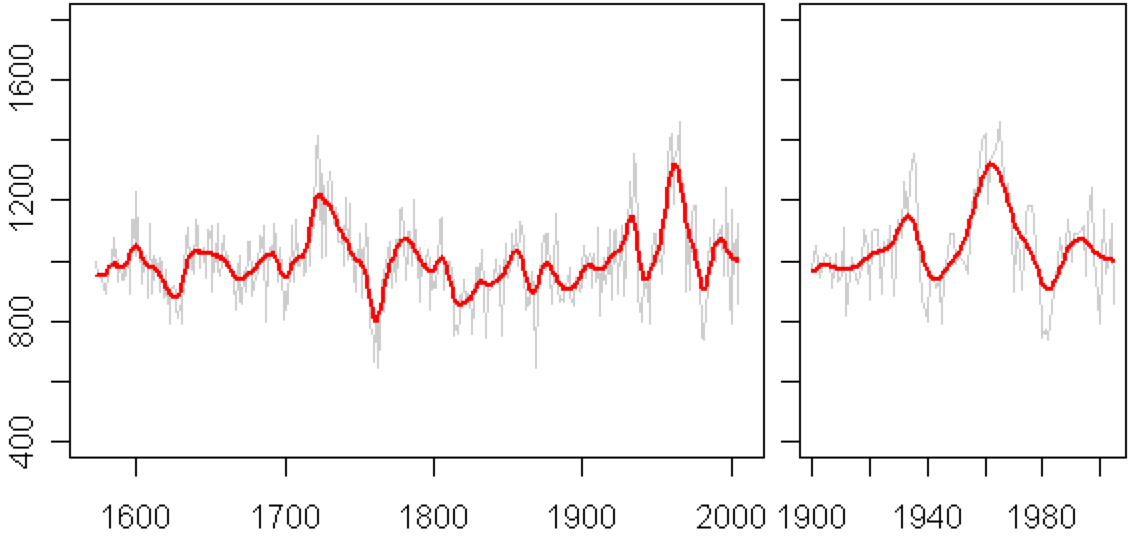
Image Filtering

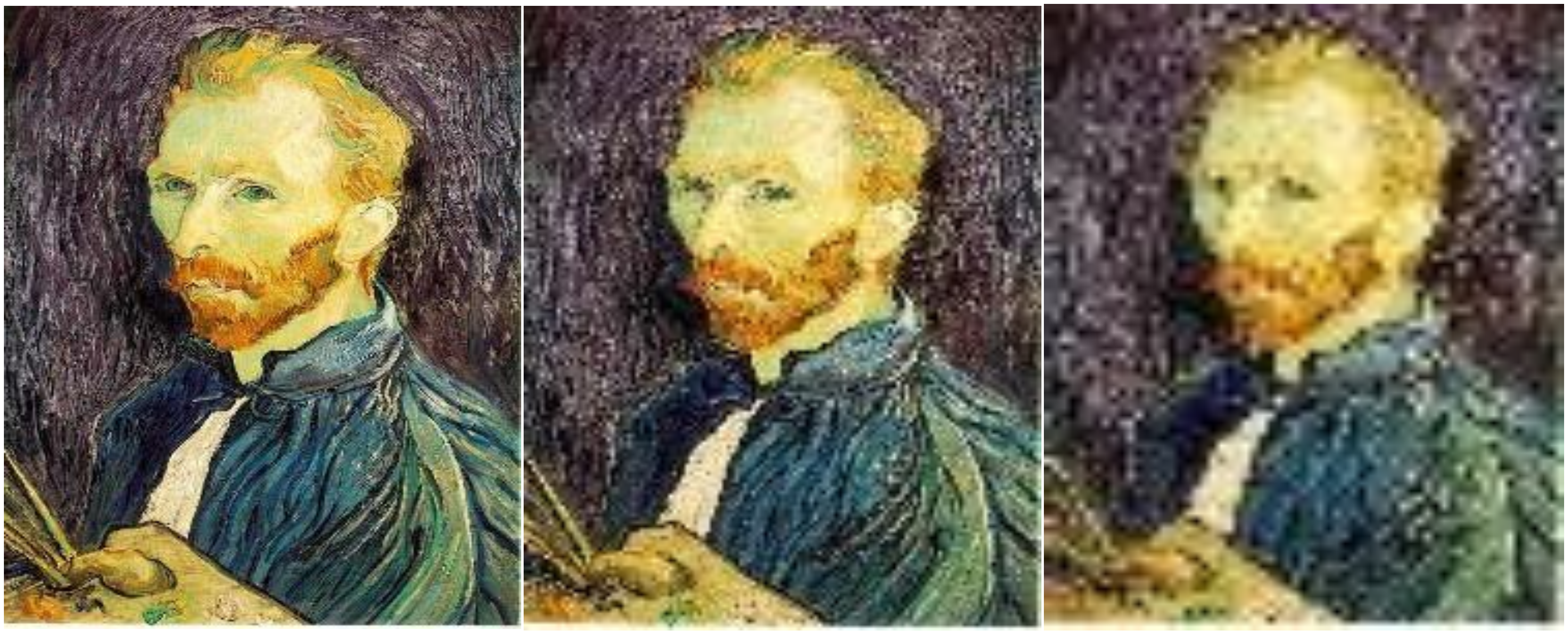
Filtering

- In signal processing, a **filter** is a process that removes from a signal some unwanted component or feature

1D Signal Filtering



2D Image Filtering



2D Image Filtering

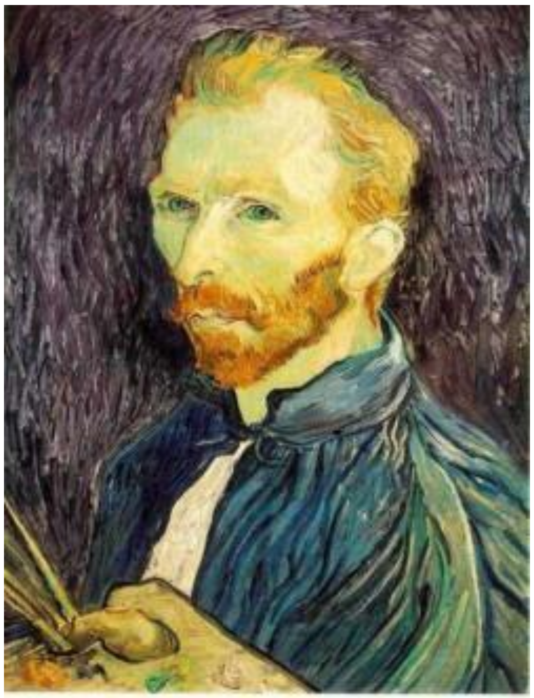
  

Image Filtering

Image filtering: change ***range*** of image

$$g(x) = h(f(x))$$

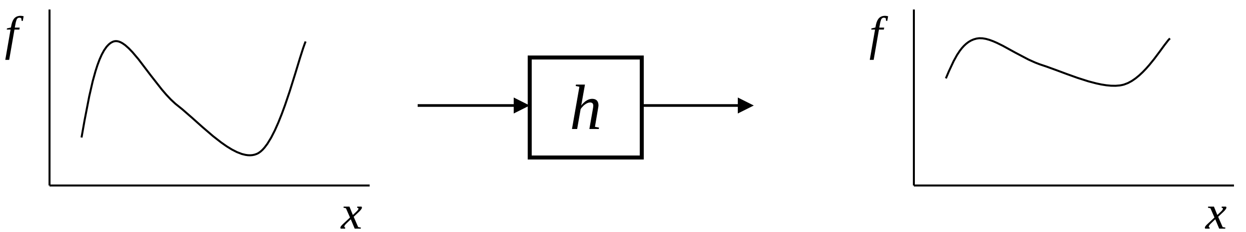


Image warping: change ***domain*** of image

$$g(x) = f(h(x))$$

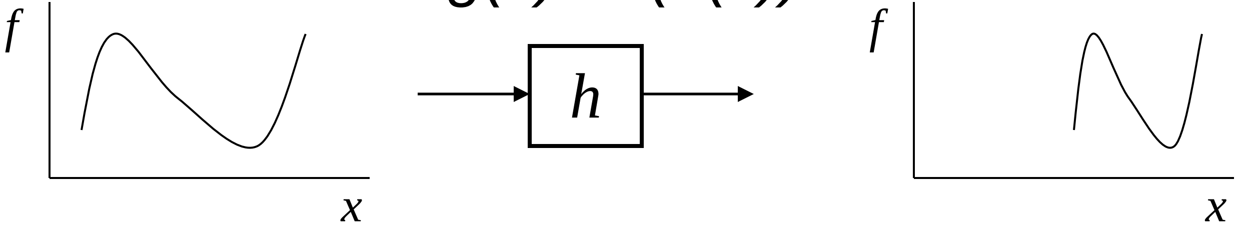


Image Filtering

Image filtering: change ***range*** of image

$$g(x) = h(f(x))$$

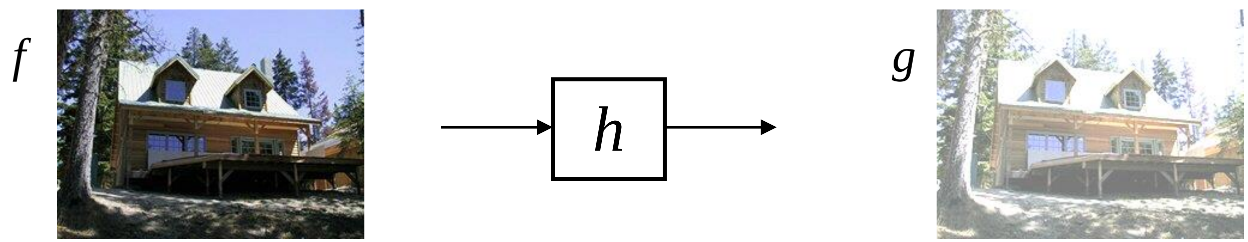
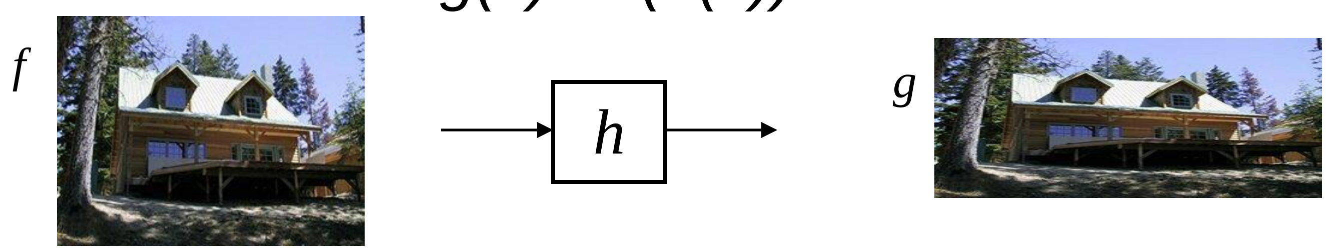


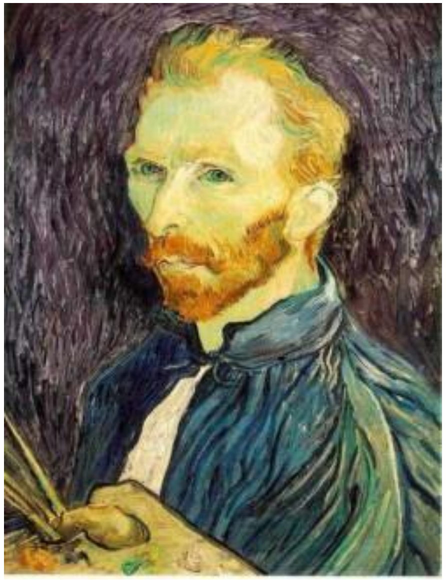
Image warping: change ***domain*** of image

$$g(x) = f(h(x))$$



Filtering in Spatial Domain

=



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

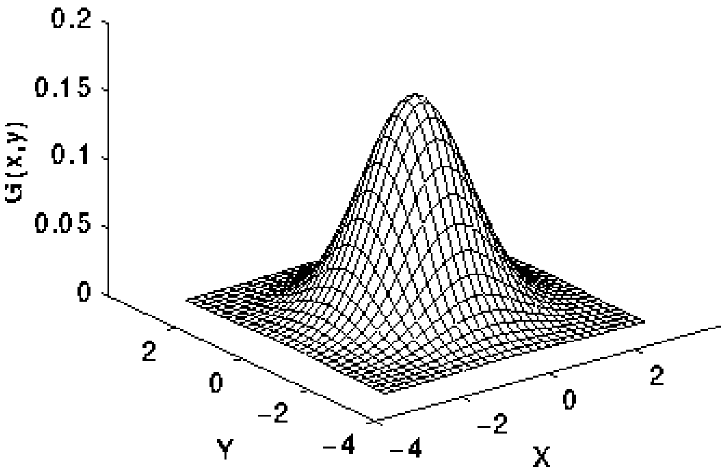
Filtered image

Input image

Filter function

Gaussian Filtering in Spatial Domain

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



$\frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Discrete approximation to Gaussian function with σ =1.0

Filtering in Spatial Domain

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

$\otimes \frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Discrete approximation to Gaussian function with $\sigma = 1.0$

Filtering in Spatial Domain

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	$\boxed{I_{45}}$	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

$$\otimes \quad \frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Discrete approximation to Gaussian function with $\sigma = 1.0$

Filtering in Spatial Domain

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

$$\otimes \quad \frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Discrete approximation to Gaussian function with σ =1.0

Filtering in Spatial Domain

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	$\boxed{I_{45}}$	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}	I_{58}	I_{59}
I_{61}	I_{62}	I_{63}	I_{64}	I_{65}	I_{66}	I_{67}	I_{68}	I_{69}

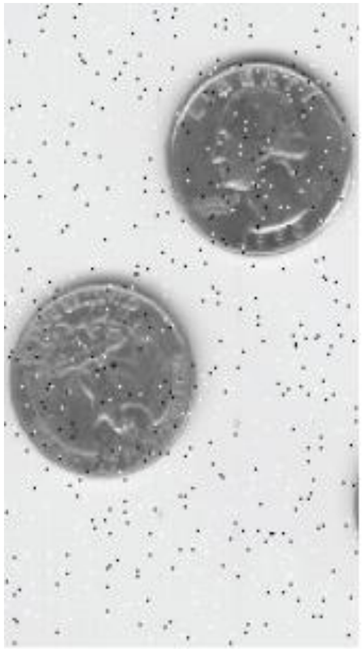
$\otimes \quad \frac{1}{273}$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

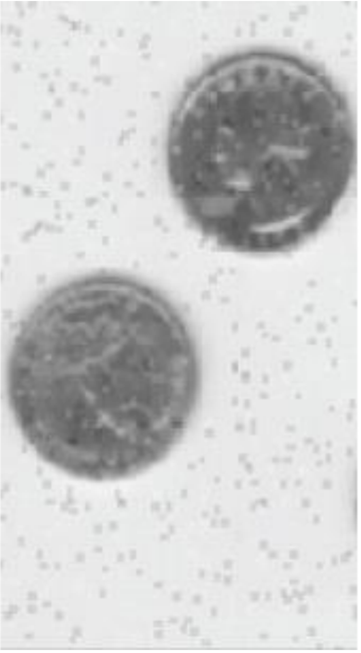
Discrete approximation to Gaussian function with $\sigma = 1.0$

$$\text{Filtered\_I}_{45} = \sum_{pixels \in window} \begin{bmatrix} I_{23} & I_{24} & I_{25} & I_{26} & I_{27} \\ I_{33} & I_{34} & I_{35} & I_{36} & I_{37} \\ I_{43} & I_{44} & I_{45} & I_{46} & I_{47} \\ I_{53} & I_{54} & I_{55} & I_{56} & I_{57} \\ I_{63} & I_{64} & I_{65} & I_{66} & I_{67} \end{bmatrix} \times \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Filtering



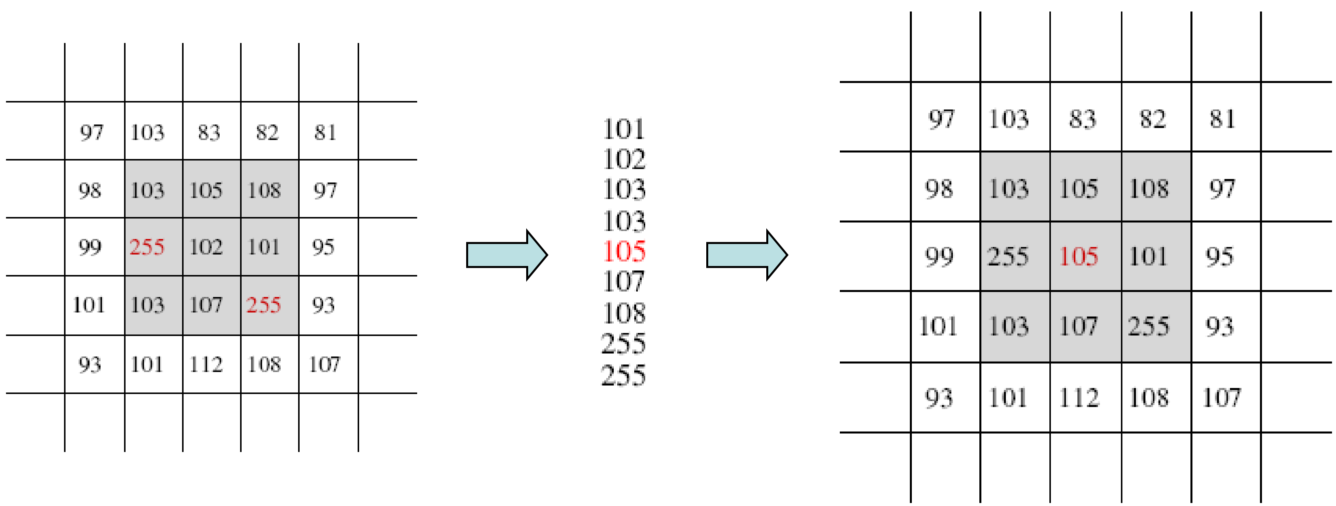
input



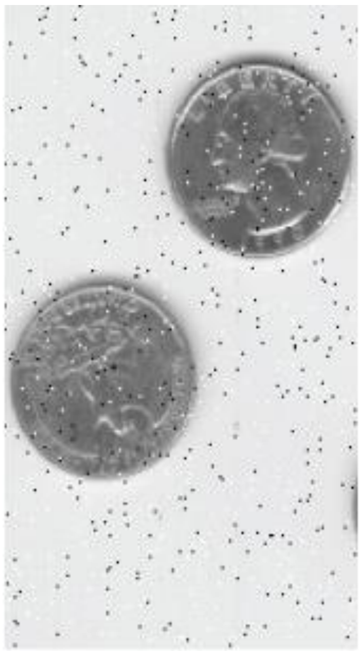
Gaussian filter

Median Filter

- For each neighbor in image, sliding the window
- Sort pixel values
- Set the center pixel to the median



Median Filter



input

Gaussian filter

Median filter

Median Filter Examples

input

Median 7X7

Median Filter Examples

Median 3X3

Median 11X11

Median Filter Examples

Median 3X3 Median 11X11

Straight edges kept

Sharp features lost

Median Filter Properties

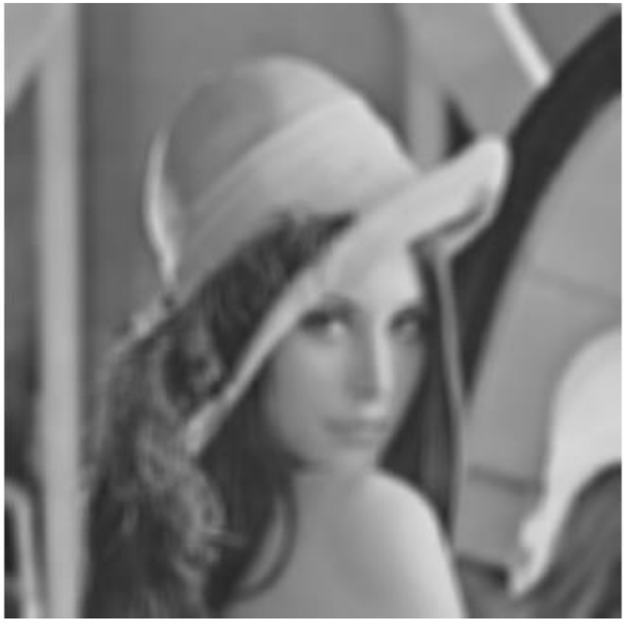
Can remove outliers (peppers and salts)

Window size controls size of structure

Preserve some details but sharp corners and edges might get lost

Comparison of Mean, Gaussian, and Median

original



Mean with 6 pixels

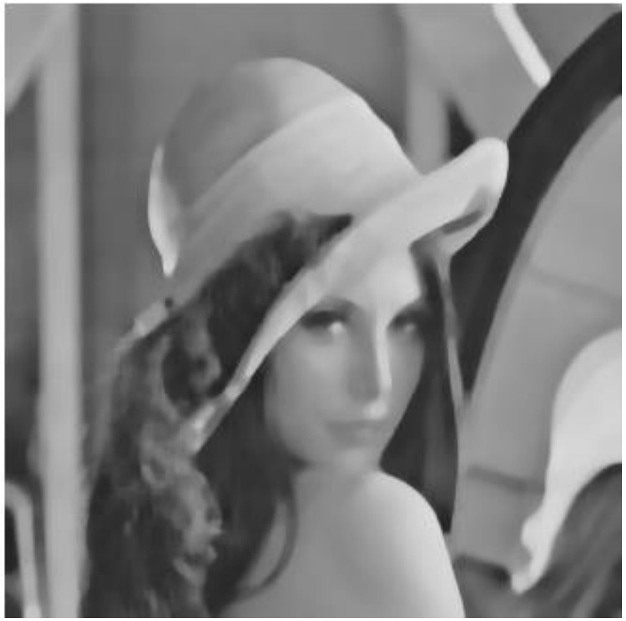
Comparison of Mean, Gaussian, and Median

original

Gaussian with 6 pixels

Comparison of Mean, Gaussian, and Median

original



Median with 6 pixels

Laplacian Filter

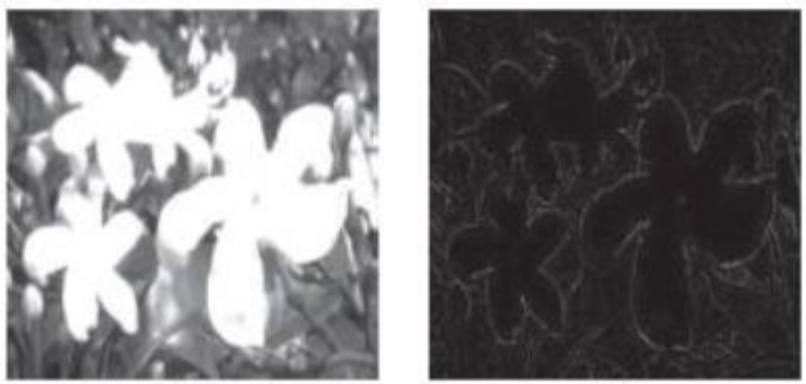
- A **Laplacian filter** is an edge detector used to compute the second derivatives of an image, measuring the rate at which the first derivatives change. This determines if a change in adjacent pixel values is from an edge or continuous progression.

Sharpening Filters

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

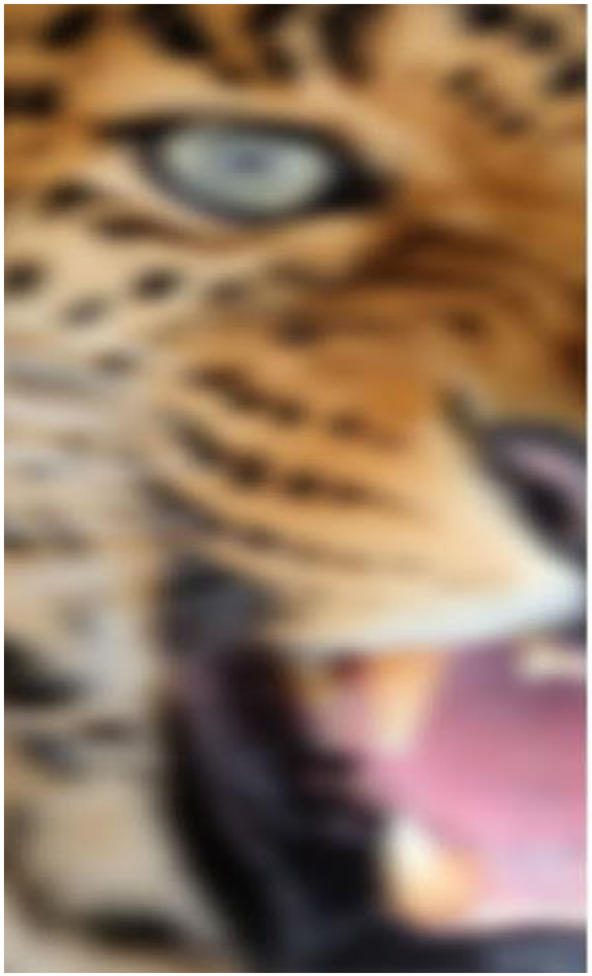
$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Fig. 5.24 Four sample Laplacian masks



(a) (b)

Fig. 5.25 Image sharpening spatial filters (a) Original image (b) Laplacian high-pass filter result




Line Detection

$$M_1 = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}, M_2 = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}, M_3 = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix}, M_4 = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

(a)

Prewitt Edge Operator

-1	-1	-1
0	0	0
-1	-1	-1

Horizontal

-1	0	-1
-1	0	-1
-1	0	-1

Vertical

Sobel Edge Operator

-1	-2	-1
0	0	0
1	2	1

Horizontal

-1	0	1
-2	0	2
-1	0	1

Vertical

High-Boost Filter

$$\text{High-boost image} = (A)\,(\text{Original}) - (\text{Low-pass})$$

$$= (A-1)\,(\text{Original}) + (\text{Original} - \text{Low-pass})$$

$$= (A-1)\,(\text{Original}) + (\text{High-pass})$$

(b)

(b) Result of a high-boost filter

Unsharp Masking

The procedure for implementing an unsharp mask is as follows:

1. Read the image.
2. Blur the image using any image smoothing filters. This stage requires a convolution based smoothing filter. Let the smooth or blurred image be $\bar{f}(x, y)$.
3. Let the mask = original image – $\bar{f}(x, y)$.

   Subtracting the blurred version from the original image results in an image where there is a visible emphasis in edges.
4. Add to the original image the weighted portion of the mask, to restore some of the lost visual information.

$$g(x, y) = f(x, y) + k \times \text{mask}$$