

Noise, Filtering and Smoothing



Dr. Elli Angelopoulou

Pattern Recognition Lab (Computer Science 5)

University of Erlangen-Nuremberg

Noise Sources



- Photon noise: variation in the #photons falling on a pixel per time interval T .
- Saturation: each pixel can only generate a limited amount of charge.
- Blooming: saturated pixel can overflow to neighboring pixels.



Noise Sources - continued



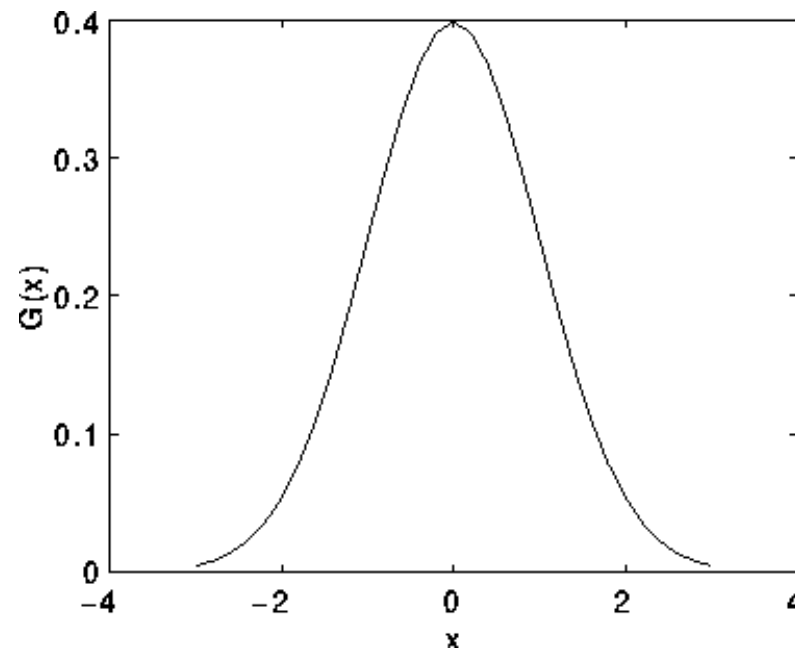
- Thermal noise: heat can free electrons and generate a response when there is no real signal.
- Electronic noise.
- Burned pixels.
- Black is not black.
- Keep in mind: Camera response may not be linear over the number of photons falling on a surface (camera gamma)



Detector Noise



- Source of noise: the discrete nature of radiation, i.e. the fact that each imaging system is recording an image by counting photons.
- Can be modeled as an independent additive noise which can be described by a zero-mean Gaussian.



Salt and Pepper Noise



- A common form of noise is caused by *data drop-out noise*.
- It is also known as commonly referred to as intensity spikes, speckle, or salt and pepper noise.
- Sources of error:
 - Errors in the data transmission.
 - Burned pixels: the corrupted pixels are either set to the maximum value (which looks like snow in the image) or are set to zero ("peppered" appearance), or a combination of the two.
 - Single bits are flipped over.
- Isolated/localized noise. It only affects individual pixels.

Filtering



- Most of the images we capture are noisy.

- Goal:



- This notion of filtering is more general and can be used in a wide range of transformations that we may want to apply to images.



- Mathematically, a filter H can be treated as a function on an input image I :

$$H(I) = R$$

- Note: We use the terms *filter* and *transformation* interchangeably.

Linear Transformation



- A transformation H is **linear** if, for any inputs $I_1(x,y)$ and $I_2(x,y)$ (in our case input images), and for any constant scalar α we have:

$$H(\alpha I_1(x,y)) = \alpha H(I_1(x,y))$$

and

$$H(I_1(x,y) + I_2(x,y)) = H(I_1(x,y)) + H(I_2(x,y))$$

- This means:
 - Scaling of the input corresponds to scaling of the output.
 - Filtering an additive image is equivalent to filtering each image separately and then adding the results.

Shift-Invariant Transformation



- A transformation H is **shift-invariant** if for every pair (x_0, y_0) and for every input image $I(x, y)$, such that

$$H(I(x, y)) = R(x, y)$$

we get

$$H(I(x - x_0, y - y_0)) = R(x - x_0, y - y_0)$$

- This means that the filter H does not change as we shift it in the image (as we move it from one position to the next).

Convolution



- If a transformation (or filter) is linear shift-invariant (LSI) then one can apply it in a systematic manner over every pixel in the image.
- **Convolution** is the process through which we apply **linear shift-invariant filters** on an image.



- Convolution is defined as:

$$R(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} H(x - i, y - j) I(i, j)$$

and is denoted as:

$$R = H * I$$



Another Look at Convolution

- Filtering often involves replacing the value of a pixel in the input image F with the weighted sum of its neighbors.
- Represent these weights as an image, H
- H is usually called the **kernel**
- The operation for computing this weighted sum is called **convolution**.

$$R = H * I$$

- Convolution is:

- commutative, $H * I = I * H$
- associative, $H_1 * (H_2 * I) = (H_1 * H_2) * I$
- distributive, $(H_1 + H_2) * I = (H_1 * I) + (H_2 * I)$

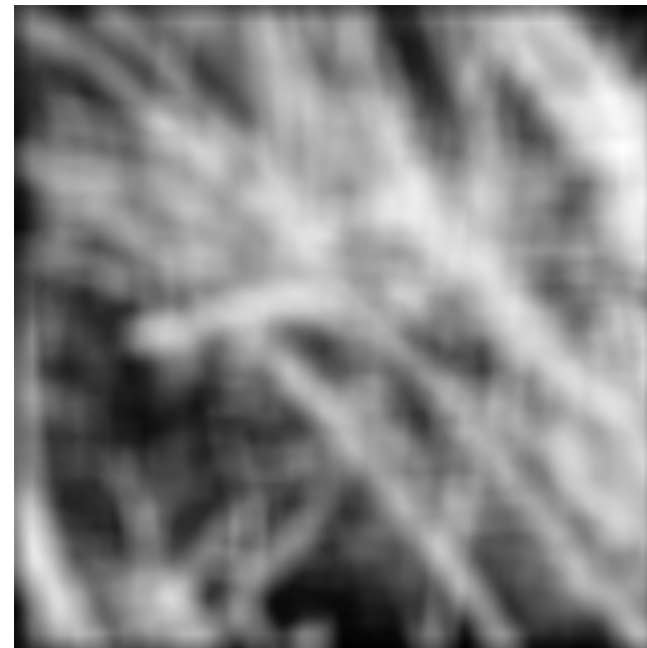
Smoothing via Simple Averaging



- One of the simplest filters is the mean filter: $H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$
- In this case, $R(x, y) = \sum_{i=-1}^1 \sum_{j=-1}^1 I(x-i, y-j)H(i, j)$
- It is used for removing image noise, i.e. for smoothing.



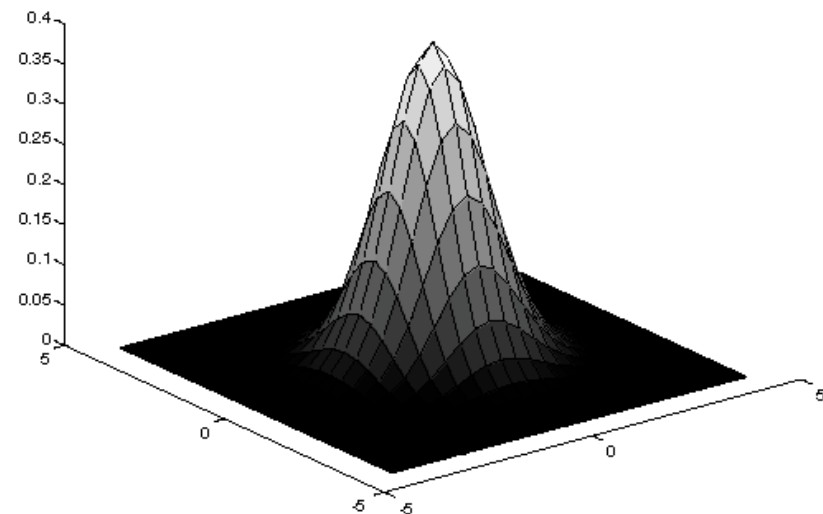
*  =



Gaussian Smoothing



- Idea: Use a weighted average. Pixels closest to the central pixel are more heavily weighted.
- The Gaussian function has exactly that profile.
- Gaussian also better approximates the behavior of a defocused lens.



Isotropic Gaussian Filter



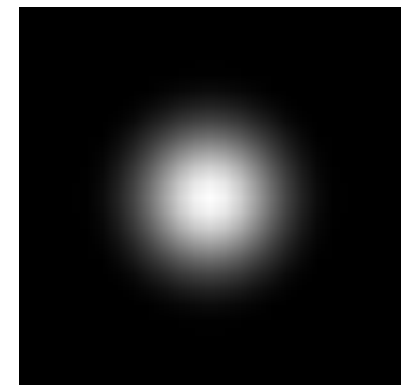
- To build a filter H , whose weights resemble the Gaussian distribution, assign the weight values on the matrix H according to the Gaussian function:

$$H(i, j) = e^{-(i^2 + j^2)/2\sigma^2}$$

$$H_{Gauss} = \begin{bmatrix} 1/16 & 1/8 & 1/16 \\ 1/8 & 1/4 & 1/8 \\ 1/16 & 1/8 & 1/16 \end{bmatrix}$$

- Small σ , almost no effect, weights at neighboring points are negligible.
- Large σ , blurring, neighbors have almost the same weight as the central pixel.
- Commonly used σ values: Let w be the size of the kernel H . Then $\sigma = w/5$.

For example for a 3x3 kernel, $\sigma = 3/5 = 0.6$




Gaussian Smoothing Example



- Compared to mean filtering, Gaussian filtering exhibits no “ringing” effect.



Original image

*  =

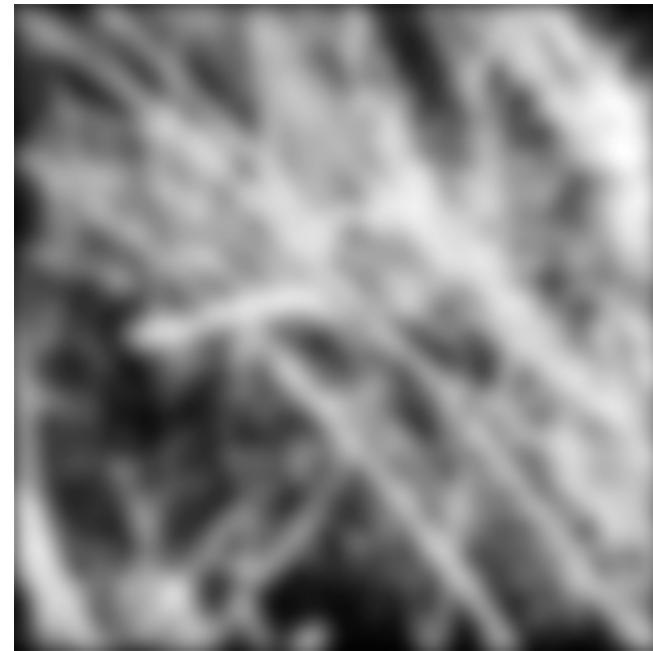


Image after Gaussian filtering (25x25 kernel)



“Ringing” effect



Original image



Image after Mean filtering (25x25 kernel)

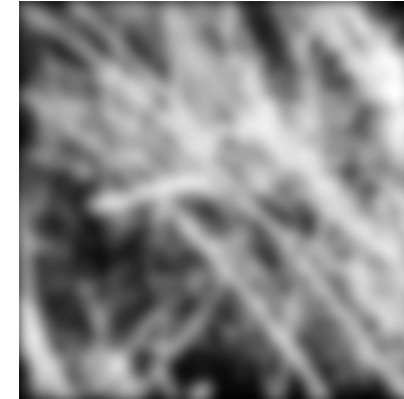
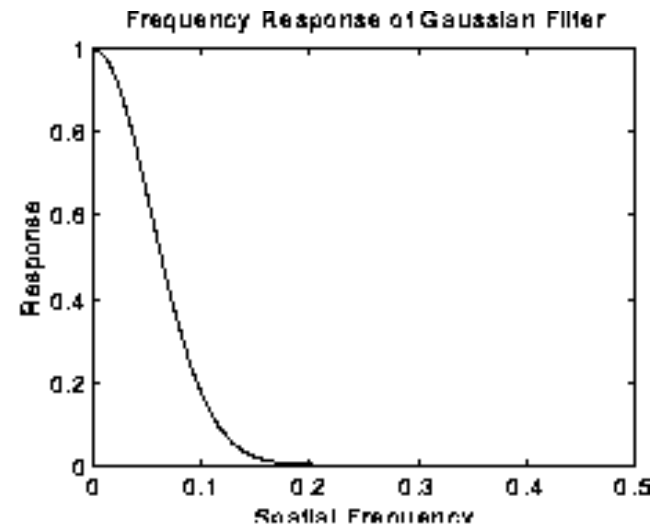
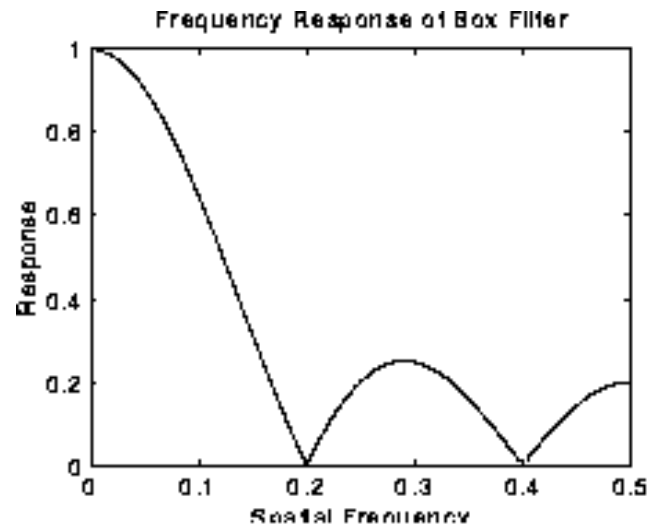


Image after Gaussian filtering (25x25 kernel)

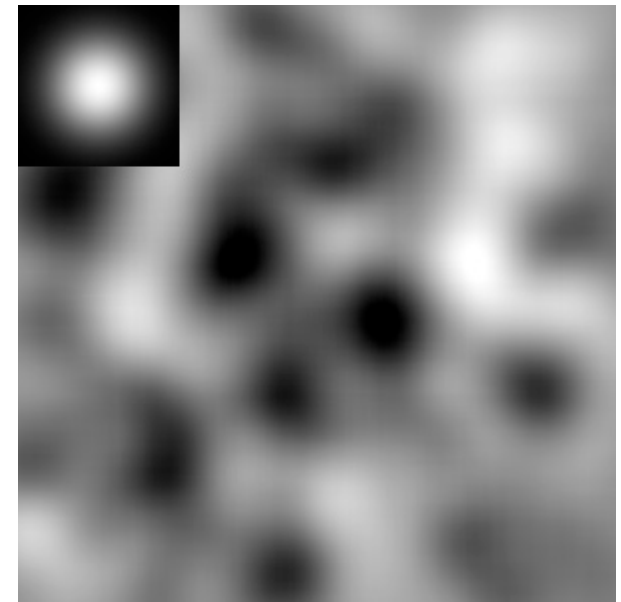
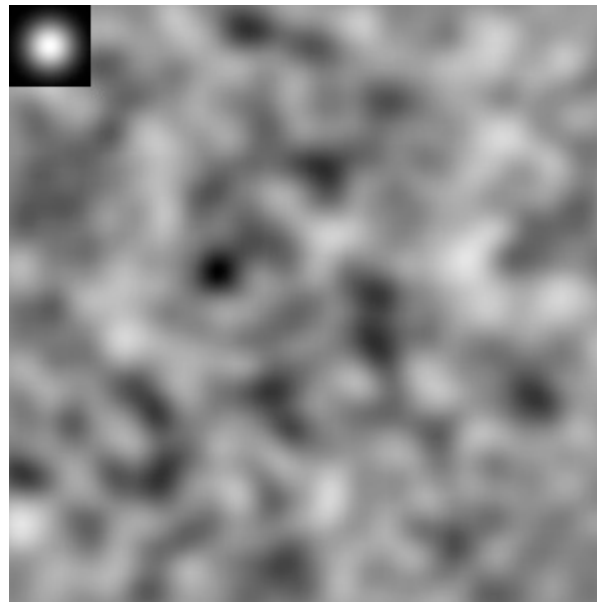
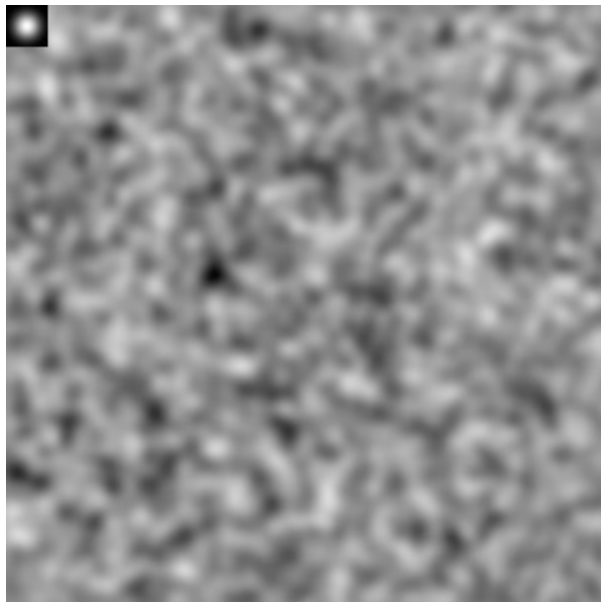


A close look at the frequency response of the two filters show that: compared to Gaussian filtering, mean filtering exhibits oscillations



The Effect of σ

- Different σ values affect the amount of blurring, but also emphasize different characteristics of the image.



Non-Linear Smoothing



- The **median** filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings.
- It replaces a pixel value with the median of all pixel values in the neighborhood.
- It is a relatively slow filter, because it involves sorting.
- Can **not** be implemented via convolution.



Smoothing Examples



Original image corrupted by a zero mean Gaussian noise with $\sigma=8$.



Image after 5x5 Mean filtering



Image after 5x5 Gaussian filtering

Mean Filter



Original image



Image after 3x3
Mean filtering



Image after 7x7
Mean filtering



Image after applying
3 times 3x3 Mean
filtering

- Mean filtering is sensitive to outliers.
- It typically blurs edges.
- It often causes a ringing effect.

Gaussian Filtering and Salt & Pepper Noise



Original image



Image with salt-pepper noise (1% prob. that a bit is flipped)



Image after 5x5 Gaussian filtering, $\sigma=1.0$



Image after 9x9 Gaussian filtering, $\sigma=2.0$

- Gaussian filtering works very well for images affected by Gaussian noise.
- It is not very effective in removing Salt and Pepper noise.

Median Filtering and Salt & Pepper Noise



Original image



Image with salt-pepper noise (5% prob. that a bit is flipped)



Image after 3x3 Median filtering



Image after 7x7 Median filtering



Image after applying 3 times 3x3 Median filtering

- Median filtering preserves high spatial frequency details.
- It works well when less than half of the pixels in the smoothing window have been affected by noise.
- It is not very effective in removing Gaussian noise.



Image Sources

1. "Image with salt & pepper noise", Marko Meza.
2. Many of the smoothing and edge detection images are from the slides by D.A. Forsyth, University of California at Urbana-Champaign.
3. The examples in slides 18-21 are courtesy of R. Fisher, S. Perkins, A. Walker and E. Wolfart