Spatial Domain Filtering & Image Enhancement

EE551

Week 3 – Spatial Domain Filtering & Image Enhancement

Ref. Chapter 3 Gonzalez & Woods

2023

In this section…

- Filtering
  - Basic concepts
  - Low pass filters
  - High pass filters
- Edge enhancement
  - LaPlacian of Gaussian (LoG)
  - Unsharp masking

Filtering

- One of the most important concept in imaging.
- In spatial-domain filtering, each pixel is replaced by a value which is determined by the surrounding values in the *neighbourhood.*
  - (Can also do frequency-domain filtering – later)
- The neighbourhood is defined by the user
- The pixel value is usually a weighted sum of the neighbourhood pixels
- The weighting factors are specified in the *kernel*
- Linear filters: output pixel is a linear combination of neighbourhood pixels

Enhancement

- Very important application of filtering
- Objective is to "improve" the image somehow e.g. remove noise or add some effect (e.g. enhance edges).
- The definition of "improve" is very application-dependent and subjective
- It could be for human consumption
- It could be for computer vision
- They're not the same!

Enhancement using spatial filtering

- Spatial filtering is one of the most important and widely used method of image enhancement
- The operations are fundamentally the same as filtering operations discussed in signals and systems, DSP modules
  - i.e. it is a convolution operation in the spatial domain
- Each output filtered output pixel is the product of an input pixel, pixels in a “neighbourhood”, and the filter kernel weights
- Spatial filtering is used for noise reduction, edge detection, edge enhancement, tone mapping etc

Linear (Convolution) filtering

- Define the filter kernel
- Slide the kernel over the image so that its center pixel coincides with each (target) pixel in the image.
- Multiply the pixels lying beneath the kernel by the corresponding values (weights) in the kernel above them and sum the total.
- Copy the resulting value to the same locations in a new (filtered) image.

Linear filter example – 3*3 averaging filter

Input (edge with noise)

153	159	158	100	109	107
150	154	153	101	100	105
156	150	157	101	107	110
158	154	159	105	108	104
162	162	157	101	110	106
153	155	157	101	104	103
154	157	154	100	108	101
160	158	162	105	107	107
153	152	156	100	102	100
162	155	162	108	105	107
157	155	151	109	106	109
159	155	151	103	107	108

Filter

1	1	1
1	1	1
1	1	1

Filtered output

154	137	121	104
155	137	121	105
157	138	123	106
157	139	122	105
157	138	121	104
157	139	122	104
156	138	122	103
158	140	123	105
156	139	122	105
156	139	122	107

Neighbourhood

Filter kernel

Output pixel

Target pixel

$$Output = \frac{153 \times 1 + 159 \times 1 + 158 \times 1 + 150 \times 1 + 154 \times 1 + 153 \times 1 + 156 \times 1 + 150 \times 1 + 157 \times 1}{9} = 154$$

NUI Galway
OÉ Gaillimh

Linear filter example – 3*3 averaging filter

Input (edge with noise)

153	159	158	100	109	107
150	154	153	101	100	105
156	150	157	101	107	110
158	154	159	105	108	104
162	162	157	101	110	106
153	155	157	101	104	103
154	157	154	100	108	101
160	158	162	105	107	107
153	152	156	100	102	100
162	155	162	108	105	107
157	155	151	109	106	109
159	155	151	103	107	108

Filter

1	1	1
1	1	1
1	1	1

Filtered output

154	137	121	104
155	137	121	105
157	138	123	106
157	139	122	105
157	138	121	104
157	139	122	104
156	138	122	103
158	140	123	105
156	139	122	105
156	139	122	107

Target pixel

Neighbourhood

Filter kernel

Output pixel

$$Output = \frac{159 \times 1 + 158 \times 1 + 100 \times 1 + 154 \times 1 + 153 \times 1 + 101 \times 1 + 150 \times 1 + 157 \times 1 + 101 \times 1}{9} = 137$$

This process is repeated for all pixels in the image

Linear filter example – edge detection

Input (edge with noise)

153	159	158	100	109	107
150	154	153	101	100	105
156	150	157	101	107	110
158	154	159	105	108	104
162	162	157	101	110	106
153	155	157	101	104	103
154	157	154	100	108	101
160	158	162	105	107	107
153	152	156	100	102	100
162	155	162	108	105	107
157	155	151	109	106	109
159	155	151	103	107	108

Filter

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

Output

-4	144	-177	-40
-41	179	-182	22
-29	185	-160	20
41	162	-193	48
-16	169	-183	2
3	137	-198	36
16	214	-149	33
-52	146	-207	-23
-8	210	-127	-1
-12	110	-121	-8

Target pixel

Neighbourhood

Filter kernel

Output pixel

$$Output = 154 \times -1 + 153 \times -1 + 101 \times -1 + 150 \times -1 + 157 \times 8 + 101 \times -1 + 154 \times -1 + 159 \times -1 + 105 \times -1 = 179$$

Note: the sum of the coefficients is already 0, so there is no need to divide by the sum of the weights

Linear filter example – edge detection

Input (edge with noise)

153	159	158	100	109	107
150	154	153	101	100	105
156	150	157	101	107	110
158	154	159	105	108	104
162	162	157	101	110	106
153	155	157	101	104	103
154	157	154	100	108	101
160	158	162	105	107	107
153	152	156	100	102	100
162	155	162	108	105	107
157	155	151	109	106	109
159	155	151	103	107	108

Filter

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

Output

-4	144	-177	-40
-41	179	-182	22
-29	185	-160	20
41	162	-193	48
-16	169	-183	2
3	137	-198	36
16	214	-149	33
-52	146	-207	-23
-8	210	-127	-1
-12	110	-121	-8

Target pixel

Neighbourhood

Filter kernel

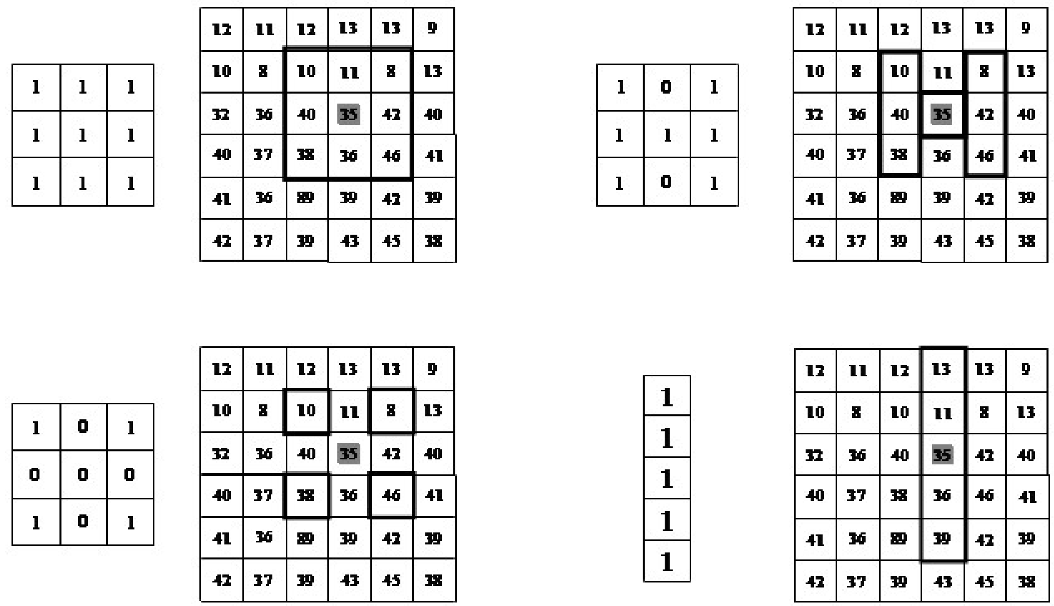
Output pixel

$$Output = 153 \times -1 + 101 \times -1 + 100 \times -1 + 157 \times -1 + 1101 \times 8 + 107 \times -1 + 159 \times -1 + 105 \times -1 + 108 \times -1 = -182$$

Note: the sum of the coefficients is already 0, so there is no need to divide by the sum of the weights

Other filter kernel designs

- Many filter kernel designs are possible
- Filter design involves choosing the right combination of kernel size, shape and weights, to perform a given task



Linear Filters

$$f_i = \sum_{i=1}^{9} w_i I_i(i)$$
$$= (-1x10) + (-1x11) + (-1x8) + (-1x40) + (8x35)$$
$$+ (-1x42) + (-1x38) + (-1x36) + (-1x46) = 14$$

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Image

12	11	12	13	13	9
10	8	10	11	8	13
32	36	40	35	42	40
40	37	38	36	46	41
41	36	89	39	42	39
42	37	39	43	45	38

Filter kernel slides over the underlying image, addressing each pixel

At the edges of the image….

- A few options …
- Simply leave unchanged any target pixels which are located where they would cause the filter to breach the image boundary.
- Consider those pixels in the kernel which remain within the boundary of the image and apply these only to the image pixels lying underneath.
- “Fill in” the missing pixels at the edges (usually by copying) so that the kernel has a full neighbourhood to work on
- Can also “crop” the output image

Spatial filtering - summary

- Spatial filtering is a very powerful and widely used method for image processing
  - Noise removal, edge detection & enhancement, CNNs etc
- The steps are as follows:
  - Define a filter kernel, based on task requirements
    - i.e. define size of kernel, weights
  - Convolve the filter kernel with the input pixel image data, to generate a filtered output
    - i.e. "slide" the filter kernel over each pixel in the image
    - If the sum of the weights is not equal to zero, then divide the output of the filter by the sum of the weights
      - If this is not done, the mean of the image will change – this manifests as a change in brightness
  - Define a strategy for the boundary of the image
    - i.e. how to handle pixels at the edge of the image

Simple Exercise

- Apply the 3x3 filter on the left at each of the image locations indicated -
  - a) The shaded pixel of value 37.
  - b) The shaded pixel of value 35.
  - c) The shaded pixel of value 13.

- Give the result for all the conventions concerning target pixels at the edges

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}$$

12	11	12	13	13	9
10	8	10	11	8	13
32	36	40	35	42	40
40	37	38	36	46	41
41	36	89	39	42	39
42	37	39	43	45	38

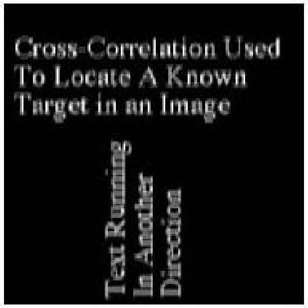
Mean or average filter

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{20}\begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{10}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$$

- Mean/averaging filter smooths image
- A form of (scaled) integration
- In signal processing terms, the mean filter is a low pass filter – removes “high frequencies” (fine detail)
- Size of kernel is analogous to the “bandwidth” (but in an “inverse” way – recall “time-frequency duality” from 1D signal processing ….
- Larger kernel does more averaging – removes more detail => “lower bandwidth filter”

Mean or average filter

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{20}\begin{array}{|c|c|c|c|}\hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline \end{array} \qquad \frac{1}{10}\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline 1 & 1 \\ \hline \end{array}$$



Mean or average filter

- Good for noise removal …
- … but not without side effects
- See code example

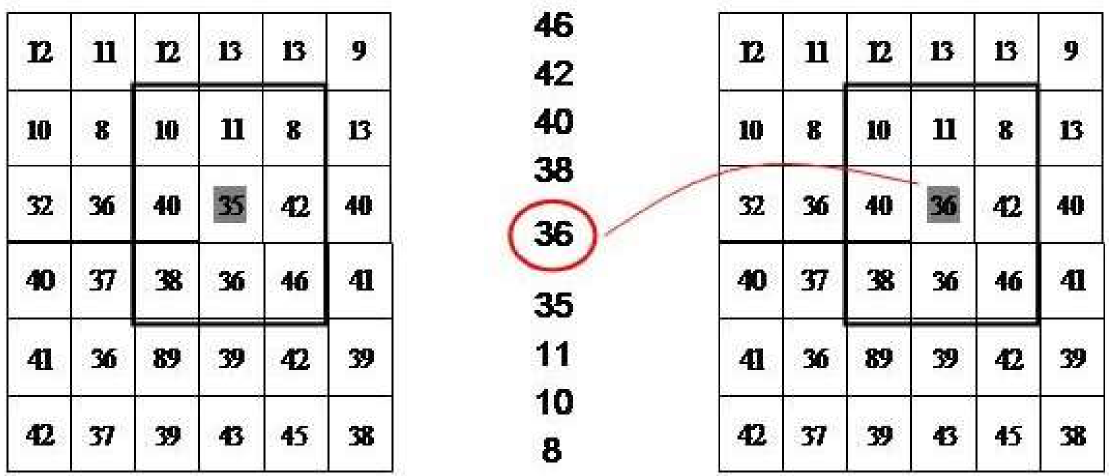
Motion blur Filter

FILTER KERNEL

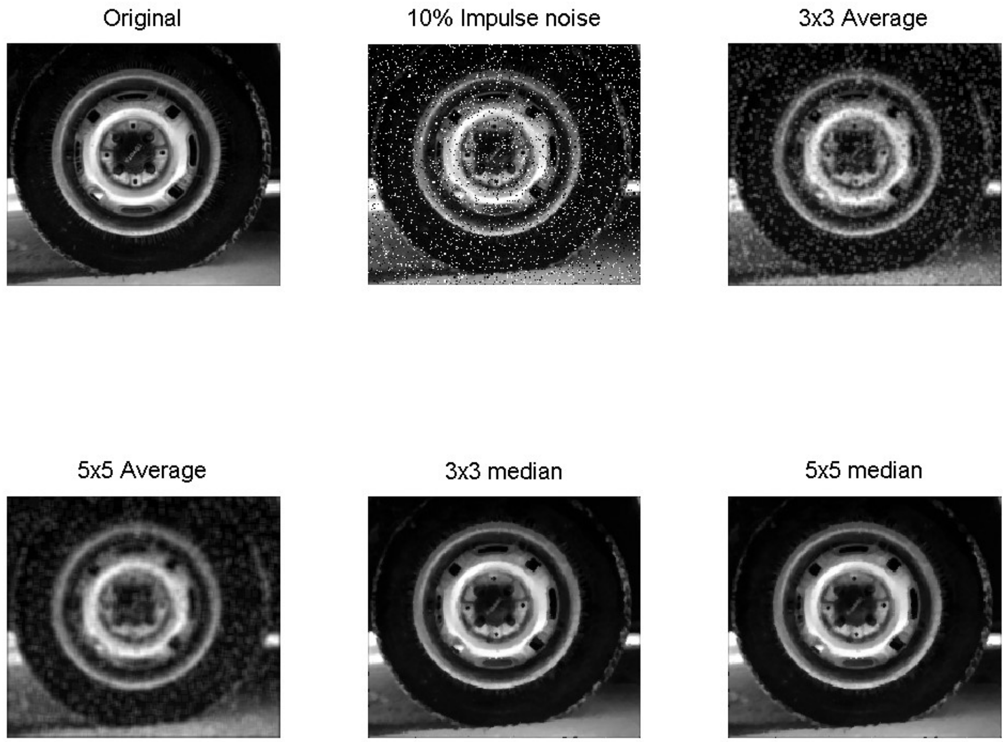
1
1
1
1 0
1
1
1
1
0 1
1
1

Median filter

- Form of non-linear filtering
- The median *m* of a set of numbers is that number for which half the numbers are less than *m*, and half greater than *m*
- Take median ranked value in neighbourhood. Eliminates outliers



Median versus mean filter



Order Filters

A general non-linear ranking filter:

- Define the neighbourhood of the target pixel. (Assume this comprises *N* pixels)
- Rank them in ascending order (1st is lowest value, *N*th is highest value).
- Choose the order of the filter (from 1 to *N*)
- Set the filtered value to be equal to the value of the chosen rank pixel.
- Median filter is just a special case of order filter

Order filters

Original

Min

Max

Max-Min

Gaussian filters

- The Gaussian filter is a very important one both for theoretical and practical reasons.

$$f(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp-\left(\frac{x^2+y^2}{2\sigma^2}\right)$$

- Discrete approximations to this filter are specified by 2 parameters:
  - i) The desired size of the kernel.
  - ii) The value of the standard deviation.

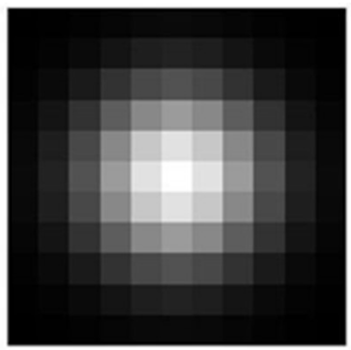
Gaussian filters

8	12	8
12	20	12
8	12	8

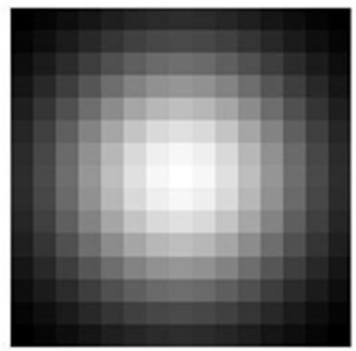
D	D	D	D	D	D	D	D	D	D	D
D	D	D	D	D	1	D	D	D	D	D
D	D	D	1	1	1	1	1	D	D	D
D	D	1	1	2	2	2	1	1	D	D
D	D	1	2	3	4	3	2	1	D	D
D	1	1	2	4	4	4	2	1	1	D
D	D	1	2	3	4	3	2	1	D	D
D	D	1	1	2	2	2	1	1	D	D
D	D	D	1	1	1	1	1	D	D	D
D	D	D	D	D	1	D	D	D	D	D
D	D	D	D	D	D	D	D	D	D	D

1	1	1	1	2	2	2	2	2	2	2	1	1	1	1
1	1	2	2	3	3	4	4	4	3	3	2	2	1	1
1	2	2	3	4	5	5	5	5	5	4	3	2	2	1
1	2	3	4	5	6	7	7	7	6	5	4	3	2	1
2	3	4	5	6	8	8	8	8	8	6	5	4	3	2
2	3	5	6	8	9	10	10	10	9	8	6	5	3	2
2	4	5	7	8	10	11	11	11	10	8	7	5	4	2
2	4	5	7	8	10	11	11	11	10	8	7	5	4	2
2	4	5	7	8	10	11	11	11	10	8	7	5	4	2
2	3	5	6	8	9	10	10	10	9	8	6	5	3	2
2	3	4	5	6	8	8	8	8	8	6	5	4	3	2
1	2	3	4	5	6	7	7	7	6	5	4	3	2	1
1	2	2	3	4	5	5	5	5	5	4	3	2	2	1
1	1	2	2	3	3	4	4	4	3	3	2	2	1	1
1	1	1	1	2	2	2	2	2	2	2	1	1	1	1

3x3, $\sigma=1$



11x11 $\sigma=2$

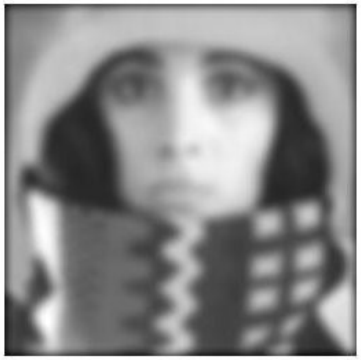


15x15 $\sigma=4$

Gaussian filtering

3x3, $\sigma=1$

11x11 $\sigma=2$



15x15 $\sigma=4$

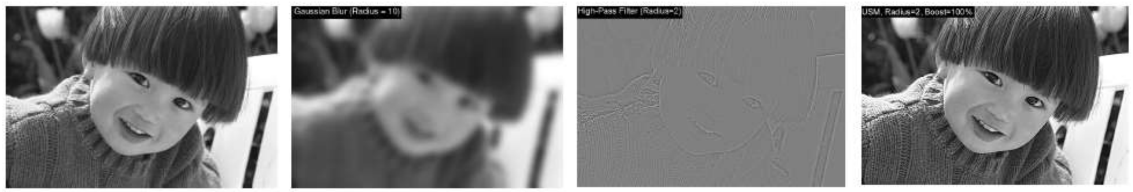
The importance of Gaussian Filters in image processing

- Gaussian filters play a fundamental role in image processing algorithms:
  - Horizontal and vertical separable
  - Isotropic – circularly symmetric – if the horizontal and vertical variance values are the same (the only separable filter satisfying this condition)
  - Excellent approximation of Gaussian filters can be achieved at a fixed cost per pixel independent of the filter size by repeated box filtering with a small number of iterations, which can be efficiently implemented using the concept of the integral image

The importance of Gaussian Filters in image processing

- Parametric (by changing the variance) Gaussian filters can efficiently perform:
  - Low-pass filtering (Gaussian filters are naturally weighted averaging filters)
  - High-pass filtering (by subtracting a low-pass image from an original)
  - Sharpening (by adding a scaled high-pass image to the original)
  - Diverse set of contrast enhancement or noise reduction algorithms
  - Dynamic range compression (with bilateral filtering)
  - Frequency-selective (notch filtering) in the Fourier Domain

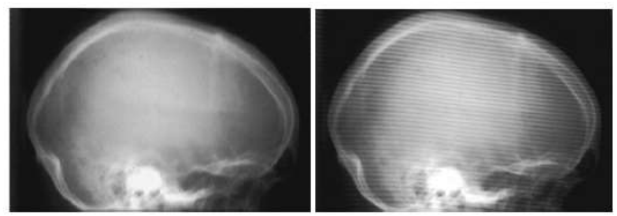
Gaussian filter applications

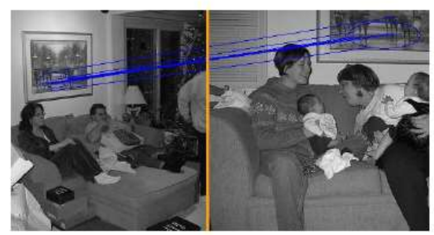


Original | Low-Pass | High-Pass | Sharpened

Original

Enhanced





Discontinuity Filters

- Detection *of discontinuities in an image* is central to the whole field of image processing - image segmentation (i.e. separating an image into *objects* and *background*).
- The simplest and most common form of discontinuity corresponds to a sudden change in the image intensity.
- Differentiation operations are important here

Discrete derivatives

Derivative	Continuous	Discrete
$\frac{\partial f}{\partial x}$	$\lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x}$	$f(x+1,y) - f(x,y)$
$\frac{\partial f}{\partial y}$	$\lim_{\Delta y \to 0} \frac{f(x, y+\Delta y) - f(x,y)}{\Delta y}$	$f(x,y+1) - f(x,y)$
$\nabla f(x,y)$	$\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$	$[f(x+1) - f(x,y), f(x,y+1)$ $- f(x,y)]$
$\frac{\partial^2 f}{\partial x}$	$\lim_{\Delta x \to 0} \frac{\frac{\partial f}{\partial x}(x+\Delta x, y) - \frac{\partial f}{\partial x}(x,y)}{\Delta x}$	$f(x+1,y) - 2f(x,y) + f(x-1,y)$
$\frac{\partial^2 f}{\partial y}$	$\lim_{\Delta y \to 0} \frac{\frac{\partial f}{\partial y}(x, y+\Delta y) - \frac{\partial f}{\partial y}(x,y)}{\Delta y}$	$f(x,y+1) - 2f(x,y) + f(x,y-1)$
$\nabla^2 f(x,y)$	$\frac{\partial^2 f}{\partial x} + \frac{\partial^2 f}{\partial y}$	$f(x+1,y) + f(x-1,y) - 4f(x,y)$ $+ f(x,y+1) + f(x,y-1)$

Derivative operators: formal continuous definition and corresponding discrete approximations

Derivative filters

- Differentiation is a *linear operation* - can thus be implemented by the kernel method
- Filter kernel response must be zero in completely smooth regions. => the weights in the kernel mask must sum to zero.
- "Straight derivatives" not generally the filter kernels of choice in practice. Detection of edges (which is the main application of derivative filters) is generally assisted by first smoothing them.
- Filter analogy: differentiation emphasizes edges, i.e. "high frequencies" so it's like a high-pass filter (so noise might upset things …)

Derivative filters (first order)

Roberts

x derivative

1	0
0	-1

y derivative

0	1
-1	0

Prewitt

x derivative

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

y derivative

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

Sobel

x derivative

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

y derivative

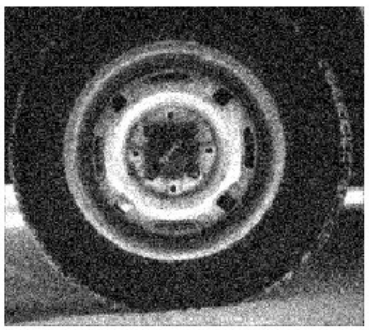
1	2	1
0	0	0
-1	-2	-1

Derivative filters

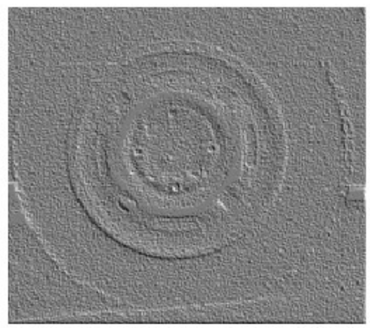
- The Roberts cross operators are simplest possible form for derivative operators.
- Prewitt and Sobel filters are better. They combine the derivative response with a degree of smoothing. This makes them *less susceptible to noise.*

Roberts

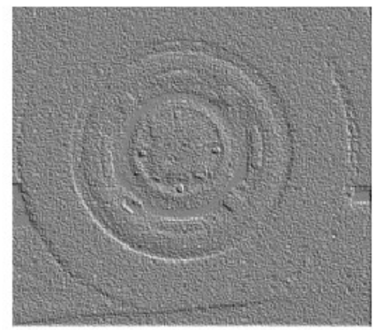
10% gaussian noise



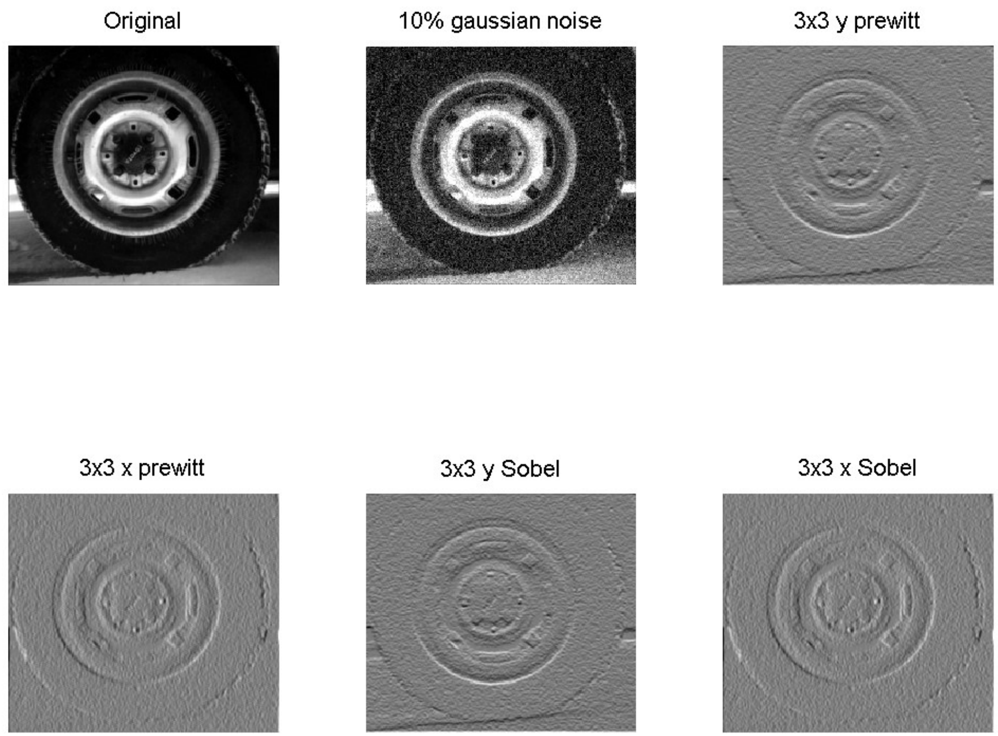
3x3 Roberts



3x3 Roberts



Sobel and Prewitt responses



Laplacian filter

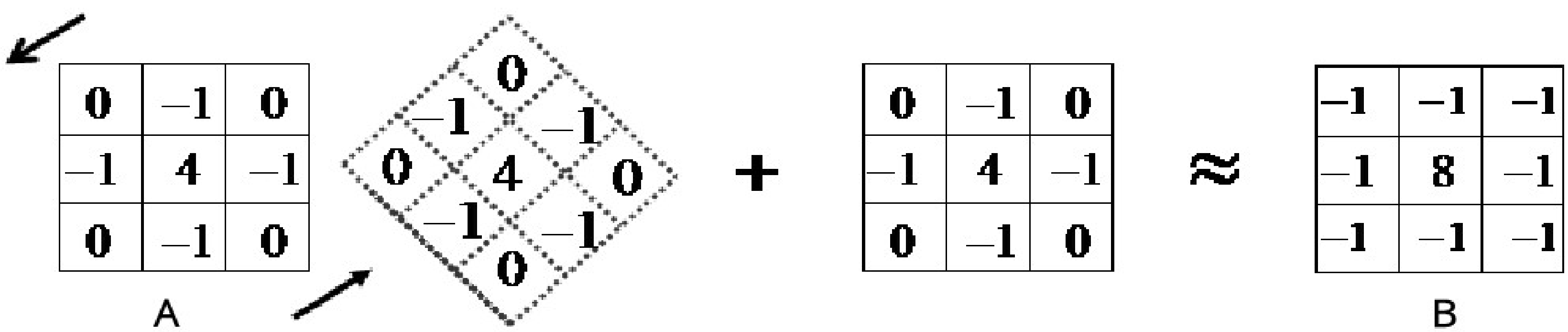
- First-order derivatives more commonly used for edge detection
- Other variants used for enhancement
- 2nd order derivative filter which also responds to edges.

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

- Derivative of derivative => tends to produce “sharper” edges
- Sensitive to noise
- Example

Laplacian filter - construction

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



Laplacian operator as edge detector

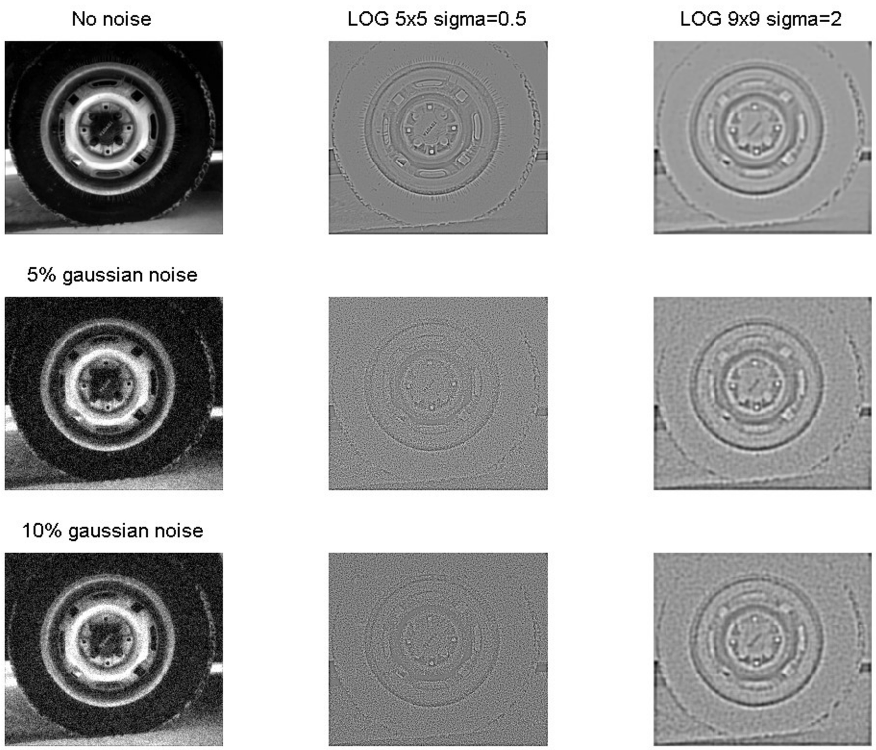
- To suppress sensitivity to noise we *can first smooth using a Gaussian filter*.
- Gaussian is a better smoothing filter to use than a straight “mean” filter
- This gives the Laplacian of Gaussian (LOG)
- Example

Image Sharpening using Laplacian

Laplacian responds to the fine detail in the image (those image regions where the change in gradient is significant) but has a zero response to constant regions and regions of smooth gradient in the image

Take the original image and add or subtract the Laplacian, we may expect to artificially enhance the fine detail in the image

Laplacian (LOG) as edge detector

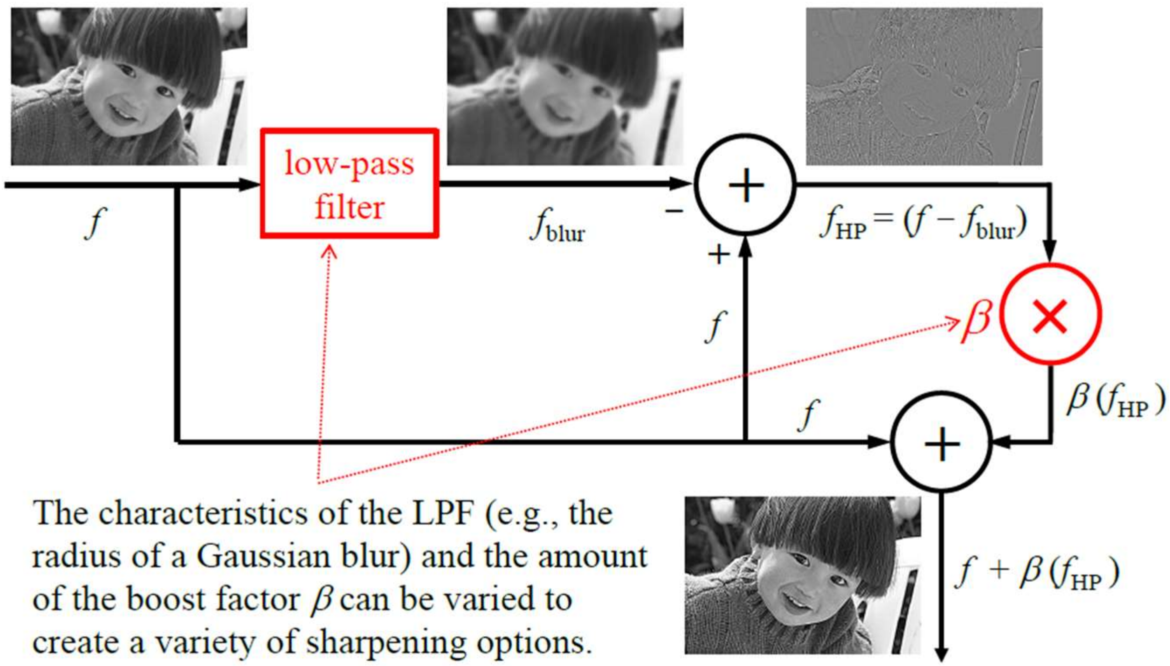


NUI Galway
OÉ Gaillimh

Unsharp Masking

- Subtract a smoothed version of an image, typically obtained by filtering with an averaging or a Gaussian filter kernel, from the original image itself. Add this to original.
- Why does this work ? Smooth regions of the original image will *not be changed significantly* by the smoothing filter. Secondly by contrast, edges and other regions in the image in which the intensity changes rapidly *will be affected*
- Example

Unsharp Masking



The characteristics of the LPF (e.g., the radius of a Gaussian blur) and the amount of the boost factor β can be varied to create a variety of sharpening options.

Edge Enhancement - Summary

- Edge detection and edge enhancement is used to:
  - Make images appear sharper
  - To detect boundaries, discontinuities within an image
- It is typically based on derivates within the image
  - First order – Sobel, Prewitt etc
  - Second order – LaPlacian
- Edge enhancement operations are often sensitive to noise
  - Sometimes a blur operation is performed beforehand
- Note: edge enhancement does not add information to the image
  - Existing details are accentuated, but no new details are added

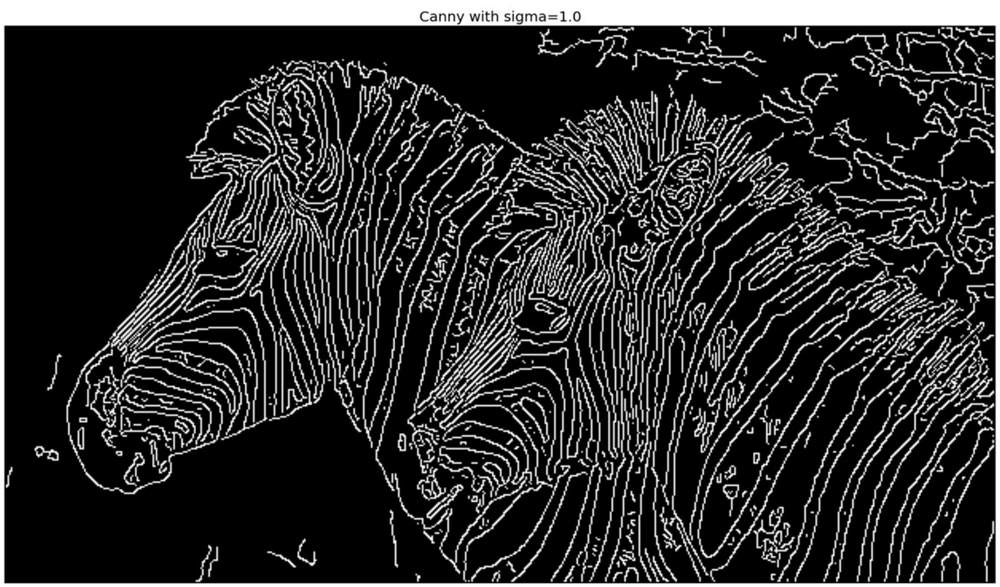
Edge Detection - Better Alternatives to Derivatives

- Laplacian of Gaussian
  - Gaussian does some noise removal
  - Laplacian looks for transitions – search for zero-crossings in Laplacian output
- Canny Edge Detector
  - Good all-round edge detection method – widely used in computer vision
  - Gaussian filtering used first – parameters of this impact the results (the bigger the kernel, the more smoothing and the stronger edges have to be to be detected)

Canny Edge Detector - Overview

- Algorithm criteria:
  - Low error rate: important that edges occurring in images should not be missed, and there should be no response where there is not an edge
  - The detected edge should be well localized: the distance between the edge pixels as found by the edge detector and the actual edge should be minimum
  - There should be only one response to a single edge*
- Algorithm steps
  - Smoothing – Gaussian blur
  - Find edge strength (H & V Sobel operators)
  - Calculate edge direction (Tan^{-1})
  - Digitize edge direction (0°, 45°, 90°, 135°)
  - Nonmaximal suppression (thins the edge)
  - Thresholding with Hysteresis

Comparison – LOG & Canny Edge Detectors



Questions?