Topologically correct reconstruction of tortuous contour forests

John Edwards Chandrajit Bajaj

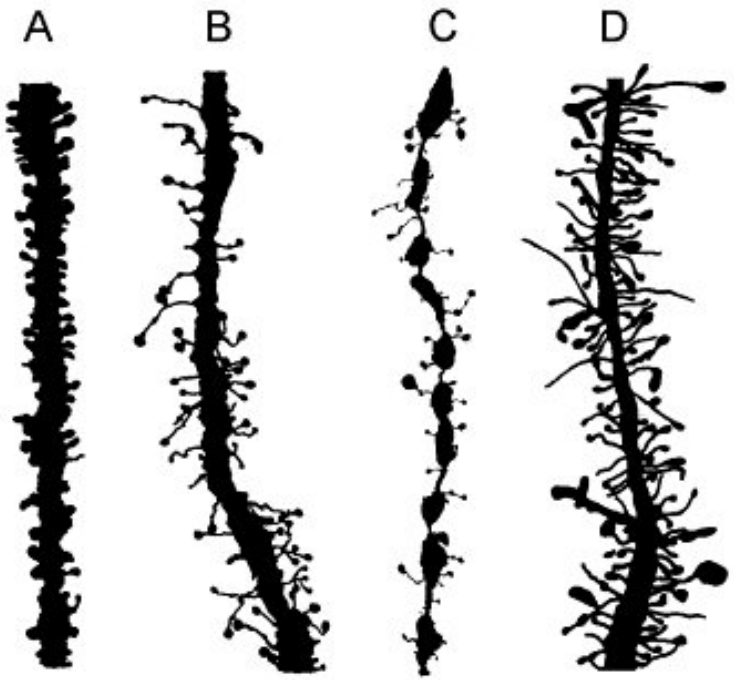
Department of Computer Science
Computational Visualization Center
Institute of Computational Engineering and Sciences
University of Texas at Austin

Symposium on Solid and Physical Modeling, 2010

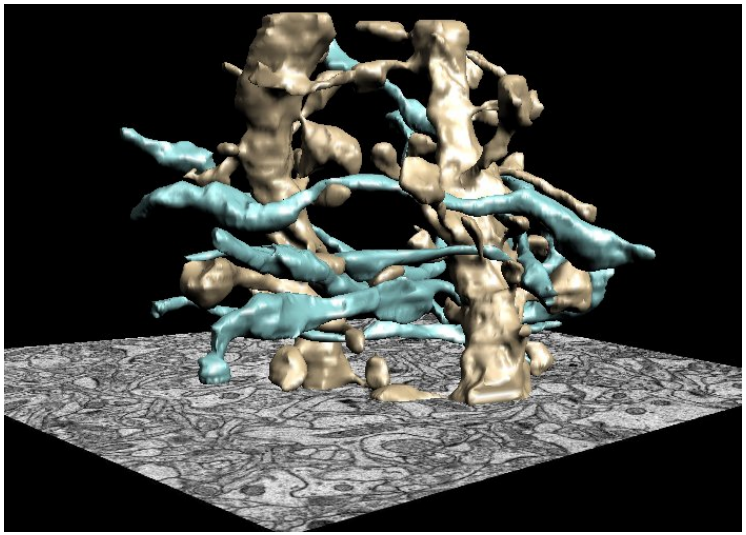
Outline

Outline

Motivation - neuronal modeling



Fiala, Spacek, and Harris 2002

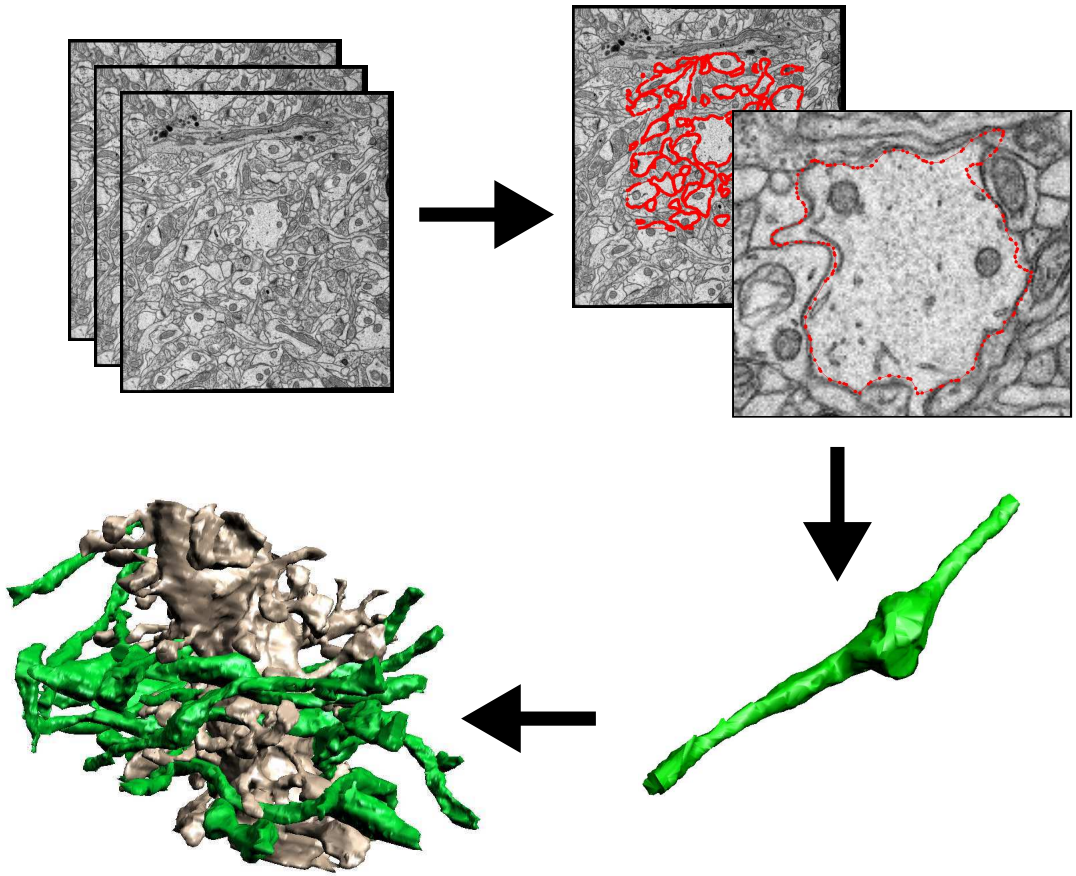


- Dendritic spine distribution and shape in the hippocampus correlates with certain types of brain disease.
  - A Neurologically normal (6 months).
  - B Mentally retarded (12 months).
  - C Alzheimer's (adult).
  - D Fragile X syndrome (adult).
- Simulating the effect of spine modification requires accurate geometry and stable methods for modeling electric activity.
- This talk will focus on creating accurate meshed geometries suitable for electrophysiological finite-element simulations.

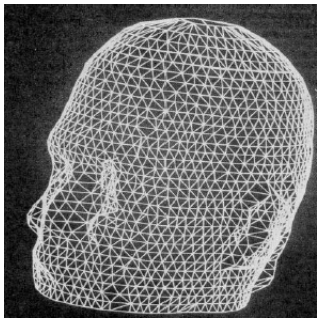
Reconstruction

The high-level outline of a typical 3D reconstruction process is as follows:

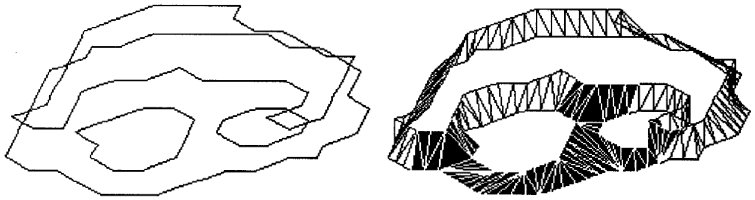
- Begin with series of 2D Electron Microscopy (EM) images
- Generate contours around components of interest (axons, dendrites, etc in our case)
- Use some method to reconstruct 3D objects from 2D contours
- Combine 3D objects into a forest of structures



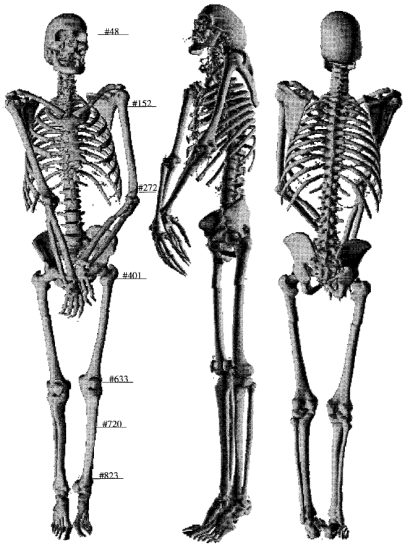
Related work - single-component reconstruction



[6]



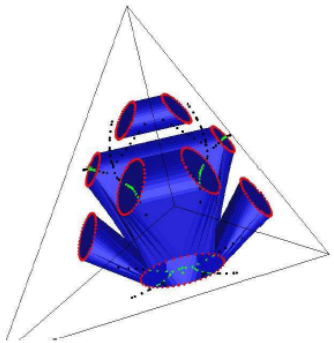
[3]



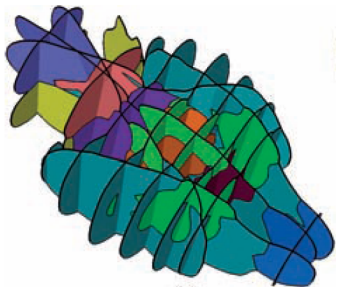
[1]

- One of the seminal works was Fuchs et al. [6] who posed the problem and presented a triangulation solution.
- Barequet and Sharir [3] introduced a method using linear interpolations between slices of medical images.
- Bajaj et al. [1] expanded on their work to support arbitrary topologies.

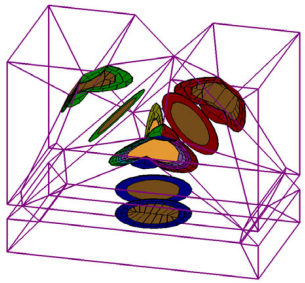
Related work – multi-component reconstruction



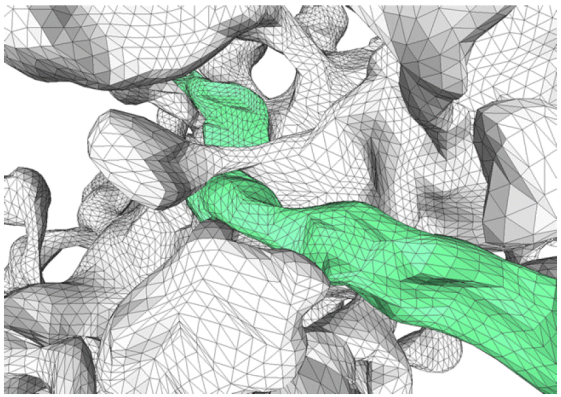
[5]



[7]



[4]



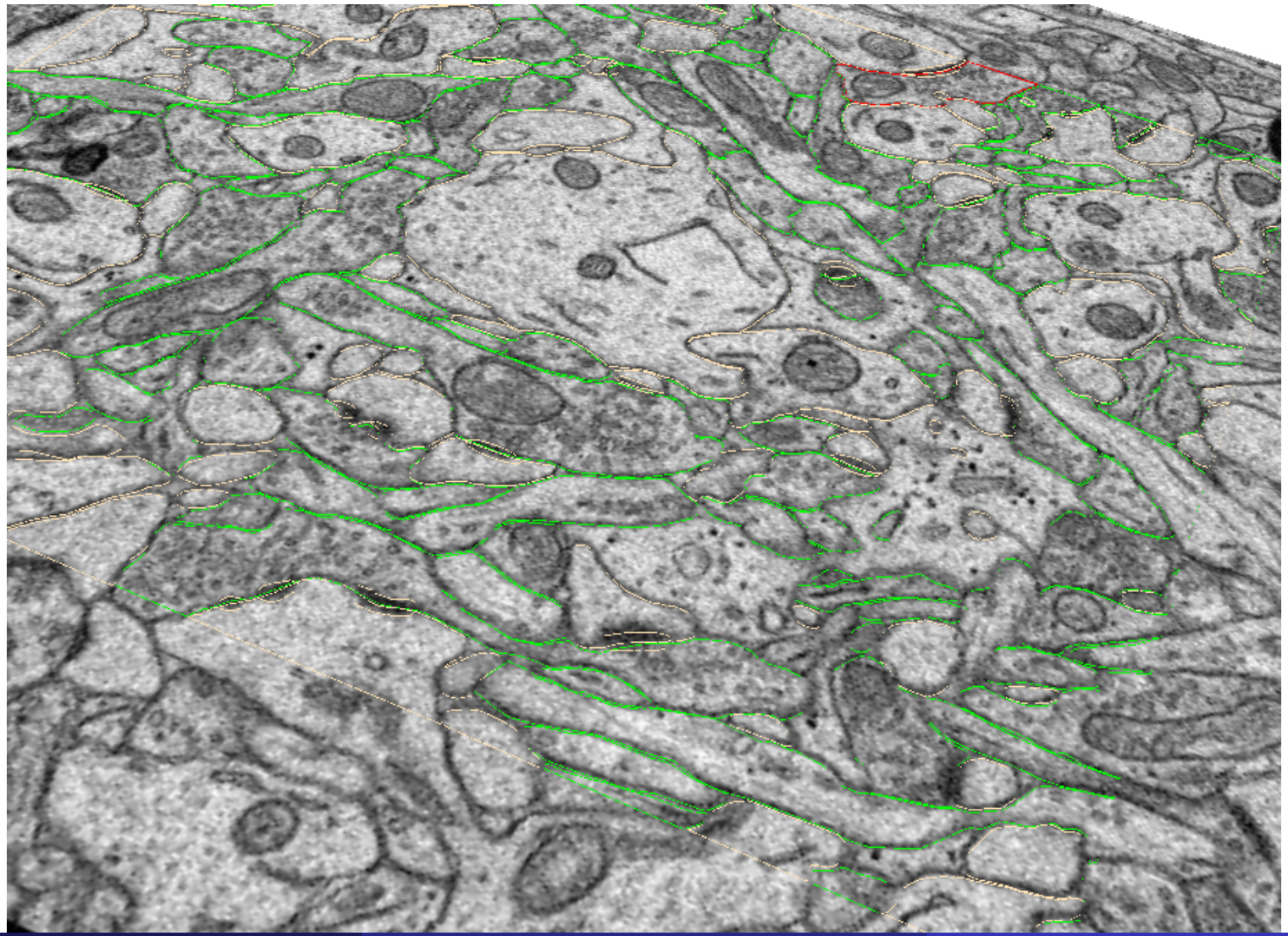
[2]

- Recent work by Boissonnat and Memari [5] reconstructs single structures from non-parallel slices.
- Two approaches by Liu et al. [7] and Barequet and Vaxman [4] reconstruct from non-parallel slices and additionally reconstruct multiple components at the same time, avoiding inter-component intersections.
- Bajaj and Gillette [2] perform single component reconstruction using [1] and then remove intersections by removing contour overlaps in intermediate planes.

Outline

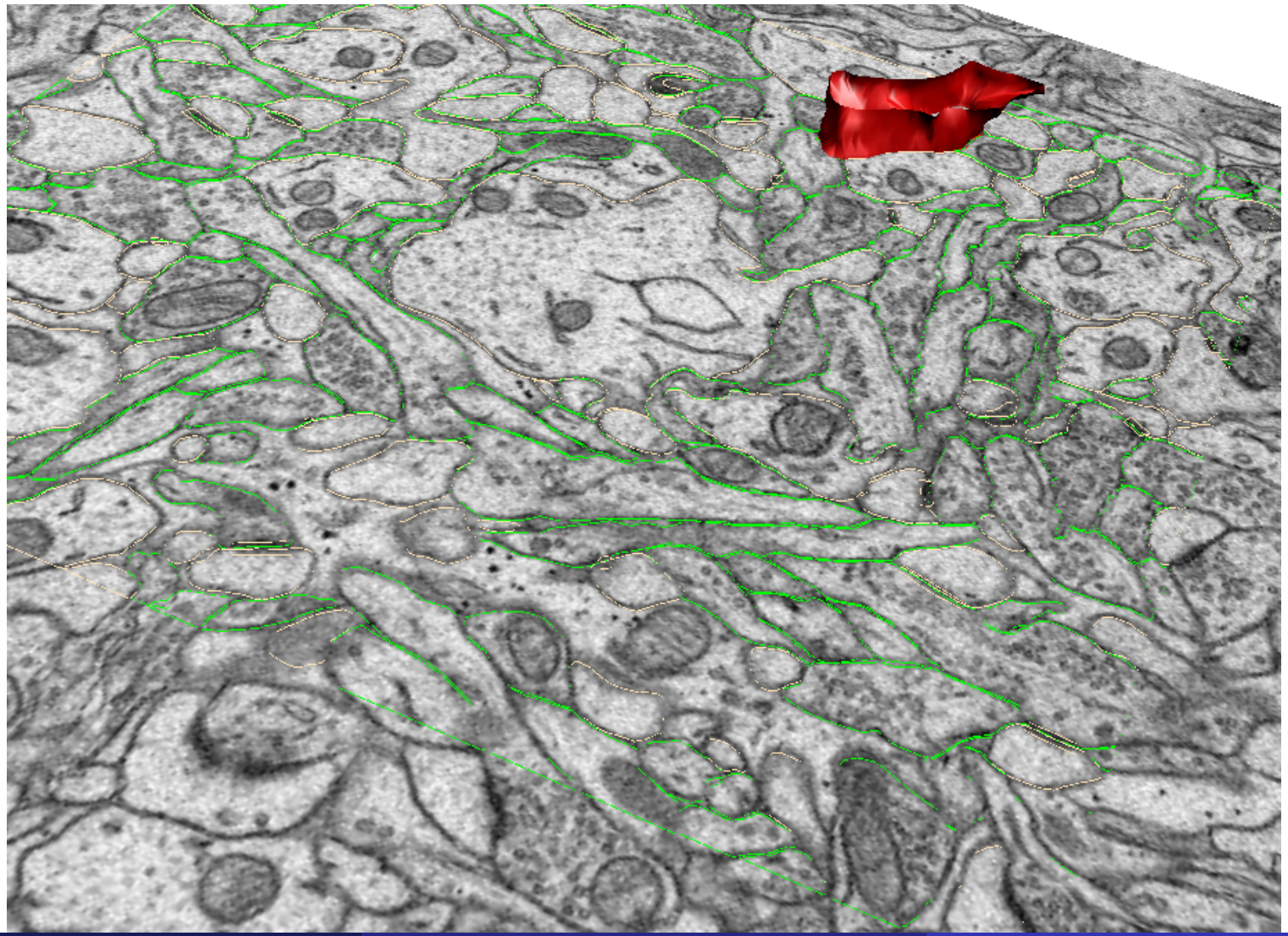
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



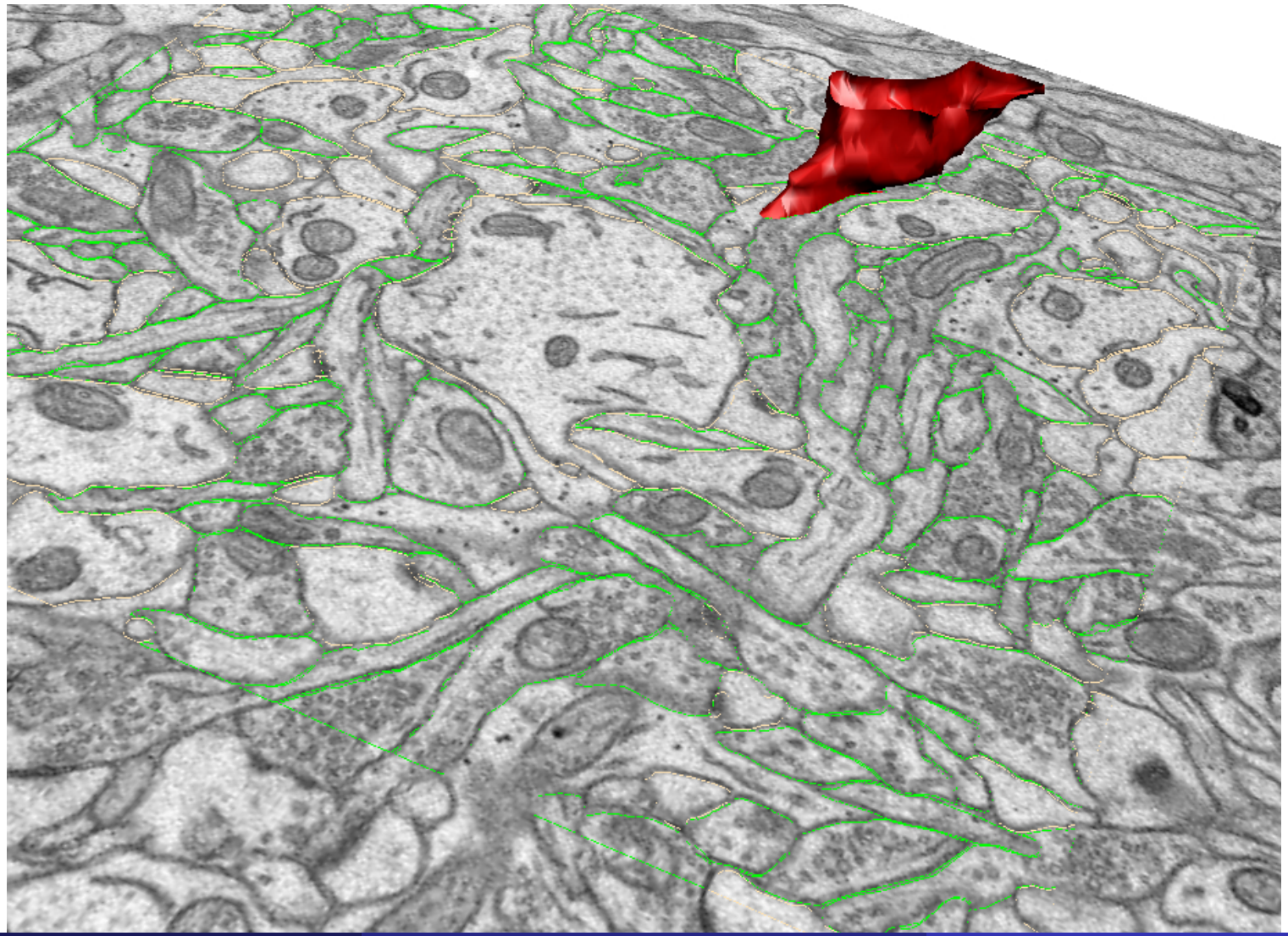
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



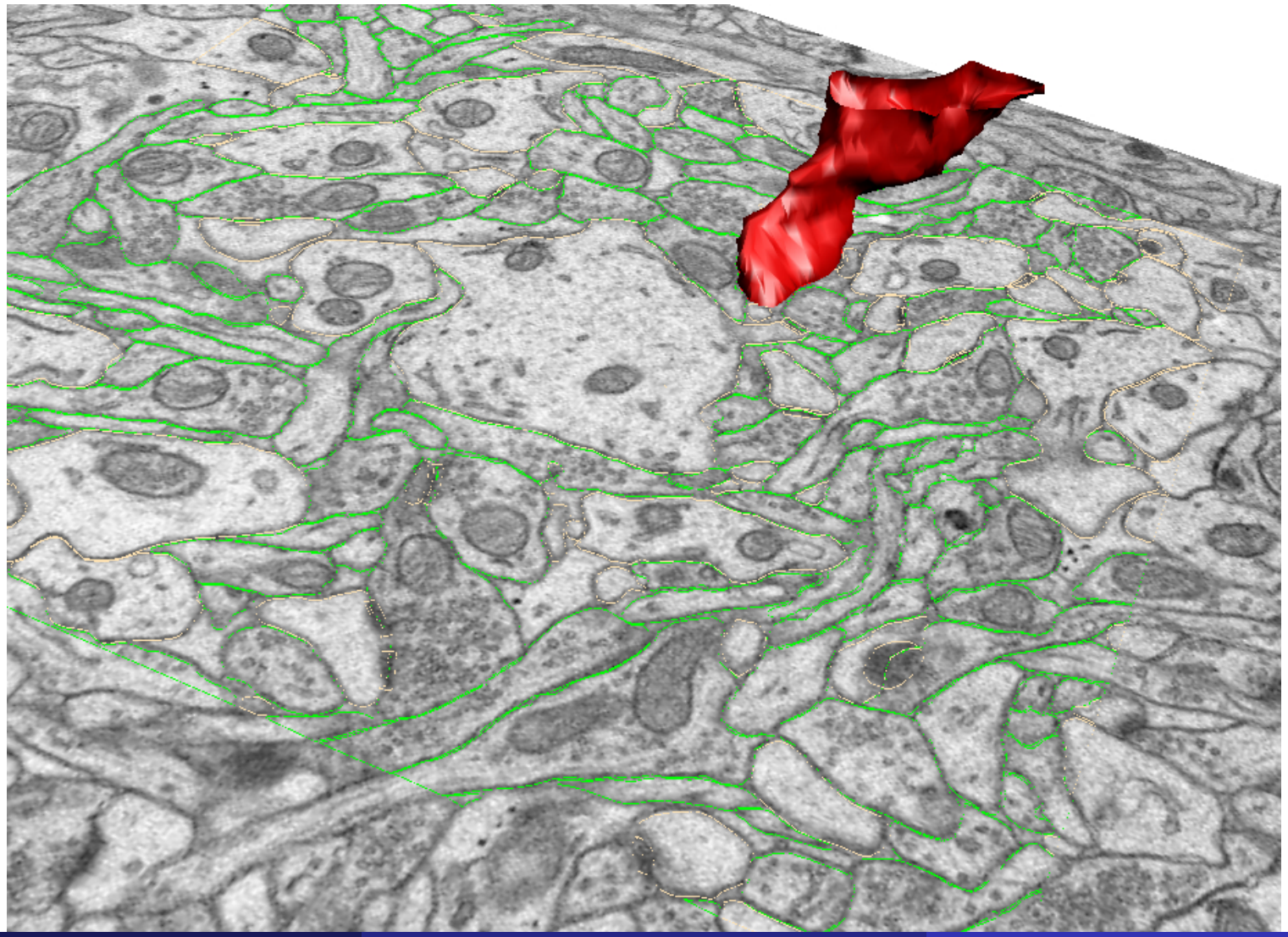
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



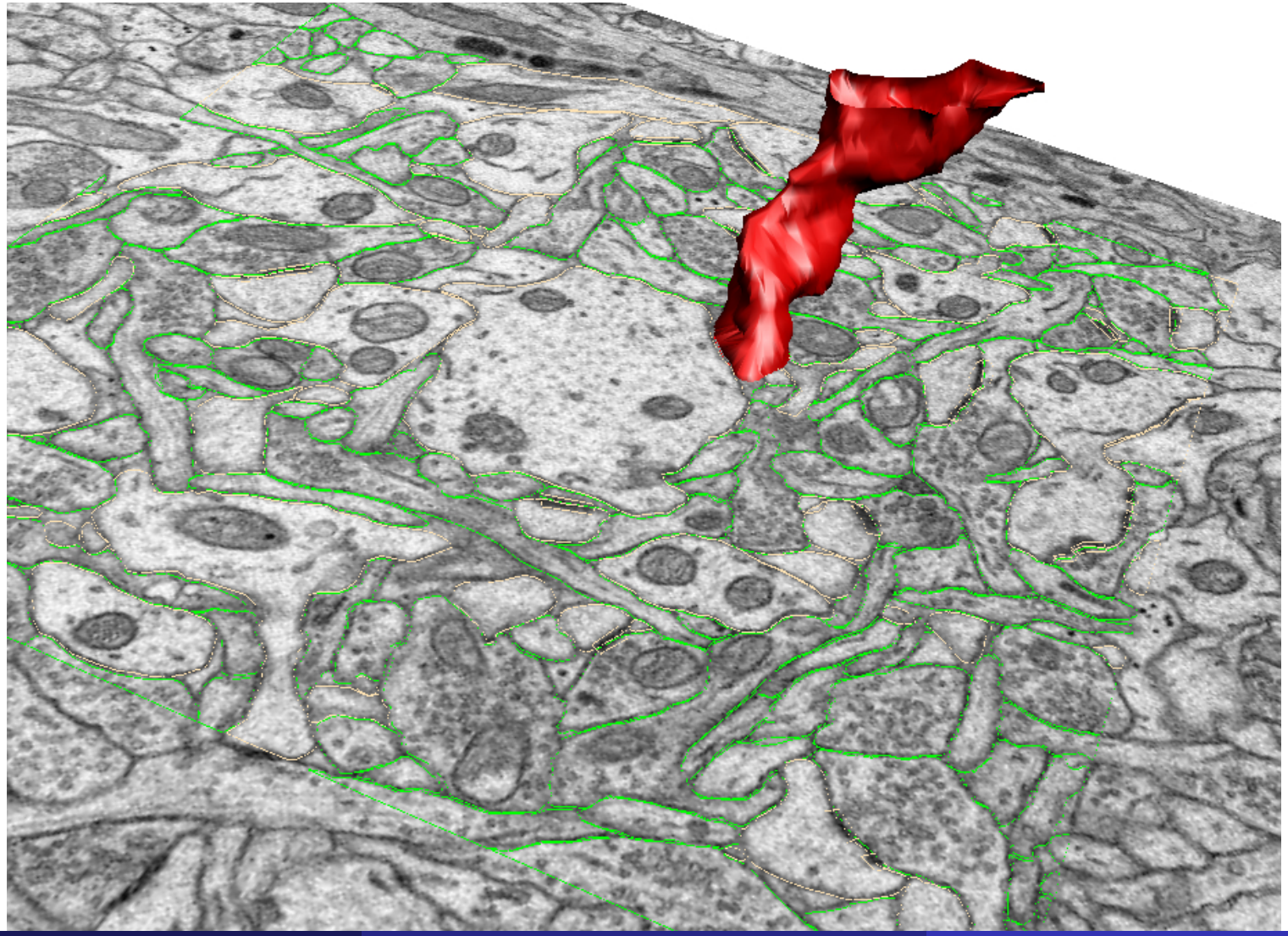
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



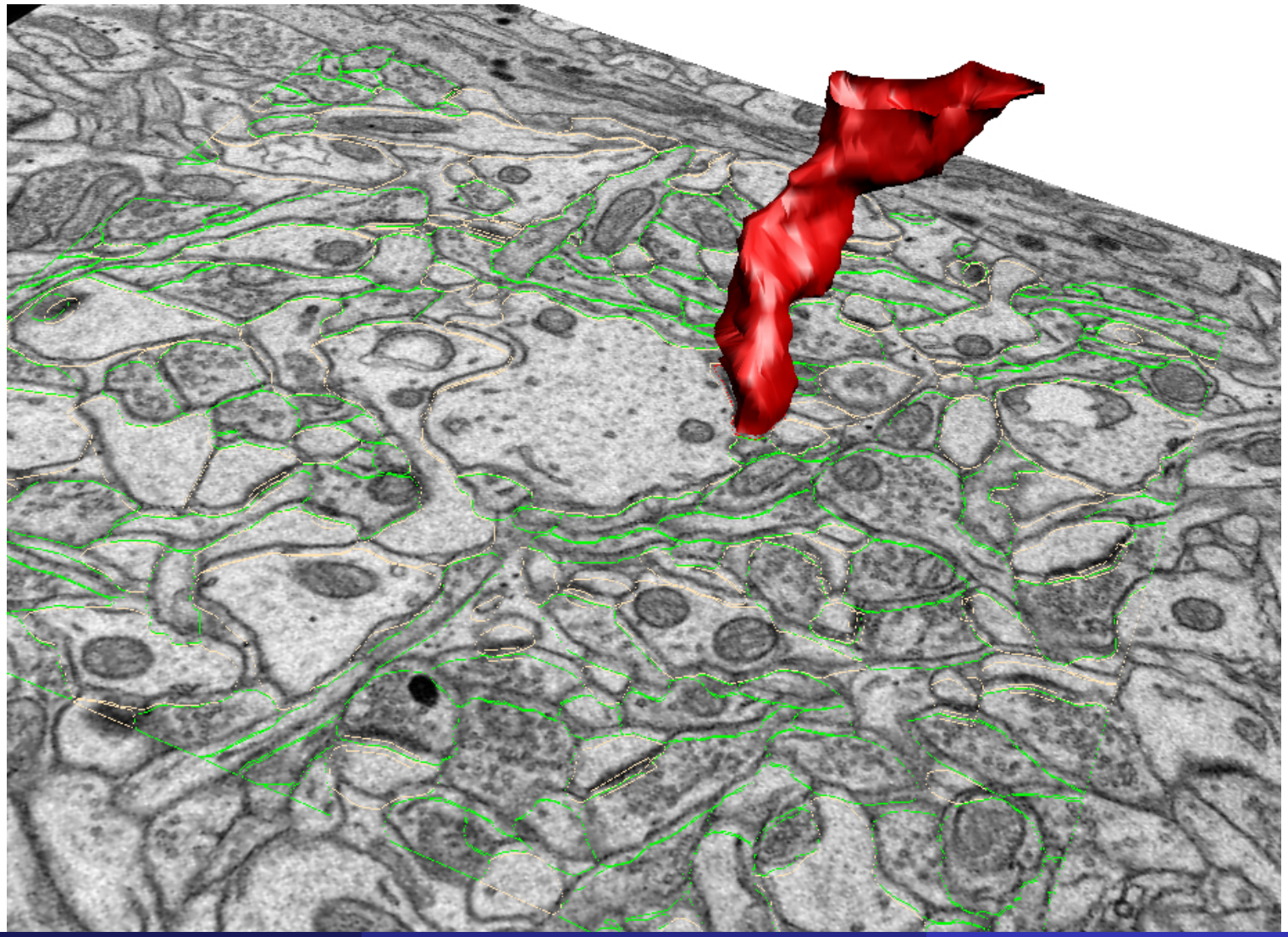
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



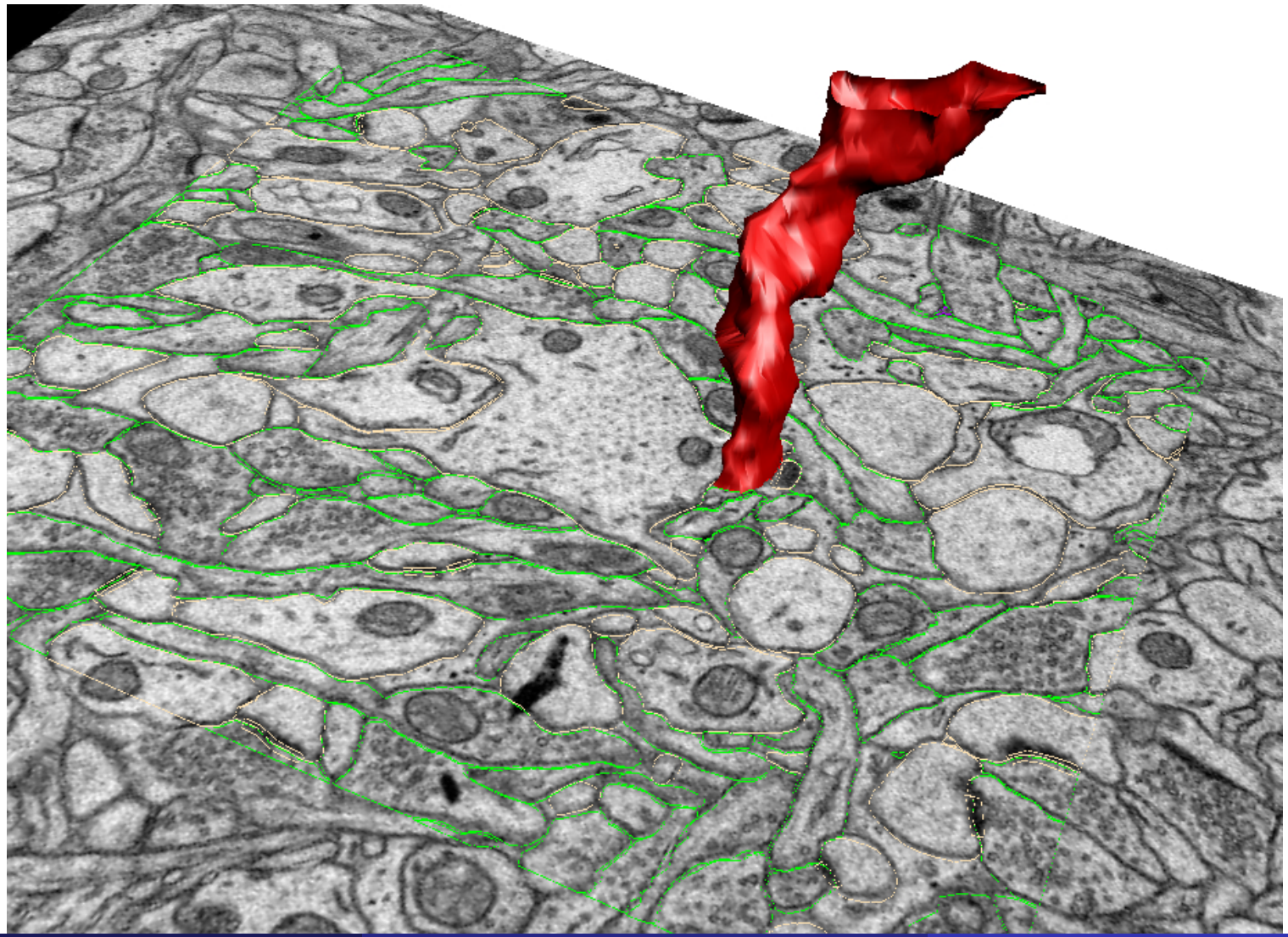
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



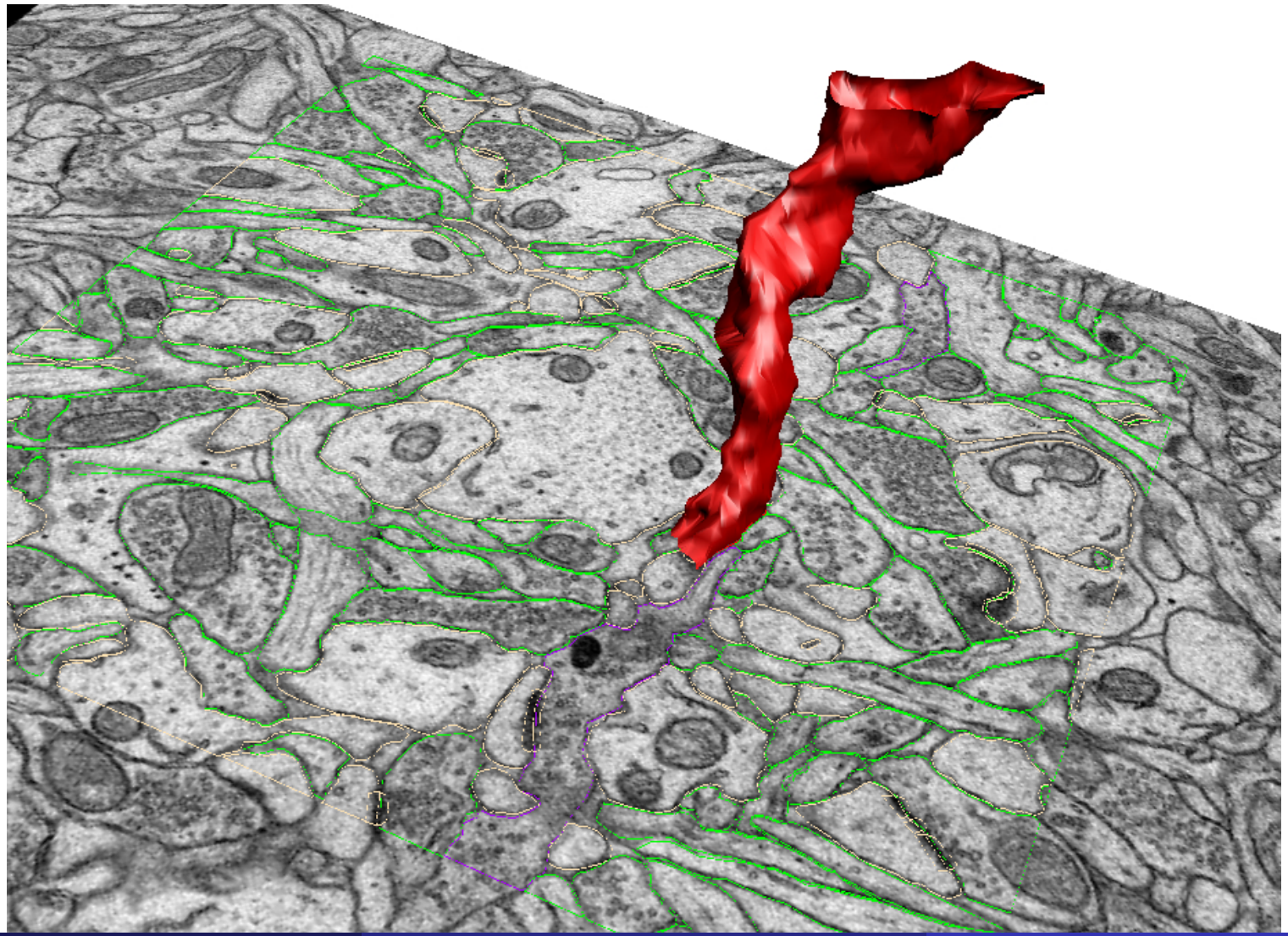
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



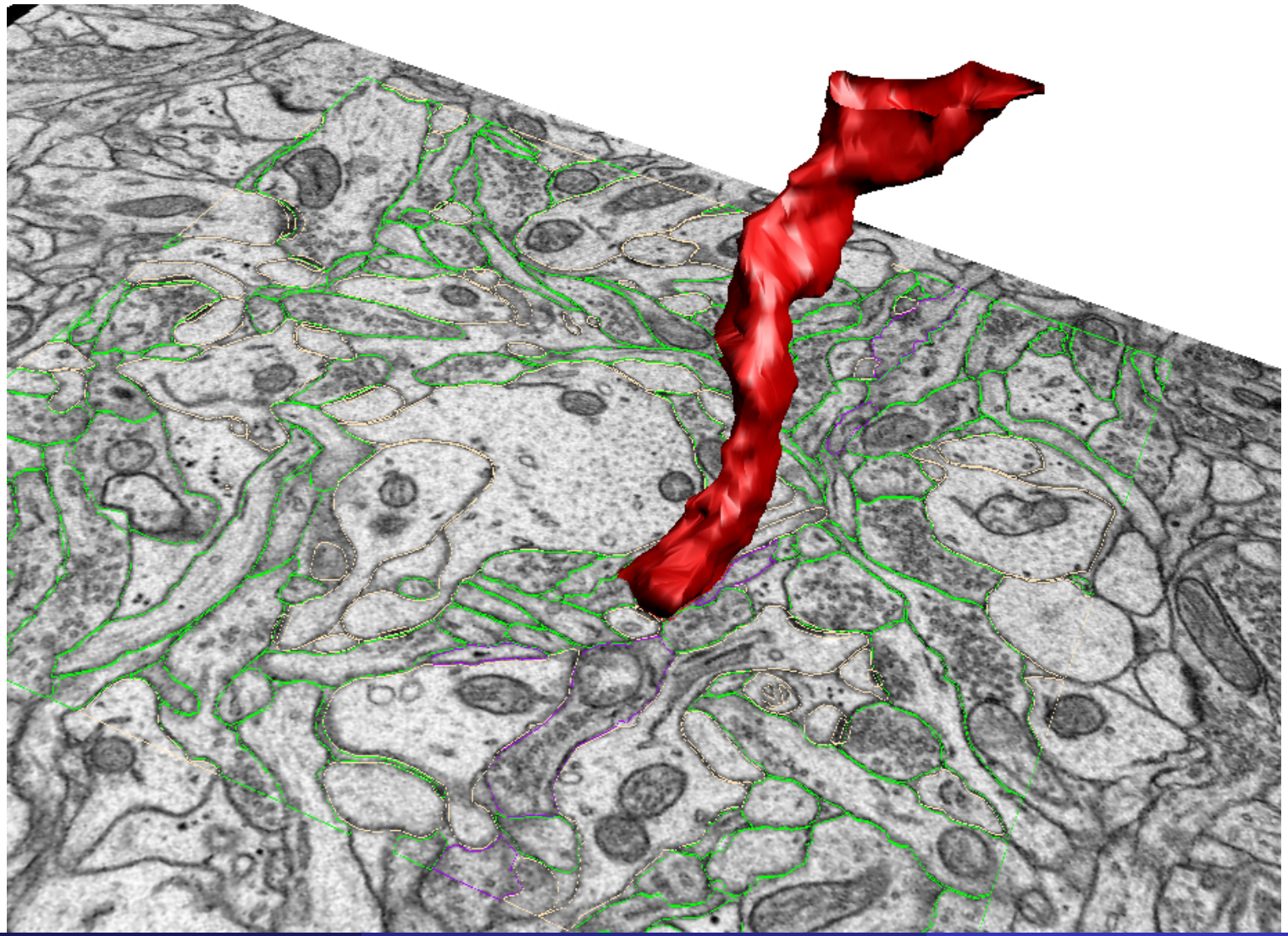
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



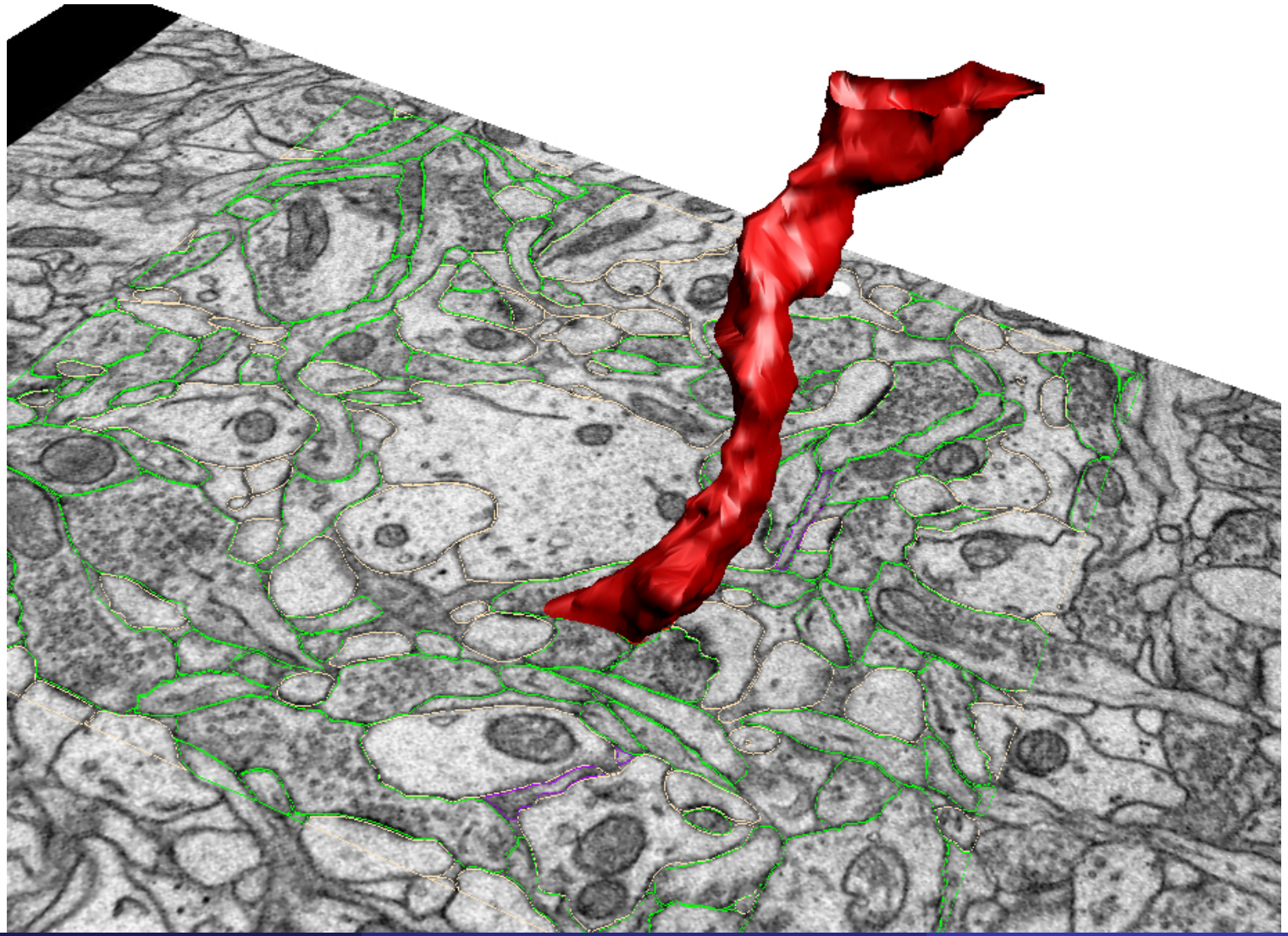
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



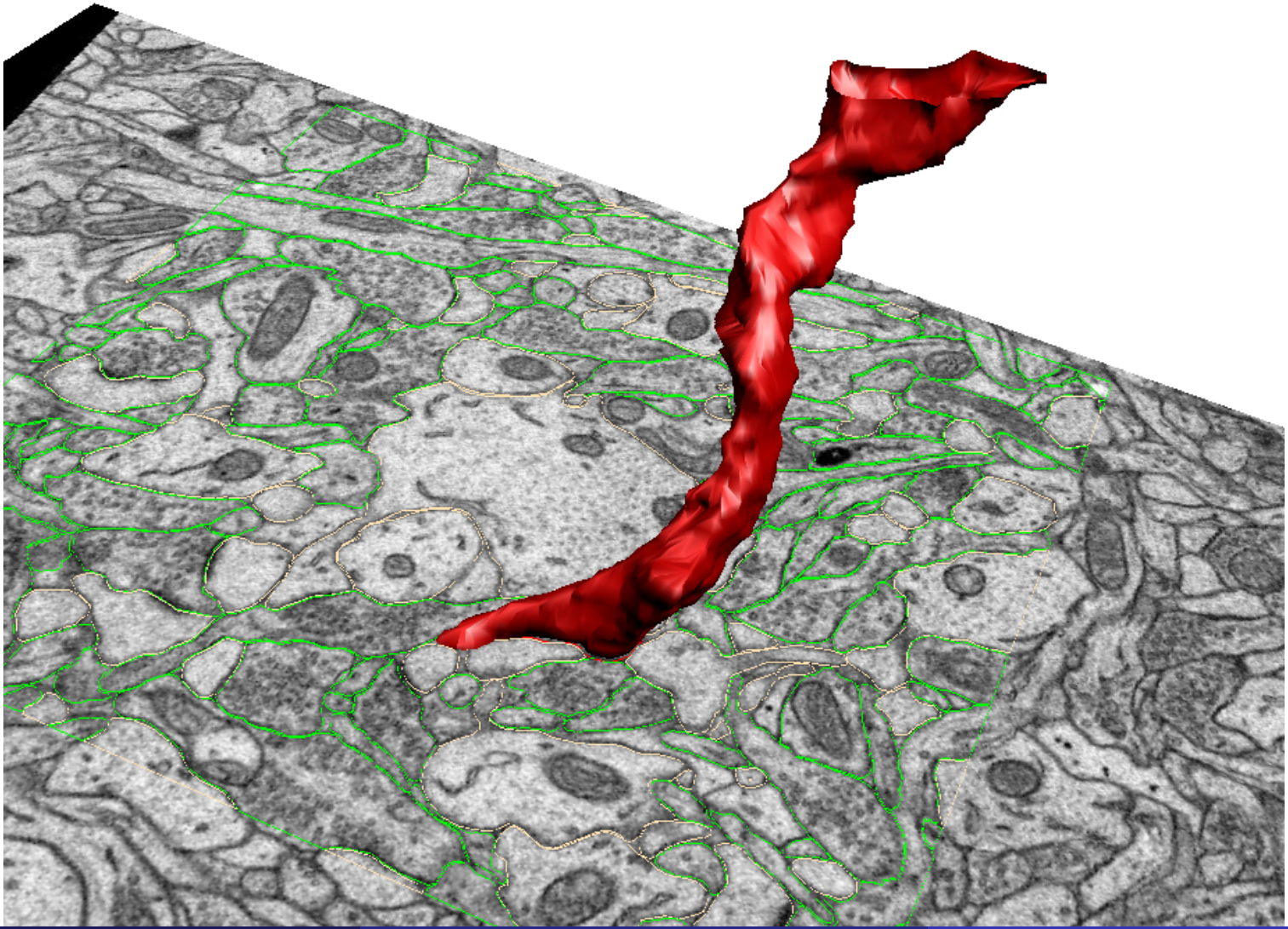
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



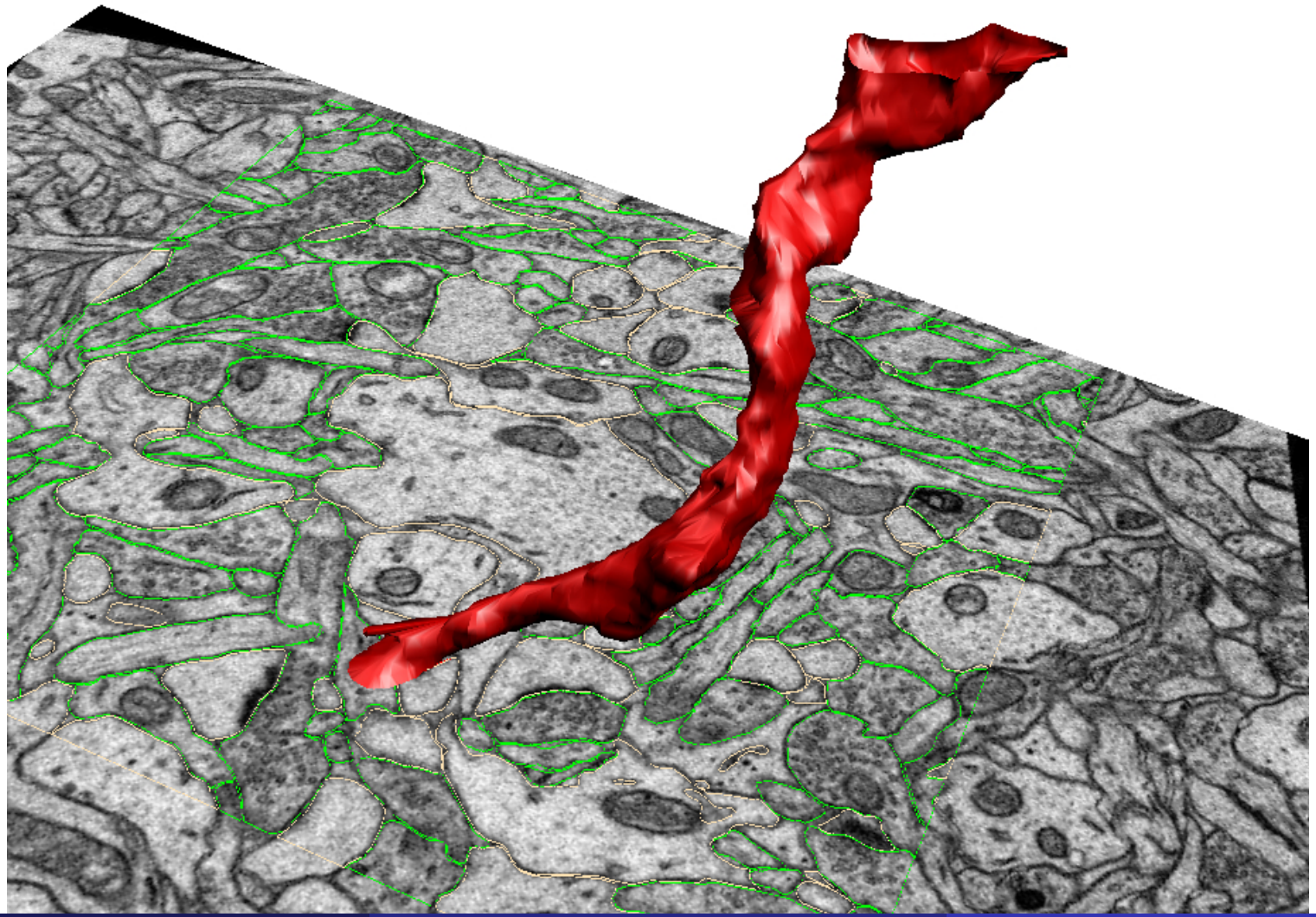
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



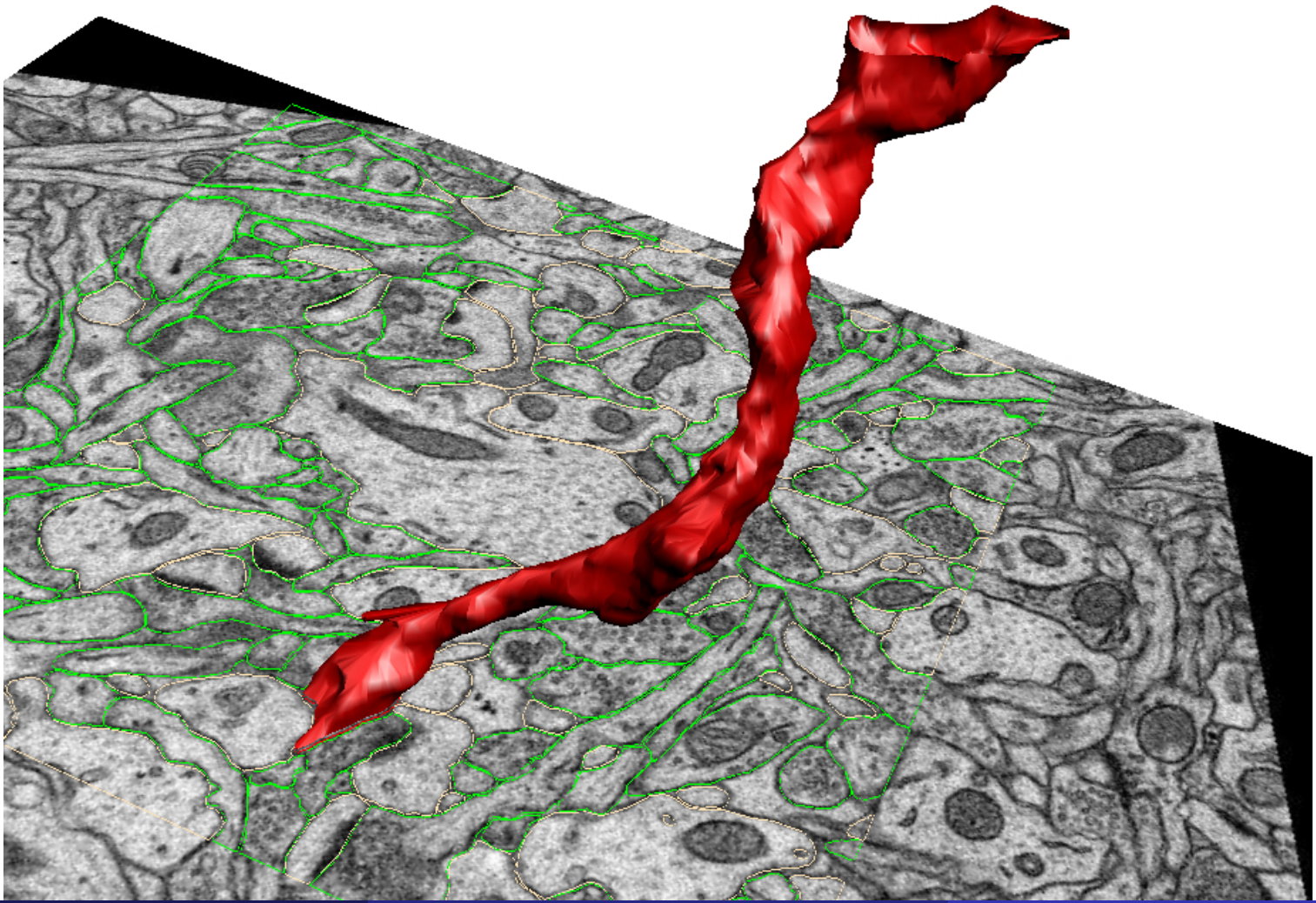
Single component reconstruction

With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.

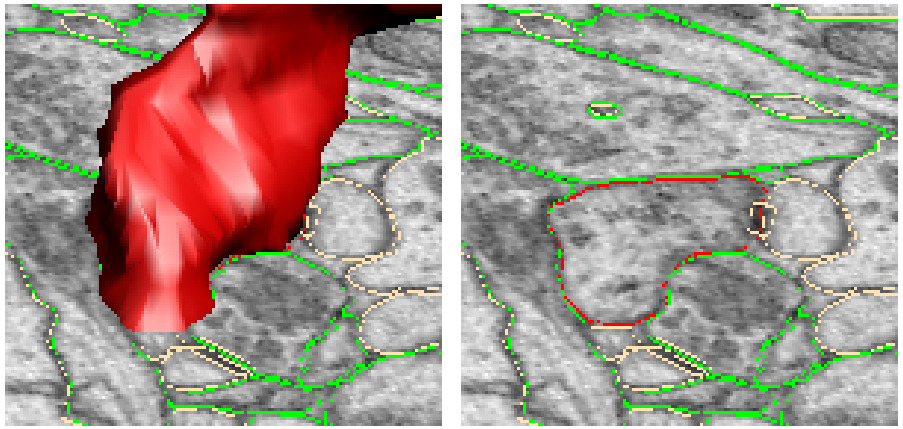


Single component reconstruction

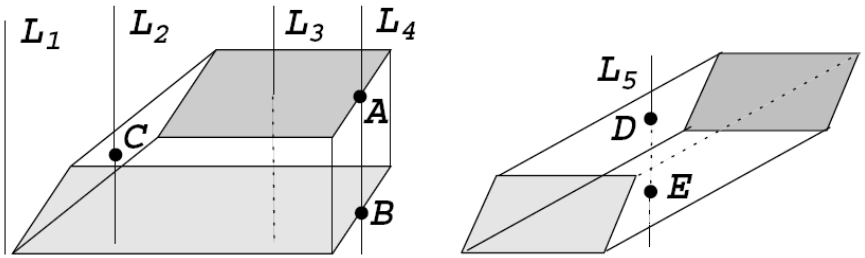
With Bajaj's algorithm [1] we can produce nice single-component reconstructions. The reconstructions can be combined to produce a forest.



Single component reconstruction - properties



(2)



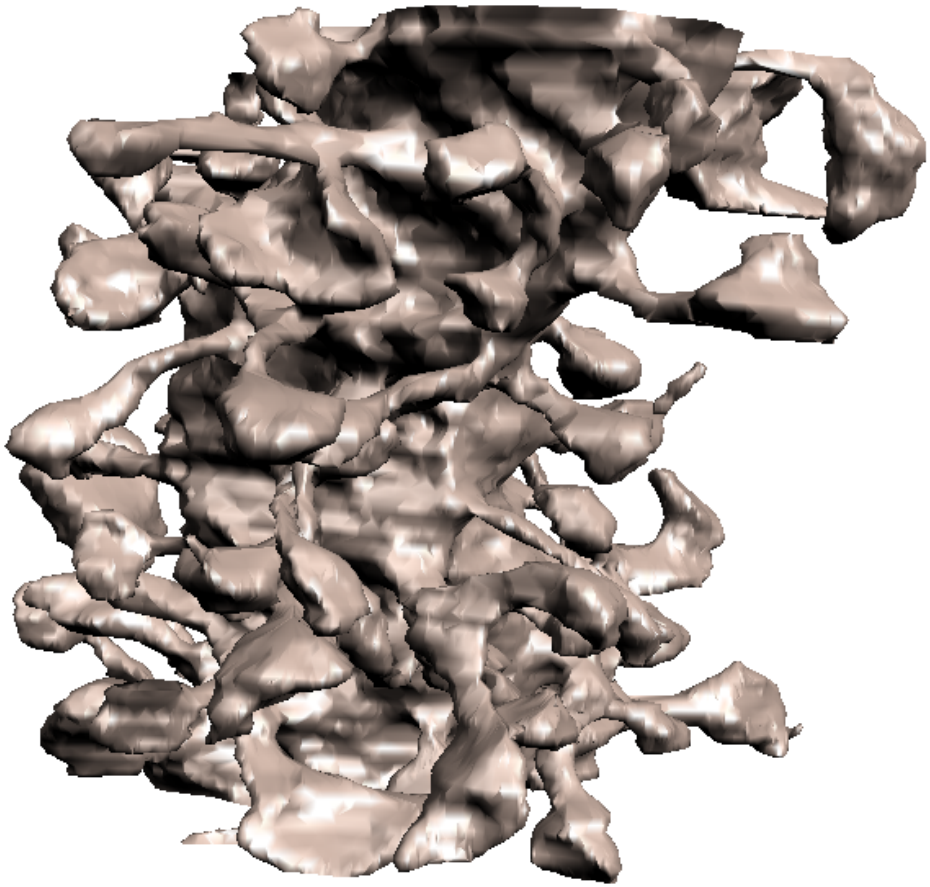
(3)

Properties that we will take advantage of:

1. The reconstructed surface is a piecewise closed surface of polyhedra.
2. Slicing the reconstructed surface on any of the original slices produces exactly the input contours.
3. Any vertical (parallel to the z axis) line segment between two adjacent slices intersects a single component exactly 0 or 1 times, or along exactly one line segment.

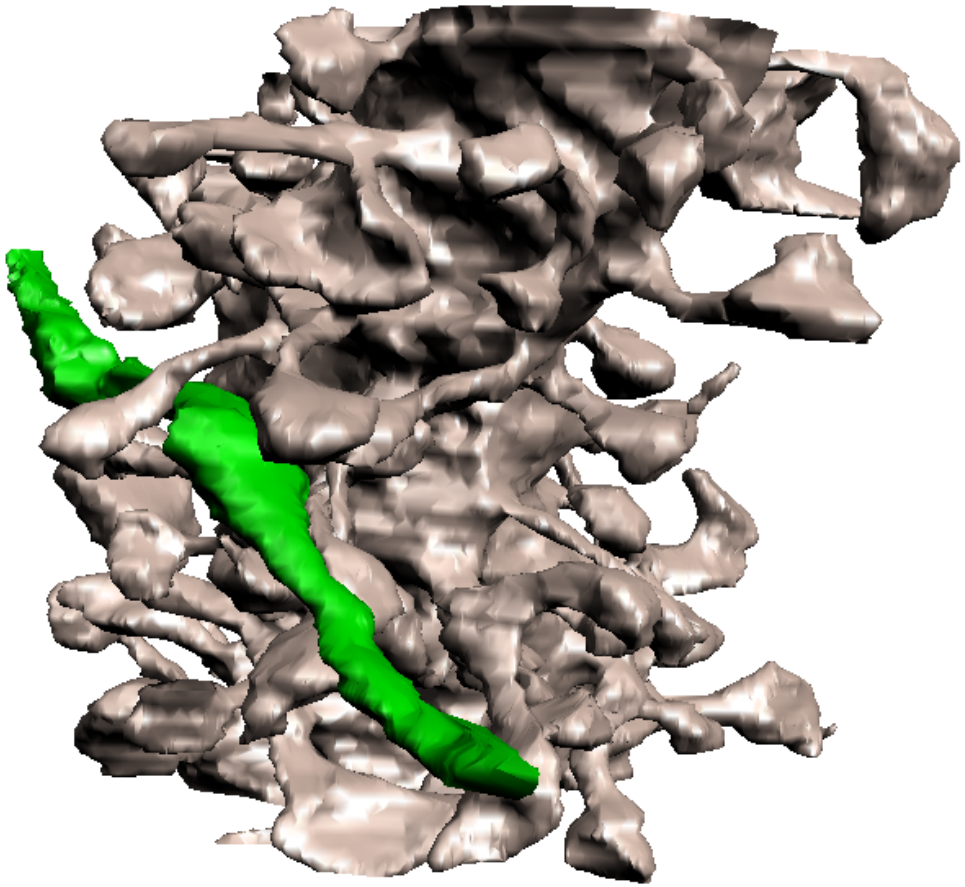
Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.



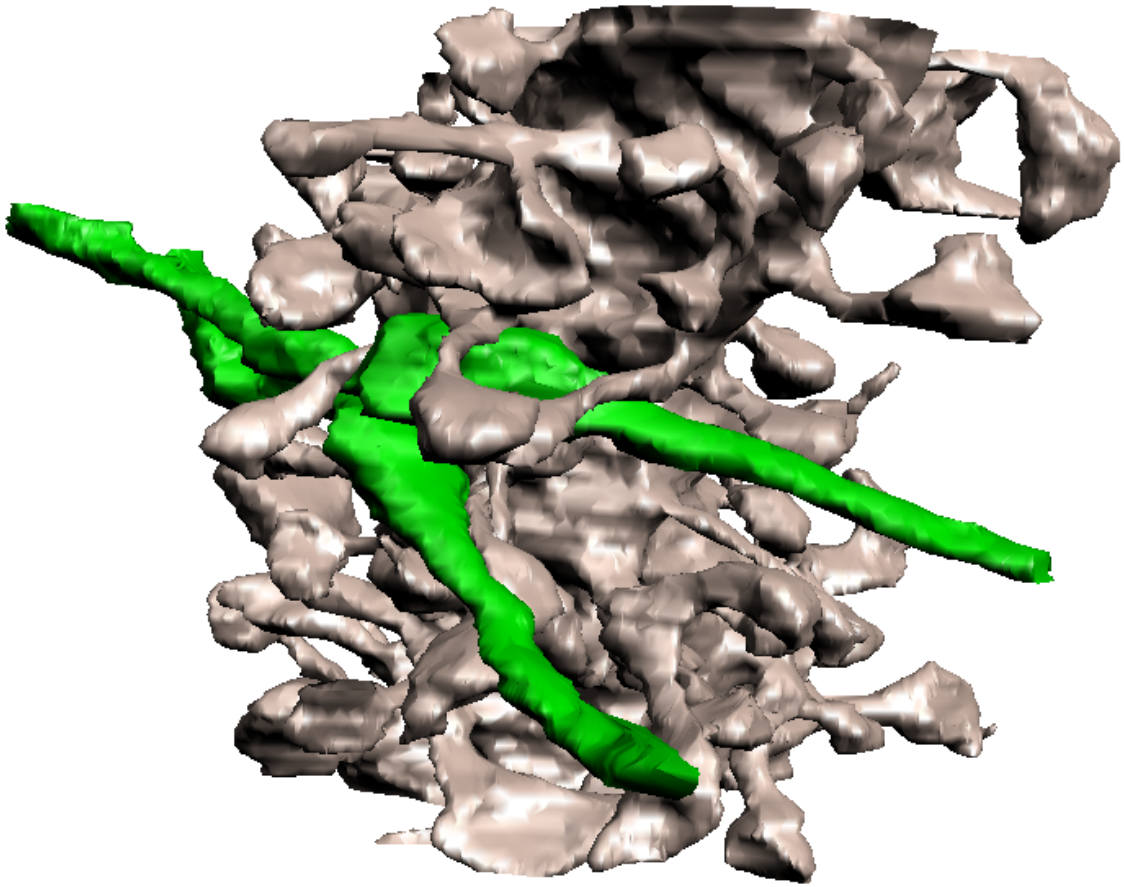
Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.



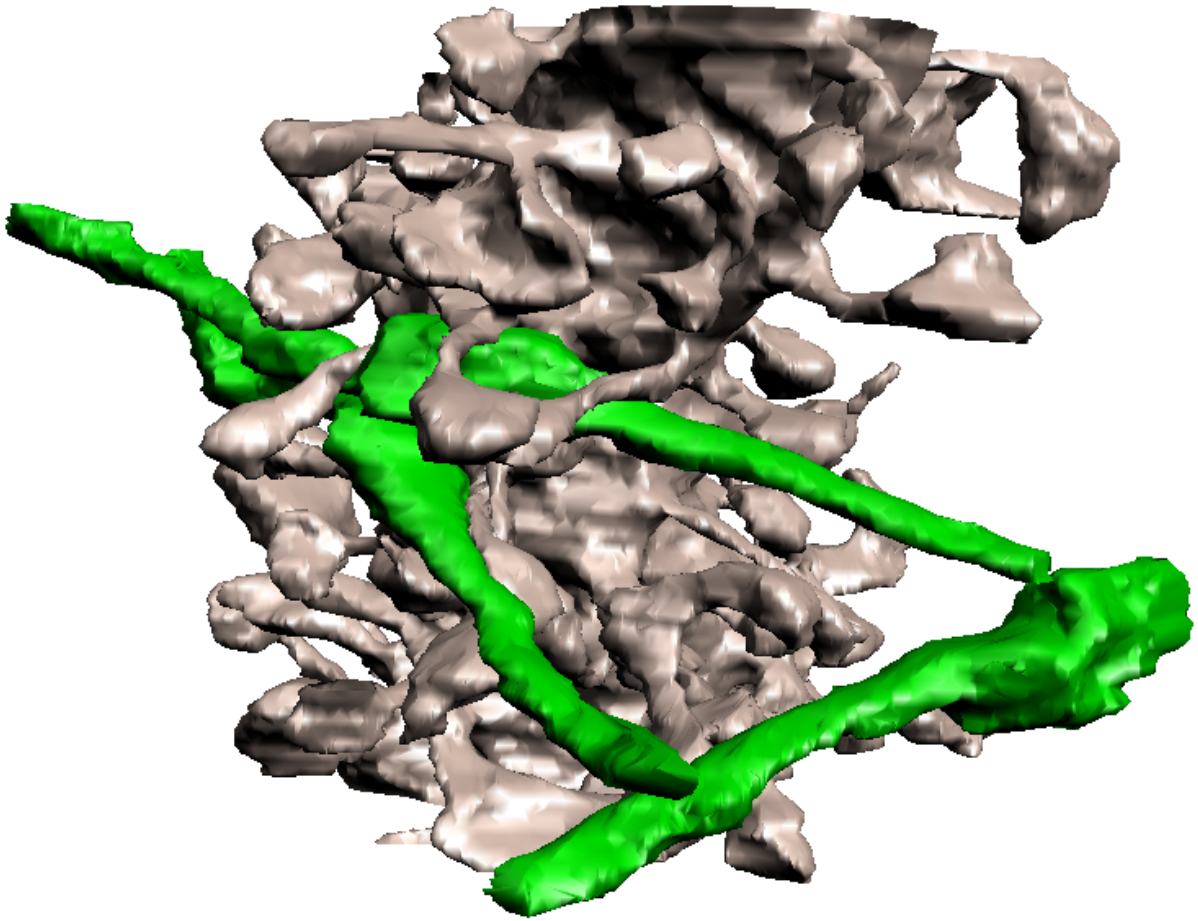
Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.



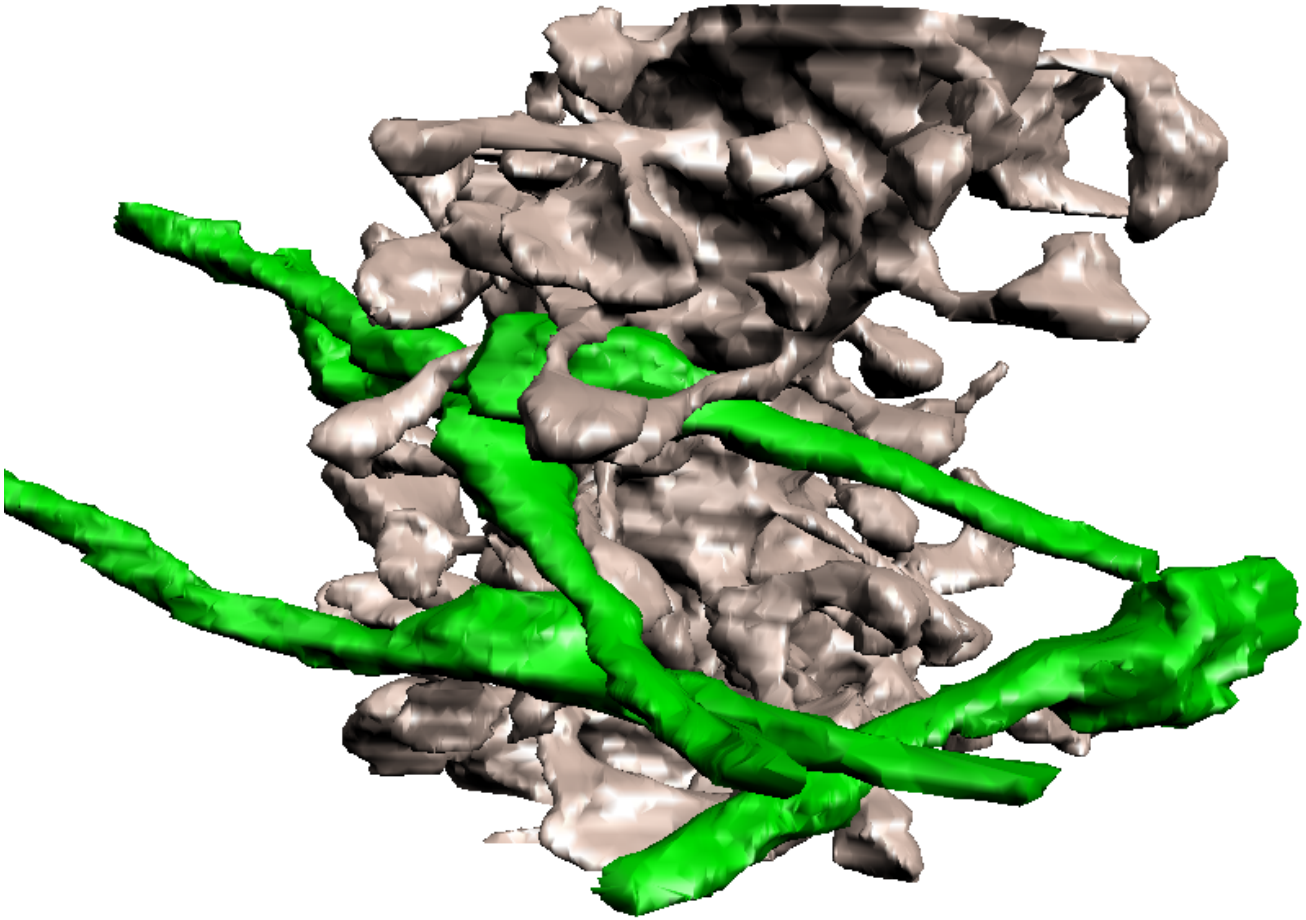
Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.



Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.



Forest reconstruction

To produce a forest of components we reconstruct components one-by-one.

Inter-component intersections

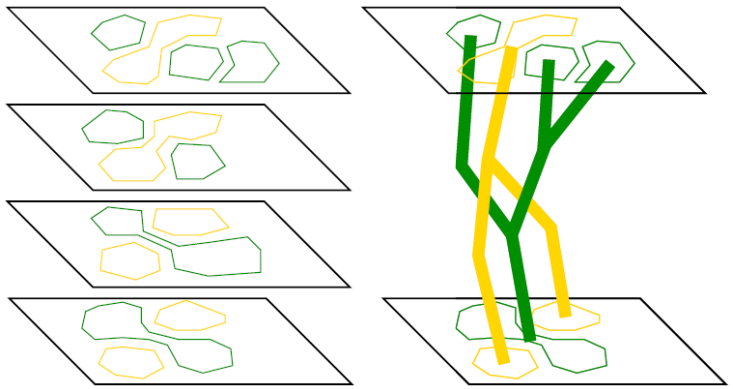
The problem: intersections can occur between components.

Such intersections occur when data is

- highly anisotropic
- tightly packed
- tortuous

Intersection removal

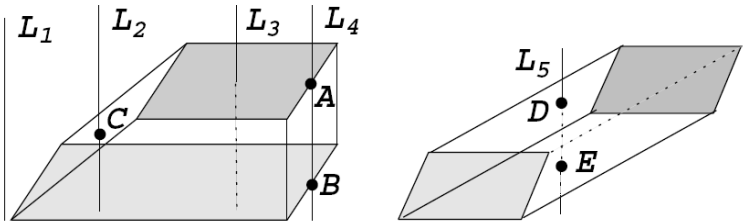
- Our algorithm removes intersections by adjusting only z-values of existing tiles (triangles)
  1. Can remove intersections without iterating through mid-slices
  2. Will not cause additional intersections
  3. Branching treated just like any other intersection
- Intersections often occur at branching points, but may occur where there is no branching. Our algorithm handles both intersections the same way.

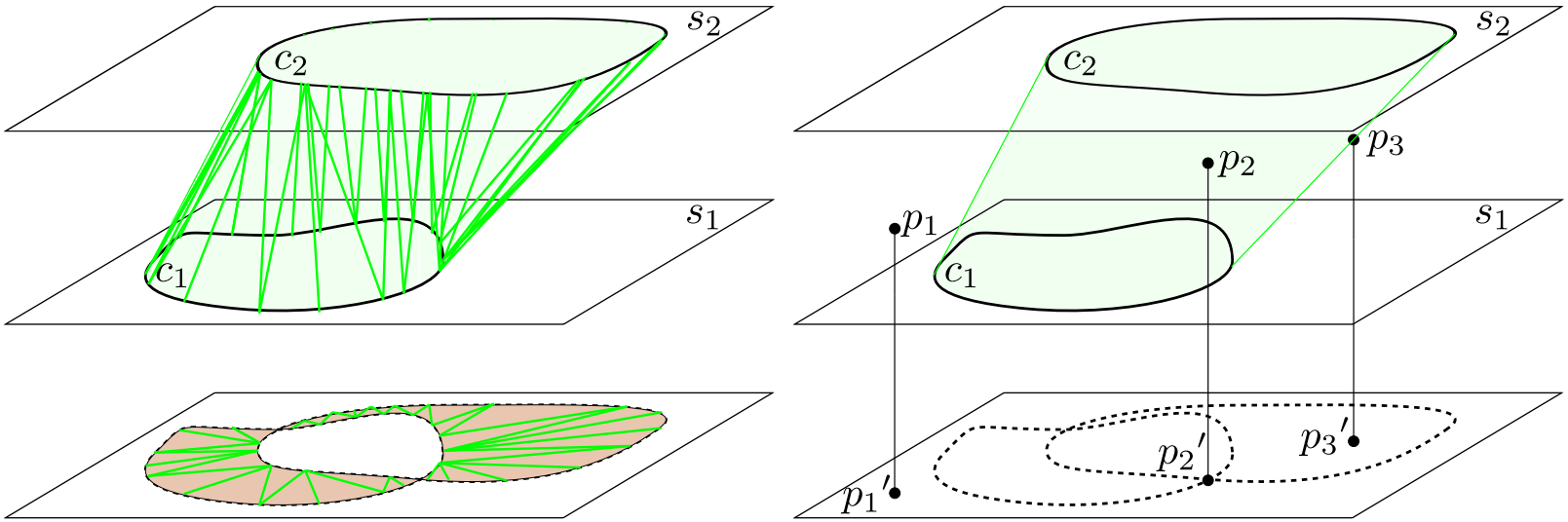


Penumbral contours

Fact

All intersections occur in *penumbral regions*. A point's *penumbral contour* is the contour whose projection contains the projected point.



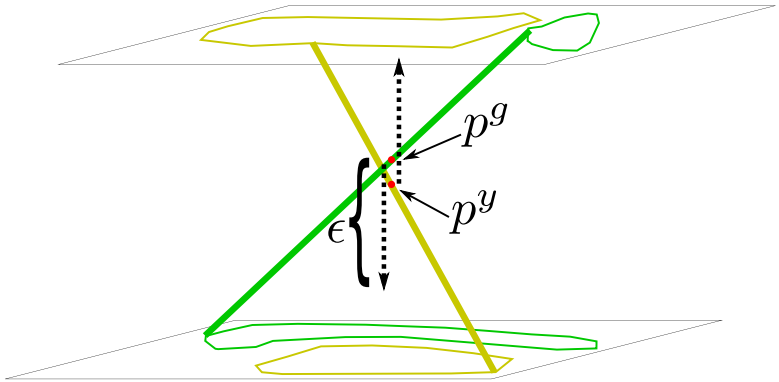


Conflict points

A *conflict point* is a point of intersection. Somewhat more formally:

Definition

Point p^g is called a *conflict point* if there is some point p^y such that the projections are equal ($p^{y\prime} = p^{g\prime}$) and p^y is closer to p^g's penumbral contour than p^g is.



Fact

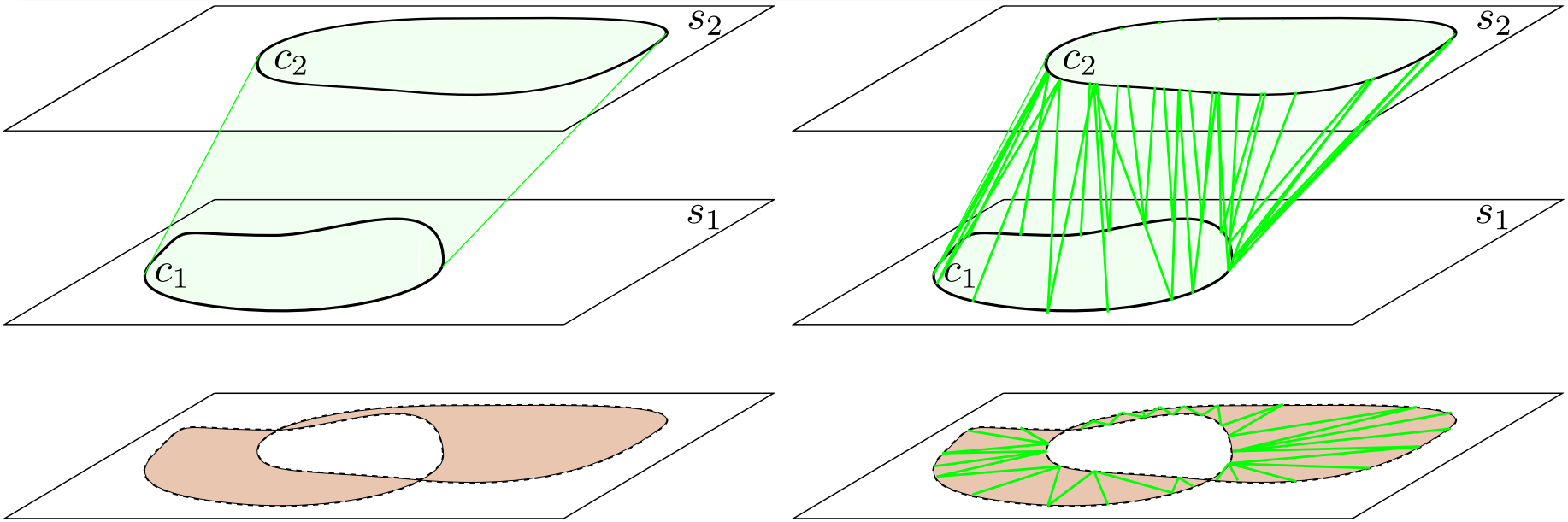
Two components C^g and C^y intersect if and only if there is at least one conflict point on the surface of either component.

Moving conflict points

Our algorithm will remove intersections without causing other intersections.

Theorem

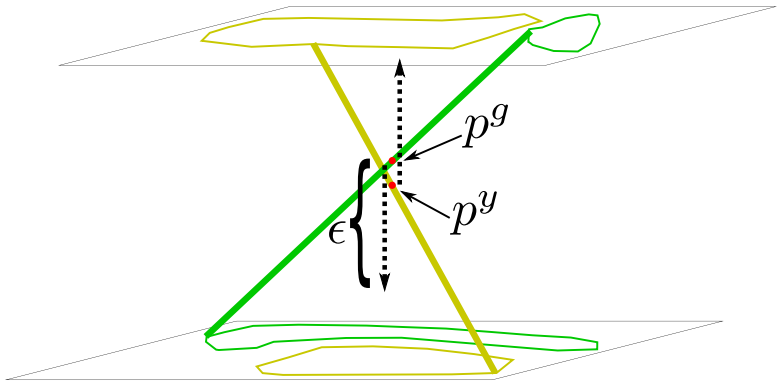
Moving any conflict point p^g in the direction of its penumbral contour will not generate any additional conflict points among any pair of components.

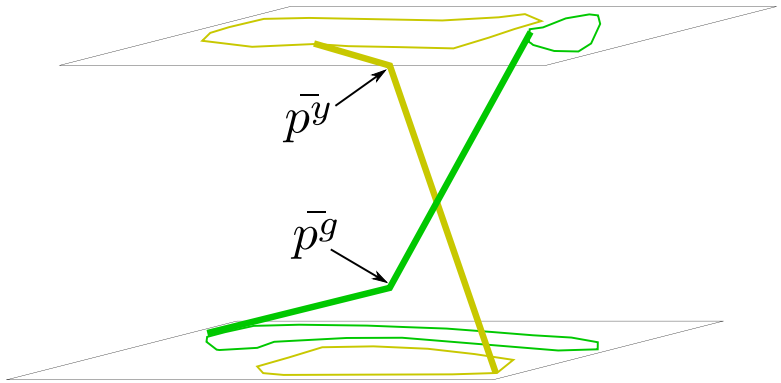


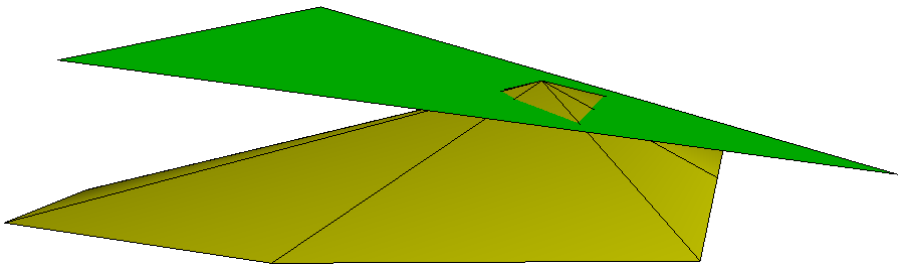
Idea of proof: as a point moves toward its penumbral contour it won't enter any component because only two components can intersect in a given penumbra.

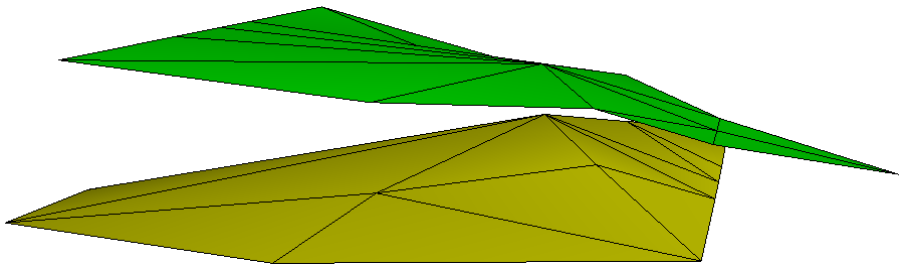
Removing conflicts

We can resolve conflict points by moving them in the directions of their penumbral contours without worrying about causing additional intersections. Once all conflict points are resolved, all intersections are removed.



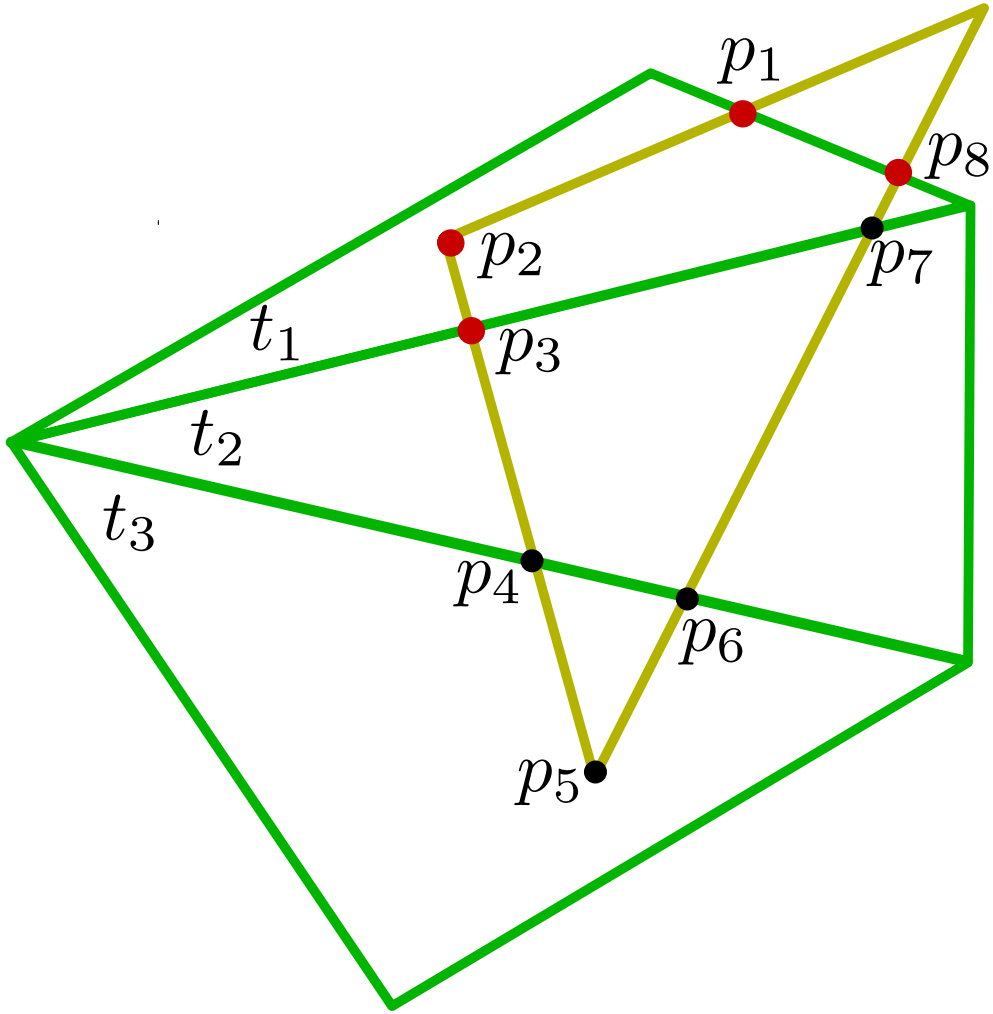






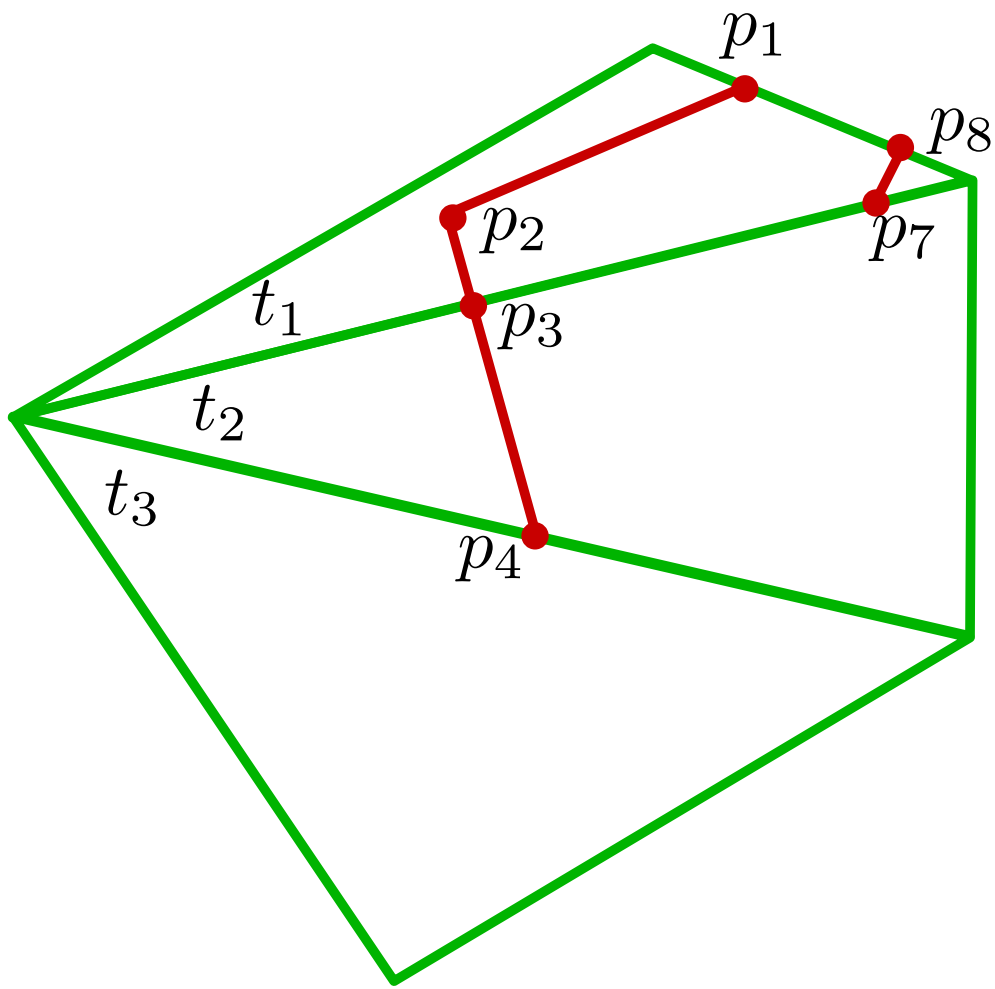
Conflict removal algorithm

1. Detect conflict points.
2. Trace paths between conflict points along edges of yellow tile. We call these *cut paths*.
3. Use original tiles and cut paths to induce new polygons.
4. Triangulate polygons and move conflict points along z-axis.



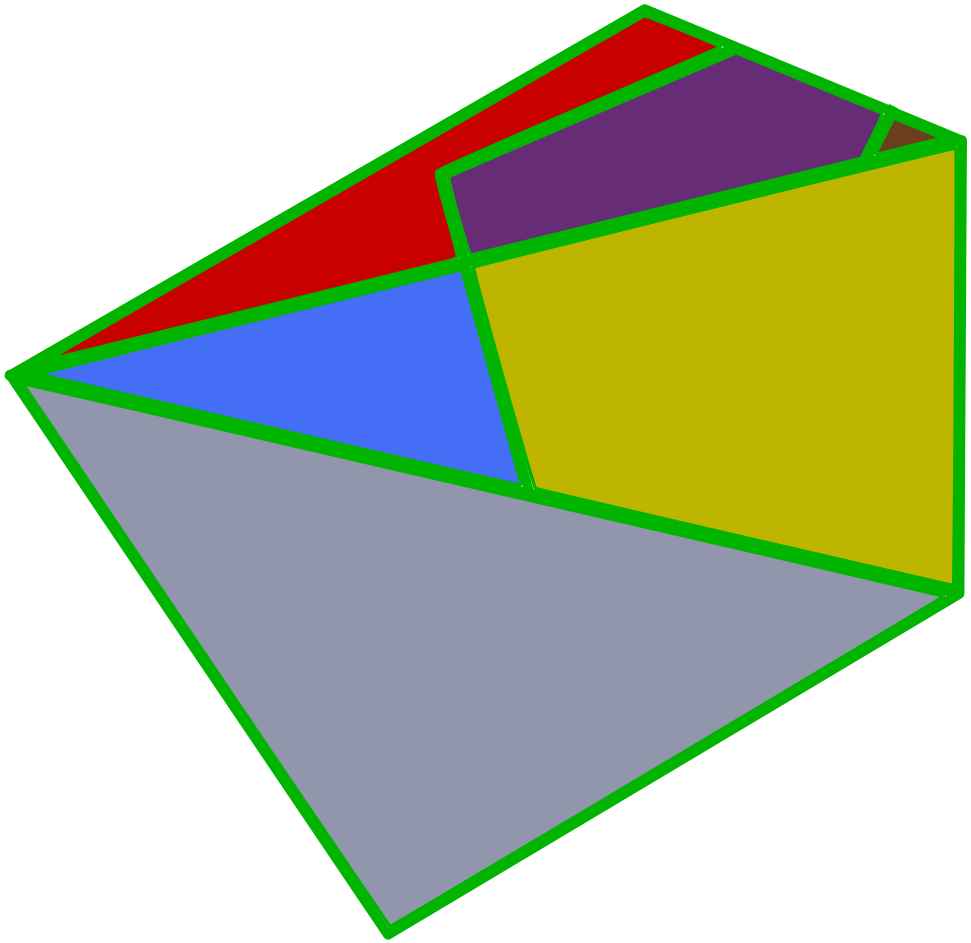
Conflict removal algorithm

1. Detect conflict points.
2. Trace paths between conflict points along edges of yellow tile. We call these *cut paths*.
3. Use original tiles and cut paths to induce new polygons.
4. Triangulate polygons and move conflict points along z-axis.



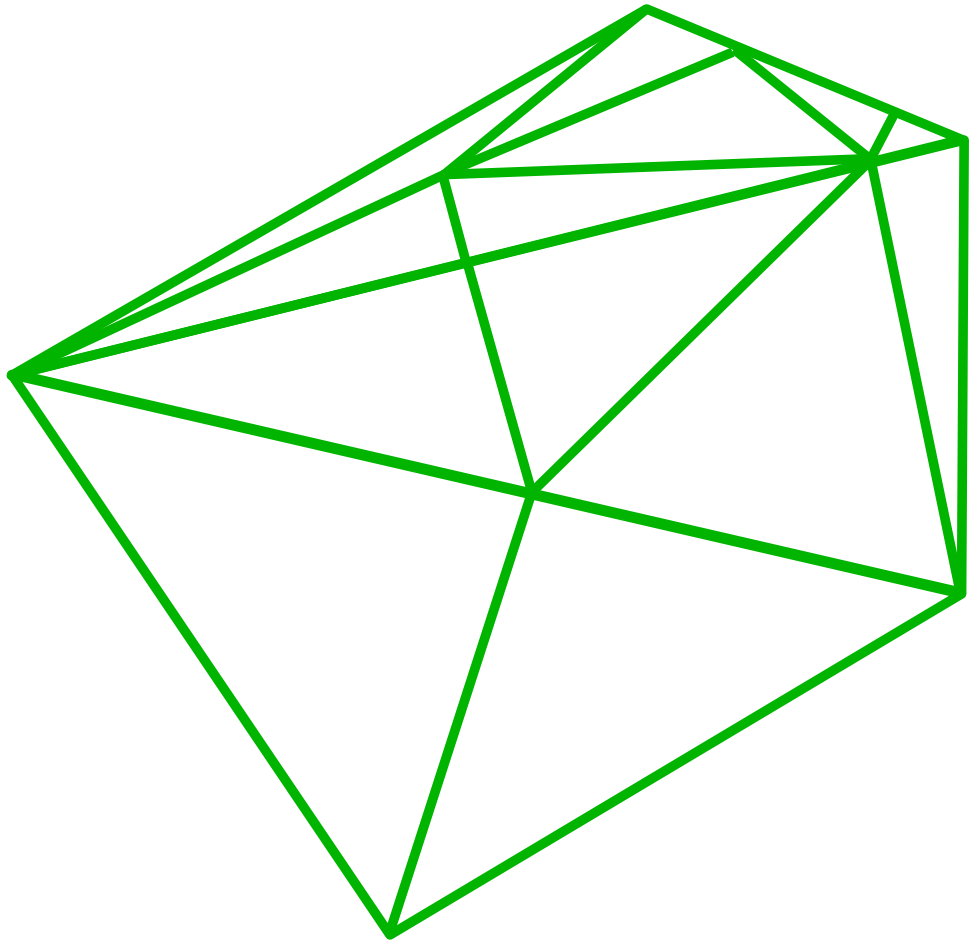
Conflict removal algorithm

1. Detect conflict points.
2. Trace paths between conflict points along edges of yellow tile. We call these *cut paths*.
3. Use original tiles and cut paths to induce new polygons.
4. Triangulate polygons and move conflict points along z-axis.



Conflict removal algorithm

1. Detect conflict points.
2. Trace paths between conflict points along edges of yellow tile. We call these *cut paths*.
3. Use original tiles and cut paths to induce new polygons.
4. Triangulate polygons and move conflict points along z-axis.

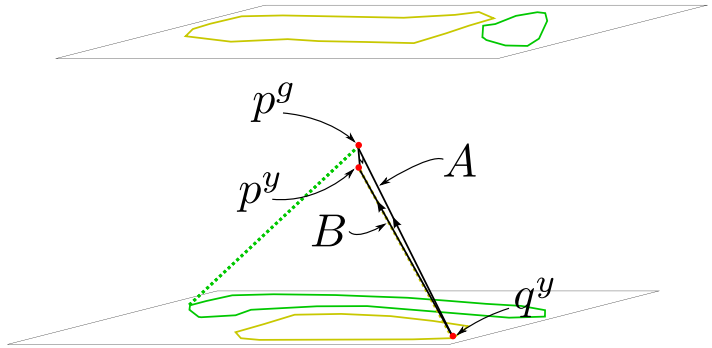


Separating by a given delta

$d = \frac{|\overline{\mathbf{A}} \times \overline{\mathbf{B}}|}{|\overline{\mathbf{B}}|}$

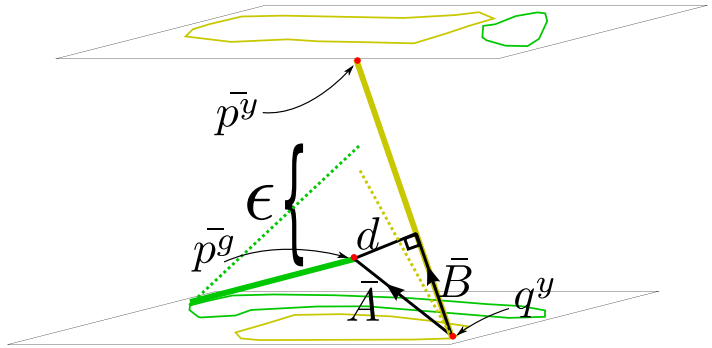
Substituting for $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$:

$$\begin{aligned} d^2 = & \quad ((A_y(B_z - \epsilon) - (A_z + \epsilon)B_y)^2 \\ & + \quad ((A_z + \epsilon)B_x - A_x(B_z - \epsilon))^2 \\ & + \quad (A_x B_y - A_y B_x)^2)/(B_x^2 + B_y^2 + (B_z + \epsilon)^2) \end{aligned}$$

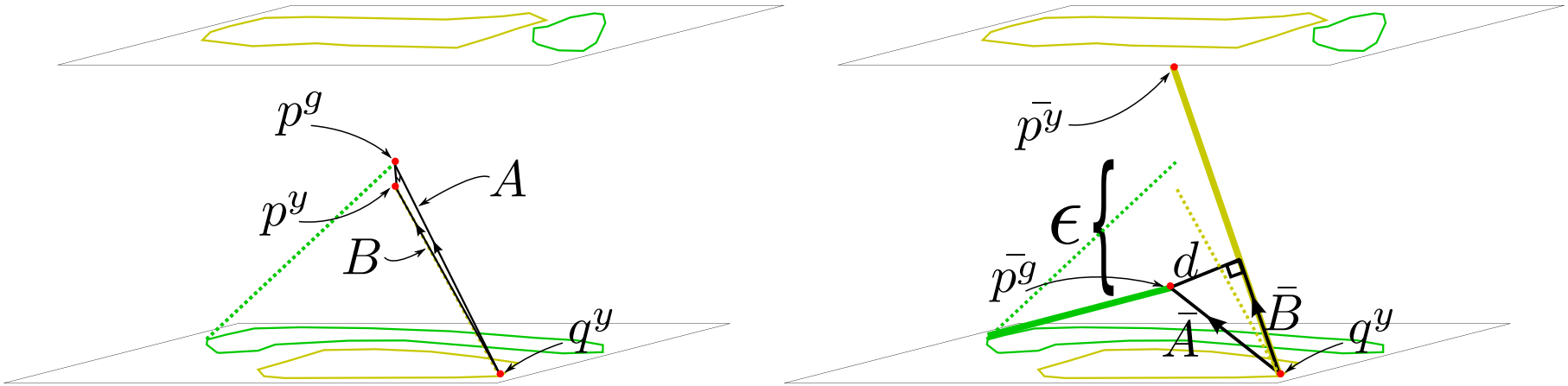


After collecting ϵ:

$$\begin{aligned} 0 = & \quad \epsilon^2((A_y + B_y)^2 + (A_x + B_x)^2 - d^2) \\ & + \quad \epsilon(2)((A_x + B_x)(A_z B_x - A_x B_z) \\ & \quad -(A_y + B_y)(A_y B_z - A_z B_y) - d^2 A_z) \\ & + \quad (A_y B_z - A_z B_y)^2 + (A_z B_x - A_x B_z)^2 \\ & \quad +(A_x B_y - A_y B_x)^2 - d^2(B_x^2 + B_y^2 + B_z^2) \end{aligned}$$



Separating by a given delta



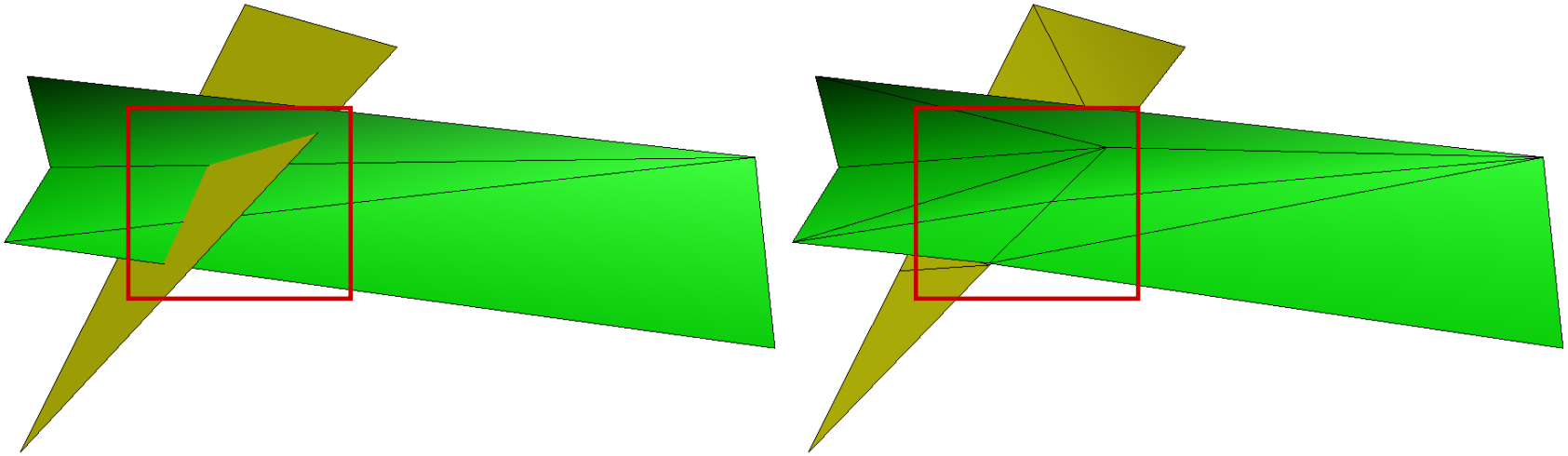
Theorem

$\epsilon < |p^g - \mathscr{Z}(p^g)|$ and $\epsilon < |p^y - \mathscr{Z}(p^y)|$

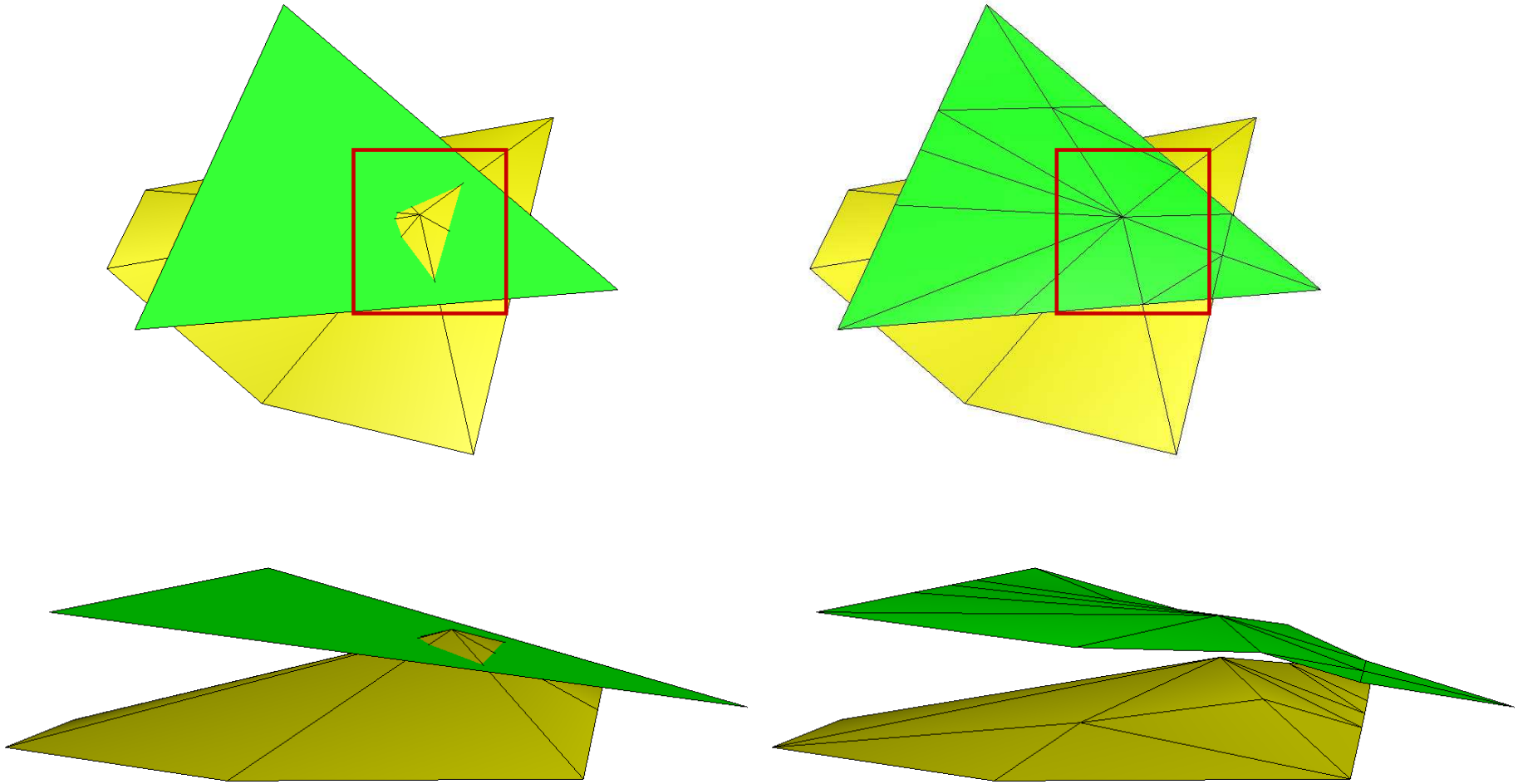
Idea of proof: as points approach original contours, which are separated by d, the chords will be separated by at least d in the limit.

Outline

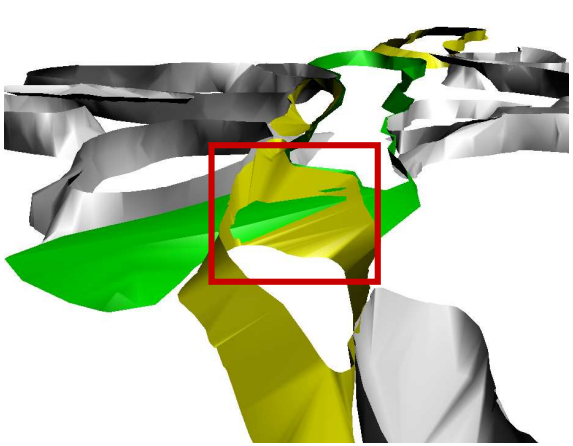
Illustrative examples

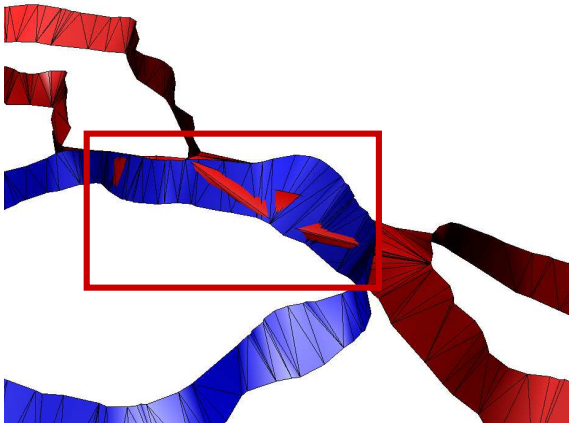


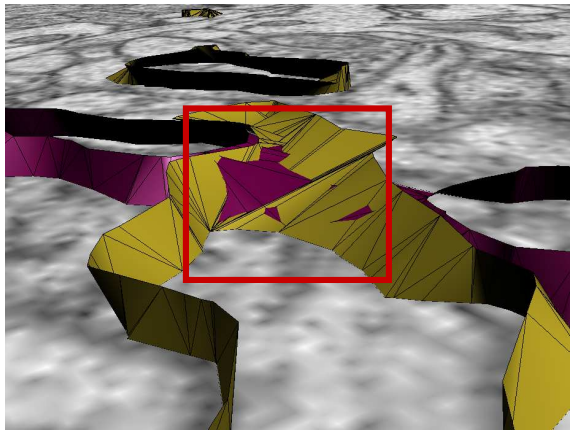
Illustrative examples

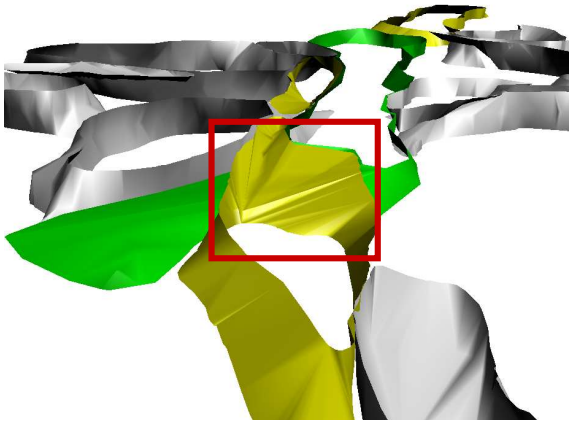


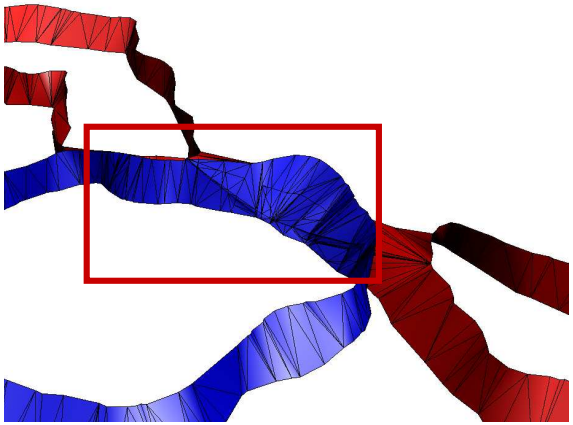
Results

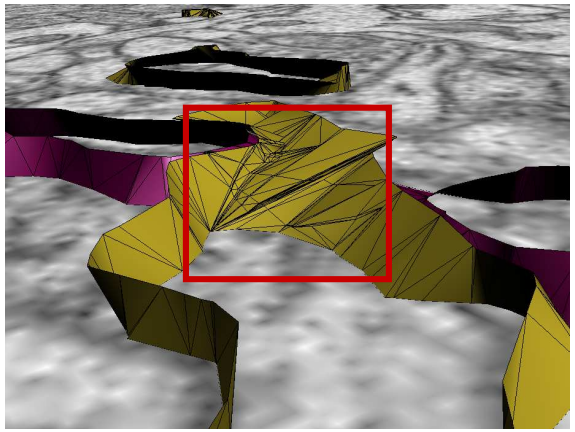




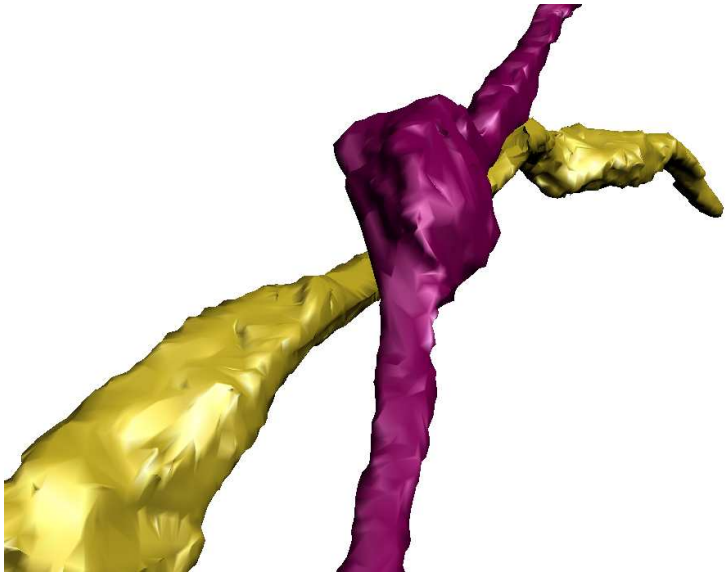


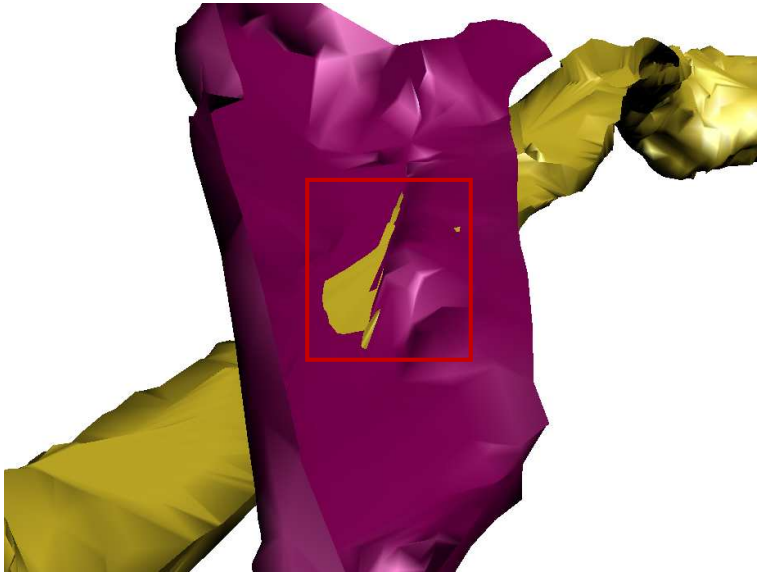


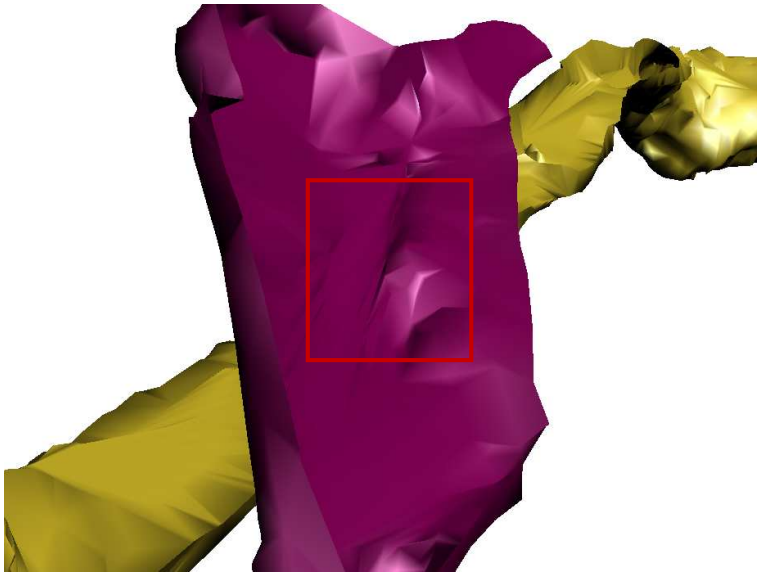


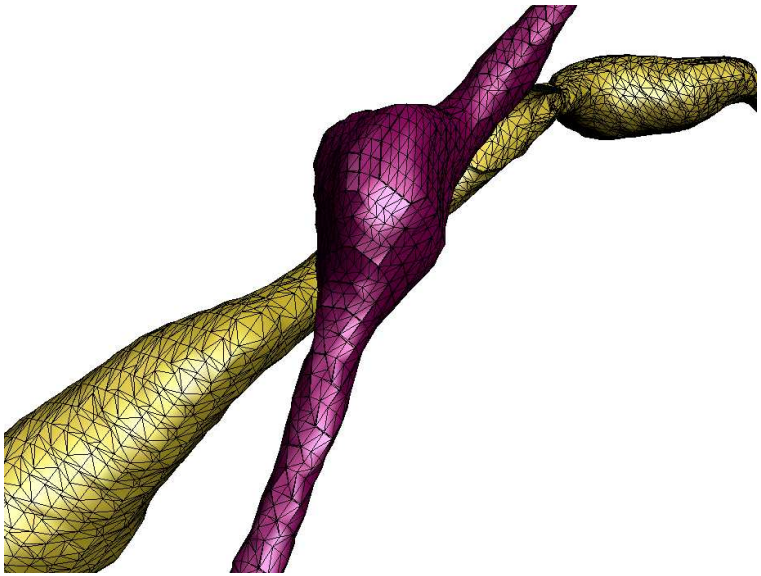


Results

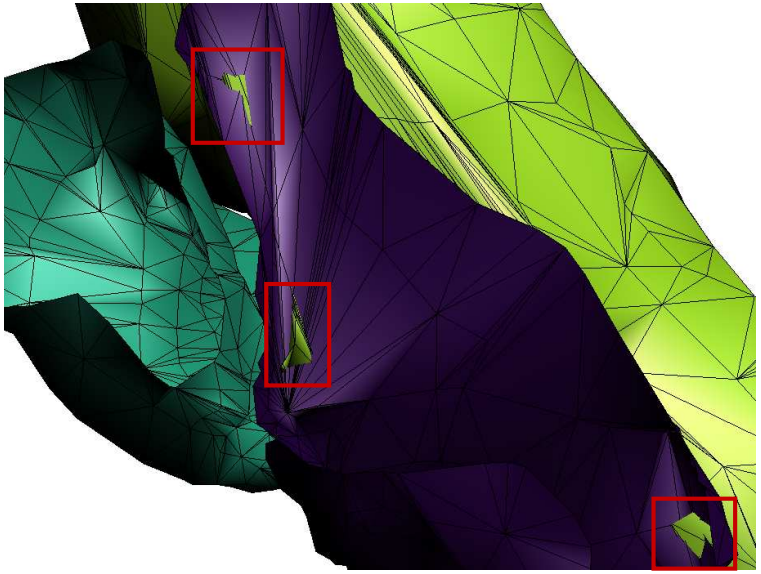


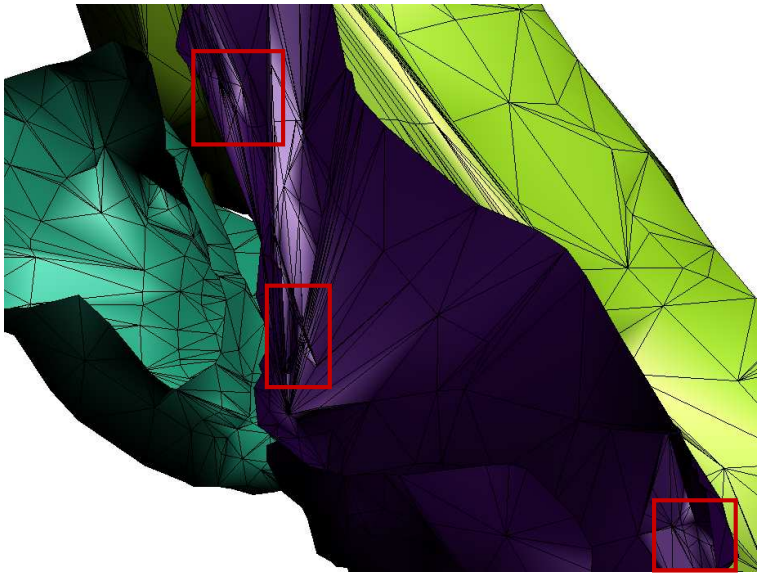


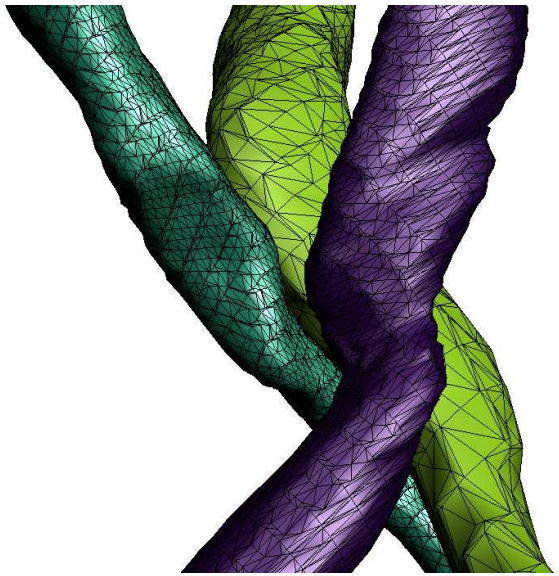




Results

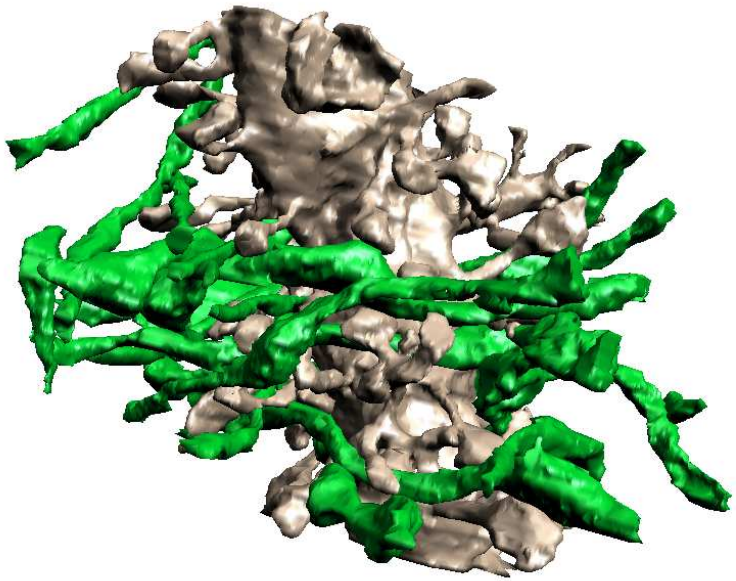






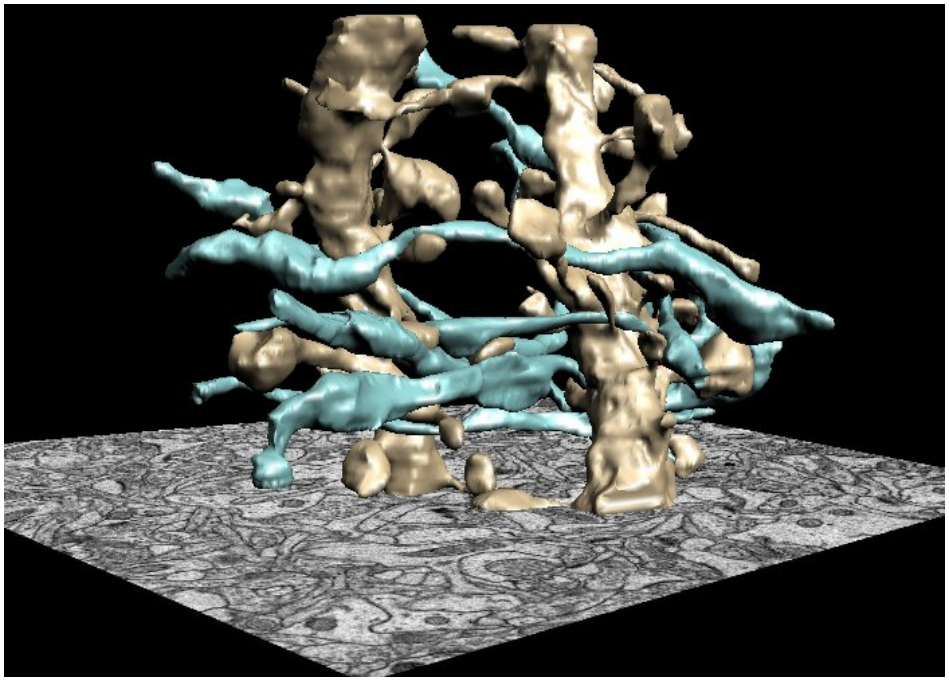
Conclusions and notes

- Algorithm is $O(n^2)$ where n is the number of tiles.
  - Average case is closer to $n \log n$ complexity of sweep line algorithm as large majority of 2D intersections are not conflict points.
- Original contours remain unchanged – only makes changes in interpolated data between slices
- Topologically correct and water tight
- Generates large number of extra triangles in intersecting regions



Acknowledgements

- All EM imagery was provided by Dr. Kristen Harris at the Center for Learning and Memory and Institute for Neuroscience at UT Austin.
- This research was supported in part by NIH contracts R01-EB00487, R01-GM074258, and a grant from the UT-Portugal colab project.



References

C. L. Bajaj, E. J. Coyle, and K. Lin.

Arbitrary topology shape reconstruction from planar cross sections.
Graph. Models Image Process., 58(6):524–543, 1996.

C. L. Bajaj and A. Gillette.

Quality meshing of a forest of branching structures.
In *Proceedings of the 17th International Meshing Roundtable*, pages 433–449. Springer-Verlag, October 2008.

G. Barequet and M. Sharir.

Piecewise-linear interpolation between polygonal slices.
In *Computer Vision and Image Understanding*, pages 93–102, 1994.

G. Barequet and A. Vaxman.

Reconstruction of multi-label domains from partial planar cross-sections.
In *SGP '09: Proceedings of the Symposium on Geometry Processing*, pages 1327–1337, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

J.D. Boissonnat and P. Memari.

Shape reconstruction from unorganized cross-sections.
In *Proceedings of the fifth Eurographics symposium on Geometry processing*, page 98. Eurographics Association, 2007.

H. Fuchs, Z. M. Kedem, and S. P. Uselton.

Optimal surface reconstruction from planar contours.
Commun. ACM, 20(10):693–702, 1977.

L. Liu, C. Bajaj, L. O. Deasy, D. A. Low, and T. Ju.

Surface reconstruction from non-parallel curve networks.
Computer Graphics Forum, 27(2):155–163, 2008.