Digital Image Processing

Dr. Vrijendra Singh

Indian Institute of Information Technology Allahabad

Outline

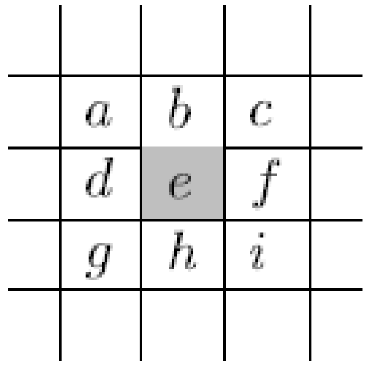
- Spatial Filtering
  - Averaging
  - Smoothing,
  - Sharpening filters
- Noise
- Edges

Spatial Filtering

- Apply function to the neighborhood of each pixel
- Like moving a mask/shape (e.g. rectangle) over the image
- Combination of mask and function : filter
- If it is linear function of all grey values: linear filter e.g. averaging, Laplacian filters etc.

Averaging Filter

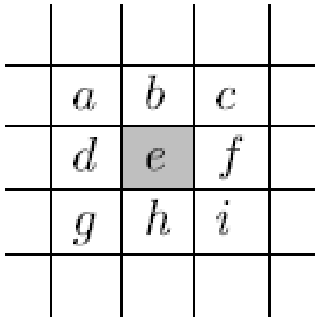
- The 3x3 average filter:



$$\longrightarrow \frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

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

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$



$$\longrightarrow a - 2b + c - 2d + 4e - 2d + g - 2h + i$$

Edge Problem

- Ignore the edges
- Pad with zero

- Check- filter2 in matlab with valid, same and full attribute
- filter2(‘average’,image,’full’)

High and Low frequency Component

- **High frequency components** are characterized by large changes in grey values over small distances;
- Example of high frequency components are edges and noise.
- **Low frequency components** are parts of the image characterized by little change in the grey values.
- Example backgrounds, skin textures etc.

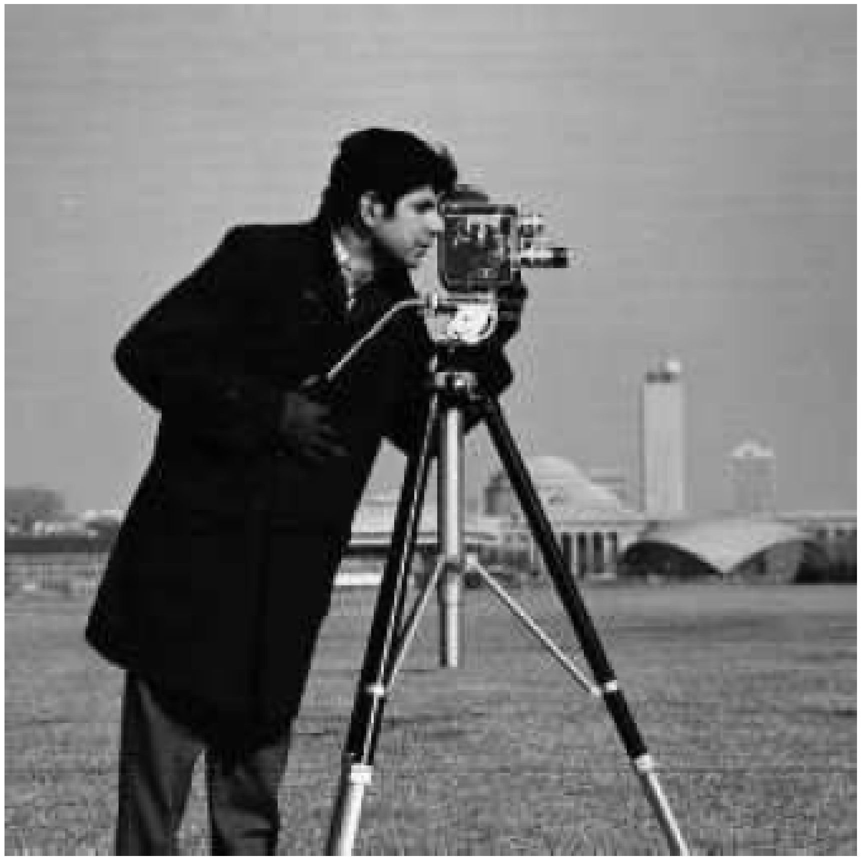
High/Low Pass Filter

- **High Pass Filter/Sharpening Filter:** reduces low frequency components
- **Low Pass Filter/Smoothing Filter:** reduces high frequency components
- E.g. of low pass filter: 3x3 averaging filter as it tends to blur edges
- $$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

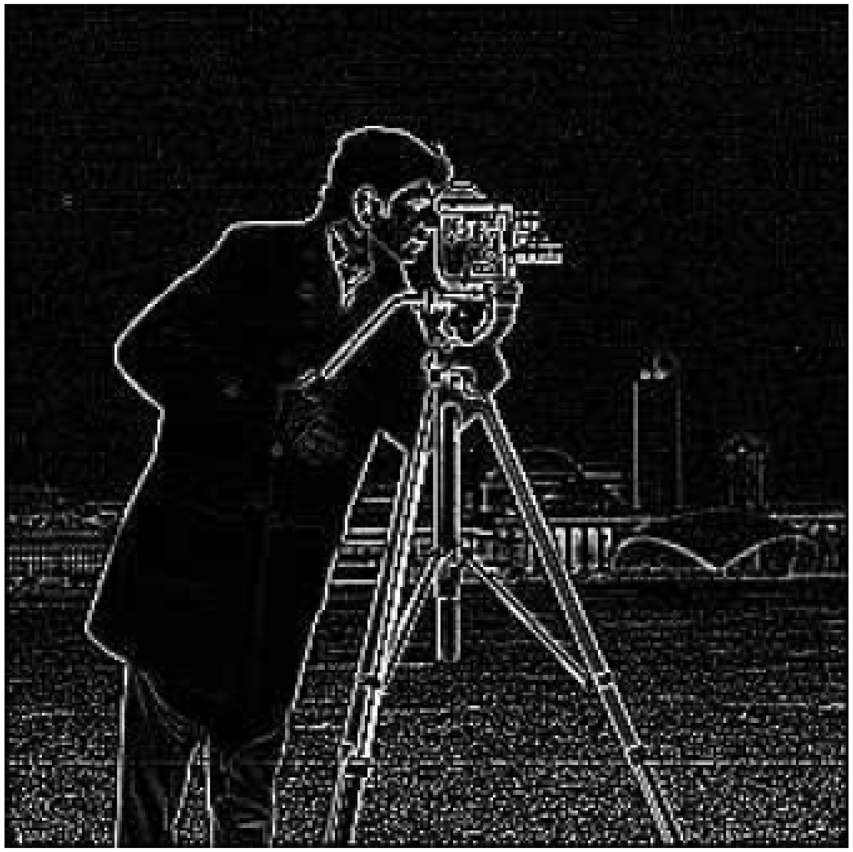
  is a high pass filter. Since the sum is zero. This means that in a low frequency part of an image, where the grey values are similar, the result of using this filter is that the corresponding grey values in the new image will be close to zero.

High Pass Filter

Original Image



After applying Laplacian Filter



Three discrete approximations to the Laplacian filter

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

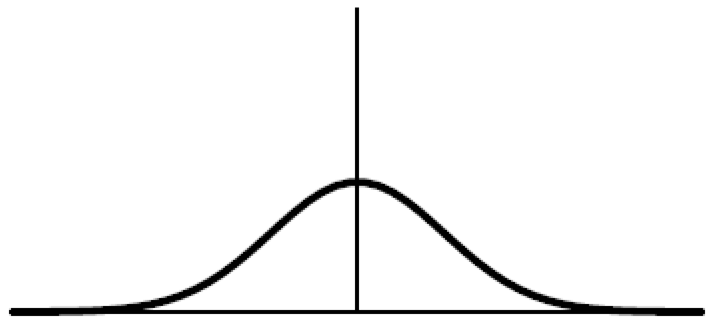
1	-2	1
-2	4	-2
1	-2	1

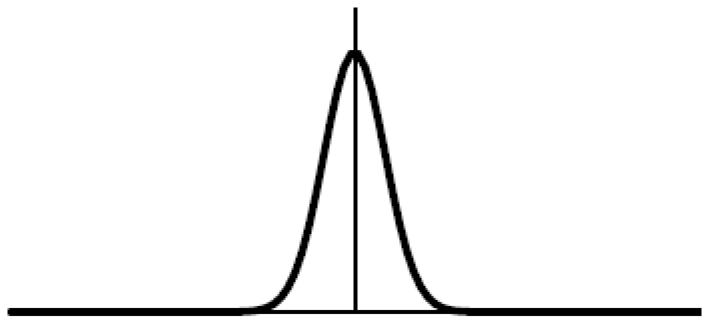
Out of range value handling

- Take <0 values =0 and >255 values = 255 rest as it is
- Use Normalization (mat2gray)

Non Linear Filter: Gaussian Filter

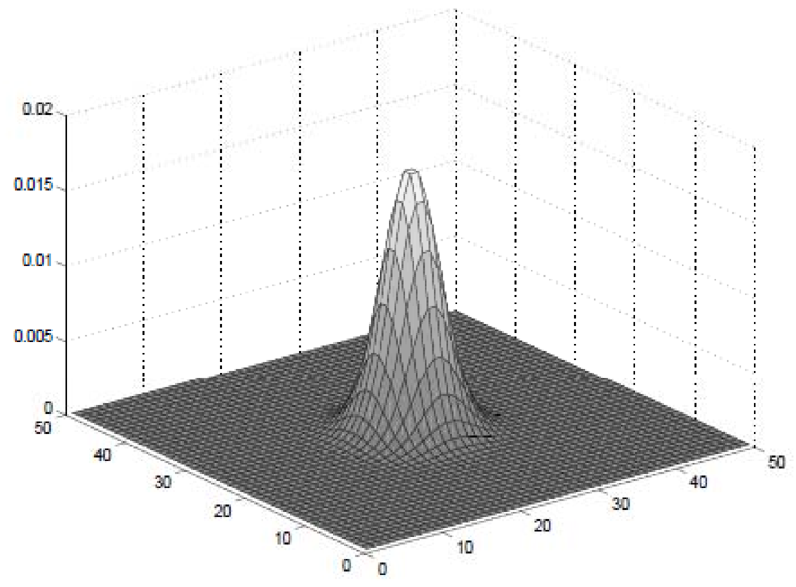
- Gaussian filters are a class of low-pass filters, all based on the Gaussian probability distribution function with sigma as standard deviation $f(x) = e^{-\frac{x^2}{2\sigma^2}}$


Large value of σ

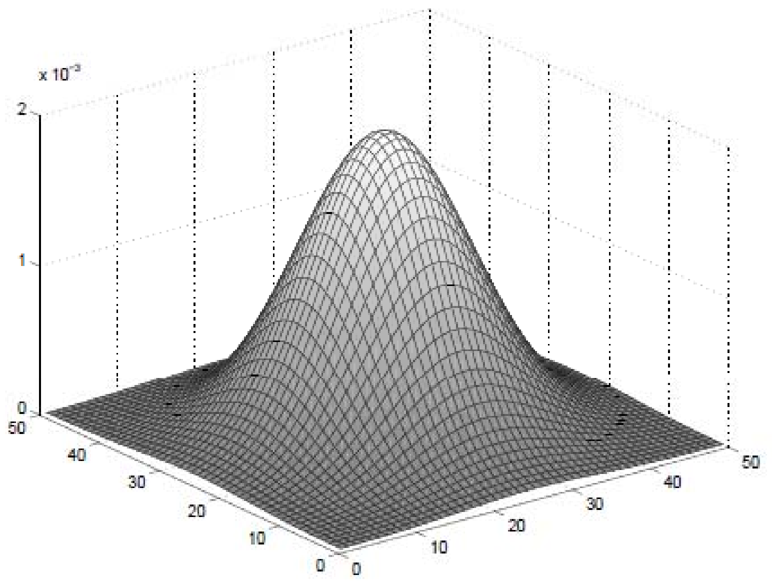

Small value of σ

Two dimensional Gaussian Filter

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

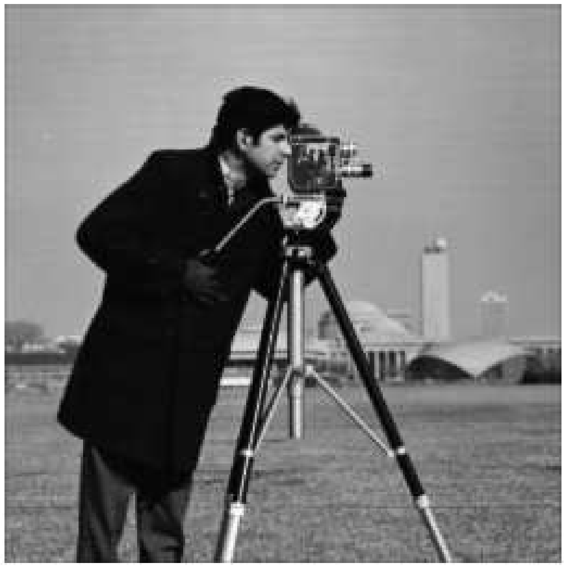


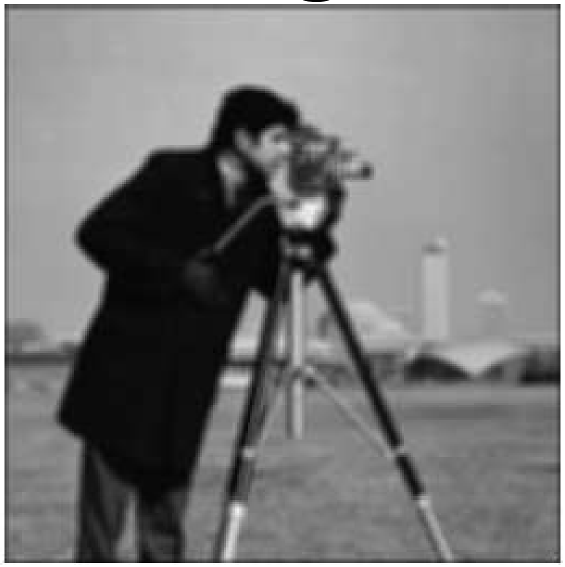
$\sigma = 3$

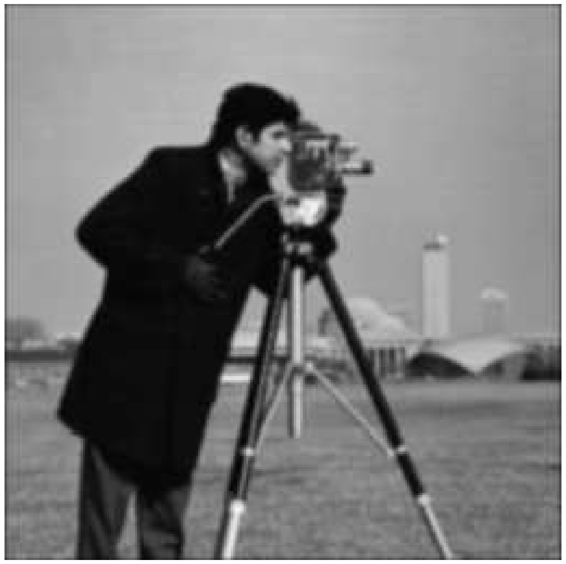


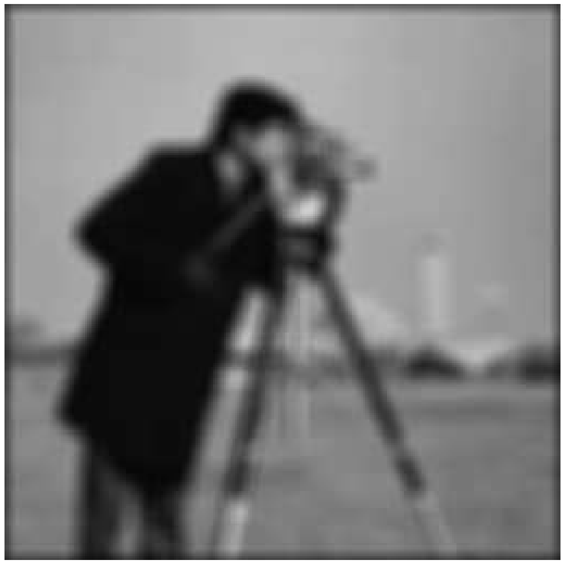
$\sigma = 9$

Gaussian Filter E.g.


$5 \times 5, \sigma = 0.5$

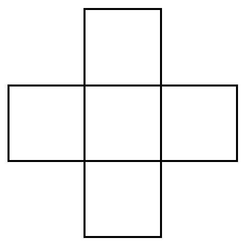

$5 \times 5, \sigma = 2$


$11 \times 11, \sigma = 1$


$11 \times 11, \sigma = 5$

Non linear filter: Rank Order Filter

- Order the set and take the n-th value
- It allows us to choose the median of non – rectangular mask like:



- Median is the special case of Rank Order

Non linear filter: Median Filter

- Median is the mean of the middle two

$$\begin{array}{|c|c|c|}\hline 50 & 65 & 52 \\ \hline 63 & 255 & 58 \\ \hline 61 & 60 & 57 \\ \hline \end{array} \longrightarrow 50 \quad 52 \quad 57 \quad 58 \quad \boxed{60} \quad 61 \quad 63 \quad 65 \quad 255 \longrightarrow 60$$

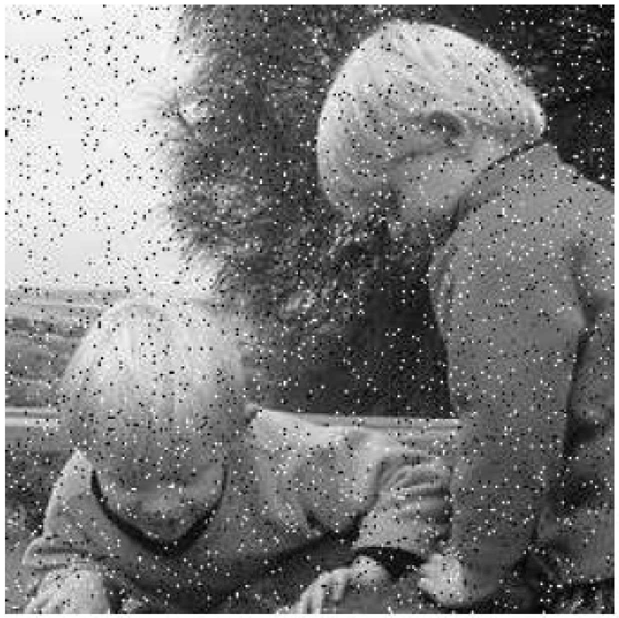
Noise

- Degradation in the image signal, caused by external disturbance
- Cause may be transmission through network
- Types of noise:
  - Salt & Pepper Noise
  - Gaussian Noise
  - Speckle Noise
  - Periodic Noise

Salt &Pepper Noise

- Impulse noise, shot noise, or binary noise
- This degradation can be caused by sharp, sudden disturbances in the image signal;
- Its appearance is randomly scattered white or black (or both) pixels over the image.

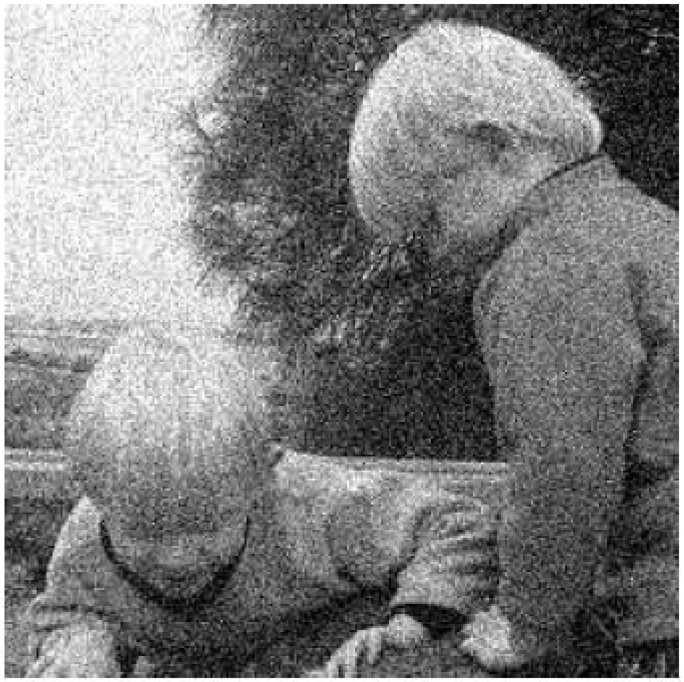
(a) Original image


(b) With added salt & pepper noise

Gaussian Noise/Additive Noise

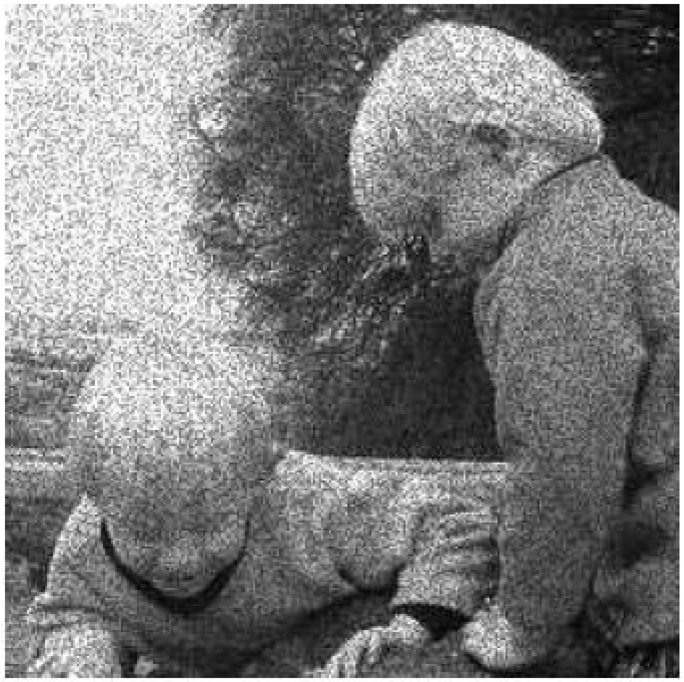
- Gaussian noise is an idealized form of white noise, which is caused by random fluctuations in the signal.
- We can observe white noise by watching a television which is slightly mistuned to a particular channel.
- Gaussian noise is white noise which is normally distributed (= I + N simple add)

(a) Original image

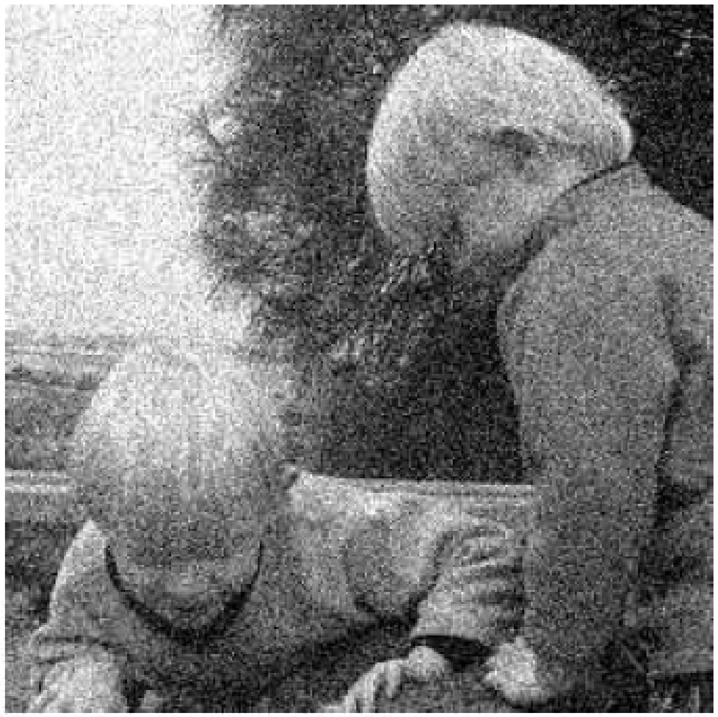

(b) Gaussian noise

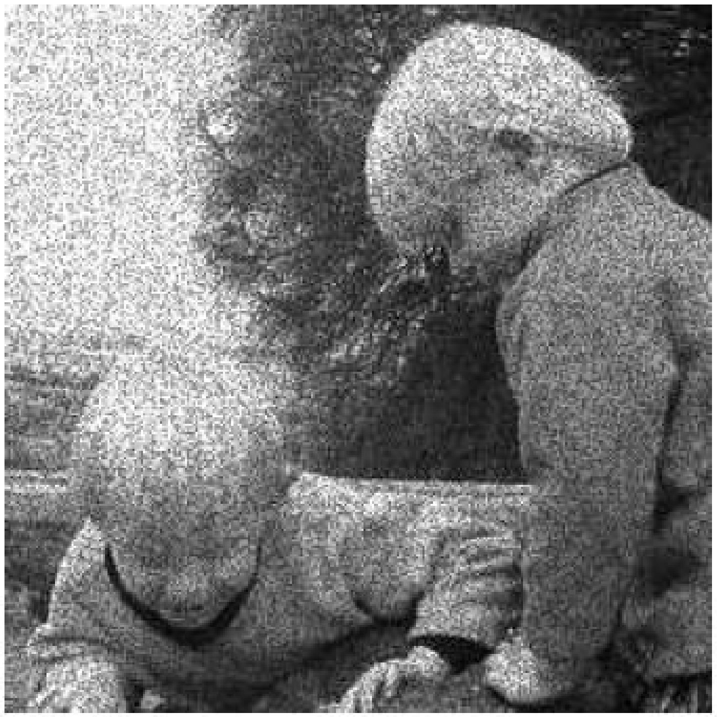
Speckle/Multiplicative Noise

(a) Original image


(b) Speckle noise

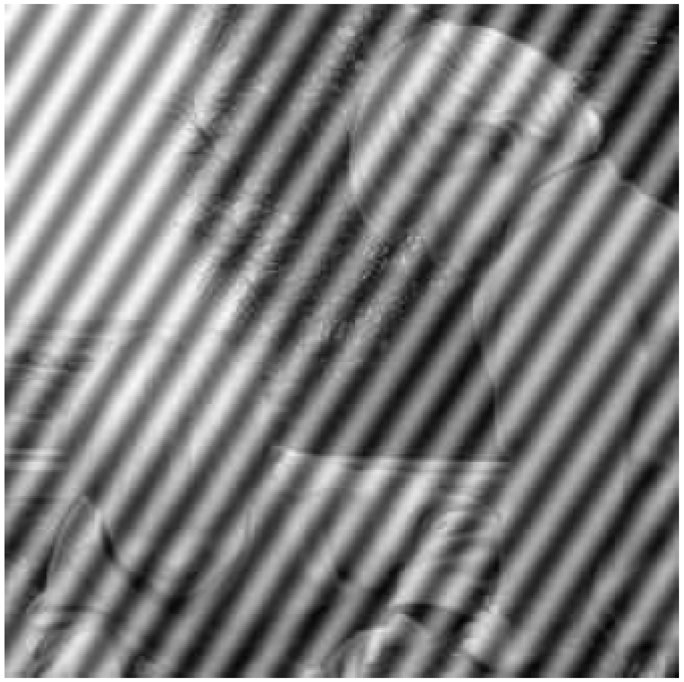
Comparison


(a) Gaussian noise

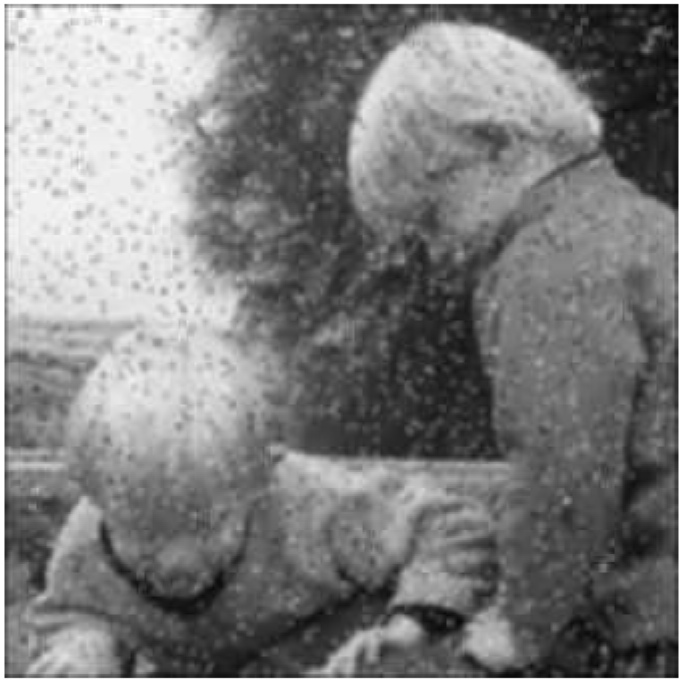

(b) Speckle noise

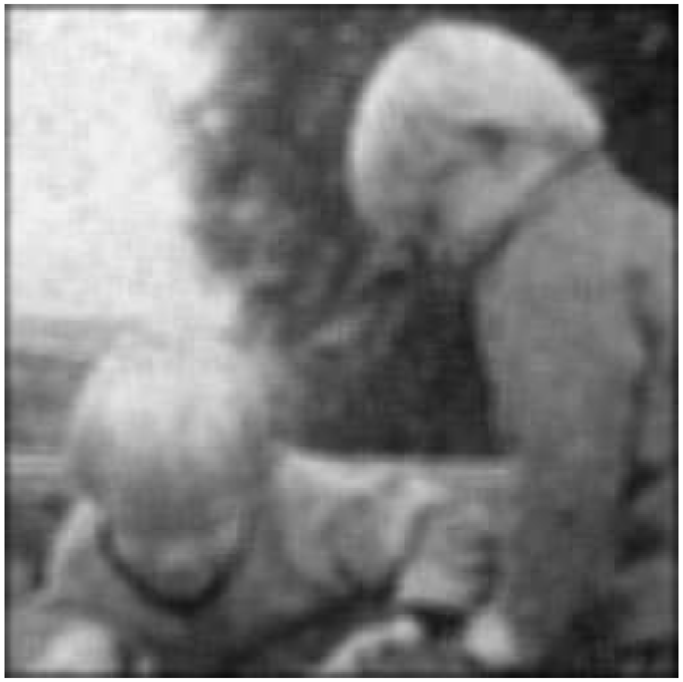
Periodic Noise

- If the image signal is subject to a periodic, rather than a random disturbance, we might obtain an image corrupted by periodic noise.
- The effect is of bars over the image.



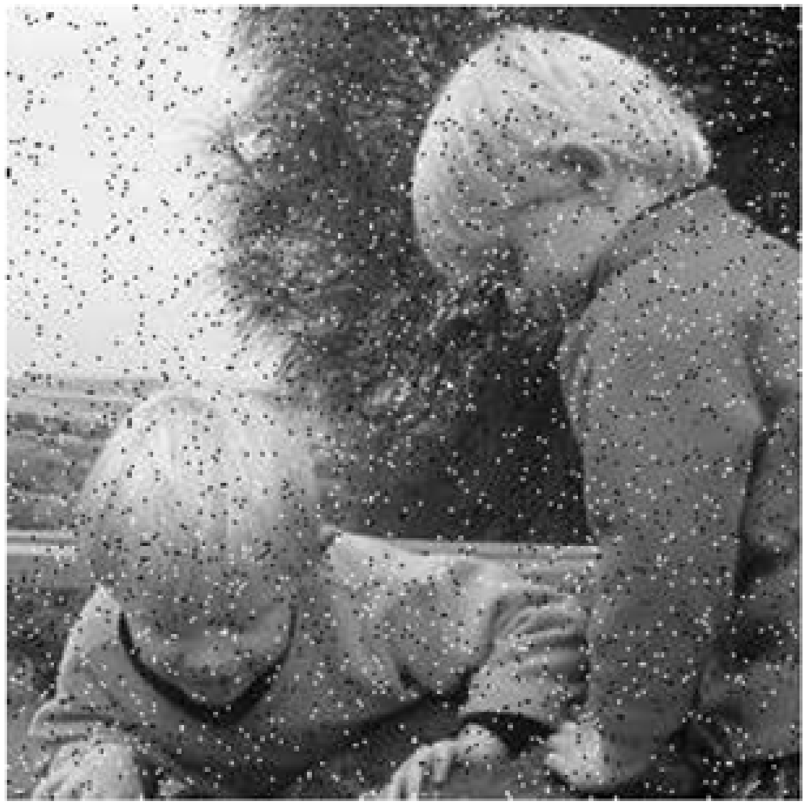
Denoising: Salt & Pepper Noise

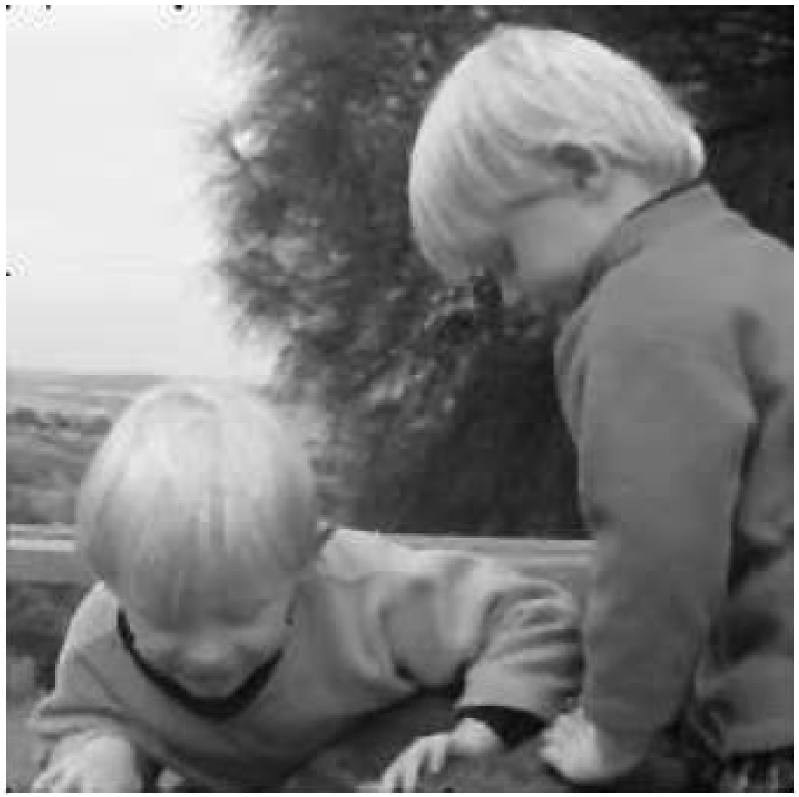

(a) 3 × 3 averaging


(b) 7 × 7 averaging

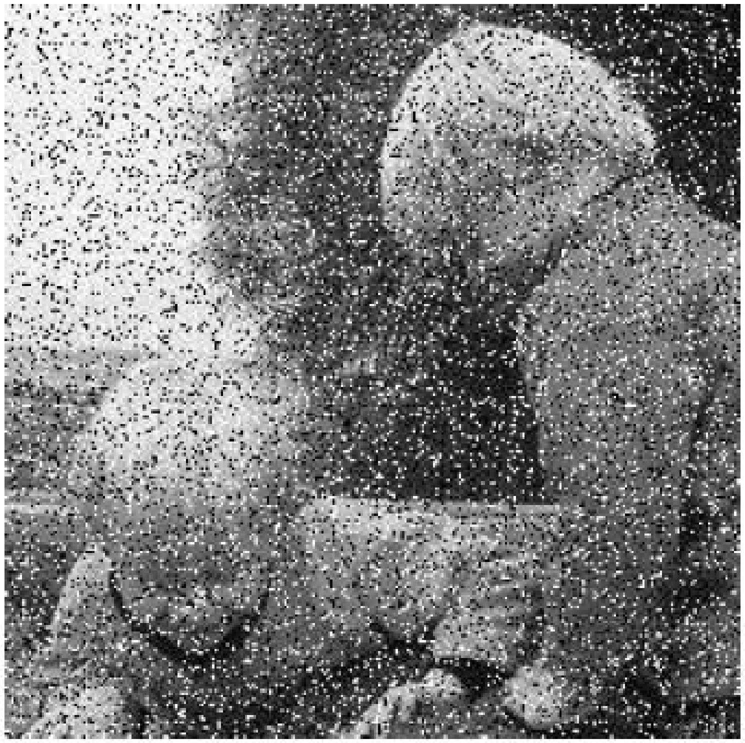
Attempting to clean salt & pepper noise with average filtering

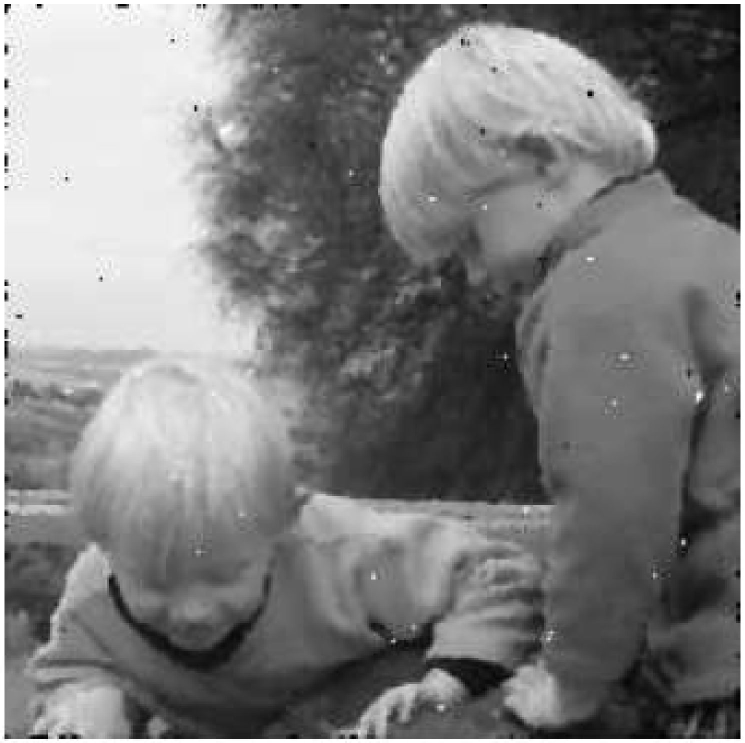
Denoising: Salt & Pepper Noise


With added salt & pepper noise


Cleaning salt and pepper noise with a median filter

Denoising: Salt & Pepper Noise

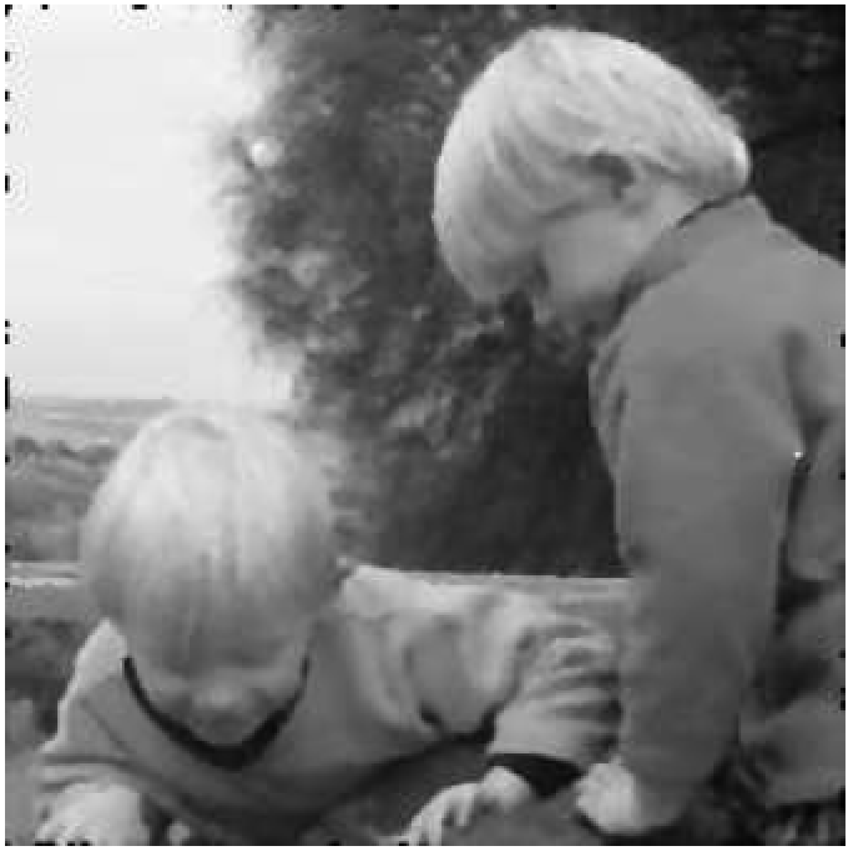

(a) 20% salt & pepper noise


(b) After median fitering

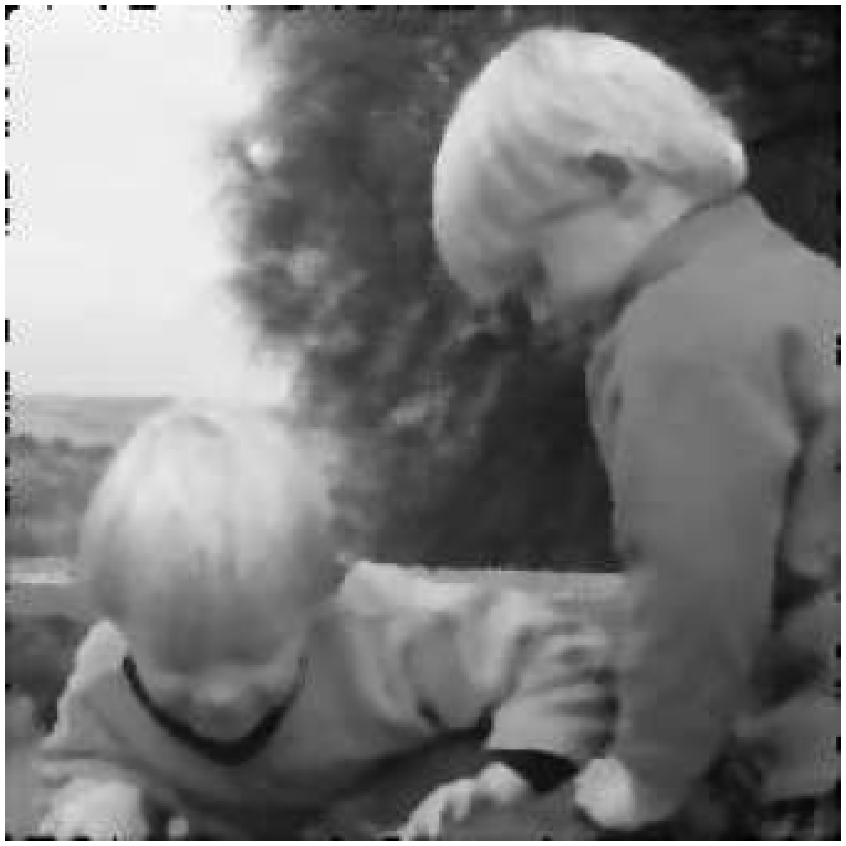
Using a 3 × 3 median filter on more noise

Denoising: Salt & Pepper Noise

Using twice median filter

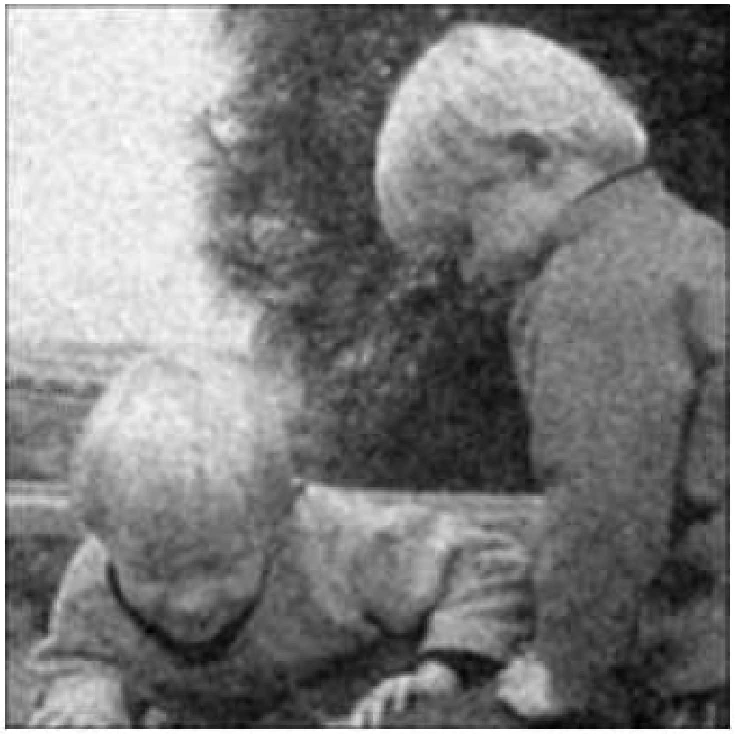


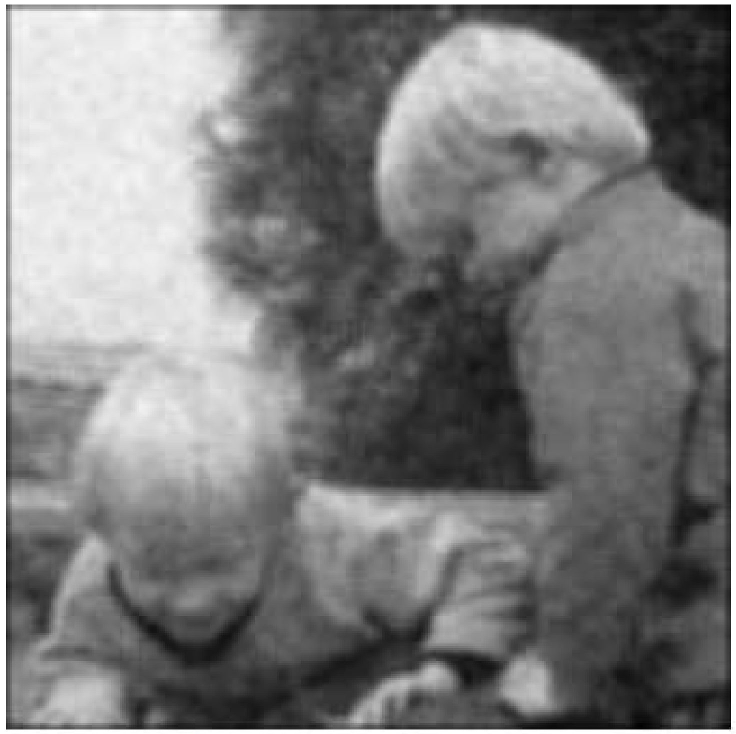
Using a 5x5 median filter



Denoising: Gaussian Noise

- Averaging Filter


(a) 4×3 averaging


(b) 5×5 averaging

Non Linear Filter: Weiner Filter

- If M is the original image and R is the restored image, then, Least Square Error can be given by

$$\sum (m_{i,j} - r_{i,j})^2$$

- It is also the measure of the closeness between the two images
- If we can minimize this value, we may be sure that our procedure has done a very good job.
- Filters which operate on this principle of least squares are called Wiener filters.

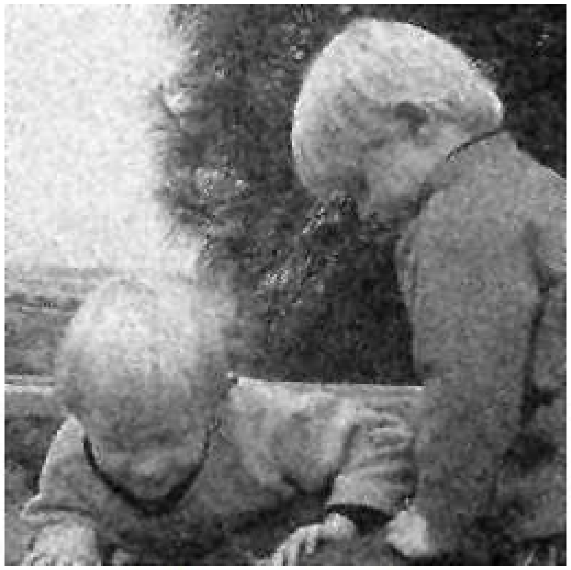
Weiner Filter

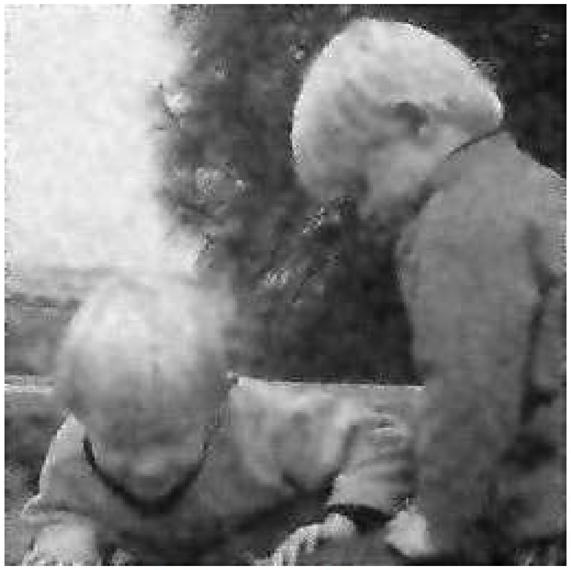
$$m_f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_g^2}(g - m_f)$$

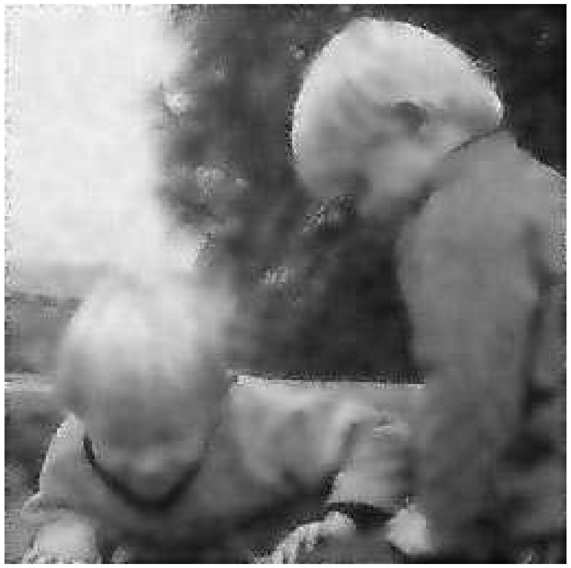
- Were m_f & sigma_f is the mean and variance of the mask and sigma_g is the variance of the noise over the entire image
- Since finding the sigma_g is difficult, Wiener filtering uses a slight variant of the above as:

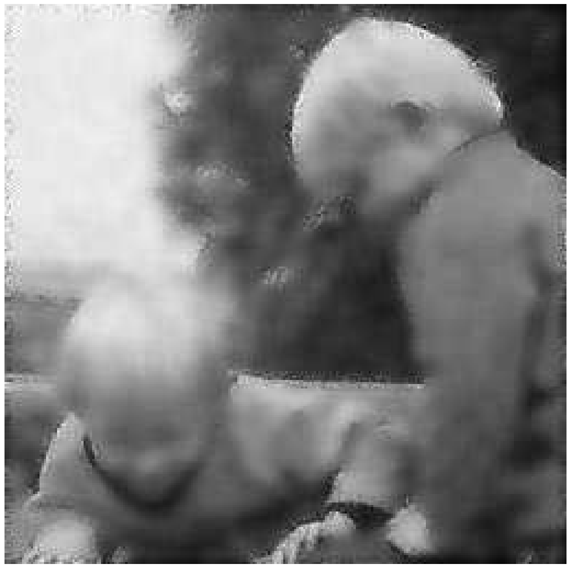
$$m_f + \frac{\max\{0, \sigma_f^2 - n\}}{\max\{\sigma_f^2, n\}}(g - m_f)$$

- Where n is the computed noise variance, and is calculated by taking the mean of all values of sigma_f square over the entire image


(a) 3×3 filtering


(b) 5×5 filtering


(a) 7×7 filtering


(b) 9×9 filtering

Examples of Wiener filtering to remove Gaussian noise

Edges

- An edge may be loosely defined as
  - a line of pixels showing an observable difference

51	52	53	59
54	52	53	62
50	52	53	68
55	52	53	55

50	53	150	160
51	53	150	170
52	53	151	190
51	53	152	155

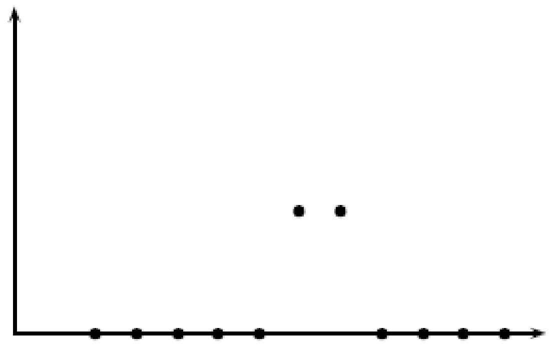
- Human eye can pick out grey differences of this magnitude with relative ease

Differences Measure for Edges

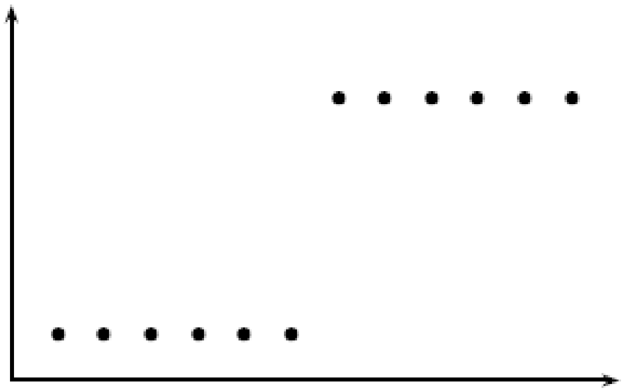
- The differences between each grey value and its predecessor from the ramp edge
- Suppose if the pixel values are as follows:

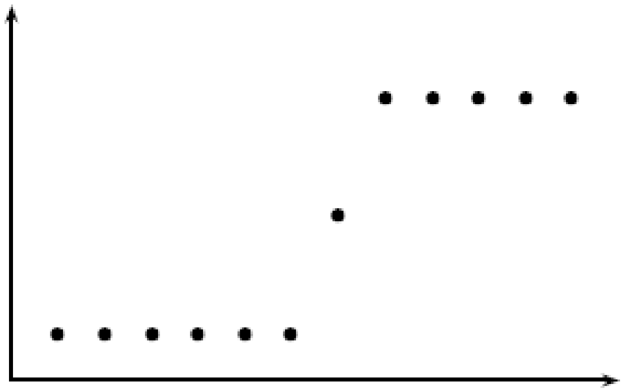
  [20 20 20 20 20 20 100 180 180 180 180 180]
- The difference Metrics can be constructed as follows:

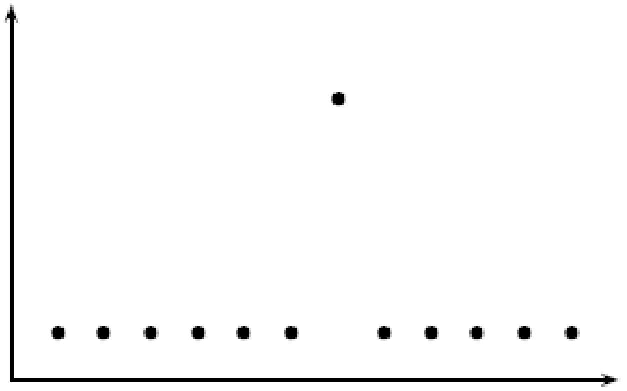
  [0 0 0 0 0 80 80 0 0 0 0]

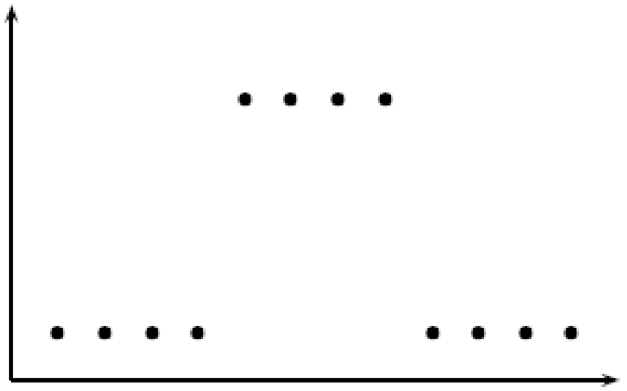


Differences of the edge function


(a) A “step” edge

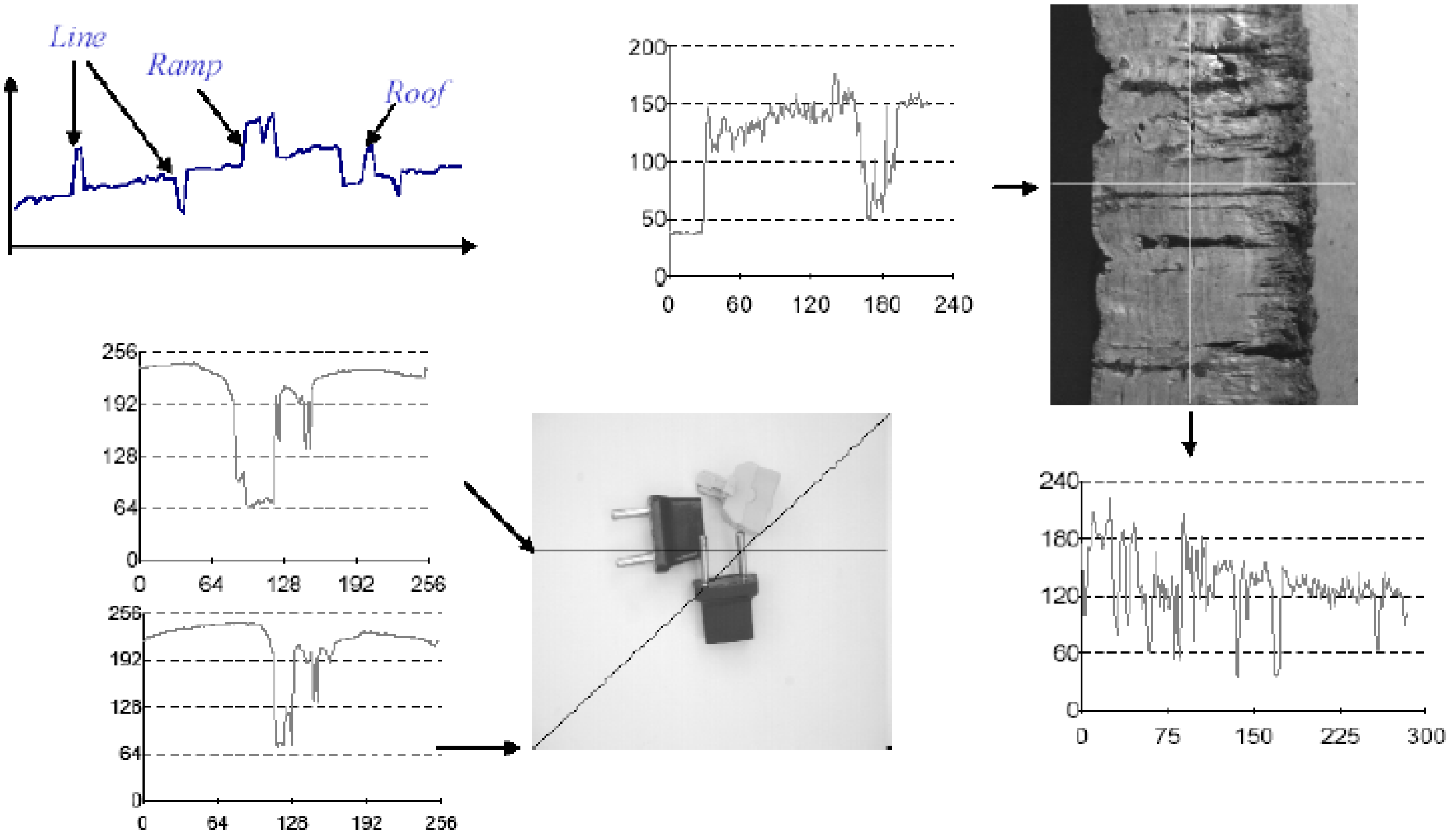

(b) A “ramp” edge


(c) A line


(d) A “roof”

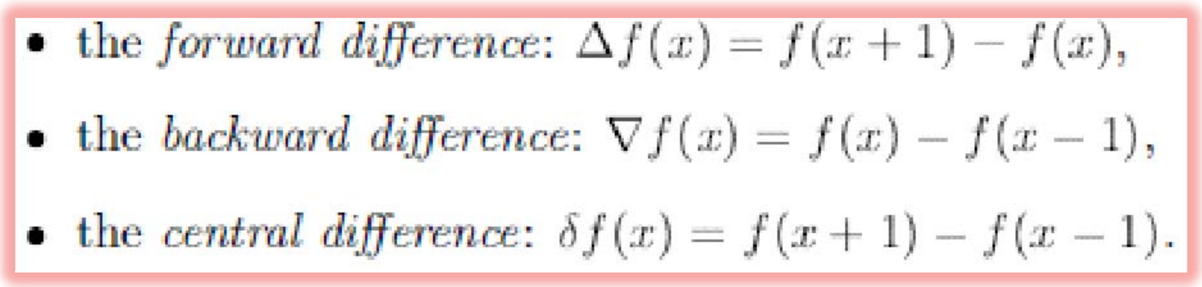
Grey values across edges

Types of Edges



Difference Measure

- The difference tends to enhance edges, and reduce other components.
- Differences can be defined as follows:



- the *forward difference*: $\Delta f(x) = f(x+1) - f(x)$,
- the *backward difference*: $\nabla f(x) = f(x) - f(x-1)$,
- the *central difference*: $\delta f(x) = f(x+1) - f(x-1)$.

2 – Dim Case: Image

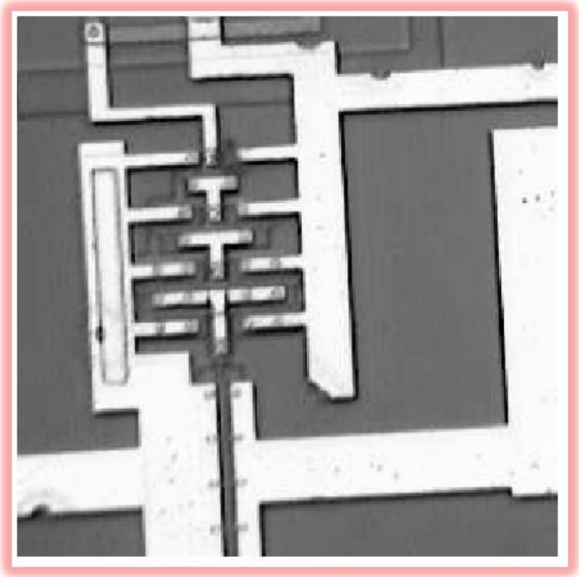
$$
\begin{aligned}
\Delta_x f(x,y) &= f(x+1,y) - f(x,y) & \Delta_y f(x,y) &= f(x,y+1) - f(x,y) \\
\nabla_x f(x,y) &= f(x,y) - f(x-1,y) & \nabla_y f(x,y) &= f(x,y) - f(x,y-1) \\
\delta_x f(x,y) &= f(x+1,y) - f(x-1,y) & \delta_y f(x,y) &= f(x,y+1) - f(x,y-1)
\end{aligned}
$$

Prewitt Filter

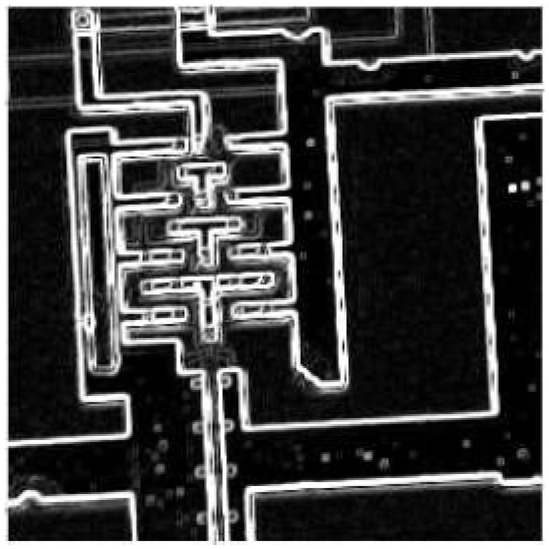
- For finding the vertical Edges: $P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$
- For Horizontal Edges $P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
- For Reconstruction choose any one:

1. $v = \max\{|p_x|, |p_y|\}$,
2. $v = |p_x| + |p_y|$,
3. $v = \sqrt{p_x^2 + p_y^2}$.

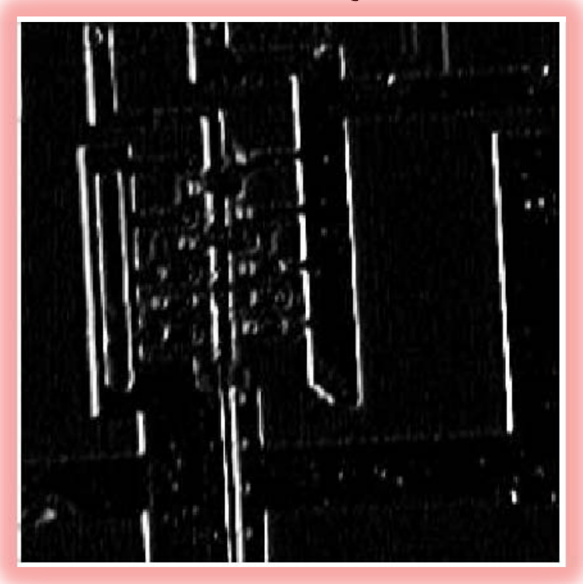
Original Image



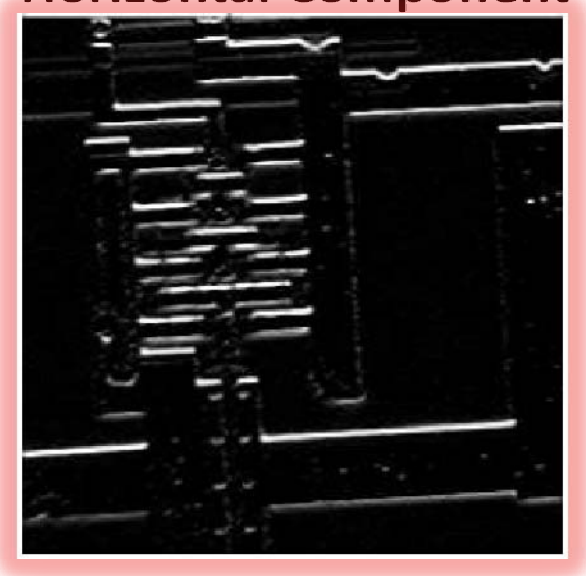
Prewitt Filter Image



Vertical Component



Horizontal Component



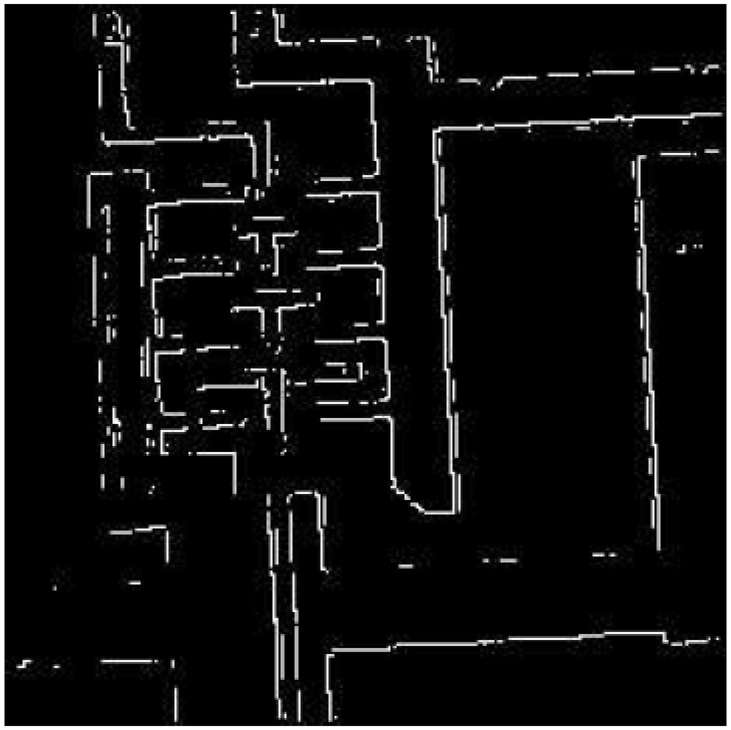
Roberts Cross Gradient Filter

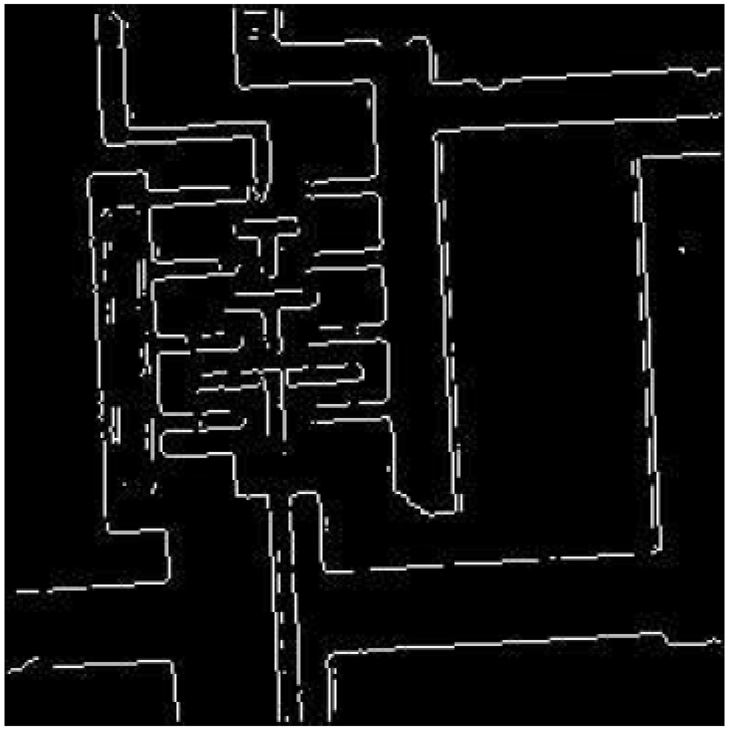
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Sobel Filter

- Sobel filters are similar to the prewitt filter
- Some pixels have been weighted more heavily than others

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & -2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$


(a) Roberts edge detection


(b) Sobel edge detection

Second Differences

- Another class of edge-detection method is obtained by considering the difference of differences; these are called second differences.
- To calculate a (central) second difference, take the backward difference of a forward difference:

$$\begin{aligned}
\delta_x^2 &= \nabla\Delta f(x,y) \\
&= \nabla(f(x+1,y) - f(x,y)) \\
&= (f(x+1,y) - f(x,y)) - (f(x,y) - f(x-1,y)) \\
&= f(x+1,y) - 2f(x,y) + f(x-1,y)
\end{aligned}$$

- This can be implemented by the following filter: $$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

- Corresponding filter for second difference δ_y^2 in y direction,

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- The sum of both the direction

$$\nabla^2 = \delta_x^2 + \delta_y^2$$

- Is implemented by the following (discrete Laplacian) filter

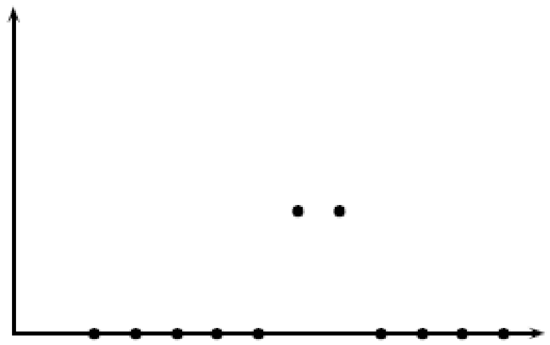
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Differences Measure for Edges

- The differences between each grey value and its predecessor from the ramp edge
- Suppose if the pixel values are as follows:

  [20 20 20 20 20 20 100 180 180 180 180 180]
- The difference Metrics can be constructed as follows:

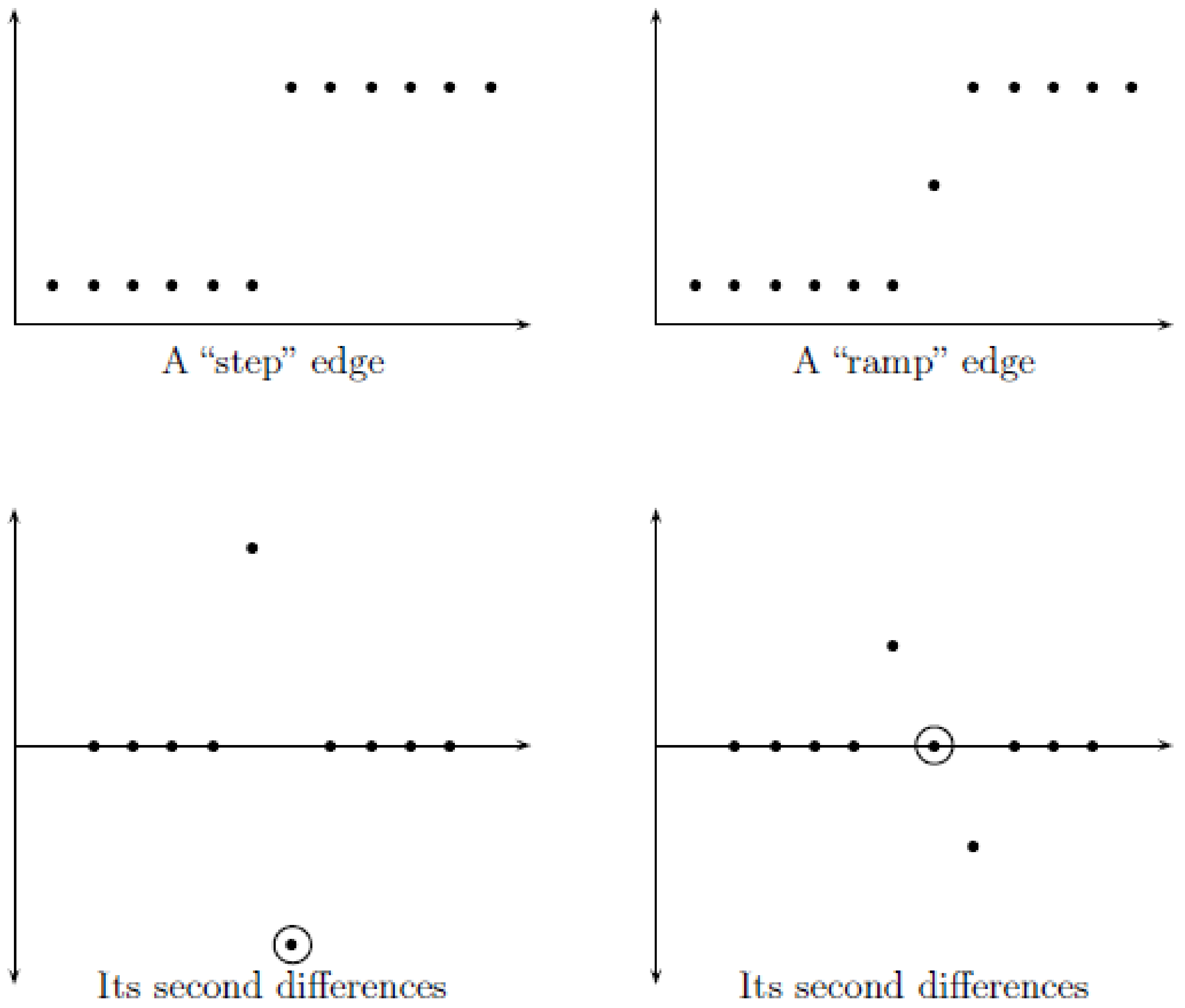
  [0 0 0 0 0 80 80 0 0 0 0]



Differences of the edge function

Second Difference

- [20 20 20 20 20 20 100 180 180 180 180 180]
- [0 0 0 0 0 80 80 0 0 0 0]
- [0 0 0 0 80 0 -80 0 0 0]



Edges and second differences

Other Laplacian Mask

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

$$\frac{1}{\alpha+1}\begin{bmatrix} \alpha & 1-\alpha & \alpha \\ 1-\alpha & -4 & 1-\alpha \\ \alpha & 1-\alpha & \alpha \end{bmatrix}$$

Zero Crossing in Image using Laplacian

- More appropriate use of Laplacian is to find the position of edges by locating zero crossings
- Zero crossing are the place where filter changes its sign
- The important point is to note that across any edge there can be only one zero-crossing.
- We define the zero crossings in such a filtered image to be pixels which satisfy either of the following:
  - they have a negative grey value and are next to (by four-adjacency) a pixel whose grey value is positive,
  - they have a value of zero, and are in between negative and positive valued pixels.

50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	200	200	200	200	200	200	50	50
50	50	50	50	200	200	200	200	50	50
50	50	50	50	200	200	200	200	50	50
50	50	50	50	50	50	50	50	50	50
50	50	50	50	50	50	50	50	50	50

(a) A simple image

−100	−50	−50	−50	−50	−50	−50	−50	−50	−100
−50	**0**	150	150	150	150	150	150	**0**	−50
−50	150	**−300**	**−150**	**−150**	**−150**	**−150**	**−300**	150	−50
−50	150	**−150**	0	0	0	0	**−150**	150	−50
−50	150	**−150**	0	0	0	0	**−150**	150	−50
−50	150	**−300**	**−150**	0	0	0	**−150**	150	−50
−50	**0**	150	300	**−150**	0	0	**−150**	150	−50
−50	0	0	150	**−300**	**−150**	**−150**	**−300**	150	−50
−50	0	0	**0**	150	150	150	150	**0**	−50
−100	−50	−50	−50	−50	−50	−50	−50	−50	−100

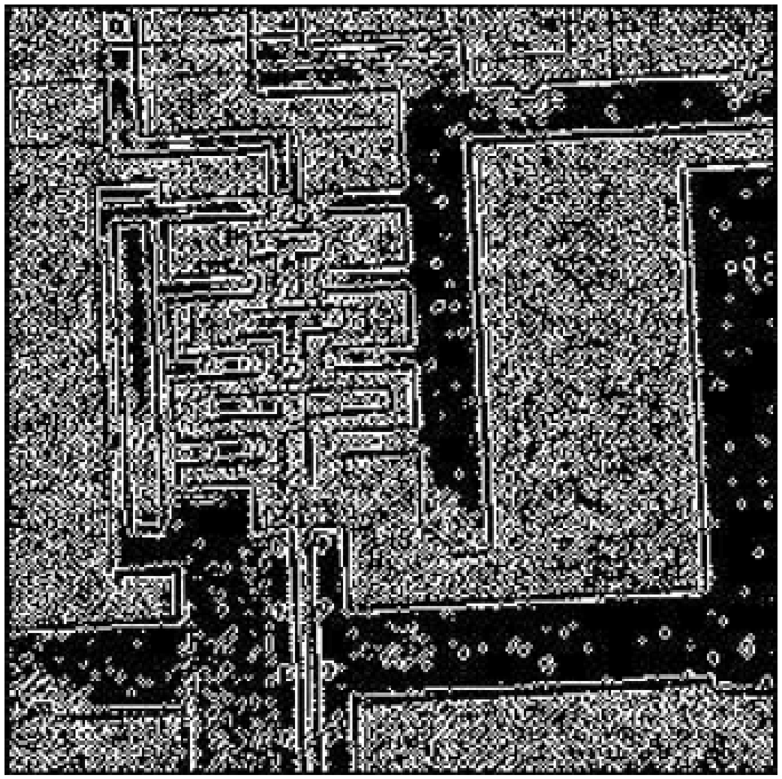
(b) After laplace filtering

Locating zero crossings in an image

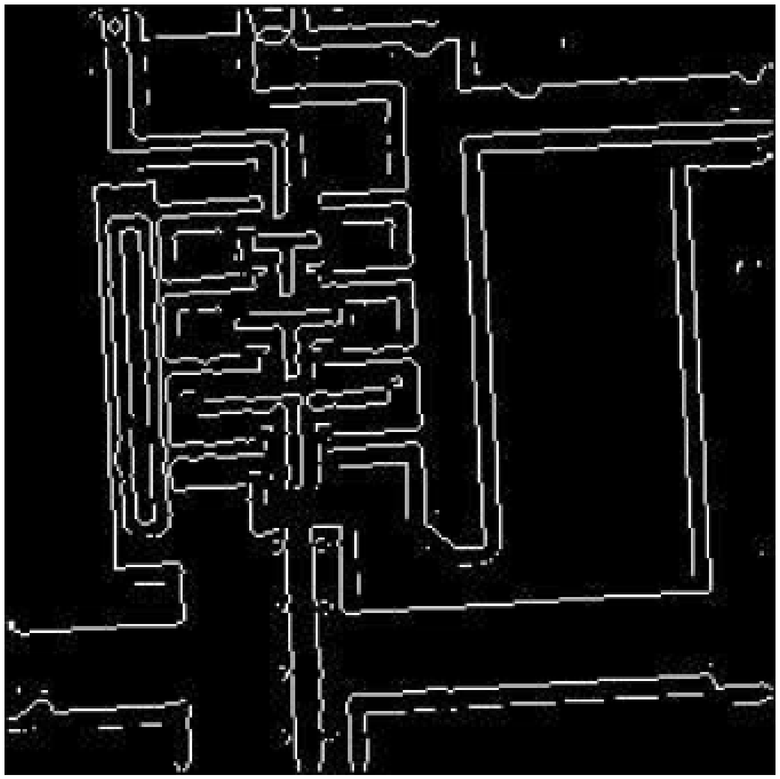
Marr- Hildreth Method

- Smooth the image with a Gaussian filter,
- Convolve the result with a laplacian,
- Find the zero crossings.

The first two steps can be combined into one, to produce a Laplacian of Gaussian or LoG filter



(a) Zeros crossings



(b) Using an LoG filter first

Edge detection using zero crossings