Filtering (II)

Dr. Gerhard Roth

COMP 4102A
Winter 2016
Version 1

Image Filtering

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

N(p)

p

10	30	10
20	11	20
11	9	1

$f(p)$

	5.7	

Pseudo Code for Image Filter

- For every row i
- For every column j
- Take Pixel(i, j) and all the pixels in the nbhd
- Nbhd is a small set of pixels around Pixel(i, j)
- Nbhd can be any size or shape but is small
- A function f that has input all pixels in nbhd
- Pixel in New Image(i, j) = f (pixels in nbhd)
- New image and old image are distinct!
- Question is what is nbhd and what is f?
- Nbhd square(not always) and odd size

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

Image

*

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

Kernel

=

	5	

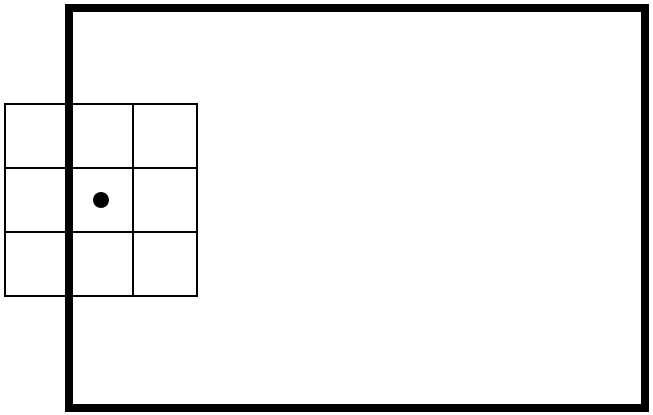
Filter Output

Handle Border Pixels

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

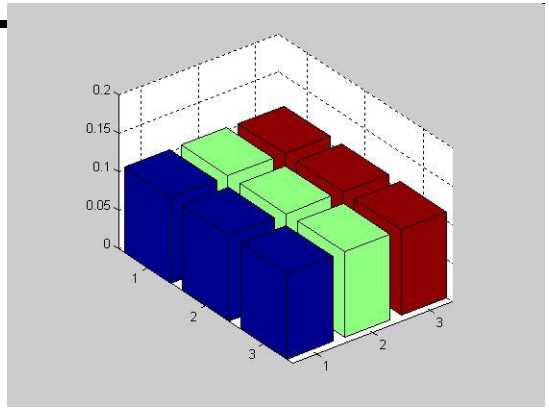
- Set the value of all non-included pixels to zero.

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



Smoothing by Averaging

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

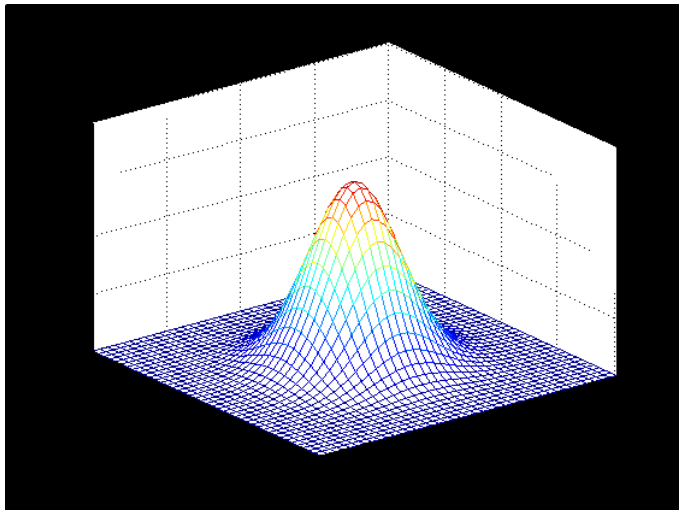


$$* \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$$

Convolution can be understood as weighted averaging.

Gaussian Filter

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

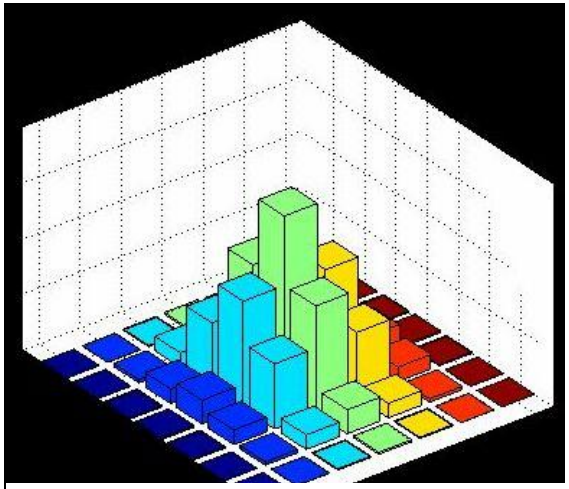


Discrete Gaussian kernel:

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

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

Gaussian Filter



$$* \quad \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} =$$

$$\sigma = 1$$

Which kernels are Separable?

- A kernel is separable if it can be written as the outer product of two 1d kernels
  - Say 1d horizontal kernel is V – dimensions m by 1
  - And 1d vertical kernel is H^T- dimensions 1 by m
- Then the kernel is V H^T has dimensions of m x 1 times 1 x m, which is m x m
- Many important kernels are separable
- For such kernels the complexity of the convolution is O(m) instead of O(m^2)
  - This is a very important computational advantage
  - Can have larger kernels (say up to m 10) on large images
  - Can also do multiple operations with different kernels

Outer product – examples

$$\begin{bmatrix} ax & ay & az \\ bx & by & bz \\ cx & cy & cz \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix}$$

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

Gaussian Kernel is Separable

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

$$\boldsymbol{I} \; * \; \frac{1}{38}\begin{array}{|c|c|c|c|c|}\hline 1 & 9 & 18 & 9 & 1 \\ \hline\end{array} \; = \; \boldsymbol{I_r}$$

$$\boldsymbol{I_r} \; * \; \frac{1}{38}\begin{array}{|c|}\hline 1 \\ \hline 9 \\ \hline 18 \\ \hline 9 \\ \hline 1 \\ \hline\end{array} \; = \; \text{result}$$

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

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

Outer product – more examples

- Most important kernels used in practice are separable

average

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

bilinear

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

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

Gaussian

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Sobel

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\frac{1}{4}\begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix} \quad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Gaussian vs. Average

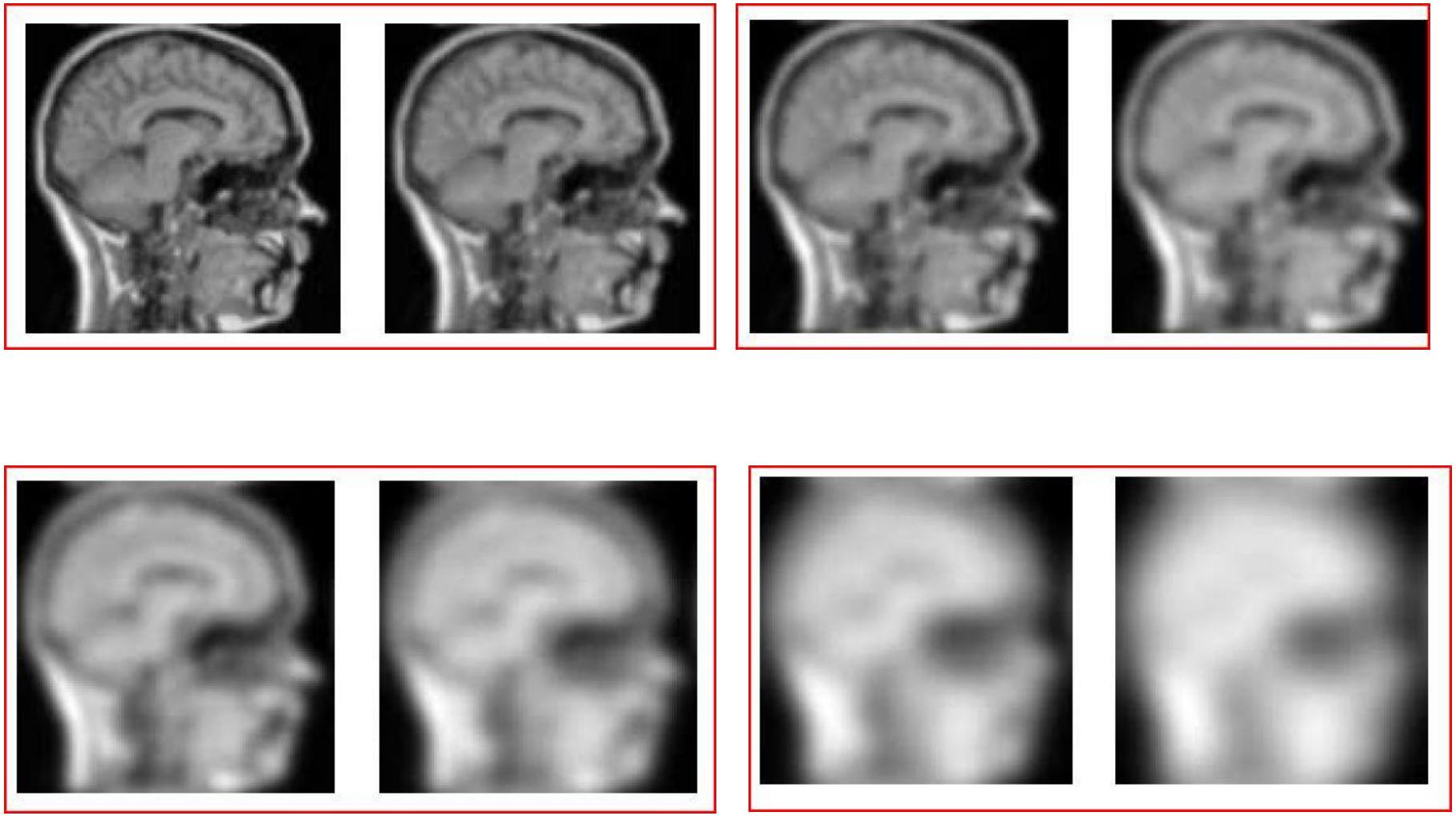
- Gaussian and average are smoothing linear filters
- In this case sum of all kernel entries is one
- So that new pixel is in same value range as old

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).

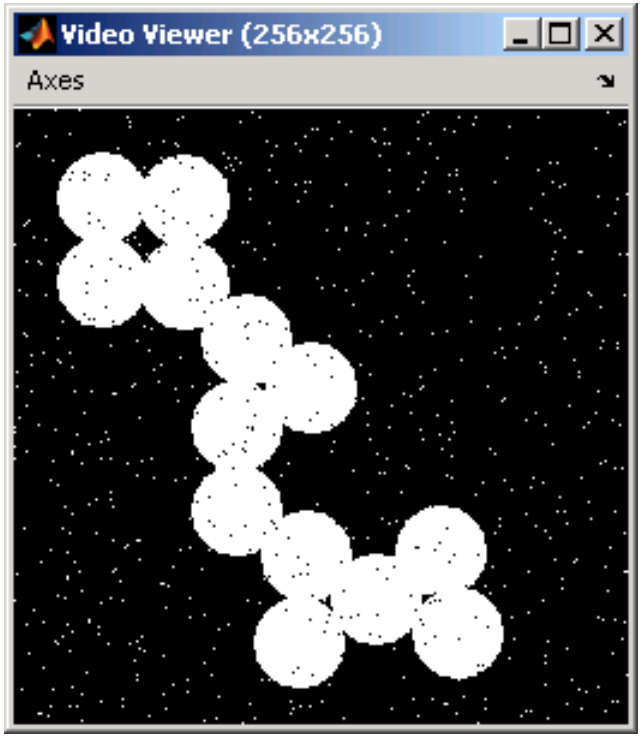
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

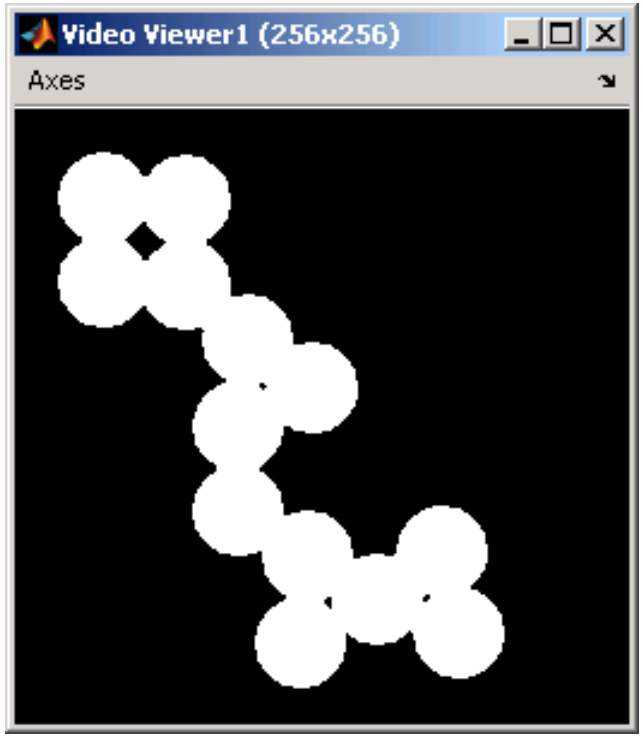
115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Median Filter

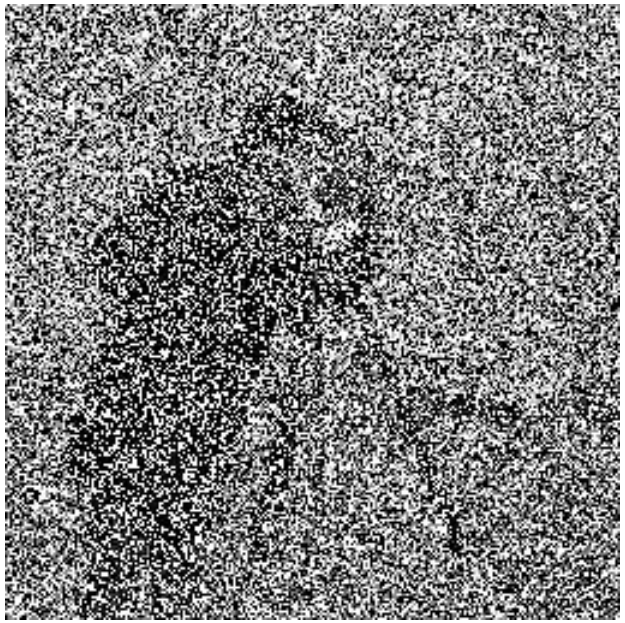


Salt-and-pepper noise



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.

Images are a type of signal

- All signals can be written as a sum of individual frequencies (like your voice)
- Images are a type of signal with both low and high frequencies
  - Low frequencies = area of little change (i.e. constant
  - High frequencies = areas with a lot of change (i.e. edges)
- Filters change image frequency content
  - Smoothing (all kernel coefficients positive) suppress high frequencies (low pass = pass low frequencies)
  - Attenuate or suppress edges
  - Edge magnify (some kernel coefficients negative) suppress low frequencies (high pass = pass high frequencies
  - Enhance or magnify edges

Image filtering summary

- Image filters have an image as input and as output and operate on small neighbourhoods
- Type of function used to compute new pixels determines the type of filter that you have
- Linear filters are defined by convolution
- There are low pass and high pass linear filters (low = only +, high = some – in kernels)
- Applying filters to an image makes a new image with different frequency content
- Smoothers enhance low frequency, edge magnifiers enhance high frequency