Image processing
“Linear filtering”

Mathieu Delalandre
University of Tours, Tours city, France
mathieu.delalandre@univ-tours.fr

Lecture available at http://mathieu.delalandre.free.fr/teachings/image.html

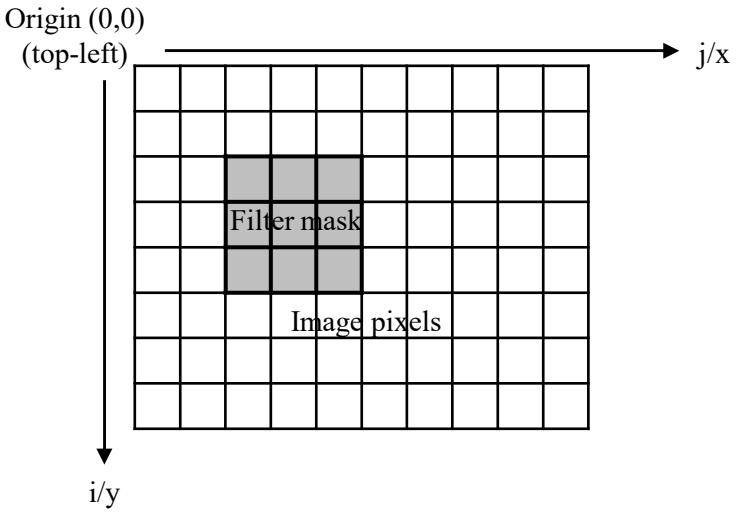
Linear filtering

1. Fundamentals of linear filtering
2. Mean filtering
3. Derivative filters - 1st order
4. Derivative filters - 2sd order
5. Combining operators

Fundamentals of linear filtering (1)

Spatial filtering is one of the principal tools for image processing. The term “filtering” is borrowed from frequency domain processing, to accept (i.e. to pass) or to reject certain frequency components.

Spatial filter consists of (1) a neighborhood (typically a small rectangle) referred to as a filter mask (2) a predefined operation that is performed on the image pixel encompassed by this neighborhood.



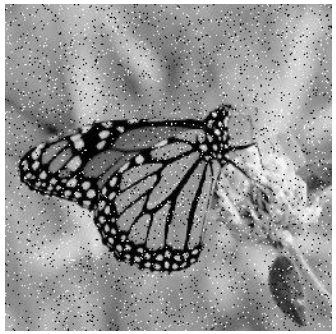
Linear filtering is related to a special case of spatial filtering, where the operation performed on the image pixels is linear. Otherwise, the filter is nonlinear.

Fundamentals of linear filtering (2)

Spatial filtering is one of the principal tools for image processing. The term “filtering” is borrowed from frequency domain processing, to accept (i.e. to pass) or to reject certain frequency components.

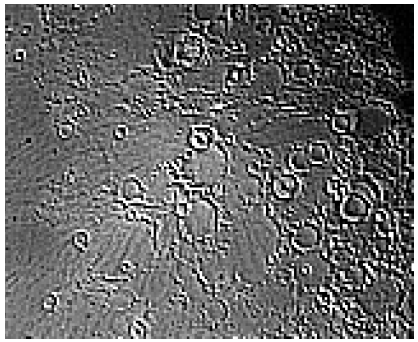
It can be used for a broad spectrum of applications. The main is image enhancement, the other include edge and point of interest detection.

Image enhancement



Edge detection

Image sharpening



Interest point detection

Fundamentals of linear filtering (3)

Convolution and correlation: there are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is convolution and the other is correlation.

	Equation	Comments
1D convolution	$F(x) = g(x) \bullet f(x) = \int_{u=-\infty}^{+\infty} g(x) \times f(x-u)\, du$	Convolution *F(x)* of two functions *f(x)*, *g(x)* is defined as integral.
2D discrete convolution	$F(x,y) = w(x,y) \bullet f(x,y)$ $F(x,y) = \sum_{u=-a}^{+a} \sum_{v=-b}^{+b} w(u,v) \times f(x-u, y-v)$	When convolution is applied to digital image, the above formulation changes in tow ways: (i) a double integral must be used. (ii) integration must be changed into discrete summation, in this new form, *w(x,y)* is the spatial convolution mask.
2D discrete correlation	$F(x,y) = w(x,y) \circ f(x,y)$ $F(x,y) = \sum_{u=-a}^{+a} \sum_{v=-b}^{+b} w(u,v) \times f(x+u, y+v)$	The fact that the mask has to be inverted before it is applied is inconvenient for visualizing the process of convolution. The usual approach is to invert the mask, convolution can be then reformulated in term of correlation.

Fundamentals of linear filtering (4)

Convolution and correlation: there are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is convolution and the other is correlation.

e.g. $a=b=1$, w is

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

f is

P_{11}	P_{12}	P_{13}
P_{21}	P_{22}	P_{23}
P_{31}	P_{32}	P_{33}

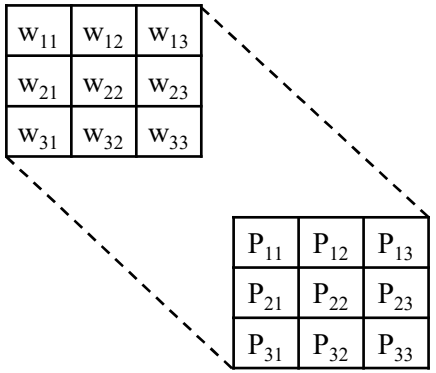
we consider here $f(x,y)$ with $x,y=2$

Correlation is

$$F(x,y) = w(x,y) \circ f(x,y)$$

$$F(x,y) = \sum_{u=-a}^{+a} \sum_{v=-b}^{+b} w(u,v) \times f(x+u, y+v)$$

$$\begin{aligned} F(x,y) = {} & w_{11}P_{11} + w_{12}P_{12} + w_{13}P_{13} \\ & + w_{21}P_{21} + w_{22}P_{22} + w_{23}P_{23} \\ & + w_{31}P_{31} + w_{32}P_{32} + w_{33}P_{33} \end{aligned}$$

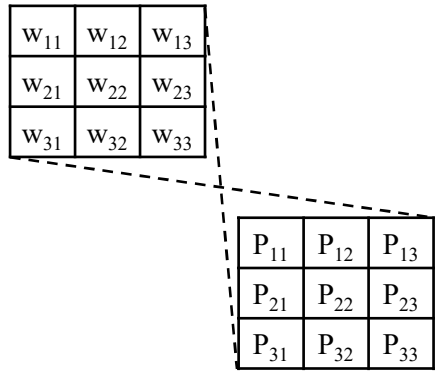


Convolution is

$$F(x,y) = w(x,y) \bullet f(x,y)$$

$$F(x,y) = \sum_{u=-a}^{+a} \sum_{v=-b}^{+b} w(u,v) \times f(x-u, y-v)$$

$$\begin{aligned} F(x,y) = {} & w_{11}P_{33} + w_{12}P_{32} + w_{13}P_{31} \\ & + w_{21}P_{23} + w_{22}P_{22} + w_{23}P_{21} \\ & + w_{31}P_{13} + w_{32}P_{12} + w_{33}P_{11} \end{aligned}$$



Fundamentals of linear filtering (5)

Convolution and correlation: there are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is convolution and the other is correlation.

Convolution equals correlation when

if we rotate w of π

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

$+\pi$

w_{33}	w_{32}	w_{31}
w_{23}	w_{22}	w_{21}
w_{13}	w_{12}	w_{11}

if w is symmetric

c	a	c
a	b	a
c	a	c

$+\pi$

c	a	c
a	b	a
c	a	c

rotated w is then similar

Fundamentals of linear filtering (6)

Convolution and correlation: there are two closely related concepts that must be understood clearly when performing linear spatial filtering. One is convolution and the other is correlation.

The discrete unit impulse *δ(x,y)* is a generalized function such that it is zero for all values of the parameter except at one point. In the context of mathematic and signal processing, it is often referred to as the Dirac delta function.

$\delta(x, y)$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	**1**	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Correlation of a mask *w(u,v)* with discrete unit impulse *δ(x,y)* yields a rotated version of the mask at the position of the impulse.

$w(u, v)$

1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

$w(u, v) \circ \delta(x, y)$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	**9**	**8**	**7**	0	0
0	0	**6**	**5**	**4**	0	0
0	0	**3**	**2**	**1**	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Convolution of a mask *w(u,v)* with discrete unit impulse *δ(x,y)* yields a copy (i.e. not rotated) of the mask at the position of the impulse.

$w(u, v)$

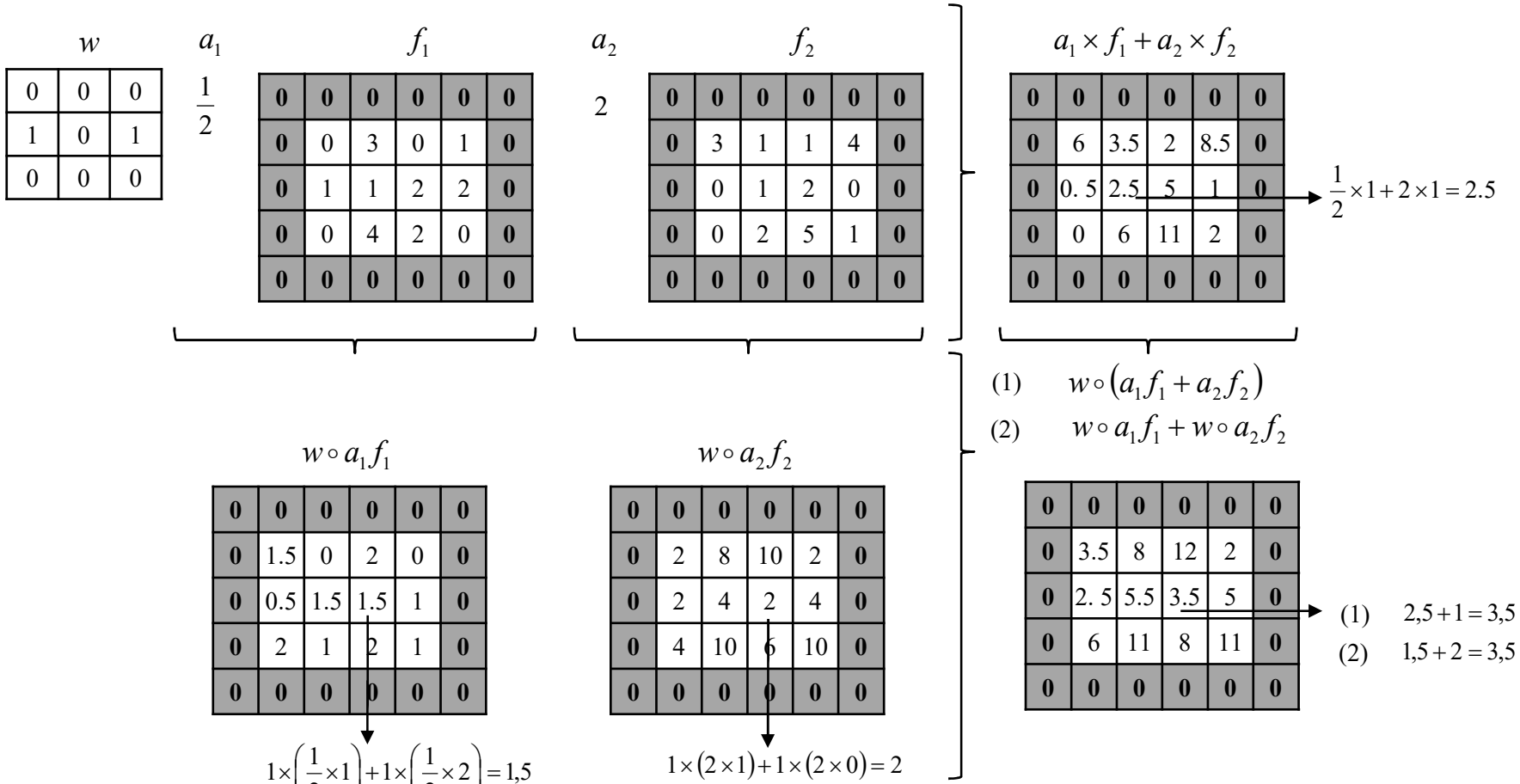
1	**2**	**3**
4	**5**	**6**
7	**8**	**9**

$w(u, v) \bullet \delta(x, y)$

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	**1**	**2**	**3**	0	0
0	0	**4**	**5**	**6**	0	0
0	0	**7**	**8**	**9**	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

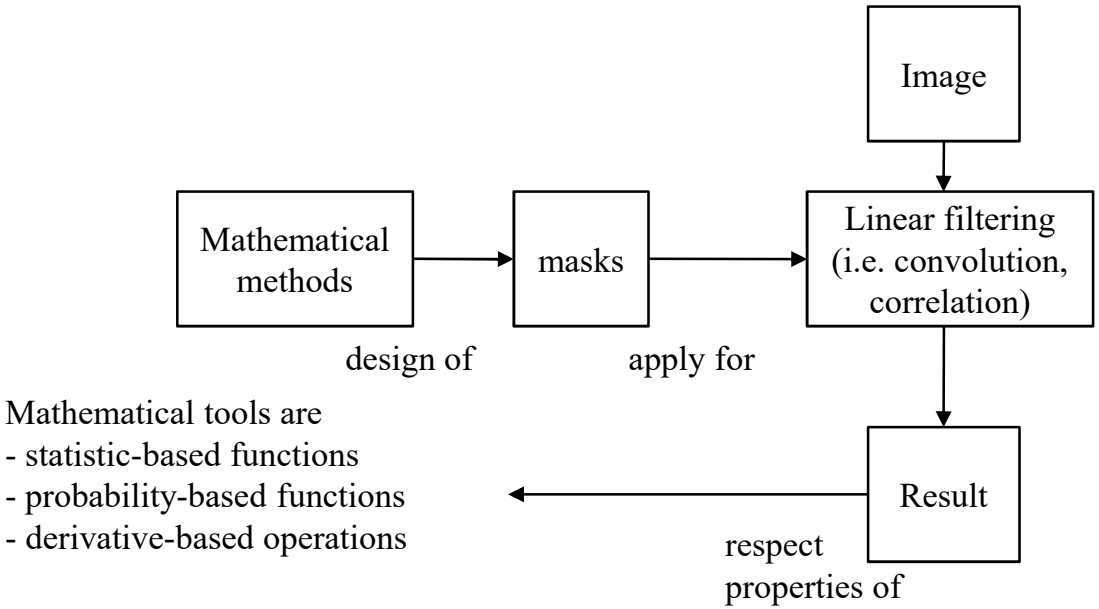
Fundamentals of linear filtering (7)

Preserving operations: the linear operators, convolution and correlation, preserve the operations of vector addition and scalar multiplication i.e. $w \circ a_1 f_1 + w \circ a_2 f_2 = w \circ (a_1 f_1 + a_2 f_2)$



Fundamentals of linear filtering (8)

Generating spatial filter mask: convolution and correlation are ways to process input of pixel. The real challenge with linear filtering is to generate spatial filter masks *w* and their coefficients. These coefficients are defined according to mathematical methods, keeping in mind that all we can do with linear filtering is to implement sum of products.



Fundamentals of linear filtering (9)

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Linear filtering

1. Fundamentals of linear filtering
2. Mean filtering
3. Derivative filters - 1st order
4. Derivative filters - 2sd order
5. Combining operators

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Mean filtering (1)

Filters	Mask	Type	Coefficient rules		
Mean filter	a a a a a a a a a	Symmetric	$a = \frac{1}{n^2}$	$\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 1$	
Parametric filter	a a a a b a a a a ; a a b a a	Symmetric	$a, b > 0$	$\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 1$	specification of a,b is free
Gaussian filter	a b a b c b a b a	Symmetric	$a, b, c > 0$	$\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 1$	a,b,c are defined according to a gaussian function

Mean filtering (1)
"Mean (or averaging) filtering"

Mean filtering: considering a mask *w*, a mean filtering is the standard average of the pixel under the mask.

a	a	a
a	a	a
a	a	a

$$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 1$$

$$a = \frac{1}{n^2}$$

$$\frac{1}{9}\times \begin{array}{|c|c|c|}\hline 1&1&1\\\hline 1&1&1\\\hline 1&1&1\\\hline\end{array}$$

0	**0**	**0**	**0**	**0**	**0**
0	0	0	0	0	**0**
0	0	1	2	2	**0**
0	0	0	2	2	**0**
0	**0**	**0**	**0**	**0**	**0**

$\rightarrow \frac{1}{9}\times$

0	**0**	**0**	**0**	**0**	**0**
0	1	3	5	4	**0**
0	1	5	9	8	**0**
0	1	5	9	8	**0**
0	**0**	**0**	**0**	**0**	**0**

$$\frac{1}{9}\times(0+0+0+0+1+2+0+0+2)=\frac{5}{9}$$

image with salt and pepper

mean filter 3×3

mean filter 5×5

Mean filtering (2)
"Parametric filtering"

Parametric mean filtering: considering a mask *w*, a parametric mean filtering is a weighted average of the pixel under the mask.

	a	
a	b	a
	a	

a	a	a
a	b	a
a	a	a

$$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 1$$

specification of a, b is free

$$\frac{1}{12}\times\begin{bmatrix}1&1&1\\1&4&1\\1&1&1\end{bmatrix}$$

0	**0**	**0**	**0**	**0**	**0**
0	0	0	0	0	**0**
0	0	1	2	2	**0**
0	0	0	2	2	**0**
0	**0**	**0**	**0**	**0**	**0**

$\rightarrow \frac{1}{12}\times$

0	**0**	**0**	**0**	**0**	**0**
0	1	3	5	4	**0**
0	1	8	15	14	**0**
0	1	5	15	14	**0**
0	**0**	**0**	**0**	**0**	**0**

$$\frac{1}{12}\times(0+0+0+0+4\times1+2+0+0+2)=\frac{8}{12}$$

Comparing to the mean filter, the parametric filter is less restrictive.

More b is higher, less the filter will have impact on salt and pepper noise.

image with salt and pepper

mean filter 5×5

parametric low-pass filter

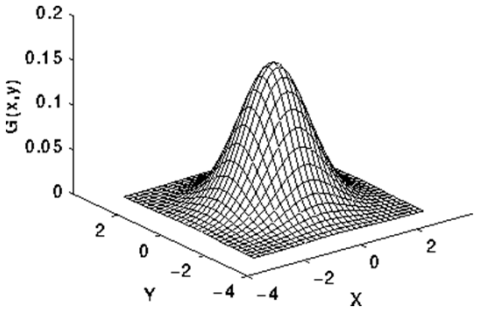
Mean filtering (3)
"Gaussian filtering"

Gaussian filtering: considering a mask *w*, a Gaussian filtering is a parametric mean filtering where weights are defined according to a Gaussian function.

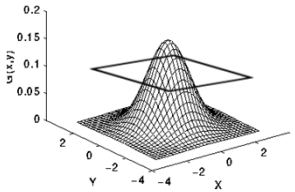
2D Gaussian function (with mean = 0) is

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

Considering σ = 1, the values are



We can catch the values within a *w* mask



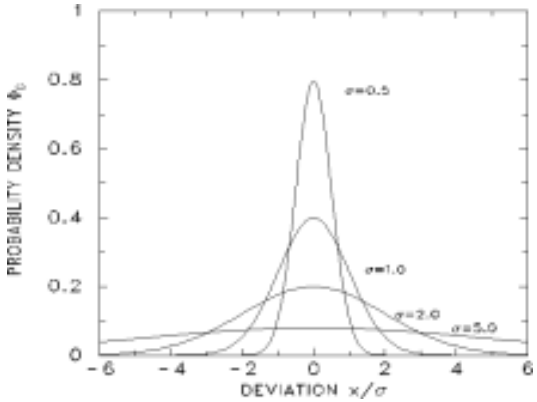
$$\rightarrow \quad \frac{1}{209} \times \begin{array}{|c|c|c|}\hline 16 & 26 & 16 \\ \hline 26 & 41 & 26 \\ \hline 16 & 26 & 16 \\ \hline \end{array} \quad \Leftrightarrow \quad \begin{array}{|c|c|c|}\hline a & b & a \\ \hline b & c & b \\ \hline a & b & a \\ \hline \end{array}$$

$$\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 1$$

initial image

gaussian filtering
with a se of size = 5×5, and σ = 5

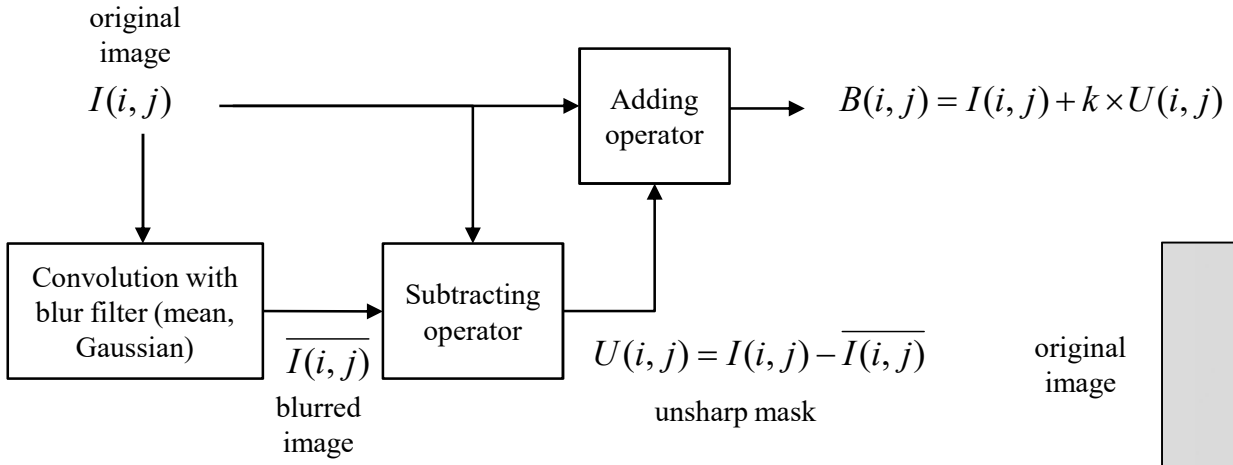
The size of w must depend of the σ value, 97% of the Gaussian sum is covered in the range ±3σ, then the mask should be designed such as $n \leq 6\sigma$, with n the size of the mask.

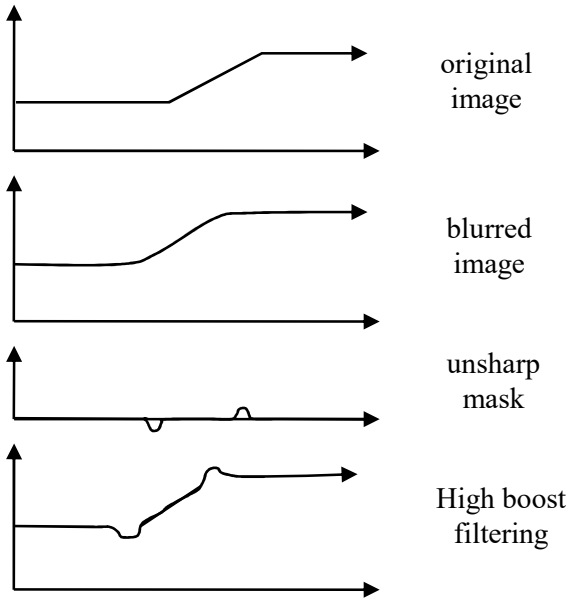


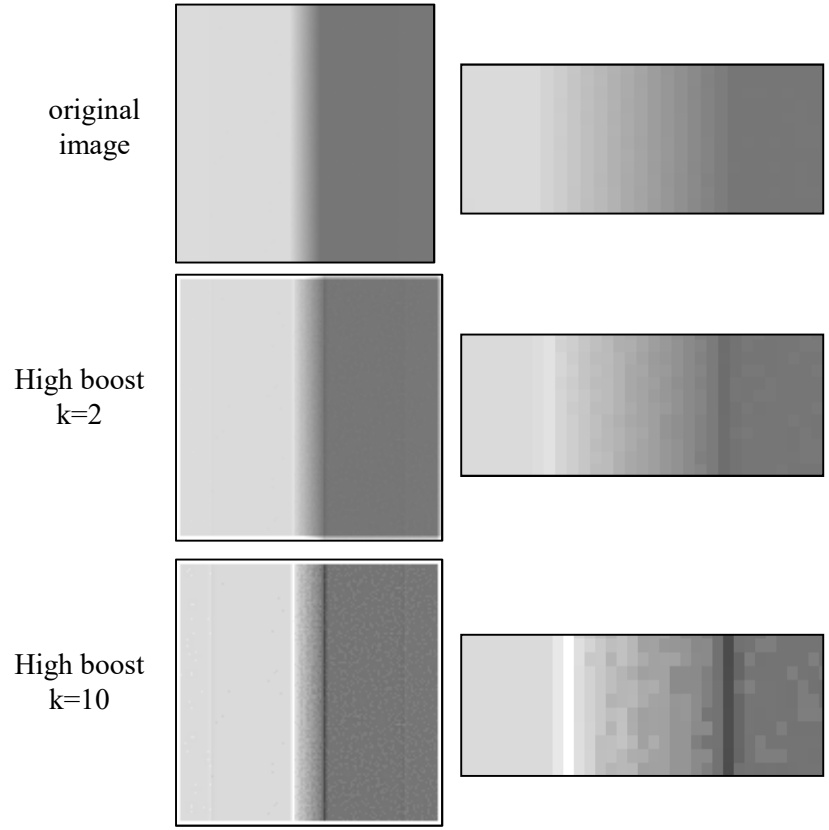
Mean filtering (4)
"High boost filtering"

K is the weight of the high boost filter

k	Filter
0	no filter
<1	weak unsharp masking
1	Unsharp masking
>1	High Boost filtering







Linear filtering

1. Fundamentals of linear filtering
2. Mean filtering
3. Derivative filters - 1st order
4. Derivative filters - 2^{sd} order
5. Combining operators

Derivative filters - 1st order "Introduction"

Filters	Mask	Type	Coefficient rules
Basic operator	-1 1	asymmetric	$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$
Roberts operator	1 ; ; -1	asymmetric	$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$
Improved operator	-1 0 1	asymmetric	$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$
Prewitt operator	-1 0 1 ; -1 0 1 ; -1 0 1	asymmetric	$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$
Sobel operator	-1 0 1 ; -2 0 2 ; -1 0 1	asymmetric	$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 1st order "Basic operators" (1)

		A		B		C				A				B				
Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	5	0	0	0	0	Na

Edge detection: many approaches to image interpretation are based on edges, since tit is insensitive to contrast modifications. The edge is at the position of the step change,

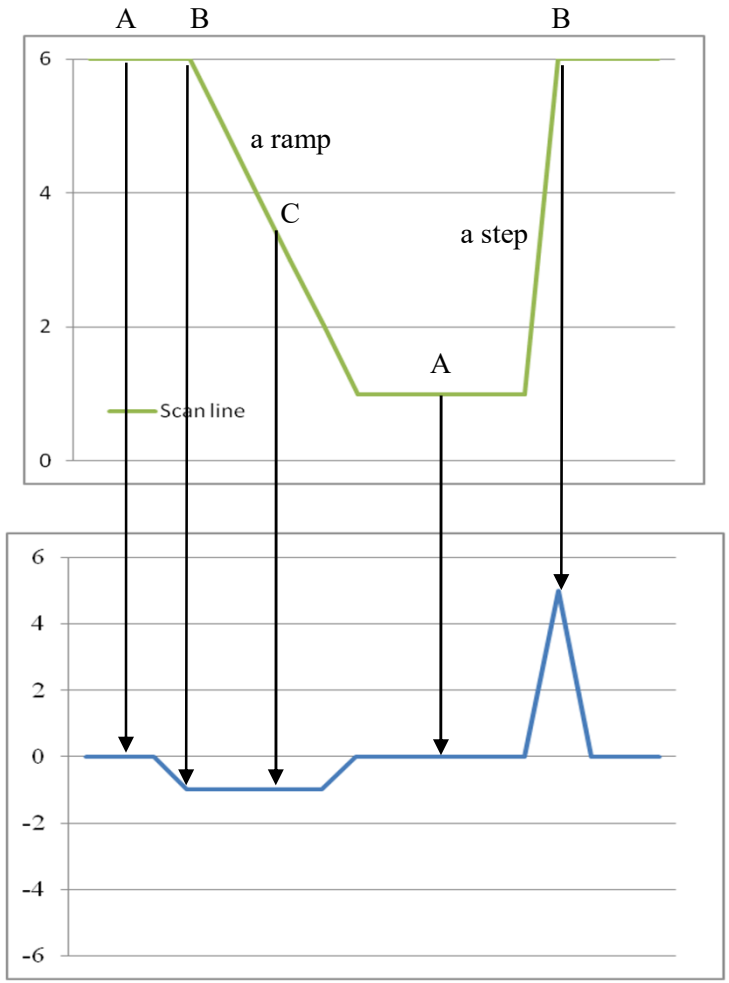
To detect edge we can use first-order differentiation. The basic definition of the first-order of a one dimensional function f(x) is the difference

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

When computing the first-derivative, we subtract the value of the function at that location from the next point. It is a look ahead operation

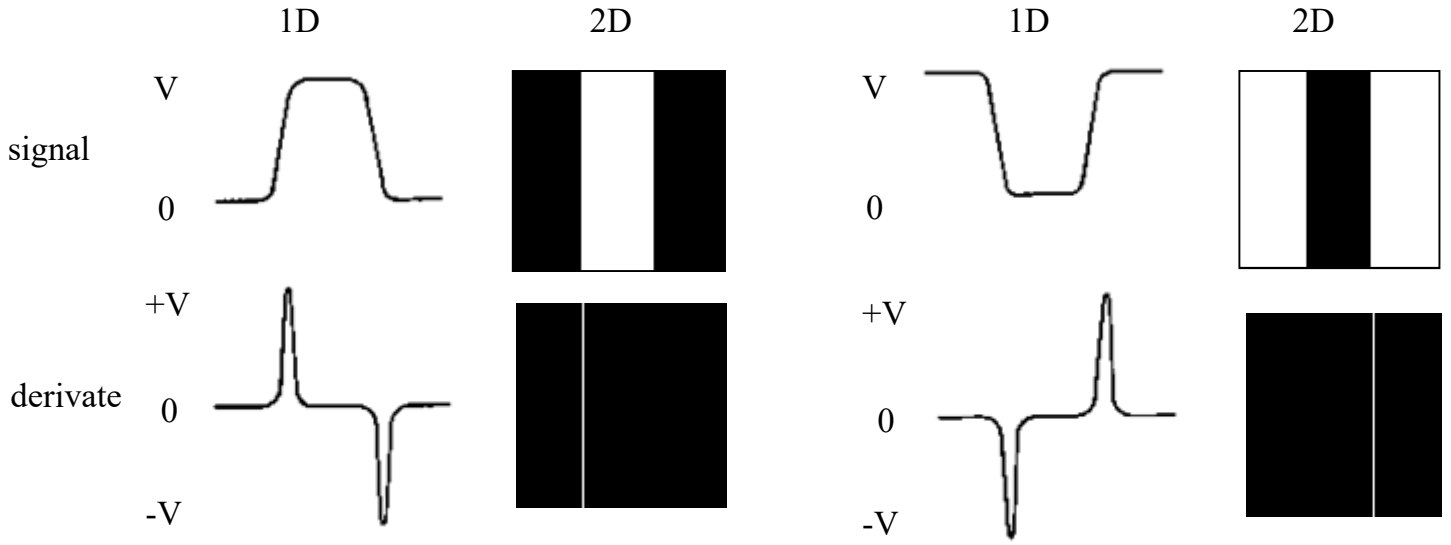
Properties of 1st derivative are

A. Must be zero on areas of constant intensity
B. Must be non zero at the onset of an intensity ramp or end of a step
C. Must be non zero along ramps



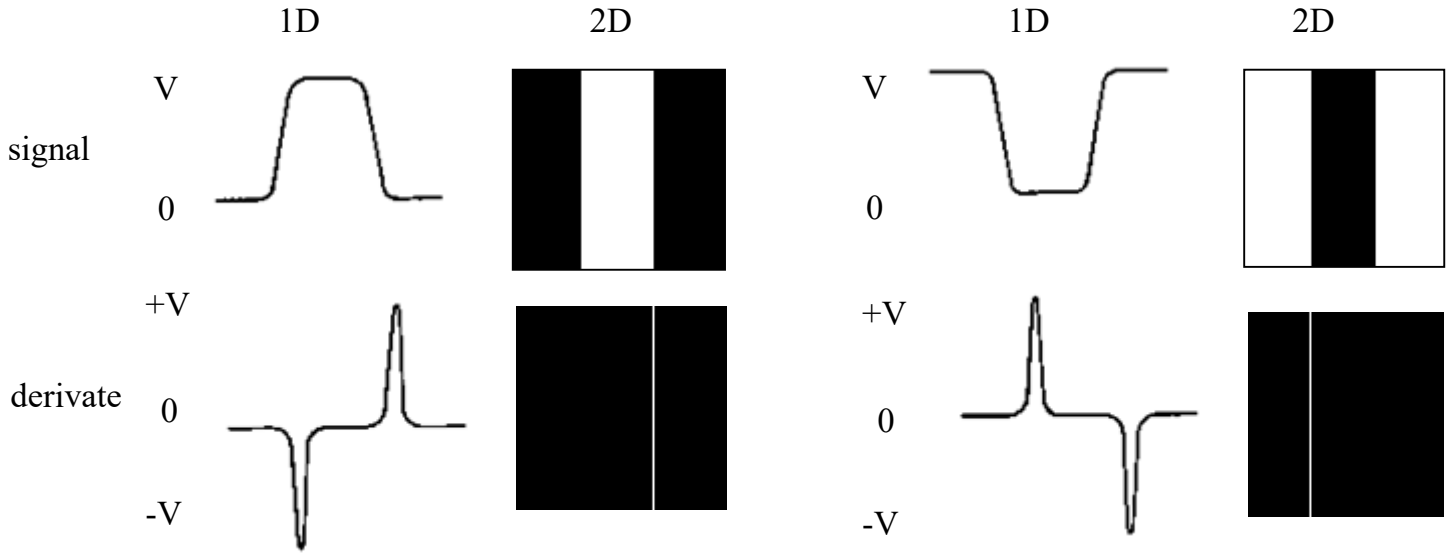
Derivative filters - 1st order
"Basic operators" (2)

Implementing a first-order differentiation $\frac{\partial f}{\partial x} = \frac{f(x+h_x) - f(x)}{h_x}$ with a convolution (i.e. correlation) operation, considering $h_x = 1$, is obtained with masks $[1 \quad -1]$ for convolution ($[-1 \quad 1]$ for correlation).



Derivative filters - 1st order
"Basic operators" (3)

In practice, negative version of a first-order differentiation $-\frac{\partial f}{\partial x} = \frac{f(x) - f(x + h_x)}{h_x}$ is also commonly used, considering $h_x = 1$, masks are $[-1 \quad 1]$ for convolution ($[1 \quad -1]$ for correlation).



Derivative filters - 1st order "Basic operators" (4)

Horizontal edge detector: differencing horizontality adjacent points will detect vertical changes and is called horizontal edge detector, it can be obtained with

$$\left|\frac{\partial f}{\partial x}\right| = \frac{f(x+h_x, y) - f(x, y)}{h_x} = f(x+1, y) - f(x, y) \quad \text{with} \quad h_x = 1$$

correlation mask is

-1	1

Vertical edge detector: differencing vertically adjacent points will detect horizontal changes and is called vertical edge detector, it can be obtained with

$$\left|\frac{\partial f}{\partial y}\right| = \frac{f(x, y+h_y) - f(x, y)}{hy} = f(x, y+1) - f(x, y) \quad \text{with} \quad h_y = 1$$

correlation mask is

-1
1

Derivative filters - 1st order "Basic operators" (5)

Horizontal/Vertical edge detector: combining the two gives an operator that can detect vertical and horizontal edge together, that is

$$\left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right| = \frac{f(x+h_x, y) - f(x,y)}{h_x} + \frac{f(x, y+h_y) - f(x,y)}{hy}$$
$$= f(x+1, y) - f(x,y) + f(x, y+1) - f(x,y) \qquad \text{with} h_x = h_y = 1$$
$$= f(x+1, y) + f(x, y+1) - 2f(x,y)$$

convolution mask is

-2	1
1	

Roberts cross operator (1965) was one of the earliest edge detection operators. It implement a version of basic first-order detection and used two masks that differentiate pixel values in a diagonal manner, as opposed to along the axes directions.
The two masks are called M^+ and M^-.

M^-

1	
	-1

M^+

	1
-1	

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 1st order
"Improved operators" (1)

Analysis of the basic operators: Taylor series reveals that differencing adjacent points provides an estimation of the first-order derivative at a point.

	Equation	Comments
Taylor series (general form)	$\sum_{n=0}^{\infty} \frac{f^n(x)}{n!}(x_0 - x)^n$ $f(x) + \frac{f'(x)}{1!}(x_0 - x) + \frac{f''(x)}{2!}(x_0 - x)^2 + \ldots$	We consider • *f(x)*, a real (or complex-valued) function that is infinitely differentiable • *x,* a real (or complex) number where *f(x)* can be defined in the neighborhood • x_0, a variable describing f The Taylor series of *f(x)* in *x* is
Taylor series of *f(x+h), f(x-h)*	$f(x+h) = f(x) + f'(x)h + f''(x)\frac{h^2}{2} + O(h^3)$ $f(x-h) = f(x) - f'(x)h + f''(x)\frac{h^2}{2} - O(h^3)$ *with* $h = x_0 - x$ *and* $O(h^3)$ *the error term at* $n = 3$	If the difference is taken between points separated by *h*, then the Taylor expansion for *f(x+h)* and *f(x-h)* are
Precision of first derivative with an error term *O(h)*	$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2}h - O(h^2) = \frac{f(x+h) - f(x)}{h} - O(h)$ $f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{f''(x)}{2}h + O(h^2) = \frac{f(x) - f(x-h)}{h} + O(h)$	By re- arrangement, the first-order derivative *f'(x)* can be defined as follow, with an error term *O(h)*
Precision of first derivative with an error term *O(h²)*	$f'(x) = \frac{f(x+h) - f(x-h)}{h} + O(h^2)$	By differencing *f(x+h)*, *f(x-h)*, we obtain the first-order derivative as follow, with an error term $O(h^2)$, With h<1, this error is clearly smaller than an error term *O(h)*

Derivative filters - 1st order "Improved operators" (2)

Horizontal edge detector: differencing horizontality adjacent points will detect vertical changes and is called horizontal edge detector, it can be obtained with

$$\left|\frac{\partial f}{\partial x}\right| = \frac{f(x+h_x,y)-f(x-h_x,y)}{h_x} = f(x+1,y)-f(x-1,y) \quad \text{with} \quad h_x = 1$$

convolution mask is

-1	0	1

Vertical edge detector: differencing vertically adjacent points will detect horizontal changes and is called vertical edge detector, it can be obtained with

$$\left|\frac{\partial f}{\partial y}\right| = \frac{f(x,y+h_y)-f(x,y-h_y)}{hy} = f(x,y+1)-f(x,y-1) \text{ with } \quad h_y = 1$$

convolution mask is

-1
0
1

Derivative filters - 1st order
"Improved operators" (3)

Prewitt operator: it is prudent to incorporate averaging within the edge detection process, we can then extend the horizontal and vertical edge detectors as follow. These give the Prewitt edge detection operator.

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

-1	0	1
-1	0	1
-1	0	1

$\frac{\partial f}{\partial y}$

-1	-1	-1
0	0	0
1	1	1

The digital approximation to the first-order derivatives using the Prewitt masks with correlation is then given by

0	**0**	**0**	**0**	**0**
0	z_1	z_2	z_3	**0**
0	z_4	z_5	z_6	**0**
0	z_7	z_8	z_9	**0**
0	**0**	**0**	**0**	**0**

$$\frac{\partial f}{\partial x} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

$$\frac{\partial f}{\partial y} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

Derivative filters - 1st order
"Improved operators" (4)

Soblel operator: a full discussion of edge detection problem involves consideration of the accuracy with which edge magnitude and orientation can be estimated. Prewitt operator was found to lead to an angular error varying from 0° to 7,38° [Pratt79]. To reduce the error (to 1,36°), the weight at the central pixel can be doubled. This gives the Sobel operator that was the most popular edge detector until the development of edge detection techniques.

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

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

$\frac{\partial f}{\partial y}$

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

The digital approximation to the first-order derivatives using the Sobel masks with correlation is then given by

0	**0**	**0**	**0**	**0**
0	z_1	z_2	z_3	**0**
0	z_4	z_5	z_6	**0**
0	z_7	z_8	z_9	**0**
0	**0**	**0**	**0**	**0**

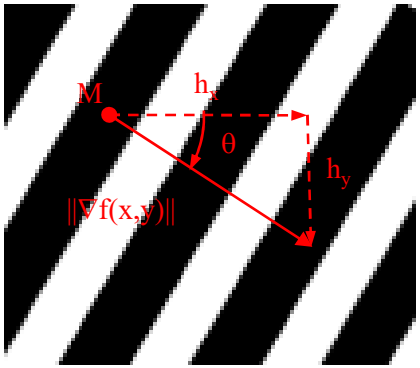
$$\frac{\partial f}{\partial x} = (z_3 + 2 \times z_6 + z_9) - (z_1 + 2 \times z_4 + z_7)$$

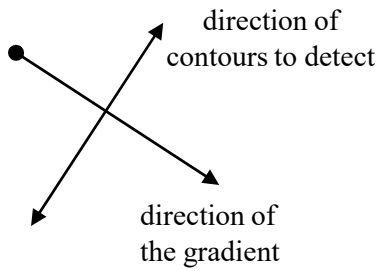
$$\frac{\partial f}{\partial y} = (z_7 + 2 \times z_8 + z_9) - (z_1 + 2 \times z_2 + z_3)$$

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 1st order
"Differential Gradient"

Differential Gradient (DG): the aim of edge detection is to find where the operator response is sufficiently large to be taken as a reliable indicator of the edge of an object. In the Differential Gradient (DG) approach, we are using gradients and their magnitudes to detect the maximum response. For a function f(x,y), the gradient, its magnitude and direction from partial derivative are:





$$\frac{\partial f}{\partial x} = \frac{f(x+h_x, y) - f(x,y)}{h_x}$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+h_y) - f(x,y)}{h_y}$$

partial derivative of f in x and y

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

gradient vector (row and column)

$$\|\nabla f(x,y)\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

or $$\|\nabla f(x,y)\| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

or $$\|\nabla f(x,y)\| = \max\left(\left|\frac{\partial f}{\partial x}\right|, \left|\frac{\partial f}{\partial y}\right|\right)$$

magnitude of the gradient

$$\theta = arctg\left(\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right)$$

direction of the gradient

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 1st order
"Canny edge detector" (1)

Canny edge detector is interested with optimal edge detection, involving

- ✓a good detection – the algorithm should mark as many real edges in the image as possible.
- ✓a good localization – edges marked should be as closed as possible to the edges in the real image.
- ✓a minimal response – a given edge in the image should only be marked once.

The optimal function in Canny's detector is described by the sum of four exponential terms, but it can be approximated by the first derivative of a Gaussian. Summarizing, the Canny edge detection algorithm consists of the following basic steps

1. **Noise reduction** smoothes the input image with a Gaussian filter.
2. **Finding intensity gradient** returns the values of 1st derivative filters, compute the gradient magnitude and angles.
3. **Non-maximum suppression** applies non-maxima suppression to the gradient magnitude image.
4. **Hysteresis thresholding** uses double thresholding and connectivity analysis to detect and link edges.

Derivative filters - 1st order
"Canny edge detector" (2)

1. **Noise reduction:** Smooth the input image with a Gaussian filter.
   e.g. mask 3×3 with a $\sigma^2 = 1$

Derivative filters - 1st order
"Canny edge detector" (3)

2. **Finding intensity gradient:** return the values of 1st derivative filters, compute the gradient magnitude and angles. e.g. we compute $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$ with Sobel based operators, and then the magnitude and angle image

$$\frac{\partial f}{\partial x} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

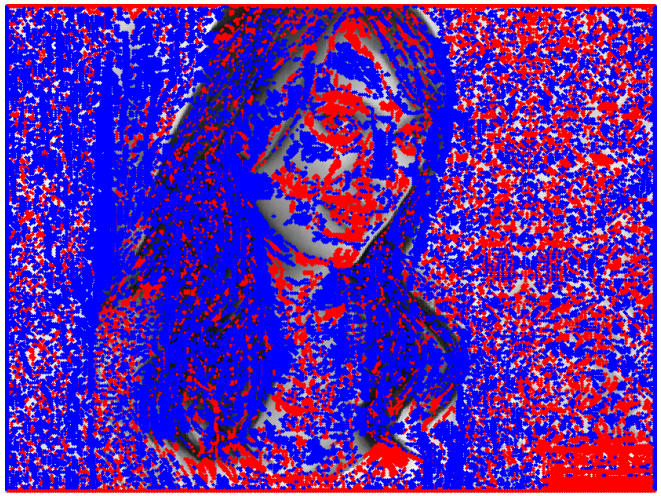
$$\frac{\partial f}{\partial y} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Derivative filters - 1st order
"Canny edge detector" (4)

2. **Finding intensity gradient:** return the values of 1st derivative filters, compute the gradient magnitude and angles. e.g. we compute $\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$ with Sobel based operators, and then the magnitude and angle image

$$M(x,y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\alpha(x,y) = arctg\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$



$[+\pi/2,- +\pi/2] \pm \pi/8$

$0 \pm \pi/8$

Derivative filters - 1st order
"Canny edge detector" (5)

3. **Non-maximum suppression:** Apply non-maxima suppression to the gradient magnitude image.
   a. to discretize α(x,y) in four main directions: horizontal, vertical and the two diagonals
   a. considering the direction $d_k = \alpha(x,y)$, if the value $M(x,y)$ is less than at least one of its two neighbors along d_k let $gN(x,y) = 0$, otherwise let $gN(x,y) = M(x,y)$

$$M(x,y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

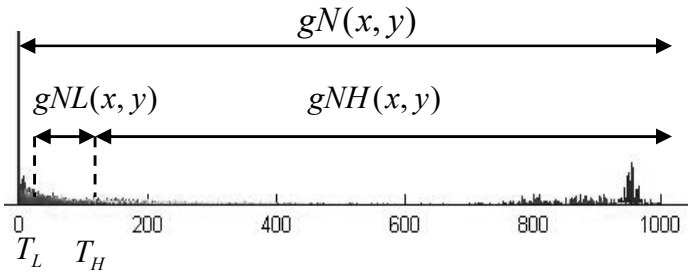
$gN(x,y)$

$M(x,y) - gN(x,y)$

Derivative filters - 1st order
"Canny edge detector" (6)

4. **Hysteresis thresholding** uses double thresholding and connectivity analysis to detect and link edges.

a. to compute the *gNH* and *gNL* images, corresponding to low and high thresholds T_L,T_H, of *gN*
e.g. with T_L=5 and T_H=70



$$gNH(x,y) = gN(x,y) \geq T_H$$

$$gNL(x,y) = gN(x,y) \geq T_L$$

$$gNL(x,y) = gN(x,y) - gNH(x,y)$$

or $gNL(x,y) = gN(x,y) \in [T_L, T_H]$

Derivative filters - 1st order
"Canny edge detector" (7)

4. **Hysteresis thresholding** uses double thresholding and connectivity analysis to detect and link edges.

b. for each pixel *p* in *gNH*, to mark as valid edges in *gNH* all the weak pixels *N8(p)* in *gNL* (i.e. 8 connected to p) let say this new image *gNHC(x,y)*

$gNH(x,y)$ $gNHC(x,y)$ $gNHC(x,y) - gNH(x,y)$

Derivative filters - 1st order
“Canny edge detector” (8)

Linear filtering

1. Fundamentals of linear filtering
2. Mean filtering
3. Derivative filters - 1st order
4. Derivative filters - 2^{sd} order
5. Combining operators

Derivative filters - 2sd order "Introduction" (1)

		A		B		C			B		A				B	B			
Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
2sd derivative	Na	0	0	-1	0	0	0	0	1	0	0	0	0	5	-5	0	0	0	Na

The basic definition of the second-order of a one dimensional function f(x) is the difference

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

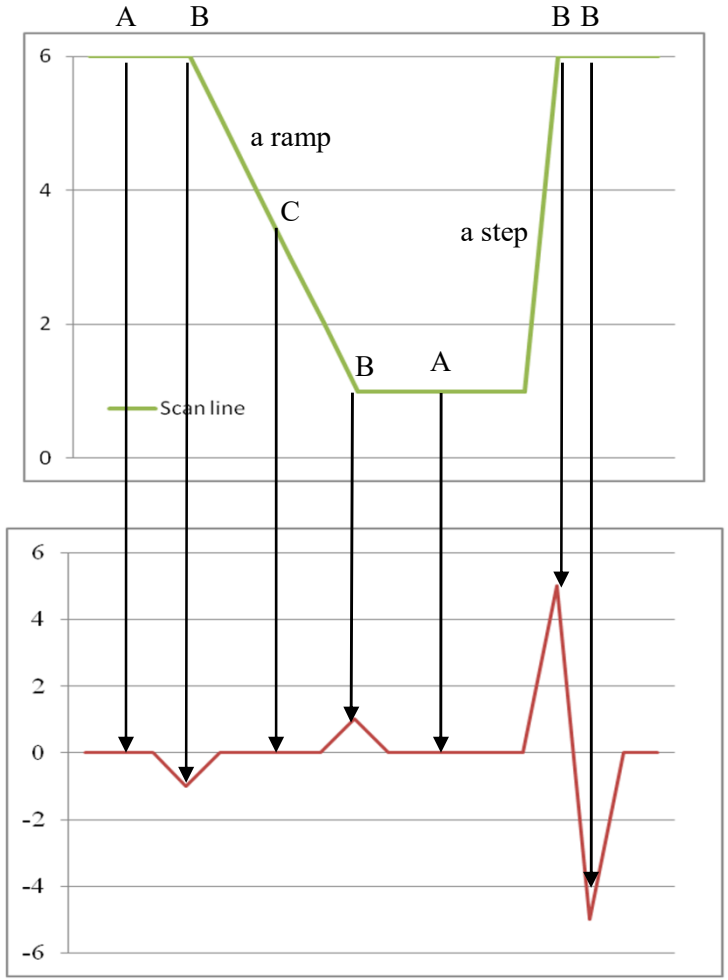
$$\frac{\partial^2 f}{\partial^2 x} = \frac{f'(x+h_x) - f'(x)}{h_x}$$

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

This expansion is about point x+1, our interest is on the second derivative about point x, so we subtract h_x

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

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



Derivative filters - 2sd order "Introduction" (2)

		A		B		C			B		A				B	B			
Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
2sd derivative	Na	0	0	-1	0	0	0	0	1	0	0	0	0	5	-5	0	0	0	Na

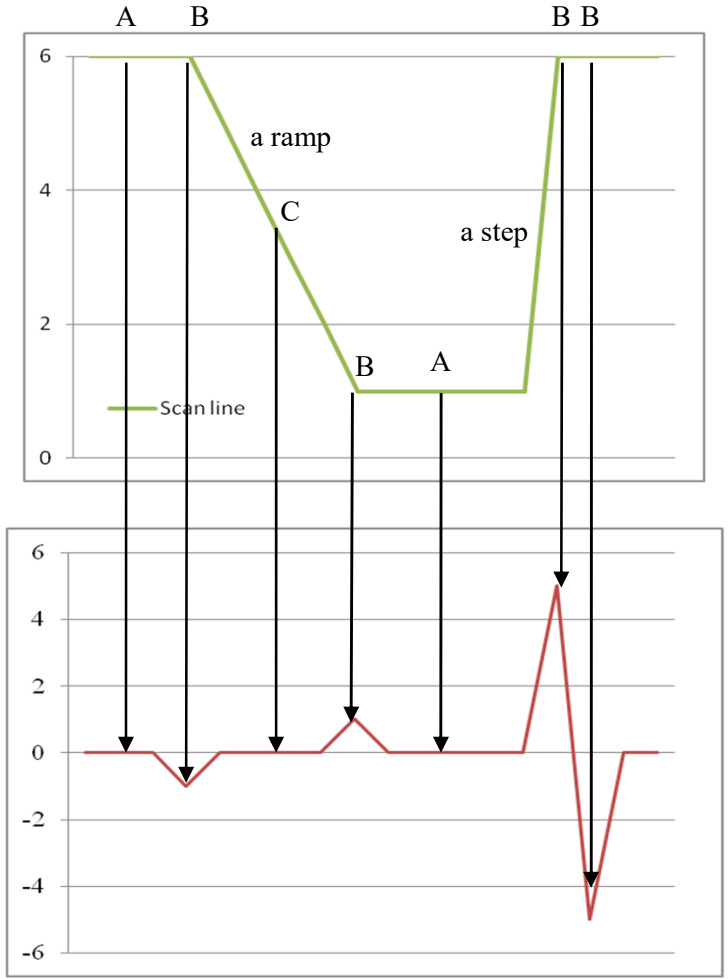
The basic definition of the second-order of a one dimensional function f(x) is the difference

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

When computing the second-derivative, we use the previous and the next point sin the computation. We avoid the outside range by limiting the computation from the second to the penultimate points.

Properties of 2sd derivative are

A. Must be zero on areas of constant intensity
B. Must be non zero at the onset and end of an intensity ramp or step
C. Must be zero along ramps of constant slope



Derivative filters - 2^{sd} order "Introduction" (3)

Filters	Mask			Type	Coefficient rules
Laplacian filter	0 a 0 a b a 0 a 0	a a a a b a a a a	c a c a b a c a c	Symmetric	$b \times a < 0$ $b \times c > 0$ $\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 0$
Unsharp filter	0 a 0 a b a 0 a 0	a a a a b a a a a	c a c a b a c a c	Symmetric	$b > 0$ $a < 0$ $c < 0$ $\sum_{u=1}^{n} \sum_{v=1}^{n} w_{u,v} = 1$

Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 2^{sd} order
"Laplacian detector" (1)

Laplacian operator: in image processing, second-order derivative are implemented using Laplacian.
For a function f(x,y) the Laplacian is defined as :

$$\frac{\partial f}{\partial x} = \frac{f(x+h_x, y) - f(x,y)}{h_x}$$

$$\frac{\partial f}{\partial y} = \frac{f(x_x, y+h_y) - f(x,y)}{h_y}$$

} partial derivative of f in x and y

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

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

} The partial second derivative of f in x and y

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

} The Laplacian

Derivative filters - 2^{sd} order "Laplacian detector" (2)

Laplacian operator: in image processing, second-order derivative are implemented using Laplacian.
How to convert Laplacian to structuring element ?

$h_x = 1$, $h_y = 0$

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

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

0	0	0
1	-2	1
0	0	0

$h_x = 0$, $h_y = 1$

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

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

0	1	0
0	-2	0
0	1	0

$h_x = 1$, $h_y = 1$

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

$$\nabla^2 f(x,y) = \frac{f(x+h_x,y) + f(x-h_x,y) - 2f(x,y)}{h_x^{\,2}} + \frac{f(x,y+h_y) + f(x,y-h_y) - 2f(x,y)}{h_y^{\,2}}$$

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

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

0	1	0
1	-4	1
0	1	0

These are basic Laplacian operators, other operators (diagonal Laplacian, line filter, etc.) can been obtained using similar computations.

Derivative filters - 2sd order
"Laplacian detector" (3)

f is a convolution operator
and the structuring elements **w** is

0	a	0
a	b	a
0	a	0

a	a	a
a	b	a
a	a	a

c	a	c
a	b	a
c	a	c

$$\sum_{u=1}^{n}\sum_{v=1}^{n} w_{u,v} = 0$$

$b \times a < 0$ $\quad$ $b \times c > 0$

	1	
1	-4	1
	1	

1	1	1
1	-8	1
1	1	1

1	4	1
4	-20	4
1	4	1

and more ….

1	1	1				
1	-8	1				
1	1	1				

0	**0**	**0**	**0**	**0**	**0**	**0**
0	5	9	9	9	5	**0**
0	5	9	17	17	5	**0**
0	5	9	17	17	5	**0**
0	5	9	9	9	5	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

I

→

0	**0**	**0**	**0**	**0**	**0**	**0**
0	-17	-27	-11	-19	-9	**0**
0	-3	4	-40	-52	13	**0**
0	-3	4	-40	-52	13	**0**
0	-17	-27	-11	-19	-9	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

I_L

Laplacian filtering

↓

0	**0**	**0**	**0**	**0**	**0**	**0**
0	35	25	41	33	43	**0**
0	49	56	12	0	65	**0**
0	49	56	12	0	65	**0**
0	35	25	41	33	43	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

$I_m = I_L - \min(I_L)$

←

0	**0**	**0**	**0**	**0**	**0**	**0**
0	113	81	132	107	139	**0**
0	158	181	39	0	210	**0**
0	158	181	39	0	210	**0**
0	113	81	132	107	139	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**

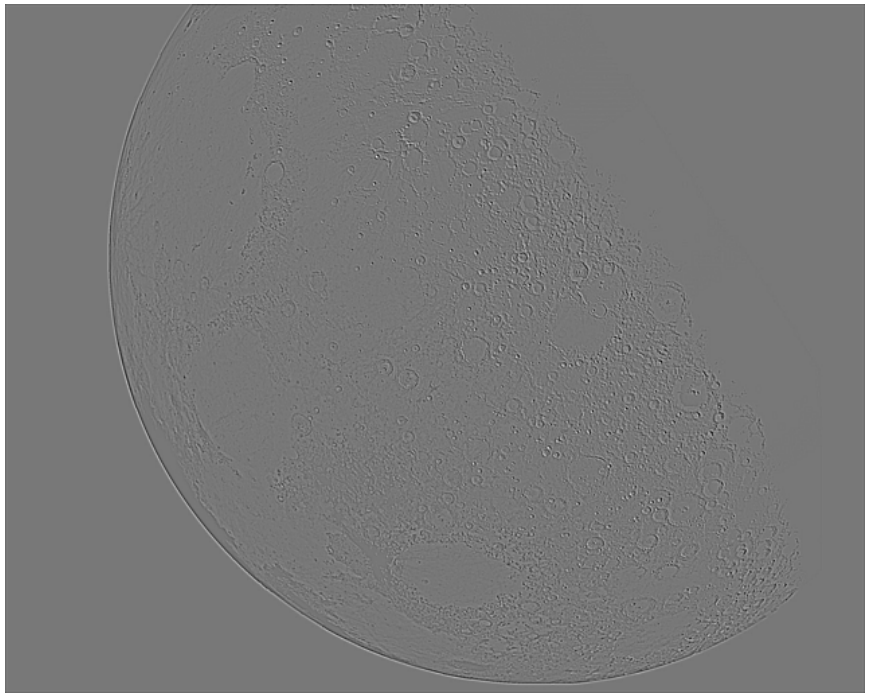
$I_s = K\left(I_m \Big/ \max(I_m) \right)$

Scaling

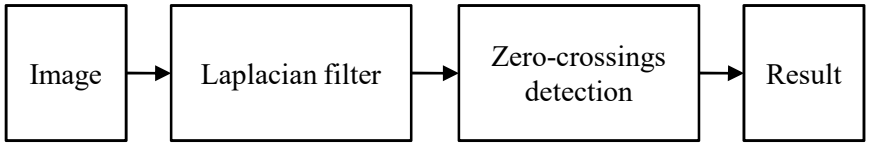
Derivative filters - 2^{sd} order "Laplacian detector" (4)

Laplacian

scaled Laplacian

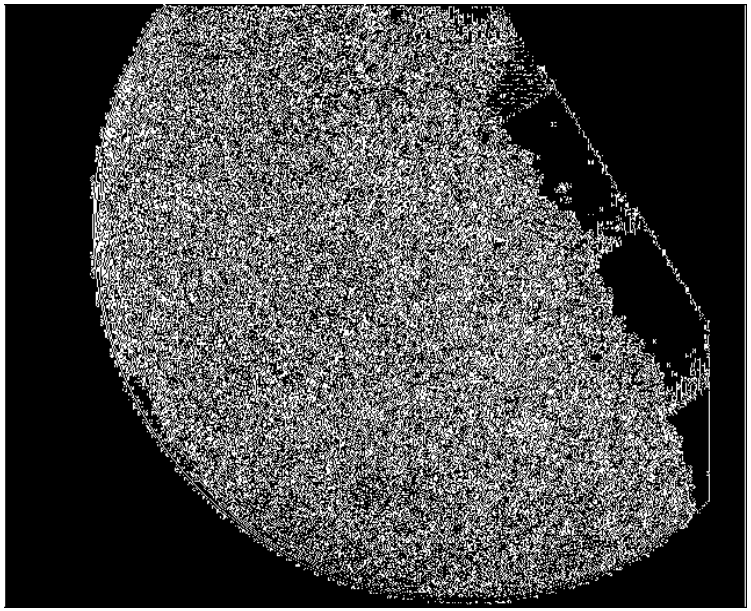


Derivative filters - 2^{sd} order "Laplacian detector" (5)

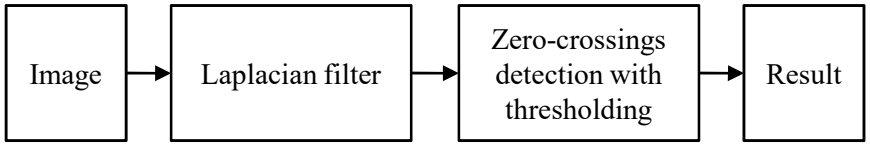


$$I_r(i,j) = \begin{cases} white & if \quad I(i,j) \times I(i-1,j) < 0 \\ black & otherwise \end{cases}$$

Zeros-crossing without thresholding

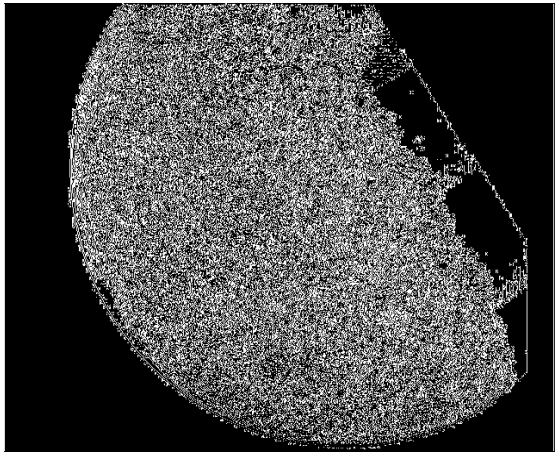


Derivative filters - 2^{sd} order "Laplacian detector" (6)

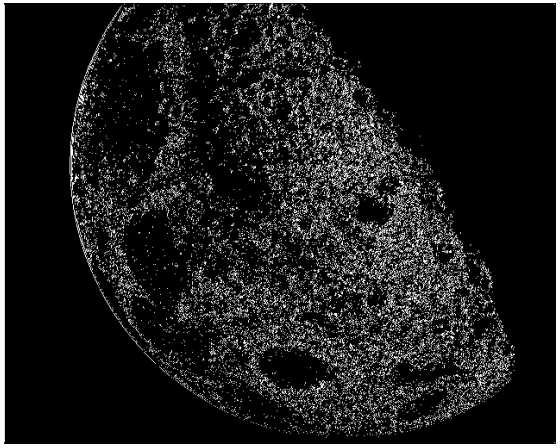


$$I_r(i,j) = \begin{cases} white & if \quad I(i,j) \times I(i-1,j) < 0 \quad and \quad \left\| I(i,j) + I(i-1,j) \right\| > th \\ black & otherwise \end{cases}$$

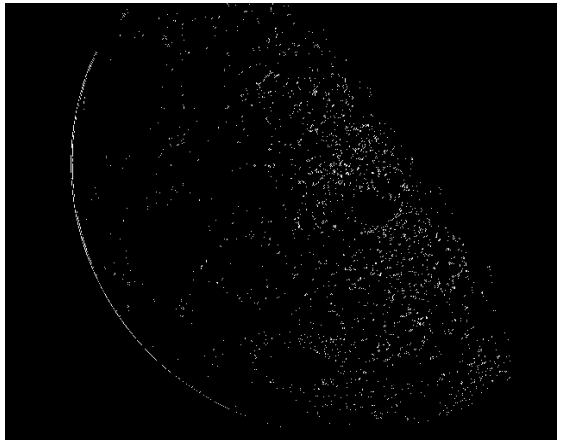
Zeros-crossing without thresholding



Zeros-crossing with thresholding 2^6

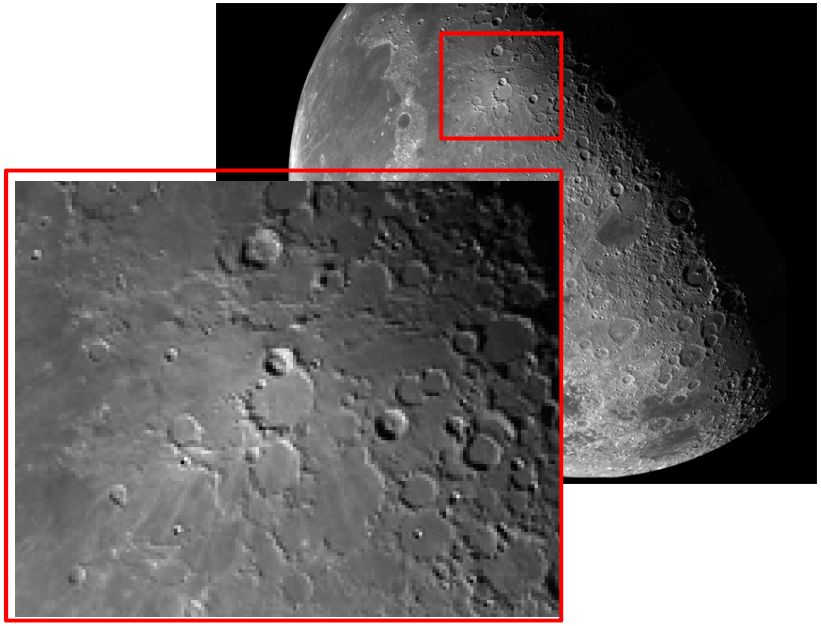


Zeros-crossing with thresholding 2^8

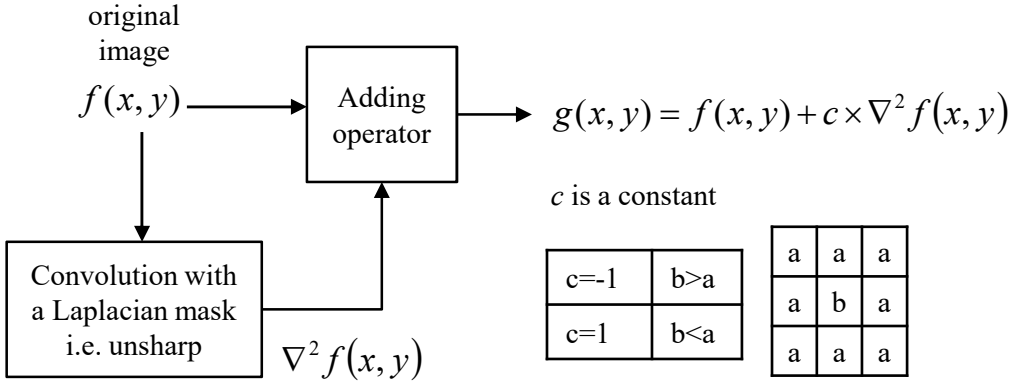


Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 2^{sd} order "Sharpening filter"



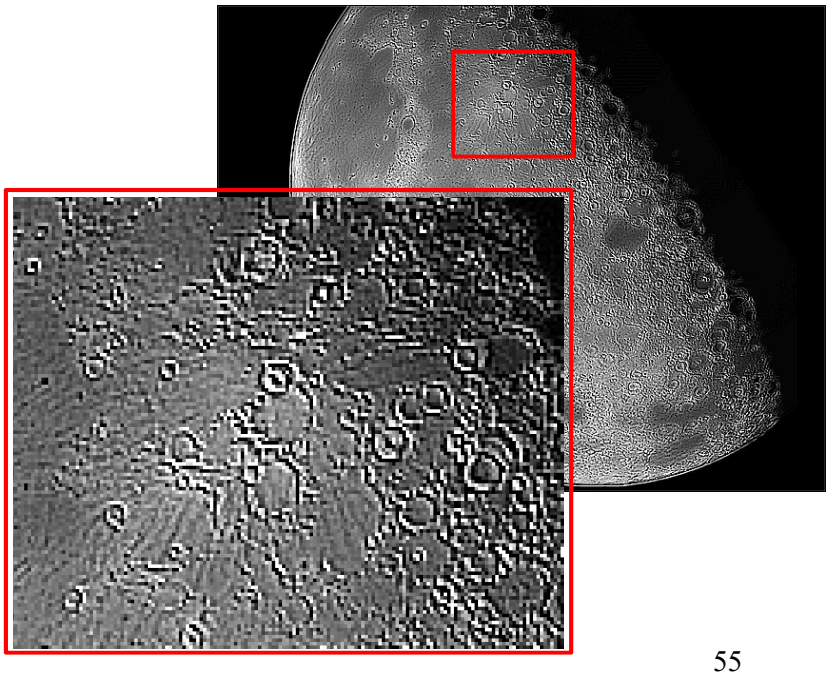
Two steps sharpening filter



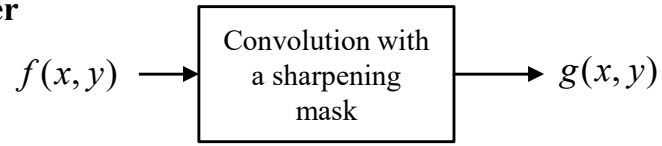
c is a constant

c=-1	b>a
c=1	b<a

a	a	a
a	b	a
a	a	a



One step sharpening filter



$$g(x,y) = f(x,y) + c \times \nabla^2 f(x,y)$$

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

$$g(x,y) = f(x,y) - \left(f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) - 4f(x,y)\right)$$

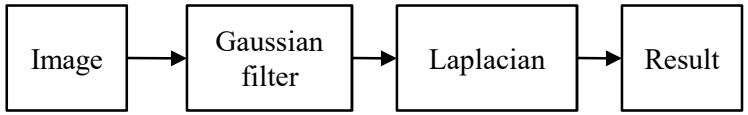
$$g(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

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

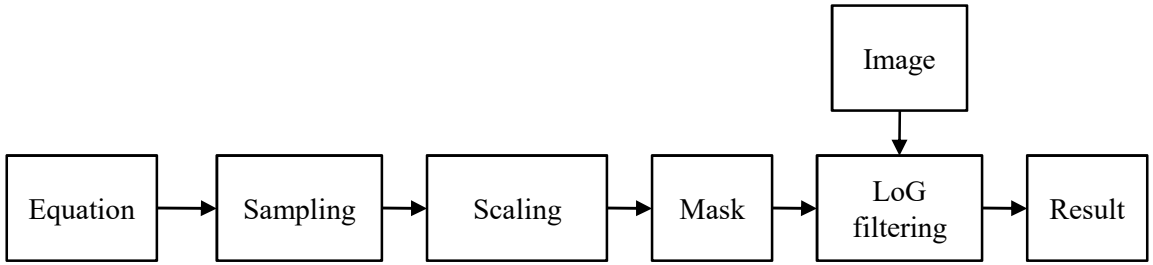
Methods	**Type**	**Application**
Mean filter	Mean filtering	Enhancement
Parametric low-pass filter		
Gaussian filter		
High boost filter		Sharpening
Basic operators	Derivative filters 1st order	Edge detection
Improved operators		
Differential Gradient		
Canny-edge operator		
Laplacian detector	Derivative filters 2sd order	Key-point detection
Sharpening filter		Sharpening
Laplacian of Gaussian (LoG)		Key-point detection

Derivative filters - 2sd order
"Laplacian of Gaussian (LoG)" (1)

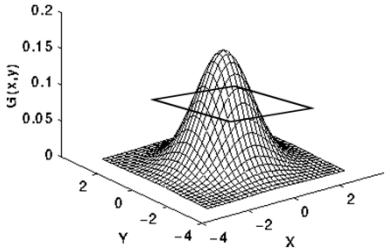
Two steps LoG



One step LoG



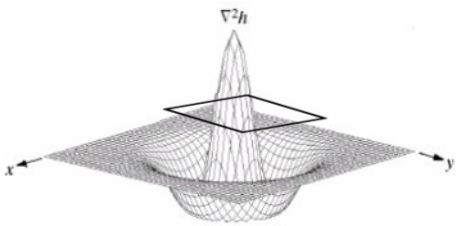
$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$\frac{1}{209}\times\begin{array}{|c|c|c|}\hline 16 & 26 & 16\\ \hline 26 & 41 & 26\\ \hline 16 & 26 & 16\\ \hline\end{array}$$

$$\sum_{u=1}^{n}\sum_{v=1}^{n}w_{u,v}=1$$

$$LoG=\nabla^2 G'(x,y)=\frac{1}{\sigma^2}\left[\frac{x^2+y^2}{\sigma^2}-2\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & -1 & 0 & 0\\ \hline 0 & -1 & -2 & -1 & 0\\ \hline -1 & -2 & 16 & -2 & -1\\ \hline 0 & -1 & -2 & -1 & 0\\ \hline 0 & 0 & -1 & 0 & 0\\ \hline\end{array}$$

$$\sum_{u=1}^{n}\sum_{v=1}^{n}w_{u,v}=0$$

LoG is also called
"Mexican hat"

Derivative filters - 2sd order "Laplacian of Gaussian (LoG)" (2)

One step LoG

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

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

$$LoG = \nabla^2 G'(x,y) = \frac{\partial^2 G(x,y)}{\partial^2 x} + \frac{\partial^2 G(x,y)}{\partial^2 y}$$

$$LoG = \nabla^2 G'(x,y) = \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

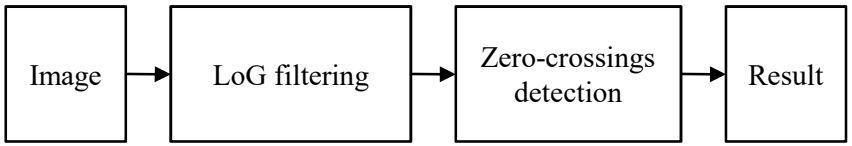
$$LoG = \nabla^2 G'(x,y) = \left[\frac{-1}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{-x}{\sigma^2}\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \left[\frac{-1}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} + \frac{-y}{\sigma^2}\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

$$LoG = \nabla^2 G'(x,y) = \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

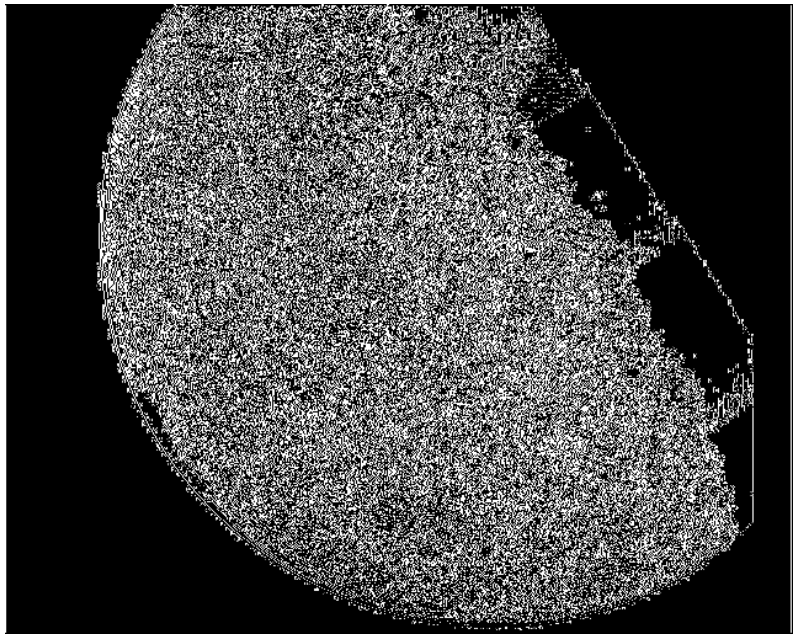
$$LoG = \nabla^2 G'(x,y) = \frac{1}{\sigma^2}\left[\frac{x^2+y^2}{\sigma^2} - 2\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Derivative filters - 2^{sd} order "Laplacian of Gaussian (LoG)" (3)

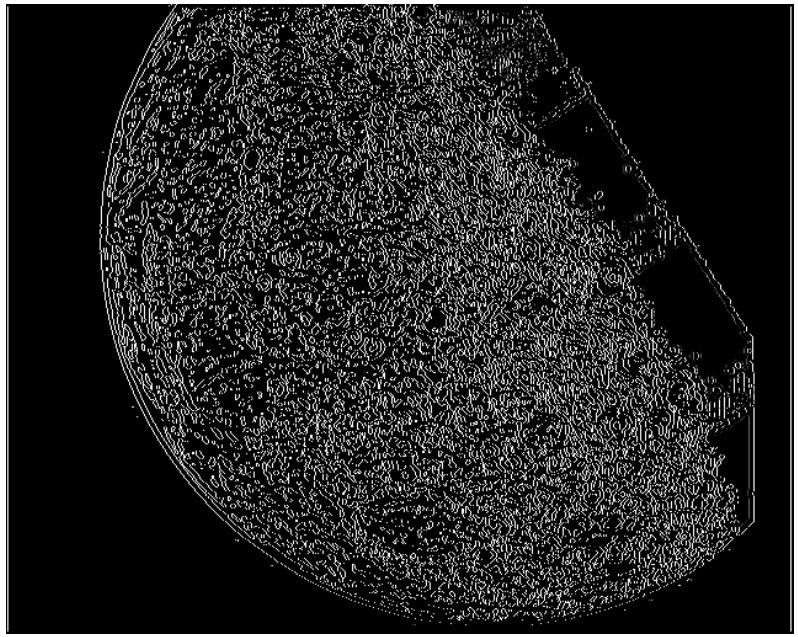
The Marr-Hildreth detector



Zero-crossings detection with Laplacian



Zero-crossings detection with LoG



Derivative filters - 2sd order
"Laplacian of Gaussian (LoG)" (4)

It is possible to approximate the LoG filter by a difference of Gaussians (DoG) in the following equation, with $\sigma_1 > \sigma_2$

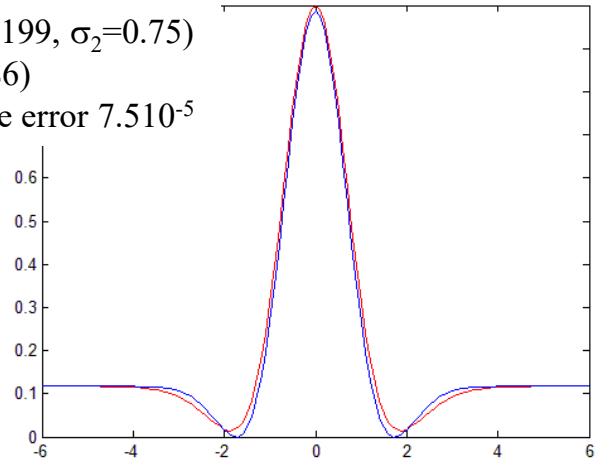
$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

To make meaningful comparison between the LoG and DoG, the value of for the LoG must be selected in the following equation so that the LoG and DoG have the same zero crossings.

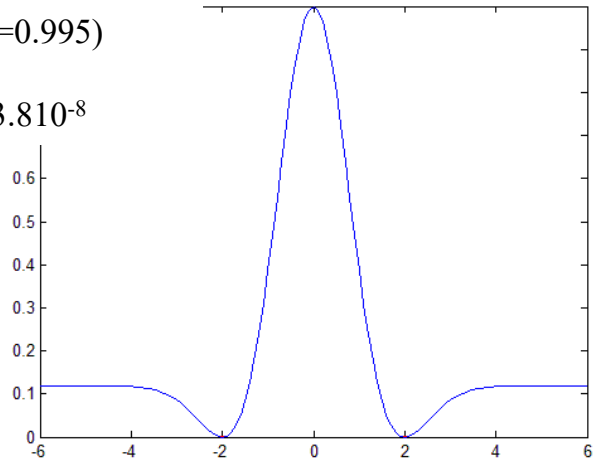
$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left(\frac{\sigma_1^2}{\sigma_2^2}\right) \qquad DoG(x,y) \approx LoG(x,y)$$

Although the zeros crossings of the LoG and Dog will be the same, their amplitude scales will be different.
We can make them compatible by scaling both functions.

DoG (σ_1=1.199, σ_2=0.75)
LoG (σ=0.86)
Mean square error 7.510^{-5}



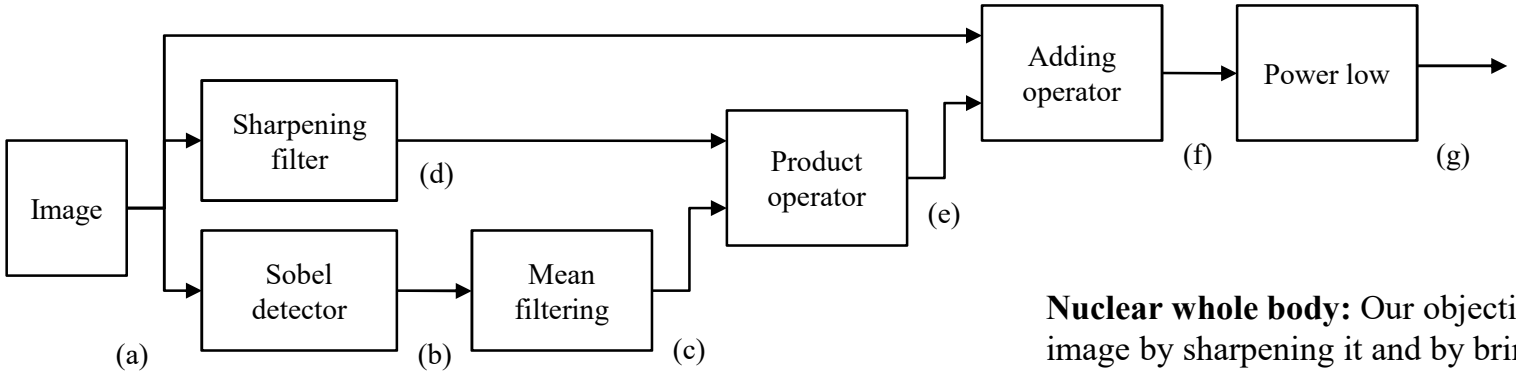
DoG (σ_1=1.005, σ_2=0.995)
LoG (σ=1.0)
Mean square error 3.810^{-8}



Linear filtering

1. Fundamentals of linear filtering
2. Mean filtering
3. Derivative filters - 1st order
4. Derivative filters - 2sd order
5. Combining operators

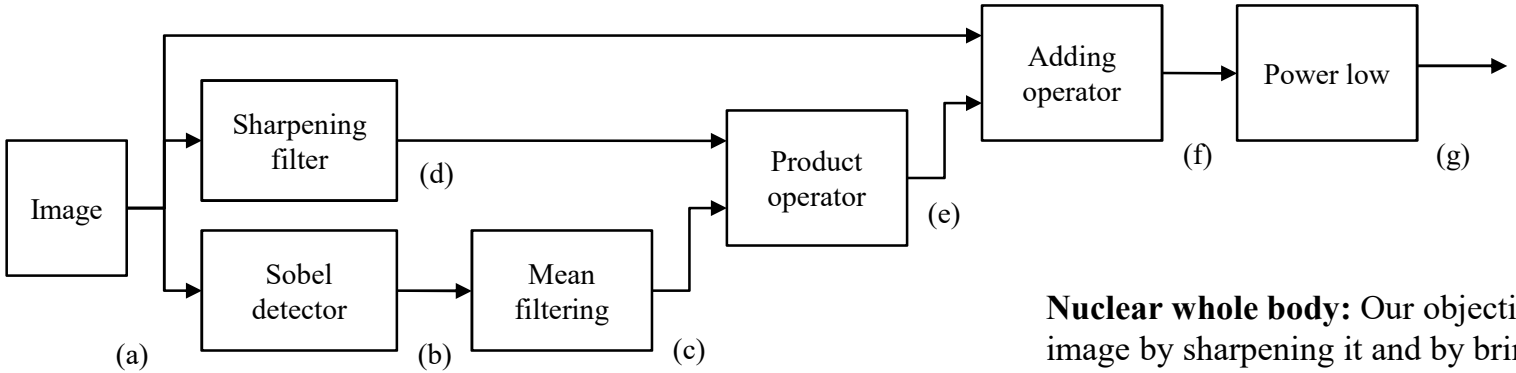
Combining operators (1)



Nuclear whole body: Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal details

(a)

Combining operators (2)

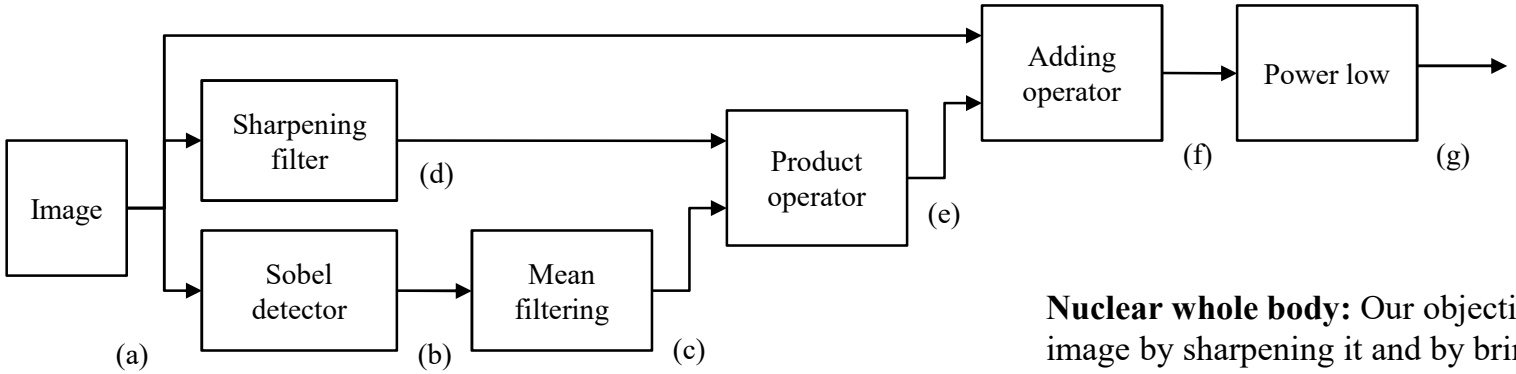


Nuclear whole body: Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal details

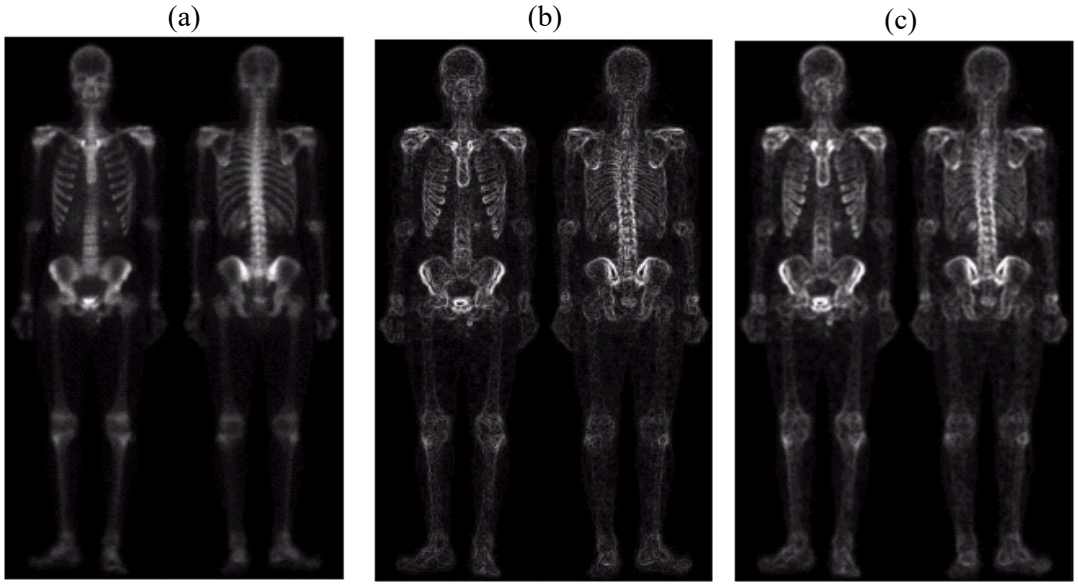
(a)

(d)

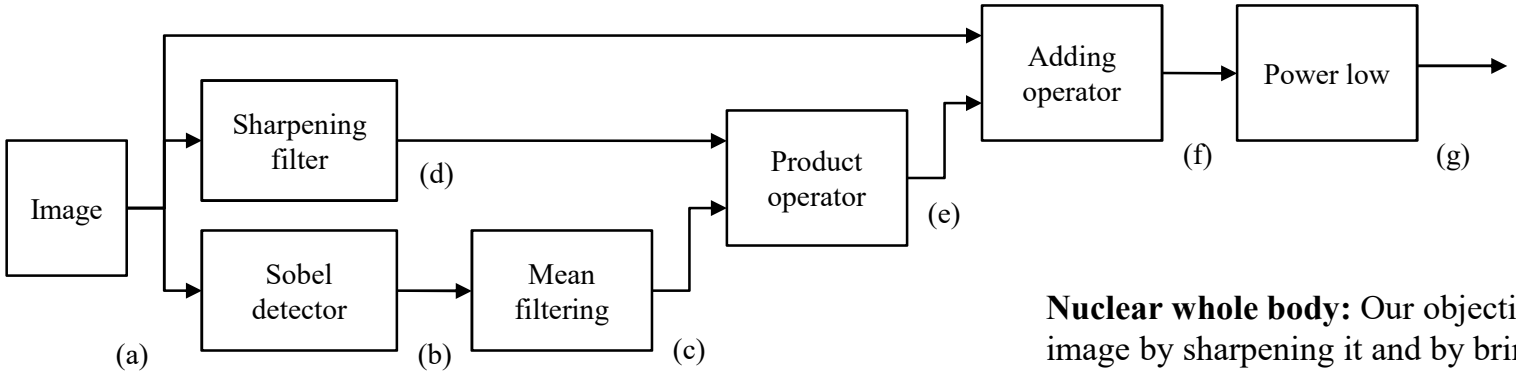
Combining operators (3)



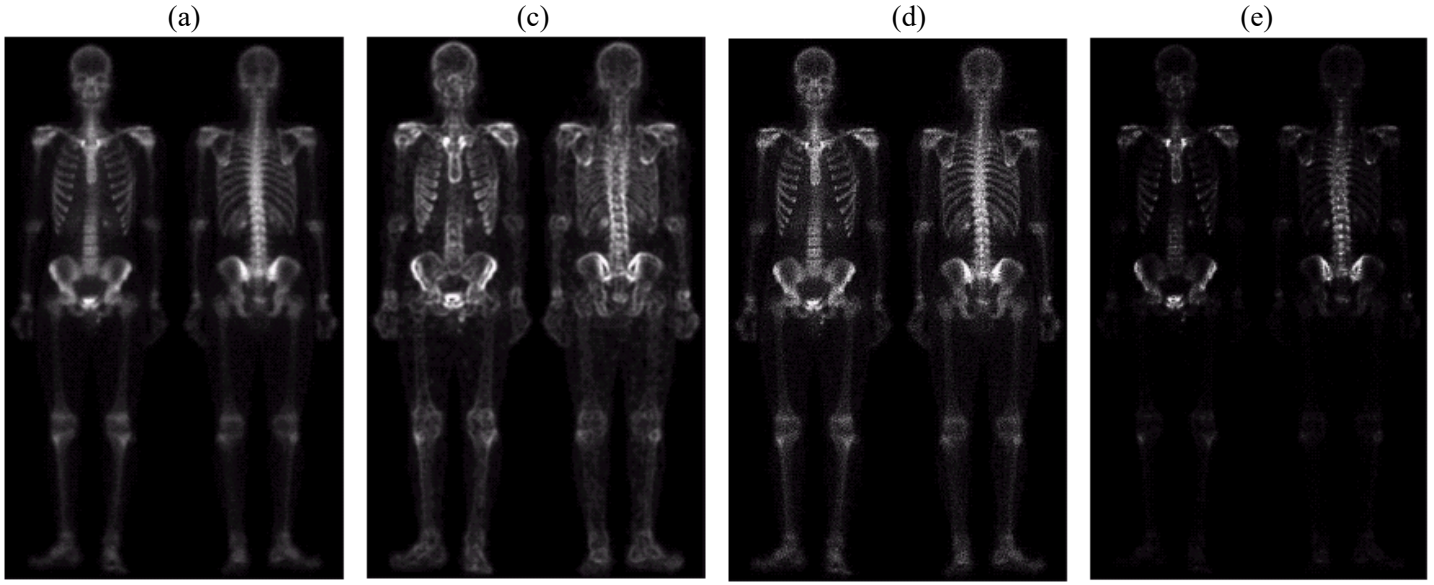
Nuclear whole body: Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal details



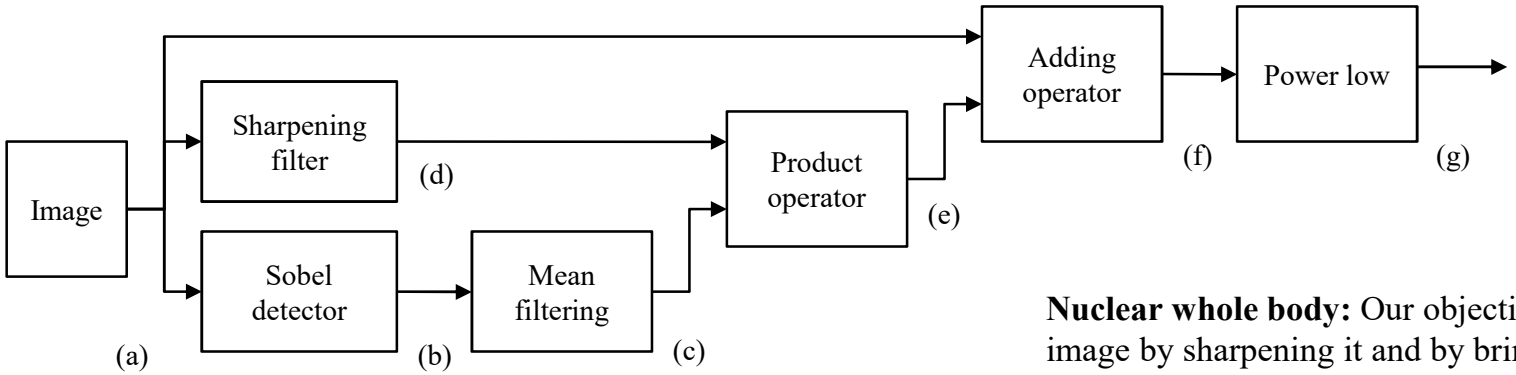
Combining operators (4)



Nuclear whole body: Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal details



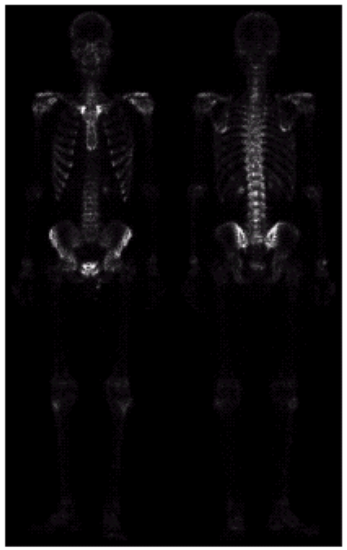
Combining operators (5)



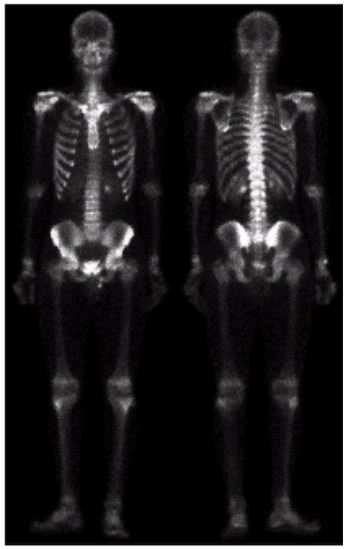
Nuclear whole body: Our objective is to enhance this image by sharpening it and by bringing out more of the skeletal details

(a)

(e)



(f)



(g)