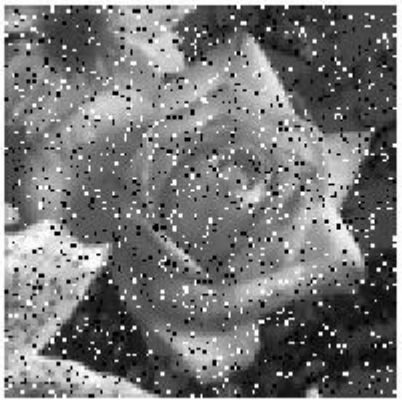
Spatial Linear Filtering - Why?

- Image processing
- Remove noise from images (e.g. poor transmission (from space), measurement (X-Rays))

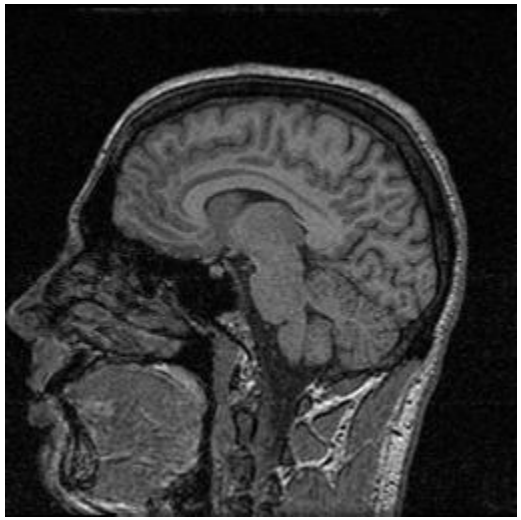
Original Image

Corrupted Image

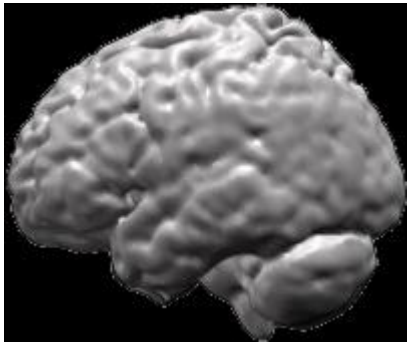


Filtered Image

Edge Detection

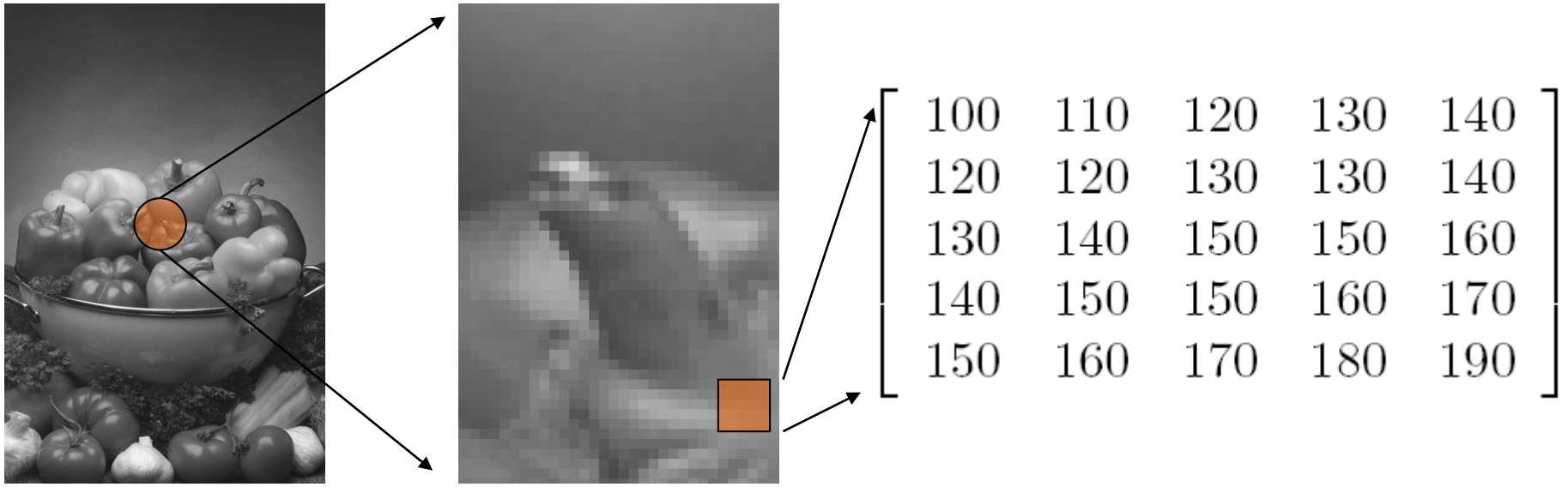


- e.g. could digitally extract brain for display
- could check if machined parts are correct



How does it work?

- Lets use a grey image for ease
- Each pixel has a single value between 0 (black) and 255 (white)



$$\begin{bmatrix} 100 & 110 & 120 & 130 & 140 \\ 120 & 120 & 130 & 130 & 140 \\ 130 & 140 & 150 & 150 & 160 \\ 140 & 150 & 150 & 160 & 170 \\ 150 & 160 & 170 & 180 & 190 \end{bmatrix}$$

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1

$$\begin{bmatrix} 100 & 110 & 120 & 130 & 140 \\ 120 & 120 & 130 & 130 & 140 \\ 130 & 140 & 150 & 150 & 160 \\ 140 & 150 & 150 & 160 & 170 \\ 150 & 160 & 170 & 180 & 190 \end{bmatrix}$$

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1

100	110	120	130	140
120	120	130	130	140
130	140	150	150	160
140	150	150	160	170
150	160	170	180	190

Sum=1x100+3x110+1x120+
3x120+9x120+3x130+
1x130+3x140+1x150
= 3080

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1

100	110	120	130	140
120	120	130	130	140
130	140	130	150	160
140	150	150	160	170
150	160	170	180	190

1	3	1
3	9	3
1	3	1

Sum=1x110+3x120+1x130+
3x120+9x130+3x130+
1x140+3x150+1x150
= 3260

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1

100	110	120	130	140
120	120	130	190	140
130	140	150	130	160
140	150	150	160	170
150	160	170	180	190

Sum=1x120+3x130+1x140+
3x130+9x130+3x140+
1x150+3x150+1x160
= 3390

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1

100	110	120	130	140
120	130	130	130	140
130	140	150	150	160
140	150	150	160	170
150	160	170	180	190

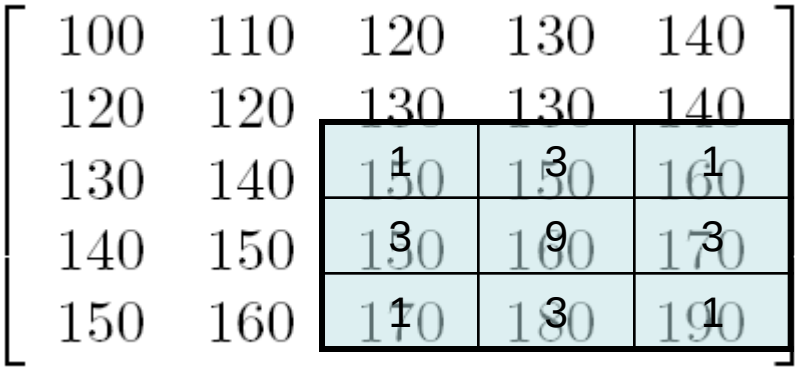
1	3	1
3	9	3
1	3	1

Sum=1x120+3x120+1x130+
3x130+9x140+3x150+
1x140+3x150+1x150
= 3450

Filter Kernel

- Now select a filter kernel
- e.g. Gaussian blur (removes noise)
- For each pixel in the image
- Can the kernel be centred over the pixel?
- If so, calculate the sum of the products

1	3	1
3	9	3
1	3	1



Result

*	*	*	*	*
*	3080	3260	3390	*
*	3450	3620	3740	*
*	3720	3870	4060	*
*	*	*	*	*

Cross Correlation

In principle, given a 3×3 subimage of pixels:

$$I_{ij} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}$$

and a 3×3 filter kernal:

$$M_{ij} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

operating on pixel p_{22}, the new value p'_{22} is given by:

$$p'_{22} = m_{11}p_{11} + m_{12}p_{12} + \cdots + m_{32}p_{32} + m_{33}p_{33}$$

*What about the edges?

- * indicates filter could not be centred
- 3x3 = 1 pixel edge all the way around
- 5x5 = 2 pixels edge all the way around
- nxm = ? Exercise!

*	*	*	*	*
*	3080	3260	3390	*
*	3450	3620	3740	*
*	3720	3870	4060	*
*	*	*	*	*

*What about the edges?

- Reduce size of image – but could be counter intuitive for users.
- Put a single colour border or pad the image beforehand.
- Use old pixel values (not sensible for e.g. edge detectors)
- Use known neighbours colour.
- Use some fudge (e.g. have some filters reduced in size so they can fit against the edges when needed)

*	*	*	*	*
*	3080	3260	3390	*
*	3450	3620	3740	*
*	3720	3870	4060	*
*	*	*	*	*

Aren't those numbers too big?

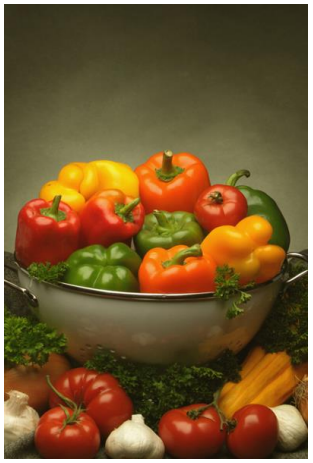
- Yes, they are no longer in the range 0-255
- We could have negative numbers (for a high pass filter)
- So we need to map the new numbers onto our range 0-255
- First calculate the max and min values
- max=4060, min=3080
- Then for each intermediate value I, calculate **I'=((I-min)*255)/(max-min)**

*	*	*	*	*
*	0	47	81	*
*	96	141	172	*
*	167	206	255	*
*	*	*	*	*

Normalisation

- The last step:
- **I’=((I-min)*255)/(max-min)**
- Is called *Normalisation*
- As used, it maps pixels values to the range 0..255. Pixels could be mapped to other ranges (e.g. You might have a 12 bit display, or you might want to map back to the same range the image currently uses)

And colour?



Calculate sum of products for each colour channel independently
Normalise as before, but replace red with new red, etc.

140	130	110	128	130
120	190	130	120	120
120	130	110	140	145
147	168	165	162	160
150	170	167	167	165

120	130	110	108	110
110	95	87	80	78
90	85	82	82	81
80	82	85	86	89
92	95	100	101	102

14	35	11	12	13
32	91	30	12	12
12	33	15	14	14
20	21	22	23	26
19	18	16	20	21

What about the filters?

1	1	1
1	1	1
1	1	1

- Low pass 3x3 box filter (nxm box filter has size nxm with all 1's). The bigger the filter, the larger the blur (and time – think about that)

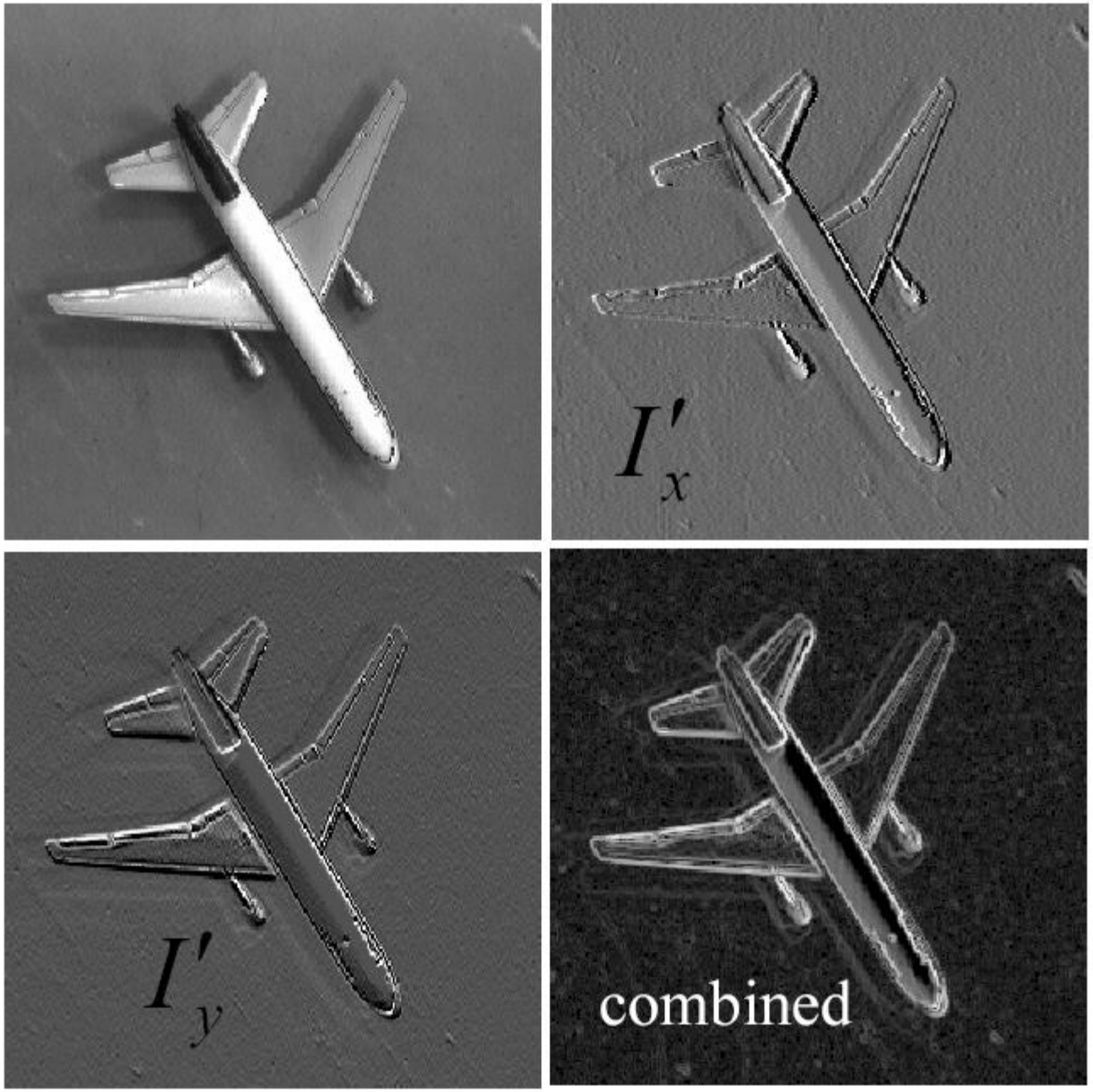
-1	-1	-1
-1	8	-1
-1	-1	-1

- High pass filter (like *sharpen* in Photoshop

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

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

- Sobel edge detectors – X-direction and Y-direction


I'_x
I'_y
combined

Low Pass Filter - Gaussian

7x7

1	4	7	10	7	4	1
4	12	26	33	26	12	4
7	26	55	71	55	26	7
10	33	71	91	71	33	10
7	26	55	71	55	26	7
4	12	26	33	26	12	4
1	4	7	10	7	4	1

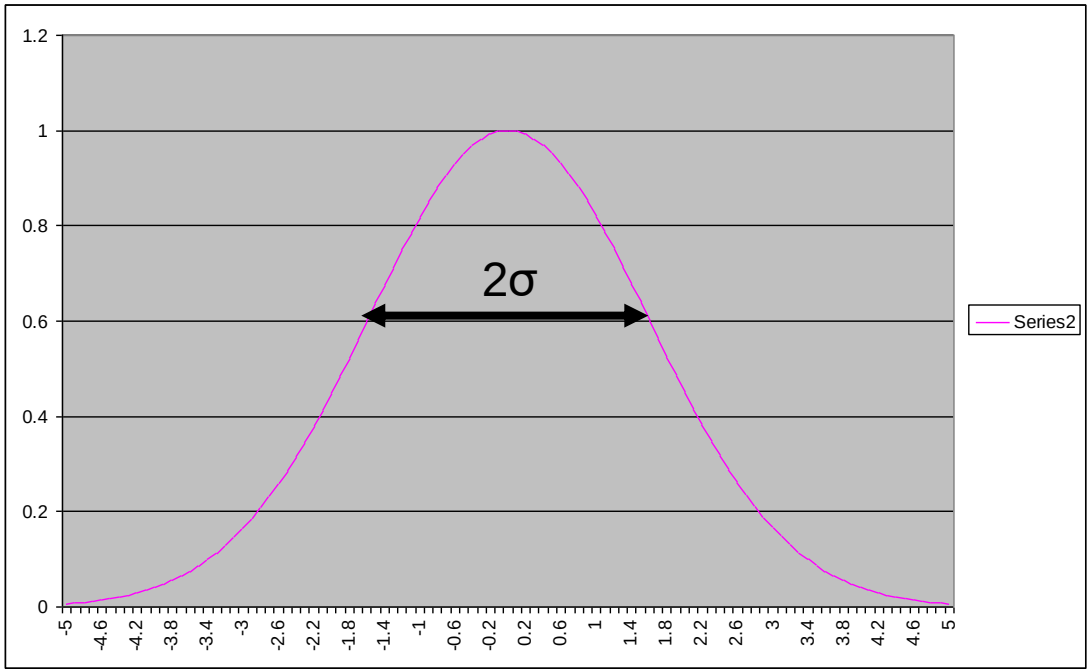
3x3

1	3	1
3	9	3
1	3	1

Where do these numbers come from?

Gaussian

- 1 dimensional
- $G(x)=e^{-(x^2/2\sigma^2)}$
- In Excel power(exp(1), -(x*x/2*σ*σ))



Gaussian

- 2D
- $G(x,y)=e^{-((x^2+y^2)/2\sigma^2)}$