Computer Graphics and Image Processing

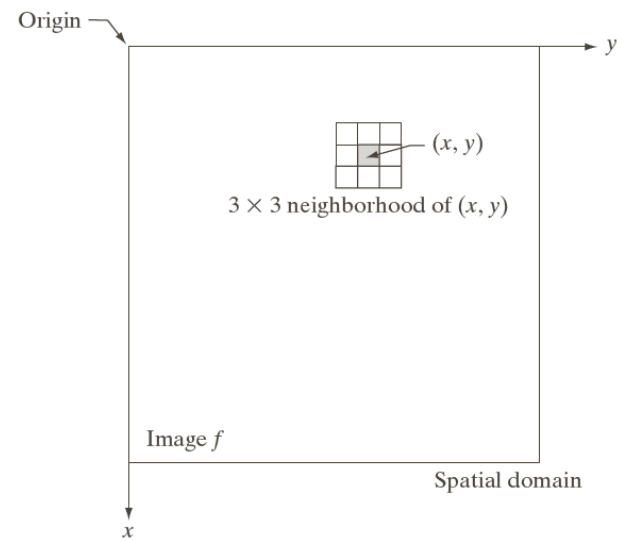
Part 3: Image Processing

4 – Image Filtering

Martin Urschler, PhD

Intensity mapping vs. spatial filtering

- Up to now: Focus on greyvalue transformations
  - Modify contrast and brightness through linear transformation
  - Nonlinear transformation for histogram equalization
  - Input and output is a **single pixel**



- Spatial image filtering operates on a **neighborhood** (moving window)
  - Intensity transformations can be seen as operating on a $1 \; x \; 1$ neighborhood
  - Image filtering: $n \; x \; n$ neighborhood ($n > 1$, often n is odd!)
  - Again, linear or nonlinear transformations possible

Spatial image filtering

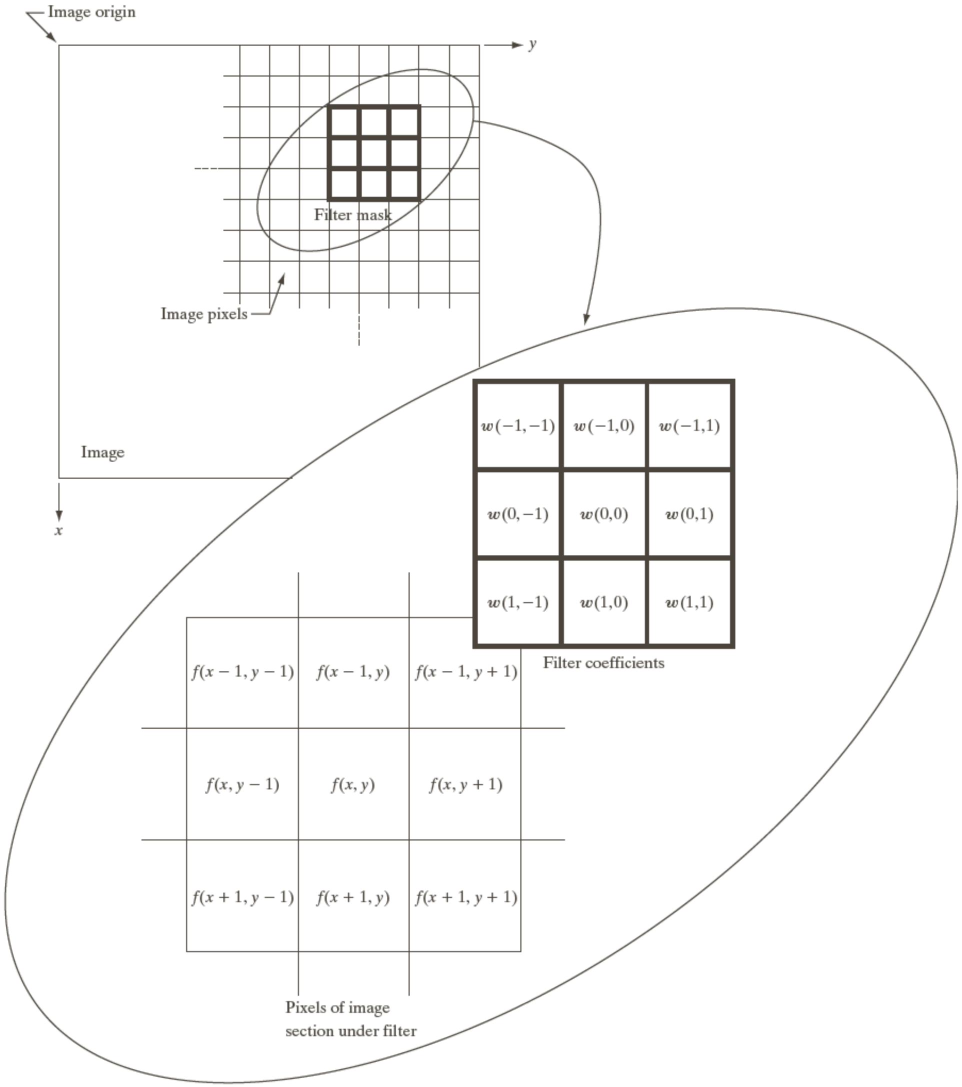
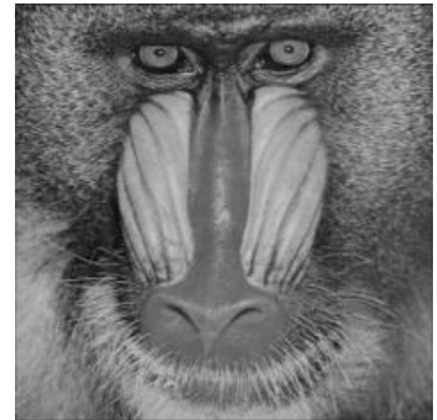


FIGURE 3.28 The mechanics of linear spatial filtering using a 3 × 3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

Example: Mean filter 5 x 5



Input:
235 x 235 px

Smoothing =
noise filtering

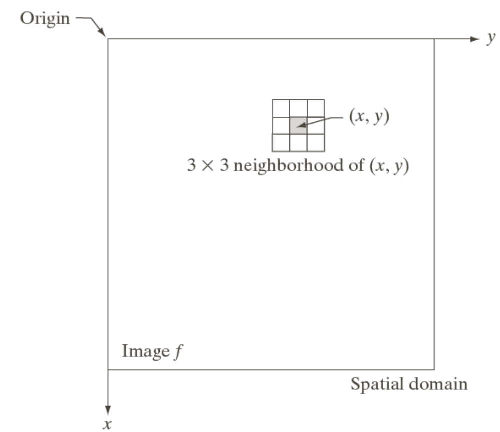
Images from: Gonzalez & Woods, Digital Image Processing, 3rd ed.

Basic principles of filtering

- Output filtered image $g(x,y)$ is obtained from input image $f(x,y)$ by applying moving window transform in $(2k+1)\ x\ (2l\ +\ 1)$ neighborhood (window): $g\ =\ MWT(f)$
  - Value $g(x,y)$ at each pixel location (x,y) is a certain linear or non-linear function of values of the original image $f(x,y)$
  - The values of f are taken in a $(2k+1)\ x\ (2l+1)$ rectangle being centered on pixel location (x,y)
    - E.g. $(k=1, l=1)$ for a $3\ x\ 3$ window and $(k=1, l\ =\ 3)$ for a $3\ x\ 7$ window
  - A general linear MWT multiplies each filter coefficient $w(\xi,\eta)$ with the image value that lies directly beneath it, and sums up these terms.
  - Example: Mean filter is just a sum with fixed weight!

$$\mu(x,y) = \frac{1}{(2k+1)(2l+1)} \sum_{\xi=-k}^{k} \sum_{\eta=-l}^{l} f(x+\xi, y+\ \eta)$$

Basic principles of filtering



- Windows are centered at each location (x, y)
- Popular windows – rectangles with odd sizes, e.g. $3\ x\ 3$, $5\ x\ 5$, $3\ x\ 1$, etc.

- The $3\ x\ 3$ square window gives a set of offsets representing the neighborhood:

$$\{(\xi, \eta) :\ \xi = -1,0,1;\ \eta = -1,0,1\}$$

- The $(2k+1)\ x\ (2l+1)$ rectangular window gives a set of offsets:

$$\{(\xi, \eta) :\ \xi = -k, \dots, -1,0,1, \dots, k;\ \eta = -l, \dots, -1,0,1, \dots, l\}$$

Spatial image filtering

- Example: Mean filter 5 x 5 applied to f

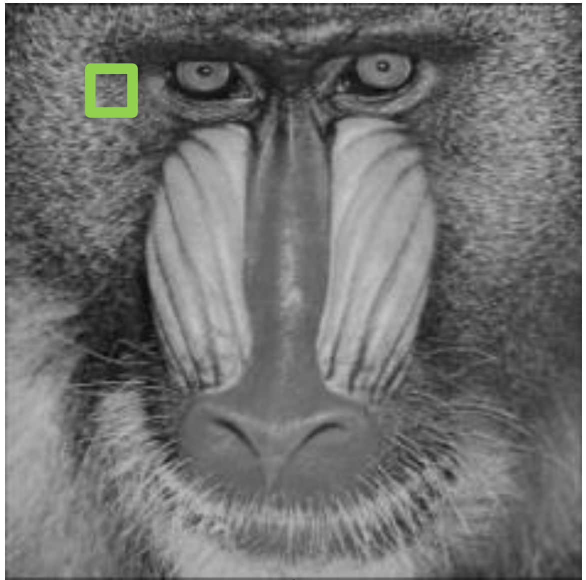
- Rectangular window $(k = 2, l = 2)$:

$$\{(\xi,\eta):\ \xi = -2,-1,0,1,2;\ \eta = -2,-1,0,1,2\}$$

- Mean filter generating output g:

$$g(x,y) = \mu(x,y) = \frac{1}{25}\sum_{\xi=-2}^{2}\sum_{\eta=-2}^{2} f(x+\xi, y+\eta)$$

$w(\xi,\eta) = 1$



$f(x,y)$

$g(x,y)$

Spatial image filtering

- Example: Mean filter 5 x 5 applied to f

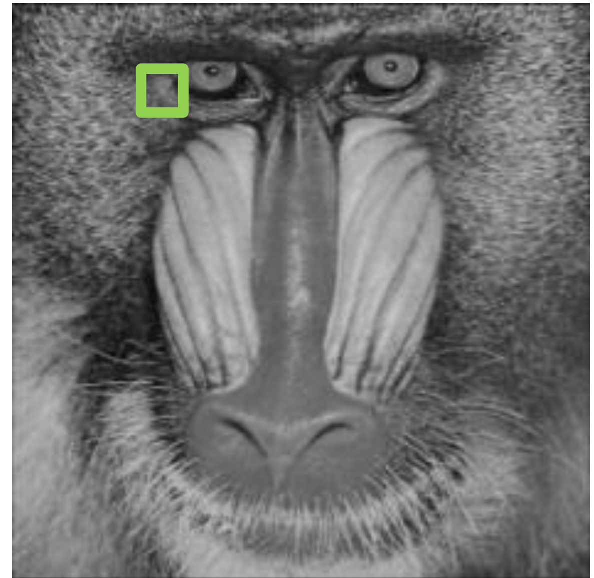
- Rectangular window $(k = 2, l = 2)$:

$$\{(\xi,\eta):\ \xi = -2,-1,0,1,2;\ \eta = -2,-1,0,1,2\}$$

- Mean filter generating output g:

$$g(x,y) = \mu(x,y) = \frac{1}{25}\sum_{\xi=-2}^{2}\sum_{\eta=-2}^{2} f(x+\xi, y+\eta)$$

$w(\xi,\eta) = 1$



$f(x,y)$

$g(x,y)$

Spatial image filtering

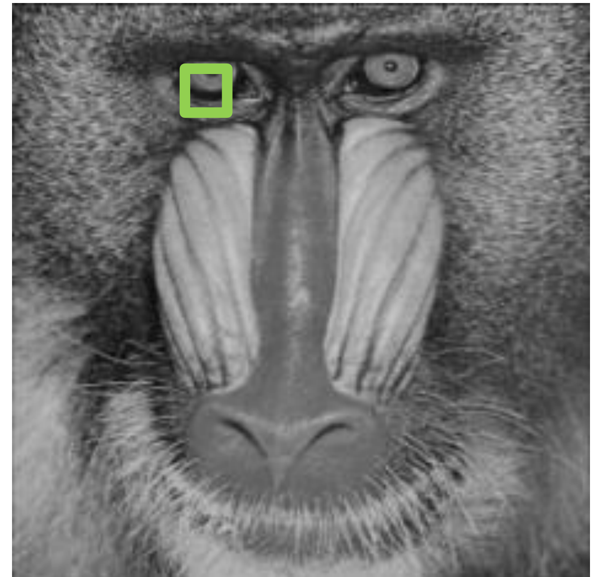
- Example: Mean filter 5 x 5 applied to f
- Rectangular window $(k = 2, l = 2)$:

$$\{(\xi,\eta):\ \xi = -2,-1,0,1,2;\ \eta = -2,-1,0,1,2\}$$

- Mean filter generating output g:

$$g(x,y) = \mu(x,y) = \frac{1}{25}\sum_{\xi=-2}^{2}\sum_{\eta=-2}^{2} f(x+\xi, y+\eta)$$

$w(\xi,\eta) = 1$



$f(x,y)$

$g(x,y)$

Spatial image filtering

- Generic linear MWT:

$$g(x,y) = \frac{1}{(2k+1)(2l+1)} \sum_{\xi=-k}^{k} \sum_{\eta=-l}^{l} w(\xi,\eta) f(x+\xi, y+\eta)$$

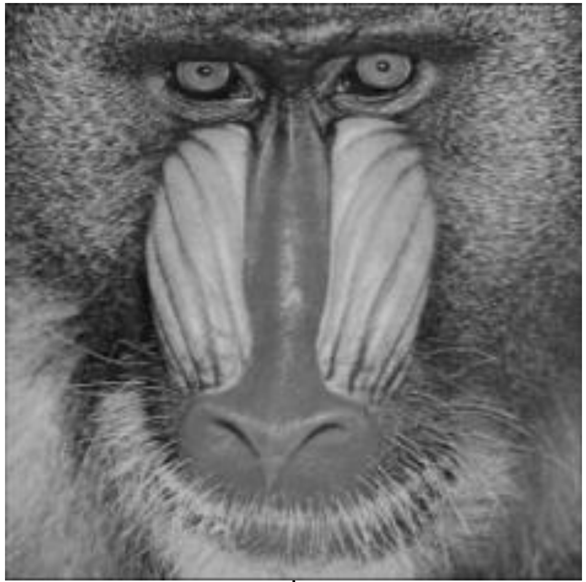
- Mean filter 3 x 3:

$$w(\xi,\eta) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Vertical edge filter 3 x 3:

We will come back to edges later!

$$w(\xi,\eta) = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$



Smoothing: Mean filter

- Mean, box or average filtering:
  - "Smoothing" images by reducing the variation of intensities between neighboring pixels -> reduce white (Gaussian) noise
  - Each value is replaced with the average value of the neighboring pixels, including itself, normalized by window size to stay inside original greyvalue range!
- Potential problems:
  - A single "outlier" can significantly affect the average of all the pixels in a neighborhood
  - Edge blurring: when the moving window crosses an edge, the filter will interpolate new pixel values on the edge
    - This may be a problem if sharp output edges are required!

Smoothing: Mean filter

- One dimensional (1D) example

x		0	1	2	3	4	5	6	7	8	9	
f	3 ←	3	9	4	52	3	8	6	2	2	9	→ 9

Moving window		Computing the mean		Rounding the mean and assigning to $g(x)$		
$W_0 : (3, 3, 9)$	$\Rightarrow$	$(3+3+9)/3 = 5$	$\Rightarrow$	$g(0) = \text{round}\{5\}$	$=$	5
$W_1 : (3, 9, 4)$	$\Rightarrow$	$(3+9+4)/3 = 5.33$	$\Rightarrow$	$g(1) = \text{round}\{5.33\}$	$=$	5
$W_2 : (9, 4, 52)$	$\Rightarrow$	$(9+4+52)/3 = 21.67$	$\Rightarrow$	$g(2) = \text{round}\{21.67\}$	$=$	22
$W_3 : (4, 52, 3)$	$\Rightarrow$	$(4+52+3)/3 = 19.67$	$\Rightarrow$	$g(3) = \text{round}\{19.67\}$	$=$	20
$W_4 : (52, 3, 8)$	$\Rightarrow$	$(52+3+8)/3 = 21$	$\Rightarrow$	$g(4) = \text{round}\{21\}$	$=$	21
$W_5 : (3, 8, 6)$	$\Rightarrow$	$(3+8+6)/3 = 5.67$	$\Rightarrow$	$g(5) = \text{round}\{5.67\}$	$=$	6
$W_6 : (8, 6, 2)$	$\Rightarrow$	$(8+6+2)/3 = 5.33$	$\Rightarrow$	$g(6) = \text{round}\{5.33\}$	$=$	5
$W_7 : (6, 2, 2)$	$\Rightarrow$	$(6+2+2)/3 = 3.33$	$\Rightarrow$	$g(7) = \text{round}\{3.33\}$	$=$	3
$W_8 : (2, 2, 9)$	$\Rightarrow$	$(2+2+9)/3 = 4.33$	$\Rightarrow$	$g(8) = \text{round}\{4.33\}$	$=$	4
$W_9 : (2, 9, 9)$	$\Rightarrow$	$(2+9+9)/3 = 6.67$	$\Rightarrow$	$g(9) = \text{round}\{6.67\}$	$=$	7

g	5	5	22	20	21	6	5	3	4	7

For $g(0)$ and $g(9)$, $f(0)$ and $f(9)$, respectively, are extended outside the boundaries.

2D Mean filter: 3 x 3 window

1: Keeping border values unchanged.

Averaging: $\mathrm{round}\{(1+4+0+2+2+4+1+0+1)/9\} = 2$

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Input $\mathbf{f}$

1	4	0	1	3	1
2	2	2	2	1	3
1	2	1	1	1	0
1	2	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Output $\mathbf{g}$

2D Mean filter: 3 x 3 window

2: Extending border values outside with the boundary values.

Averaging: $\text{round}\{(1+1+4+2+2+2+1+1+0)/9\} = 2$

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Input **f**

2	2	2	2	2	2
2	2	2	2	1	2
1	2	1	1	1	2
2	2	1	1	1	2
2	2	2	2	2	2
2	2	3	3	2	2

Output **g**

2D Mean filter: 3 x 3 window

3: Extending border values outside with zeros (zero padding).

Averaging: $\text{round}\{(0+1+4+0+2+2+0+1+0)/9 = 1$

0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

Input f

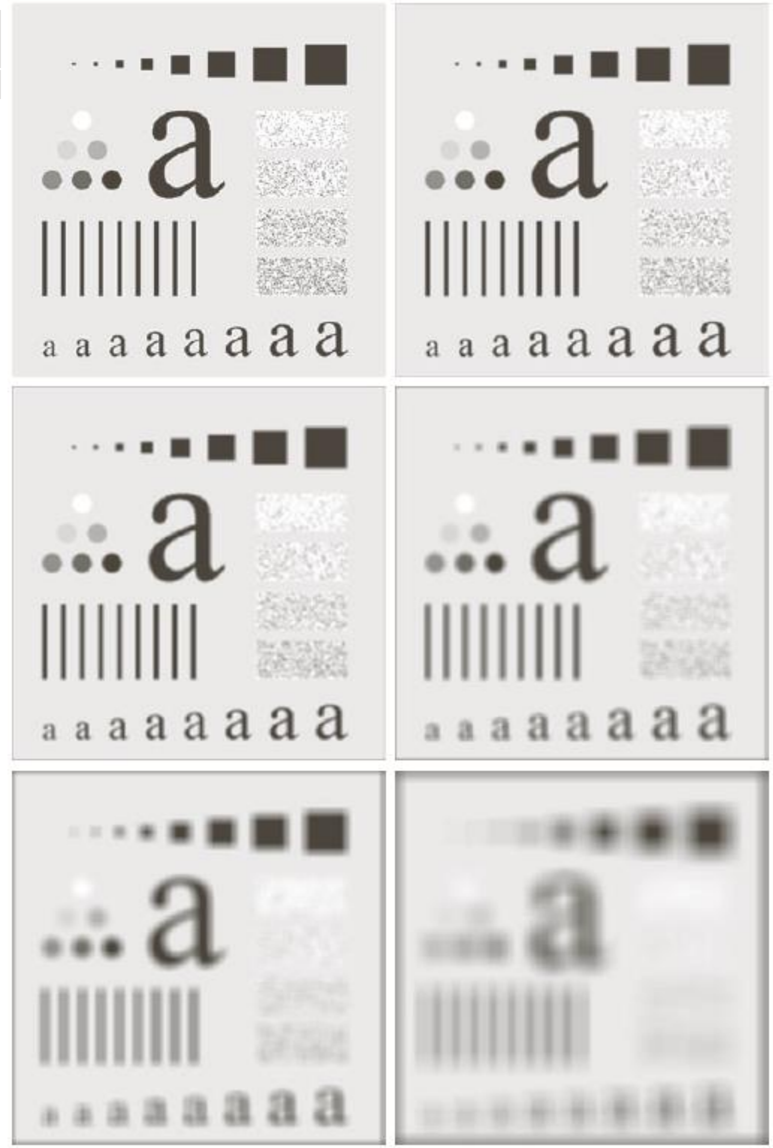
1	1	1	1	1	1
1	2	2	2	1	1
1	2	1	1	1	1
1	2	1	1	1	1
1	2	2	2	2	2
1	2	2	2	1	1

Output g

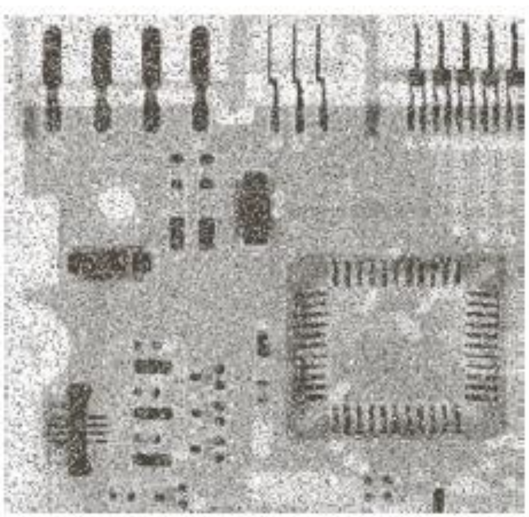
2D Mean filter: more examples

FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

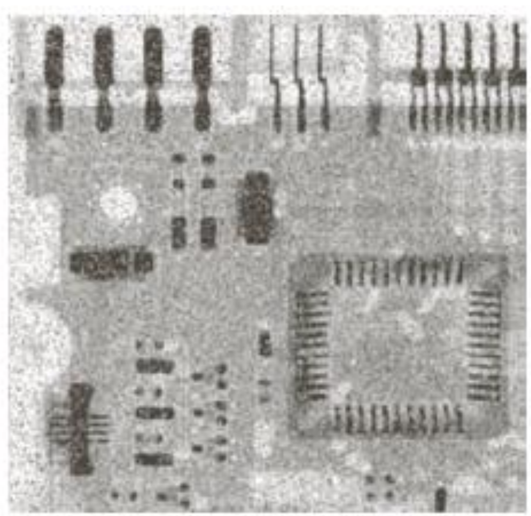
a	b
c	d
e	f



Images from: Gonzalez & Woods, Digital Image Processing, 3rd ed.



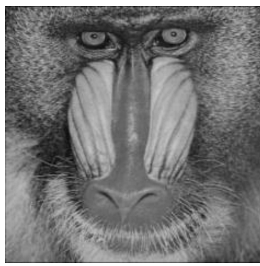
Noisy X-ray image of circuit board



Noise reduction with 3x3 averaging filter

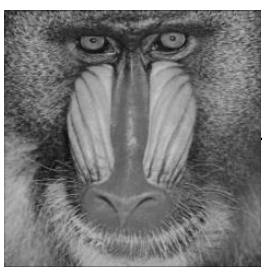
2D Mean filter: Separability

- For larger filter windows (think e.g. 15 x 15), two nested for loops are increasingly inefficient.
- Separable filter windows can be implemented more efficiently by composition of two operations with 1D filters!



$$w(\xi,\eta) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

O(k^2) operations per pixel



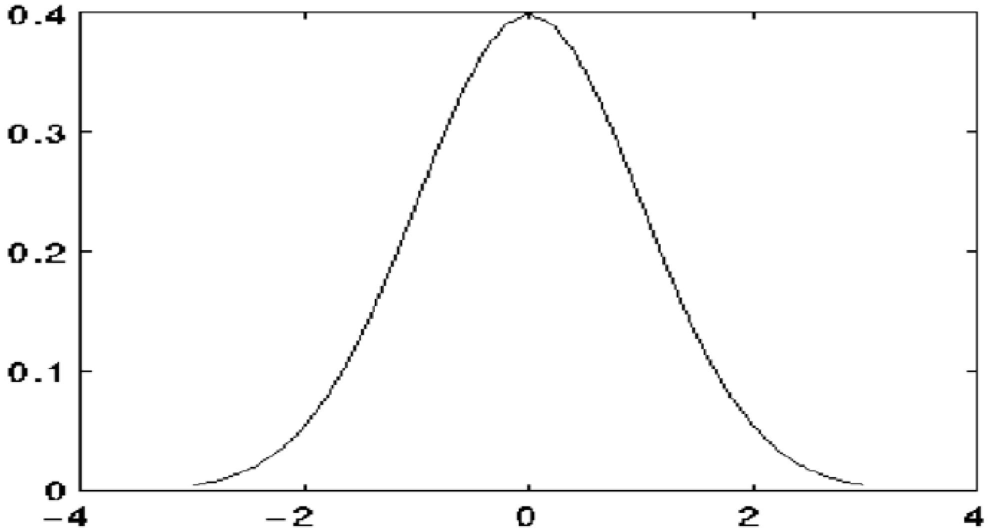
$$w_{row}(\xi,\eta) = \frac{1}{3}\,[1\ 1\ 1]$$

$$w_{col}(\xi,\eta) = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

O(k) operations per pixel, but needs memory for intermediate image!

Gaussian linear filter

- To blur images and remove noise and fine detail
- One-dimensional Gaussian function (zero mean)
- The pixels further away contribute less weight…



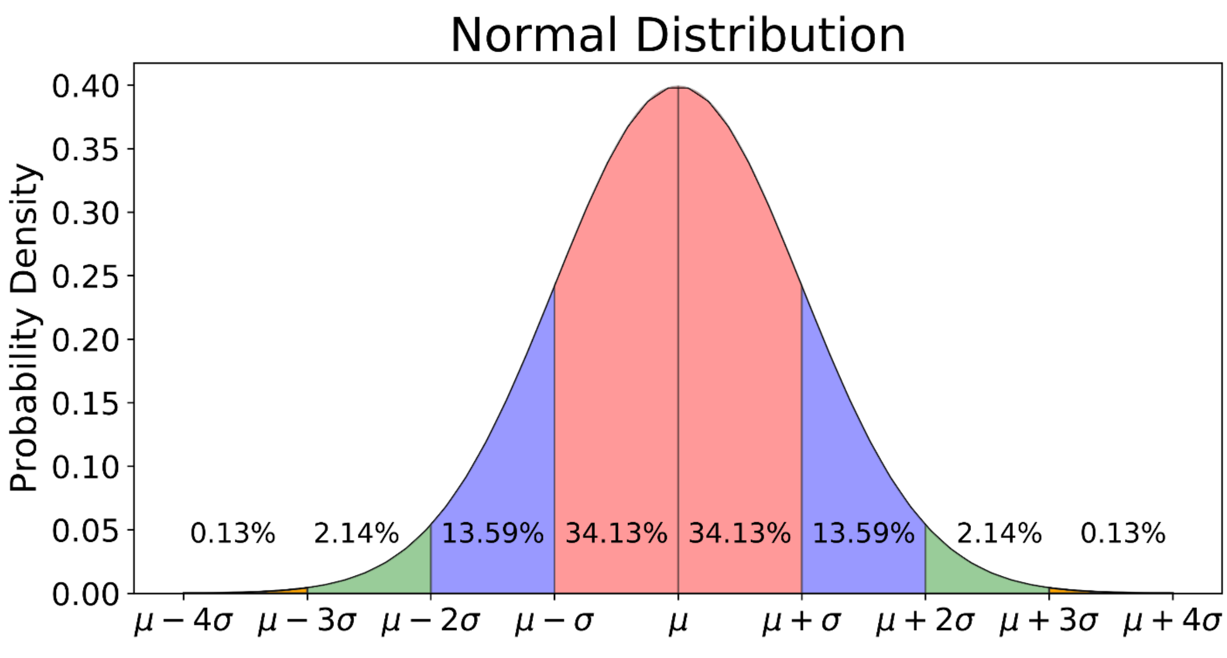
$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

σ – standard deviation

$\kappa = \frac{x}{\sigma}$	0	1	2
$G(\kappa)$	0.399	0.242	0.054
$\frac{G(\kappa)}{G(0)}$	1.000	0.607	0.135

Gaussian linear filter

- Standard deviation σ of Gaussian probability density function guides its behaviour:
  - 68% of the x-values are in the range $[mean - \sigma, mean + \sigma]$
  - 95% of the x-values are in the range $[mean - 2\sigma, mean + 2\sigma]$
  - 99.7% of the x-values are between $[mean - 3\sigma, mean + 3\sigma]$



1D Gaussian linear filter

- Built using normalized 1D Gaussian $\mathrm{G}(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$
- Filter kernel $s(x)$: a discrete approximation of the normalized Gaussian; e.g. for the $1 \; x \; 5$ filter with $\sigma = 1$:

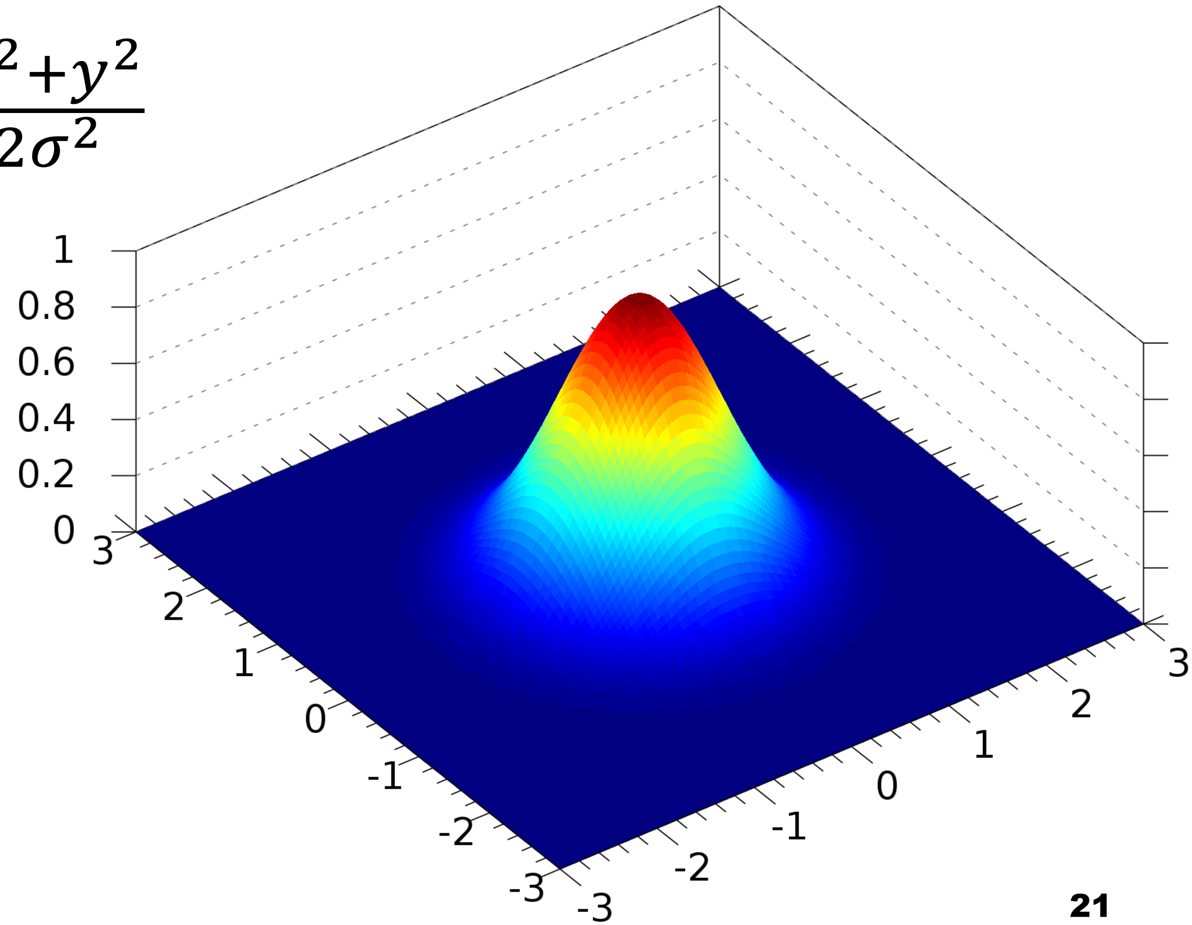
x	-2	-1	0	1	2
$s(x)$	0.054	0.242	0.399	0.242	0.054
Old: $\lfloor 10s(x) \rceil$	1	2	4	2	1
Old: kernel	$\frac{1}{10}$	$\frac{2}{10}$	$\frac{4}{10}$	$\frac{2}{10}$	$\frac{1}{10}$

- Rule of thumb: set filter half-width to about 2-3σ
- Still often used filter kernel for $1 \; x \; 3$ filter with $\sigma = 1$ is $\frac{1}{4}[1\ 2\ 1]$

Gaussian linear filter

- 2D Gaussian filter is built using the (isotropic) 2D Gaussian function

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



- 2D Gaussian is separable!

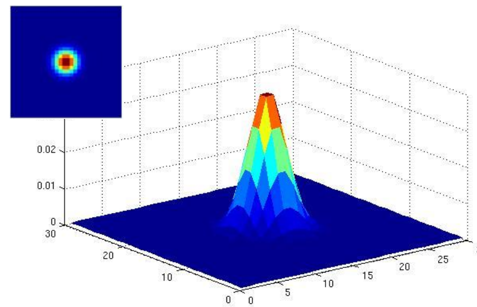
$$G(x, y) = G(x)G(y)$$

2D Gaussian linear filter

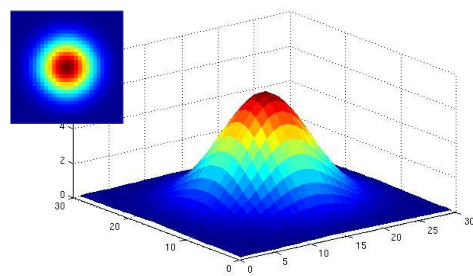
- Filter kernel $s(x)$: a discrete approximation of the normalized continuous 2D Gaussian; e.g. for the $3\ x\ 3$ and $5\ x\ 5$ filters with $\sigma = 1$:

$$s_{3x3}(x) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad s_{5x5}(x) = \frac{1}{100}\begin{bmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

- The larger the value of σ, the wider the peak of the Gaussian and the larger the blurring
- Non-uniform averaging: low pass filtering
- Rotational symmetry with no directional bias
- Fast computations due to separability
- Might not preserve image brightness

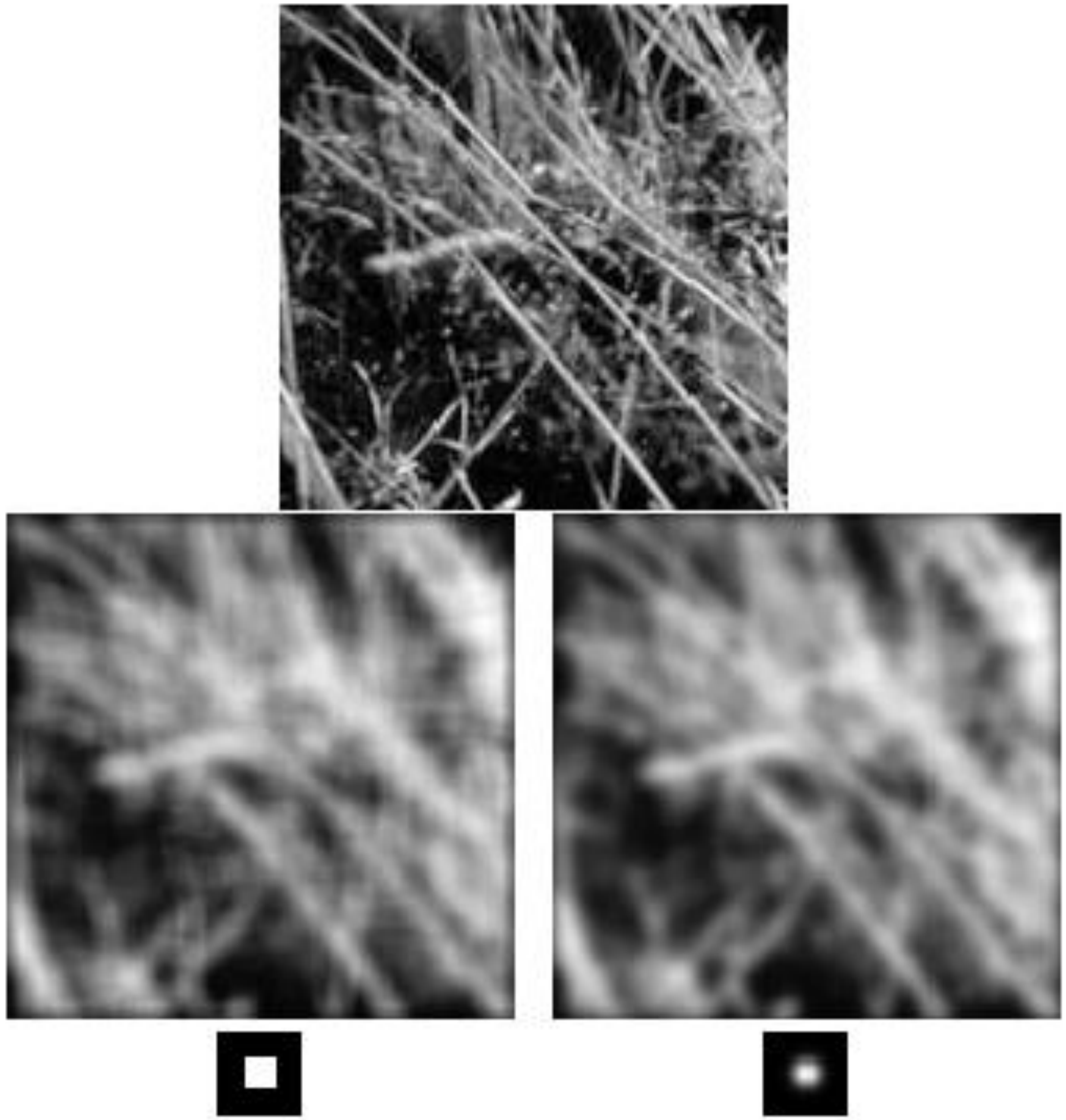


$\sigma\ = 2;\ 30\ x\ 30\ kernel$



$\sigma\ = 5;\ 30\ x\ 30\ kernel$

Gaussian vs. mean (box) filter



Gaussian filter for scale space

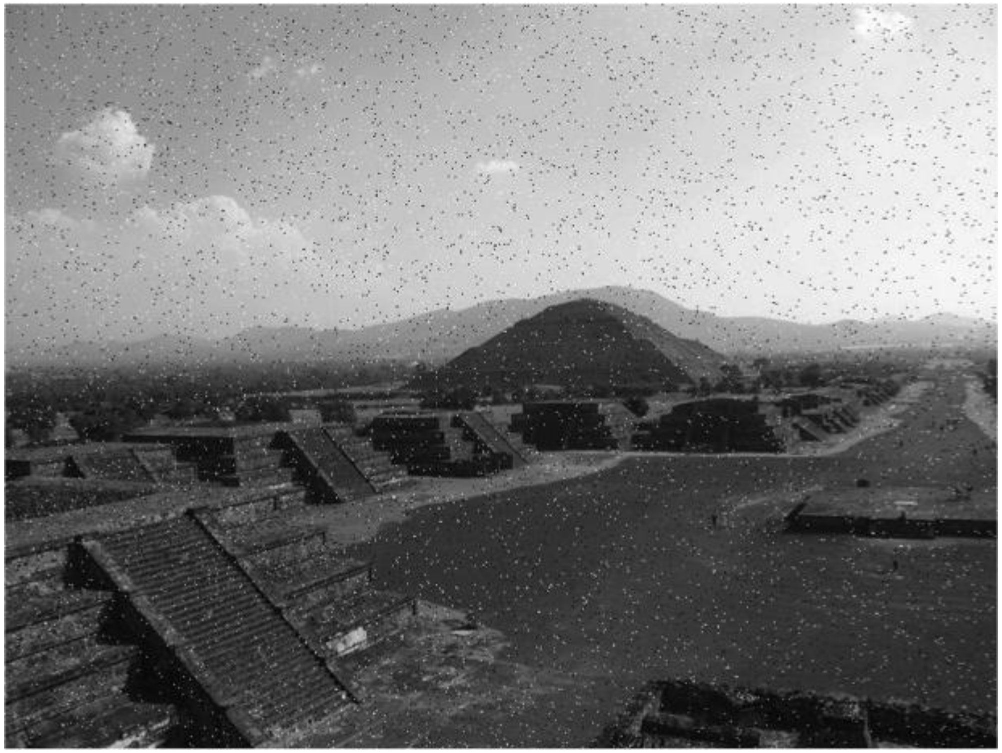
$Orig$

$3x3$

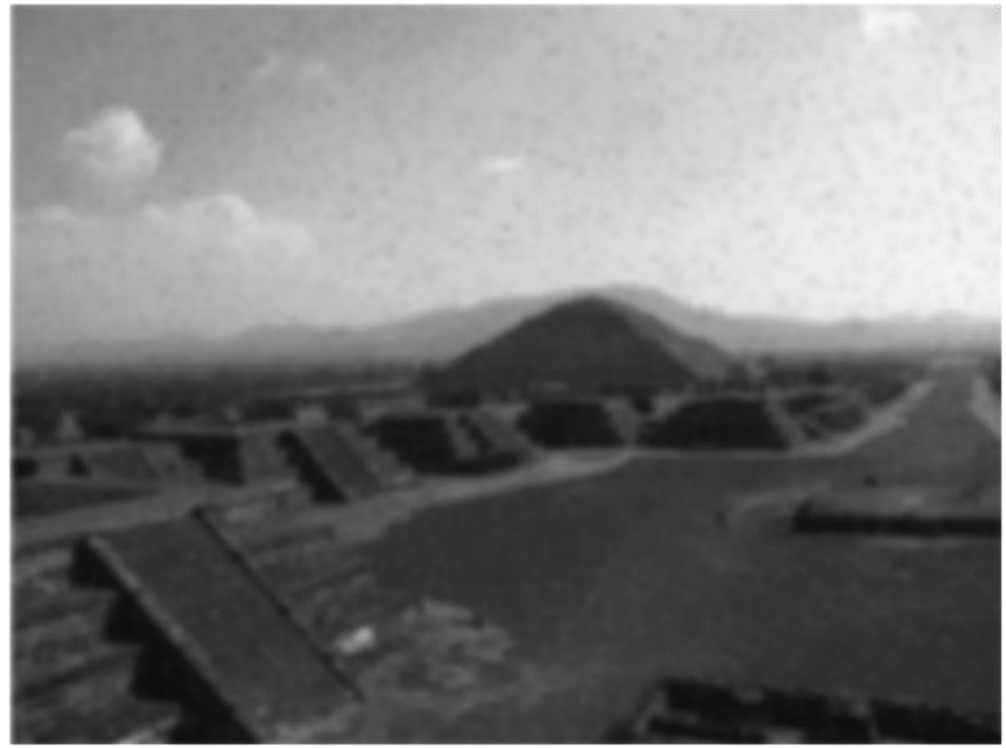
$5x5$

$11x11$

Gaussian filter with Salt-and-pepper noise

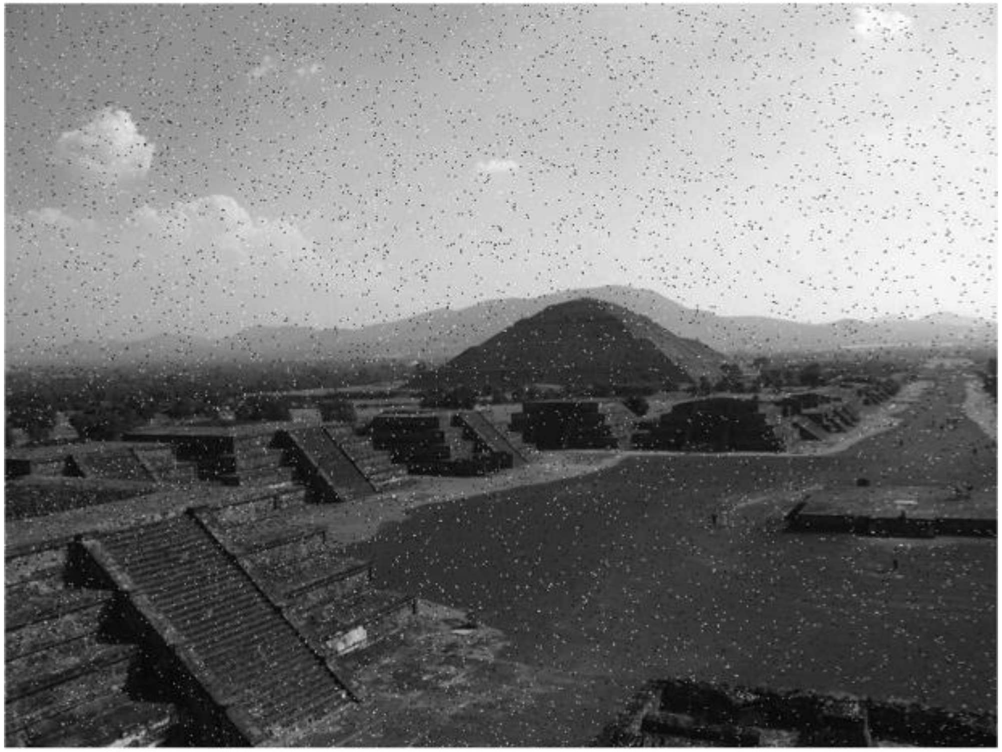


Noisy input image

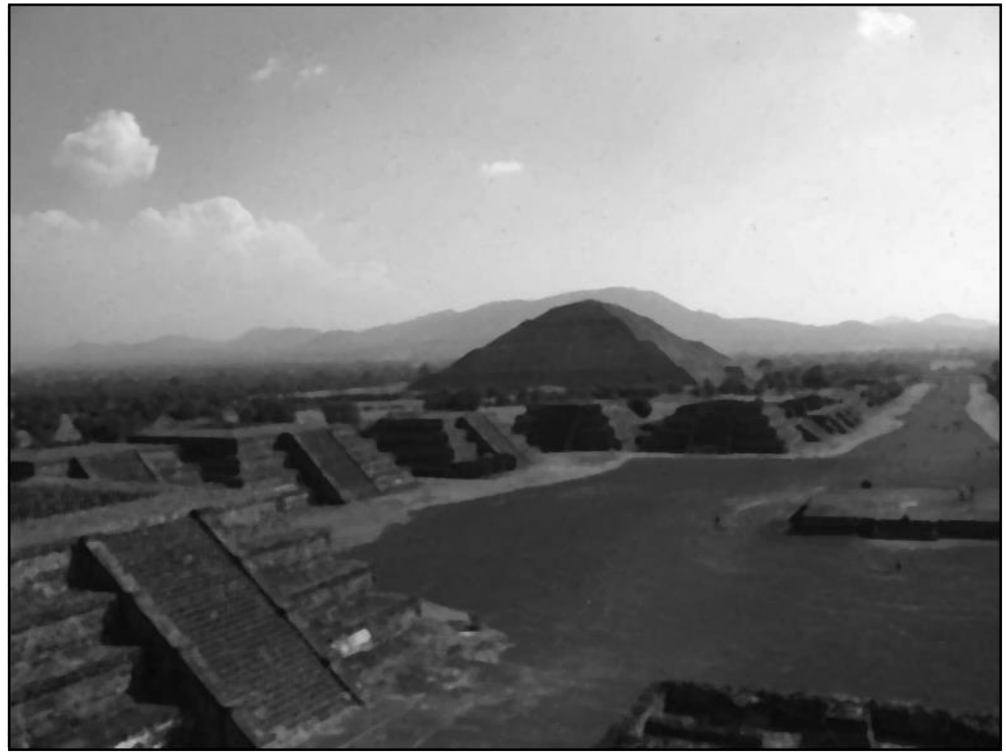


Gaussian filter: blurred edges, residual noise

Median filter



Noisy input image



Median filter output (5x5)

Median filter

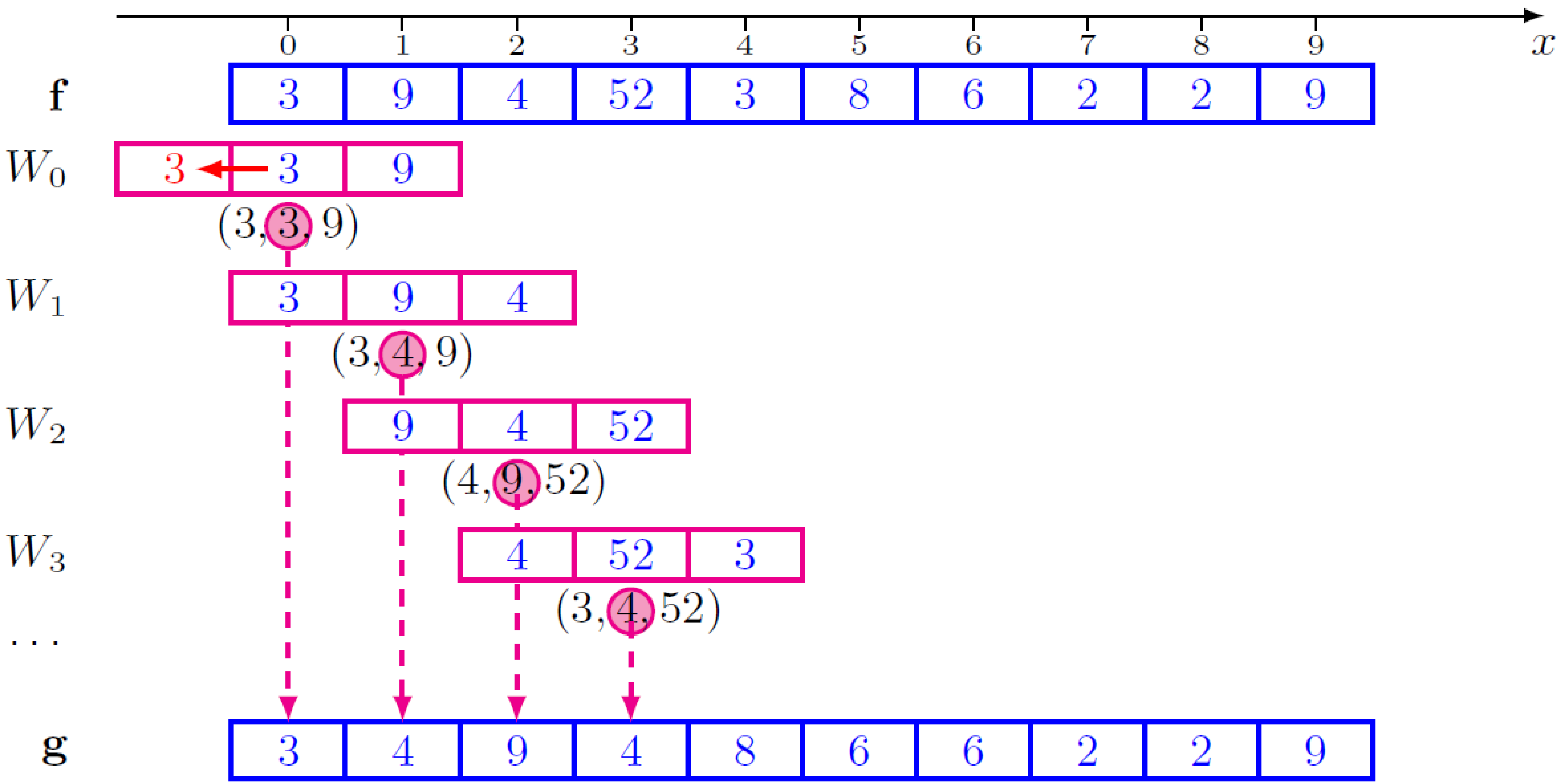
- Effective nonlinear filter, being used frequently to remove “salt-and-pepper” noise while preserving edges
- Replace each pixel with the median of the neighborhood of K pixels:

$$g(x, y) = median\{f(x + \xi_i, y + \eta_i) : i = 1, \dots, K\}$$

- Computation for each pixel (x, y):
  1. Sort all K values from the neighboring window of $f(x, y)$ into ascending numerical order: $v_{[0]} \leq v_{[1]} \leq \cdots \leq v_{[K-1]}$
  2. Select the middle value, $g(x, y) = v_{[K/2]}$, if K is odd or the average, $g(x, y) = 0.5(v_{[(K-1)/2]} + v_{[(K+1)/2]})$, of the two middle values if K is even

Median filter: 1D example

Median filtering of a simple 1D signal $\mathbf{f} = (f(x) : x = 0, 1, \ldots, 9)$ with a moving window $W_x = [x-1, x, x+1]$ of size 3.



Median filter: 1D example

		0	1	2	3	4	5	6	7	8	9	x
f	3 ←	3	9	4	52	3	8	6	2	2	9	→ 9

Moving window		Sorting		Selecting the median and assigning to $g(x)$	
$W_0 : \ (3, 3, 9)$	$\Rightarrow$	$(3, \mathbf{3}, 9)$	$\Rightarrow$	$g(0) = \text{median}\{3, 3, 9\}$	$= 3$
$W_1 : \ (3, 9, 4)$	$\Rightarrow$	$(3, \mathbf{4}, 9)$	$\Rightarrow$	$g(1) = \text{median}\{3, 9, 4\}$	$= 4$
$W_2 : \ (9, 4, 52)$	$\Rightarrow$	$(4, \mathbf{9}, 52)$	$\Rightarrow$	$g(2) = \text{median}\{9, 4, 52\}$	$= 9$
$W_3 : \ (4, 52, 3)$	$\Rightarrow$	$(3, \mathbf{4}, 52)$	$\Rightarrow$	$g(3) = \text{median}\{4, 52, 3\}$	$= 4$
$W_4 : \ (52, 3, 8)$	$\Rightarrow$	$(3, \mathbf{8}, 52)$	$\Rightarrow$	$g(4) = \text{median}\{52, 3, 8\}$	$= 8$
$W_5 : \ (3, 8, 6)$	$\Rightarrow$	$(3, \mathbf{6}, 8)$	$\Rightarrow$	$g(5) = \text{median}\{3, 8, 6\}$	$= 6$
$W_6 : \ (8, 6, 2)$	$\Rightarrow$	$(2, \mathbf{6}, 8)$	$\Rightarrow$	$g(6) = \text{median}\{8, 6, 2\}$	$= 6$
$W_7 : \ (6, 2, 2)$	$\Rightarrow$	$(2, \mathbf{2}, 6)$	$\Rightarrow$	$g(7) = \text{median}\{6, 2, 2\}$	$= 2$
$W_8 : \ (2, 2, 9)$	$\Rightarrow$	$(2, \mathbf{2}, 9)$	$\Rightarrow$	$g(8) = \text{median}\{2, 2, 9\}$	$= 2$
$W_9 : \ (2, 9, 9)$	$\Rightarrow$	$(2, \mathbf{9}, 9)$	$\Rightarrow$	$g(9) = \text{median}\{2, 9, 9\}$	$= 9$

g	3	4	9	4	8	6	6	2	2	9

For $g(0)$ and $g(9)$, $f(0)$ and $f(9)$, respectively, are extended outside the boundaries.

Median filter

Keeping border values unchanged (processing no border pixels).

Sorted: (0, 0, 1, 1, 1, 2, 2, 4, 4)

1	4	0	1	3	1
2	2	4	2	2	3
1	0	1	0	1	0
1	2	1	0	2	2
2	5	3	1	2	5
1	1	4	2	3	0

Input f

1	4	0	1	3	1
2	1	1	1	1	3
1	1	1	1	2	0
1	1	1	1	1	2
2	2	2	2	2	5
1	1	4	2	3	0

Output g

Median filter

1: Extending border values outside with the boundary values.

Sorted: (0, 0, 1, 1, 1, 2, 2, 4, 4)

1	1	4	0	1	3	1	1
1	1	4	0	1	3	1	1
2	2	2	4	2	2	3	3
1	1	0	1	0	1	0	0
1	1	2	1	0	2	2	2
2	2	5	3	1	2	5	5
1	1	1	4	2	3	0	0
1	1	1	4	2	3	0	0

Input f

2	2	2	2	2	2
1	1	1	1	1	1
1	1	1	1	2	2
1	1	1	1	1	2
1	2	2	2	2	2
1	2	2	3	2	2

Output g

Median filter

2: Extending border values outside with zeros.

Sorted: $(0,\ 0,\ 1,\ 1,\ 1,\ 2,\ 2,\ 4,\ 4)$

0	0	0	0	0	0	0	0
0	1	4	0	1	3	1	0
0	2	2	4	2	2	3	0
0	1	0	1	0	1	0	0
0	1	2	1	0	2	2	0
0	2	5	3	1	2	5	0
0	1	1	4	2	3	0	0
0	0	0	0	0	0	0	0

Input f

0	2	1	1	1	0
0	1	1	1	1	1
0	1	1	1	2	1
0	1	1	1	1	1
1	2	2	2	2	2
0	1	1	1	1	0

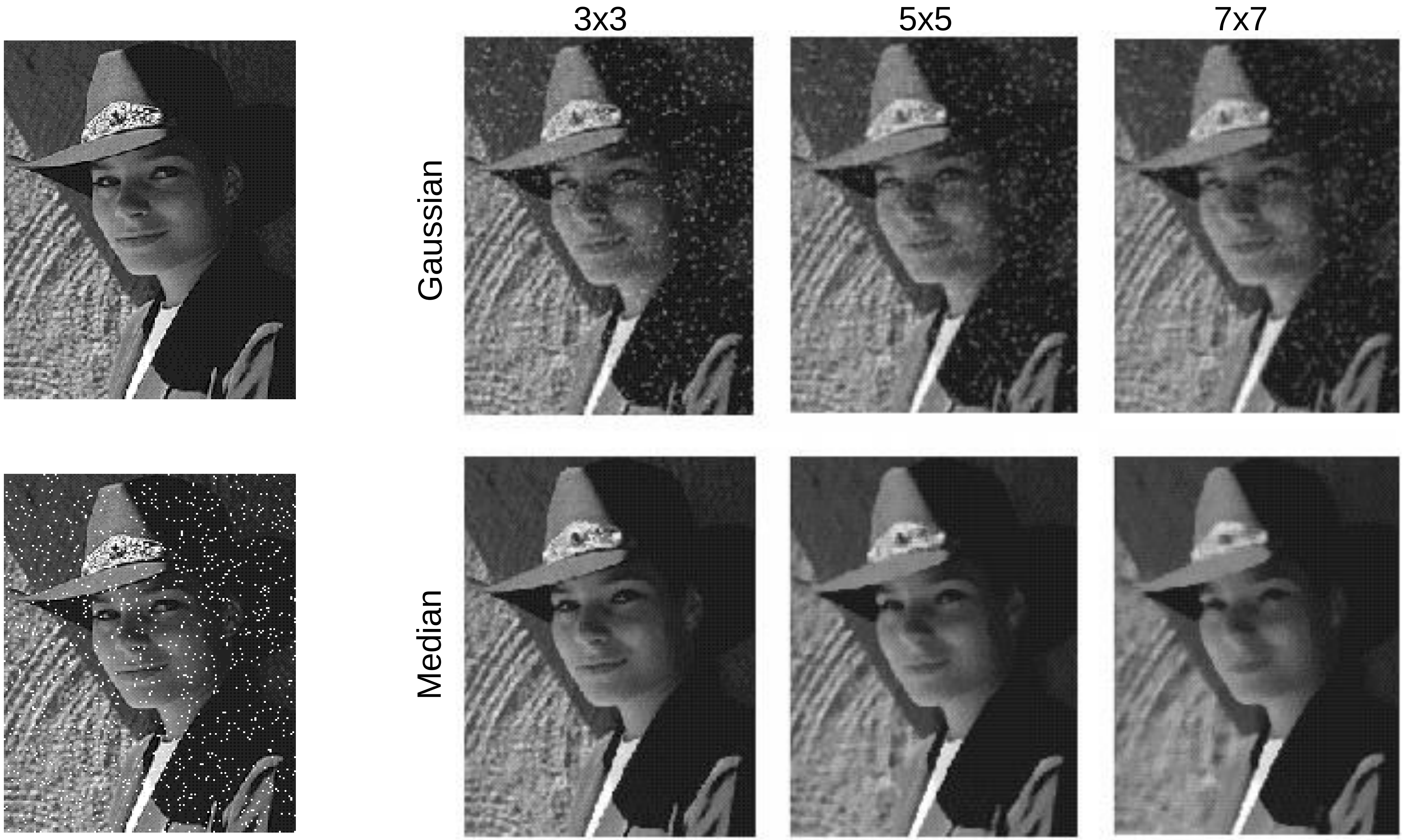
Output g

Gauss vs. median filter

3×3 GF 5×5 GF 11×11 GF

3×3 MF 5×5 MF 11×11 MF

Gauss vs. median filter



Gauss vs. median filter

- Use depends on noise assumption
  - White noise: mean or Gaussian filter
  - Salt-and-pepper noise: median filter (robust to outliers!)
- Local intensity transitions
  - Remain unchanged with mean or Gaussian filter
  - Can get destroyed by median filter
  - Gauss filter: Important for Gaussian scale space (multi-scale representation to inspect image at different complexity levels)