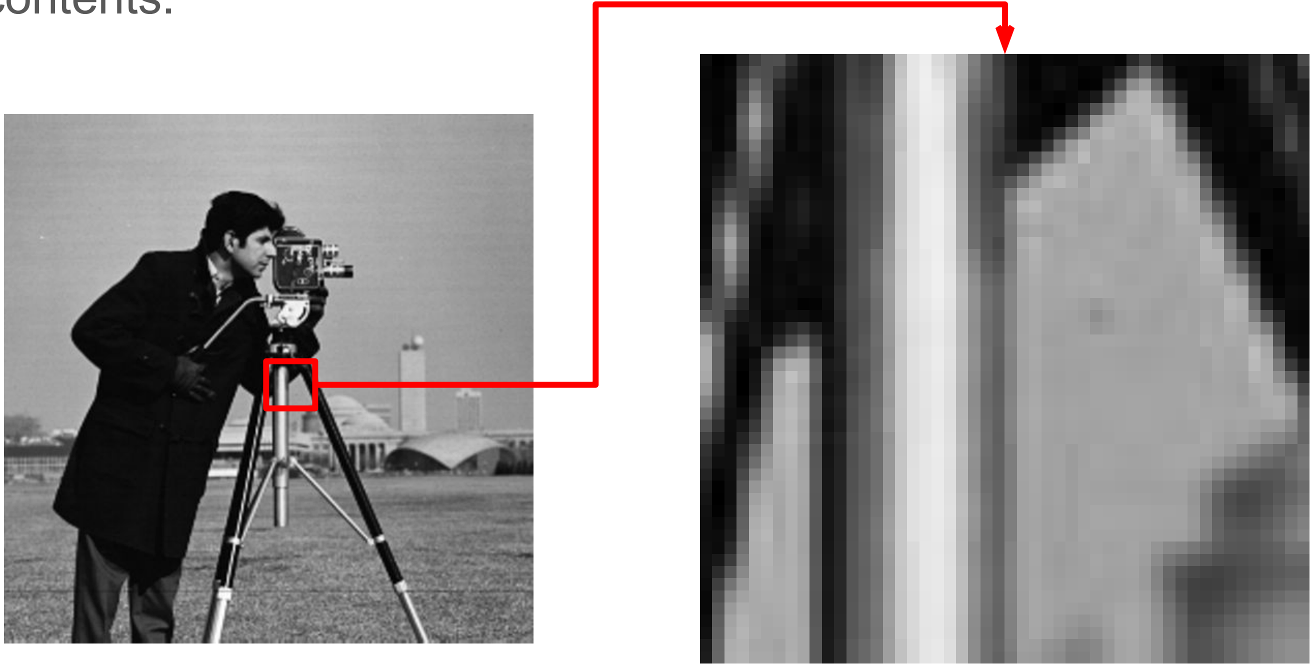
Filtering

Deepayan Bhowmik

Image = Information + Noise

Noise = random variations in pixel values not directly attributable to scene contents.



Noise

Image Encoding:

- Lossy image compression techniques remove information from image to safe space:
  - E.g. JPEG / MPEG: most widely used.
- Remove non-perceivable detail: “no noticeable difference”.
- Result: “compression artefacts” in the image.
- Lossy = compression artefacts = noise.

PNG (+ others) offers lossless compression.

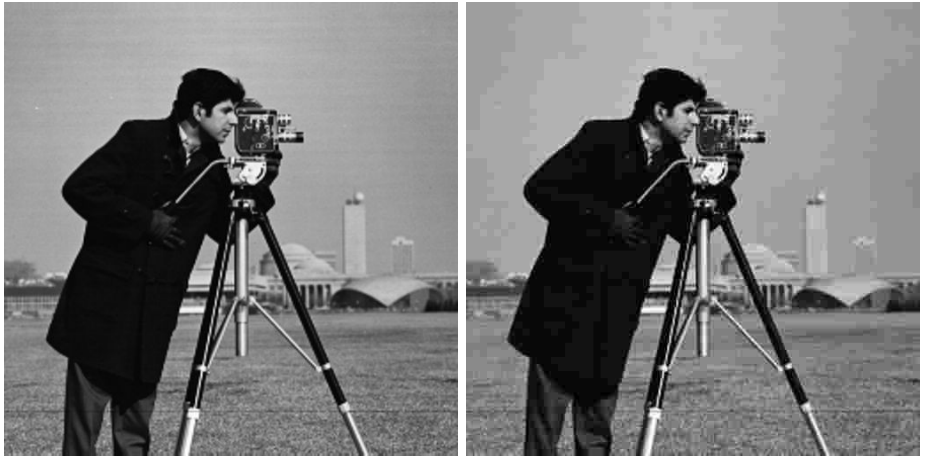


Image = Information + Noise

Lighting (mainly in image understanding tasks):

- Lighting in an environment is never constant:
    - Sunlight changes: time of day, weather and season.
    - Varying artificial light sources: street lamps, headlights etc..
    - Interior light sources oscillate with power supply frequency.
    - Objects cast shadows, as they move shadows change. Shadows cause false image features.

Image = Information + Noise

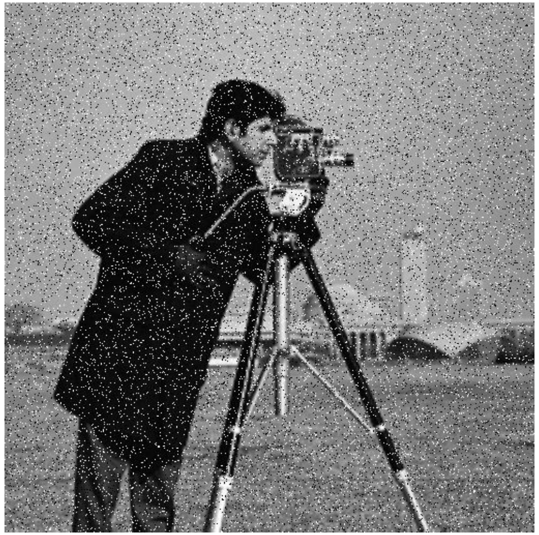
Occlusion (mainly in image understanding tasks):

- Objects are frequently obscured by other objects: occlusion:
  - big problem for recognition systems.
- “You cannot see what you cannot see”.
- “You don't know what you cannot see”.

Noise: Types

Salt & Pepper Noise:

- Random white or black (high/low) value pixels into the image.
- Less common in modern image sensors but present in cheap sensors.
    - Can effect single colour channels.
- Also known as impulse noise.
- “Dead pixel” noise - when a pixel in the sensor (or display) fails to respond to input correctly.
- Follows a binary high-low bi-modal noise distribution.

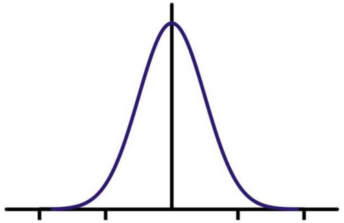


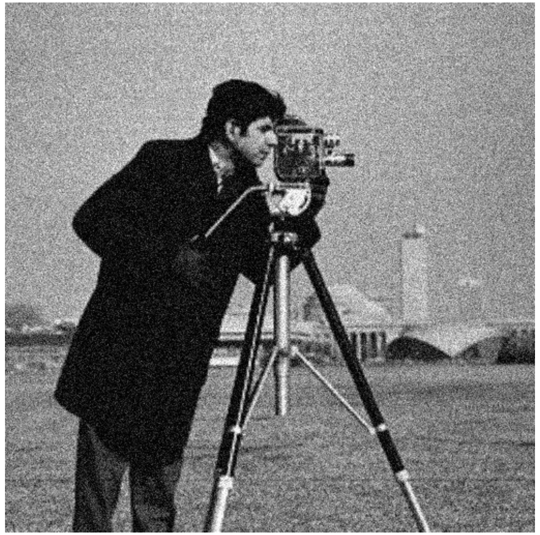
Noise: Types

Gaussian Noise:

- Small random variation of the image signal around its true value following the Gaussian or normal distribution.
- Most common used noise model in image processing.
- Effectively describes most random noise encountered in the image processing pipeline:
    - especially sensor noise.
- Also known as additive noise.

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





Sampling & Noise

No digital image is a perfect representation of the original 2D signal:

- It is limited in resolution by sampling and contains noise.

Noise removal = a major goals of image processing:

- Why ?: limit effects on image visualisation and analysis.

If images where noise free, then life would be easy:

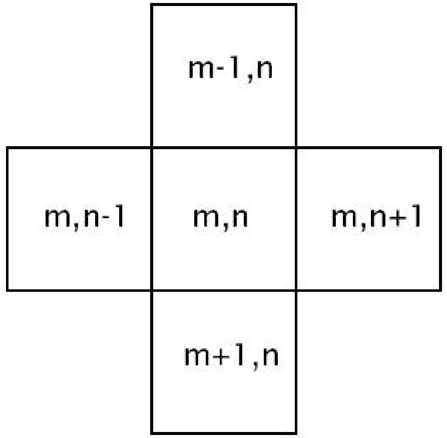
- There not, so its not (and hence big companies employ lots of software engineers in the global software industry!).

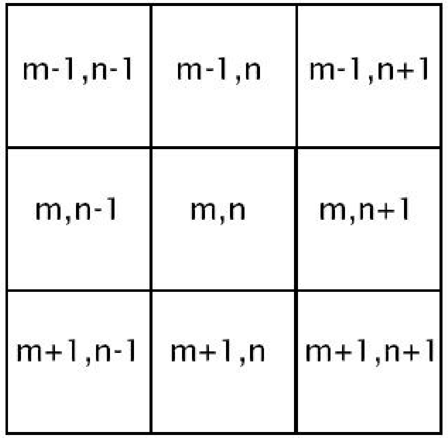
Q: How can we remove image noise?
A: “filter it out” using a range of image filtering techniques.

Filtering: Pixel Neighbourhoods

Image Filtering and many other operations use *N x N* image neighbourhoods:

- Local image neighbourhoods define connectivity or a local area of influence, relevance or interest.



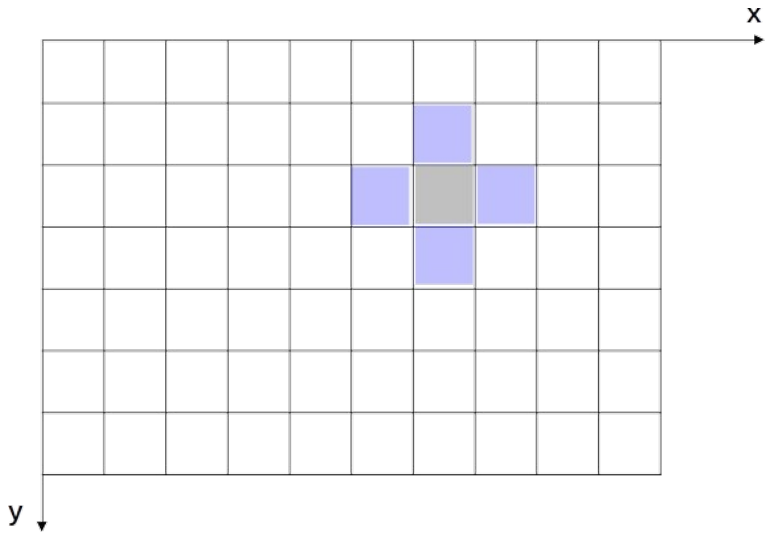


Connectivity choice: 4 or 8 connectivity (in 2D!).

Filtering: Pixel Neighbourhoods

A pixel p at coordinates *(x,y)* has four horizontal and vertical neighbours whose coordinates are given by:

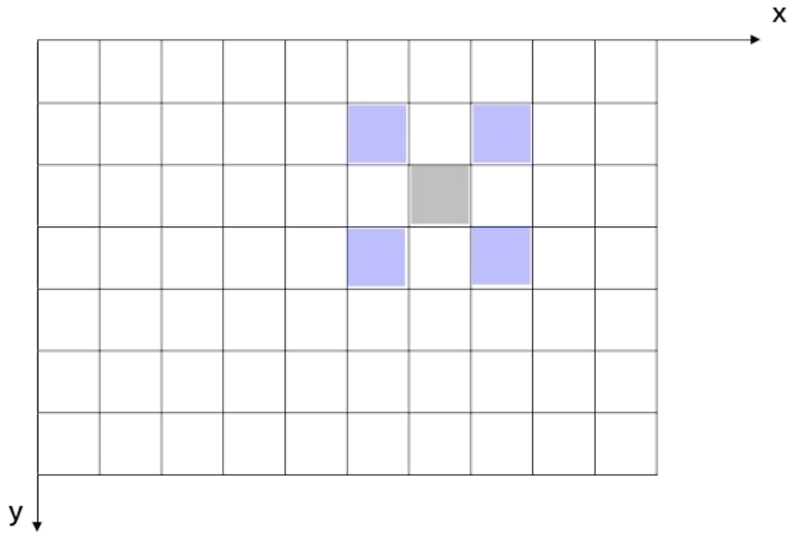
(x+1,y), (x-1,y), (x,y+1), (x,y-1)



Filtering: Pixel Neighbourhoods

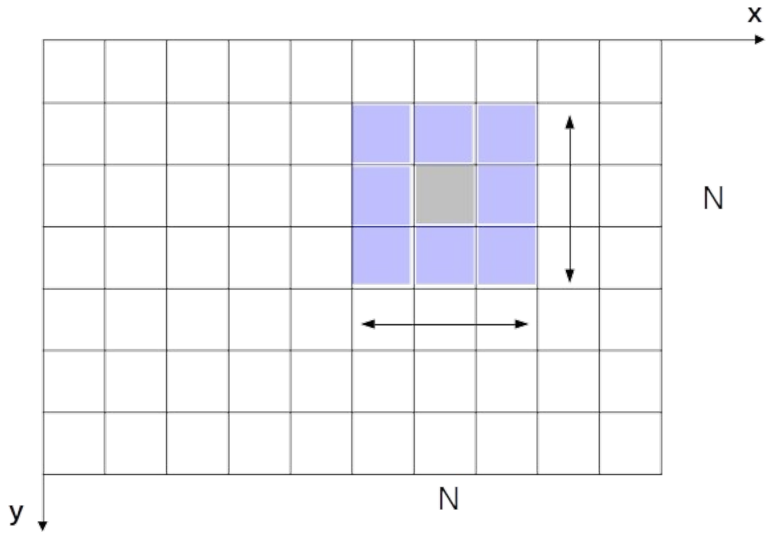
A pixel p at coordinates *(x,y)* has four diagonal neighbours whose coordinates are given by:

(x+1,y+1), (x+1,y-1), (x-1,y+1), (x-1,y-1)



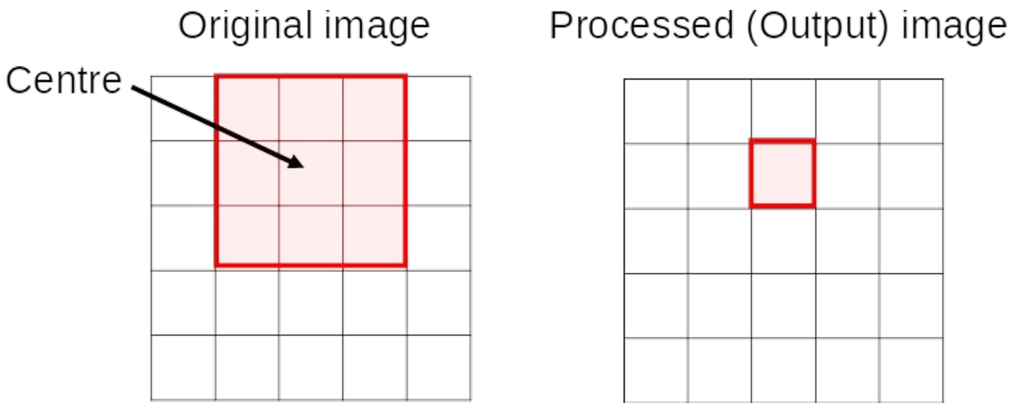
Filtering: Pixel Neighbourhoods

Together they form the local pixel neighbourhood (also called image neighbourhoods).
In general: defined as *N x N* from central origin.



Filtering: Non-Linear

As in the case of non-linear filtering, we consider a neighbourhood *N_{xy} (N x N)* centred at a certain pixel *(x,y)* of the image.
The new value of the image at that pixel *(x,y)* is a non-linear function of the intensity values in the corresponding neighbourhood, *N_{xy}*.



Filtering: Non-Linear

For an *N x N* image neighbourhood, N_{xy}, centred at pixel *(x,y)* indexed by *(s,t)* the following simple statistical filters can be defined to replace each pixel with the min/max/median of the input neighbourhood:

$$I_{ouput}(x,y) = \min_{(s,t)\in N_{xy}} \{I_{input}(s,t)\}$$

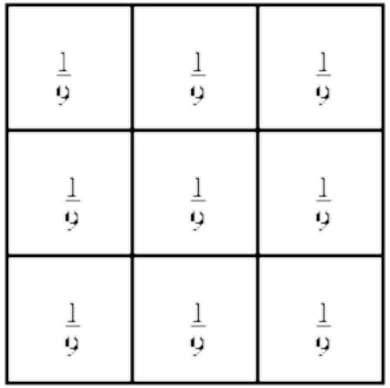
$$I_{output}(x,y) = \max_{(s,t)\in N_{xy}} \{I_{input}(s,t)\}$$

$$I_{output}(x,y) = \operatorname*{median}_{(s,t)\in N_{xy}} \{I_{input}(s,t)\}$$

Filtering: Mean

Operation: replace a given pixel with the mean of its *NxN* image neighbourhood.

For *N*=3, filter mask =



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

Effect: eliminates sudden intensity jumps which could be caused by some noise processes

- i.e. large deviations from the norm.

Filtering: Mean

S = window enclosing *N* × *M* neighbours centred at *(i, j)*.
NM is thus the total number of pixels (enclosed by S).

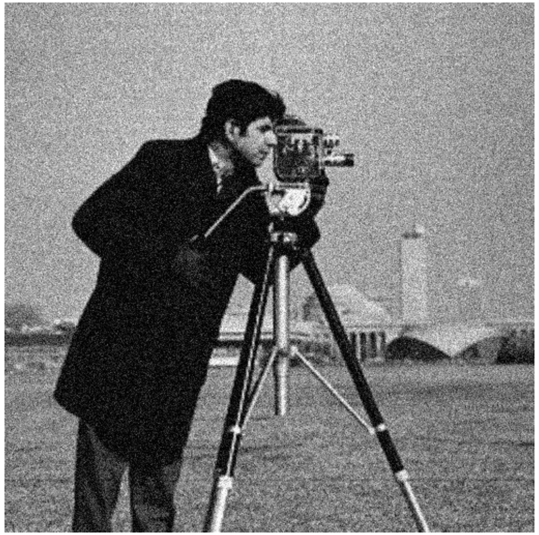
In general:

- *N=M* for most applications.
- *N* is odd:
  - Makes computation easier.

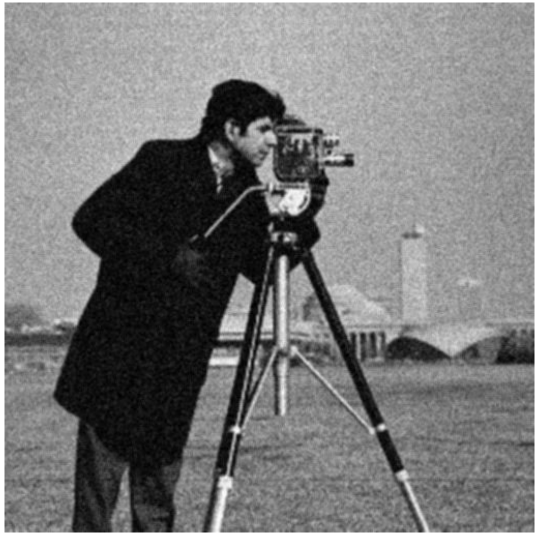
$$I_{output}(i,j) = \frac{1}{NM} \sum_{(N,M)\in S} I_{input}(i,j)$$

Filtering: Mean

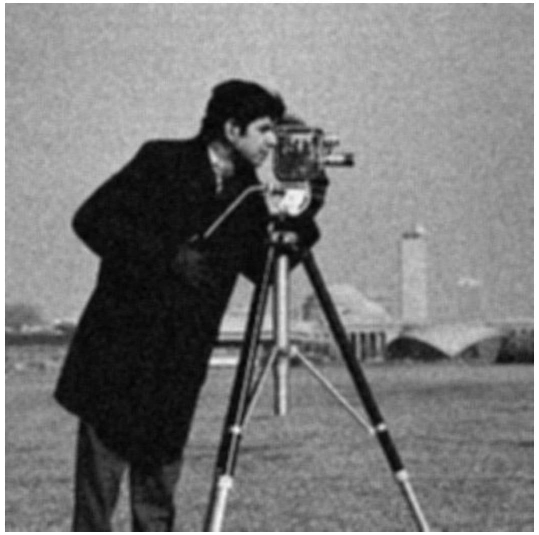
Gaussian Noise



3x3

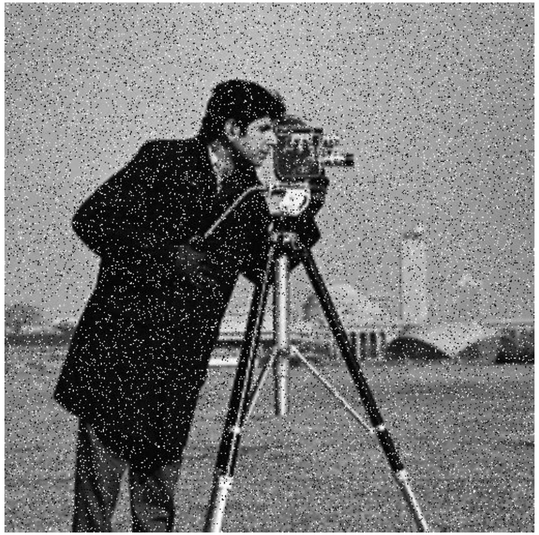


5x5

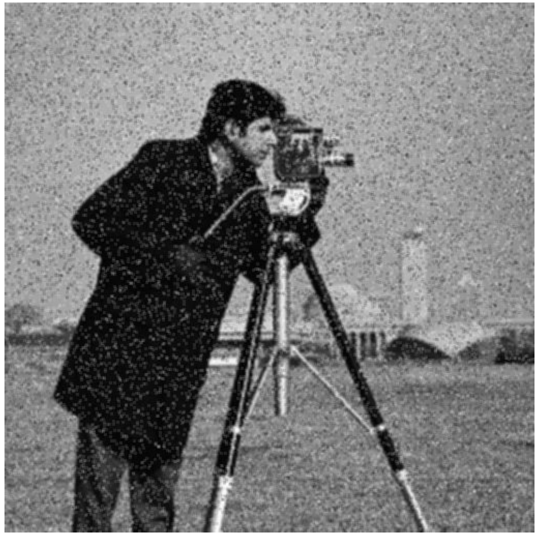


Filtering: Mean

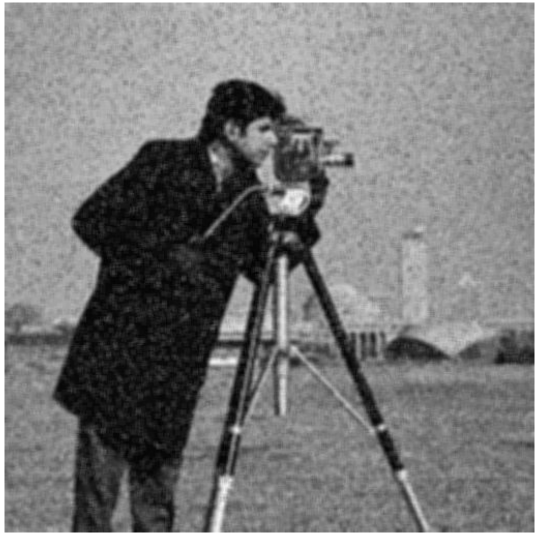
Salt & Pepper Noise



3x3



5x5



Filtering: Mean

Drawbacks:

Mean filter not robust to large noise deviations (i.e. statistical outliers):

- E.g. single pixel with a very unrepresentative value.

Mean filter causes edge blurring:

- Removes the high frequency sharp detail.

Filtering: Mean

- Alpha trimmed mean:

$$I_{output}(x,y)=\frac{1}{MN-2d}\sum_{(s,t)\in N_{xy}} I_{input}(s,t)$$

- Harmonic mean:

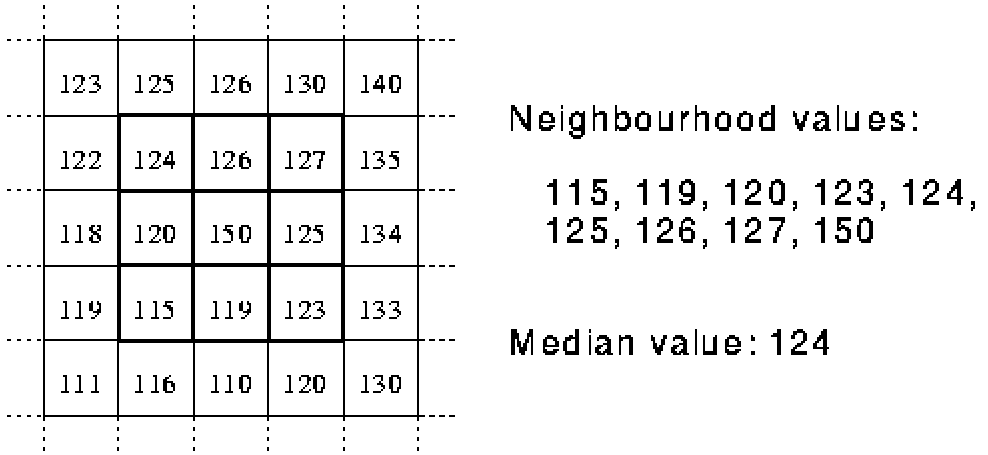
$$I_{output}(x,y)=\frac{MN}{\sum_{(s,t)\in N_{xy}} \frac{1}{I_{input}(s,t)}}$$

The dimension of the neighbourhood N_{xy} is *M x N*. The *d* lowest and the *d* highest intensity levels of *g(s,t)* in N_{xy} are deleted with $I_{input}(s,t)$ denoting the remaining (reduced set of) *MN*-d pixels in N_{xy}.

Filtering: Median

Operation: replace a given pixel with the median of its *NxN* image neighbourhood.

For *N* = 3:



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

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

Median value: 124

Effect: eliminates sudden intensity jumps which could be caused by some noise processes:

- I.e. large deviations from the norm.
- But robust to statistical outliers.

Filtering: Median

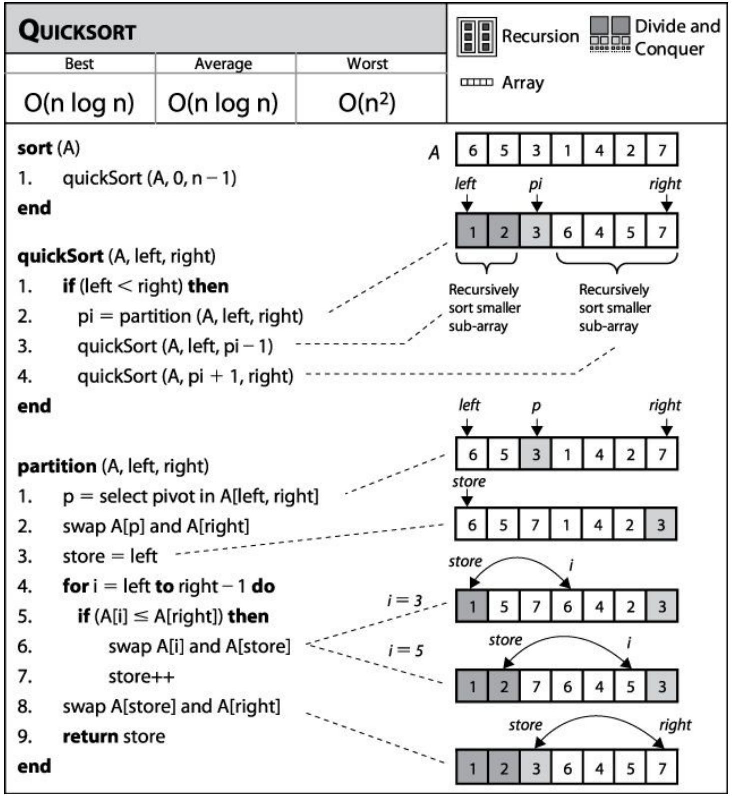
Median requires sort operation.

Common method: Bubble Sort (i.e. simple sorting):

- *$O(n^2)$* for *n* items.
- *$O(N^4)$* for an *NxN* neighborhood.

Improvement: Quicksort:

- *$O(n\log_2 n)$* for n items.
- *$O(N^2 \log_2 N^2)$* for an NxN image neighborhood.



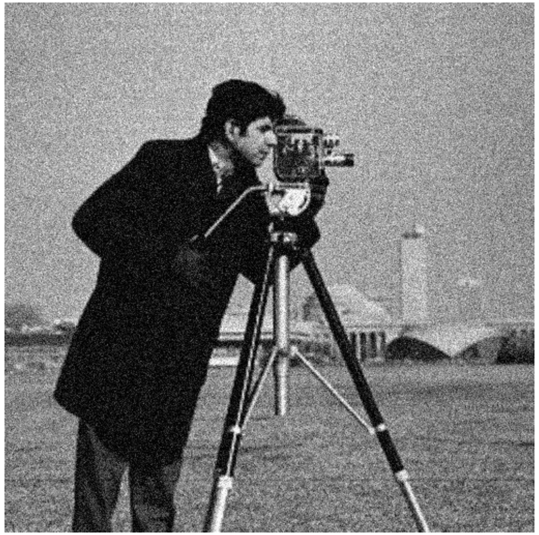
Quicksort

Best	Average	Worst
O(n log n)	O(n log n)	$O(n^2)$

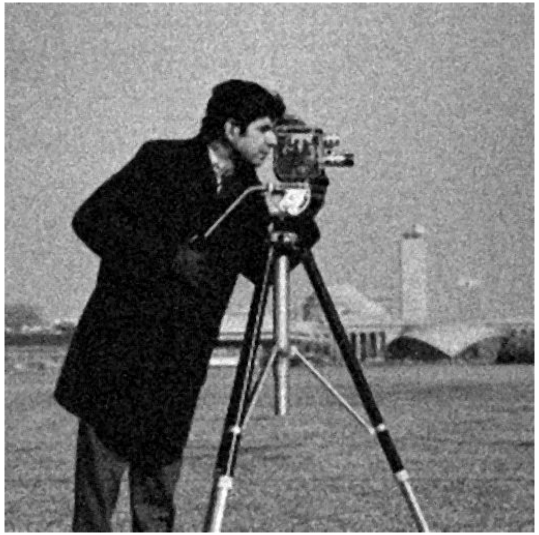
```
sort (A)
1.  quickSort (A, 0, n – 1)
end

quickSort (A, left, right)
1.  if (left < right) then
2.    pi = partition (A, left, right)
3.    quickSort (A, left, pi – 1)
4.    quickSort (A, pi + 1, right)
end

partition (A, left, right)
1.  p = select pivot in A[left, right]
2.  swap A[p] and A[right]
3.  store = left
4.  for i = left to right – 1 do
5.    if (A[i] ≤ A[right]) then
6.      swap A[i] and A[store]
7.      store++
8.  swap A[store] and A[right]
9.  return store
end
```

Filtering: Median

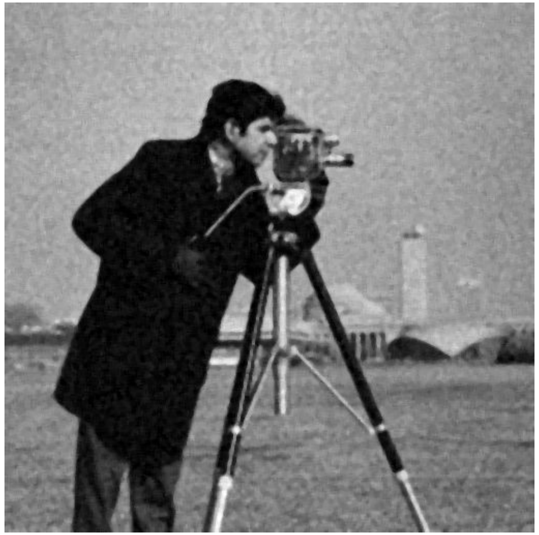
Gaussian Noise



3x3

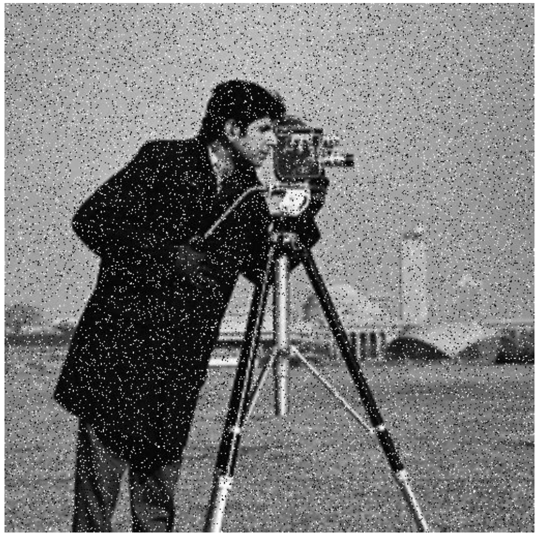


5x5

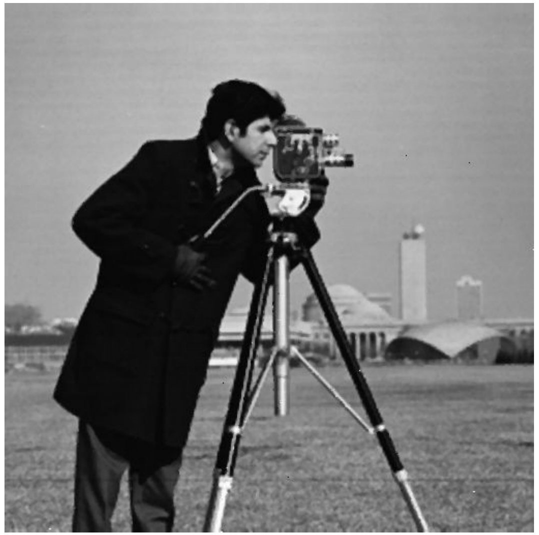


Filtering: Median

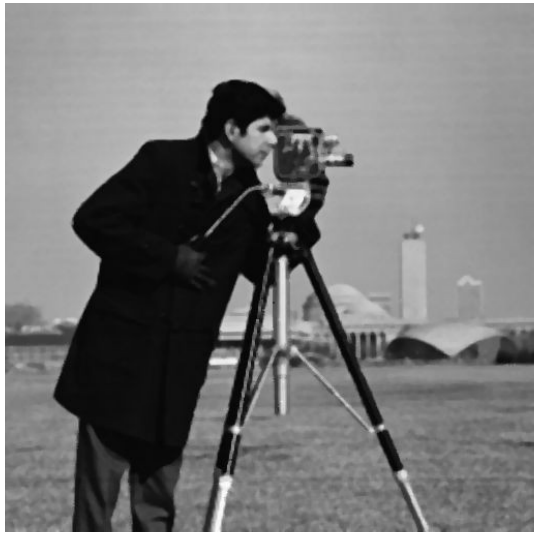
Salt & Pepper Noise



3x3



5x5

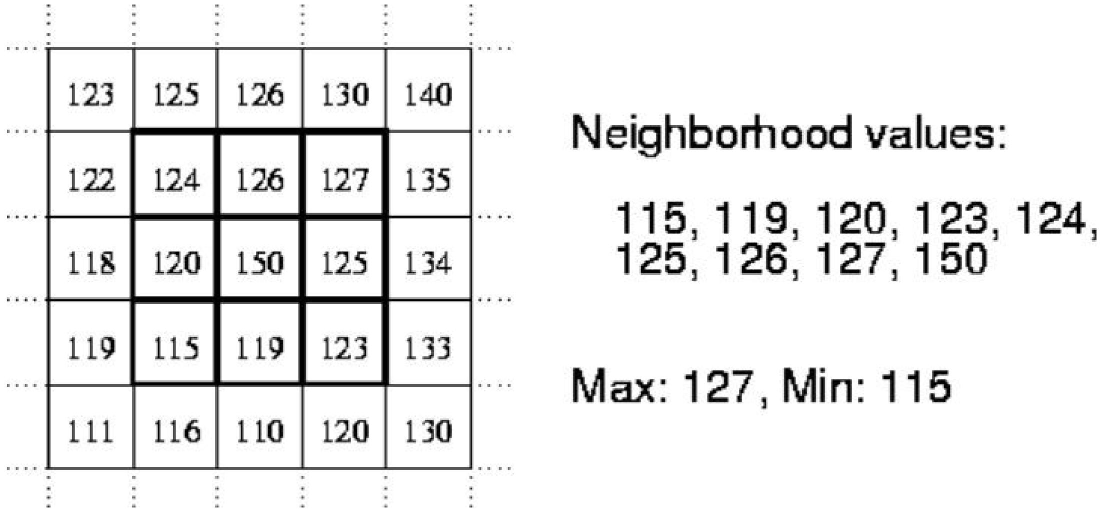


Filtering: Conservative Smoothing

Operation: compare pixel value to min and max of *N x N* neighbourhood

- Replace by min if < min.
- Replace by max if > max.

For *N* = 3:



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighborhood values:

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

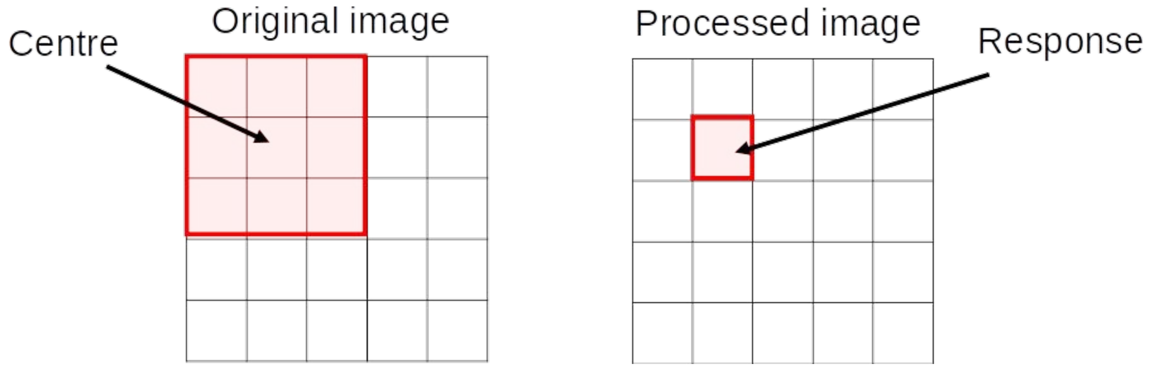
Max: 127, Min: 115

Effect: eliminates sudden intensity jumps.

Filtering: Convolution

Convolution of an image by a 3x3 mask:

$$I_{output}(i,j)=\sum_{k=1}^{3}\sum_{l=1}^{3} I_{input}(i-2+k, j-2+l)\, m_{kl}$$



In the implementation of convolution, the matrix should be updated only after all responses have been computed.

Filtering

Processing the matrix of an image with mask convolutions is called **spatial linear filtering**.

Why is it called linear?

- The mask operates on the image linearly, i.e. component-wise multiplication and summation.

The mask is also called **filter**, **kernel**, **template**, **window**.

Filtering

Example: Find the convolution of *Image*$_A$ by *M*.

$$Image_A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	1/9	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	1/9	0	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	1/9	0	..
..	2/9	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	1/9	0	..
..	2/9	4/9	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Original image

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Processed image

..	..	..	..	..	..	..
..	1/9	2/9	2/9	1/9	0	..
..	2/9	4/9	4/9	2/9	0	..
..	2/9	4/9	4/9	2/9	0	..
..	1/9	2/9	2/9	1/9	0	..
..	..	..	..	..	..	..
..	..	..	..	..	..	..

Filtering

In this ample, each pixel of the original image is replaced by the average of its neighbours.

The convolution mask:

$$M = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

It is just the equivalent to the **mean** (or average) **filter** and when applied to an image it smoothes it.

Filtering: Discrete Convolution in 2D

Discrete convolution as a 2D filter can thus be represented as follows for a generalised N x M convolution mask:

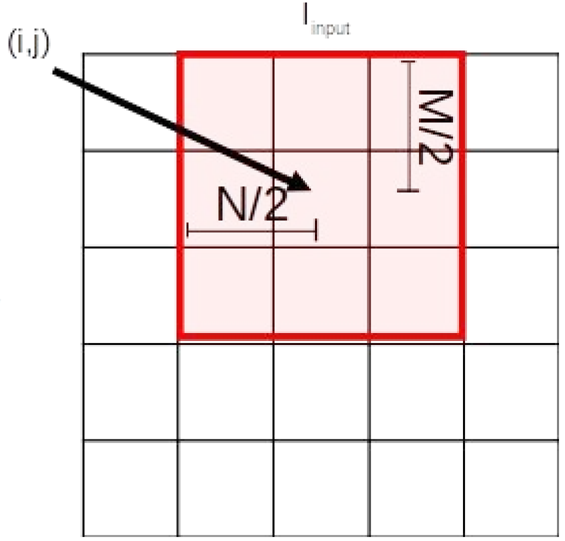
$$I_{output}(i,j)=\sum_{k=1}^{N}\sum_{l=1}^{M} I_{input}(i-(N-1)+k,j-(M-1)+l)m_{kl}$$

In terms of image processing this is essentially the localised weighted sum of the image (*I*) and the convolution kernel (*m*) over a *NxM* pixel neighbourhood, at a given location within the image *(i,j)*.

Filtering: Discrete Convolution in 2D

Discrete convolution as a 2D filter can thus be represented as follows for a generalised N x M convolution mask:

$$I_{output}(i,j)=\sum_{k=-\lfloor N/2\rfloor}^{\lfloor N/2\rfloor}\sum_{l=-\lfloor M/2\rfloor}^{\lfloor M/2\rfloor} I_{input}(i+k,j+l)\,m_{kl}$$

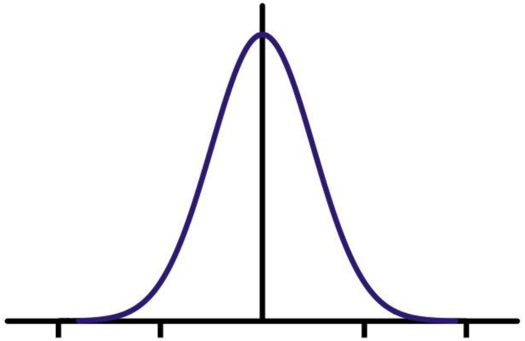


Often it is easier (and simpler) to think of the central pixel (i,j) with +/- indices up ↔ down and left ↔ right as per above.

Filtering: Gaussian Filter

The elements of the mask are given by the Gaussian function *g()* of the distance, *d* = |*p* - *p'*|, between the centres of central pixel *p* and pixel *p'*.

The graph of a Gaussian function *g()* has the characteristic bell shape.



Its general form is given by:

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

where σ is a parameter controlling the width of the Gaussian “bell-curve”.

Filtering: Gaussian Filter

Example of a 3x3 Gaussian filter:

$$M_3 = \begin{bmatrix} g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) \\ g_\sigma(1) & g_\sigma(0) & g_\sigma(1) \\ g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) \end{bmatrix} = \begin{bmatrix} 0.011 & 0.083 & 0.011 \\ 0.083 & 0.619 & 0.083 \\ 0.011 & 0.083 & 0.011 \end{bmatrix}$$

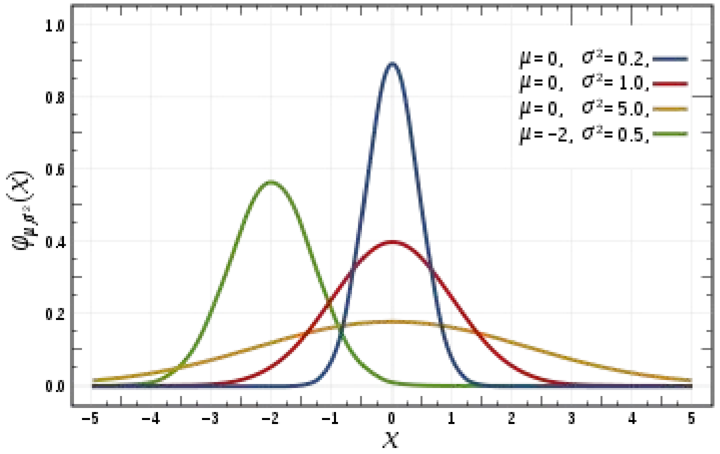
Filtering: Gaussian Filter

Example of a 5x5 Gaussian filter:

$$M_5 = \begin{bmatrix} g_\sigma(2\sqrt{2}) & g_\sigma(\sqrt{5}) & g_\sigma(2) & g_\sigma(\sqrt{5}) & g_\sigma(2\sqrt{2}) \\ g_\sigma(\sqrt{5}) & g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) & g_\sigma(\sqrt{5}) \\ g_\sigma(2) & g_\sigma(1) & g_\sigma(0) & g_\sigma(1) & g_\sigma(2) \\ g_\sigma(\sqrt{5}) & g_\sigma(\sqrt{2}) & g_\sigma(1) & g_\sigma(\sqrt{2}) & g_\sigma(\sqrt{5}) \\ g_\sigma(2\sqrt{2}) & g_\sigma(\sqrt{5}) & g_\sigma(2) & g_\sigma(\sqrt{5}) & g_\sigma(2\sqrt{2}) \end{bmatrix}$$

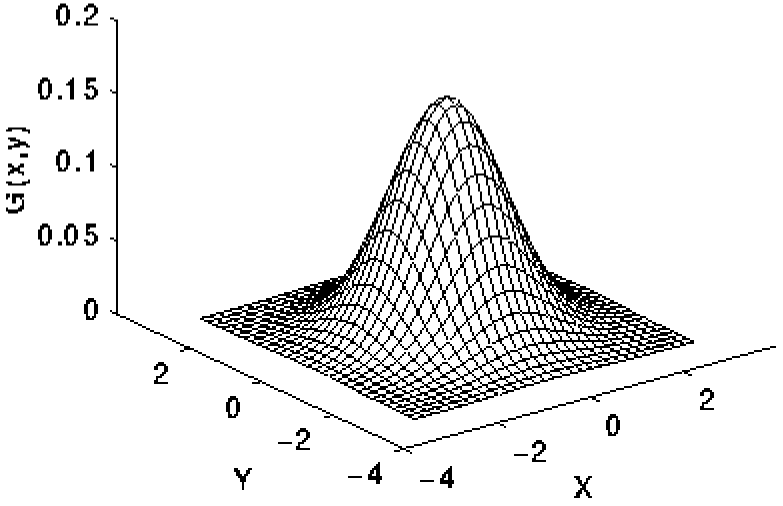
$$= \begin{bmatrix} 0.0000 & 0.0000 & 0.0002 & 0.0000 & 0.0000 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0002 & 0.0837 & 0.6187 & 0.0837 & 0.0002 \\ 0.0000 & 0.0113 & 0.0837 & 0.0113 & 0.0000 \\ 0.0000 & 0.0000 & 0.0002 & 0.0000 & 0.0000 \end{bmatrix}$$

Filtering: Gaussian Filter



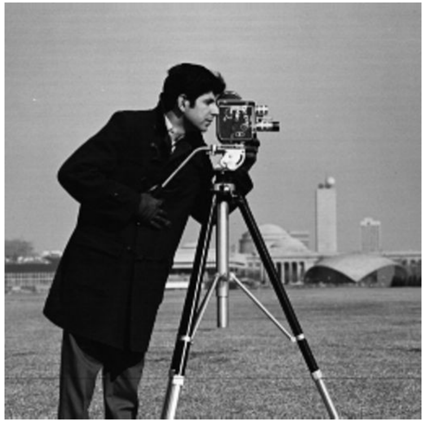
$\frac{1}{273}$

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

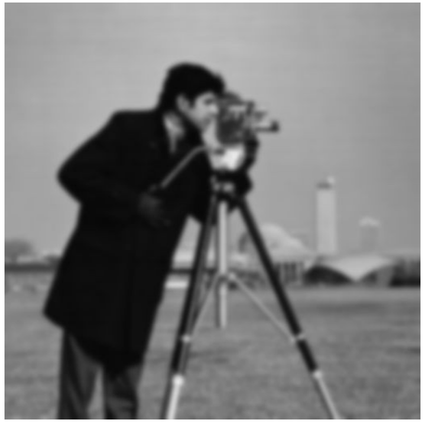


Filtering: Gaussian Filter

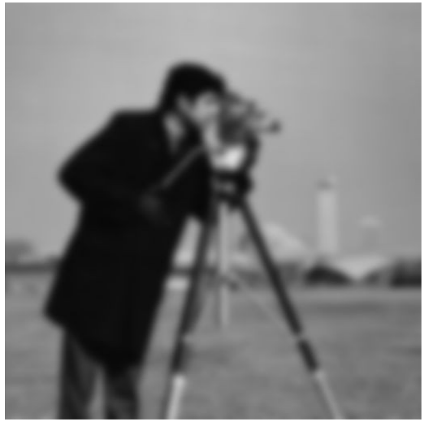
Input



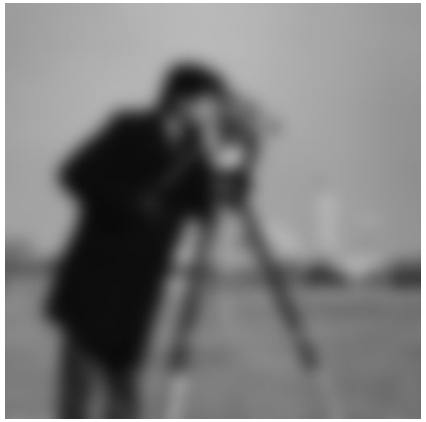
σ=3



=5



=10



Filtering: Gaussian Filter

We can describe the effect of the Gaussian filters by the equation:

$$I_p^{output} = \frac{\sum_{p' \in \Omega_{NM}} g(|p - p'|) I_{p'}}{\sum_{p' \in \Omega_{NM}} g(|p - p'|)}$$

The new intensity I_p^{output} of the central pixel *p* is a linear combination of the intensities of the pixels *p′* in a neighborhood *Ω* which determines the size of the filter.
The denominator of the fraction normalises the expression, making the sum of the coefficients equal to 1.

Filtering: Laplacian Filter

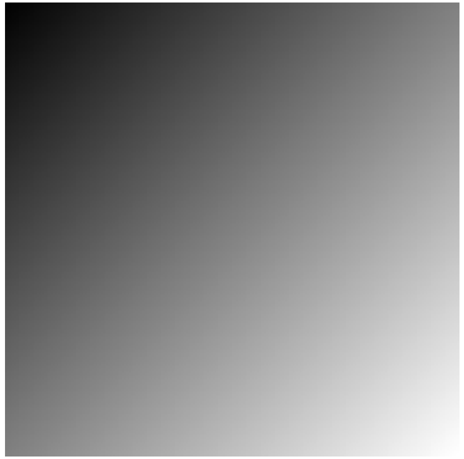
The Laplacian filter:

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

Notice that the Laplacian filter contains negative numbers.

Filtering: Laplacian Filter

The response to the Laplacian filter is zero at areas with smooth intensity changes. In particular, the response is zero at the areas of constant intensity.



The response to the Laplacian filter is everywhere zero at these images.

Filtering: Laplacian Filter

Operation: highlights areas of rapid intensity change: i.e. edges.

Laplacian = sum of partial derivatives of image pixel values.

$$I_{laplacian}(x,y) = \frac{\partial^2 I_{input}(x,y)}{\partial x^2} + \frac{\partial^2 I_{input}(x,y)}{\partial y^2}$$

Commonly implemented via discrete convolution filters:

either of.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

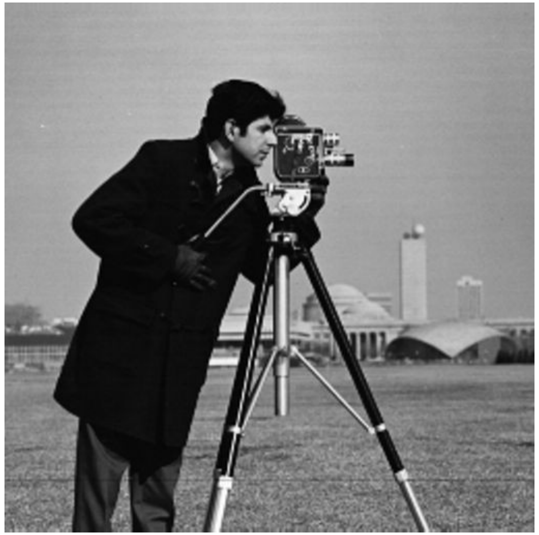
Filtering: Laplacian Filter

When the variation of the intensities is not smooth, the response to the Laplacian filter can be non-zero.

The less smooth the variation of the intensities, the higher in absolute value of the response to the Laplacian filter.

The Laplacian filter has higher response in absolute value at the **edges** of the image.

Filtering: Laplacian Filter



Filtering: Laplacian Filter

The Laplacian mask that takes into account the two diagonal directions too.

$$M = \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ 2/3 & -10/3 & 2/3 \\ 1/6 & 2/3 & 1/6 \end{bmatrix}$$

Filtering: Bilateral Filter

In Gaussian filters, the influence of one pixel over another is determined only by spatial proximity. The nearer they are, the larger the influence.

In bilateral filters, the influence of one pixel over the other is determined by two factors: spatial proximity and a measure of similarity between the two pixels.

Thus, strictly speaking, the bilateral filter is not a spatial filter (but has a spatial component).

Filtering: Bilateral Filter

Typically, the similarity of two pixels is measured by the difference of their intensities $(I_p - I_{p'})$.

Large differences of intensities mean small similarity between two pixels and thus small influence over one another in the filtering process.

We expect bilateral filtering to preserve the edges better than spatial Gaussian filtering because it is essentially “edge aware”.

Filtering: Bilateral Filter

Bilateral filters are described by the equation:

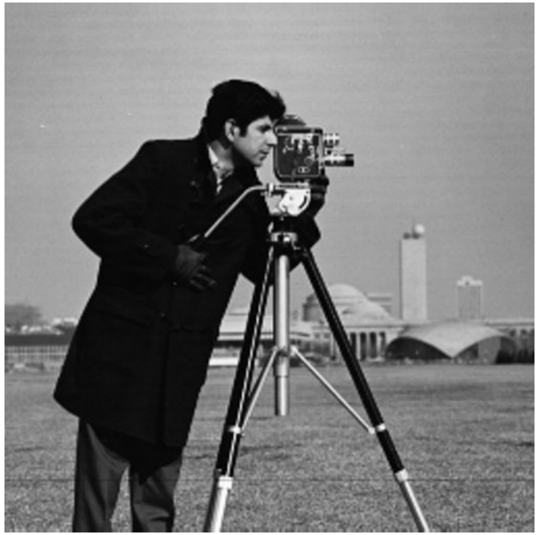
$$I_{output}(p) = \frac{\sum_{p' \in \Omega} g_1(|p - p'|) g_2(I_p - I_{p'}) I_{p'}}{\sum_{p' \in \Omega} g_1(|p - p'|) g_2(I_p - I_{p'})}$$

The two Gaussian functions g_1 and g_2 determine the influence of spatial proximity and similarity on the filtering process.

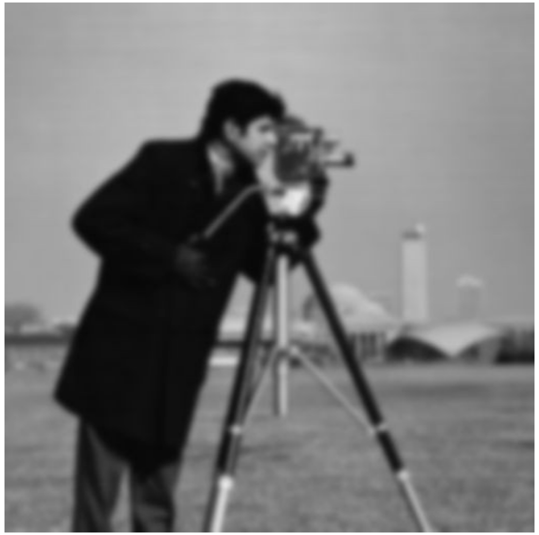
The neighbourhood Ω, over which we do the summation, determines the size of the filter.

Filtering: Bilateral Filter

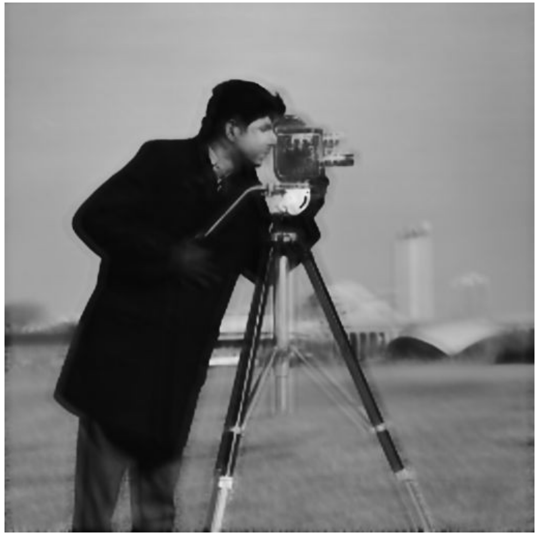
Input



Gaussian *σ=3*



Bilateral *σ=5*



Filtering: Non-Local Means

Key Concept:

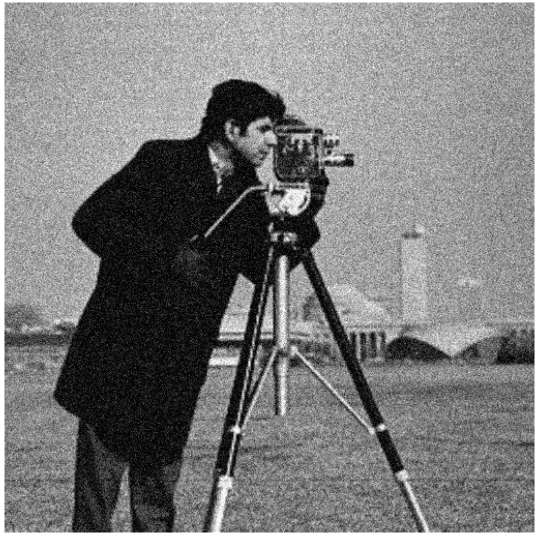
- Search image for similar pixel neighbourhoods to the current one being filtered.
- Replace the current pixel with a weighted mean of multiple similar neighbourhoods.
- Weights defined as similarity between pixel neighbourhoods.
- I.e. the average (mean) of neighbourhoods that are not local within the image to the target pixel neighbourhood – hence the name.

$$I_{output}(i,j) = \frac{\sum_{neighbourhood(k,l) \in I_{input}} W_{neighbourhood}(k,l)\, I_{input}(k,l)}{\sum W_{neighbourhood}(k,l)\, I_{input}}$$

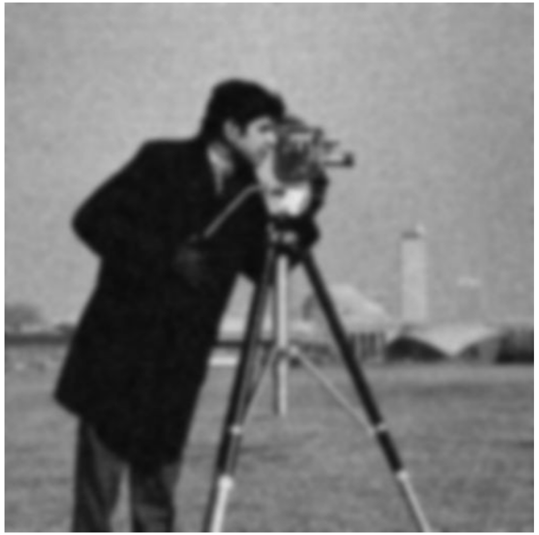
$$W_{neighbourhood(k,l)} = e^{\left(\frac{-((neighbourhood(k,l) - neighbourhood(i,j))^2)}{(h^2)}\right)}$$

Filtering: Non-Local Means

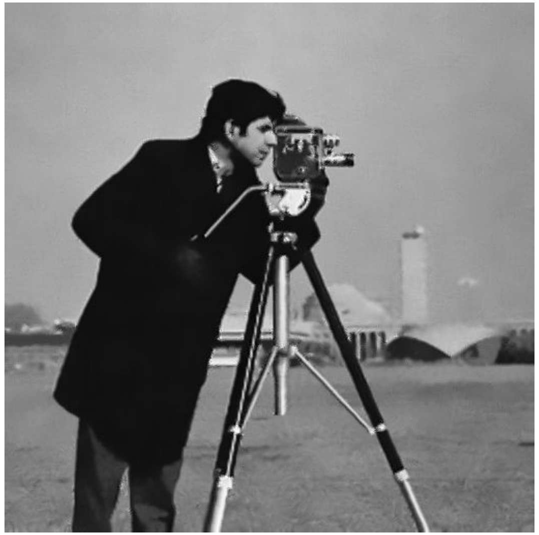
Gaussian Noise



Gaussian



Non-Local Means



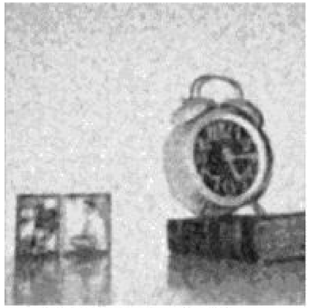
Applications

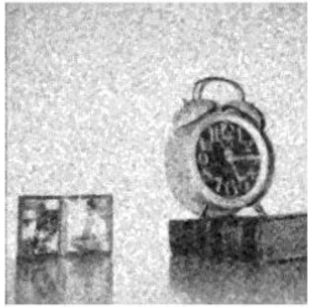
The proposed algorithm firstly defines the region direction gradient to detect the image edge pixel with dynamic threshold control, and then it implements a modified hybrid filter method to each edge pixel and each smoothing pixel, which aims at preserving the image edges and suppressing the noises in the operation of image filtering.

L. Zhao, J. Lu and J. Zhang, "A Novel Image Filtering Algorithm in Image Enhancement," 2009 5th International Conference on Wireless Communications, Networking and Mobile Computing, Beijing, 2009, 1-4.

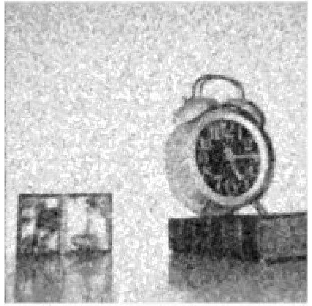
(a) AMWM in the Mixed Noise

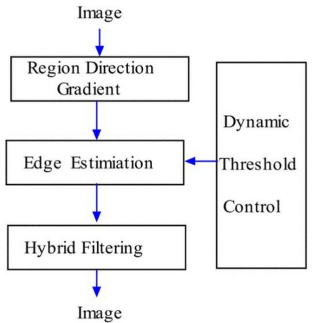
(a) AMWM in Impulse Noise

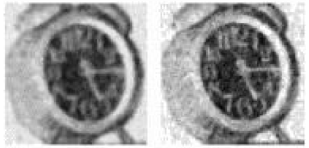

(a) AMWM in Gaussion Noise


(b) EdgeFilter in the Mixed Noise

(b) EdgeFilter in Impulse Noise


(b) EdgeFilter in Gaussion Noise




(c)Part Image (300% amplified)