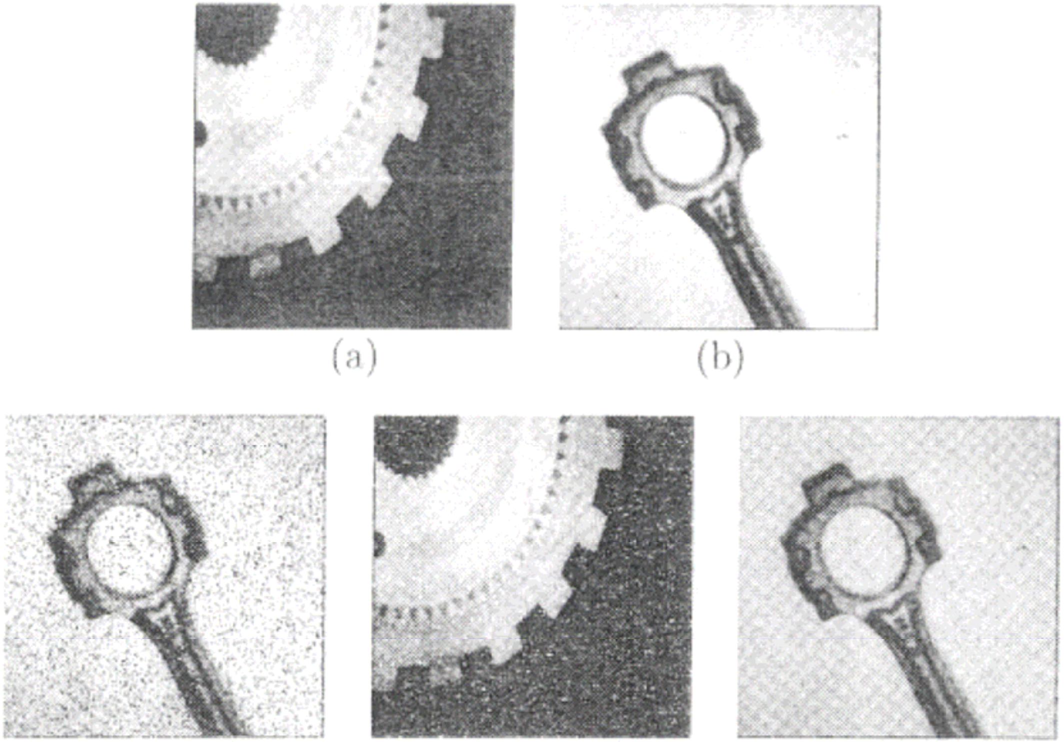
Visione artificiale
(a.a. 2006/07)

Image Filtering

Noise

- Salt and Pepper Noise
  - random occurrences of black and white pixels
- Impulse noise
  - Random occurrences of white pixels only
- Gaussian noise
  - Variations of intensity that are drawn from a Gaussian or normal distribution



(a) (b)

(c) (d) (e)

Figure 4.5: Examples of images corrupted by salt and pepper, impulse, and Gaussian noise. (a) & (b) Original images. (c) Salt and pepper noise. (d) Impulse noise. (e) Gaussian noise.

Convolution in 1-D

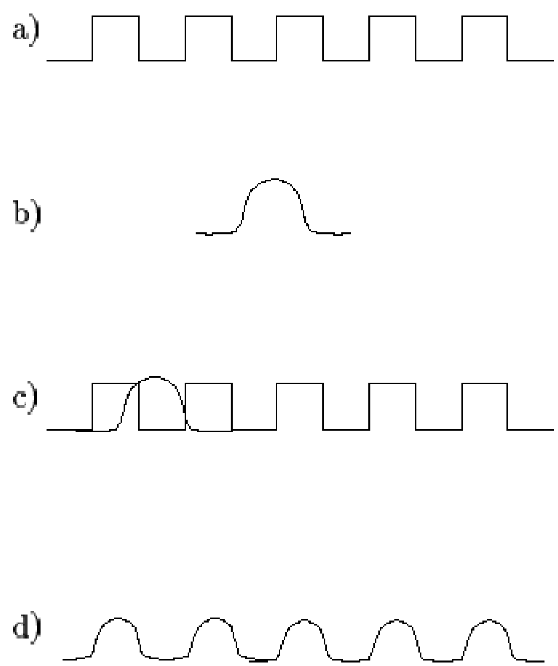


Figure 4: Illustration of one-dimensional convolution (see the text).

$$g(x) = \int_{\infty}^{\infty} f(x - \xi)h(\xi)d\xi$$

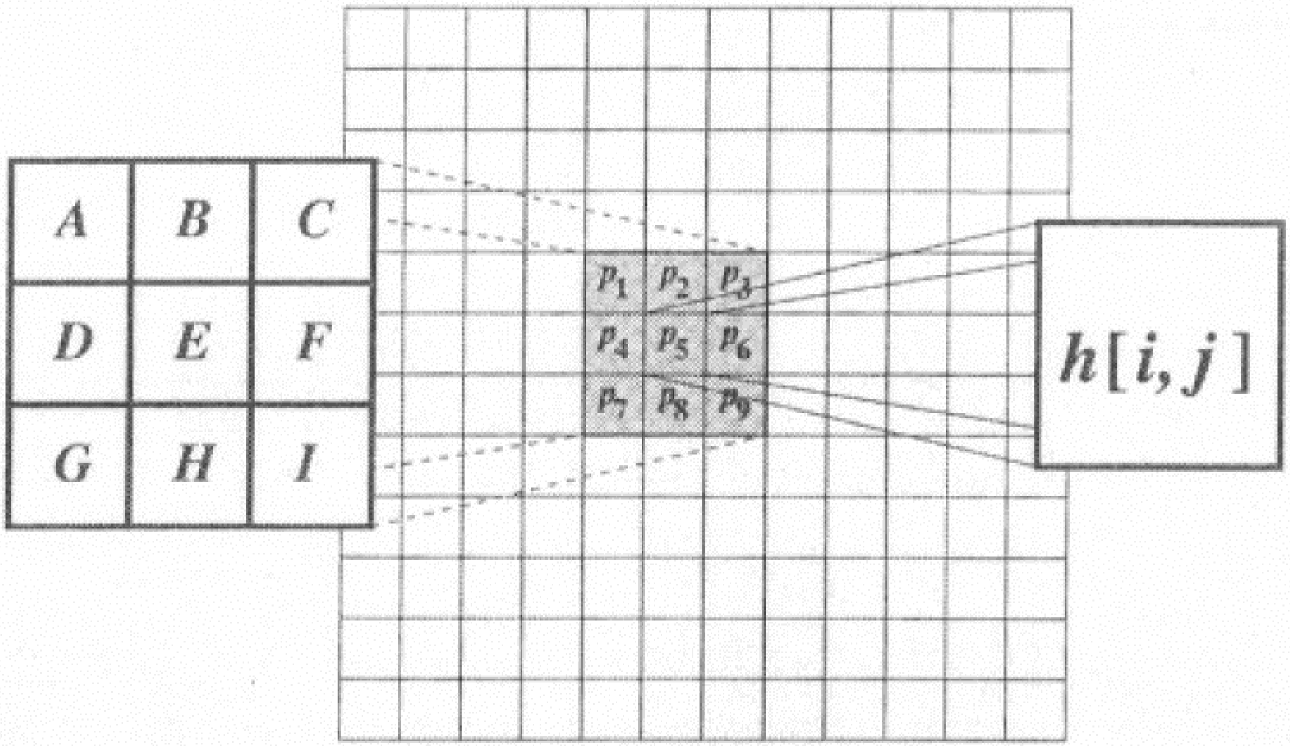
Convolution in 2-D

$$h(x,y) = f(x,y) \star g(x,y)$$
$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x',y')\, g(x-x', y-y')\, dx'\, dy'.$$

$$h[i,j] = f[i,j] \star g[i,j]$$
$$= \sum_{k=1}^{n}\sum_{l=1}^{m} f[k,l]\, g[i-k, j-l].$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

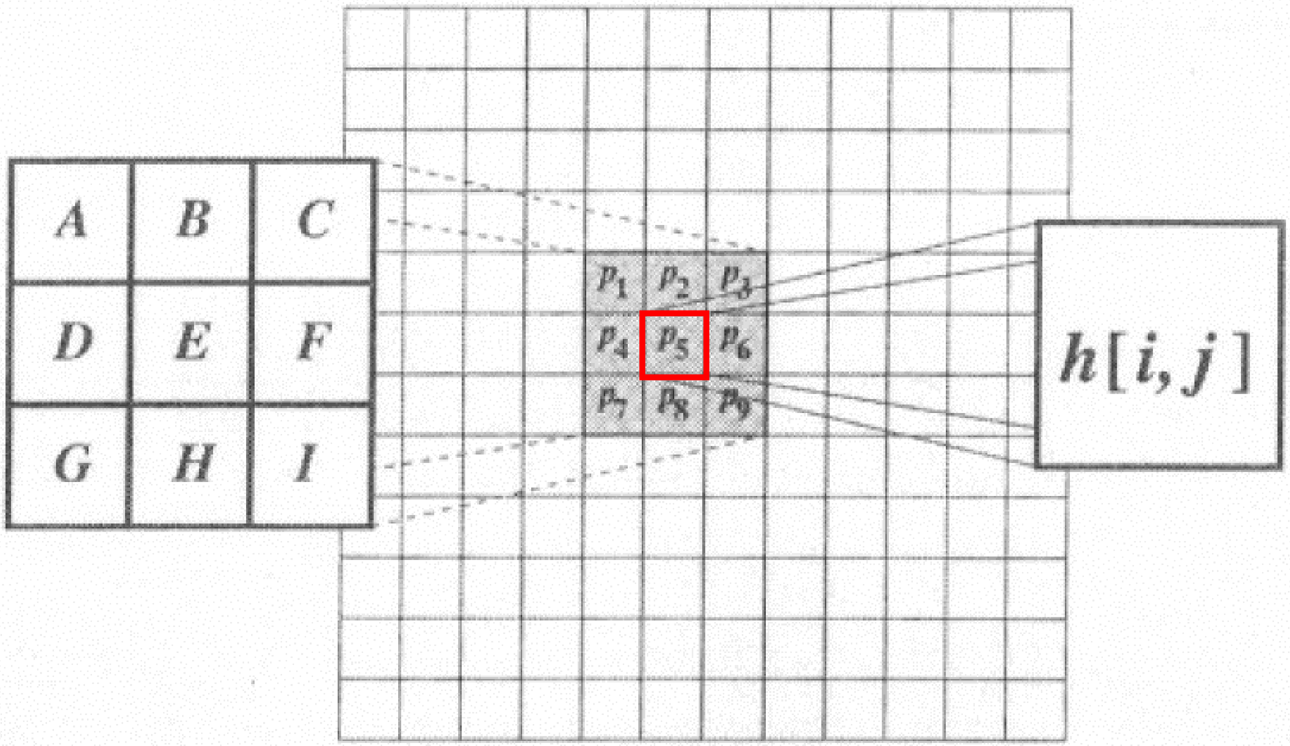
Example of 3x3 convolution mask



$$h[i,j] = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Example of 3x3 convolution mask



$$h[i,j] = A\,p_1 + B\,p_2 + C\,p_3 + D\,p_4 + E\,p_5 + F\,p_6 + G\,p_7 + H\,p_8 + I\,p_9$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Properties of Convolution

Convolution is *commutative*, which can be seen by a simple substitution, $\alpha = x - \xi$, $\beta = y - \eta$ then rename α to ξ and β to η,

$$a \otimes b = b \otimes a \ .$$

Convolution is also associative

$$(a \otimes b) \otimes c = a \otimes (b \otimes c) \ .$$

These two properties are very useful because they allow us to rearrange computations in whatever fashion is most convenient (or efficient).

Linear Shift Invariant Systems

Convolutions are equivalent to linear shift invariant systems (LSI) — a topic central to much of signal processing which we will only touch on here. Say you are given a black box h, such that when the function f_1 is input to the box the function g_1 is output, and when the function f_2 is input, the function g_2 is output,

$$f_1 \longrightarrow \boxed{\text{h}} \longrightarrow g_1$$

$$f_2 \longrightarrow \boxed{\text{h}} \longrightarrow g_2$$

We say that h is linear shift invariant (or LSI) when it obeys linearity,

$$\alpha f_1 + \beta f_2 \longrightarrow \boxed{\text{h}} \longrightarrow \alpha g_1 + \beta g_2 \text{ for any } \alpha, \beta$$

and it is shift invariant

$$f_1(x-a, y-b) \longrightarrow \boxed{\text{h}} \longrightarrow g_1(x-a, y-b) \text{ for any } a, b\,.$$

Mean Filters

- Arbitrary neighborhood

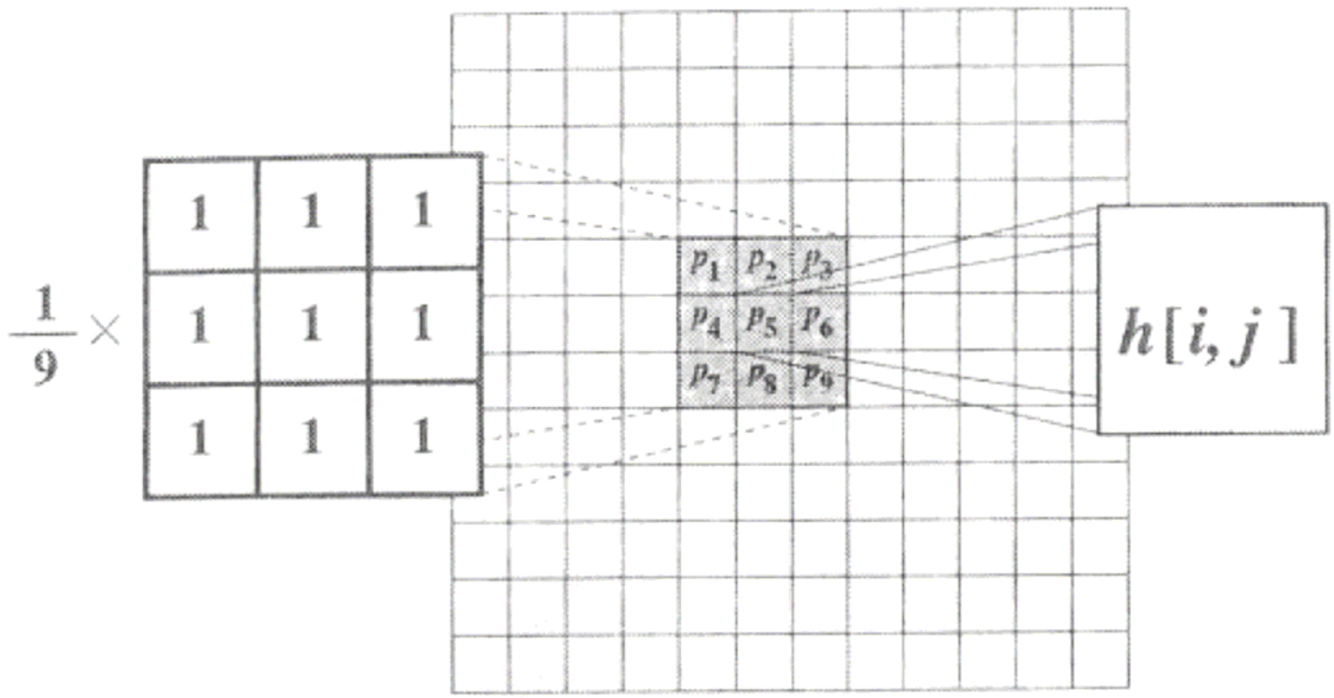
$$h[i,j] = \frac{1}{M} \sum_{(k,l)\in N} f[k,l]$$

- For a 3x3 neighborhood

$$h[i,j] = \frac{1}{9} \sum_{k=i-1}^{i+1} \sum_{k=j-1}^{j+1} f[k,l].$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

3x3 Mean Filter



[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

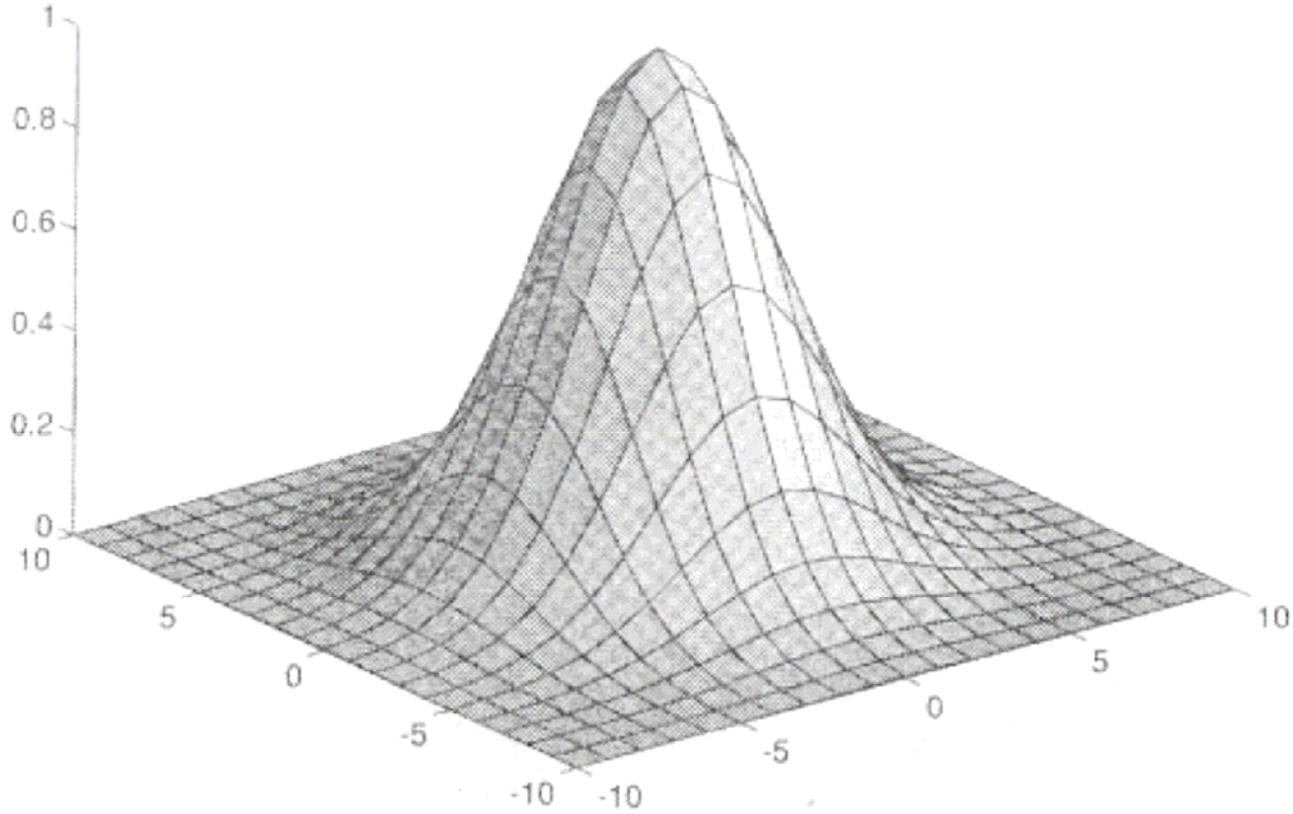
3x3 Linear Smoothing Filter

In general, it is a good idea to have only a single peak in your smoothing filter:

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Gaussian Smoothing



[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

The Gaussian Function

- Zero mean 1D Gaussian

$$g(x) = e^{-\frac{x^2}{2\sigma^2}},$$

- Zero mean 2D gaussian for image processing applications

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}},$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Gaussian Properties

- Rotationally symmetric in 2D
- Has a single peak
- The width of the filter and the degree of smoothing are determined by sigma
- Large Gaussian filters can be implemented very efficiently using small Gaussian filters

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Rotational Symmetry

- Original formula

$$g[i,j] = e^{-\frac{(i^2+j^2)}{2\sigma^2}}.$$

- Switch to polar coordinates

- Result

$$r^2 = i^2 + j^2$$

$$g(r,\theta) = e^{-\frac{r^2}{2\sigma^2}},$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Gaussian Separability

$$
\begin{aligned}
g[i,j] \star f[i,j] &= \sum_{k=1}^{m} \sum_{l=1}^{n} g[k,l]\, f[i-k, j-l] \\
&= \sum_{k=1}^{m} \sum_{l=1}^{n} e^{-\frac{(k^2+l^2)}{2\sigma^2}} f[i-k, j-l] \\
&= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left\{ \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f[i-k, j-l] \right\}.
\end{aligned}
$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Gaussian Separability

$$g[i,j] \star f[i,j] = \sum_{k=1}^{m} \sum_{l=1}^{n} g[k,l]\, f[i-k, j-l]$$

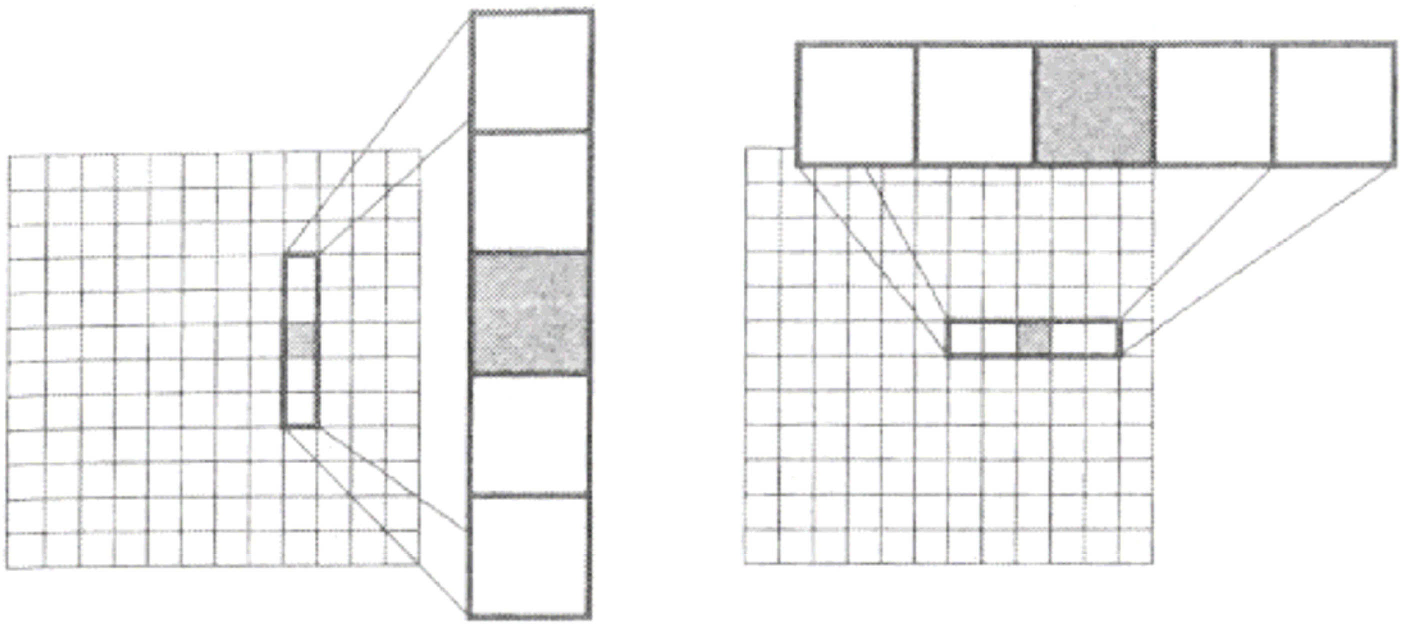
$$= \sum_{k=1}^{m} \sum_{l=1}^{n} e^{-\frac{(k^2+l^2)}{2\sigma^2}} f[i-k, j-l]$$

$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left\{ \sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f[i-k, j-l] \right\}.$$

The convolution of the input image f[i,j] with a vertical 1D Gaussian function

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Cascading Gaussians



[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

The convolution of a Gaussian with itself yields a scaled Gaussian with larger sigma

$$
\begin{aligned}
g(x) \star g(x) &= \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} \, d\xi \\
&= \int_{-\infty}^{\infty} e^{-\frac{(\frac{x}{2}+\xi)^2}{2\sigma^2}} e^{-\frac{(\frac{x}{2}-\xi)^2}{2\sigma^2}} \, d\xi, \quad \xi \rightarrow \xi + \frac{x}{2} \\
&= \int_{-\infty}^{\infty} e^{-\frac{(2\xi^2+\frac{x^2}{2})}{2\sigma^2}} \, d\xi \\
&= e^{-\frac{x^2}{4\sigma^2}} \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{\sigma^2}} \, d\xi \\
&= \sqrt{\pi}\sigma e^{-\frac{x^2}{2(\sqrt{2}\sigma)^2}}.
\end{aligned}
$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Properties

The product of the convolution of two Gaussian functions with a spread σ is a Gaussian function with a spread $\sqrt{2}\sigma$ scaled by the area of the Gaussian filter

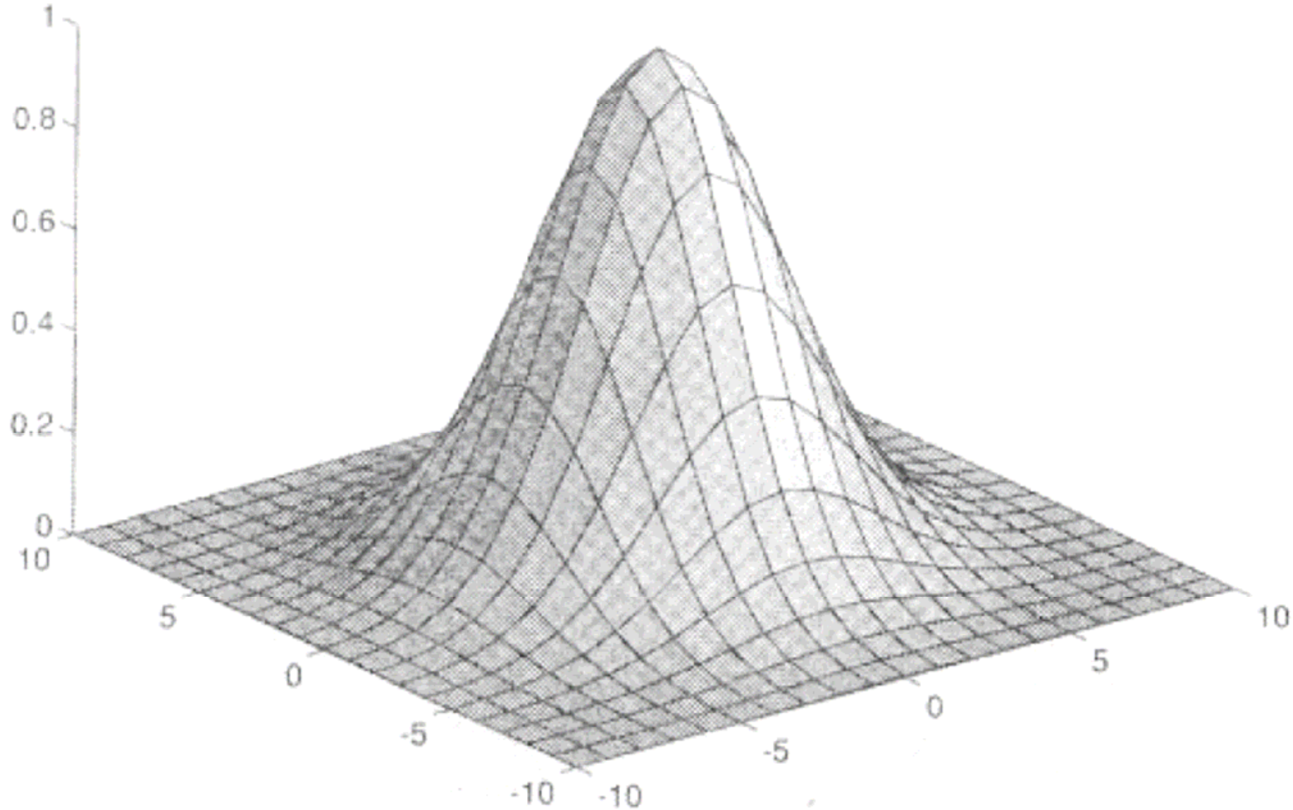
[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Properties of Discrete Gaussian Filters

- Step 1: smooth with n x n discrete Gaussian Filter
- Step 2: smooth the intermediary result from Step 1 with m x m discrete Gaussian Filter
- Step 1 + Step 2 are equivalent to smoothing the original with (n+m-1)x(n+m-1) discrete Gaussian Filter

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Designing Gaussian Filters



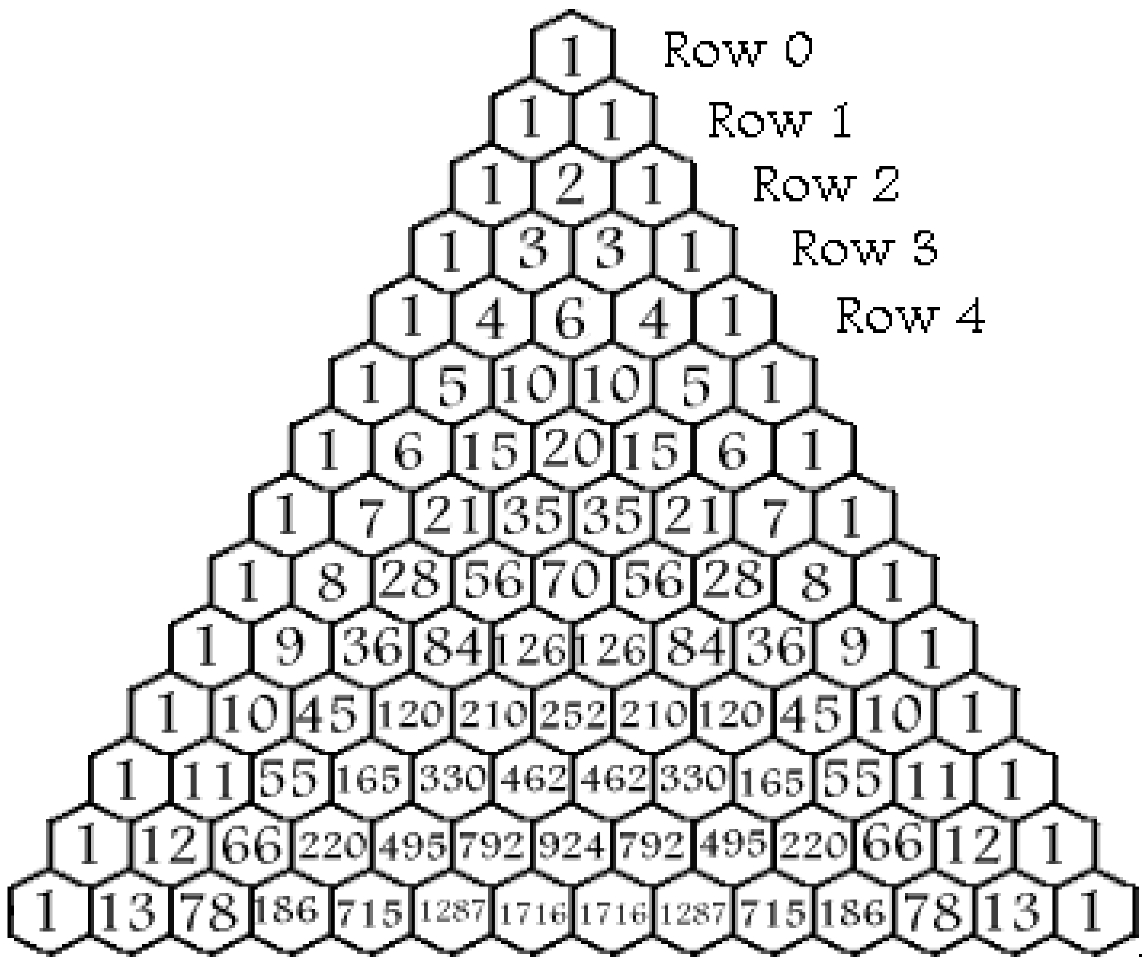
[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Pascal's Triangle (Binomial Expansion)

$$(1+x)^n = \binom{n}{0} + \binom{n}{1} x + \binom{n}{2} x^2 + \cdots + \binom{n}{n} x^n.$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Pascal’s Triangle



[http://ptri1.tripod.com/]

A Five Point Approximation

1	4	6	4	1

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Another Way: Compute the Weights

- Start with a discrete Gaussian

$$g[i,j] = c\, e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

- Normalize the weights

$$\frac{g[i,j]}{c} = e^{-\frac{(i^2+j^2)}{2\sigma^2}}$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Example: sigma^2=2, n=7

$[i, j]$	-3	-2	-1	0	1	2	3
-3	.011	.039	.082	.105	.082	.039	.011
-2	.039	.135	.287	.368	.287	.135	.039
-1	.082	.287	.606	.779	.606	.287	.082
0	.105	.368	.779	1.000	.779	.368	.105
1	.082	.287	.606	.779	.606	.287	.082
2	.039	.135	.287	.368	.287	.135	.039
3	.011	.039	.082	.105	.082	.039	.011

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

To keep them all integers

$$\frac{g[3,3]}{k} = e^{-\frac{(3^2+3^2)}{2(2)^2}} = 0.011 \implies k = \frac{g[3,3]}{0.011} = \frac{1.0}{0.011} = 91.$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Integer Weights

[i, j]	−3	−2	−1	0	1	2	3
−3	1	4	7	10	7	4	1
−2	4	12	26	33	26	12	4
−1	7	26	55	71	55	26	7
0	10	33	71	91	71	33	10
1	7	26	55	71	55	26	7
2	4	12	26	33	26	12	4
3	1	4	7	10	7	4	1

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Normalization constant

$$\sum_{i=-3}^{3}\sum_{j=-3}^{3} g[i,j] = 1115.$$

$$h[i,j] = \frac{1}{1115}(f[i,j] \star g[i,j])$$

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

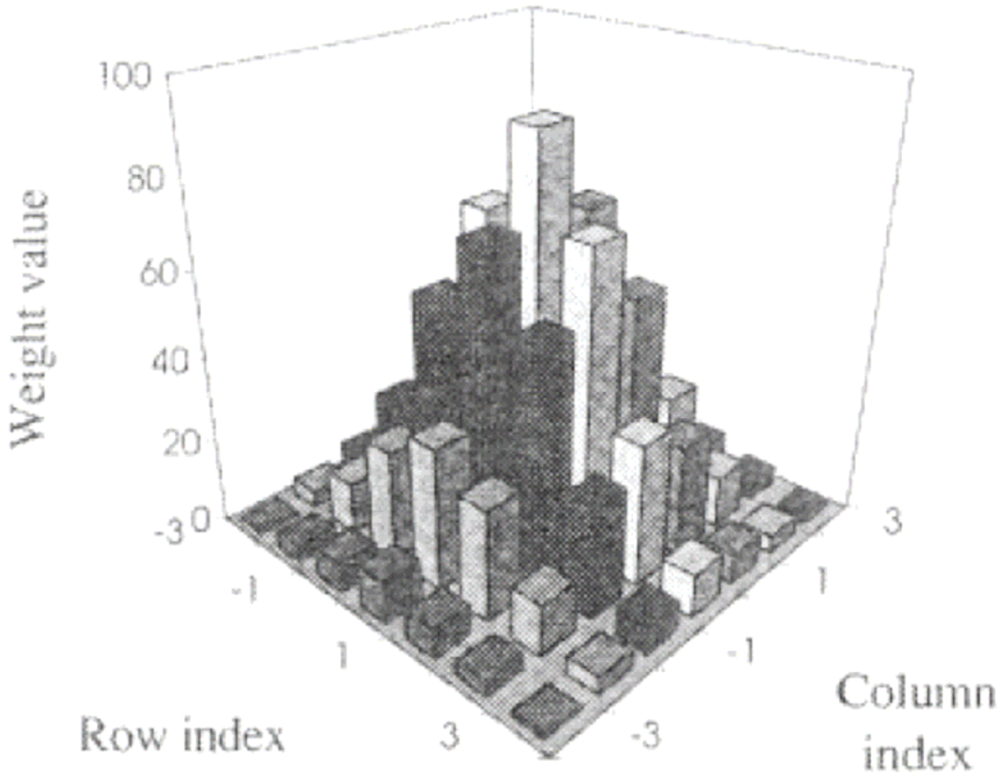
Discrete Gaussian Filters

7x7 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

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

3D Plot of the 7x7 Gaussian



[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

15 x 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

[Jain, Kasturi, and Schunck (1995). Machine Vision, Ch. 4]

Median filter

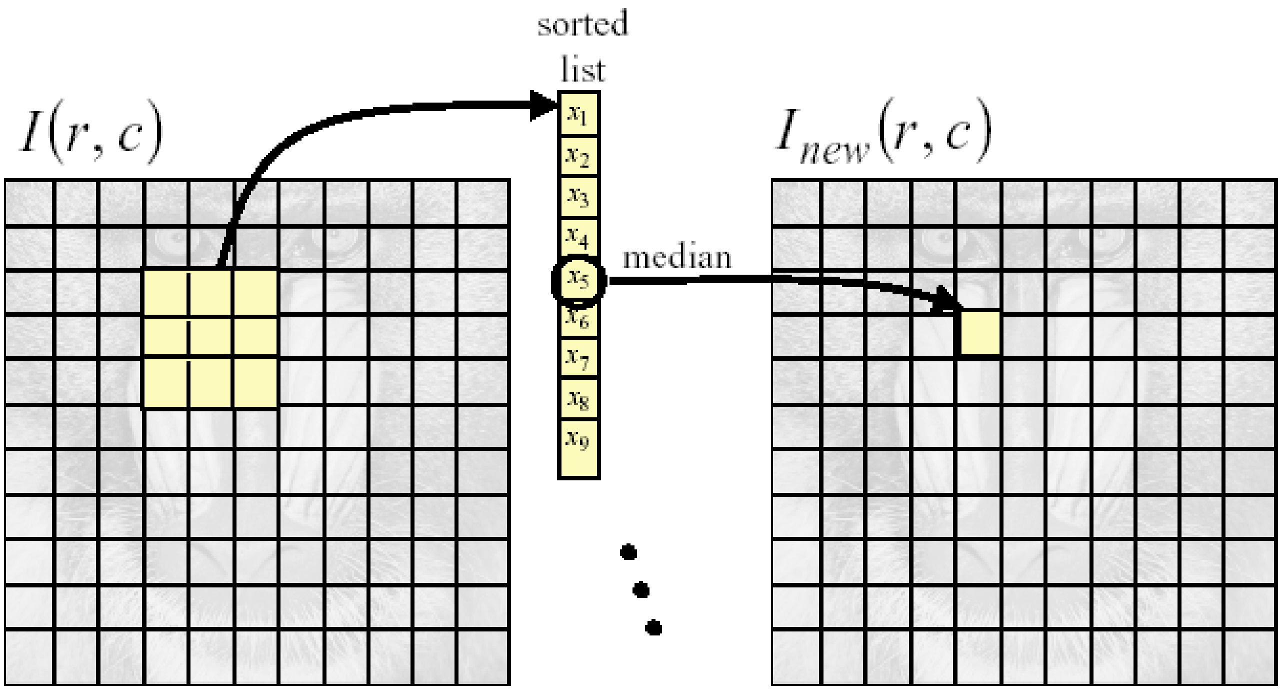
An example of nonlinear smoothing: the **median filter**

- Specify a window size, such as 3 x 3
- For each position of the window within the original image, compute the median of pixel values that lie in the window
- This becomes the new value in the output image

$$
\begin{aligned}
I_{new}(r,c) = \text{median}\{ \; & I(r-1,c-1), && I(r-1,c), && I(r-1,c+1), \\
& I(r,c-1), && I(r,c), && I(r,c+1), \\
& I(r+1,c-1), && I(r+1,c), && I(r+1,c+1) \}
\end{aligned}
$$

Median filter

Illustration with 3 x 3 window



Median filter

- Image with impulsive noise

(sometimes this is called "salt-and-pepper" noise)

- Result after 3x3 median filter

Compare with linear smoothing

Compare median filtering with linear smoothing:

1	2	1
2	4	2
1	2	1

- Image with impulsive noise
- Result using linear filter shown above