COMP0026: Image Processing

Image Filtering

UCL

[Popular Science]

COMP0026: Image Processing

Image Filtering

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
imshow(I), title('Original Image')
figure, imshow(I2), title('Filtered Image')
```

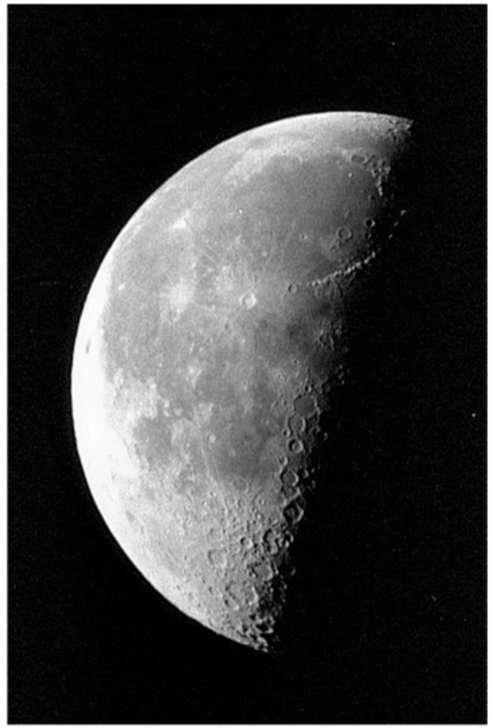
UCL

[Popular Science]

COMP0026: Image Processing

Image Filtering

```
I = imread('moon.tif');
h = fspecial('unsharp');
I2 = imfilter(I,h);
imshow(I), title('Original Image')
figure, imshow(I2), title('Filtered Image')
```



UCL

[Popular Science]

Lectures will be Recorded

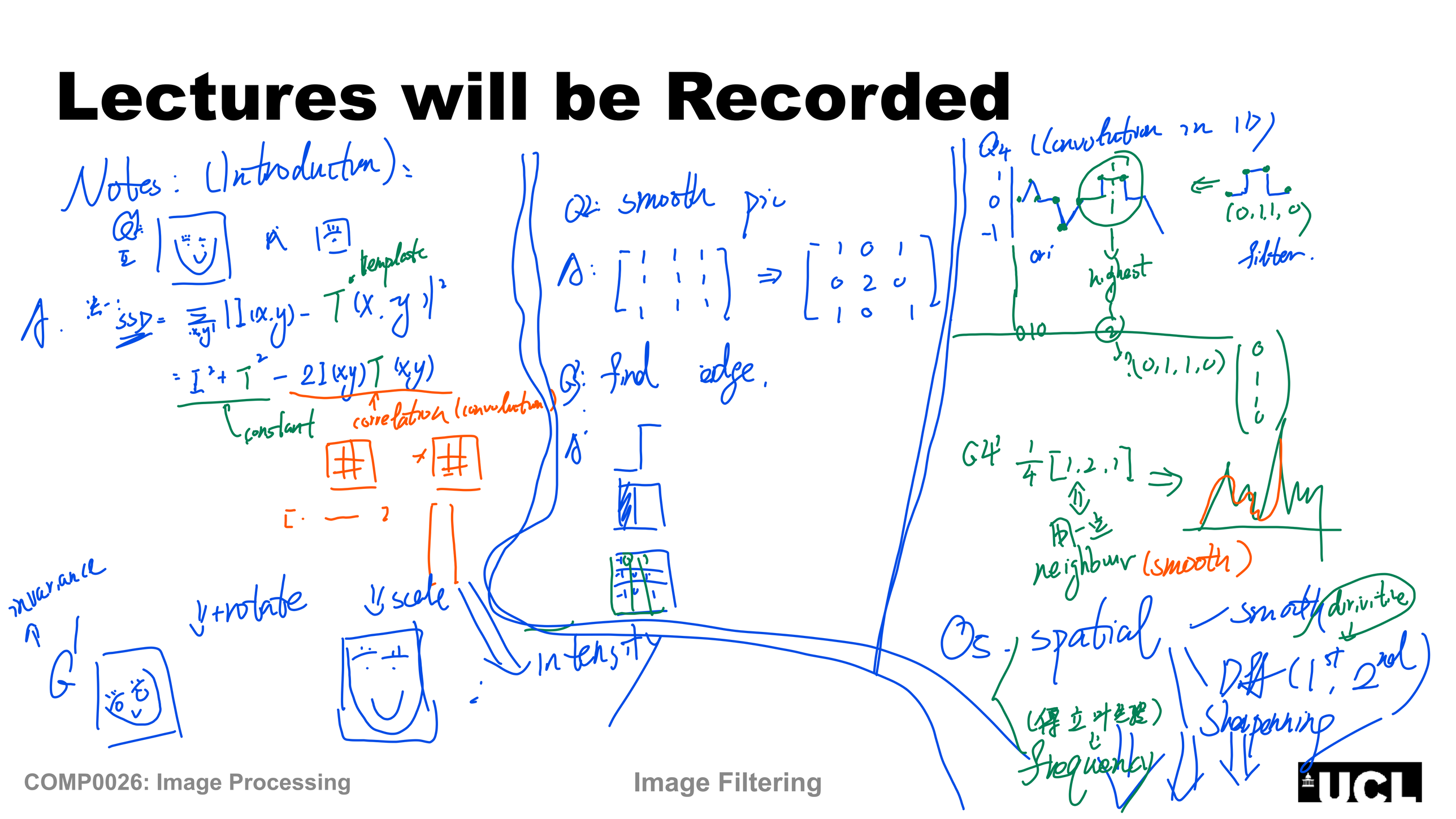


Image Filtering

input image ⟶ filter function ⟶ output image

Image Filtering

input image ⟶ filter function ⟶ output image

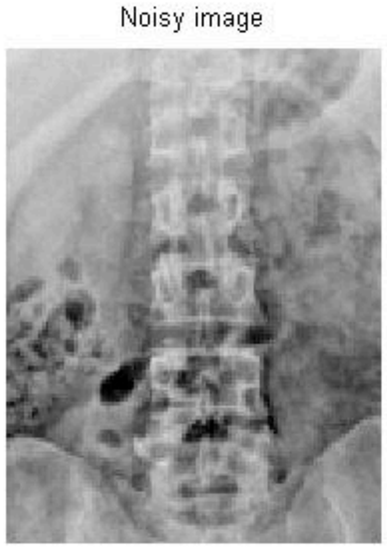
Functions
- **linear/non-linear**

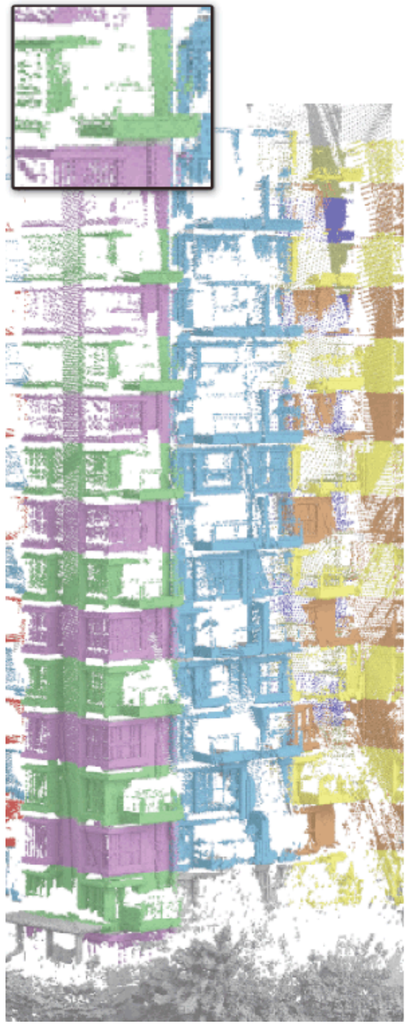
Neighborhoods
- **local/non-local**

Goals

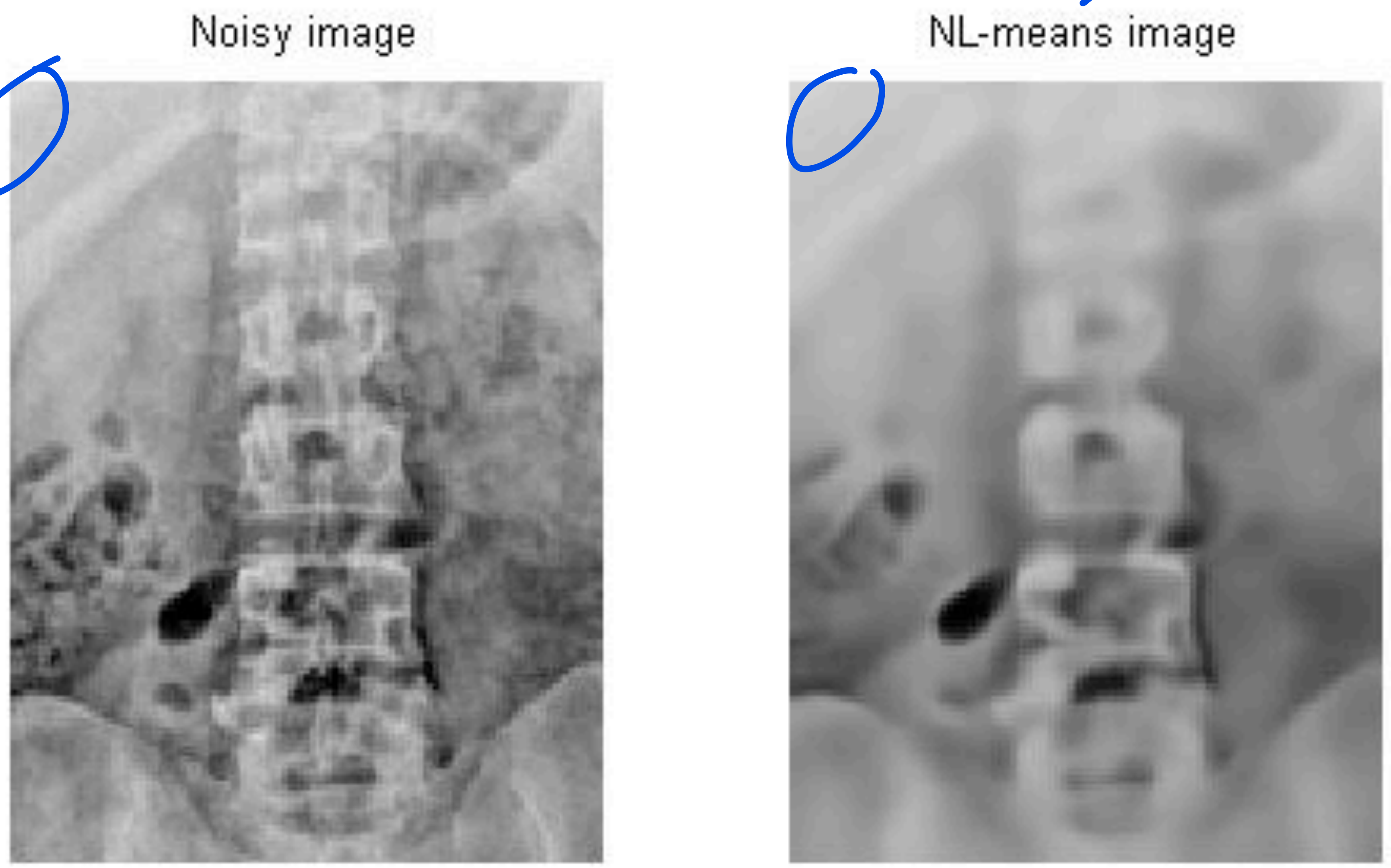
- Low-level data (e.g., pixel) processing operations
- Smoothing and noise reduction
- Feature extraction and enhancement

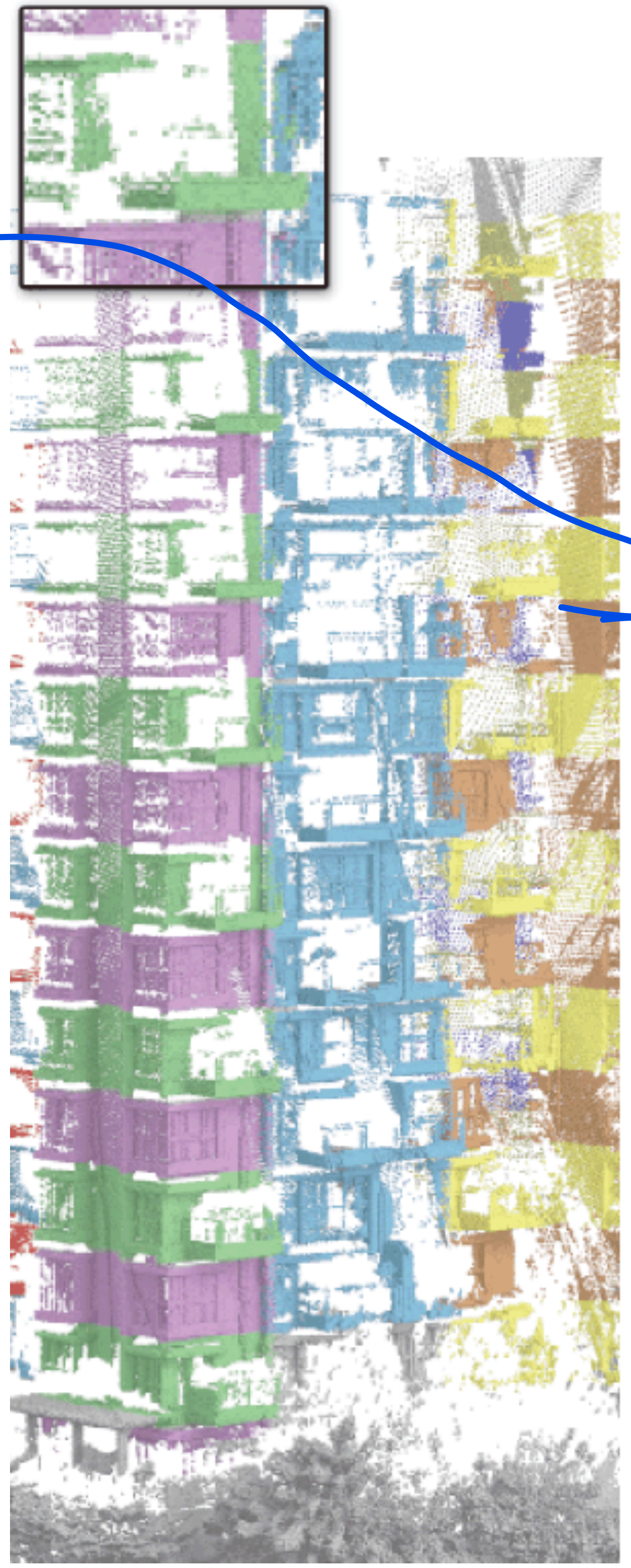
Non-local Filtering





Non-local Filtering



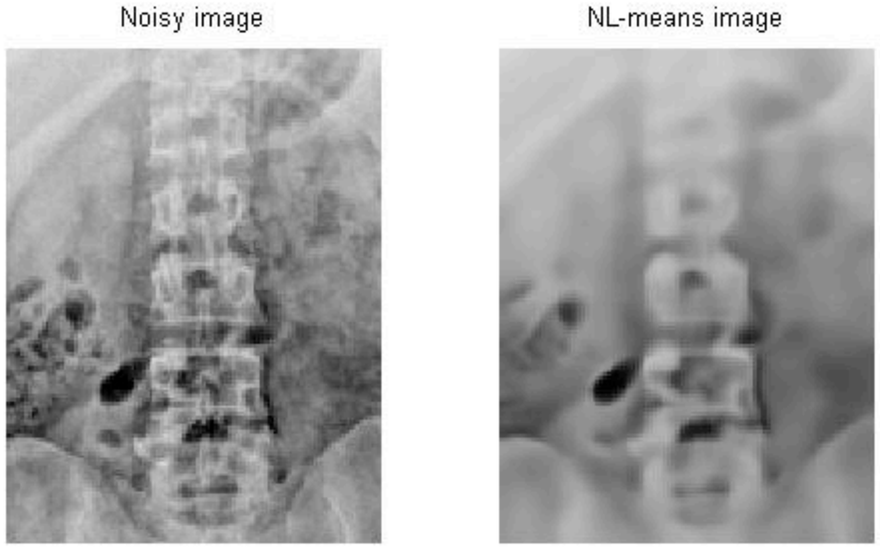


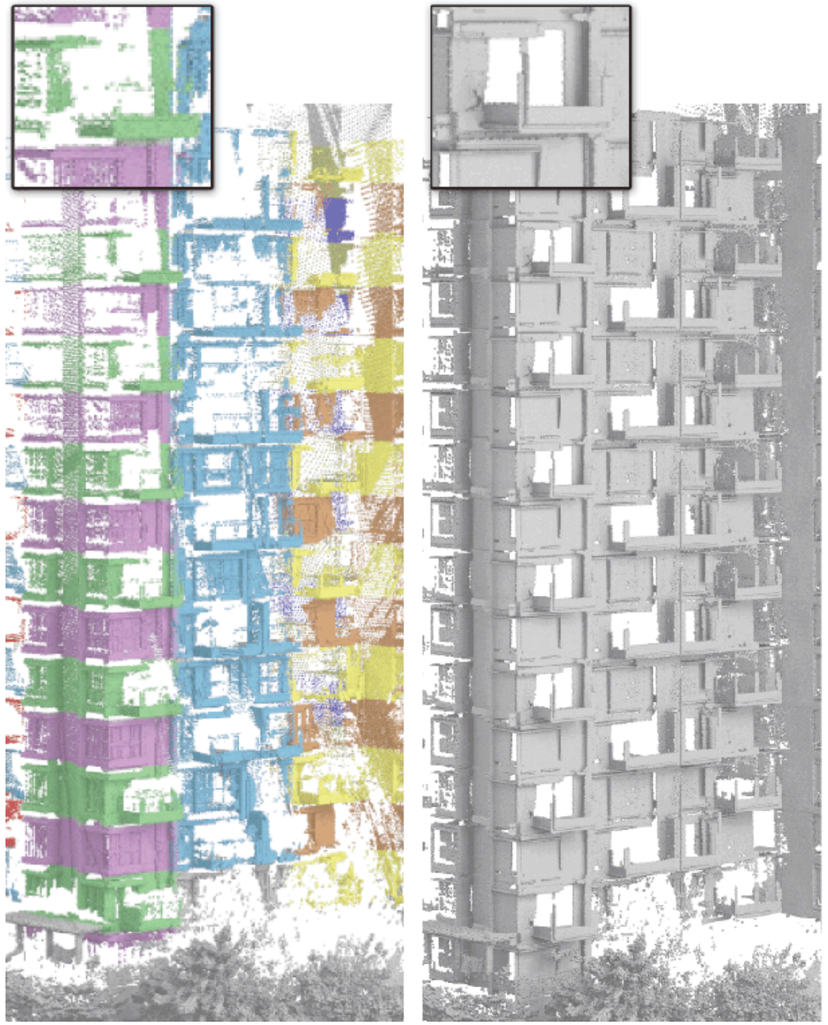
correlation
(how similar)
(or 和 convolution 不同的概念)

look all the pixels

对每个 □ (像素块)
在全图中找 similar 的 □
加 weight 去 denoise
↑
根据 similarity 多少决定

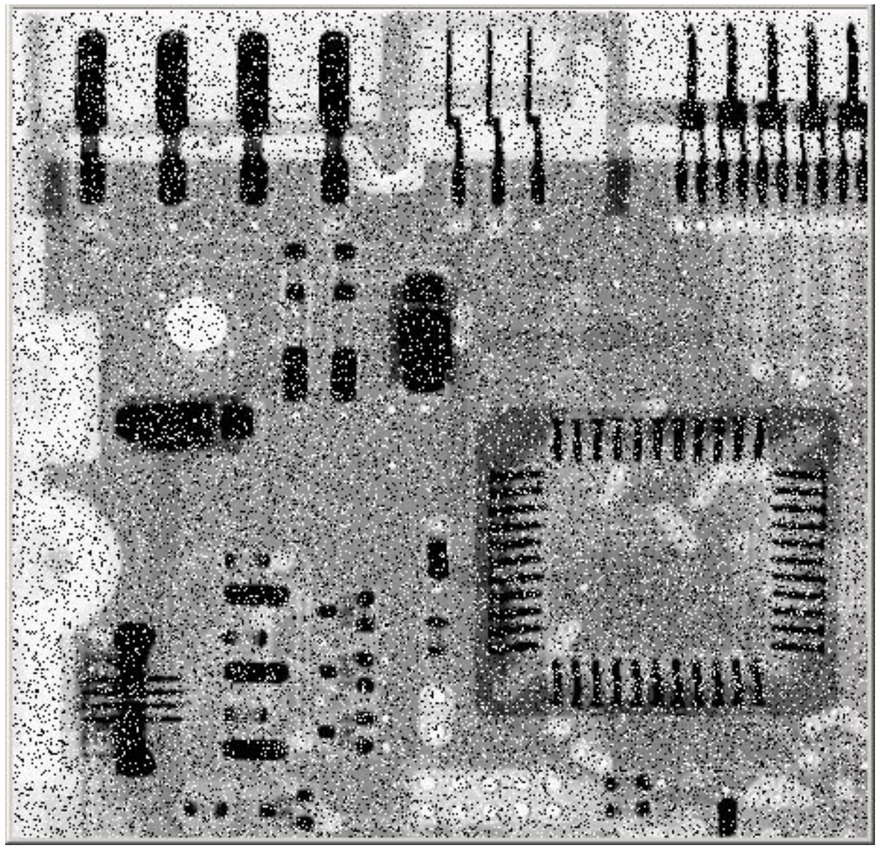
Non-local Filtering

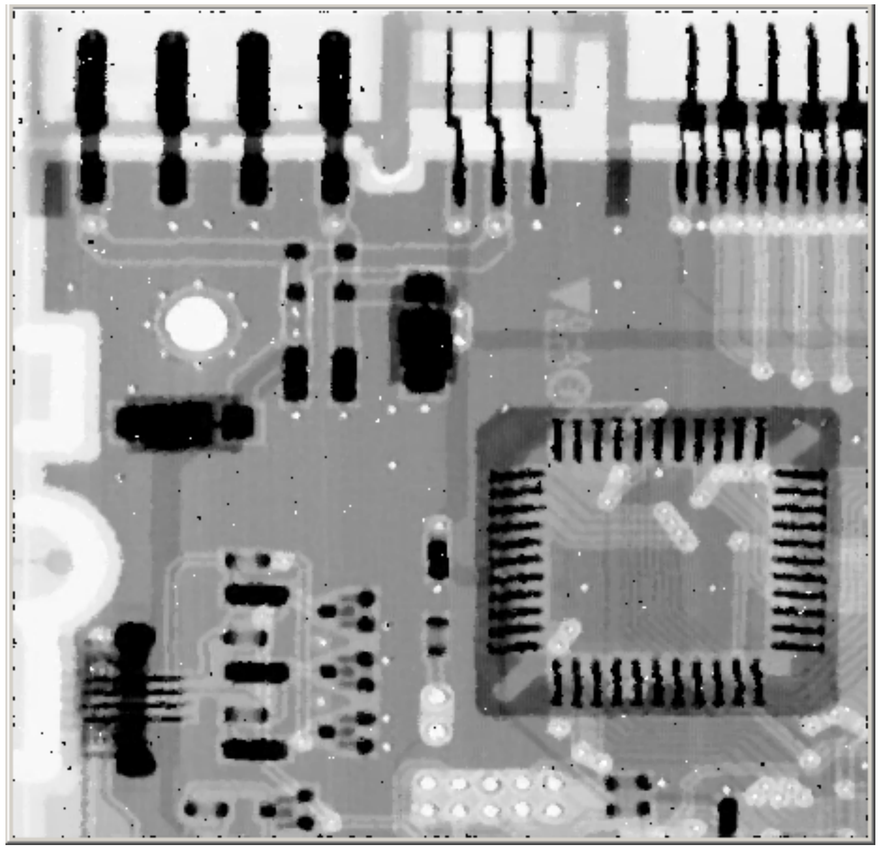




Example: Remove “Noise”

(if you know how to model that noise specifically!)





median filter

Input image from DIP by Gonzalez+Woods, 2008

Outline

- Linear filtering
- Convolution operations
  - Smoothing
  - Low-pass and high-pass filters
  - Sharpen
- Filtering in the Fourier domain
- Rank filters

Linear Mappings

- **Equivalently, *L* is *linear* if**

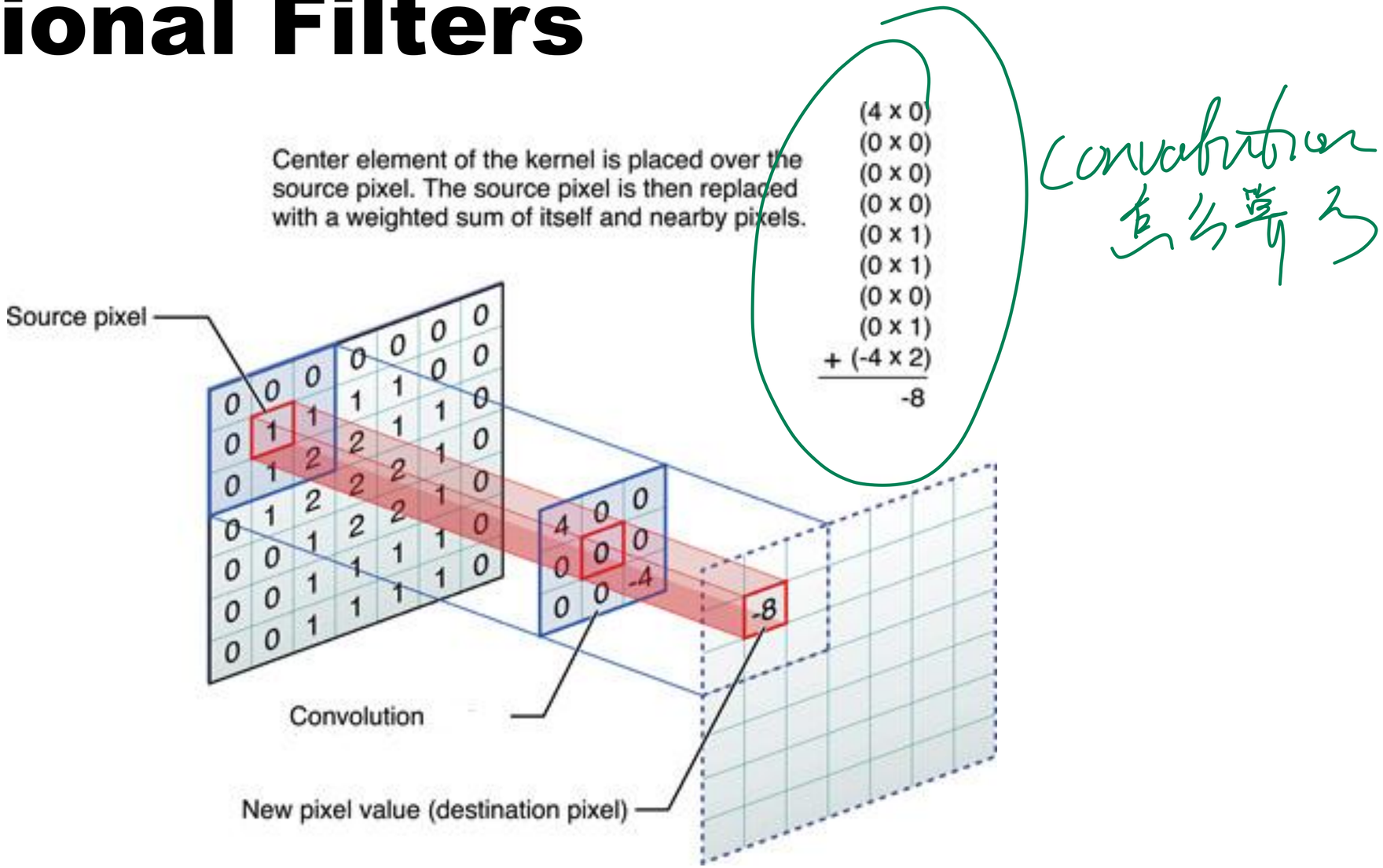
$$L(I_1 + I_2) = L(I_1) + L(I_2)$$

$$L(\alpha I) = \alpha L(I)$$

(i.e., *L* preserves linear combinations)

$$L(\alpha I_1 + \beta I_2) = \alpha L(I_1) + \beta L(I_2)$$

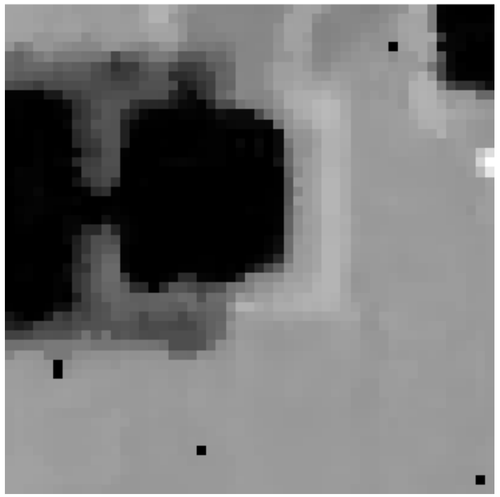
Convolutional Filters



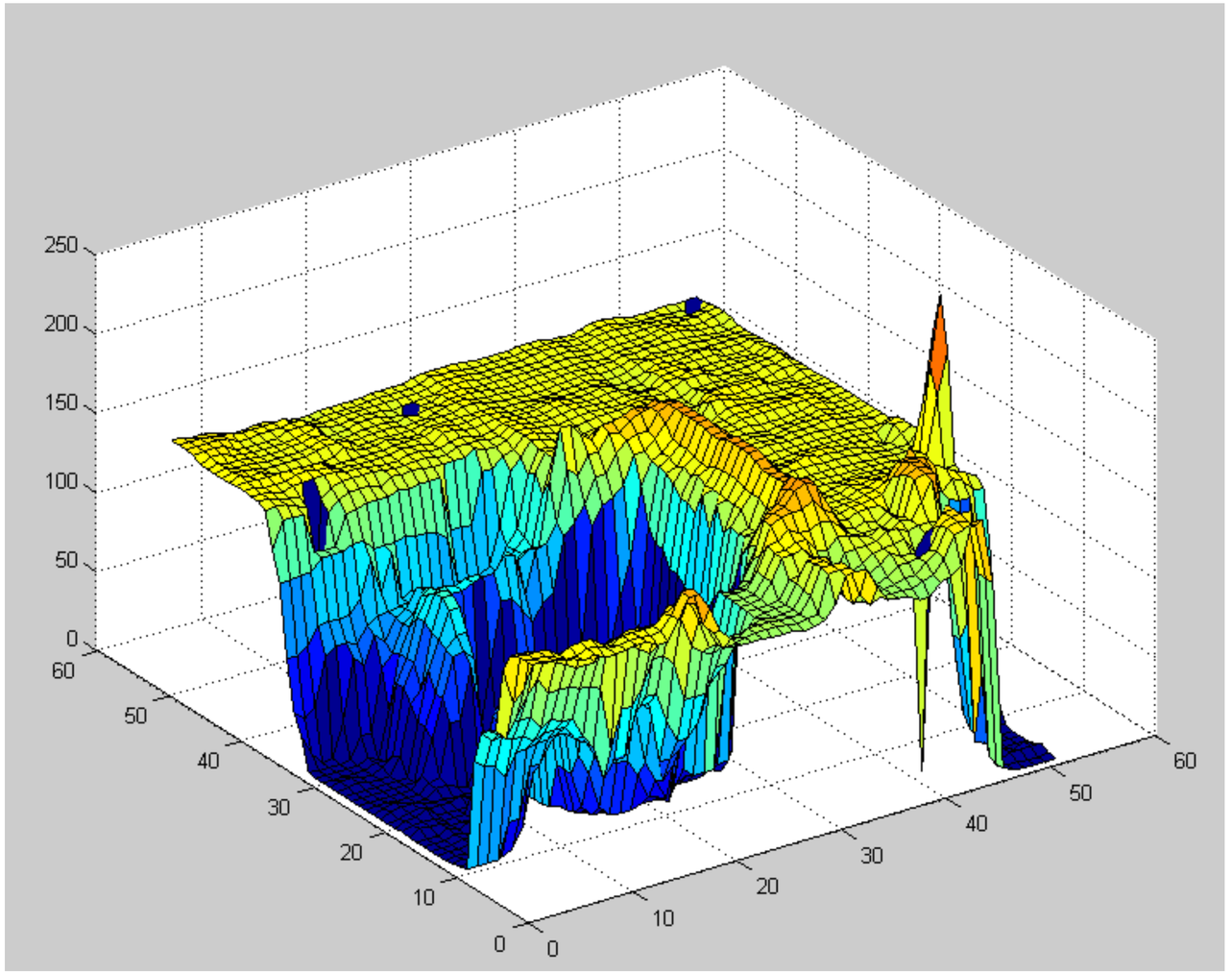
convolution 怎么算？

Image ~ Heightfield

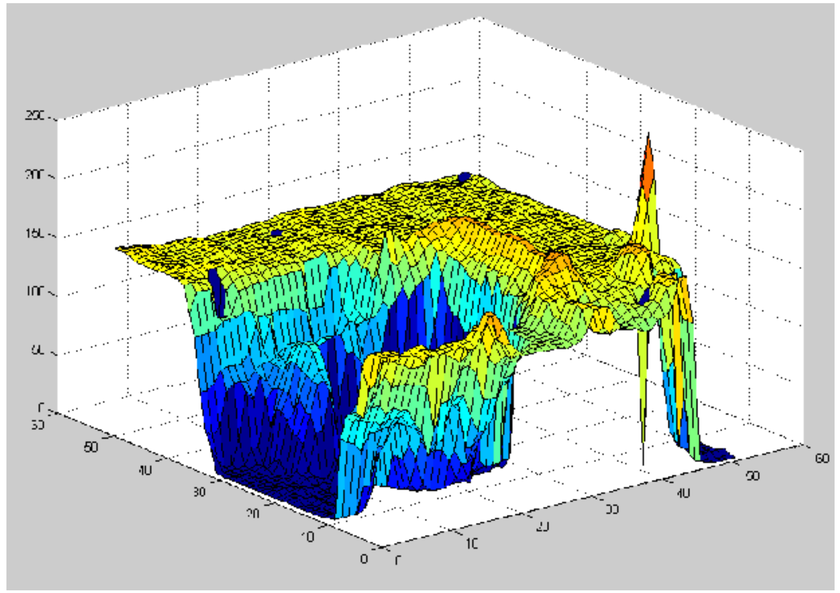
I

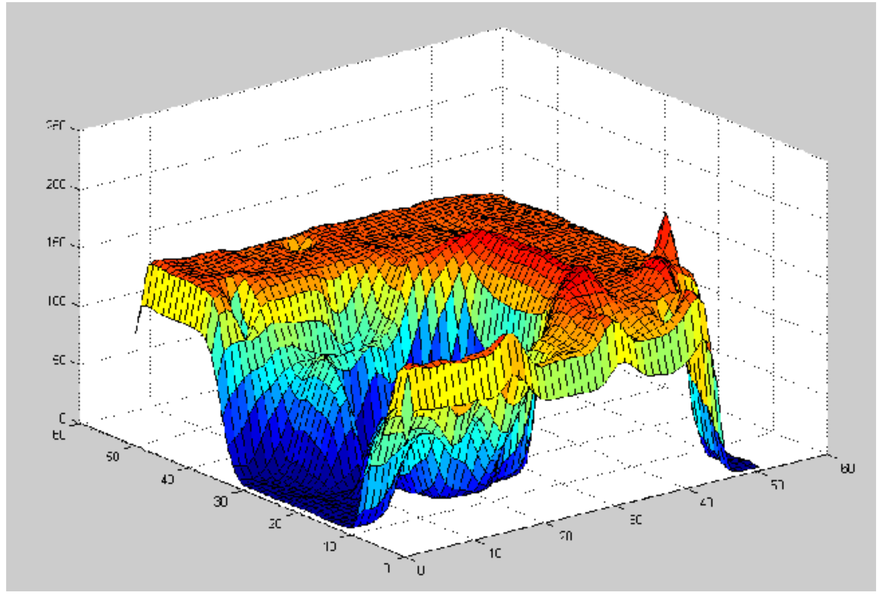


z = *I*'s intensity



Filter: Let Intensity Through Selectively

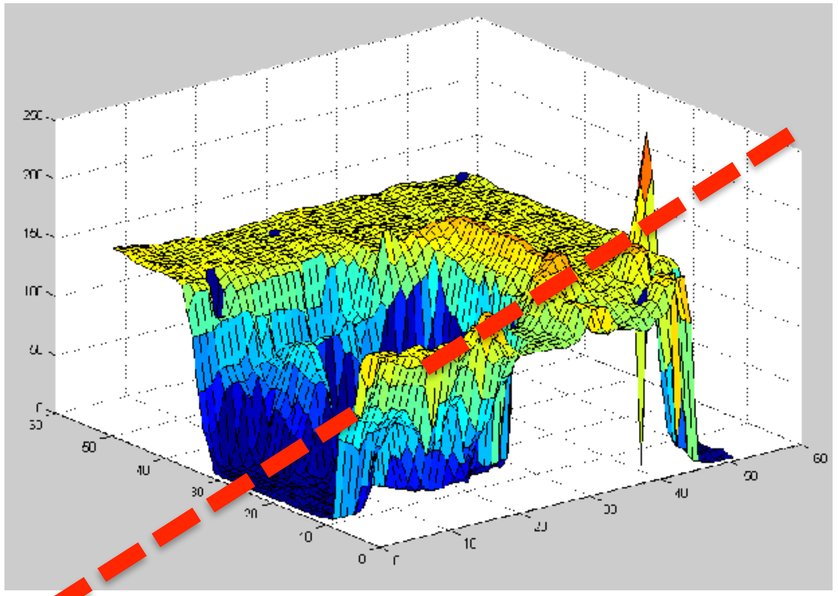


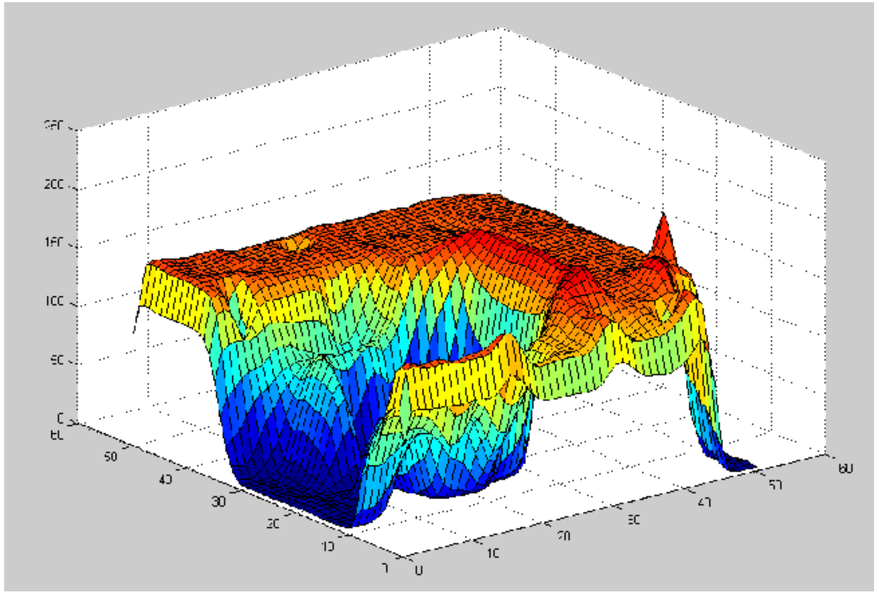


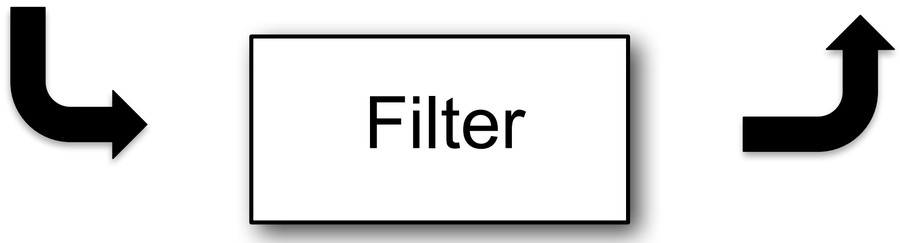
Filter

let the intensity through conditionally

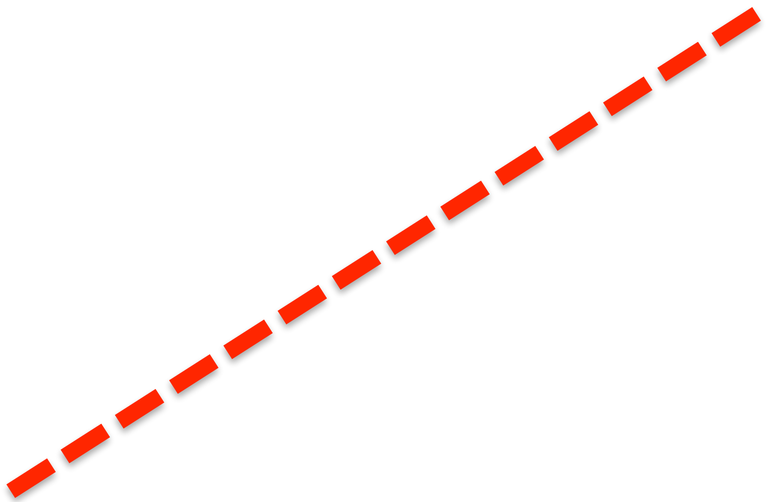
Filter: Let Intensity Through Selectively





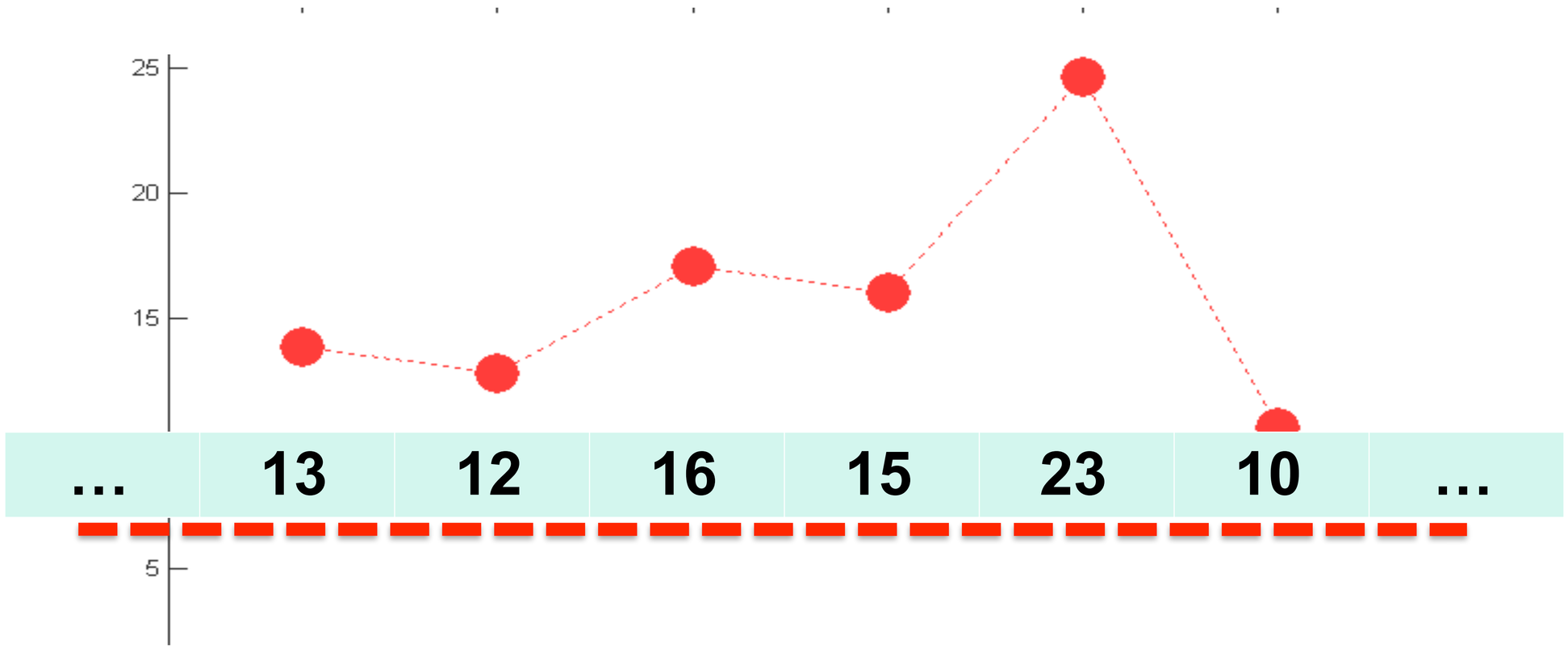


Filter: Let Intensity Through Selectively

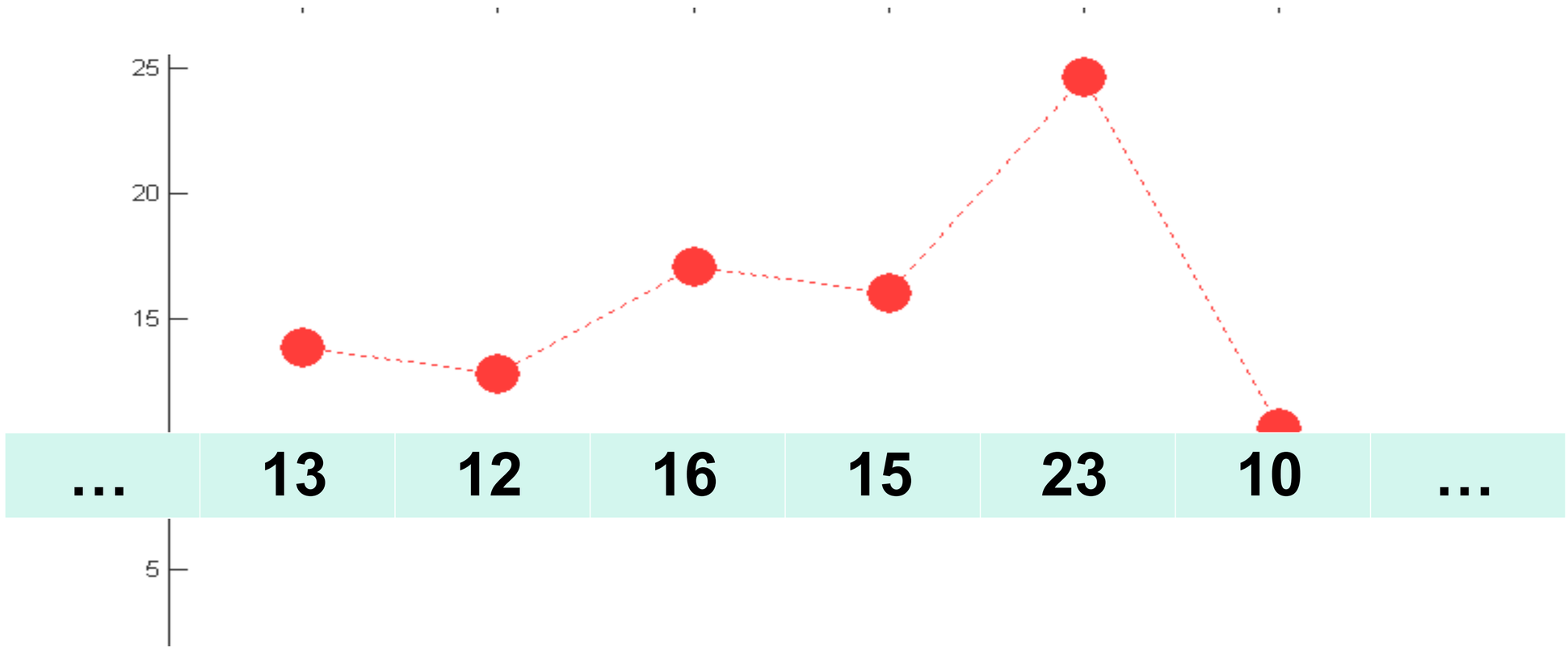


Filter: Let Intensity Through Selectively

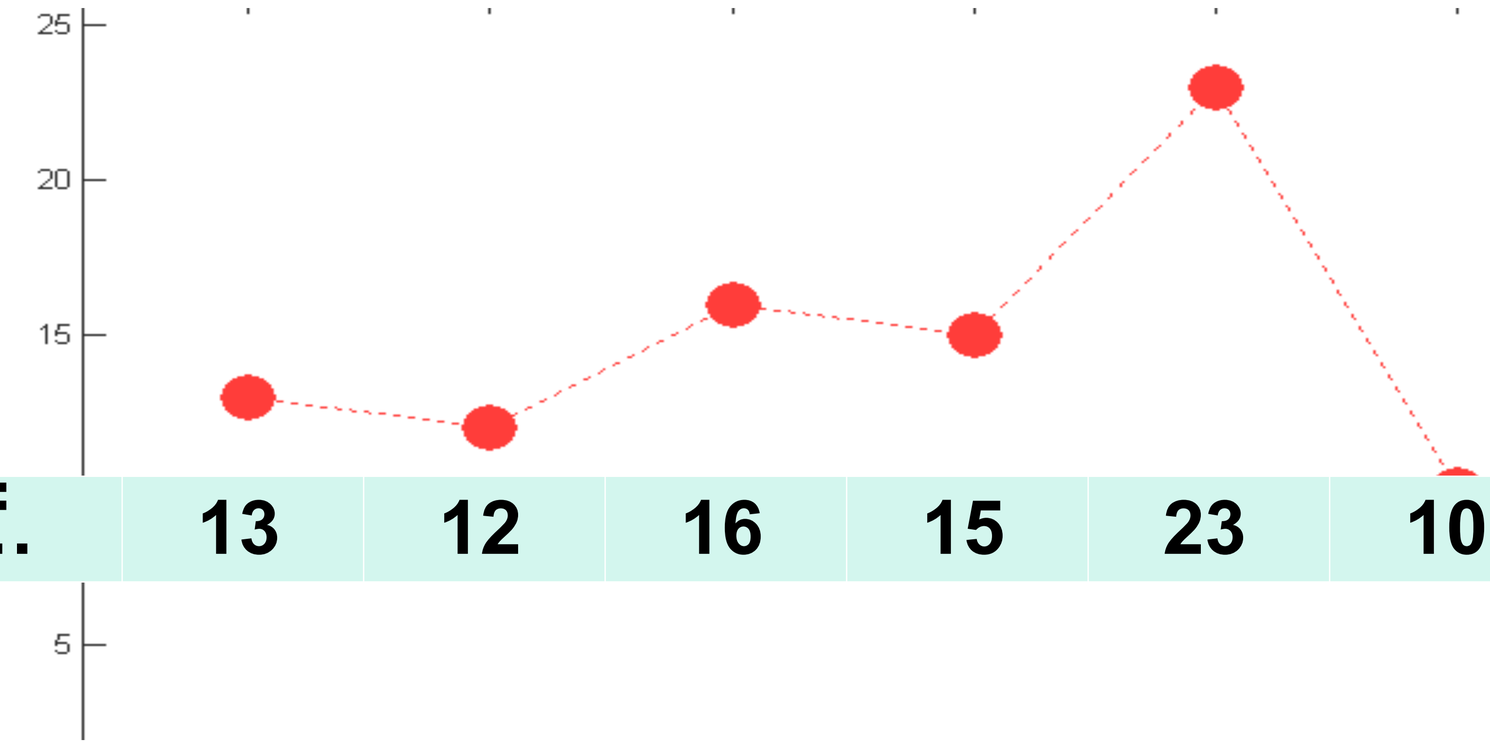
A Row of Pixel Intensities



A Row of Pixel Intensities



Linear Operations: Weighted Sum

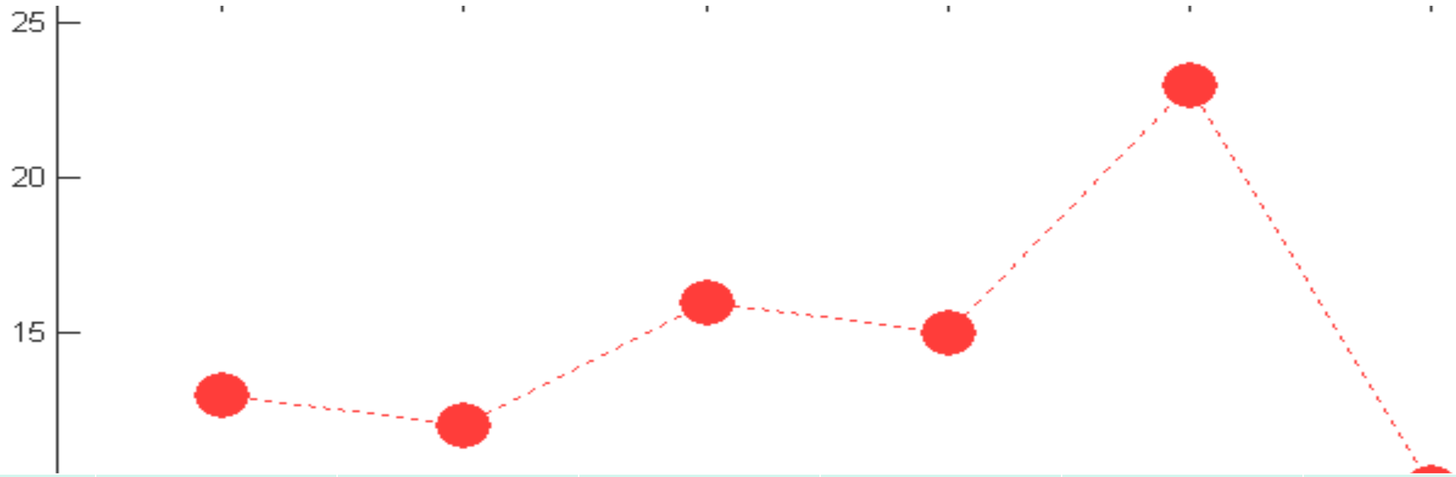


$I =$

...	13	12	16	15	23	10	...

$I' =$

...	??	??	??	??	...



$I =$ | ... | 13 | 12 | 16 | 15 | 23 | 10 | ... |

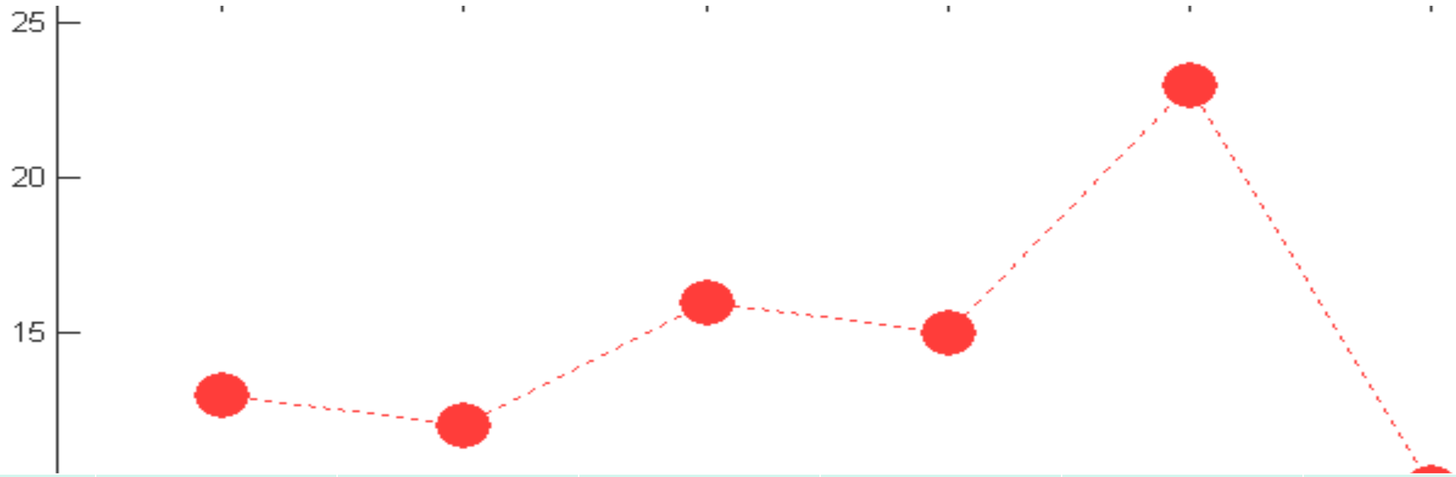
$K =$ | 1/3 | 1/3 | 1/3 |

* * *

Σ

$I' =$ | ... | ?? | ?? | ?? | ?? | ... |

Neighborhood: $N(x) = x \pm 1$
Operation: Average

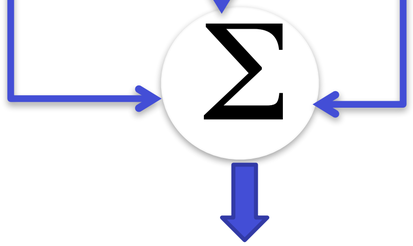


$I =$

...	13	12	16	15	23	10	...

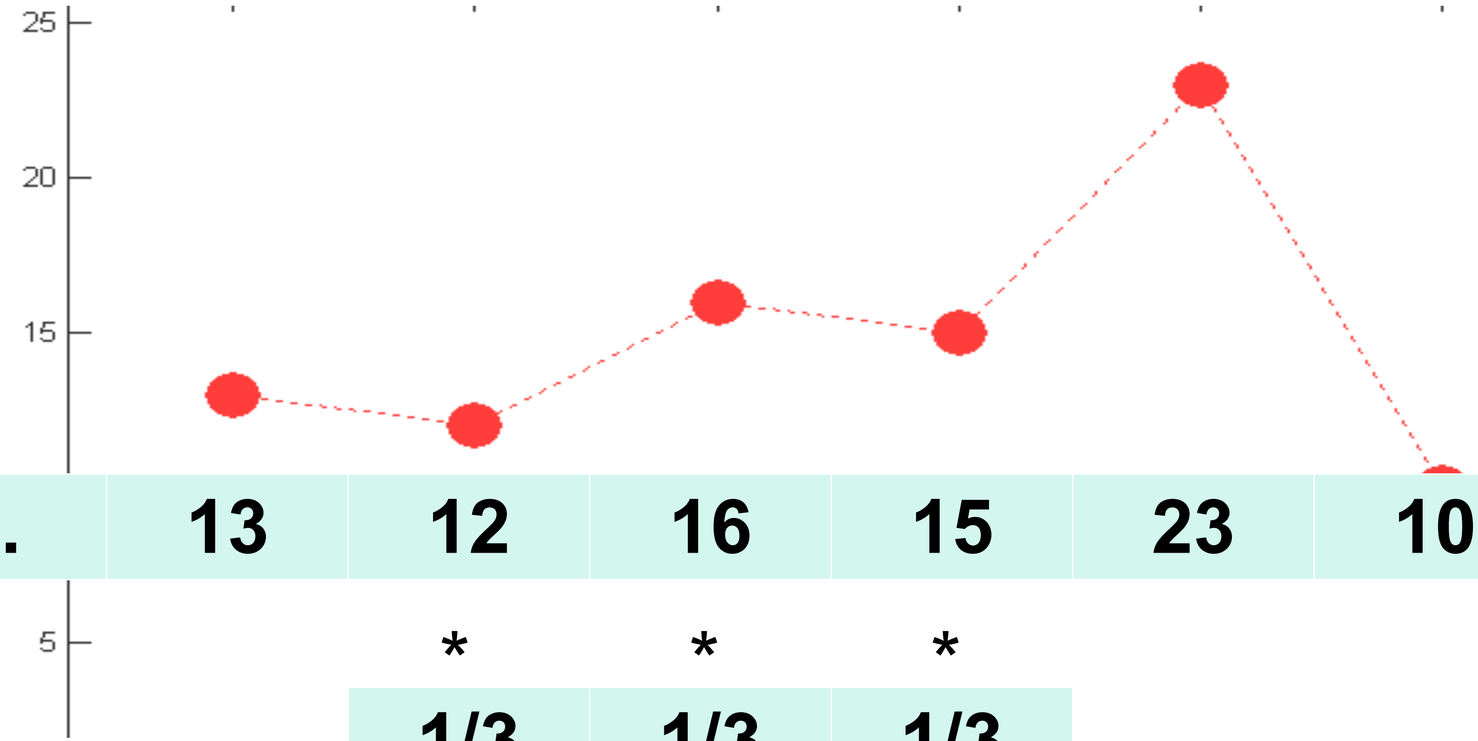
$K =$

	*		*		*	
	1/3		1/3		1/3	



$I' =$

...	13.66	??	??	??	...



$I =$

...	13	12	16	15	23	10	...

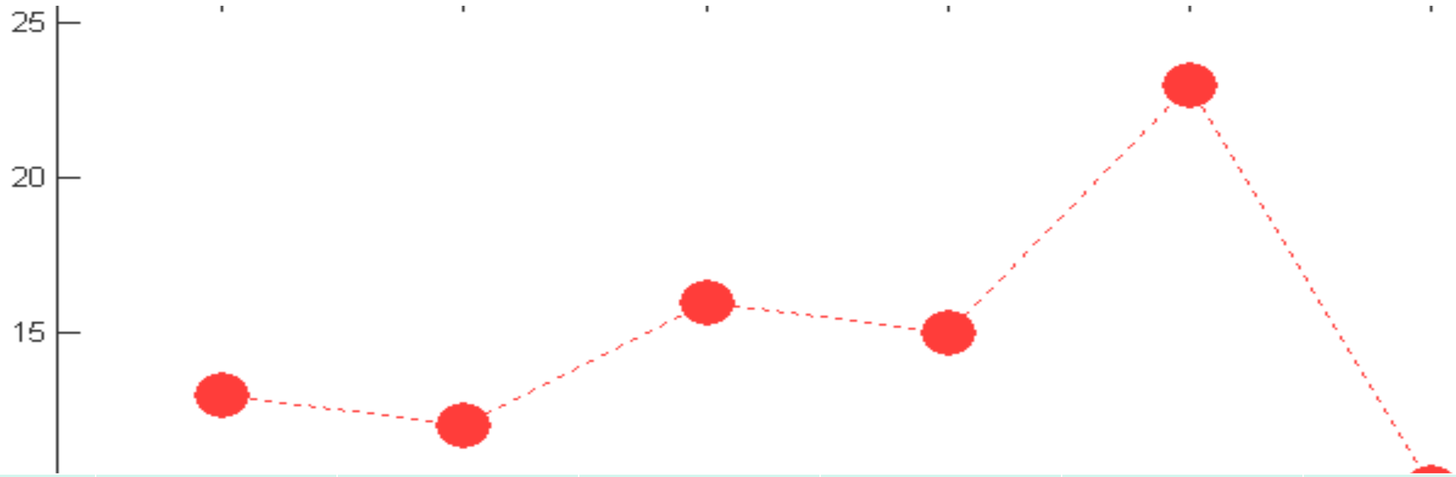
$K =$

	*		*		*	
	1/3		1/3		1/3	

Σ

$I' =$

...	13.66	??	??	??	...

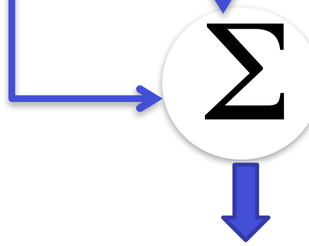


$I =$

...	13	12	16	15	23	10	...

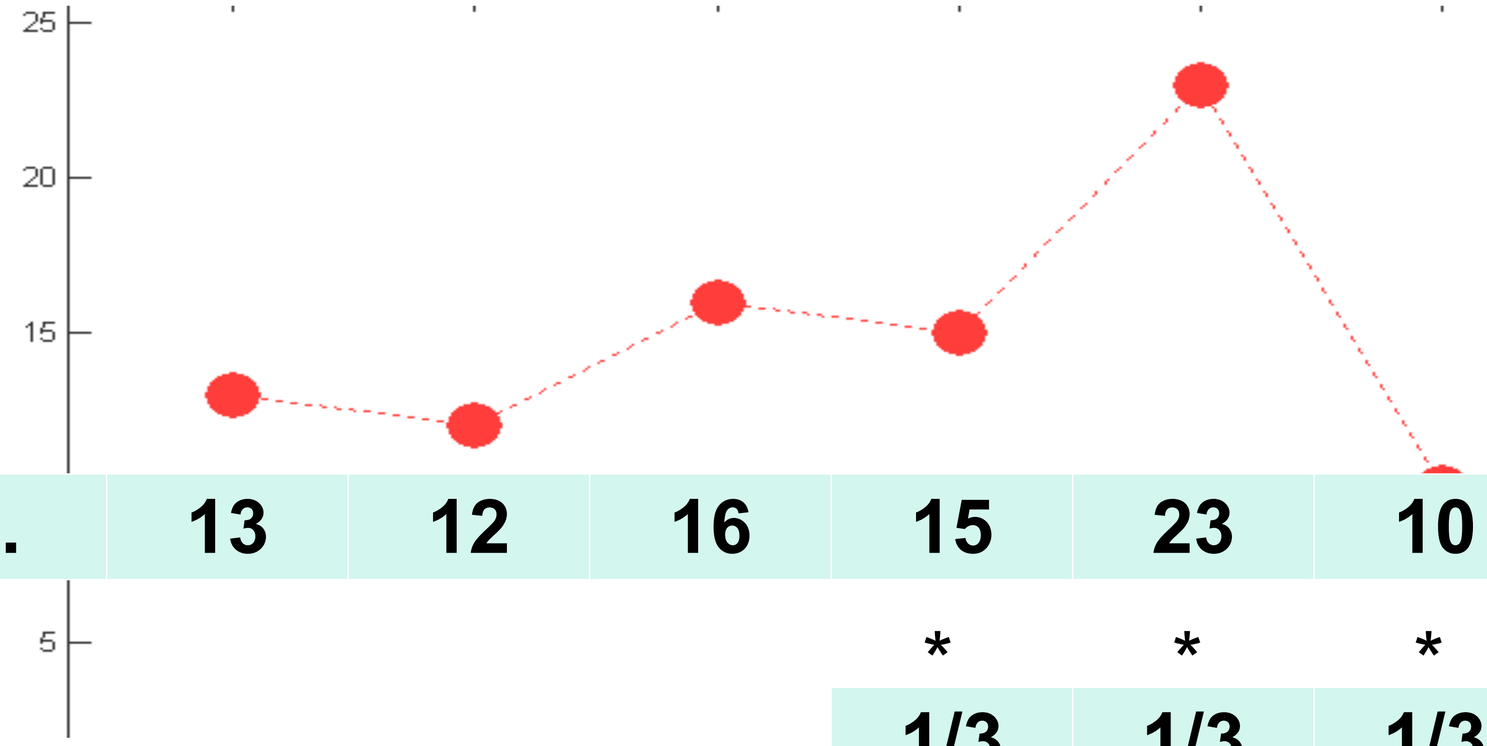
$K =$

*	*	*
1/3	1/3	1/3



$I' =$

...	13.66	14.33	??	??	...

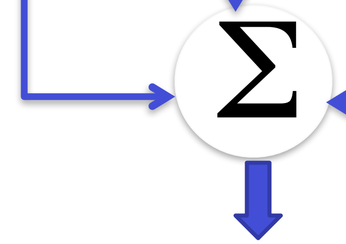


$I =$

...	13	12	16	15	23	10	...

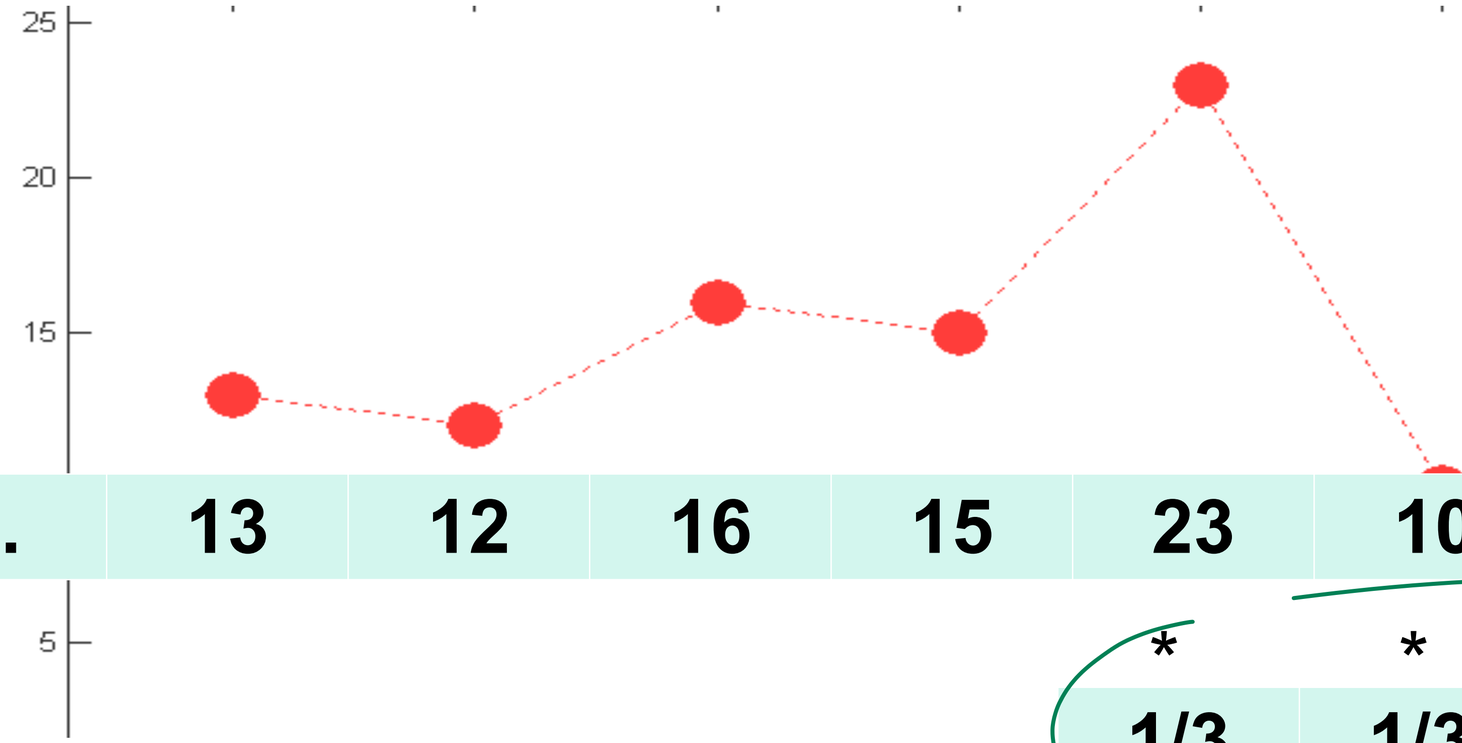
$K =$

			*	*	*	
			1/3	1/3	1/3	



$I' =$

...	13.66	14.33	18	??	...



$I =$

…	13	12	16	15	23	10	…

$K =$

*	*	*
1/3	1/3	1/3

Σ

add up to 1

$I' =$

…	13.66	14.33	18	16	…

Correlation

- **Linear operation in 1D:**

$$I'(x) = \sum_{i=-a}^{a} K(i)I(x+i)$$

- ***I* is the input image; *I'* is the output of the operation**
- ***K* is the *kernel* of the operation; many choices!**
- ***N*(*x*) is a neighbourhood of size (*2a+1)***

Correlation

- **Linear operation in 1D:**

$$I'(x) = \sum_{i=-a}^{a} K(i) I(x+i)$$

- **Linear operation in 2D:**

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

- ***I* is the input image; *I'* is the output of the operation**
- ***K* is the *kernel* of the operation; many choices!**
- ***N*(*x*,*y*) is a neighbourhood of size (*2a+1)* x *(2b+1)***

box filter :-

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

K

1	1	1
1	1	1
1	1	1

1/9

I'

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) =
1/9.(90) = 10

Slide credit: Christopher Rasmussen

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

K

1	1	1
1	1	1
1	1	1

1/9

I’

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) =
1/9.(90) = 10

Slide credit: Christopher Rasmussen

Correlation with a Unit-Impulse Image

$I =$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

$K =$ | 5 | 6 | 7 |

$I' =$ | _ | _ | _ | _ | _ | _ | _ | _ |

和 convolution 有点像

Correlation with a Unit-Impulse Image

$I =$

0	0	0	1	0	0	0	0

$K =$

5	6	7

$I' =$

0	0	7	6	5	0	0	0

Correlation reveals copy of *K*, rotated by 180°

Correlation with a Unit-Impulse Image

$I =$

0	0	0	1	0	0	0	0

$K =$

5	6	7

$I' =$

0	0	7	6	5	0	0	0

Correlation reveals copy of *K*, rotated by 180°

Would be nice to get result that is **NOT** rotated!

Correlation/Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

Correlation/Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

Convolution

Correlation/Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

Convolution

$$\begin{aligned} I'(x,y) &= \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x-i, y-j) \\ &= \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(-i,-j) I(x+i, y+j) \end{aligned}$$

Correlation/Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x-i, y-j)$$

flipping the kernel

$$= \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(-i,-j) I(x+i, y+j)$$

$$K(i,j) = K(-i,-j) \quad \Leftrightarrow \quad \text{convolution} \equiv \text{correlation}$$

Correlation/Convolution

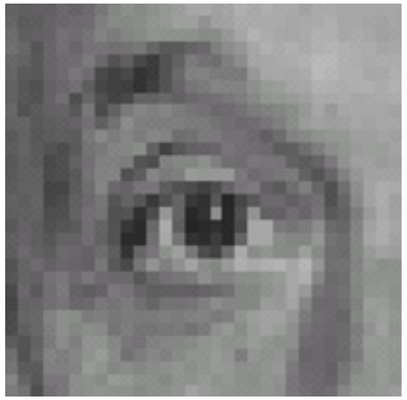
$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x+i, y+j)$$

Convolution

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x-i, y-j)$$

$$I' = K * I$$

Linear Filtering (warm-up)



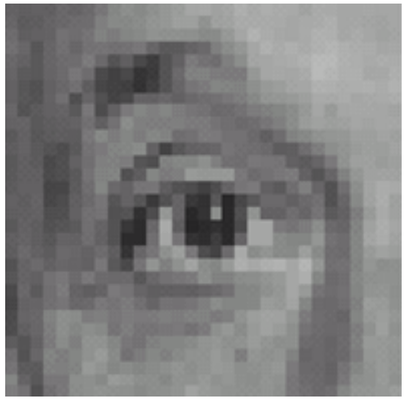
Original

0	0	0
0	1	0
0	0	0

?

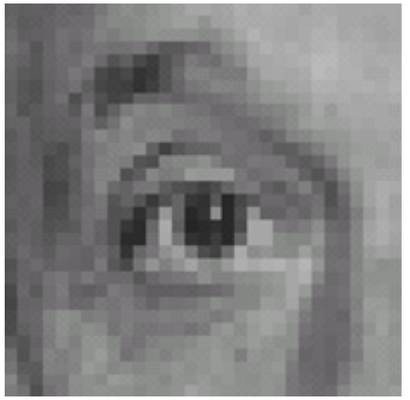
Slide credit: D.A. Forsyth

Linear Filtering (warm-up)



Original

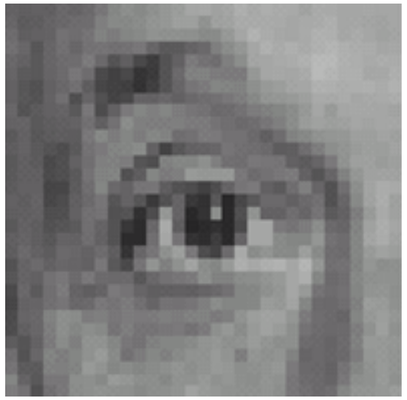
0	0	0
0	1	0
0	0	0



Filtered
(no change)

Linear Filtering (Convolution)

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x-i, y-j)$$



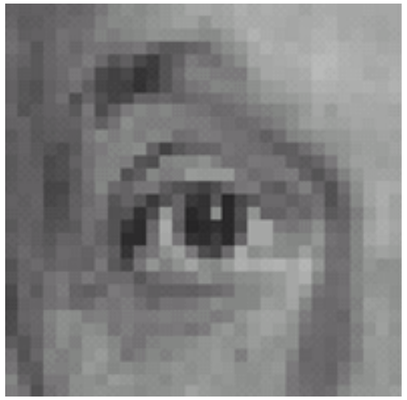
Original

0	0	0
1	0	0
0	0	0

?

Linear Filtering (Convolution)

$$I'(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} K(i,j) I(x-i, y-j)$$

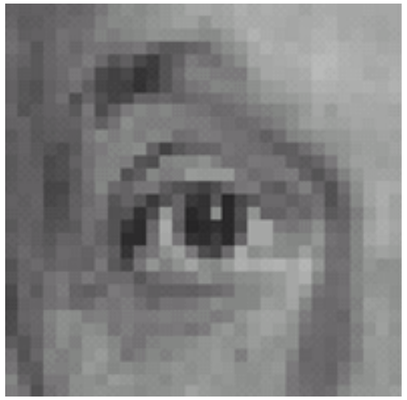


Original

0	0	0
1	0	0
0	0	0

Shifted left
By 1 pixel

Linear Filtering



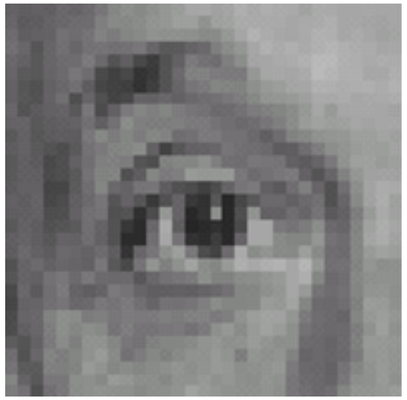
Original

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

?

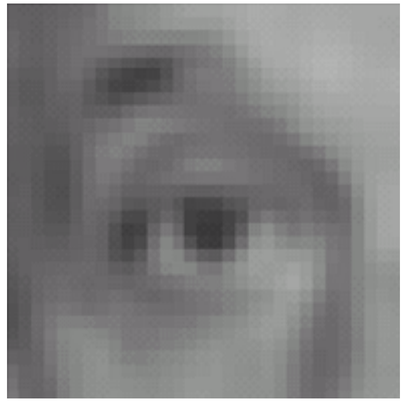
Linear Filtering

BOX FILTER



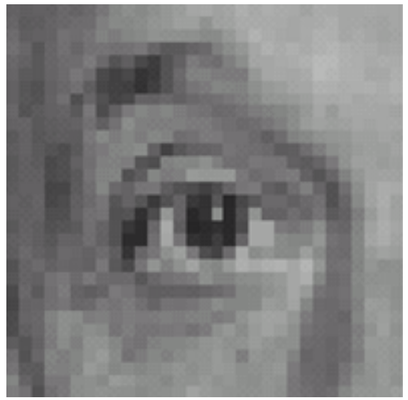
Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blur (with a box filter)

Linear Filtering

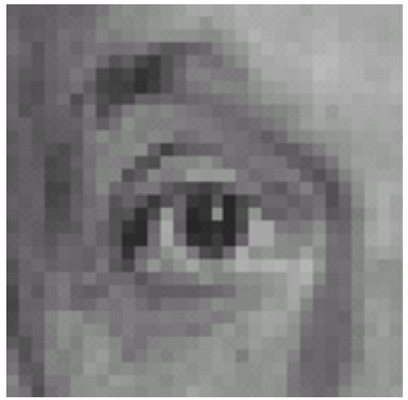


Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad ?$$

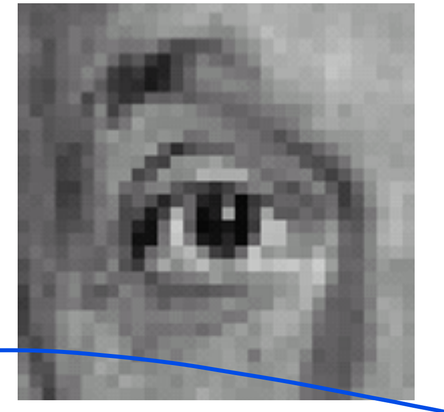
(Note that filter sums to 1)

Linear Filtering



Original

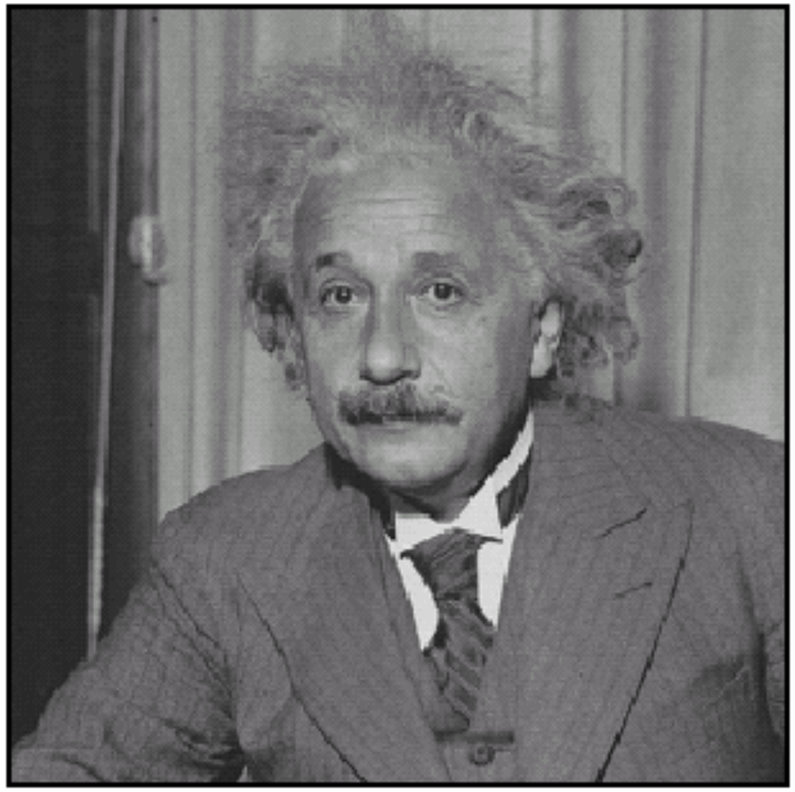
$$\begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$



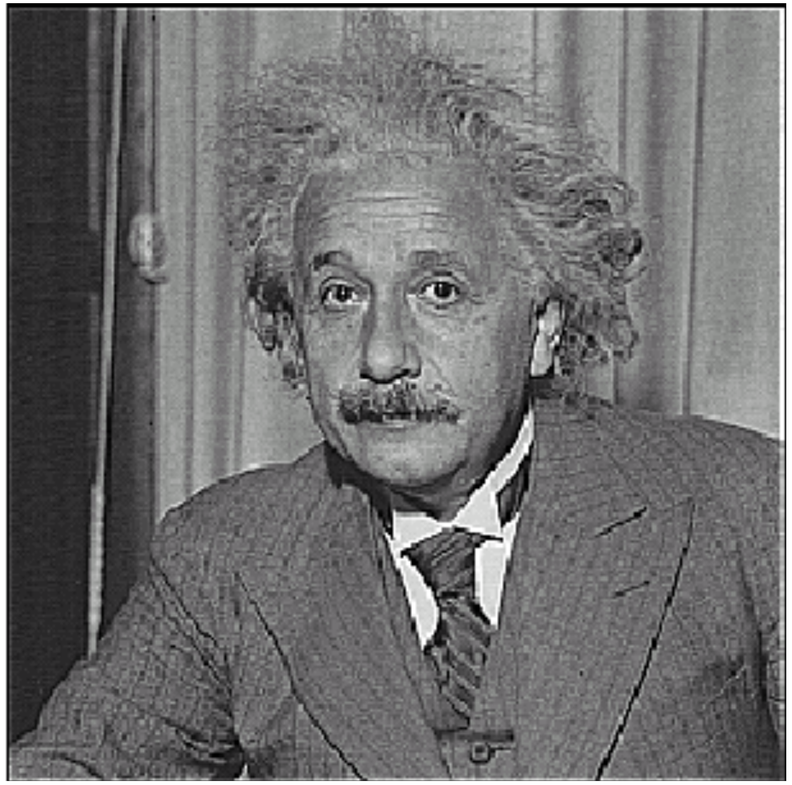
Sharpening filter

- Amplifies differences with local average
- Also known as Laplacian

Sharpening



before



after

Example

K=ones(9,9);

I2=conv2(I,K);

Example

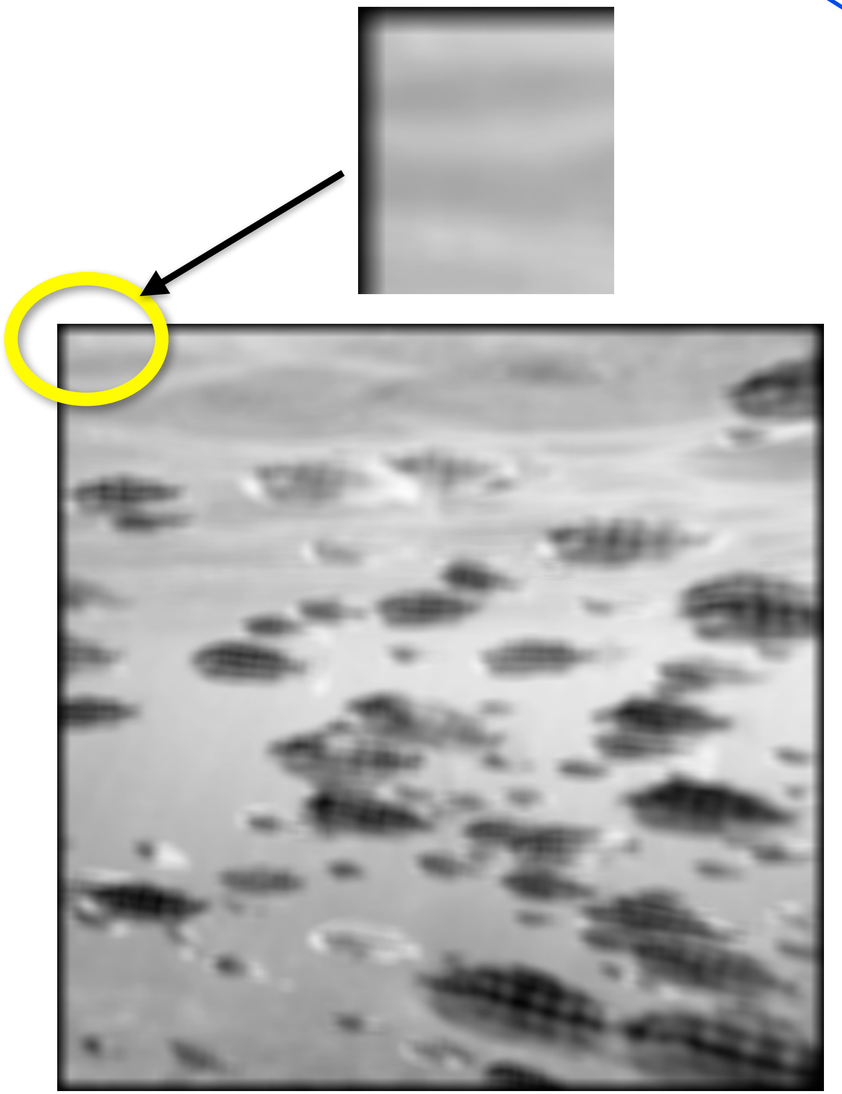
K=ones(9,9);

I2=conv2(I,K);

Example

K=ones(9,9);

I2=conv2(I,K);



‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

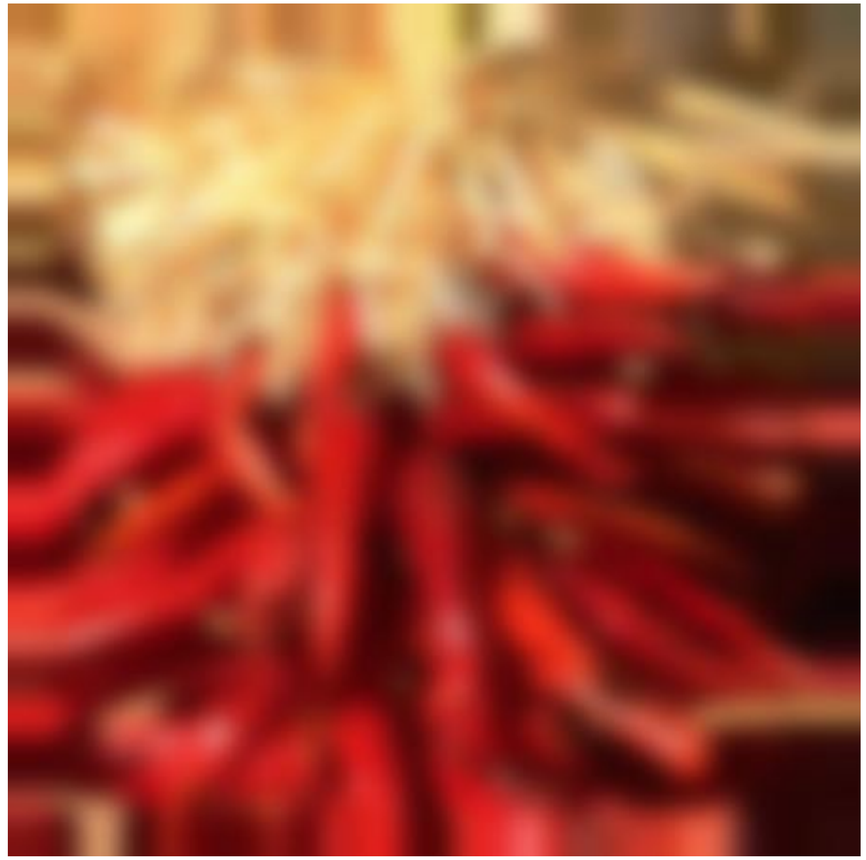
- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge



‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

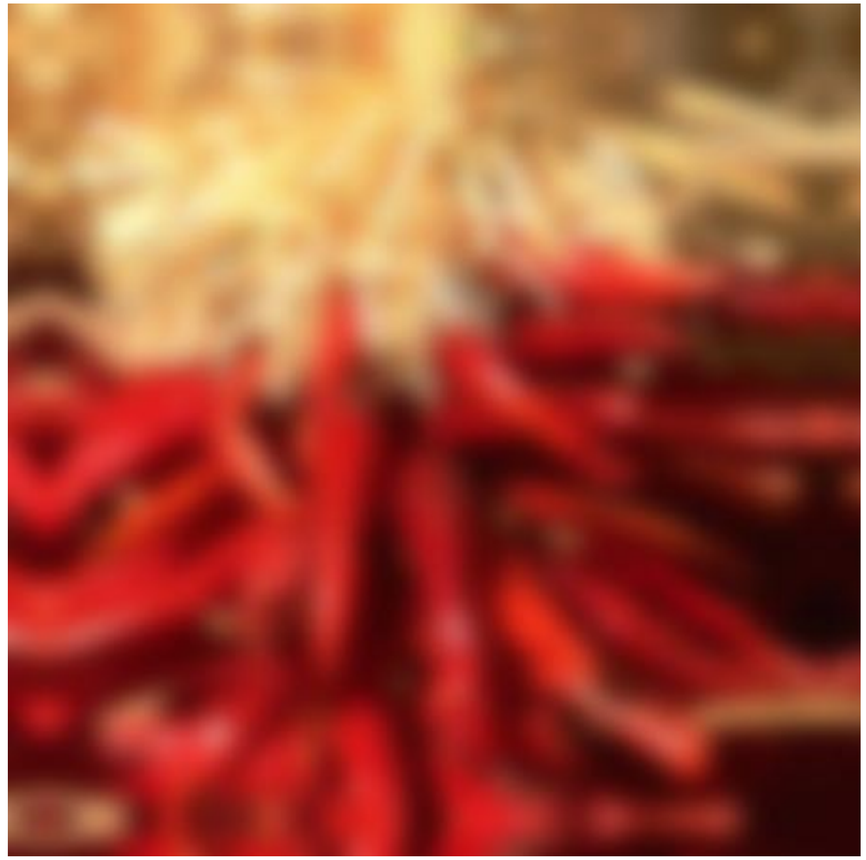
- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge

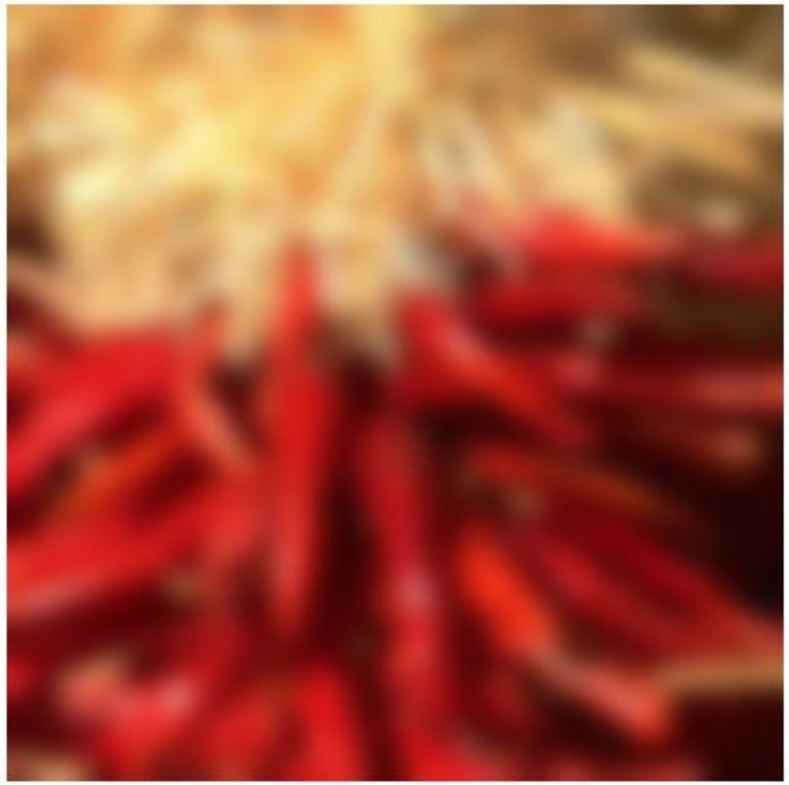
‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge



‘Boring’ Details

- When filter window falls off the edge of the image?
  - need to extrapolate
  - methods:
    - clip filter (black)
    - wrap around
    - copy edge
    - reflect across edge
    - vary filter near edge



Smoothing Kernels (Low-pass filters)

Mean filter:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smoothing Kernels (Low-pass filters)

Mean filter:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

giving more importance to the center &

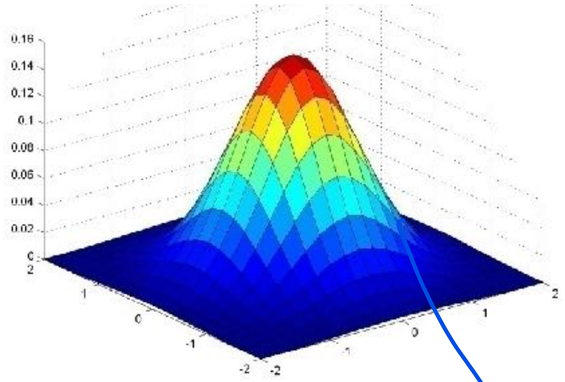
Weighted smoothing filters:

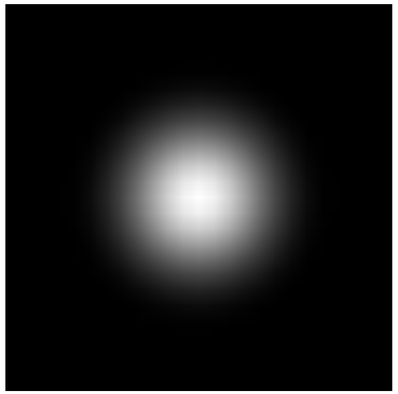
$$\frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

(a bit like gaussian)

Gaussian Kernel

- **Weight contributions of neighbouring pixels related to nearness**





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

5 x 5, σ = 1

- **Constant factor in the front makes volume sum to 1**
  - **(subject to rounding errors, so better to normalize by sum of matrix)**

Slide credit: Christopher Rasmussen

Review of Sigma (σ)

σ = sigma = standard deviation = √(variance)

$$Data, \mathbf{d} = [d_1 \quad d_2 \quad d_3 \quad \ldots \quad d_n]$$

$$Mean := \quad \mu = \frac{\sum_i \mathbf{d}_i}{}$$

$$Variance := \sum_i (\mathbf{d}_i - \mu)^2 / n$$

$$StdDeviation := \sigma = \sqrt{Variance}$$

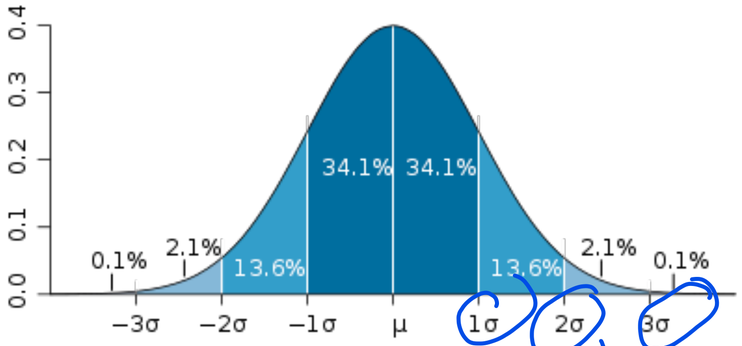
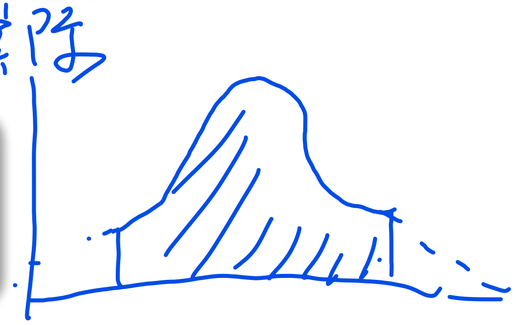
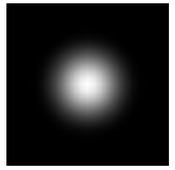


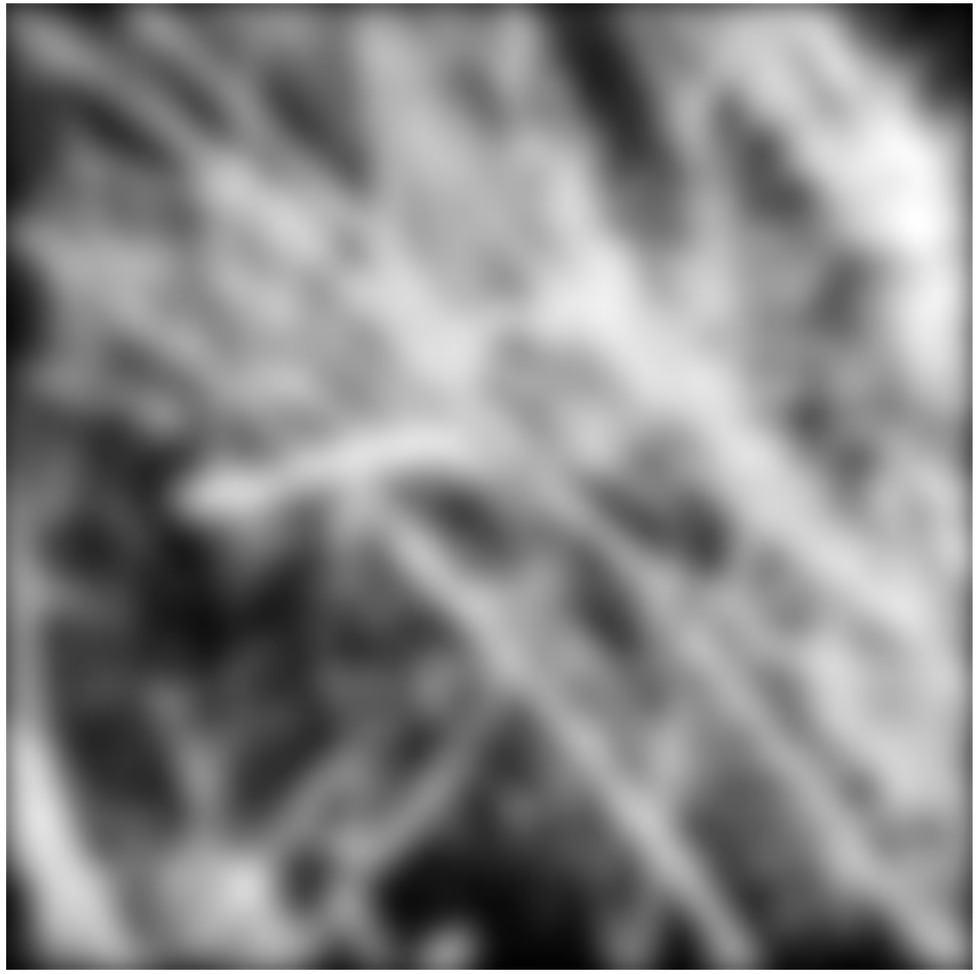
Figure by Petter Strandmark

```
mean(d), var(d), std(d)
```



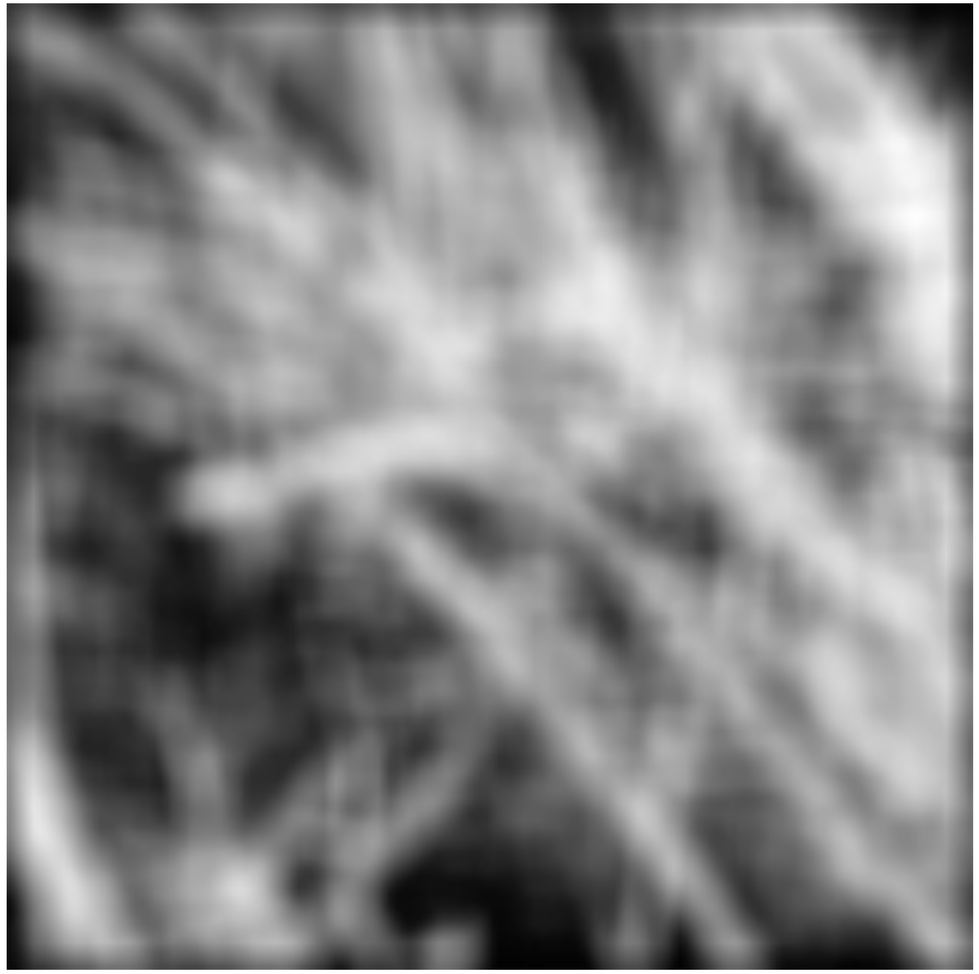
Gaussian Blur





Slide credit: D.A. Forsyth

Smoothing with Box Filter



Slide credit: D.A. Forsyth

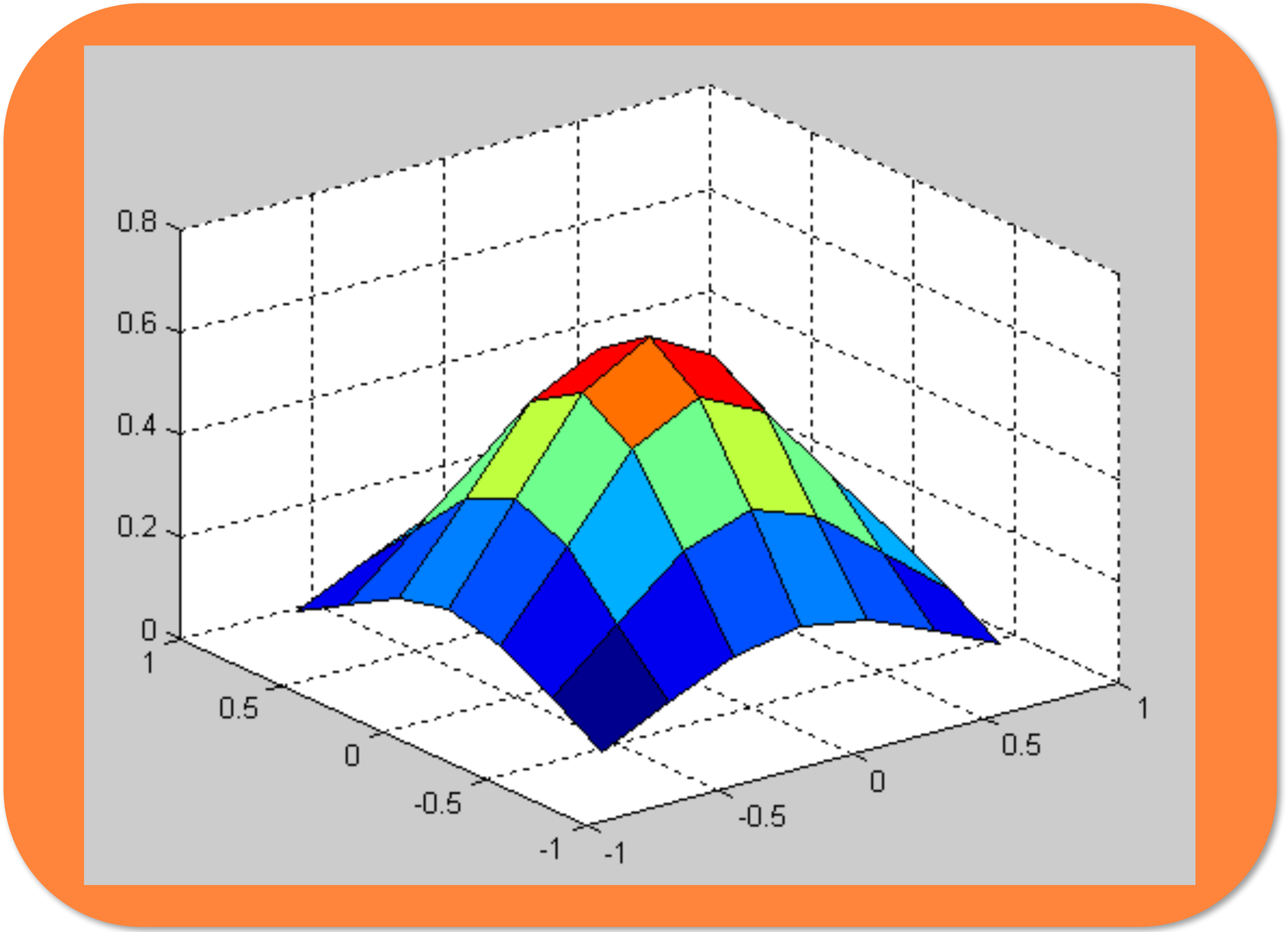
```
X =

   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500
   -0.7500   -0.5000   -0.2500         0    0.2500    0.5000    0.7500

Y =

   -0.7500   -0.7500   -0.7500   -0.7500   -0.7500   -0.7500   -0.7500
   -0.5000   -0.5000   -0.5000   -0.5000   -0.5000   -0.5000   -0.5000
   -0.2500   -0.2500   -0.2500   -0.2500   -0.2500   -0.2500   -0.2500
         0         0         0         0         0         0         0
    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500    0.2500
    0.5000    0.5000    0.5000    0.5000    0.5000    0.5000    0.5000
    0.7500    0.7500    0.7500    0.7500    0.7500    0.7500    0.7500

>> Z = (1/(2*pi*sigma.^2)) * exp(-(X.^2 + Y.^2)/(2*sigma.^2));
>> surf(X,Y,Z)    %  shown with sigma = 0.5
>>
>> % Probably want Z = Z / sum(Z(:));
```


0.8
0.6
0.4
0.2
0
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1

Separable Kernels

- **Separable filters can be written as *K*(*m*,*n*) = *f*(*m*)*g*(*n*)**
- **For a rectangular neighbourhood with size *(2M+1)*x*(2N+1)*,**

$$I'(m,n) = f \star (g \star I(N(m,n)))$$

$$I''(m,n) = \sum_{j=-N}^{N} g(j) I(m, n-j)$$

$$I'(m,n) = \sum_{i=-M}^{M} f(i) I''(m-i, n)$$

Gaussian Smoothing Kernels

$$
\begin{aligned}
g(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-y^2}{2\sigma^2}\right) \\
&= g(x)g(y)
\end{aligned}
$$

Gaussian Smoothing Kernels

$$
\begin{aligned}
g(x,y) &= \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right) \\
&= \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-x^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(\frac{-y^2}{2\sigma^2}\right) \\
&= g(x)g(y)
\end{aligned}
$$

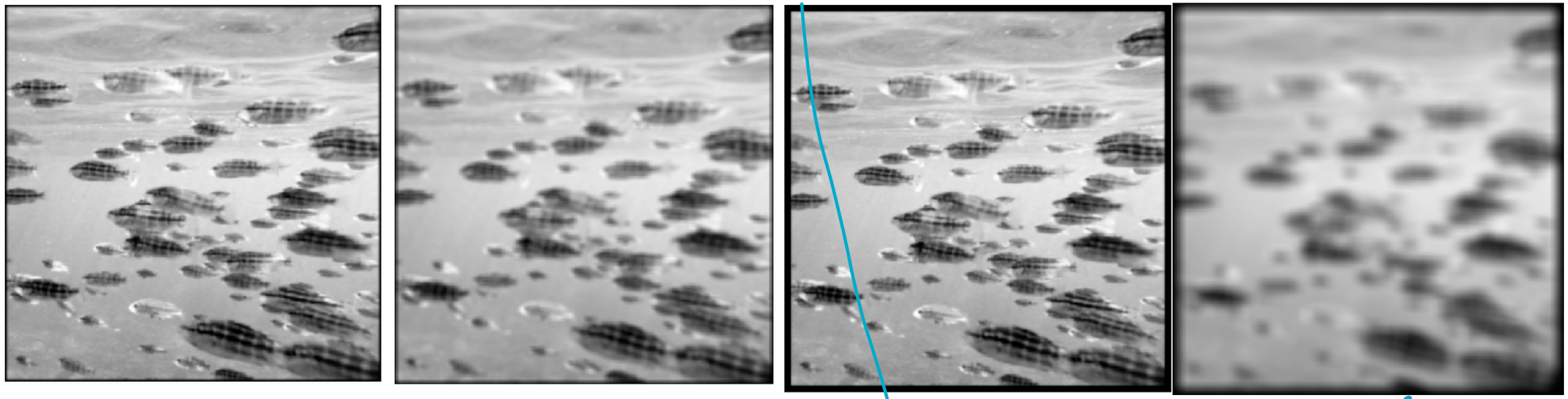
Separable!

To see how svd() can be used to obtain the separated kernels of a rank 1 kernel.

Gaussian Smoothing Kernels

- **Amount of smoothing depends on σ and window size.**
- **Width > 3σ**

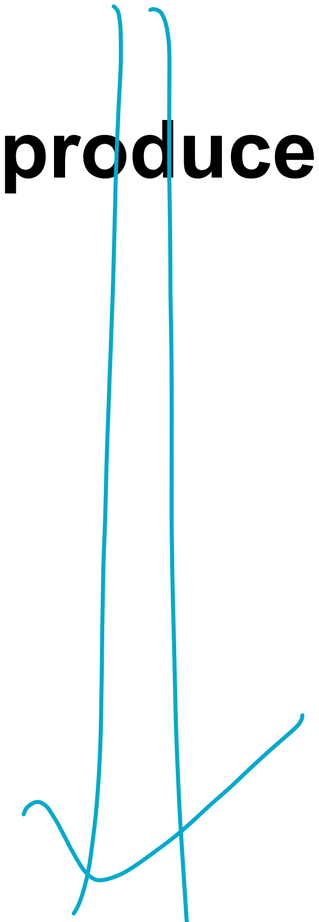
7x7; σ =1. 7x7; σ =9. 19x19; σ =1. 19x19; σ =9.



Scale Space

- **Convolution of a Gaussian with standard deviation σ with itself, is a Gaussian with standard deviation $\sigma\sqrt{2}$.**

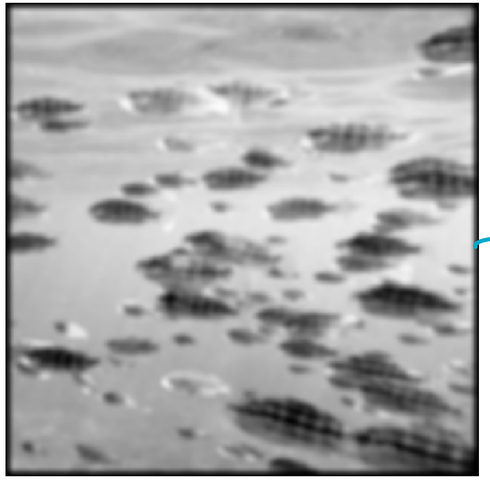
- **Repeated convolution by a Gaussian filter produces the *scale space* of an image.**



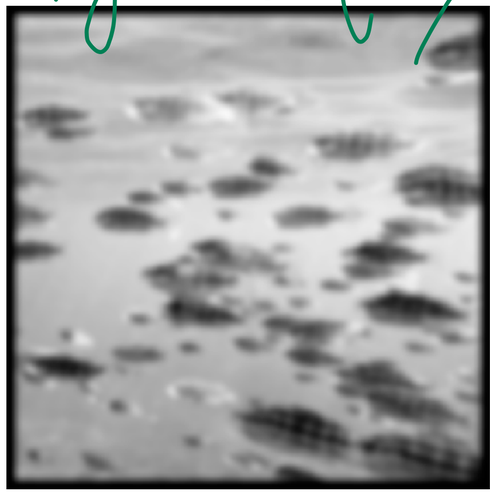
Scale Space Example

11x11; σ =3.

1



2



high frequency

3

repeated apply Gauss Filter on the last result

4

5

6

low frequency

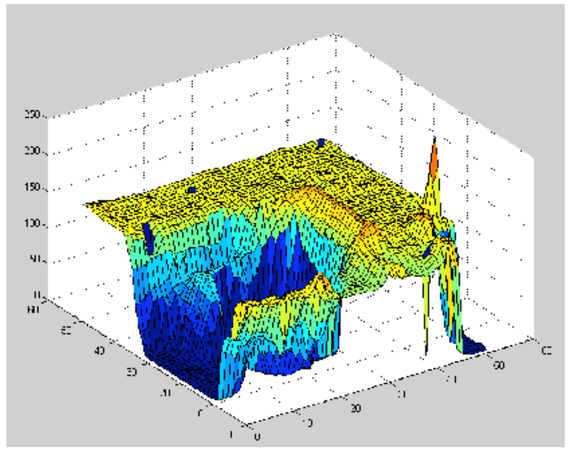
Gaussian Smoothing Kernel Top-5

1. **Rotationally symmetric**
2. **Has a single lobe/mode**
   - **Neighbour's influence decreases monotonically**
3. **Still one lobe in frequency domain**
   - **No corruption from high frequencies**
4. **Simple relationship to σ**
5. **Easy to implement efficiently**

Smoothing vs. Sharpening

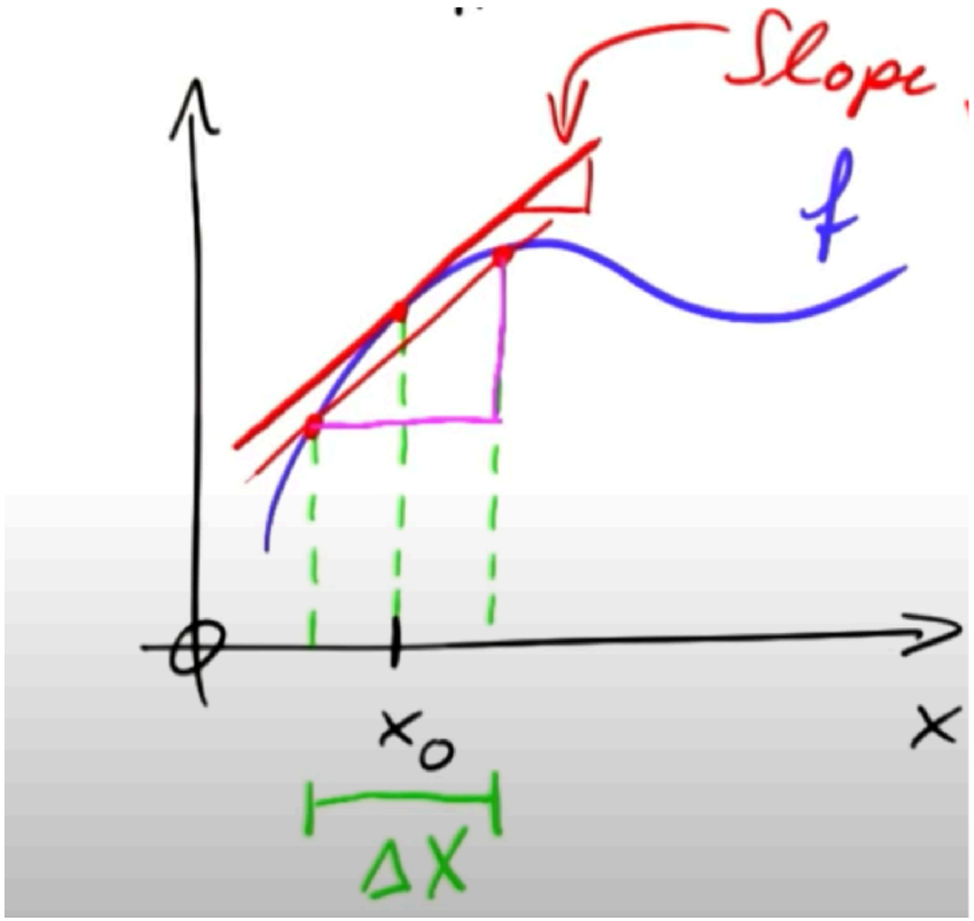
- Smoothing ↔ Integration, i.e.,
  - Summing
  - Sand-papering the height field's slope

- Sharpening ↔ Differentiation, i.e.,
  - Subtracting
  - Accentuating the height field's slope



z = *I*'s intensity

1st and 2nd Derivative



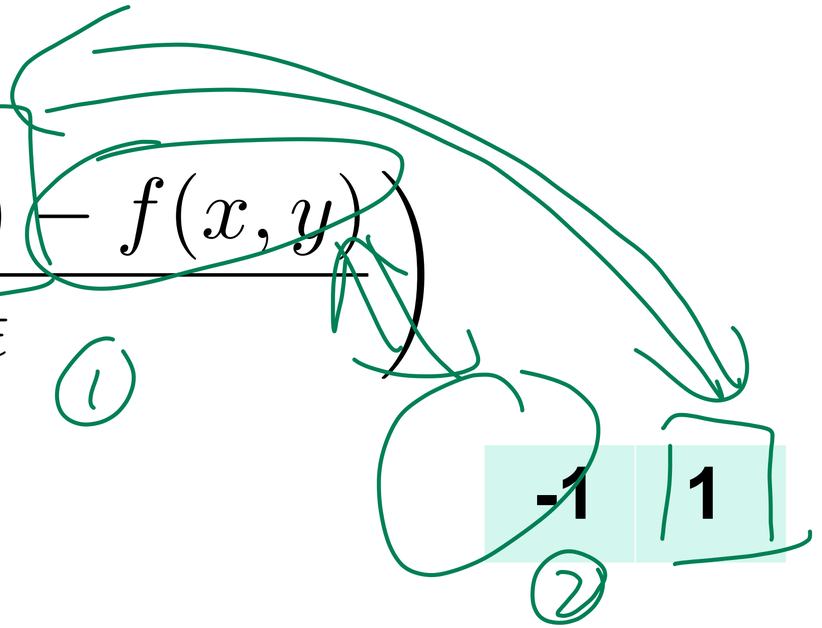
Differentiation and Convolution

- **Recall, for 2D function, f(x,y):**

$$\frac{\partial f}{\partial x} = \lim_{\epsilon \to 0} \left(\frac{f(x+\epsilon, y) - f(x,y)}{\epsilon} \right)$$

- **This is linear and shift invariant, so the result of a convolution**
- We could approximate this as (which is obviously a convolution)

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$



1st and 2nd Derivative

The derivative of a function f at a point x is defined by the limit

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Approximation of the derivative when h is small

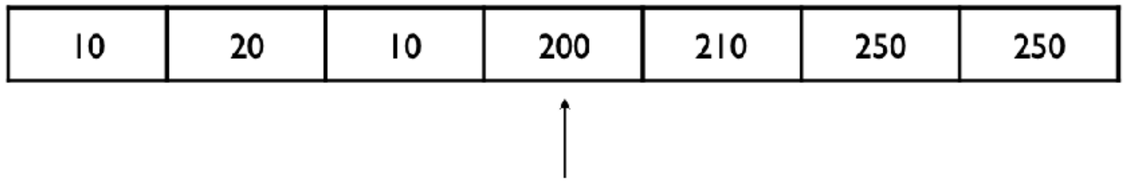
This definition is based on the 'forward difference' but ...

1st and 2nd Derivative

it turns out that using the 'central difference' is more accurate

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

How do we compute the derivative of a discrete signal?



1st and 2nd Derivative

it turns out that using the 'central difference' is more accurate

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h}$$

How do we compute the derivative of a discrete signal?

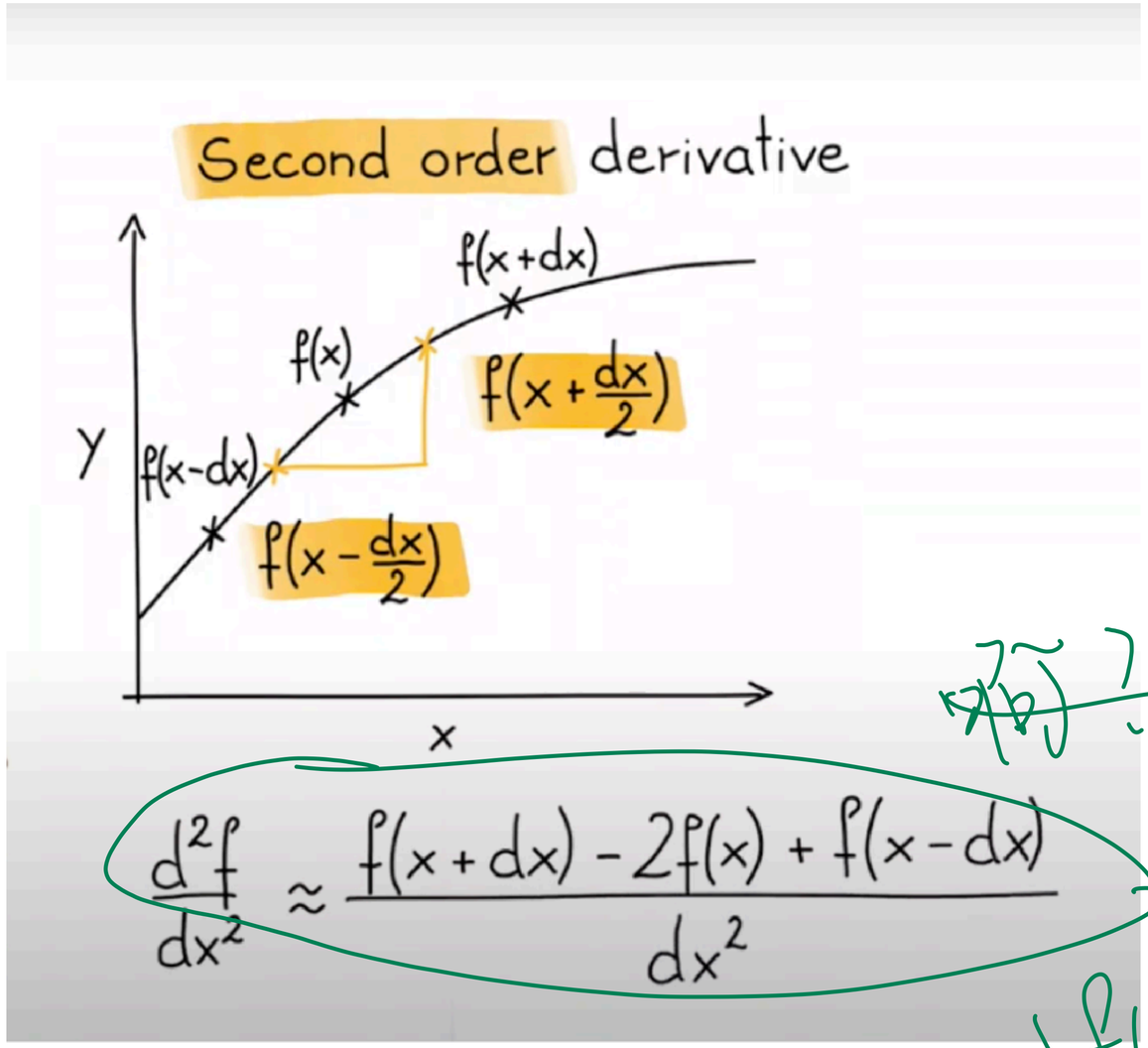
10	20	10	200	210	250	250

$$f'(x) = \frac{f(x+1) - f(x-1)}{2} = \frac{210 - 10}{2} = 100$$

-1	0	1

1D derivative filter

2nd Derivative



1st and 2nd Derivative

(Look ahead)

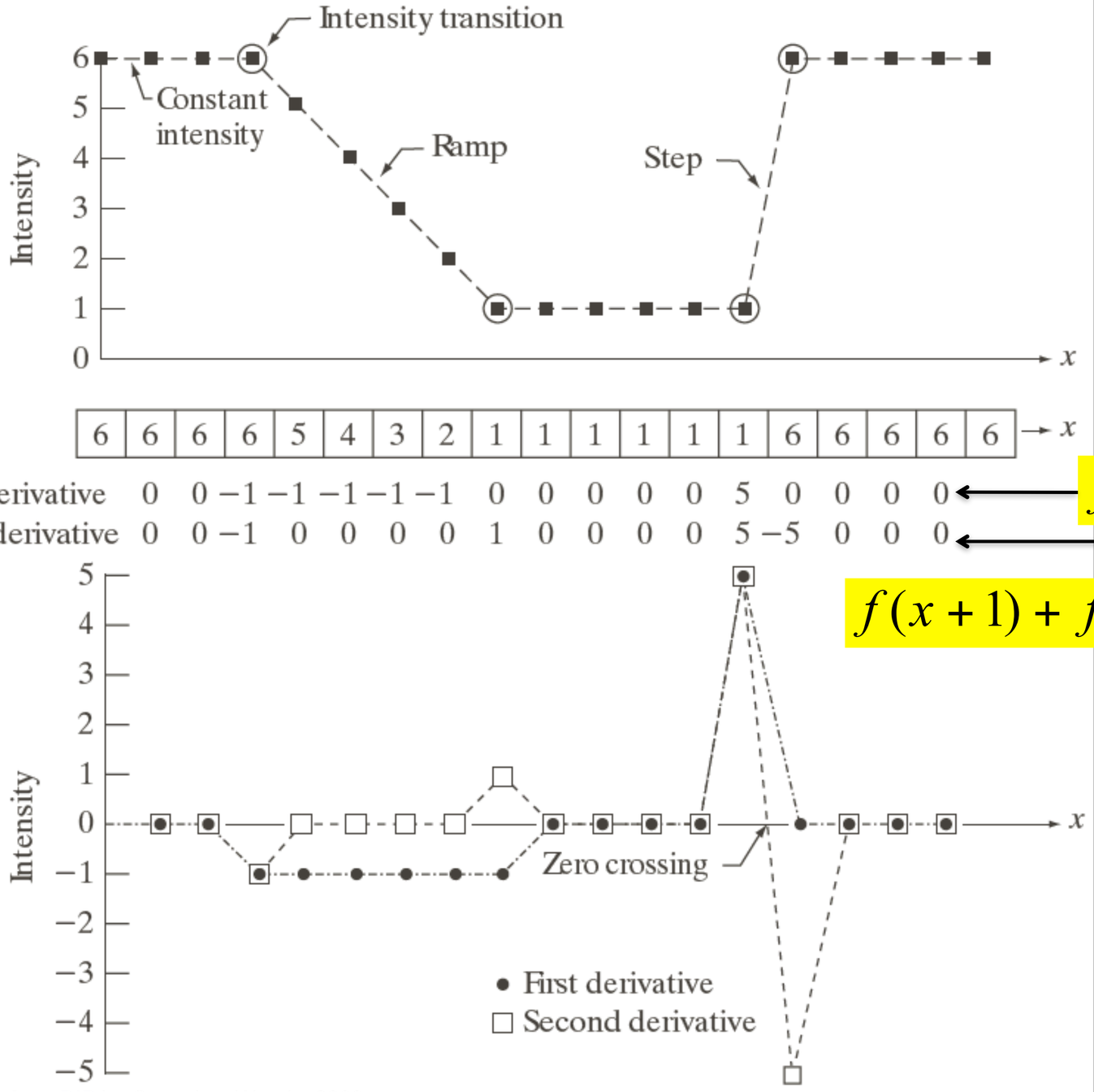
$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

(Look ahead and look back)

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

左1 右1 中

[1 -2 1]



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

$$f(x+1) + f(x-1) - 2f(x)$$

Figure from DIP by Gonzalez+Woods, 2008

Vertical Gradients from Finite Differences

Slide credit: D.A. Forsyth

Differential Filters

Prewitt operator:

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

⇒ derivative respect to 1

give more weight to pixel in the middle

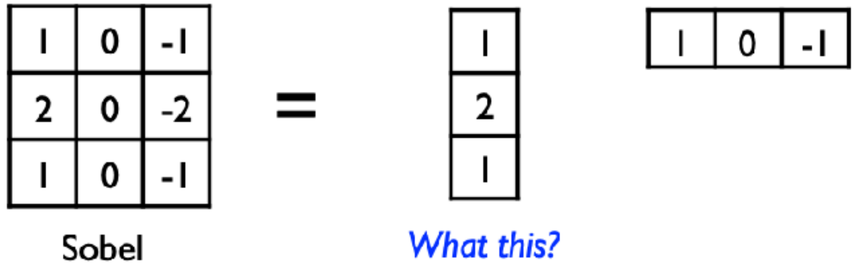
Sobel operator:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Sobel 算子

Differential Filters

Decomposing the Sobel filter



1st Derivative: From 1D to 2D

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$ in 1D ?2D: $$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

- Gradient is anisotropic
- $mag(\nabla f) = \sqrt{g_x^2 + g_y^2}$ is isotropic
- Orientation of gradient: $\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$

Imagine g_x and g_y as vectors that form two sides of a right triangle.
The result of vector-addition points in the direction of the gradient!

2nd Derivative: From 1D to 2D

$$\frac{\partial^2 f}{\partial x} = f(x+1) + f(x-1) - 2f(x) \quad \text{in 1D ? 2D:} \quad \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

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

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

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Incorporating diagonals, isotropic in increments of 45°

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

Laplacian: use this (negative) version of kernel for convenience

Image Sharpening

- Also known as *enhancement*
- Increases the high frequency components to enhance edges
- $I' = I + \alpha(k * I)$, where k is a high-pass filter kernel and α is a scalar in [0,1]

Image Sharpening: Example 1

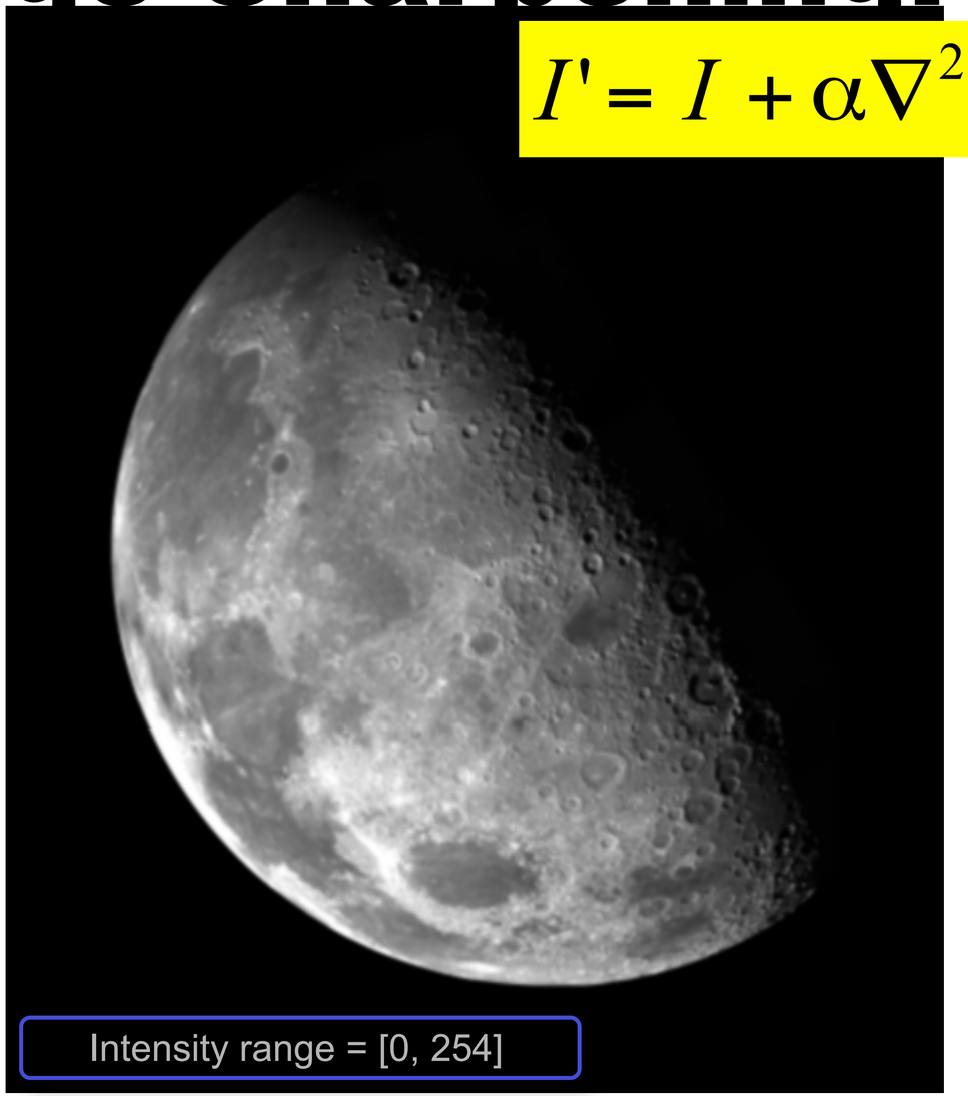
$$I' = I + \alpha \nabla^2 I = I + \alpha (K * I)$$

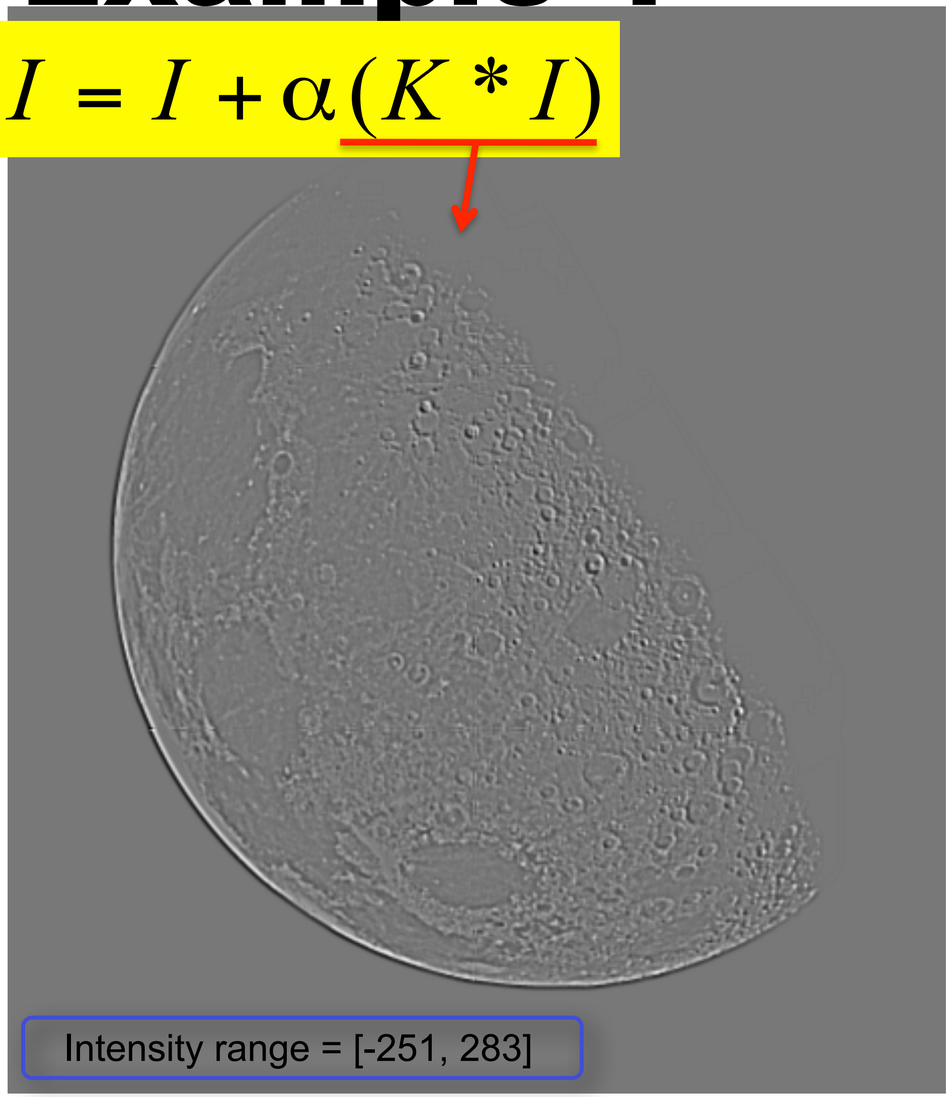
-1	-1	-1
-1	8	-1
-1	-1	-1

Figure from NASA, obtained on DIP

Image Sharpening: Example 1

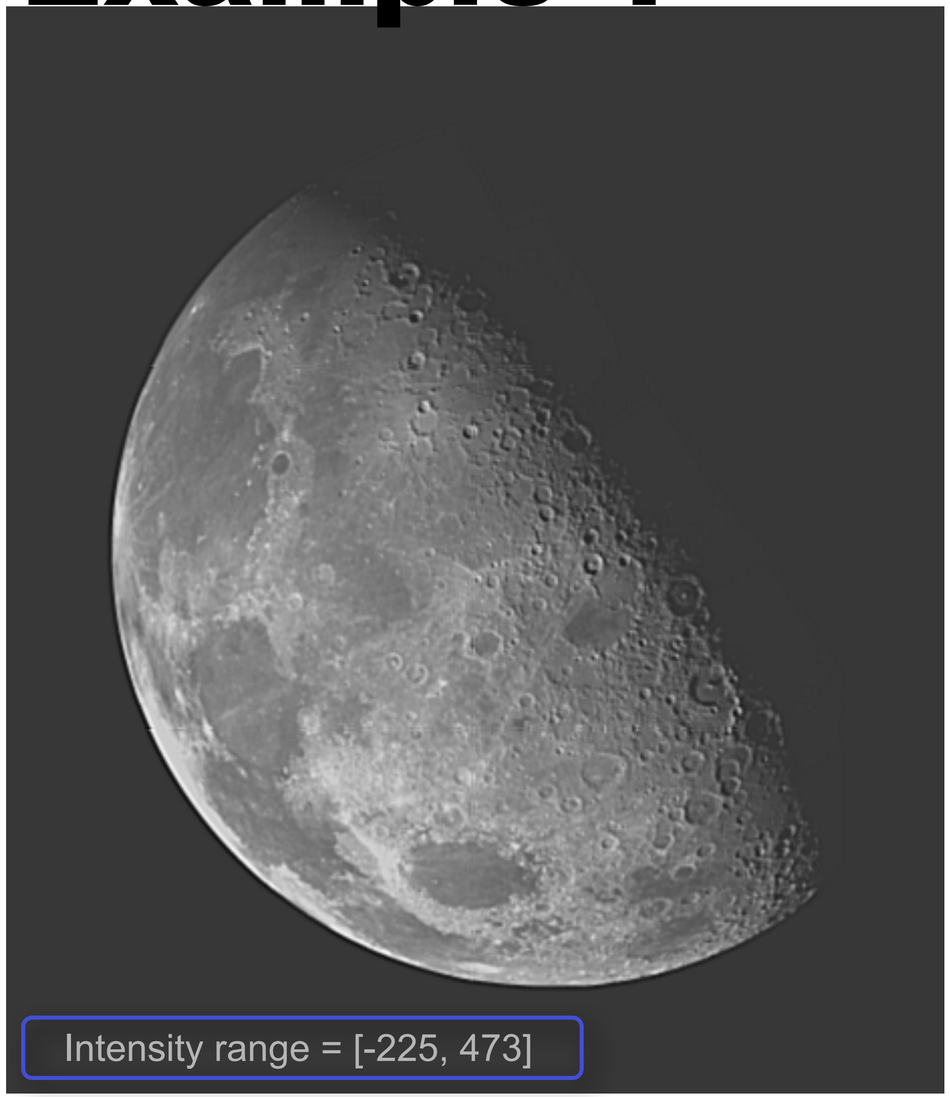
$$I' = I + \alpha \nabla^2 I = I + \alpha (K * I)$$





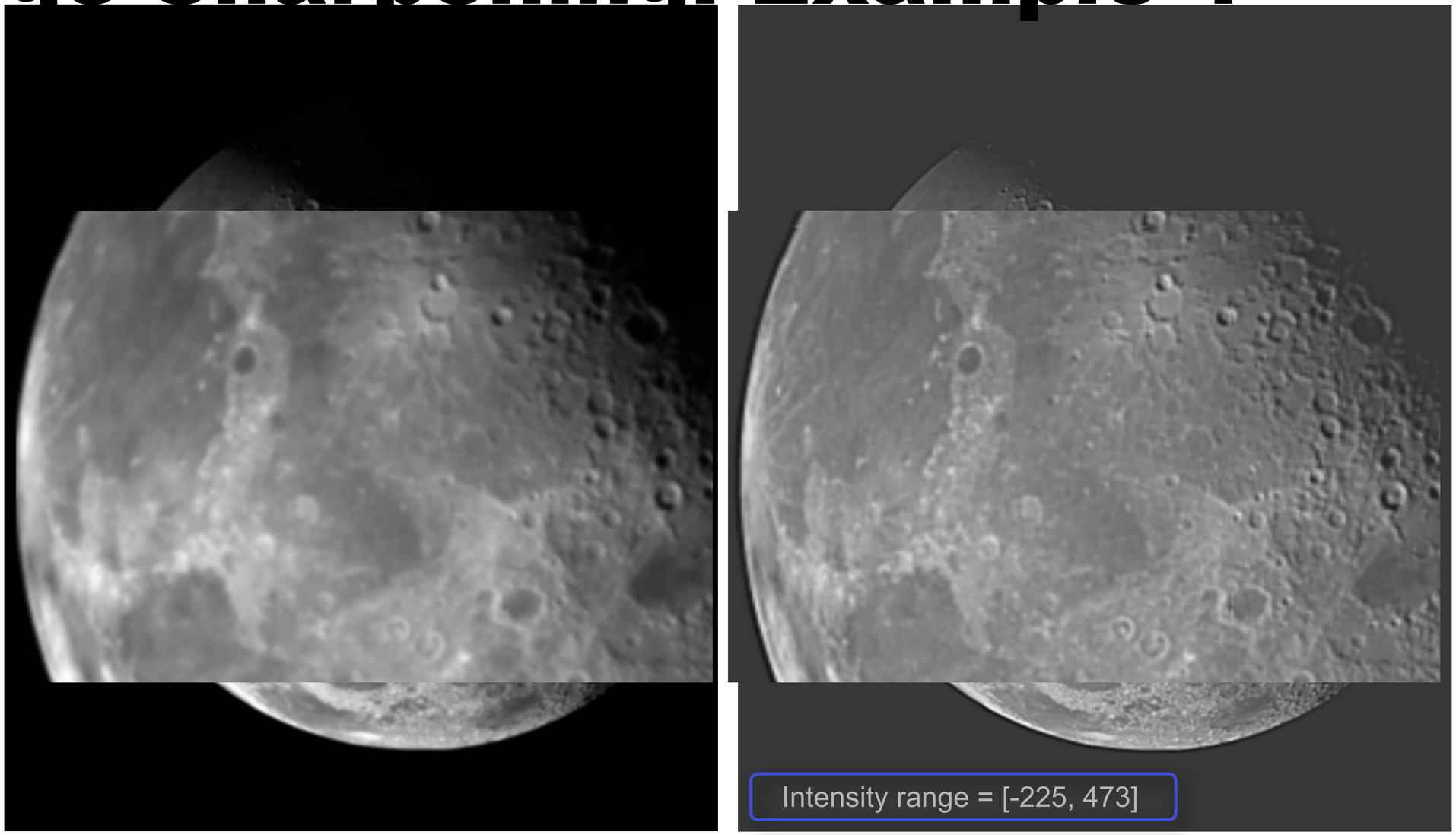
I $K*I$

Image Sharpening: Example 1



I I'

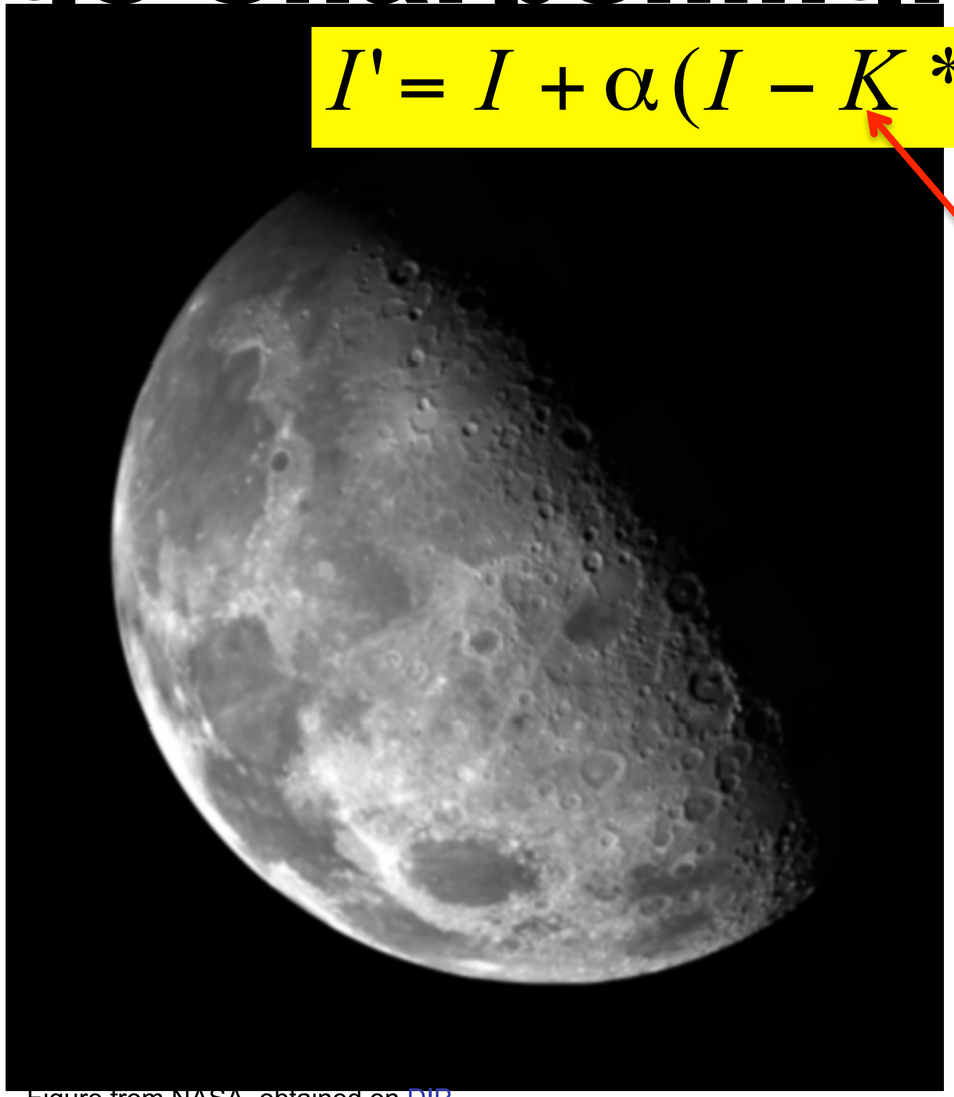
Image Sharpening: Example 1



I I'

Image Sharpening: Example 2

$$I' = I + \alpha(I - K * I) \Leftarrow \text{"Unsharp Mask"}$$



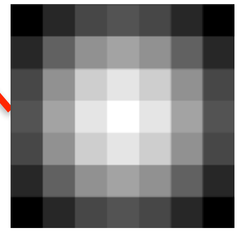


Figure from NASA, obtained on DIP

I

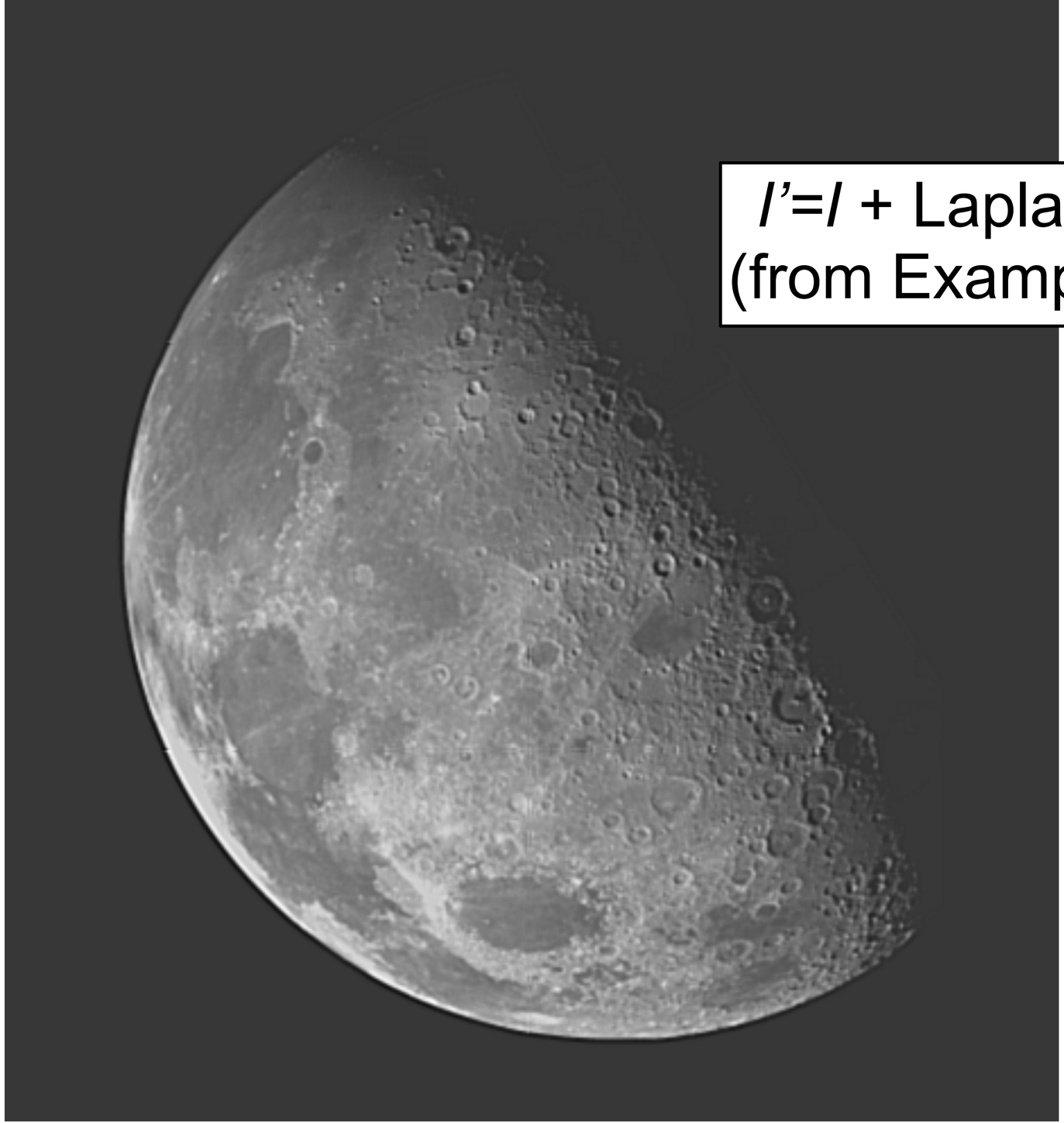
Image Sharpening: Example 2

$$I' = 2I - K * I \Leftarrow \text{"Unsharp Mask"}$$

I I'

I

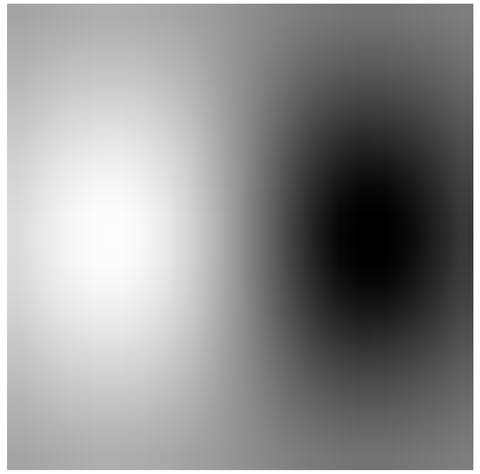
I’= Unsharp Mask
(from Example 2)

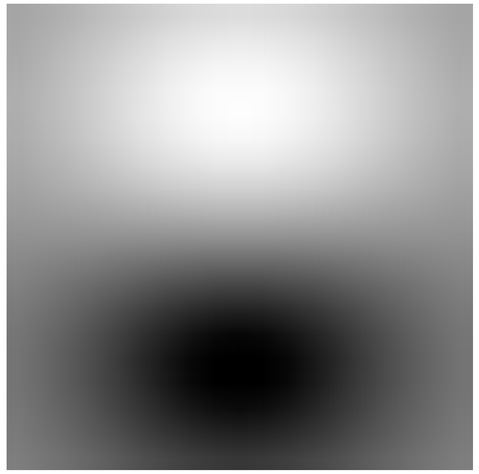


I’=I + Laplacian
(from Example 1)

Filters are Templates

- **Filter at some point can be seen as taking a dot-product between the image and some vector**
- **Filtering the image is a set of dot products**
  - filters look like the effects they are intended to find
  - filters find effects they look like





Slide credit: D.A. Forsyth

Scaled Representations

- Find *correlations/convolutions* at all scales
  - e.g., when finding hands or faces, we don't know what size they will be in a particular image
  - Template size is constant, but image size changes

- Efficient search for *correspondence*
  - look at coarse scales, then refine with finer scales
  - much less cost, but may miss best match

- Examining all *levels of detail*
  - Find edges with different amounts of blur
  - Find textures with different spatial frequencies *(levels of detail)*

Slide credit: D.A. Forsyth