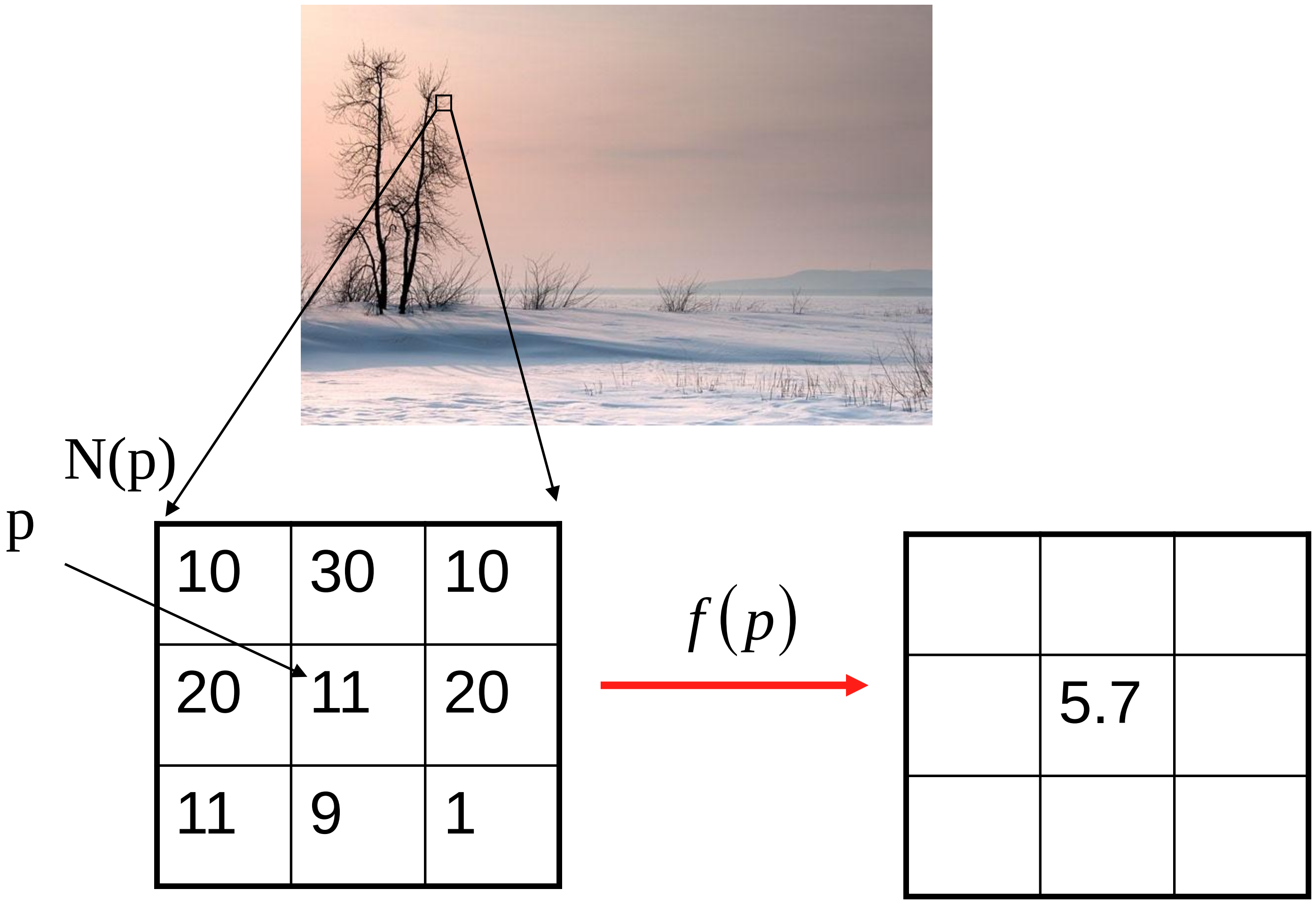
Filtering (II)

Dr. Gerhard Roth

COMP 4102A
Winter 2013

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.

Image:

1	3	0
2	10	2
4	1	1

*

Kernel:

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

=

Filter Output:

	5	

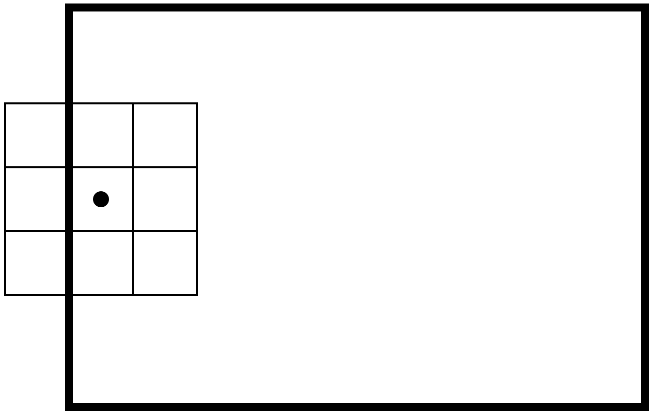
Image | Kernel | Filter Output

Handle Border Pixels

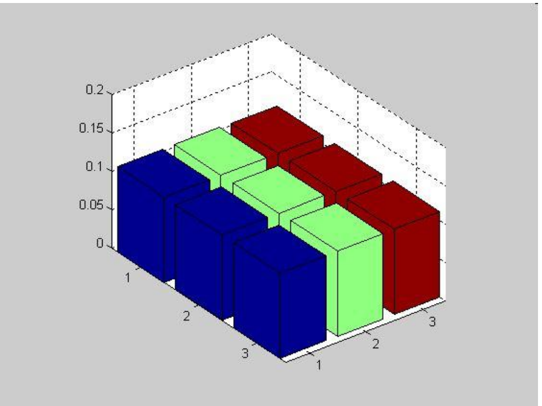
Near the borders of the image, some pixels do not have enough neighbours. Two possible solutions are:

- 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



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

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

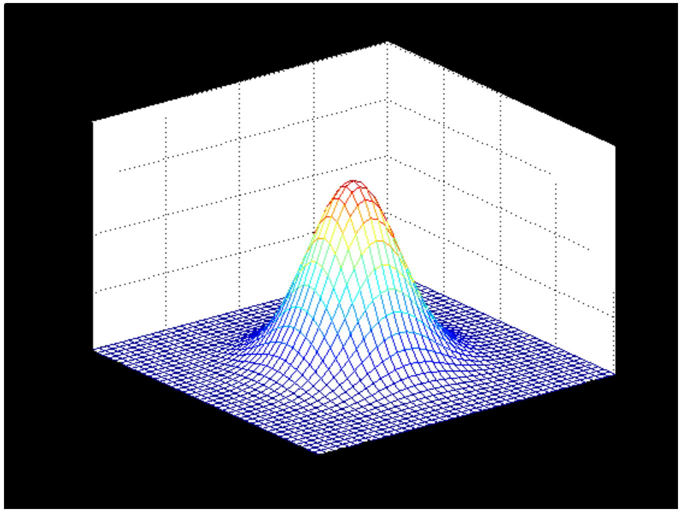
1	1	1
1	1	1
1	1	1

$=$

Convolution can be understood as weighted averaging.

Gaussian Filter

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

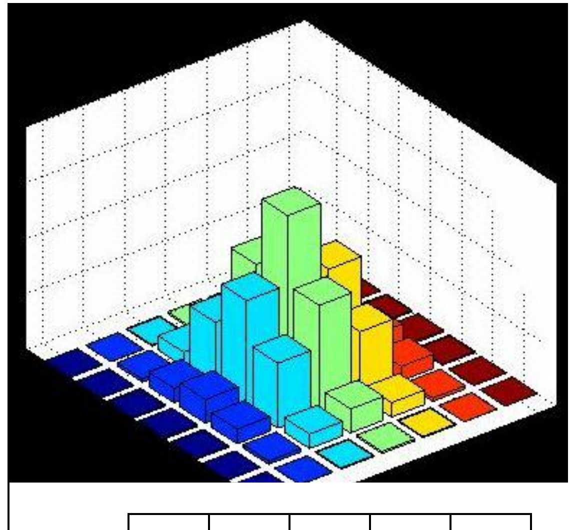


Discrete Gaussian kernel:

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

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

Gaussian Filter



$$* \ \frac{1}{273}$$

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.

$$I * \frac{1}{38}\begin{bmatrix} 1 & 9 & 18 & 9 & 1 \end{bmatrix} = I_r$$

$$I_r * \frac{1}{38}\begin{bmatrix} 1 \\ 9 \\ 18 \\ 9 \\ 1 \end{bmatrix} = \text{result}$$

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

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 – m by 1
  - And 1d vertical kernel is H^T- 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

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1v_1 & u_1v_2 & u_1v_3 \\ u_2v_1 & u_2v_2 & u_2v_3 \\ u_3v_1 & u_3v_2 & u_3v_3 \\ u_4v_1 & u_4v_2 & u_4v_3 \end{bmatrix}$$

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

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

Outer product – more examples

- Most important kernels used in practice are separable

average

$\frac{1}{K^2}$

1	1	$\cdots$	1
1	1	$\cdots$	1
$\vdots$	$\vdots$	1	$\vdots$
1	1	$\cdots$	1

$\frac{1}{K}$

1	1	$\cdots$	1

bilinear

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{4}$

1	2	1

Gaussian

$\frac{1}{256}$

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

$\frac{1}{16}$

1	4	6	4	1

Sobel

$\frac{1}{8}$

−1	0	1
−2	0	2
−1	0	1

$\frac{1}{4}$

−1
−2
−1

$\frac{1}{2}$

−1	0	1

Gaussian vs. Average

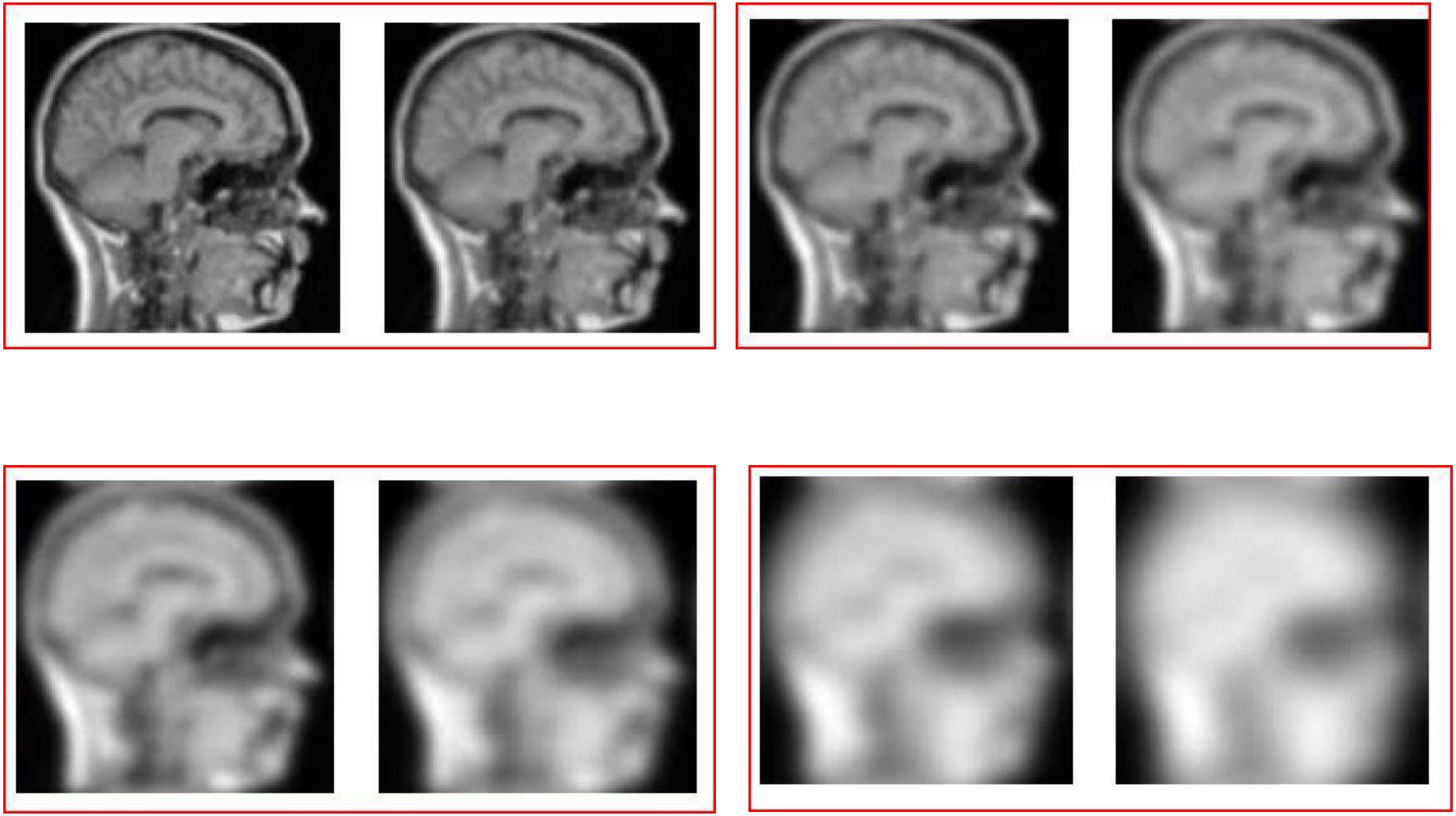
- Gaussian and average are smoothing linear filters
- In this case sum of all kernel entries is one
- So that new pixel is in same value range as old

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

After Averaging

After Gaussian Smoothing

•Gaussian smoothing preserves edges better than average filter

•Gaussian filter best at removing Gaussian noise (can prove this)

Noise Filtering

•Neither Gaussian nor average filter removes salt and pepper noise

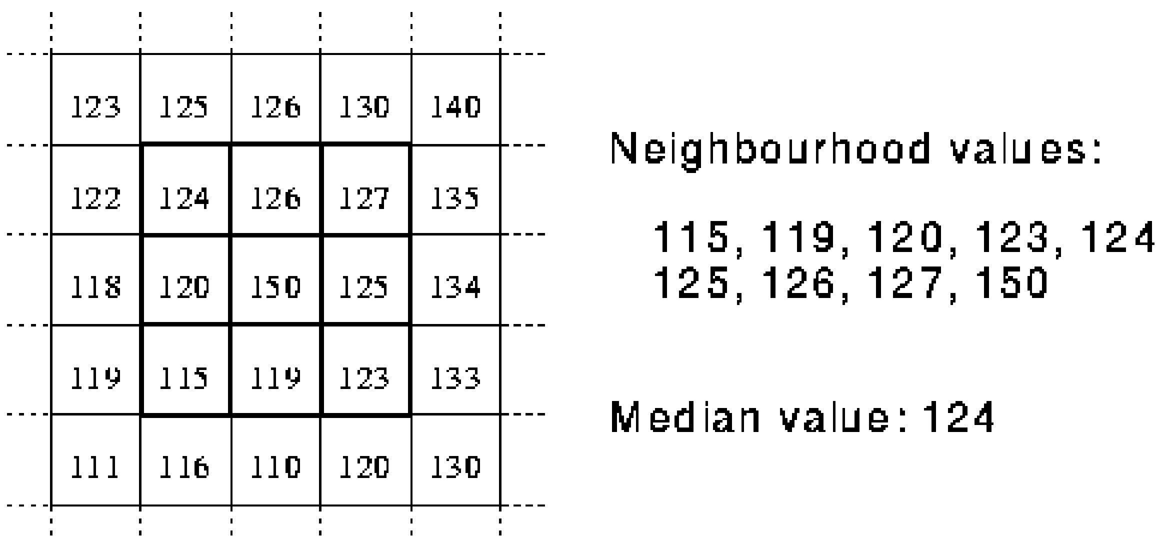
After averaging

Salt-and-pepper noise

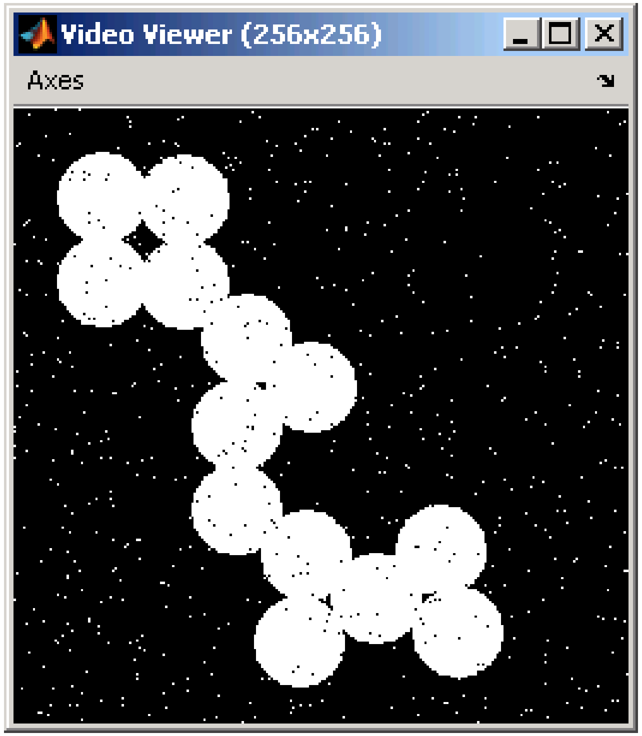
After Gaussian smoothing

Nonlinear Filtering – median filter

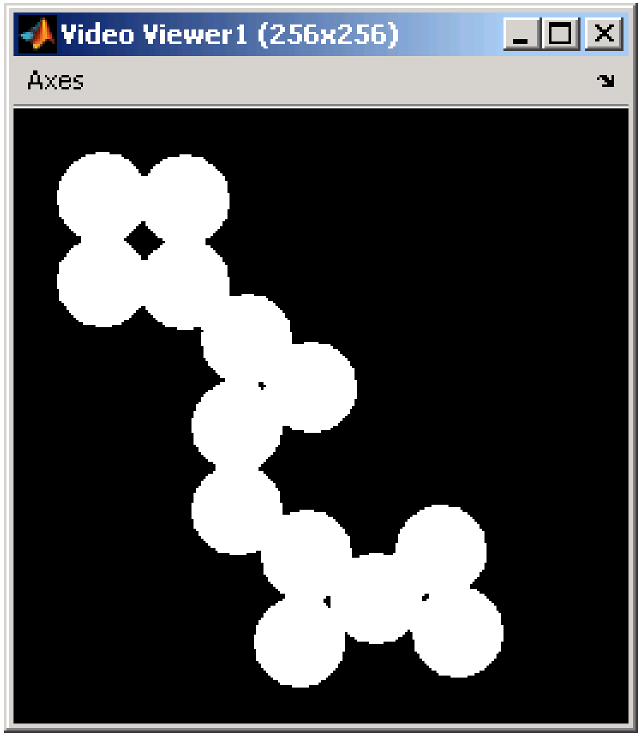
Replace each pixel value $I(i, j)$ with the median of the values found in a local neighbourhood of (i, j).



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.