Department of Computer Science
University of Bristol

COMS30030 - Image Processing and Computer Vision

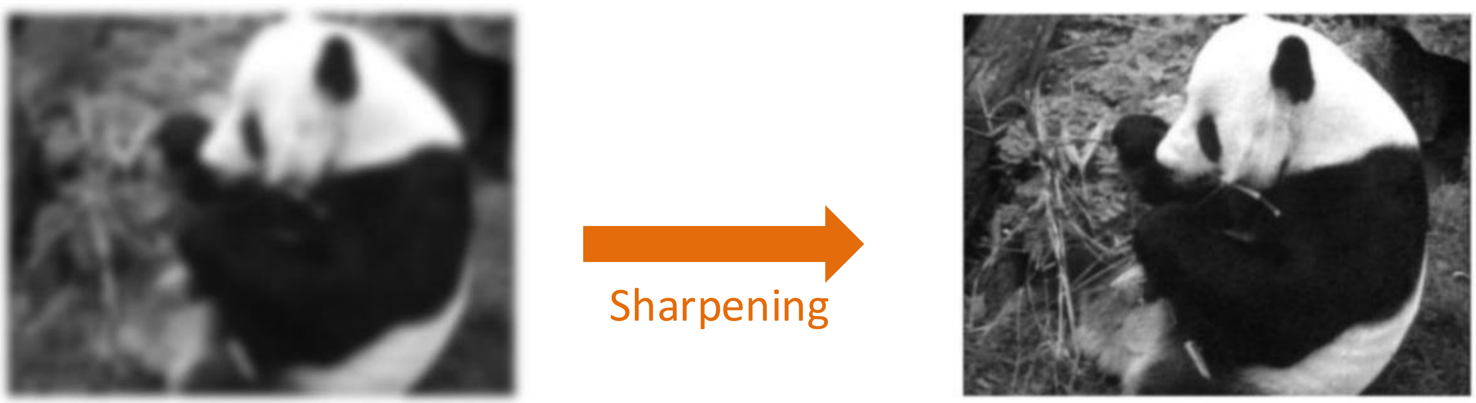
Lecture 02

Filtering Images

Majid Mirmehdi | majid@cs.bris.ac.uk

Image Transforms

- **Image Transform →** a ***new representation*** of the input data (e.g. by re-encoding it in another [parameter] space, e.g. Fourier, Hough, Wavelet, Haar, etc.)
- Image Transforms are classic processing techniques, used in filtering, compression, feature extraction, and much more



Convolution

Convolution = A mathematical operation on two functions (***f*** and ***g***) that produces a third function (***h=f*g***), representing how the shape of one is modified by the other.
(Wikipedia)

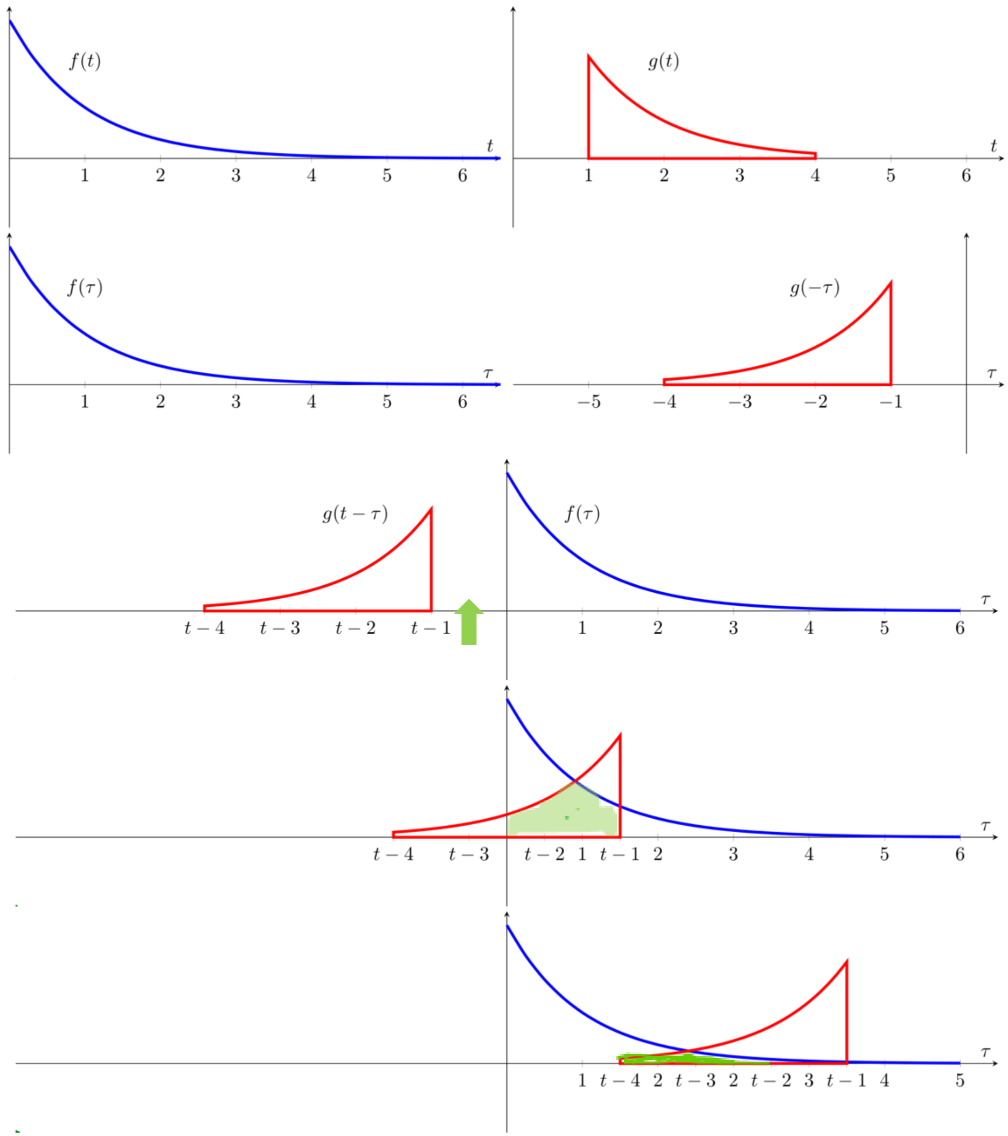
$$(f * g)(t) = \int_{-\infty}^{\infty} \underbrace{f(t-x)}_{\textbf{Signal}} \overbrace{g(x)}^{\textbf{Kernel}} \, \partial t$$

- Used for filtering images in the spatial domain
- Application in other parameter spaces, e.g. frequency domain
- Fundamental to Convolutional Neural Nets for deep Learning
- much, much more

An Intuitive look at Convolution

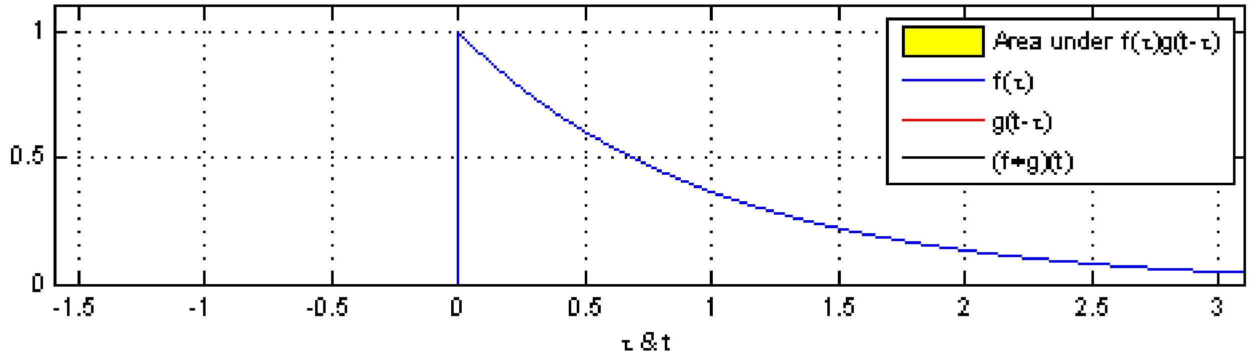
$$f * g = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\,\partial t$$

$$f * g = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,\partial t$$

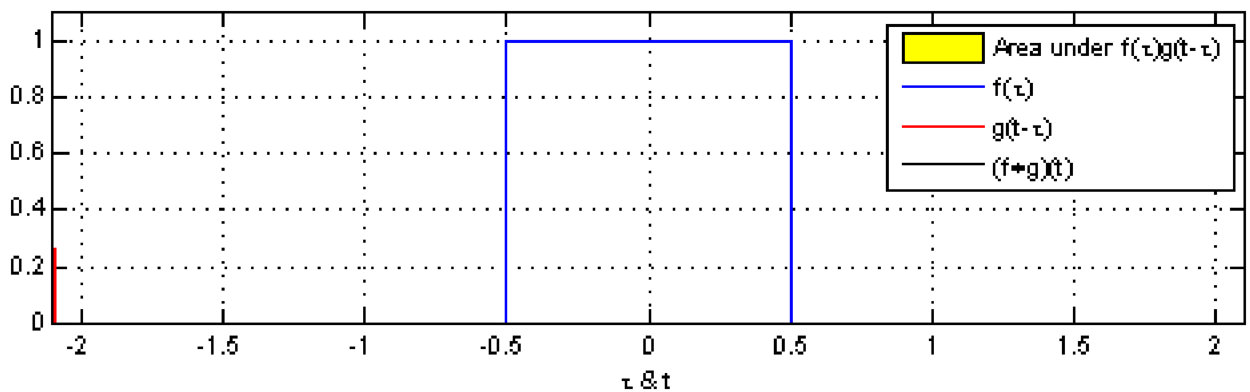


By Krishnavedala

An Intuitive look at Convolution



$$f * g = \int_{-\infty}^{\infty} f(t - \tau) g(\tau)\, \partial t$$



Autocorrelation

Animations Source: Brian Amberg

1D Discrete Convolution

$$g(x) = h(x) * f(x)$$

- f is the signal, h is the convolution filter
  - h has an origin

-1	3	-1

Example 1D kernel

  - Normalization factor (sum of the absolute values of the filter) is also part of the filter!

- The discrete version of convolution is defined as:

$$g(x) = \sum_{m=-s}^{s} f(x-m)h(m) \qquad \text{for } s \geq 1$$

2D Discrete Convolution

- The discrete version of 2D convolution is defined as:

$$g(x,y) = \sum_m \sum_n f(x-m, y-n)h(m,n)$$

		$y-1$	y	$y+1$	
$x-1$		**43**	**12**	**61**	
x		**44**	**45**	**60**	
$x+1$		**43**	**50**	**61**	

	$n=-1$	$n=0$	$n=1$
$m=-1$	**-1**	**0**	**1**
$m=0$	**-2**	**0**	**2**
$m=1$	**-1**	**0**	**1**

f * ***h*** **= -68**

$$\begin{aligned}
& f(x+1,y+1)h(-1,-1) \\
&+ f(x+1,y)h(-1,0) \\
&+ f(x+1,y-1)h(-1,1) \\
&+ f(x,y+1)h(0,-1) \\
&+ f(x,y)h(0,0) \\
&+ f(x,y-1)h(0,1) \\
&+ f(x-1,y+1)h(1,-1) \\
&+ f(x-1,y)h(1,0) \\
&+ f(x-1,y-1)h(1,1)
\end{aligned}$$

2D Discrete Correlation

- The discrete version of 2D correlation is defined as:

$$g(x, y) = \sum_{m} \sum_{n} f(x+m, y+n) h(m, n)$$

	n=-1	*n*=0	*n*=1
m=-1	**-1**	**0**	**1**
m=0	**-2**	**0**	**2**
m=1	**-1**	**0**	**1**

		$y-1$	y	$y+1$	
$x-1$		**43**	**12**	**61**	
x		**44**	**45**	**60**	
$x+1$		**43**	**50**	**61**	

68

Correlation ≡ Convolution when kernel is symmetric under 180° rotation, e.g.

1	**2**	**1**
1	**2**	**1**
1	**2**	**1**

Spatial Low/High Pass Filtering

Low Pass

$$h = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

$\boldsymbol{f}$

$\boldsymbol{f * h}$

High Pass

$$h = \begin{array}{|c|c|c|} \hline 0 & -1 & 0 \\ \hline -1 & 5 & -1 \\ \hline 0 & -1 & 0 \\ \hline \end{array}$$

Properties of Convolution

- Commutative:

$$f * h = h * f$$

- Associative

$$(f * g) * h = f * (g * h)$$

- Distributes over addition

$$f * (g + h) = (f * g) + (f * h)$$

- Scalars factor out

$$af * g = f * ag = a(f * g)$$

- Identity:

Given unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$ then $f * e = f$

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

$$g(x,y) = \sum_{m}\sum_{n} f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

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

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10							

$$g(x,y) = \sum_{m}\sum_{n} f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20						

$$g(x,y) = \sum_{m}\sum_{n} f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

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

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20	30					

$$g(x,y) = \sum_{m}\sum_{n} f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20	30	30				

$$g(x,y) = \sum_m \sum_n f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20	30	30				
				?					

$$g(x,y) = \sum_m \sum_n f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20	30	30				
						?			
				50					

$$g(x,y) = \sum_m \sum_n f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Image filtering example

$h(m,n)$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f(x,y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x,y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$g(x,y) = \sum_{m}\sum_{n} f(x+m, y+n)h(m,n)$$

Credit: S. Seitz

Normalising the Convolution Result!

The weights will affect pixel values and the result could fall outside the 0-255 range ➔ Normalise them such that they sum to 1.

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

Smooth

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

3x3 Gaussian Blur

$$\frac{-1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Unsharp Masking

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

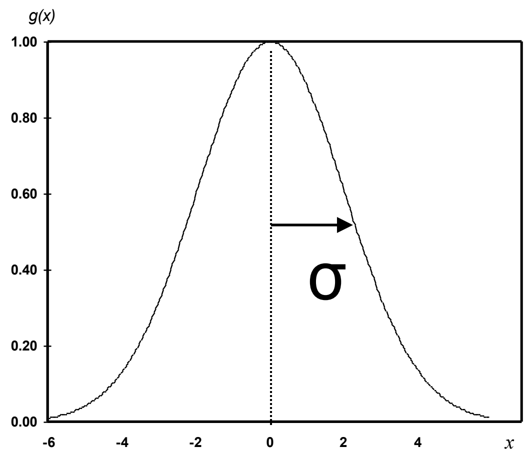
Sharpen

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

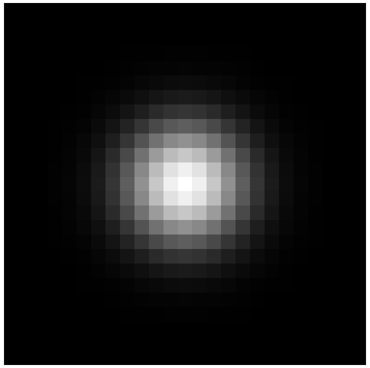
Basic Edge Detector

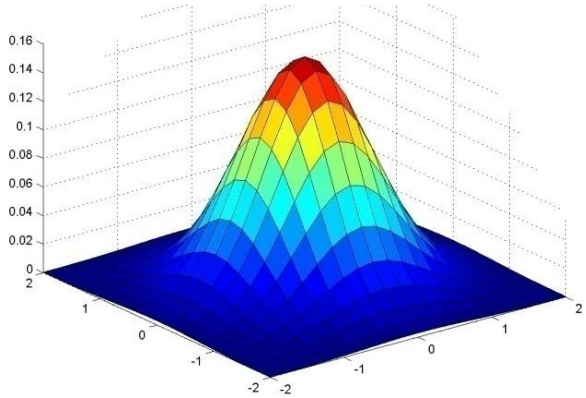
Gaussian Filter Kernel

The *Gaussian* filter is very commonly used in image processing. The parameter σ determines the width of the filter and hence the amount of smoothing.



$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

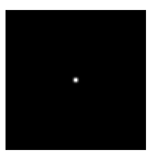




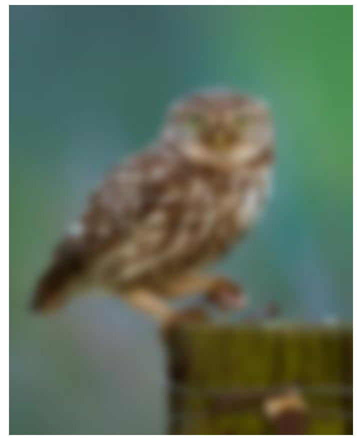
$$g(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

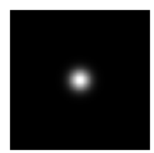
Applying the Gaussian filter

Original

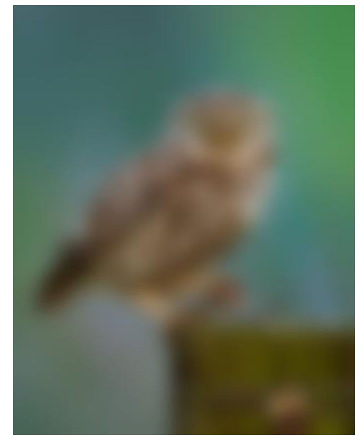


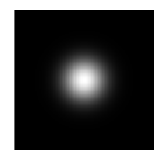
$\boldsymbol{\sigma = 1}$



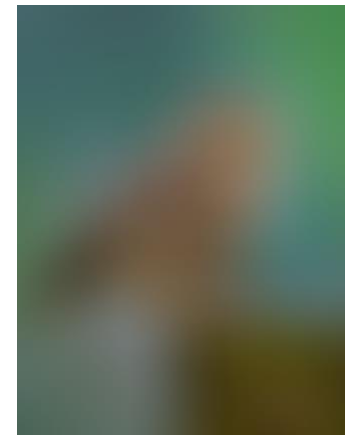


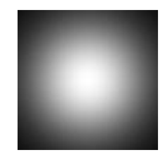
$\boldsymbol{\sigma = 5}$





$\boldsymbol{\sigma = 10}$



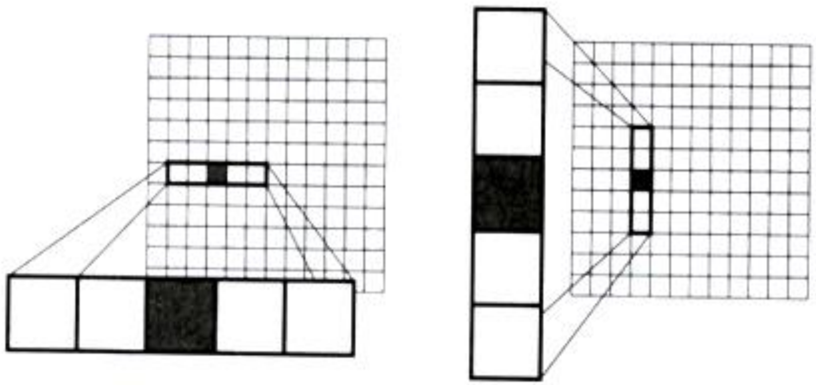


$\boldsymbol{\sigma = 30}$

Source: Noah Snavely

Efficient Gaussian Filtering

Filtering with a 2D Gaussian can be achieved faster using two 1D Gaussian filters! This is because the linear Gaussian kernel is *separable*.



$$K = \frac{1}{256}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \frac{1}{16}\begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} x \frac{1}{16}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

Filtering an m x m image with an n x n kernel is $\mathcal{O}(m^2n^2)$ complexity, but with a separable kernel, it would be $\mathcal{O}(m^2n)$.
→ a significant reduction in computations!

Sources from fiveko.com and E.G.M. Petrakis

Separability of the Gaussian Filter

Separability means that a 2D convolution can be reduced to the product of two 1D convolutions - one on the rows and one on the columns:

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

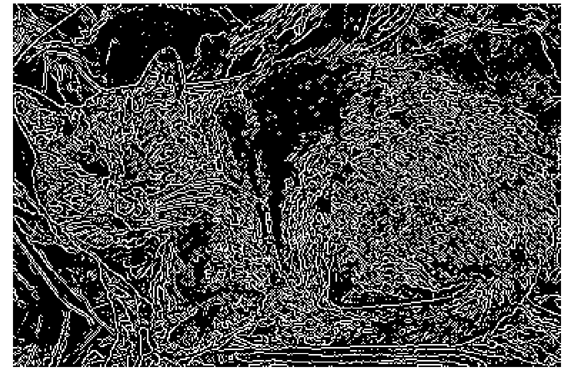
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}}\right)$$

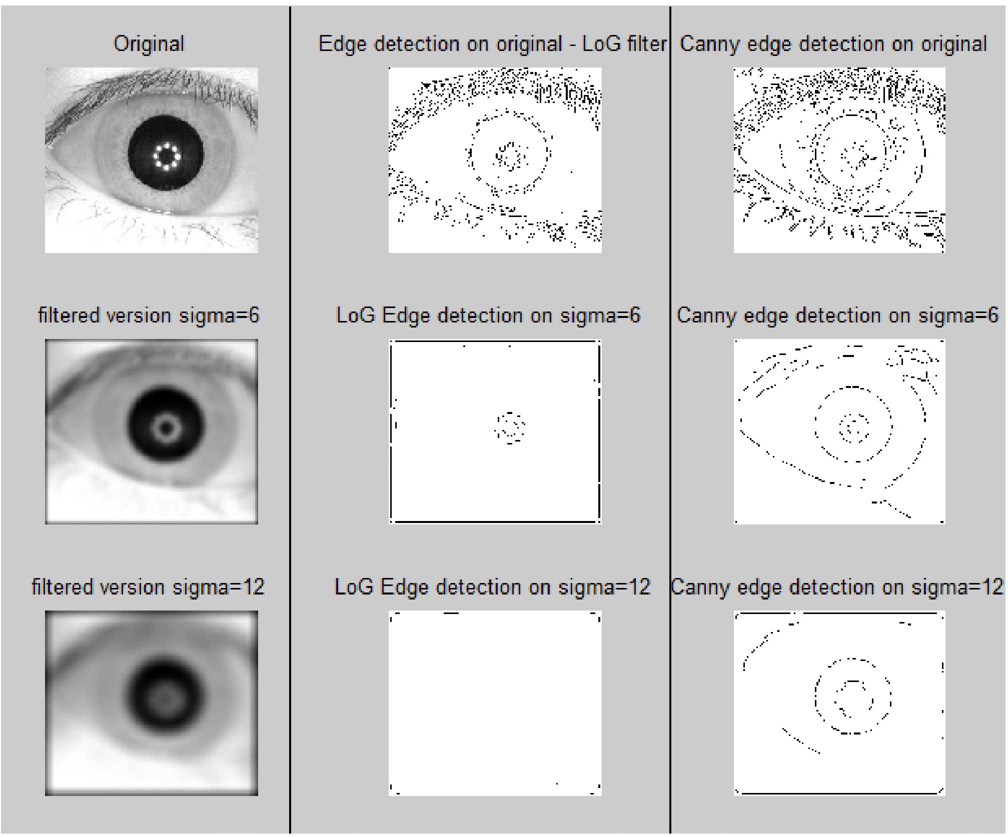
$$= g(x)\, g(y)$$

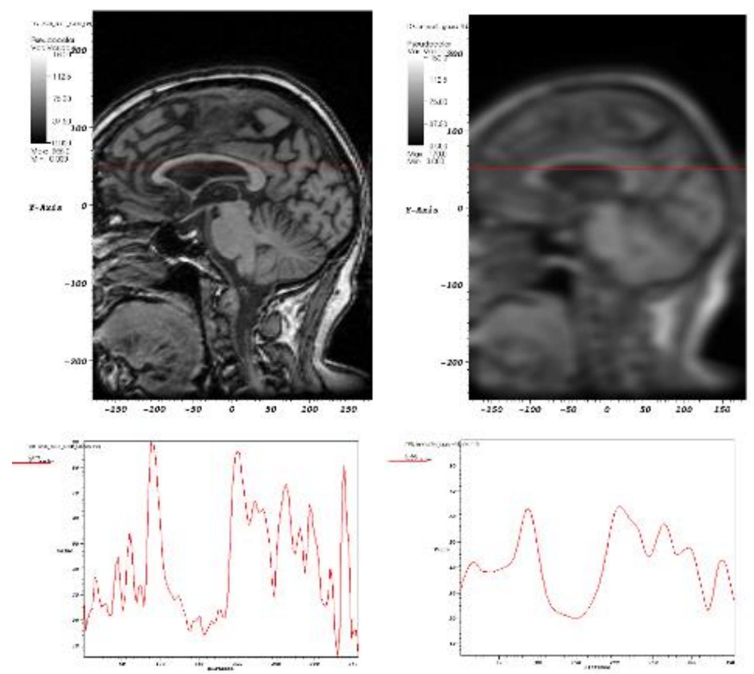
Sources from fiveko.com and E.G.M. Petrakis

Example benefit of the Gaussian Filter

When we smooth an image, the easier the next stage of processing, in this case edge detection



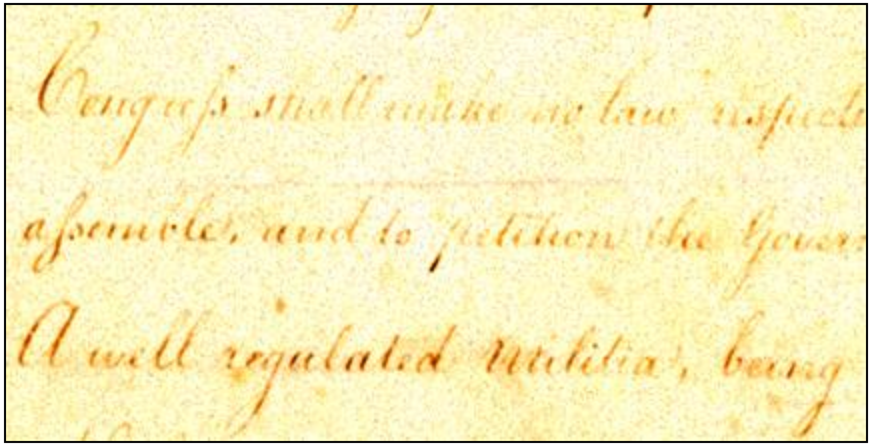




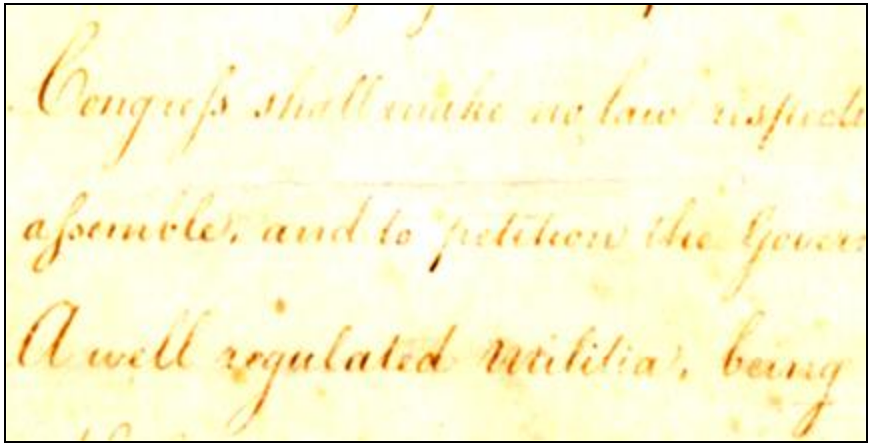
Noise Removal: The Median Filter

- Returns the median value of the pixels in a neighborhood
- Relationship to a uniform blurring filter?
- Is non-linear

original



median filtered



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124, 125, 126, 127, 150

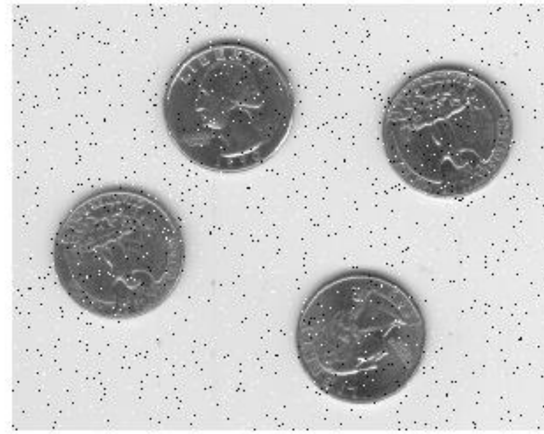
Median value: 124

Slide adapted from Richard Peters II

Noise Removal: The Median Filter

Original

salt & pepper noise added



3x3 averaging filter

3x3 median filter

Noise Removal: The Median Filter

original

corrupted (*i.e.* p=5% that a bit is flipped)

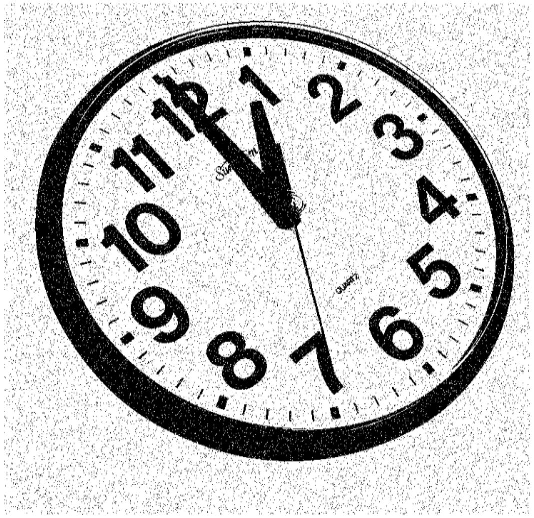
median filtered

Slide adapted from Richard Peters II

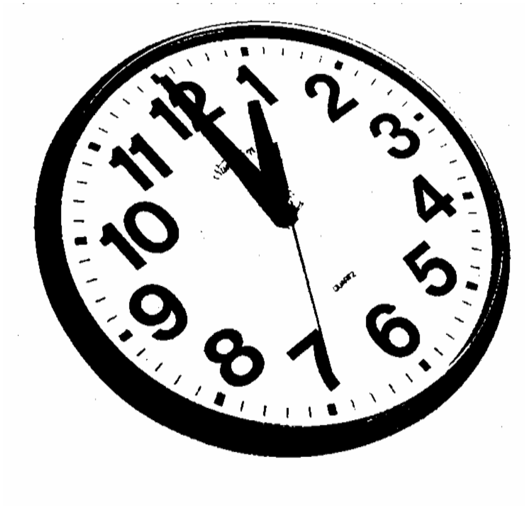
Median Filter: the algorithm

1. Let I be a monochrome image.
2. Let Z define a neighborhood of arbitrary shape.
3. At each pixel location, $\mathbf{p}=(x,y)$, in I …
4. … select the n pixels in the Z-neighborhood of $\mathbf{p}$,
5. … sort the n pixels in the neighborhood of $\mathbf{p}$ by value into a list $L(j)$ for $j = 1,\ldots,n$.
6. The output value at $\mathbf{p}$ is $L(m)$, where $m = \lfloor n/2 \rfloor + 1$.

original



median filtered



Slide by Richard Alan Peters II

Sharpening Revisited

$h(m, n)$

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

Sharpening spatial filters seek to highlight fine detail

- Remove blurring from images
- Highlight edges

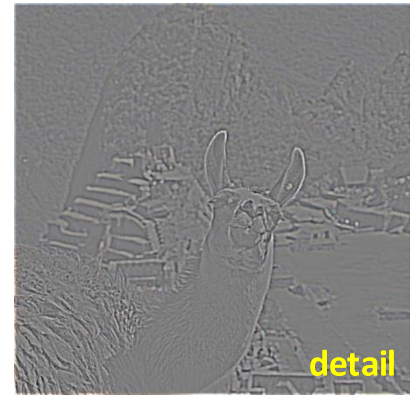
Sharpening filters are based on *spatial differentiation.*

Sharpening

What does blurring take away?

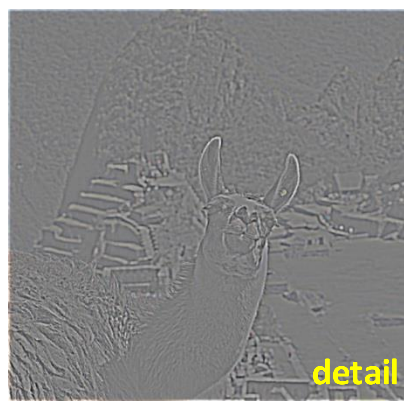
−

=



Let’s add it back:

$+ \ \alpha$



=

Photo credit: https://www.flickr.com/photos/geezaweezer/16089096376/

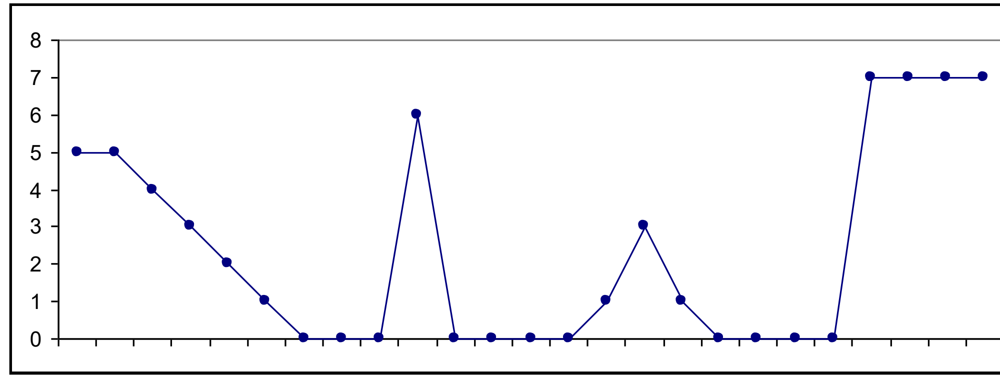
1^{st} Derivative

The 1^{st} derivative of a function is:

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

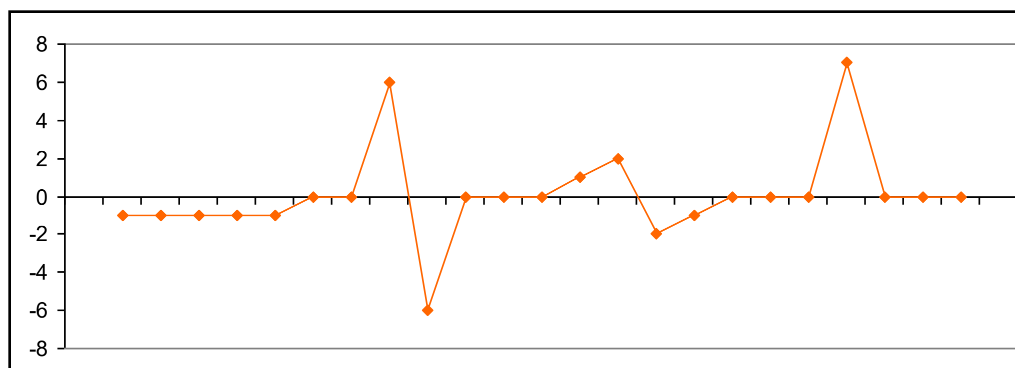
→ it's the difference between subsequent values and measures the rate of change of the function.

$f(x)$



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

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



$\frac{\partial f}{\partial x}$

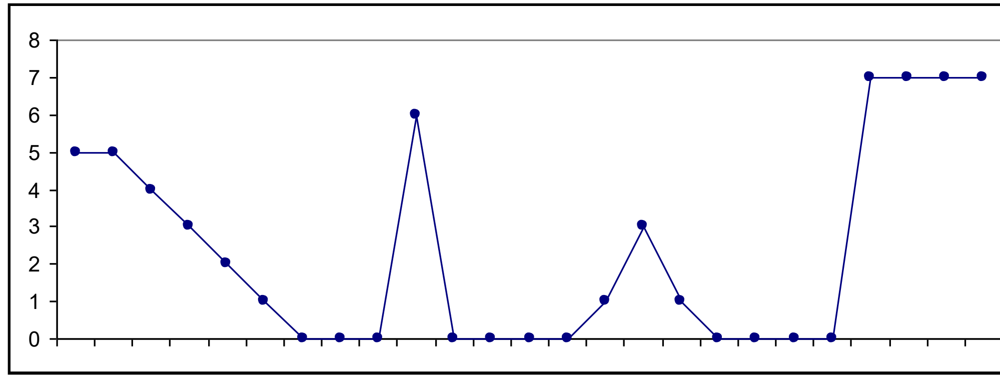
2nd Derivative

The 2nd derivative of a function is:

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

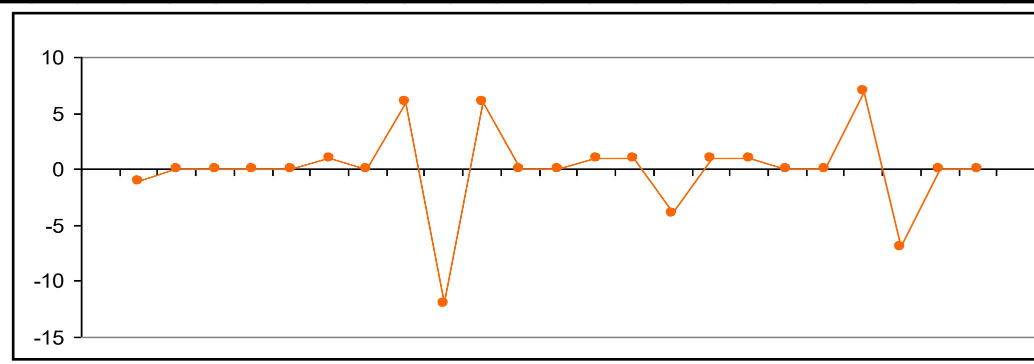
→ Takes into account the values both before and after the current value.

$f(x)$



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

	-1	0	0	0	0	1	0	6	-12	6	0	0	1	1	-4	1	1	0	0	7	-7	0	0	



$\frac{\partial^2 f}{\partial x^2}$

Using Second Derivatives For Image Enhancement

Now in 2-dimensions for images:

The 2nd derivative is more useful for image sharpening than the 1st derivative

- ✓ Stronger response to fine detail

The *Laplacian Filter:*

- ✓ Isotropic
- ✓ One of the simplest filters for sharpening

0	1	0
1	-4	1
0	1	0

The Laplacian

The Laplacian is defined as:

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

where the partial 1st order derivative in the x direction is:

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

and in the y direction:

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

So, the Laplacian can be:

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

0	1	0
1	-4	1
0	1	0

About the Laplacian Operator

As it is a derivative operator:

– it highlights gray-level discontinuities in an image.

– it de-emphasizes regions with slowly varying gray-levels.

– Very sensitive to noise as taking derivatives increases noise!

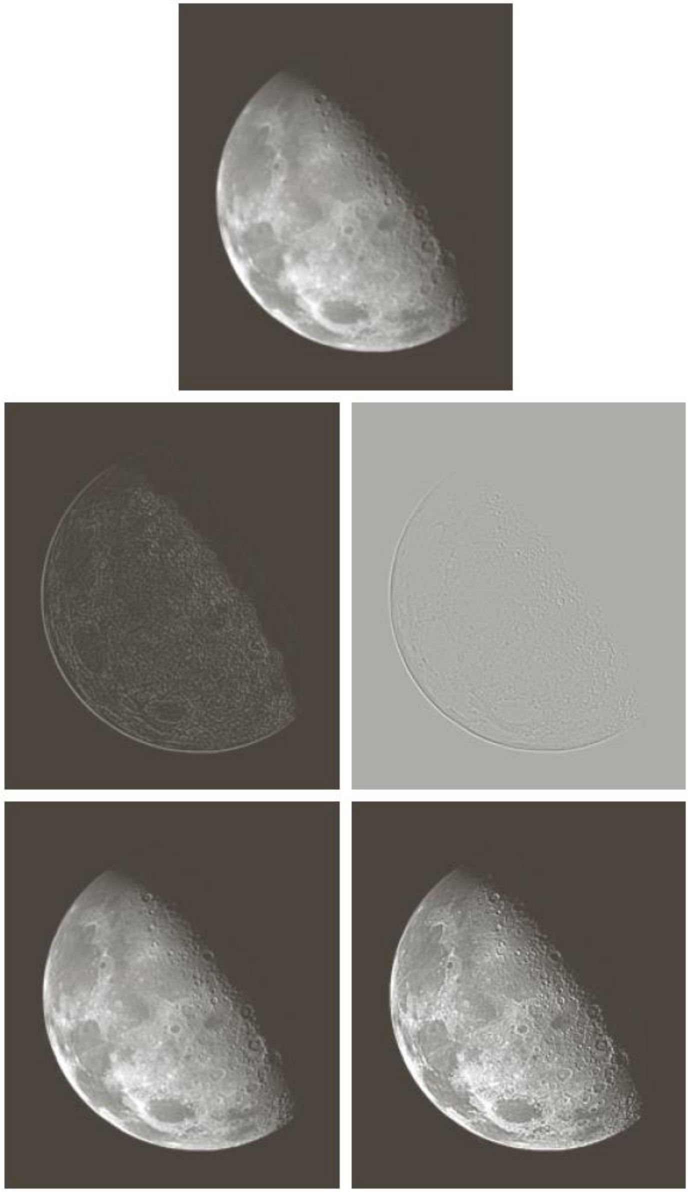
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a
b c
d e

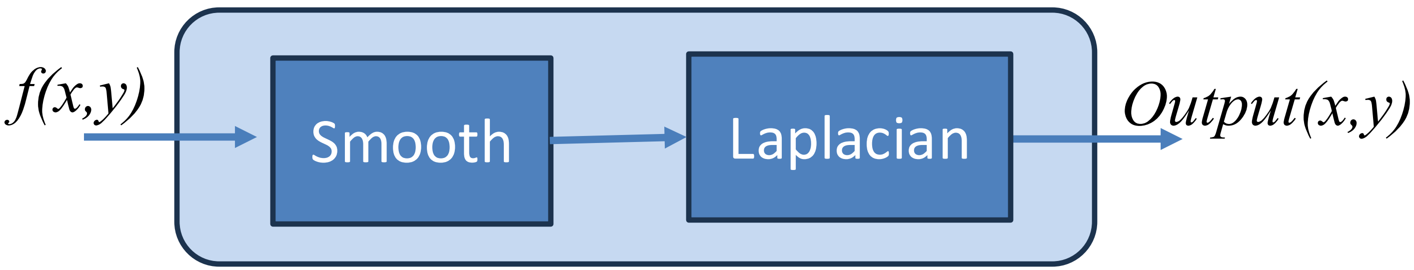


(a) Blurred image of the North Pole. (b) Laplacian without scaling and (c) with scaling, (d) and (e) Image sharpening using two different masks.

LoG: Laplacian of Gaussian

Because the Laplacian is noise sensitive, it is always combined with a smoothing operation ☺

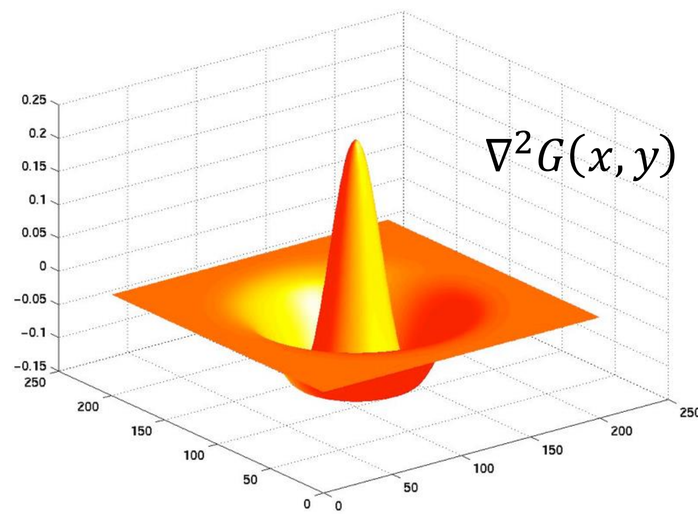
$$Output(x, y) = \nabla^2(f(x, y) * G(x, y))$$



$$\nabla^2(f(x, y) * G(x, y)) = \nabla^2 G(x, y) * f(x, y)$$

Laplacian of a Gaussian-filtered image

Laplacian of Gaussian of a filtered image



Summary: Image Filtering

- Images are often corrupted by random variations in intensity, illumination, or have poor contrast and cannot be used directly

- *Filtering* allows us to achieve:
  - *Enhancement:* improves contrast
  - *Smoothing:* removes noise
  - *Template matching*: detects known patterns
  - Feature Extraction: provides clues about objects etc., for further analysis
  - Many other uses…

Adapted from E.G.M. Petrakis