Visualization
– Surface Visualization

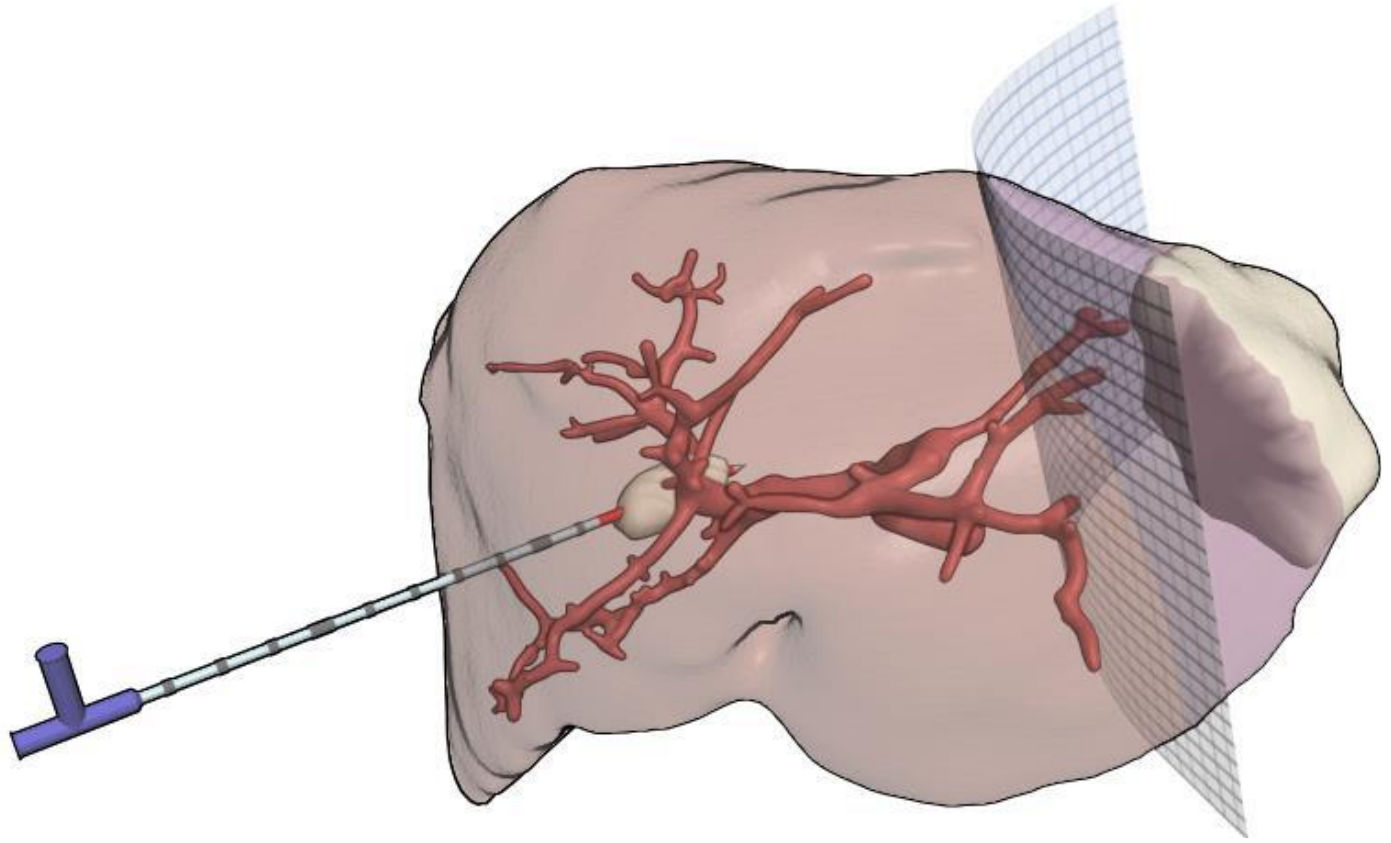
J.-Prof. Dr. habil. Kai Lawonn

Outline

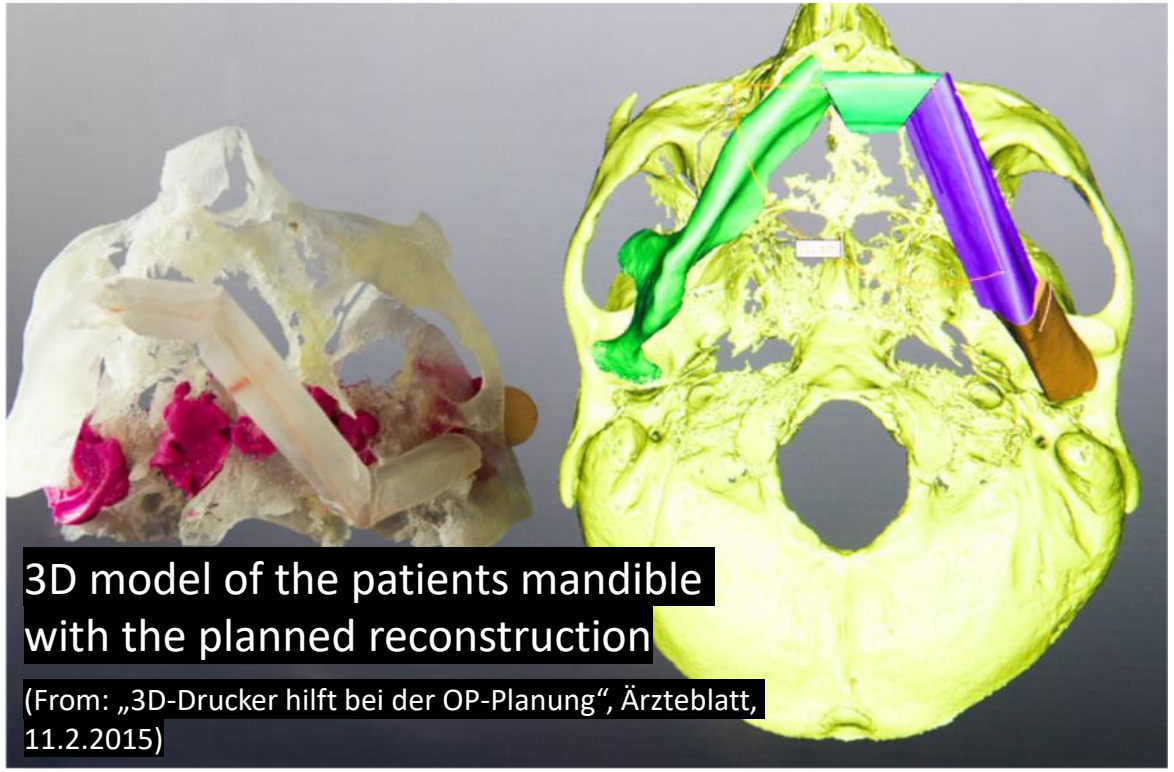
- Cuberille
- From contours in layers to surfaces
- Marching Cubes and its improvements
- Smoothing of polygonal surfaces
- Context-aware smoothing
  - Distance-aware smoothing
  - Staircase-aware smoothing
- Quad-meshes

Motivation

- Surface models for:
  - Diagnosis,
  - Intervention planning,
  - Medical education,
  - Biomedical simulations,
  - Rapid prototyping

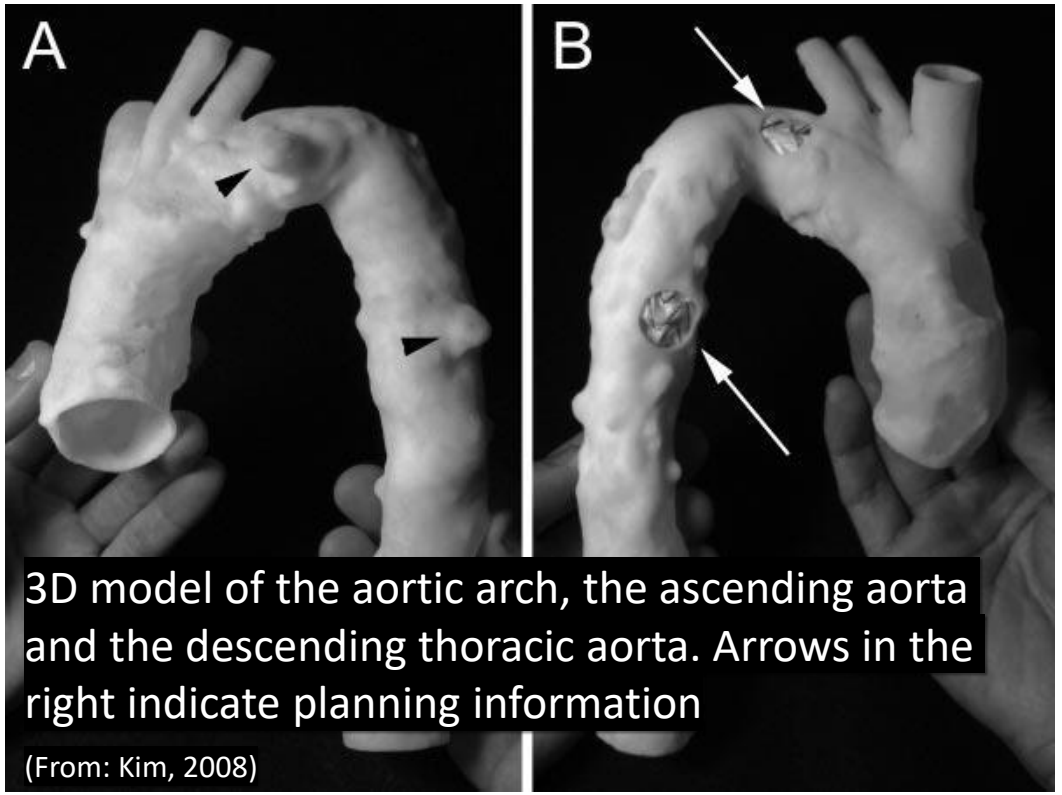


(From: Ragnar Bade)



3D model of the patients mandible with the planned reconstruction

(From: „3D-Drucker hilft bei der OP-Planung“, Ärzteblatt, 11.2.2015)



3D model of the aortic arch, the ascending aorta and the descending thoracic aorta. Arrows in the right indicate planning information

(From: Kim, 2008)

Motivation

Rapid prototyping

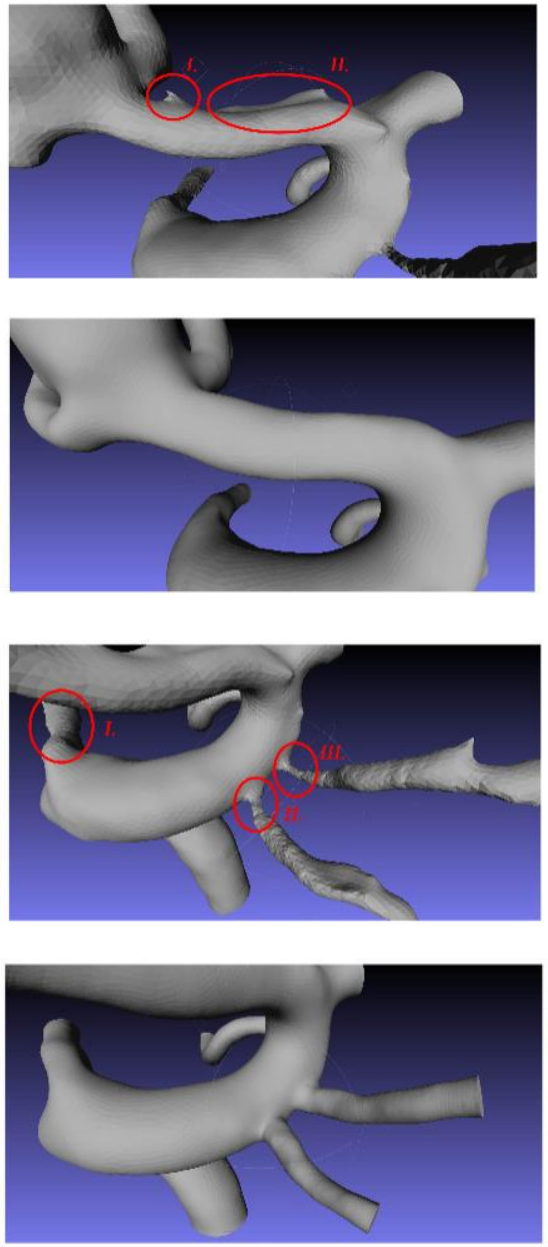
- Increasingly used for planning complex facial and heart surgery

Motivation

Model generation for biomedical simulation:

Surface models should be

- Accurate and smooth (constrained smoothing)
- Free of artifacts (mesh editing)
- As small as possible (clipping, cutting, mesh decimation)
- Of high element quality (remeshing to avoid long and thin triangles)



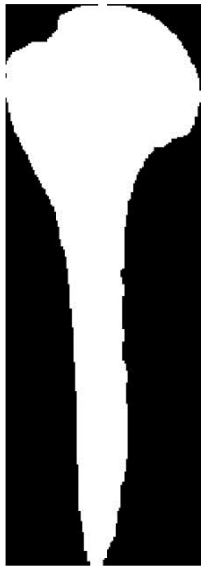
Motivation

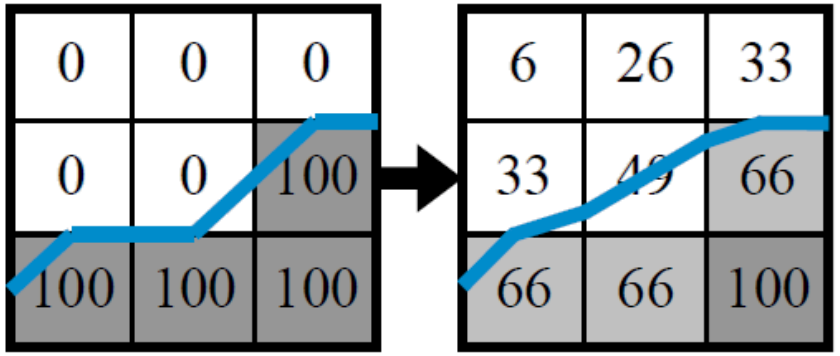
- Surface models for fluid simulation should not exhibit artifacts
- Meshes are edited, smoothed, remeshed to achieve a good element quality
- To reduce the data, vessel trees are clipped

(From: Mönch, 2011)

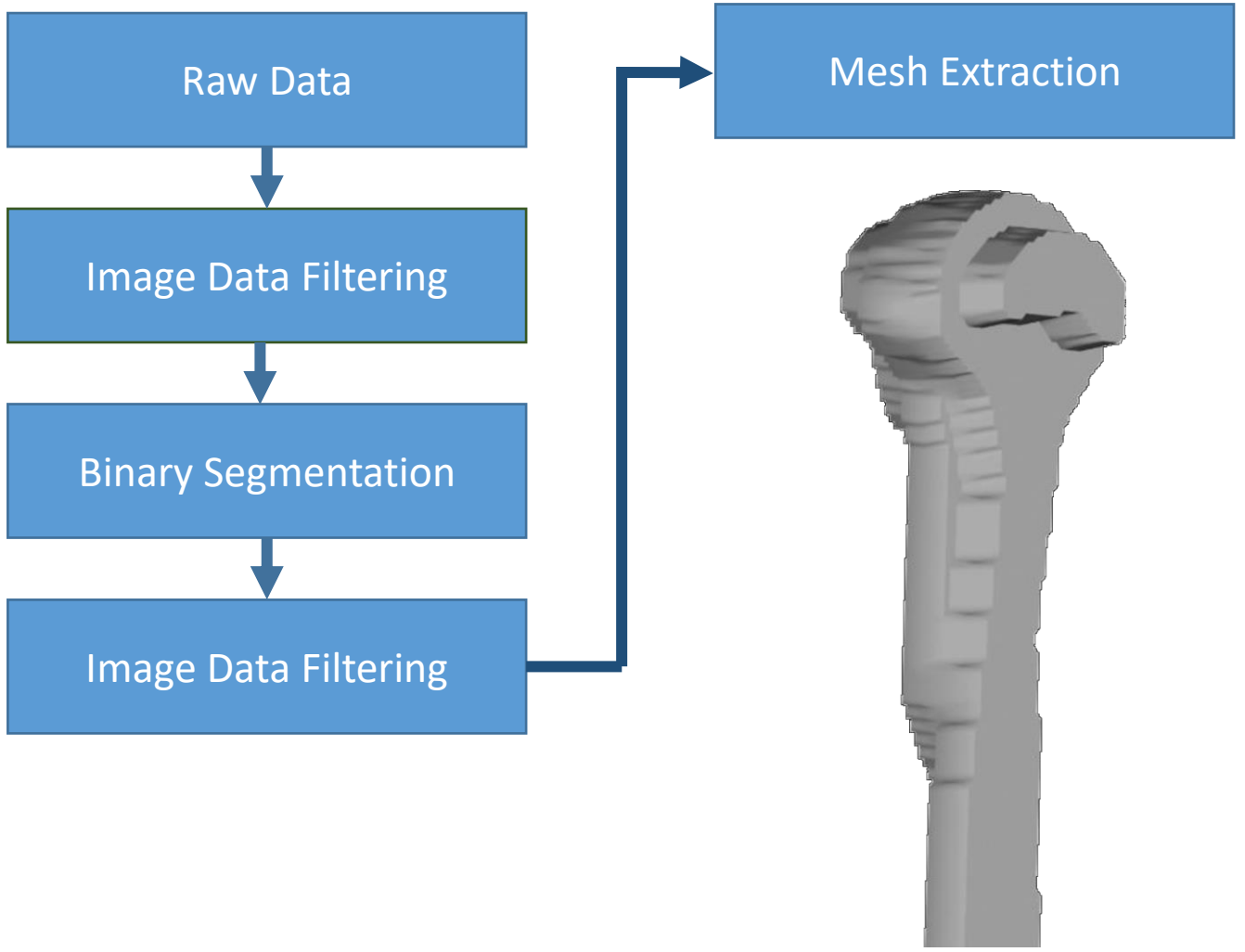
Problem Analysis

- Surface extraction of segmented objects





(From: Neubauer et al. [2004])

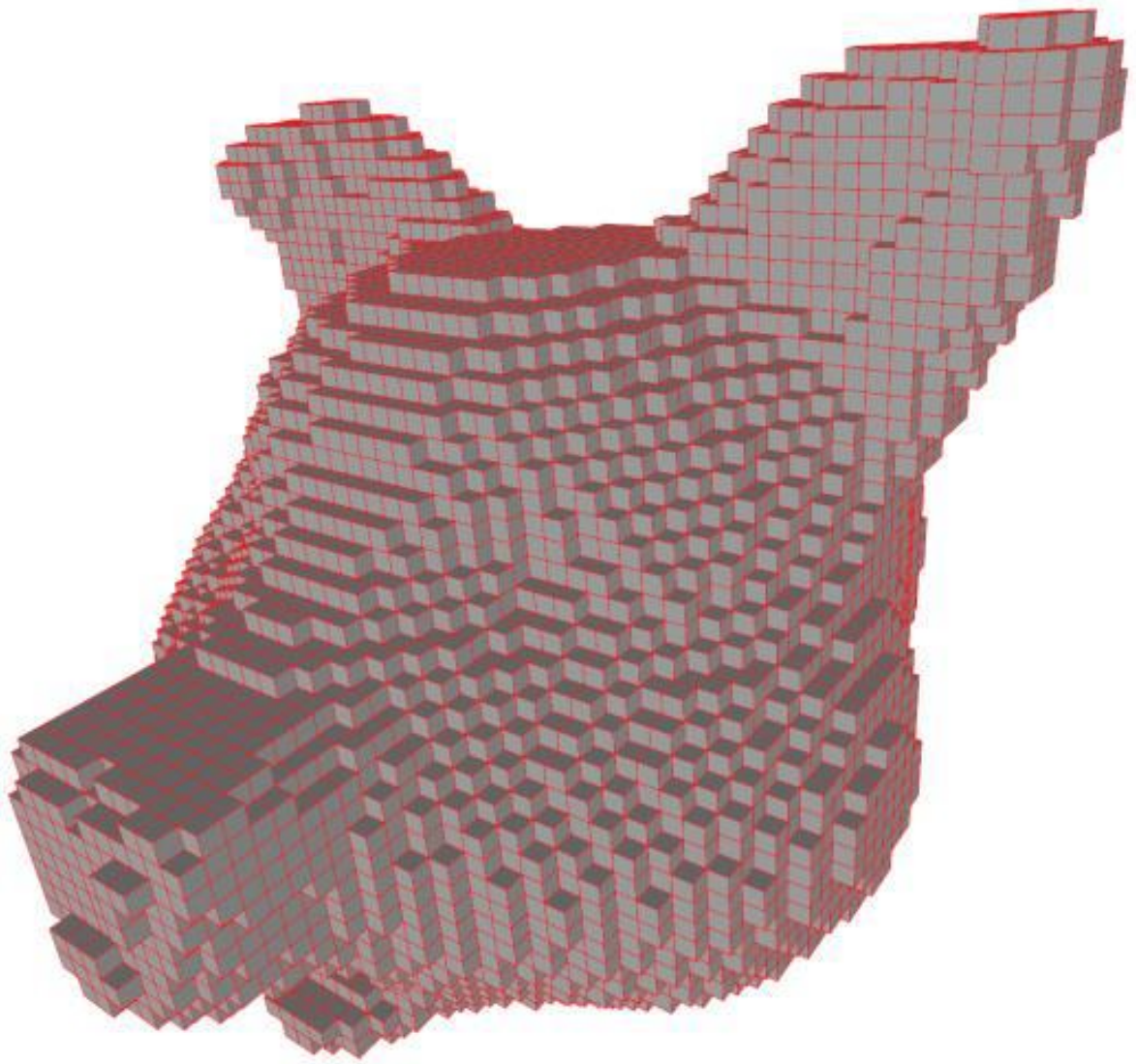


Cuberille

Cuberille

Assumption:

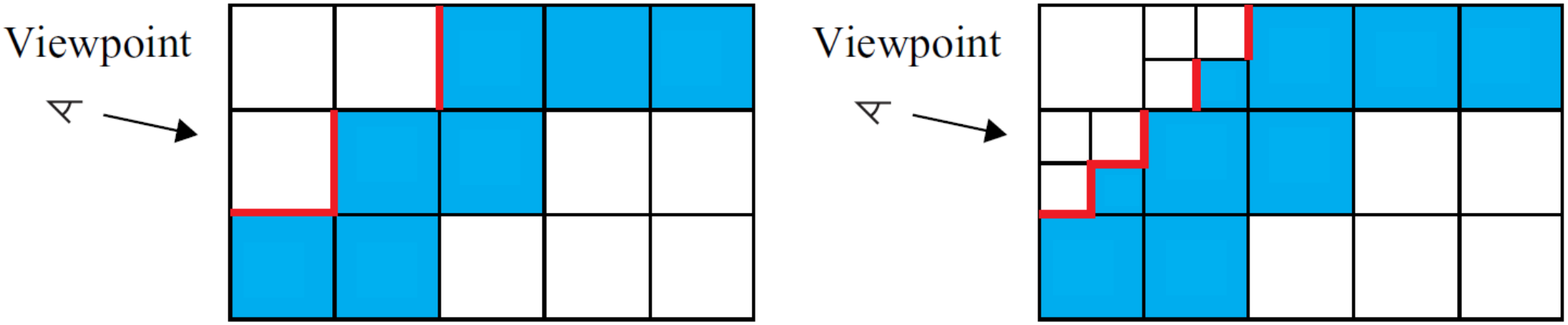
- Herman & Liu 1979
- Segmentation result is binary represented on the voxel plane
  (1 = foreground, 0 = background)
- Voxels represented with cubes
  - (each voxel=1 one cube) -> do better



(From: J. Andreas Bærentzen)

Cuberille

- Surface approximation with cubes
- Adaptive subdivide for improvement

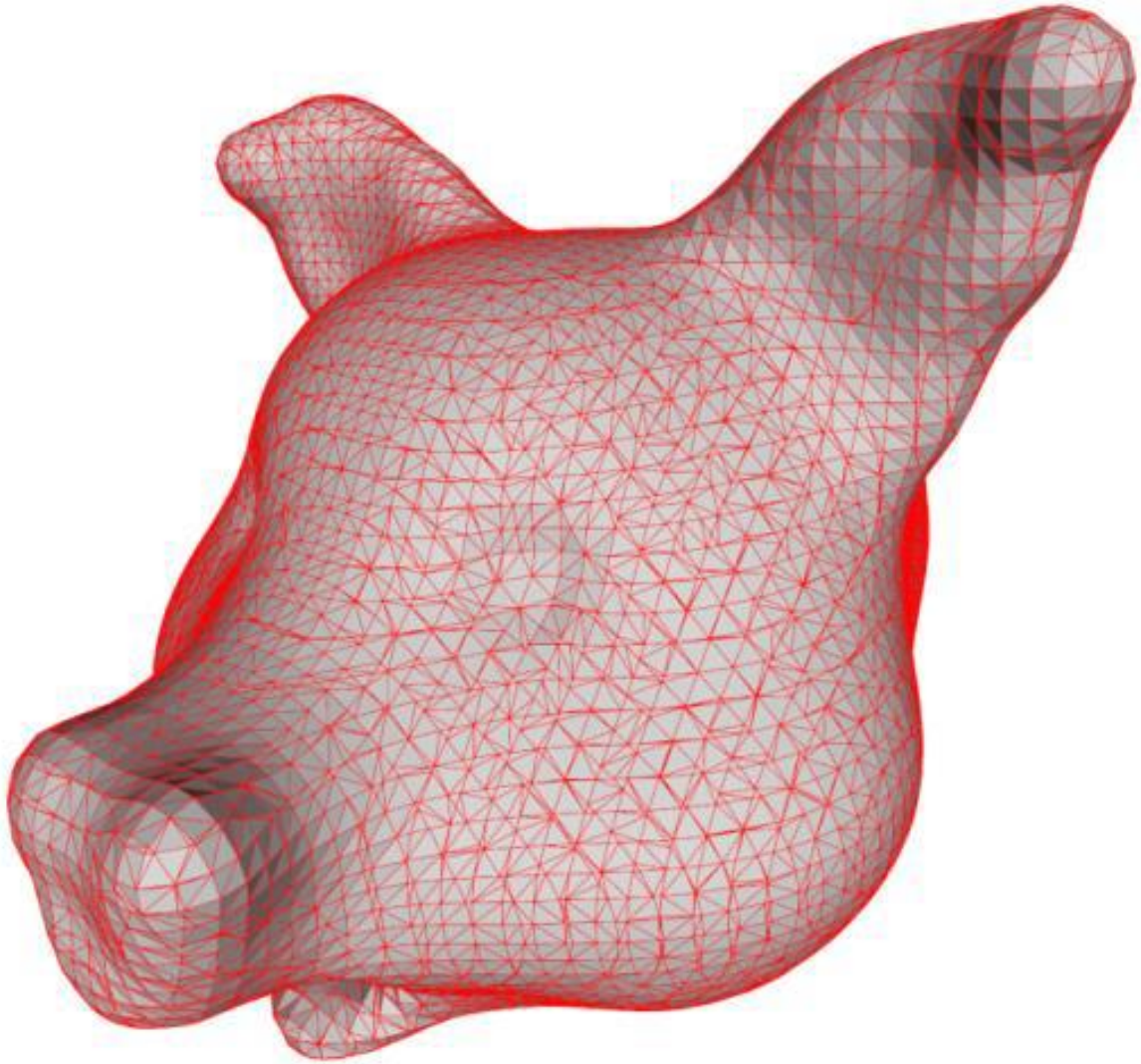


Visible faces shown in red. (Preim and Bartz 2007)

Motivation

Visualization, better idea:

- Mapping to a polygonal surface (triangle mesh)
- Determination of edge points and normals, triangulation
- Rendering with massive use of graphics hardware



(From: J. Andreas Bærentzen)

From Contours in Layers to Surfaces

From Contours in Layers to Surfaces

Implementation:

- Identification of contours in 2D layers
- Tracking of contours and connection in 3D
  - Very difficult, many case distinctions

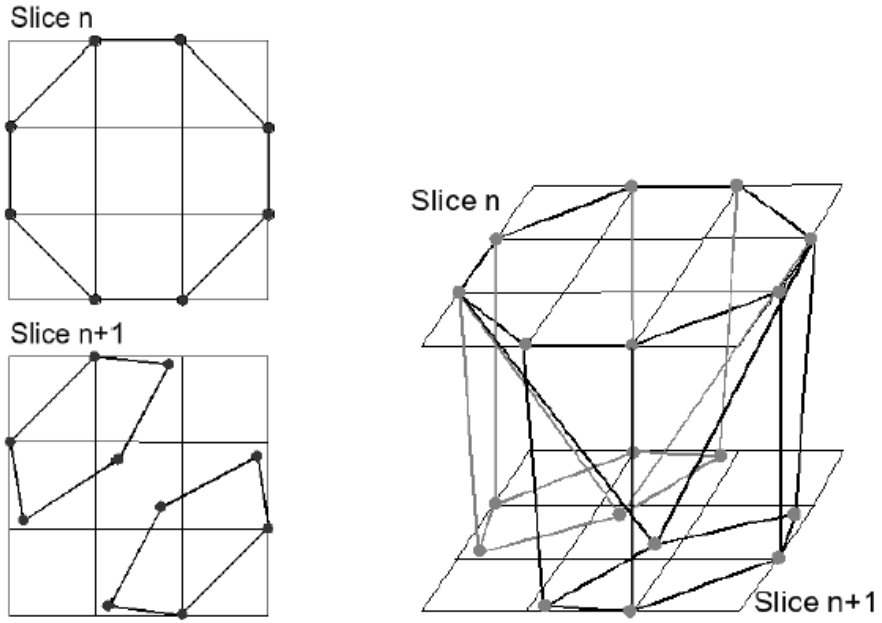
Locally independent viewing of the cells.

- Determine how the cells are cut from the surface
  - Marching Cubes (patented in 1985, published in 1987)
  - Marching Tetrahedra
  - Dividing Cubes

From Contours in Layers to Surfaces

Which problems must be solved?

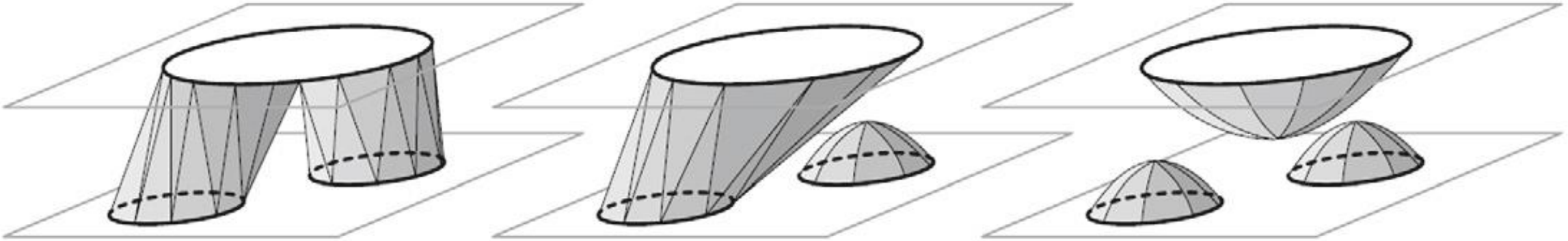
- Correspondence
  - Which contour of one layer belongs to a contour on the next layer?
- Triangulation (tiling)
  - Let C_1 and C_2 be corresponding contours. How can they be connected by triangle meshes?
- Branching problem
  - A suitable triangulation is required when the number of contours in a layer S_n is unequal to the number of contours in the neighbor layer S_{n+1}.



(From: Meyers et al. [1992])

From Contours in Layers to Surfaces

- Correspondence problem

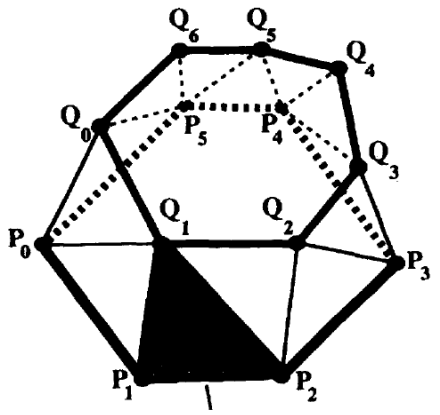


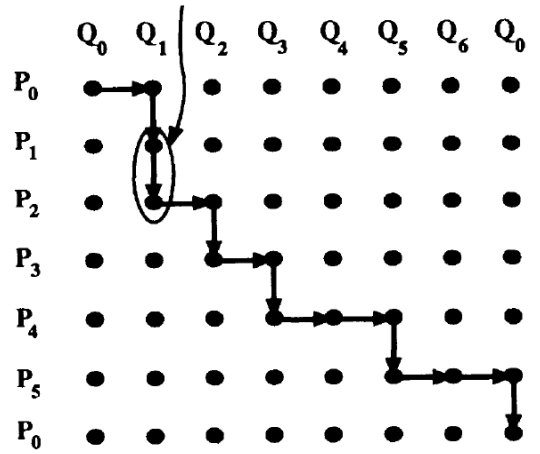
(From: Ragnar Bade)

From Contours in Layers to Surfaces

Tiling problem: Connect *n* edge points of the contour in one layer P_i with *m* edge points of the contour in the neighbor layer Q_j.

- Criteria:
  - Resulting volume shall be maximal
  - Resulting surface shall be minimal

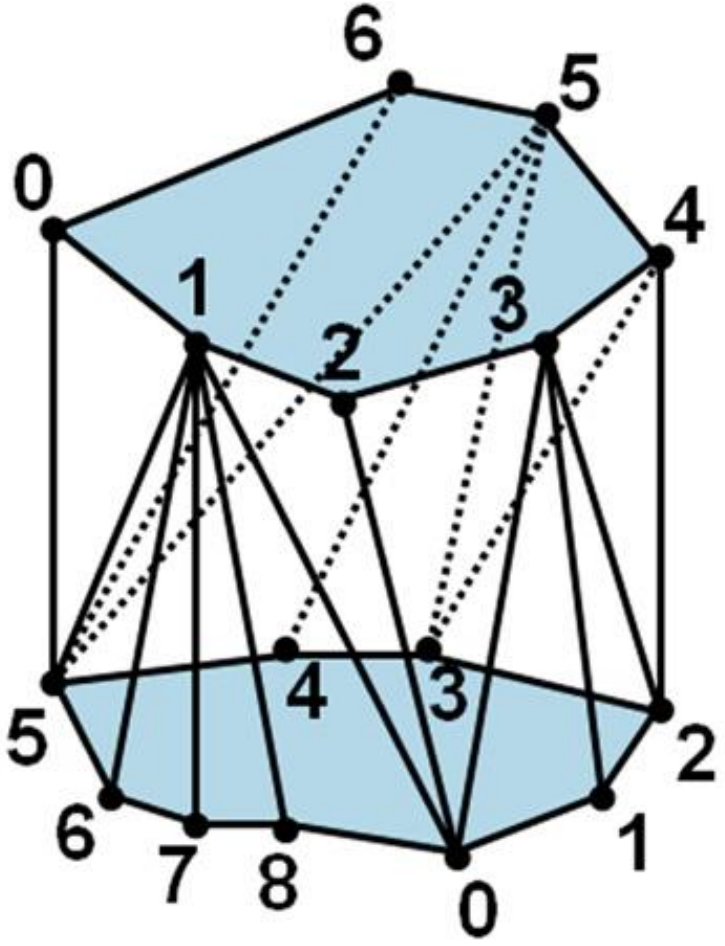




- Solution:
  - Mapping to graph search (Keppel [1975], Fuchs [1977]), remember LiveWire graph search

From Contours in Layers to Surfaces

- Tiling problem

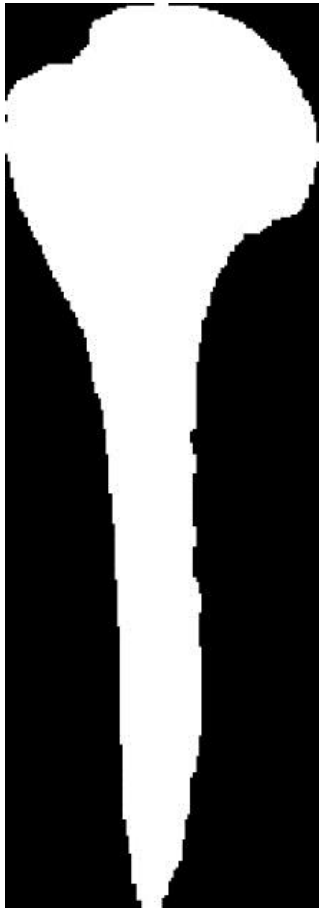


(From: Ragnar Bade)

Marching Cubes

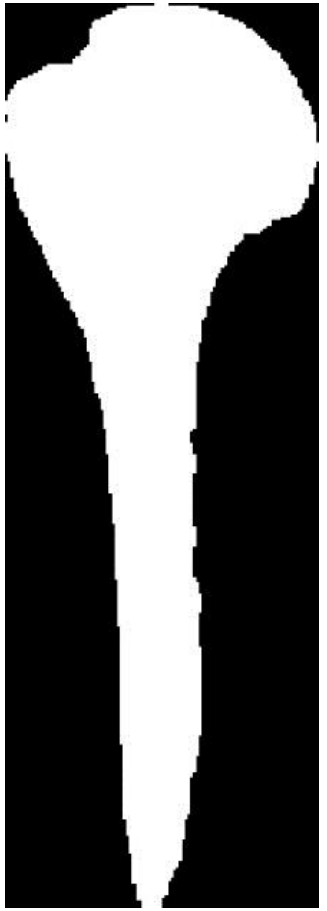
First: Marching Squares

- Assume we have a segmented image
- Task: Find the contour that separates the segmented object



First: Marching Squares

- Assume we have a segmented image
- Task: Find the contour that separates the segmented object

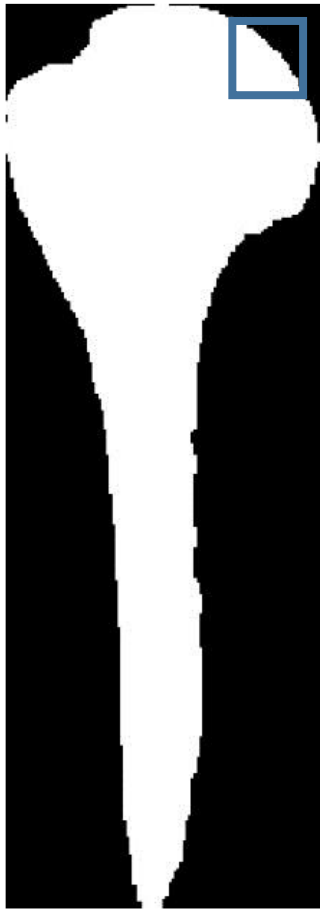


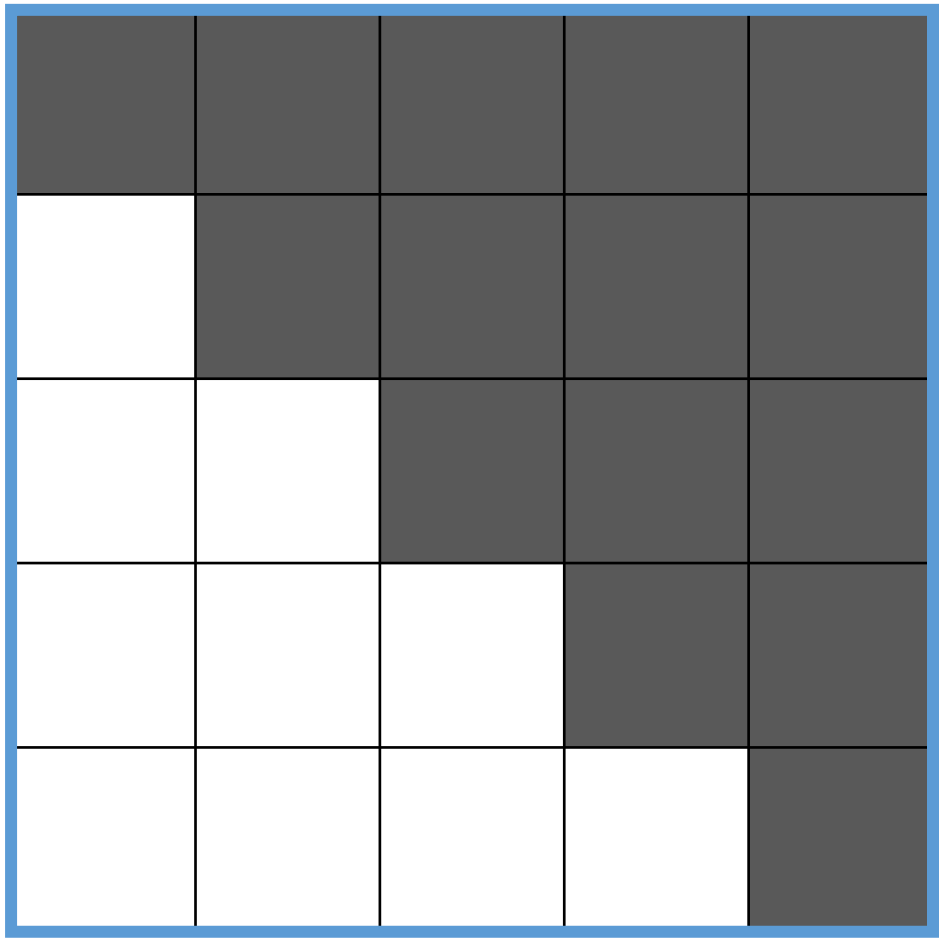
Process:

- Divide image into squares
- For every point determine if it lies inside/outside the object
- Determine edges connecting points that are inside and outside
- Calculate contour point on this edge
- Connect the points

First: Marching Squares

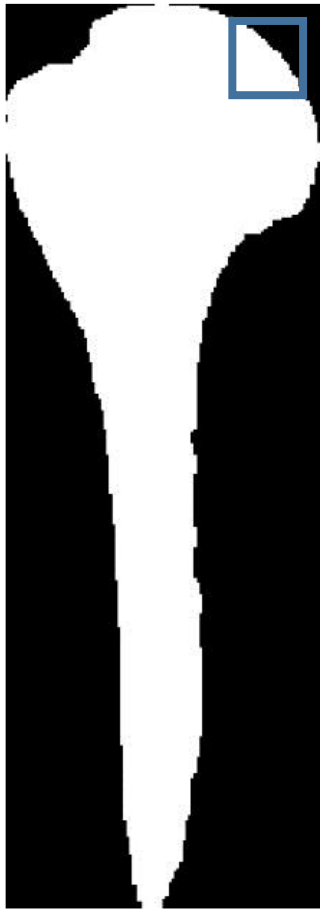
- Example:
- Points (squares) = midpoints of pixels

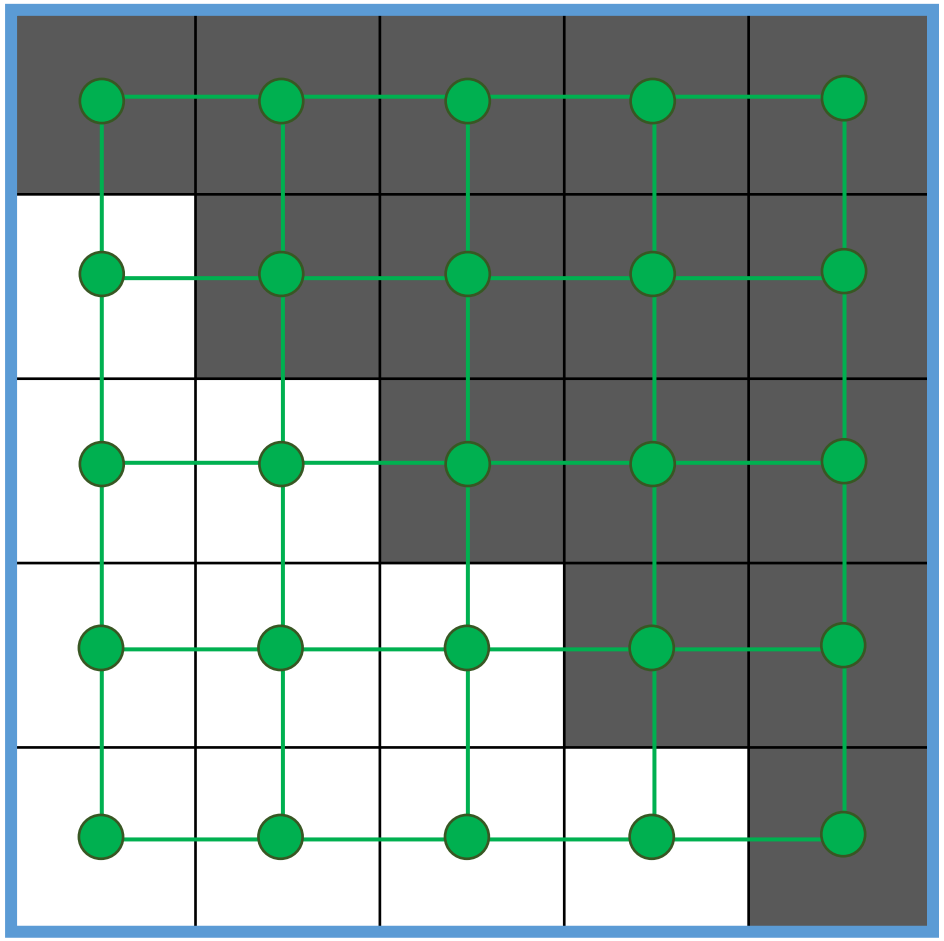




First: Marching Squares

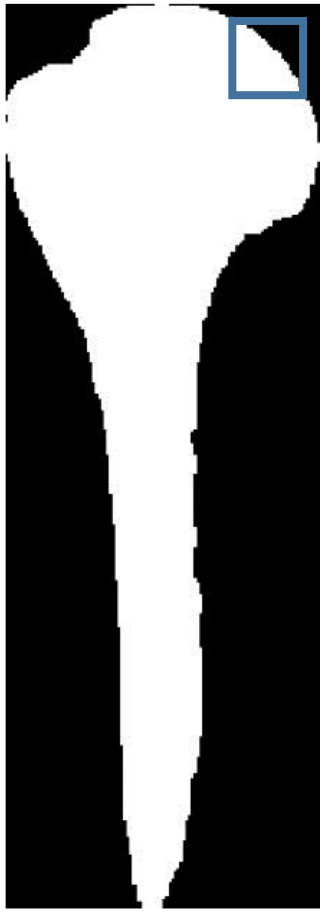
- Example:
- Points (squares) = midpoints of pixels

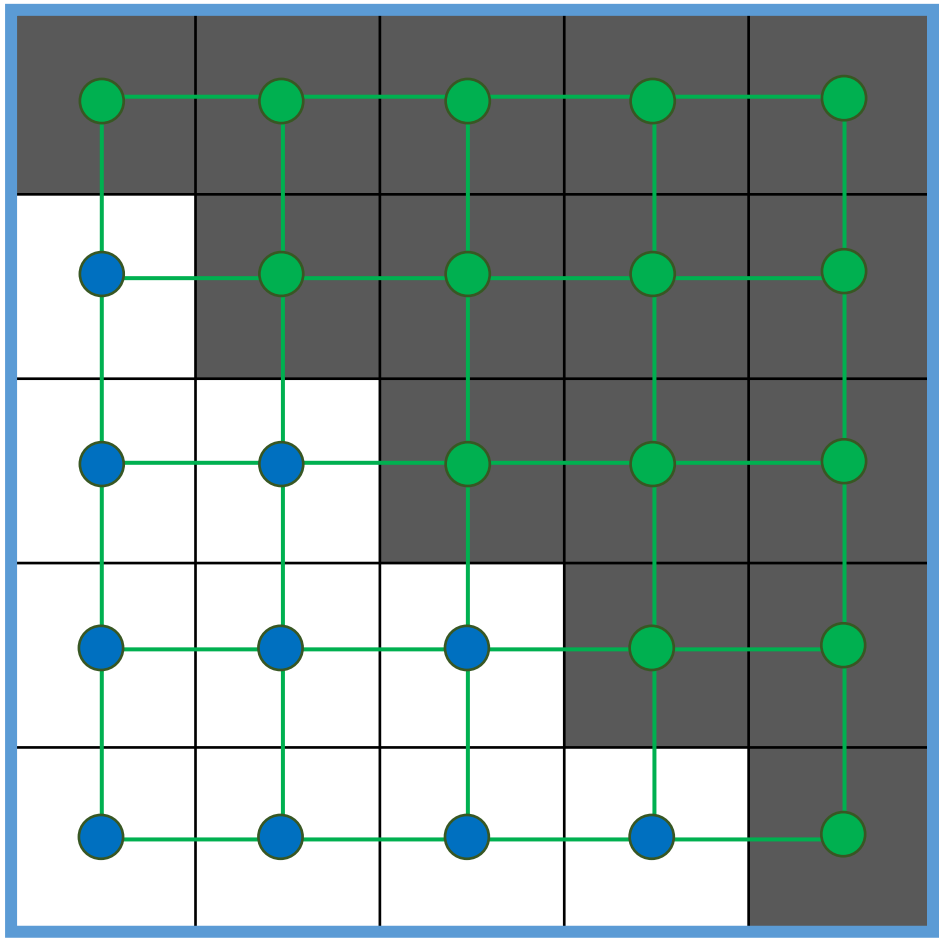




First: Marching Squares

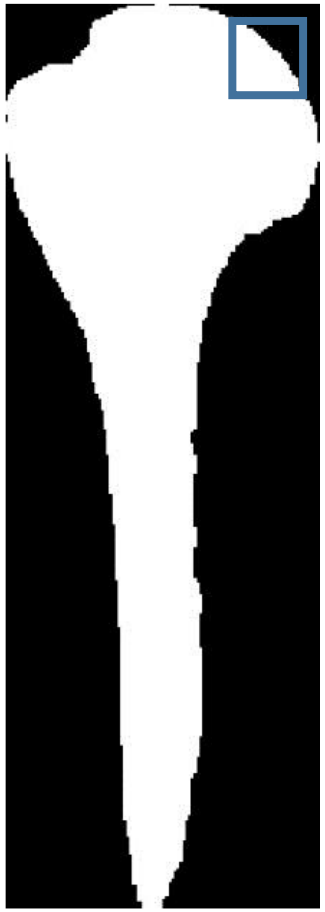
- Example:
- Points (squares) = midpoints of pixels

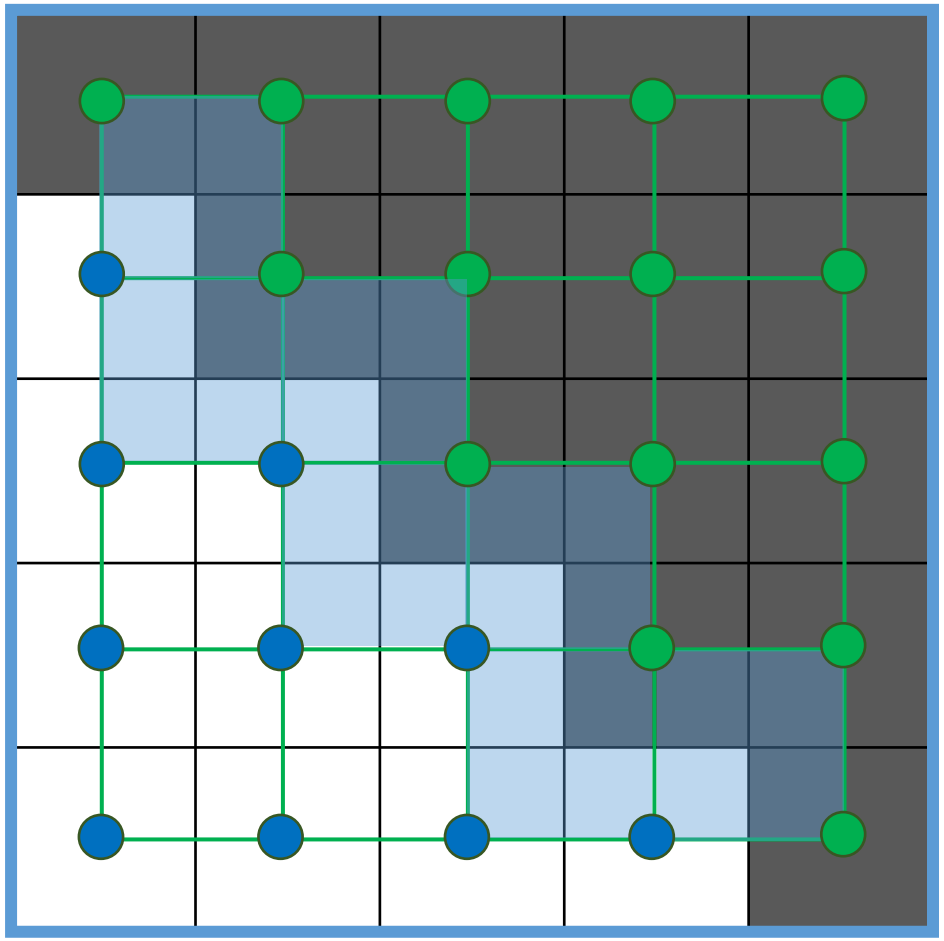




First: Marching Squares

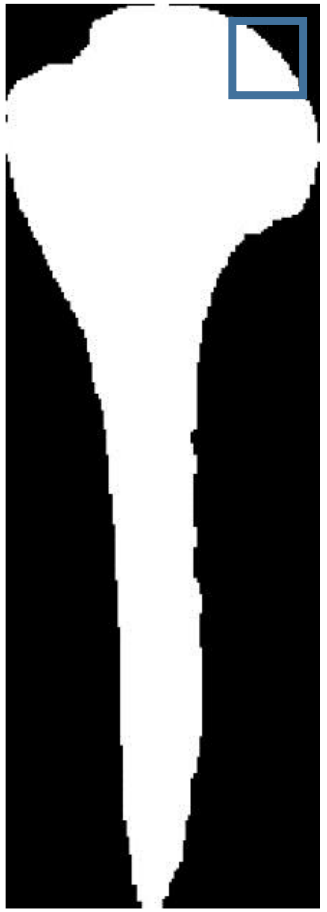
- Example:
- Points (squares) = midpoints of pixels

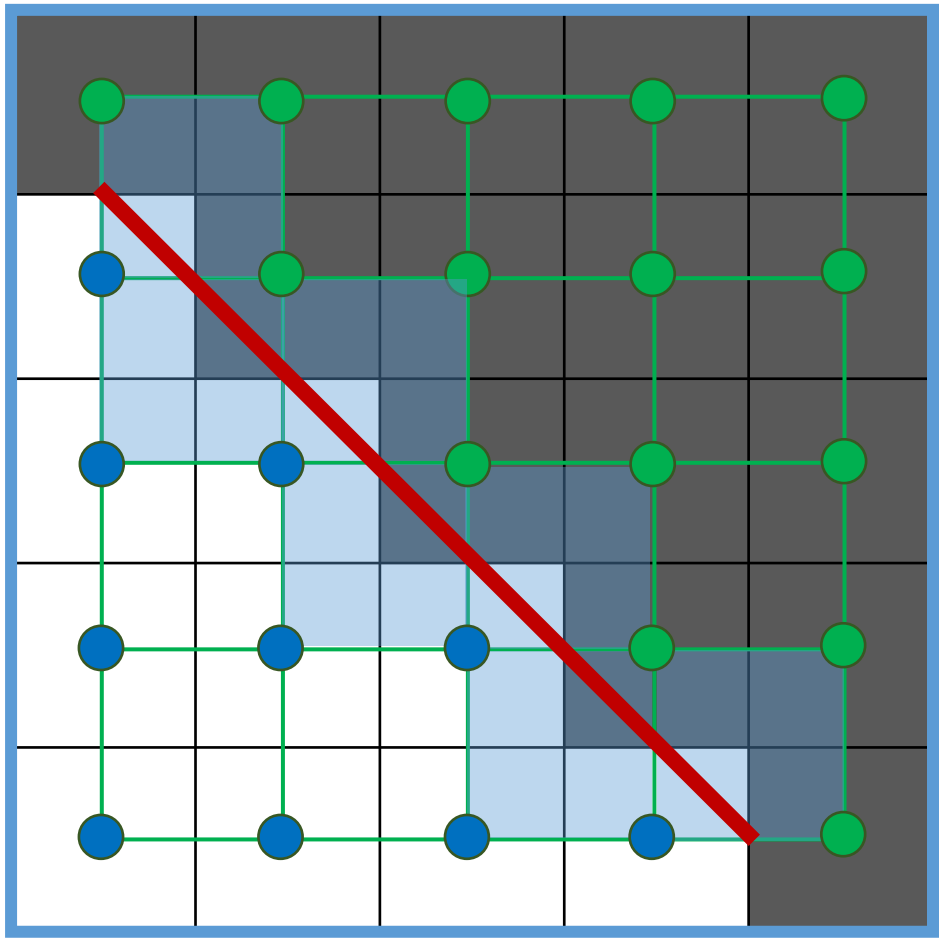




First: Marching Squares

- Example:
- Points (squares) = midpoints of pixels



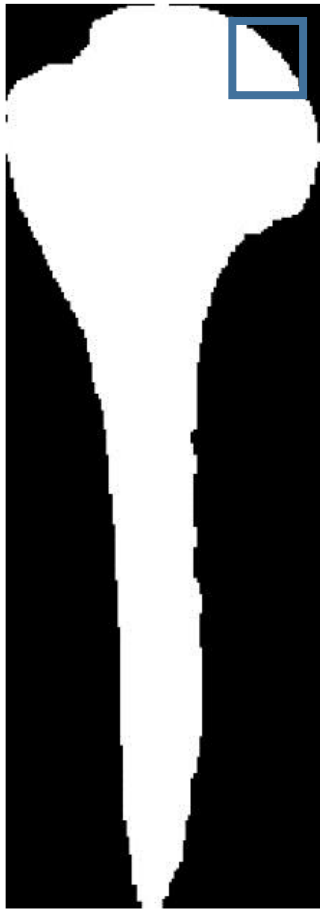


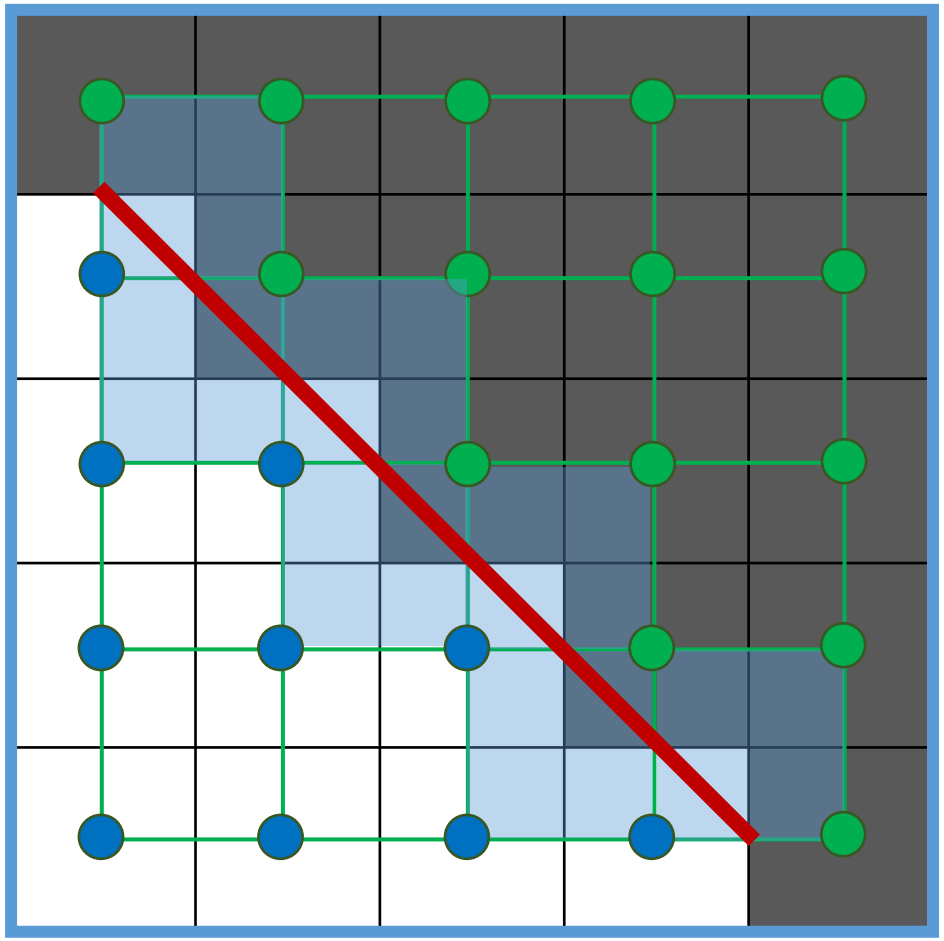
First: Marching Squares

- Example:
- Points (squares) = midpoints of pixels

- Result similar to edge detection, but

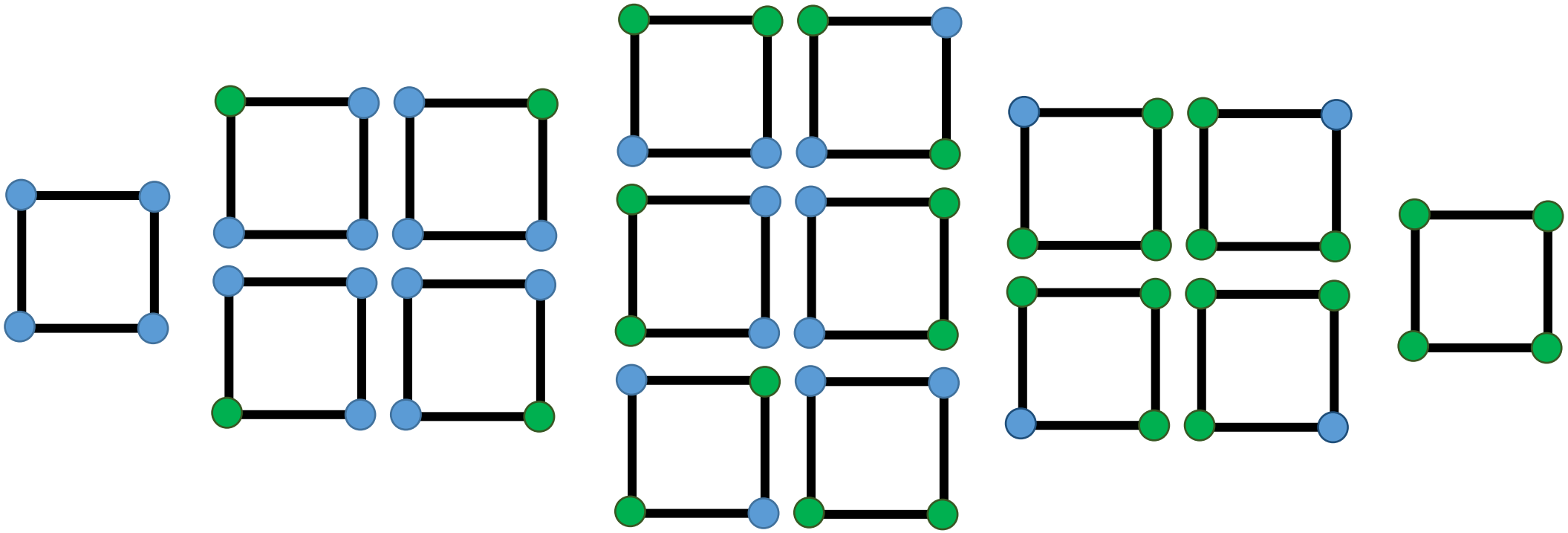
  Edge ≠ Isoline





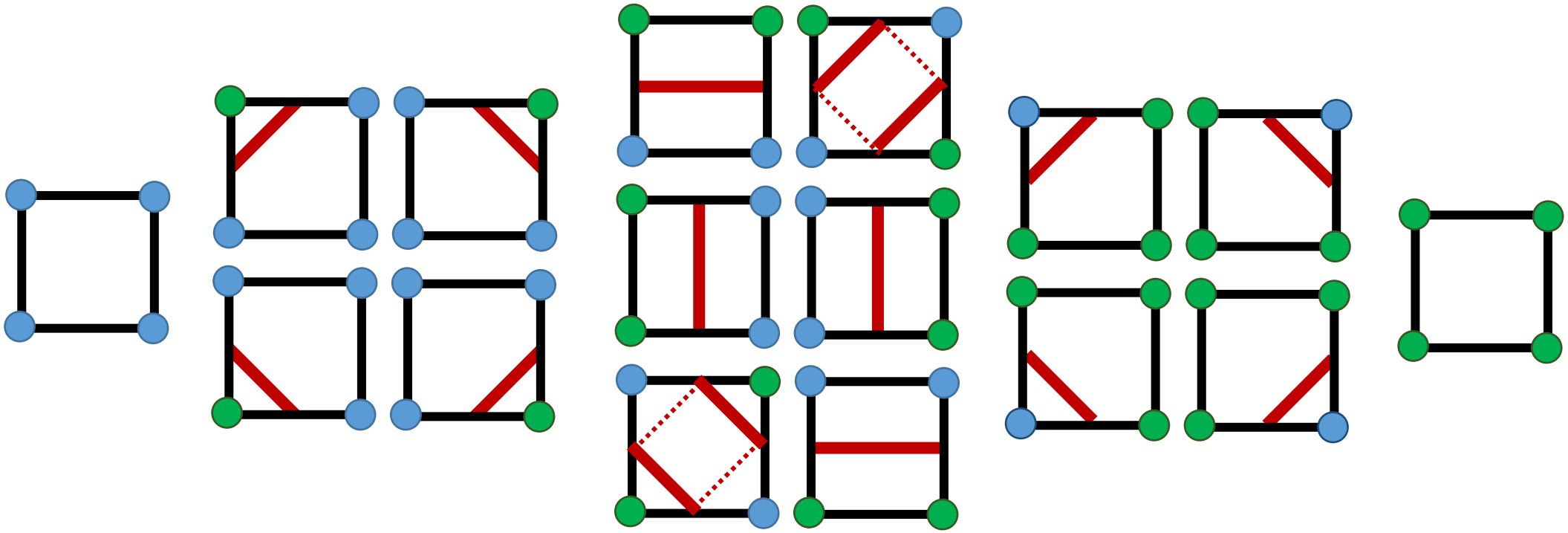
First: Marching Squares

- Combinations



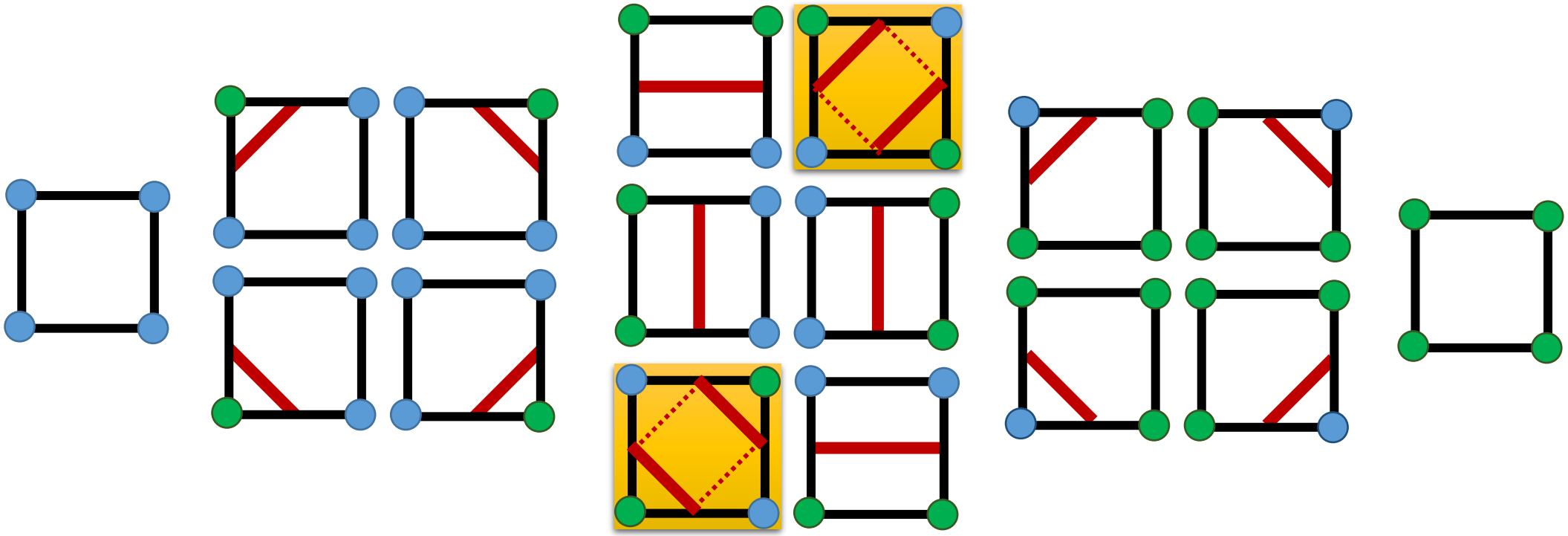
First: Marching Squares

- Combinations



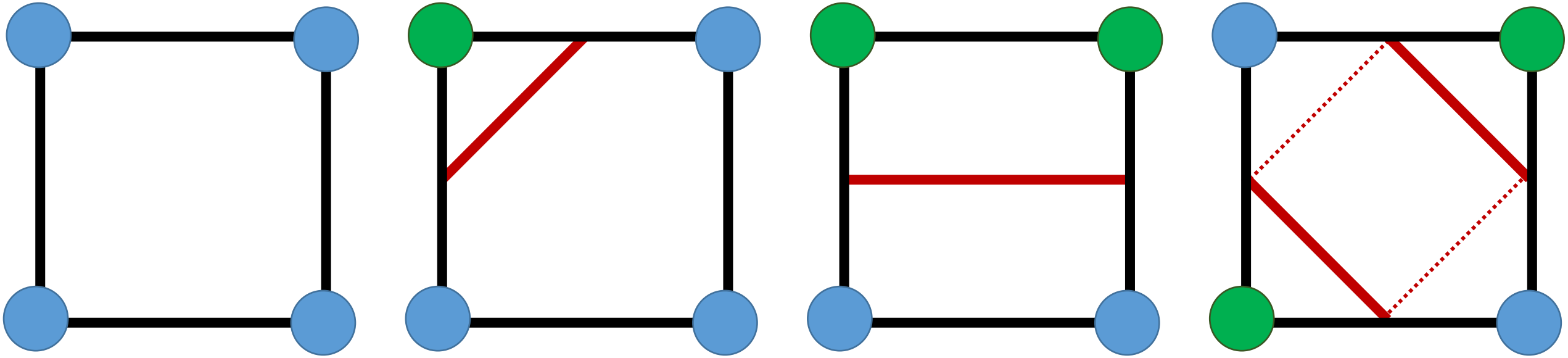
First: Marching Squares

- Combinations



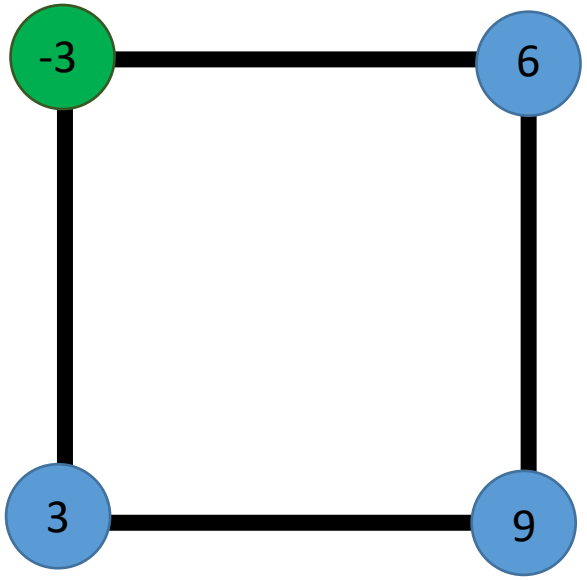
First: Marching Squares

- Combinations



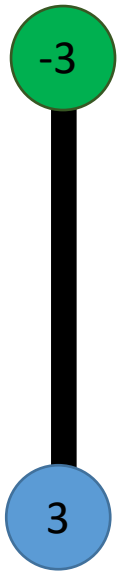
First: Marching Squares

- Finding the point on the edge with an isovalue of 0



First: Marching Squares

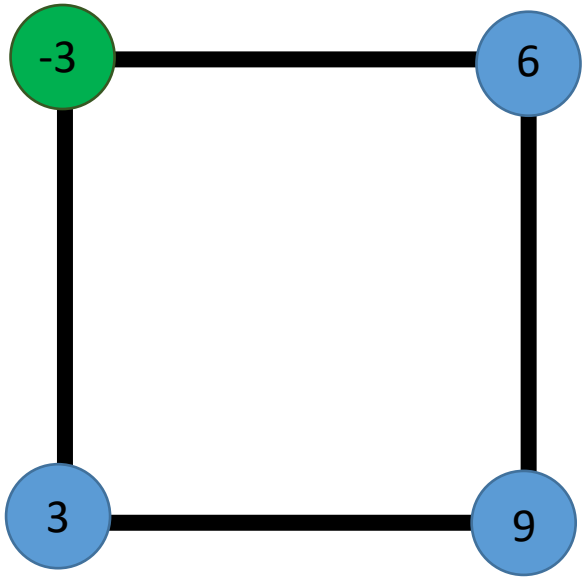
- Finding the point on the edge with an isovalue of 0



$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

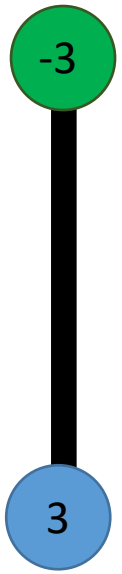
$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0



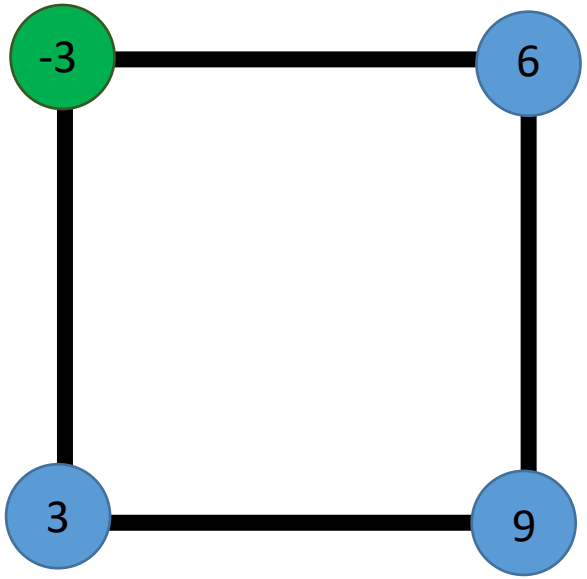
$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$

$$\lambda = \frac{3}{3 + 3}$$

$$\lambda = \frac{1}{2}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0

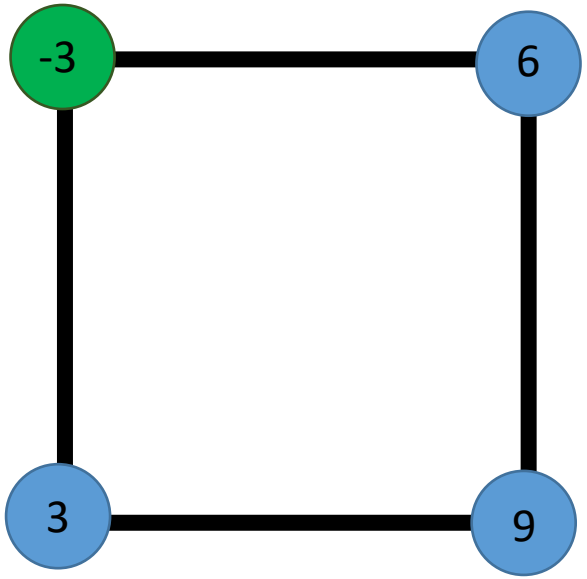
$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$

$$\lambda = \frac{3}{3 + 3}$$

$$\lambda = \frac{1}{2}$$



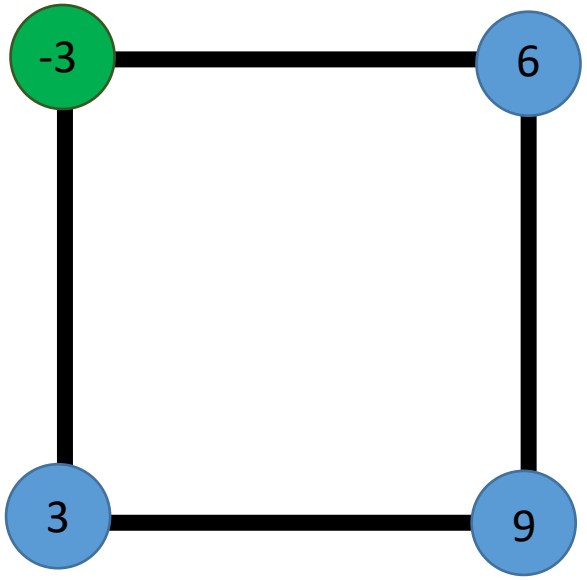
First: Marching Squares

- Finding the point on the edge with an isovalue of 0

$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0

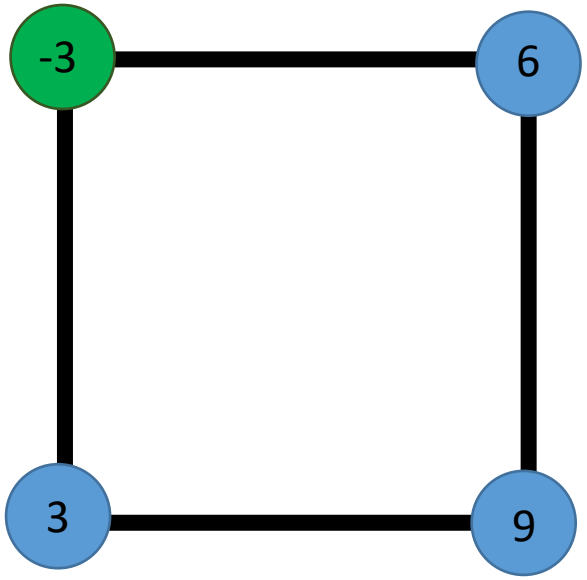
$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$

$$\lambda = \frac{6}{6 + 3}$$

$$\lambda = \frac{2}{3}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0

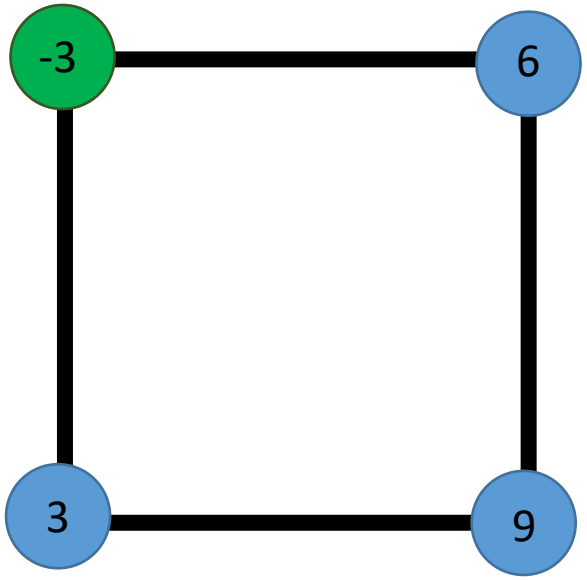
$$0 = \lambda \cdot v_1 + (1 - \lambda) \cdot v_2$$

$$0 = \lambda \cdot (v_1 - v_2) + v_2$$

$$\lambda = \frac{v_2}{v_2 - v_1}$$

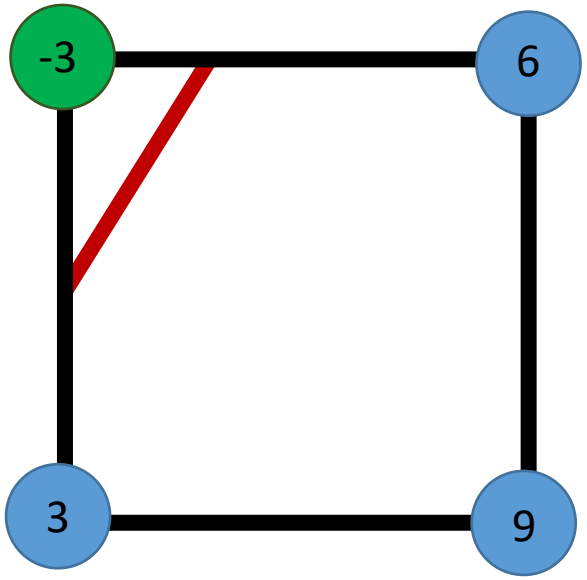
$$\lambda = \frac{6}{6 + 3}$$

$$\lambda = \frac{2}{3}$$



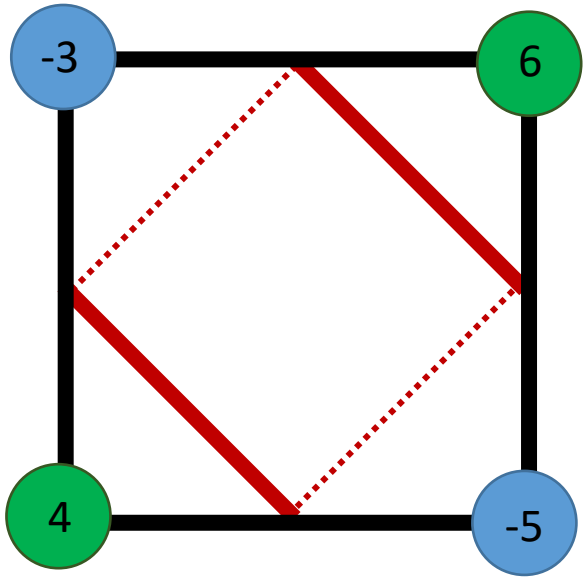
First: Marching Squares

- Finding the point on the edge with an isovalue of 0



First: Marching Squares

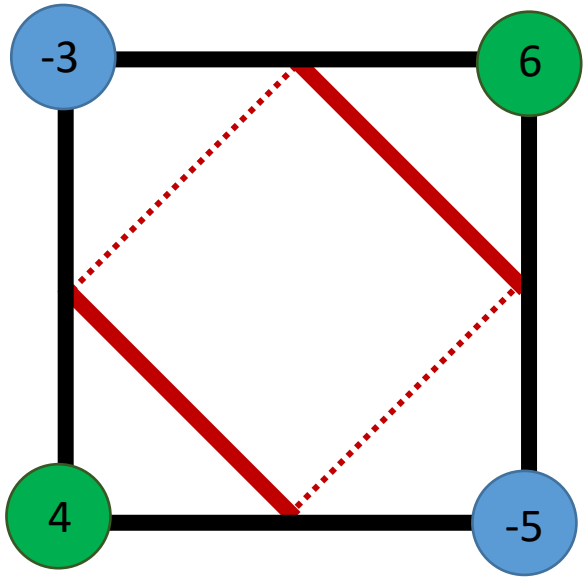
- Finding the point on the edge with an isovalue of 0



First: Marching Squares

- Finding the point on the edge with an isovalue of 0
- Calculate the isovalue in the middle by averaging all corners:

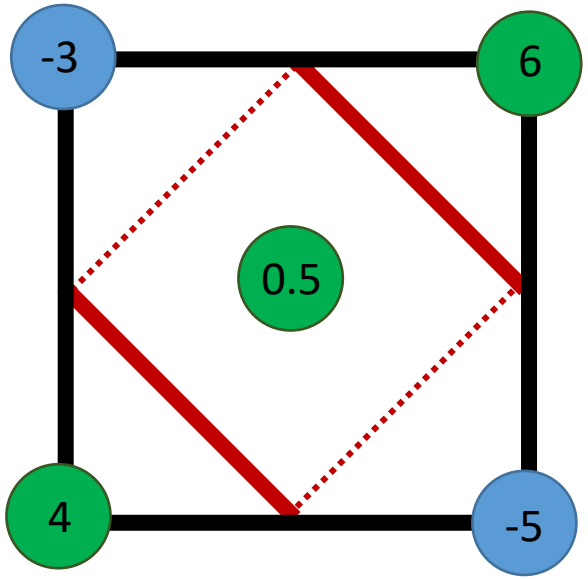
$$\frac{-3+6+4-5}{4} = \frac{1}{2}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0
- Calculate the isovalue in the middle by averaging all corners:

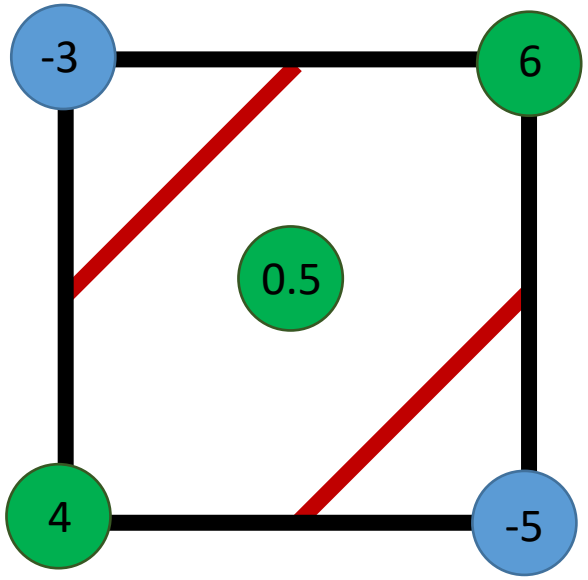
$$\frac{-3+6+4-5}{4} = \frac{1}{2}$$



First: Marching Squares

- Finding the point on the edge with an isovalue of 0
- Calculate the isovalue in the middle by averaging all corners:

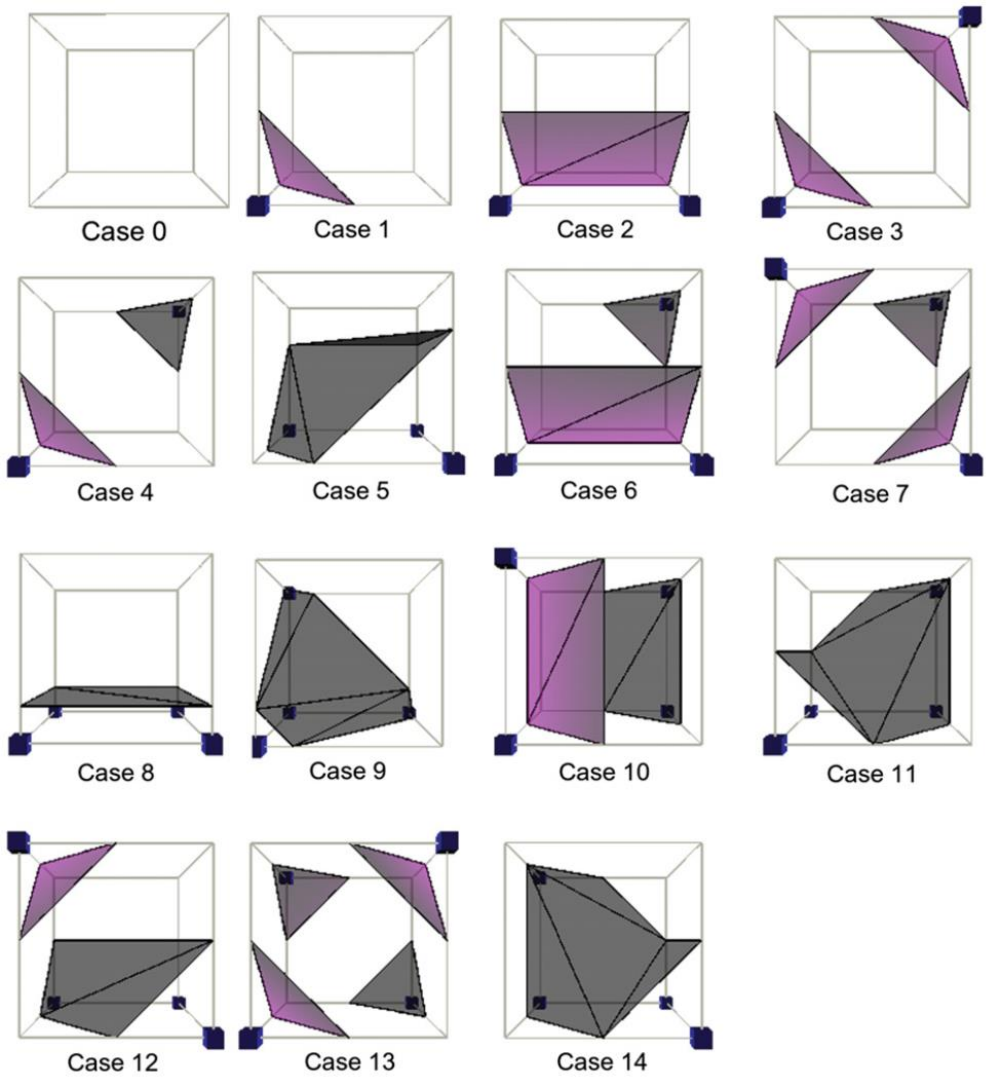
$$\frac{-3+6+4-5}{4} = \frac{1}{2}$$



Marching Cubes

- Extension from squares to cubes
- Instead of generating lines -> surfaces
- 15 combinations

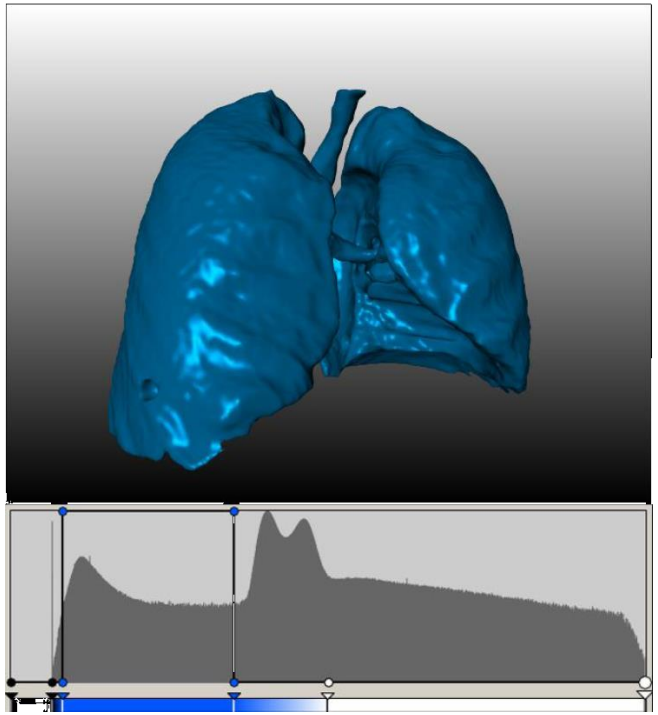
- Ambiguities:
  - Connected, or separated surfaces arise
  - Holes

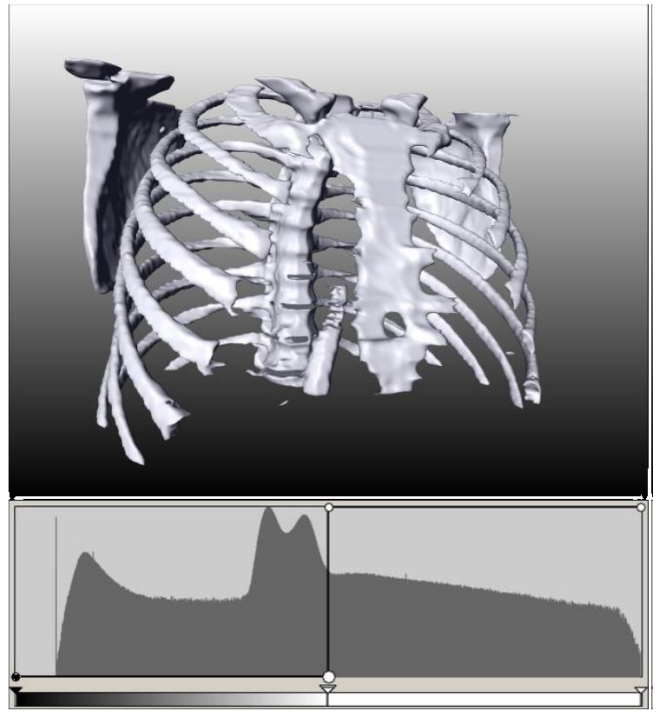


Marching Cubes and its Improvements

- Marching Cubes applied to original data
- Challenge: determine isovalue

- Histogram depiction of CT thorax data sets
- For selection of isovalues





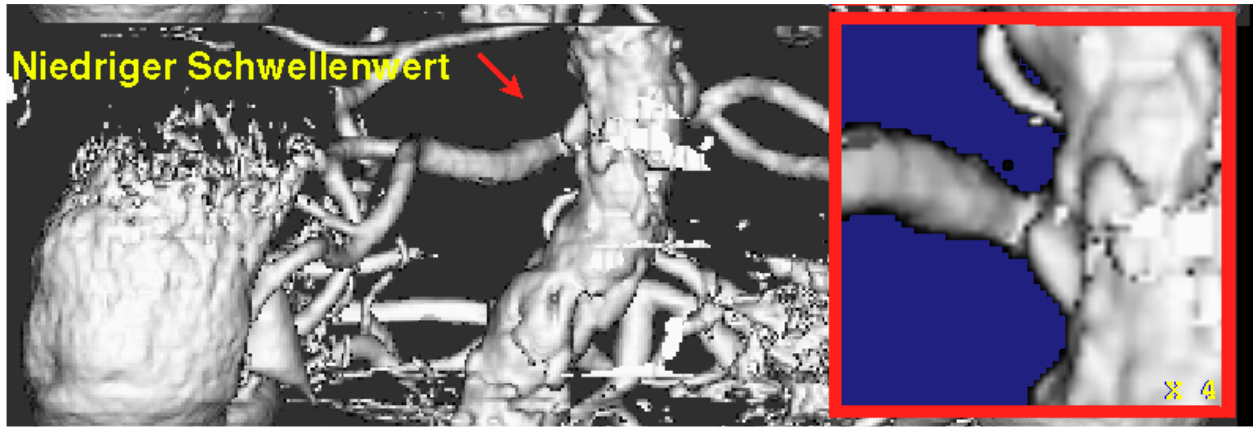
(From: Preim & Botha [2013])

Marching Cubes and its Improvements

- Thresholds for mesh extraction are not ideal (even if carefully selected).
- The presence of noise and partial volume effects may lead to inhomogeneous contour regions.
- As a consequence, there is considerable uncertainty about the true location of boundary surfaces (Wei, 2015)

Marching Cubes and its Improvements

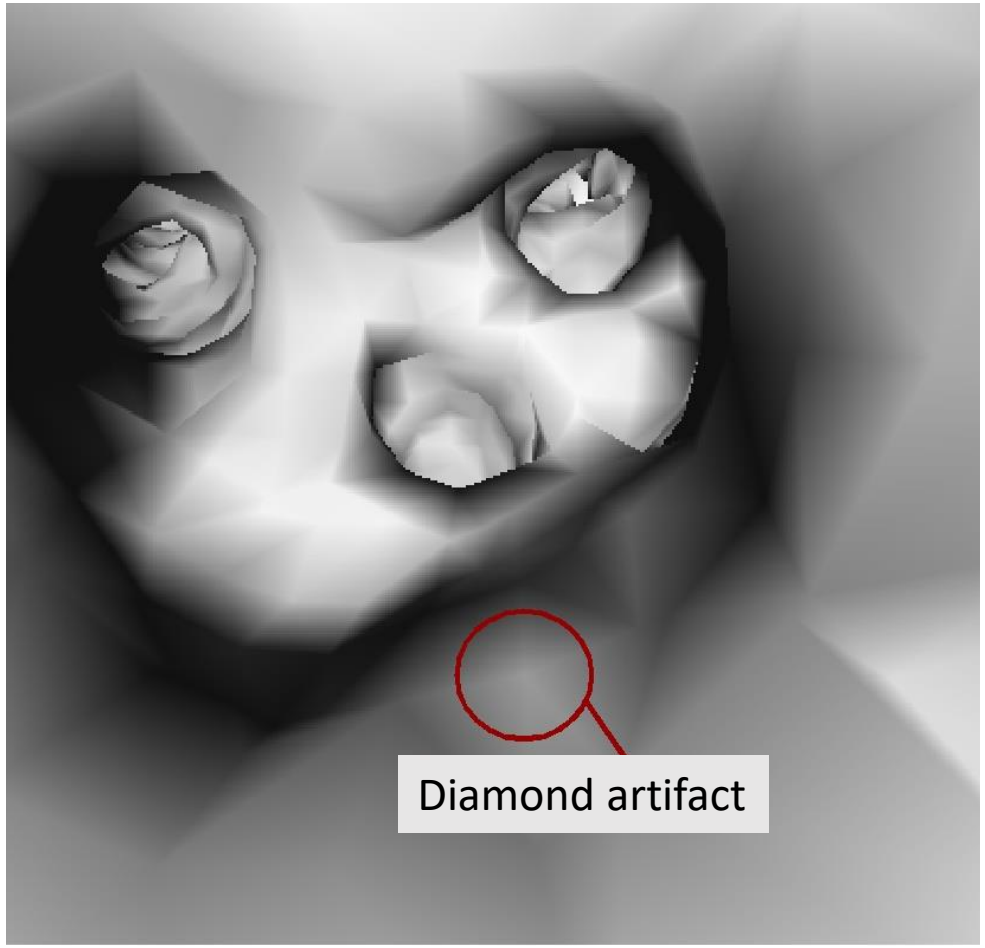
- Small changes of the threshold (isovalue) may severely affect the resulting surface and the diagnostic consequences



(Courtesy of Hoen-Oh Shin, Medical School Hannover)

Marching Cubes and its Improvements

- Quality problems through linear interpolation in binary (segmented) data and Gouraud shading



(Virtual bronchoscopy

From: Dirk Bartz)

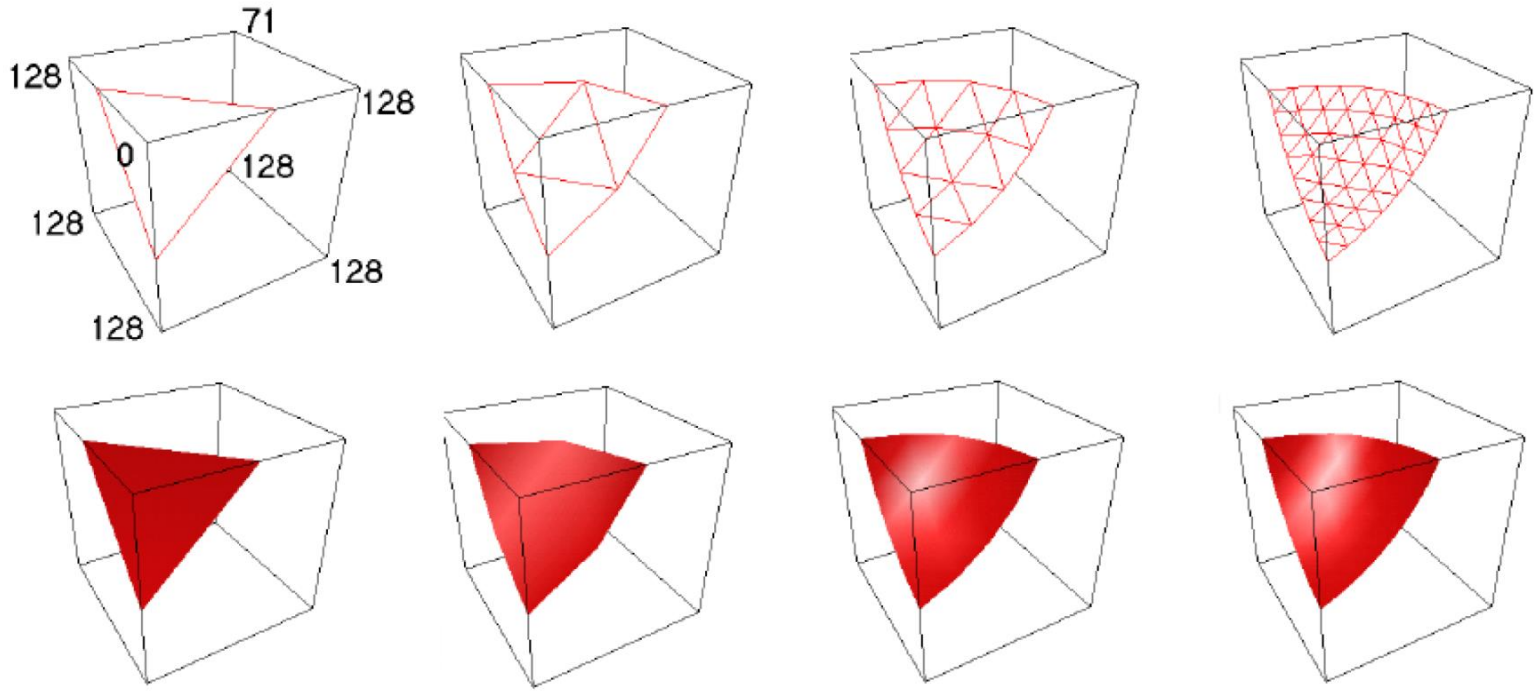
Marching Cubes and its Improvements

What is important about Marching Cubes?

- It is very simple
- Compared to Cuberille: Better depiction through linear interpolation.
  - **But**: Viewers are also sensitive for discontinuities of the 1st and 2nd derivative
- Ambiguities and inconsistencies, no treatment of the correspondence problem, no optimal solution for the tiling and branching problem
  - **Alternatives**: Marching Tetrahedron, Precise Marching Cubes
- Quite fast procedure
  - **But**: many time is wasted on cells that do not contribute to the surface
- Fast Rendering

Precise Marching Cubes

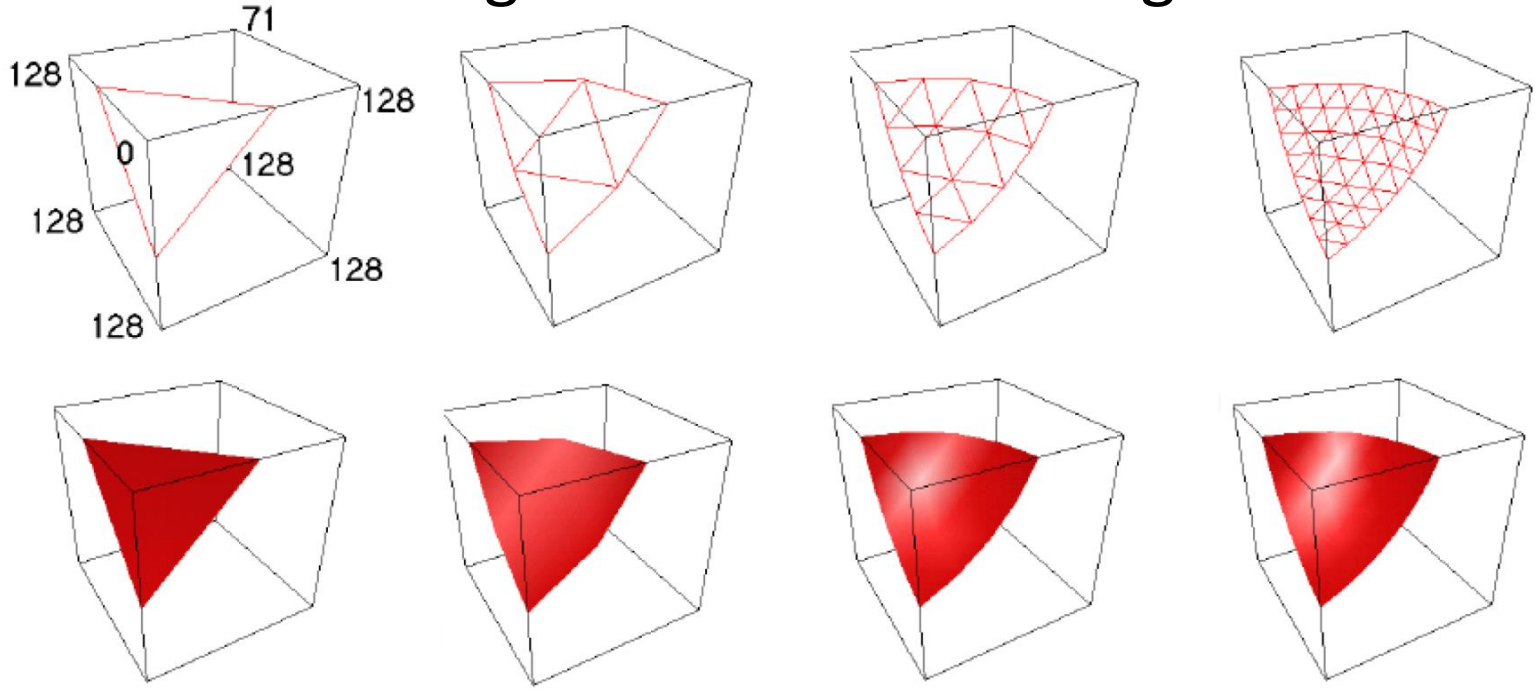
- Strategy: Adaptively refine the surface
- Triangles refined (based on error threshold)



(From: Allamandri et al. [1998])

Precise Marching Cubes

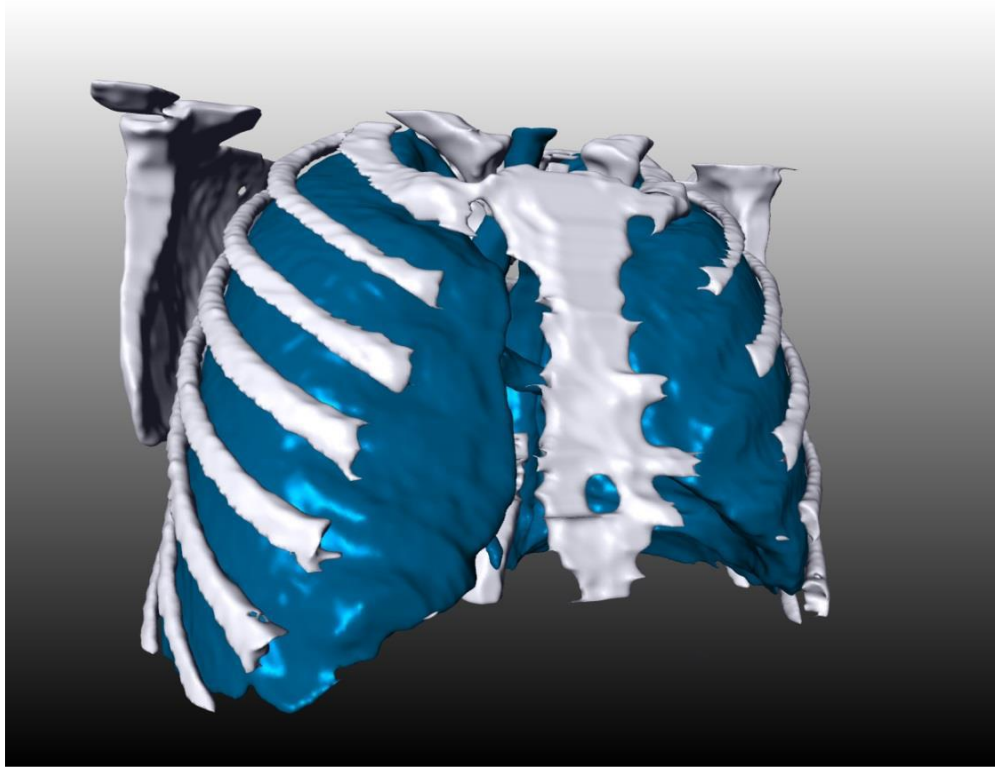
- Surface determined by trilinear interpolation
- 100 more expensive than marching cubes
- Surface 2-3 larger in terms of # triangles

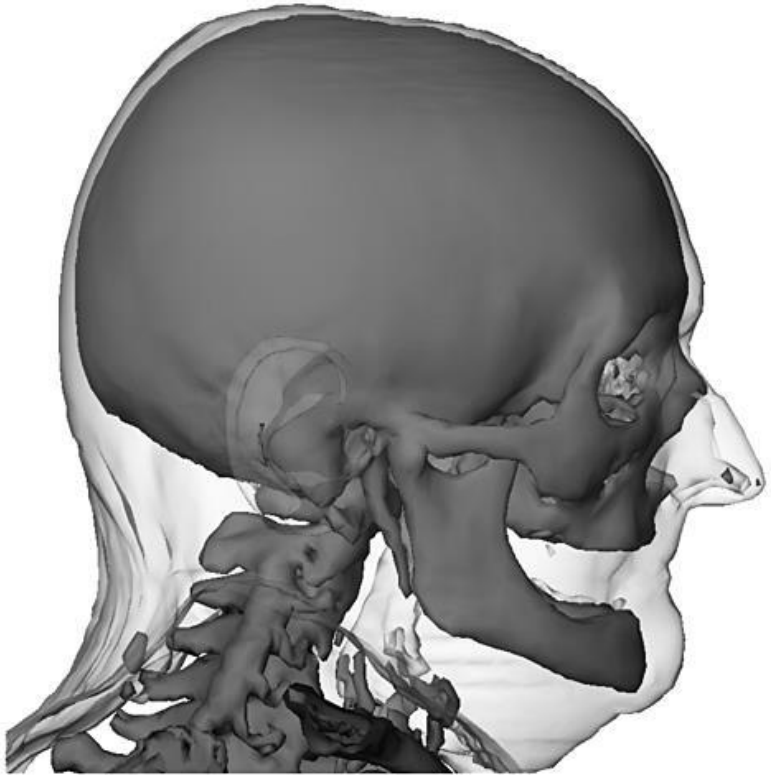


(From: Allamandri et al. [1998])

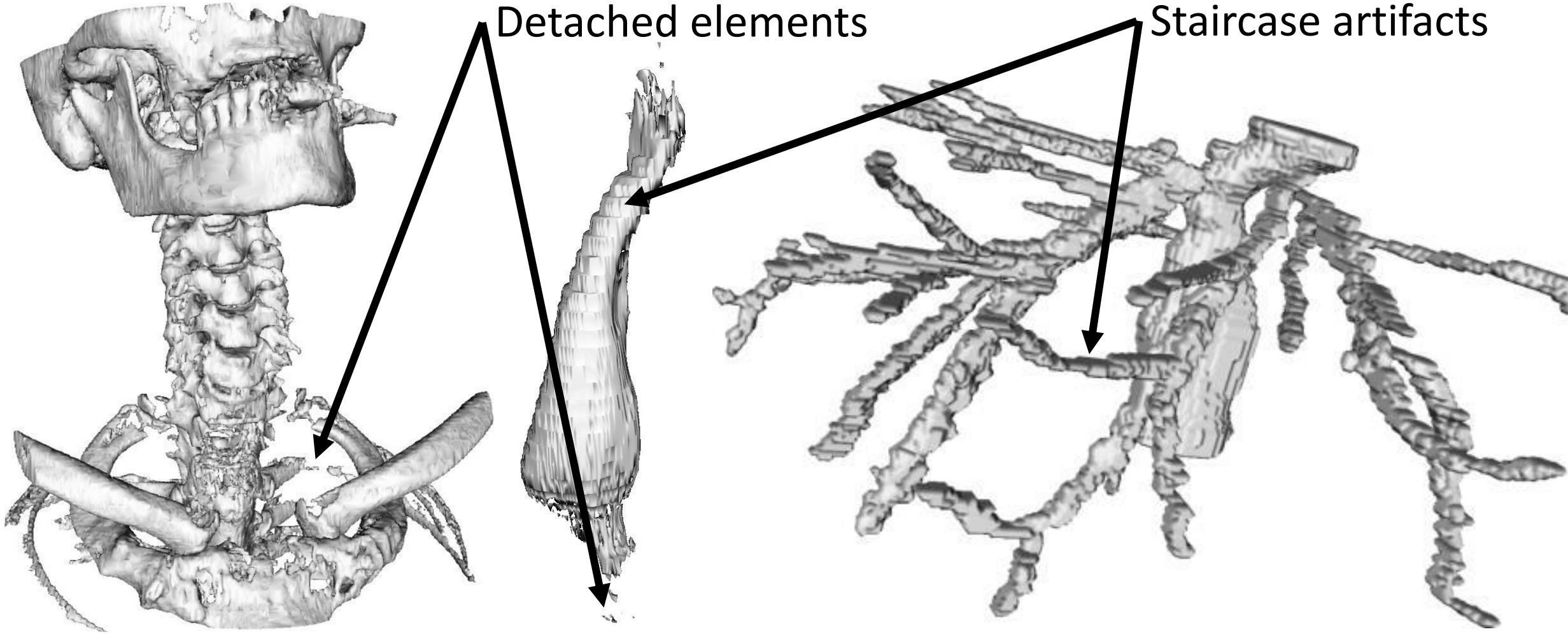
Marching Cubes and its Improvements

- Simultaneous illustration of several surfaces. Right: outer surface is semi-transparent



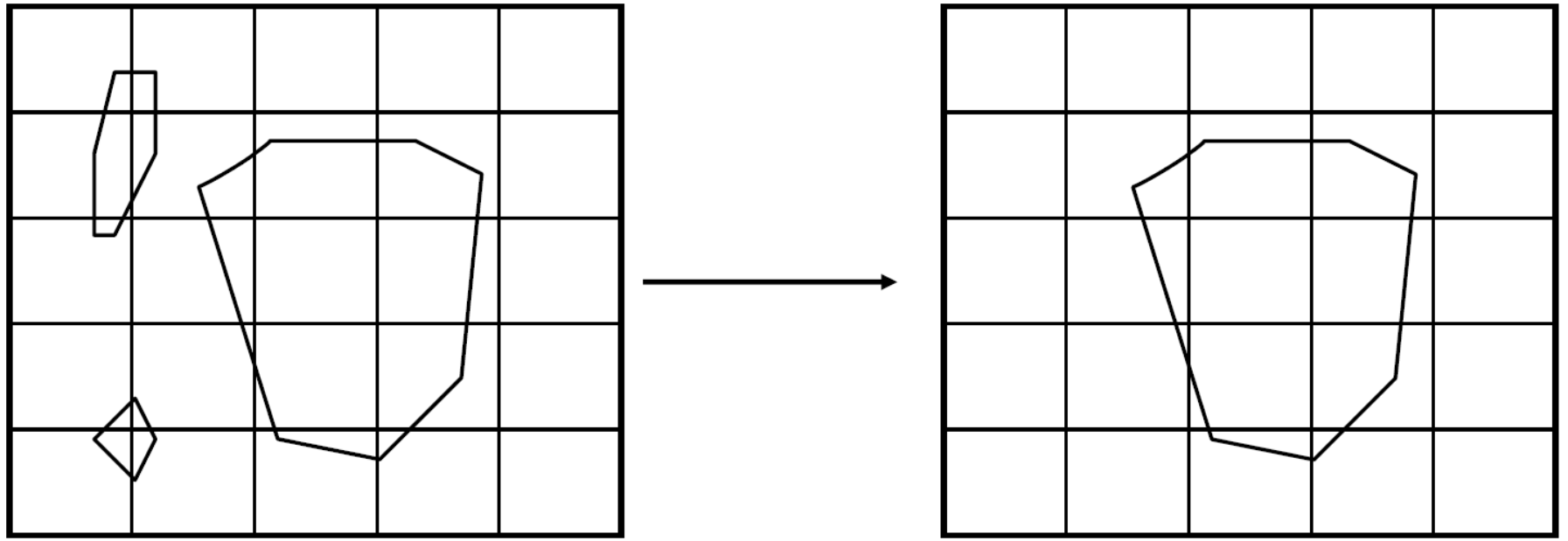


Problem – Detached Elements



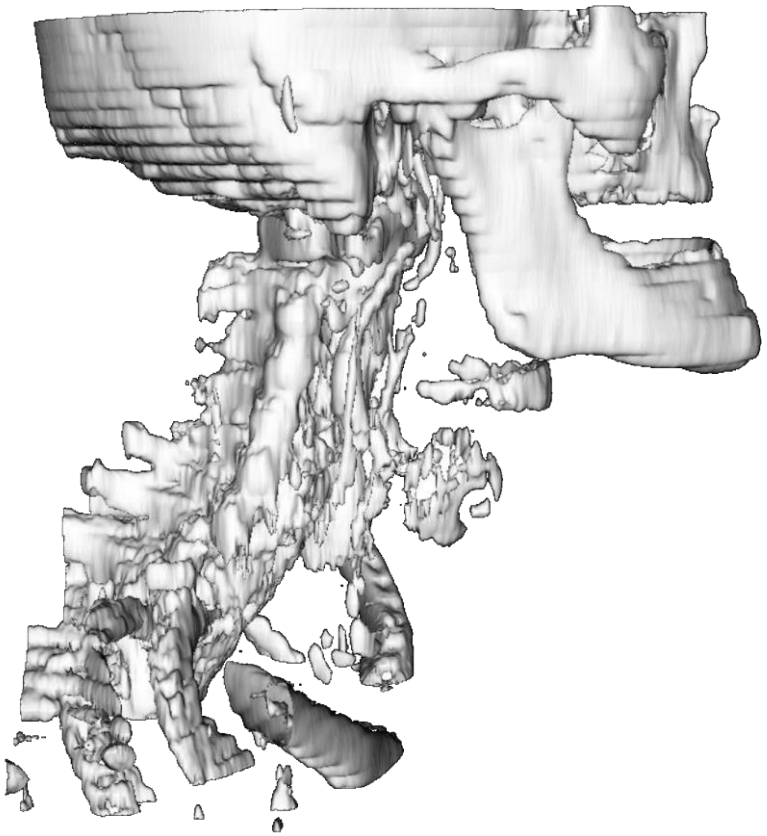
Connected Component Analysis

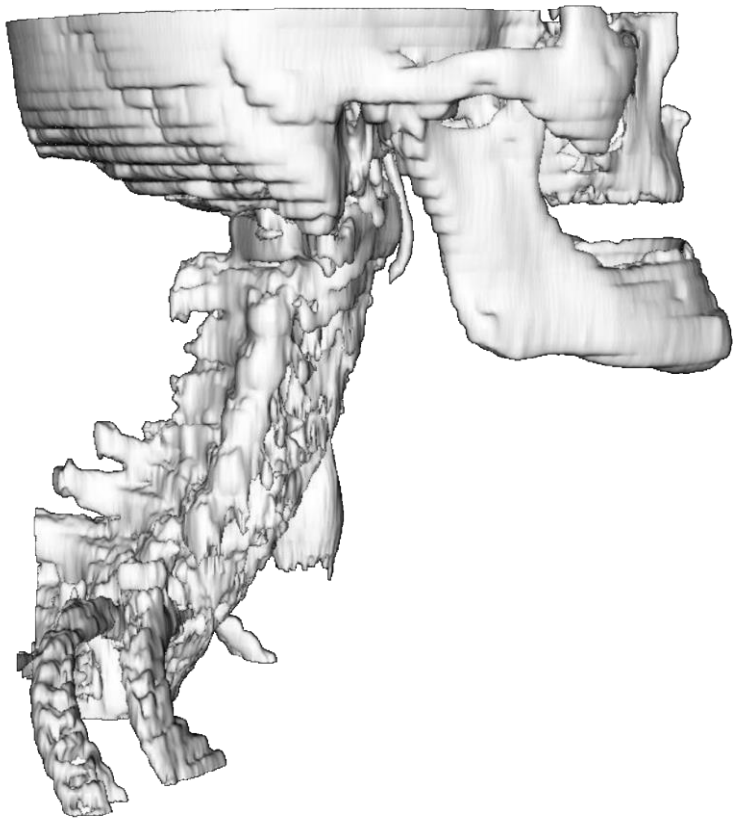
- Problem: Generation of many separated surfaces
- Goal: Limitation of surface extraction to the largest surface (or a certain quantity)
- Method: Connected Component Analysis (Schroeder et al. [1998])



Connected Component Analysis

- Depiction of the largest connected component





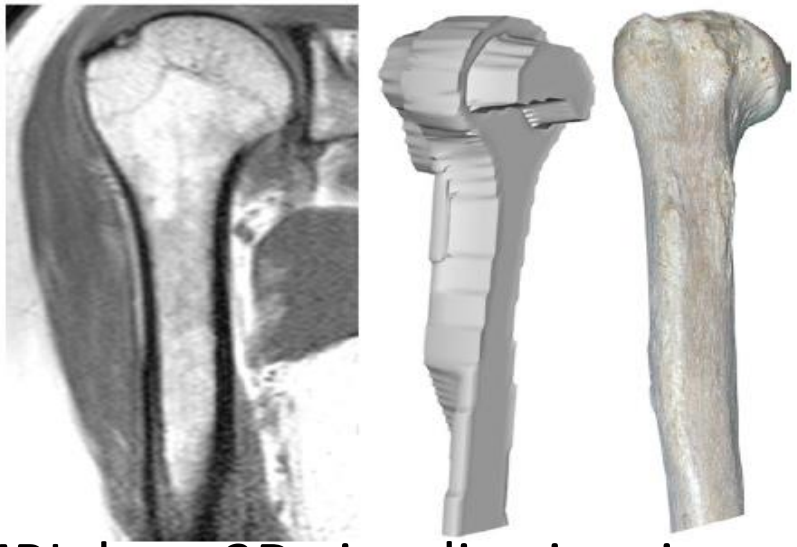
Connected Component Analysis

- Depiction of the largest connected component (cerebral and neck arteries).
- Limitation: target structure is not connected and a part of it is removed when only the largest component is shown.

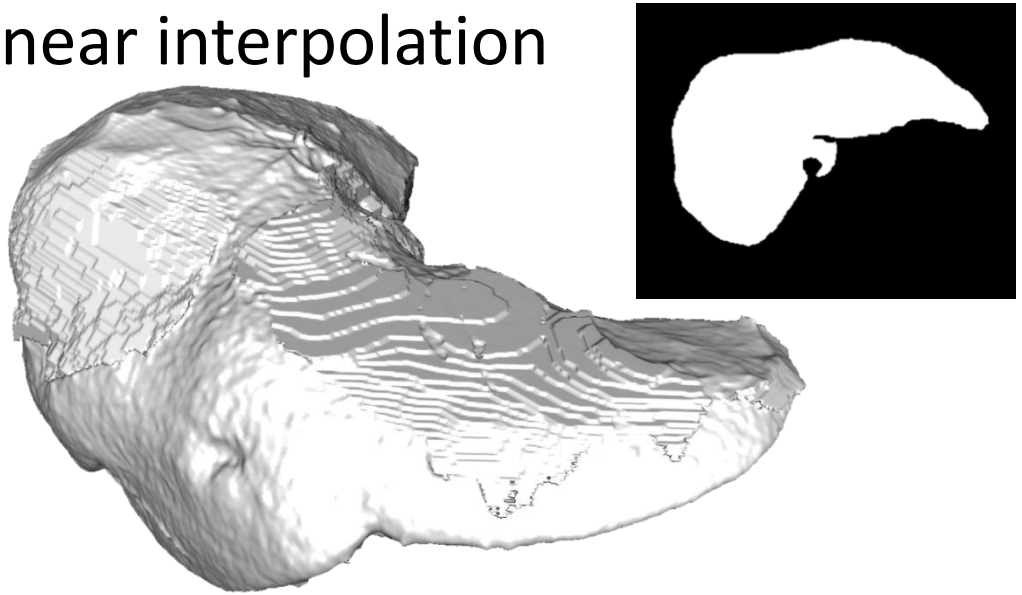
Smoothing of Polygonal Surfaces

Problem:

- Generation of surface models from segmentation results causes many artifacts, especially in case of strongly anisotropic data
- Marching Cubes is based on linear interpolation

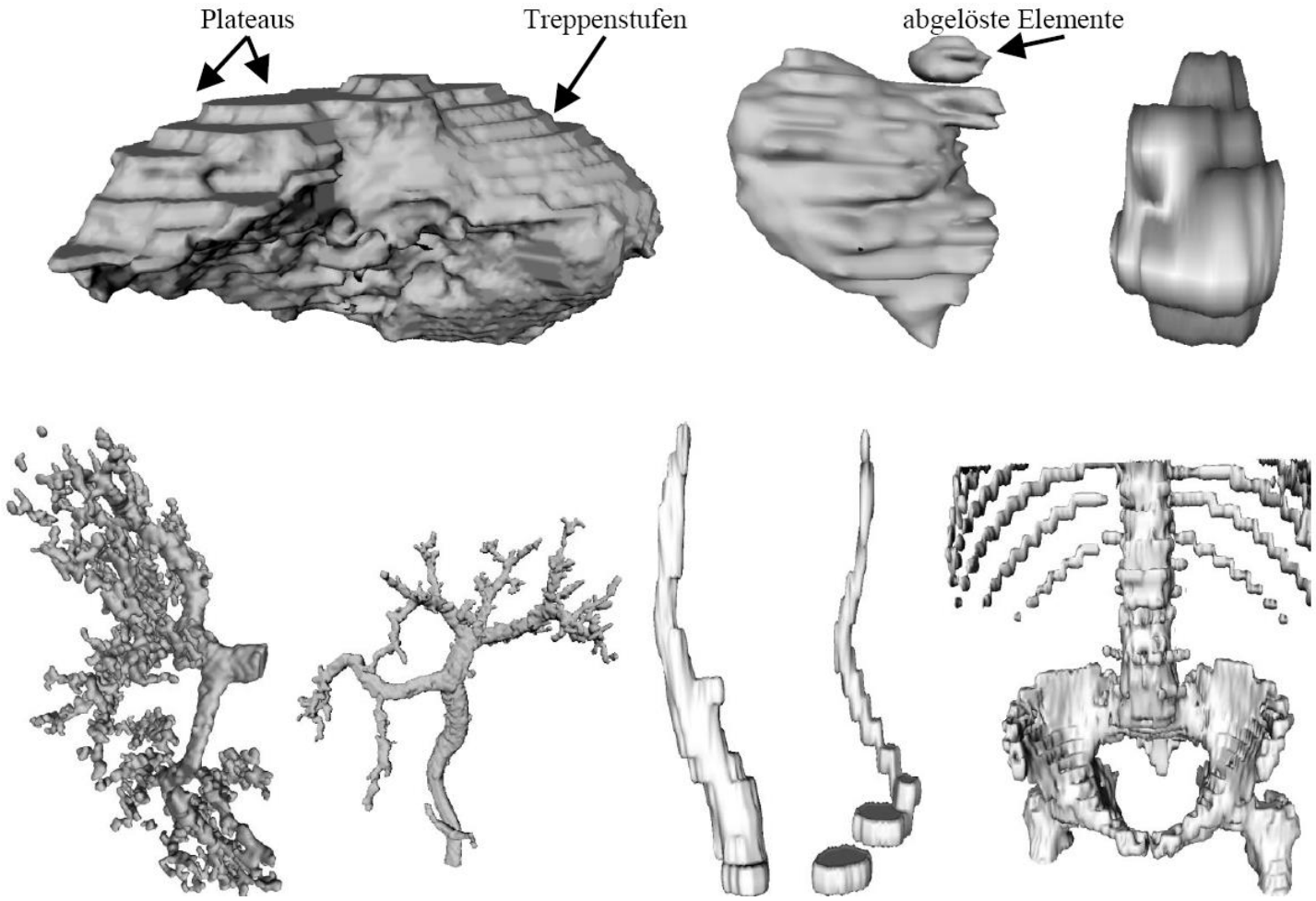


MRI data, 3D visualization, image



Liver, CT data (isotropic voxels)

Smoothing: Problem Statement



Smoothing of Polygonal Surfaces

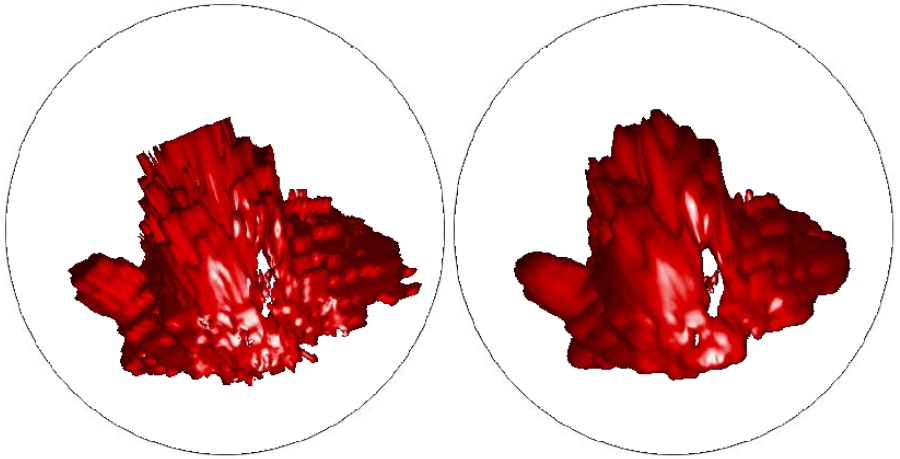
Smoothing of Image Data

- Smoothing of the segmentation result with morphological filters

0	0	0	0	0	0	0
0	0	0	100	100	100	0
0	100	100	100	100	100	0
0	100	100	100	100	100	0
0	0	0	100	100	100	0
0	0	0	0	0	0	0

→

0	6	26	33	33	26	6
20	33	54	66	66	54	26
33	66	66	100	100	66	33
33	66	66	100	100	66	33
20	33	54	66	66	54	26
0	6	26	33	33	26	6



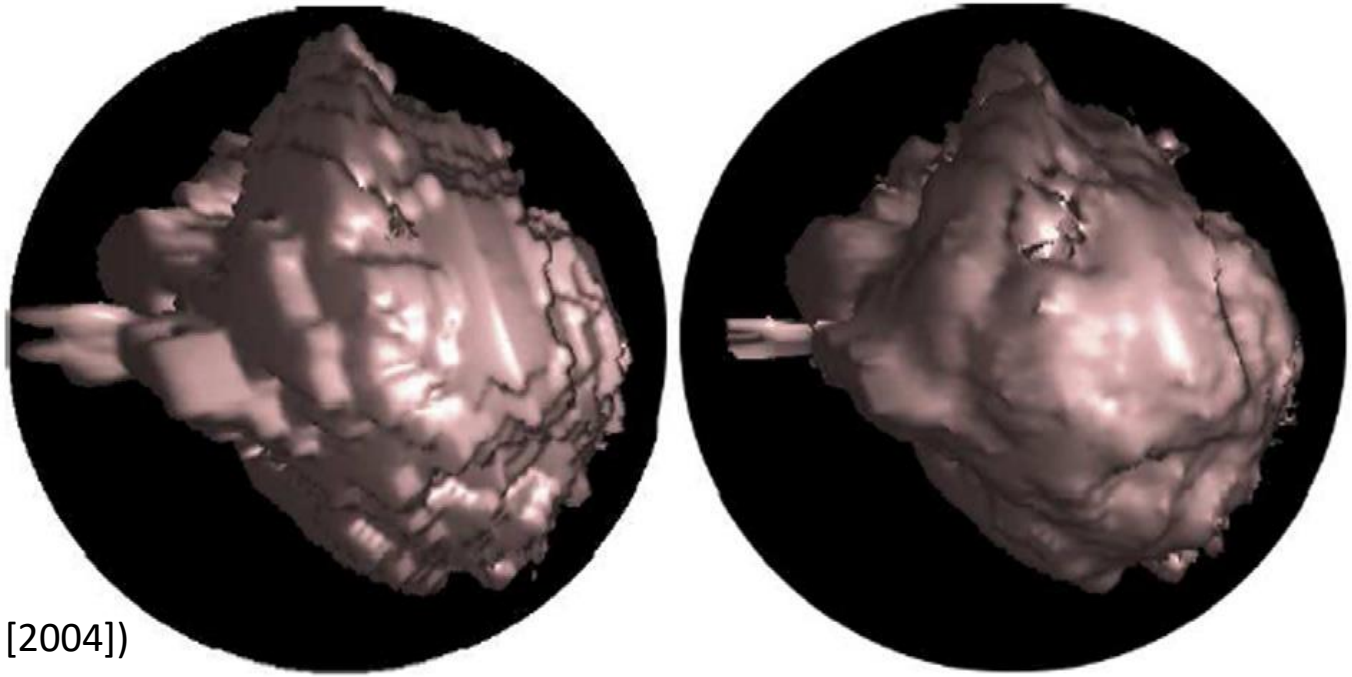
(From: Neubauer et al. [2004])

1st: Erosion and modification:

2nd: Dual dilatation and modification:

Smoothing of Image Data

- Algorithm yields a near-linear value degradation from the reference mask
- It retains the basic object shape and volume



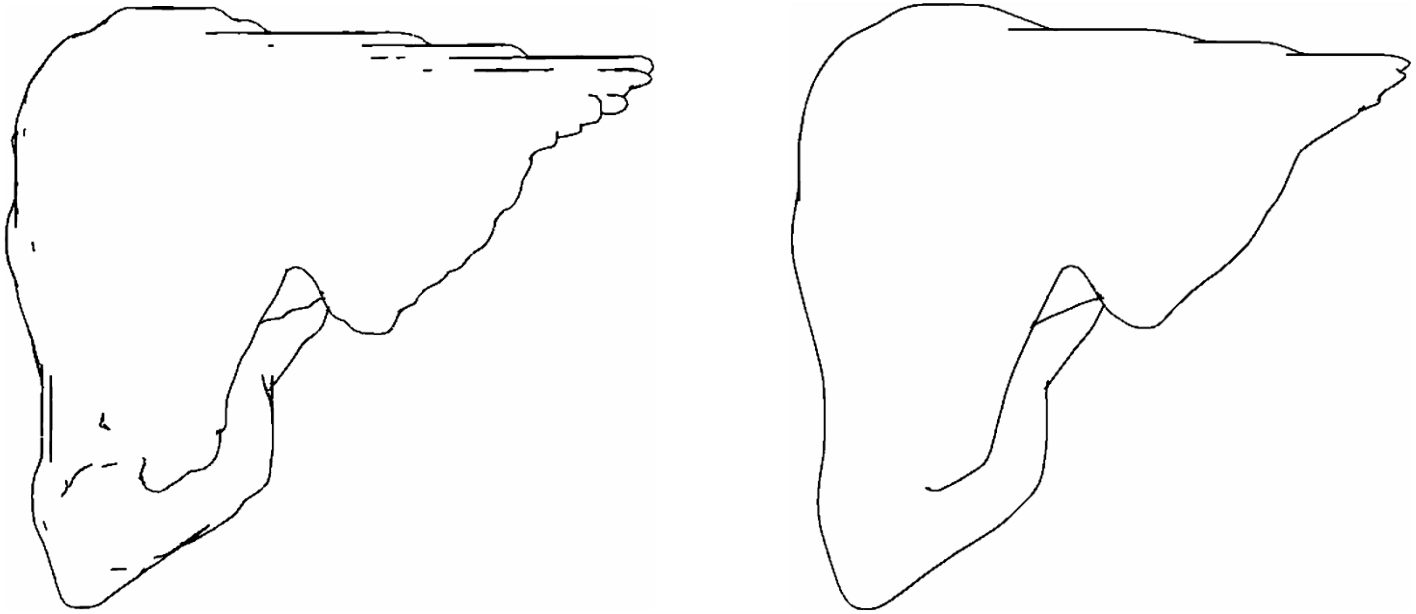
(From: Neubauer et al. [2004])

Smoothing of Polygonal Surfaces

- In the following standard smoothing algorithm
- These algorithms does neither change the number of vertices nor the topology
- Only vertex coordinates are changes

Smoothing of Polygonal Surfaces

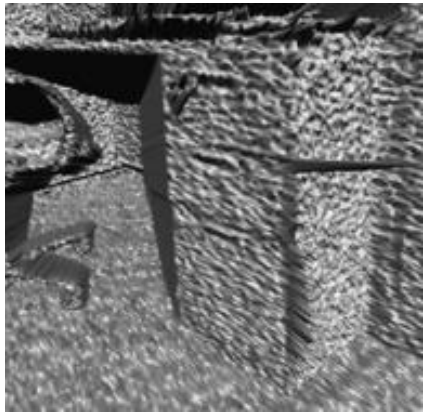
- Illustrative techniques (such as silhouettes) highlight these artifacts

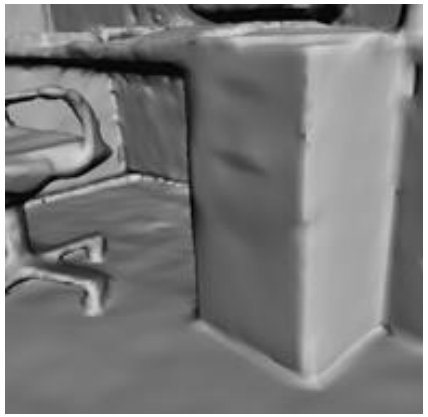


3D Studio, Relaxation filter, 7 iterations, relaxation factor 1.0

Smoothing of Surfaces

- Many different methods available
- Clear application orientation in the field of CAD and the smoothing of models that were acquired with laser scanners
  - Preservation of sharp (rectangular) edges with preferably optimum smoothing of planar areas
- Medical surface models: nearly no sharp edges, some curvatures change very fast, "large" models





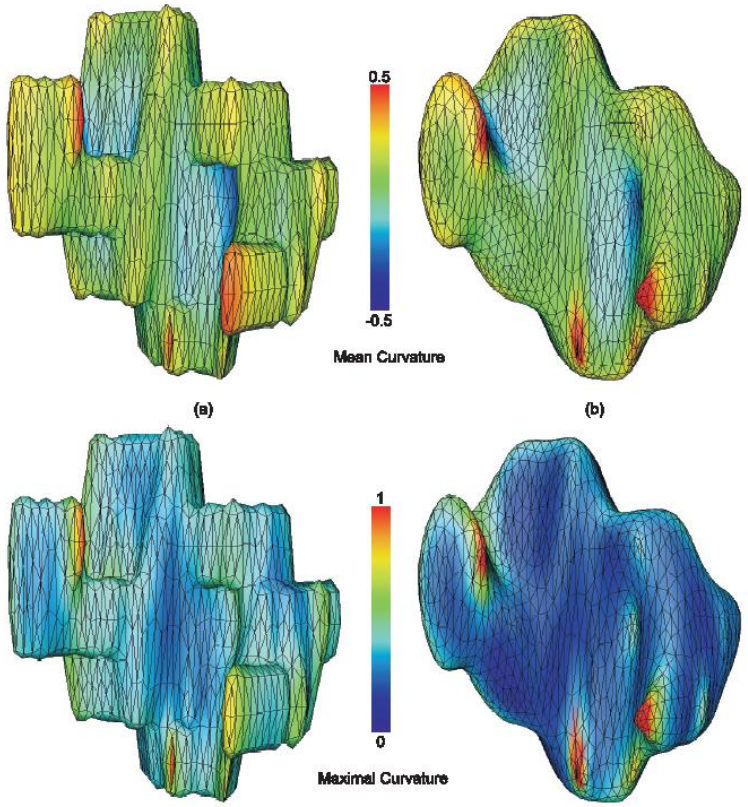
Source:
Tasdizen & Whitaker 2003

Requirements

- Analogous to smoothing of image data:
  - Elimination of high-frequent noise while maintaining features
  - Measures/evaluation:
    - Curvature plots, total curvature (mean or Gaussian curvature)
    - low distances between original surface and smoothed surface
    - volume preservation
  - Speed
  - Accuracy

Costly Approaches

- Smoothing as an optimization problem
- Possible criteria:
  - Minimization of total curvature
  - Minimization of total surface
- Most important publications:
  - Witkin and Welch (1994)
  - Kobbelt et al. (1997)



Iterative Mesh Smoothing – Basics

Iterative Mesh Smoothing

- Based on diffusion flow (e.g., heat diffusion)

$$\frac{\partial f(\mathbf{x}_i, t)}{\partial t} = \lambda \Delta f(\mathbf{x}_i, t)$$

- 2nd order linear partial differential equation

Iterative Mesh Smoothing

- Based on diffusion flow (e.g., heat diffusion)

$$\frac{\partial f(\mathbf{x}_i, t)}{\partial t} = \lambda \Delta f(\mathbf{x}_i, t)$$

- 2nd order linear partial differential equation
- Examples: Blurring images

Iterative Mesh Smoothing

- Assume the time axis is divided into regular intervals of size h

$$f(\mathbf{x}_i, t+h) = f(\mathbf{x}_i, t) + h\lambda\Delta f(\mathbf{x}_i, t)$$

Iterative Mesh Smoothing

- Assume the time axis is divided into regular intervals of size h

$$f(\mathbf{x}_i, t+h) = f(\mathbf{x}_i, t) + h\lambda \underbrace{\Delta f(\mathbf{x}_i, t)}_{???}$$

Discrete Laplace–Beltrami Operator

- The operator can be presented as:

$$\Delta f(\mathbf{p}_i) = \sum_j w_{ij}(f(\mathbf{p}_j) - f(\mathbf{p}_i))$$

- Different weights give different Laplace-Beltrami operators on meshes

Discrete Laplace–Beltrami Operator

- Combinatorial

$$w_{ij} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

Discrete Laplace–Beltrami Operator

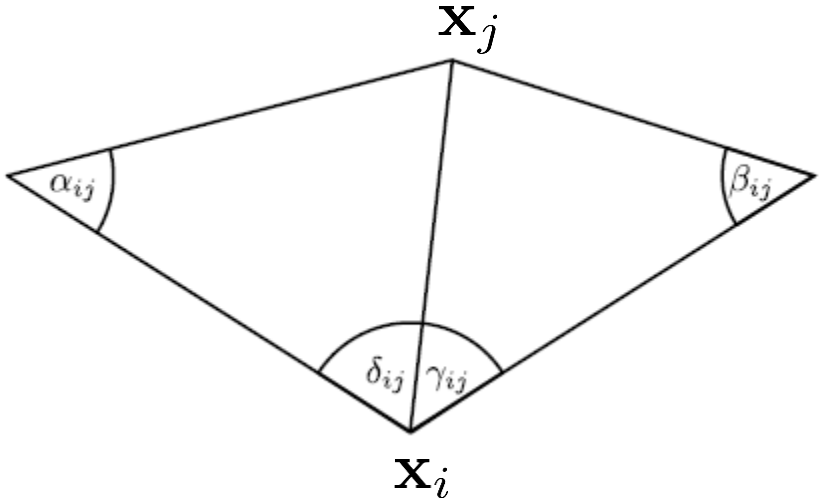
- Uniform/Laplace

$$w_{ij} = \begin{cases} \frac{1}{N(i)}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

Discrete Laplace–Beltrami Operator

- Floater's mean value

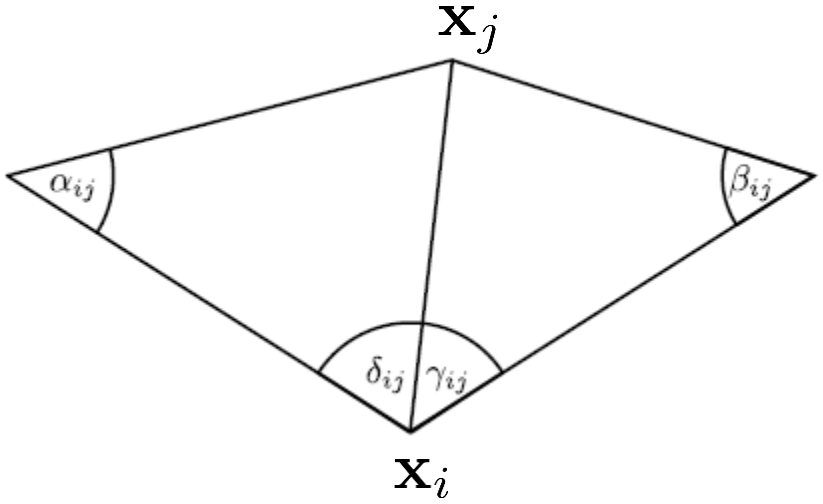
$$w_{ij} = \begin{cases} \frac{\tan(\delta_{ij}/2)+\tan(\gamma_{ij}/2)}{|\mathbf{x}_i - \mathbf{x}_j|}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$



Discrete Laplace–Beltrami Operator

- Cotangent weights

$$w_{ij} = \begin{cases} \frac{\cot(\alpha_{ij})+\cot(\beta_{ij})}{2}, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$



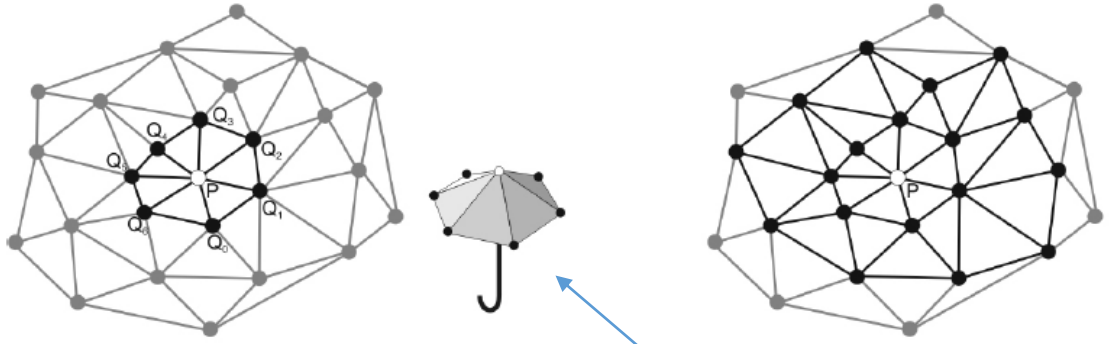
Iterative Mesh Smoothing

- Smoothing

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \lambda \sum_j w_{ij}(\mathbf{x}_j - \mathbf{x}_i)$$

Iterative Mesh Smoothing

- Iterate over all vertices and replace them by a weighted average of the former value and the surrounding vertices.
- Which surrounding?
  - Vertices in a certain distance (*Eucl. distance*)
  - Vertices that are connected to the current vertex (directly or via a path of length *n*) (*topological distance*)
  - Typical choice: Vertices in topological distance of 1 or 2



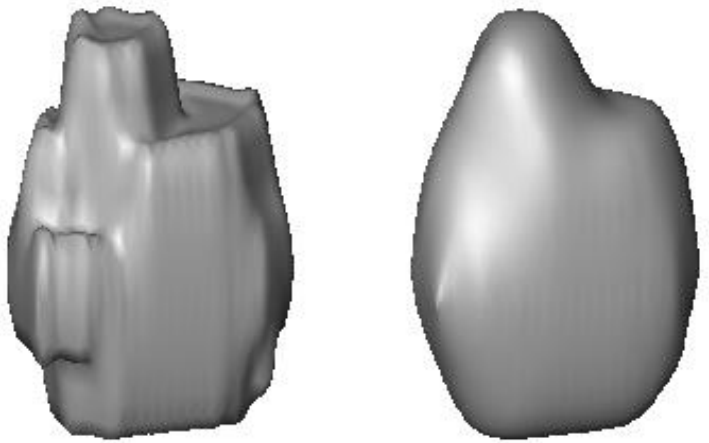
Iterative Mesh Smoothing

Laplace Smoothing

- Vertices in topological distance of 1
- Parameter: smoothing factor λ (0< λ<1) and number of iterations

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \lambda \sum_j \frac{1}{N(i)} (\mathbf{x}_j - \mathbf{x}_i)$$

- Smoothing of a lymphnode model with 0.5 as smoothing factor and 20 iterations

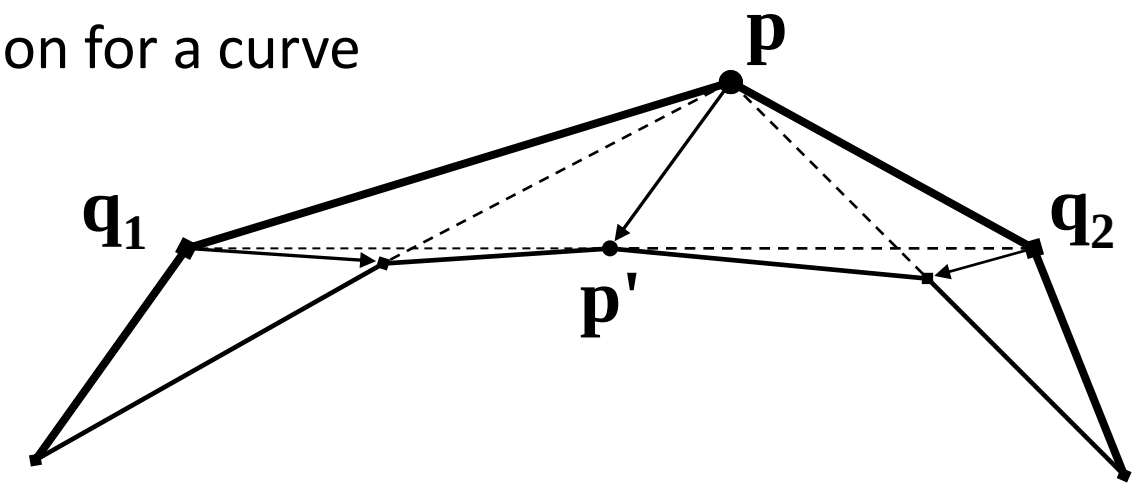


Original
V=100%

Laplace
V=80,8%

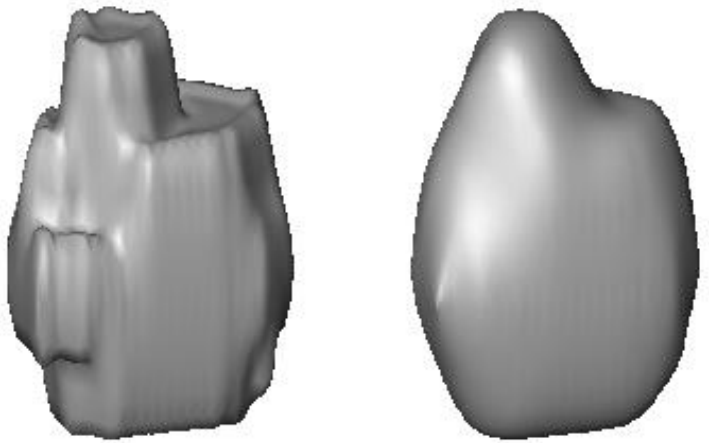
Laplace Smoothing

- Basic algorithms:
  - Iterative smoothing: Knots are moved towards the center of the neighboring knots
  - Illustration for a curve



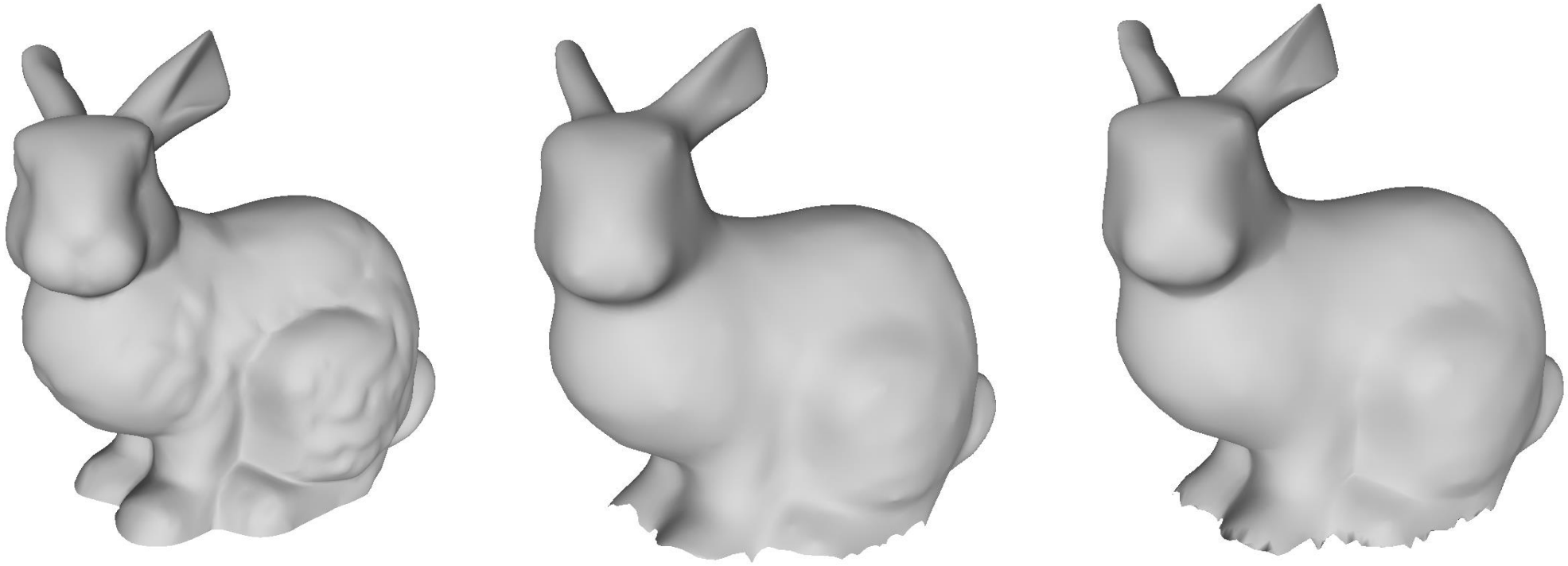
Laplace Smoothing

- Results in a strong (uncontrolled) shrinking and often reaches the desired smoothing only through "total smoothing" of smaller characteristics



Laplace/Cotangent Smoothing

- Uniform/Laplace vs. Cotangent



Original Laplace Cotangent

Laplace Smoothing

Discussion:

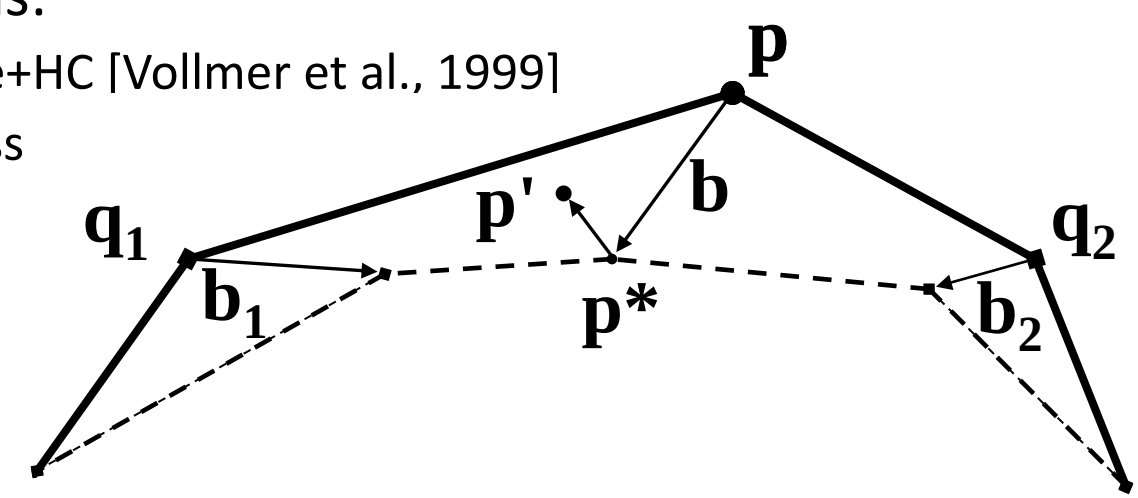
- Simple and effective method with two parameters
- Efficient implementations possible, including GPU-based techniques
- May lead to loss of features and volume loss
- Mesh quality may be degraded in particular in concave regions

Refinements:

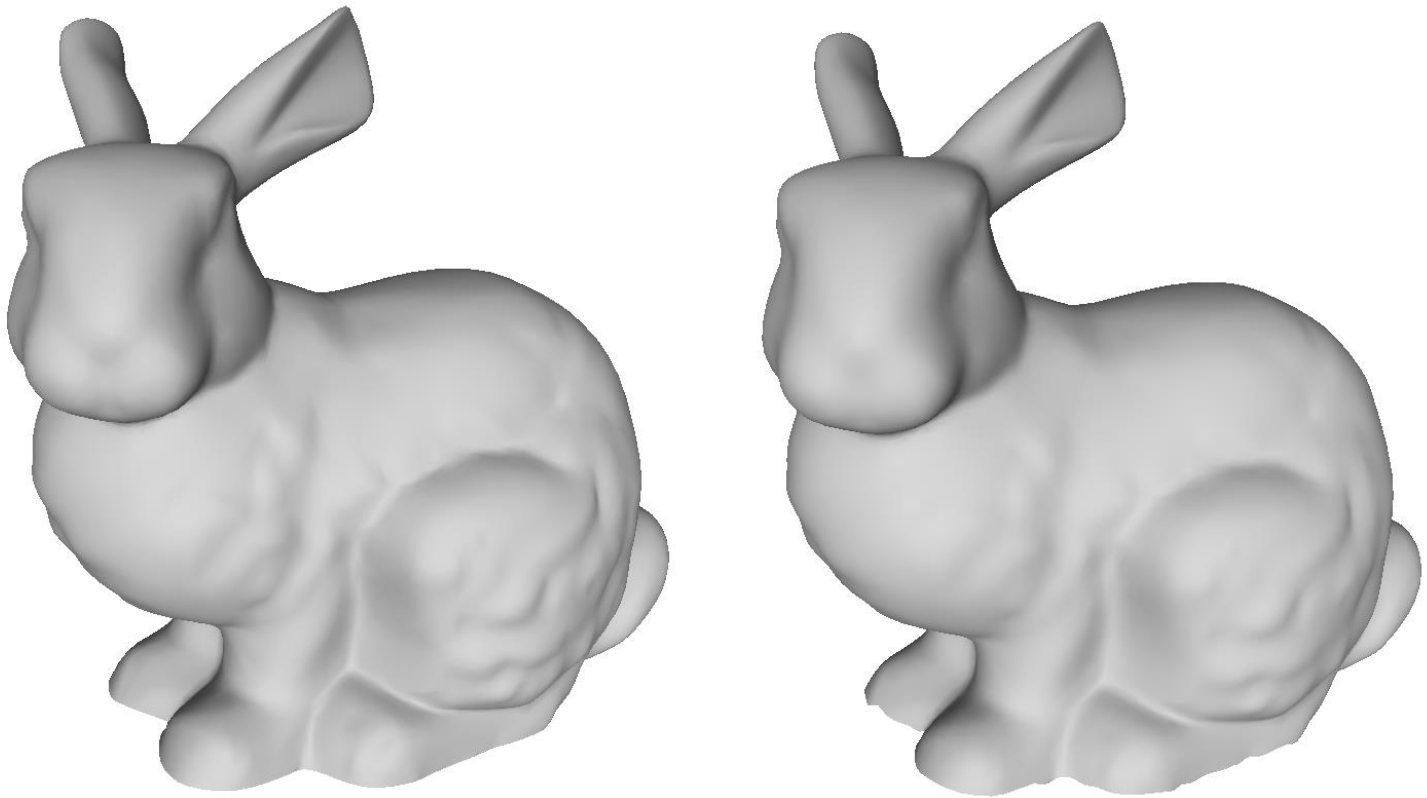
- Correction steps to improve accuracy (next slide)
- Smart Laplacian smoothing: Combination with mesh optimization (meshes are simultaneously reduced and element quality is ensured)
- (Owen, 1998, Canann, 1998)
  - Repositioning of nodes only when element quality does not suffer

Laplace Smoothing with Correction

- Correction of Laplace smoothing:
  - In each step, modified knots are back-shifted by a certain value
- Additional parameters:
  - How strong is the shifting towards the original point?
  - How is the shifting of the neighbors incorporated?
- Algorithms:
  - Laplace+HC [Vollmer et al., 1999]
  - LowPass



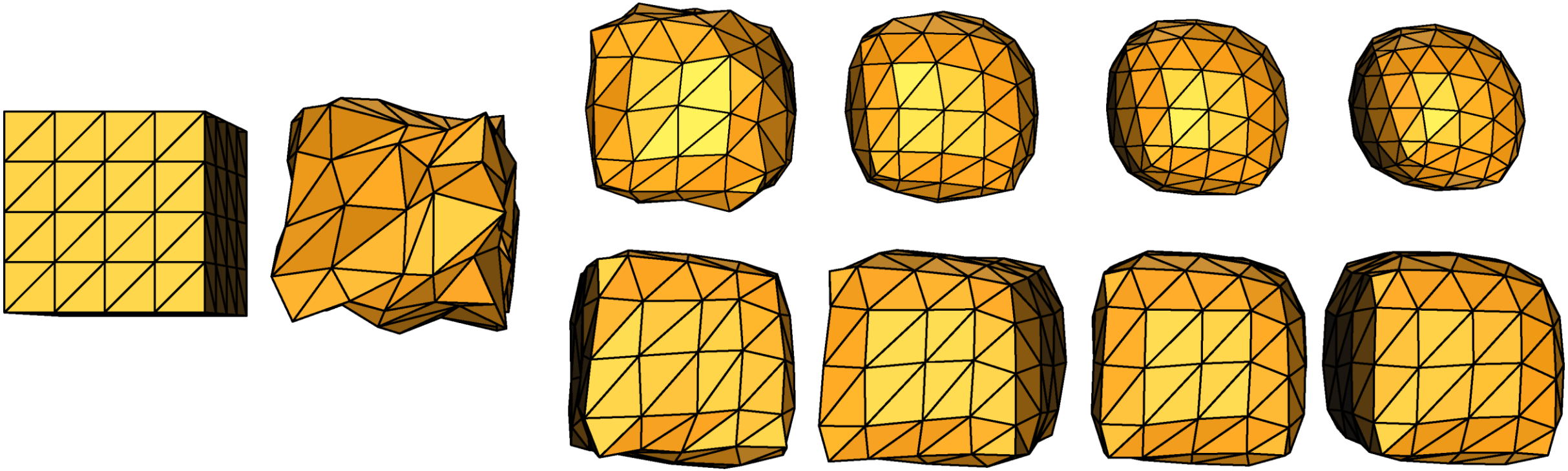
Laplace+HC



Original

Laplace+HC

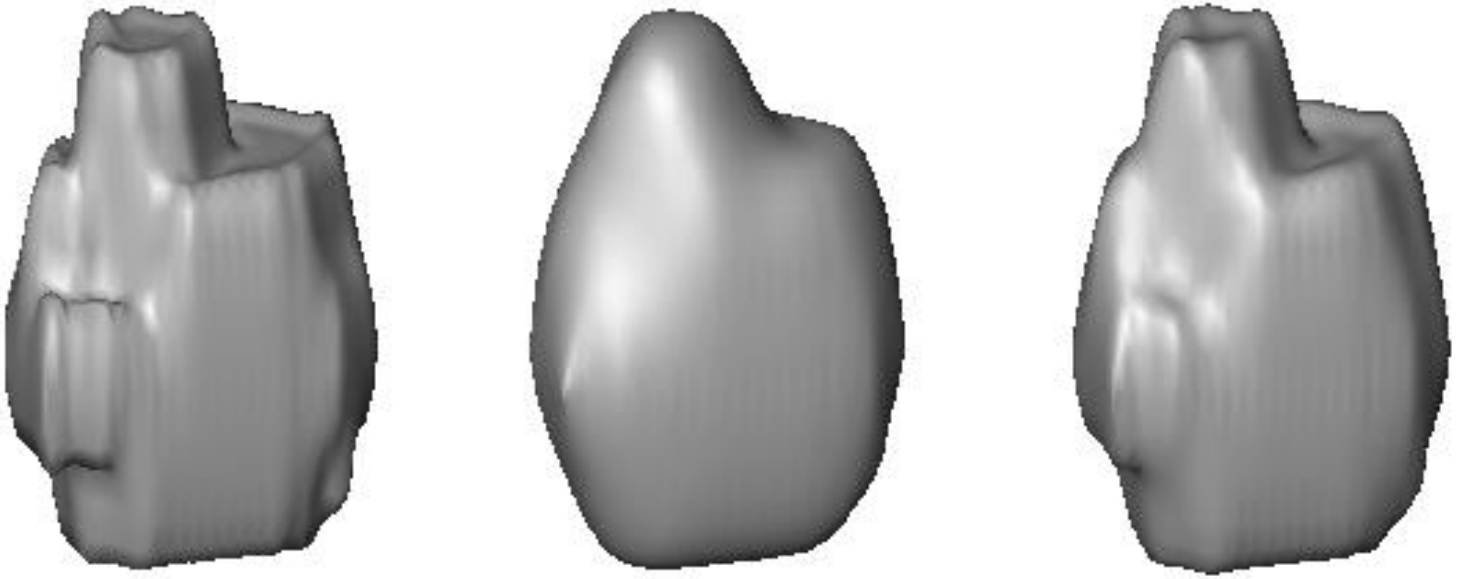
Laplace+HC



[Vollmer et al., 1999]

Laplace+HC

- Comparison: Laplace smoothing of a lymph node with and without correction



Images: Ragnar Bade

LowPass Filtering

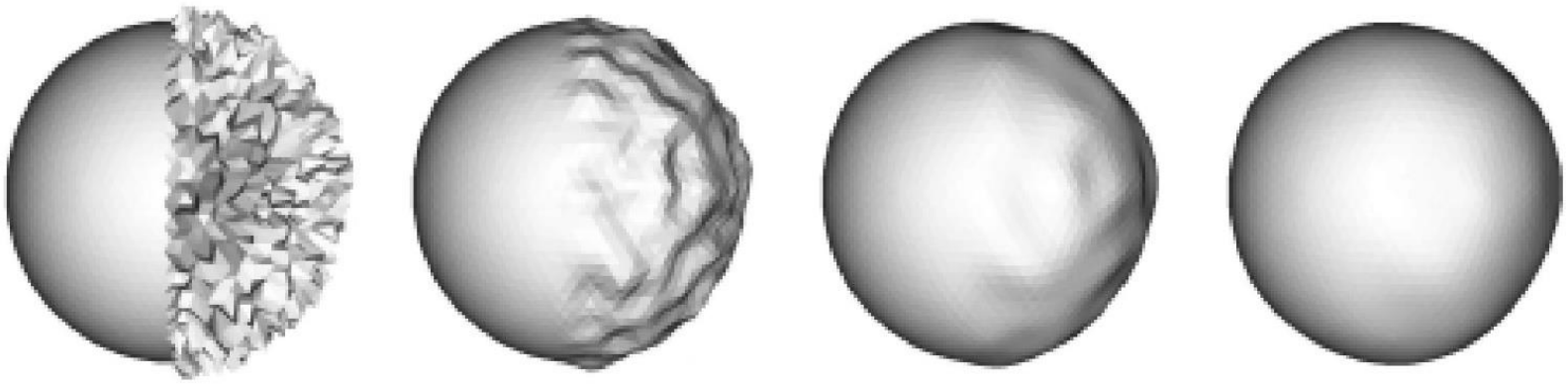
- Implementation of two Laplace-like filterings with factors λ>0 and μ<0

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \lambda \sum_j \frac{1}{N(i)} (\mathbf{x}_j - \mathbf{x}_i)$$ Shrink

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mu \sum_j \frac{1}{N(i)} (\mathbf{x}_j - \mathbf{x}_i)$$ Inflate

LowPass Filtering

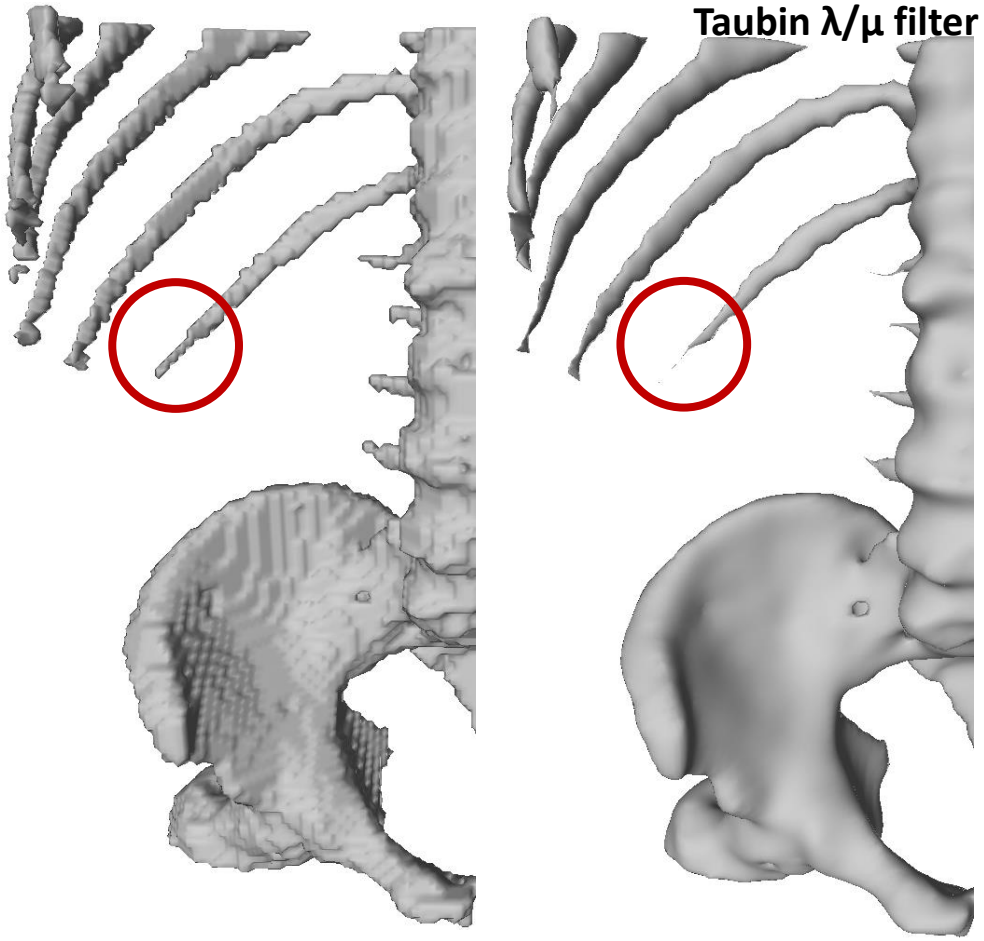
- Filtering: usually 1/n (all neighbors have the same influence, such as with Laplace)
    - Modifications include, e.g., the inner angles of the adjacent triangles
- Default: $\mu = -1.02\ \lambda$ (Taubin, 1995)



[Taubin, 1995]

LowPass Filtering

- General problem (also during filtering with correction):
  - Preservation of small details cannot be guaranteed!



Images: Ragnar Bade

Comparisons

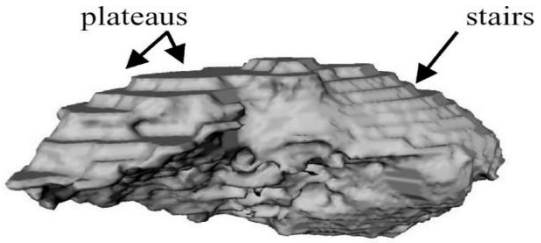
Comparison of Methods

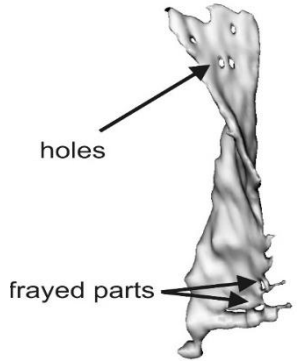
- Classification - Different objects:
  - possess varying artifacts
  - different behavior during smoothing
  - require different strategies (accuracy requirements)

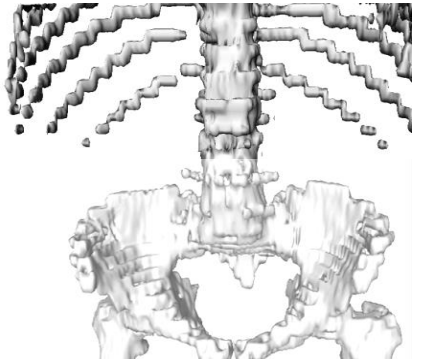
compact

flat

elongated

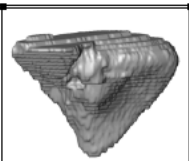
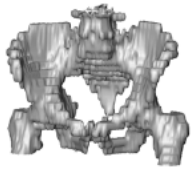
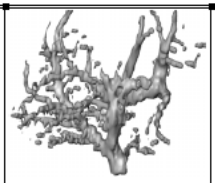






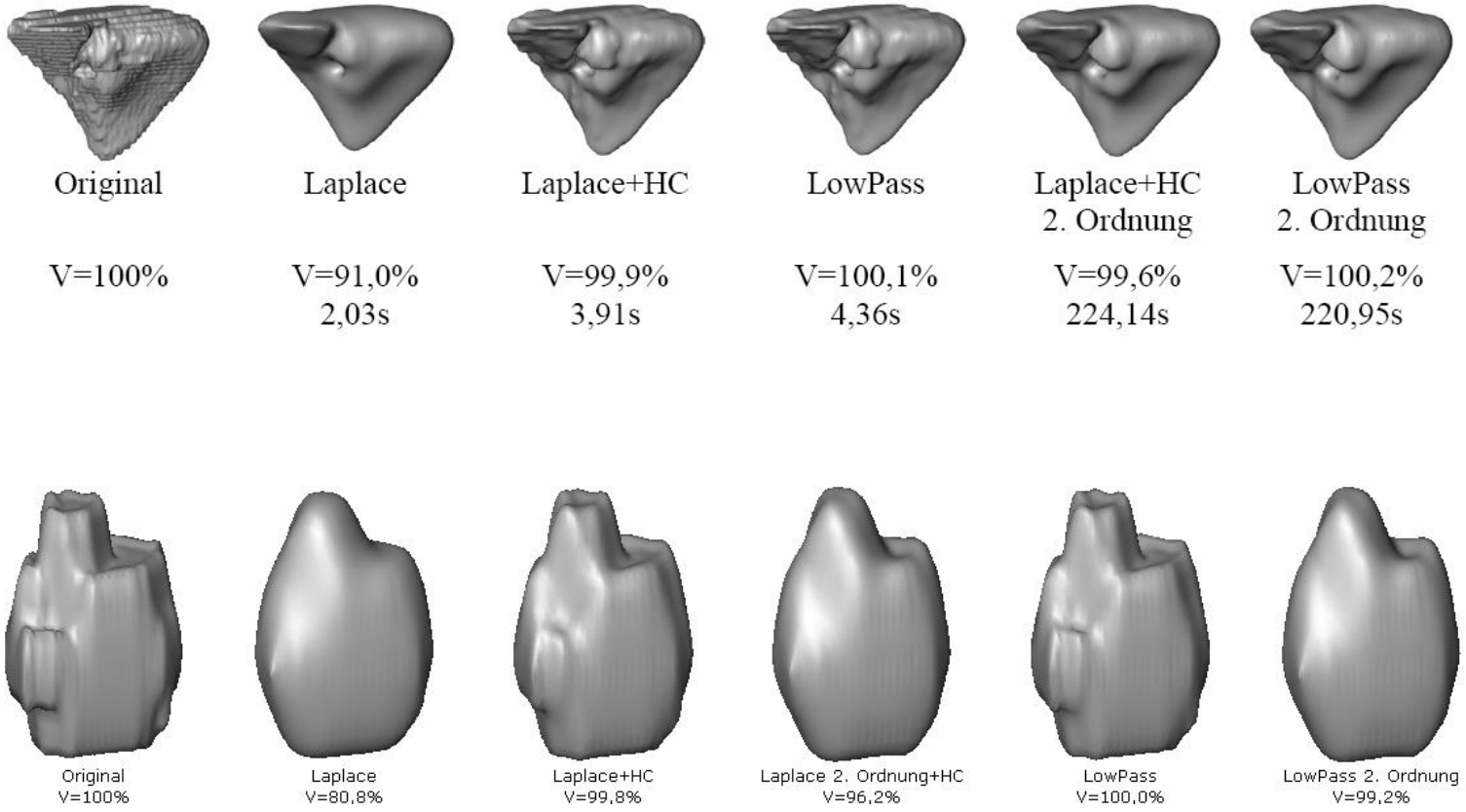
Comparison of Methods

- Criteria: Quality, volume preservation (run time)
- Methods/parameters:
  - Laplace, Laplace with correction, LowPass
  - Different iteration steps: 5, 10, 20, 50
  - Different weighting factors: 0.05, 01., 0.3, 0.5, 0.7, 0.9
  - Different neighborhood: 1, 2 (topological)

	 Leber	 Lymphknoten	 Kopfwendemuskel	 Beckenknochen	 Gefäßbaum	 Halsschlagader
Faces	37.148	3.412	9.616	53.930	23.236	1.956
Vertices	18.576	1.708	4.804	27.211	11.820	982
Voxel	1.696.250	1.664	101.035	430.318	96.807	16.404

Comparison of Methods

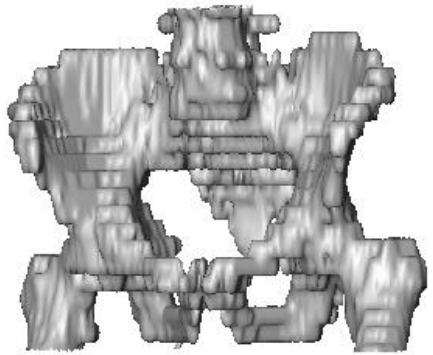
- Smoothing factor: 0.5, 20 iterations



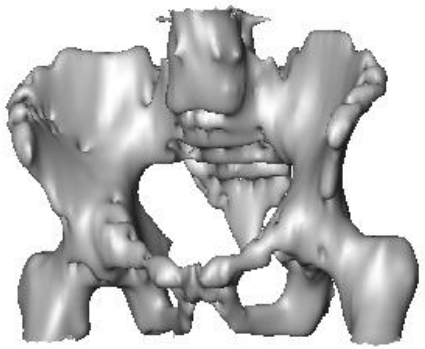
Images: Ragnar Bade

Comparison of Methods

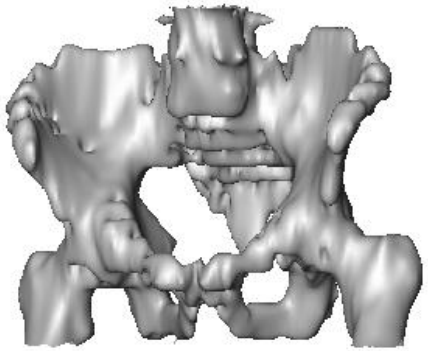
- All images with smoothing factor 0.5 and 10 iterations



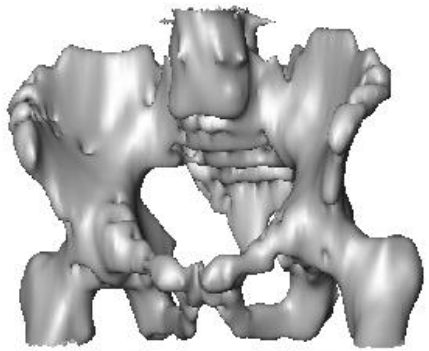
Original
V=100%



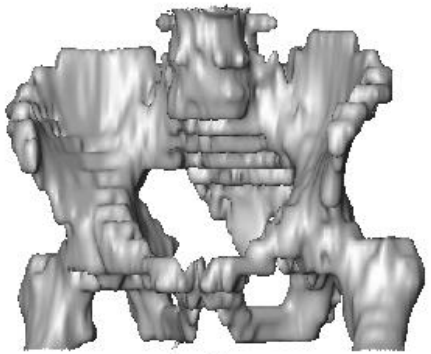
Laplace
V=94,2%



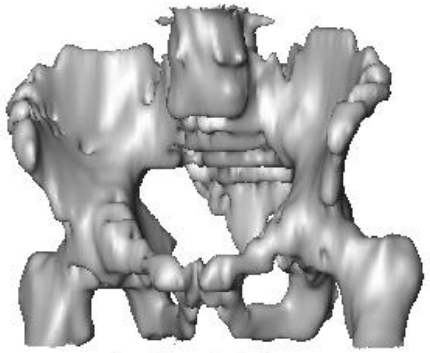
Laplace+HC
V=99,7%



Laplace 2. Ordnung+HC
V=98,9%



LowPass
V=100,4%

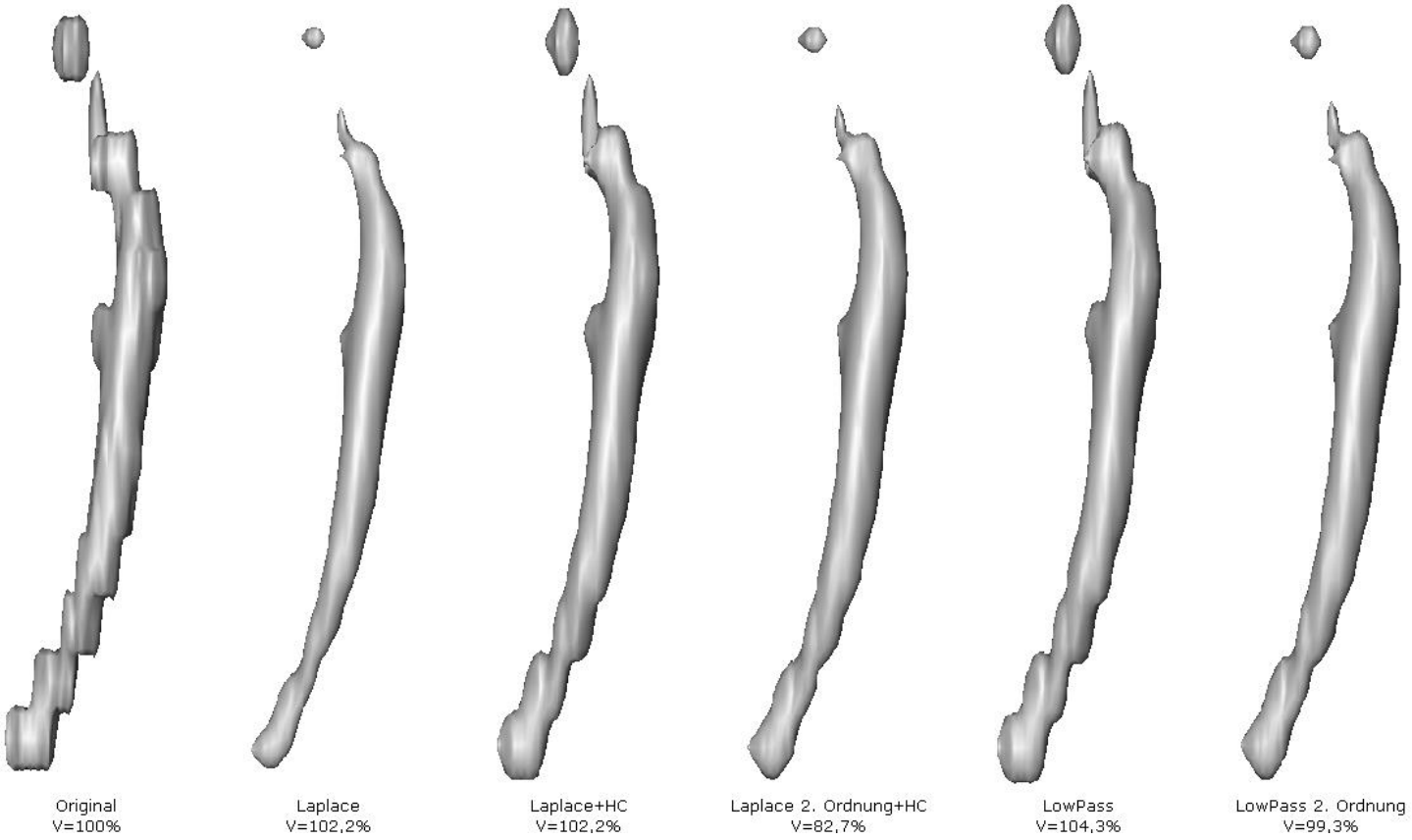


LowPass 2. Ordnung
V=100,1%

Images: Ragnar Bade

Comparison of Methods

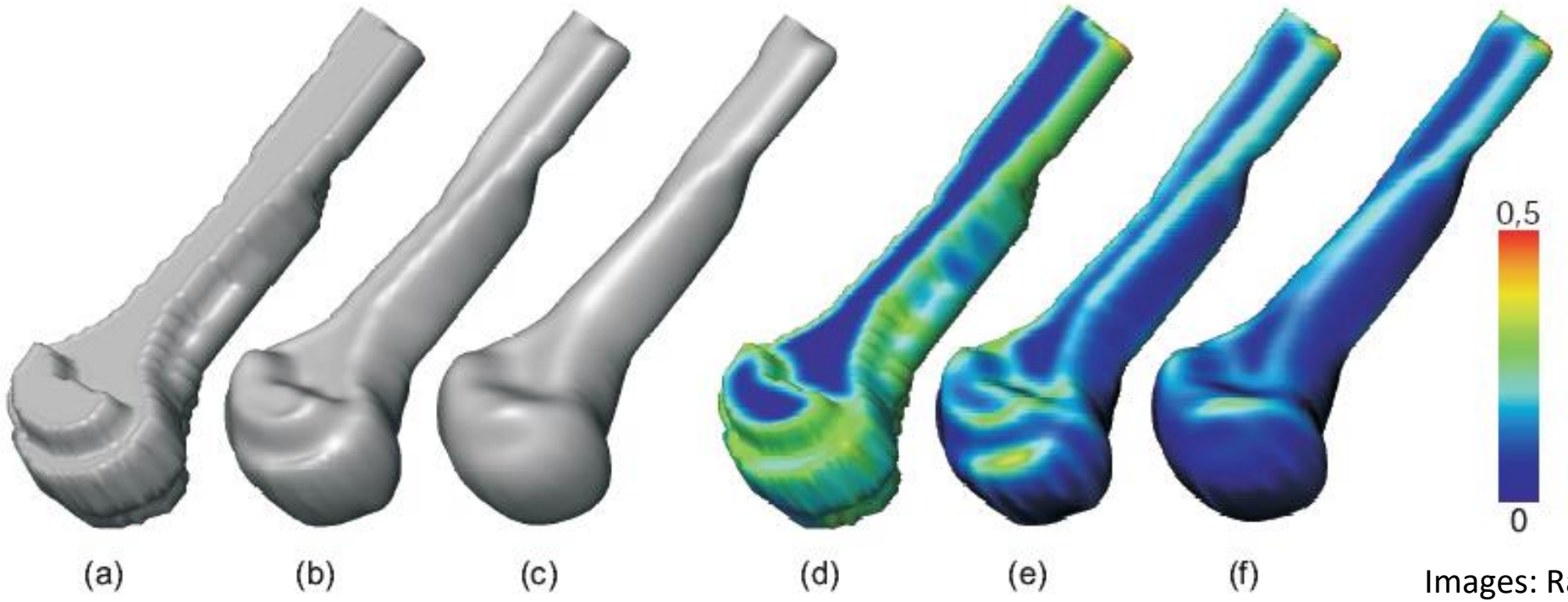
- Carotid artery: smoothing factor: 0.7 and 10 iterations



Images: Ragnar Bade

Comparison of Methods

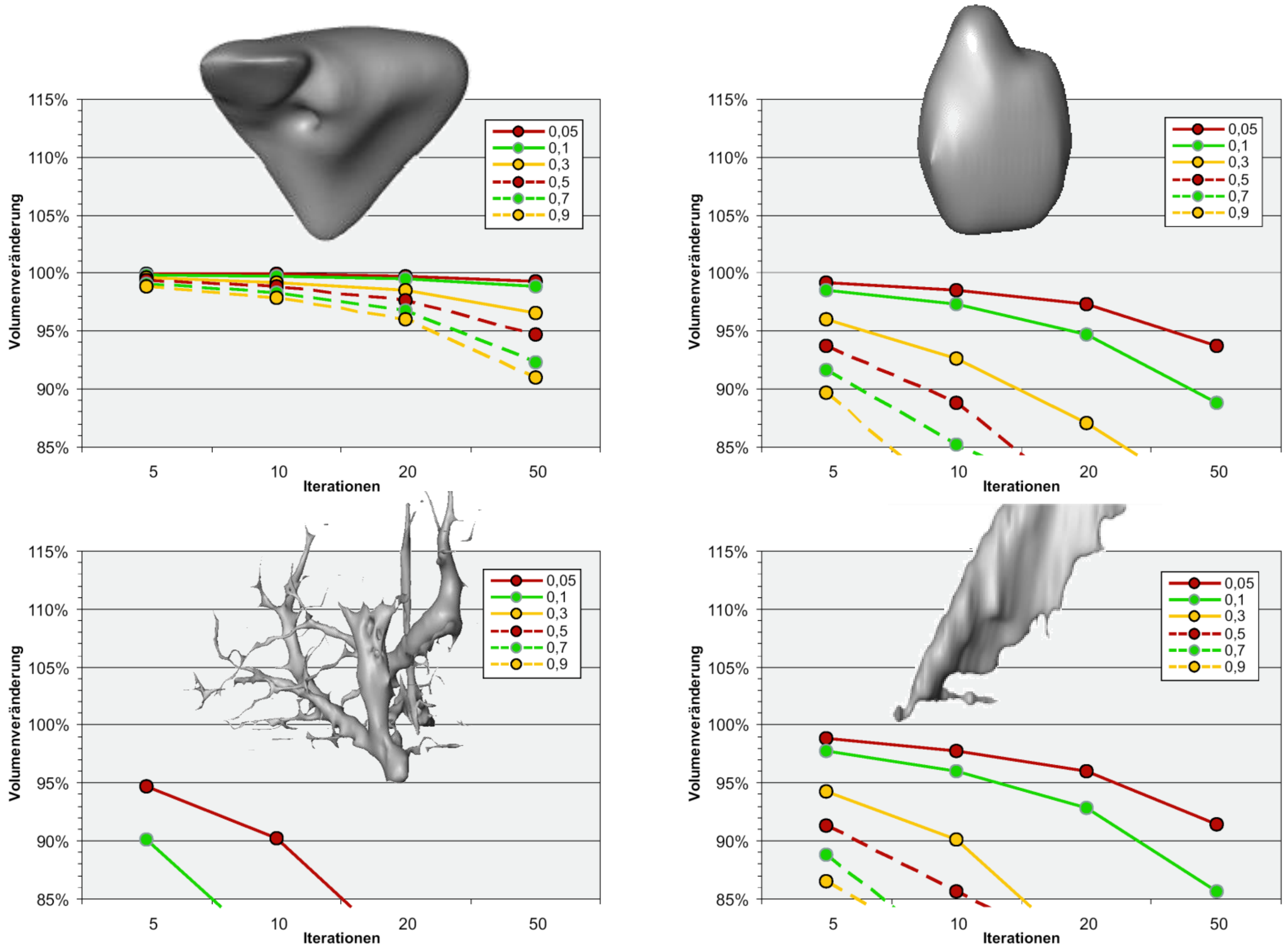
• Original, Lowpass filtering with 1th neighborhood, with extended neighborhood, and corresponding curvature values.



Images: Ragnar Bade

Comparison of Methods

- Laplace:



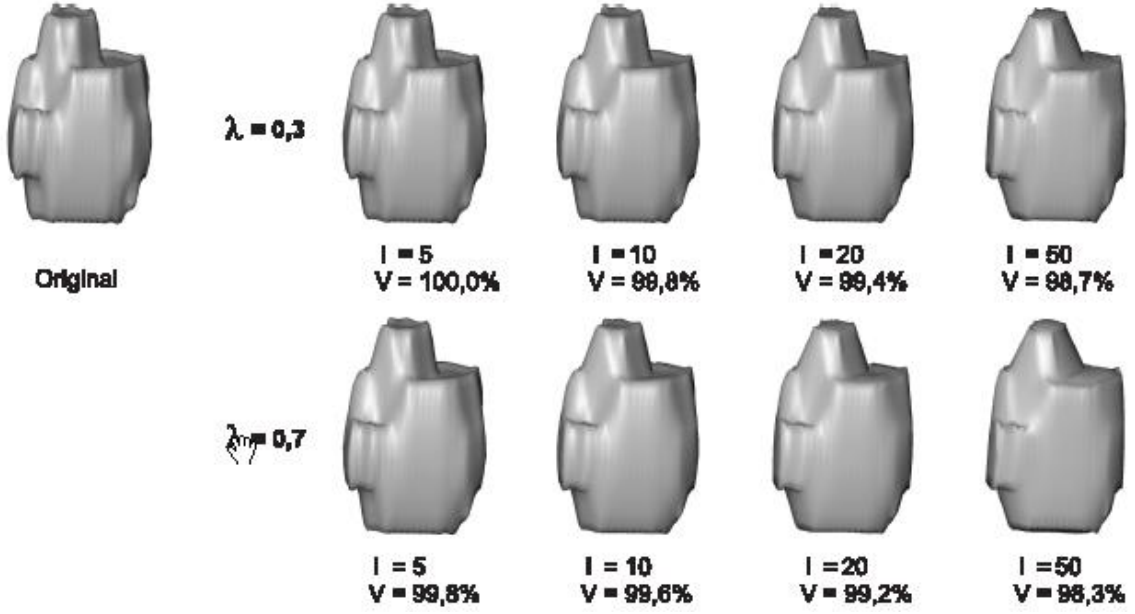
Images: Ragnar Bade

Further Smoothing Methods

- Mean value filtering
  - Replaces the surface normal for the current point by the average normal of the considered surrounding and adjusts the coordinates
- Median filtering
  - Sorts the normals in the neighborhood according to their angle to the current normal and replaces the current normal by the normal that lies in the middle (after sorting)
- Mean curvature flow
  - Uses curvature measures for modification of edge points

Further Smoothing Methods

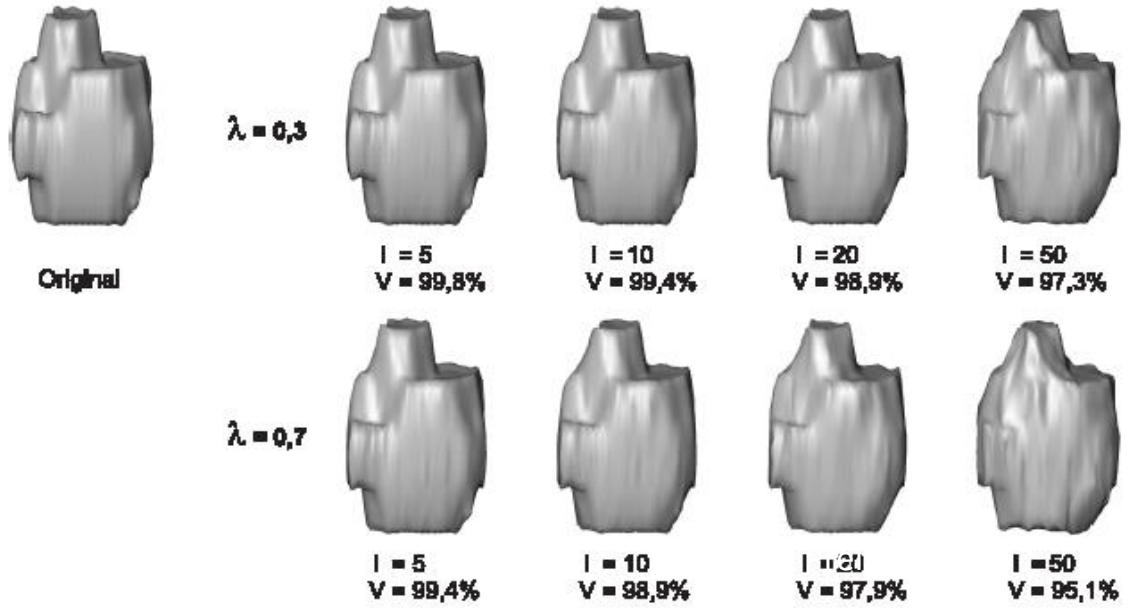
- Mean value filtering:
  - Good volume preservation
  - Good smoothing for large and compact objects
  - New artifacts (sharp edges) occur in smaller objects, especially in case of a high factor and many iterations



Images: Ragnar Bade

Further Smoothing Methods

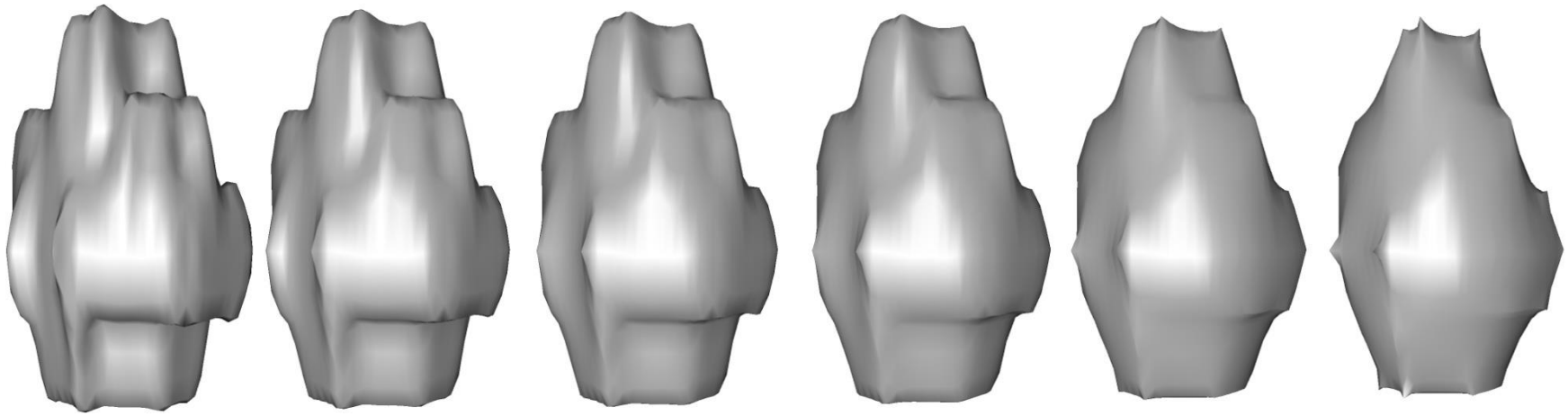
- Median filtering:
  - Main application: Preservation of sharp edges in noisy data
  - Similar problems like with mean value filter, partially even more distinctive



Images: Ragnar Bade

Further Smoothing Methods

- Mean Curvature Flow:
  - Especially difficult for compact objects with stronger smoothing factors



Images: Ragnar Bade

Original and smoothing after 5, 10, 20, 50 and 100 iterations.

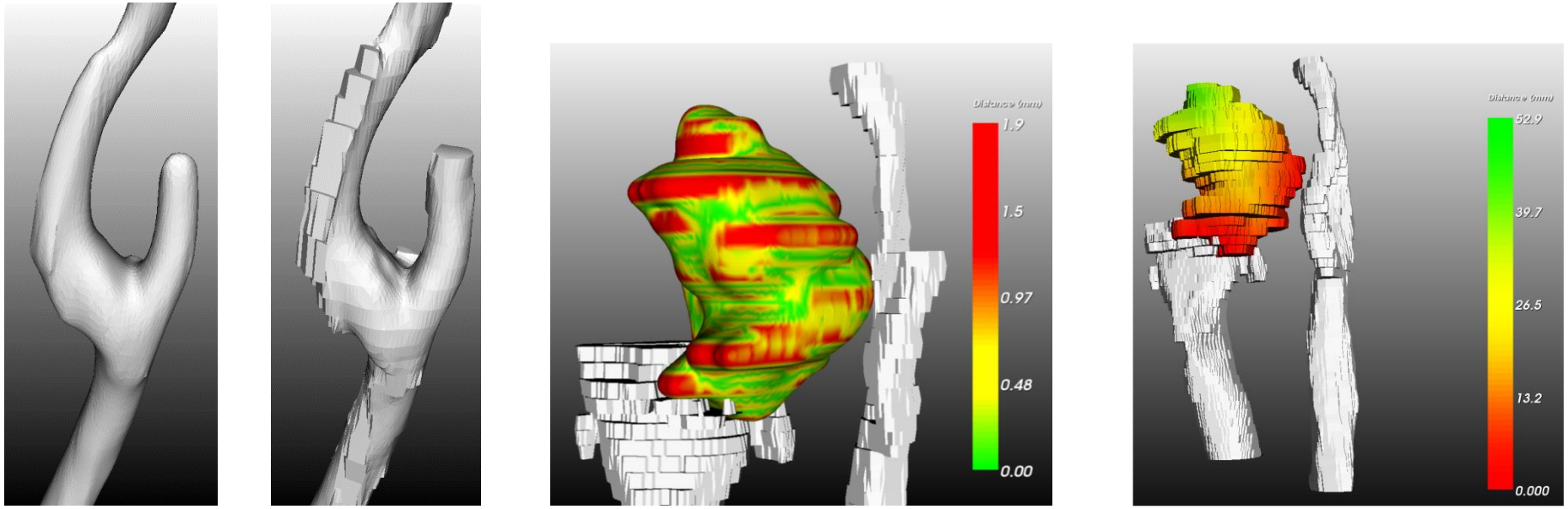
Recommendations

- Lowpass filter is the best solution for all object classes.
- For small objects
  - Topological neighborhood of size 2, 20-50 iterations, weighting: 0.7
- For flat objects, especially with problem areas:
  - Topological neighborhood: 1, approx. 20 iterations
- For elongated, branched objects:
  - Partially no suitable filter
- Few branched structures:
  - Lowpass filter with topological neighborhood of 1, weighting factor: 0.5 and 10 iterations

Constrained Mesh Smoothing

Distance–Aware Smoothing

- Smoothing under consideration of clinically relevant basic conditions
  - Volume loss (e.g., tumors)
  - Shape change
  - Distance changes between structures

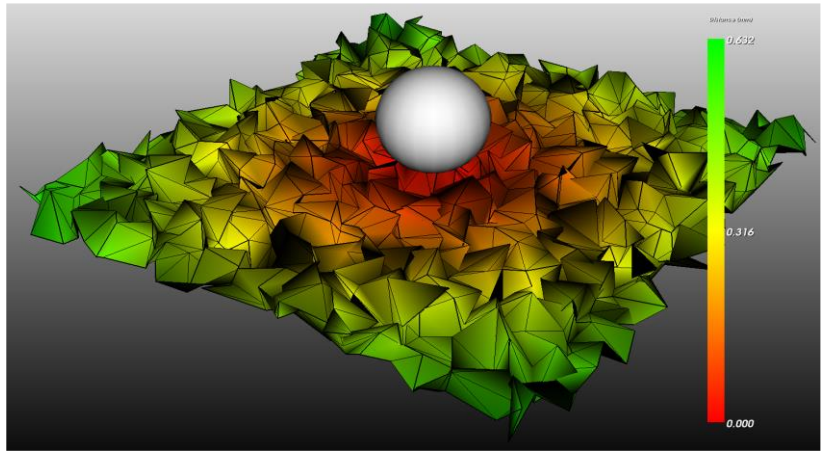


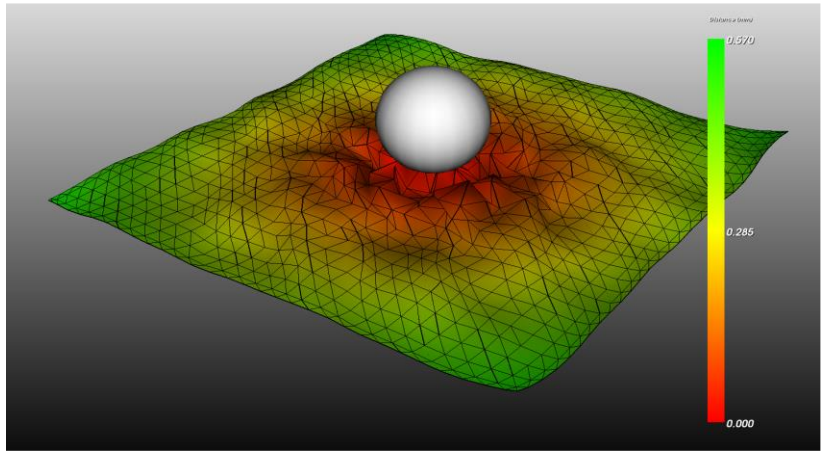
Source: [Mönch, 2010]

Distance-Aware Smoothing

- Clinically relevant aspects
  - Assessment of safety margins
  - Risk assessment during intervention and surgery planning

→ Consideration of spatial relations during smoothing

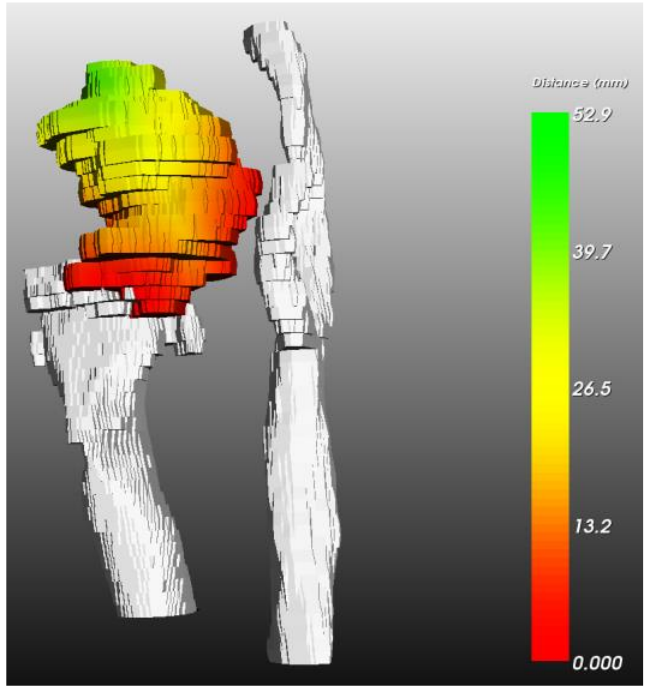


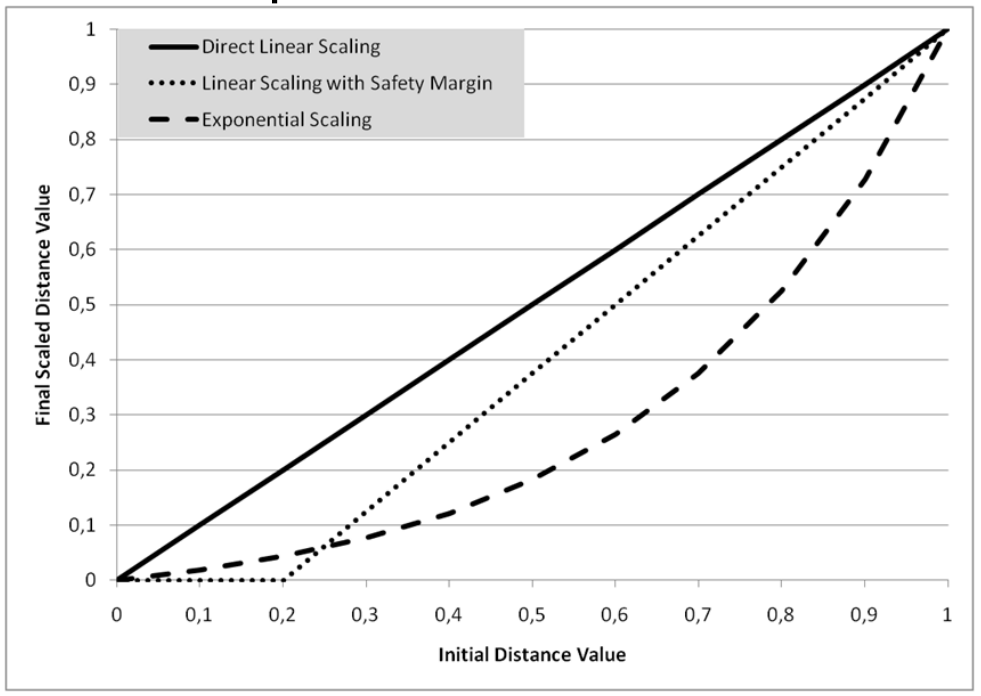


Source: [Mönch, 2010]

Distance-Aware Smoothing

- Calculation of minimum Euclidean distances to all (relevant) neighboring structures
- Scaling of the feasible deviation depends on the distance to risk structures





Source: [Mönch, 2010]

Distance–Aware Smoothing

- Calculation of minimum Euclidean distances to all (relevant) neighboring structures
- Scaling of the feasible deviation depends on the distance to risk structures

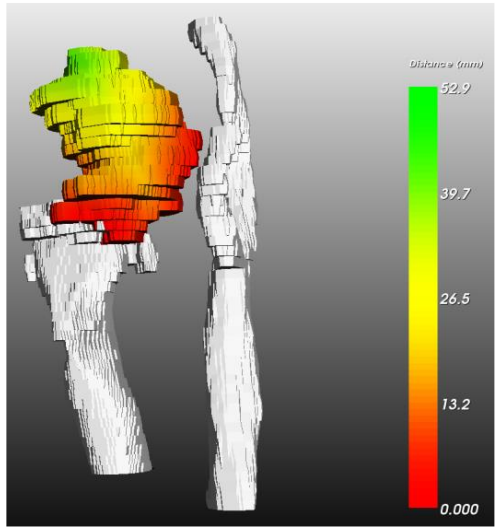
$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \lambda \sum_j \frac{1}{N(i)} (\mathbf{x}_j - \mathbf{x}_i)$$

λ is a distance-based weighting factor

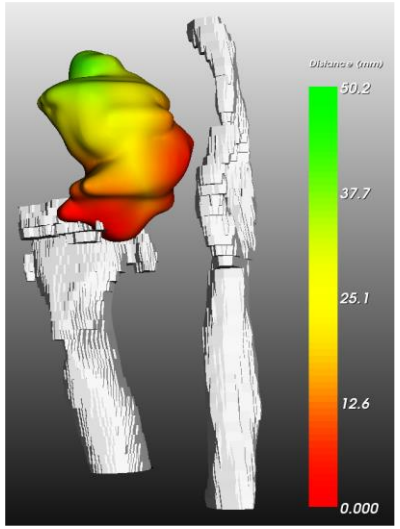
Distance–Aware Smoothing

Smoothing method	Min. Euclid. Dist.	Volume preservation
No smoothing	0.35 mm	100%
Default Laplace	2.17 mm	88.91%
Distance-Aware Laplace	0.35 mm	93.46%

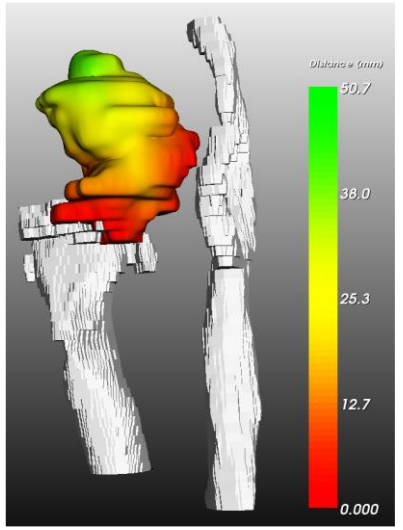
Original model



Consistent Laplace smoothing



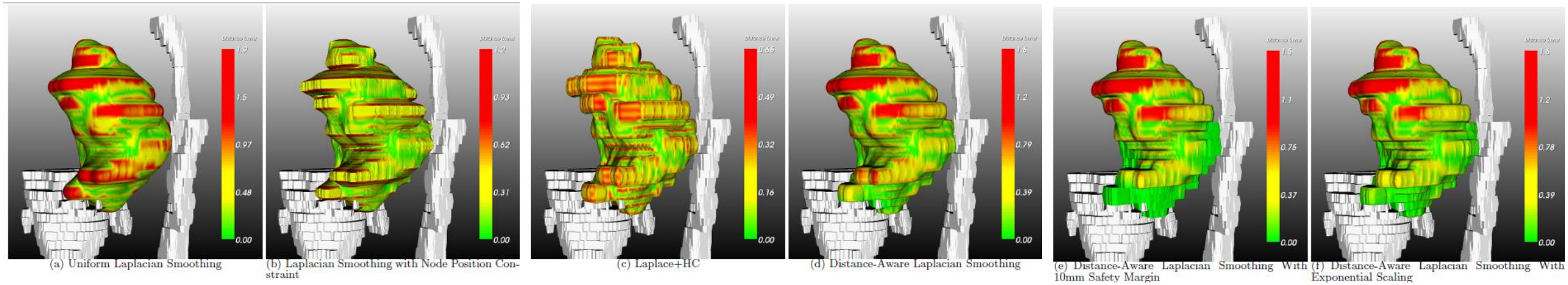
Distance-Aware Laplace



Source: [Mönch, 2010]

Distance–Aware Smoothing

- Volume-preserving methods (b, d):
  - No distance changes
  - Insufficient staircase reduction

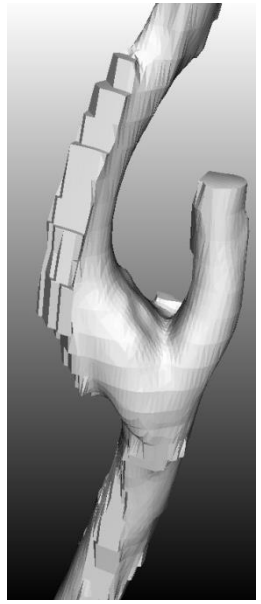


(a) Uniform Laplacian Smoothing (b) Laplacian Smoothing with Node Position Constraint (c) Laplace+HC (d) Distance-Aware Laplacian Smoothing (e) Distance-Aware Laplacian Smoothing With 10mm Safety Margin (f) Distance-Aware Laplacian Smoothing With Exponential Scaling

Color coding of the deviation to the original model

Source: [Mönch, 2010]

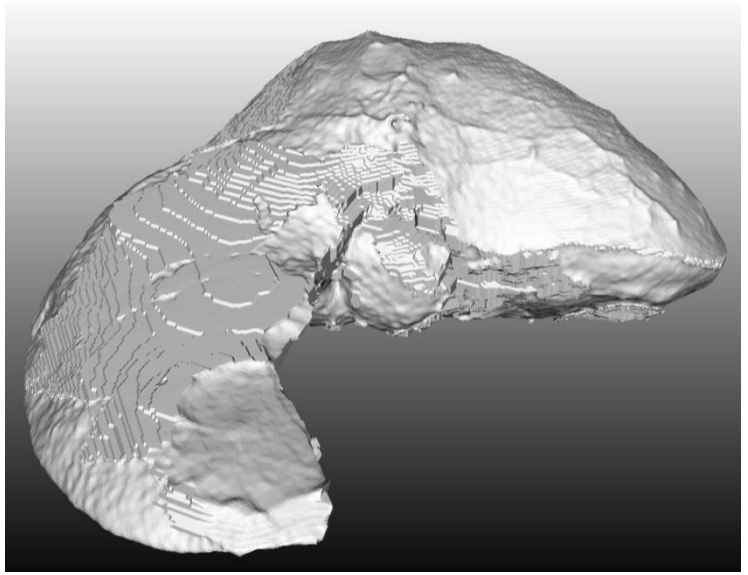
Staircase–Aware Smoothing



- Smoothing …
  - leads to volume loss and
  - removes potentially relevant details

  ↕

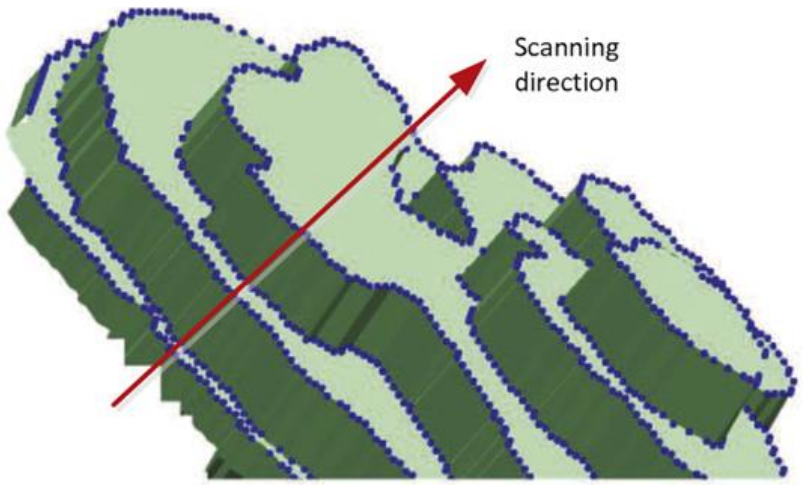
  - Insufficient staircase reduction



- Identification of critical artifacts
- Limitation of smoothing to these areas
- Preservation of volume, shape and details

Staircase–Aware Smoothing

- Features of staircase artifacts?
  - depend on the orientation of the image data → represented in the DICOM data
  - the size depends on the slice distance
  - exhibit sharp edges



Source: [Wei, 2015]

Staircase–Aware Smoothing

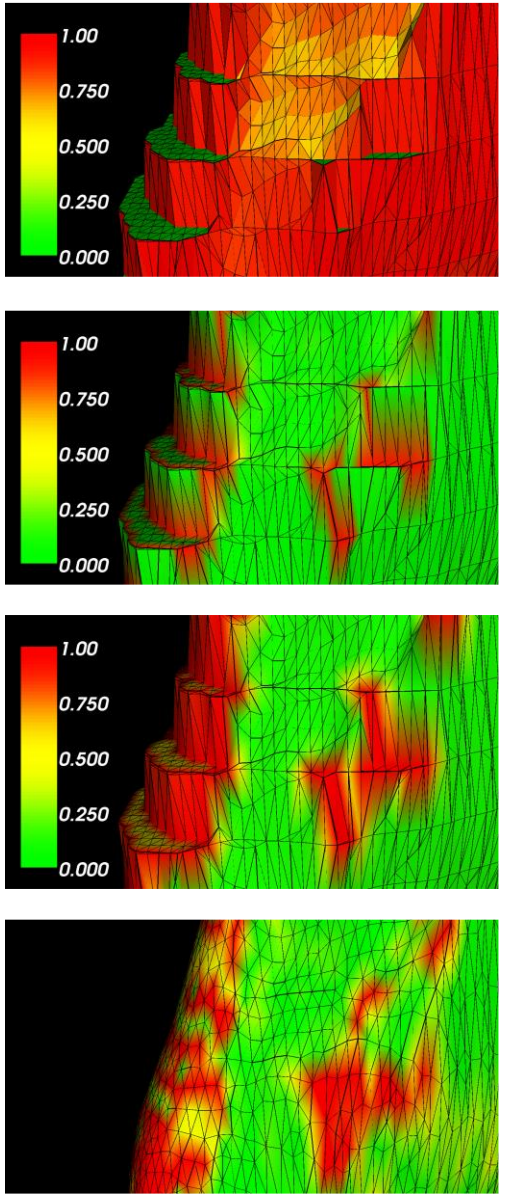
Definition of staircasiness

1. Calculation of the surface orientation relative to the slice direction

2. Identification of changes of the surface orientation

3. Calculation of the weights according to the distance to the edges at “staircases”

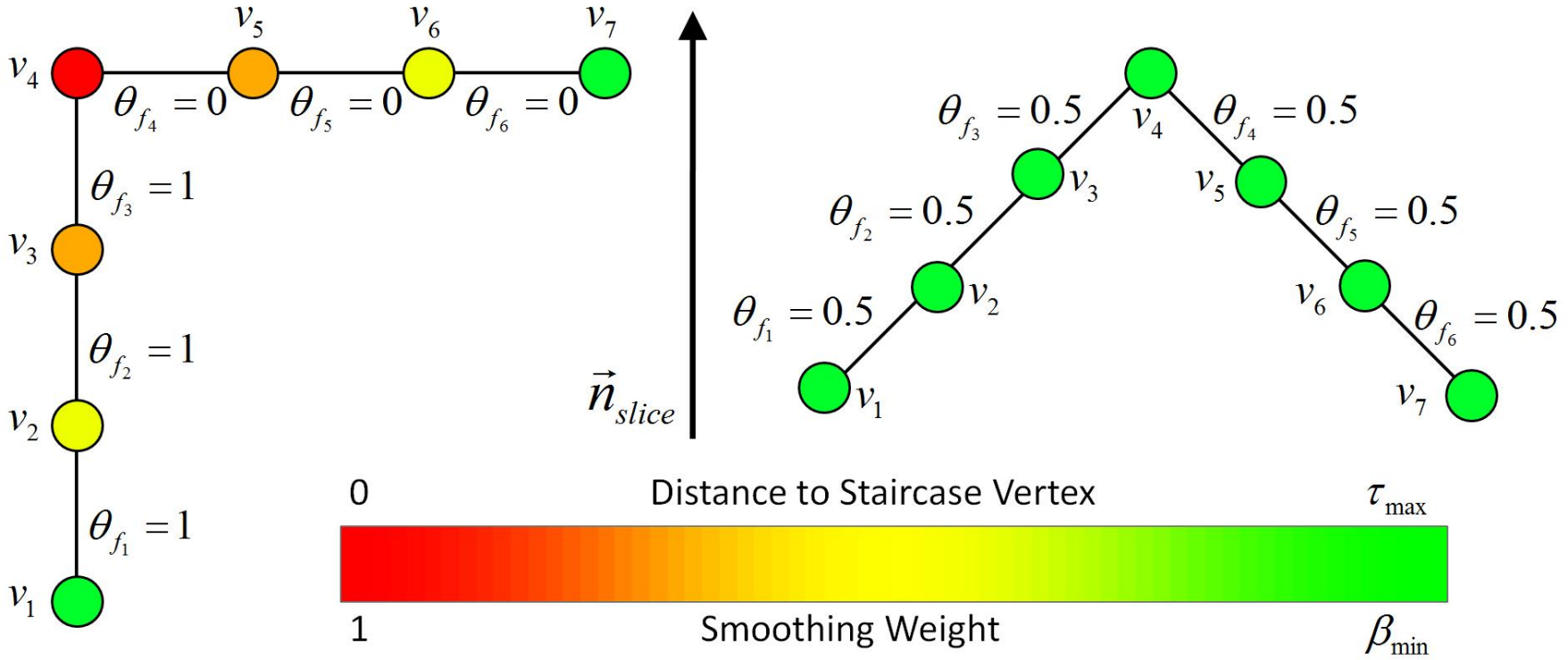
4. Application of distance-based weighting to the smoothing



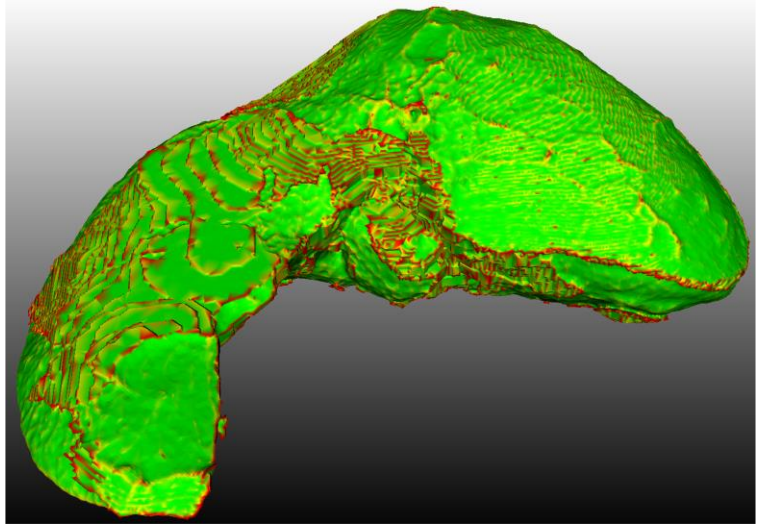
Source: [Mönch, 2011]

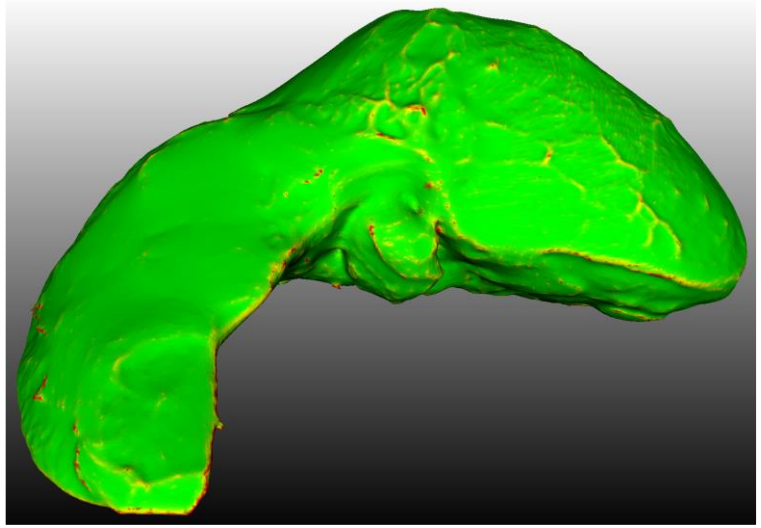
Staircase–Aware Smoothing

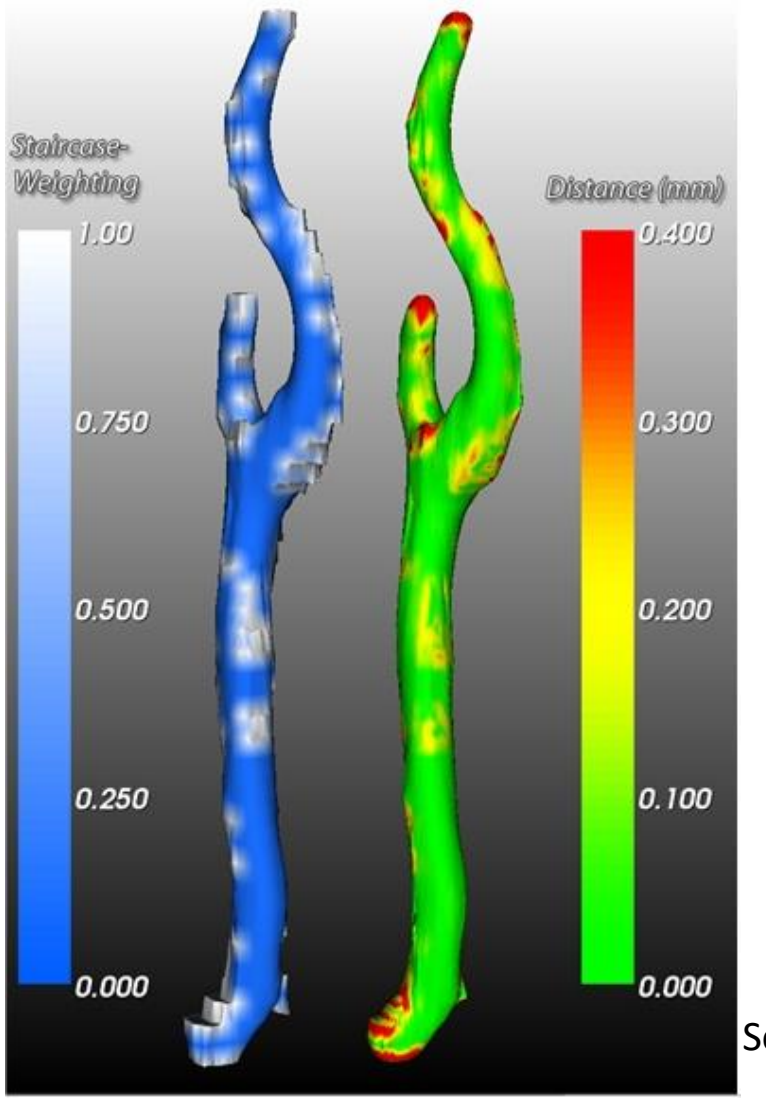
- Schematic depiction
  - Relation between model and layer orientation



Staircase–Aware Smoothing

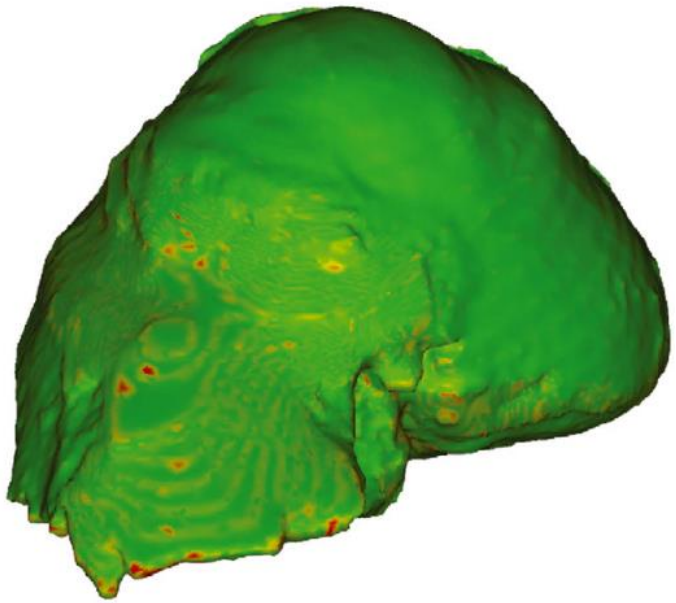


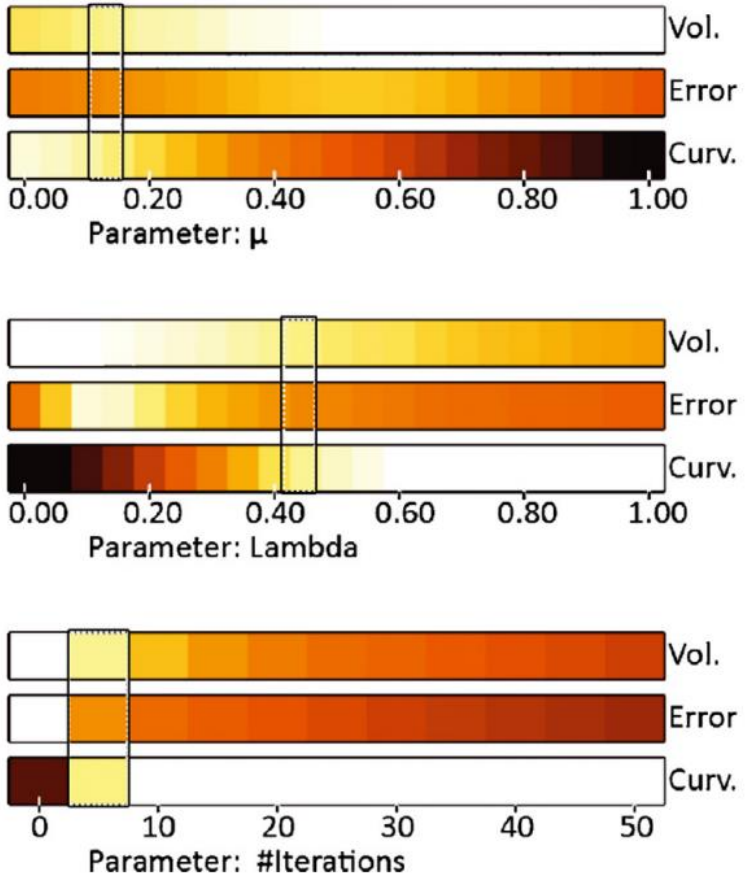




Source: [Mönch, 2011]

Interactive Smoothing





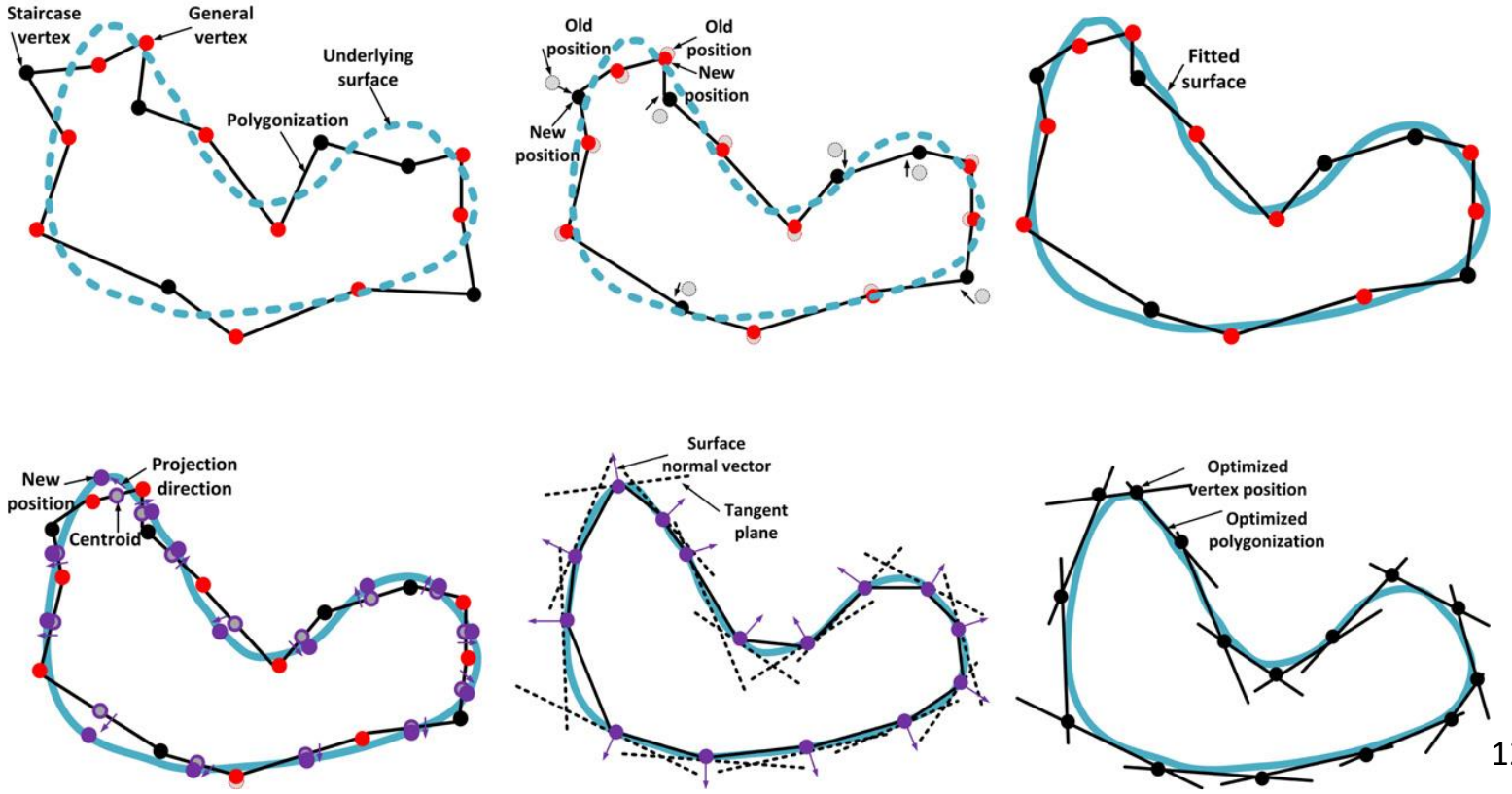
Source: [Mönch, 2013]

Staircase–Aware Smoothing

- Alternative solution (Wei, 2015)
- Staircase vertices are determined in a two-step process.
- First, the mesh is divided in „normal“ regions and strong edges.
- For the determination of the slicing direction, only faces that share strong edges are used.
- For these faces, the method of Mönch et al. is applied to define the slicing direction and to identify staircase vertices
- Smoothing is performed with quadratic Bezier surfaces (9 degrees of freedom), where a smoothing weight is adjusted to change primarily staircase vertices.

Staircase–Aware Smoothing

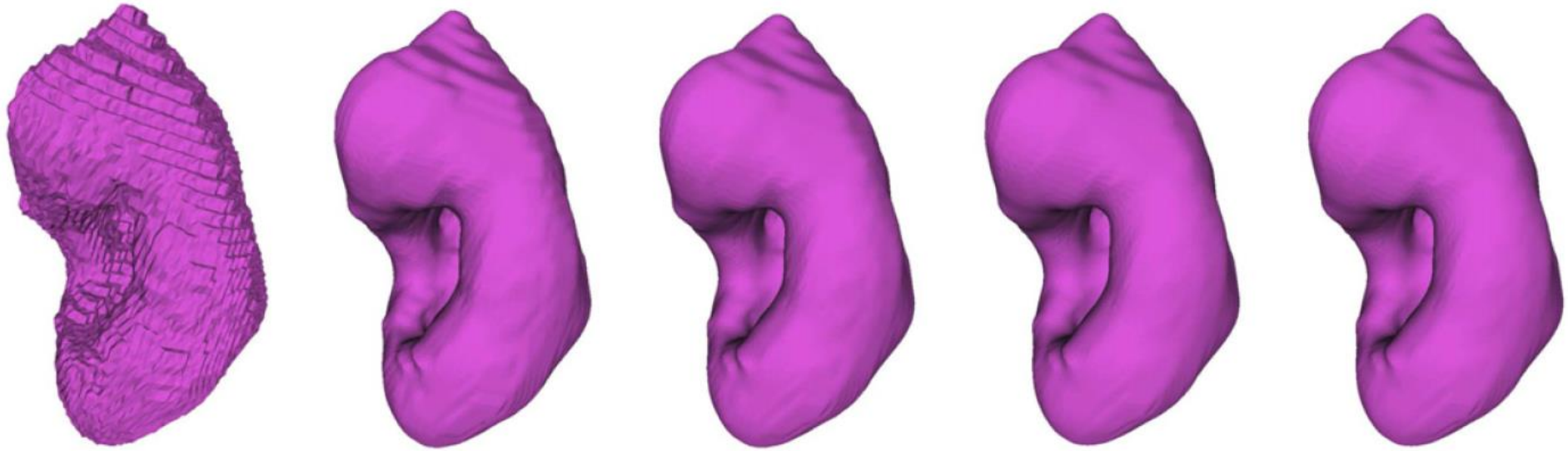
- 1st: Smoothing with staircase-based weighting
- 2nd: Fitting to the quadratic Bezier surfaces



(From: [Wei, 2015])

Staircase–Aware Smoothing

- Iterative staircase-aware smoothing (left original surface, second to fifth image: number of iterations is increasing).



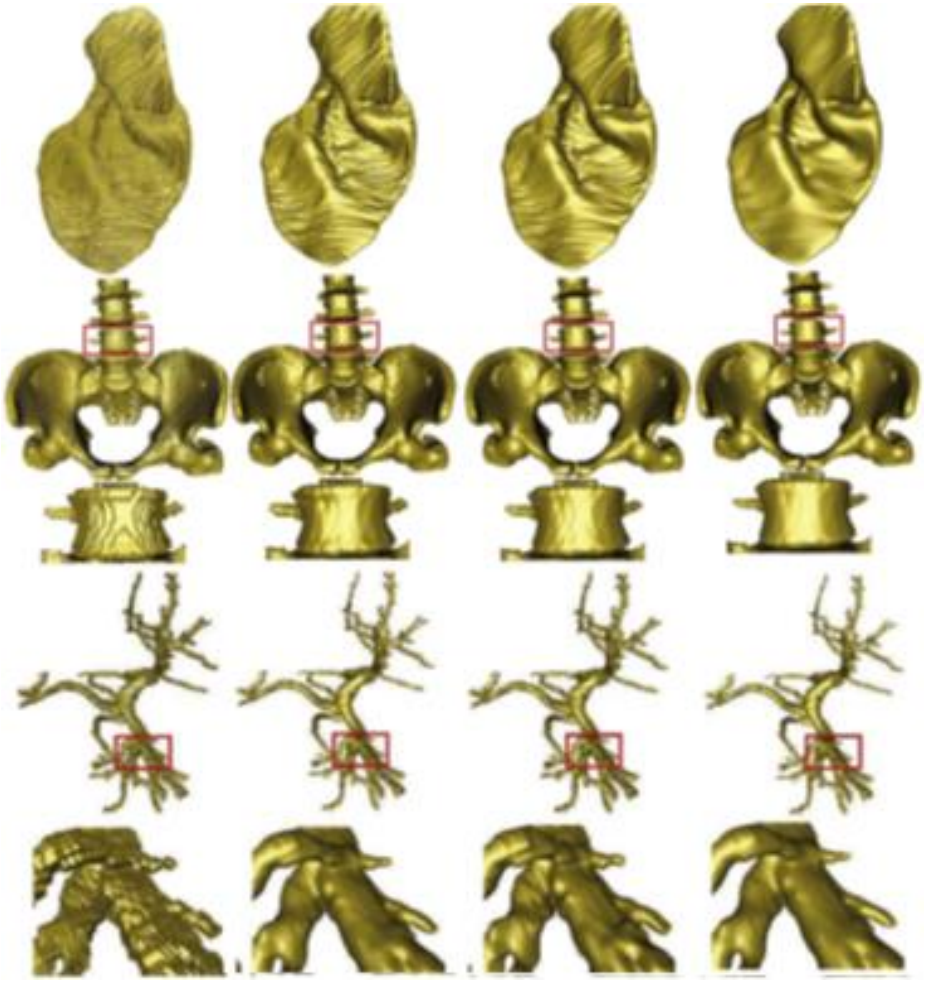
(f) (g) (2,2). (h) (2,4). (i) (2,6). (j) (2,8).

(From: [Wei, 2015])

Staircase–Aware Smoothing

- Comparison of three methods to smooth the models shown in the left column:
- Taubin‘s λ/μ filter, Vollmer‘s Laplace with correction, Wei‘s staircase-aware filter.
- Wei‘s filter is clearly the best for all models (liver, bones, vertebrae, vascular tree).
- Performance: for larger models (bones, liver ~600 K faces): 7 minutes on a modern PC with 8 Gbyte RAM.

(From: [Wei, 2015])

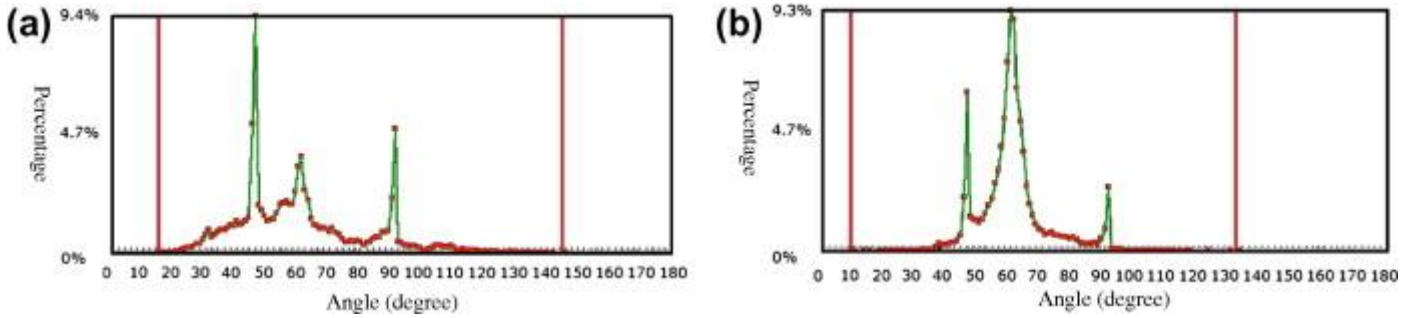


Evaluation of Smoothing Algorithms

Accuracy is studied by looking at

- (Hausdorff) Distance Histogramms (original-smoothed),
- Angular histogramms

It is essential not only to compare mean values between smoothing approaches.



Angle histogramms to compare two smoothing algorithms (From: Wang, Yu: 2011)

Quad Meshes

Quad Meshes for Medical Visualization

- So far: exclusively triangular meshes (simple and efficient algorithms, good support by graphics hardware

- Quadrilaterals require more complex algorithms (they are not necessarily convex) but they have advantages for efficient meshing (less elements are needed).

- Quadrilateral and triangular elements may be integrated (*hybrid meshes*), often with primarily quads (quad-dominant meshing)

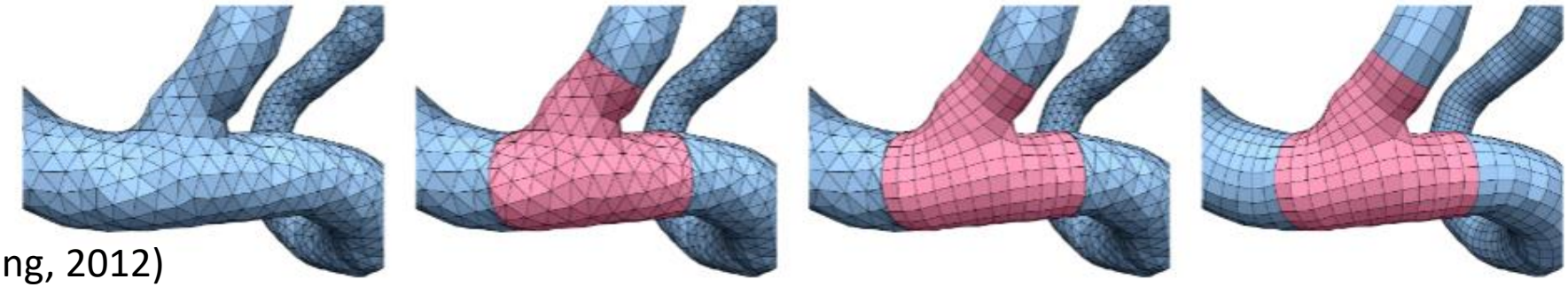
- Triangular meshes are transformed in quads by segmenting the input mesh in patches and transforming the patches individually.

Criteria for the transformation:

- Good alignment with the overall shape
- Good element quality (angles around 90 degrees)
- Reasonable size of the meshes

Quad Meshes for Medical Visualization

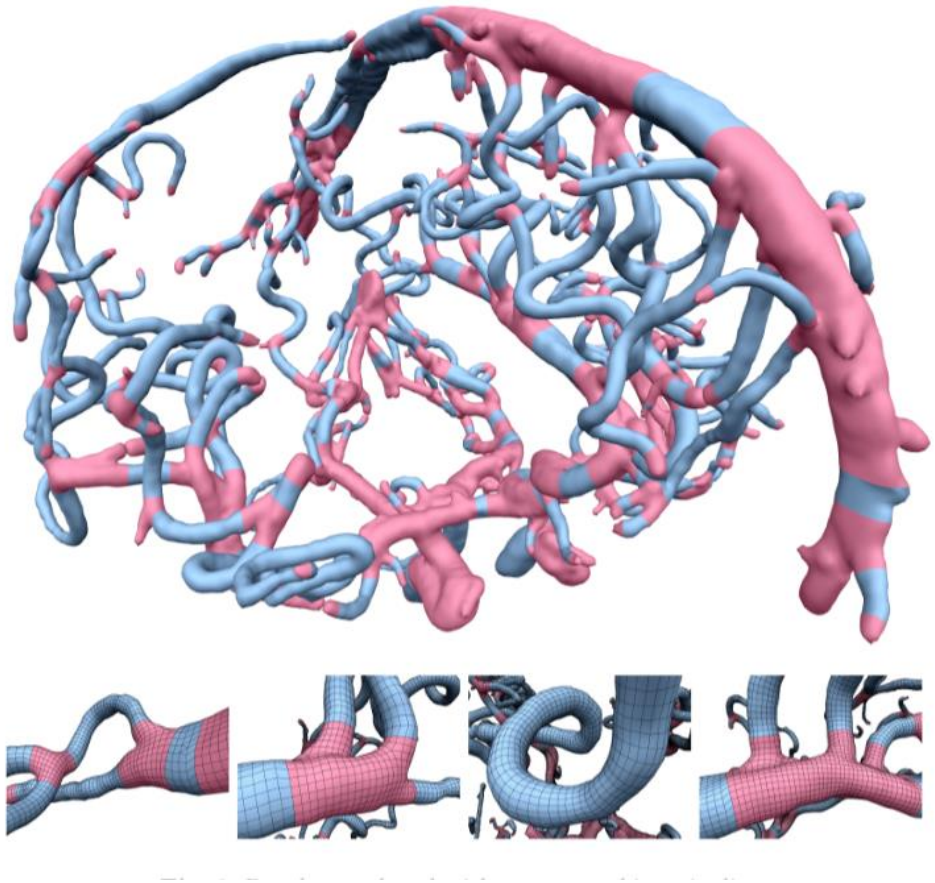
- The input mesh is post-processed Marching Cubes triangle mesh.
- It is divided in junction and tube parts with proper interfaces.
- Junctions are transformed in quads first (takes long!).
- Tubes are transformed in quads in the last step.
- Note the different resolution for smaller and larger vessels



(From: Sibbing, 2012)

Quad Meshes for Medical Visualization

- Based on a skeletonization, the mesh was divided in 215 junction and 214 tube elements. Quad-dominant mesh of a cerebral tree. 120 k faces result



(From: Sibbing, 2012)

Summary

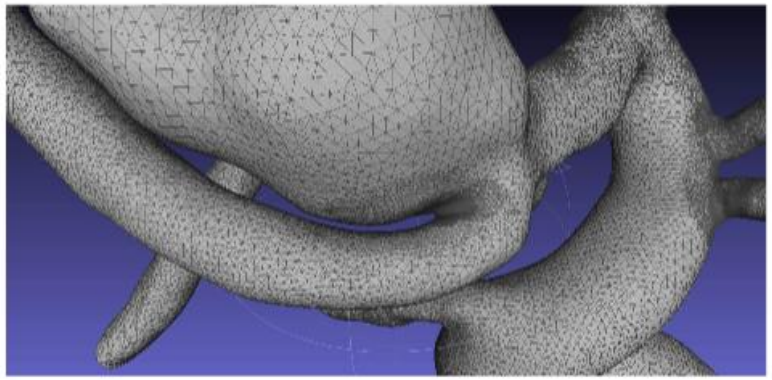
- Applications in medical education and training are often based on surface models.
- Surface models are also used as input for
  - biomedical simulations (FEM, biomechanical simulations, fluid simulations)
  - Rapid prototyping

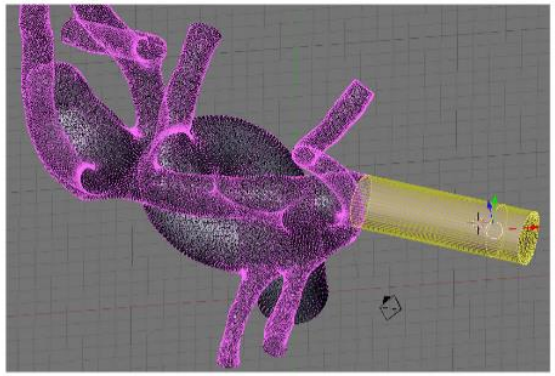
- Surface models (often triangle meshes) are derived from medical image data
- Raw surface models need postprocessing w.r.t. smoothness
- Accuracy (volume preservation, distances to other structures) has to be considered.

Summary

Mesh processing for medical applications is partially supported by:

- MevisLab (mesh smoothing)
- Blender (local edits)
- MeshLab (remeshing)
- ParaView (adaptive mesh refinement)





- Screenshots of Meshlab (after local remeshing) and Blender (after extruding a vessel)

Questions???