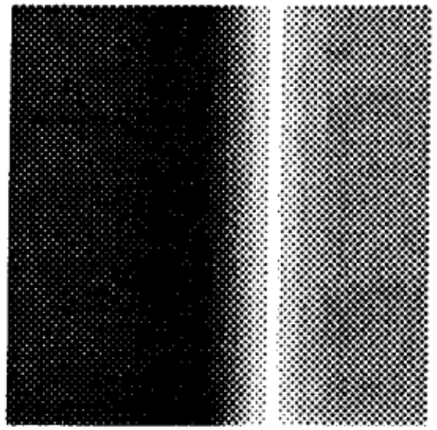
Ramp Edge Image



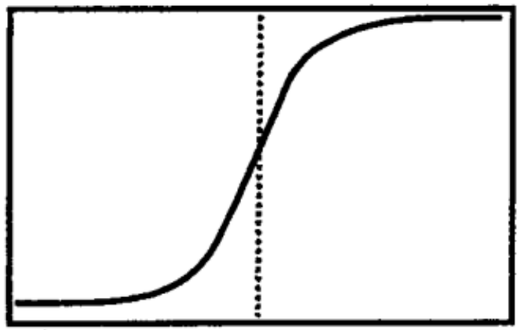
Laplacian Applied to Ramp Edge



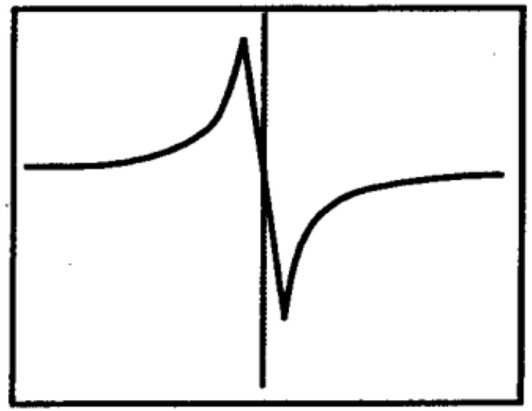
Result after Subtracting Laplacian from Original Image



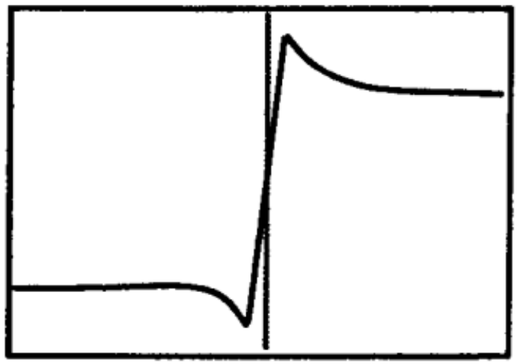
Profile of Ramp Edge



Profile of Laplacian Result



Profile of Enhanced Edge



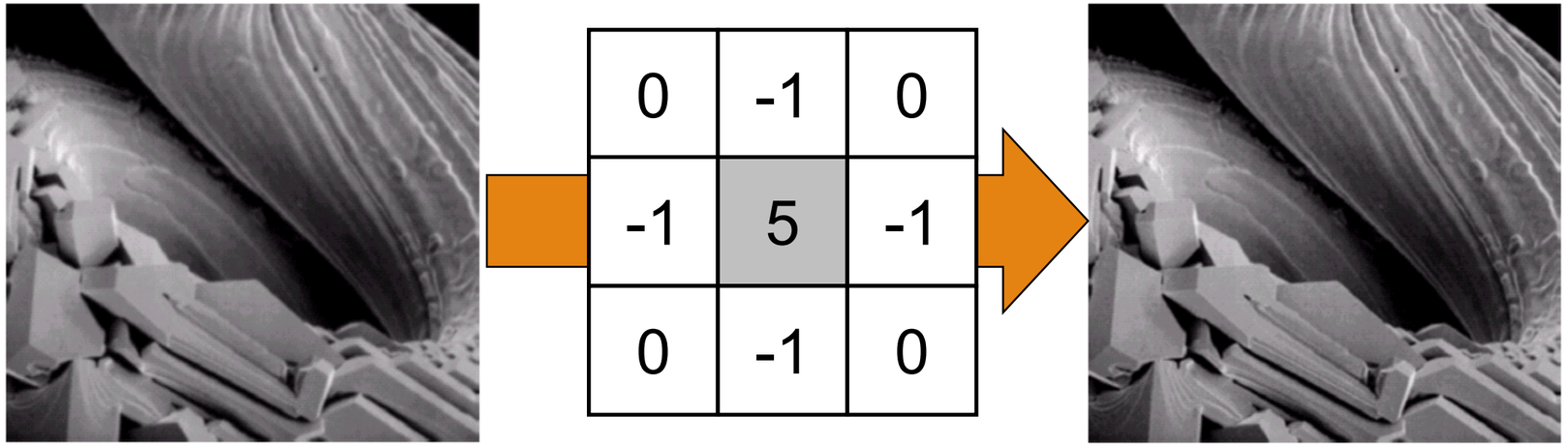
Simplified Image Enhancement

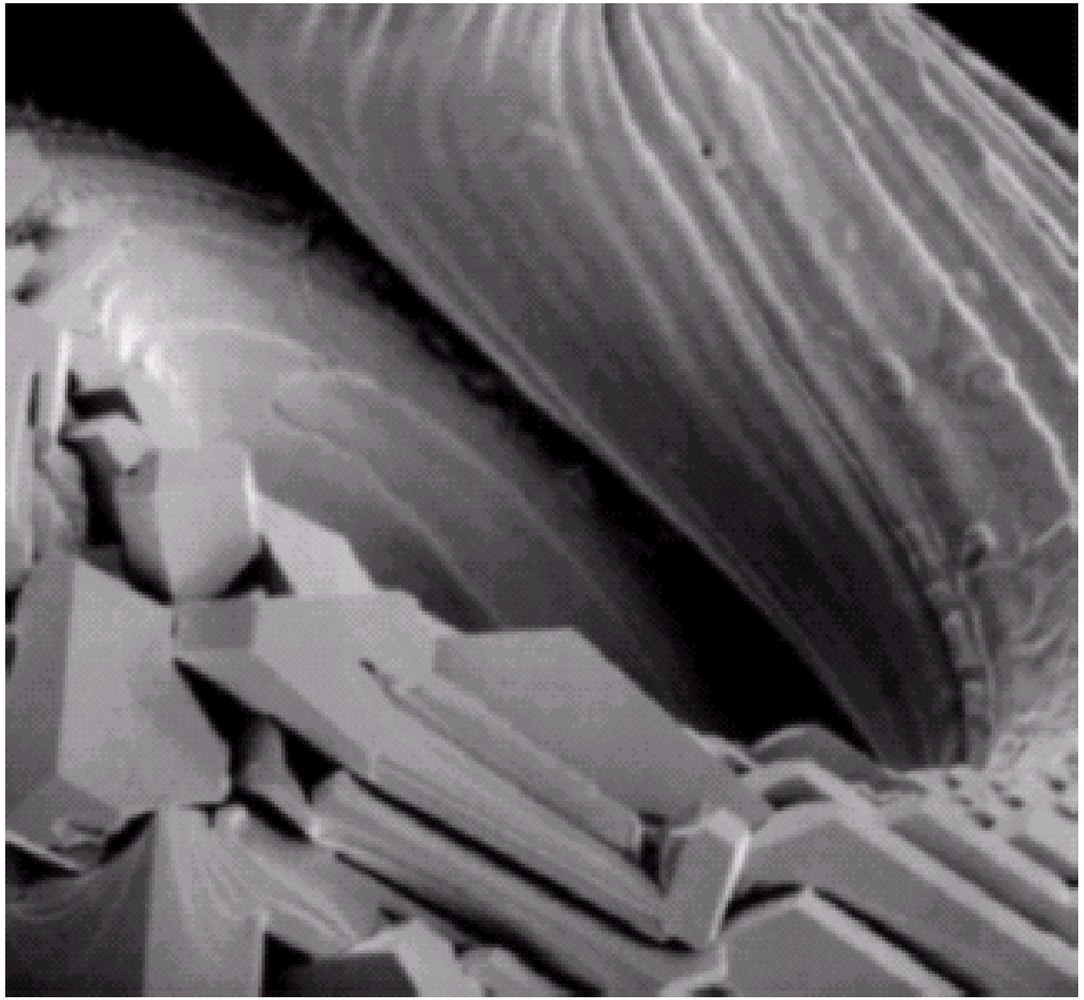
The entire enhancement can be combined into a single filtering operation

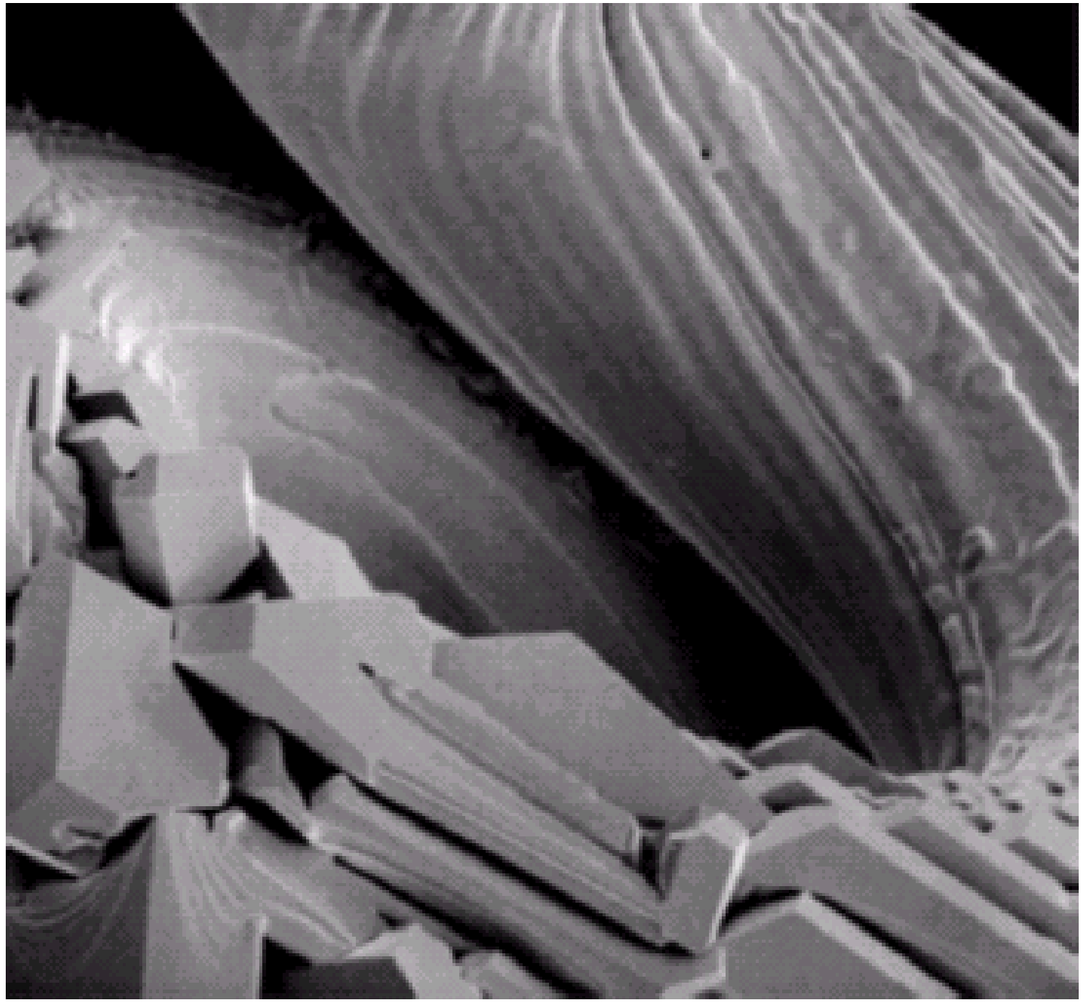
$$
\begin{aligned}
g(x,y) &= f(x,y) - \nabla^2 f \\
&= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\qquad + f(x,y+1) + f(x,y-1) \\
&\qquad - 4f(x,y)] \\
&= 5f(x,y) - f(x+1,y) - f(x-1,y) \\
&\qquad - f(x,y+1) - f(x,y-1)
\end{aligned}
$$

Simplified Image Enhancement (cont...)

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







Variants On The Simple Laplacian

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

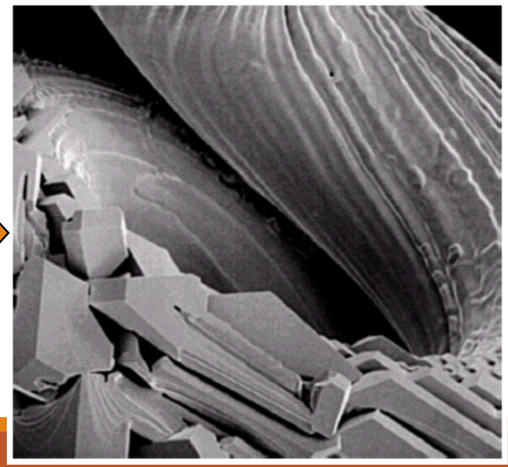
0	1	0
1	-4	1
0	1	0

Simple Laplacian

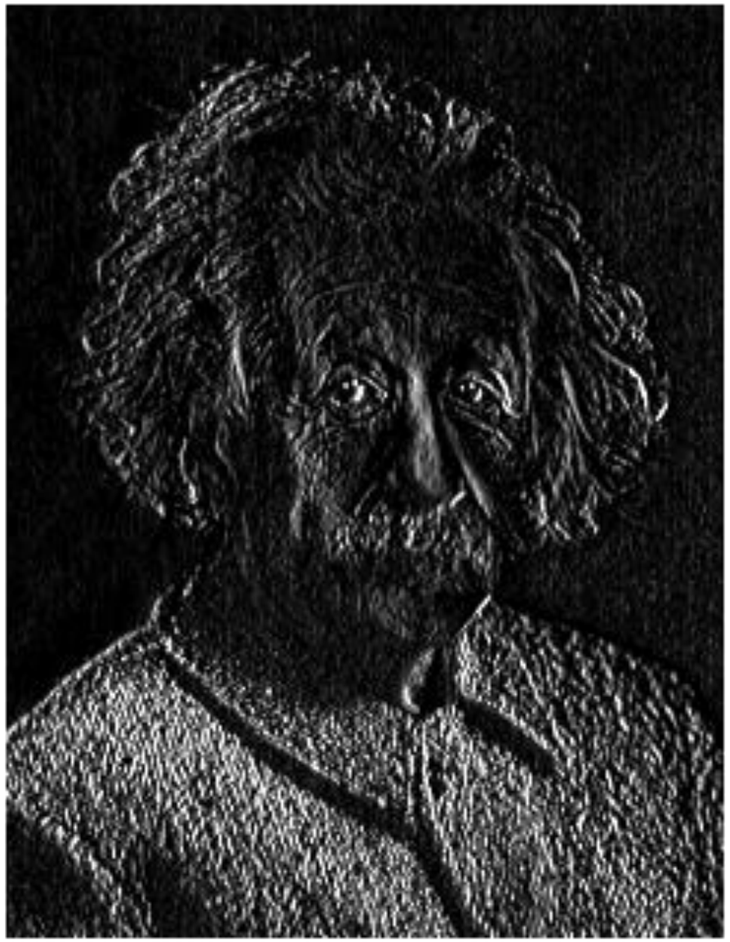
1	1	1
1	-8	1
1	1	1

Variant of Laplacian

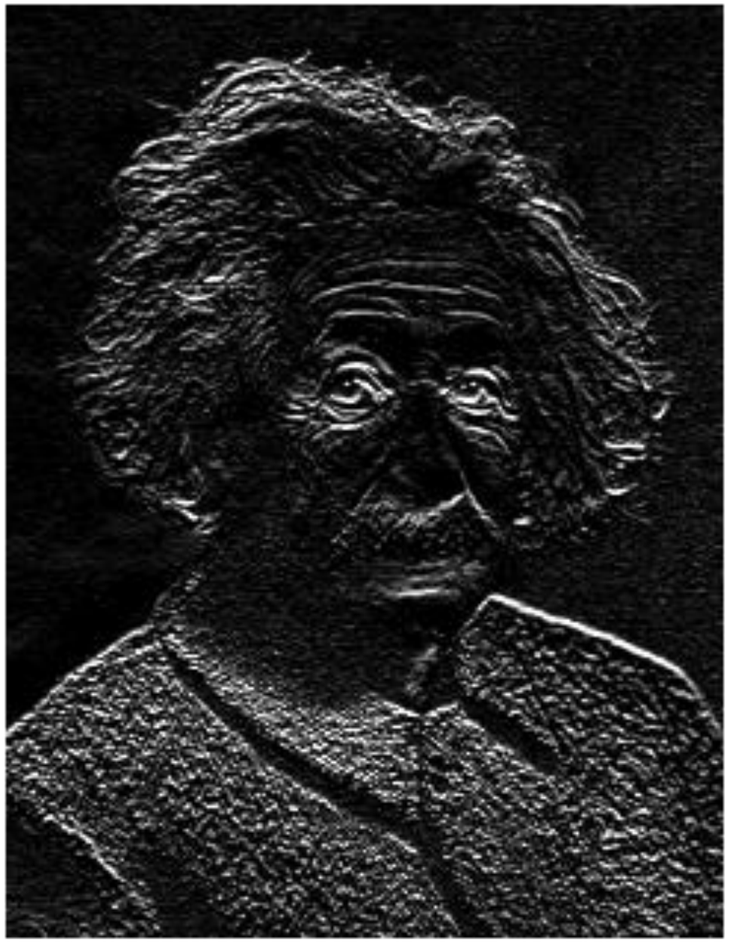
-1	-1	-1
-1	9	-1
-1	-1	-1



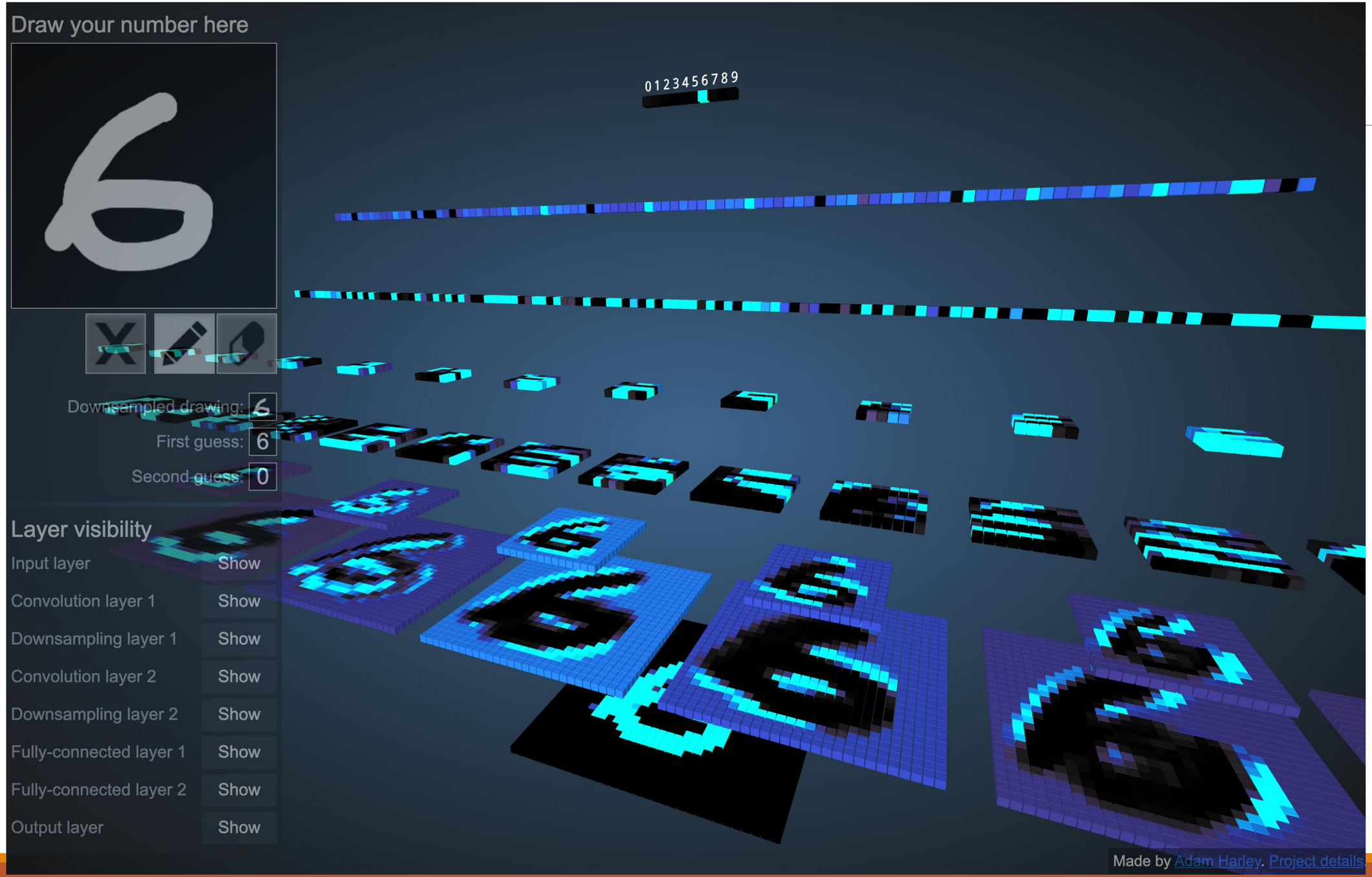
x-derivative



y-derivative



Laplacian (x and y)



https://adamharley.com/nn_vis/cnn/2d.html