FACULTY OF ENGINEERING AND ARCHITECTURE

E016712: Computer Graphics

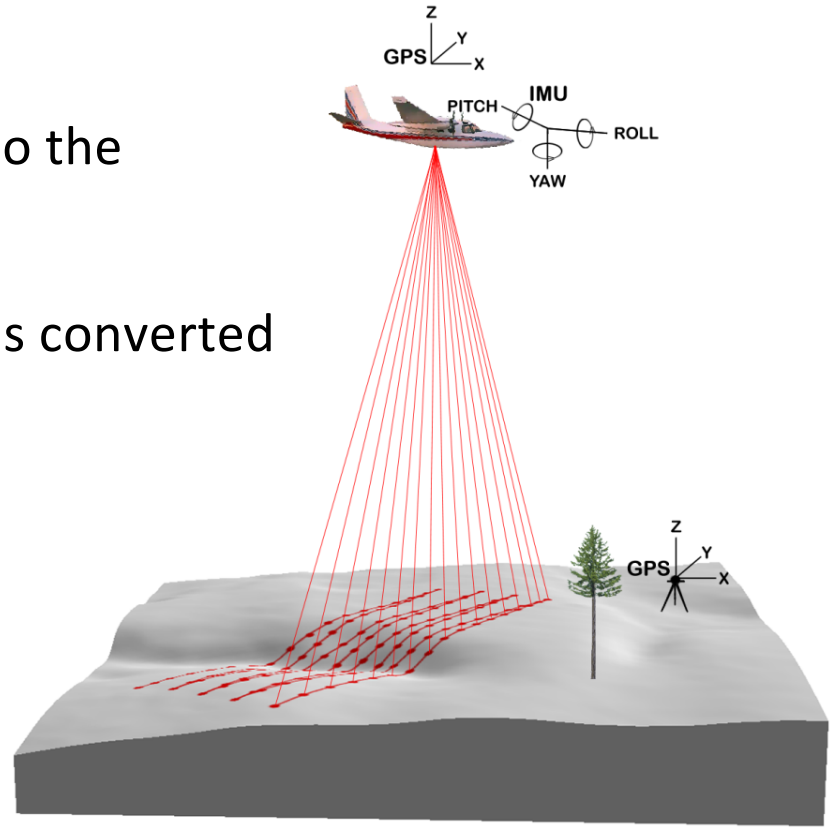
Point Clouds & Scene Graphs

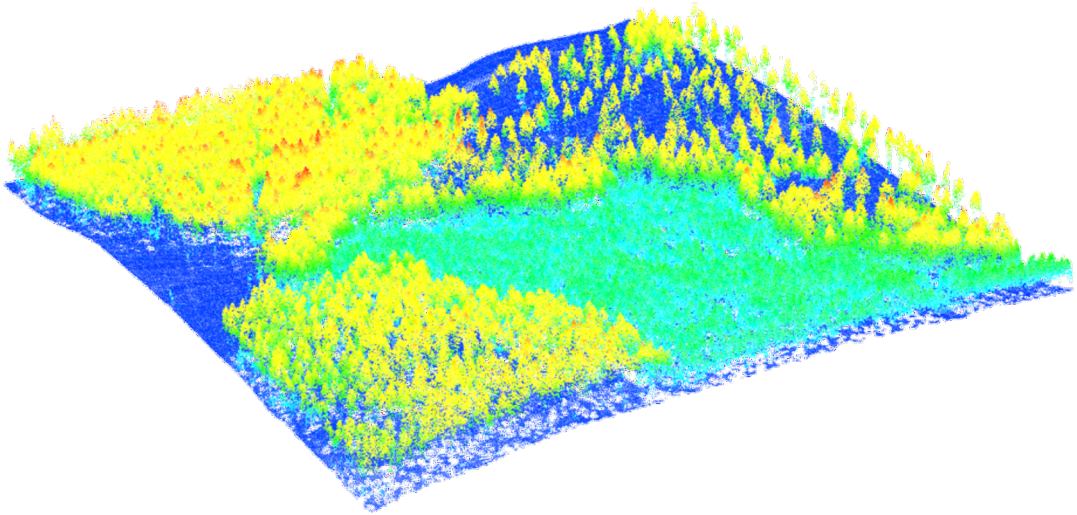
Lecturers: Aleksandra Pizurica and Danilo Babin

GHENT UNIVERSITY

Light Detection and Ranging (LiDAR)

- LiDAR is also commonly known as ***Laser Radar***
- Laser generates an optical pulse
- Pulse is reflected off an object and returns to the system receiver
- Receiver measures the time of flight which is converted to a distance.

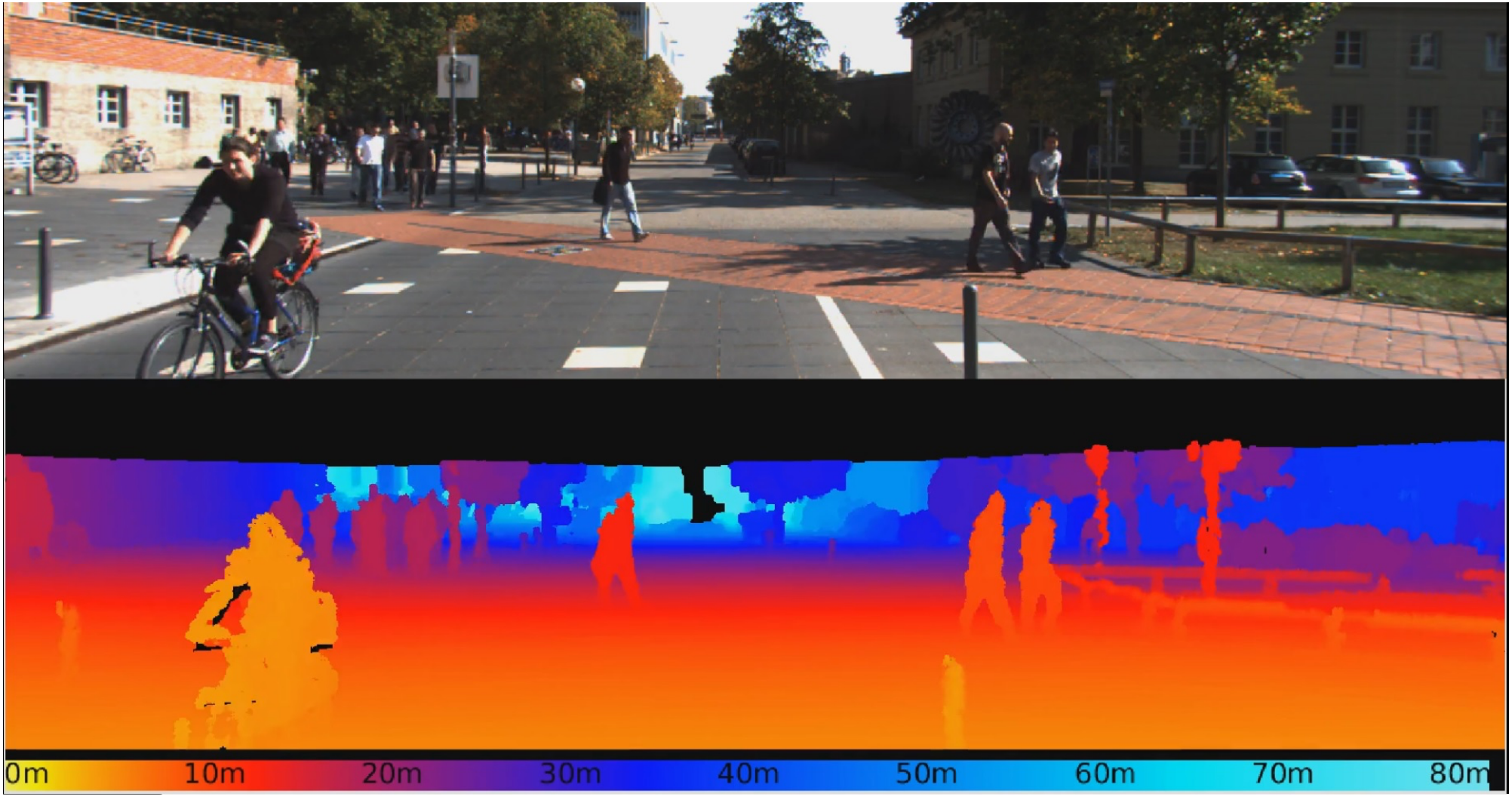




LiDAR point data colored by height

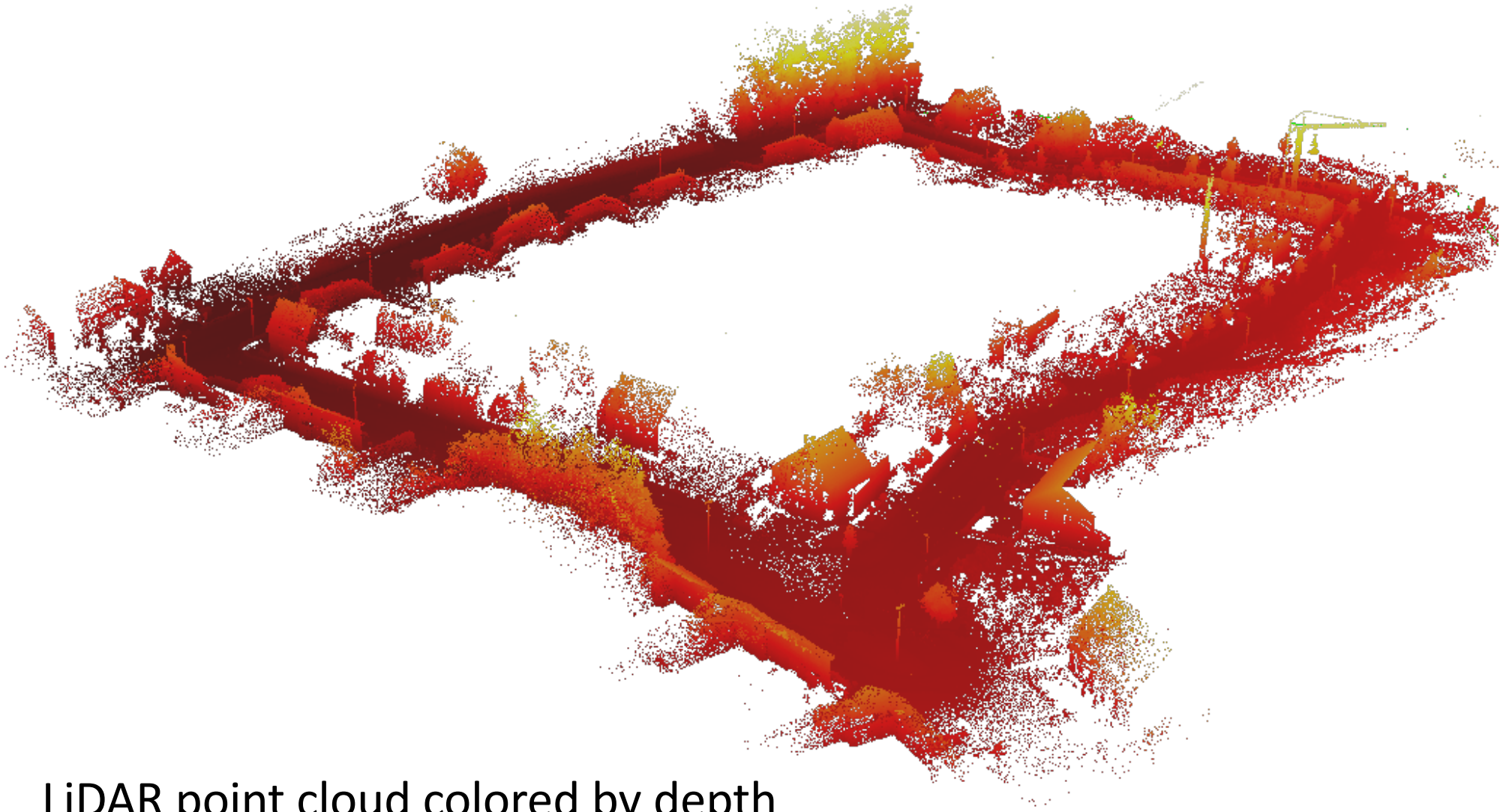
Figures from McMcGaughey
USDA Forest Service--PNW Research Station

Light Detection and Ranging (LiDAR)



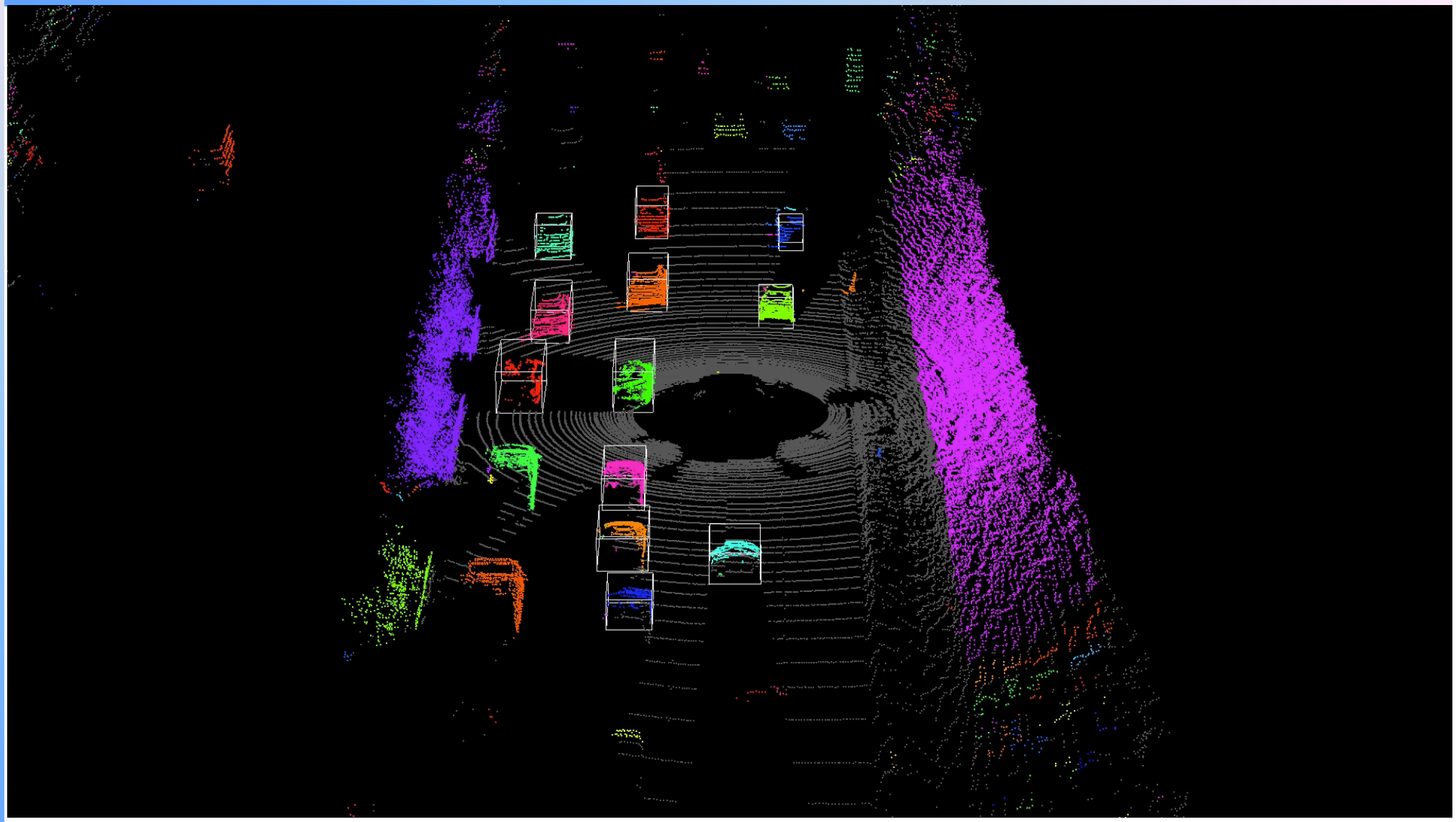
LiDAR image colored by depth

Light Detection and Ranging (LiDAR)



LiDAR point cloud colored by depth

Light Detection and Ranging (LiDAR)

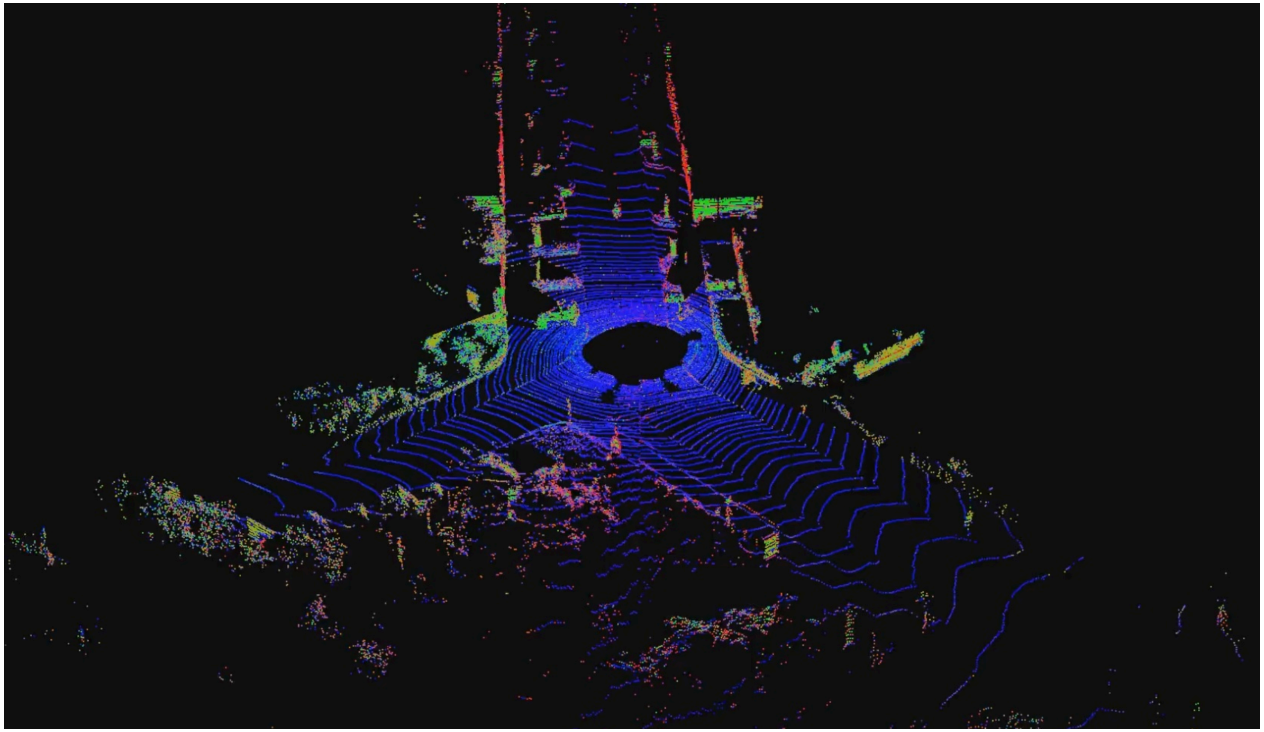


LiDAR point cloud colored by object classification

Point Cloud Reconstruction Problems

Problems in acquired data:

- Non-uniform sampling
- Holes
- Noise
- Bad scan alignment
- No (reliable) normals

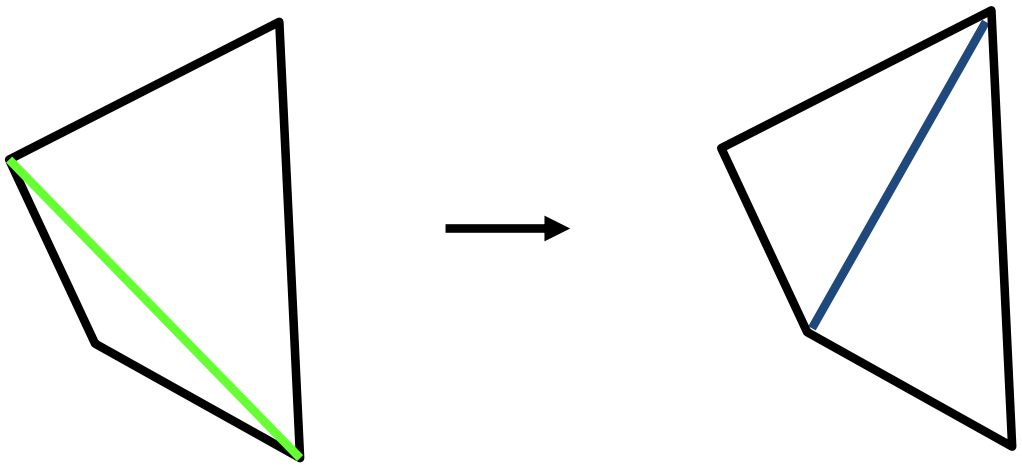


Overview

- Point cloud meshing
  - Delaunay
  - Graph cuts
  - Moving least squares
- Space partitioning (scene graphs)
  - Octree
  - K-d tree
  - BSP tree

Improving a Triangulation: Delaunay

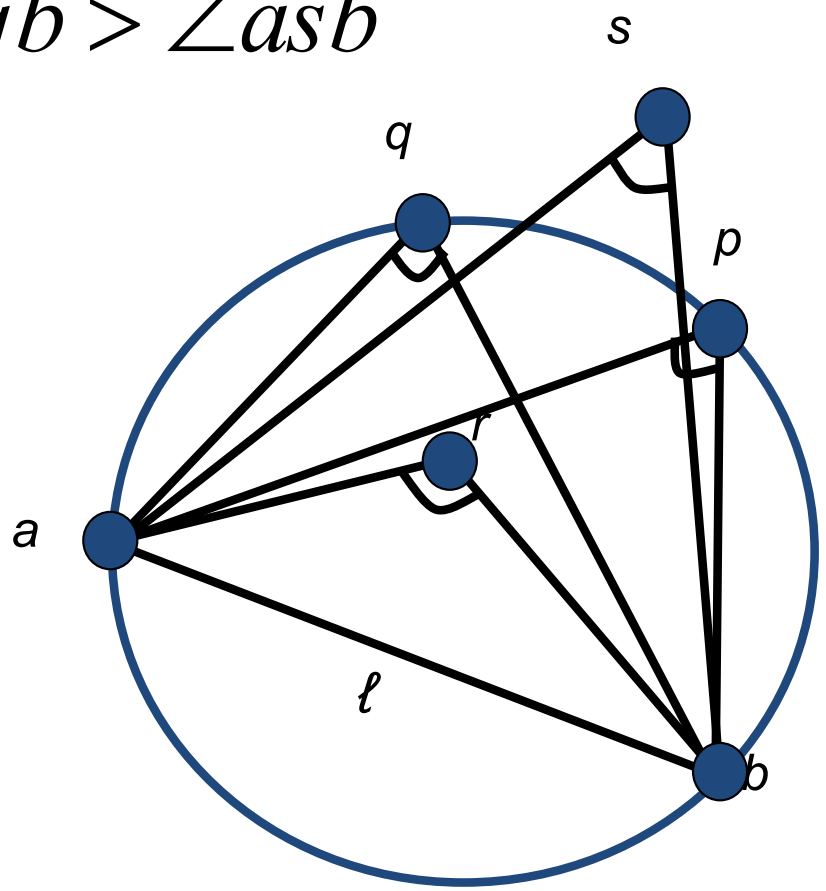
- If we compare two triangular meshes derived from the same set of points, we can say the better mesh is the one with the largest minimum interior angle of all the triangles in the mesh.
- In any convex quadrangle, an *edge flip* is possible.
- If the flip *improves* the triangulation locally, the original edge is called "illegal".



Thales's Theorem

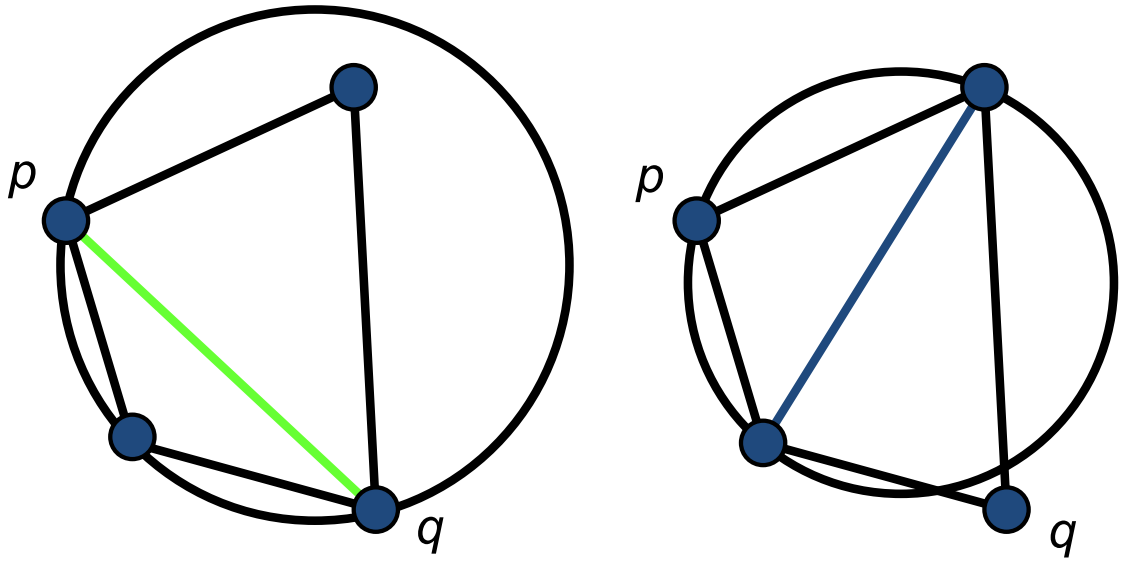
- Let C be a circle, and ℓ a line intersecting C at the points a and b. Let p, q, r, and s be points lying on the same side of ℓ, where p and q are on C, r inside C, and s outside C. Then:

$$\angle arb > \angle apb = \angle aqb > \angle asb$$



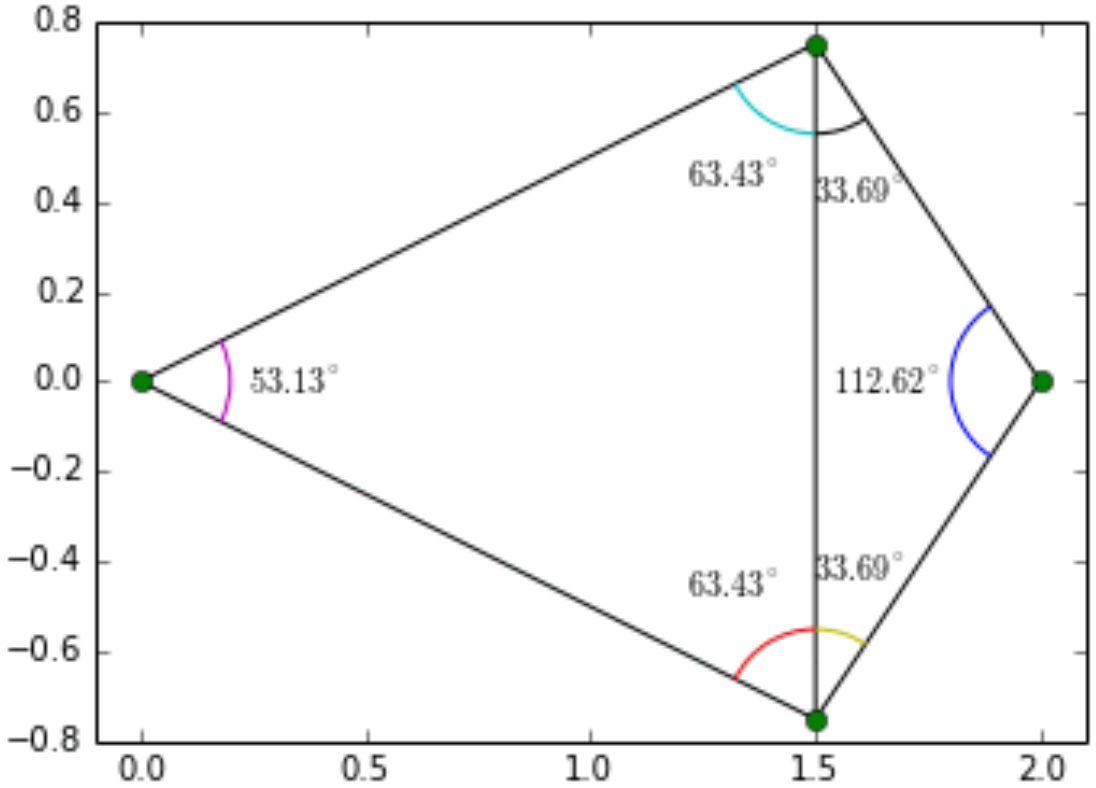
Illegal Edges

- An edge is illegal iff any of its opposite vertices are inside the circle defined by the other three vertices.
- A convex quadrangle has exactly one legal diagonal.
- A Delaunay triangulation does not contain illegal edges. (Otherwise it can be improved locally.)
- The Delaunay triangulation is not unique if more than three sites are cocircular.



Delaunay triangulation demo

- Each frame of the animation shows a Delaunay triangulation of the four points.
- Halfway through, the triangulating edge flips showing that the Delaunay triangulation maximizes the minimum angle, not the edge-length of the triangles.

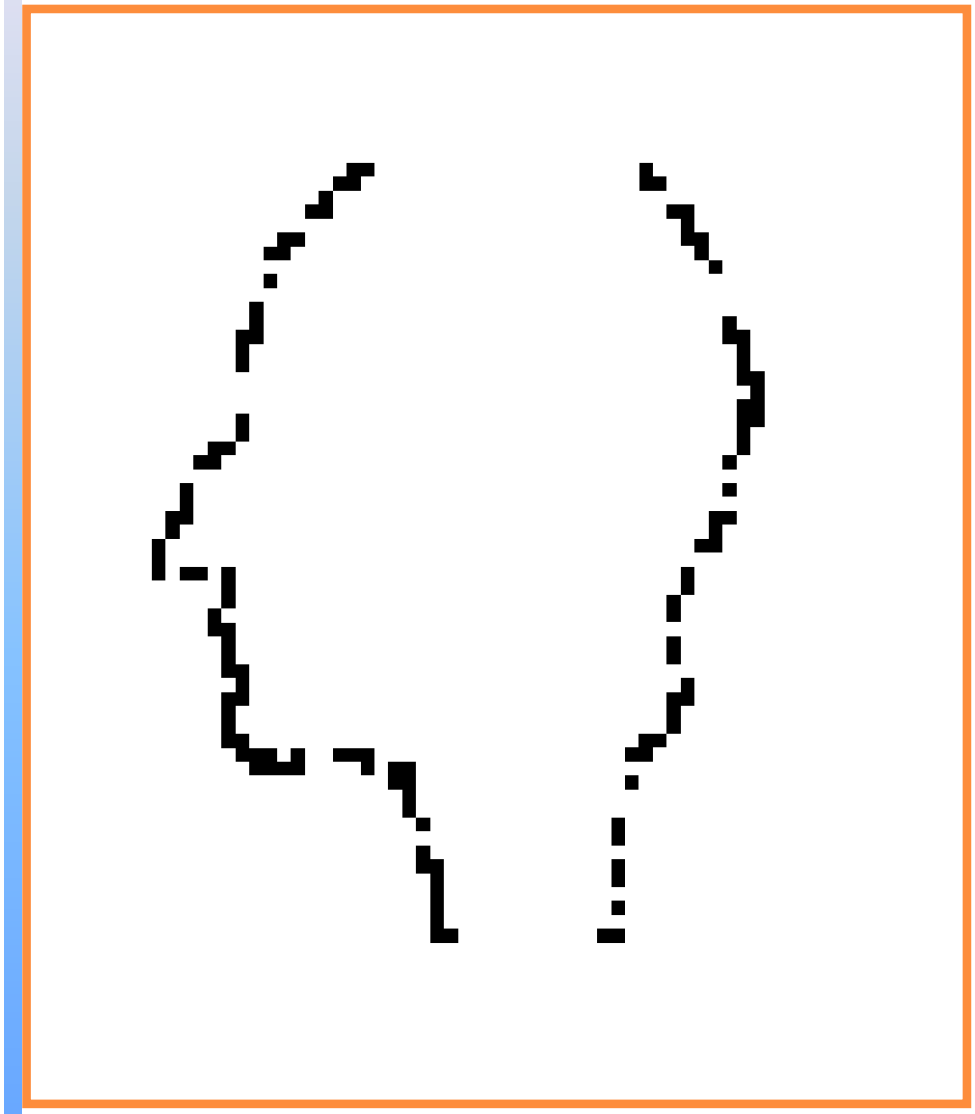


Source: Wikipedia

Delaunay triangulation use cases

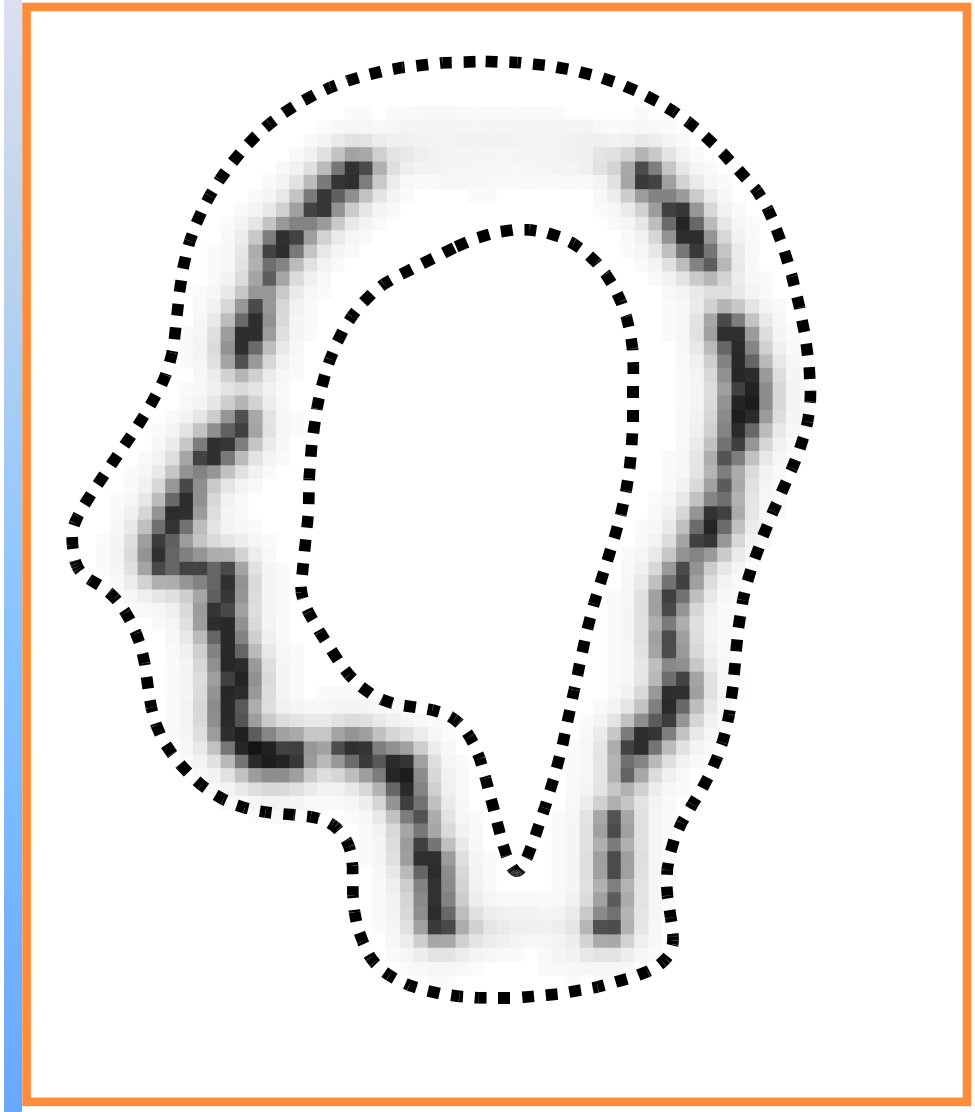
- Good for “exact” point clouds (noise-free, uniform sampling, no scan-misalignment)
- Not optimal for point clouds produced from natural scenes or moving scanners

Graph-based surface extraction



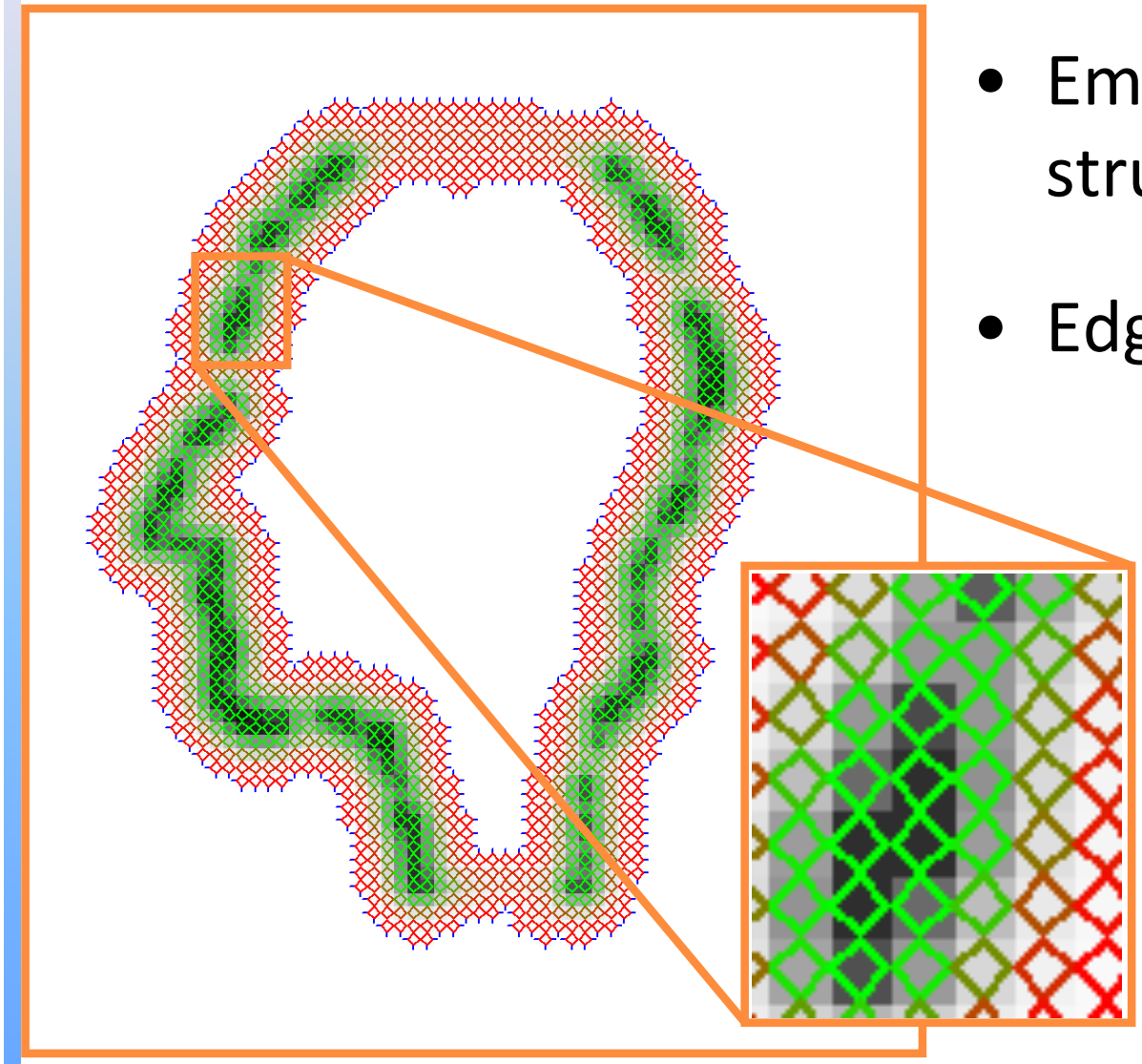
- Point cloud P
  - Holes
  - Noise

Graph-based surface extraction



- Surface confidence (distance) as “likelyhood” that surface intersects a voxel v: $\phi(v)$
- Compute “crust” containing the surface
  - Morphological dilation
  - Medial axis approximation

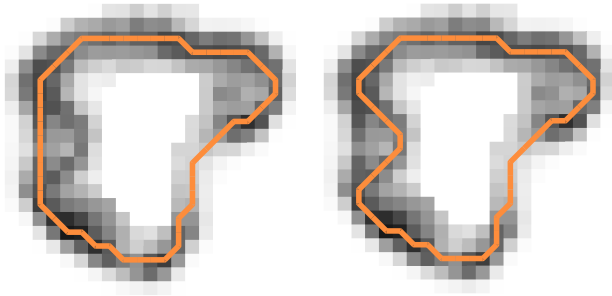
Graph-based surface extraction



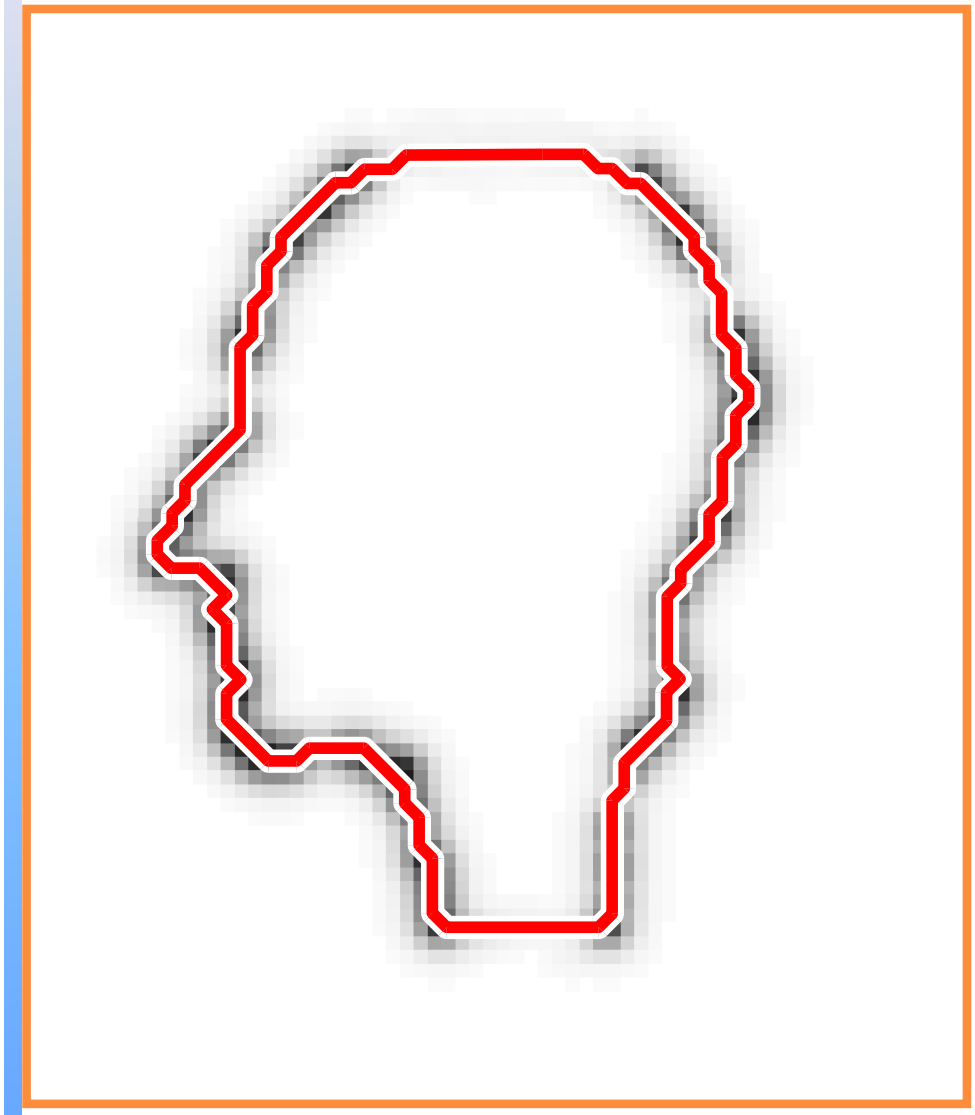
- Embed weighted graph structure G.
- Edge weights:

$$w(v) = \phi(v)^s + a$$

- "s" emphasizes strong/weak maxima.



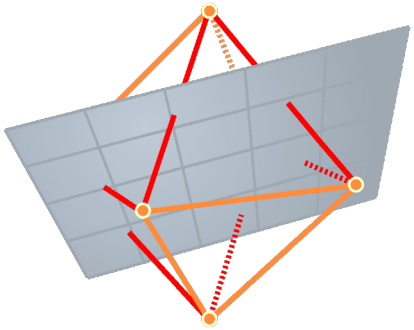
Graph-based surface extraction

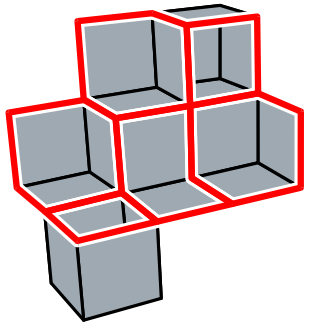


- Graph min-cut yields set of intersected surface voxels.
- Minimize energy:

$$E(S) = \int_S \phi(x)\, dx + \int_S a\, dS$$

Cut Manifold to Triangle Mesh

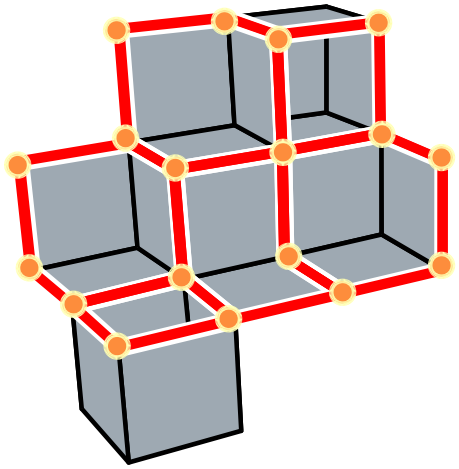




Graph *cut-edges* Loop of voxel *split-edges*

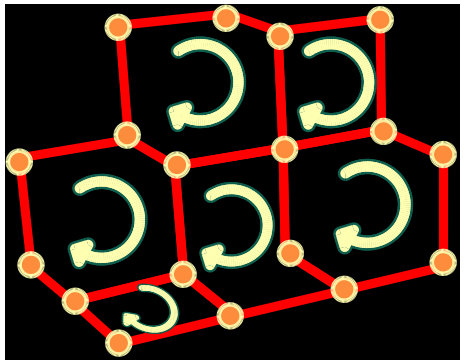
Cut Manifold to Triangle Mesh

- Loops define non-planar polygonal faces
- Mesh vertices at voxel *corners*



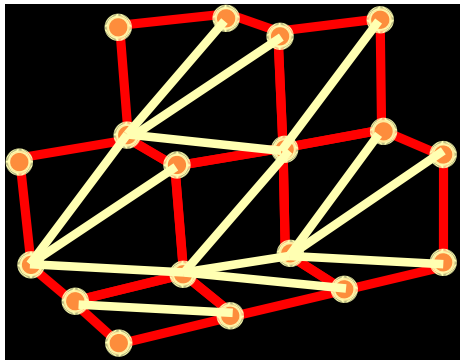
Cut Manifold to Triangle Mesh

- Loops define non-planar polygonal faces
- Mesh vertices at voxel *corners*
- Cycle along split-edges



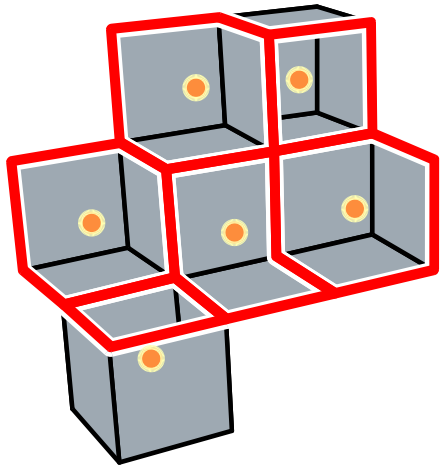
Cut Manifold to Triangle Mesh

- Loops define non-planar polygonal faces
- Mesh vertices at voxel *corners*
- Cycle along split-edges



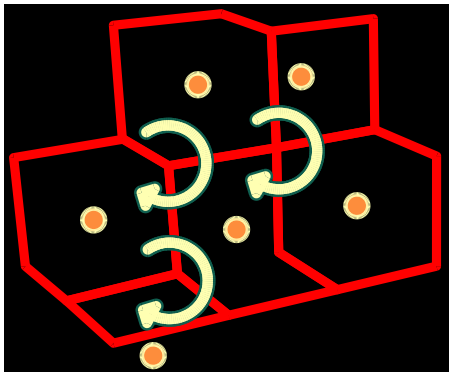
Cut Manifold to Triangle Mesh

- Mesh vertices at voxel *centers*
- Voxel *corners* correspond to non-planar faces
- Cycle over shared split-edges



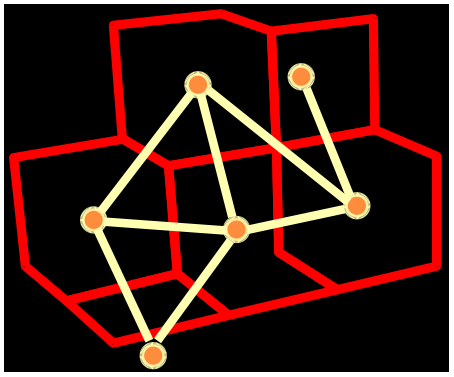
Cut Manifold to Triangle Mesh

- Mesh vertices at voxel *centers*
- Voxel *corners* correspond to non-planar faces
- Cycle over shared split-edges

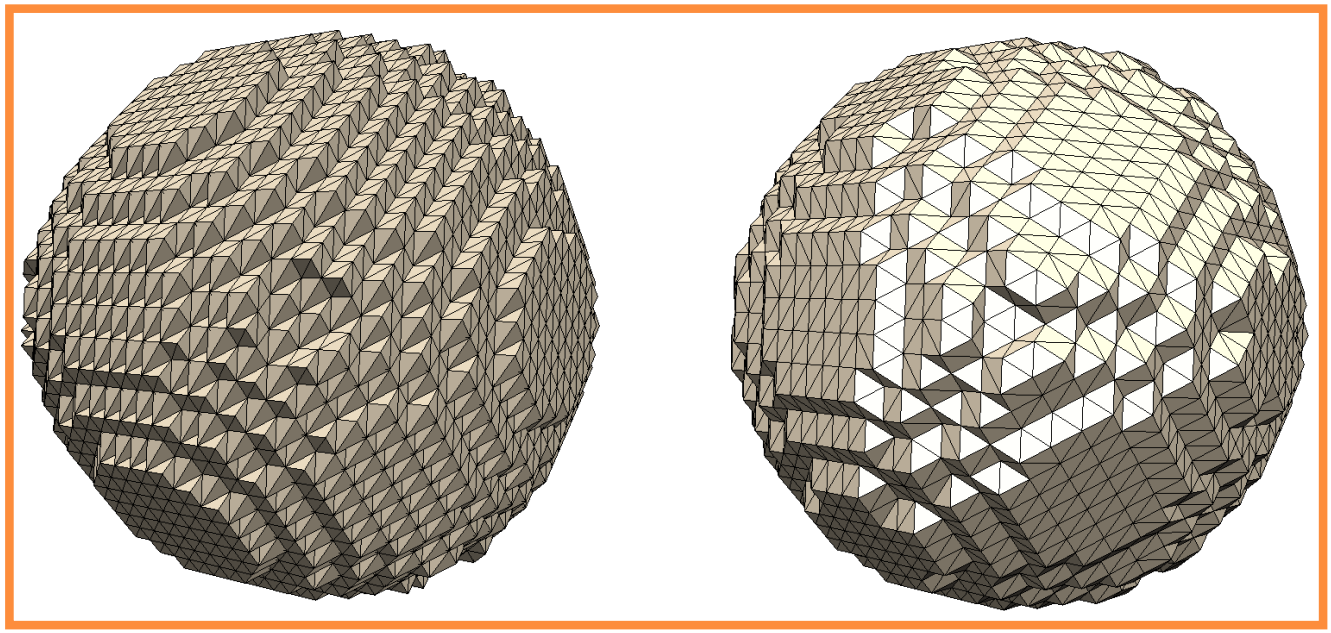


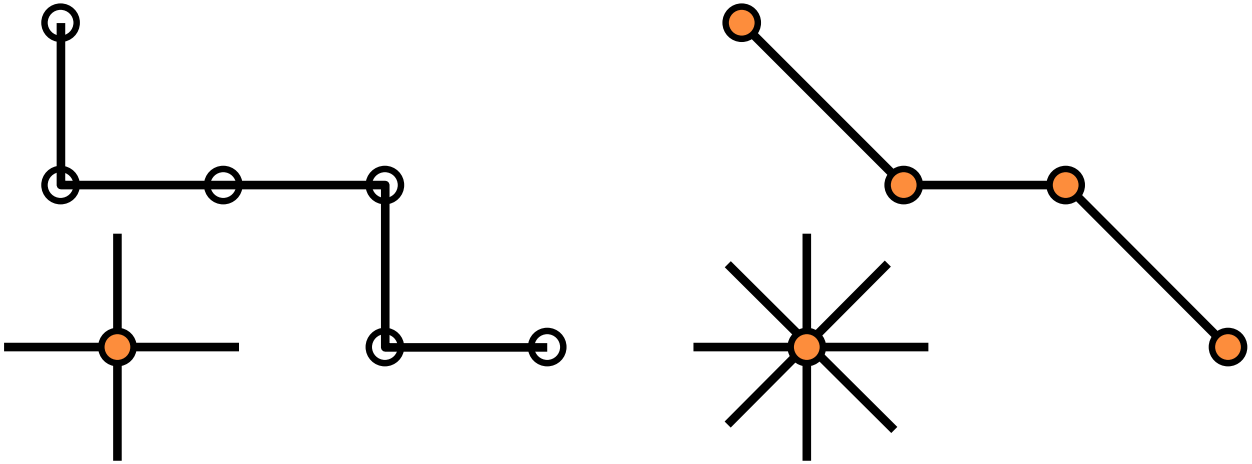
Cut Manifold to Triangle Mesh

- Mesh vertices at voxel *centers*
- Voxel *corners* correspond to non-planar faces
- Cycle over shared split-edges

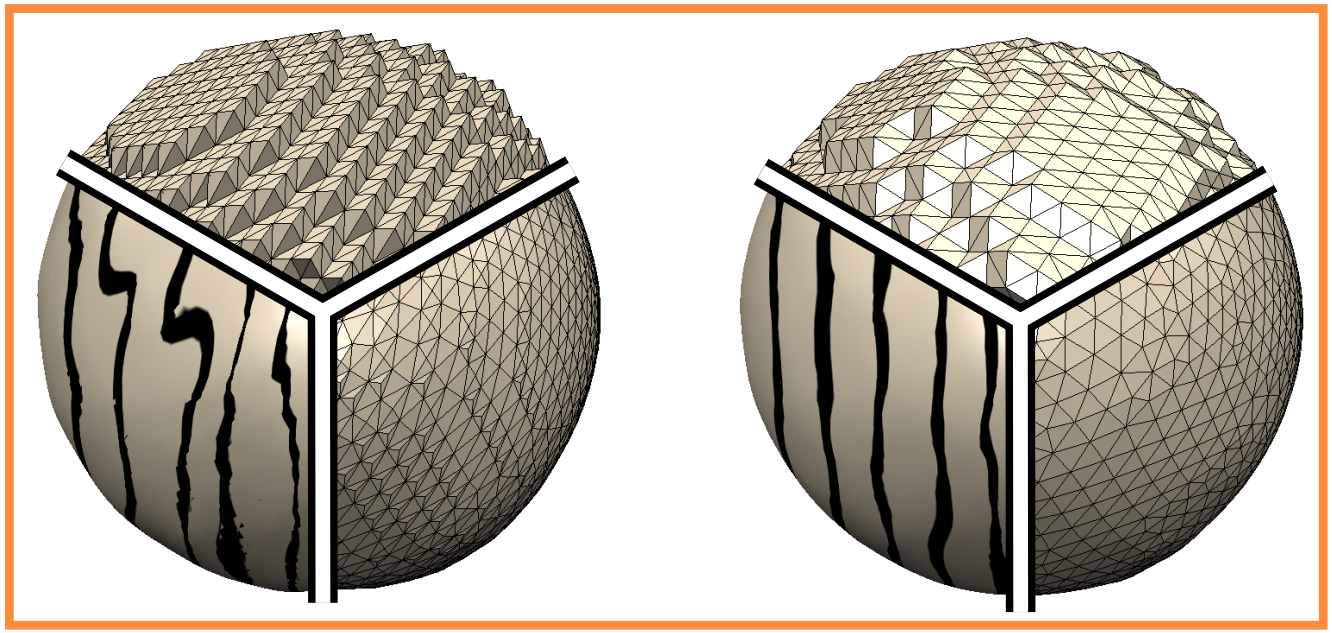


Cut Manifold to Triangle Mesh





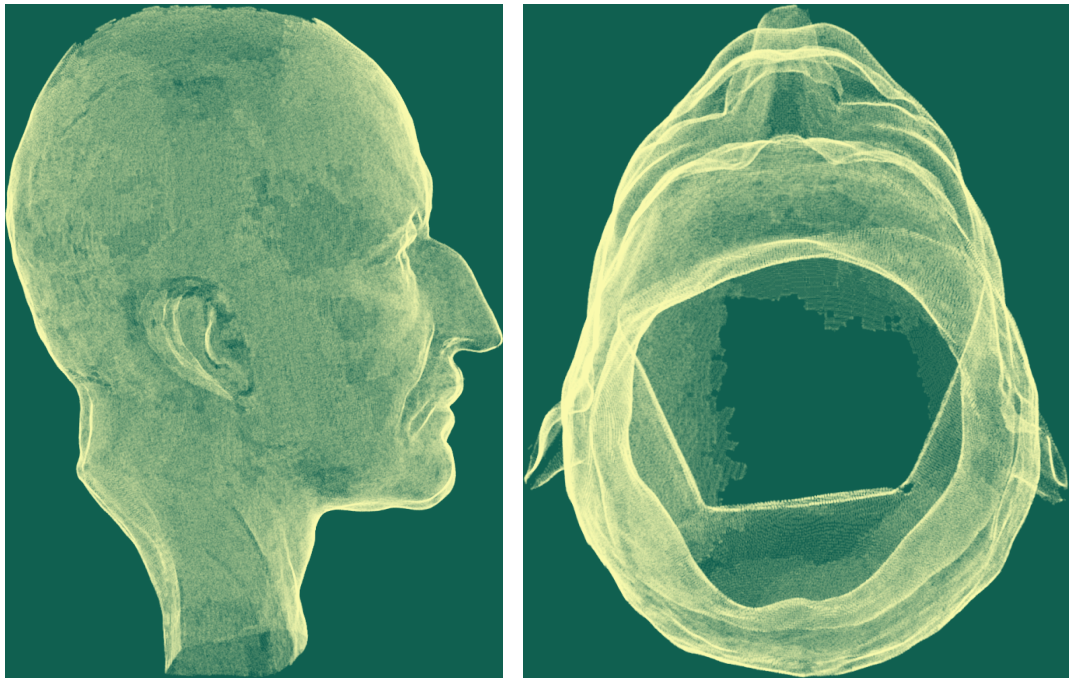
Cut Manifold to Triangle Mesh

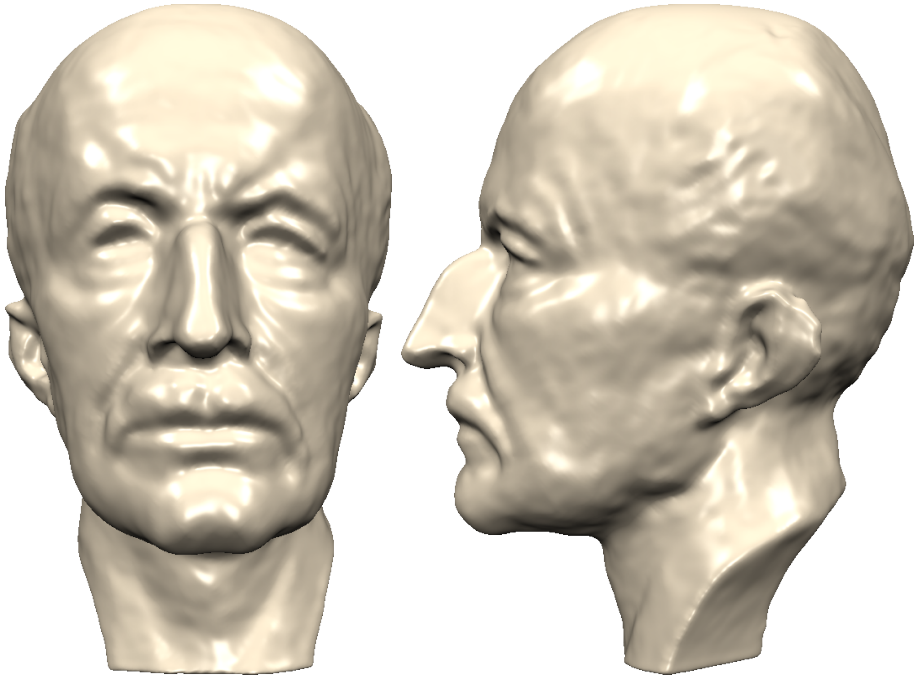


- Elimination of grid artifacts using error-controlled Bi-Laplacian smoothing
  - Based on surface confidence

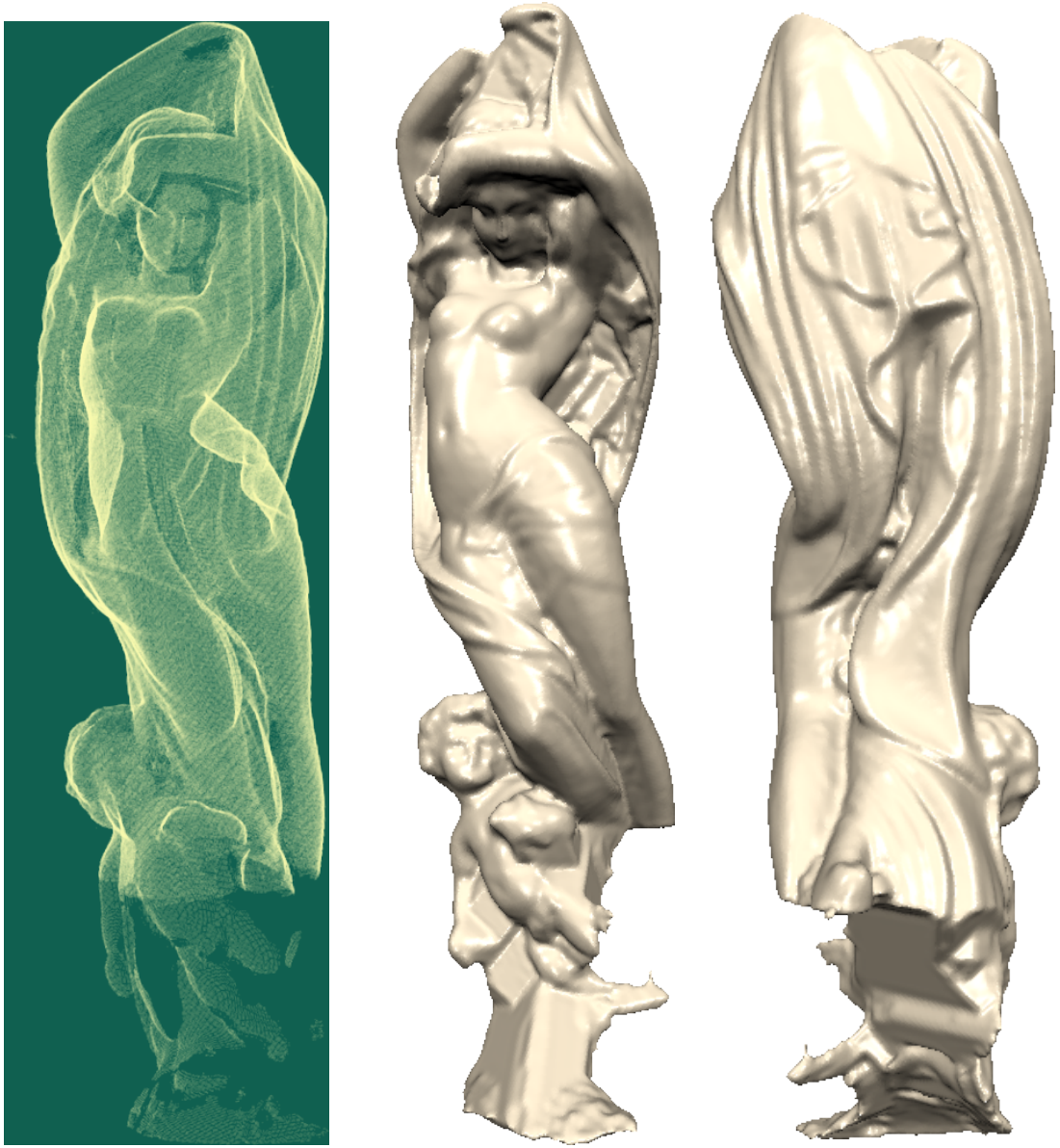
Example: Max Planck

- Note the hole in the point cloud viewed from the top
- Holes need to be “filled in”



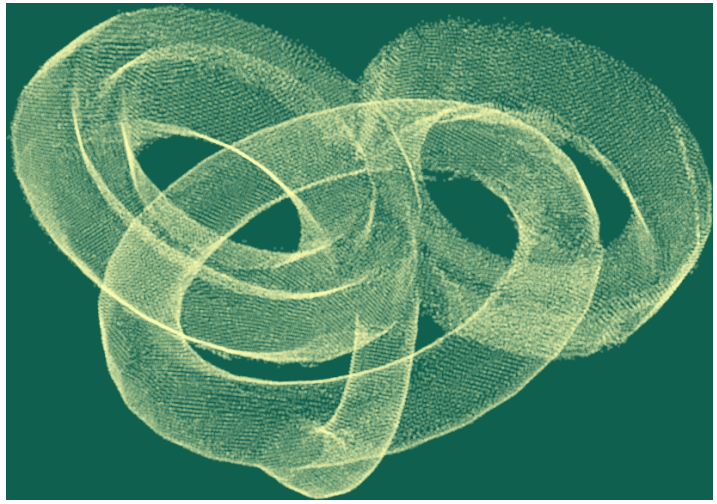


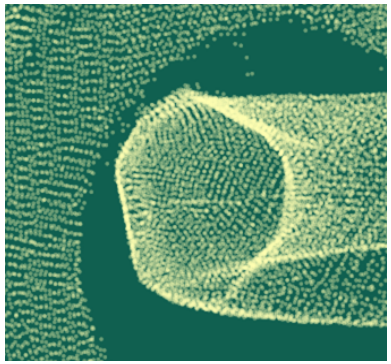
Example: Statue

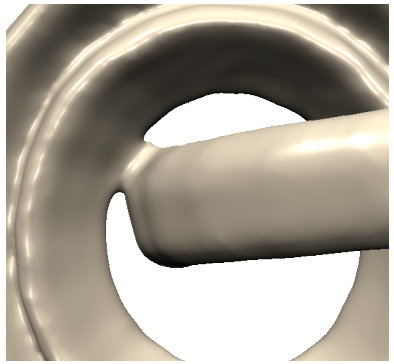


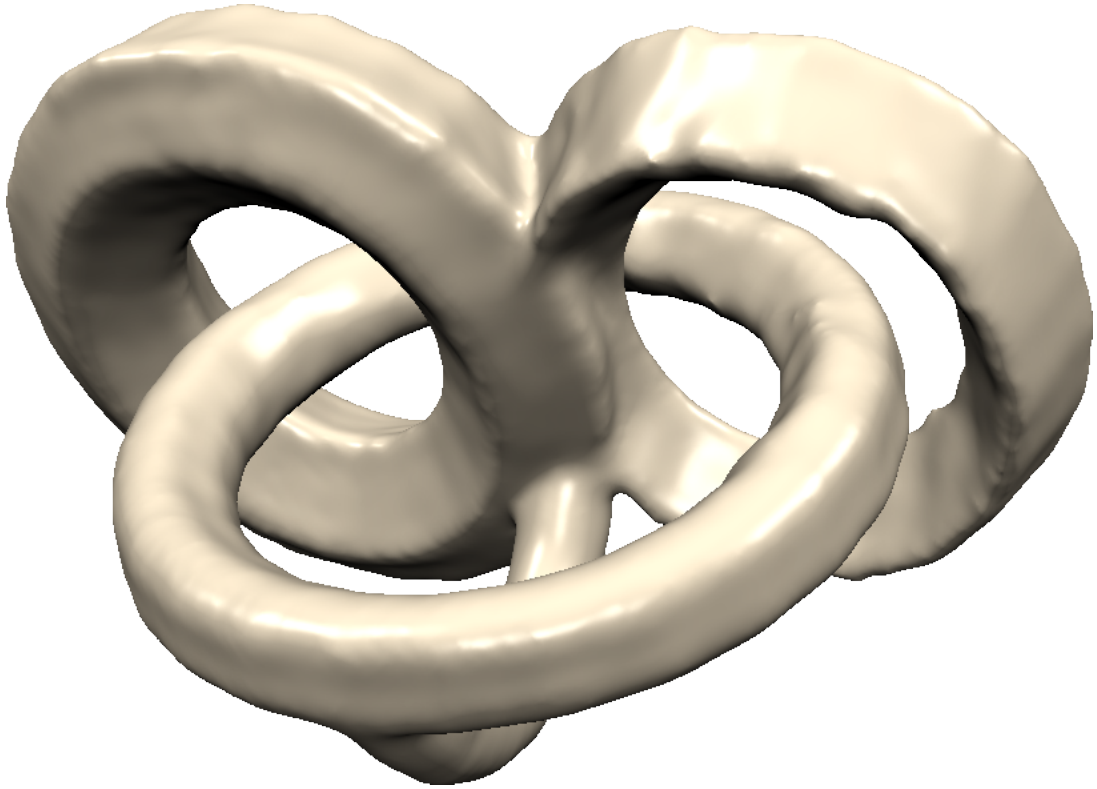
Example: Rings

- Note that certain parts of the mesh are connected, although they should not be (result of holes and noise in point cloud)



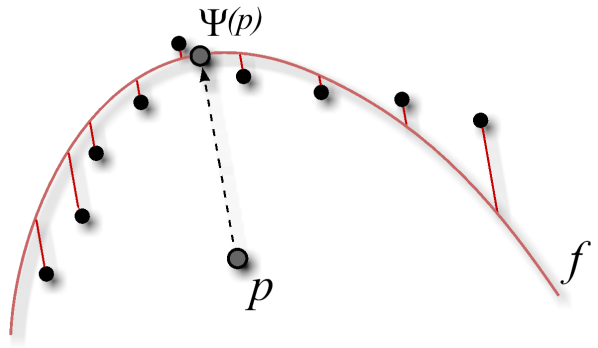


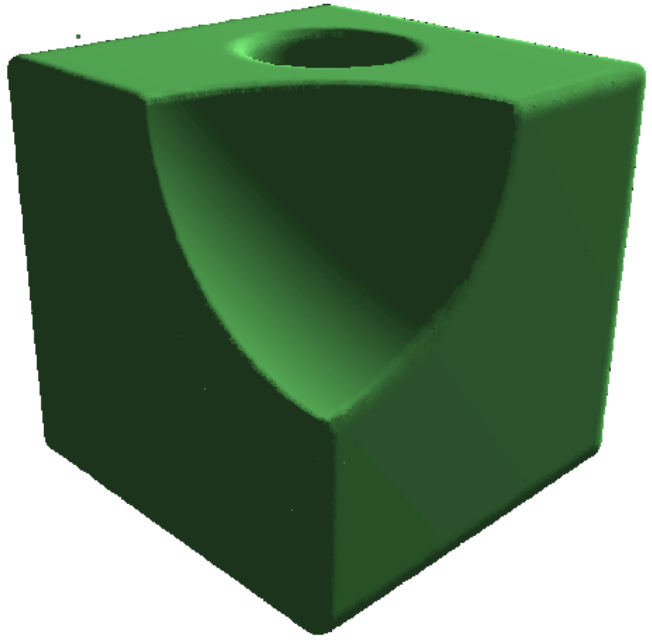




Point Set Surfaces

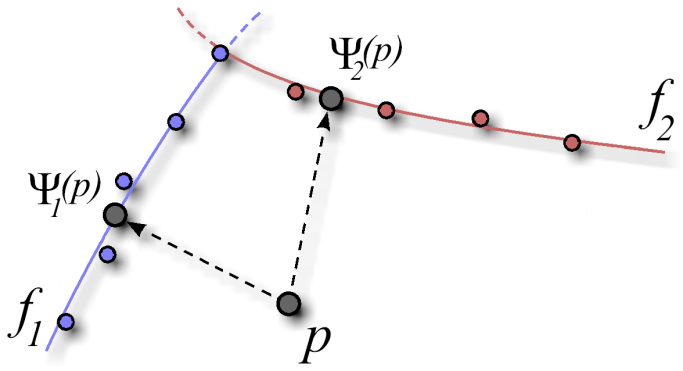
- Moving Least Squares (MLS)
  - Approximate surface fit over local neighborhood
  - Project point to the surface approximation
  - Relaxes noise and features arbitrarily

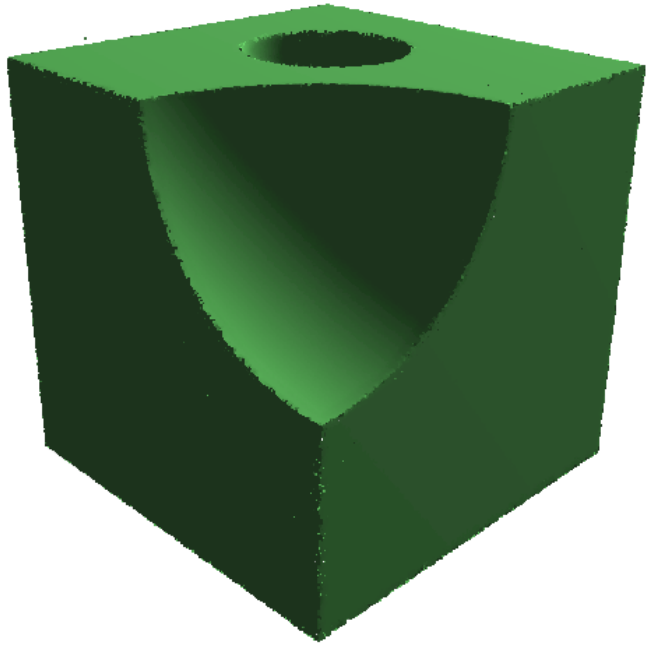




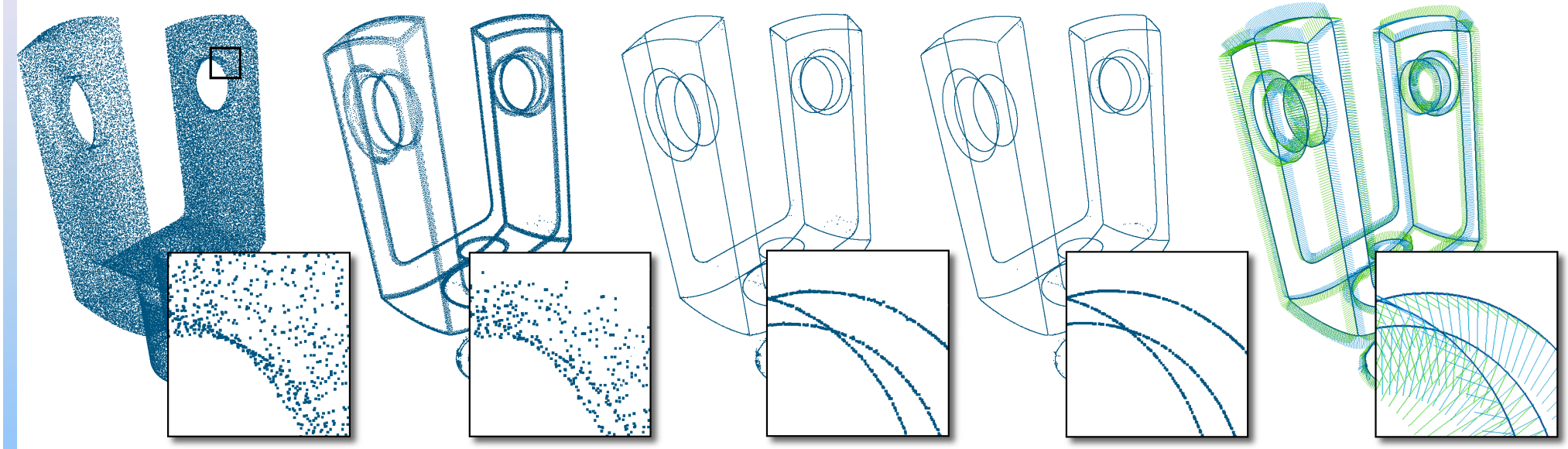
Point Set Surfaces

- Robust Moving Least Squares (RMLS)
  - Intelligent selection of local neighborhoods
    - Measured residual of an MLS fit
    - Threshold test determines appropriateness
    - Define multiple neighborhoods (surfaces)
    - Iteratively insert and remove points based on statistics of the residual error
  - Smooth noise while reconstructing sharp features





Algorithm Overview

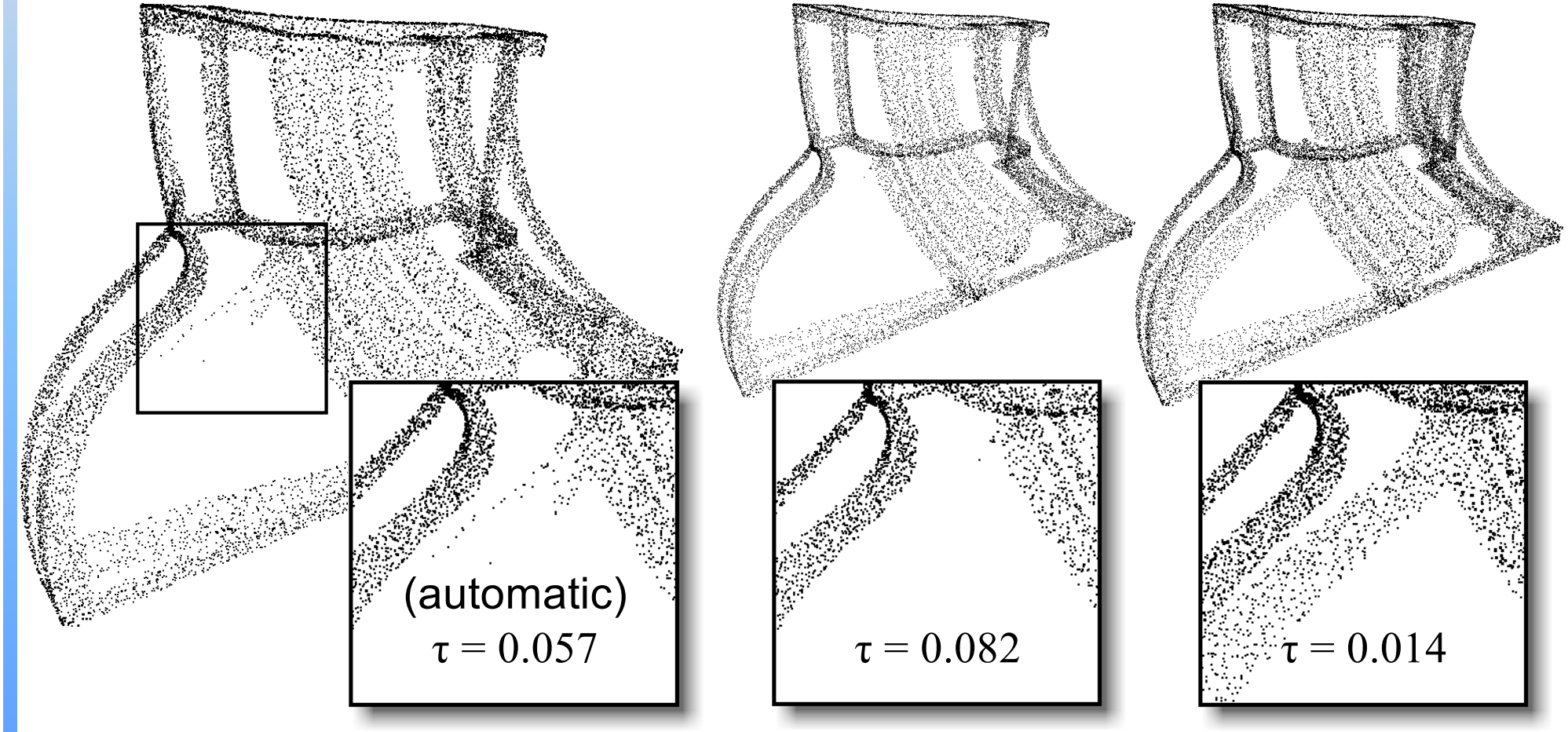


Input Cloud (Stage 1) (Stage 2) (Stage 3) (Stage 4) Output Features

1. Extract points near features
2. Project to edges using RMLS fit surfaces
3. PCA smoothing procedure
4. Grow feature curves through points

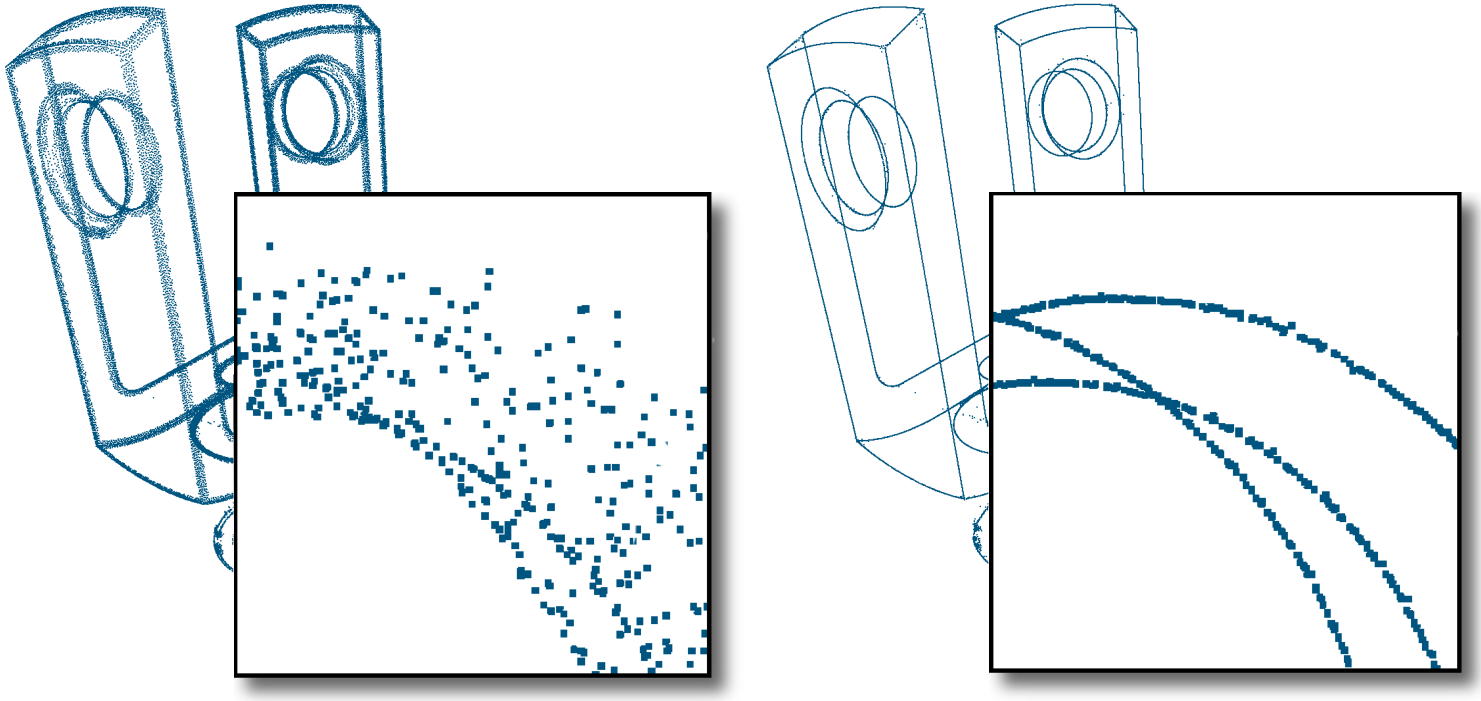
(Stage 1) Tolerance Tuning

- Residual: max distance of neighborhood points to the MLS surface fit
- Threshold test: Automatic (mean residual) vs. user tuned value

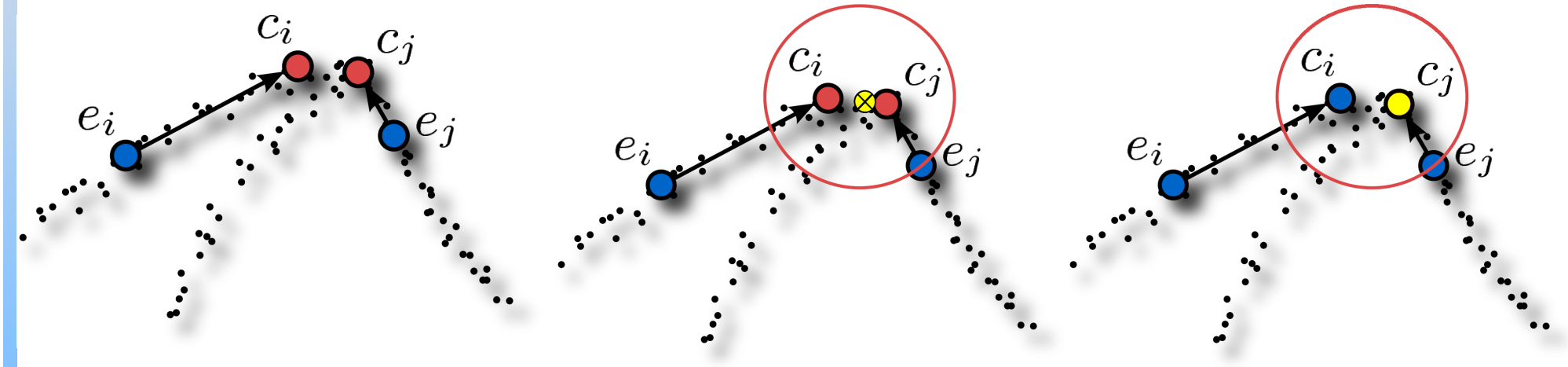


(Stage 2) Feature Projections

- RMLS defined surfaces
  - Newton method to find intersections
  - Project to the intersection of closest two surfaces (edge point)
  - Project to the intersection of three surfaces (corner point)
    - Associate weight as the inverse distance to original point



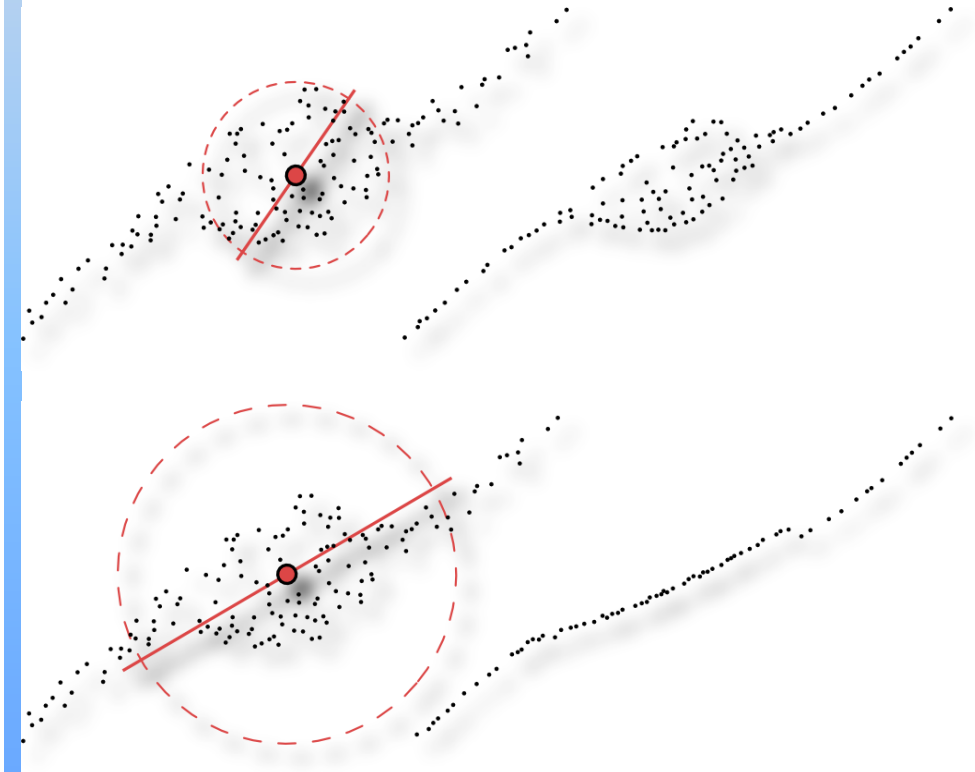
(Stage 2) Corner Computation

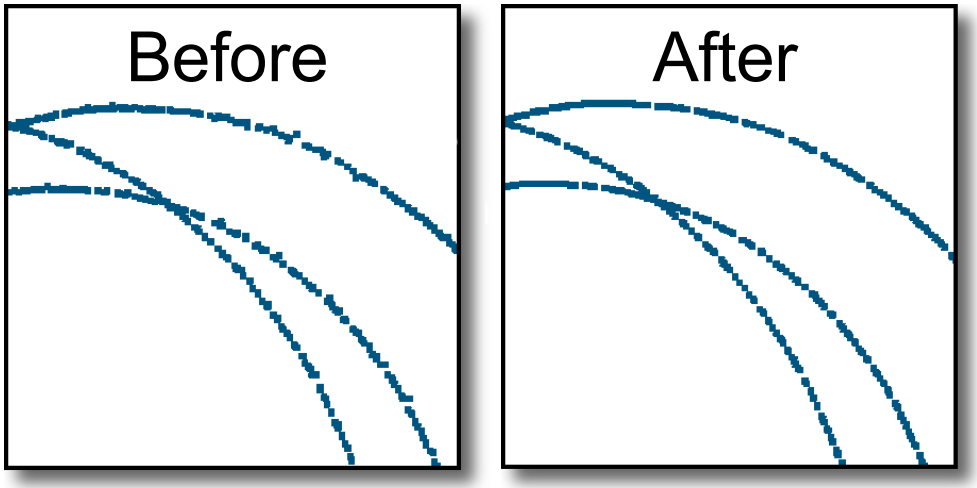


- Cluster corner points within a user defined radius
- Weighted average computed for each cluster
- Save closest corner point to average

(Stage 3) PCA Smoothing

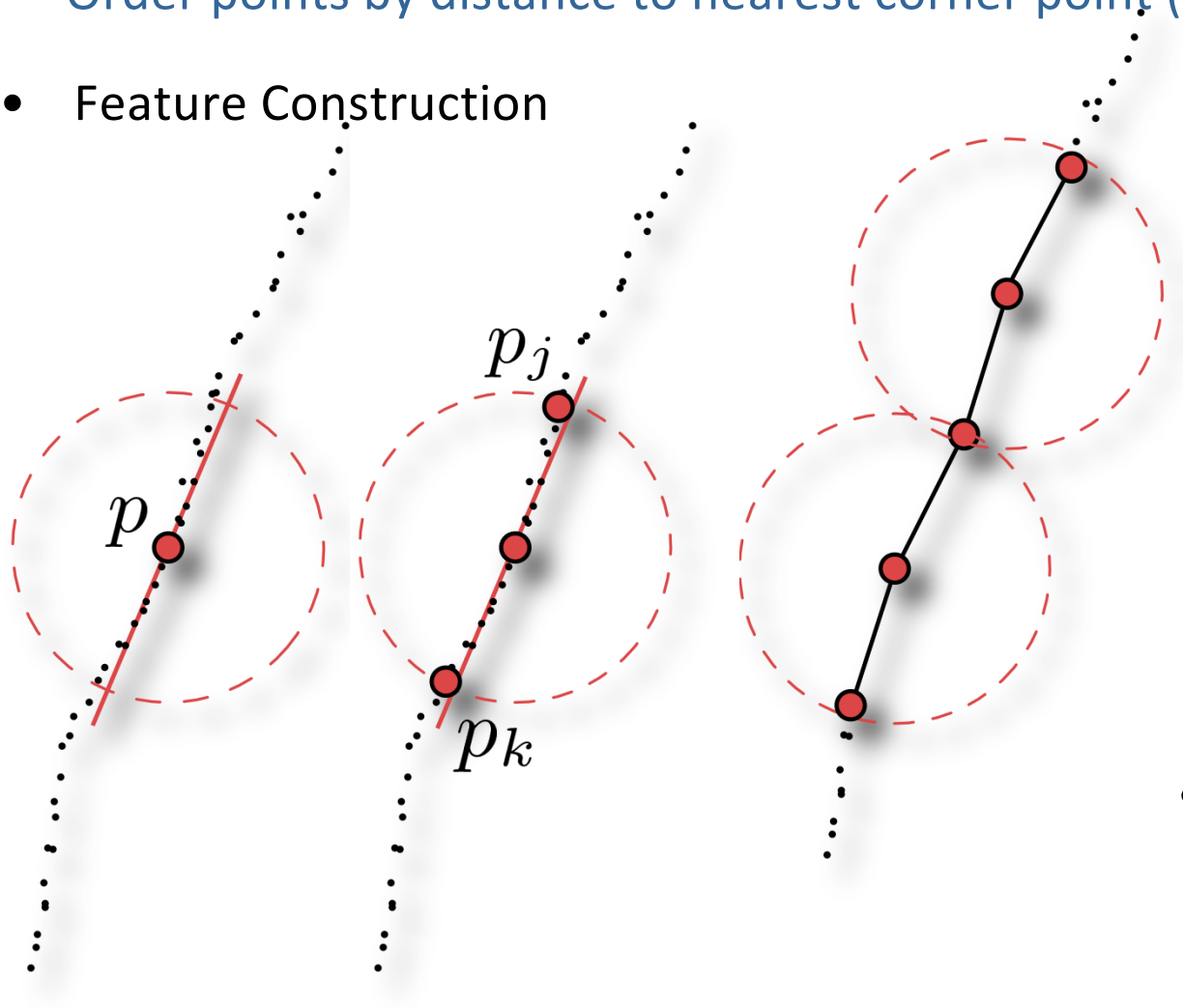
- PCA based smoothing removes RMLS noise
  - Neighborhood size determined by correlation to principle component

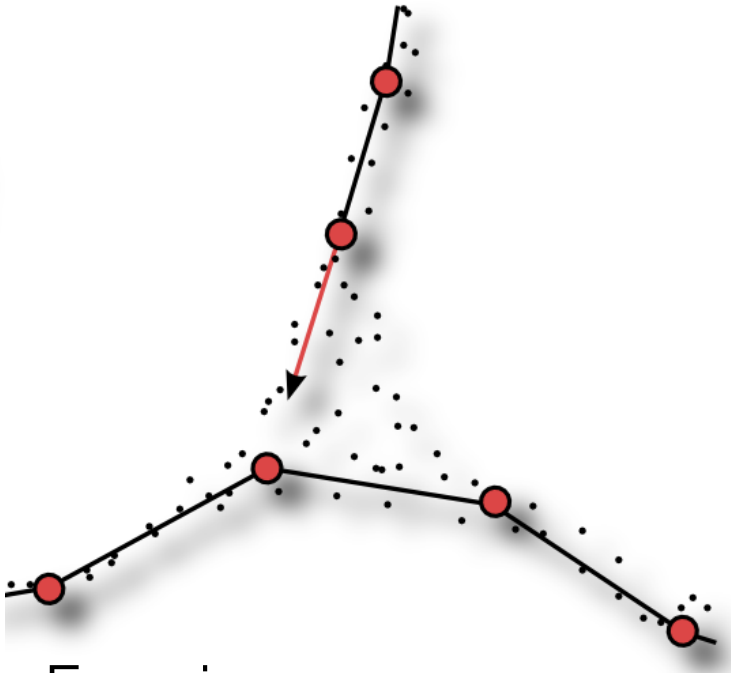




(Stage 4) Feature Growth

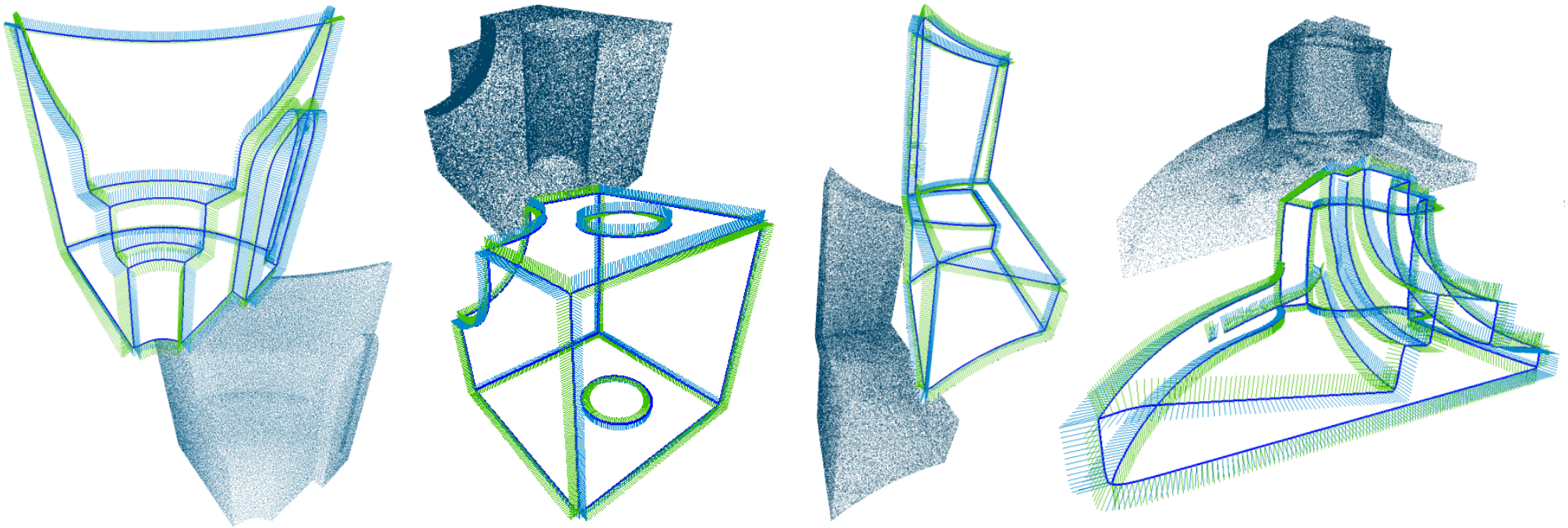
- Priority Queue
  - Order points by distance to nearest corner point (furthest to closest)
- Feature Construction





- Forming new corners

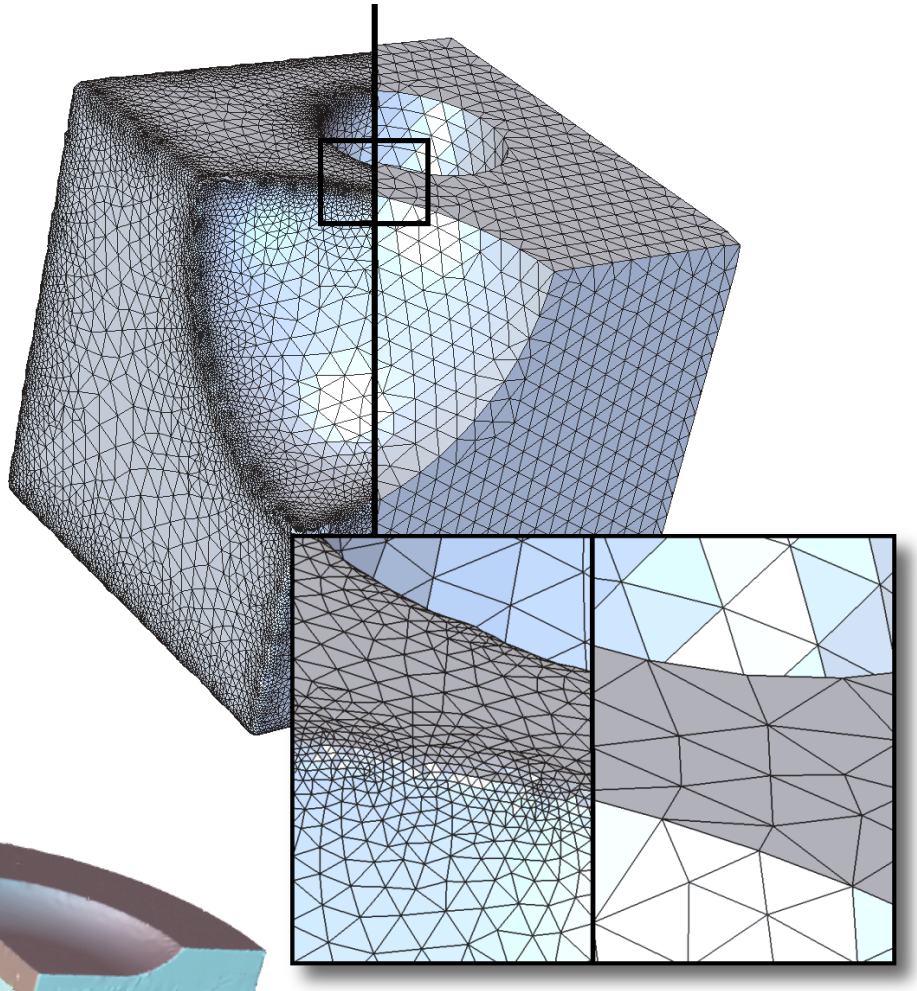
Constructed Features

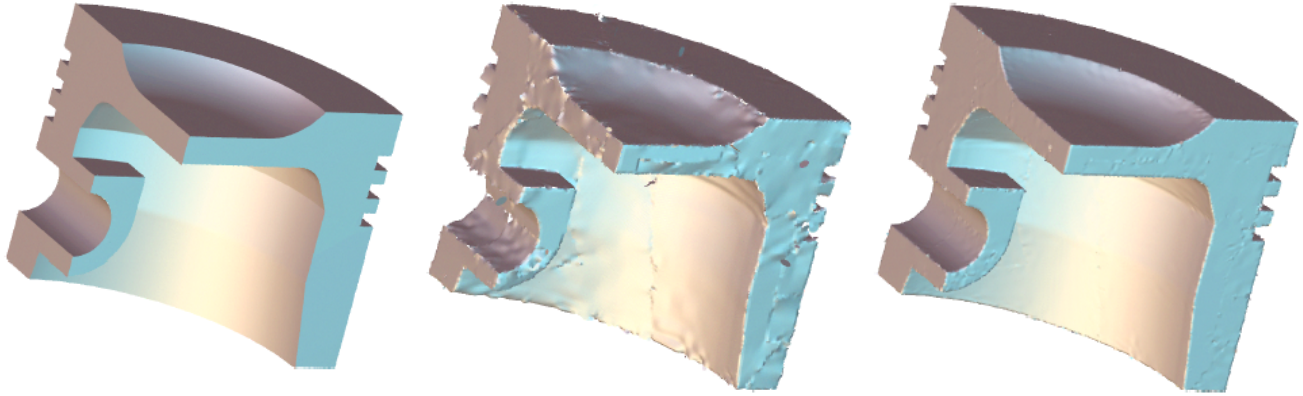


- Surface normals associated with curves during growth
- Relaxation iteration removes perturbations

Applications – Feature Meshing

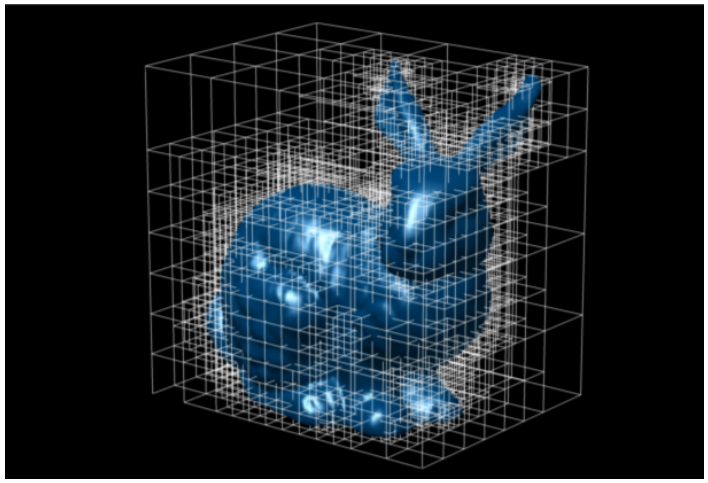
- Re-mesh segmented regions
- Achieve sharp features without modification of algorithm
- Reduced triangle count
  - Triangle sizes are determined by the curvature *along* the feature edge rather than *across* it!
- Compression





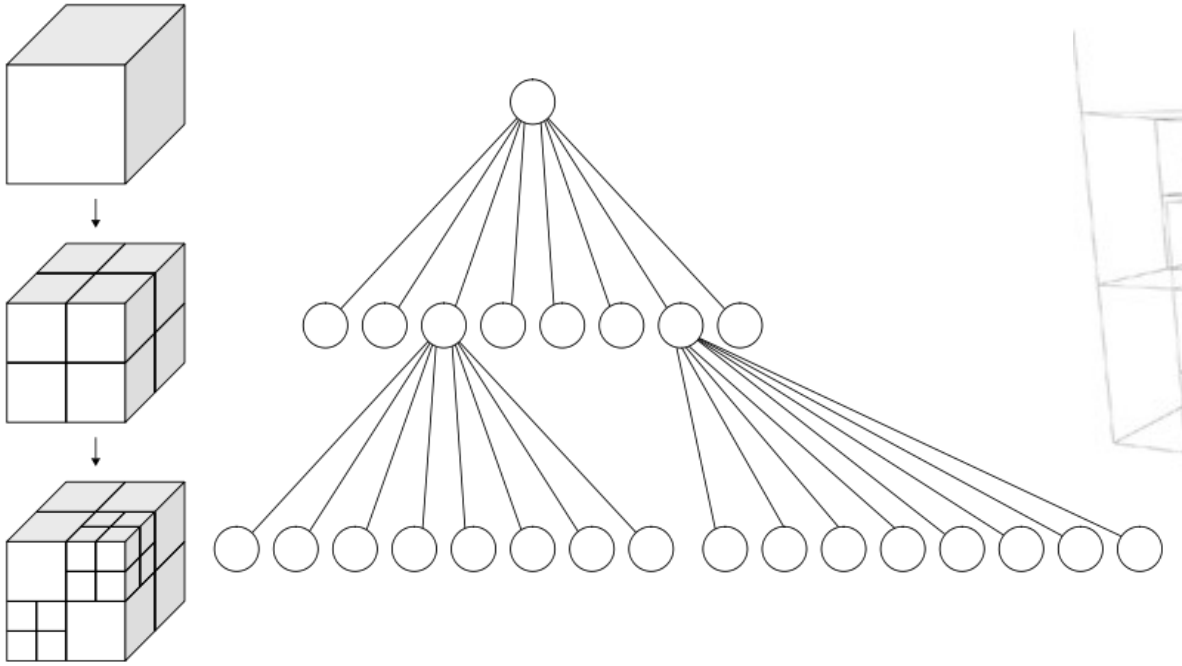
3D space partitioning

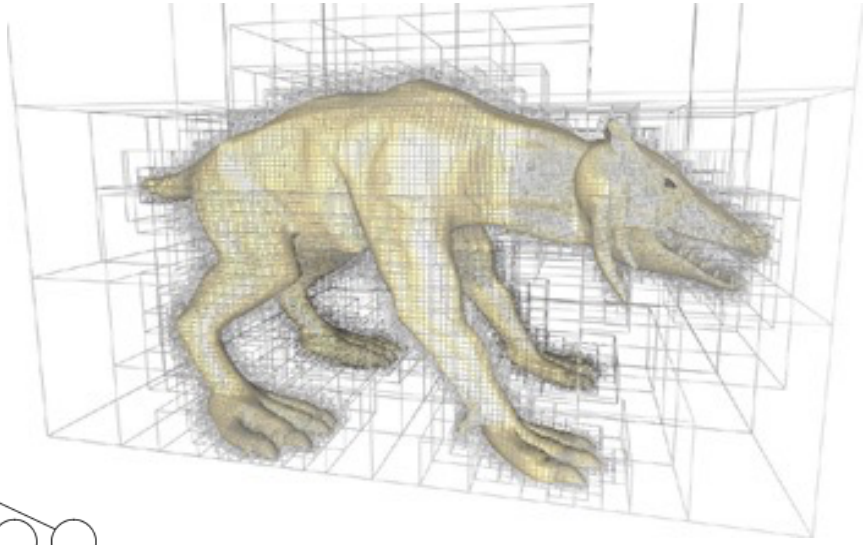
- 3D space is divided into non-overlapping regions
  - Often organized as a hierarchical tree data structure
    - Each region is recursively subdivided by applying the same space-partitioning system.
- Often used in CG to quickly perform certain kinds of geometry queries
  - Ray tracing – What did the player just shoot?
  - Frustum culling - What can the player see?
  - Collision detection – Did the player hit a wall?
- 3 commonly used space partitioning algorithms
  - K-d tree, octree, BSP tree



Octree

- Octree
  - Extension of the quad tree (in 2D)
  - Starts with a bounding box of the entire scene
  - On each level cubes are subdivided in 8 smaller cubes of the same size
    - Further subdivision only when the voxel is occupied



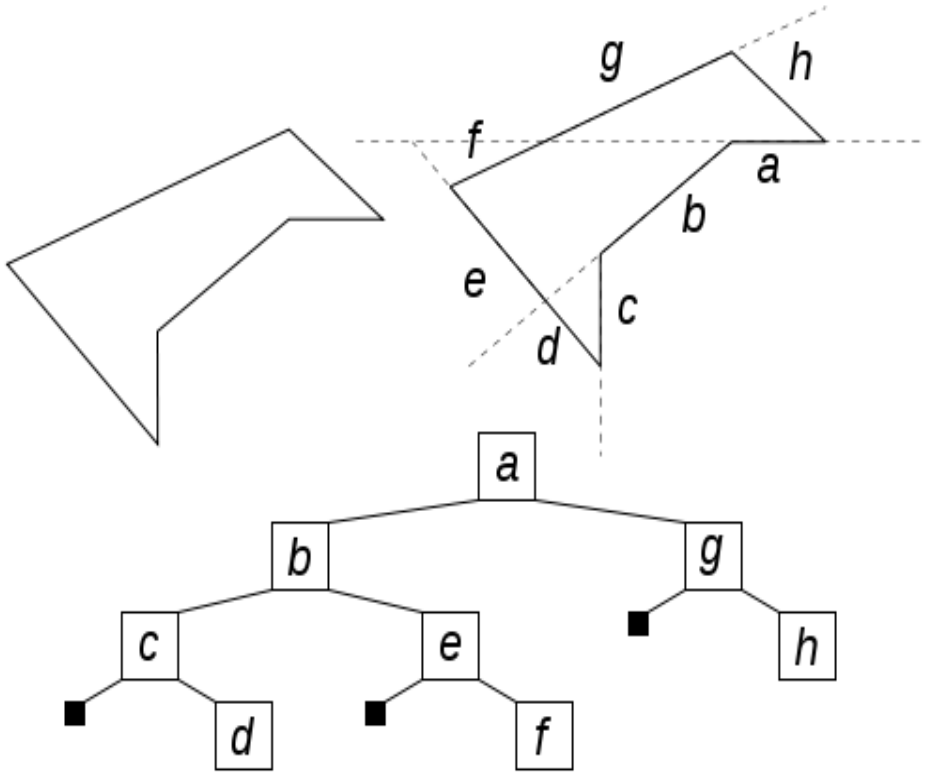


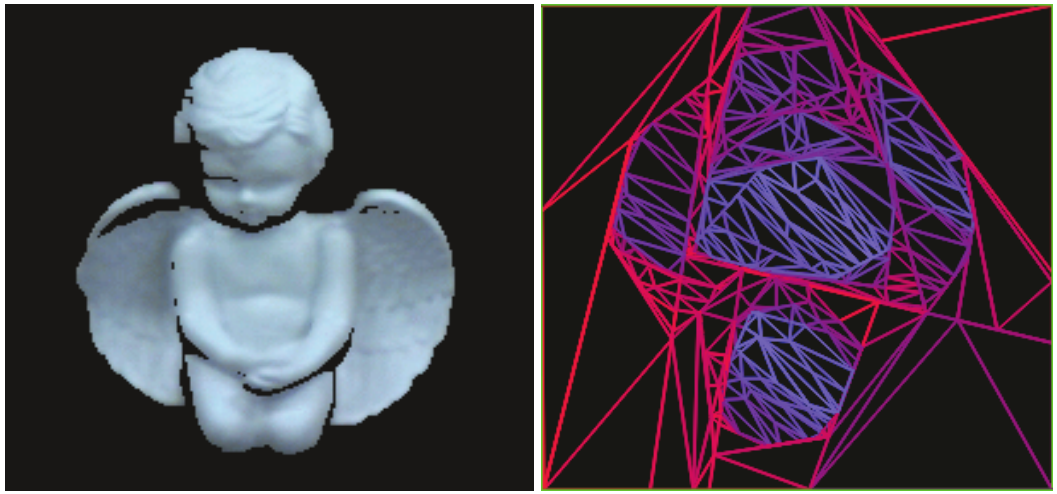
Octrees: What Are They Good For?

- Handling Observer-Object Interactions
  - Subdivide the octree until each leaf's region intersects only a small number of objects.
  - Each leaf holds a list of pointers to objects that intersect its region.
  - Find out which leaf the observer is in. We only need to test for interactions with the objects pointed to by that leaf.
- Inside/Outside Tests for Odd Shapes
  - The root node represent a square containing the shape.
  - Then the octree contains information allowing us to check quickly whether a given point is inside the shape.
- Sparse Arrays of Spatially-Organized Data
  - Store array data in the quadtree or octree.
  - Only subdivide if that region of space contains interesting data.

BSP tree

- Binary space partitioning (BSP)
  - Binary search tree
  - Generalization of K-d tree
  - Hyperplanes may have any orientation
    - Rather than being aligned with the coordinate axes as they are in K-d trees



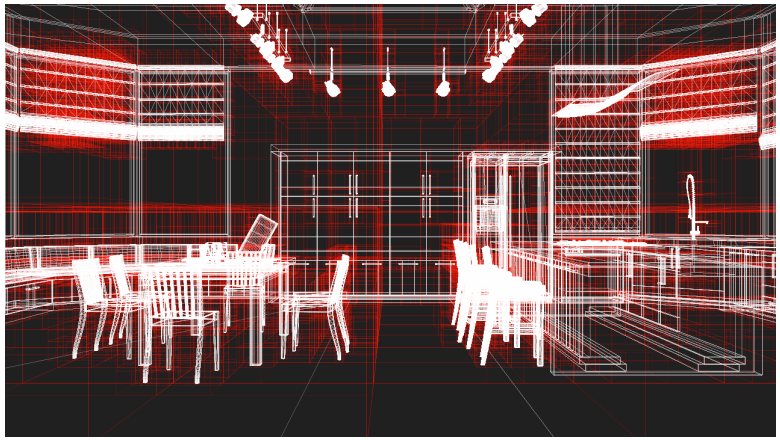


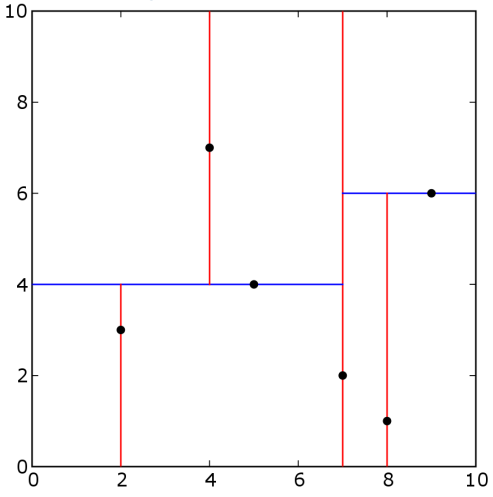
BSP Trees: What Are They Good For?

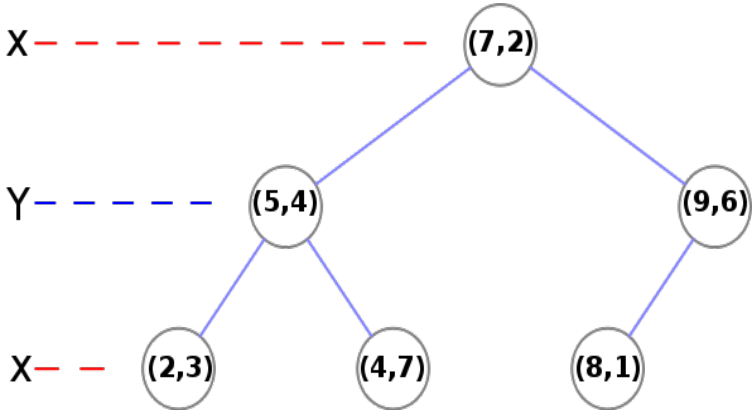
- BSP trees are primarily useful when a back-to-front or front-to-back ordering is desired:
  - For hidden-surface removal (HSR).
  - For translucency via blending.
- Since it can take some time to construct a BSP tree, they are useful primarily for:
  - Static scenes.
  - **Some** dynamic objects are acceptable.
- BSP-tree techniques are generally a waste of effort for small scenes. We use them on:
  - Large, complex scenes.

K-d tree

- K-dimensional tree
  - Binary search tree
  - Partitioning is applied on alternating dimension
  - Split only one dimension at a time (i.e. x or y or z)







(2D in the picture on the right)

White lines are the wireframe lines for the geometry itself, red lines represent KD-Tree node boundaries