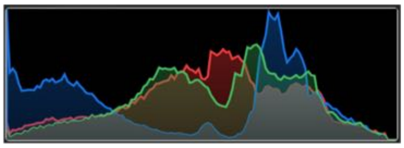
Image Processing
INT3404 20

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.
Email: ngocdiep@vnu.edu.vn
Slide & code: https://github.com/chupibk/INT3404_20

Recall week 3: Histogram



An image with L-level intensities
r_k : intensity level k (k = 0, 1, 2, ... , L-1)
n_k : number of pixels with intensity r_k

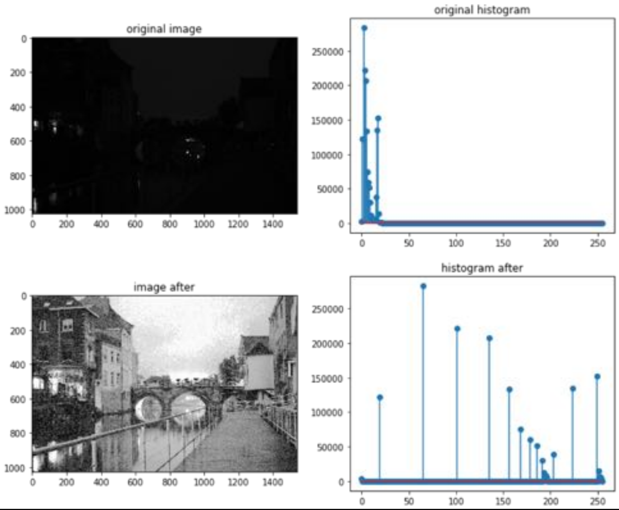
Unnormalized histogram:

$$h(r_k) = n_k$$

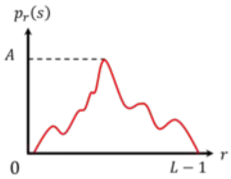
Normalized histogram:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

Recall week 3:
Histogram and image appearance

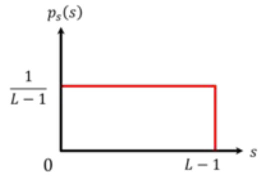


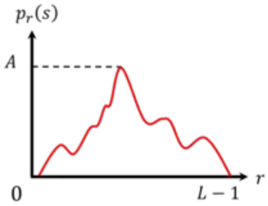
Recall week 3:
Histogram equalization and matching



Histogram Equalization

A transformation function that generates an output image with a uniform histogram

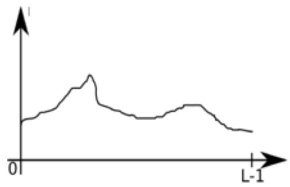




source

Histogram Matching

A transformation function that generates an output image with a specified histogram



template

Schedule

Week	Content	Homework
1	Introduction	Set up environments: Python 3, OpenCV 3, Numpy, Jupyter Notebook
2	Digital image – Point operations Contrast adjust – Combining images	HW1: adjust gamma to find the best contrast
3	Histogram - Histogram equalization – Histogram-based image classification	Self-study
4	Spatial filtering - Template matching	Self-study
5	Feature extraction Edge, Line, and Texture	Self-study
6	Morphological operations	HW2: Barcode detection → Require submission as mid-term test
7	Filtering in the Frequency domain Announcement of Final project topics	Final project registration
8	Color image processing	HW3: Conversion between color spaces, color image segmentation
9	Geometric transformations	Self-study
10	Noise and restoration	Self-study
11	Compression	Self-study
12	Final project presentation	Self-study
13	Final project presentation Class summarization	Self-study

5

Week 4: Spatial filtering

Neighbors of a pixel

	(x-1, y)	
(x, y – 1)	**(x, y)**	(x, y+1)
	(x+1, y)	

4 - neighbors

(x-1, y-1)	(x-1, y)	(x-1, y+1)
(x, y – 1)	**(x, y)**	(x, y+1)
(x+1, y-1)	(x+1, y)	(x+1, y+1)

8 - neighbors

Distance between two pixels (1/2)

2 pixels p=(x, y) and q=(u,v)

Euclidean distance: $D_e(p,q) = \left[(x-u)^2 + (y-v)^2\right]^{\frac{1}{2}}$

City-block distance: $D_4(p,q) = \underbrace{|x-u| + |y-v|}_{\text{All pixels that are less than or equal to some value d form a diamond centered at (x, y)}}$

Example:

$$\begin{array}{ccc} & 1 & \\ 1 & 0 & 1 \\ & 1 & \end{array}$$

D_4 = 1 (→ 4 neighbors)

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

D_4 = 2

Distance between two pixels (2/2)

2 pixels p=(x, y) and q=(u,v)

Chessboard distance: $D_8(p,q) = \max(\underbrace{|x-u|, |y-v|}_{\text{All pixels that are less than or equal some value d form a square centered at (x, y)}})$

Example:

$$\begin{matrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{matrix}$$

D_8 = 1 (→ 8 neighbors)
square size: 3x3

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

D_8 = 2
square size: 5x5

Spatial filter kernel

- Also called: mask, template, window, filter, kernel
- A kernel: an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter
- Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors

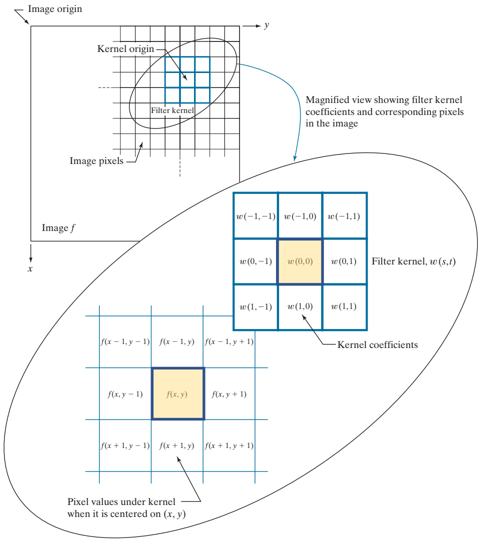
Linear spatial filtering mechanism

- A linear spatial filter performs a sum-of-products operation between an image *f* and a filter kernel, *w*
- Kernel center *w(0,0)* aligns with the pixel at location *(x,y)*

Kernel size: m x n
m = 2a + 1
n = 2b + 1
Image size: M x N

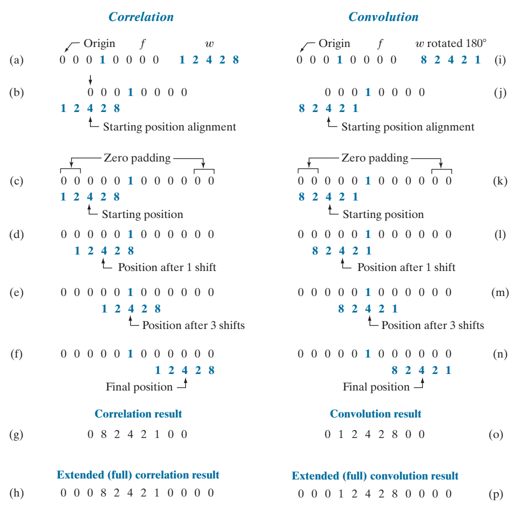
Linear spatial filtering:

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$



Source: Fig. 3.28, Gonzalez

Spatial correlation and convolution in 1D



At the location of the impulse (1)

Source: Fig. 3.29, Gonzalez

Spatial correlation and convolution in 1D

Correlation

Origin f w

$0\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ 0 \qquad \mathbf{1\ 2\ 4\ 2\ 8}$

Correlation result

$0\ 8\ 2\ \boxed{4}\ 2\ 1\ 0\ 0$

yield a copy of w,
but rotated by 180º

Convolution

Origin f w rotated 180°

$0\ 0\ 0\ \boxed{1}\ 0\ 0\ 0\ 0 \qquad \mathbf{8\ 2\ 4\ 2\ 1}$

Convolution result

$0\ 1\ 2\ \boxed{4}\ 2\ 8\ 0\ 0$

yield a copy of w,
by pre-rotating w before performing
shifting/sum-of-products

At the location of the impulse (1)

Source: Fig. 3.29, Gonzalez

Correlation and Convolution in 2D

Origin f $\quad w$

$$\begin{matrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{matrix} \qquad \begin{matrix} 1&2&3\\ 4&5&6\\ 7&8&9 \end{matrix}$$

(a)

Padded f

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(b)

Initial position for w

$$\begin{matrix} 1&2&3&0&0&0&0\\ 4&5&6&0&0&0&0\\ 7&8&9&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(c)

Correlation result

$$\begin{matrix} 0&0&0&0&0\\ 0&9&8&7&0\\ 0&6&5&4&0\\ 0&3&2&1&0\\ 0&0&0&0&0 \end{matrix}$$

(d)

Full correlation result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&9&8&7&0&0\\ 0&0&6&5&4&0&0\\ 0&0&3&2&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(e)

Rotated w

$$\begin{matrix} 9&8&7&0&0&0&0\\ 6&5&4&0&0&0&0\\ 3&2&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(f)

Convolution result

$$\begin{matrix} 0&0&0&0&0\\ 0&1&2&3&0\\ 0&4&5&6&0\\ 0&7&8&9&0\\ 0&0&0&0&0 \end{matrix}$$

(g)

Full convolution result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&2&3&0&0\\ 0&0&4&5&6&0&0\\ 0&0&7&8&9&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(h)

Source: Fig. 3.30, Gonzalez

Correlation vs Convolution

Correlation

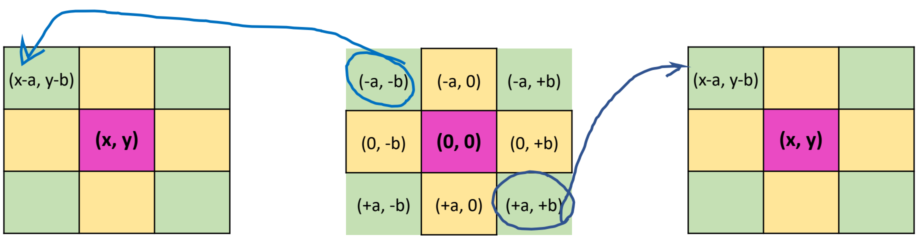
$$(w \star f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Convolution

$$(w \bigstar f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Correlation vs convolution

$$(w \star f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$



$$(w \bigstar f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Fundamental properties of convolution and correlation

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

Boundary issues

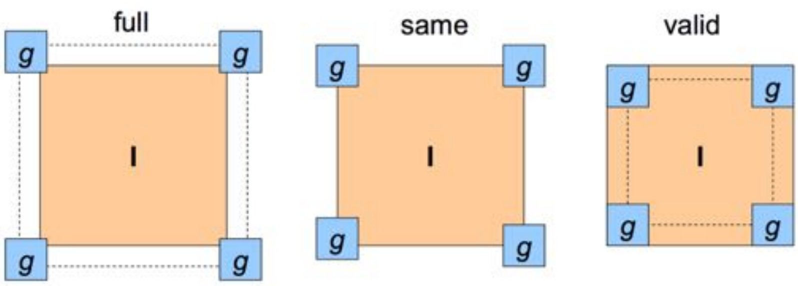


Image size: MxN
Kernel size: mxn

Output: (M + m – 1) x (N +n – 1) M x N (M-m+1) x (N – n + 1)

What to do around the borders

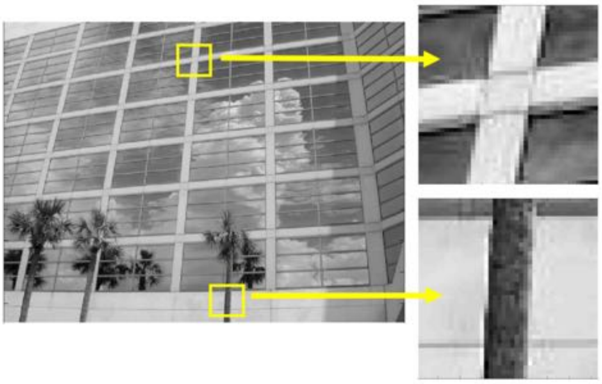
- Pad a constant value (black)
- Wrap around (circulate the image)
- Copy edge (replicate the edges' pixels)
- Reflect across edges (symmetric)



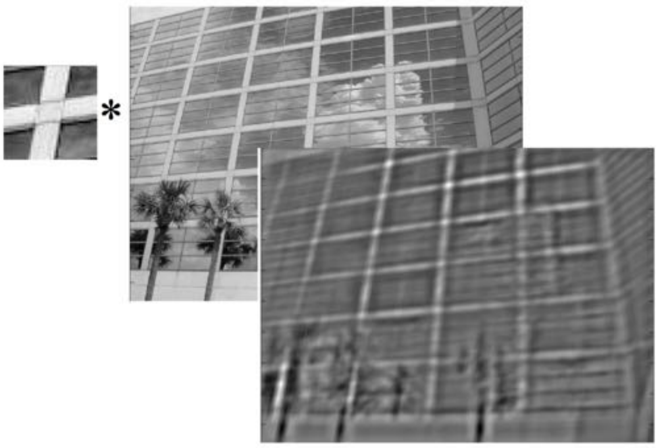

Template matching

Template matching

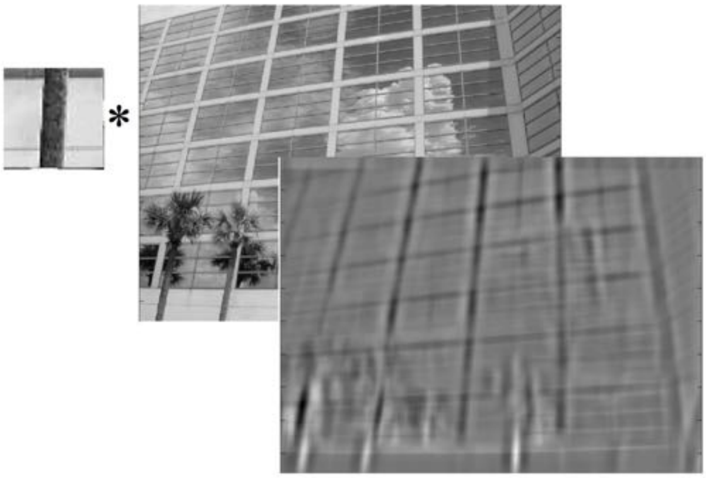
- What if we cut little pictures out from an image, then tried to do cross correlation them with the same or other images?



Template matching



Template matching



Actually...

I subtracted the mean gray value from both the image and the template before doing cross correlation.

Why?

Problem with correlation of raw image template

Consider correlation of template with an image of constant grey value:

$$\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline v & v & v \\ \hline v & v & v \\ \hline v & v & v \\ \hline \end{array}$$

Result: v*(a+b+c+d+e+f+g+h+i)

Problem with correlation of raw image template

Now consider correlation with a constant image that is twice as bright

$$\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array} \otimes \begin{array}{|c|c|c|}\hline 2v & 2v & 2v \\ \hline 2v & 2v & 2v \\ \hline 2v & 2v & 2v \\ \hline \end{array}$$

Result: 2*v*(a+b+c+d+e+f+g+h+i)
> v*(a+b+c+d+e+f+g+h+i)

Larger score, regardless of what the template is!

Solution

- Subtract off the mean value of the template
- In this way, the correlation score is higher only when darker parts of the template overlap darker parts of the image, and the brighter parts of the template overlap brighter parts of the image

Correspondence problem

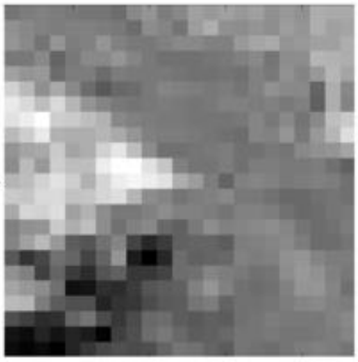
Finding corresponding feature across two or more views

Image 1 Image 2

Note: this is a stereo pair from the NASA mars rover.
The rover is exploring the “El Capitan” formation

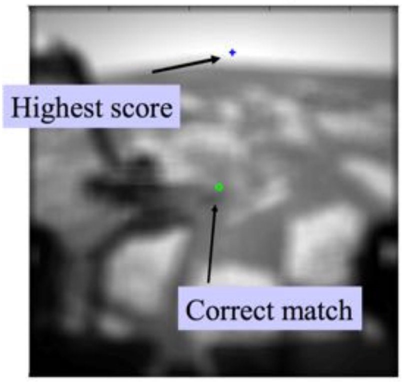
Example

Image 1



Template
(image patch)

Example: Raw cross-correlation



Score around
correct match

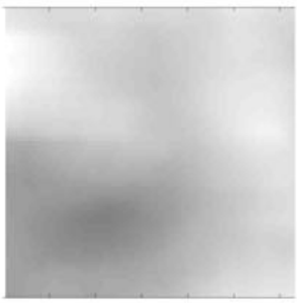
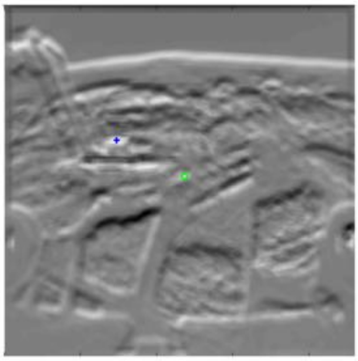


Image 2

Example: Cross-correlation with zero-mean template




Better! But highest score is still not the correct match.
Note: highest score IS best within local neighborhood of correct match.

"SSD" or "Block matching" (Sum of squared differences)

$$\sum_{[i,j]\in R} (f(i,j) - g(i,j))^2$$

1 – The most popular matching score
2 – T&V claim it works better than cross-correlation

Relation between SSD and Correlation

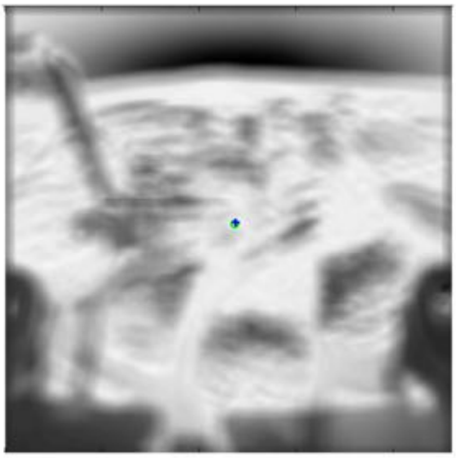
$$SSD = \sum_{[i,j]\in R} (f-g)^2$$

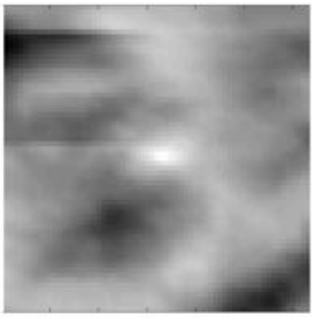
$$= \sum_{[i,j]\in R} f^2 + \sum_{[i,j]\in R} g^2 - 2 \sum_{[i,j]\in R} fg$$

$$C_{fg} = \sum_{[i,j]\in R} f(i,j)g(i,j)$$

Correlation!

SSD





Best match (highest score) in image coincides with correct match in this case!

Handling intensity changes

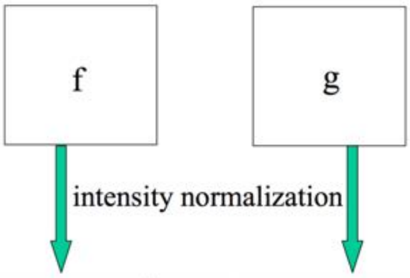
- the camera taking the second image might have different intensity response characteristics than the camera taking the first image
- Illumination in the scene could change
- The camera might have auto-gain control set, so that it's response changes as it moves through the scene.

Intensity normalization

- When a scene is imaged by different sensors, or under different illumination intensities, both the SSD and the C_{fg} can be large for windows representing the same area in the scene!
- A solution is to NORMALIZE the pixels in the windows before comparing them by subtracting the mean of the patch intensities and dividing by the std.dev.

$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum(f - \bar{f})^2}} \qquad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum(g - \bar{g})^2}}$$

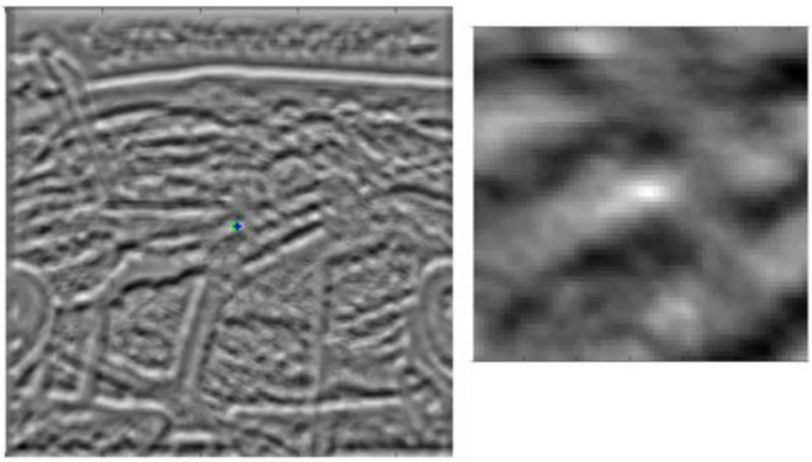
Normalized cross correlation



$$\hat{f} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}} \qquad \hat{g} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}$$

$$\boxed{\mathrm{NCC(f,g)} = C_{fg}(\hat{f}, \hat{g}) = \sum_{[i,j] \in R} \hat{f}(i,j)\hat{g}(i,j)}$$

Normalized cross correlation



Highest score also coincides with correct match.
Also, looks like less chances of getting a wrong match.

Normalized cross correlation

- Important point about NCC:
  - Score values range from 1 (perfect match) to -1 (completely anti-correlated)
- Intuition: treating the normalized patches as vectors, we see they are unit vectors. Therefore, correlation becomes dot product of unit vectors, and thus must range between -1 and 1

Spatial filter kernels

Filter design

- Based on mathematical properties
  - E.g.: A filter that computes the average of pixels in a neighborhood blurs an image
  - E.g.: A filter that computes the local derivative of an image sharpens the image
- Based on sampling a 2D spatial function whose shape has a desired property
  - E.g.: Samples from a Gaussian function to construct a weighted-average filter
- Based on a specified frequency response

Smoothing filters

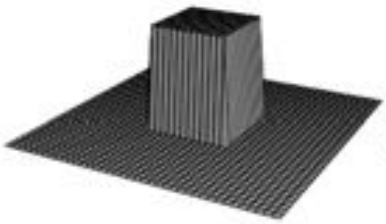
- Used to reduce sharp transitions in intensity
  - Reduce irrelevant detail in an image (e.g., noise)
  - Smooth the false contours that result from using an insufficient number of intensity levels in an image
- Filter kernels:
  - Box filter
  - Lowpass Gaussian filter
  - Order-statistic (nonlinear) filter

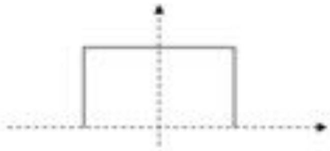
Box filter kernels

An array of 1's

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

Normalizing constant

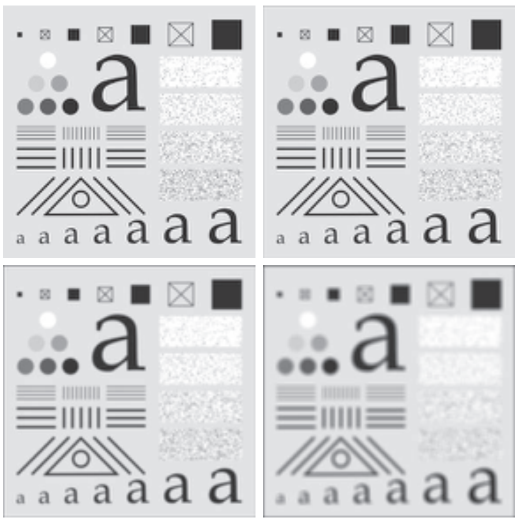




Box filters tend to favor blurring along perpendicular directions

Box filter example

Original image (1024x1024) Kernel size: 3x3



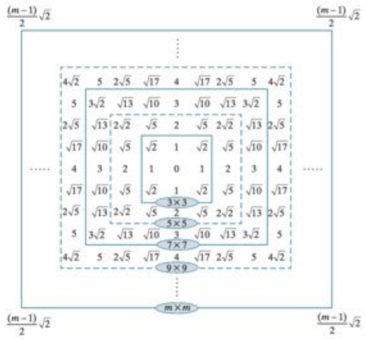
Kernel size: 11x11 Kernel size: 21x21

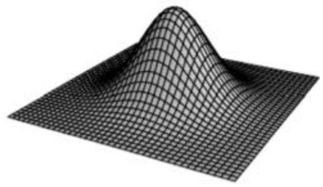
Source: Fig. 3.33, Gonzalez

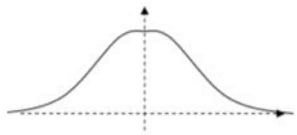
Gaussian filter kernels

When images with a high level of detail, with strong geometrical components

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2+t^2}{2\sigma^2}}$$







Gaussian filter example

Pattern image, 1024x1024

Gaussian filter size 21x21 $\sigma = 3.5$

Gaussian filter size 43x43 $\sigma = 7$

Source: Fig. 3.36. Gonzalez

Box vs Gaussian kernels

Original image

Box kernel, 21x21

Gaussian kernel, 21x21

Significantly less blurring

Note on Gaussian kernels

nothing to be gained by using a Gaussian kernel larger than $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$

→ We get essentially the same result as if we had used an arbitrarily large Gaussian kernels

a b c

FIGURE 3.37 (a) Result of filtering Fig. 3.36(a) using a Gaussian kernels of size 43×43, with $\sigma = 7$. (b) Result of using a kernel of 85×85, with the same value of σ. (c) Difference image.

Shading correction using Gaussian filters

2048x2048 checkerboard image
Inner square size: 128x128

Gaussian kernel
Size: 512x512 (=4x square size)
K=1, $\sigma = 128$ (=1x square size)

Blur-out the squares
to obtain the shading pattern

Dividing original to shading pattern

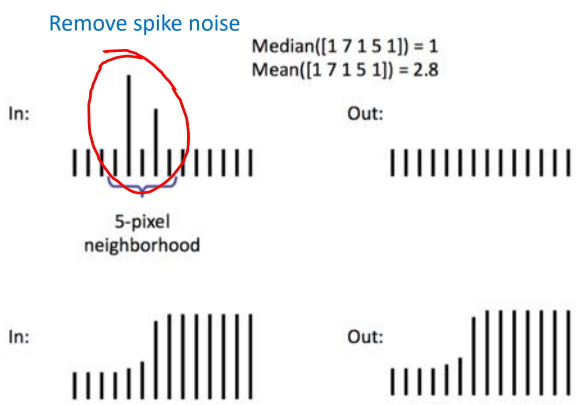
Order-statistic filters

- Nonlinear spatial filter
- Based on ordering (ranking) the pixels contained in the region encompassed by the filter
- Smoothing by replacing the value of the center pixel with the value determined by the ranking result
- Best-known filter:
  - Median filter
- Others:
  - Max filter
  - Min filter

Median filter

- Replaces the value of the center pixel by the median of the intensity values in the neighborhood of that pixel
- Excellent noise reduction:
  - Random noise
  - Impulse noise (salt-and-pepper noise)

Median filter in 1D



Median filter in 2D

10	15	20
23	**90**	27
33	31	30

10	15	20
23	**27**	27
33	31	30

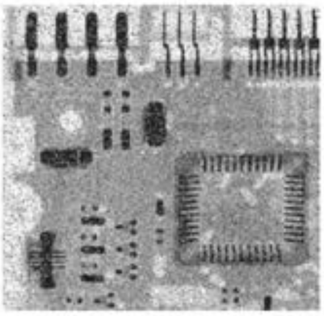
Sort

Replace

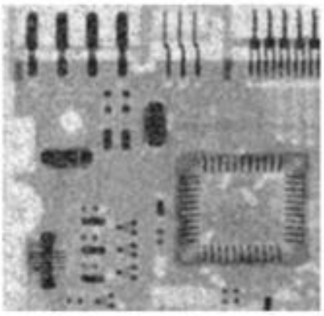
10	15	20	23	$\boxed{27}$	30	31	33	90

Median value

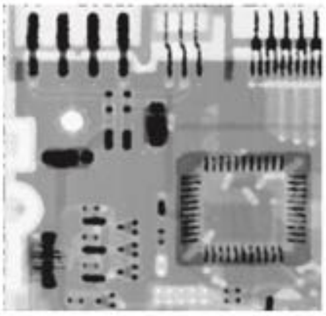
Median filter example



X-ray image of a circuit board



Applied 19x19 Gaussian kernel
$\sigma = 3$



Applied median kernel, 7x7

Source: Fig. 3.43, Gonzalez

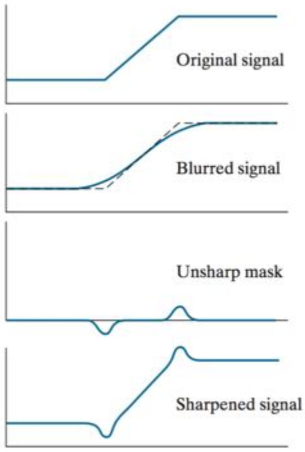
Unsharp masking

(1)
$f(x, y)$

(2)
$\bar{f}(x, y)$

(3) = (1) – (2)
$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$

(4) = (1) + k* (3)
$g(x, y) = f(x, y) + k\, g_{\text{mask}}(x, y)$



k = 1 → unsharp masking
k > 1 → highboost filtering

Unsharp masking example

Original image, 600x259

Gaussian kernel, 31x31, $\sigma = 5$

Mask

Result of unsharp masking

Source: Fig. 3.49, Gonzalez

Spatial filtering with OpenCV

Check the source code