Linear Systems

- Many image processing (filtering) operations are modeled as a linear system

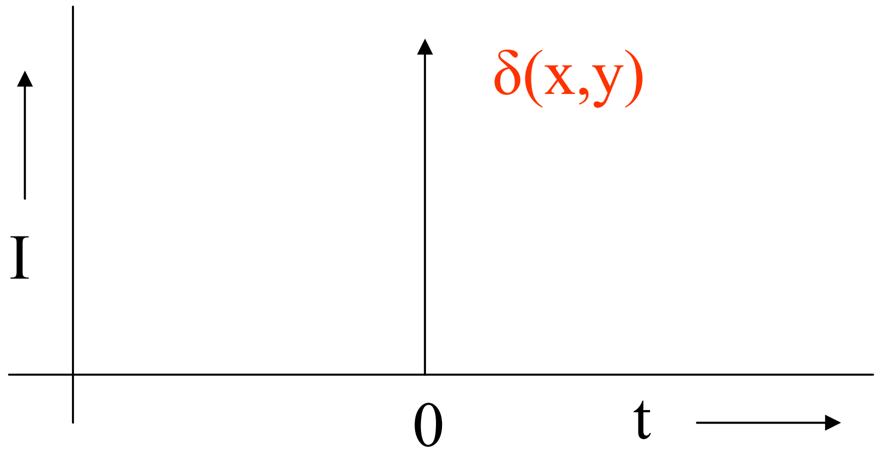
$$\delta(x,y) \longrightarrow \boxed{\textit{Linear System}} \longrightarrow h(x,y)$$

$$g(x,y) = f(x,y) * h(x,y) =$$

$$\iint_{-\infty} f(x',y')h(x-x',y-y')dxdy$$

Impulse Response

- System's output to an impulse $\delta(x,y)$



Space Invariance

- $g(x,y)$ remains the same irrespective of the position of the input pulse

$$\delta(x-x_0,y-y_0) \longrightarrow \boxed{\textit{Space Inv. Syst.}} \longrightarrow h(x-x_0,y-y_0)$$

- Linear Space Invariance (LSI)

$$af_1(x,y)+bf_2(x,y) \longrightarrow \boxed{\textit{LSI System}} \longrightarrow ah_1(x,y)+bh_2(x,y)$$

Discrete Convolution

- The filtered image is described by a discrete convolution

$$g(i,j) = f(i,j) * h(i,j) =$$

$$\sum_{k=1}^{n} \sum_{l=1}^{m} f(k,l)h(i-k, j-l)$$

- The filter is described by a n x m discrete convolution mask

Computing Convolution

- Invert the mask g(i,j) by 180°
  - not necessary for symmetric masks
- Put the mask over each pixel of f(i,j)
- For each (i,j) on image

$$h(i,j)=Ap_1+Bp_2+Cp_3+Dp_4+Ep_5+Fp_6+Gp_7+Hp_8+Ip_9$$

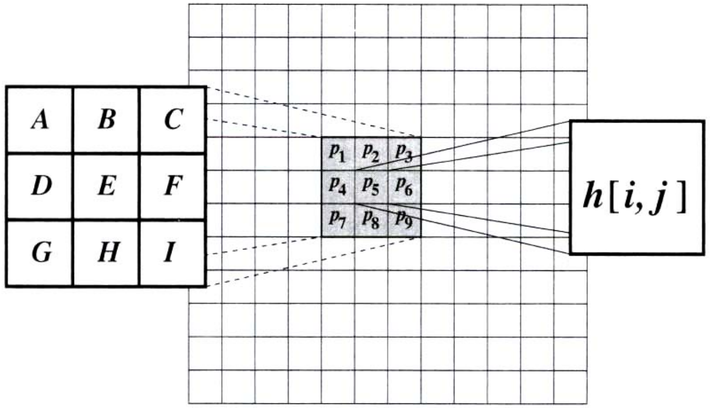
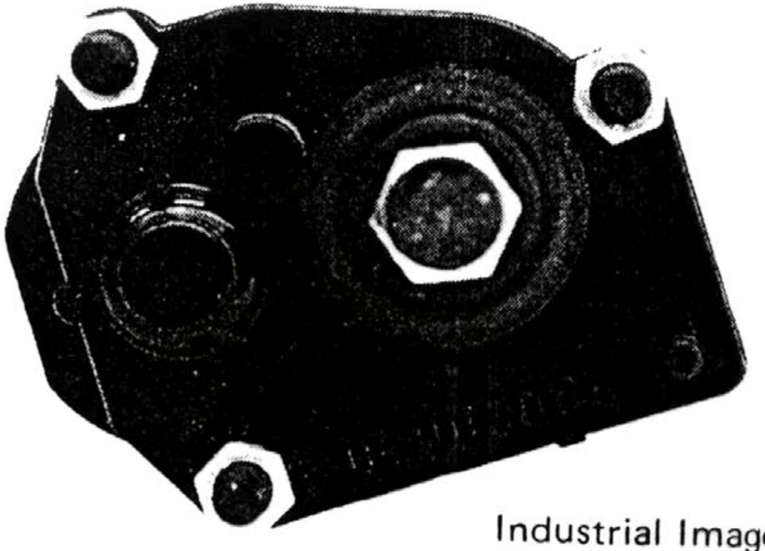


Image Filtering

- Images are often corrupted by random variations in intensity, illumination, or have poor contrast and can't be used directly
- *Filtering*: transform pixel intensity values to reveal certain image characteristics
  - *Enhancement:* improves contrast
  - *Smoothing:* remove noises
  - *Template matching*: detects known patterns

Template Matching

- Locate the template in the image



Industrial Image

Template

Computing Template Matching

- Match template with image at every pixel
  - distance → 0 : the template matches the image at the current location

$$D^2(x,y) = \sum_{x'=0}^{m} \sum_{y'=0}^{n} \left[f(x',y') - t(x'-x, y'-y) \right]^2$$

  - t(x,y): template
  - M,N size of the template

$$D^2(x,y) =$$

$$\sum_{x'=1}^{M}\sum_{y'=1}^{N}\left[f(x',y') - t(x'-x,y'-y)\right]^2 =$$

$$\sum_{x'=1}^{M}\sum_{y'=1}^{N} f(x',y')^2 +$$

background

constant

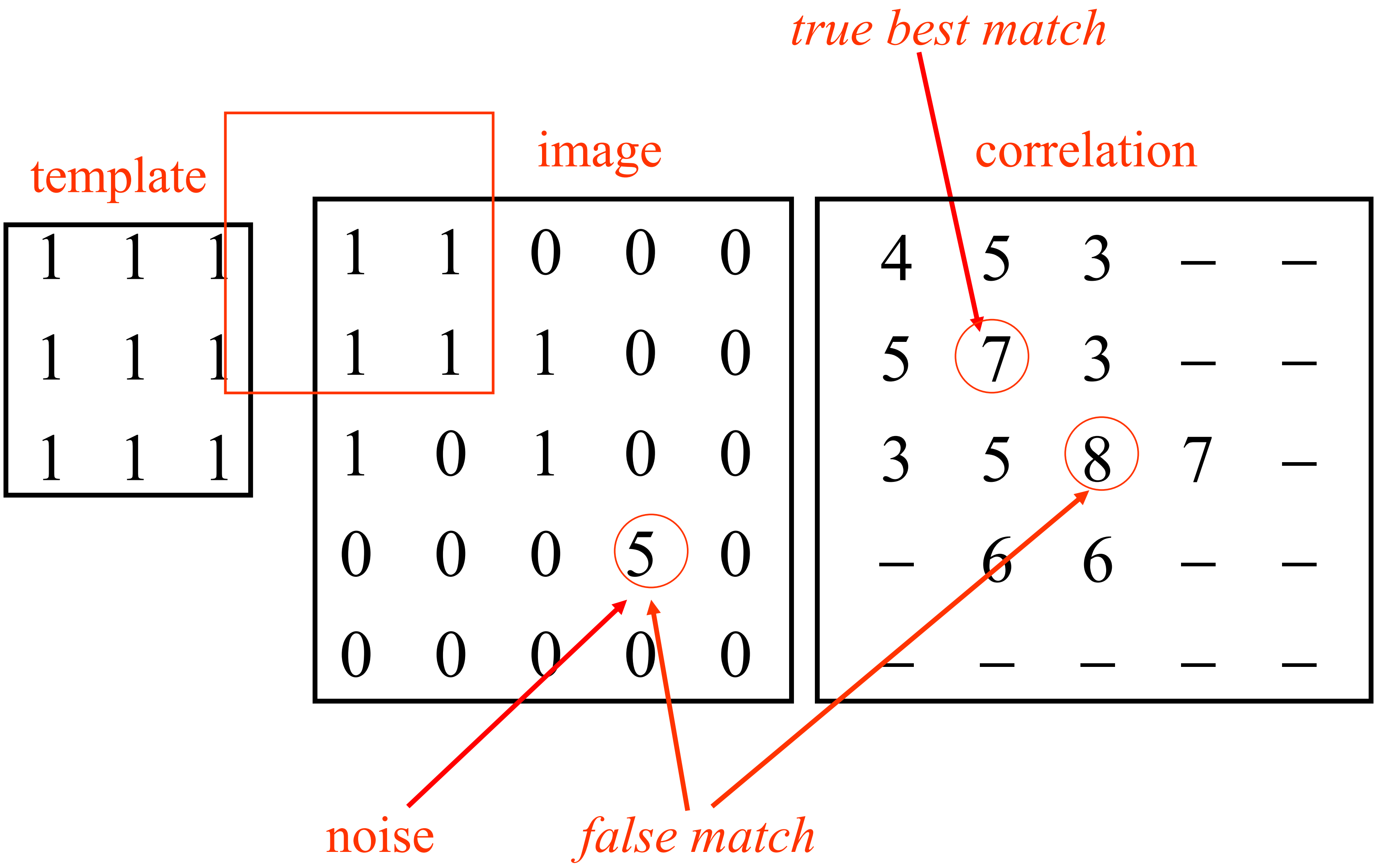
$$\sum_{x'=1}^{M}\sum_{y'=1}^{N} t(x'-x,y'-y)^2 -$$

correlation:

convolution of

f(x,y) with t(-x,-y)

$$2\sum_{x'=1}^{M}\sum_{y'=1}^{N} f(x',y')\,t(x'-x,y'-y)$$


true best match
template
image
correlation
1 1 1
1 1 1
1 1 1
1 1 0 0 0
1 1 1 0 0
1 0 1 0 0
0 0 0 5 0
0 0 0 0 0
4 5 3 – –
5 7 3 – –
3 5 8 7 –
– 6 6 – –
– – – – –
noise
false match

Observations

- If the size of f(x,y) is n x n and the size of the template is m x m the result is accumulated in a (n-m-1) x (n+m-1) matrix
- Best match: maximum value in the correlation matrix but,
  – false matches due to noise

Disadvantages of Correlation

- Sensitive to noise
- Sensitive to variations in orientation and scale
- Sensitive to non-uniform illumination
- *Normalized Correlation* (1:image, 2:template):

$$N(x,y)=\frac{E(q_1q_2)-E(q_1)E(q_2)}{\sigma(q_1)\sigma(q_2)}$$

$$\sigma(q)=\left[E(q^2)-E(q)^2\right]^{1/2}$$

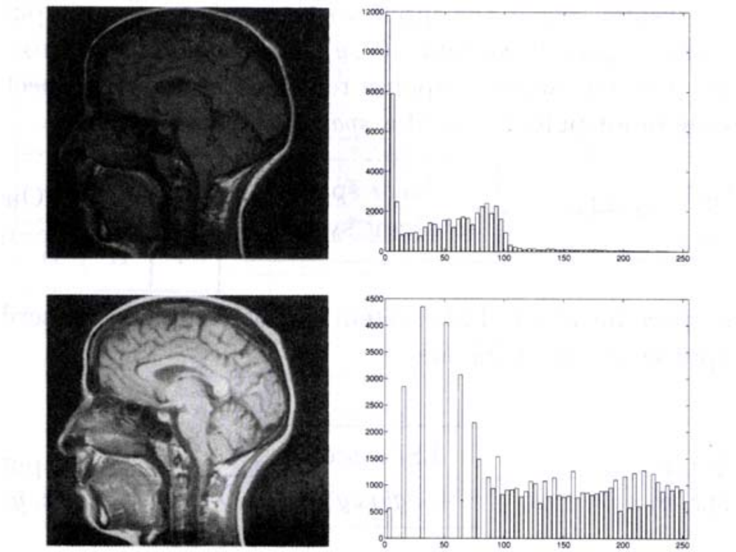
  - **E** : expected value

Histogram Modification

- Images with poor contrast usually contain unevenly distributed gray values
- Histogram Equalization is a method for stretching the contrast by uniformly distributing the gray values
  - enhances the quality of an image
  - useful when the image is intended for viewing
  - not always useful for image processing

Example

- The original image has very poor contrast
  - the gray values are in a very small range
- The histogram equalized image has better contrast



Histogram Equalization Methods

- Background Subtraction: subtract the "background" if it hides useful information
  - $f'(x,y) = f(x,y) - f_b(x,y)$
- Static & Dynamic histogram equalization methods
  - Histogram scaling (static)
  - Statistical scaling (dynamic)

Static Histogram Scaling

- Scale uniformly entire histogram range:
  - $[z_1, z_k]$: available range of gray values:
  - $[a,b]$: range of intensity values in image:
  - scale $[a,b]$ to cover the entire range $[z_1, z_k]$
  - for each z in $[a,b]$ compute

$$z' = \frac{z_k - z_1}{b - a}(z - a) + z_1$$

  - the resulting histogram may have gaps

Statistical Histogram Scaling

- Fills all histogram bins continuously
  - p_i : number of pixels at level z_i input histogram
  - q_i : number of pixels at level z_i output histogram
  - $k_1 = k_2 = \ldots$: desired number of pixels in histogram bin
- Algorithm:
  1. Scan the input histogram from left to right to find k_1:

$$\sum_{i=1}^{k_1-1} p_i \leq q_1 < \sum_{i=1}^{k_1} p_i$$

  - all pixels with values $z_1, z_2, \ldots, z_{k-1}$ become z_1

Algorithm (conted)

2. Scan the input histogram from k_1 and to the right to find k_2:

$$\sum_{i=1}^{k_2-1} p_i \leq q_1 + q_2 < \sum_{i=1}^{k_2} p_i$$

– all pixels $z_{k1}, z_{k1+1}, \ldots, z_{k2}$ become z_2

- Continue until the input histogram is exhausted
  – might also leave gaps in the histogram

Noise

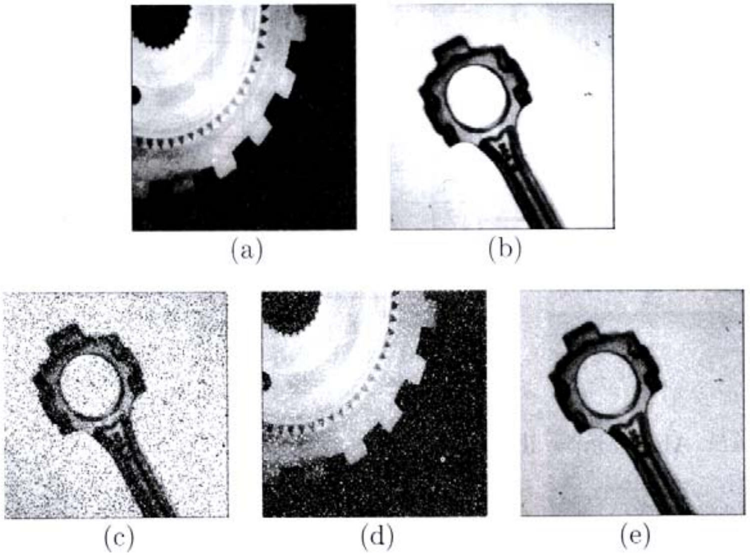
- Images are corrupted by random variations in intensity values called noise due to non-perfect camera acquisition or environmental conditions.
- *Assumptions*:
  - Additive noise: a random value is added at each pixel
  - White noise: The value at a point is independent on the value at any other point.

Common Types of Noise

- **Salt and pepper noise**: random occurrences of both black and white intensity values
- **Impulse noise**: random occurrences of white intensity values
- **Gaussian noise**: impulse noise but its intensity values are drawn from a Gaussian distribution
  - noise intensity value:
  - k: random value in [a,b]
  - **σ** : width of Gaussian
  - models sensor noise (due to camera electronics)

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

Examples of Noisy Images



a. Original image
b. Original image
c. Salt and pepper noise
d. Impulse noise
e. Gaussian noise

Noise Filtering

- Basic Idea: replace each pixel intensity value with an new value taken over a neighborhood of fixed size
  - Mean filter
  - Median filter
- The size of the filter controls degree of smoothing
  - large filter → large neighborhood → intensive smoothing

Mean Filter

- Take the average of intensity values in a m x n region of each pixel (usually m = n)
  - take the average as the new pixel value

$$h(i,j) = \frac{1}{mn} \sum_{k \in m} \sum_{l \in n} f(k,l)$$

  - the normalization factor mn preserves the range of values of the original image

Mean Filtering as Convolution

- Compute the convolution of the original image with

$$g(i,j) = \frac{1}{3 \cdot 3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  – simple filter, the same for all types of noise
  – *disadvantage*: blurs image, detail is lost

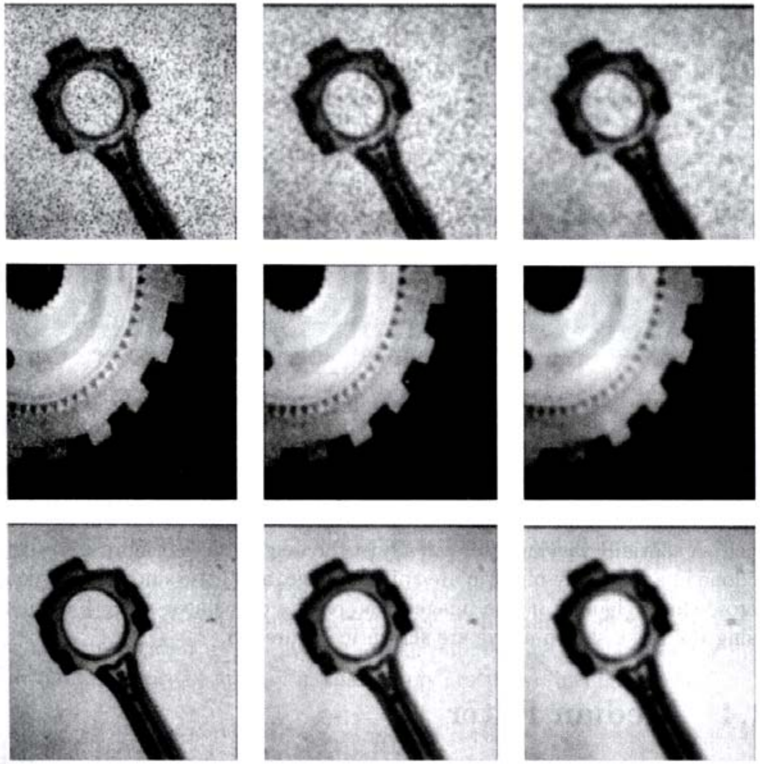
Size of Filter

- The size of the filter controls the amount of filtering (and blurring).
  - 5 x 5, 7 x 7 etc.

$$g(i,j) = \frac{1}{5 \cdot 5}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

  - different weights might also be used
  - normalize by sum of weights in filter

Examples of Smoothing

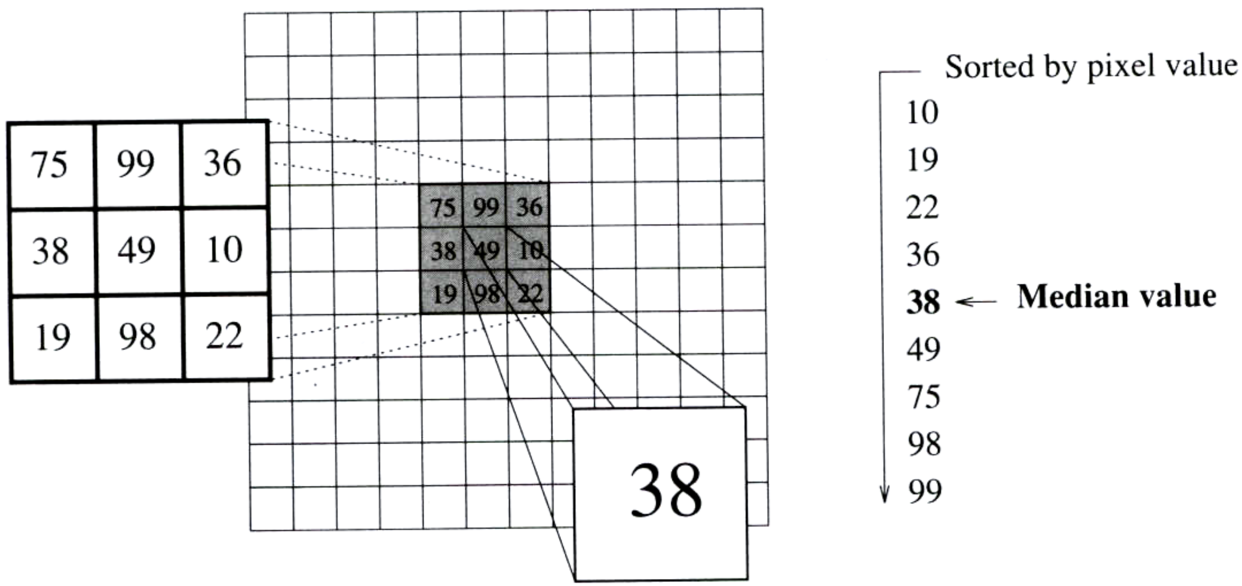


- **From left to right**: results of 3 x 3, 5 x 5 and 7 x 7 mean filters

Median Filter

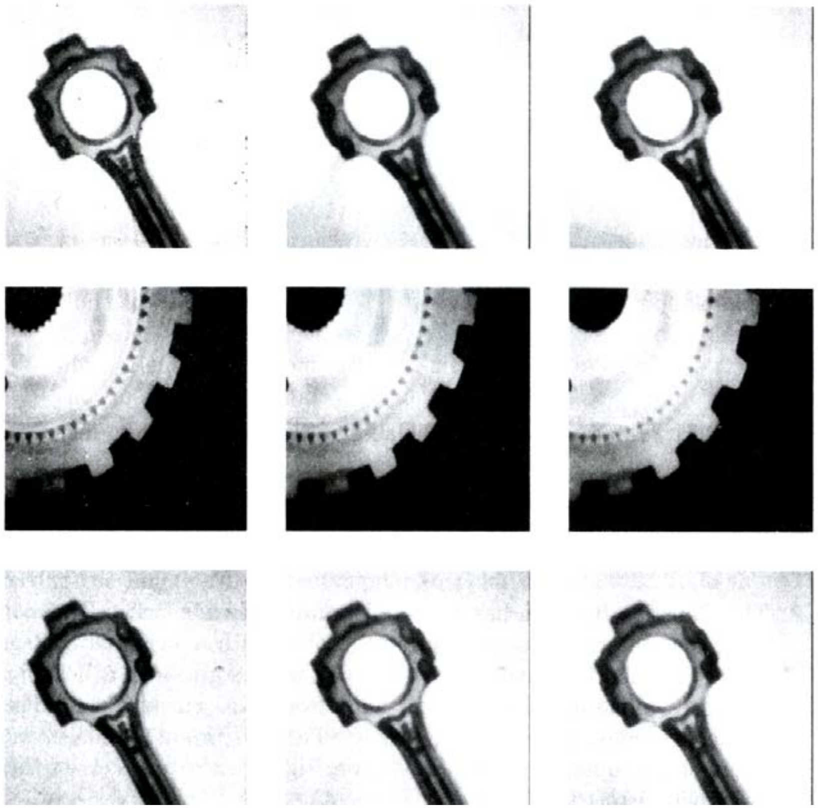
- Replace each pixel value with the median of the gray values in the region of the pixel:
  1. take a 3 x 3 (or 5 x 5 etc.) region centered around pixel (i,j)
  2. sort the intensity values of the pixels in the region into ascending order
  3. select the middle value as the new value of pixel (i,j)

Computation of Median Values



- Very effective in removing salt and pepper or impulsive noise while preserving image detail
- Disadvantages: computational complexity, non linear filter

Examples of Median Filtering



- **From left to right**: the results of a 3 x 3, 5 x 5 and 7 x 7 median filter

Gaussian Filter

- Filtering with a m x m mask
  - the weights are computed according to a Gaussian function:
  - **σ** is user defined

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

Example:

m = n = 7

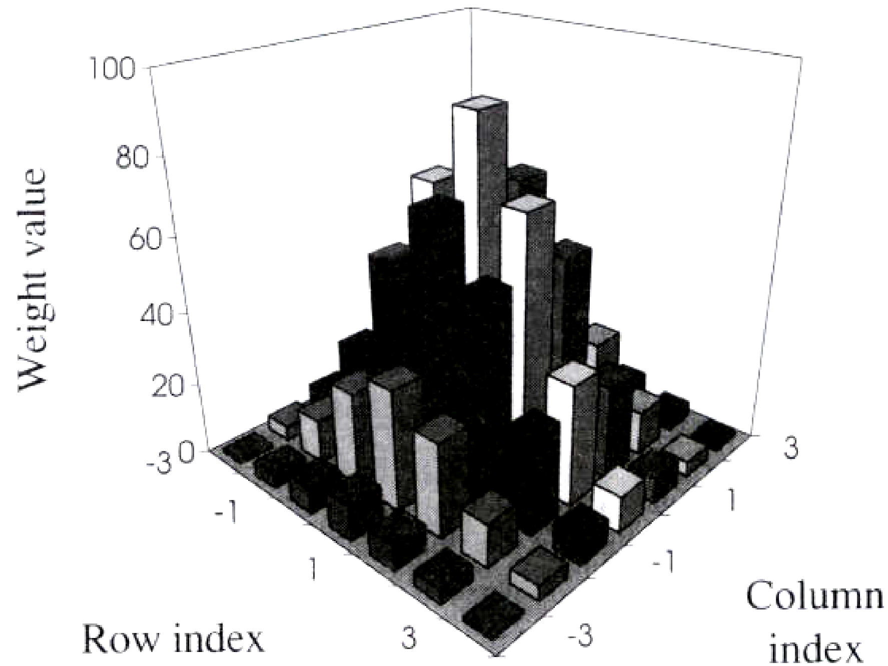
$\sigma^2 = 2$

$$\begin{bmatrix} 1 & 4 & 7 & 10 & 7 & 4 & 1 \\ 4 & 12 & 26 & 33 & 26 & 12 & 4 \\ 7 & 26 & 55 & 71 & 55 & 26 & 7 \\ 10 & 33 & 71 & 91 & 71 & 33 & 10 \\ 7 & 26 & 55 & 71 & 55 & 26 & 7 \\ 4 & 12 & 26 & 33 & 26 & 12 & 4 \\ 1 & 4 & 7 & 10 & 7 & 4 & 1 \end{bmatrix}$$

Properties of Gaussian Filtering

- Gaussian smoothing is very effective for removing Gaussian noise
- The weights give higher significance to pixels near the edge (reduces edge blurring)
- They are linear low pass filters
- Computationally efficient (large filters are implemented using small 1D filters)
- Rotationally symmetric (perform the same in all directions)
- The degree of smoothing is controlled by σ (larger σ for more intensive smoothing)

Gaussian Mask



- A 3-D plot of a 7 x & Gaussian mask: filter symmetric and isotropic

Gaussian Smoothing

- The results of smoothing an image corrupted with Gaussian noise with a 7 x 7 Gaussian mask

Computational Efficiency

- Filtering twice with g(x) is equivalent to filtering with a larger filter with $\sigma' = \sqrt{2}\sigma$
- Assumptions

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

$$h(x,y) = f(x,y) * g(x,y)$$
$$h'(x,y) = f(x,y) * g(x,y) * g(x,y)$$

$$g(x) * g(x) = \int_{-\infty}^{+\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} d\xi =$$

$$= \int_{-\infty}^{+\infty} e^{-\frac{\left(\frac{x}{2}+\xi\right)^2}{2\sigma^2}} e^{-\frac{\left(\frac{x}{2}-\xi\right)^2}{2\sigma^2}} d\xi =$$

$$= \int_{-\infty}^{+\infty} e^{-\frac{\left(2\xi^2+\frac{x^2}{2}\right)}{2\sigma^2}} d\xi = \sqrt{\pi}\sigma e^{-\frac{x^2}{2\left(\sqrt{2}\sigma\right)^2}}$$

Observations

- Filter an image with a large Gaussian $\sqrt{2}\sigma$
  - equivalently, filter the image twice with a Gaussian with small σ
  - filtering twice with a $m \times n$ Gaussian is equivalent to filtering with a $(n + m - 1) \times (n + m - 1)$ filter
  - this implies a significant reduction in computations

Gaussian Separability

$$h(i,j) = f(i,j) * g(i,j) =$$

$$= \sum_{k=1}^{m} \sum_{l=1}^{n} g(k,l) f(i-k, j-l) =$$

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

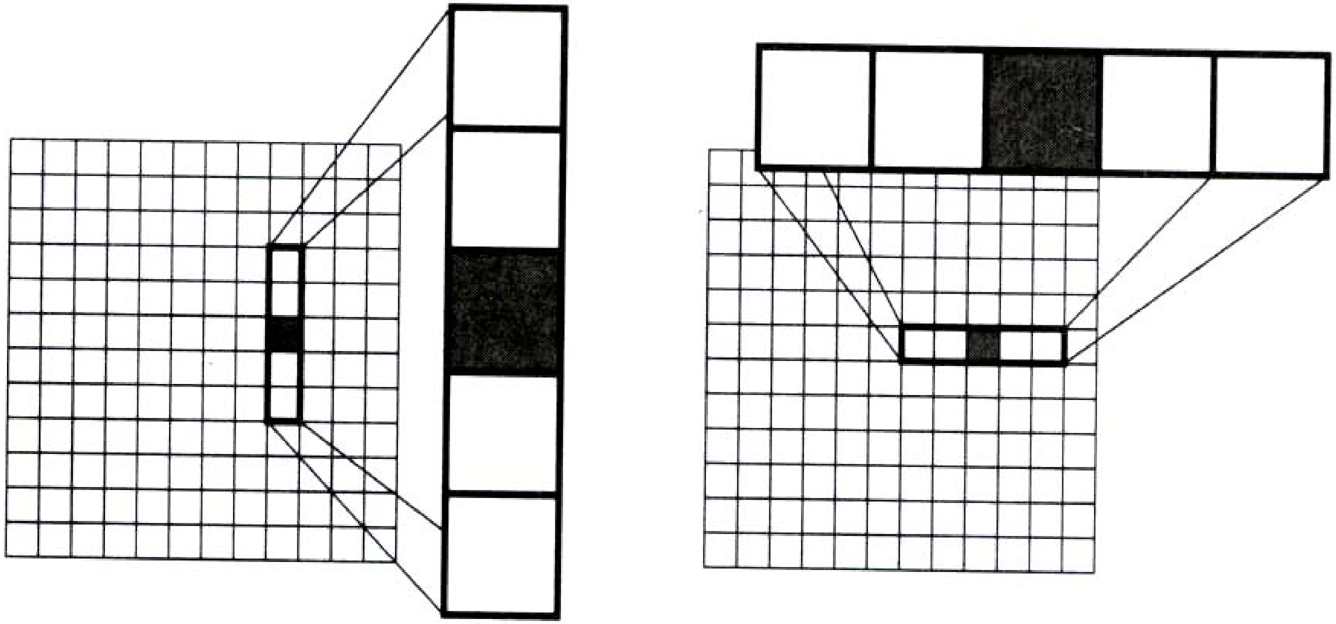
$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} \left[\underbrace{\sum_{l=1}^{n} e^{-\frac{l^2}{2\sigma^2}} f(i-k, j-l)}_{h'} \right] =$$

$$= \sum_{k=1}^{m} e^{-\frac{k^2}{2\sigma^2}} h'(i-k, j)$$

1-D Gaussian horizontally

1-D Gaussian vertically

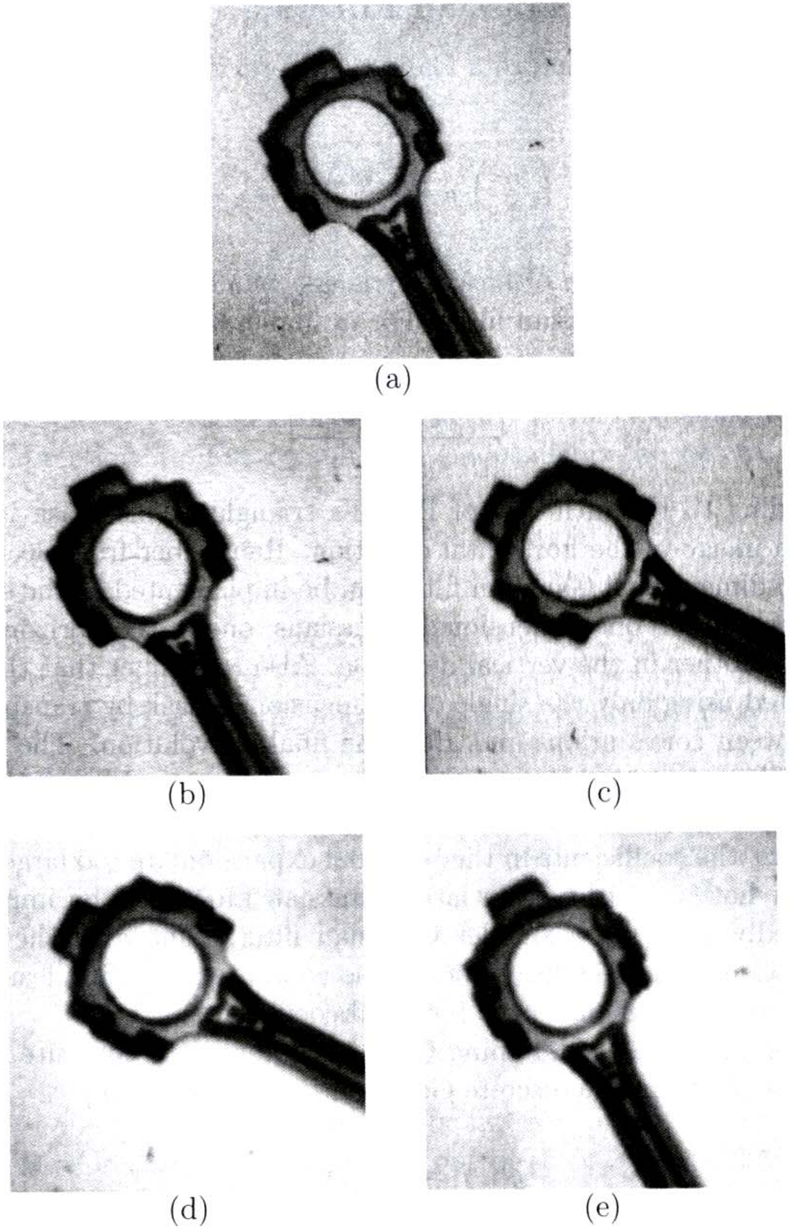
- The order of convolutions can be reversed



- An example of the separability of Gaussian convolution
  - **left**: convolution with vertical mask
  - **right**: convolution with horizontal mask

Gaussian Separability

- Filtering with a 2D Gaussian can be implemented using two 1D Gaussian horizontal filters as follows:
  - first filter with an 1D Gaussian
  - take the transpose of the result
  - convolve again with the same filter
  - transpose the result
- Filtering with two 1D Gausians is faster !!



a. Noisy image
b. Convolution with 1D horizontal mask
c. Transposition
d. Convolution with same 1D mask
e. Transposition → smoothed image