Digital filters

Francesco Isgrò

October 3, 2023

Image noise

Noise is generally grouped in two classes

- independent noise
- data dependent noise (due to wave interference)

Image noise

Noise is generally grouped in two classes

- independent noise
  - additive noise model
    $$I_{ij} = \hat{I}_{ij} + n_{ij}$$
  - it is often zero-mean, i.e. $\mu_n = 0$
  - described by its variance σ_n^2
- data dependent noise (due to wave interference)

Image noise

Noise is generally grouped in two classes

- independent noise
  - additive noise model
    $$I_{ij} = \hat{I}_{ij} + n_{ij}$$
  - it is often zero-mean, i.e. $\mu_n = 0$
  - described by its variance σ_n^2
- data dependent noise (due to wave interference)
  - multiplicative or non-linear model
    $$I_{ij} = \hat{I}_{ij} n_{ij}$$
  - more complicated

Image noise

Noise is generally grouped in two classes

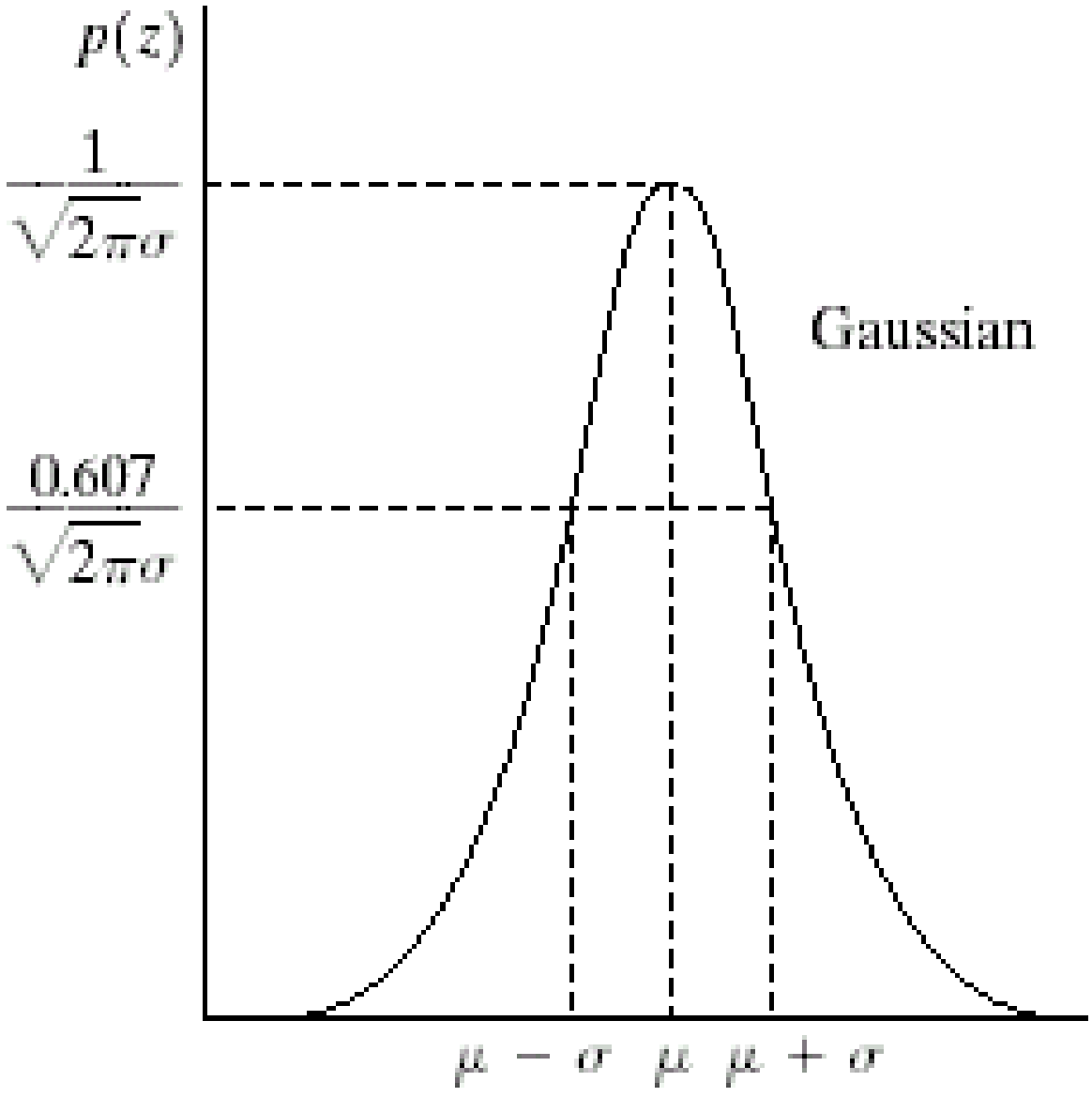
- independent noise
  - additive noise model
    $$I_{ij} = \hat{I}_{ij} + n_{ij}$$
  - it is often zero-mean, i.e. $\mu_n = 0$
  - described by its variance σ_n^2
- data dependent noise (due to wave interference)
  - multiplicative or non-linear model
    $$I_{ij} = \hat{I}_{ij} n_{ij}$$
  - more complicated
- unless necessary noise is assumed data independent

Main noise models

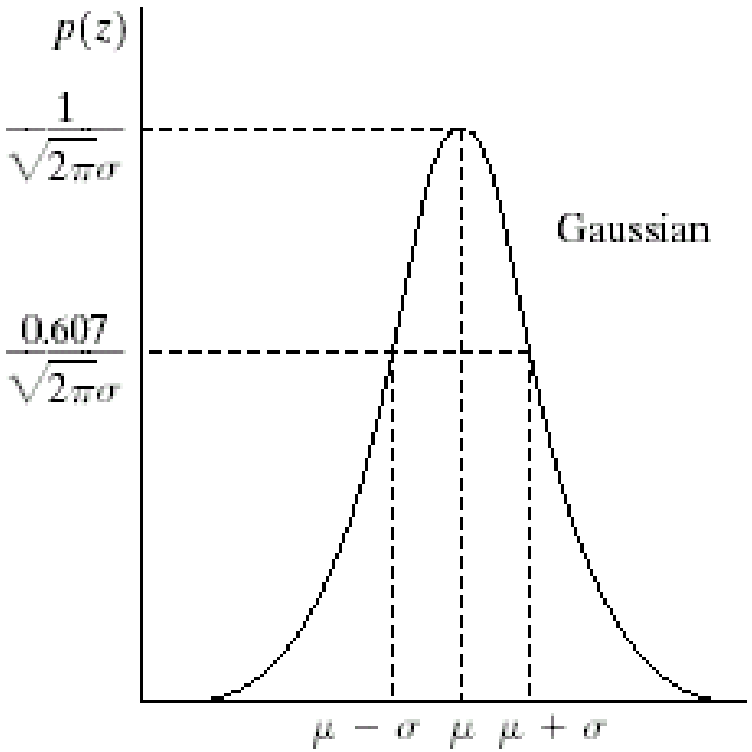
Noise is modelled as a random variable

- Gaussian noise
- Rayleigh noise
- Gamma noise
- Exponential noise
- Uniform noise
- Impulse noise

Gaussian noise

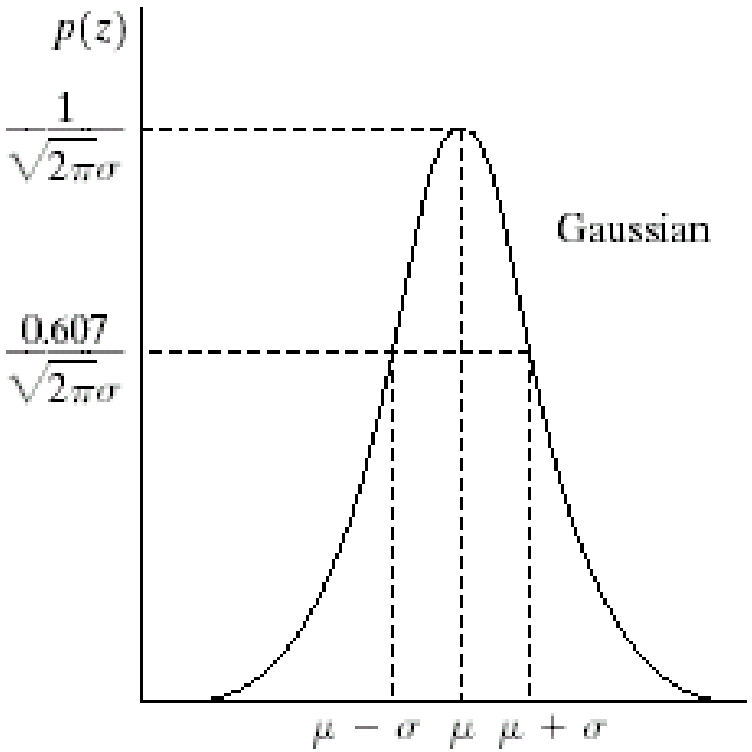


Gaussian noise



$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2}$$

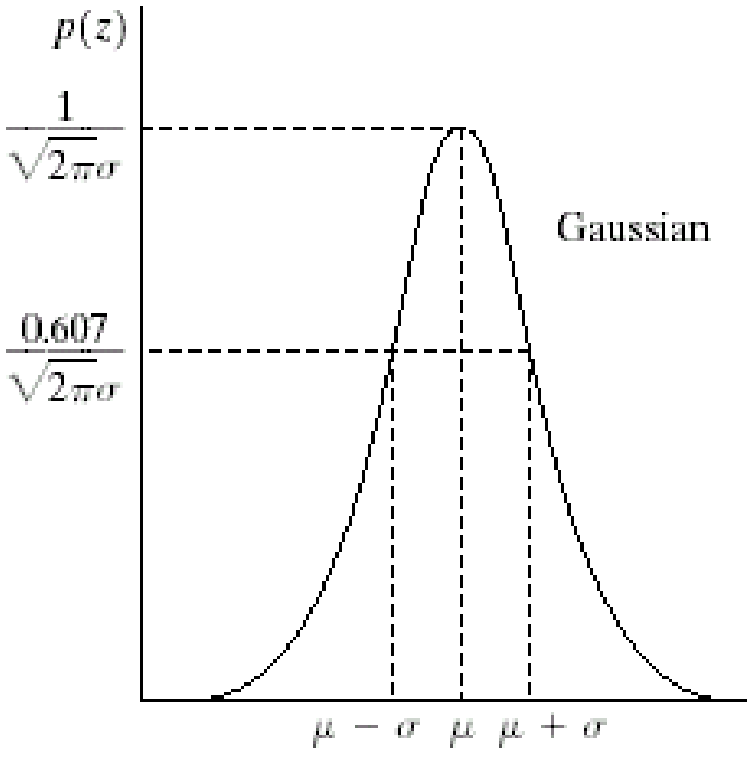
Gaussian noise

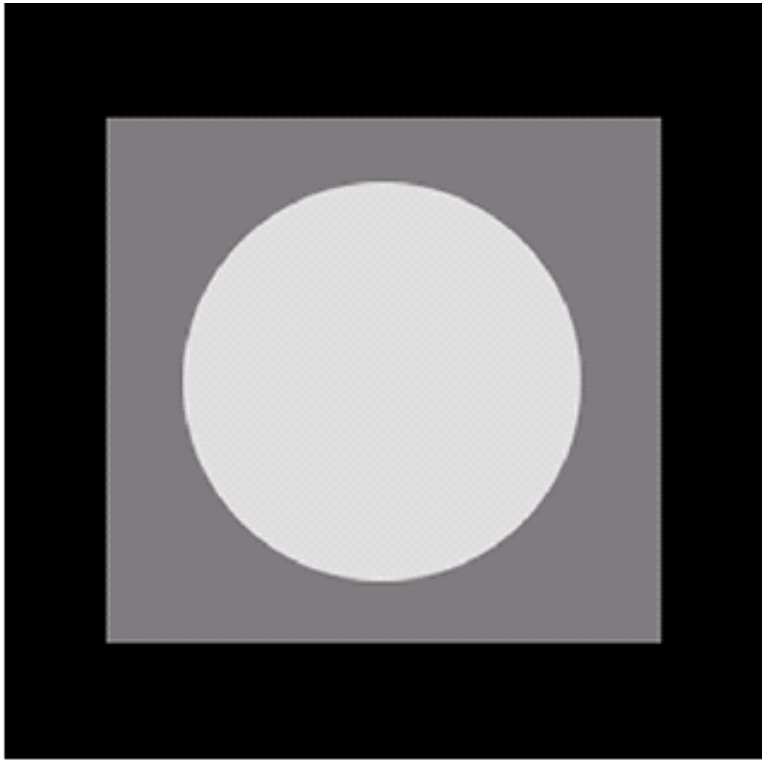


$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2}$$

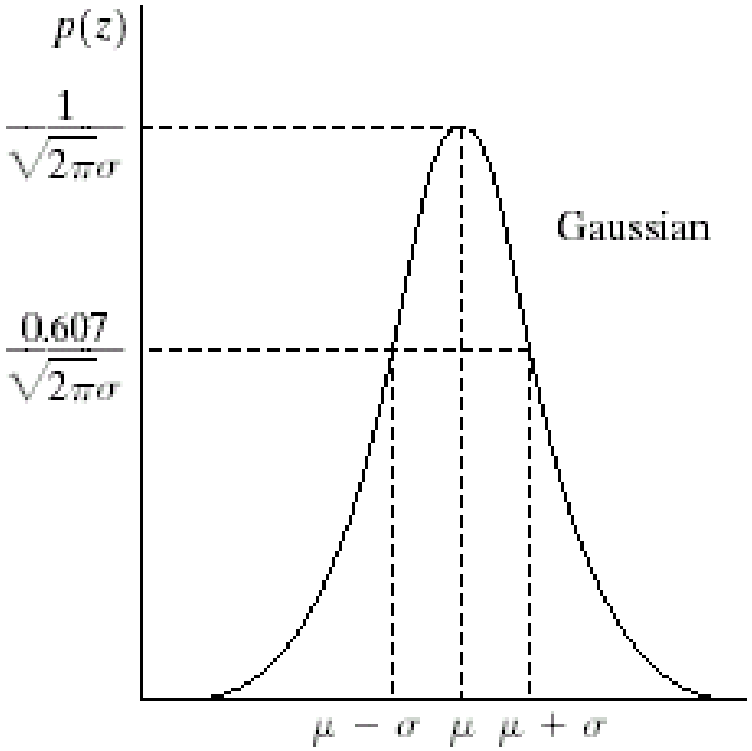
- z is the gray level
- μ is the mean
- σ is the standard deviation
- it is due to sensor noise, bad illumination, high temperature

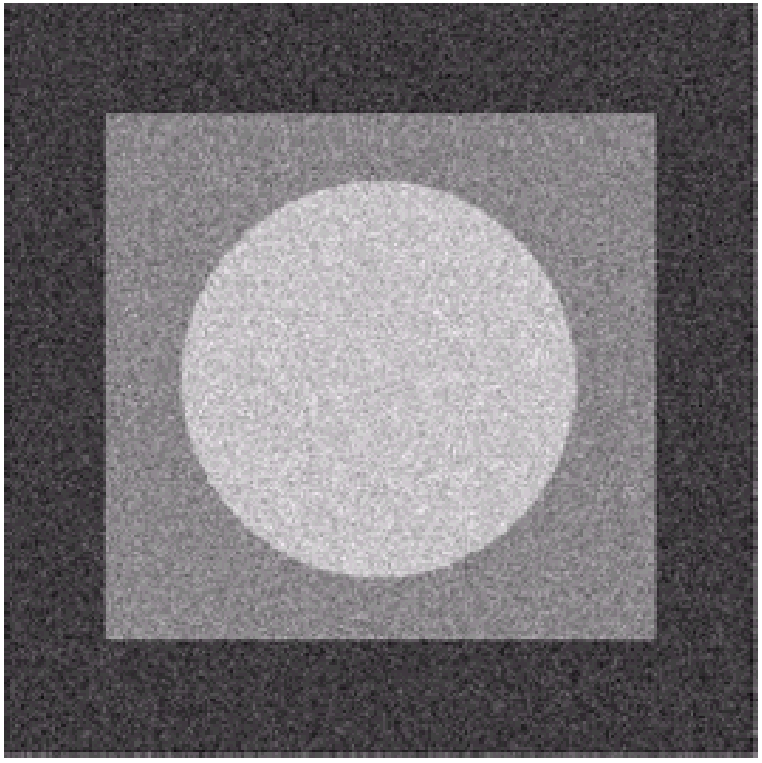
Gaussian noise



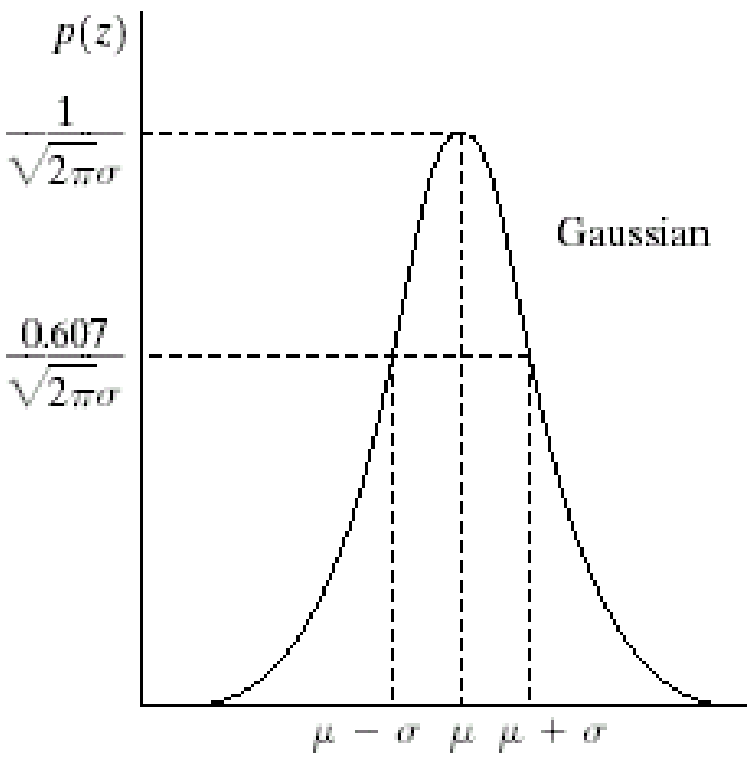


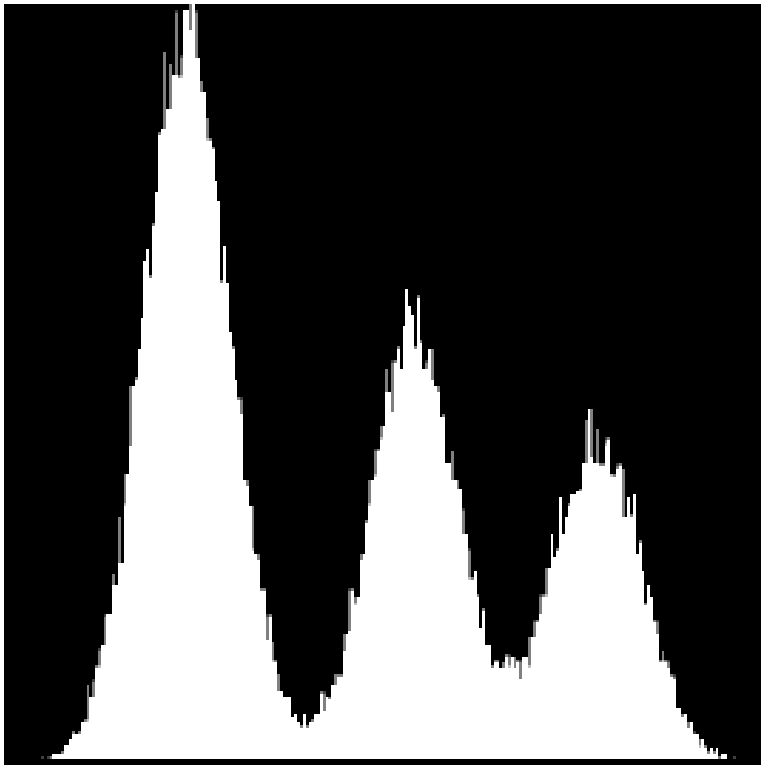
Gaussian noise



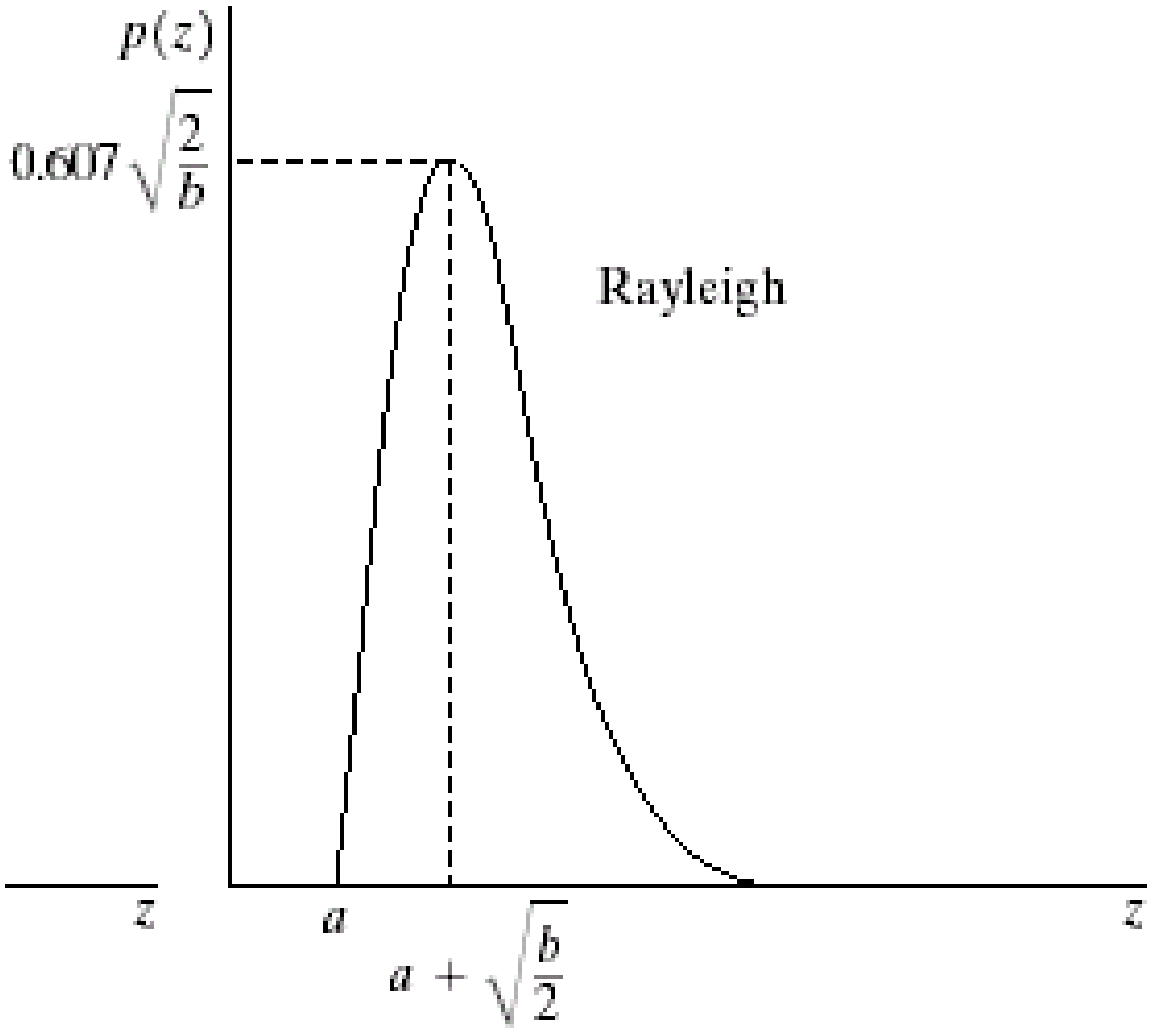


Gaussian noise

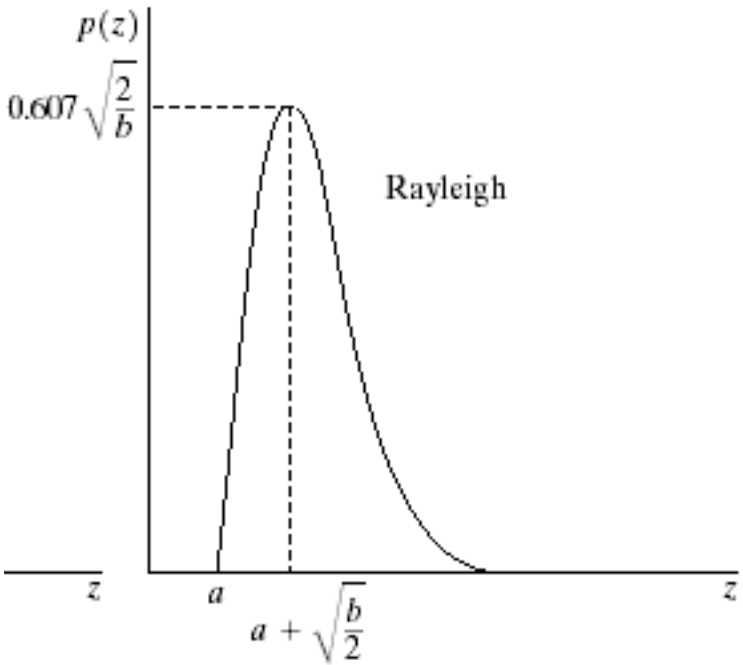




Rayleigh noise

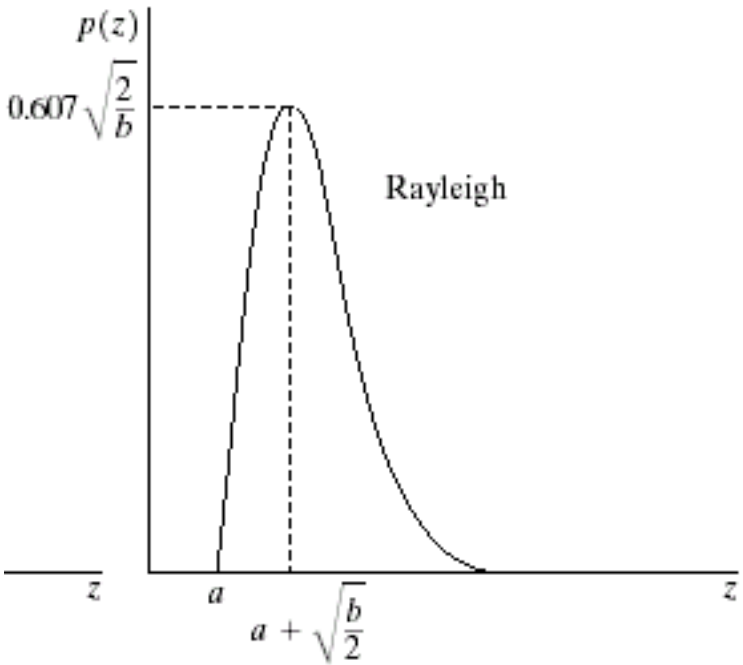


Rayleigh noise



$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{if } z \geq a \\ 0 & \text{if } z < a \end{cases}$$

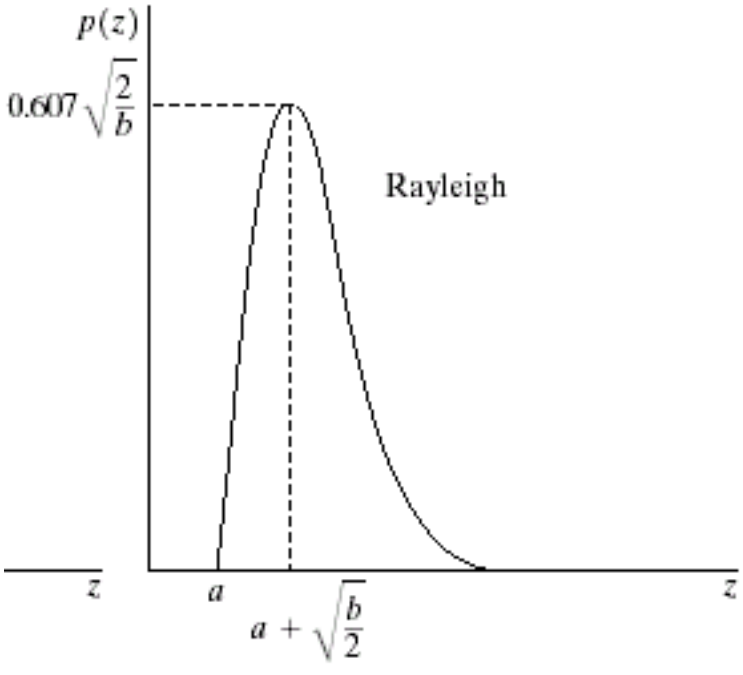
Rayleigh noise

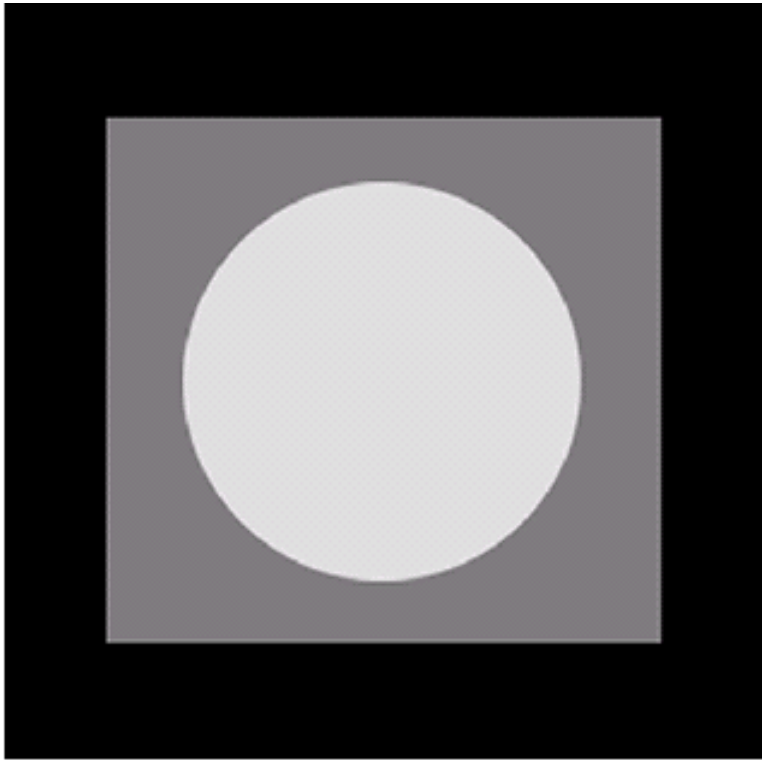


$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{if } z \geq a \\ 0 & \text{if } z < a \end{cases}$$

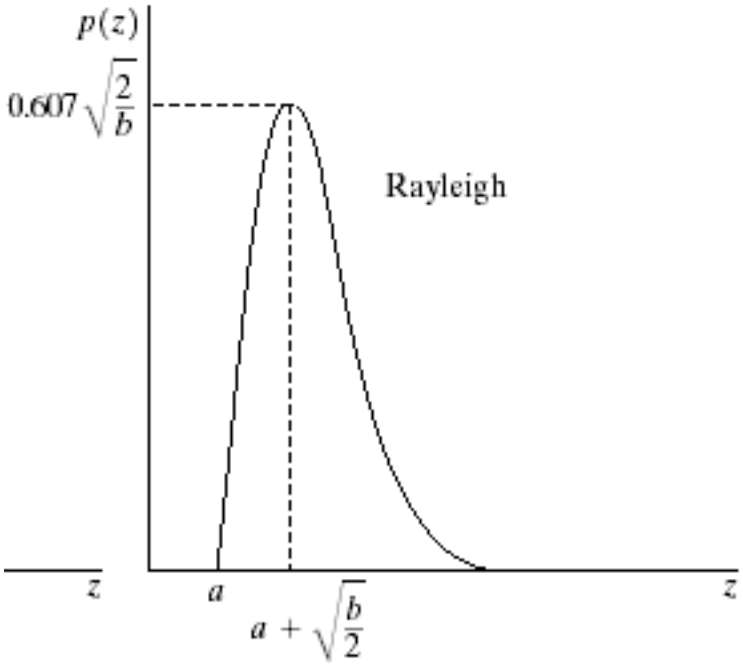
- $\mu = a + \sqrt{\pi b/4}$
- $\sigma^2 = \frac{b(4-\pi)}{4}$
- used to characterise noise in range images

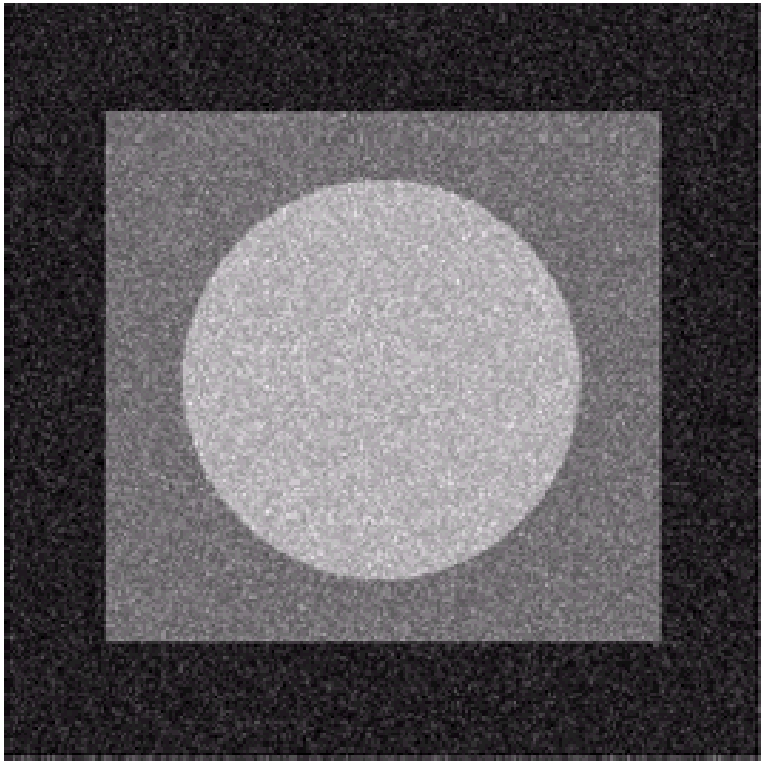
Rayleigh noise



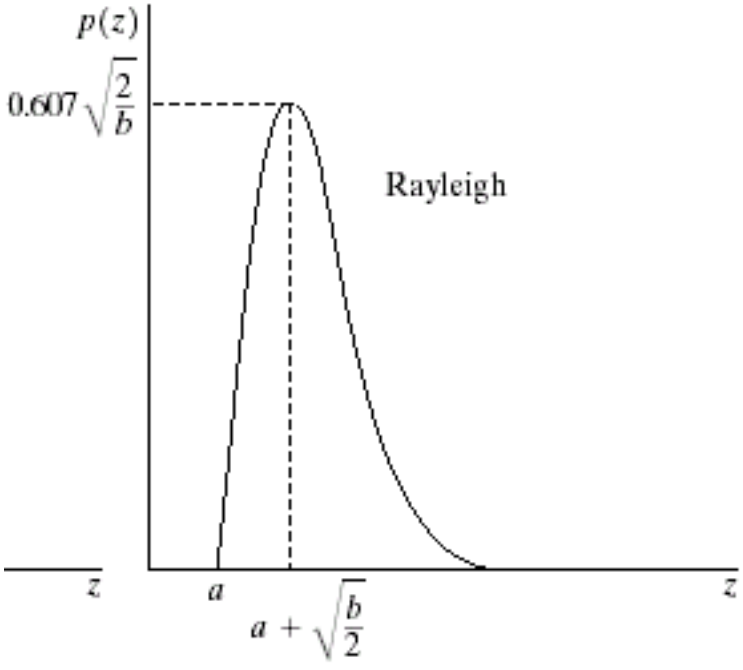


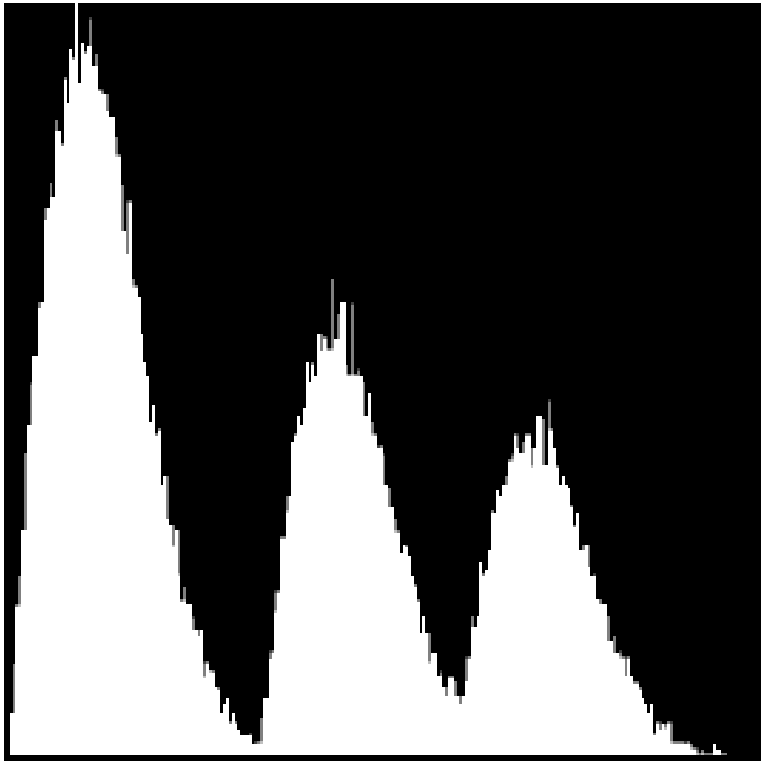
Rayleigh noise



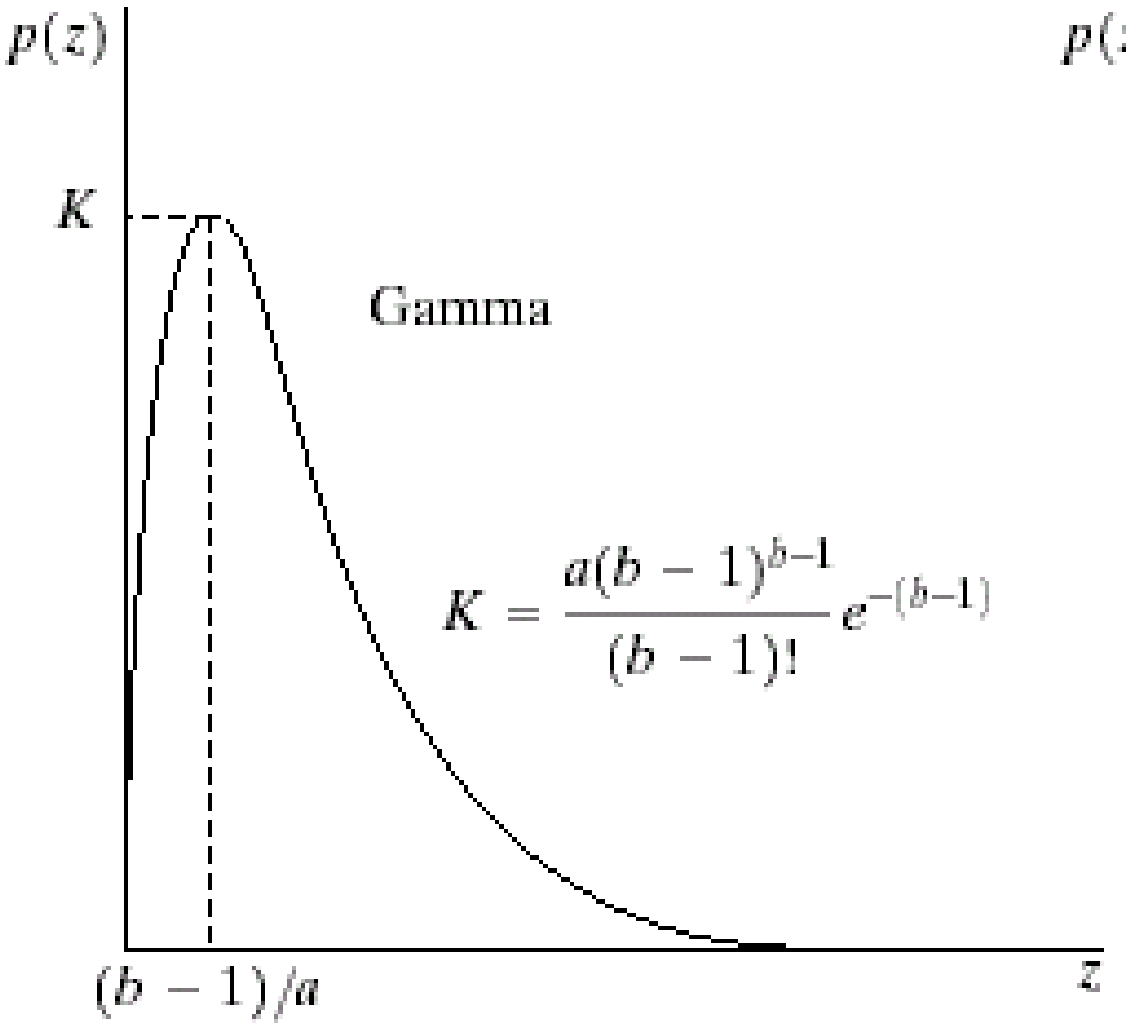


Rayleigh noise

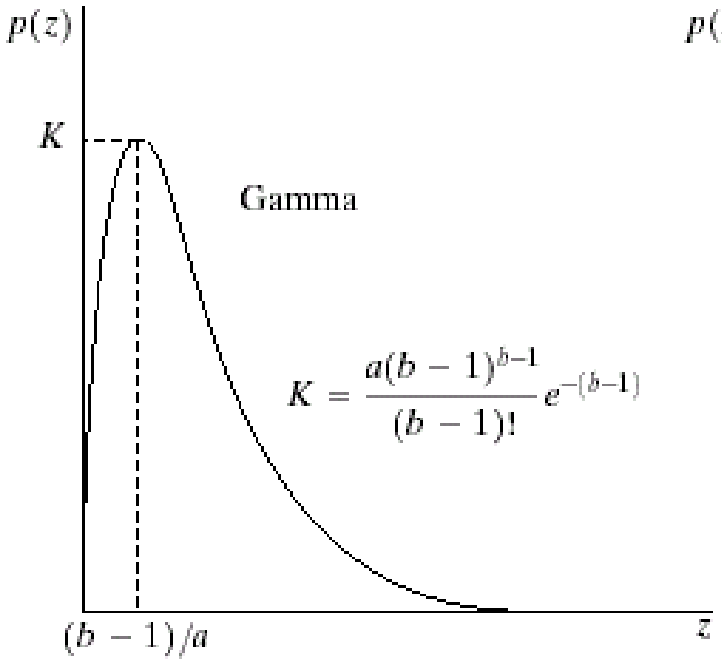




Gamma noise

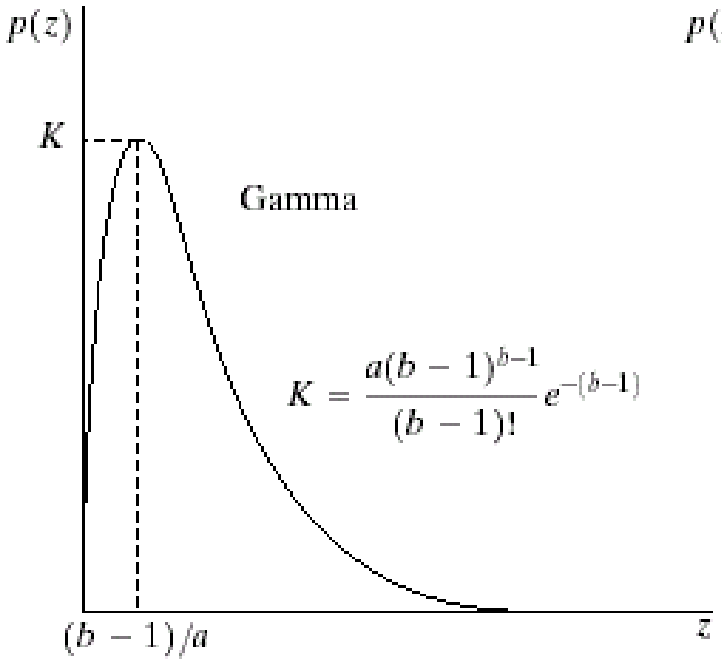


Gamma noise



$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

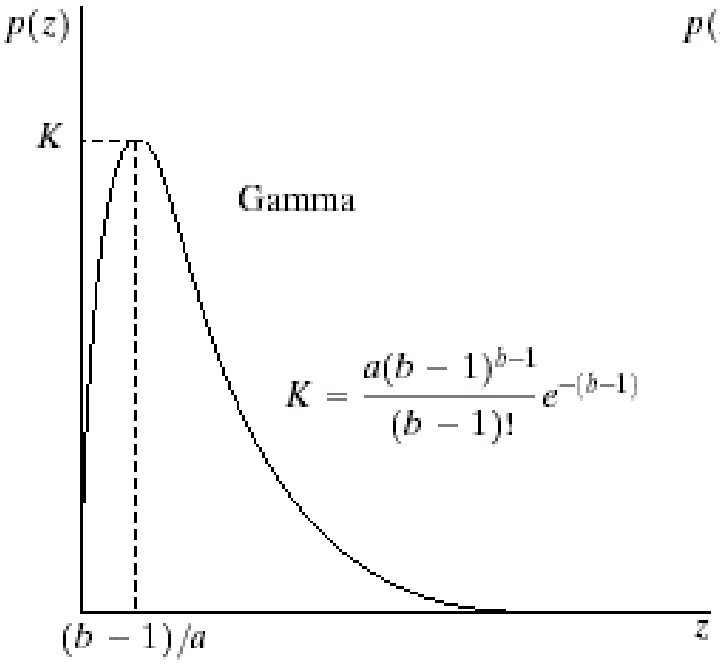
Gamma noise

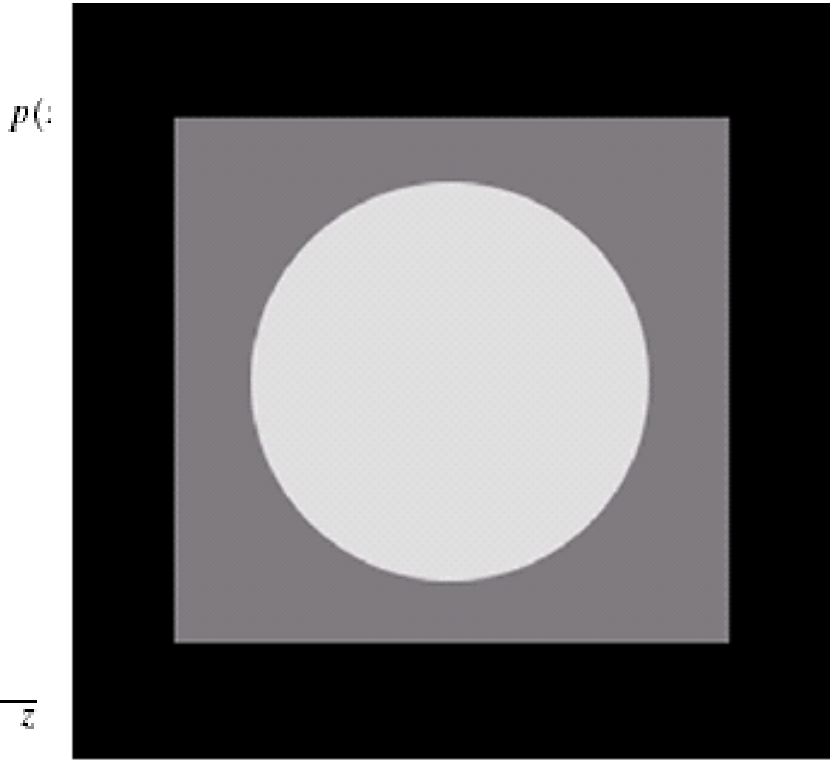


$$p(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

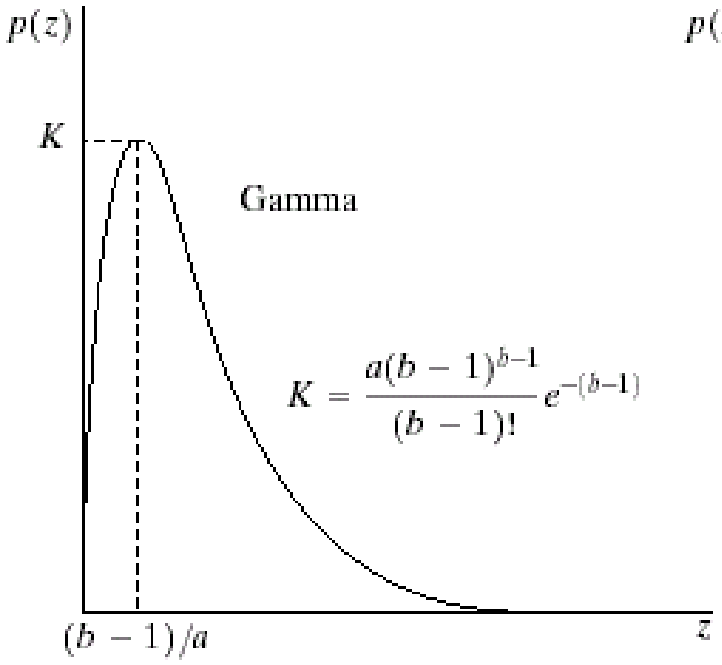
- $a > 0$
- $b \epsilon \mathbb{Z}$
- $\mu = b/a$
- $\sigma^2 = b/a^2$
- used in laser imaging

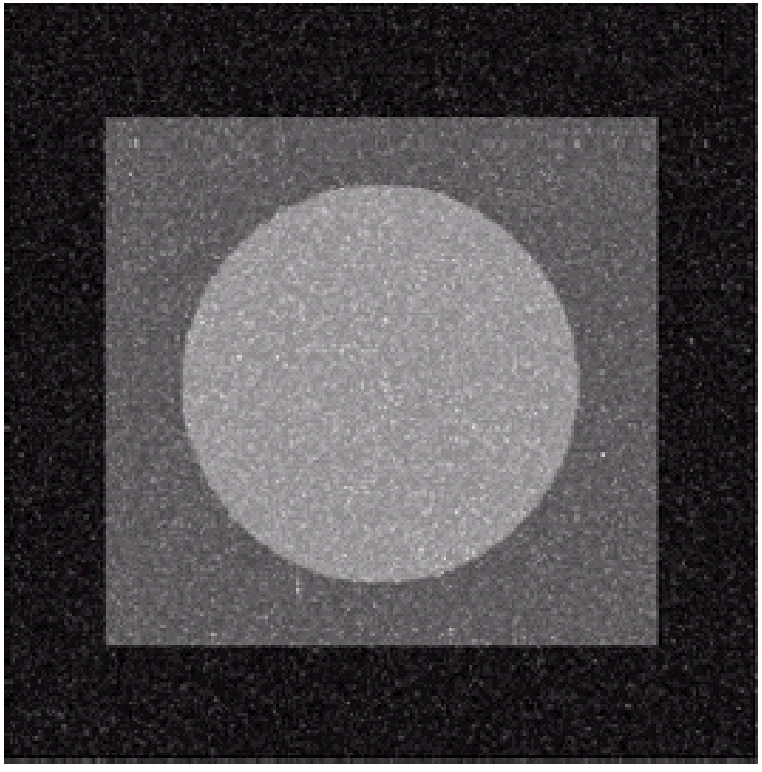
Gamma noise



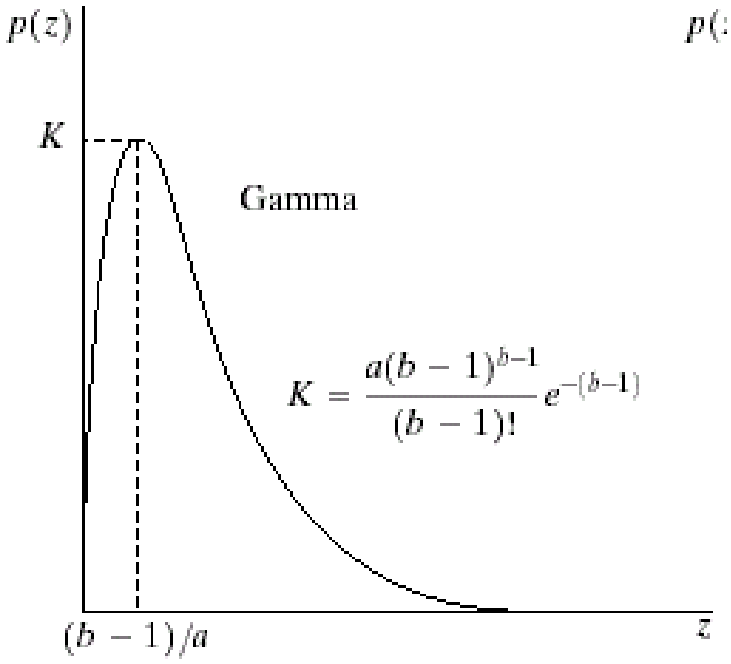


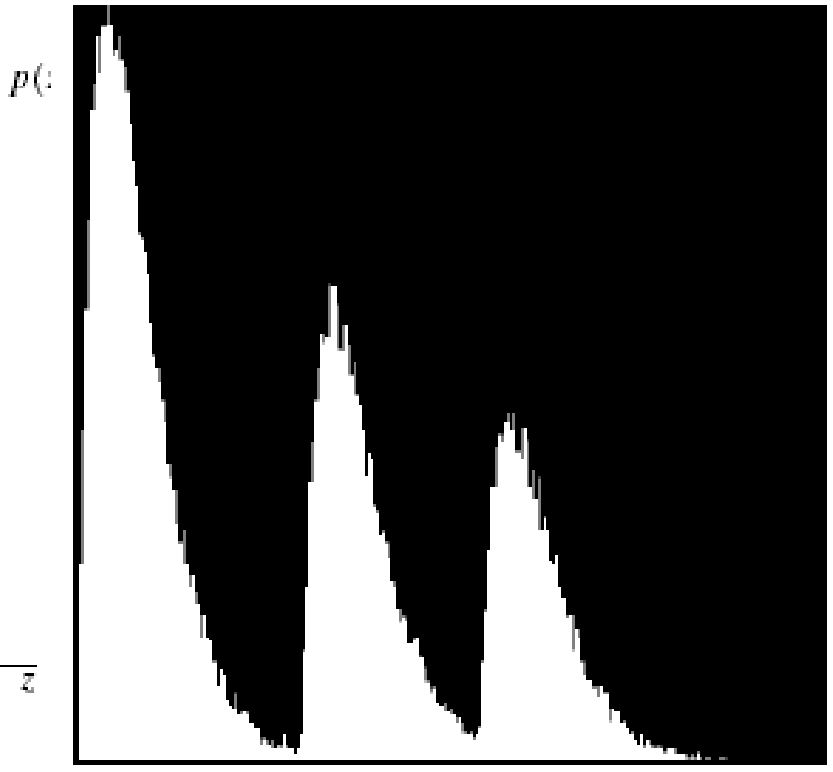
Gamma noise



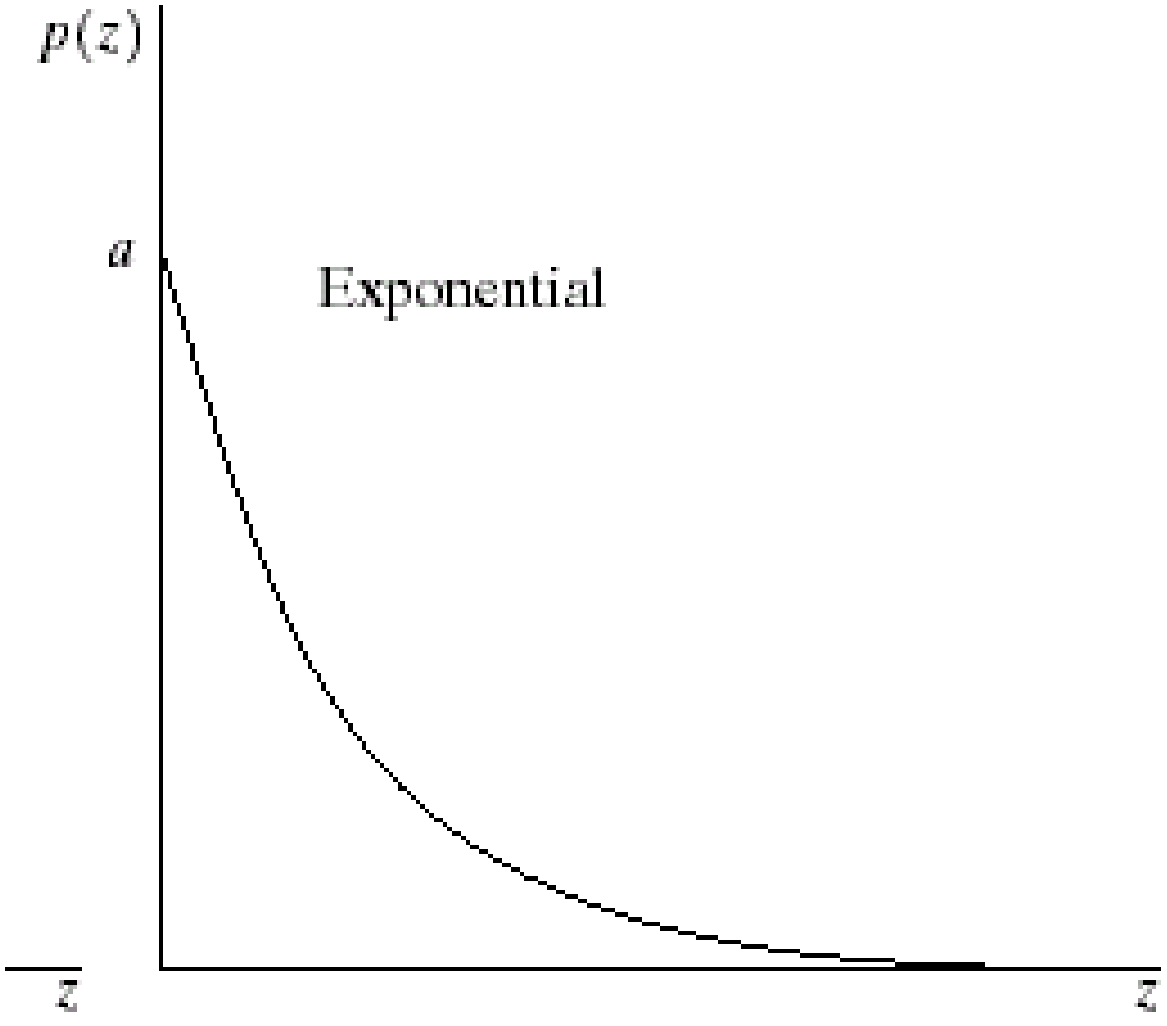


Gamma noise

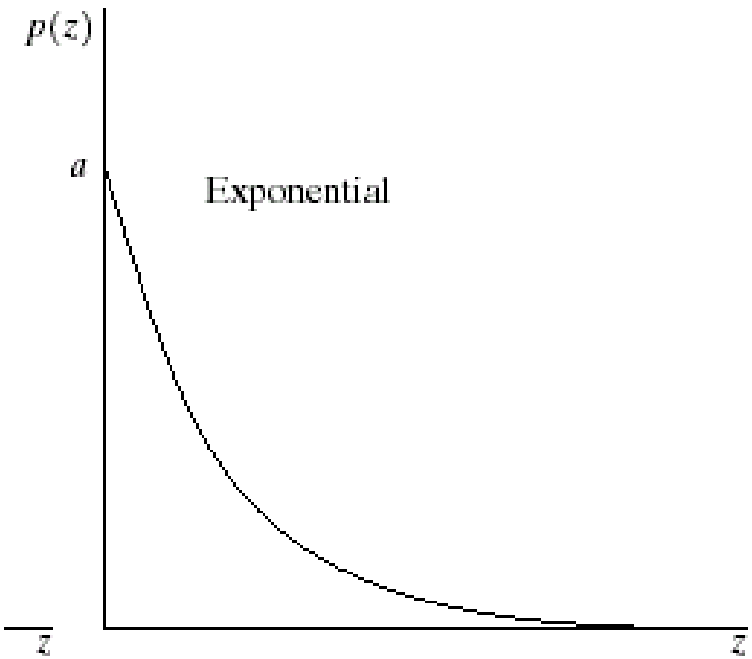




Exponential noise

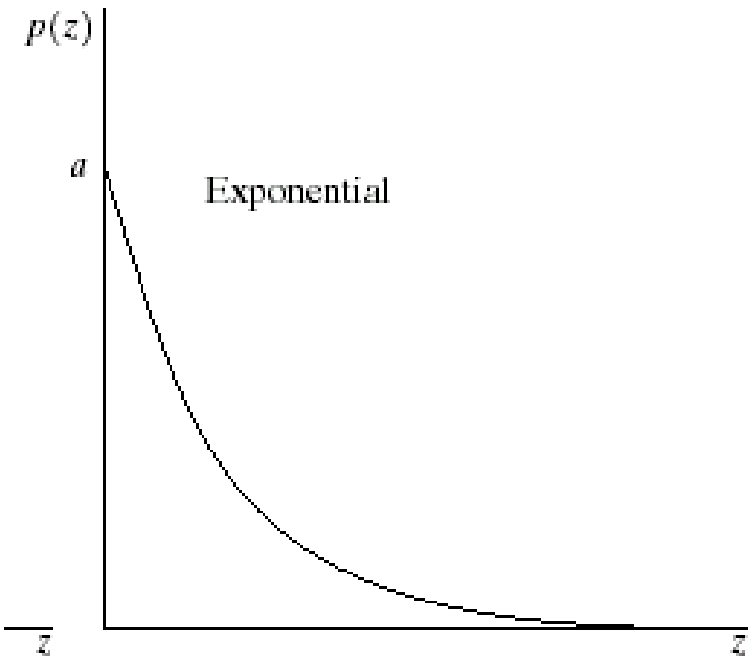


Exponential noise



$$p(z) = \begin{cases} ae^{-az} & if\ z \geq 0 \\ 0 & if\ z < 0 \end{cases}$$

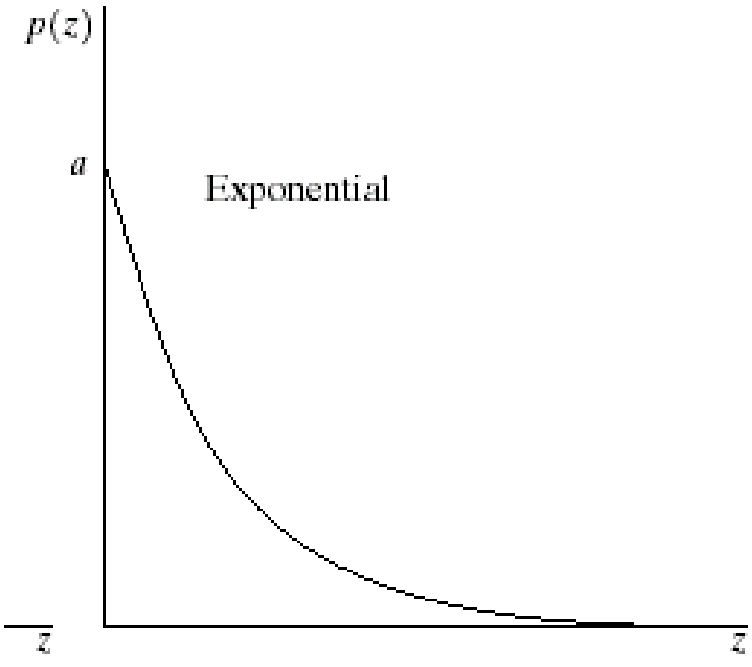
Exponential noise

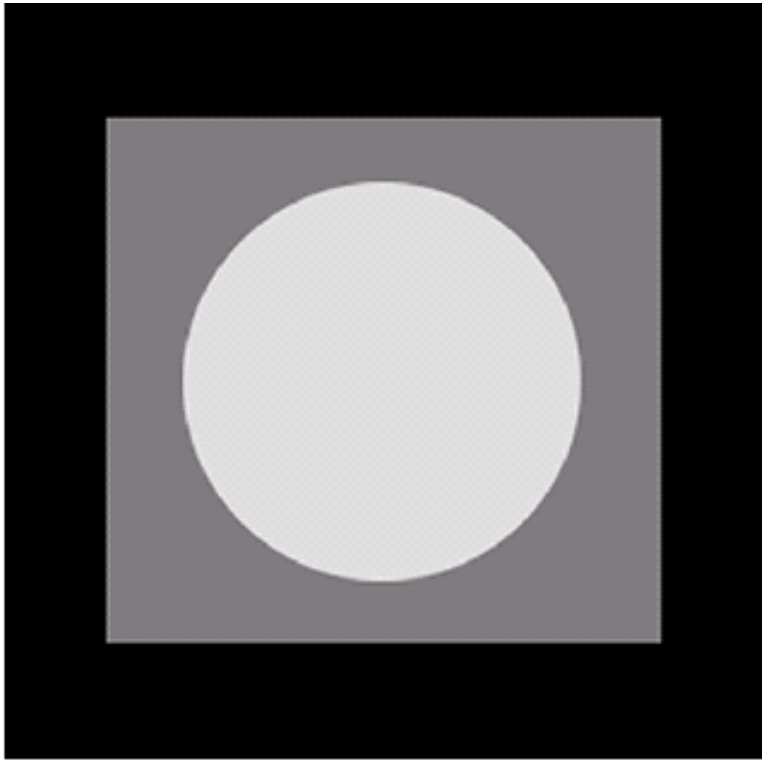


$$p(z) = \begin{cases} ae^{-az} & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

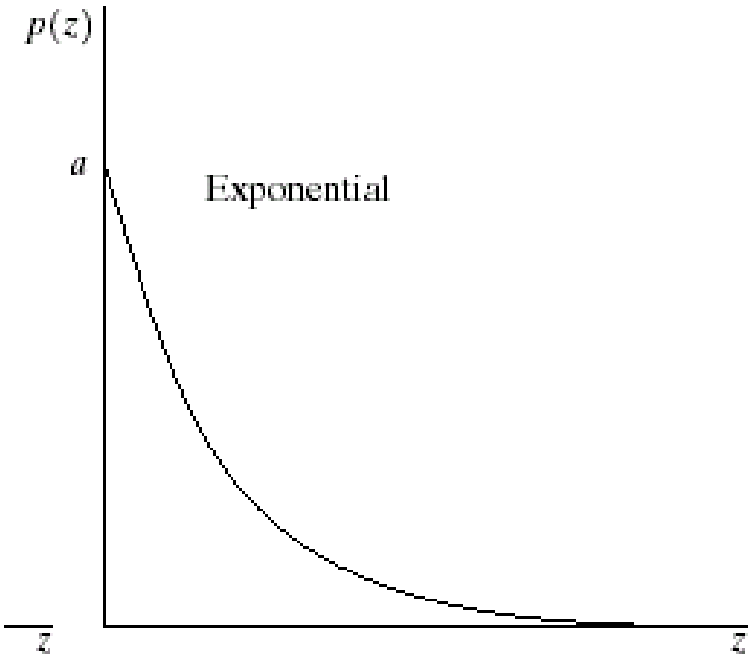
- $a > 0$
- $\mu = 1/a$
- $\sigma^2 = 1/a^2$
- it is a special case of the Gamma pdf, when $b = 1$
- used in laser imaging

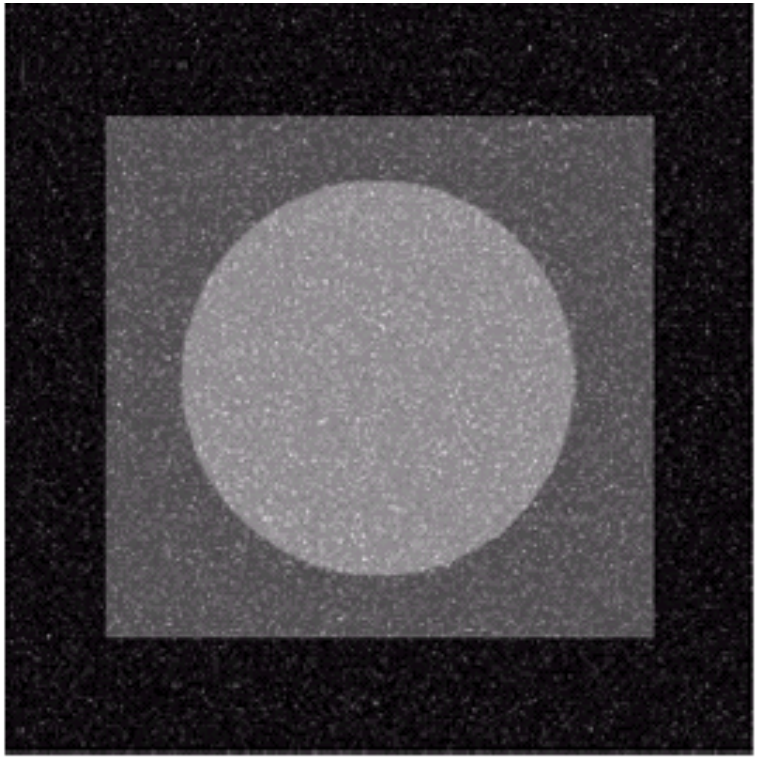
Exponential noise



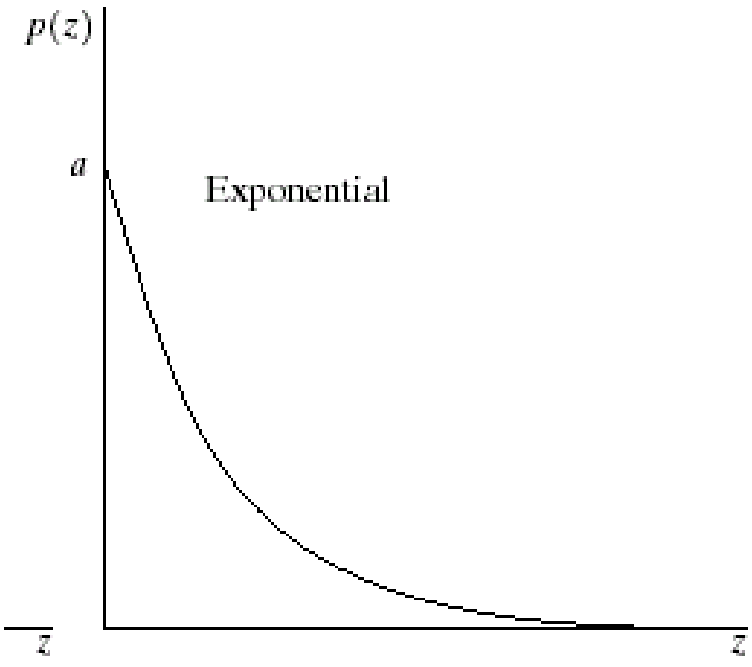


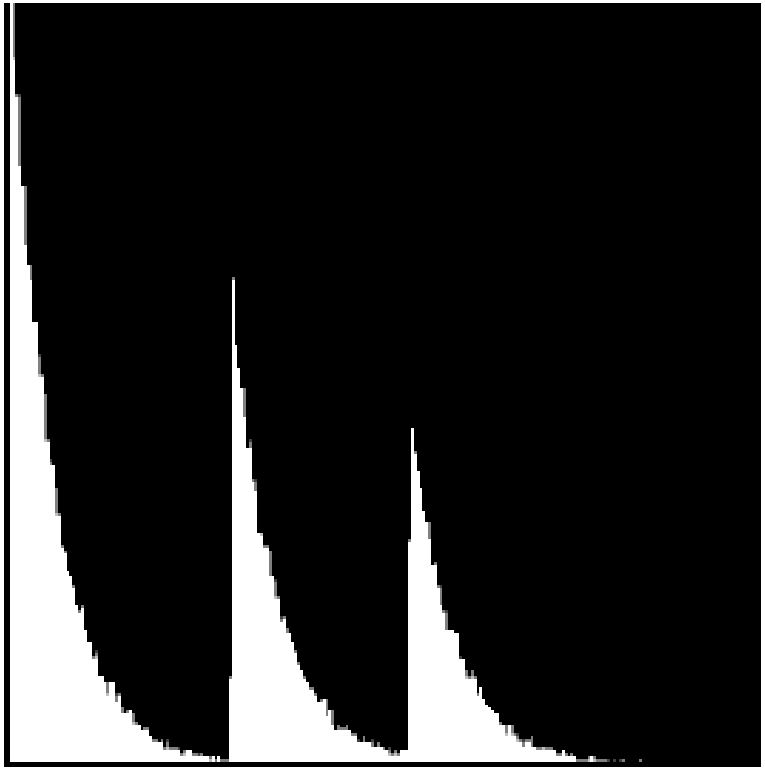
Exponential noise



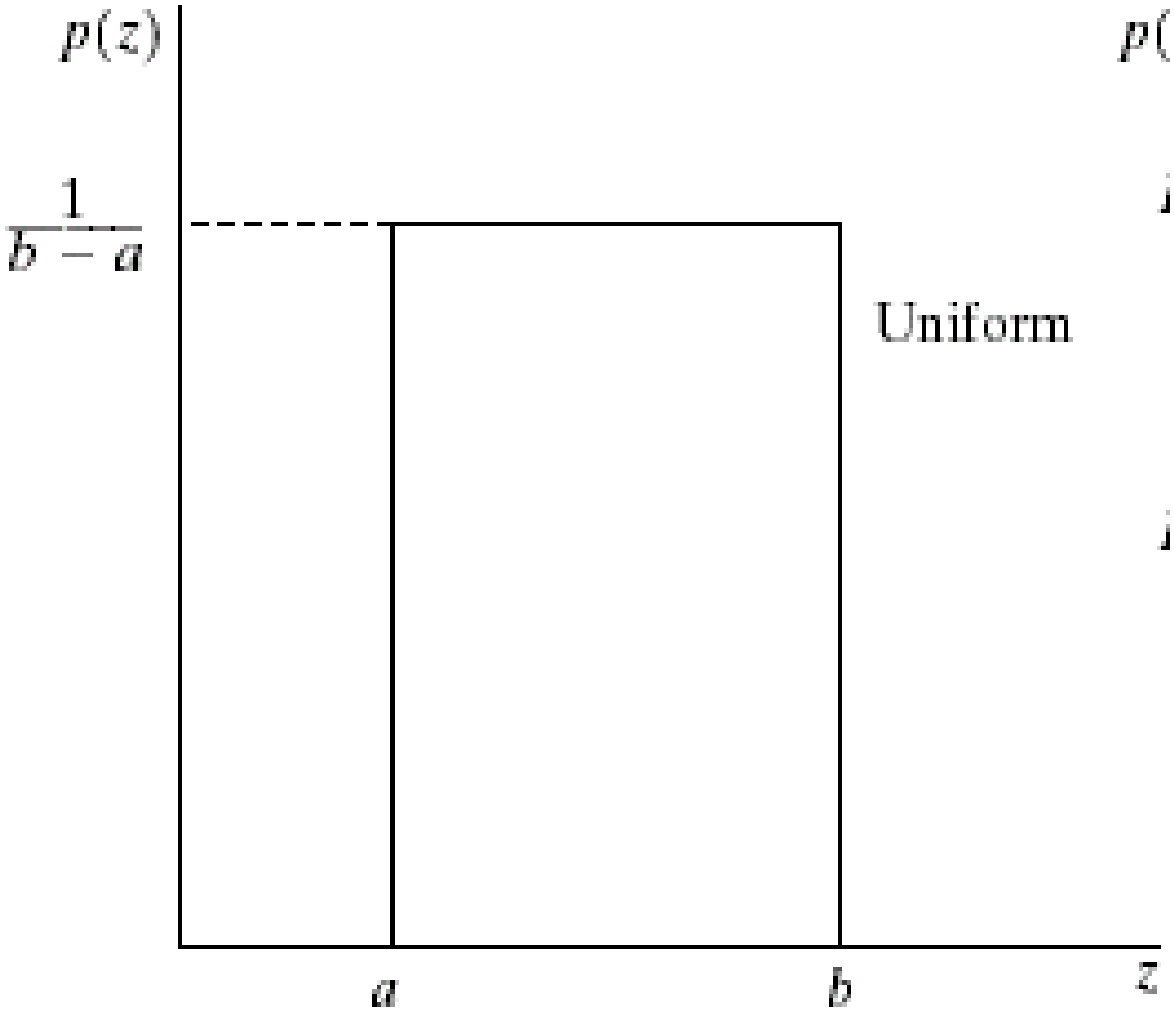


Exponential noise

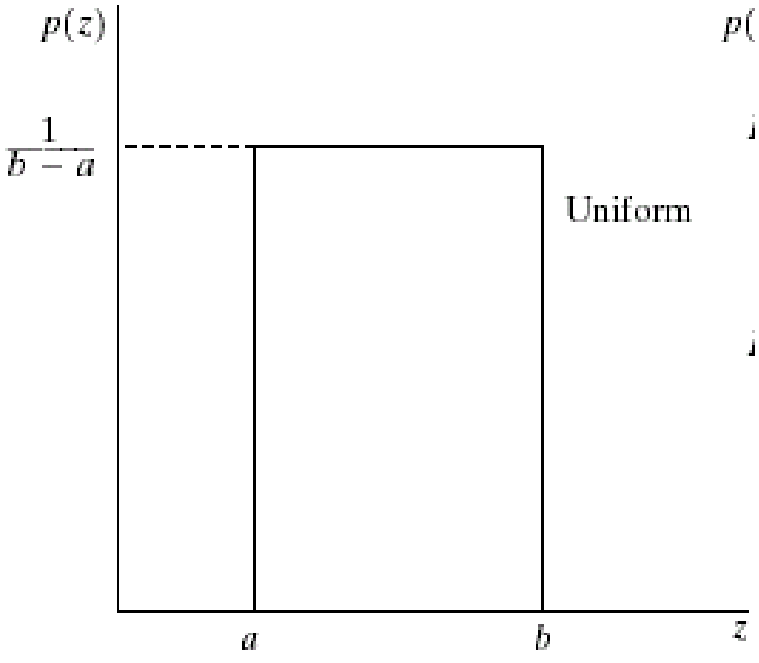




Uniform noise

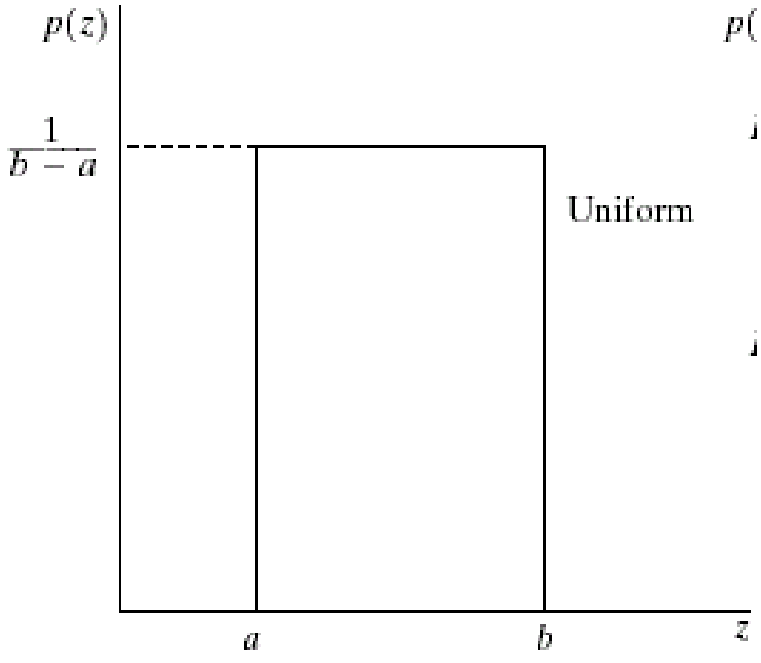


Uniform noise



$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

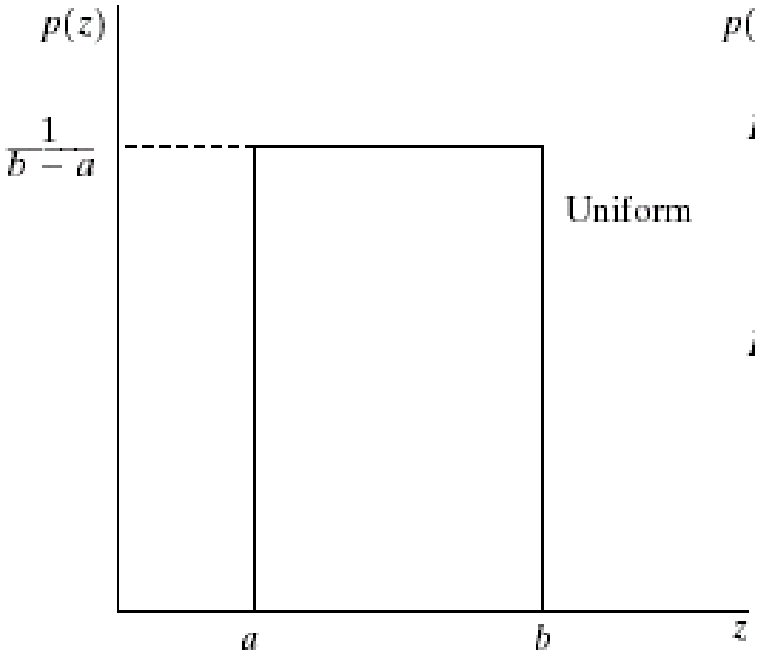
Uniform noise

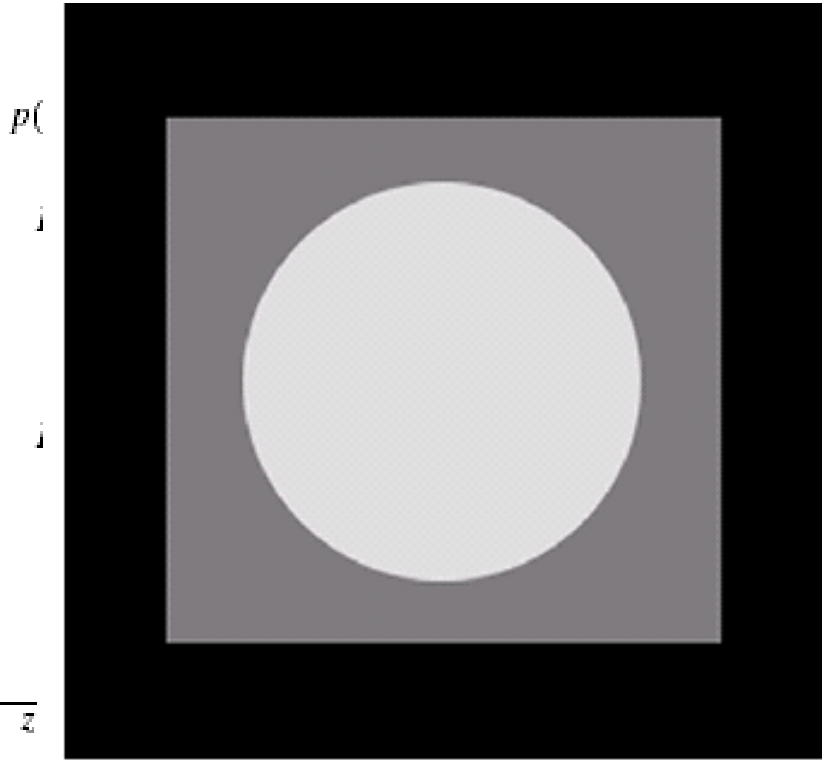


$$p(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}$$

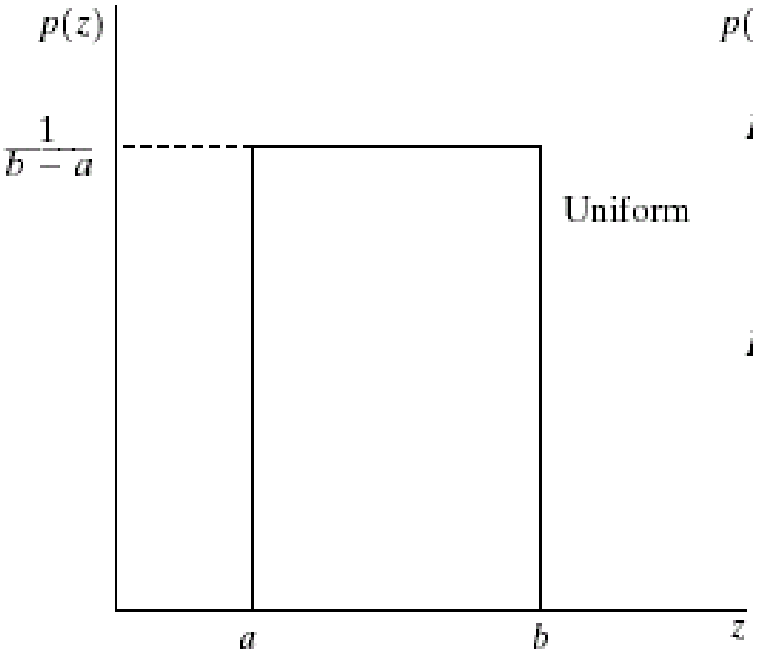
- $\mu = \frac{a+b}{2}$
- $\sigma^2 = \frac{(b-a)^2}{12}$

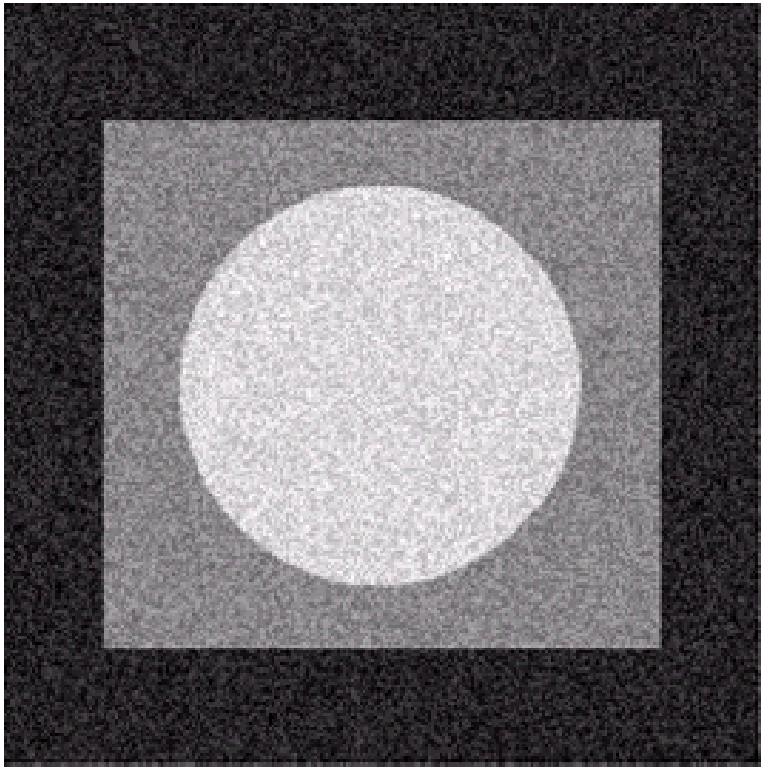
Uniform noise



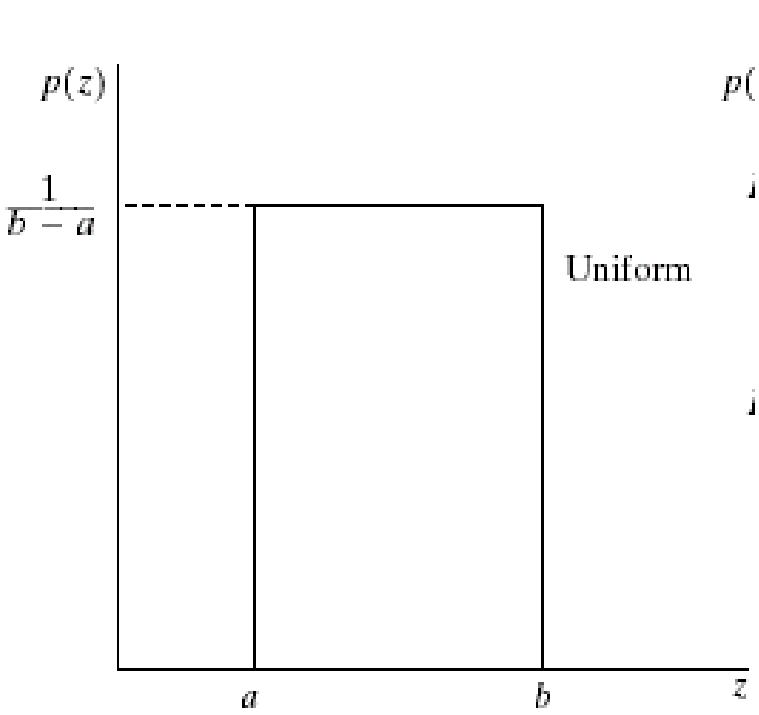


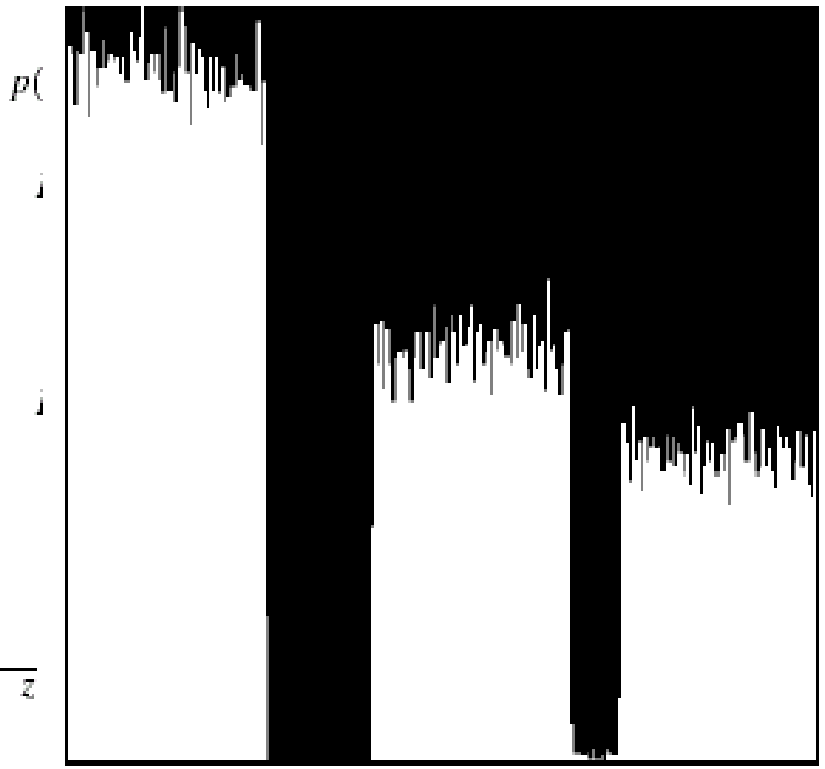
Uniform noise



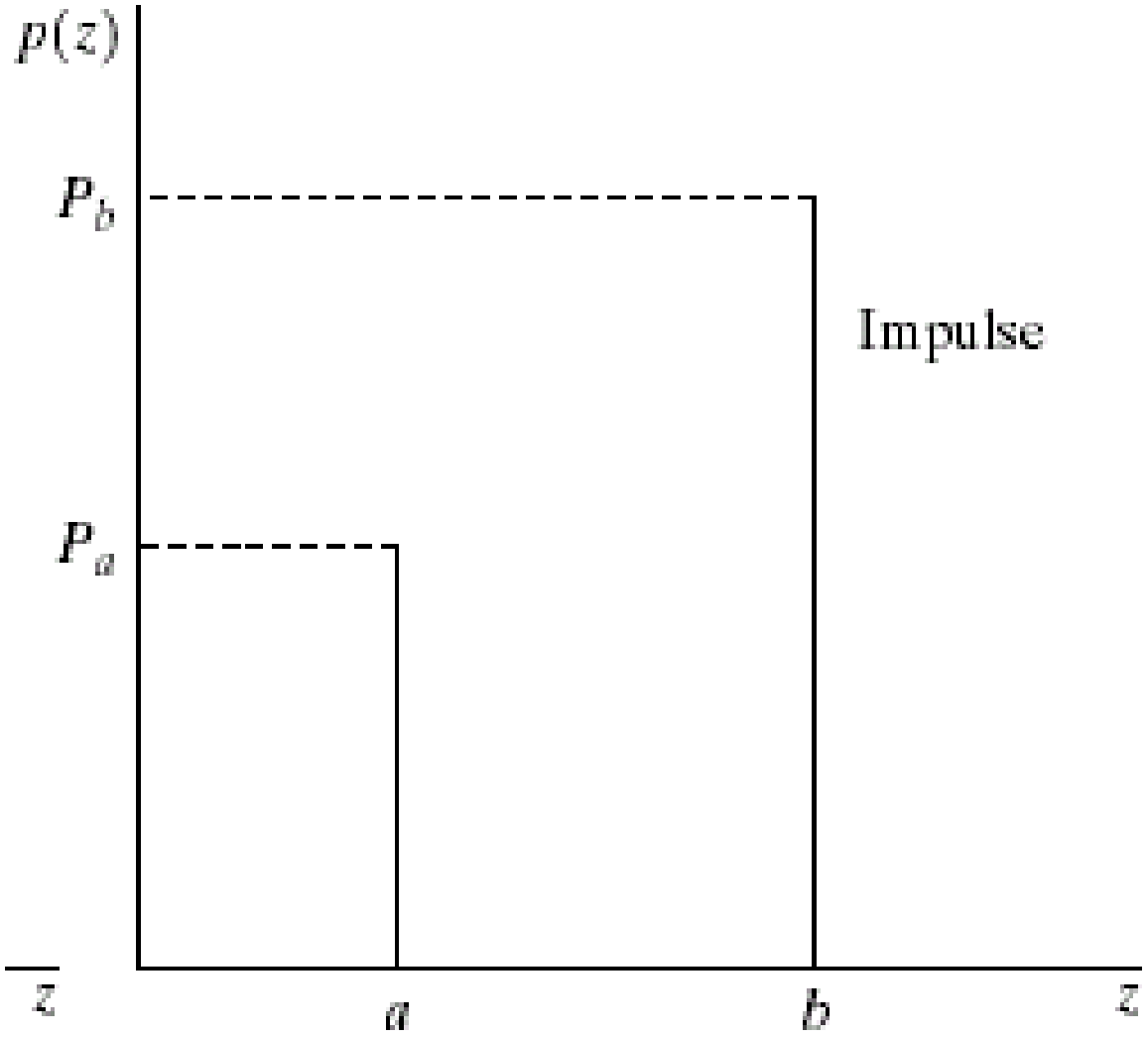


Uniform noise

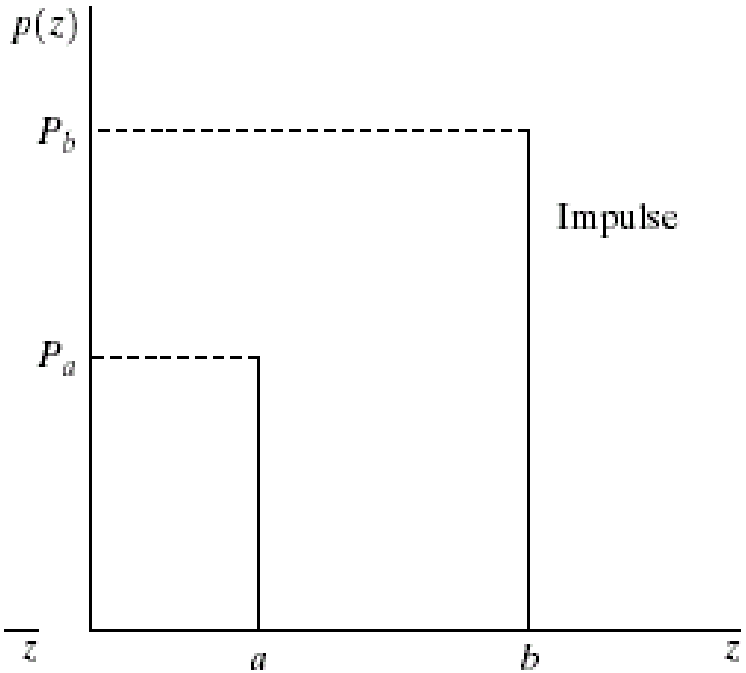




Impulsive noise

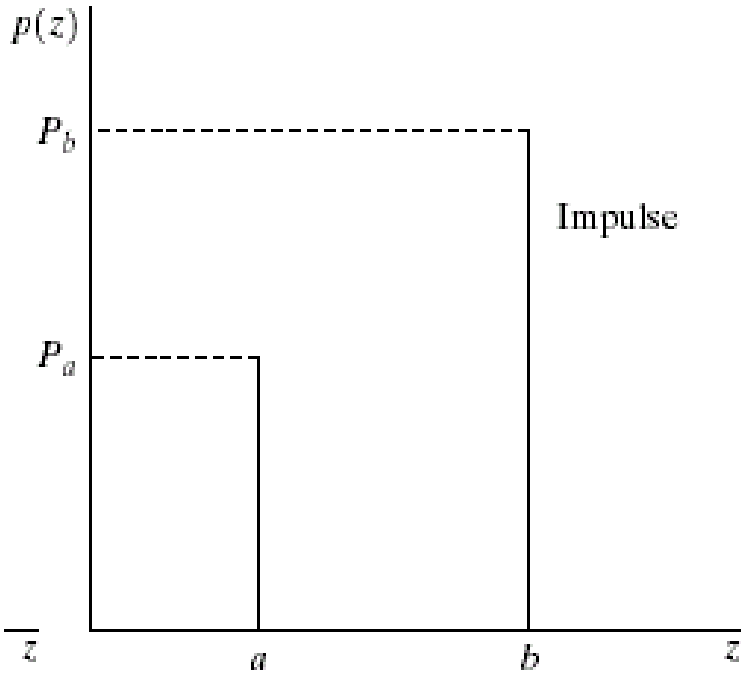


Impulsive noise



$$p(z) = \begin{cases} P_a & \text{if } z = a \\ P_b & \text{if } z = b \\ 0 & \text{otherwise} \end{cases}$$

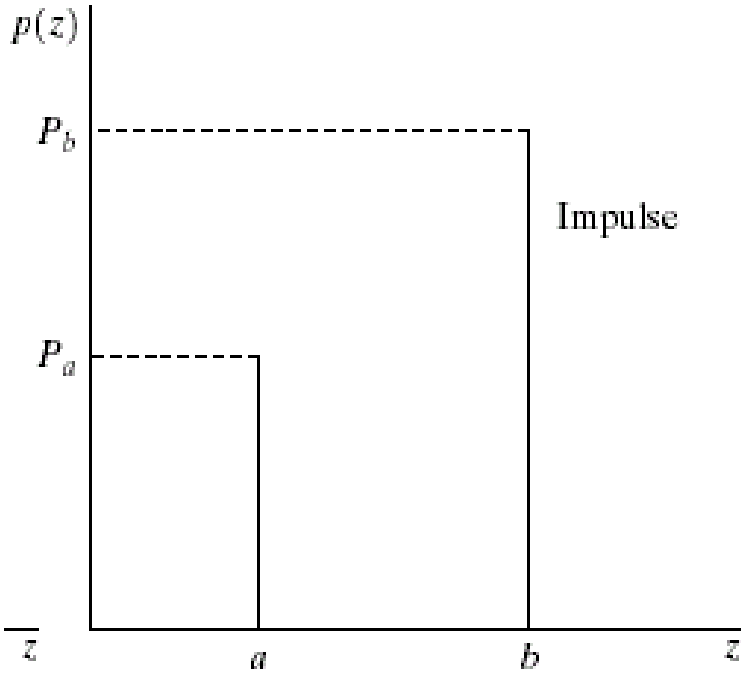
Impulsive noise

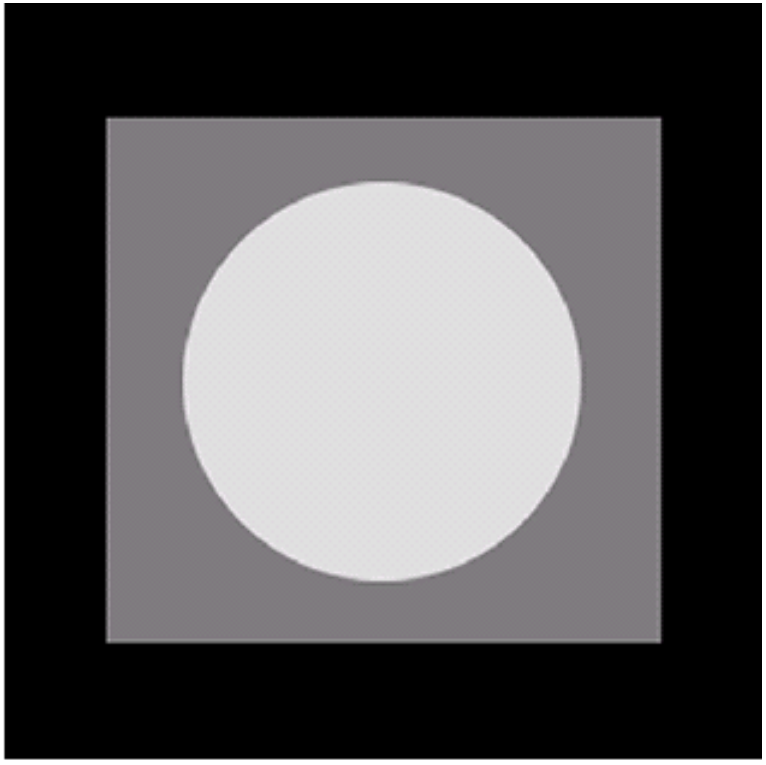


$$p(z) = \begin{cases} P_a & \text{if } z = a \\ P_b & \text{if } z = b \\ 0 & \text{otherwise} \end{cases}$$

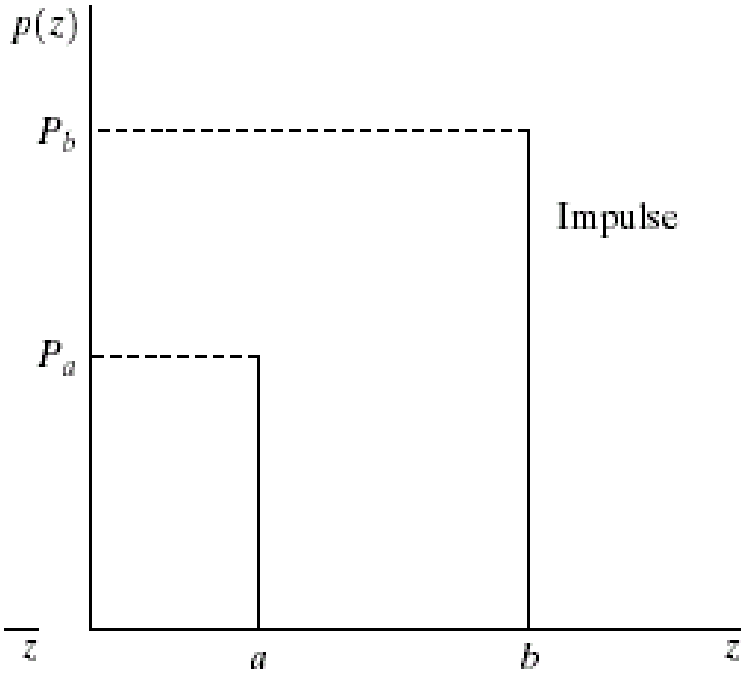
- if $P_a = 0$ or $P_b = 0$ known as *unipolar* noise
- if $P_a \approx P_b$ known as *salt and pepper* noise
- it models the noise caused by errors in the data transmission

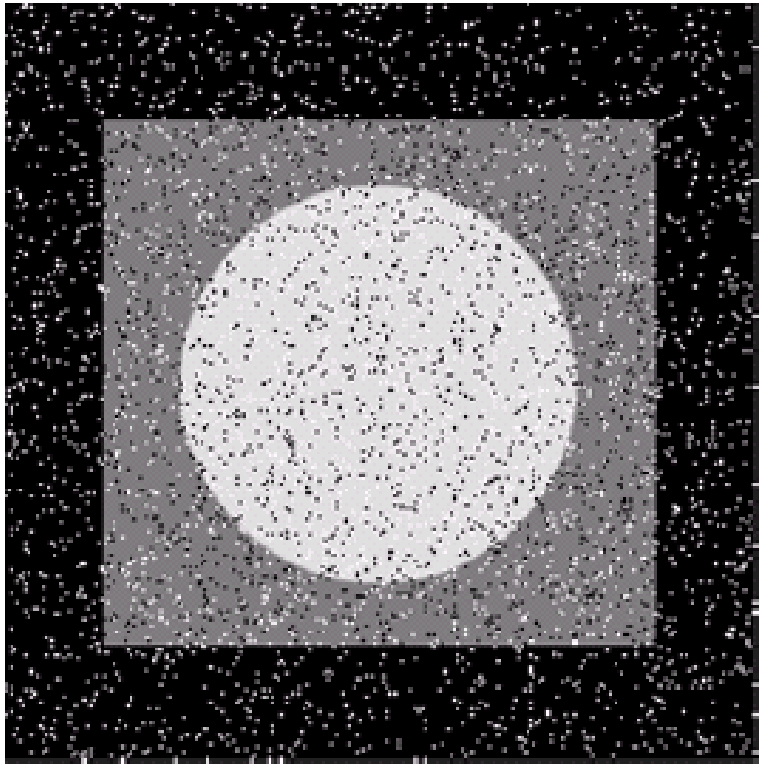
Impulsive noise



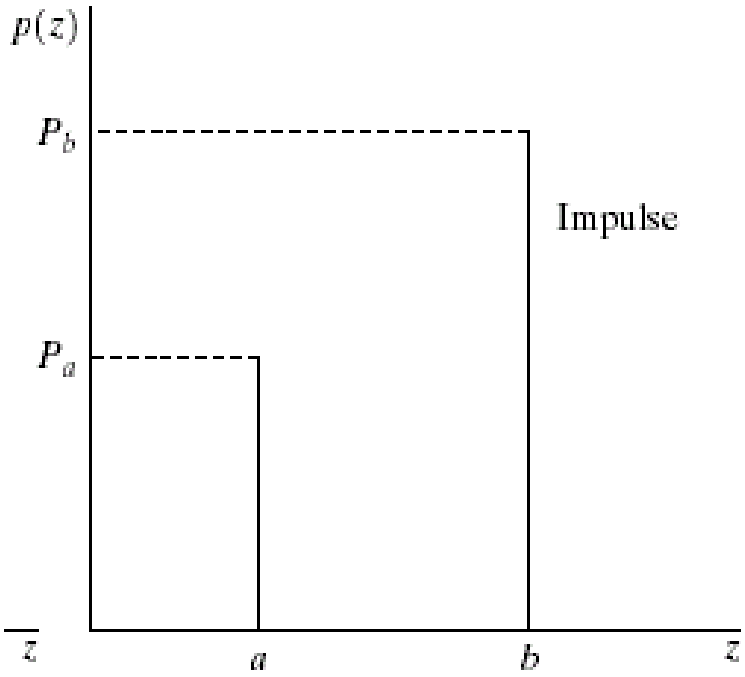


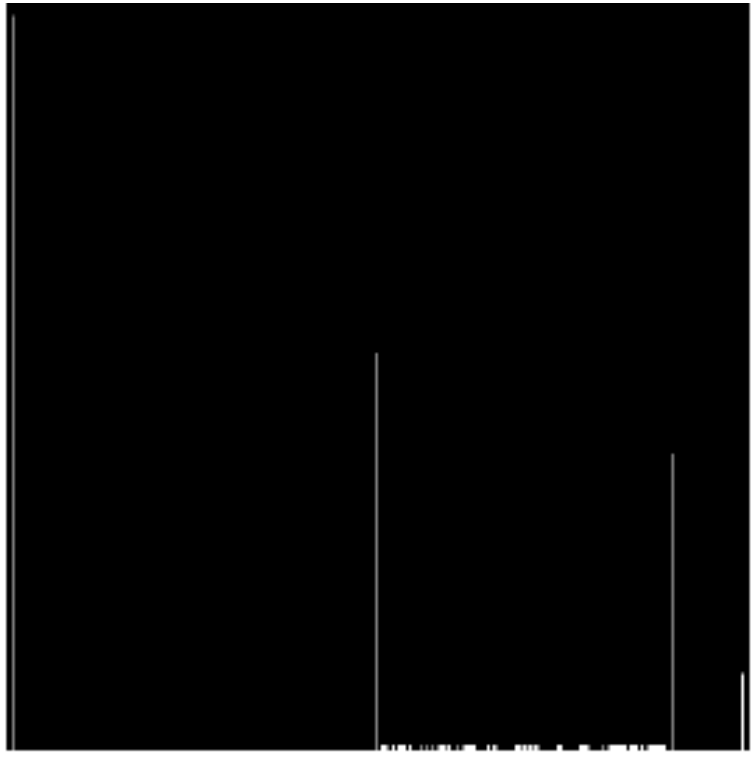
Impulsive noise





Impulsive noise





How do you guess the noise?

How do you guess the noise?

Take a few flat images and study the histogram

How do you guess the noise?

Take a few flat images and study the histogram
What if you get only the images?

How do you guess the noise?

Take a few flat images and study the histogram
What if you get only the images?

Digital filters

Filters are mainly used for:

- smoothing the image
- enhancing or detecting edges in the image

Digital filters

Filters are mainly used for:

- smoothing the image
- enhancing or detecting edges in the image

$$\mathsf{I}'_{ij} = \mathcal{O}_N(\mathsf{I}_{N_{ij}}, \mathsf{h}_N)$$

- h_N kernel of size the size of neighbourhood N
- $\mathcal{O}_N$ an operator that can be
  - linear
  - non-linear
- I'_{ij} called *filter response*

Digital filters

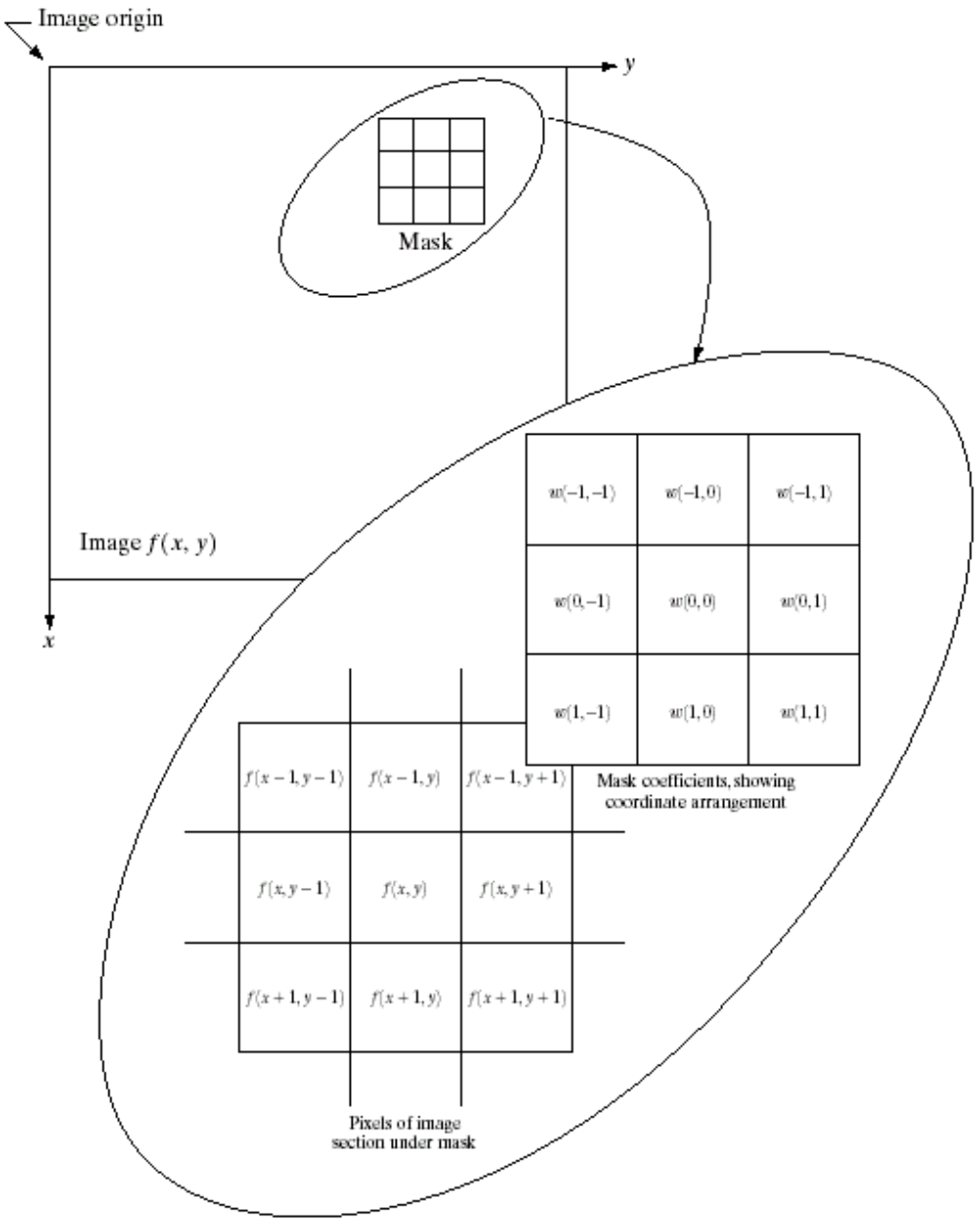
I_{11}	I_{12}	I_{13}	I_{14}	I_{15}	I_{16}	I_{17}	I_{18}	I_{19}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}	I_{26}	I_{27}	I_{28}	I_{29}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}	I_{36}	I_{37}	I_{38}	I_{39}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}	I_{46}	I_{47}	I_{48}	I_{49}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}	I_{56}	I_{57}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

h_{11}	h_{12}	h_{13}
h_{21}	h_{22}	h_{23}
h_{31}	h_{32}	h_{33}

Filter kernel

- size $m \times n$
- $m = 2a + 1$
- $n = 2b + 1$
- sometimes central pixel is $(0, 0)$

Digital filters



Linear filters

Definition

A filter is linear if

$$\mathcal{O}_N(a\mathsf{I}_1 + b\mathsf{I}_2, \mathsf{h}_N) = a\mathcal{O}_N(\mathsf{I}_1, \mathsf{h}_N) + b\mathcal{O}_N(\mathsf{I}_2, \mathsf{h}_N)$$

Linear filters

Definition

A filter is linear if

$$\mathcal{O}_N(a\mathsf{I}_1 + b\mathsf{I}_2, \mathsf{h}_N) = a\mathcal{O}_N(\mathsf{I}_1, \mathsf{h}_N) + b\mathcal{O}_N(\mathsf{I}_2, \mathsf{h}_N)$$

- linear filtering are usually called *convolution*
- kernels are usually called *convolution kernels*

Linear filters

Definition

A filter is linear if

$$\mathcal{O}_N(a\mathsf{I}_1 + b\mathsf{I}_2, \mathsf{h}_N) = a\mathcal{O}_N(\mathsf{I}_1, \mathsf{h}_N) + b\mathcal{O}_N(\mathsf{I}_2, \mathsf{h}_N)$$

- linear filtering are usually called *convolution*
- kernels are usually called *convolution kernels*

Implementation

$$\mathsf{I}'_{ij} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \mathsf{I}_{i+s,j+t} \mathsf{h}_{s-a,t-b}$$

Smoothing filters

What they are

Smoothing filters reduce abrupt changes in the image. Main effects are:

- blur the image
- remove small details
- bridge small gaps
- reduce the noise
- soften the edges

Linear smoothing filters

- mean filter
- weighted average
- Gaussian filter

Mean filter

Definition

$$I'_{ij} = \frac{1}{mn} \sum_{r=-a}^{a} \sum_{s=-b}^{b} I_{i+r,j+s}$$

Example of kernel

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Mean filter

Definition

$$I'_{ij} = \frac{1}{mn} \sum_{r=-a}^{a} \sum_{s=-b}^{b} I_{i+r,j+s}$$

Example of kernel

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Mean filter

Mean filter

3×3

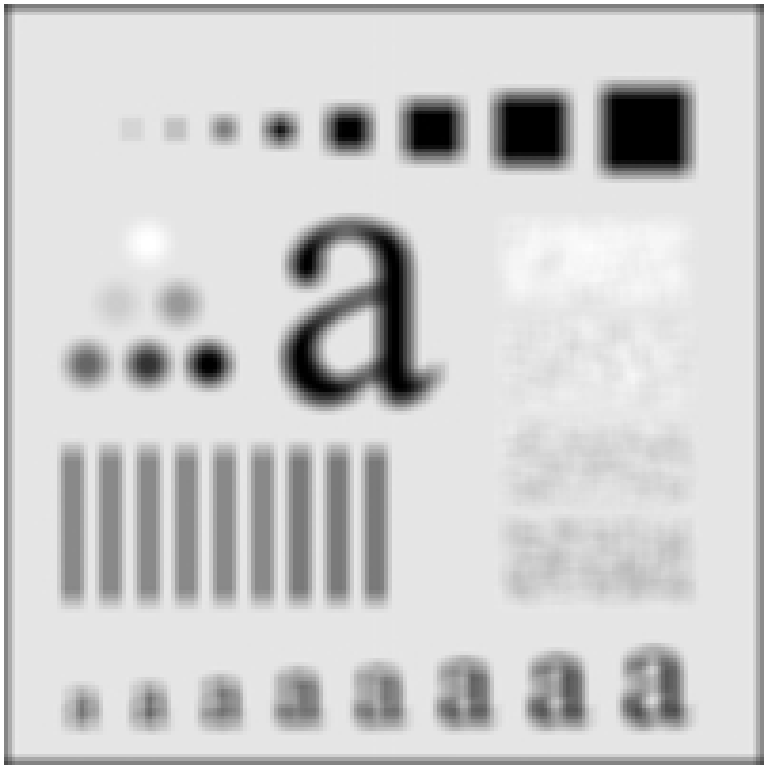
Mean filter

5×5

Mean filter

9×9

Mean filter



15×15

Mean filter

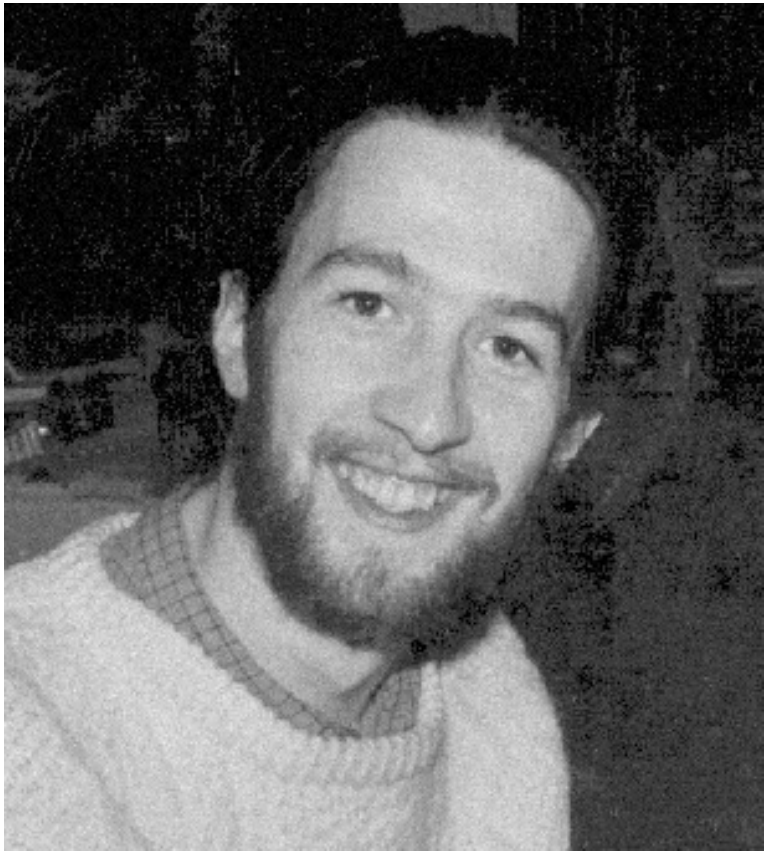
35×35

Mean filter

Original image

Mean filter

Original image



Gaussian noise $\mu = 0$ and $\sigma = 8$

Mean filter

Original image

3×3

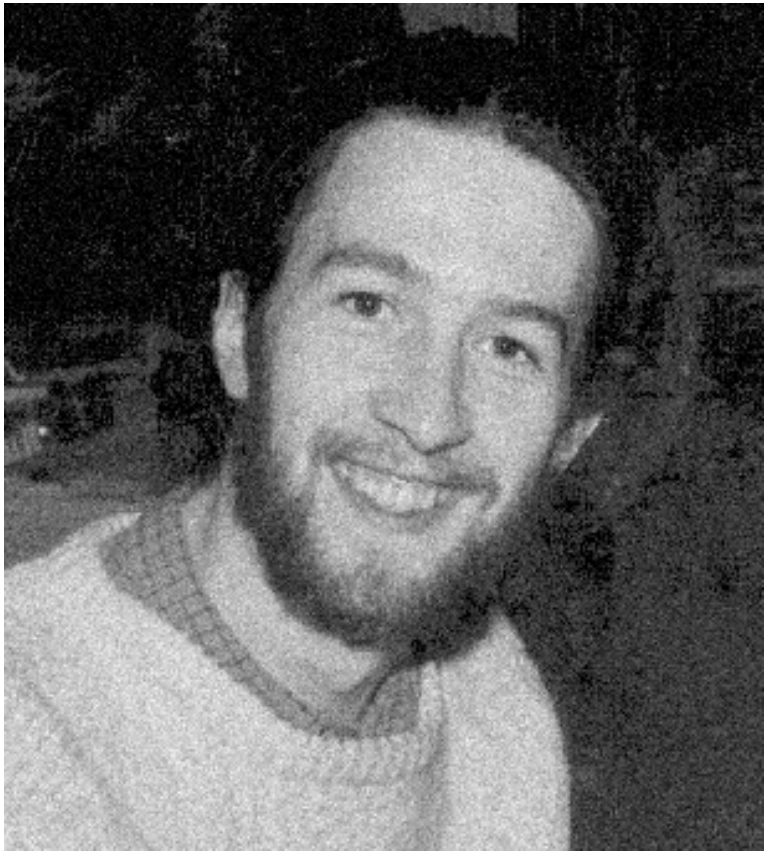
Mean filter

Original image

5×5

Mean filter

Original image



Gaussian noise $\mu = 0$ and $\sigma = 13$

Mean filter

Original image

3×3

Mean filter

Original image

salt and pepper noise

Mean filter

Original image

3×3

Mean filter

Original image

5×5

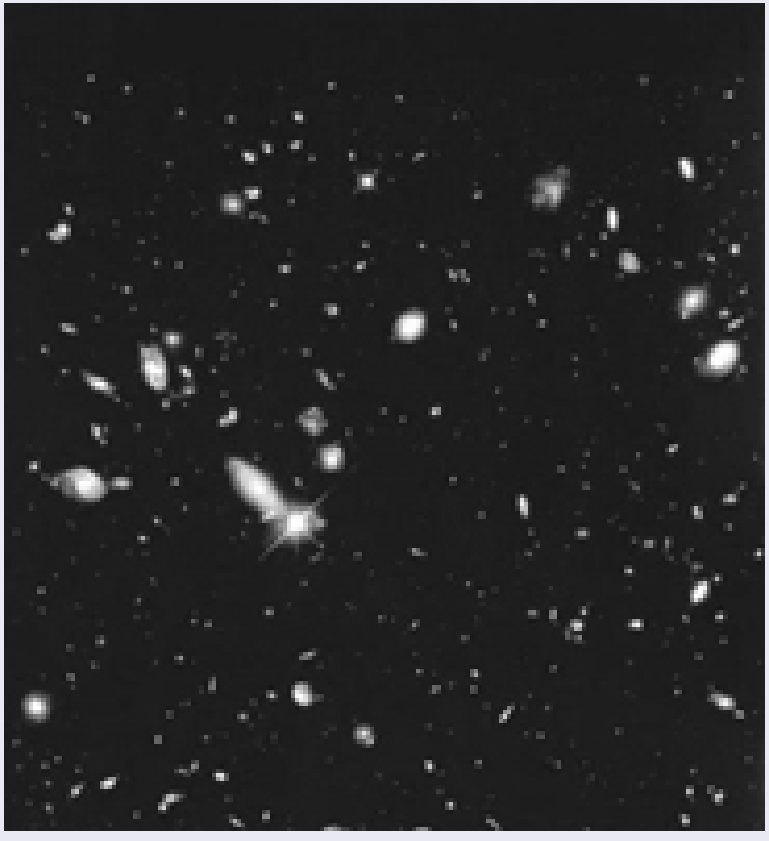
Mean filter

Main problems

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighbourhood
- When the filter neighbourhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.

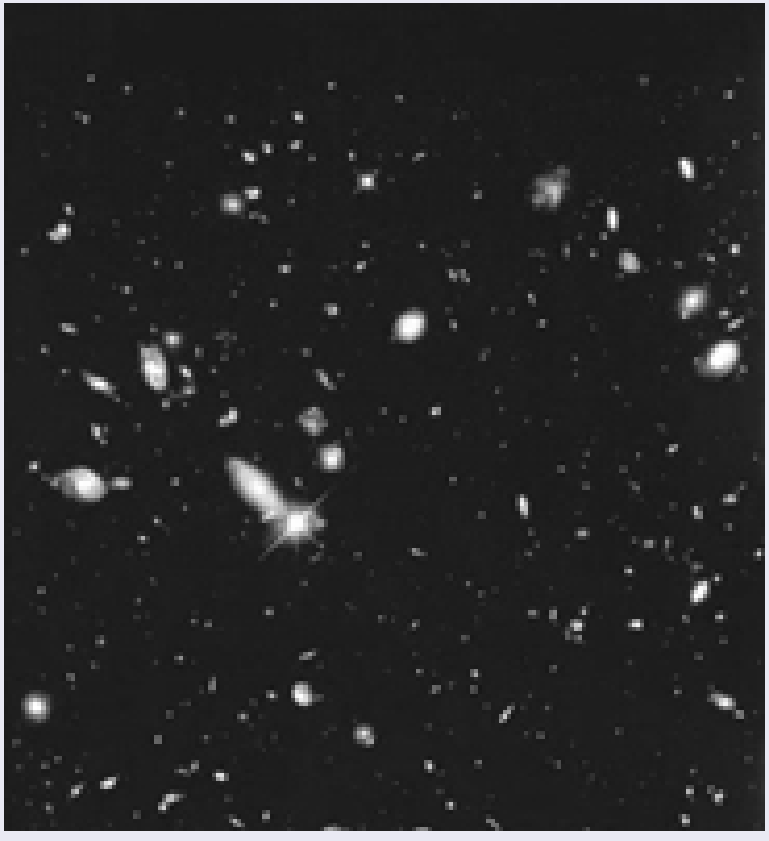
Mean filter

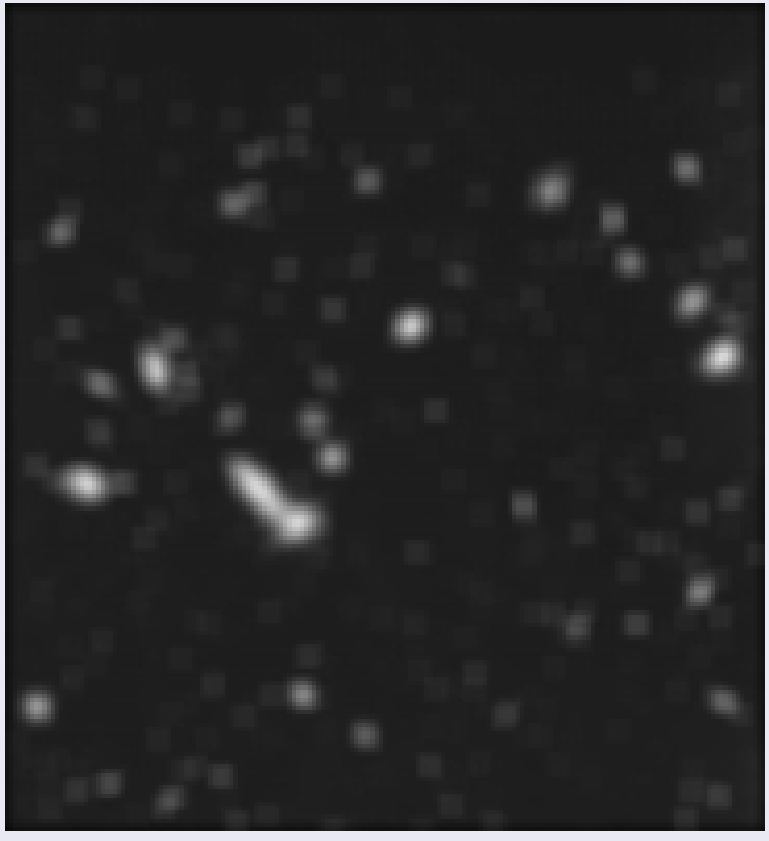
Filter objects of interest



Mean filter

Filter objects of interest

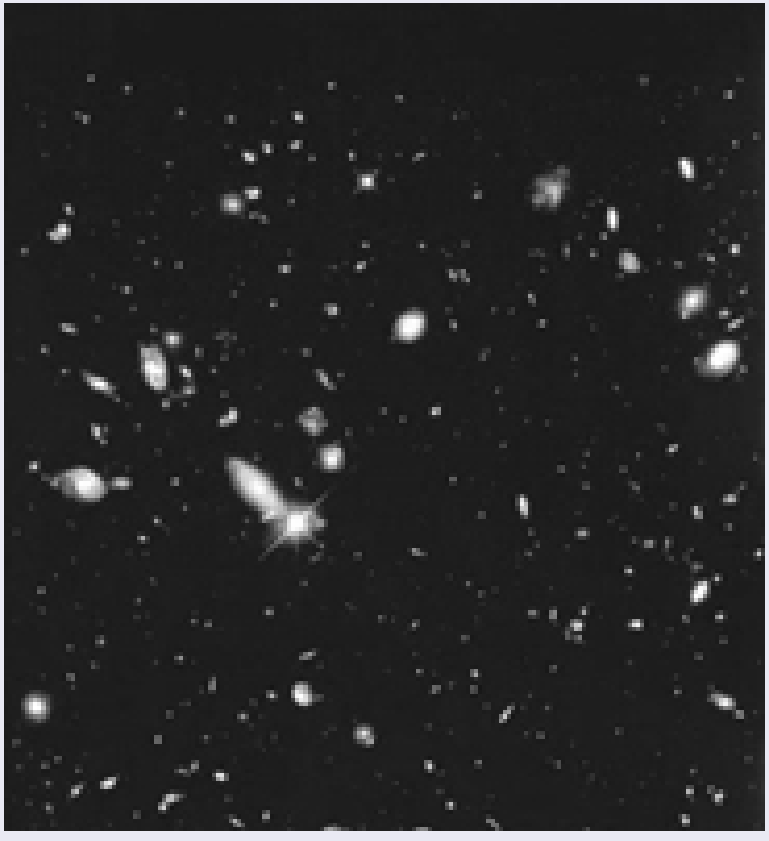




15×15

Mean filter

Filter objects of interest



Thresholded

Mean filter

What about computation?

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Mean filter

What about computation?

$$\begin{array}{|c|c|c|}\hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \hline \end{array}$$

Much better

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Weighted average

$$I'_{ij} = \sum_{s=-a}^{a} \sum_{t=-b}^{b} I_{i+s,j+t} h_{s-a,t-b}$$

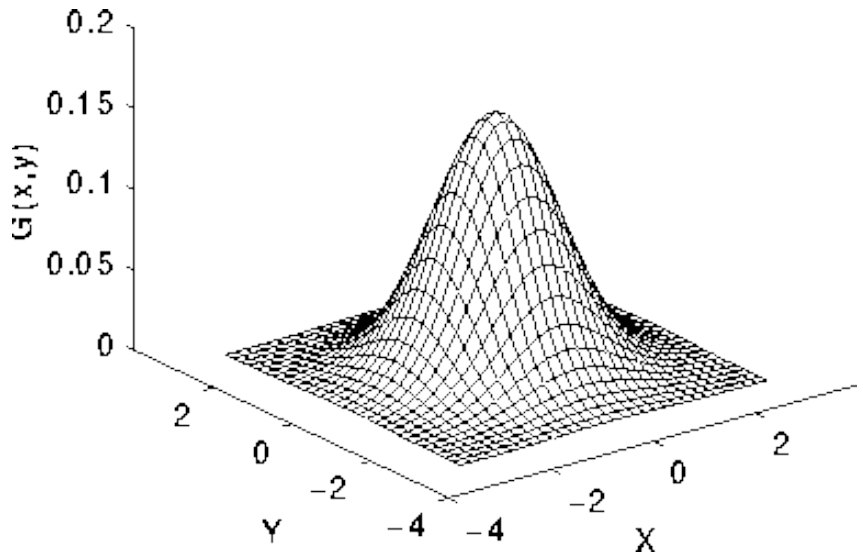
Much better to have $h_{st} \epsilon \mathbb{Z}$ and compute

$$I'_{ij} = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} I_{i+s,j+t} h_{s-a,t-b}}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} h_{s-a,t-b}}$$

Gaussian filter

In this case the weights are sampled from a Gaussian surface.
In our bi-dimensional case

$$h_{st} = e^{-\frac{s^2+t^2}{2\sigma^2}}$$



Gaussian filter

How to design a Gaussian kernel

- we want that most of the area under the Gaussian is covered
- we need a relation between the mask size and the σ
- take $2a + 1 = 5\sigma$ so that 98.76% of the area is covered

$$a = \frac{5\sigma - 1}{2}$$

Gaussian filter

How to design a Gaussian kernel

- we want that most of the area under the Gaussian is covered
- we need a relation between the mask size and the σ
- take $2a + 1 = 5\sigma$ so that 98.76% of the area is covered

$$a = \frac{5\sigma - 1}{2}$$

Algorithm

1. Compute the floating point kernel h_{st}
2. find $h_{min} = \min_{st} \mathrm{h}_{st}$
3. $\mathrm{h}_{st} = round(h_{min}^{-1} \mathrm{h}_{st})$

Gaussian filter

Gaussian kernels are separable

$$\sum_{s=-a}^{a}\sum_{t=-b}^{b} e^{-\frac{s^2+t^2}{2\sigma^2}} \mathsf{I}_{i+t,j+s} = \sum_{s=-a}^{a} e^{-\frac{s^2}{2\sigma^2}} \left(\sum_{t=-b}^{b} e^{-\frac{t^2}{2\sigma^2}} \mathsf{I}_{i+t,j+s} \right)$$

Gaussian filter

Gaussian kernels are separable

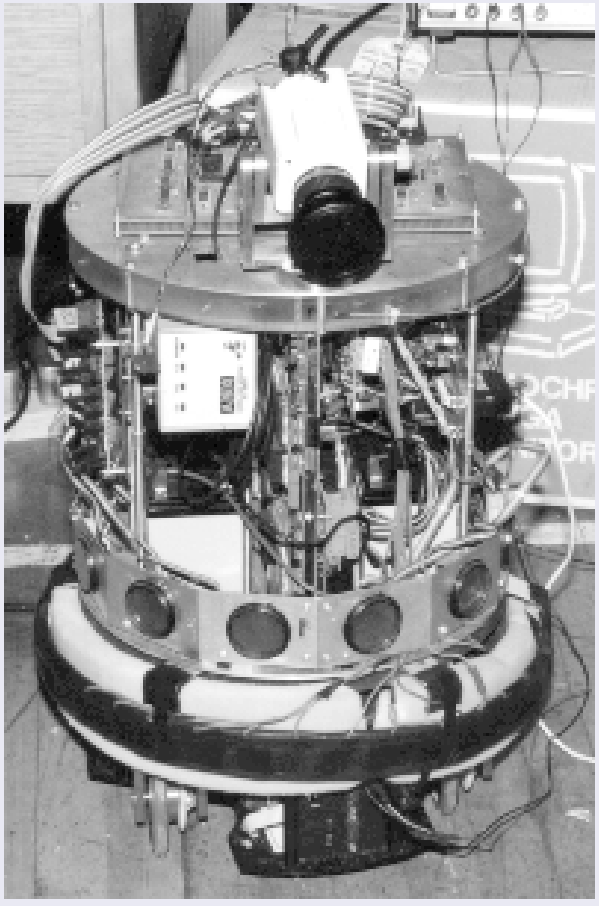
$$\sum_{s=-a}^{a}\sum_{t=-b}^{b} e^{-\frac{s^2+t^2}{2\sigma^2}} \mathsf{I}_{i+t,j+s} = \sum_{s=-a}^{a} e^{-\frac{s^2}{2\sigma^2}} \left(\sum_{t=-b}^{b} e^{-\frac{t^2}{2\sigma^2}} \mathsf{I}_{i+t,j+s} \right)$$

Implementation

- 2 cascading convolutions with a 1-dimensional masks
- first convolve all columns
- then convolve all rows
- time complexity is $O(n)$ and not $O(n^2)$ (n is the mask size)

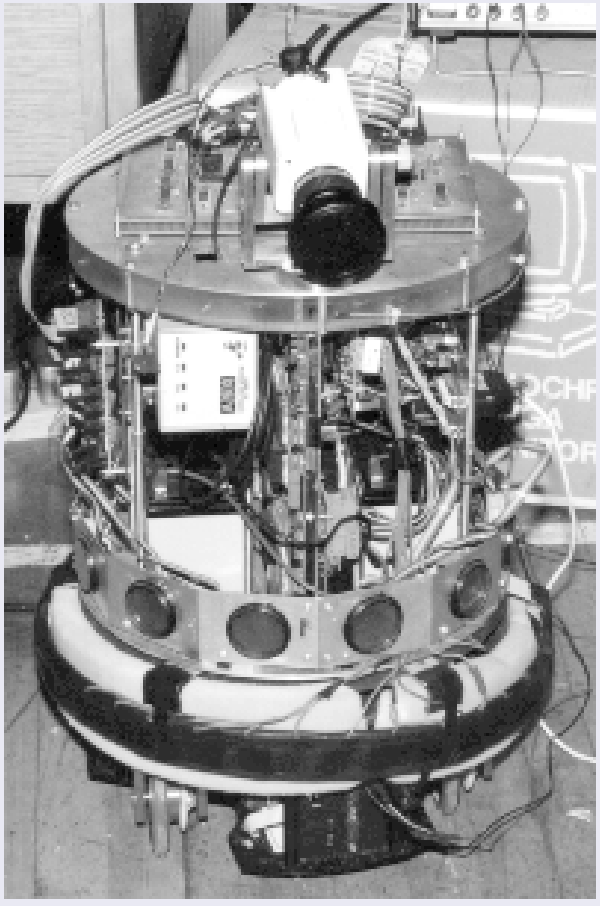
Gaussian filter

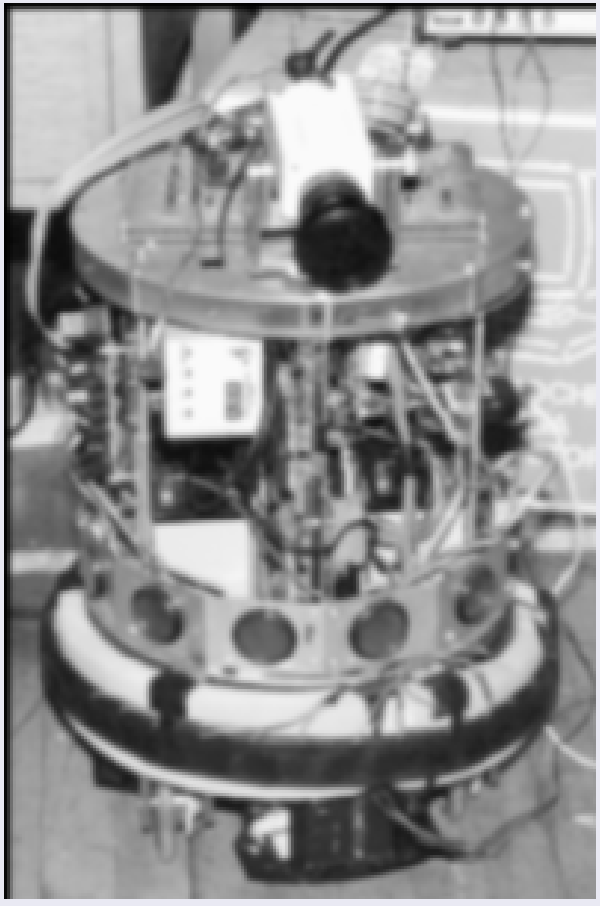
Effects of smoothing



Gaussian filter

Effects of smoothing

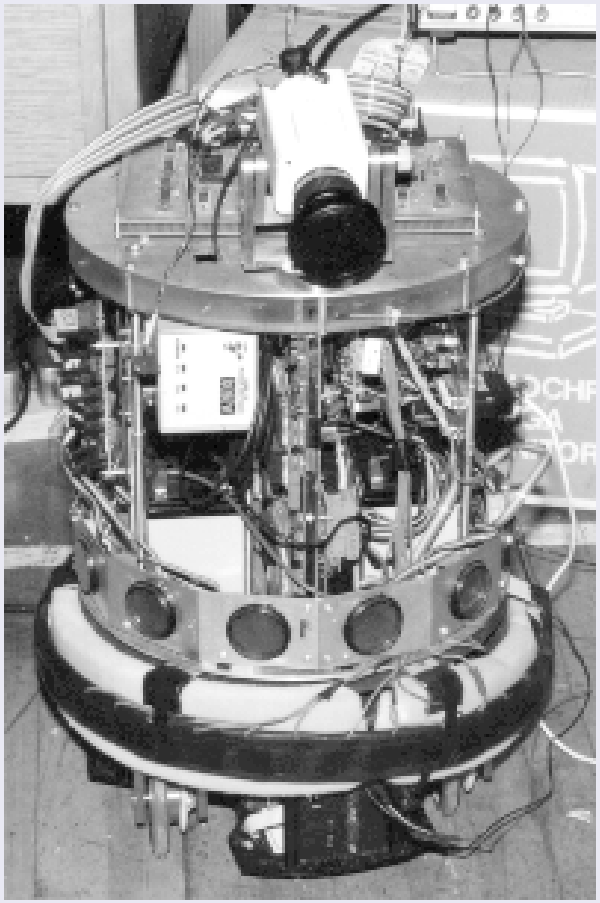


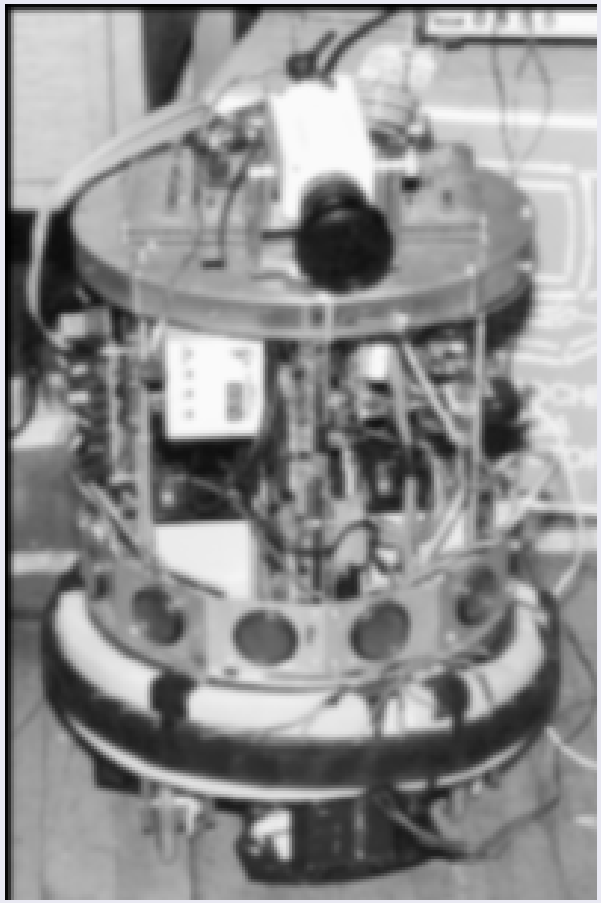


$\sigma = 1\ 5 \times 5$

Gaussian filter

Effects of smoothing

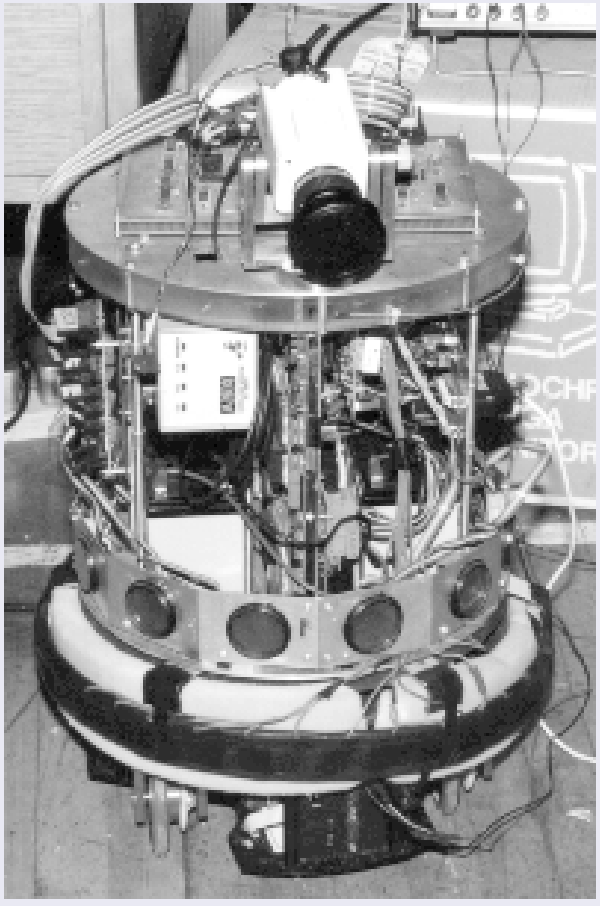


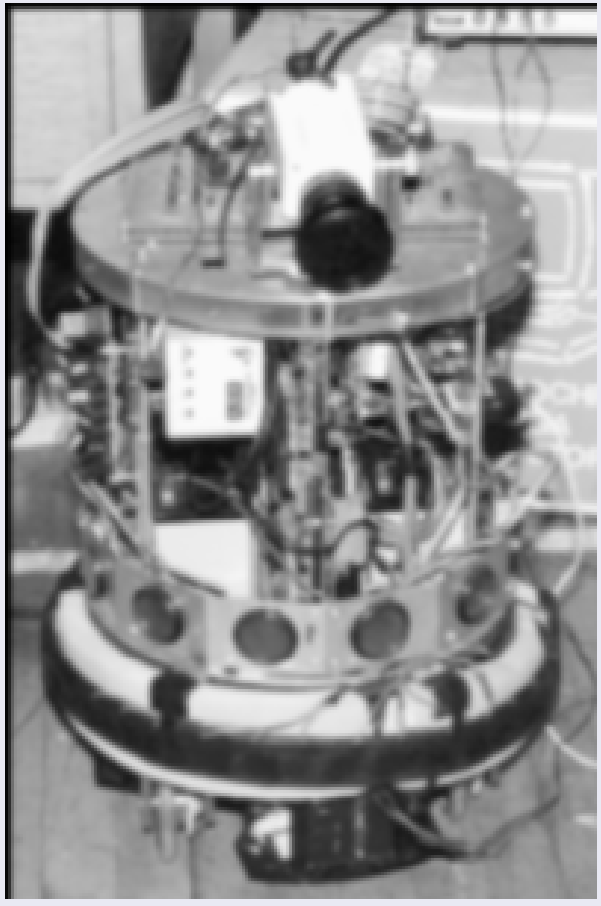


$\sigma = 2\ 9 \times 9$

Gaussian filter

Effects of smoothing

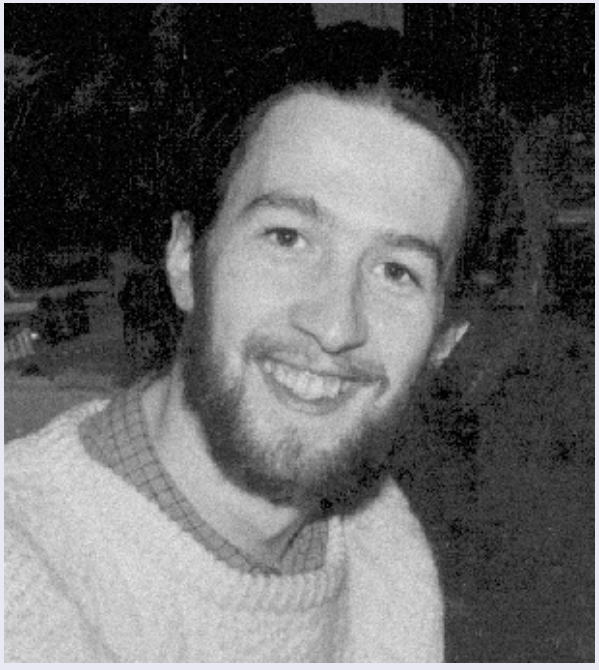




$\sigma = 4$ 15×15

Gaussian filter

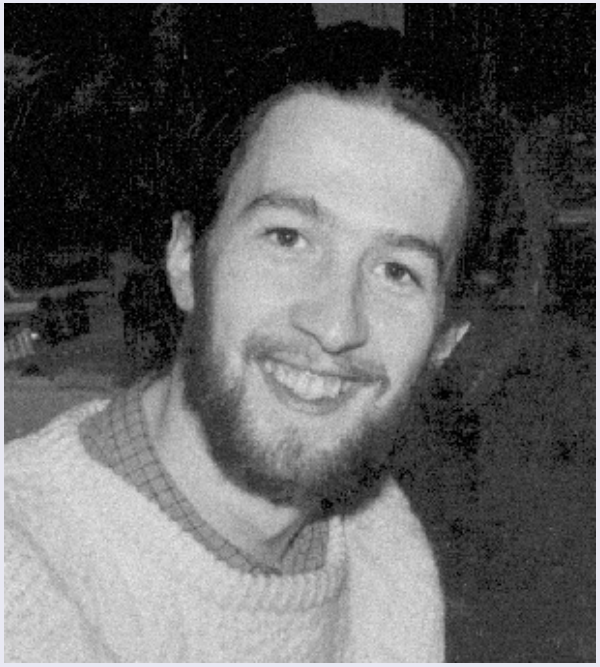
Noise reduction



Gaussian noise $\mu = 0$ and $\sigma = 8$

Gaussian filter

Noise reduction

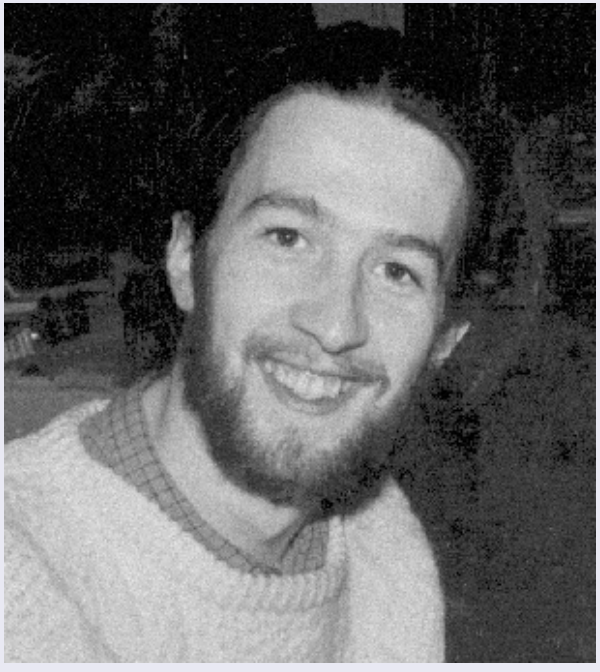


Gaussian noise $\mu = 0$ and $\sigma = 8$

5×5

Gaussian filter

Noise reduction



Gaussian noise $\mu = 0$ and $\sigma = 8$

mean 5×5

Order-statistics filters

Non-linear spatial filters the response of which is based on ranking the pixels in the area covered by the filter.

Order-statistics filters

Non-linear spatial filters the response of which is based on ranking the pixels in the area covered by the filter.

Examples

- median filter
- conservative smoothing

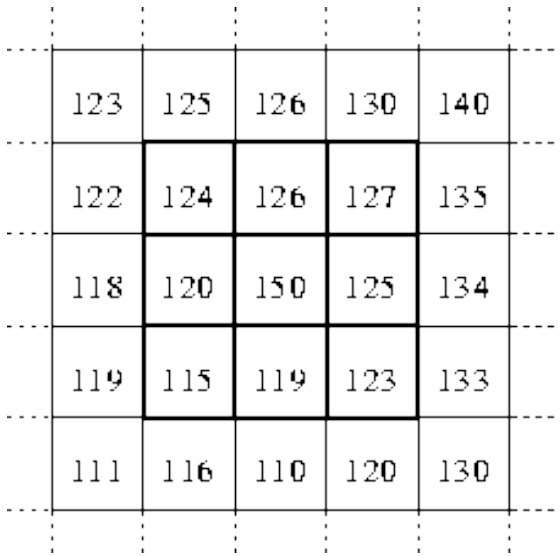
Median filter

$$I'_{ij} = \underset{(s,t)\epsilon N_{ij}}{\text{med}} I_{st}$$

- used to reduce noise
- it does preserve details better than mean filters

Median filter

$$I'_{ij} = \underset{(s,t)\epsilon N_{ij}}{\text{med}} I_{st}$$



Neighbourhood values:

115, 119, 120, 123, 124, 125, 126, 127, 150

Median value: 124

Median filter

$$I'_{ij} = \underset{(s,t)\epsilon N_{ij}}{\text{med}} I_{st}$$

Question

How do we implement median filter?

Median filter

$$I'_{ij} = \underset{(s,t)\epsilon N_{ij}}{\text{med}} I_{st}$$

Question

How do we implement median filter?

- sorting

Median filter

$$I'_{ij} = \underset{(s,t)\epsilon N_{ij}}{\text{med}} I_{st}$$

Question

How do we implement median filter?

- sorting
- select Kth largest

Median filter

```
function select(list[1..n], k)
for i from 1 to k
 minIndex = i
 minValue = list[i]
 for j from i+1 to n
   if list[j] < minValue
     minIndex = j
     minValue = list[j]
 swap list[i] and list[minIndex]
return list[k]
```

Median filter

```
function select(list, left, right, k)
select pivotIndex between left and right
pivotNewIndex = partition(list, left, right,
pivotIndex)
if k = pivotNewIndex
 return list[k]
else if k < pivotNewIndex
 return select(list, left, pivotNewIndex-1, k)
else
 return select(list, pivotNewIndex+1, right, k)
```

Median filter

Question

Can it be done in linear time?

Median filter

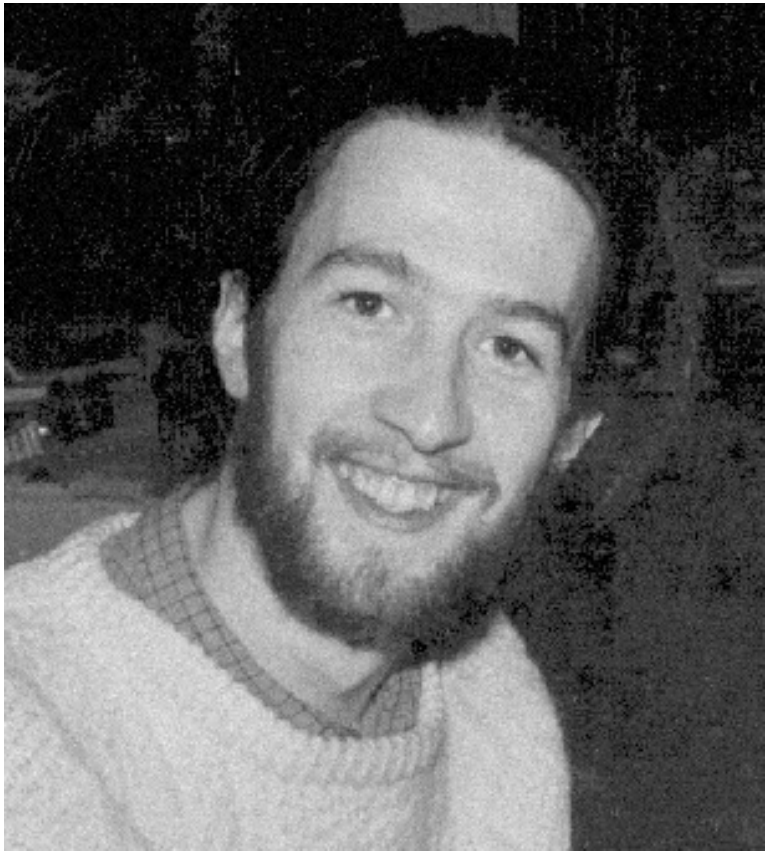
Huang's fast median filter

```
function medianfilter(I, r)
Initialise kernel histogram H
for i = 1 to m
 for j=1 to n
   for k=-r to r
     remove I_{i+k,j-r-1} from H
     add I_{i+k,j+r} to H
   end
   I'_{ij} = median(H)
 end
end
```

Median filter

- the median is a more robust average than the mean
- a single very unrepresentative pixel in a neighbourhood will not affect the median value significantly
- the median value is one of the pixels in the neighbourhood
- the median filter does not create new unrealistic pixel values when the filter straddles an edge
- it is much better at preserving sharp edges than the mean filter

Median filter



Gaussian noise $\mu = 0$ $\sigma = 8$

Median filter

3×3 mean filter

Median filter

3×3 median filter

Median filter

Gaussian noise $\mu = 0$ $\sigma = 13$

Median filter

3×3 median filter

Median filter

3×3 mean filter

Median filter

Salt and pepper noise

Median filter

3×3 median filter

Median filter

3×3 mean filter

Median filter

Salt and pepper noise

Median filter

3×3 median filter

Median filter

7×7 median filter

Median filter

3 3×3 median filter

Median filter

7×7 median filter

3 3×3 median filter

Conservative smoothing

Conservative smoothing employs a simple, fast filtering algorithm that sacrifices noise suppression power in order to preserve details (e.g. sharp edges) in an image.

- more effective with salt and pepper noise
- less effective with additive noise

Conservative smoothing

Filters attempt to make pixel's intensity consistent with those of its nearest neighbours.

- mean filter: average local intensities
- median filter: rank-selection
- conservative smoothing: ensures that each pixel's intensity is bounded within the range of intensities defined by its neighbours

Conservative smoothing

Filters attempt to make pixel's intensity consistent with those of its nearest neighbours.

- mean filter: average local intensities
- median filter: rank-selection
- conservative smoothing: ensures that each pixel's intensity is bounded within the range of intensities defined by its neighbours

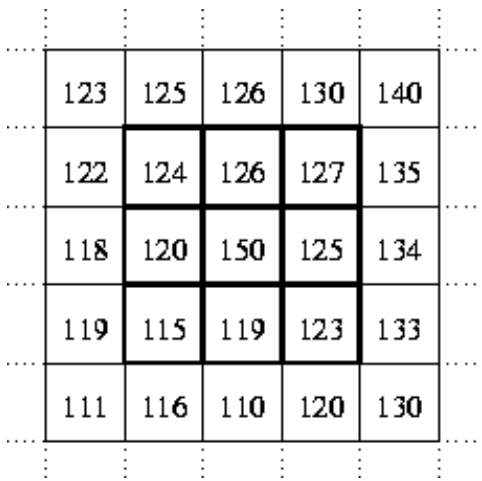
Definition

$$I'_{ij} = \begin{cases} I_{ij} & \text{if } \min(N_{ij}) \leq I_{ij} \leq \max(N_{ij}) \\ \min(N_{ij}) & \text{if } \min(N_{ij}) > I_{ij} \\ \max(N_{ij}) & \text{if } \max(N_{ij}) < I_{ij} \end{cases}$$

Conservative smoothing

Definition

$$I'_{ij} = \begin{cases} I_{ij} & \text{if } \min(N_{ij}) \leq I_{ij} \leq \max(N_{ij}) \\ \min(N_{ij}) & \text{if } \min(N_{ij}) > I_{ij} \\ \max(N_{ij}) & \text{if } \max(N_{ij}) < I_{ij} \end{cases}$$



Neighborhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Max: 127, Min: 115

Conservative smoothing

Gaussian noise $\mu = 0$ $\sigma = 13$

Conservative smoothing

3×3 mean filter

Conservative smoothing

3×3 median filter

Conservative smoothing

3×3 conservative smoothing

Conservative smoothing

- mean filter deals better with Gaussian noise: it inserts intermediate intensity values
- median filter not as good: it uses original intensity values
- median filter preserves details: it uses original intensity values
- conservative smoothing is unable to reduce much Gaussian noise as individual noisy pixel values do not vary much from their neighbours

Conservative smoothing

Salt and pepper noise

Conservative smoothing

3×3 mean filter

Conservative smoothing

3×3 median filter

Conservative smoothing

3×3 conservative smoothing

Conservative smoothing

Salt and pepper noise

Conservative smoothing

3×3 conservative smoothing

Conservative smoothing

3×3 median filter

Adaptive filters

- better performance
- reduce noise
- preserve details (e.g., edges)
- higher complexity

- adaptive local noise reduction filter
- adaptive median filter

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

Formally...

- σ_η noise standard deviation
- $\sigma_{N_{ij}}$ local image standard deviation

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

Formally...

- σ_η noise standard deviation
- $\sigma_{N_{ij}}$ local image standard deviation
- if $\sigma_\eta = 0$ then $I'_{ij} = I_{ij}$

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

Formally...

- σ_η noise standard deviation
- $\sigma_{N_{ij}}$ local image standard deviation
- if $\sigma_\eta = 0$ then $I'_{ij} = I_{ij}$
- if $\sigma_\eta \ll \sigma_{N_{ij}}$ then $I'_{ij} \simeq I_{ij}$

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

Formally...

- σ_η noise standard deviation
- $\sigma_{N_{ij}}$ local image standard deviation
- if $\sigma_\eta = 0$ then $I'_{ij} = I_{ij}$
- if $\sigma_\eta \ll \sigma_{N_{ij}}$ then $I'_{ij} \simeq I_{ij}$
- if $\sigma_\eta \simeq \sigma_{N_{ij}}$ then $I_{ij} = mean(N_{ij})$

Adaptive local noise reduction

What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

$$I'_{ij} = I_{ij} - \frac{\sigma_\eta^2}{\sigma_{N_{ij}}^2}\left(I_{ij} - \mu_{N_{ij}}\right)$$

Adaptive local noise reduction

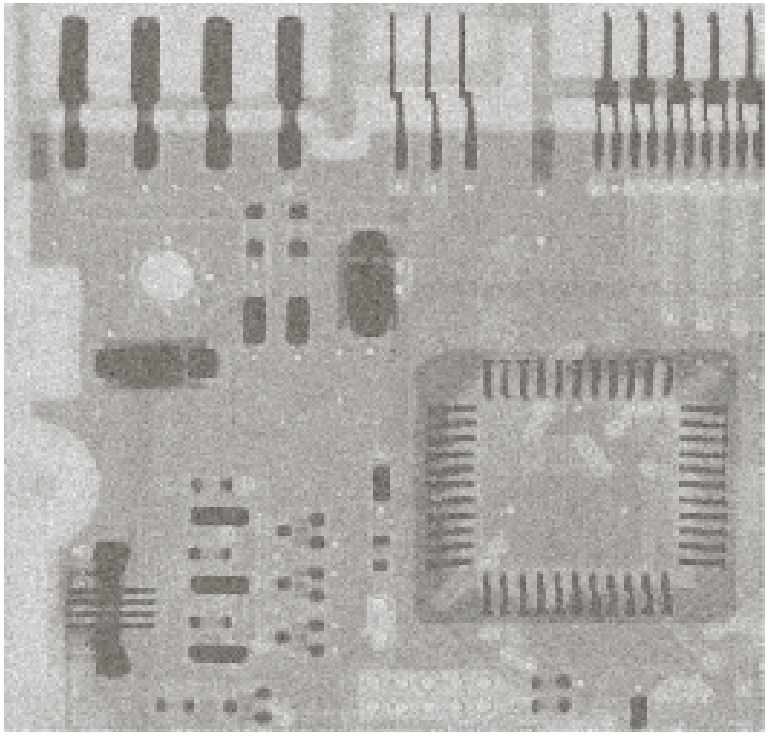
What we want

- nothing if no noise
- small intensity change close to edges
- act as mean filter where image is noisy

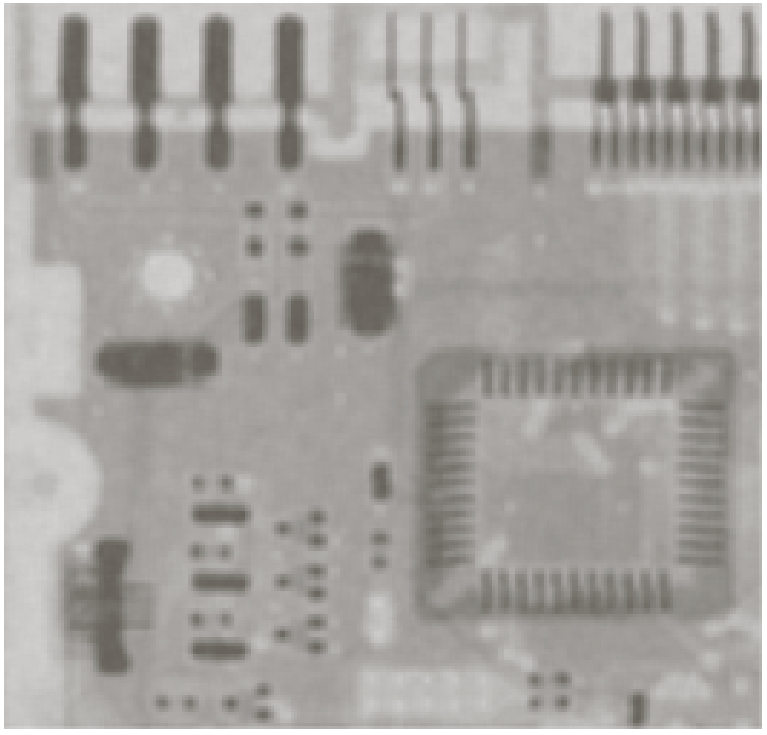
$$I'_{ij} = I_{ij} - \frac{\sigma_\eta^2}{\sigma_{N_{ij}}^2}\left(I_{ij} - \mu_{N_{ij}}\right)$$

It is assumed $\sigma_\eta \leq \sigma_{N_{ij}}$. Otherwise $\frac{\sigma_\eta^2}{\sigma_{N_{ij}}^2}$ is set to 1, preventing negative values.

Adaptive local noise reduction

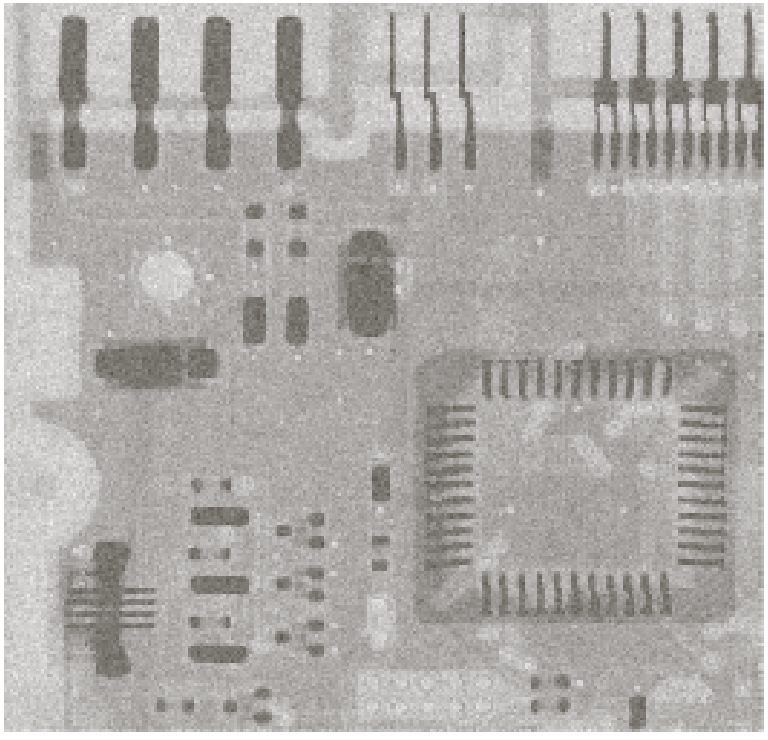


Gaussian noise $\sigma^2 = 1000$

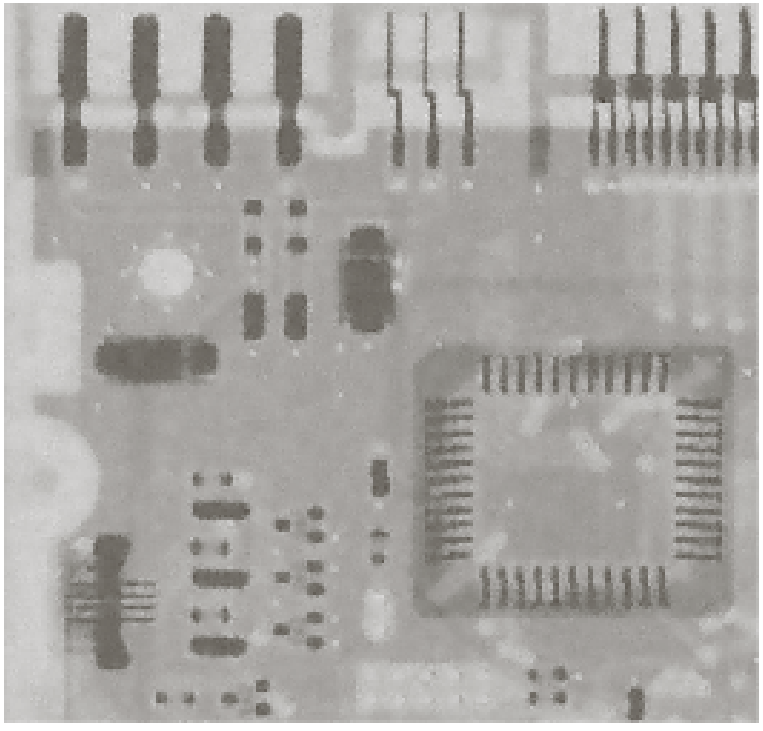


mean filter 7×7

Adaptive local noise reduction

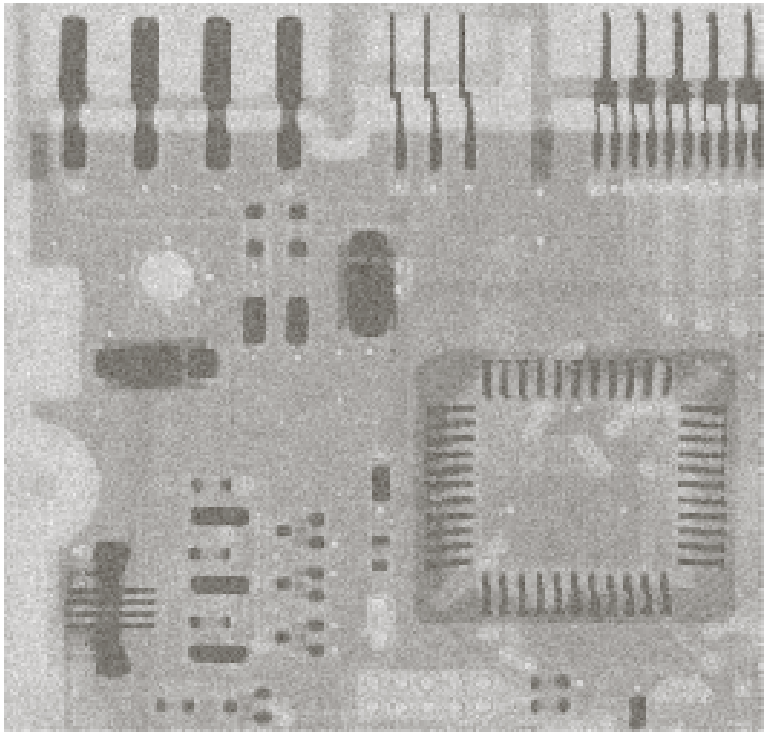


Gaussian noise $\sigma^2 = 1000$

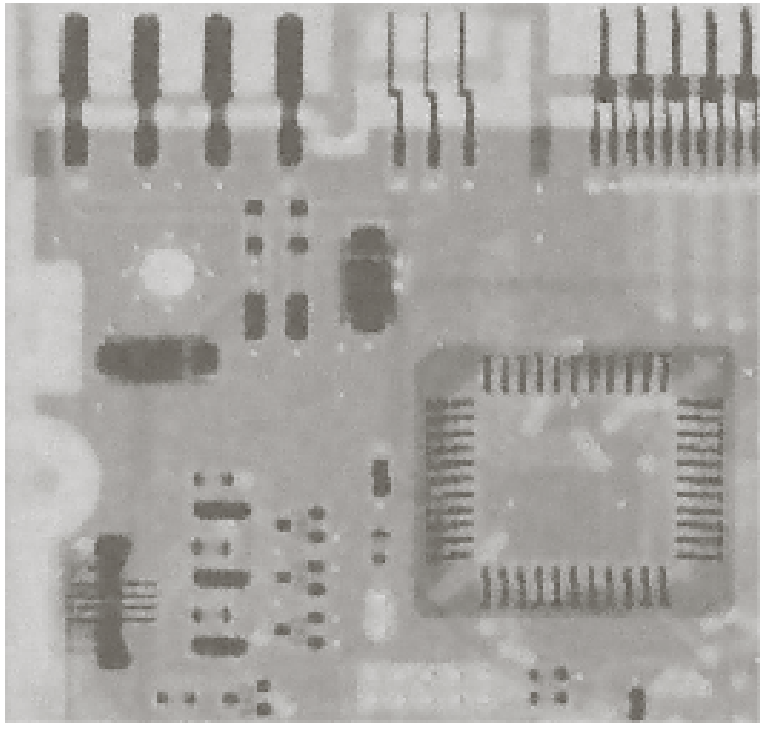


adaptive filter $7 \times 7\, \sigma_\eta^2 = 1000$

Adaptive local noise reduction



Gaussian noise $\sigma^2 = 1000$



adaptive filter 7×7 $\sigma_\eta^2 = 1000$

- not too much correction if σ_η too small: $\mathsf{I}' \simeq \mathsf{I}$
- if σ_η too large loss of details as for the mean filter

Adaptive median filter

The objective is

- remove impulse noise
- preserve object boundaries

Adaptive median filter

The objective is

- remove impulse noise
- preserve object boundaries

- median filter works well if impulse noise not too large
- adaptive median filter can handle larger corruption
- it tries to preserve details

Adaptive median filter

How it works

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse
- if yes enlarge the window

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse
- if yes enlarge the window
- if not check if it can preserve information

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse
- if yes enlarge the window
- if not check if it can preserve information
- check if original value is an impulse

Adaptive median filter

How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse
- if yes enlarge the window
- if not check if it can preserve information
- check if original value is an impulse
- if not leave pixel as it is

Adaptive median filter

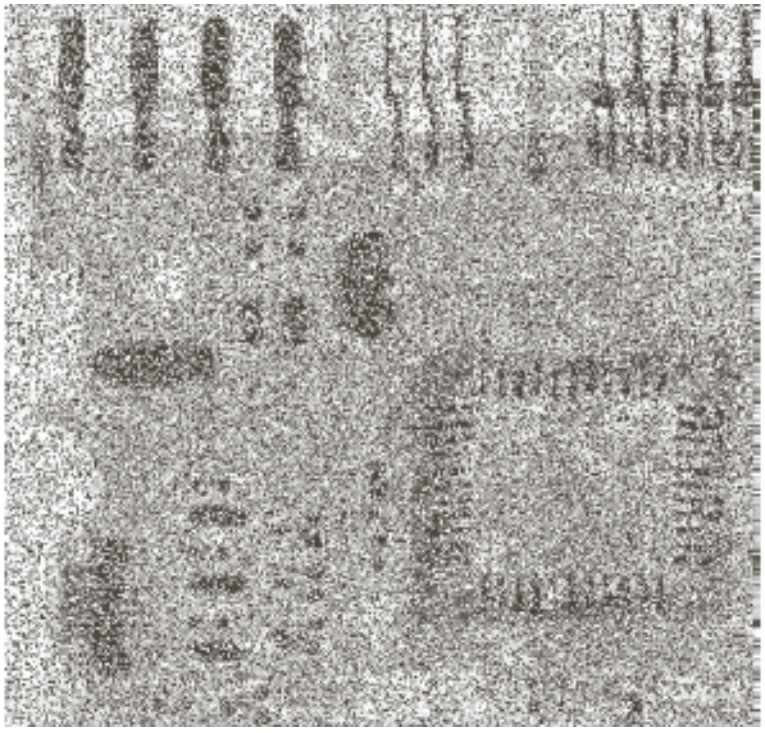
How it works

- problem with large impulse noise is that the median can be an impulse
- first check if median is an impulse
- if yes enlarge the window
- if not check if it can preserve information
- check if original value is an impulse
- if not leave pixel as it is
- otherwise use median value

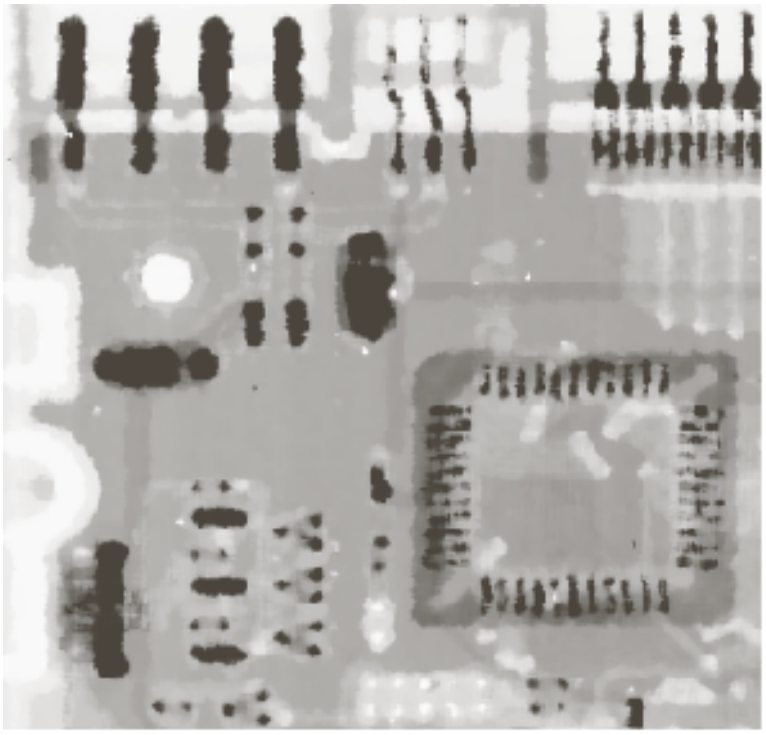
Adaptive median filter

```
function adaptive filter(I, i, j, r)
N = neighbourhood of radius r
if r > r_max then return med_N
if med_N > min_N and med_N < max_N then
 if I_ij > min_N and I_ij < max_N then
    return I_ij
 else
    return med_N
else
 r = r + 1
```

Adaptive median filter

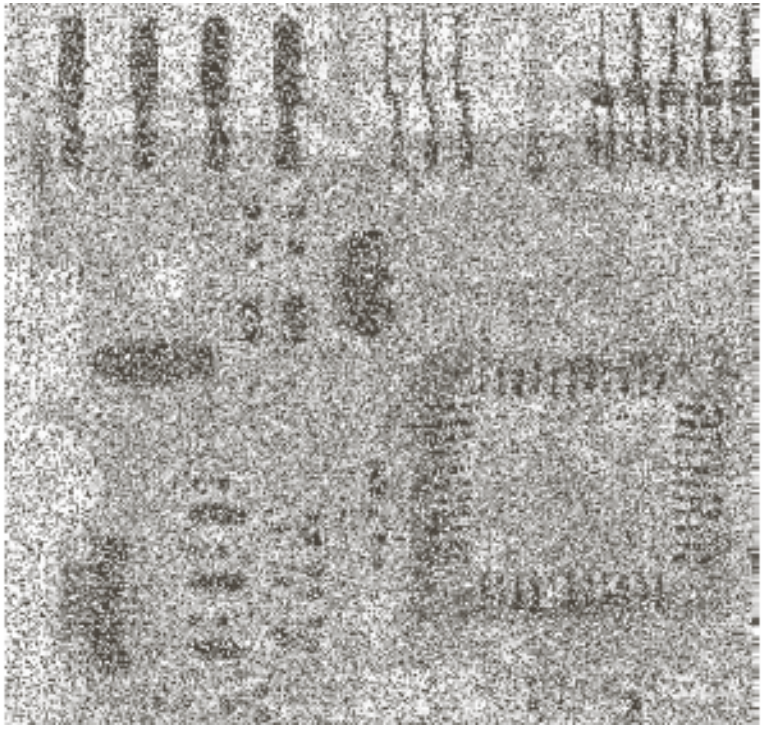


Impulse noise$Pa = P_b = 0.25$

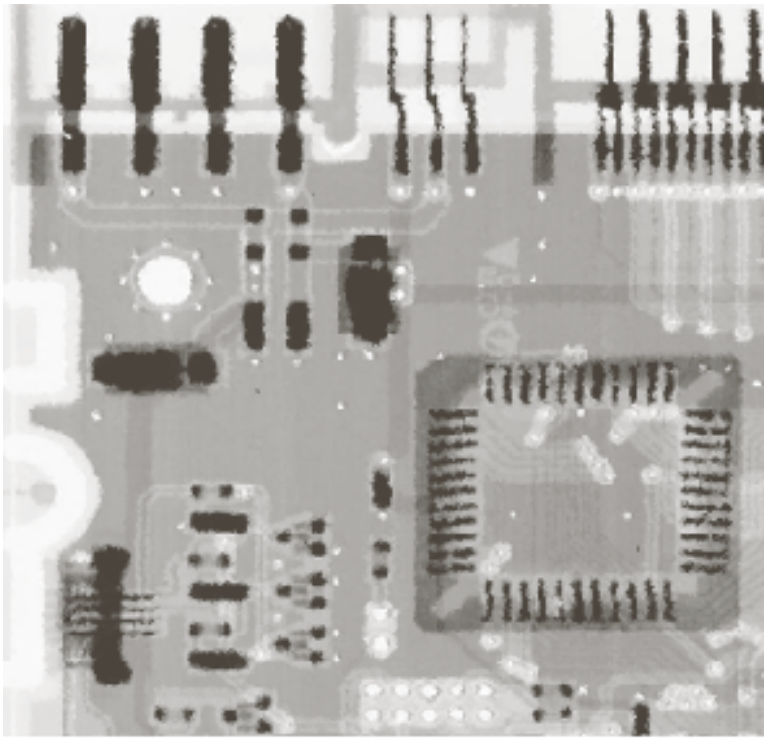


median filter 7×7

Adaptive median filter



Impulse noise$Pa = P_b = 0.25$



adaptive median filter $r_{max} = 7$

Sharpening filter

What we want to achieve

Enhance details in the image.

Sharpening filter

What we want to achieve

Enhance details in the image.
That is enhance hightlight grey level transitions in the images

Sharpening filter

What we want to achieve

Enhance details in the image.
That is enhance hightlight grey level transitions in the images
Main ingredient:

Sharpening filter

What we want to achieve

Enhance details in the image.
That is enhance hightlight grey level transitions in the images
Main ingredient: Spatial differentiation

Derivatives of a digital function

We focus on one-dimensional function.

Derivatives of a digital function

We focus on one-dimensional function.

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Derivatives of a digital function

We focus on one-dimensional function.

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

What is for a digital function?

Derivatives of a digital function

We focus on one-dimensional function.

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

What is for a digital function?

The derivatives are defined in terms of differences.

Derivatives of a digital function

We focus on one-dimensional function.

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

What is for a digital function?

The derivatives are defined in terms of differences.

Derivatives can be defined several ways.

Derivatives of a digital function

Required properties: first derivative

- must be zero in constant intervals
- must be non-zero at the onset of an intensity step or ramp
- must be non-zero along ramps

Derivatives of a digital function

Required properties: first derivative

- must be zero in constant intervals
- must be non-zero at the onset of an intensity step or ramp
- must be non-zero along ramps

Required properties: second derivative

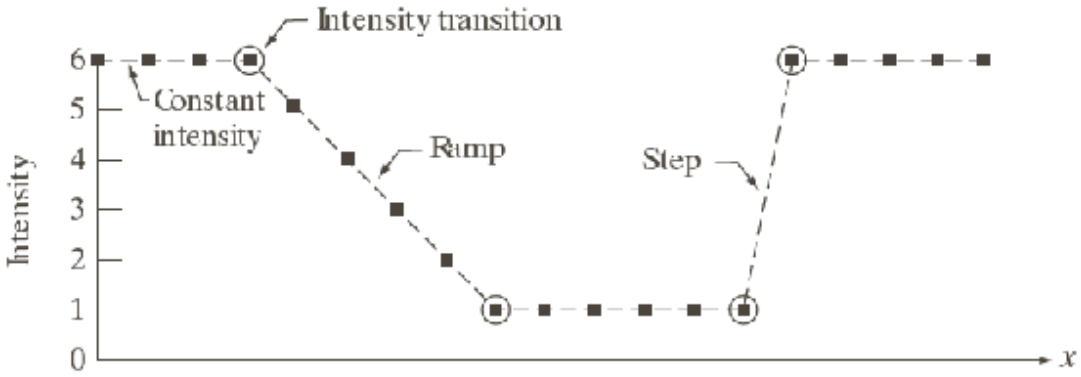
- must be zero in constant intervals
- must be non-zero at the onset and the end of an intensity step or ramp
- must be zero along ramps of constant slope

Derivatives of a digital function

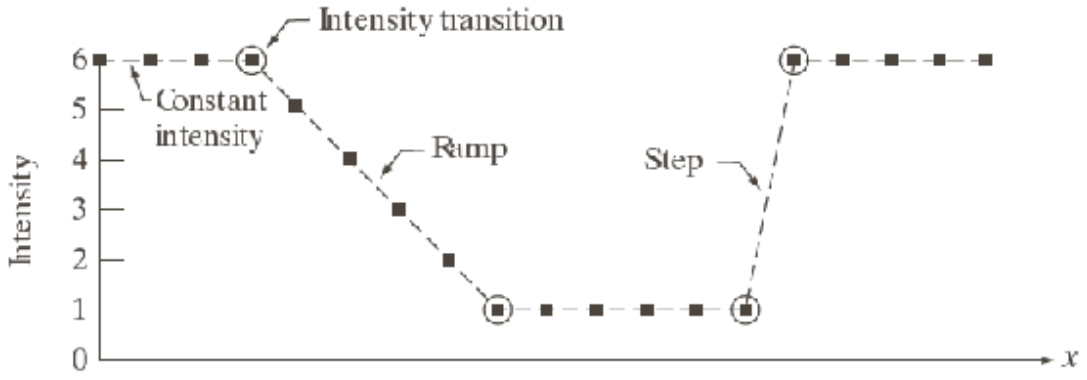
$$f'(x) = f(x+1) - f(x)$$

$$f''(x) = f(x+1) - 2f(x) + f(x-1)$$

Derivatives of a digital function

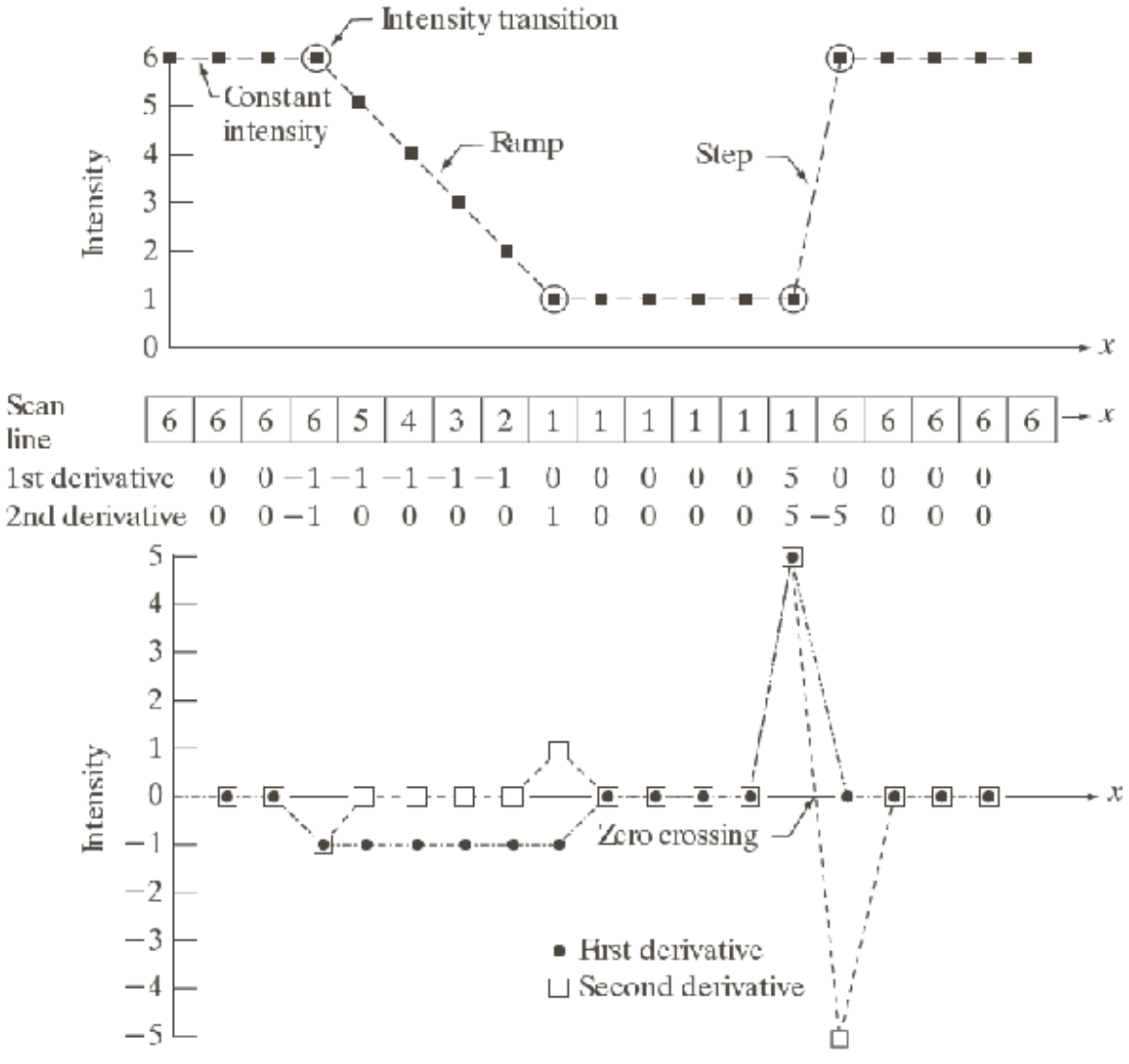


Derivatives of a digital function



Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0		
2nd derivative	0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0		

Derivatives of a digital function



Numerical differentiation

Let us write Taylors polynomial approximation of $f(x)$ at $x+h$ and $x-h$

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) + O(h^3)$$

Numerical differentiation

Let us write Taylors polynomial approximation of $f(x)$ at $x+h$ and $x-h$

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

Subtracting the second from the first

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

Numerical differentiation

Let us write Taylors polynomial approximation of $f(x)$ at $x+h$ and $x-h$

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

Summing up we get

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h)$$

Numerical differentiation

Let us write Taylors polynomial approximation of $f(x)$ at $x+h$ and $x-h$

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

Summing up we get

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h)$$

Using more samples we can derive formulae in which the truncation error vanishes more rapidly.

Numerical differentiation

Let us write Taylors polynomial approximation of $f(x)$ at $x+h$ and $x-h$

$$f(x+h) = f(x) + hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

$$f(x-h) = f(x) - hf'(x) + \tfrac{1}{2}h^2 f''(x) + O(h^3)$$

Summing up we get

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h)$$

Using more samples we can derive formulae in which the truncation error vanishes more rapidly.
These equations are called *central-difference approximations*: they use samples from a symmetric interval.

Numerical differentiation

First derivatives: central difference approximations

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$$

$$f'(x) = \frac{-f(x+2h)+8f(x+h)-8f(x-h)+f(x-2h)}{12h} + O(h^4)$$

Second derivatives: central difference approximations

$$f''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h)$$

$$f''(x) = \frac{-f(x+2h)+16f(x+h)-30f(x)+16f(x-h)-f(x-2h)}{12h^2} + O(h^3)$$

Numerical differentiation

Using asymmetric intervals we get

Forward approximation

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

Backward approximation

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h)$$

Numerical differentiation

Using asymmetric intervals we get

Forward approximation

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

Backward approximation

$$f'(x) = \frac{f(x) - f(x-h)}{h} + O(h)$$

How do we obtain the formulae above?

Computing image derivatives

Computing image derivatives

In an image we compute

$$\frac{\partial \mathsf{I}}{\partial x} = \mathsf{I}(x+1, y) - \mathsf{I}(x-1, y)$$

$$\frac{\partial \mathsf{I}}{\partial y} = \mathsf{I}(x, y+1) - \mathsf{I}(x, y-1)$$

Computing image derivatives

In an image we compute

$$\frac{\partial \mathsf{I}}{\partial x} = \mathsf{I}(x+1, y) - \mathsf{I}(x-1, y)$$

$$\frac{\partial \mathsf{I}}{\partial y} = \mathsf{I}(x, y+1) - \mathsf{I}(x, y-1)$$

The derivatives can be computed convolving rows and colums with the mask

$$[1 \;\; 0 \;\; -1]^t$$

Laplacian

Given a two variables function the Laplacian is defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

For the case of images we may take

$$\frac{\partial^2 I}{\partial x^2} = I(x+1, y) - 2I(x, y) + I(x-1, y)$$

$$\frac{\partial^2 I}{\partial y^2} = I(x, y+1) - 2I(x, y) + I(x, y-1)$$

and then

$$\nabla^2 I(x, y) = I(x+1, y) + I(x-1, y) + I(x, y+1) + I(x, y-1) - 4I(x, y)$$

Laplacian: implementation

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Laplacian: implementation

What is the result if the image rotates?

Sharpening using the Laplacian

The effect of the Laplacian is

- highlights intensity variations
- de-emphasise area with slow intensity variations
- a dark image with discontinuity highlighted

What if we want to include also the background, obtaining a meaningul image?

Sharpening using the Laplacian

The effect of the Laplacian is

- highlights intensity variations
- de-emphasise area with slow intensity variations
- a dark image with discontinuity highlighted

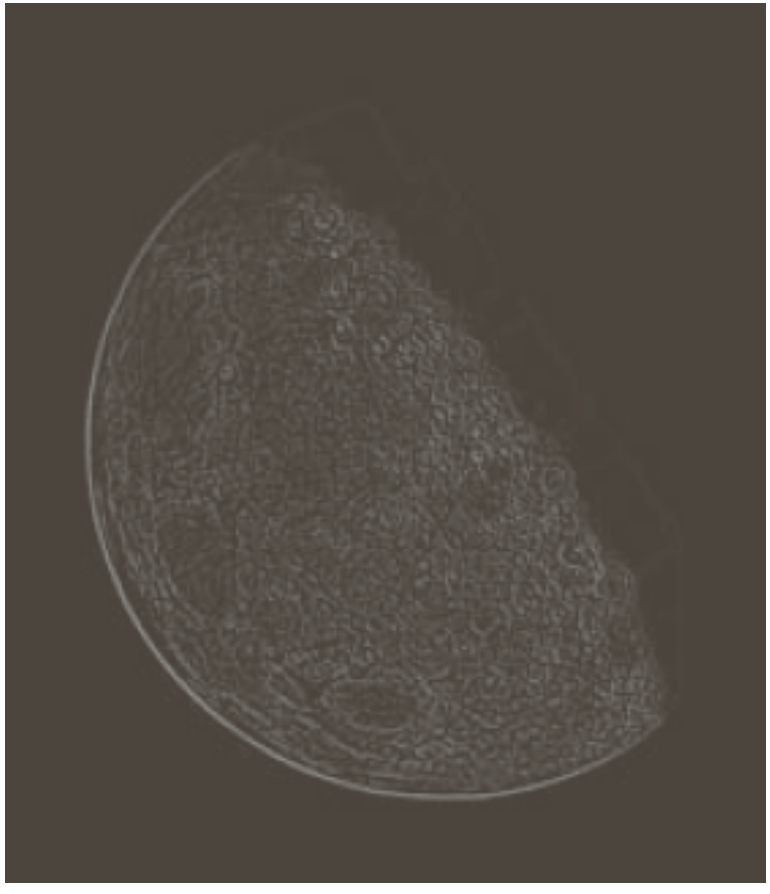
What if we want to include also the background, obtaining a meaningul image?

$$I'(x, y) = I(x, y) + c\nabla^2 I(x, y)$$

where $c = \pm 1$ depending on the mask used.

Sharpening using the Laplacian

Sharpening using the Laplacian



Laplacian

Sharpening using the Laplacian

Result using first mask

Sharpening using the Laplacian

Result using second mask

Unsharp filter

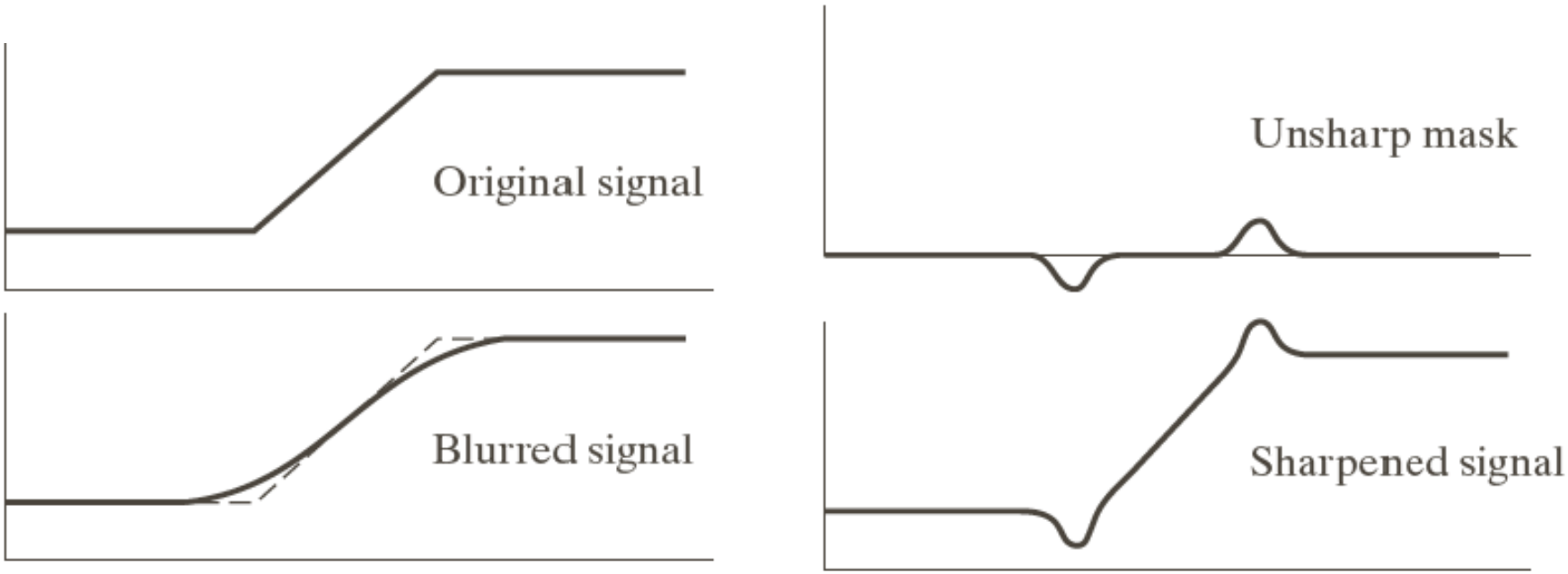
A sharpening operator which enhances discontinuities subtracting an unsharp version of the image.

Unsharp filter

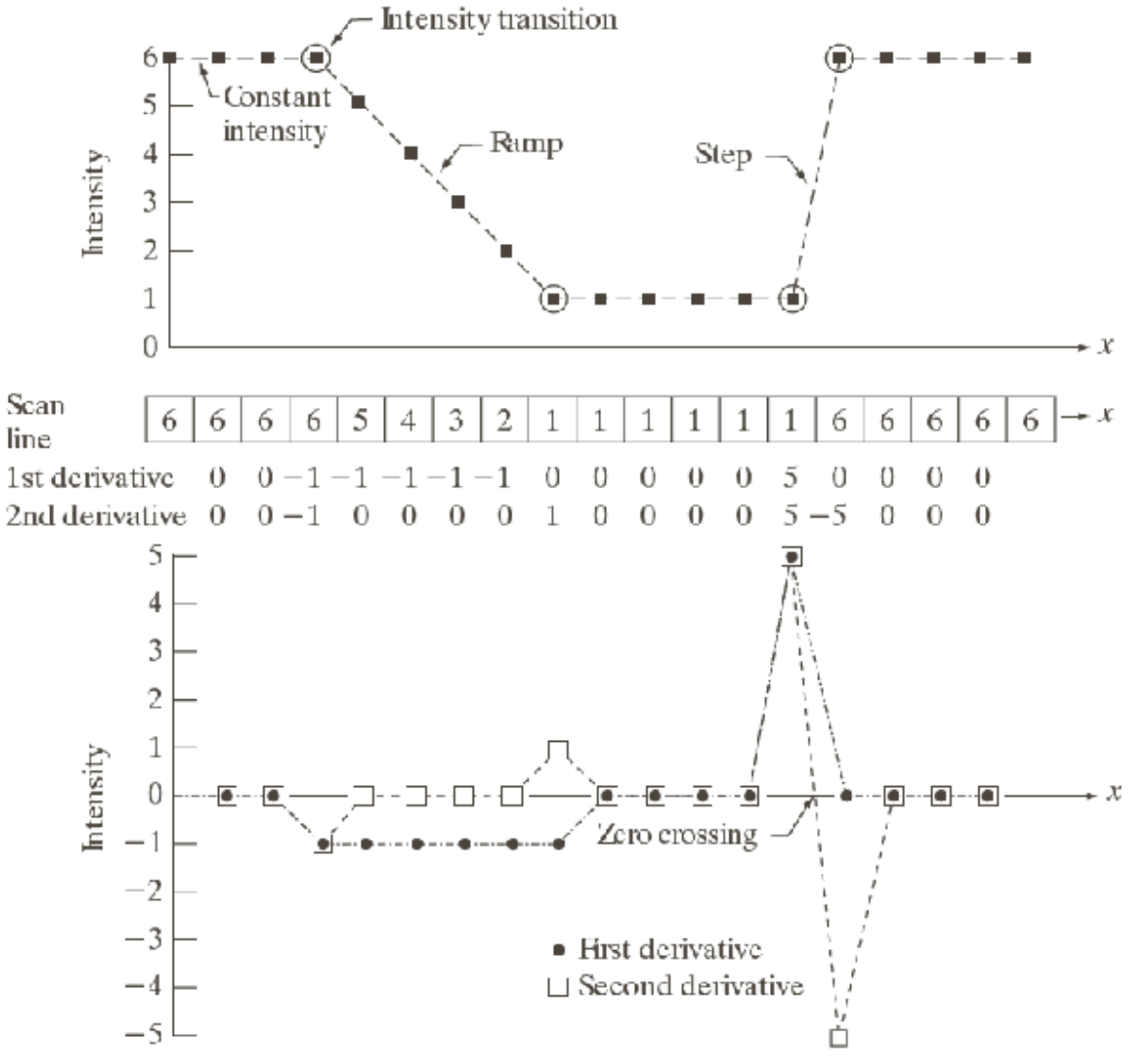
A sharpening operator which enhances discontinuities subtracting an unsharp version of the image. The process is as follows:

- Blur the original image
- Subtract the blurred image from the original
- Add the result to the original

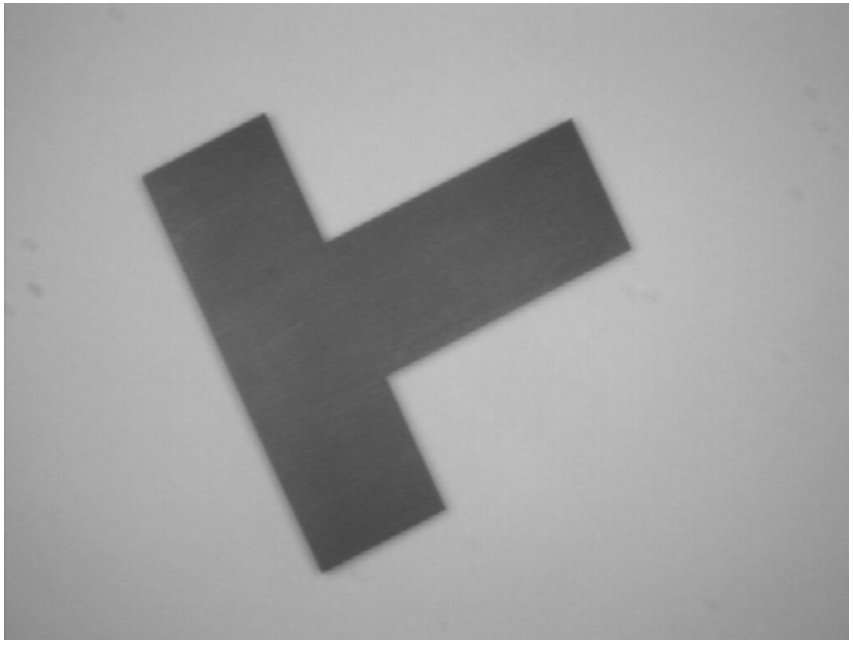
Unsharp filter

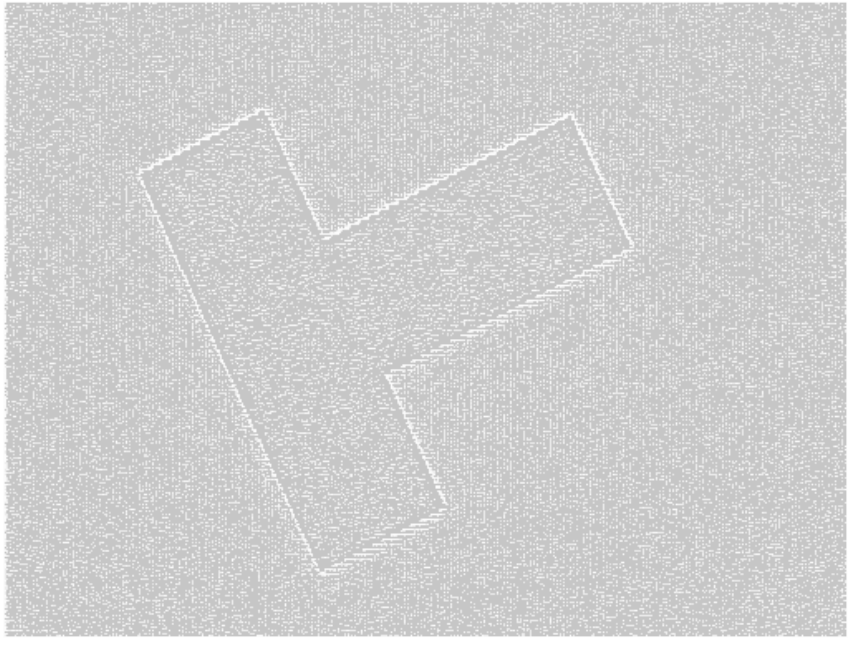


Unsharp filter



Unsharp filter





Unsharp filter