FILTERING

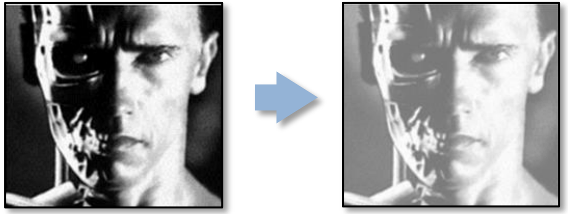
(DIGITAL IMAGE PROCESSING)

Faculty of Information Technology
Ton Duc Thang University

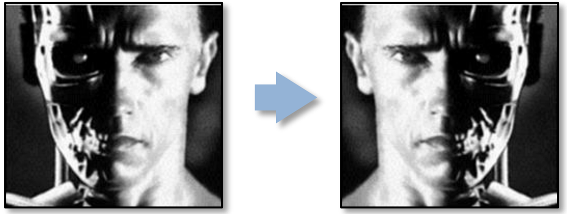
August 2023

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$

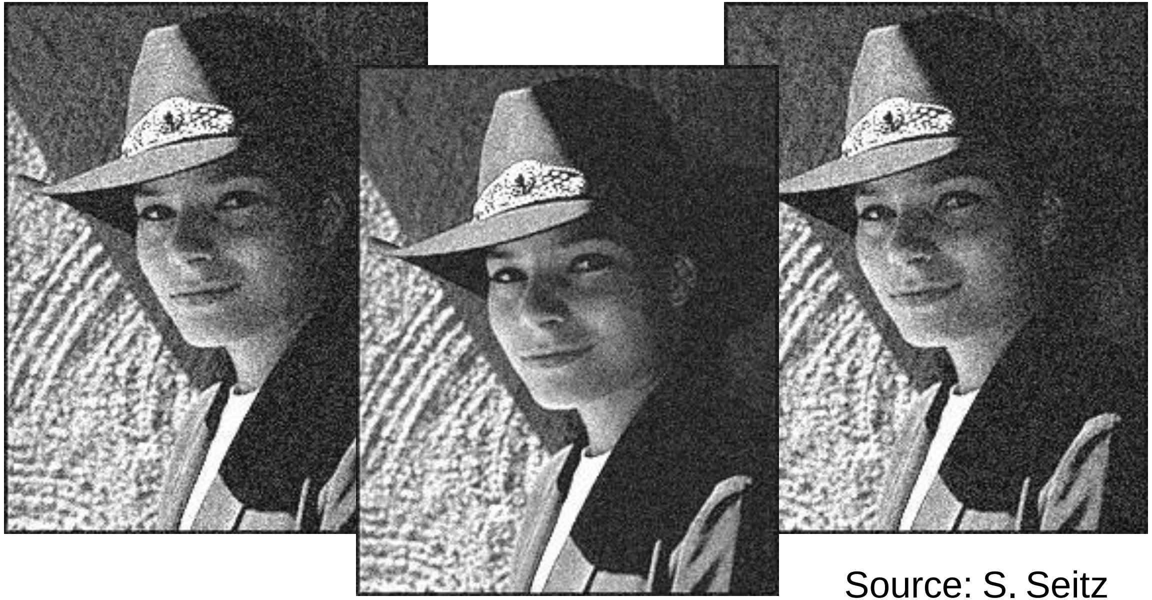


$g(x,y) = f(-x,y)$

- We’ll talk about a special kind of operator, convolution (linear filtering)

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?



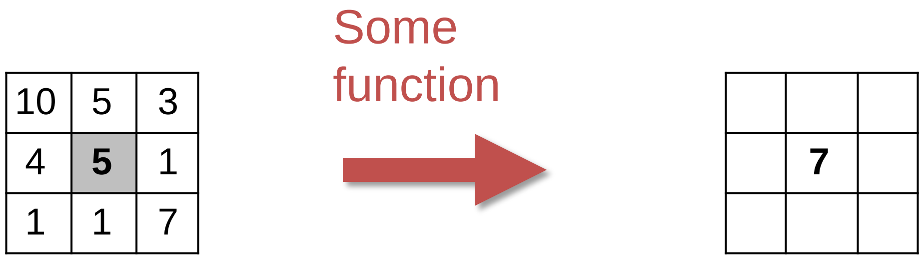
Source: S. Seitz

Take lots of images and average them!

What's the next best thing?

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel



Local image data

Modified image data

Source: L. Zhang

Image filtering

- **Filtering:**
  – Form a new image whose pixels are a combination original pixel values

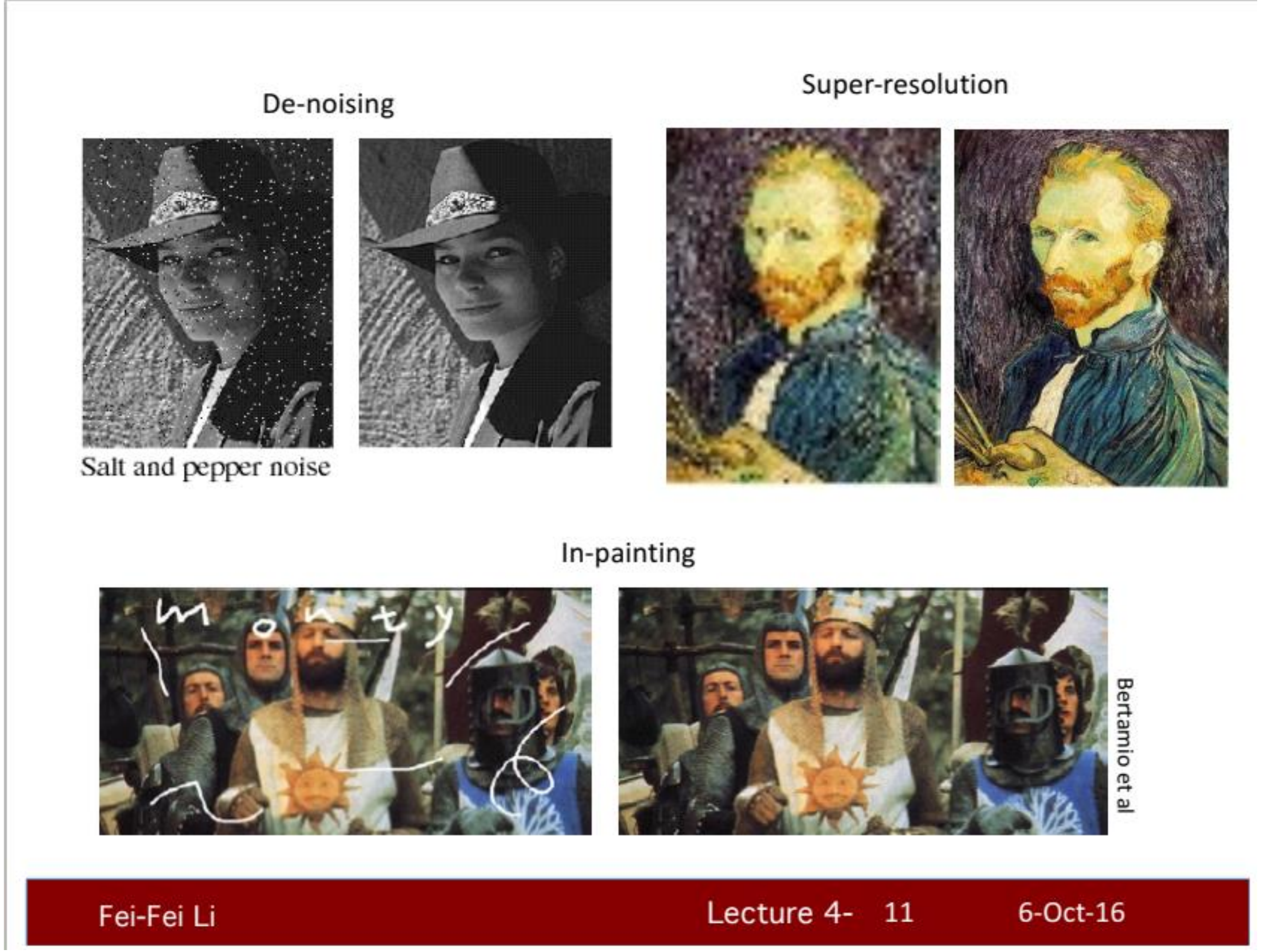
Goals:

-Extract useful information from the images

- Features (edges, corners, blobs…)

- Modify or enhance image properties:

- super-resolution; in-painting; de-noising


De-noising
Salt and pepper noise
Super-resolution
In-painting
Bertamio et al
Fei-Fei Li
Lecture 4- 11
6-Oct-16

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Correlation of Discrete Signals

Correlation is a measure of how similar signals are

$$Corr_{x,y} = \sum_{n=-\infty}^{\infty} x[n]\, y[n] \qquad Corr_{x,y} = \sum_{n=0}^{N-1} x[n]\, y[n]$$

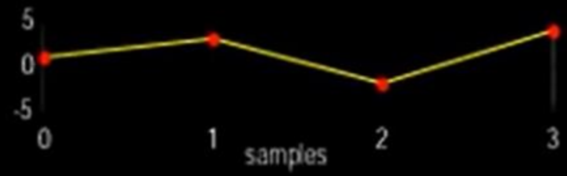
x = [1 3 -2 4]

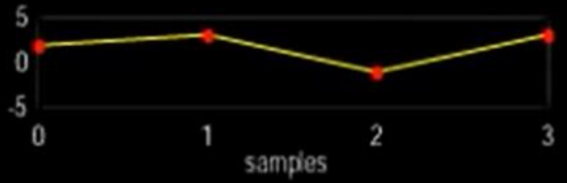
$corr_{x,y} = x[0]y[0] + x[1]y[1] + x[2]y[2] + x[3]\,y[3]$
$= (1)(2) + (3)(3) + (-2)(-1) + (4)(3)$
$= 2 + 9 + 2 + 12 = 25$

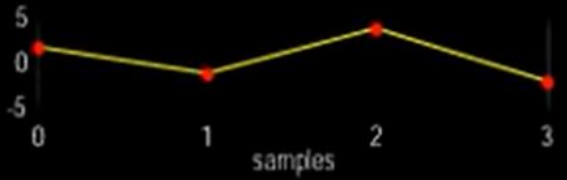
y = [2 3 -1 3]

$corr_{y,z} = y[0]z[0] + y[1]z[1] + y[2]z[2] + y[3]\,z[3]$
$= 2(2) + (3)(-1) + (-1)(4) + (3)(-2)$
$= 4 - 3 - 4 - 6 = -9$

z = [2 -1 4 -2]







YTB

Cross-correlation

- Let F be the image, H be the kernel (of size $2k+1 \times 2k+1$), and G be the output image:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called a **cross-correlation** operation:

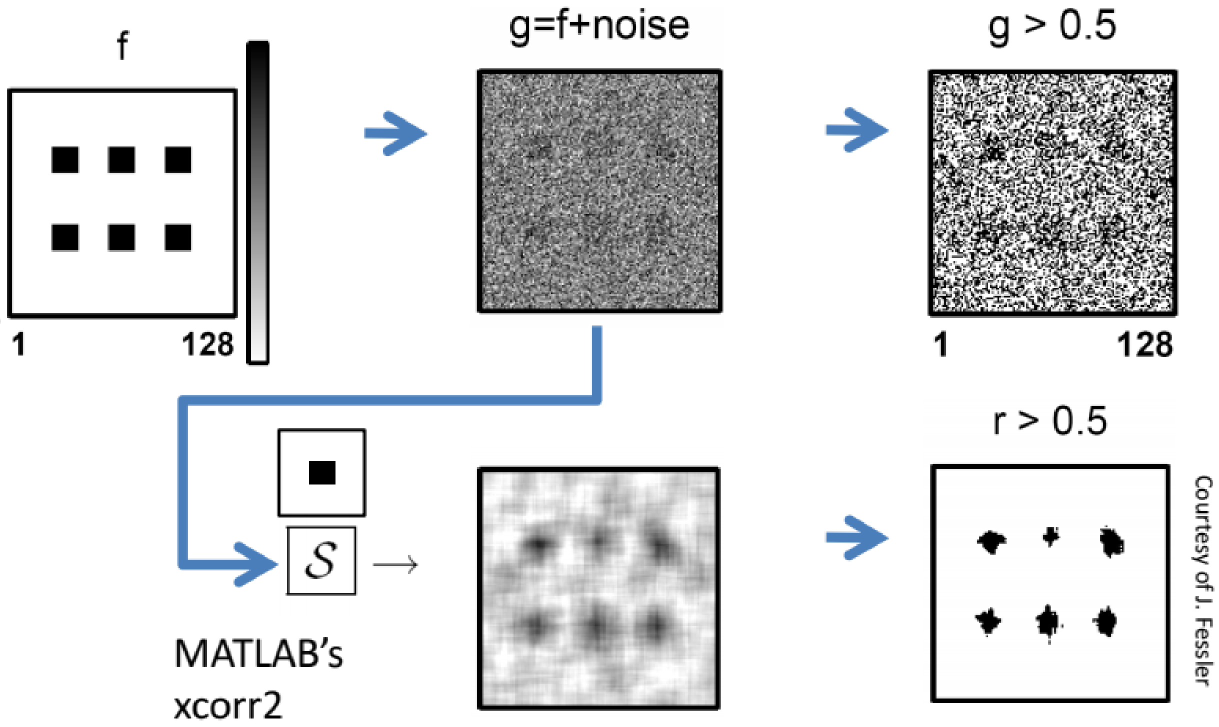
$$G = H \otimes F$$

  - When the aperture is partially outside the image, the operation interpolates outlier pixel values according to the specified border mode (refers [1])

Cross-correlation (ct)

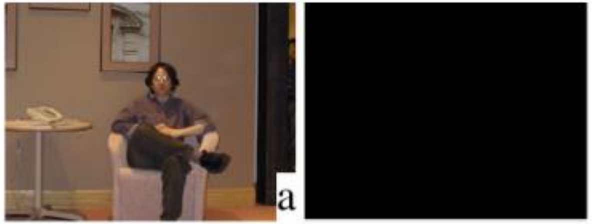
- How similar the kernel is to the image at any point [2]
  - Used for image alignment and simple image matching
  - Refers [3] [4] more about template matching and normalized cross-correlation.

(Cross) correlation – example



Cross Correlation Application: Vision system for TV remote control

- uses template matching



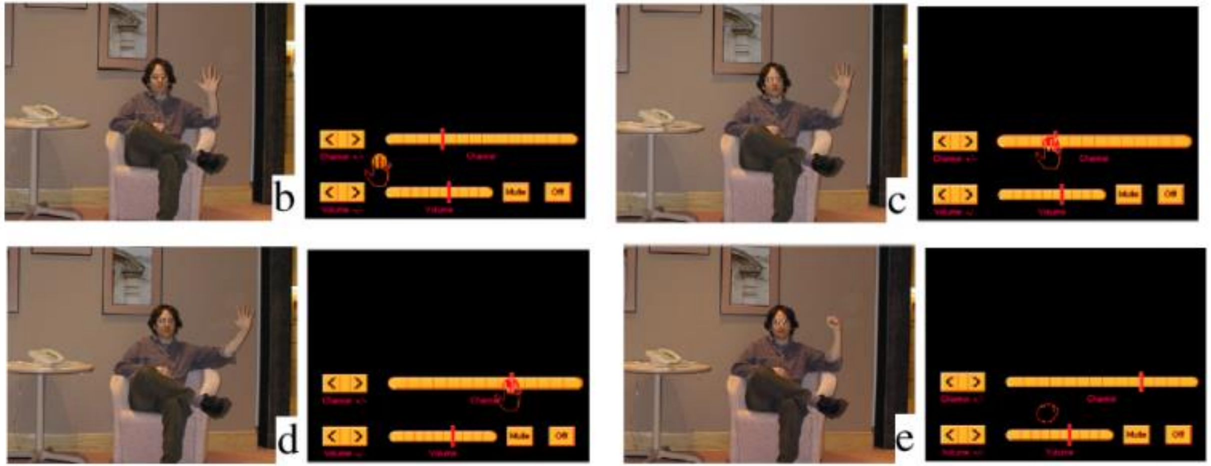


Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

Ex. 1

- Apply cross-correlation operation into the following image F:

F

1	2	3
4	5	6
8	9	10

H

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

$\Rightarrow$

G

?	?	?
?	$48/9$	?
?	?	?

Ex. 2

- Apply **normalized** cross-correlation operation to locate the best matching of H in the image F (zero padding for the border pixels):

F

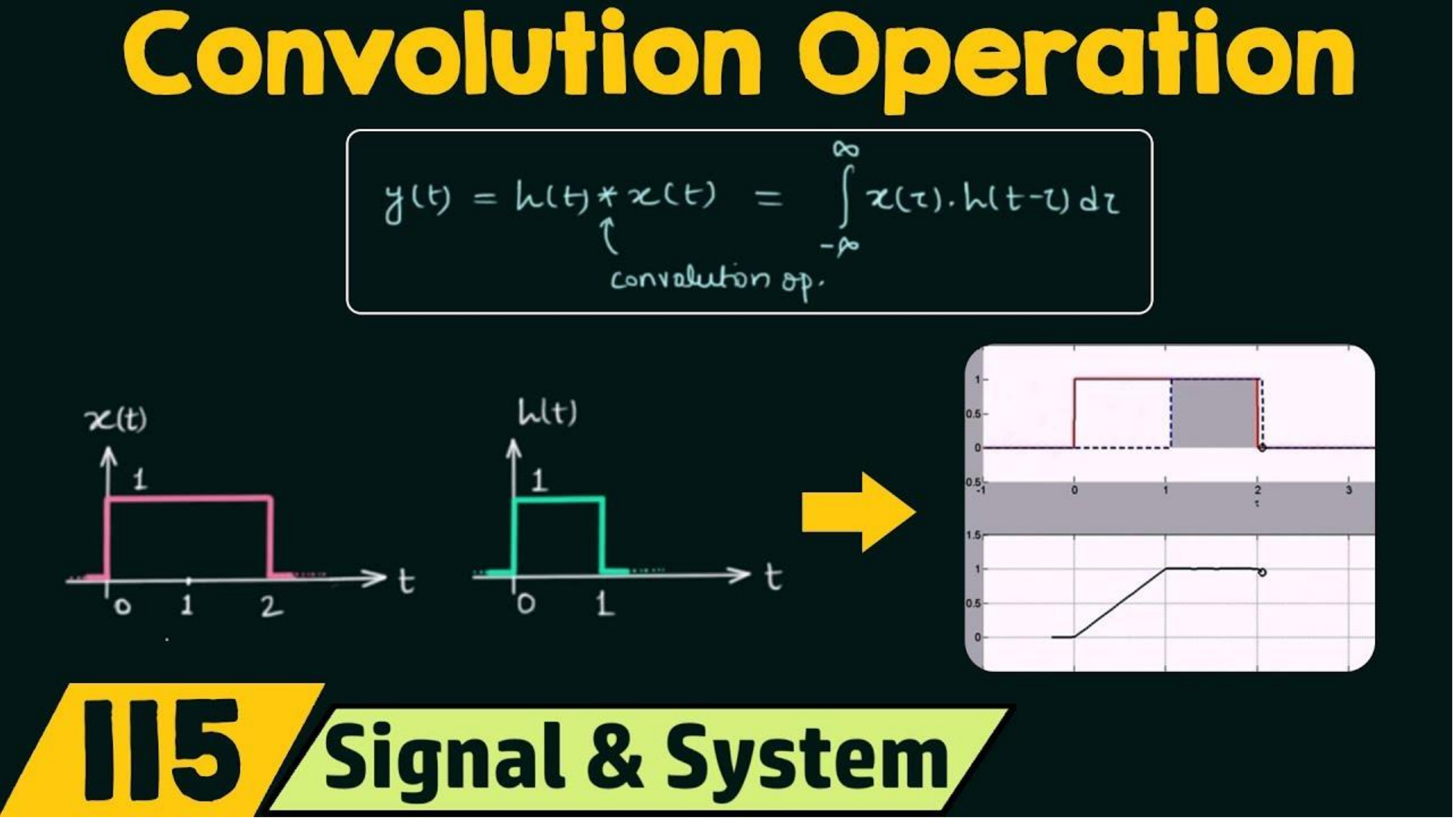
1	1	1	1	1	1
1	20	2	2	2	1
1	2	3	3	2	1
1	2	3	3	2	1
1	2	2	2	2	1
1	1	1	1	1	1

H

3	3	2
3	3	2
2	2	2

G

47	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?



https://www.youtube.com/watch?app=desktop&v=_HATc2zAhcY

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

This is called a **convolution** operation:

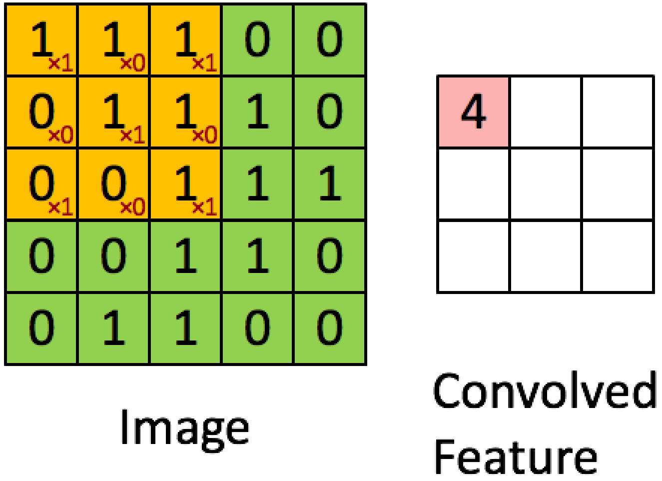
$$G = H * F$$

- Convolution is commutative and associative

2D Convolution

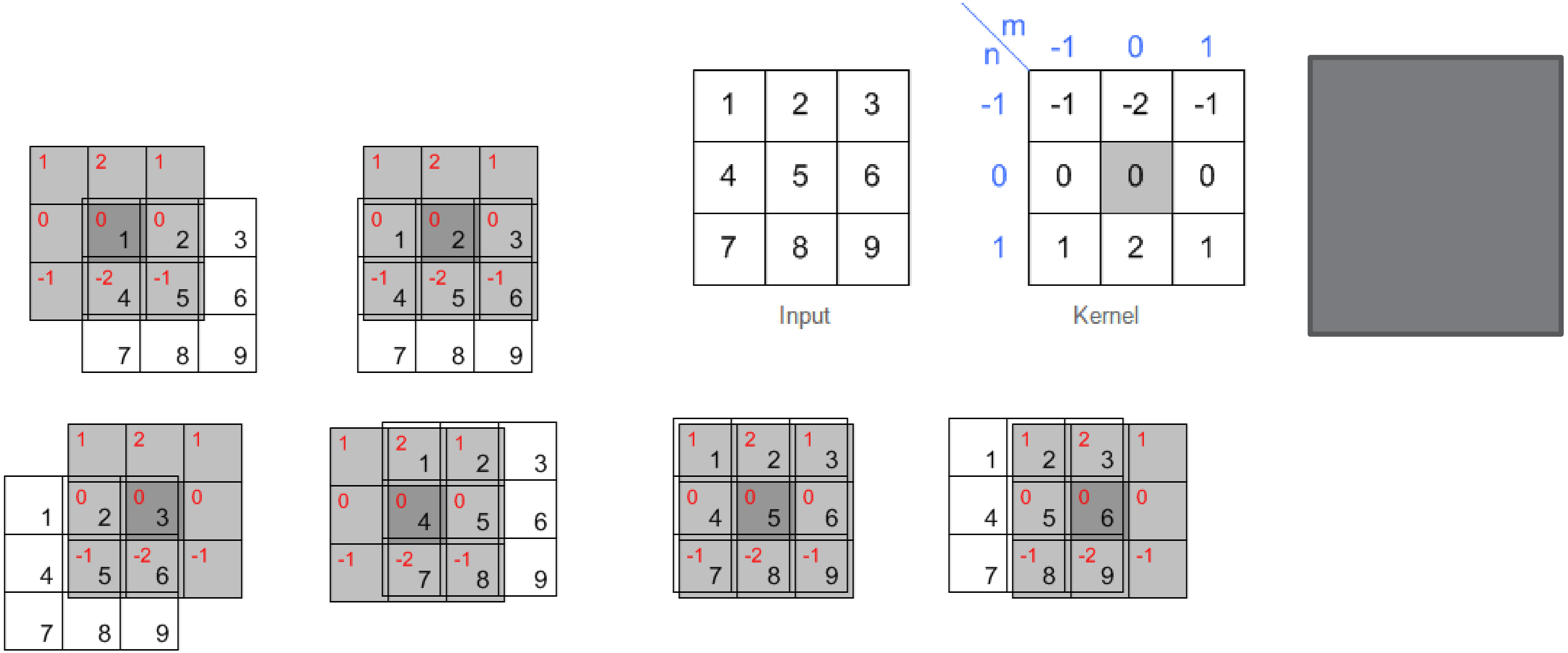
- g(x,y) = h(x,y) * f(x,y)
  - f, g: input/output
  - h: mask/filter/kernel

- **Flip** the mask (horizontally and vertically) only once
- Slide the mask onto the image.
- Multiply the corresponding elements and then add them
- Repeat this procedure until all values of the image has been calculated.

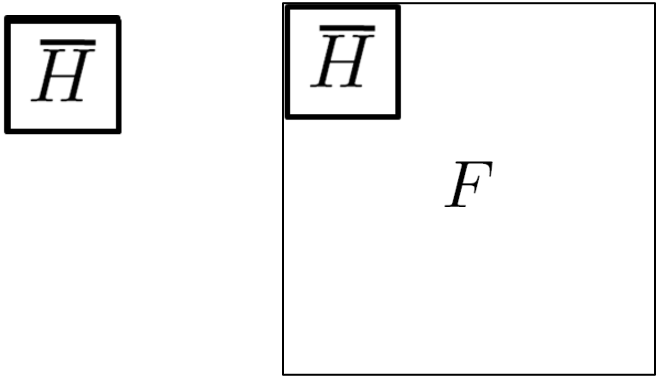


Example

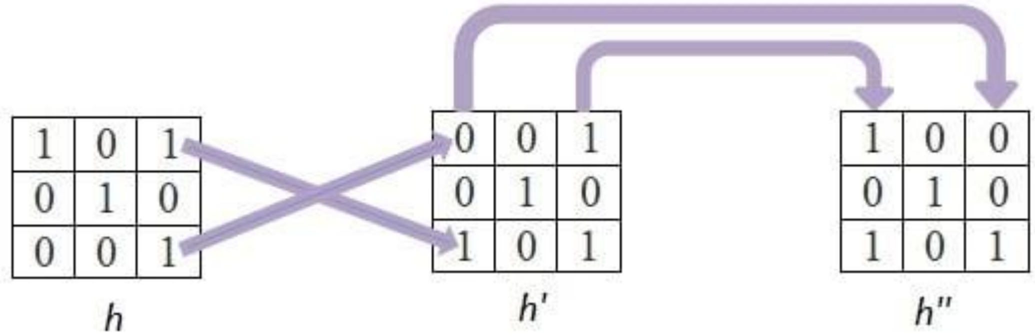
- http://www.songho.ca/dsp/convolution/convolution2d_example.html



CONVOLUTION



Adapted from F. Durand



https://www.allaboutcircuits.com/uploads/articles/Fig2_2D_Conv.jpg

Convolution applications

- Blur image
- Remove noise
- Sharpening
- Smoothing
- Edge detection
- ...

https://www.geeksforgeeks.org/python-opencv-filter2d-function/

Ex. 3

- Apply convolution operation into the following image F:

F

1	2	3
4	5	6
8	9	10

H

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

$\Rightarrow$

G

?	?	?
?	?	?
?	?	?

Convolution vs. (Cross) Correlation

- A **<u>convolution</u>** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation

- **<u>Correlation</u>** compares the ***similarity*** *of* ***two*** *sets of* ***data.*** Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best .
  - correlation is a measure of relatedness of two signals

Linear filters: examples

Original

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

Identical image

Source: D. Lowe

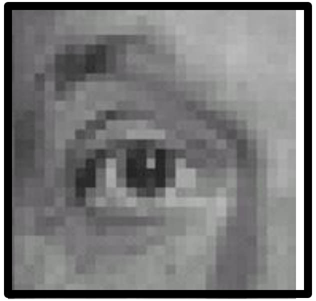
Linear filters: examples

Original

*

0	0	0
1	0	0
0	0	0

=

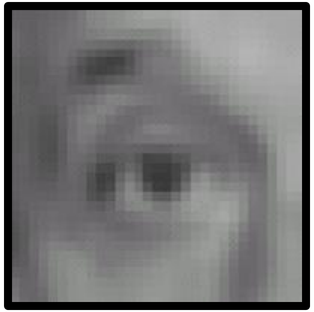


Shifted left By 1 pixel

Source: D. Lowe

Linear filters: examples

$$* \; \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$



Original

Blur (with a mean filter)

Source: D. Lowe

Mean filtering

H

$*$

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

F

$=$

	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

Ex. 4

- Apply the filtering (cross-correlation) into the following image F (zero padding at the borders):

F

1	1	1	1	1	1
1	180	180	180	180	1
1	180	96	96	180	1
1	180	96	96	180	1
1	180	180	180	180	1
1	1	1	1	1	1

H

0	0	0
1	0	0
0	0	0

→

G

?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?

Ex. 5

- Apply the filtering (cross-correlation) into the following image F (zero padding at the borders):

F

1	1	1	1	1	1
1	180	180	180	180	1
1	180	96	96	180	1
1	180	96	96	180	1
1	180	180	180	180	1
1	1	1	1	1	1

H

0	0	0
0	0	1
0	0	0

→

G

?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?

Linear filters: examples

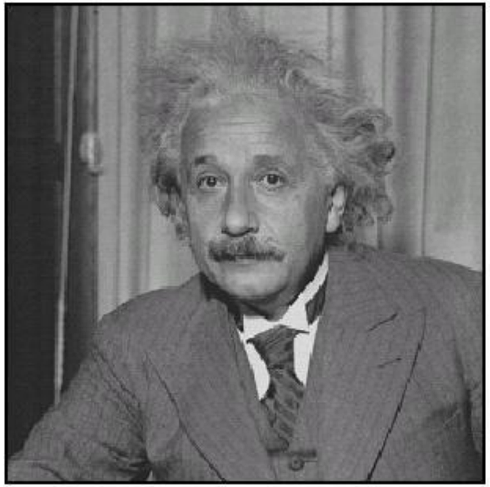
Original

$$* \left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \right) =$$

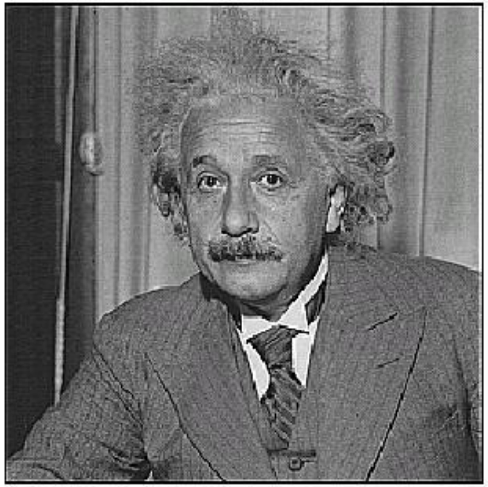
Sharpening filter
(accentuates edges)

Source: D. Lowe

SHARPENING



before

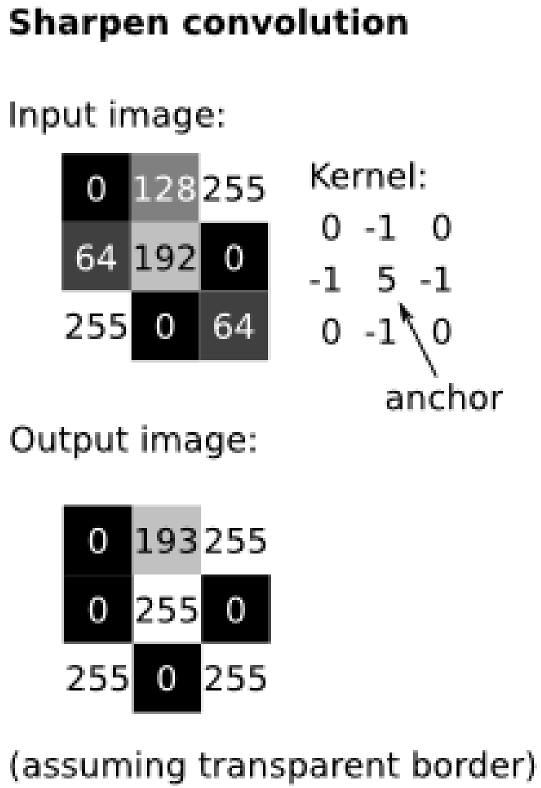


after

Source: D. Lowe

Sharpening

- emphasizes differences in adjacent pixel values
- accentuating the edges of the image
- add contrast to edges



https://i.stack.imgur.com/XXBUN.png

Ex. 6

- Apply the filtering (cross-correlation) into the following image F (zero padding at the borders):

F

20	20	20	20	20	20
20	120	120	120	120	20
20	120	20	20	120	20
20	120	20	20	120	20
20	120	120	120	120	20
20	20	20	20	20	20

H

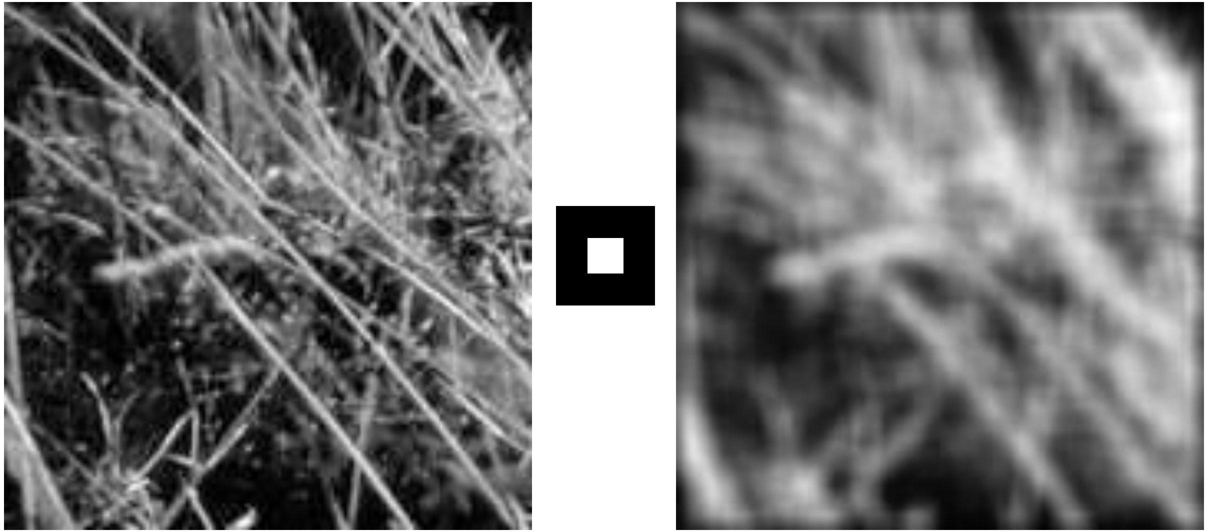
0	-1	0
-1	5	-1
0	-1	0

→

G

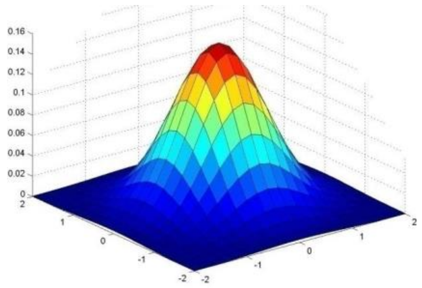
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?
?	?	?	?	?	?

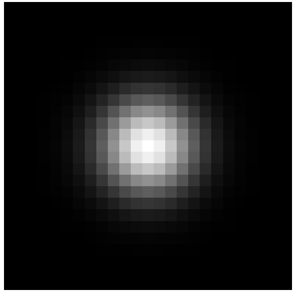
SMOOTHING WITH BOX FILTER REVISITED



Source: D. Forsyth

GAUSSIAN KERNEL





$\frac{1}{16}$ x

1	2	1
2	4	2
1	2	1

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

https://theailearner.com/2019/05/06/gaussian-blurring/

Source: C. Rasmussen

Discrete approximation of the Gaussian kernels

1/16

1	2	1
2	4	2
1	2	1

1/273

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/1003

0	0	1	2	1	0	0
0	3	13	22	13	3	0
1	13	59	97	59	13	1
2	22	97	159	97	22	2
1	13	59	97	59	13	1
0	3	13	22	13	3	0
0	0	1	2	1	0	0

https://www.researchgate.net/figure/Discrete-approximation-of-the-Gaussian-kernels-3x3-5x5-7x7_fig2_325768087

Gaussian blur

- Use a weighted mean: the values near the center pixel will have a higher weight
  - probably get a less blurred image but a natural blurred image because it handles the edge values very well

https://theailearner.com/tag/gaussian-filter/

Gaussian kernel - properties

- Gaussian kernel is linearly separable: can break any 2-d filter into two 1-d filters
  - Applying multiple successive Gaussian kernels is equivalent to applying a single, larger Gaussian blur

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

- Gaussian kernel weights(1-D) can be obtained quickly using the Pascal's Triangle

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

https://theailearner.com/tag/gaussian-filter/

Gaussian blur – code

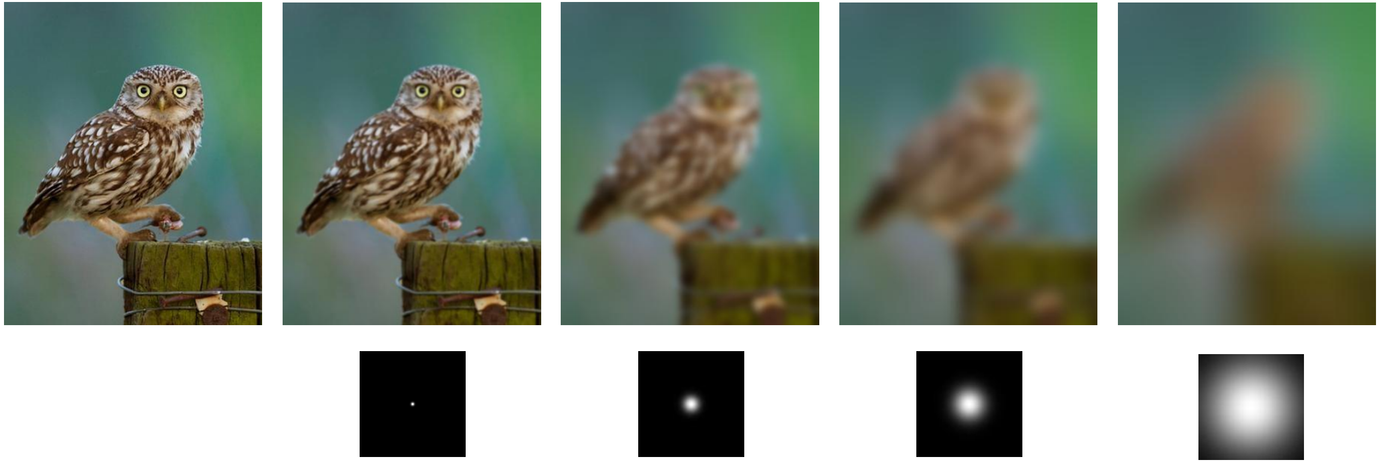
```
import cv2
img = cv2.imread('D:/downloads/opencv_logo.PNG')

# Creates a 1-D Gaussian kernel
a = cv2.getGaussianKernel(5,1)

# Apply the above Gaussian kernel. Here, I
# have used the same kernel for both X and Y
b = cv2.sepFilter2D(img,-1,a,a)

# Display the Image
cv2.imshow('a',b)
cv2.waitKey(0)
```

Gaussian filters



$\sigma = 1$ pixel

$\sigma = 5$ pixels

$\sigma = 10$ pixels

$\sigma = 30$ pixels

Sharpening revisited

–

=

Source: S. Lazebnik

Let's add it back:

+ α

=

$$\mathrm{F} + \alpha(F - H * F)$$

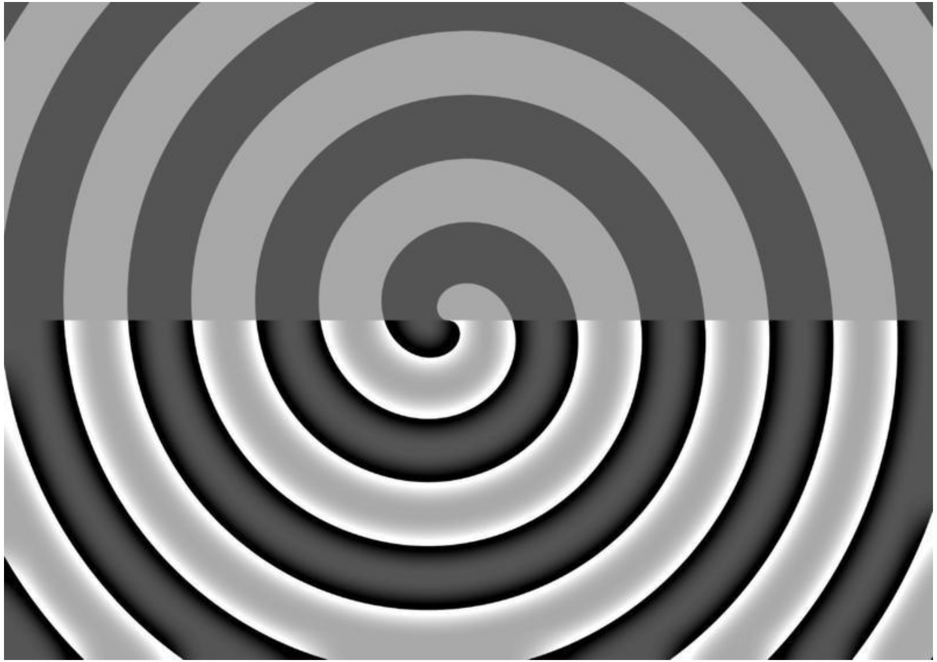
image

blurred image

Sharpen filter

unfiltered

filtered



Convolution in the real world

Camera shake

=

*

Source: Fergus, *et al.* "Removing Camera Shake from a Single Photograph", SIGGRAPH 2006

Bokeh: Blur out-of-focus regions of an image.

Source: http://lullaby.homepage.dk/diy-camera/bokeh.html

Rank filters

- Rank filters assign the k-th value of the gray levels from the window consisting of M pixels sorted in ascending order [code]
  - The special cases k = 1, k = M (MIN and MAX filter) : erosion and dilation
  - k = (M + 1)/2 : median filter
- Generalisation of flat dilation/erosion: in lieu of min or max value in window, use the k-th ranked value
- Increases robustness against noise
  - Best-known example: median filter for noise reduction
- Concept useful for both gray-level and binary images
- All rank filters are commutative with thresholding

Rank filters - benefits

- image quality enhancement, e.g., image smoothing, sharpening
- image pre-processing, e.g., noise reduction, contrast enhancement
- feature extraction, e.g., border detection, isolated point detection
- image post-processing, e.g., small object removal, object grouping, contour smoothing

Median filter

- Gray-level median filter

$$g[x,y] = median[W\{f[x,y]\}] := median(f,W)$$

- Binary images: majority filter

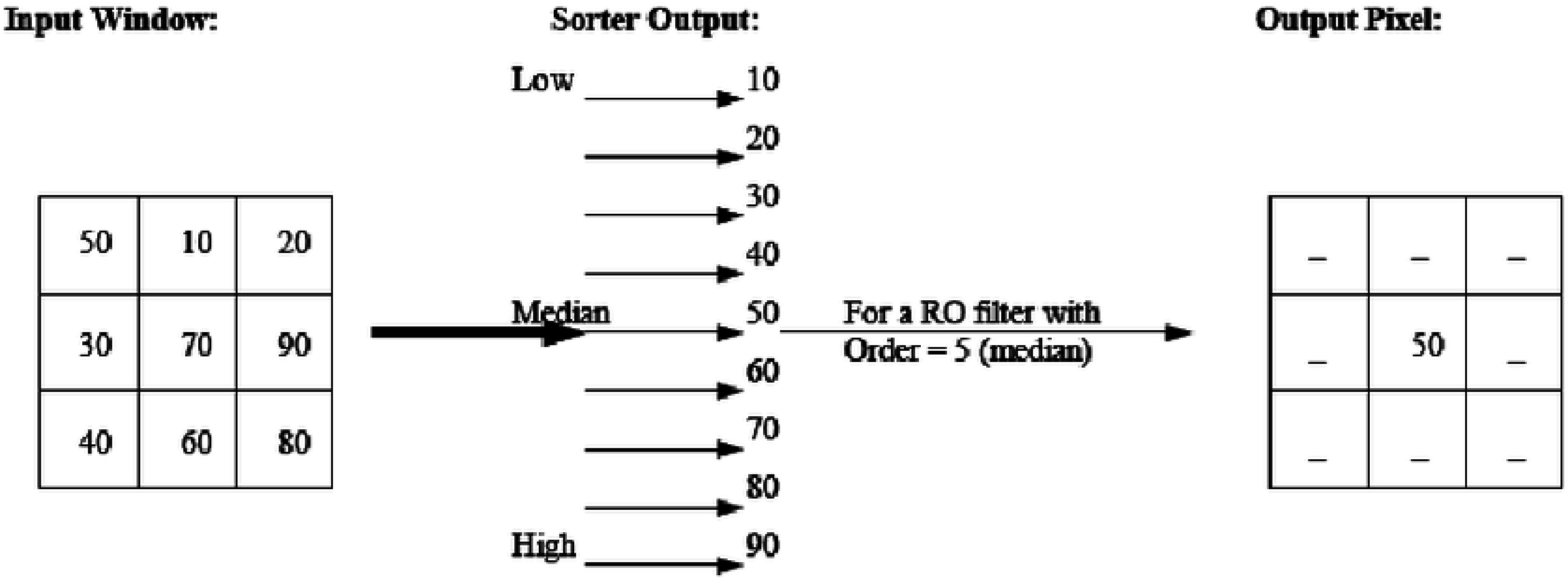
$$g[x,y] = MAJ[W\{f[x,y]\}] := majority(f,W)$$

- Self-duality

$$median(f,W) = -[median(-f,W)]$$

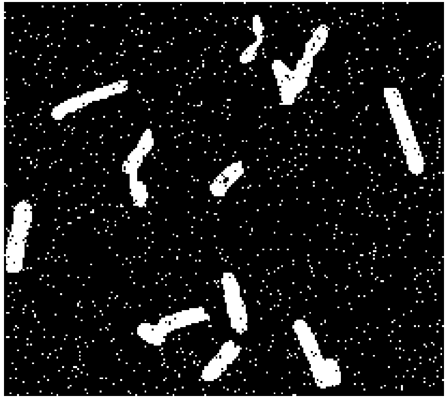
$$majority(f,W) = NOT[majority(NOT[f],W)]$$

Median filter

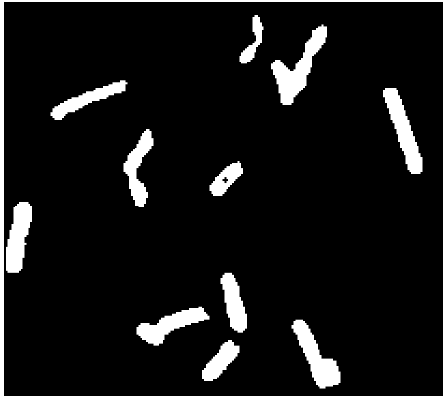


https://www.researchgate.net/figure/Graphic-Depiction-of-Rank-Order-Filter-Operation_fig6_268373873

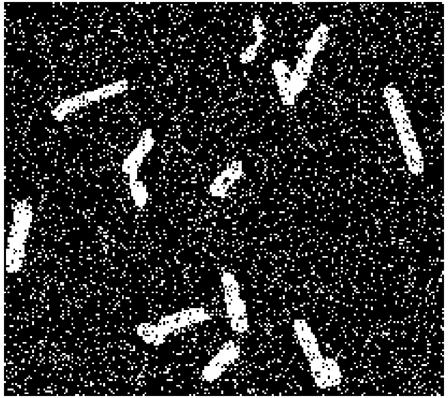
Majority filter: example



Binary image with
5% ‘Salt&Pepper’ noise



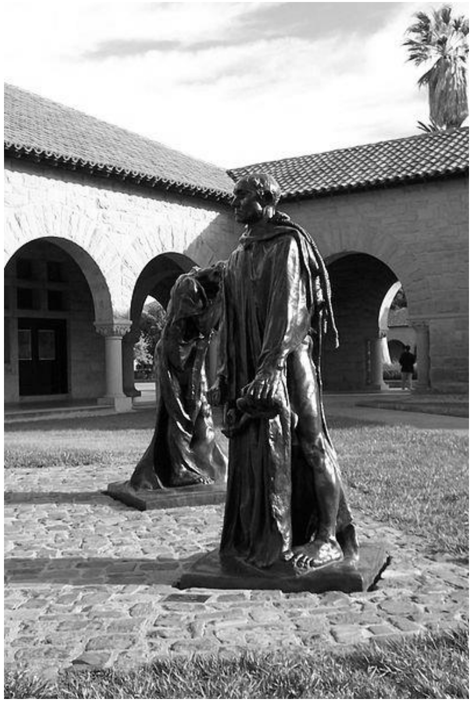
3x3 majority filter



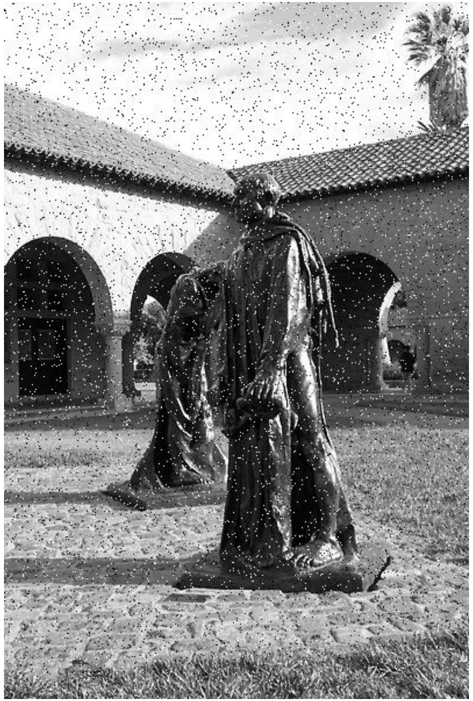
20% ‘Salt&Pepper’ noise

3x3 majority filter

Median filter: example



Original image

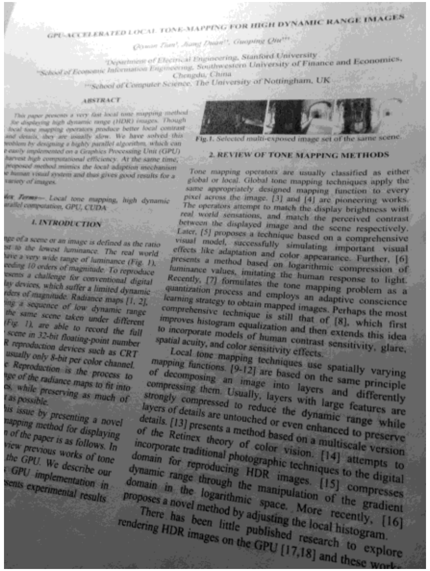


5% ‘Salt&Pepper’ noise

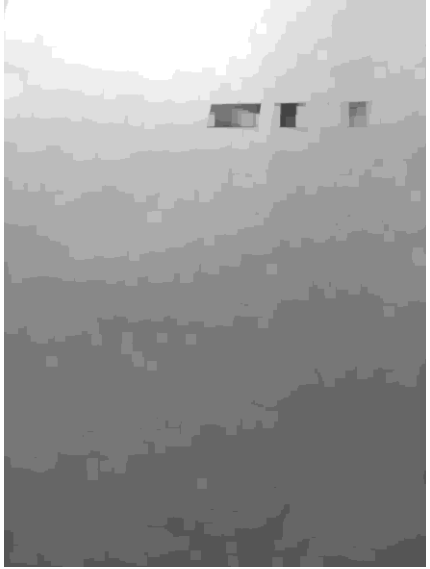
3x3 median filtering

7x7 median filtering

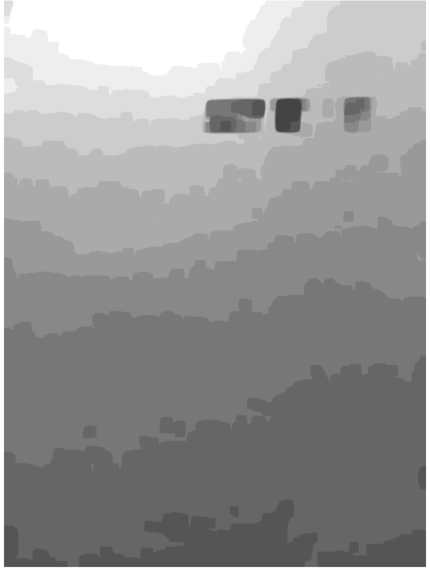
Example: non-uniform lighting compensation



Original image
1632x1216 pixels



Dilation (local max)
61x61 structuring element



Rank filter
10st brightest pixel
61x61 structuring element

References

1. https://docs.opencv.org/4.x/d4/d86/group__imgproc__filter.html
2. https://www.youtube.com/watch?app=desktop&v=kGHz-cEyjiE
3. https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html
4. https://www.youtube.com/watch?app=desktop&v=kGHz-cEyjiE
5. https://vincmazet.github.io/bip/filtering/convolution.html