
Filtering (II)

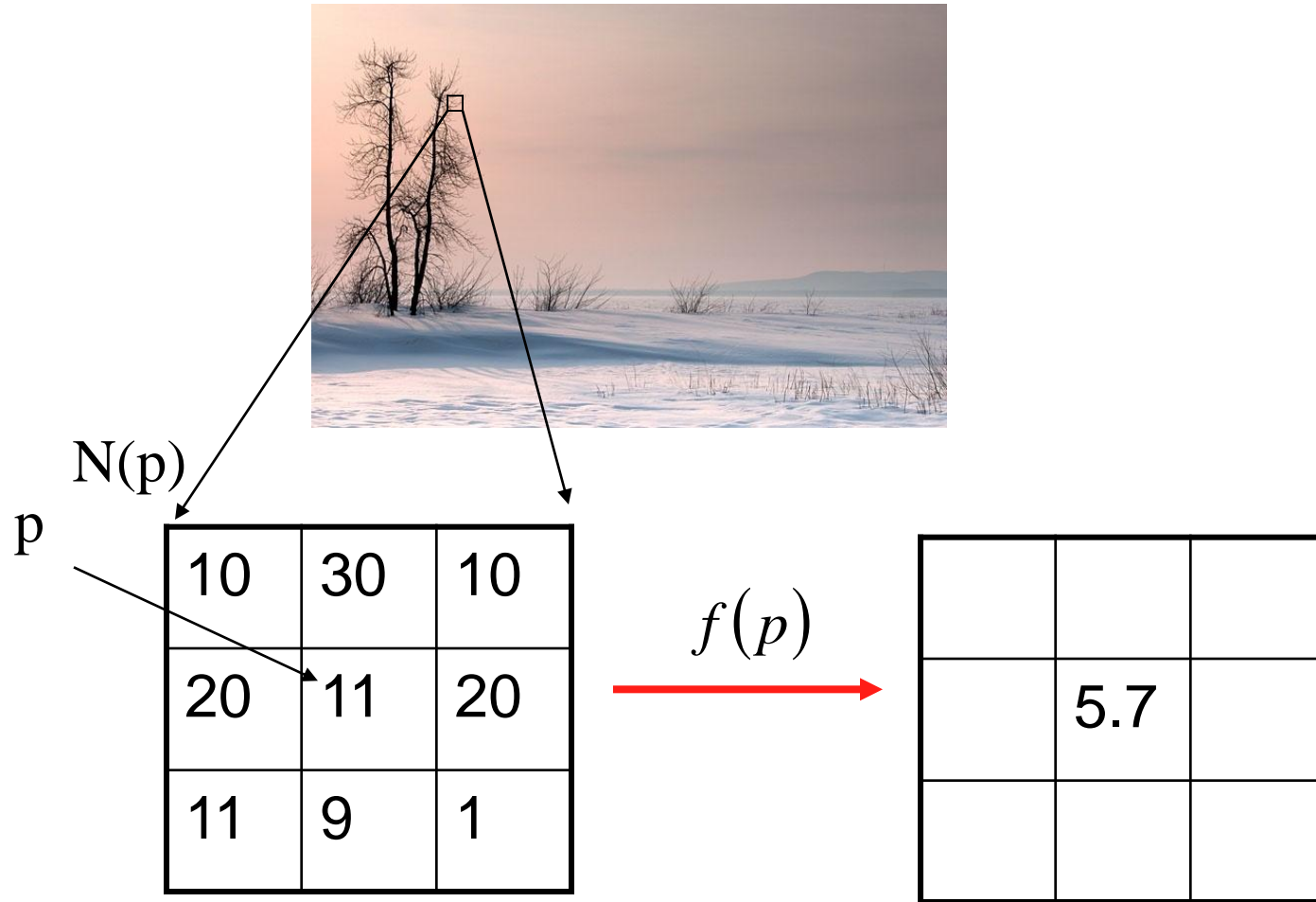
Dr. Gerhard Roth

COMP 4900D

Winter 2012

Image Filtering

Modifying the pixels in an image based on some functions of a local neighbourhood of the pixels



Linear Filtering – convolution

The output is the linear combination of the neighbourhood pixels

$$I_A(i, j) = I * A = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} A(h, k) I(i - h, j - k)$$

The coefficients come from a constant matrix A, called [kernel](#). This process, denoted by ‘*’, is called (discrete) [convolution](#).

1	3	0
2	10	2
4	1	1

*

1	0	-1
1	0.1	-1
1	0	-1

=

	5	

Image

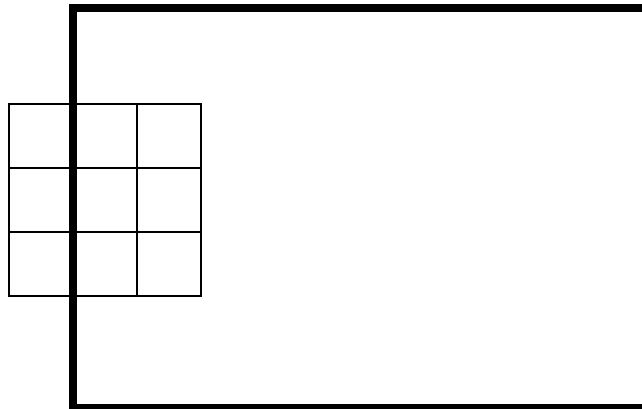
Kernel

Filter Output

Handle Border Pixels

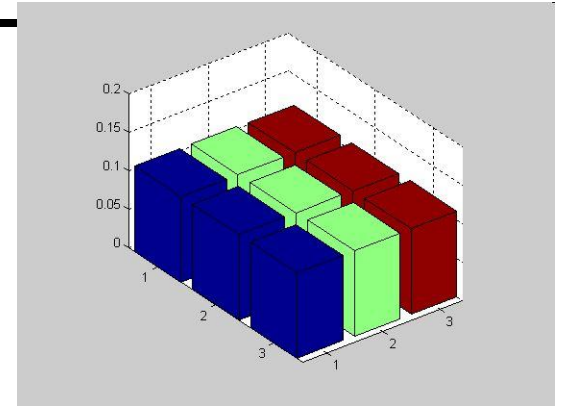
Near the borders of the image, some pixels do not have enough neighbours. Two possible solutions are:

- Set the value of all non-included pixels to zero.
- Set all non-included pixels to the value of the corresponding pixel in the input image.



Smoothing by Averaging

$$1 = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} A(h, k)$$



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

1	1	1
1	1	1
1	1	1

$=$



Convolution can be understood as weighted averaging.

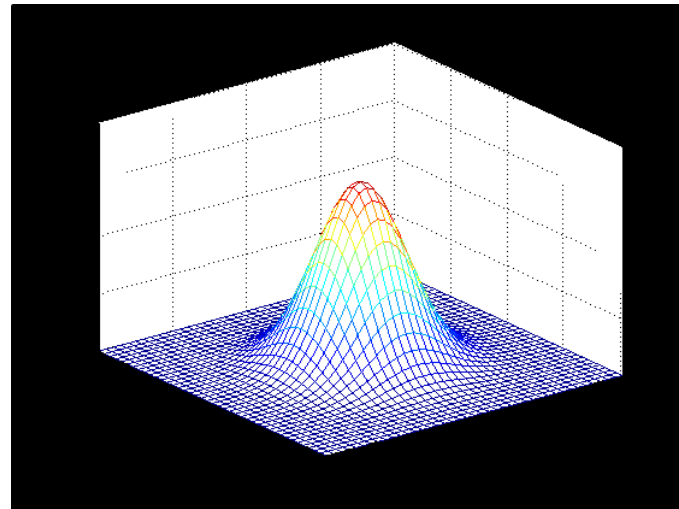
Gaussian Filter

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2 + y^2)}{2\sigma^2}\right)$$

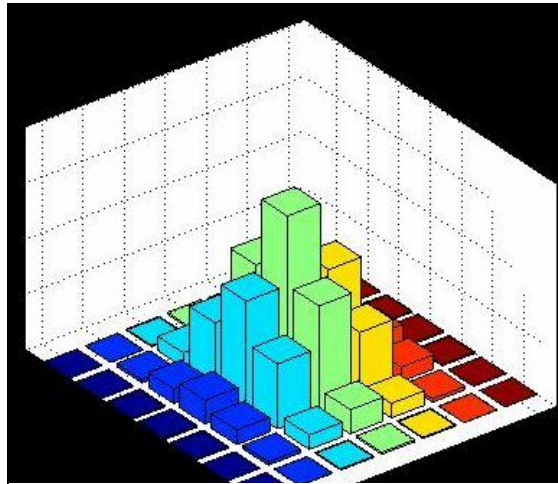
Discrete Gaussian kernel:

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

where $G(h, k)$ is an element of an $m \times m$ array



Gaussian Filter



$\ast \frac{1}{273}$

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

$=$



$$\sigma = 1$$

Gaussian Kernel is Separable

$$I_G = I * G =$$

$$= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} G(h, k) I(i-h, j-k) =$$

$$= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} e^{-\frac{h^2+k^2}{2\sigma^2}} I(i-h, j-k) =$$

$$= \sum_{h=-m/2}^{m/2} e^{-\frac{h^2}{2\sigma^2}} \sum_{k=-m/2}^{m/2} e^{-\frac{k^2}{2\sigma^2}} I(i-h, j-k)$$

since

$$e^{-\frac{h^2+k^2}{2\sigma^2}} = e^{-\frac{h^2}{2\sigma^2}} e^{-\frac{k^2}{2\sigma^2}}$$

Gaussian Kernel is Separable

Convolving rows and then columns with a 1-D Gaussian kernel.

$$\begin{array}{ccccc} \boxed{\mathbf{I}} & * & \frac{1}{38} \begin{array}{|c|c|c|c|c|} \hline 1 & 9 & 18 & 9 & 1 \\ \hline \end{array} & = & \boxed{\mathbf{I}r} \\ & & & & \\ \boxed{\mathbf{I}r} & * & \frac{1}{38} \begin{array}{|c|} \hline 1 \\ \hline 9 \\ \hline 18 \\ \hline 9 \\ \hline 1 \\ \hline \end{array} & = & \boxed{\text{result}} \end{array}$$

The complexity increases linearly with m instead of with m^2 .

Gaussian vs. Average

- Gaussian and average are smoothing linear filters
- In this case sum of all kernel entries is one
- So that new pixel is in same value range as old



Gaussian Smoothing

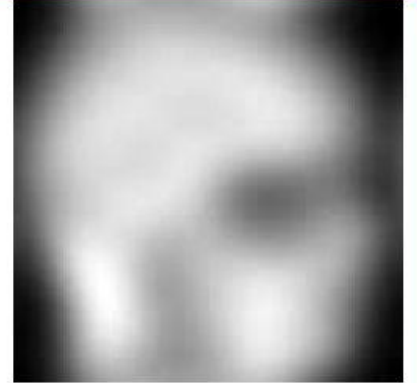
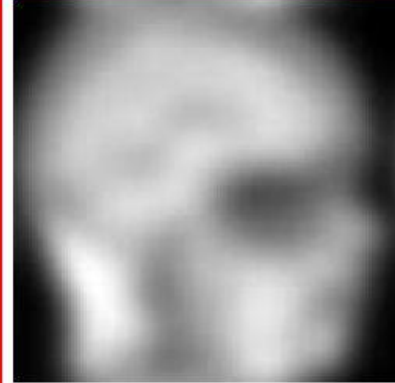
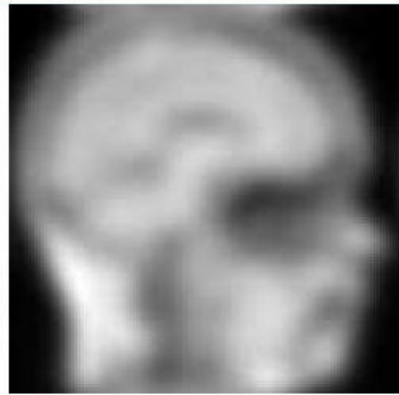
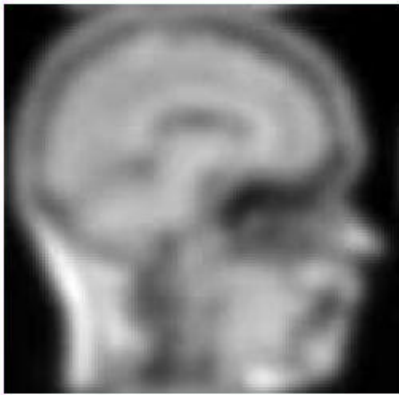
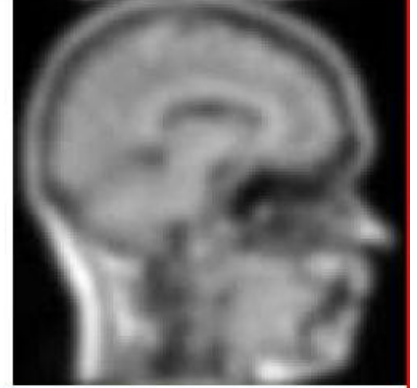
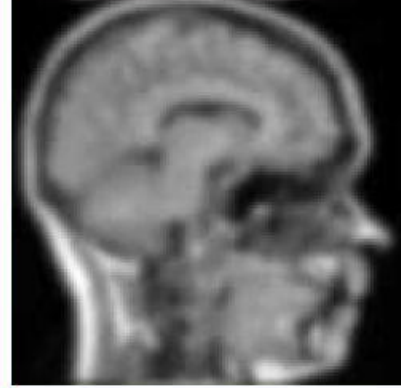
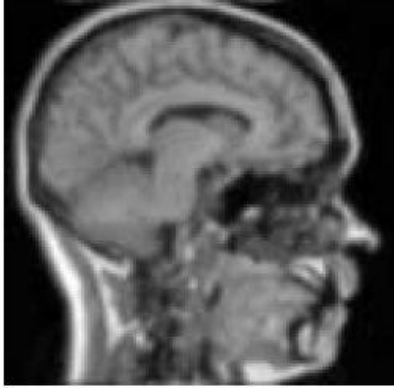


Smoothing by Averaging

Gaussian Scale Space (increasing σ)



Gaussian Scale Space (increasing σ)



Noise Filtering

- Goal is to remove noise and still preserve image structure (edges)



Gaussian Noise

- Gaussian smoothing preserves edges better than average filter
- Gaussian filter best at removing Gaussian noise (can prove this)



After Averaging



After Gaussian Smoothing

Noise Filtering

- Neither Gaussian nor average filter removes salt and pepper noise



Salt-and-pepper noise



After averaging



After Gaussian smoothing

Nonlinear Filtering – median filter

Replace each pixel value $I(i, j)$ with the median of the values found in a local neighbourhood of (i, j) .

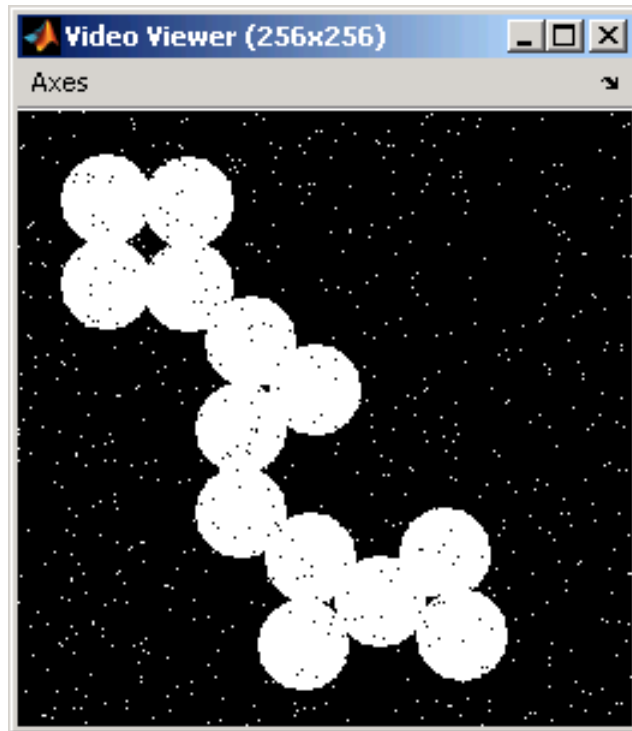
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

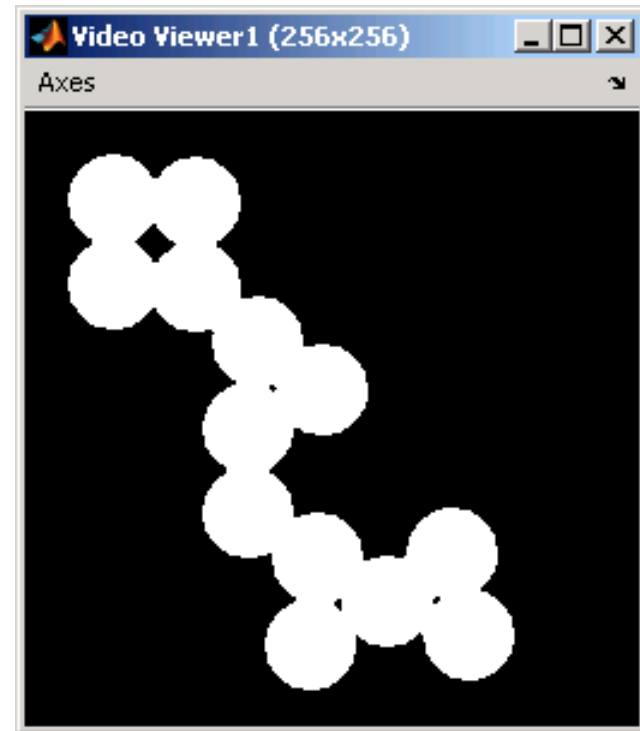
115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Median Filter

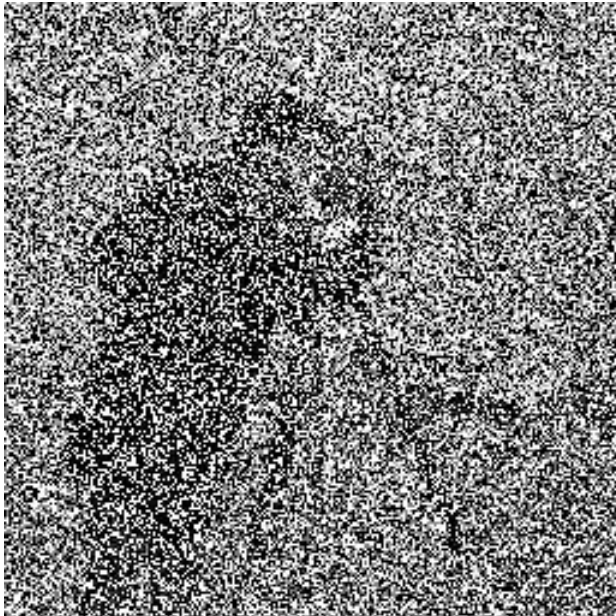


Salt-and-pepper noise



After median filtering

Remove noise and preserve edges!



Salt-and-Pepper Noise Removal by Median-type Noise Detectors and Edge-preserving Regularization

Raymond H. Chan, Chung-Wa Ho, and Mila Nikolova

IEEE Transactions on Image Processing, 14 (2005), 1479-1485.