Digital Image Processing

Berlin University of Technology (TUB),
Computer Vision and Remote Sensing Group
Berlin, Germany

Last Exercise - Theory

1. What is a digital image?

→ A projection of the continuous 3D world into a discrete 2D map. An image maps spatial coordinates $(x, y) \in \mathbb{N}$ to discrete intensity values $f(x, y)$, e.g. $f(x, y) \in \mathbb{N}^c$, where c the number of channels.

2. What does the paradigm "bottom-up processing" mean?

→ Bottom-up processing refers to a data-driven processing scheme, where low-level features are successively grouped into complex structures and thus accumulating object evidence.

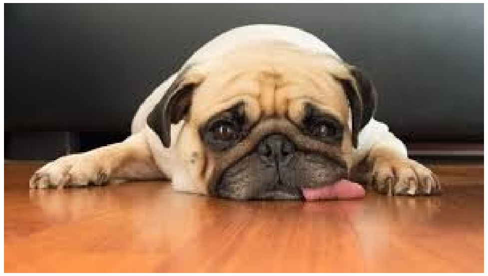
3. State at least three fundamentally different image sources!

→ Optical cameras:
passive sensor, visible part of the spectrum, object position depends on angle of visual ray

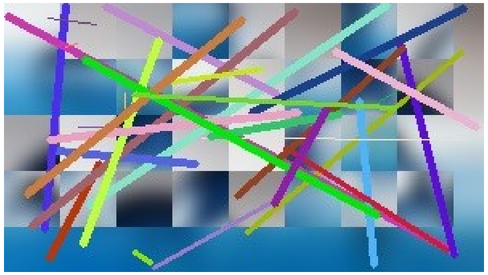
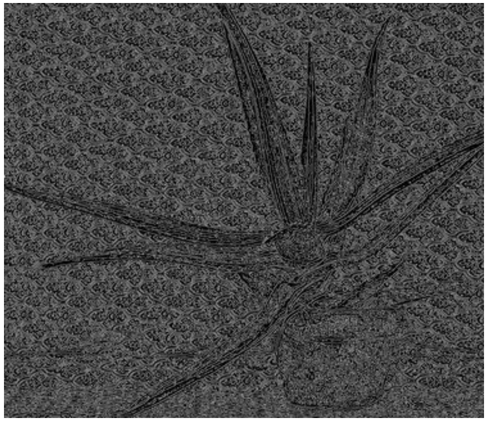
→ SAR:
active sensor, micro-waves, object position depends on distance

→ Ultrasound:
Active sensor, based on high-frequency sound pulses

Last Exercise - Examples

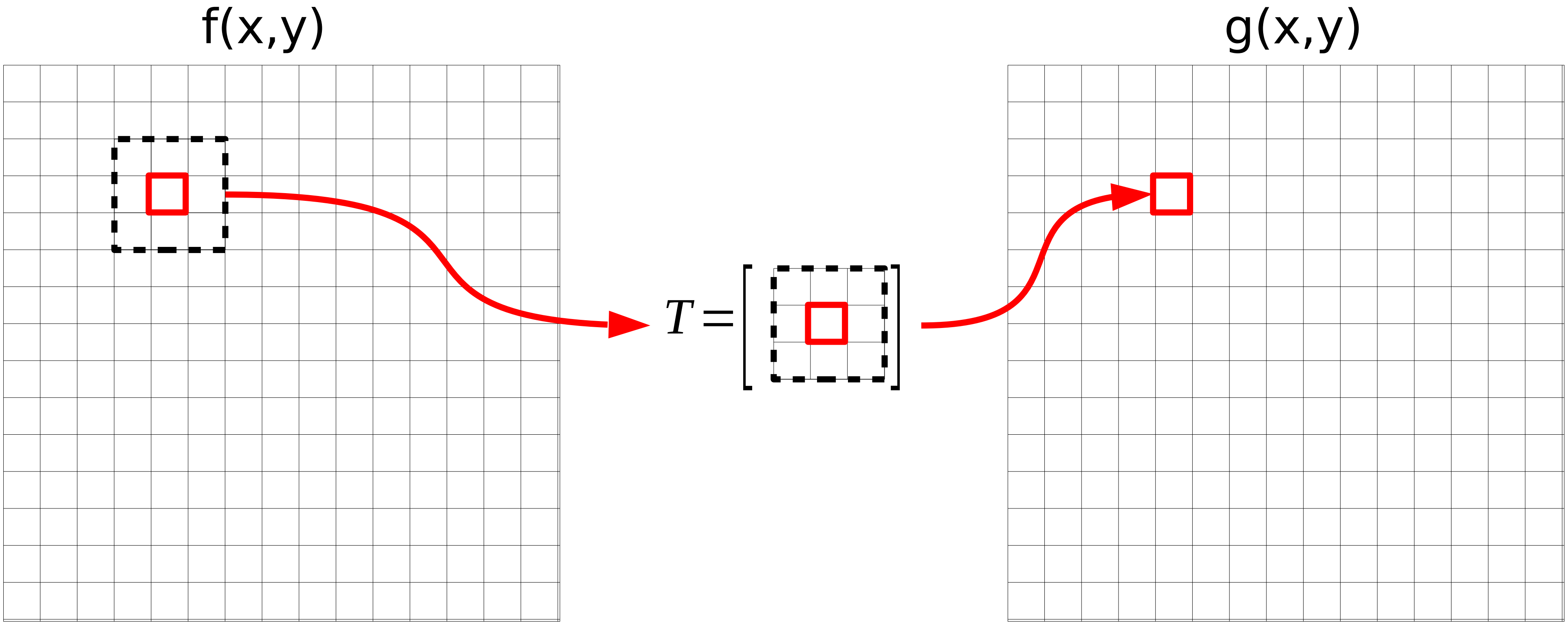



Purpose of Digital Image Processing

<u>Image restoration</u>: Improving *objective* image quality
e.g. noise suppression

Sliding Window

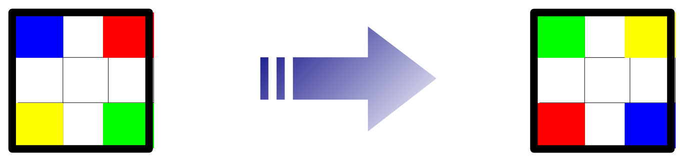


- Operator T takes into account only local information
- Result in g is based on pixel intensity and intensities of neighbours
  - '*Filter size*' refers to size of neighbourhood (e.g. 3x3 pixels)

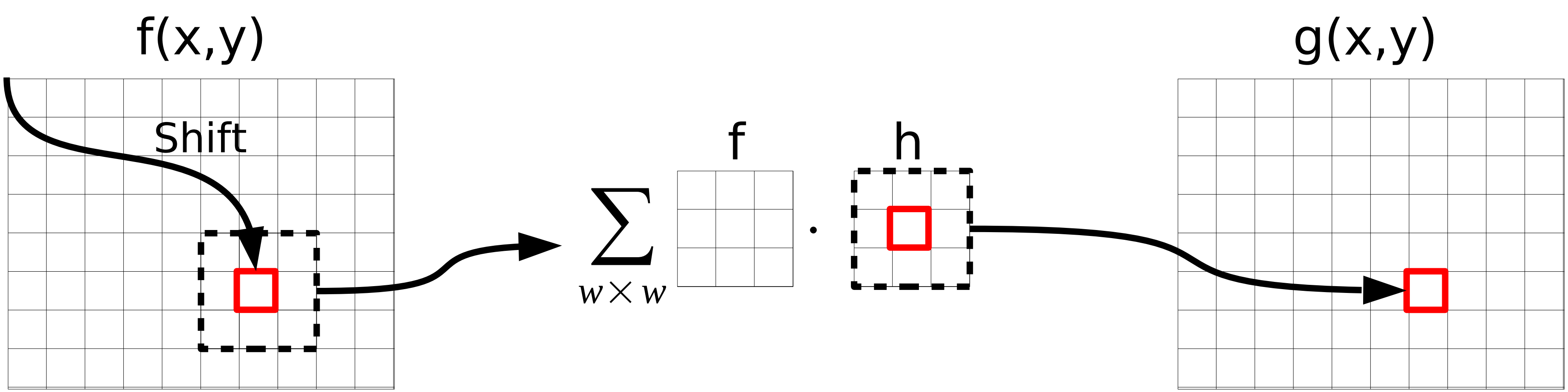
Convolution

$$g(\alpha,\beta)=\sum_{x=1}^{N}\sum_{y=1}^{M} f(x,y)\cdot h(x-\alpha,y-\beta)$$

1. **Flip filter kernel** (about the filter centre)



2. **Shift** (re-centre), **Multiply** and **Integrate**



$$\sum_{w\times w} f \cdot h$$

Convolution

- Filter consists of coefficients and has a anchor point:

$$h(r,s)=\begin{pmatrix} h(-1,-1) & h(0,-1) & h(1,-1) \\ h(-1,0) & \boxed{h(0,0)} & h(1,0) \\ h(-1,1) & h(0,1) & h(1,1) \end{pmatrix}$$

- Linear filters are applied by *convolution*:

$$g(x,y)=(f*h)(x,y)=$$

$$\sum_{3\times 3}\begin{pmatrix} f(x-1,y-1)h(1,1) & f(x,y-1)h(0,1) & f(x+1,y-1)h(-1,1) \\ f(x-1,y)h(1,0) & f(x,y)h(0,0) & f(x+1,y)h(-1,0) \\ f(x-1,y+1)h(1,-1) & f(x,y+1)h(0,-1) & f(x+1,y+1)h(-1,-1) \end{pmatrix}$$

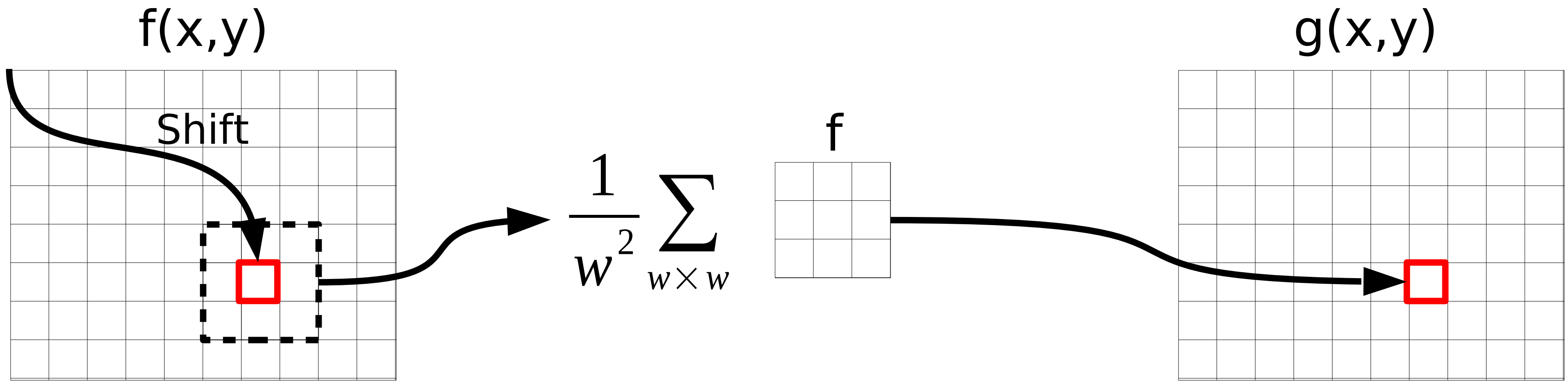
Filter Techniques

Example: Noise Suppression by Moving Average Filter

$$h(x,y)=\frac{1}{w^2}\begin{pmatrix} 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & \cdots \\ 1 & 1 & 1 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

(w x w Filter Kernel)

- Each pixel intensity is replaced by the local average...



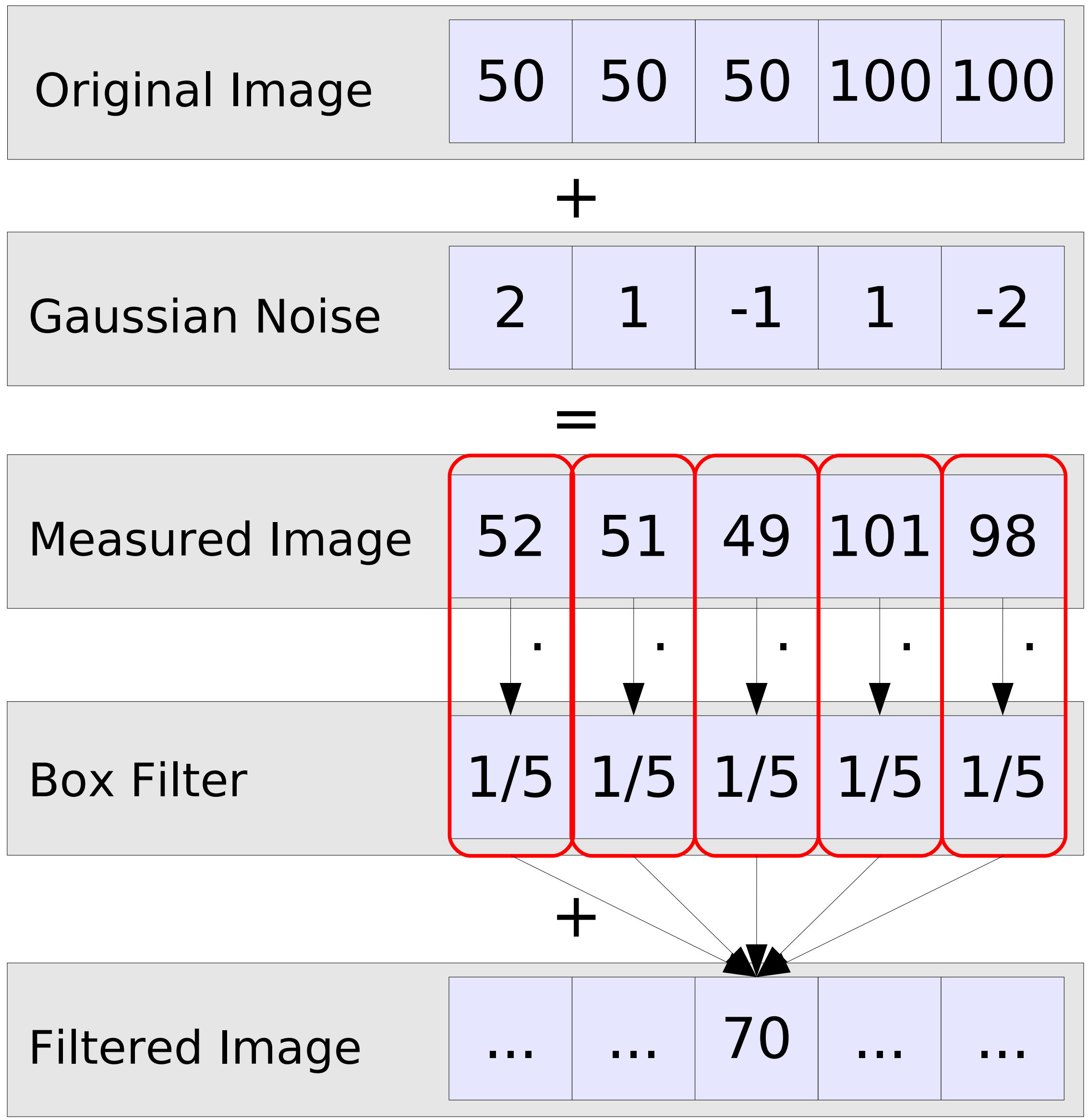
Filter Techniques

Example: Noise Suppression by Moving Average Filter

Original Image	100	100	100	100	100
+					
Gaussian Noise	2	1	-1	1	-2
=					
Measured Image	102	101	99	101	98
·	·	·	·	·	·
Box Filter	1/5	1/5	1/5	1/5	1/5
+					
Filtered Image	...	...	100	...	...

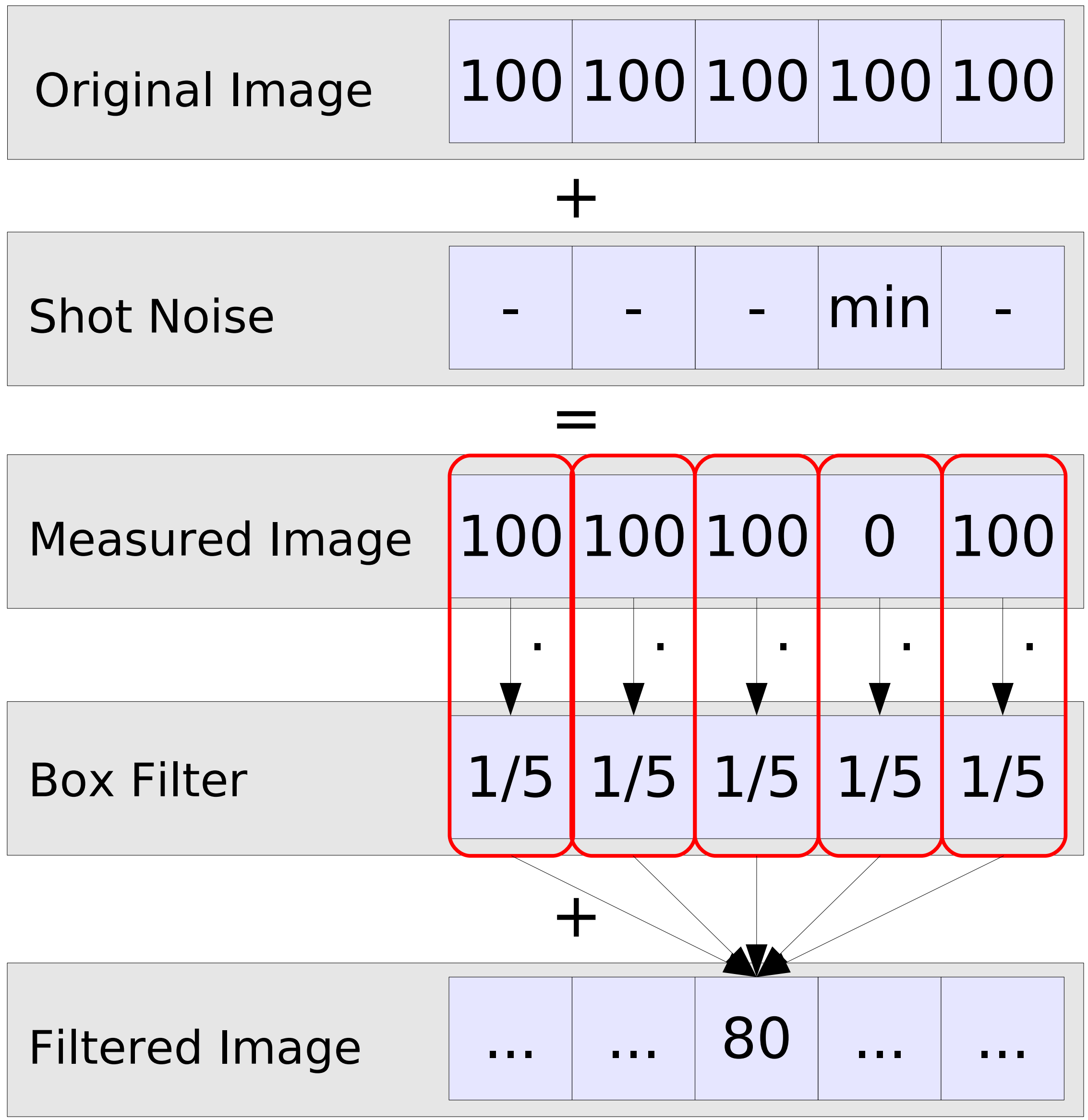
Filter Techniques

Example: Noise Suppression by Moving Average Filter



Filter Techniques

Example: Noise Suppression by Moving Average Filter



Filter Techniques

Example: Noise Suppression by Moving Average Filter

Gaussian Noise

f

w=5

w=11

w=25

Shot Noise

f

w=5

w=11

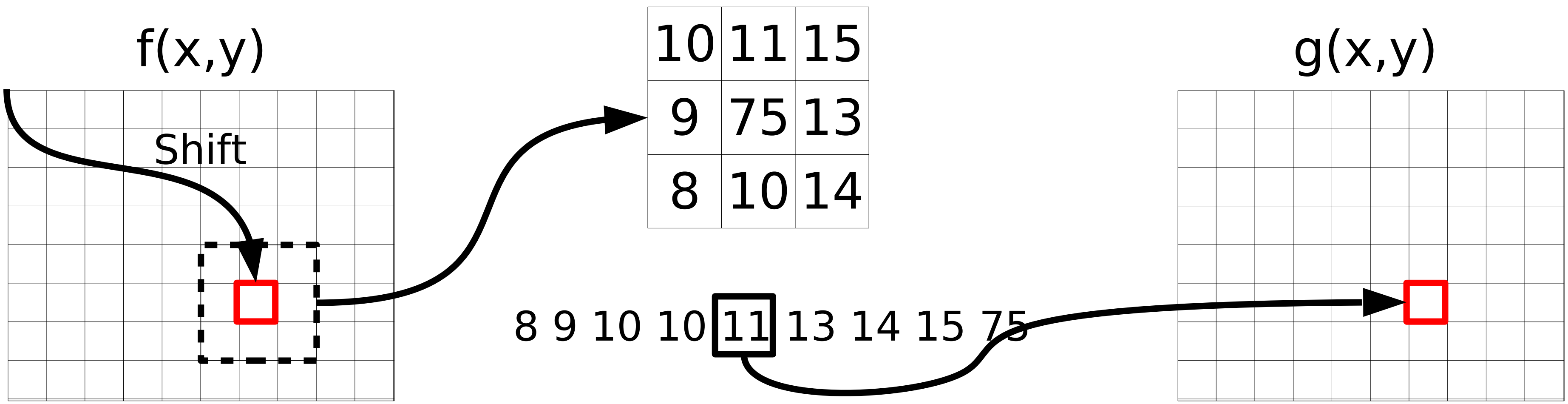
w=25

Filter Techniques

Example: Noise Suppression by Median Filter (NOTE: No convolution)

1. Consider intensities in a local NxN window
2. Sort intensities
3. Select middle value (median) as result

- Each pixel intensity is replaced by the local median...



10	11	15
9	75	13
8	10	14

- Effectively removes outliers
- Preserves sufficiently large (>> wxw) image structures

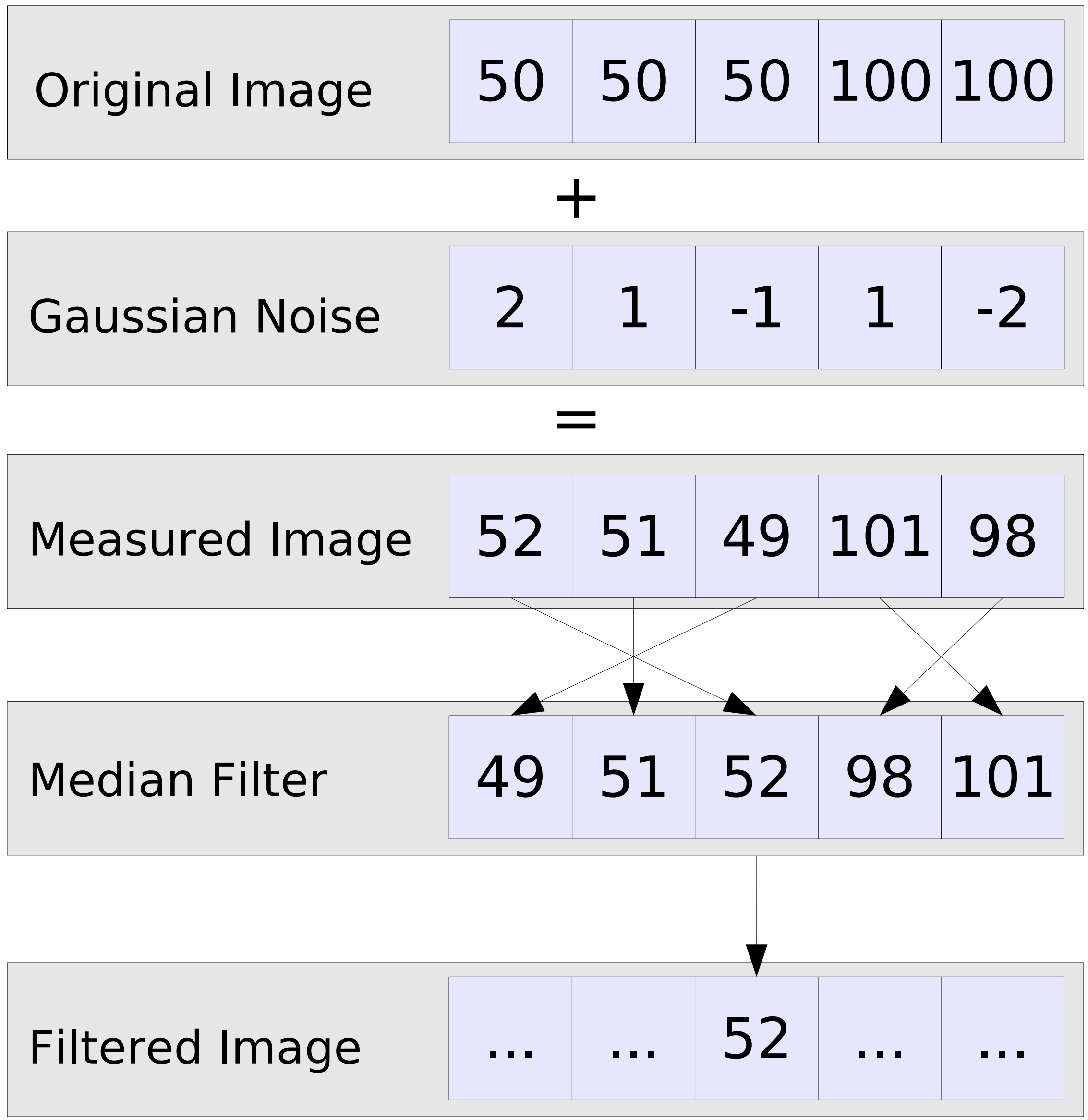
Filter Techniques

Example: Noise Suppression by Median Filter

Original Image	100	100	100	100	100
+					
Gaussian Noise	2	1	-1	1	-2
=					
Measured Image	102	101	99	101	98
Median Filter	98	99	101	101	102
Filtered Image	...	...	101	...	...

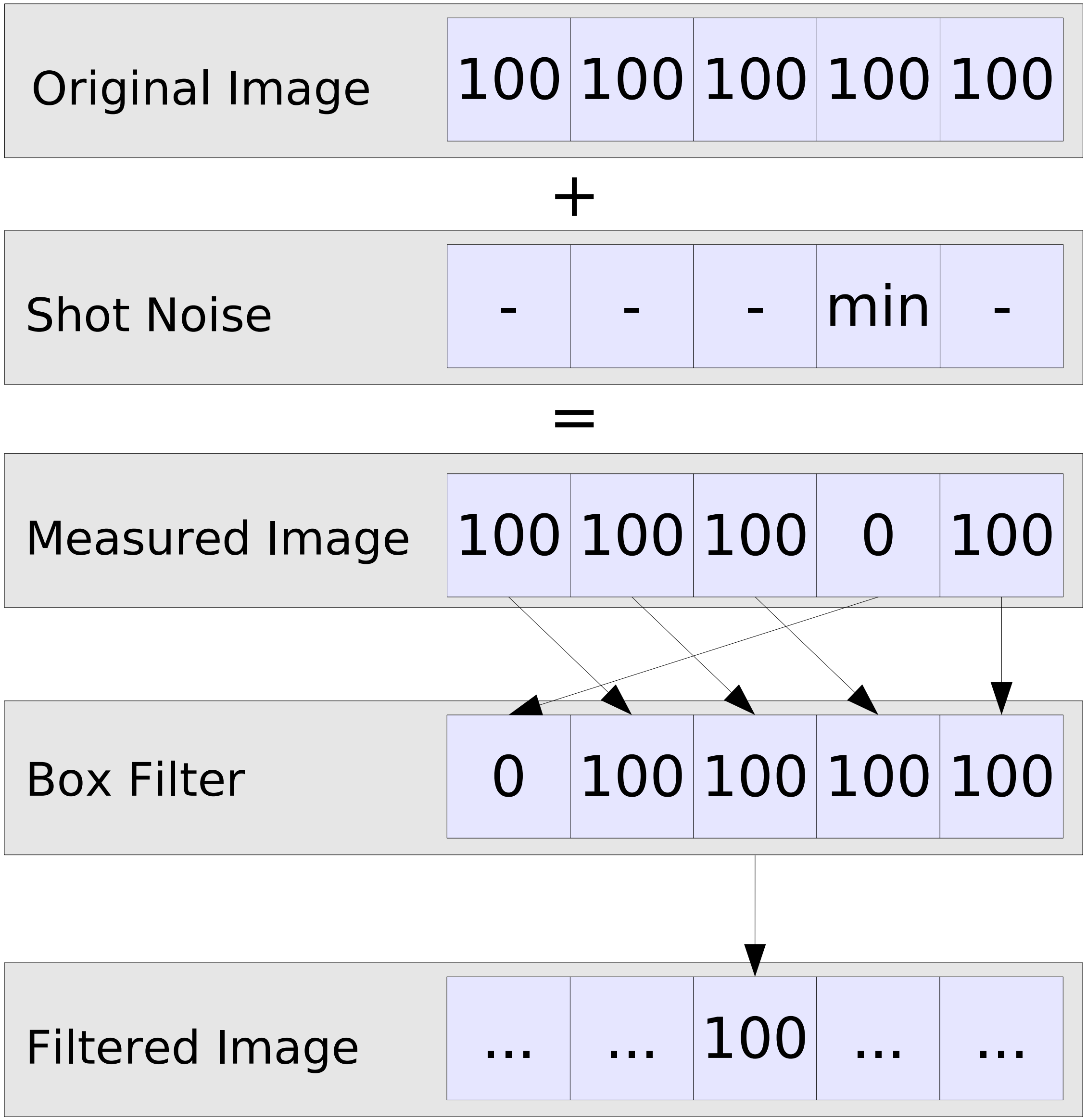
Filter Techniques

Example: Noise Suppression by Median Filter



Filter Techniques

Example: Noise Suppression by Median Filter



Filter Techniques

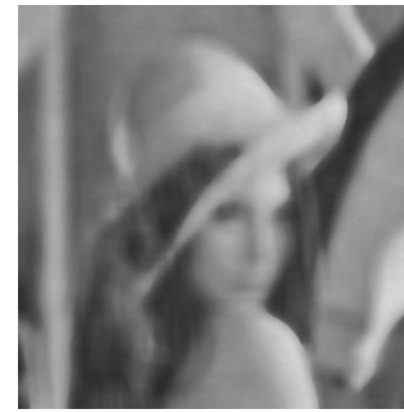
Example: Noise Suppression by Median Filter

Gaussian Noise

f

w=5

w=11

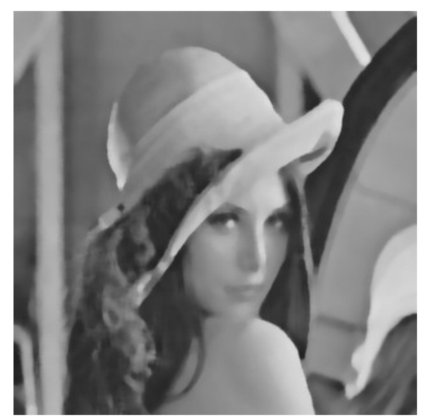


w=25

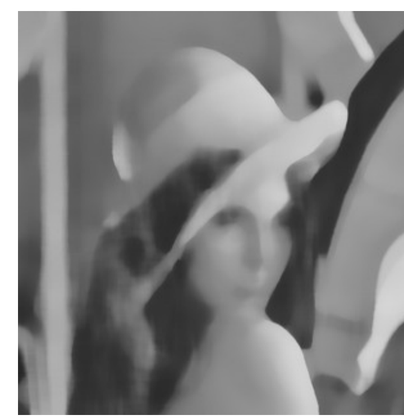
Shot Noise

f

w=5

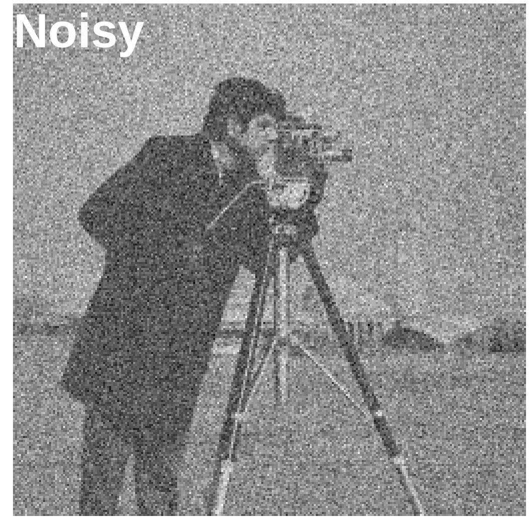


w=11

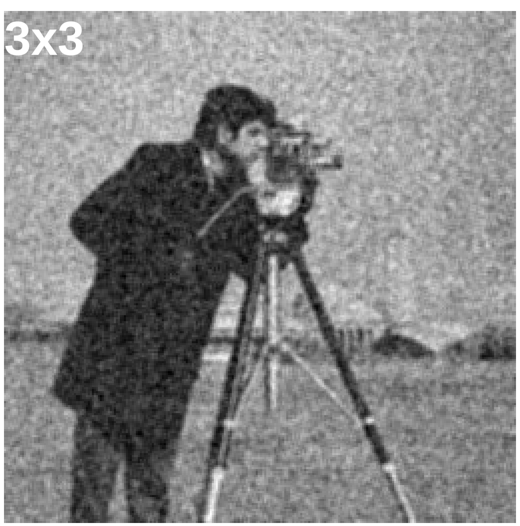


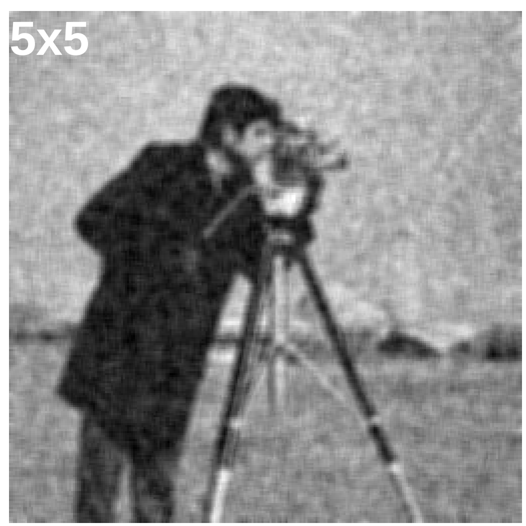
w=25

Noise Suppression vs. Resolution



Moving average filtering





Bilateral Filter

Spatial Weight:

$$h_{spat}(r,s)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(r-\mu)^2+(s-\mu)^2}{2\sigma^2}\right)$$

Radiometric Weight:

$$h_{rad}(p,q)=\frac{1}{2\pi\sigma_p^2}\exp\left(-\frac{(p-q)^2}{2\sigma_p^2}\right)$$

Combined:

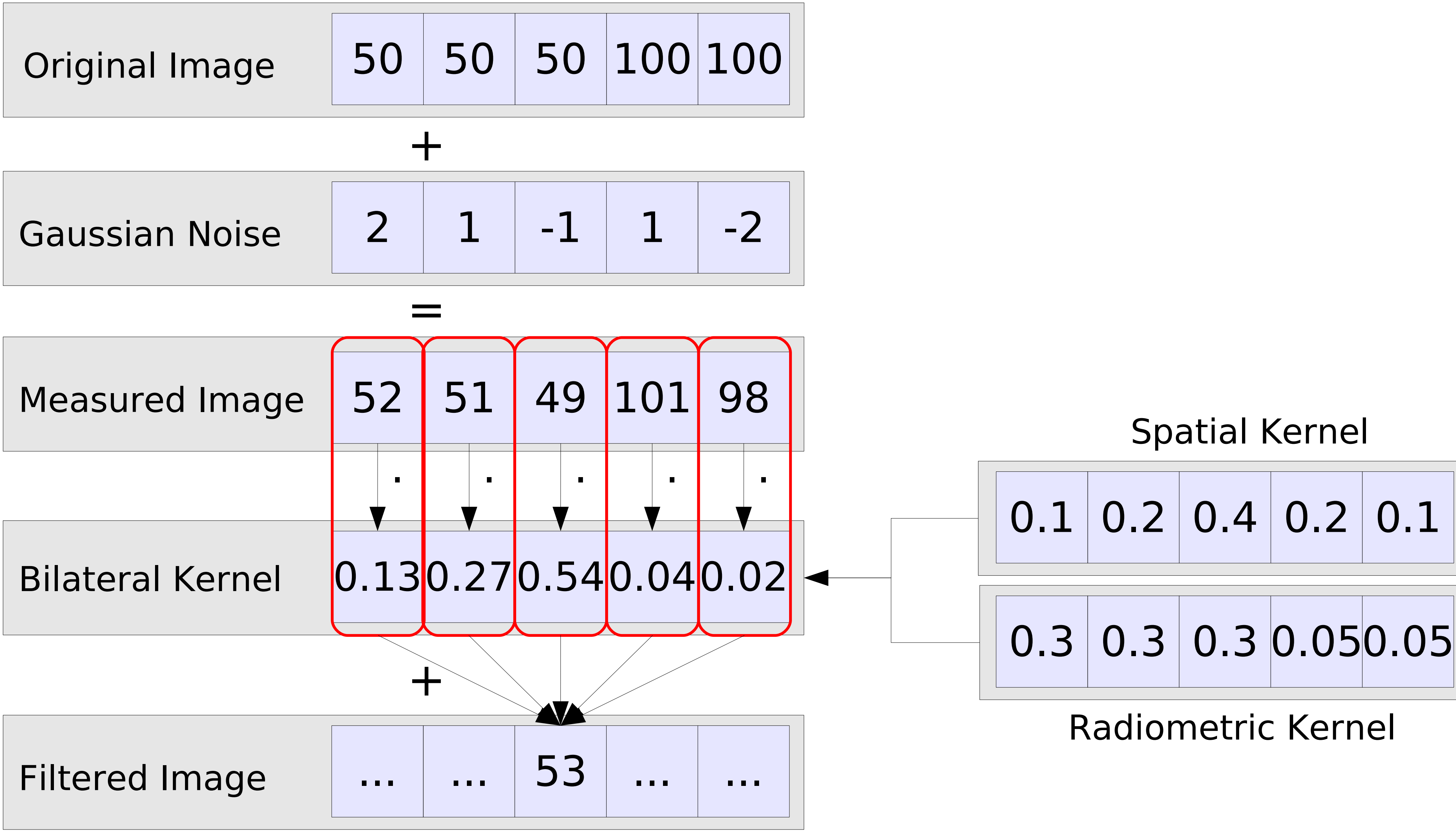
$$h(r,s,f(x,y))=h_{spat}(r,s)\cdot h_{rad}(f(x,y))$$

Output:

$$g(\vec{x})=\frac{1}{Z(\vec{x})}\sum_{\vec{x}'\in N(\vec{x})}\exp\left(-\frac{(\vec{x}'-\vec{x})^2}{2\cdot\sigma_1^2}\right)\cdot\exp\left(-\frac{(f(\vec{x}')-f(\vec{x}))^2}{2\cdot\sigma_2^2}\right)\cdot f(\vec{x}')$$

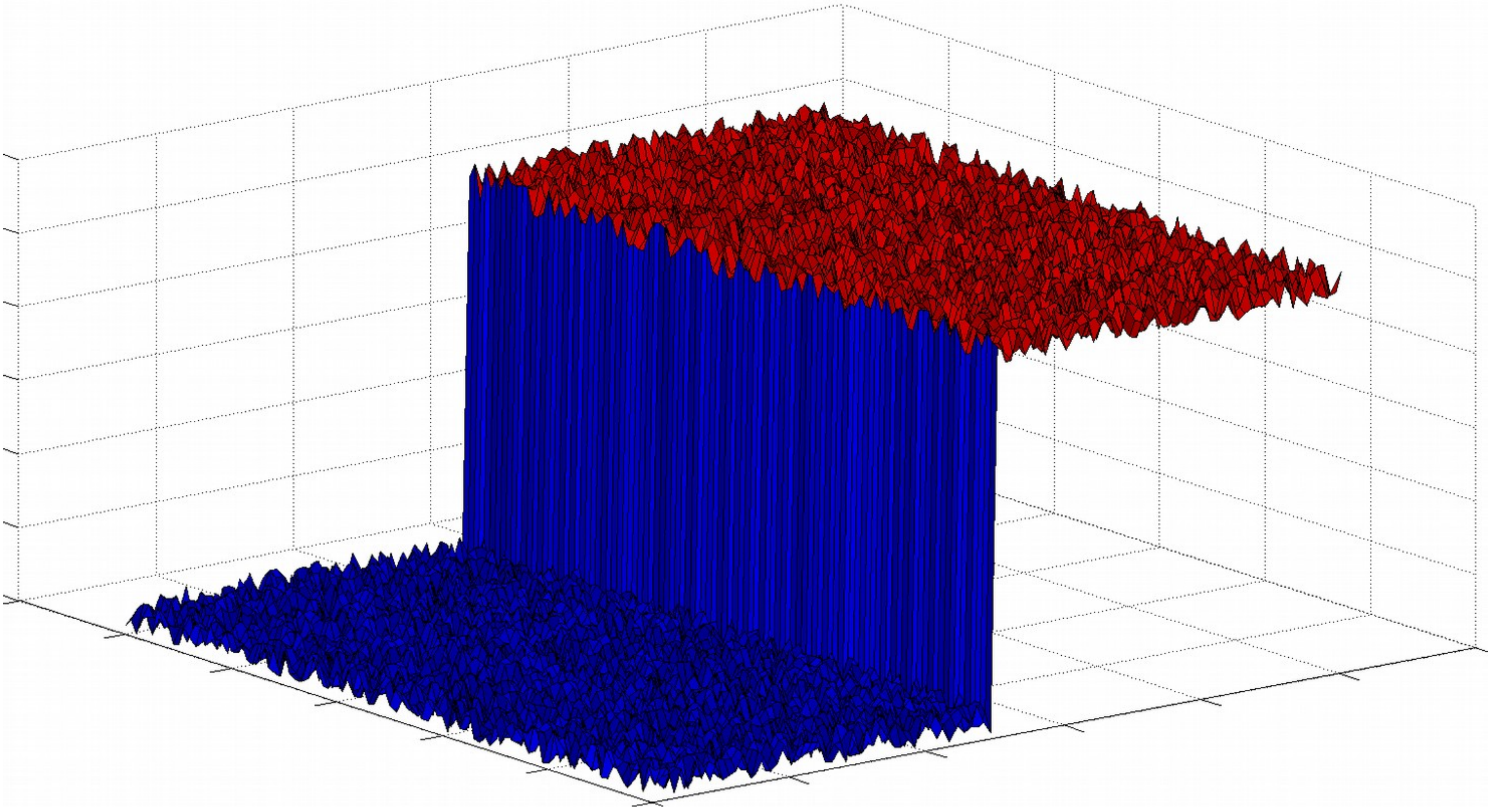
Bilateral Filter

Example: Noise Suppression by Bilateral Filter



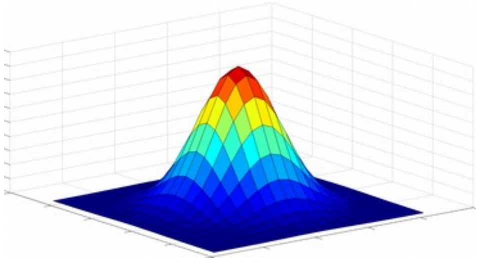
Bilateral Filter

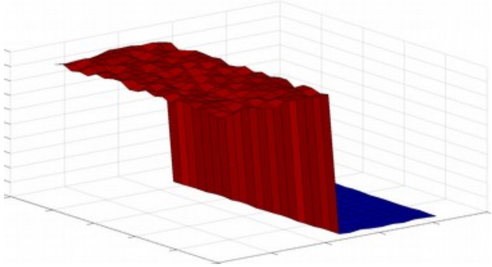
$$g(\vec{x}) = \frac{1}{Z(\vec{x})} \sum_{\vec{x}' \in \partial \vec{x}} \exp\left(-\frac{(\vec{x}' - \vec{x})^2}{2 \cdot \sigma_1^2}\right) \cdot \exp\left(-\frac{(f(\vec{x}') - f(\vec{x}))^2}{2 \cdot \sigma_2^2}\right) \cdot f(\vec{x}')$$

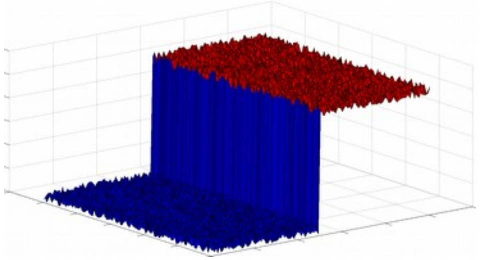


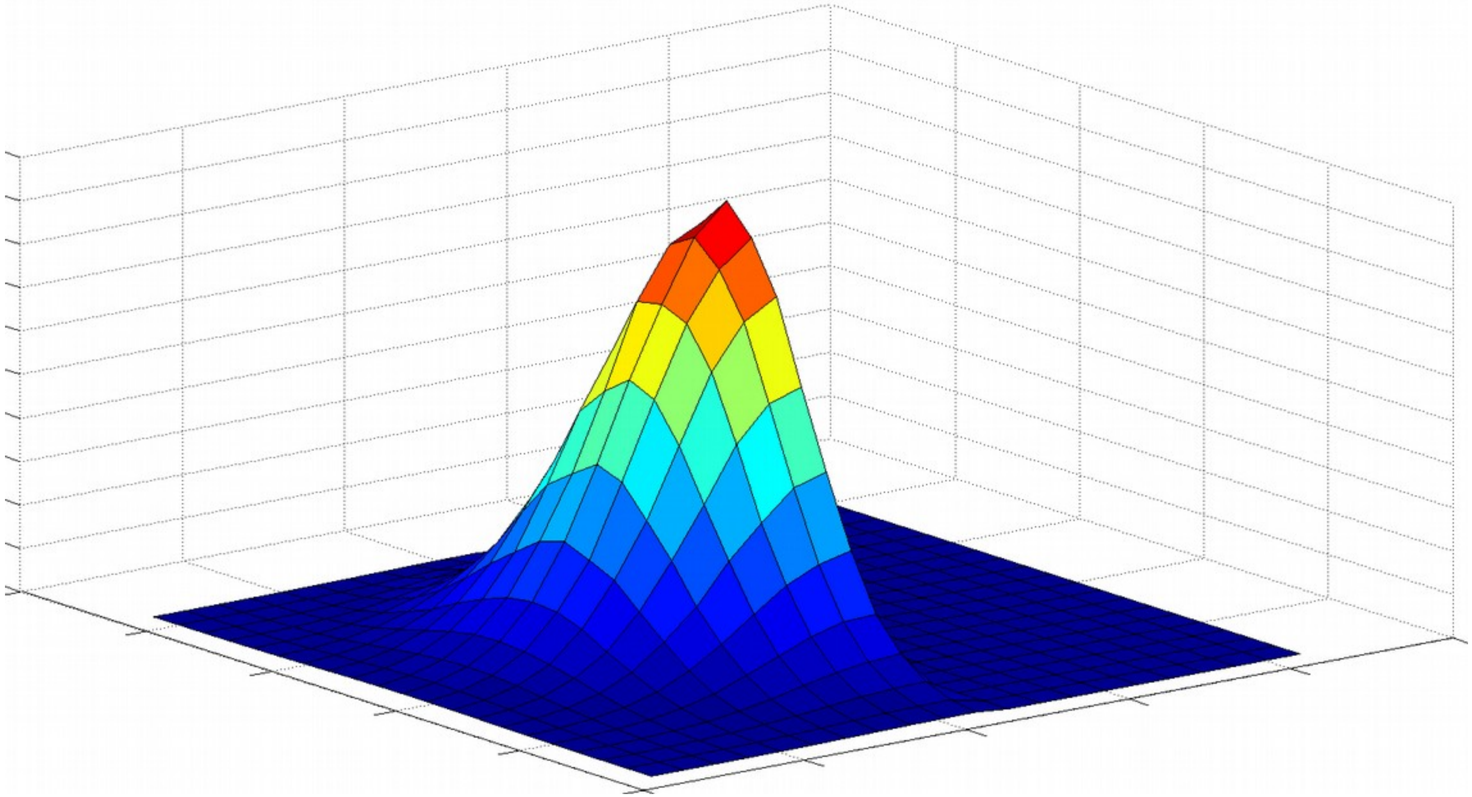
Bilateral Filter

$$g(\vec{x}) = \frac{1}{Z(\vec{x})} \sum_{\vec{x}' \in \partial \vec{x}} \exp\left(-\frac{(\vec{x}' - \vec{x})^2}{2 \cdot \sigma_1^2} \right) \cdot \exp\left(-\frac{(f(\vec{x}') - f(\vec{x}))^2}{2 \cdot \sigma_2^2} \right) \cdot f(\vec{x}')$$









Bilateral Filter

Original

Filter Result

C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images",
Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India

Bilateral Filter

Bilateral Filter

$$g(\vec{x}) = \frac{1}{Z(\vec{x})} \sum_{\vec{x}' \in \partial \vec{x}} \exp\left(-\frac{(\vec{x}' - \vec{x})^2}{2 \cdot \sigma_1^2}\right) \cdot \exp\left(-\frac{(f(\vec{x}') - f(\vec{x}))^2}{2 \cdot \sigma_2^2}\right) \cdot f(\vec{x}')$$

What happens for $\sigma_1^2 \rightarrow \infty$?

Excursus: Non-Local Means

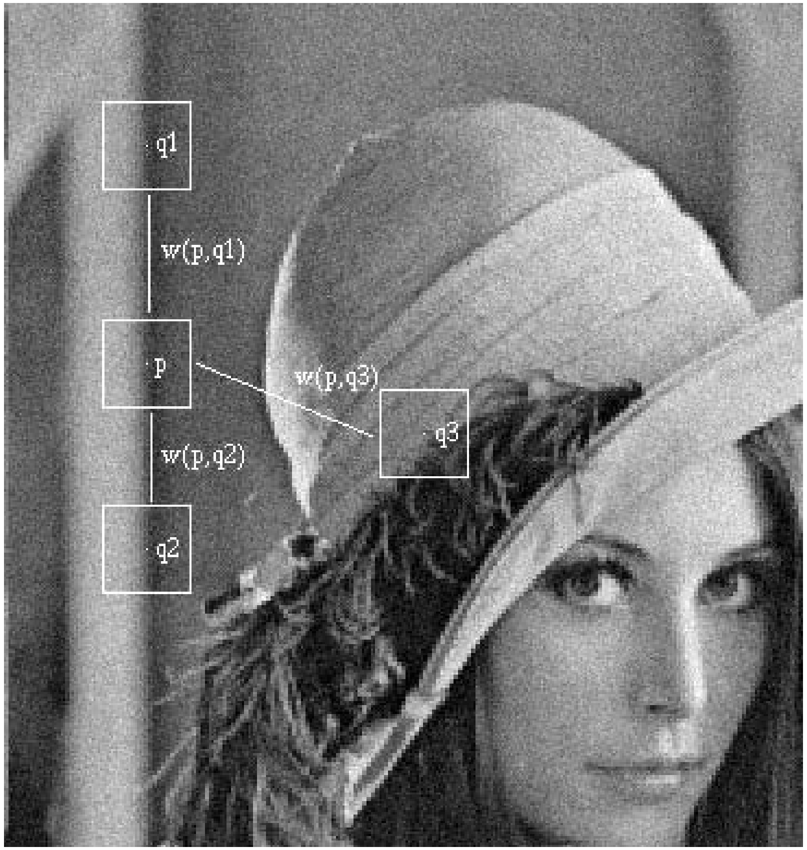
$$g(\vec{x}) = \frac{1}{Z(\vec{x})} \sum_{\vec{x}' \in \Omega} w(\vec{x}', \vec{x}) \cdot f(\vec{x}')$$

Important design questions:

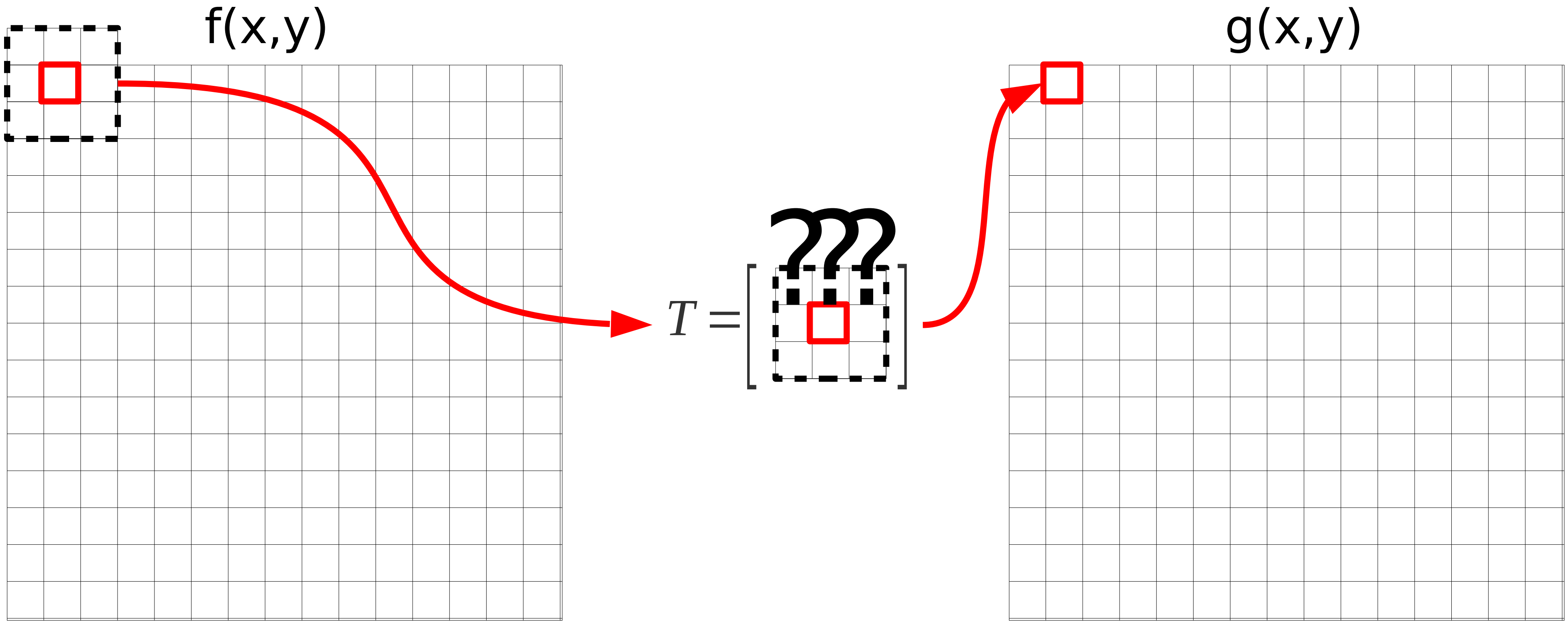
- Which search region?
  → Often restricted
  (computational complexity)

- Which pixel selection strategy?
  → Patch-based similarity measures

- Which averaging strategy?
  → Weighted average
  → Thresholding, normalizing



Border handling



- Problem: Unknown image values beyond image borders
- Possible solution
  - "Shrink" output image using only available information
  - Adapt kernel shape
  - Use "default" values (0, 255)
  - Use other image information (e.g. mirroring, wrapping)

2. Exercise – Noise Suppression

→ **Part I - Theoretical**

→ **Part II - Practical**

- Moving Average Filter
- Median Filter
- Bilateral Filtering

- Optional: Non-Local Means

2. Exercise – Theory

1. When should the median filter be applied to an image and when the moving average filter?

2. **Explain** your answer to question 1.

3. Is there a **general** better choice than the moving average filter?

4. **Explain** your answer to question 3.

2. Exercise - Given Functions

FILE: main.cpp

```
int main(int argc, char** argv)
```

- Main function
  - Usage:
    - ./main path_to_image
      - For 2 noise types:
        - Generates and saves noisy images
      - Runs multiple denoising algorithms on noisy images
        - Saves all resulting images and computes PSNR

FILE: unit_test.cpp

```
int main(int argc, char** argv)
```

- Unit tests
  - Usage: ./unit_test

2. Exercise – To Do

```
Mat spatialConvolution(Mat& src, Mat& kernel)
```

- Parameter:
  - → **`src`** : source image
  - → **`kernel`** : kernel of the convolution
  - → **`return`** : output image

- Applies convolution in spatial domain
- One method of border handling: size(src) == size(return)
- Input / output single channel (greyscale)

- Do **NOT** use convolution functions of OpenCV

2. Exercise – To Do

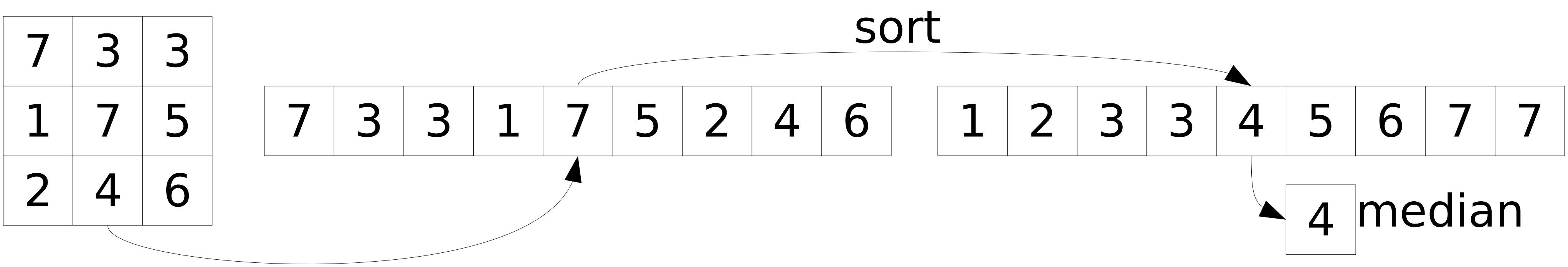
`Mat averageFilter(Mat& src, int kSize)`

- Parameter:
  - → **`src`** : noisy source image
  - → **`kSize`** : Kernel size
  - → **`return`** : output image
- Uses convolution to calculate local average
- Calls spatialConvolution(...)

`Mat medianFilter(Mat& src, int kSize)`

- Parameter:
  - → **`src`** : noisy source image
  - → **`kSize`** : Kernel size
  - → **`return`** : output image
- Applies local median filtering

2. Exercise - Median




```
#include <iostream>
#include <algorithm>

int main() {
  std::vector<int> array = { 23, 5, -10, 0, 0, 321, 1, 2, 99, 30 };
  std::sort(array.begin(), array.end());
  for (int i : elements)
     std::cout << i << ' ';
}
```

2. Exercise – To Do

```
Mat bilateralFilter(Mat& src, int kSize,
                    float sigma_spatial, float sigma_radiometric)
```

- Parameter:
  - **src** : noisy source image
  - **kSize** : size of spatial kernel
  - **sigma_spatial** : Std-dev of spatial kernel
  - **sigma_radioametric** : Std-dev of radiometric kernel
  - **return** : output image
- Implements bilateral filter

2. Exercise – To Do

```
enum NoiseType {
    NOISE_TYPE_1,
    NOISE_TYPE_2,
    NUM_NOISE_TYPES
};
```

```
enum NoiseReductionAlgorithm {
    NR_MOVING_AVERAGE_FILTER,
    NR_MEDIAN_FILTER,
    NR_BILATERAL_FILTER,
    NUM_FILTERS
};
```

```
Mat denoiseImage(Mat &src, NoiseType noiseType,
                 NoiseReductionAlgorithm noiseReductionAlgorithm)
```

- Parameter:
  - **src** : noisy source image
  - **noiseType** : type of noise in this image
  - **noiseReductionAlgorithm** : Which algorithm to use

- Denoises an image with the implemented filters
- Choose reasonable parameters (kernel size etc.) here

2. Exercise – To Do

```
enum NoiseType {
    NOISE_TYPE_1,
    NOISE_TYPE_2,
    NUM_NOISE_TYPES
};
```

```
enum NoiseReductionAlgorithm {
    NR_MOVING_AVERAGE_FILTER,
    NR_MEDIAN_FILTER,
    NR_BILATERAL_FILTER,
    NUM_FILTERS
};
```

```
NoiseReductionAlgorithm chooseBestAlgorithm(NoiseType noiseType)
```

- Which reduction algorithm is best (in general) for the type of noise?
- Look at the noisy images and deduce the noise type

2. Exercise – Optional

```
Mat nlmFilter(Mat& src, int searchSize, double sigma)
```

- Parameter:
  - **`src`** : noisy source image
  - **`searchSize`** : size of search region
  - **`sigma`** : Optional parameter for weighting function
  - **`return`** : output image

- Implements non-local means filter (optional)

Reminder

Deadline: 12th of November

But: We meet again in 1 week
(5th of November, H1028)