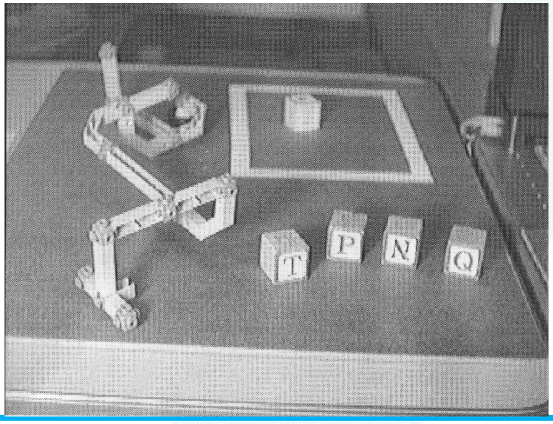
Image Filtering

What is image filtering?

f(x,y) g(x,y)

filtering



filtering

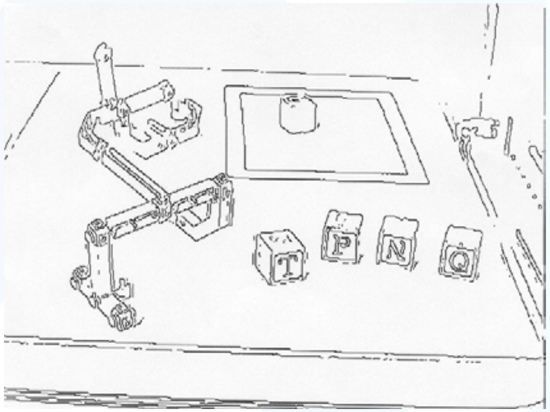
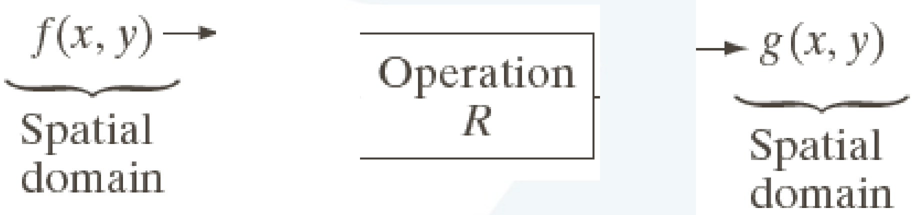
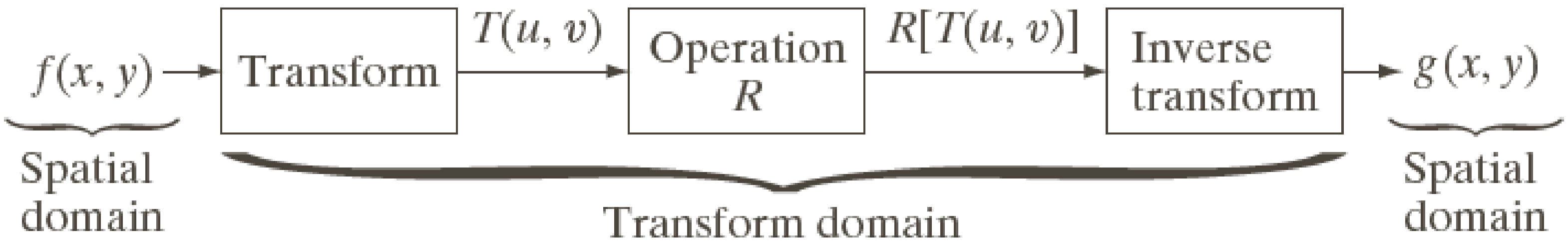


Image Filtering Methods

- Spatial Domain

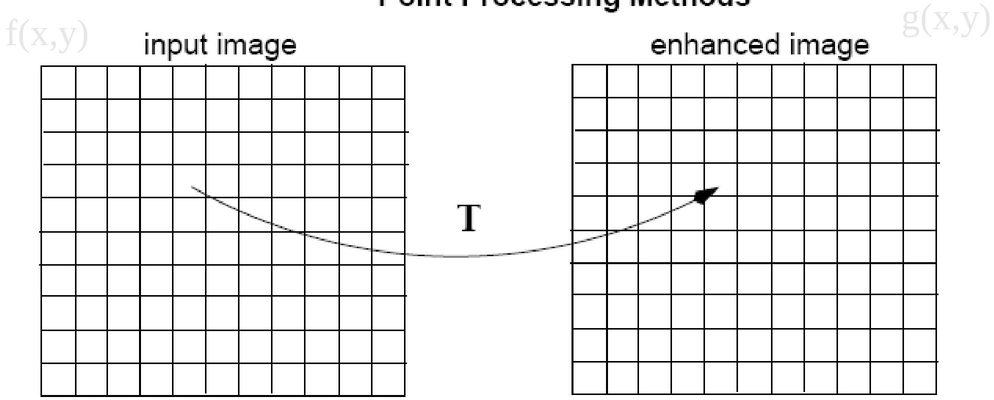


- Frequency Domain (i.e., uses Fourier Transform)



Spatial Domain Methods

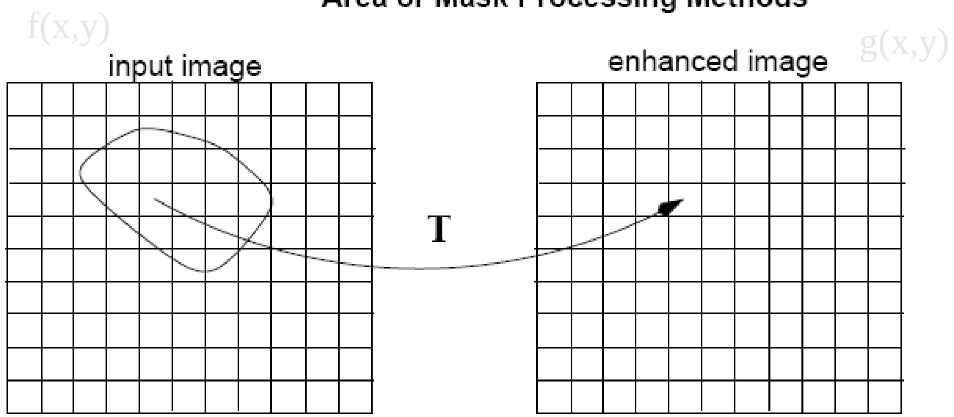
Point Processing Methods



$$g(x,y) = T[f(x,y)]$$

T operates on 1 pixel

Area or Mask Processing Methods

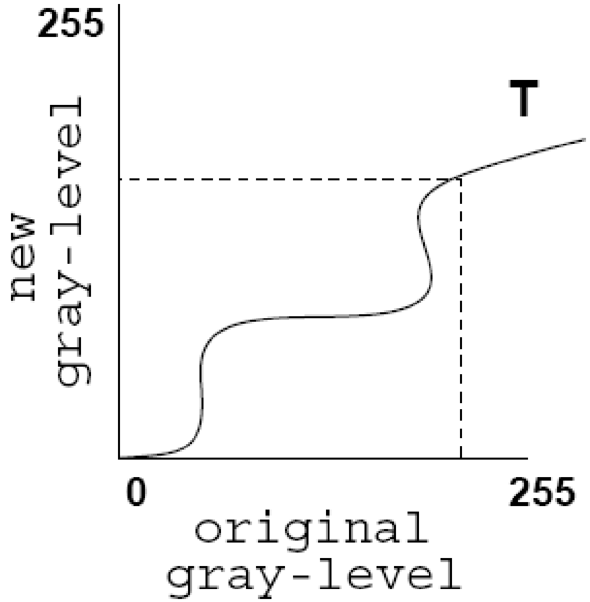


$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

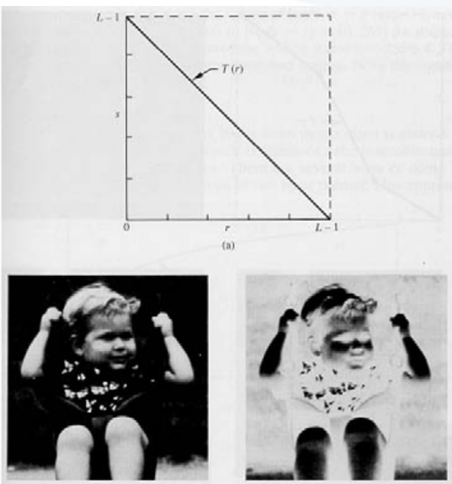
Point Processing Methods

- Convert a given pixel value to a new pixel value based on some predefined function.

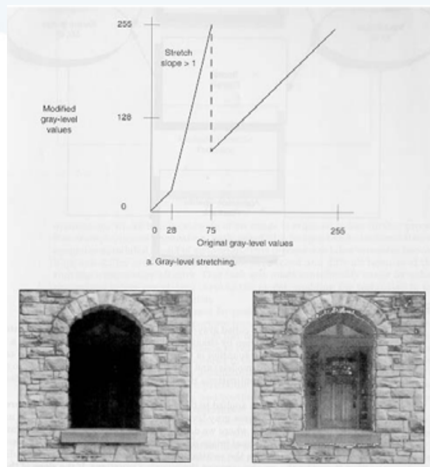


Point Processing Methods - Examples

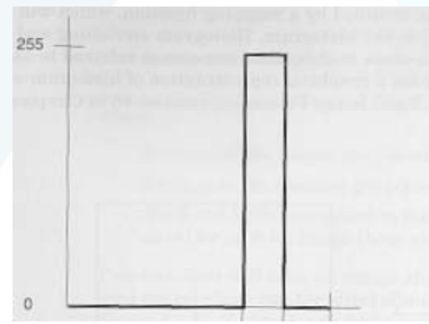
Negative

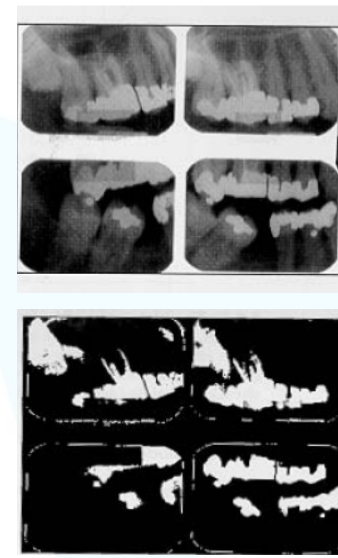


Contrast stretching



Thresholding



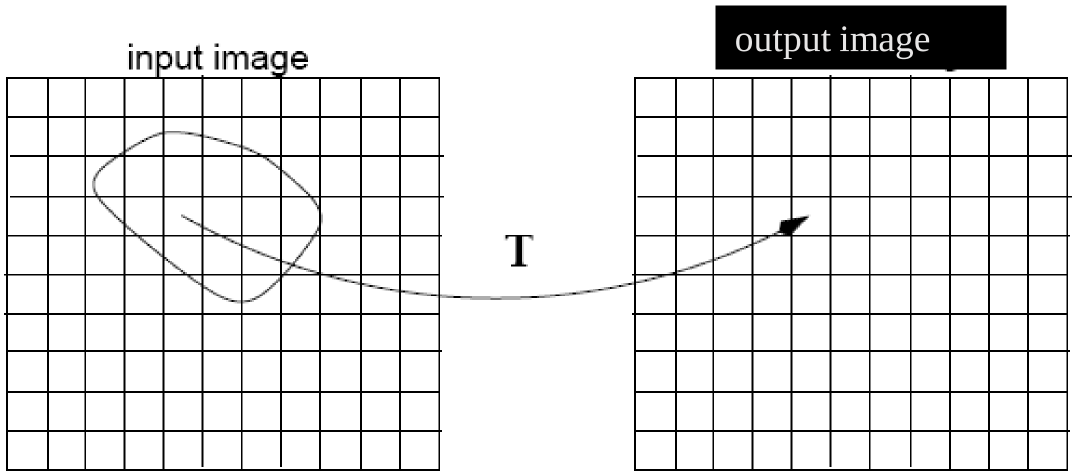


Histogram Equalization

Area Processing Methods

- Need to define:
  - (1) Area shape and size
  - (2) Operation

Area or Mask Processing Methods



$$g(x,y) = T[f(x,y)]$$

T operates on a neighborhood of pixels

Area Shape and Size

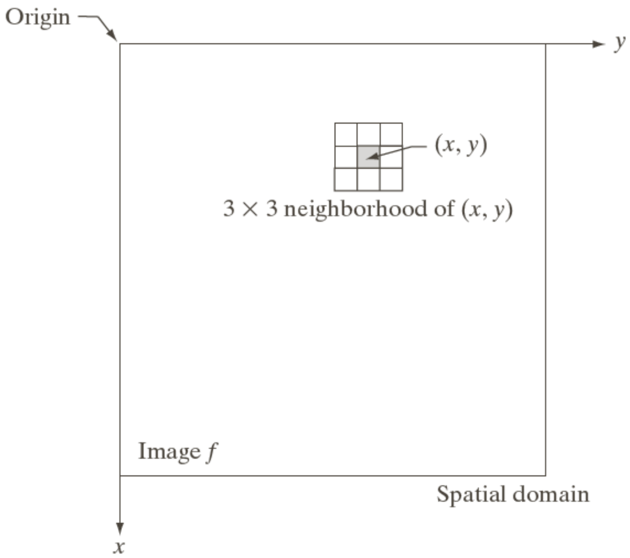
جامعة المنارة
MANARA UNIVERSITY

•

•Area shape is typically defined using a rectangular mask.

- Area size is determined by mask size.
  e.g., 3x3 or 5x5

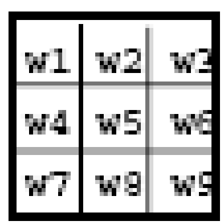
- Mask size is an important parameter!

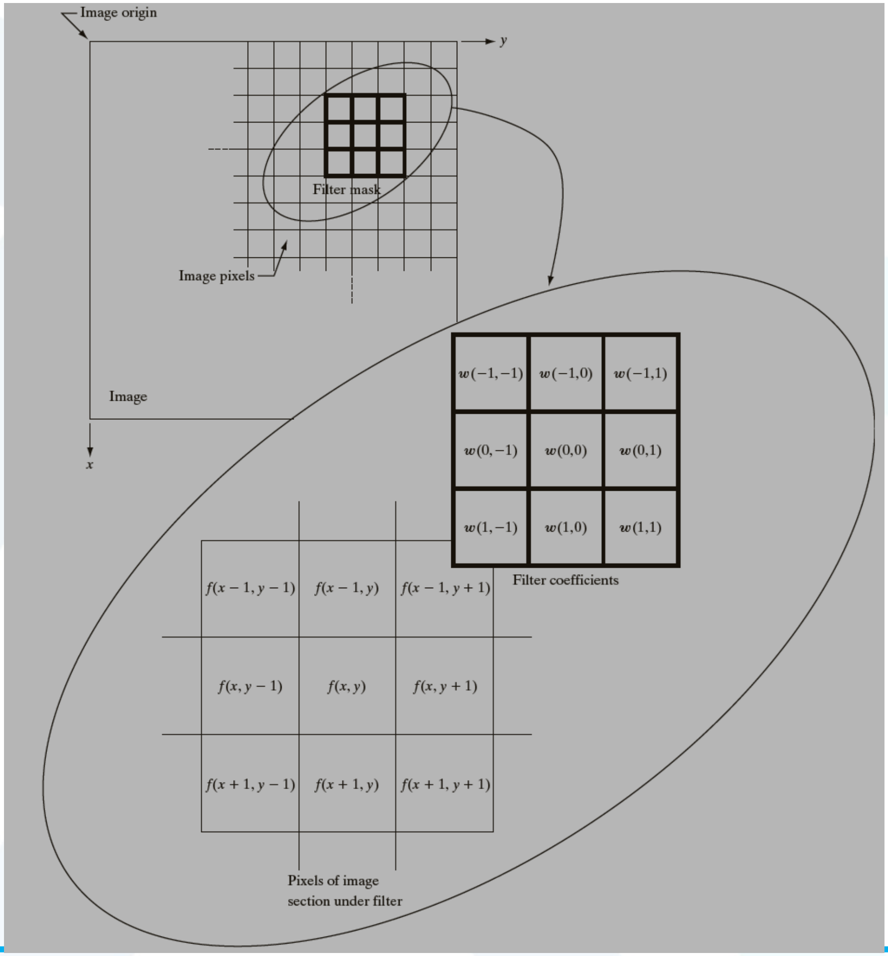


Operation

- Typically linear combinations of pixel values.
  - e.g., weight pixel values and add them together.
- Different results can be obtained using different weights.
  - e.g., smoothing, sharpening, edge detection).

mask





Example

w1	w2	w3
w4	w5	w6
w7	w8	w9

10	5	3
4	6	1
1	1	8

Local image neighborhood

0	0	0
0	0.5	0
0	1	0.5

mask

	8	

Modified image data

Common Linear Operations

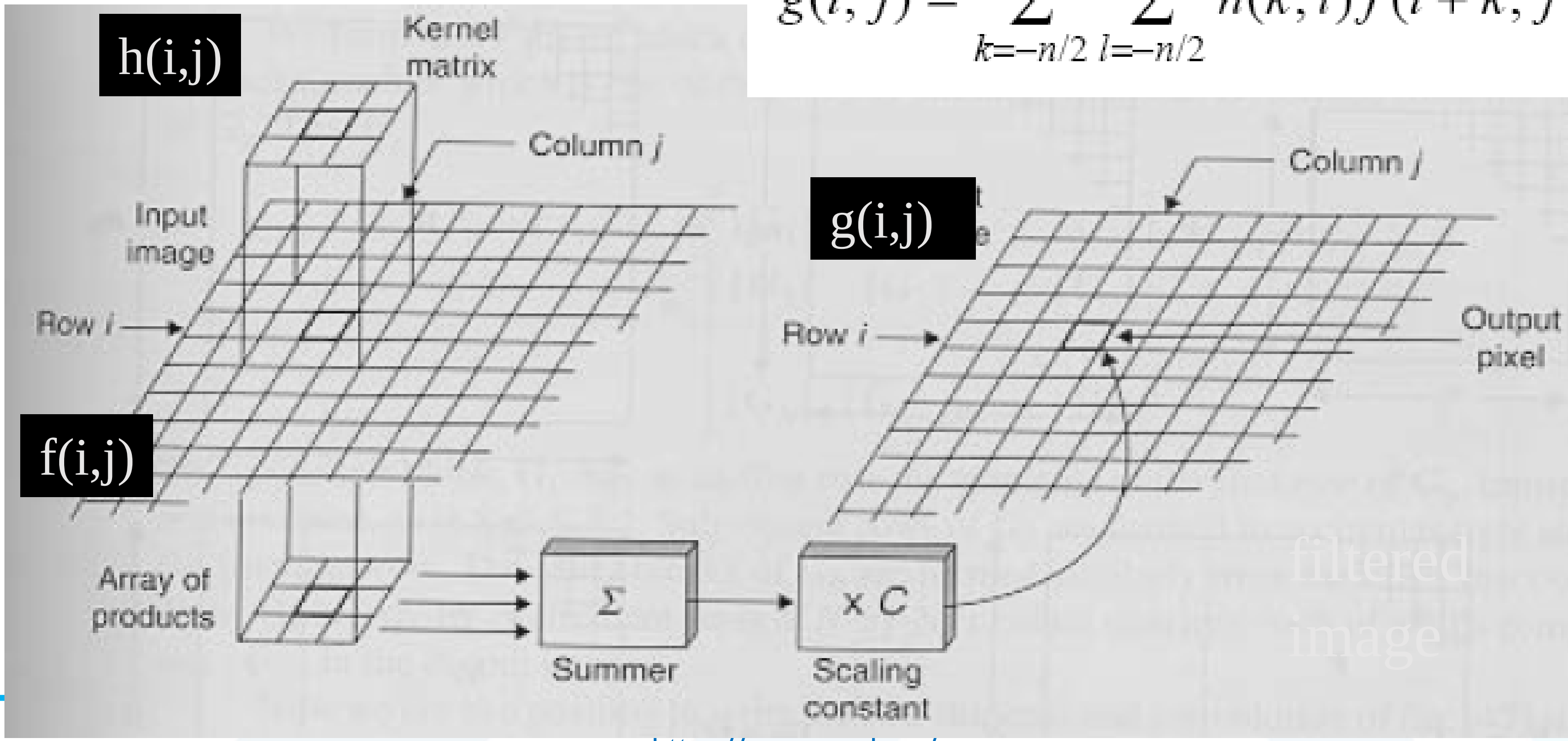
- Correlation
- Convolution

Correlation

- A filtered image is generated as the <u>center</u> of the mask visits every pixel in the input image.

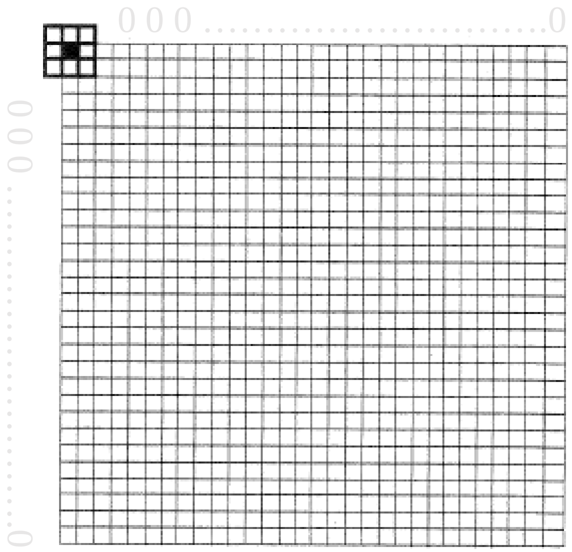
$$g(i,j)=\sum_{k=-n/2}^{n/2}\sum_{l=-n/2}^{n/2} h(k,l)\,f(i+k,j+l)$$

n x n mask



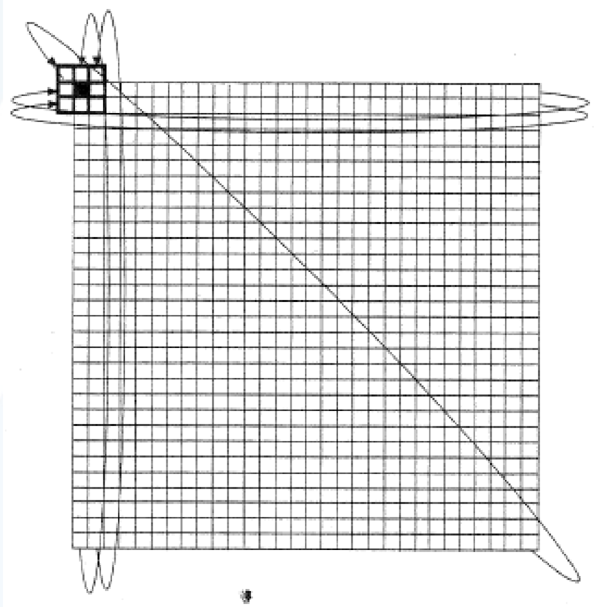
Handling Pixels Close to Boundaries

pad with zeroes



or

wrap around



Correlation – Example

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

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

	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	
[redacted]									

Geometric Interpretation of Correlation

- Suppose x and y are two n-dimensional vectors:

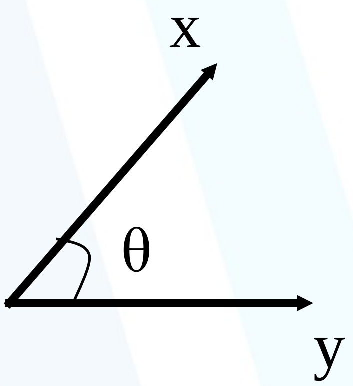
$$x = (x_1, x_2, \ldots, x_n) \qquad y = (y_1, y_2, \ldots, y_n)$$

- The dot product of x with y is defined as:

$$x.y = x_1 y_1 + x_2 y_2 + \ldots + x_n y_n$$

using vector notation:

$$x . y = |x| |y| \cos(\theta)$$



- Correlation generalizes the notion of dot product

Geometric Interpretation of Correlation (cont'd)

cos(θ) measures the similarity between x and y

$$x \cdot y = |x||y|\cos(\theta) \quad or \quad \cos(\theta) = \frac{x\,y}{|x||y|}$$

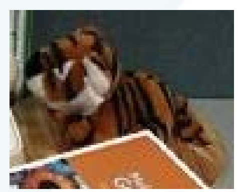
Normalized correlation (i.e., divide by lengths)

$$N(i,j) = \frac{\sum_{k=-\frac{n}{2}}^{\frac{n}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} h(k,l)\, f(i+k, j+l)}{[\sum_{k=-\frac{n}{2}}^{\frac{n}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} h^2(k,l)]^{1/2} [\sum_{k=-\frac{n}{2}}^{\frac{n}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} f^2(i+k, j+l)]^{1/2}}$$

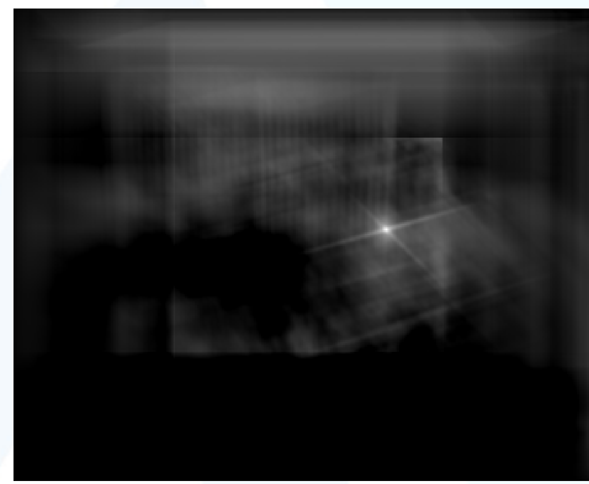
Normalized Correlation

- Measure the similarity between images or parts of images.

mask



=



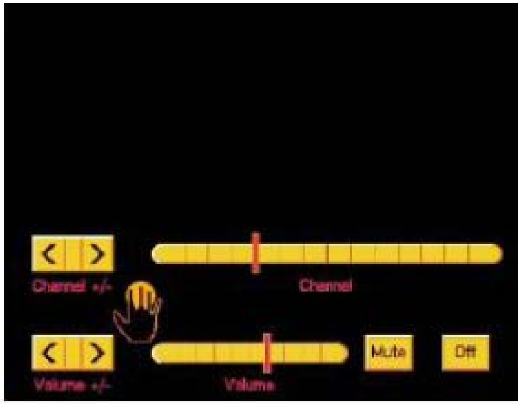
Application: TV Remote Control

Example 1: TV Remote Control

Credit: W. Freeman *et al*, “Computer Vision for Interactive Computer Graphics,” *IEEE Computer Graphics and Applications*, 1998

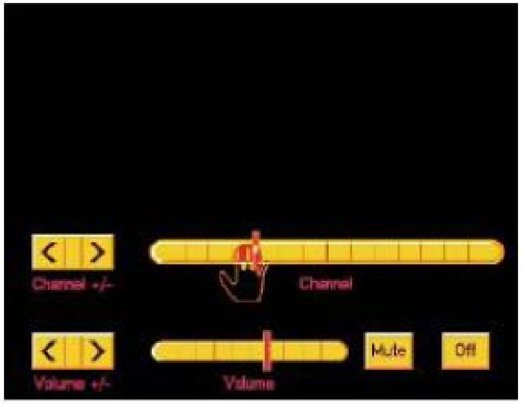
Application : TV Remote Control (cont'd)

Example 1 (cont'd)



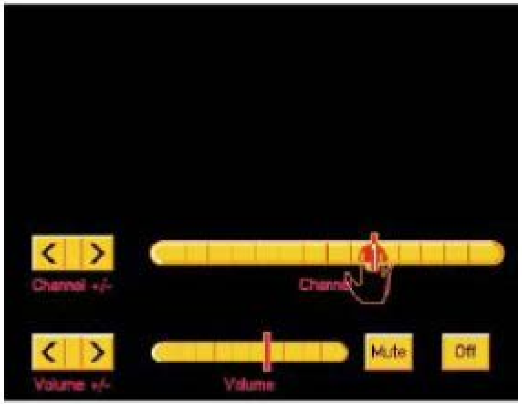
Application : TV Remote Control (cont'd)

Example 1 (cont'd)



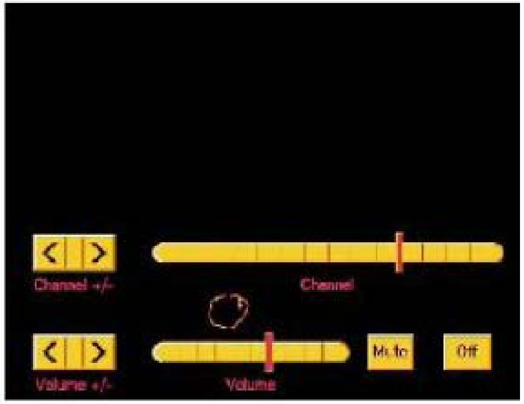
Application : TV Remote Control (cont’d)

Example 1 (cont’d)



Application : TV Remote Control (cont’d)

Example 1 (cont’d)



Application : TV Remote Control (cont’d)

Example 1 (cont’d): Normalized Correlation

Template (left), image (middle), normalized correlation (right)

Note peak value at the true position of the hand

Credit: W. Freeman *et al*, “Computer Vision for Interactive Computer Graphics,” *IEEE Computer Graphics and Applications*, 1998

Normalized Correlation (cont'd)

- Traditional correlation cannot handle changes due to:
  - size
  - orientation
  - shape (e.g., deformable objects).

?

Convolution

- Same as correlation except that the mask is **flipped,** both horizontally and vertically.

1	2	3
4	5	6
7	8	9

H →

7	8	9
4	5	6
1	2	3

V →

9	8	7
6	5	4
3	2	1

$$g(i,j) = \sum_{k=-\frac{n}{2}}^{\frac{n}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} h(k,l)\, f(i-k, j-l) = \sum_{k=-\frac{n}{2}}^{\frac{n}{2}} \sum_{l=-\frac{n}{2}}^{\frac{n}{2}} h(i-k, j-l)\, f(k,l)$$

For symmetric masks (i.e., $h(i,j)=h(-i,-j)$), convolution is equivalent to correlation!

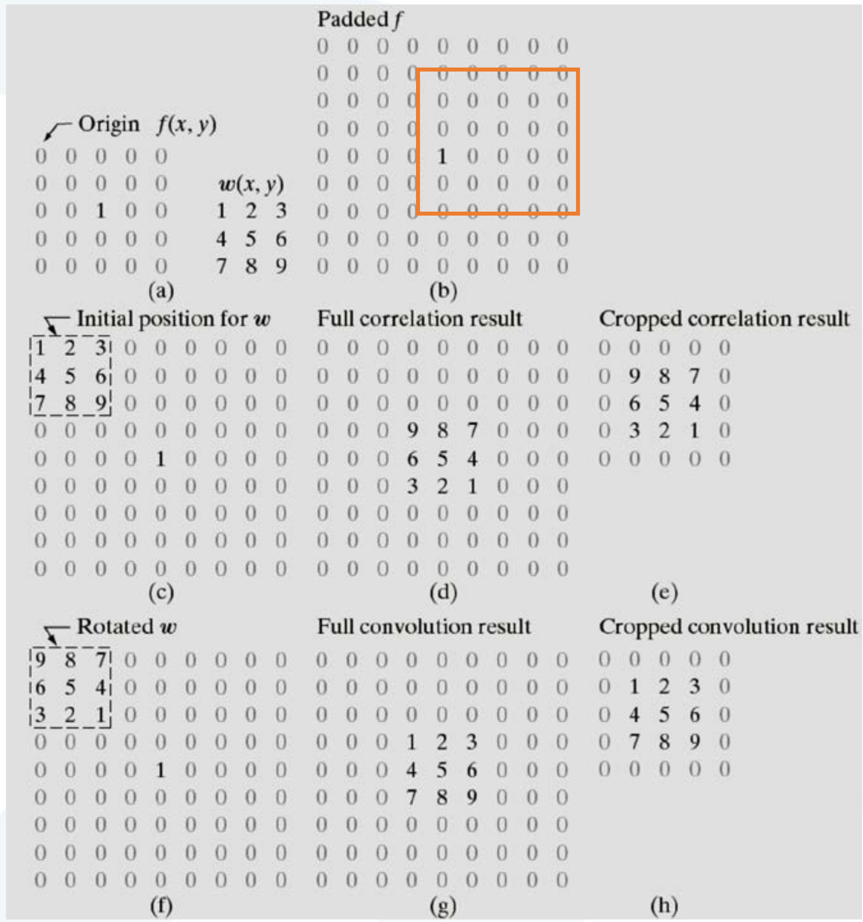
Notation:

$$h * f = f * h$$

Correlation/Convolution Examples

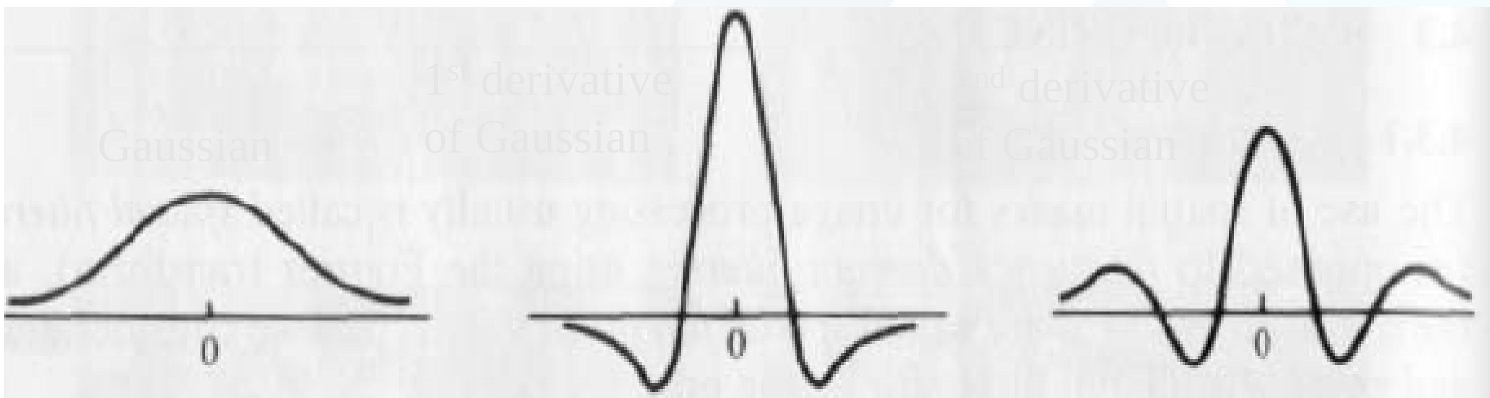
Correlation:

Convolution:



How do we choose the mask weights?

- Depends on the application.
- Usually by sampling certain functions and their derivatives.



Good for
image smoothing

Good for
image sharpening

Normalization of Mask Weights

- Sum of weights affects overall intensity of output image.
- Positive weights
  - Normalize them such that they sum to **one**.
- Both positive and negative weights
  - Should sum to **zero** (but not always)

w1	w2	w3
w4	w5	w6
w7	w8	w9

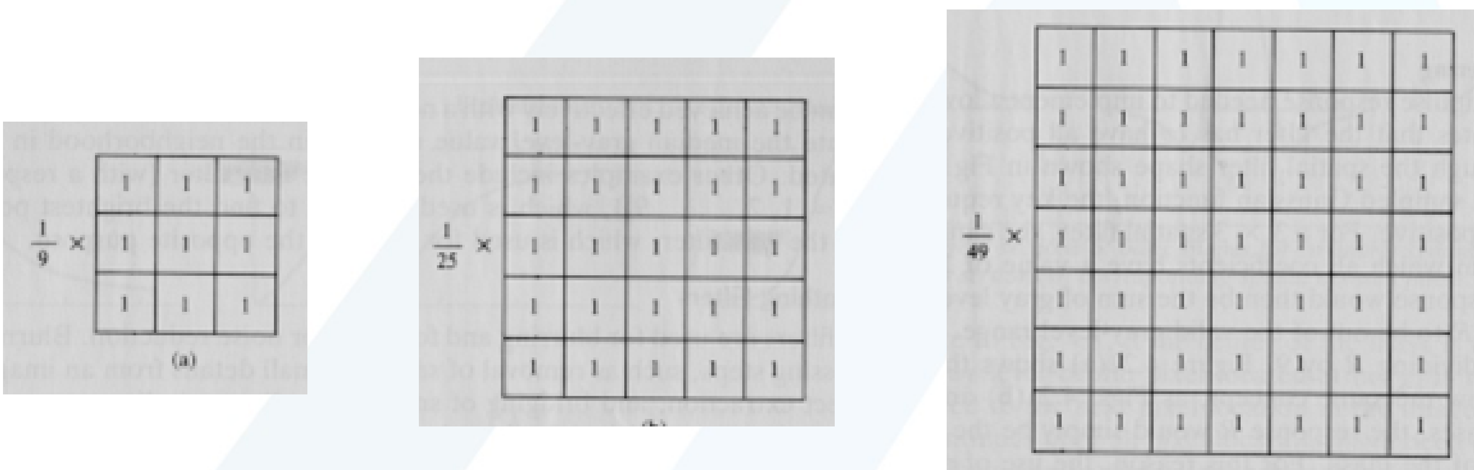
$$\frac{1}{9}\begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \qquad \frac{1}{16}\begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$

Smoothing Using Averaging

- **Idea:** replace each pixel by the average of its neighbors.

- Useful for reducing noise and unimportant details.

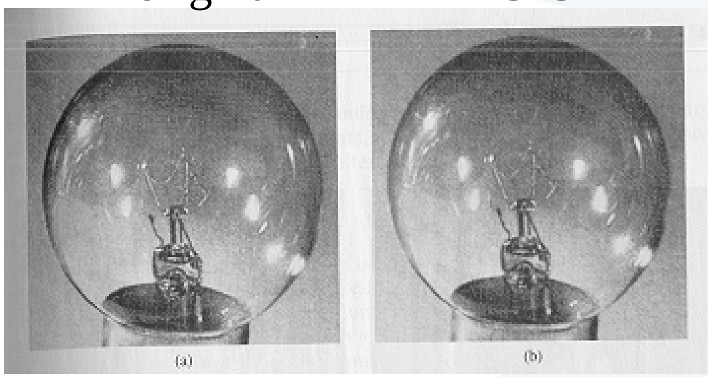
- The size of the mask controls the amount of smoothing.



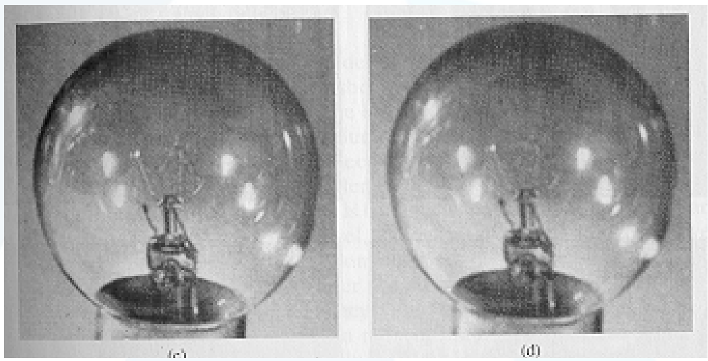
Smoothing Using Averaging (cont'd)

- **Trade-off:** noise vs blurring and loss of detail.

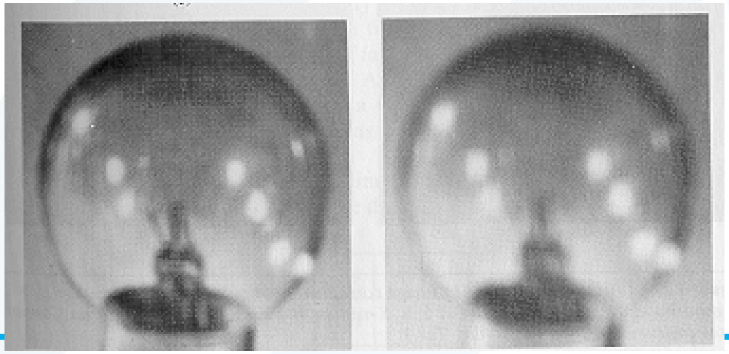
original 3x3



5x5 7x7



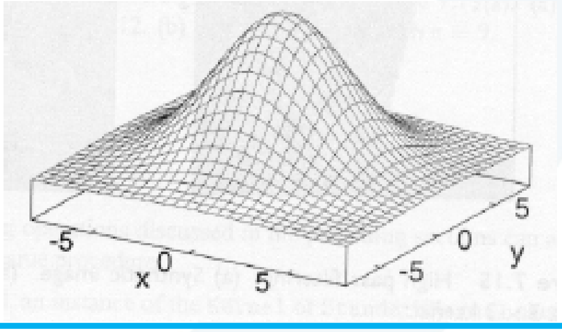
15x15 25x25



Gaussian Smoothing

- **Idea:** replace each pixel by a weighted average of its neighbors
- Mask weights are computed by sampling a Gaussian function

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



7 × 7 Gaussian mask

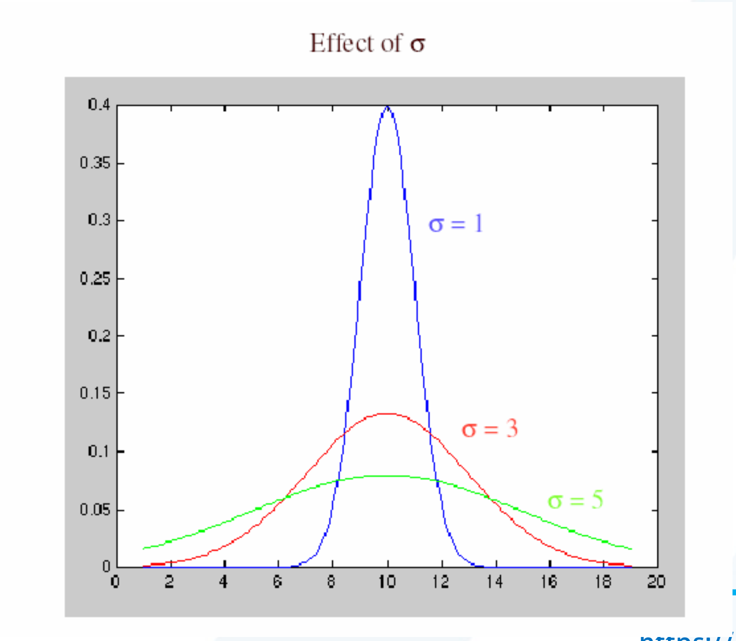
1	1	2	2	2	1	1
1	2	2	4	2	2	1
2	2	4	8	4	2	2
2	4	8	16	8	4	2
2	2	4	8	4	2	2
1	2	2	4	2	2	1
1	1	2	2	2	1	1

Note: weight values decrease with distance from mask center!

Gaussian Smoothing (cont'd)

mask size depends on **σ** : *height* = *width* = 5σ (subtends 98.76% of the area)

- **σ** determines the degree of smoothing!



σ=3

15 × 15 Gaussian mask

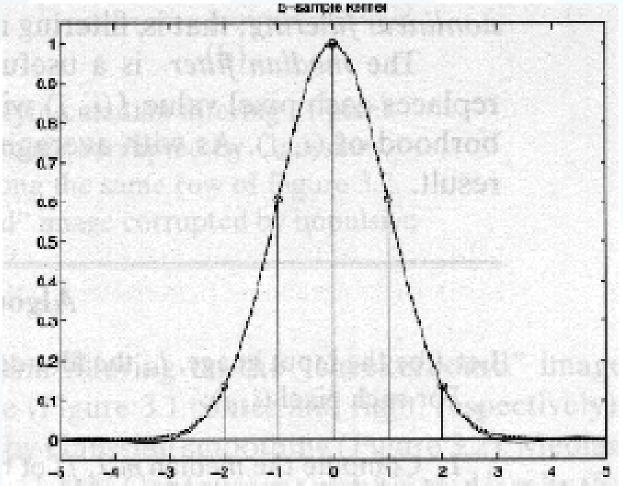
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
6	8	11	13	16	18	19	20	19	18	16	13	11	8	6
6	8	10	13	15	17	19	19	19	17	15	13	10	8	6
5	7	10	12	14	16	17	18	17	16	14	12	10	7	5
5	7	9	11	13	14	15	16	15	14	13	11	9	7	5
4	5	7	9	10	12	13	13	13	12	10	9	7	5	4
3	4	6	7	9	10	10	11	10	10	9	7	6	4	3
2	3	4	5	7	7	8	8	8	7	7	5	4	3	2
2	2	3	4	5	5	6	6	6	5	5	4	3	2	2

Gaussian Smoothing (cont'd)

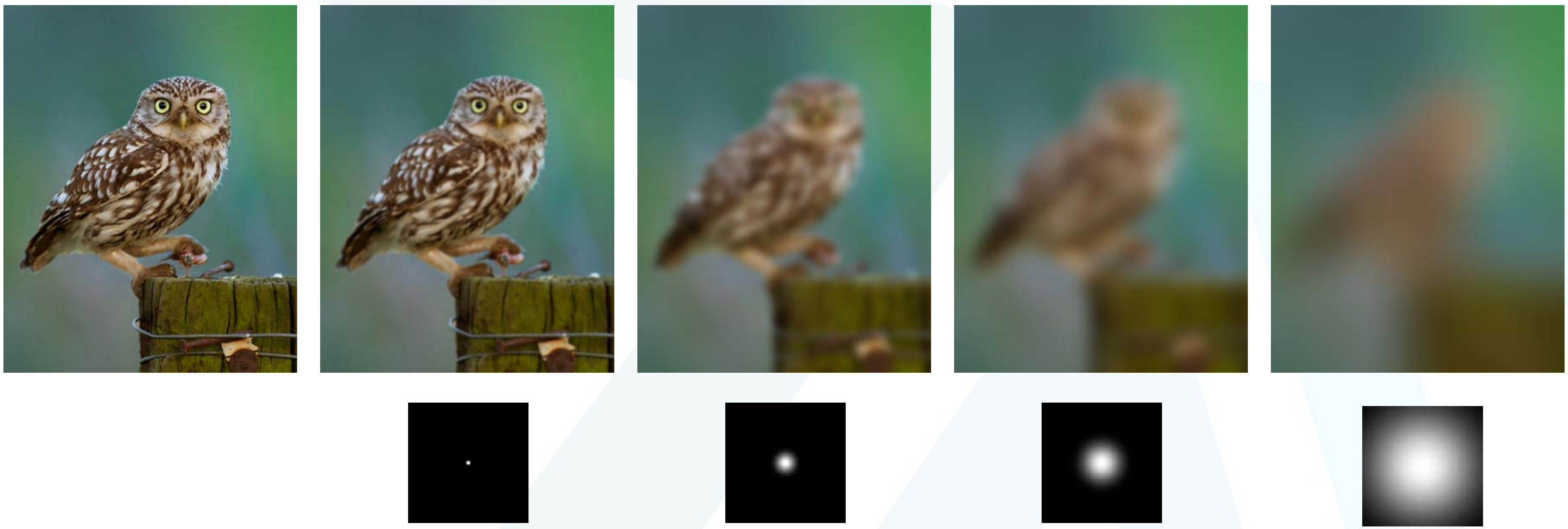
```
Gaussian(sigma, hSize, h)
float sigma, *h;
int hSize;
{
  int    i;
  float  cst, tssq, x, sum;

  cst = 1./(sigma*sqrt(2.0*PI)) ;
  tssq = 1./(2*sigma*sigma) ;

  for(i=0; i<hSize; i++) {
    x=(float)(i-hSize/2);
    h[i]=(cst*exp(-(x*x*tssq))) ;
  }
  // normalize
  sum=0.0;
  for(i=0;i<hSize;i++)
    sum += h[i];
  for(i=0;i<hSize;i++)
    h[i] /= sum;
}
```

[redacted]

```
halfSize=(int)(2.5*sigma);
hSize=2*halfSize;
if (hSize % 2 == 0) ++hSize;  // odd size
```



Gaussian Smoothing - Example



σ = 1 pixel | σ = 5 pixels | σ = 10 pixels | σ = 30 pixels

Averaging vs Gaussian Smoothing

Averaging

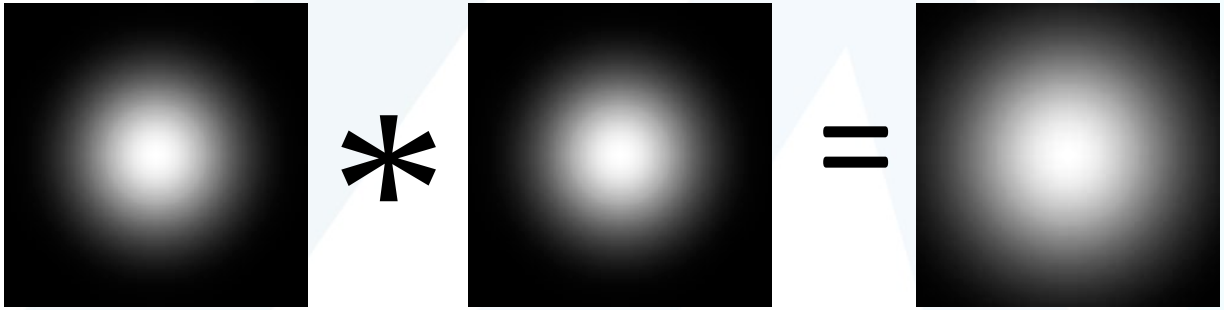
Gaussian

Properties of Gaussian

- Convolution with self is another Gaussian

$$G_{\sigma_1}(x) * G_{\sigma_2}(x) \equiv G_{\sqrt{\sigma_1^2 + \sigma_2^2}}(x)$$

- **Special case:** convolving two times with Gaussian kernel of width σ is equivalent to convolving once with kernel of width $\sigma\sqrt{2}$



Properties of Gaussian (cont'd)

- *Separable* kernel: a 2D Gaussian can be expressed as the product of two 1D Gaussians.

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

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

Properties of Gaussian (cont'd)

- 2D Gaussian convolution can be implemented more efficiently using 1D convolutions:

$$g(i,j) = \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} h(k,l) f(i-k, j-l) =$$

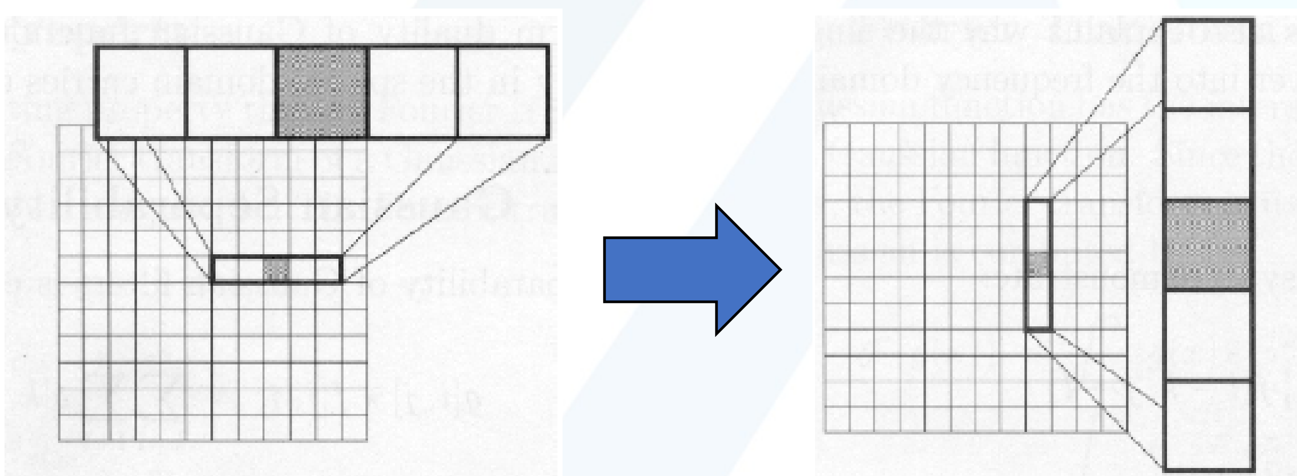
$$\sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} exp[\frac{-(k^2 + l^2)}{2\sigma^2}] f(i-k, j-l) =$$

$$\sum_{k=-n/2}^{n/2} exp[\frac{-k^2}{2\sigma^2}] \sum_{l=-n/2}^{n/2} exp[\frac{-l^2}{2\sigma^2}] f(i-k, j-l)$$

Properties of Gaussian (cont'd)

To convolve an image I with a nxn 2D Gaussian mask G with $\sigma = \sigma_g$

1. Build a 1-D Gaussian mask g, of width n, with σ_g [redacted]
2. Convolve each row of I with g, get a new image I_r
3. Convolve each column of I_r with g



Example

2D convolution
(center location only)

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix}$$

=2 + 6 + 3 = 11
= 6 + 20 + 10 = 36
= 4 + 8 + 6 = 18

65

$O(n^2)$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \times \begin{bmatrix}1&2&1\end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix}1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix} = \begin{bmatrix}&11&\\&18&\\&18&\end{bmatrix}$$

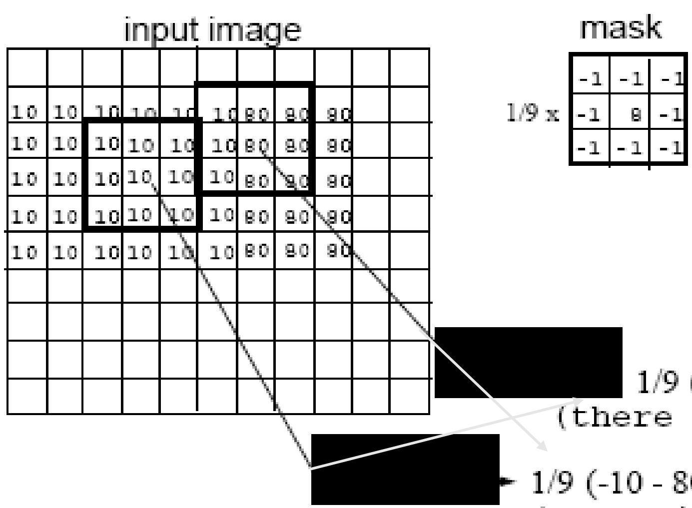
$O(2n)=O(n)$

Followed by convolution
along the remaining column:

$$\begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}&11&\\&18&\\&18&\end{bmatrix} = \begin{bmatrix}&&\\&65&\\&&\end{bmatrix}$$

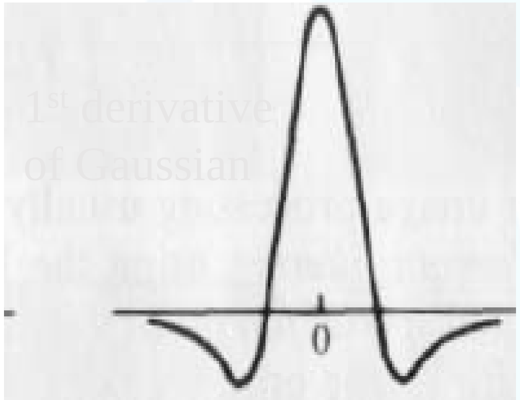
Image Sharpening

- <u>Idea:</u> compute intensity differences in local image regions.
- Useful for emphasizing transitions in intensity (e.g., in edge detection).

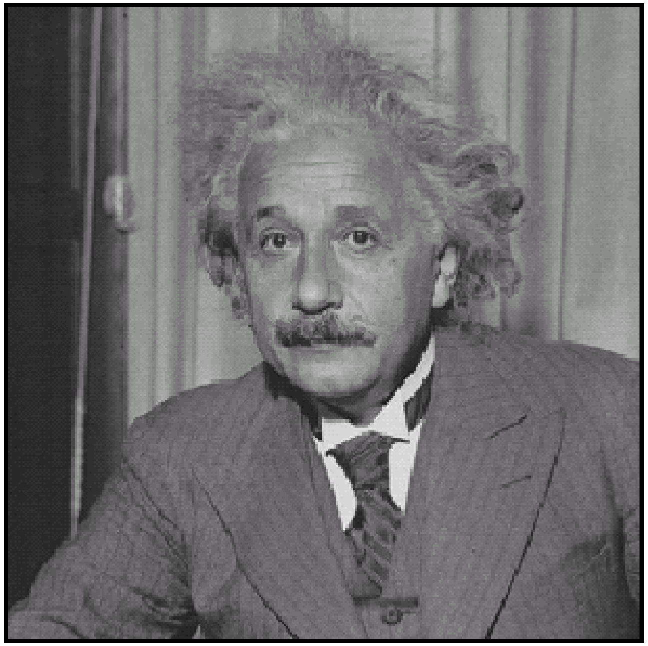


1/9 (-10 - 10 - 10 -10 + 80 -10 -10 -10 -10) = 0
(there is no variation in the gray-levels)

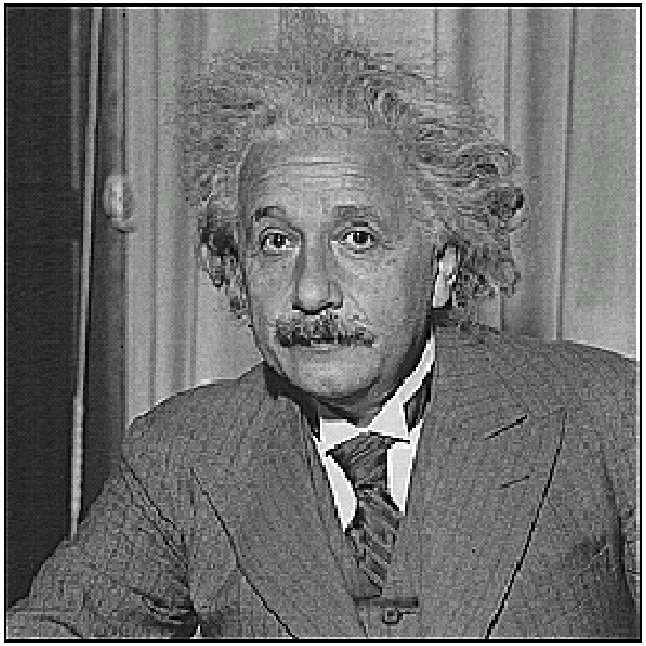
1/9 (-10 - 80 - 80 -10 + 640 -80 -10 -80 -80) = 210/9 > 0
(there is variation in the gray-levels)



Example



before



after