Visualization, DD2257

Prof. Dr. Tino Weinkauf

Data Filtering

Trend: Large-Scale Volumetric Data

Biological Applications, e.g., Ultra-Thin Serial Section Electron Microscopy 1mm^3 of brain tissue amounts to 20,000 slices à 40 gigapixel, total of **800 TB** of image data

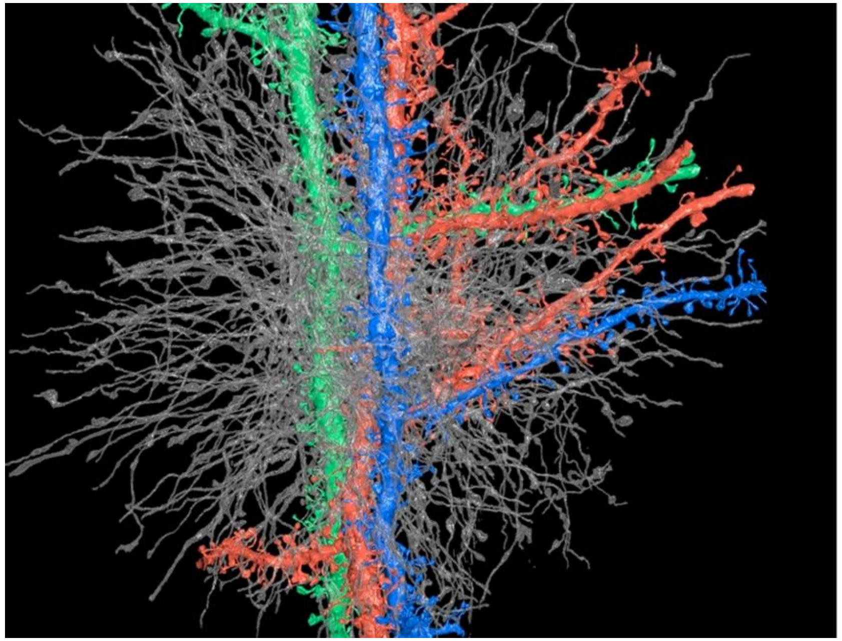
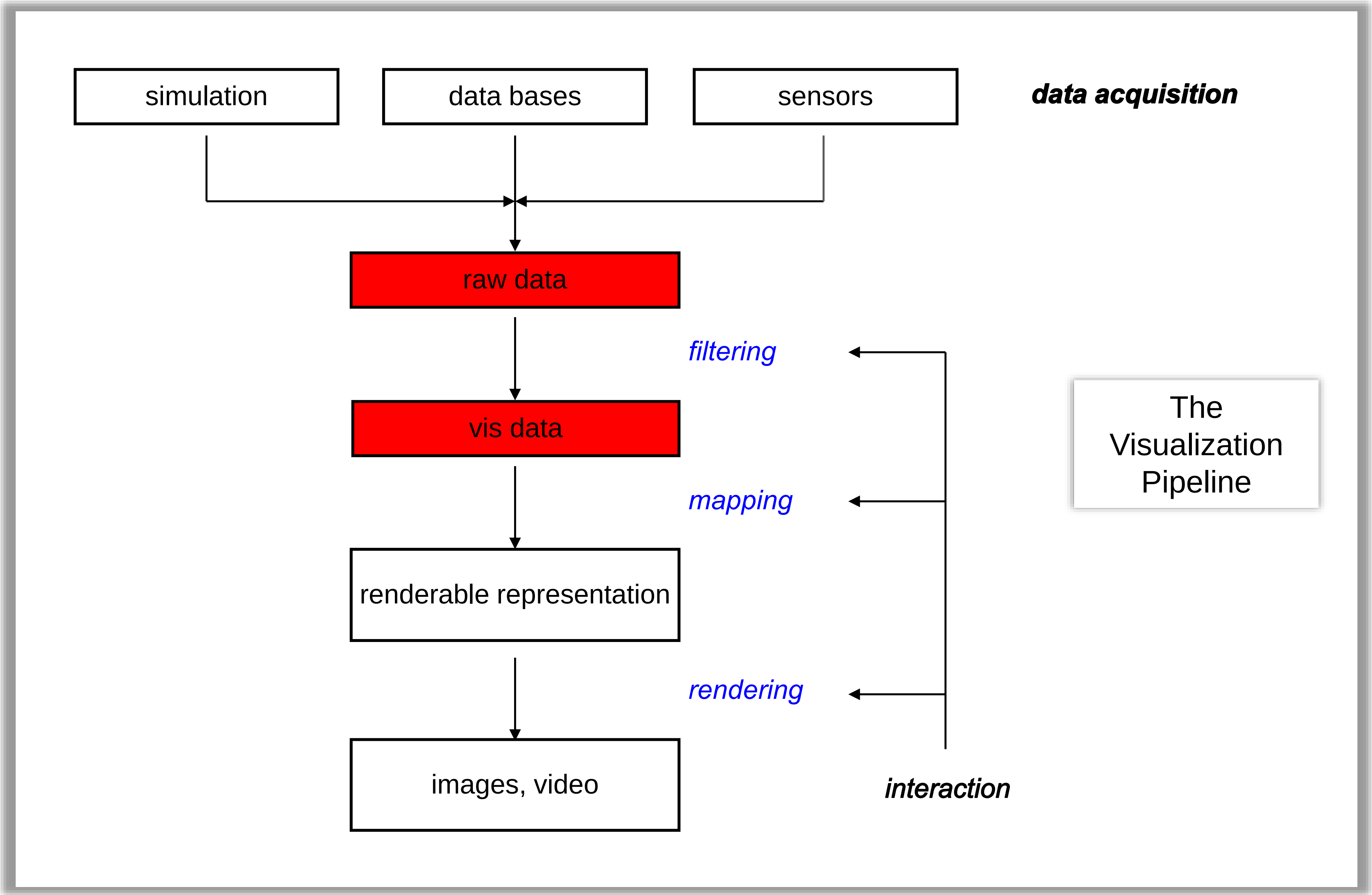
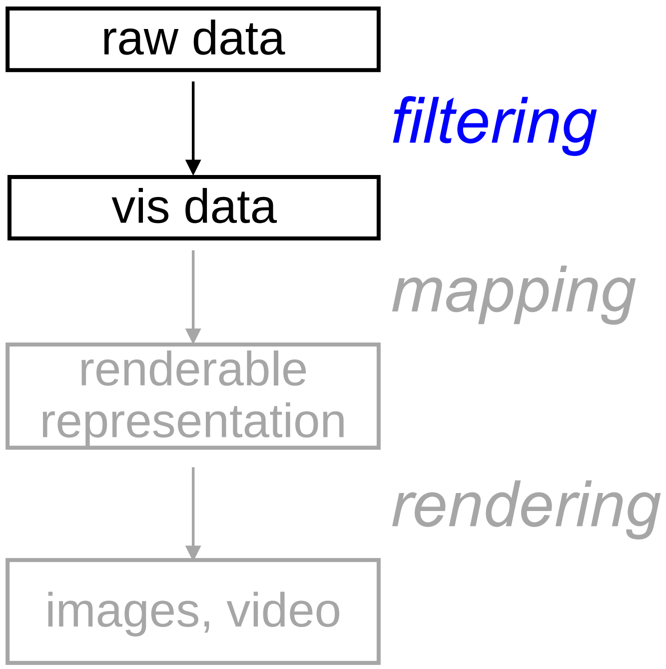


Image Source: Pfister et al., Visualization in Connectomics, arXiv 2012


simulation
data bases
sensors
data acquisition
raw data
filtering
vis data
mapping
renderable representation
rendering
images, video
interaction
The Visualization Pipeline

Operations for filtering:

- **completing/cleaning the data set**
  - if data values are missing
  - if data values are outliers
- **reduce data set**
  - remove non-relevant data by certain criteria
- **smoothing data**
  - apply filter and smoothing operators
- **compute characteristic properties of the data**
  - gradients
  - extreme values
  - metadata
- **apply conversions**
  - imperial ➔ metric
  - customary ➔ metric



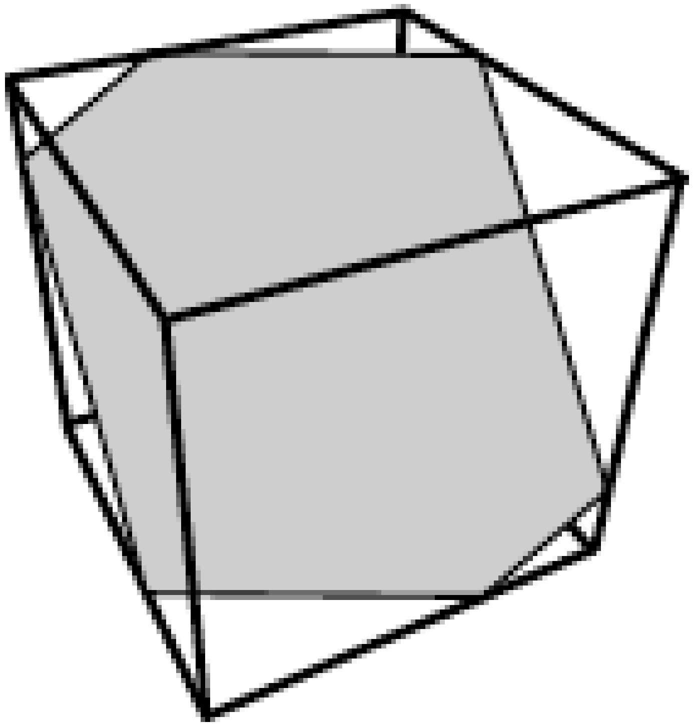
Filtering by Selection

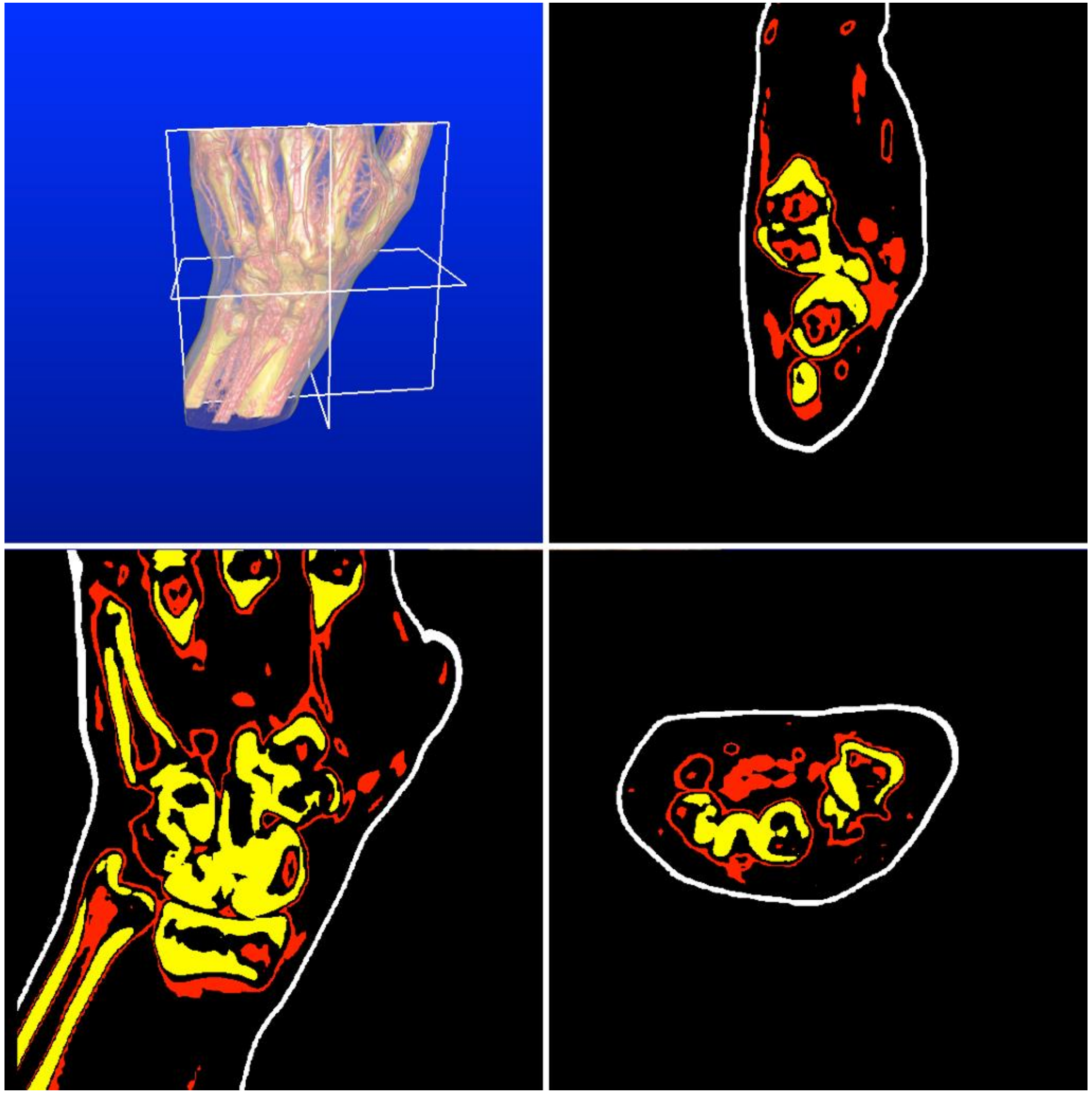
select parts of the data (domain / dependent variables)

- select variables
- sub-volume / sub-area
- slicing
- predicates

Slicing

Example: 2D cutting surface (slice) through a 3D volume





Predicates

Selection of data according to logical conditions (predicates)

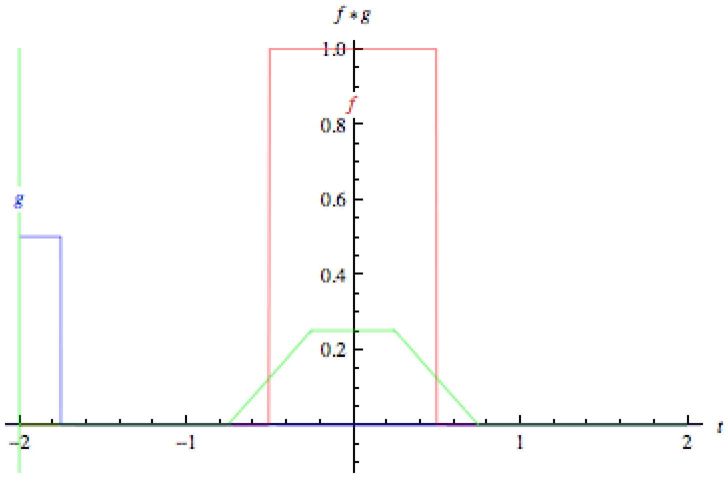
Example:

- 2D height field: (x, y, h)
- $\sigma = \{(x, y, h) \mid (x^2 + y^2 < 5\mathrm{km}) \wedge (h > 1\mathrm{km})\}$

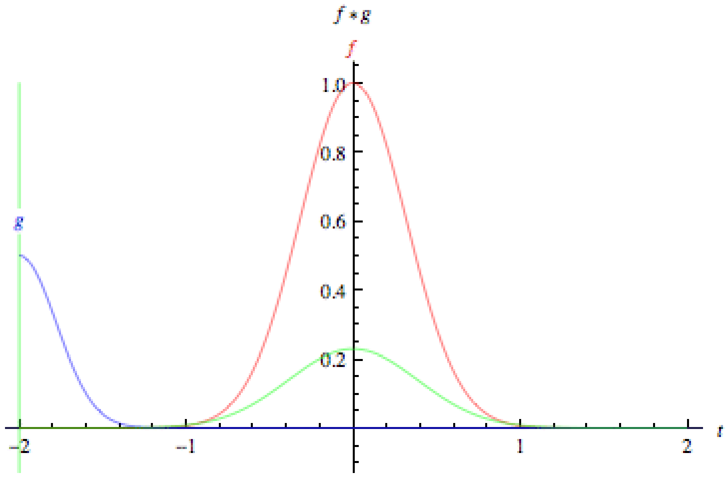
Filtering by Convolution

”Sliding a function g(x) along a function f(x)“

$$s(x) = (f * g)(x) = \int_{-\infty}^{\infty} f(u)\, g(x-u)\, \mathrm{d}u$$



Box



Gauß

http://mathworld.wolfram.com/Convolution.html

- For 2D functions f and g, their convolution is defined by:

$$s(x,y) = (f * g)(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)\, g(x-u, y-v)\,\mathrm{d}u\,\mathrm{d}v$$

- Discrete form:

$$s[m,n] = \sum_{i}\sum_{j} f[i,j]\, g[m-i, n-j]$$

Convolution in **Image Processing**

- Filtering image f using the filter kernel g
  - “Sliding the kernel g over the image f “
- Mainly applied to remove or enhance features
  - Smoothing, noise removal, edge detection ...
- Typically, filter kernel is smaller than image!

Mean Filter / Box Filter (averaging)

- Reducing the intensity variation in the data
- Replace each data value by the average of the neighbors

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Original

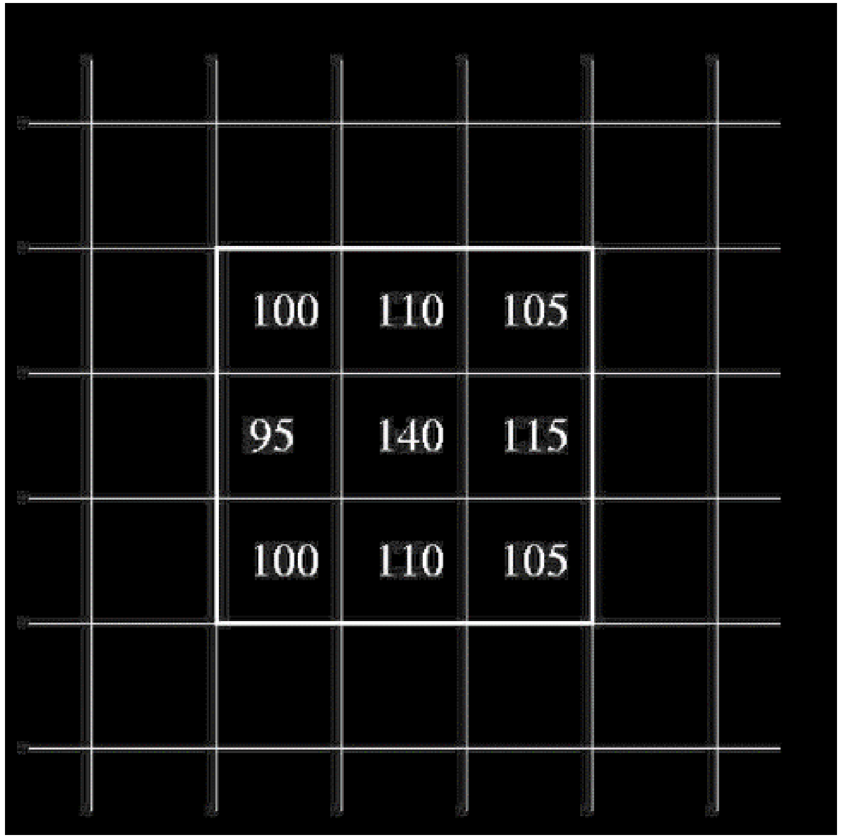
Box-Filtered

Median Filter
(not a convolution!)

- Reducing the intensity variation in the data
- Replace each value by median of neighbors
  - More robust than mean, does not create new values

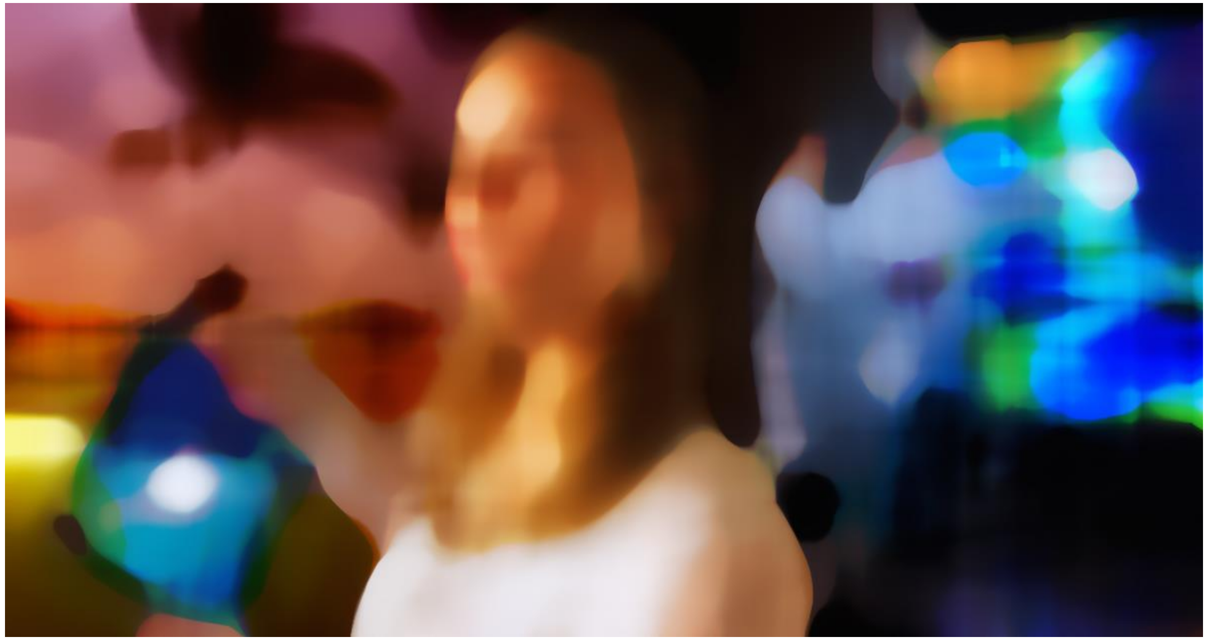
Values: 95,100,100,105,105,110,110,115,140

Median: 105



Original

Median-Filtered

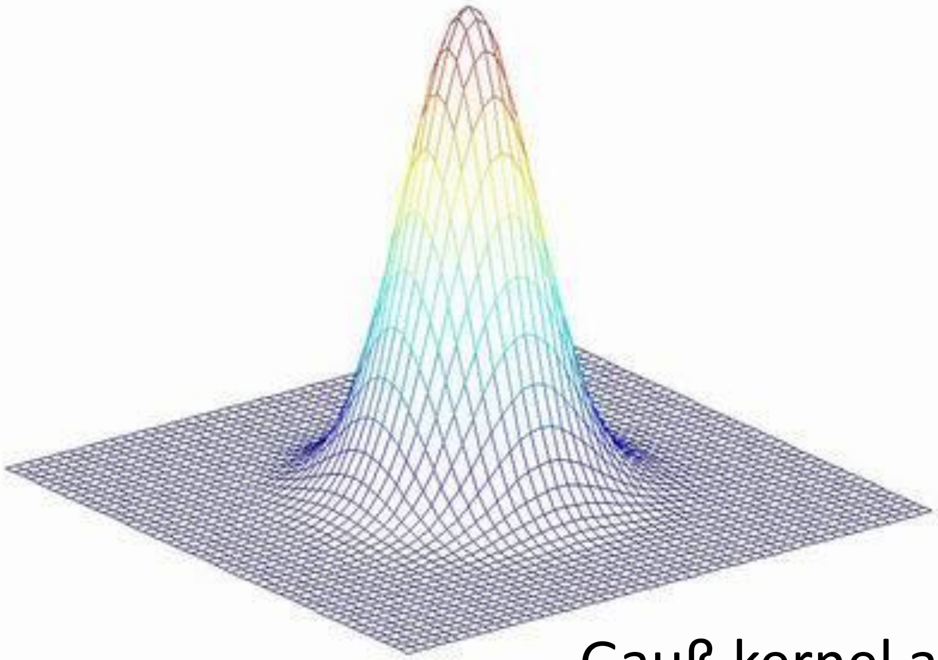


“Salt-and-Pepper Noise”

median-filtered

Gaussian Smoothing

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



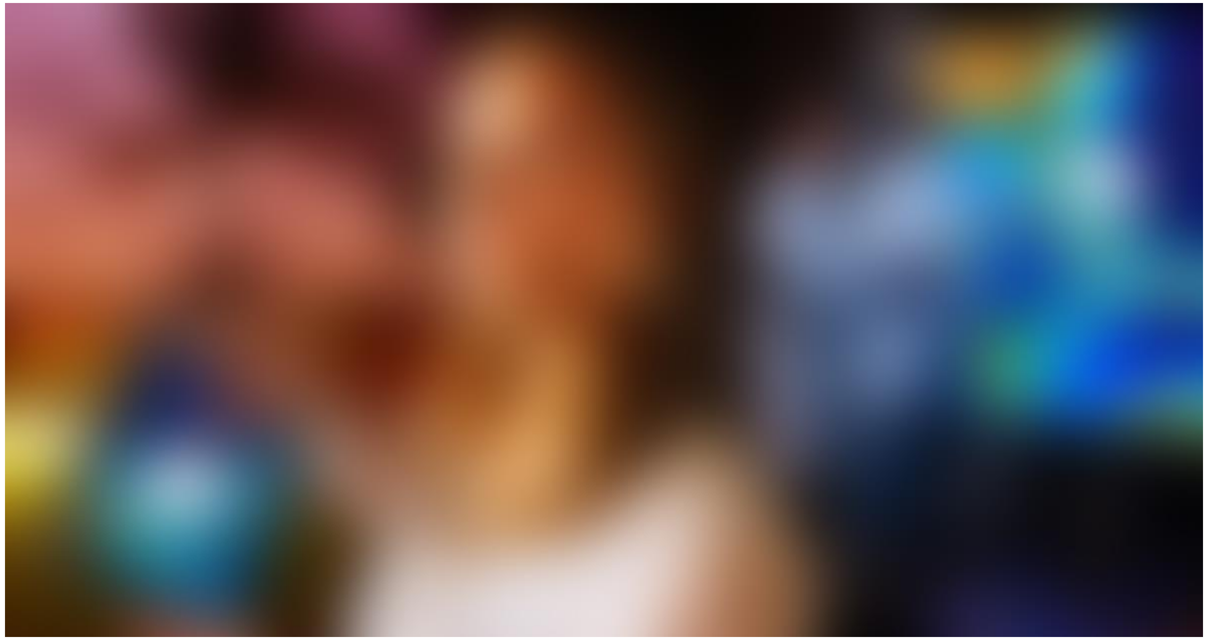
1/115

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Gauß kernel and its discrete approximation

Original

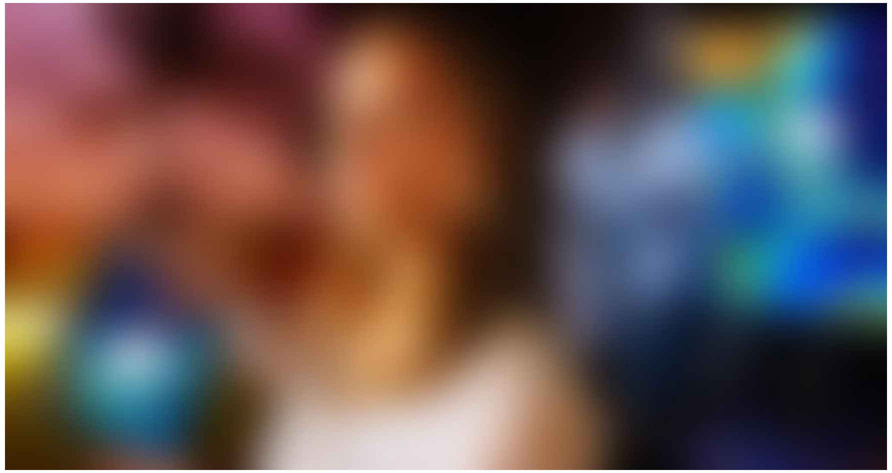
Gauß-Filtered



Original

Box

Gauß

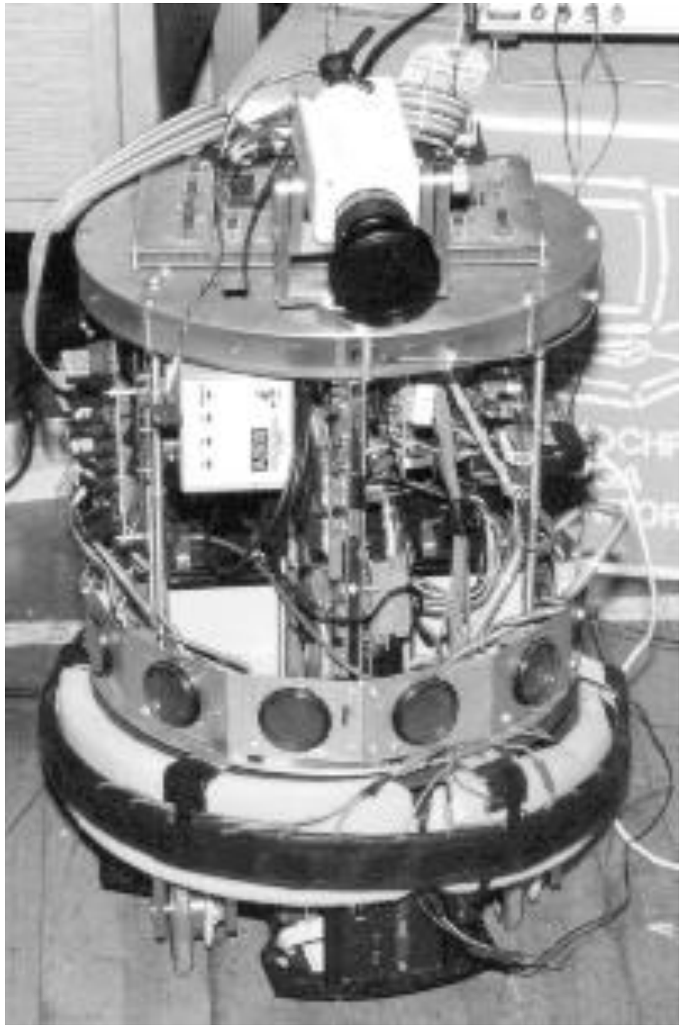


Median

2D Box & Gauß filters can be separated into two 1D filters

- First: perform 1D convolution along the x-axis
- Second: perform 1D convolution of intermediate result along the y-axis
- Much faster:
  Two times a 1D kernel of size 5
  instead of a 2D kernel of size 25

Original



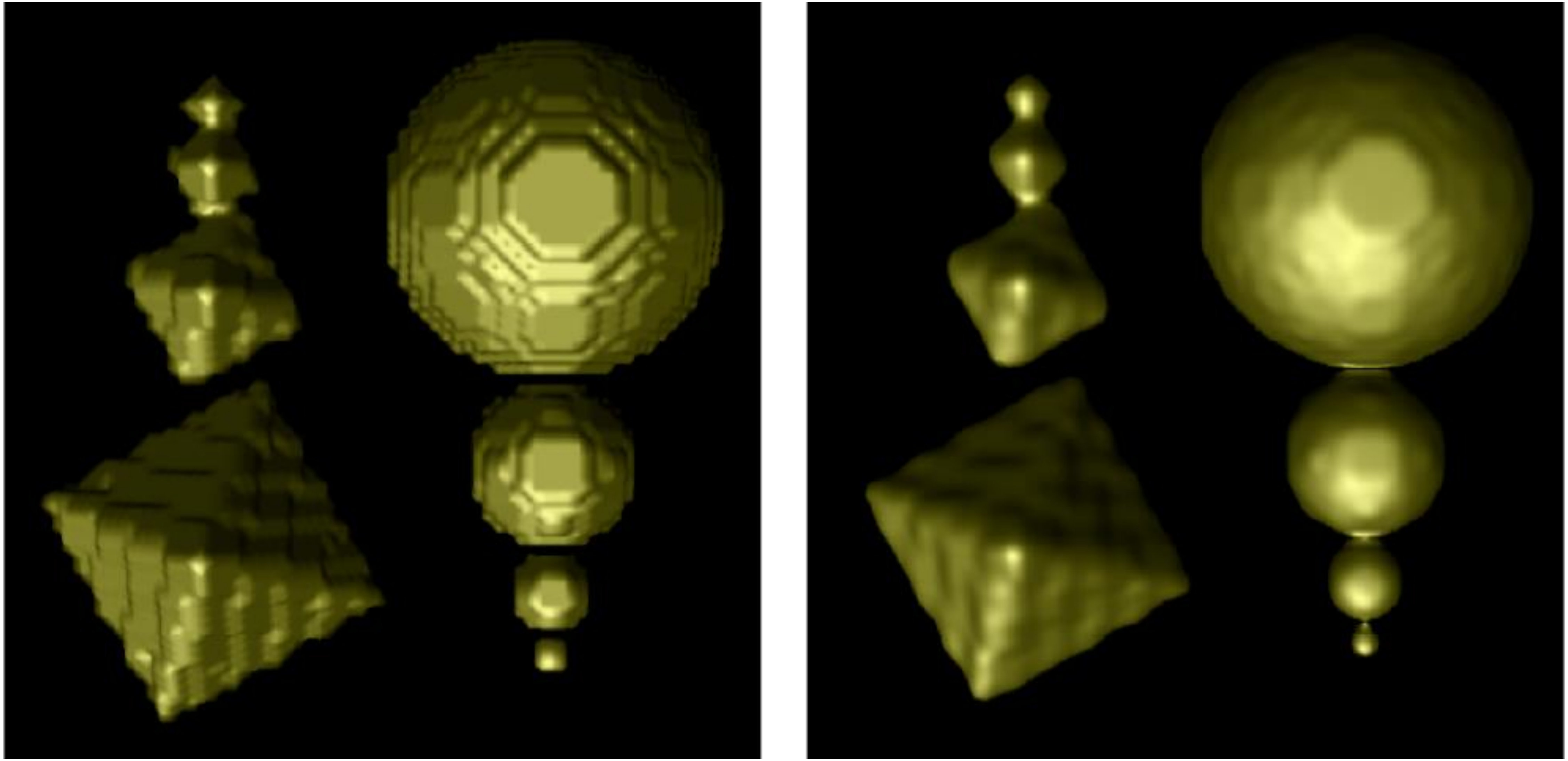
σ=1

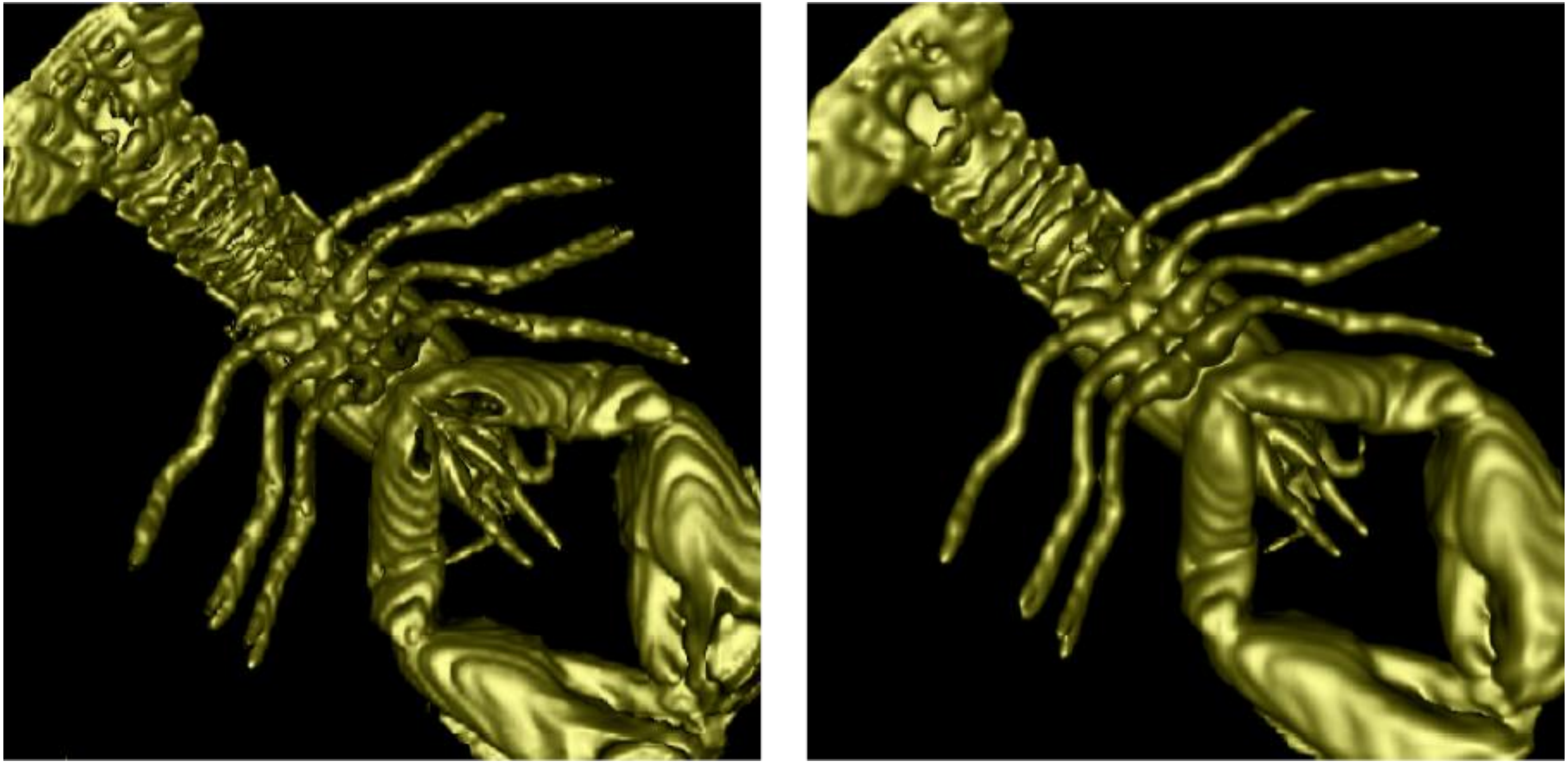
σ=4

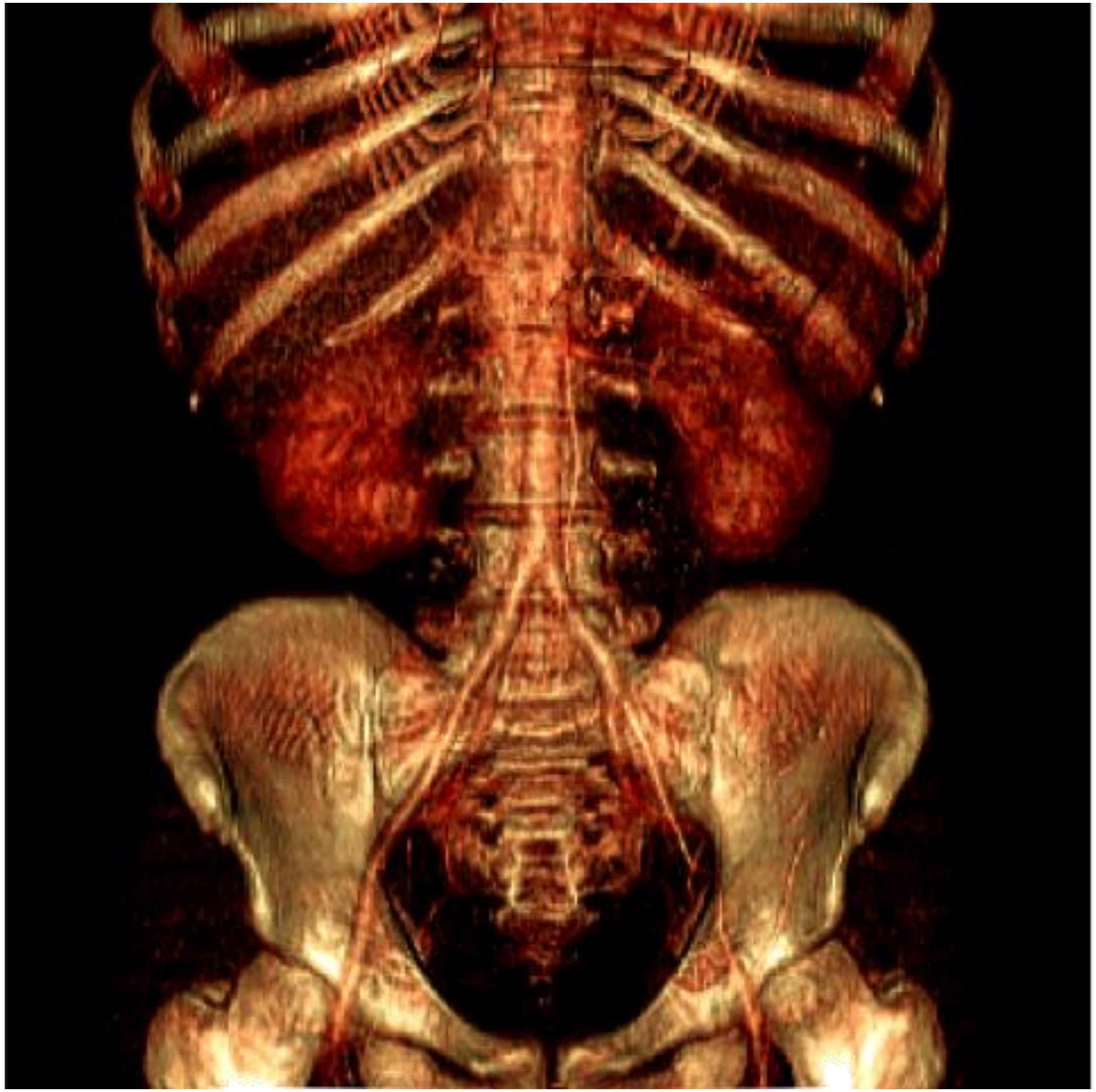

Original

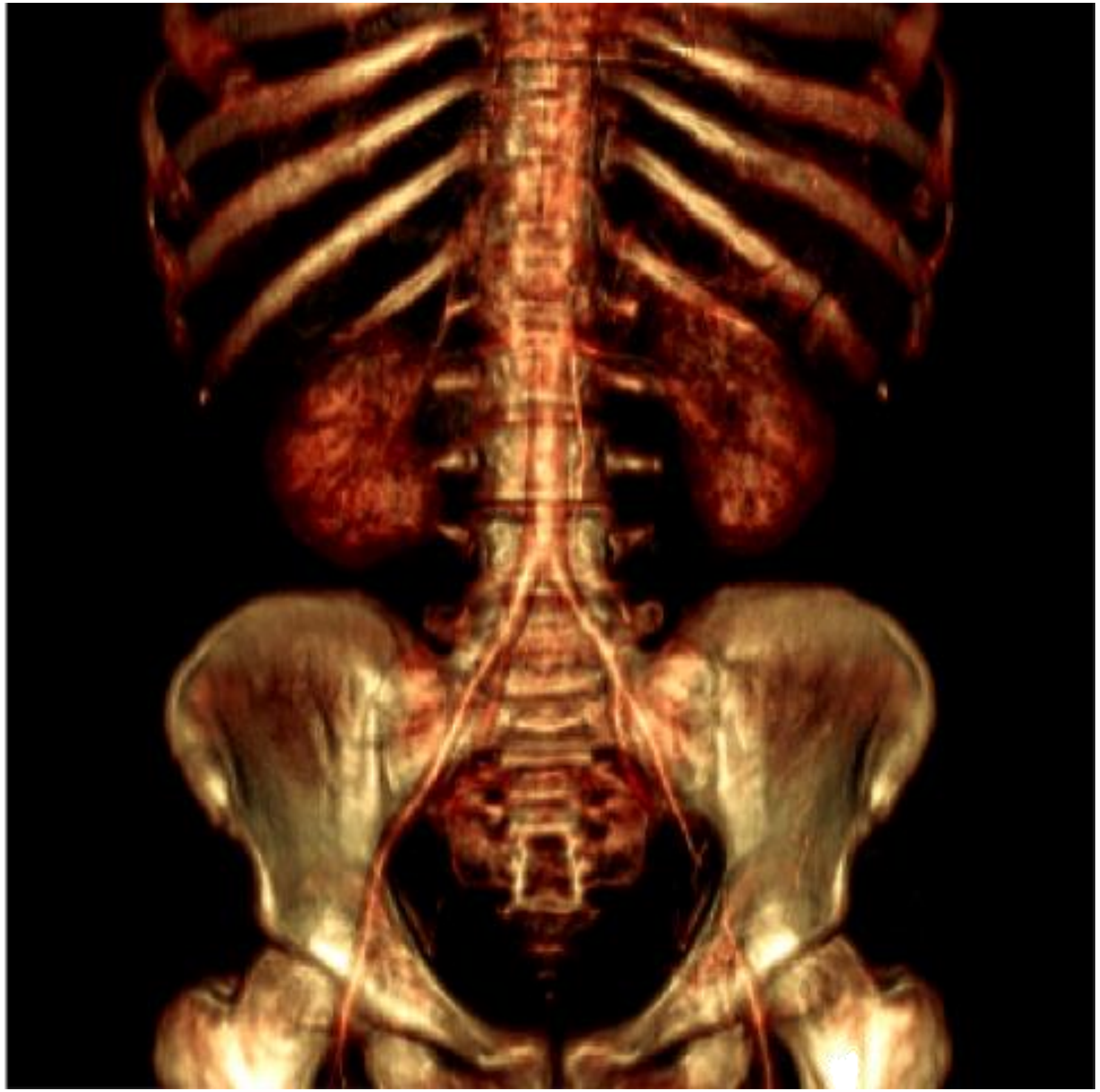

noisy


Gaussian smoothing









Sobel Filter

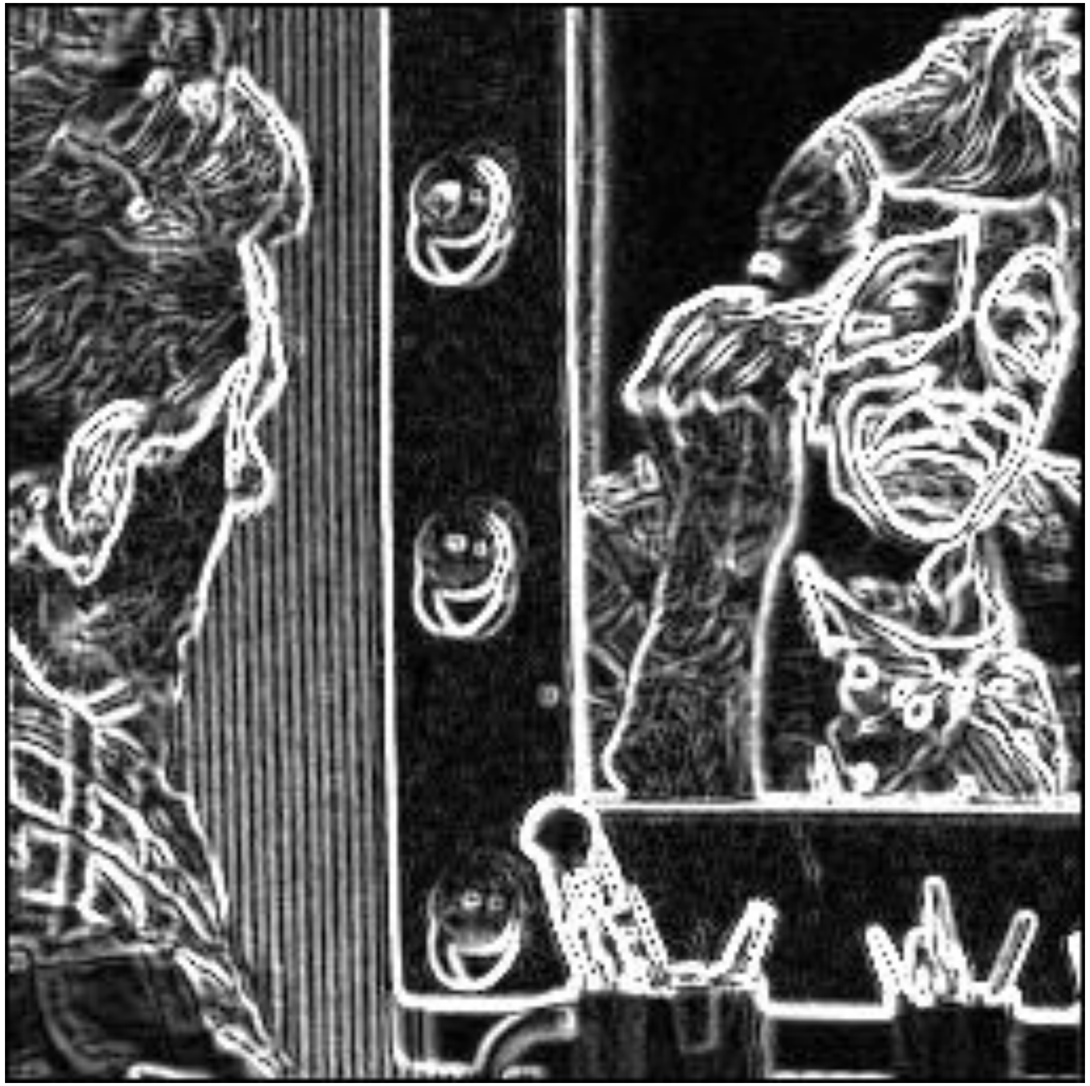
- Edge detector, derivative filter:
  - Measures the gradient thus enhancing regions of high spatial gradients like edges.
  - Consists of two pairs of filter kernels to be applied to the original image:
    $G_x = I * g_x$ and $G_y = I * g_y$
  - The final result could be the gradient magnitude: $G = \sqrt{{G_x}^2 + {G_y}^2}$

-1	0	1
-2	0	2
-1	0	1

g_x

1	2	1
0	0	0
-1	-2	-1

g_y



Canny Edge Detector

1. Smooth the image using a Gauss filter
   - Suppresses noise

2. Filter the image using a derivative filter
   - Sobel Operator

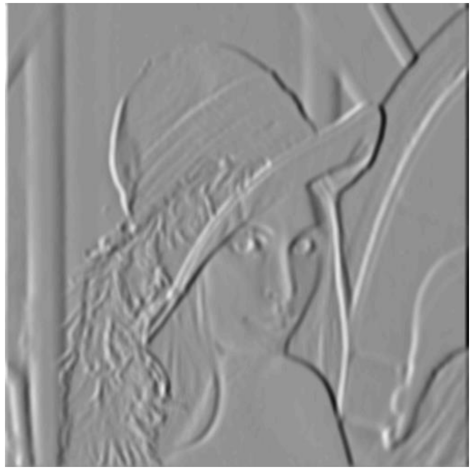
3. Determine gradient vector
   ➔ partial derivatives in x and y direction

4. Find „important“ edges by thresholding the gradient magnitudes (keep all above threshold)

5. Trace along edges (orthogonal to the gradient)
   and suppress non-edge pixels

smoothed

vertical edges



G_x

horizontal edges

G_y

norm of the gradient

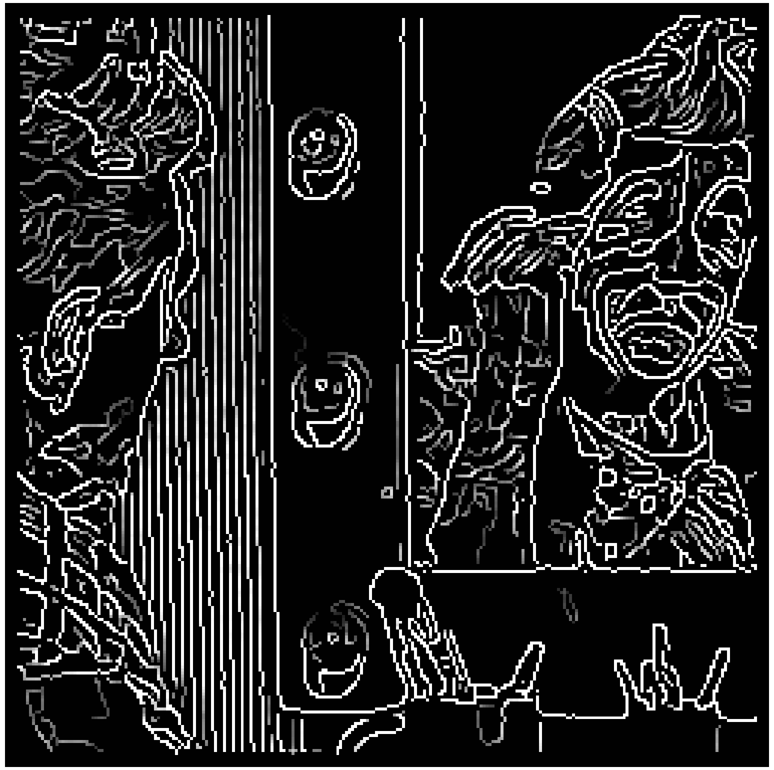
G

thresholding

thinning

- Canny edge detector

σ=1



σ=2

- Laplace of a Gaussian
  - Laplace of a 2D signal : f

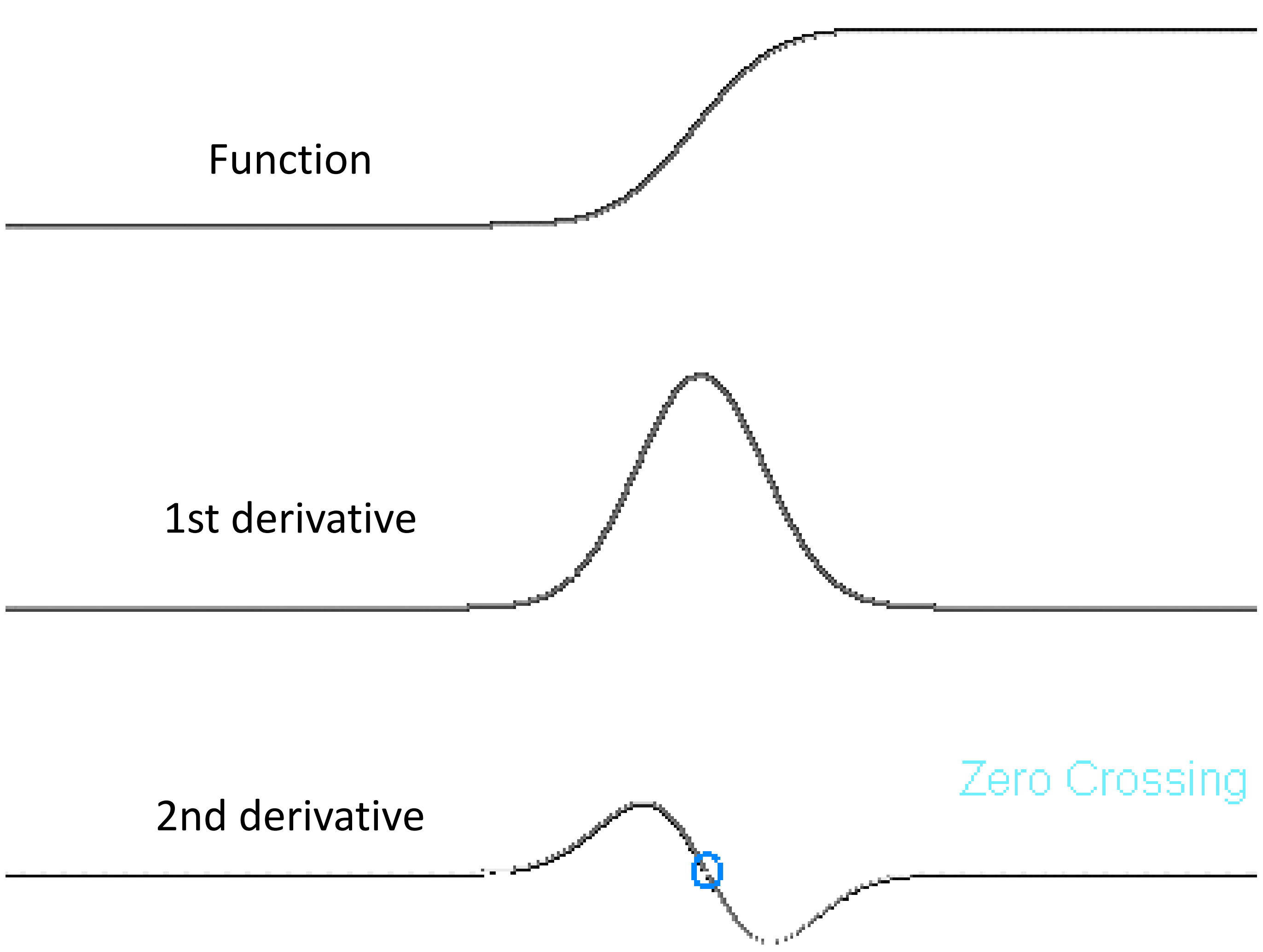
$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

    - “Deviation from average of neighbors”

  - Estimating derivatives by central differences yields:

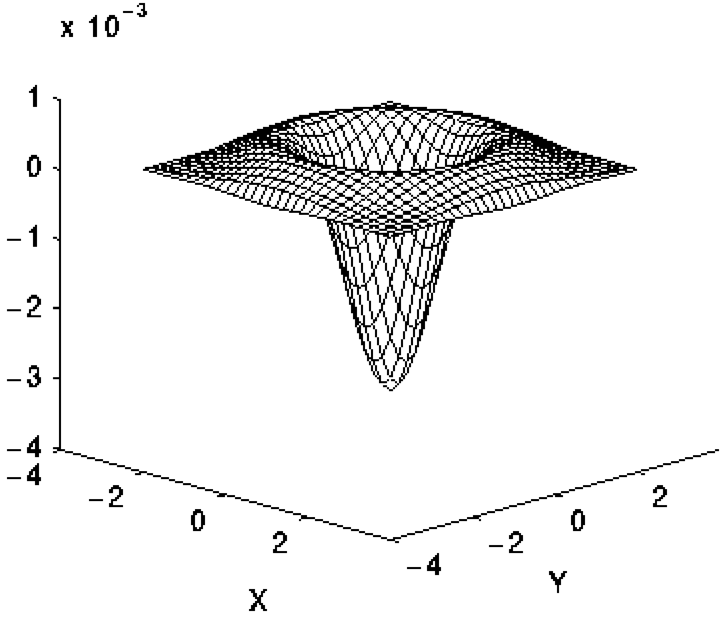
$$\Delta f \approx f[i+2] - 2f[i] + f[i-2]$$

  - Since second derivative measurements are very sensitive to noise, Gaussian smoothing is used in a first step


Function
1st derivative
2nd derivative
Zero Crossing

- **Laplace of a Gaussian (LoG)** Filter

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$



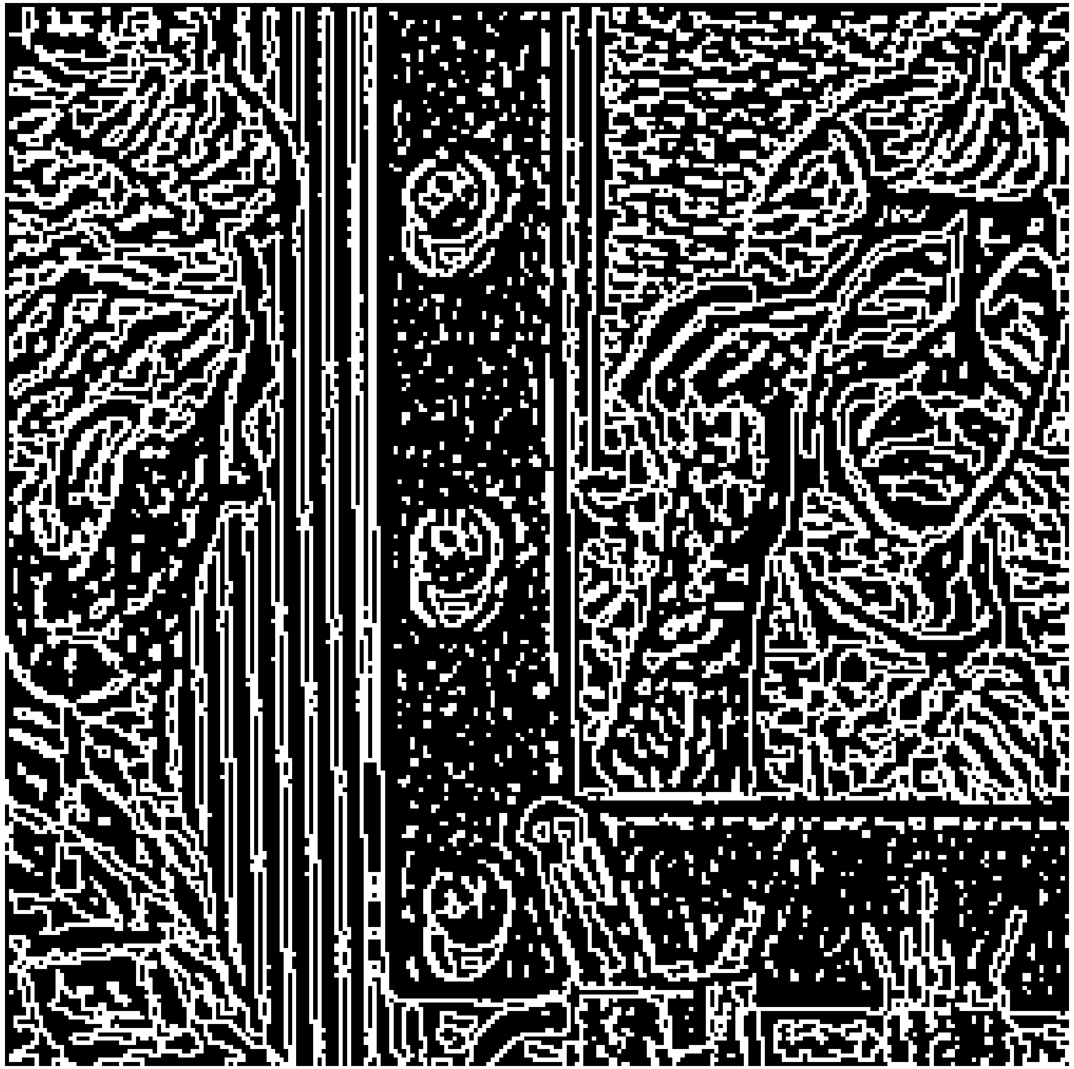
0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

LoG function and its discrete approximation

- LoG (σ=1)

LoG

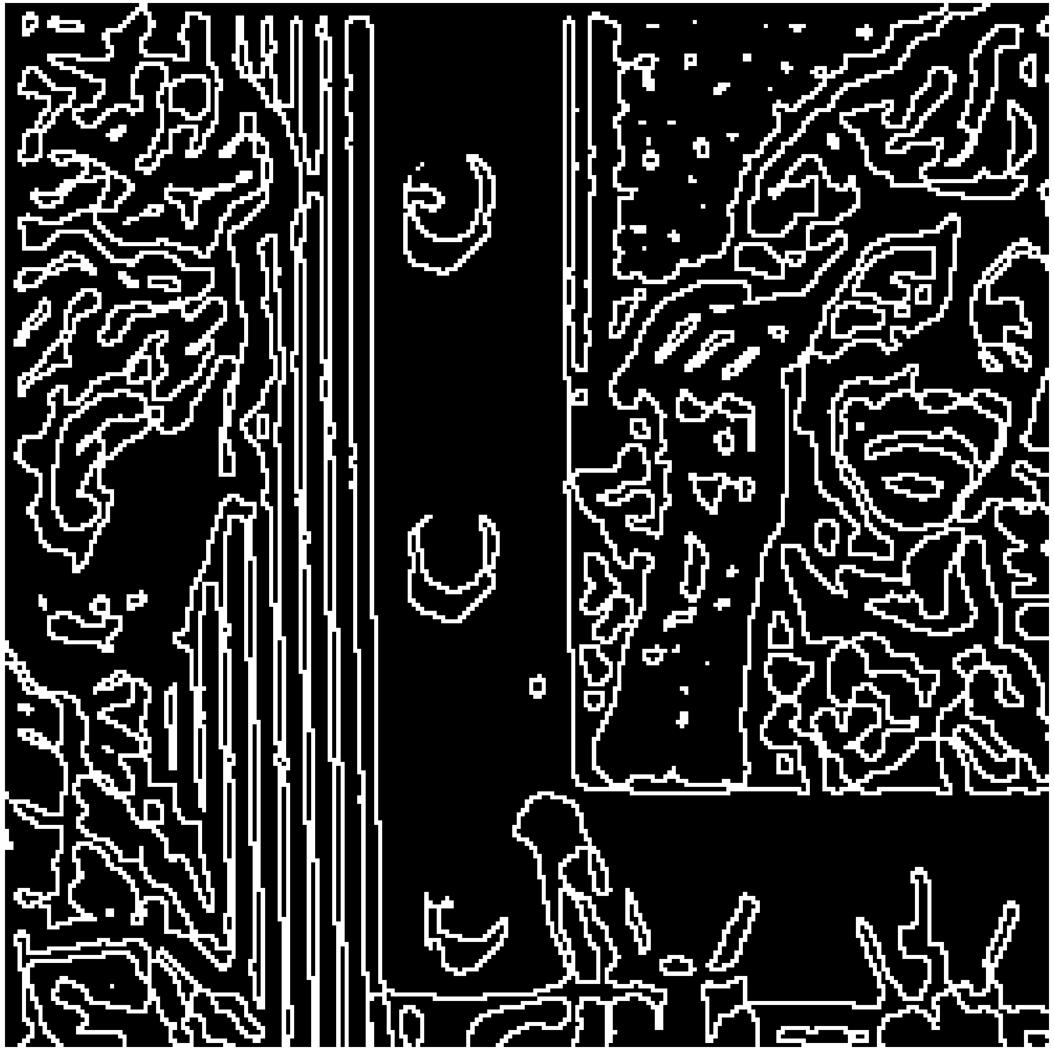
zero crossings

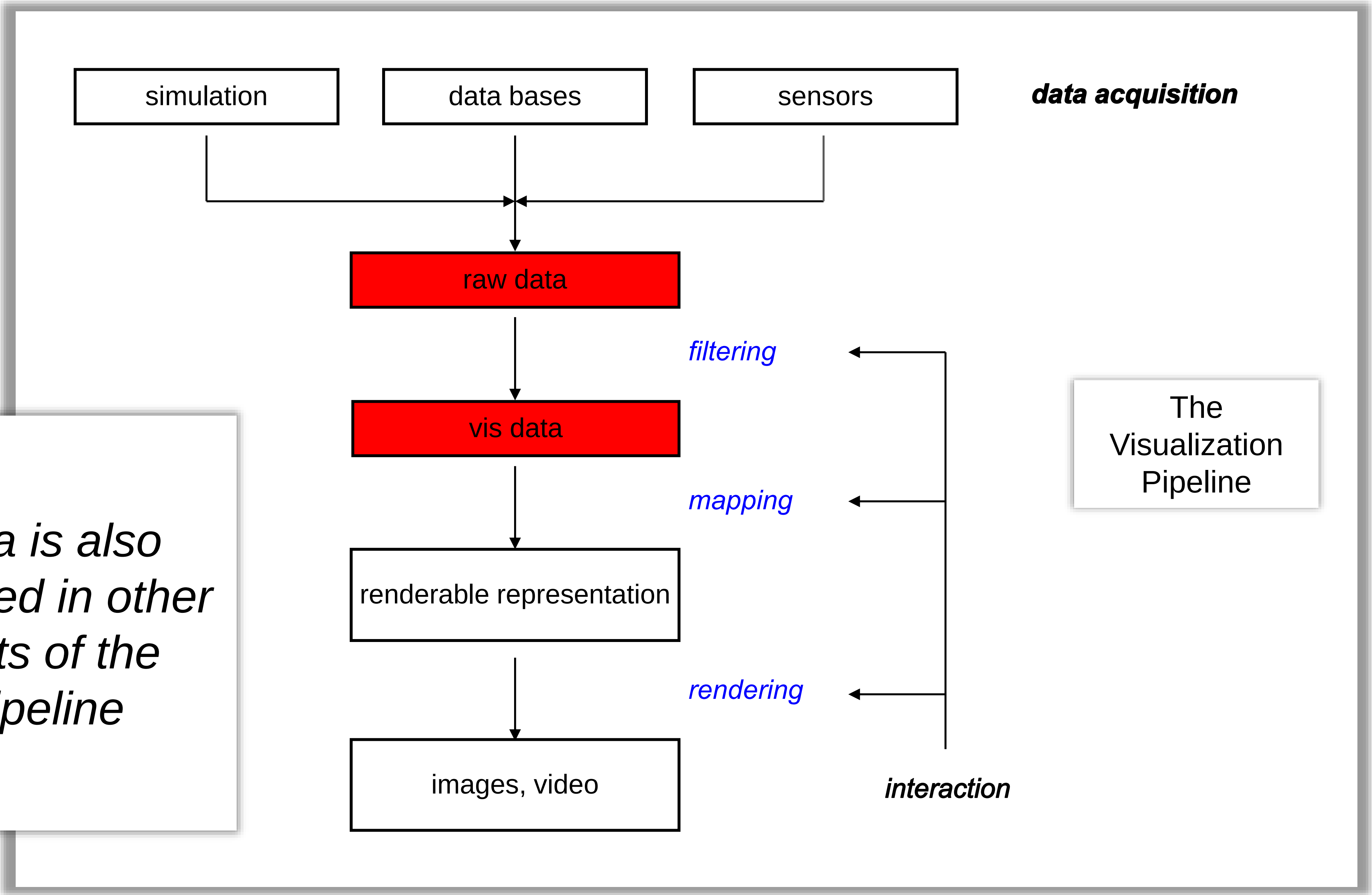


- LoG (σ=2)

LoG

zero crossings



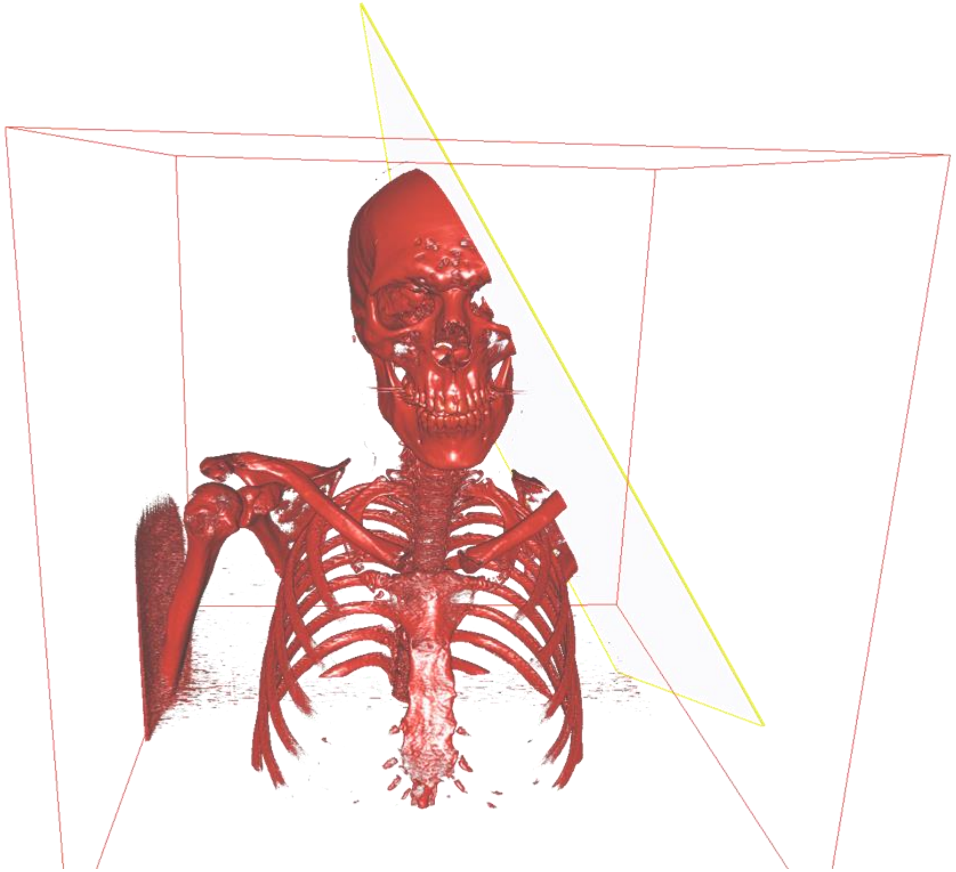


data is also reduced in other parts of the pipeline

Clipping

Remove visual content from one side of a plane

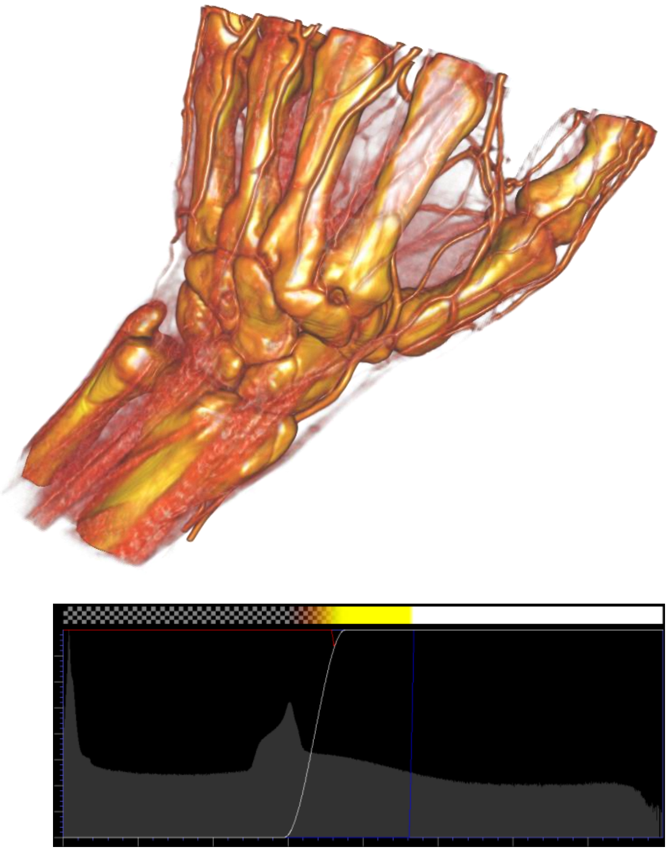
before clipping

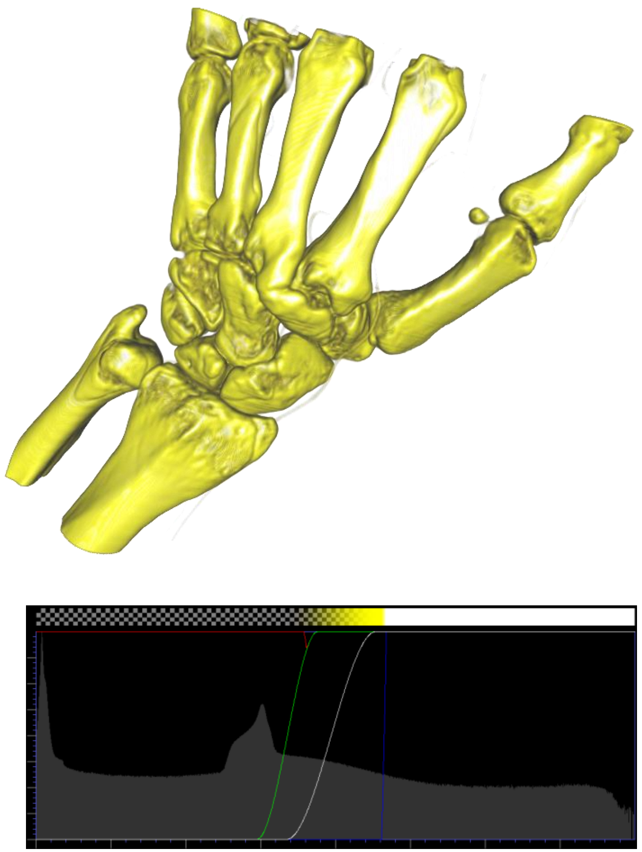


clipping using a plane

after clipping

- **Transfer Function**





Remove a certain range of data values

Summary

- Data Filtering as part of the visualization pipeline
  - data to data map
- Data reduction through selection
- Noise reduction through convolution
- Feature enhancement through convolution