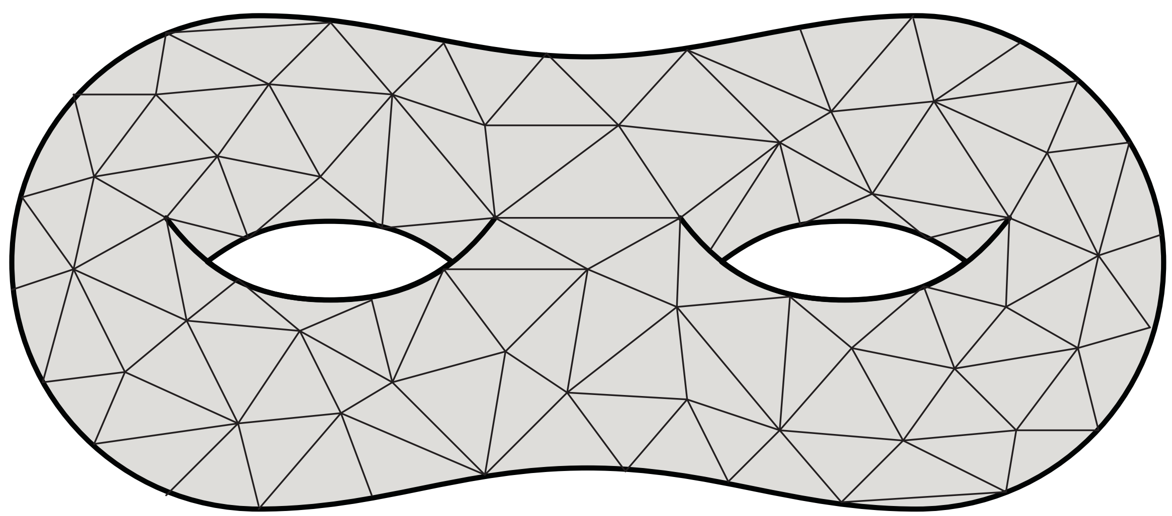
Reconstructing surfaces from point scans

TDA, fall 2025

Combinatorial surfaces

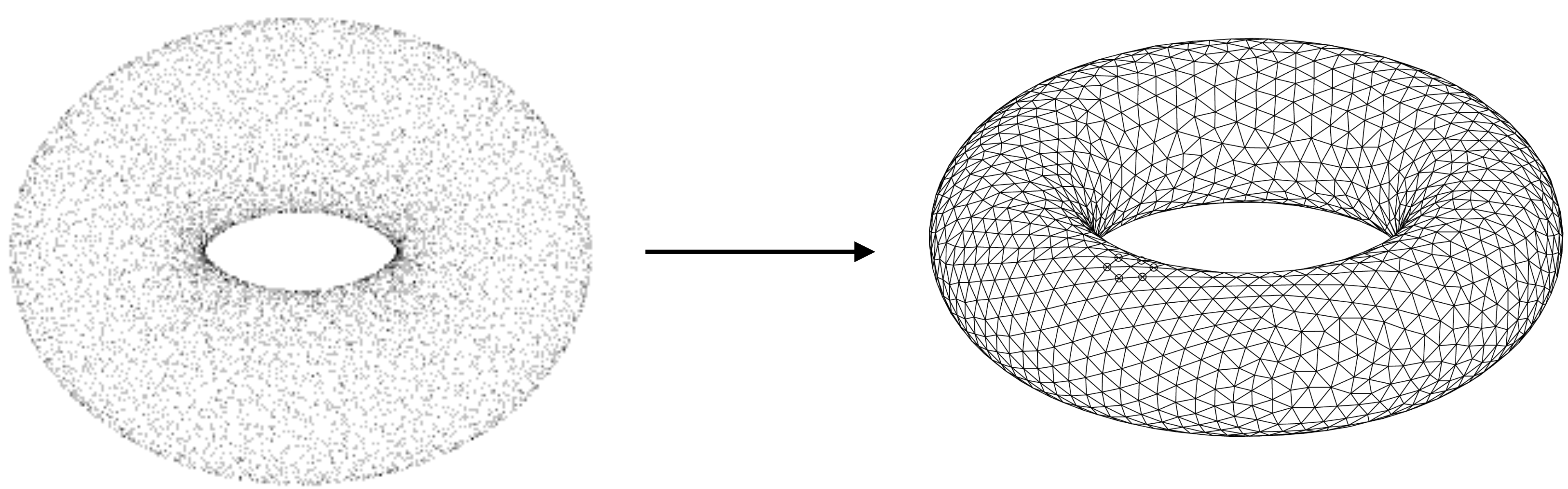
- Many times this semester, I've given you the same picture of a surface:



- But how do we get here? In graphics and geometry processing, usually we start from a point scan or some other method of scanning an object.

Representing shapes

- A fundamental problem: given a set of points scanned from some input, reconstruct the underlying shape they represent



Images courtesy of Wikipedia

Reconstructing shape

- However, sometimes it isn't so clear what shape we want:

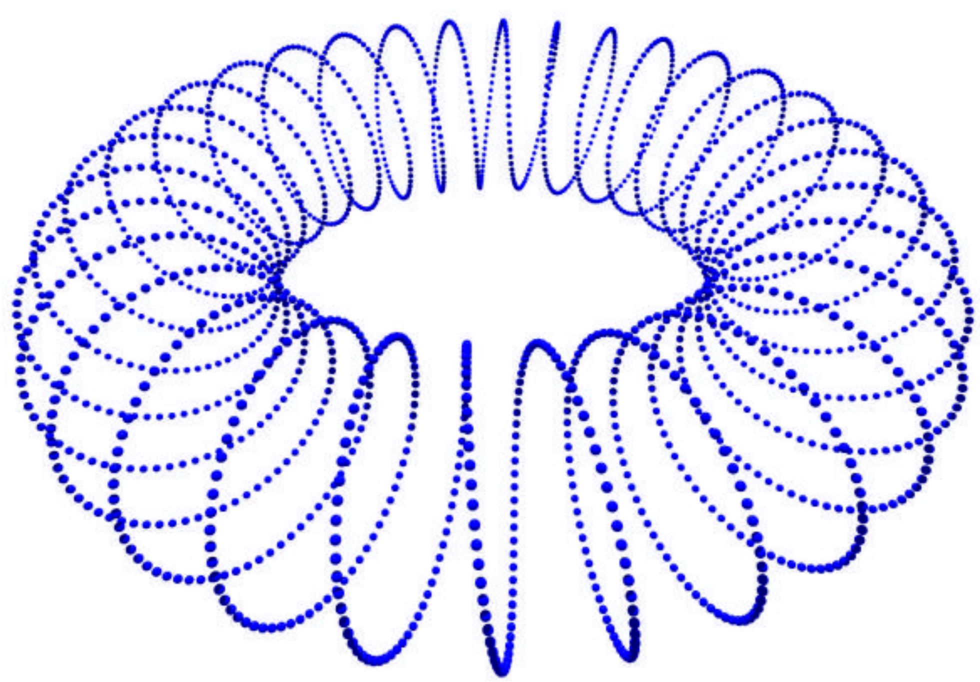


Image courtesy of the SIAM Journal of Applied Algebra and Geometry

Algorithms for shape reconstruction

- Goal today: Survey some classical shape reconstruction algorithms

- Note that this is also an active area of research, and methods vary widely

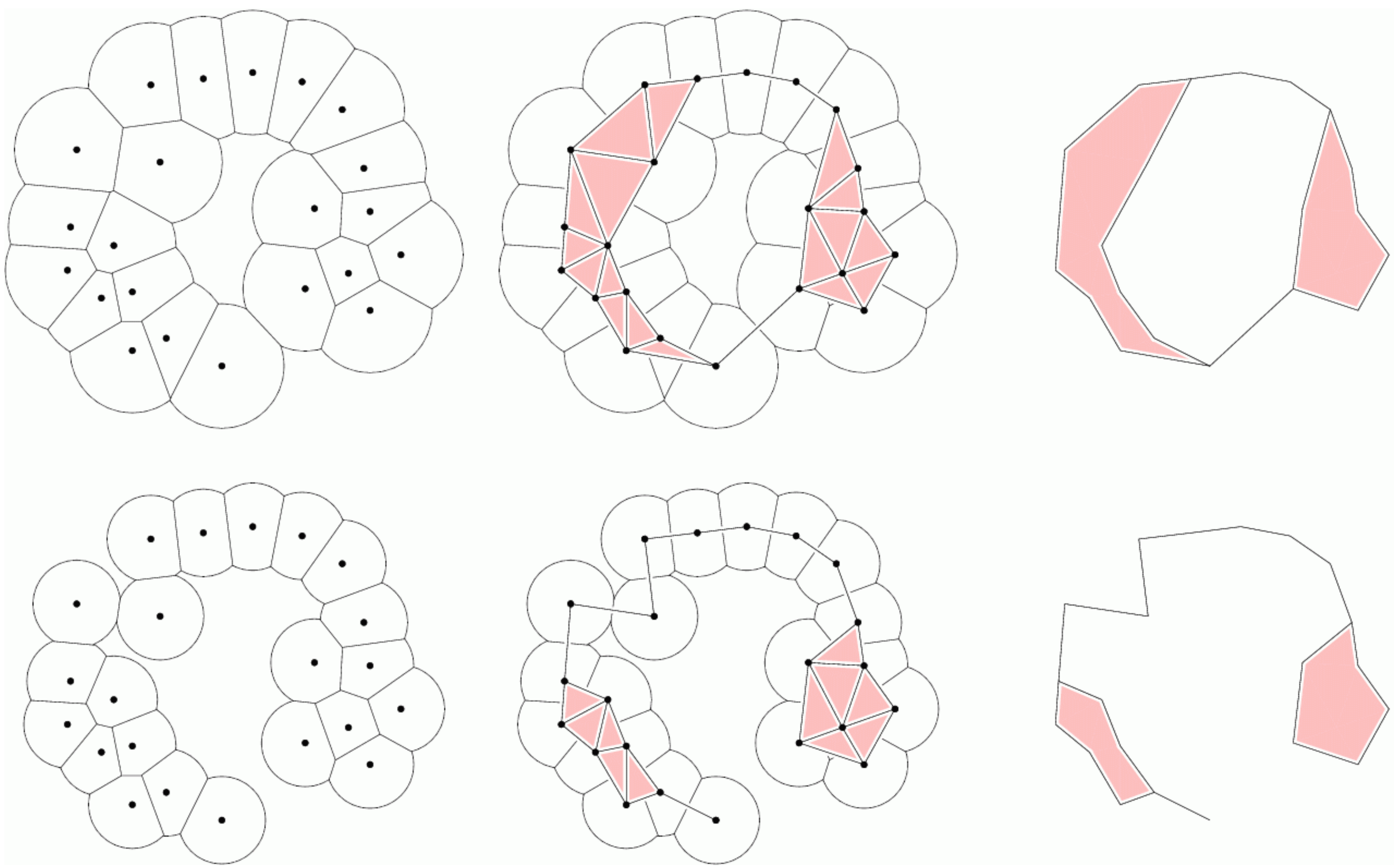
- I'll focus on computational geometry and graphics algorithms, many of which build on the complexes we discussed early in the semester

Goals for any method

- Output a triangulation which is:
  - Homeomorphic to original shape
  - Close geometrically to original shape
  - Approximates the normals
- Interestingly, while I'd classify these as topology, the computational geometry winds up being the key thing to consider.

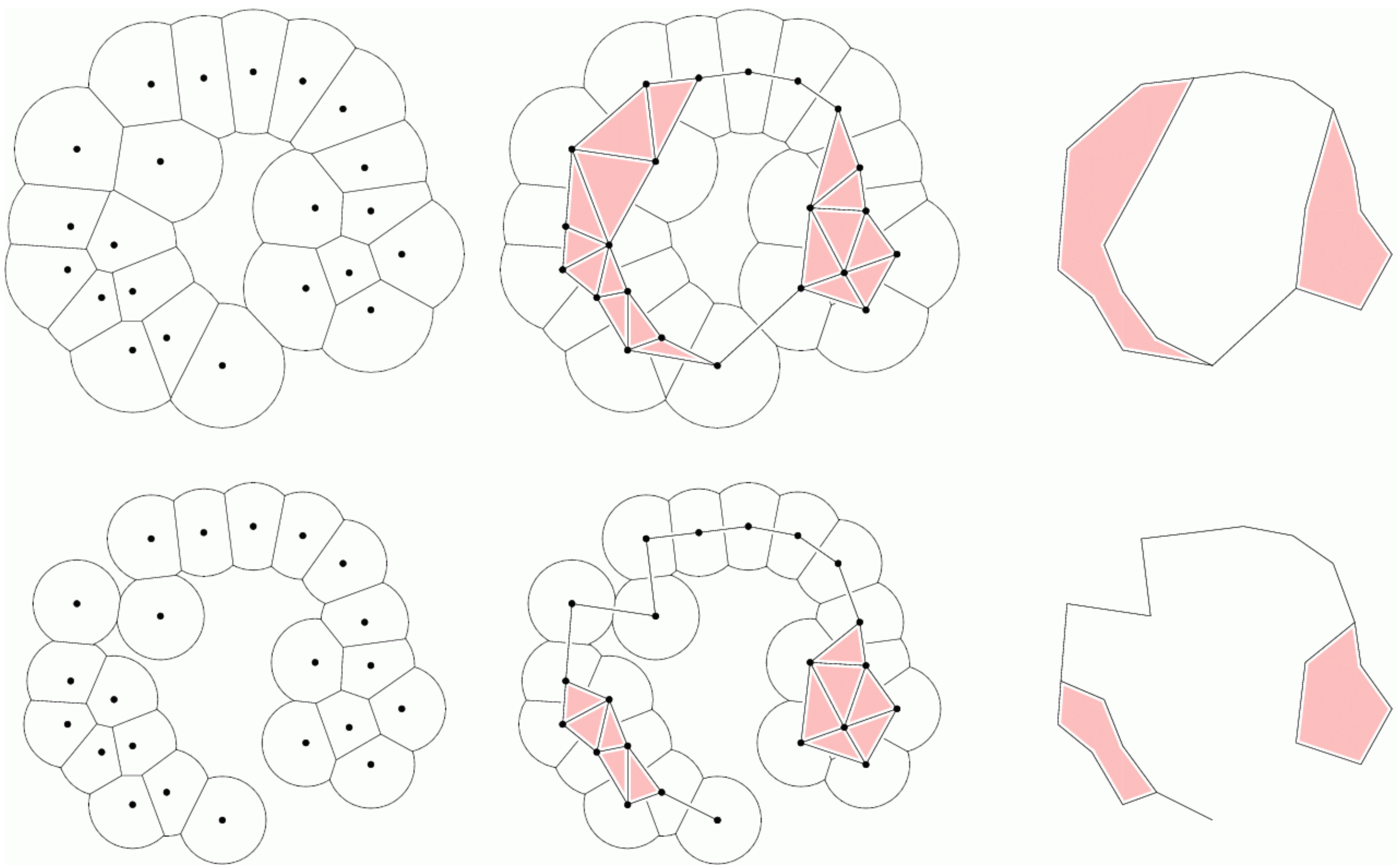
Recall: alpha shapes

- Given a radius α and a set of points, we take the union of all radius α balls at those points.



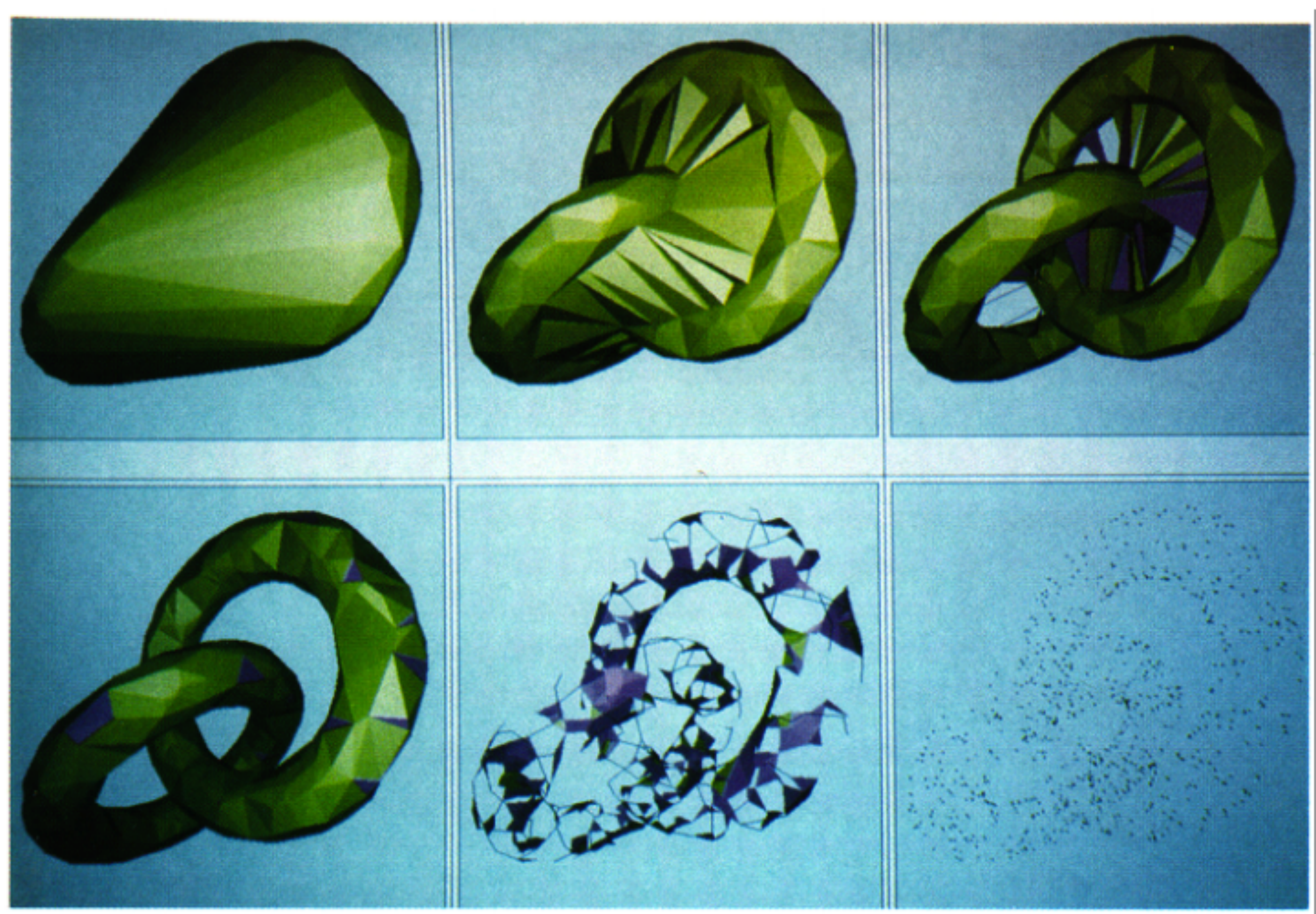
Recall: alpha complex

- The α-complex is then the nerve of this set of balls:



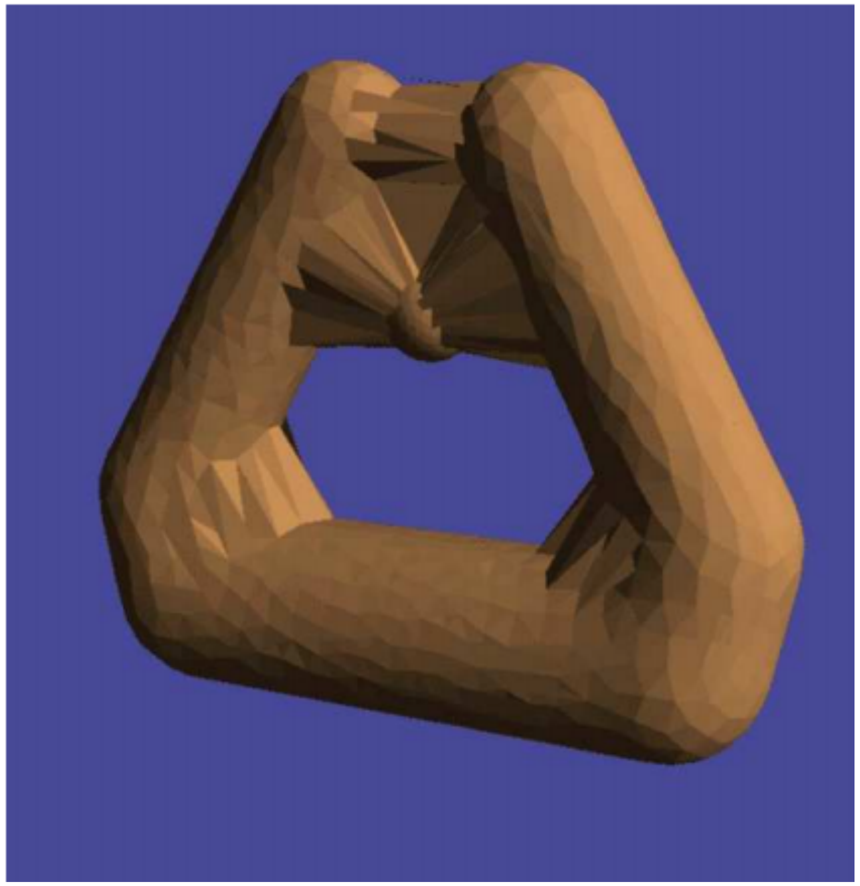
3d α-shapes

- In fact, one early reconstruction algorithm was just based on using α-shapes directly [Edelsbrunner-Mucke 1994]



Downside of alpha shapes

- However, there is a downside:
  - Finding the “perfect” alpha is difficult
  - And there might not even be one
  - Alpha shapes also overfill certain types of objects
- Some variations (anisotropic alpha shapes) attempt to compensate by scaling the ball according to local features



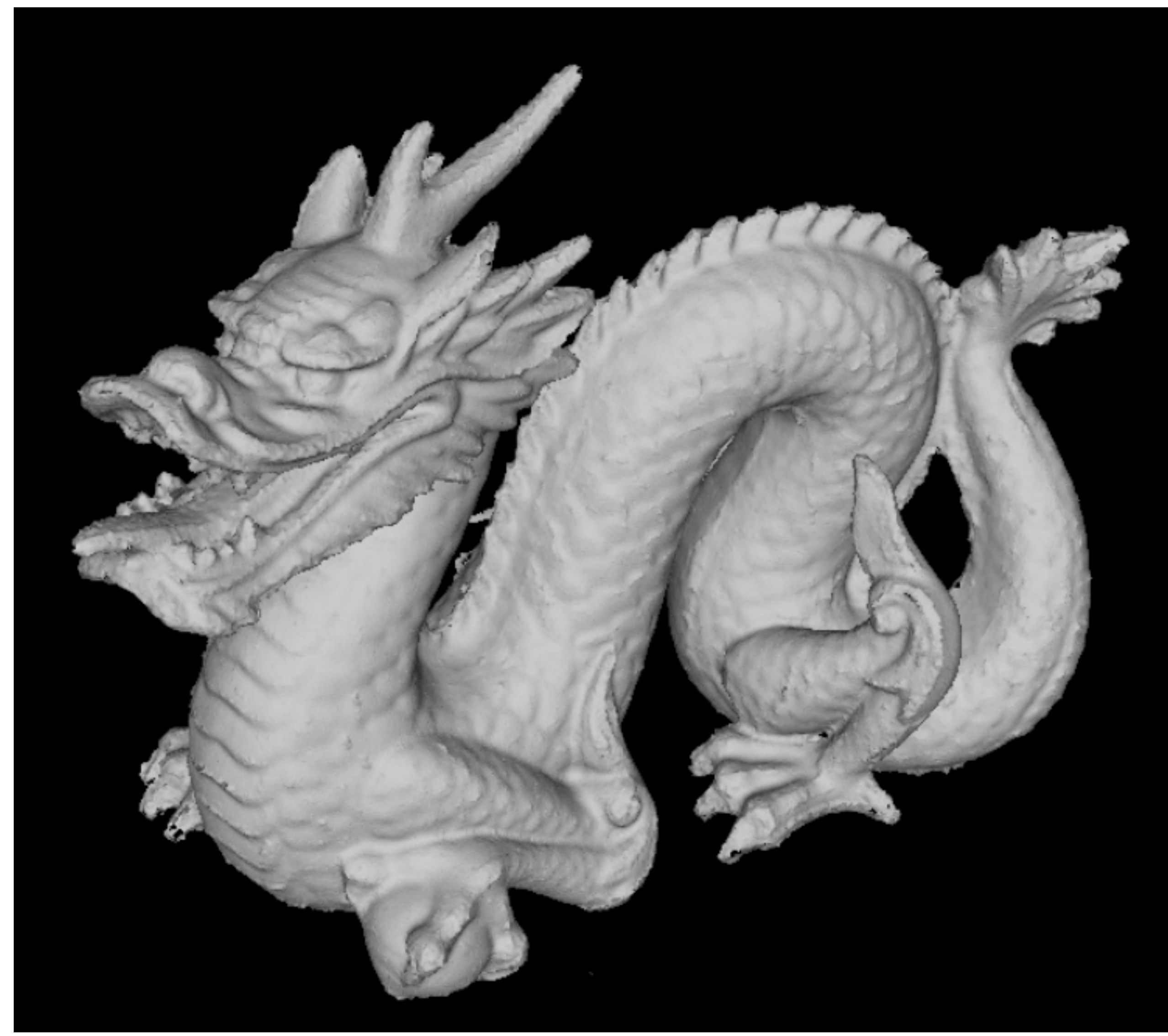
Teichmann and Capps 1998

Ball-rolling algorithm

- Another nice early extension which used the α-shape was the ball pivot algorithm [Bernardini et al]:
  - Starting at a seed triangle, pivot a ball around each edge of the triangle until a new sample point is hit.
  - Add that triangle to the mesh and continue.
- Fast and generates nice things, but also tricky

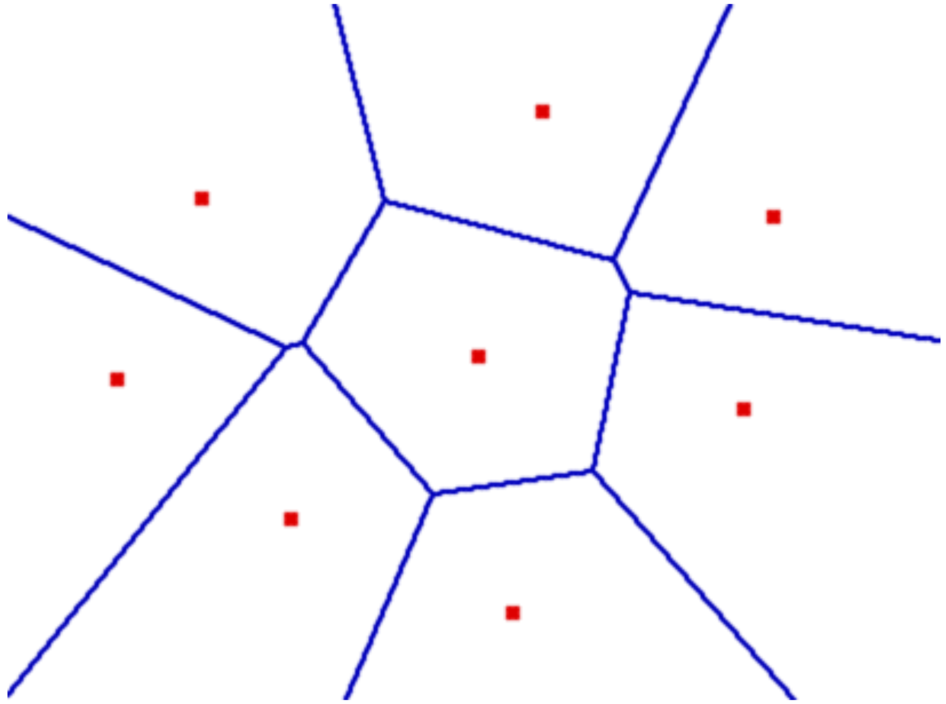
Ball rolling algorithm (cont.)

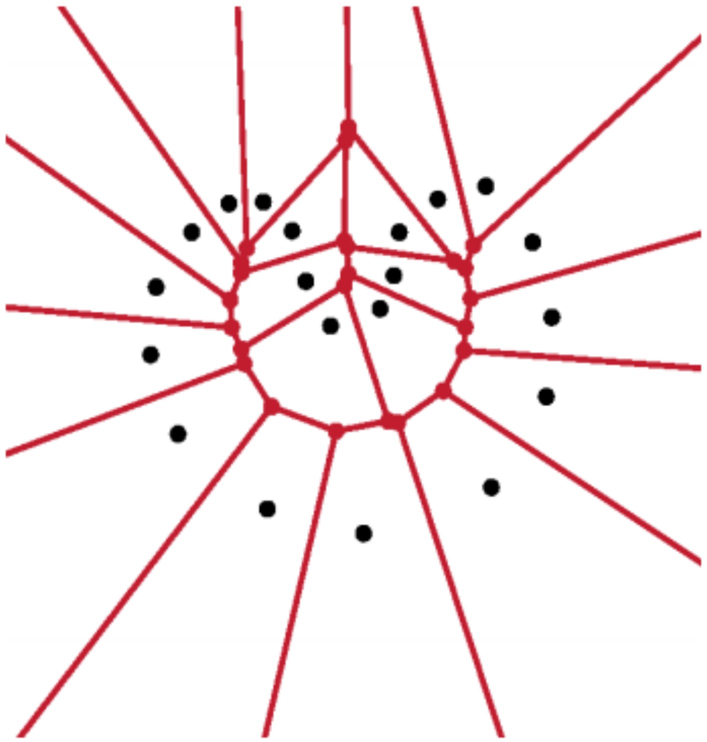
- Pros: Conceptually simple, very fast to implement
- Cons:
  - No theoretical guarantee of quality in terms of the topology
  - Not even always a surface
- Choice of ball is key: can miss things or fall through holes



The crust algorithm: 2d

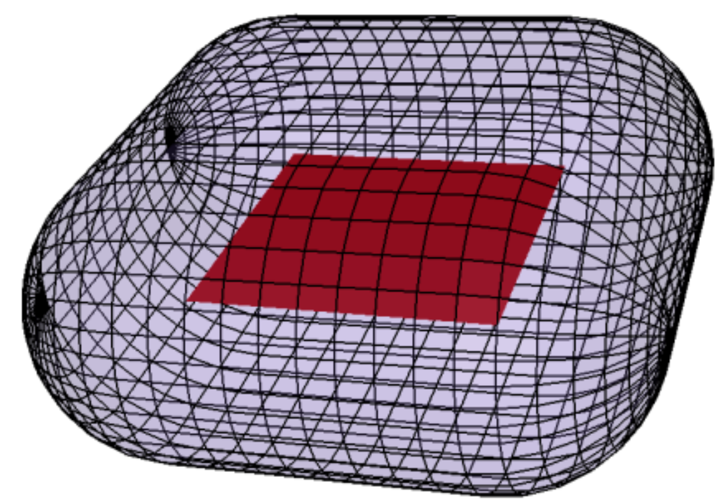
- If we go back to a 2d idea:
  - The Voronoi diagram is the division of the plane into cells where each cell consists of points closest to one of the input points:

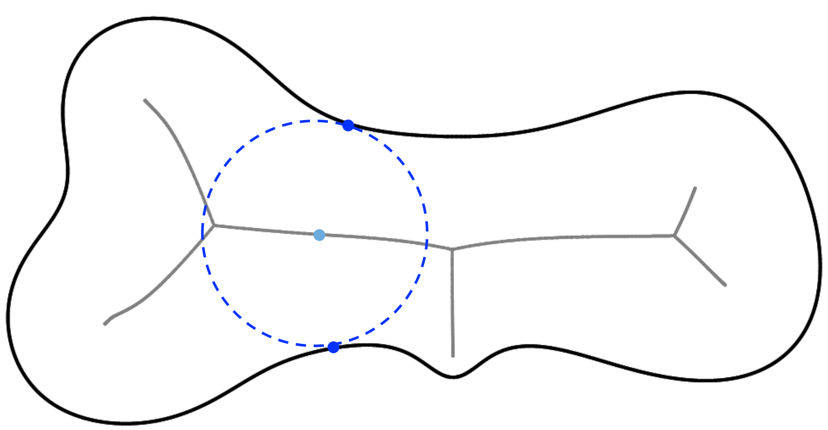


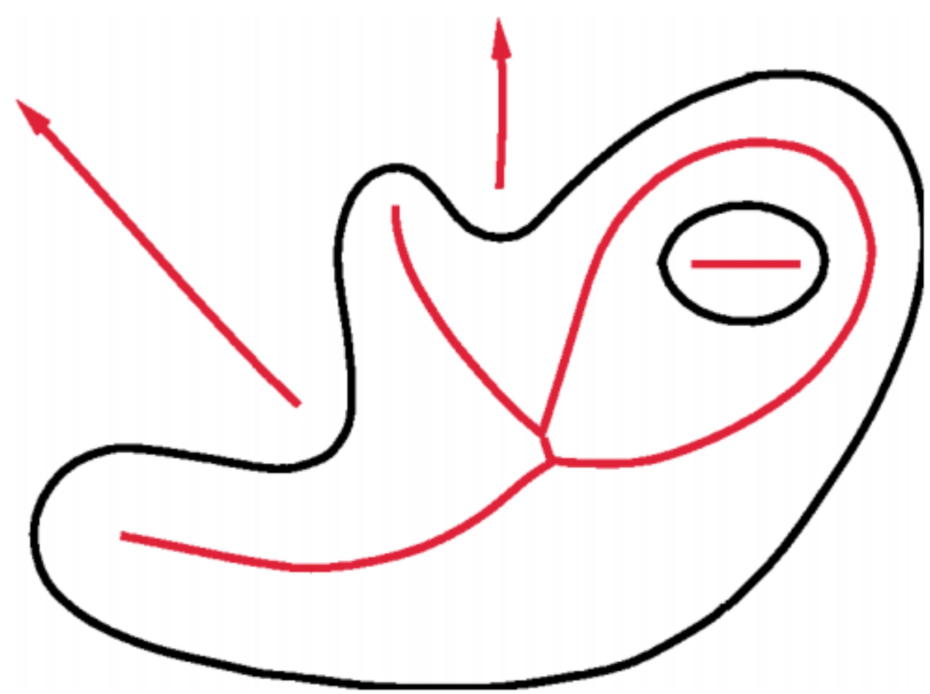


Related: medial axis

- The medial axis of a shape is the set of points with more than one closet point on the shape.
- Noisy, but correct topology

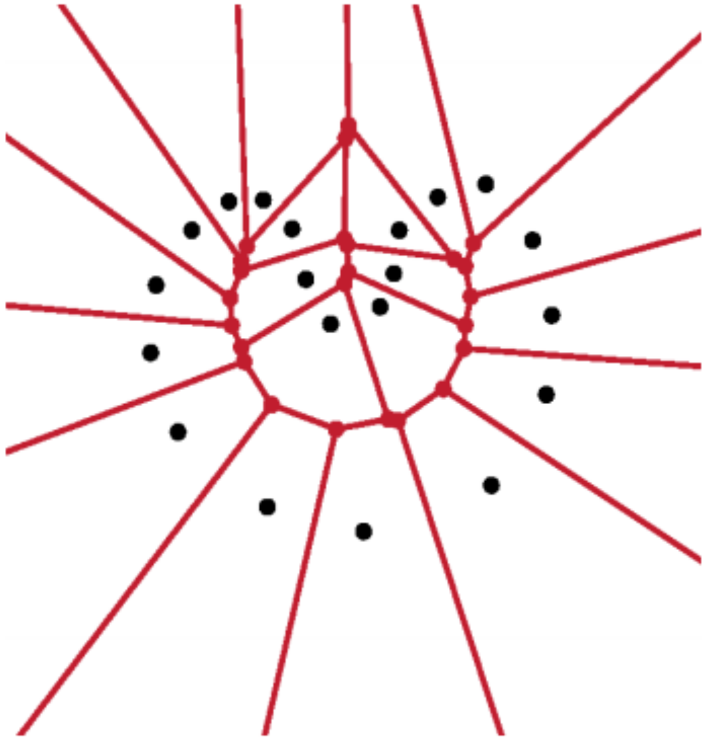






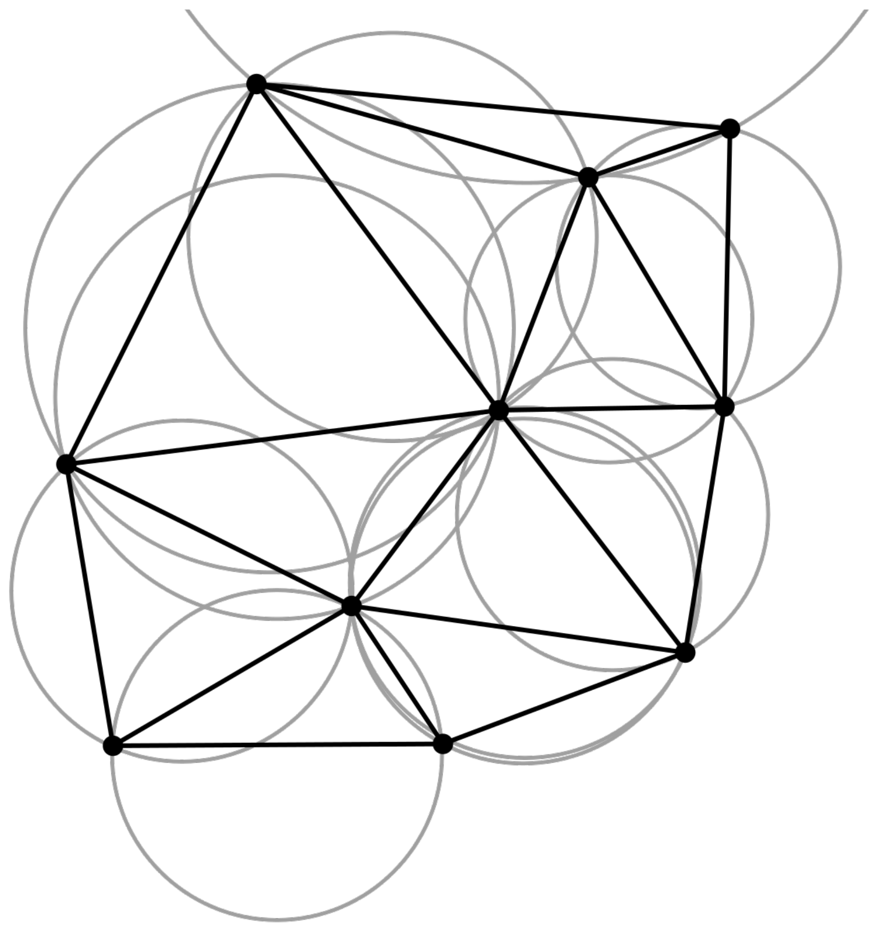
The connection

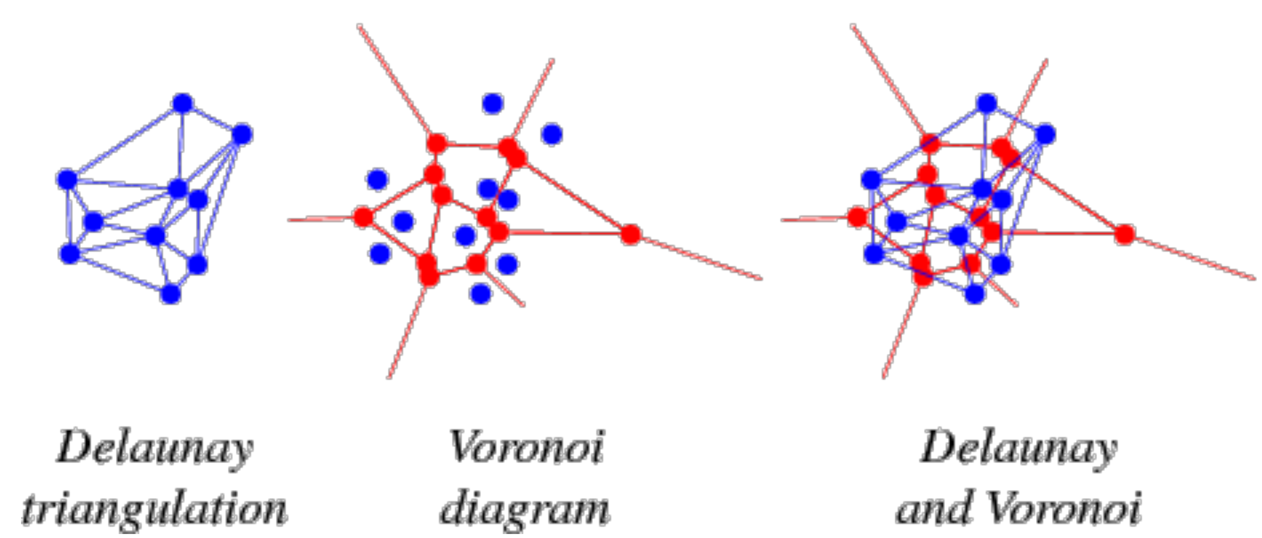
- In 2d, the Voronoi diagram of a point set that closely samples an underlying shape will contain an approximate medial axis of the shape:



Back to curve reconstruction

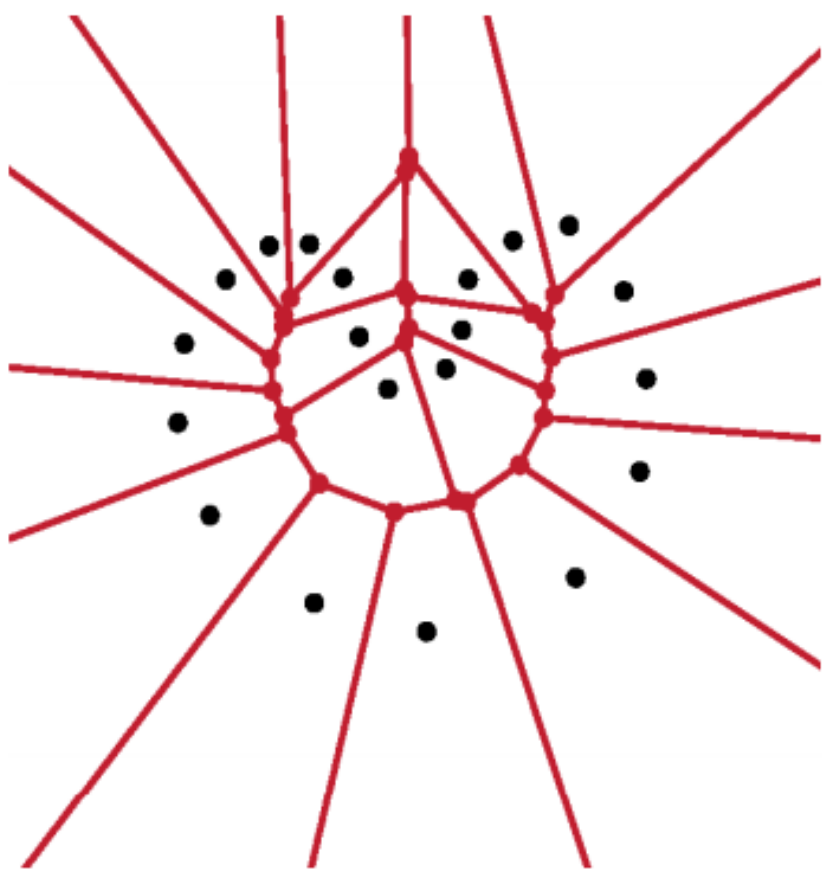
- Recall the dual to the Voronoi diagram: the Delaunay triangulation is the set of simplicies where the circumcircle of those simplicies is empty of other sites





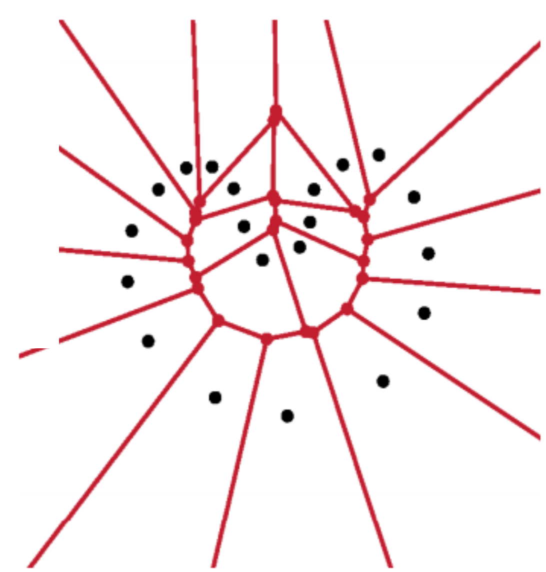
2d crust algorithm

- In 2d, we want to select any edge of the Delaunay triangulation whose circumcircle is empty not only of sample points, but also of the Voronoi vertices:



2d crust algorithm

```
Crust(P)
compute Vor P;
let V be the Voronoi vertices of Vor P;
compute Del (P ∪ V);
E := ∅;
for each edge pq ∈ Del(P ∪ V) do
    if p ∈ P and q ∈ P
        E := E ∪ pq;
    endif
output E.
```



Crust algorithm (2d)

Why?

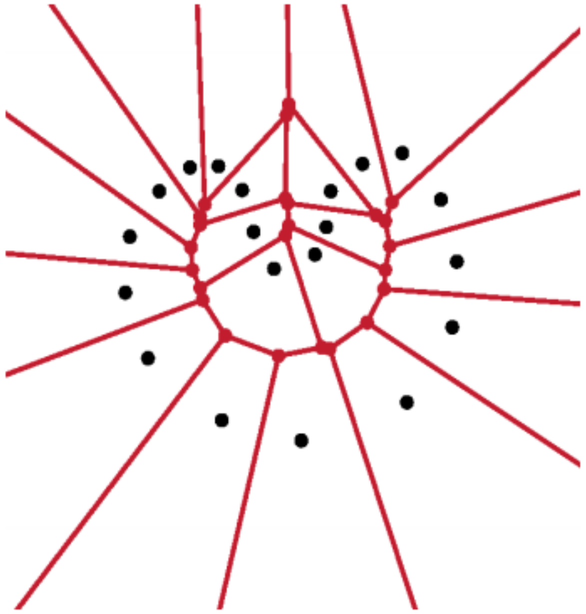
- Key lemma: Any Voronoi disk of a set of points sampled from a curve in the plane must contain a medial axis point of the curve.

    - Sketch: Essentially, the Voronoi disk's center is equidistant from more than 1 point on the curve, so it should be on the medial axis.

Why?

- Key lemma: For a fine enough sample S of a curve, an edge between two non-adjacent samples cannot be circumscribed by a circle that is empty of both Voronoi vertices and sample points.

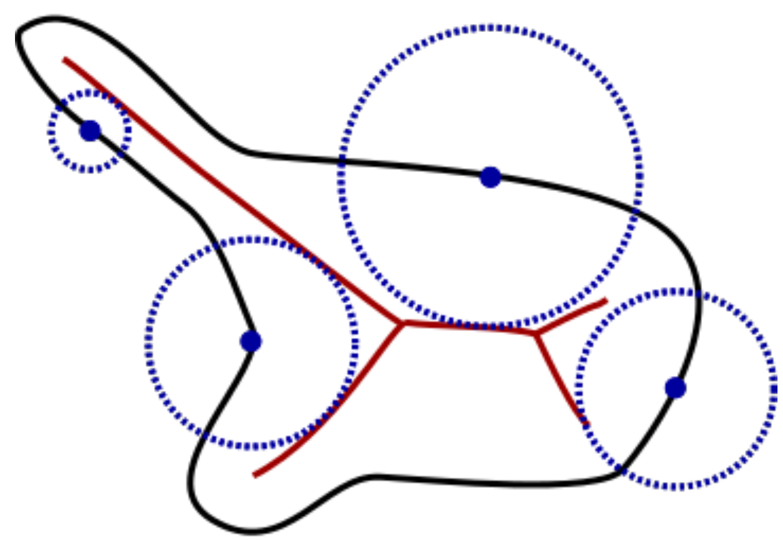
  - Proof by picture:



“Fine enough” sample

- More precisely: we must sample based on local feature size, lfs

    - For any x from the curve F, lfs(x) is the distance from x to the nearest medial axis point



- We say it is ε-sampled if every point p on the underlying curve is within ε×lfs(p) of a sample point

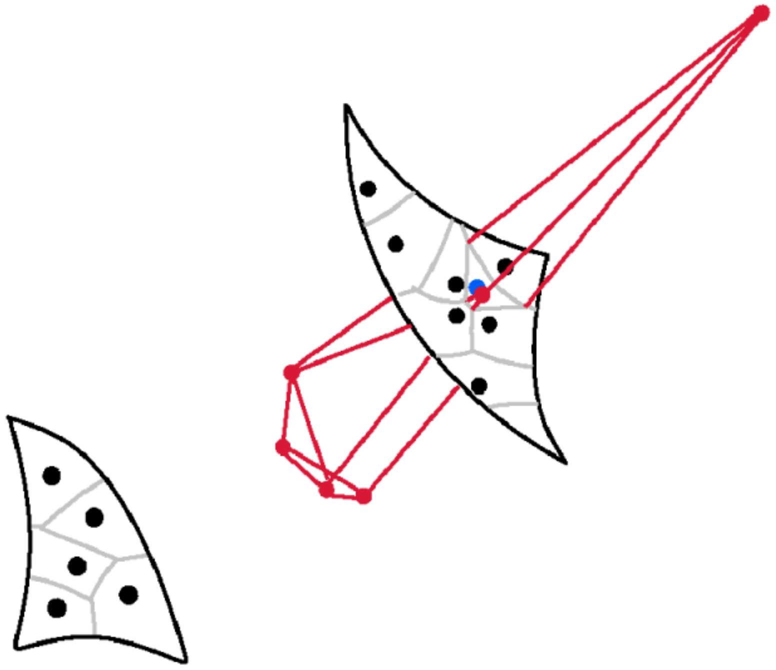
Algorithm for 2d:

- Compute the Delaunay triangulation and the Voronoi diagram of the point set. Include an edge from the triangulation if its circumcircle is empty of all Voronoi vertices.

  - Theorem: The crust of an ε-sample of a smooth (twice differentiable) curve, for ε≤.25, will connect only adjacent sample points.

Moving to 3d

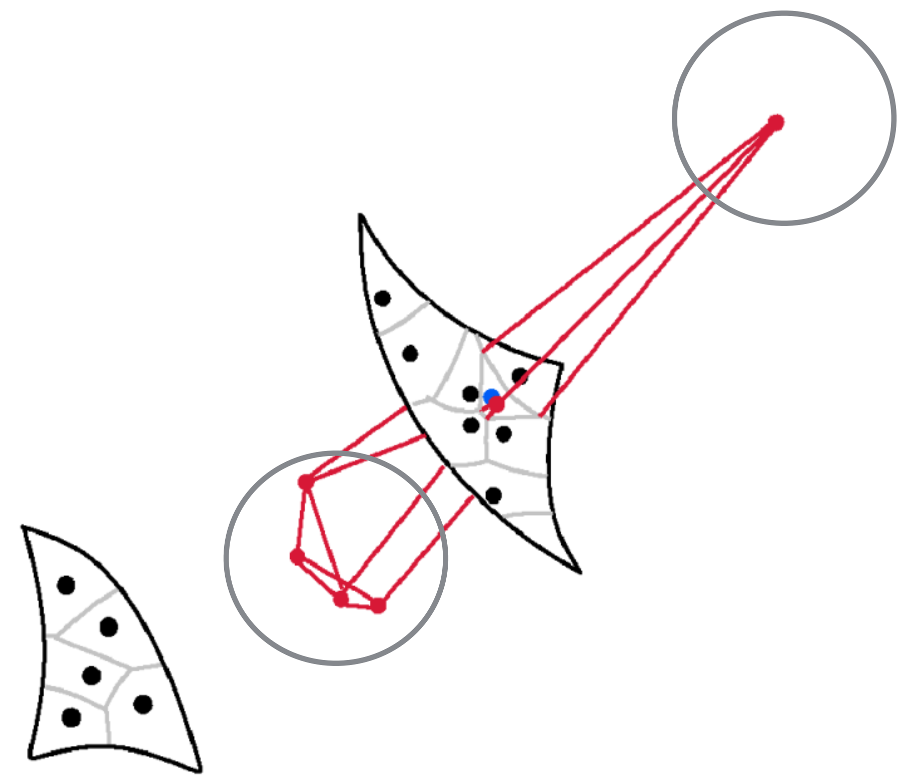
- Unfortunately, this simple filtering will NOT work for surfaces in 3d, because Voronoi vertices do not have to lie near the medial axis, no matter how dense the sample.



Finding a good subset

- However, some of the points are good!

Intuitively, we want to
take cells that exclude
the points of the cell that
are farthest away;
these are the ones
near the medial axis.



Poles

- To formalize this, in [Amenta-Bern] they define the poles of a sample point to be the two farthest vertices of its Voronoi cell, one on each side of the surface.

    - Of course, the algorithm doesn't know the surface!

- Instead, it chooses the point furthest away as the first pole, and then the second is chosen to be the farthest in the opposite half space.

How do to this:

- More formally: if s is the sample point and p the first pole chosen, among all vertices q of the Voronoi cell with the angle $\angle psq > \pi/2$, choose the furthest one

- **Lemma**: Given an ε-sample of a surface, with $\varepsilon<1/4$, and a sample point s with farthest pole p. Then the second pole v will be the farthest Voronoi vertex where the vector sv has negative dot product with sp.

How to prove:

- I won't go into detail - they get fairly technical:

Lemma 3.4 (Edge Normal.) *For an edge pq with $\|p - q\| \leq 2f(p)$, the angle $\angle_a(\overrightarrow{pq}, \mathbf{n}_p)$ is at least $\frac{\pi}{2} - \arcsin \frac{\|p-q\|}{2f(p)}$.*

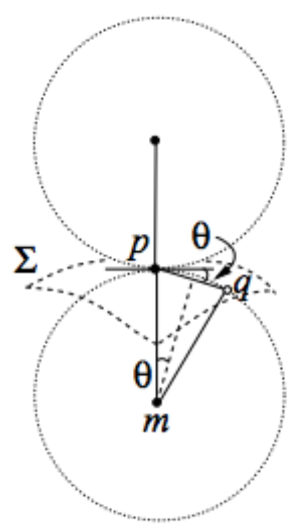
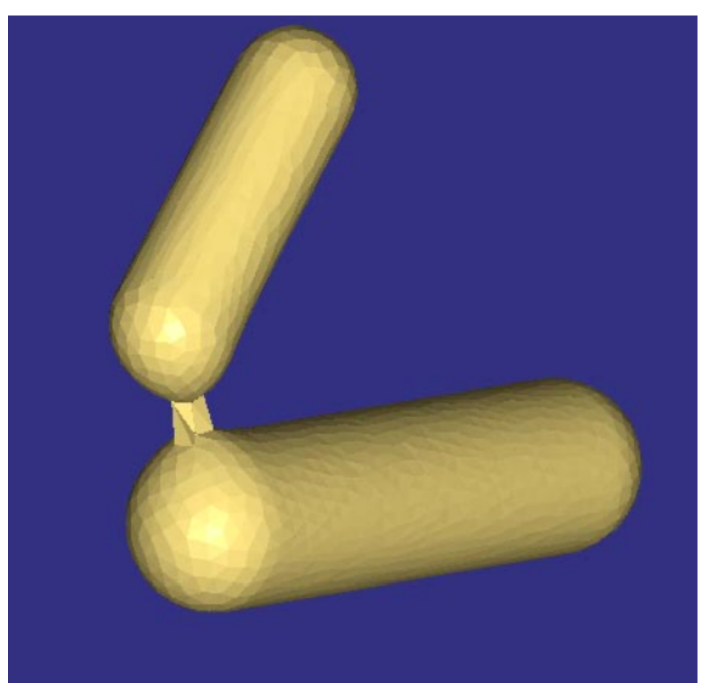


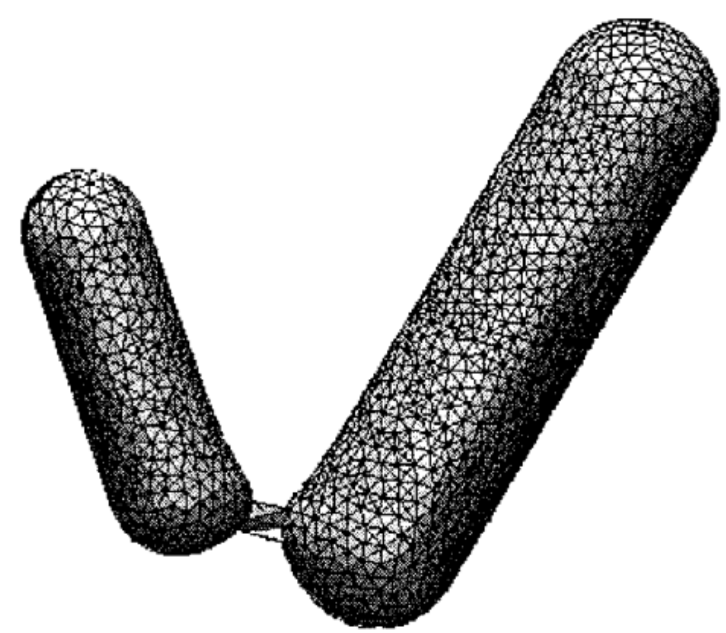
Figure 3.5: Illustration for the Edge Normal Lemma 3.4.

Dey 2006

The crust

- We then take the Delaunay triangulation of the input points and their poles.
- The **crust** is the set of Delaunay triangles from this triangulation where all three vertices are sample points.





Quality

- At this point we have a fairly weak theoretical guarantee: it is pointwise convergent to the underlying surface as the sampling density increases.

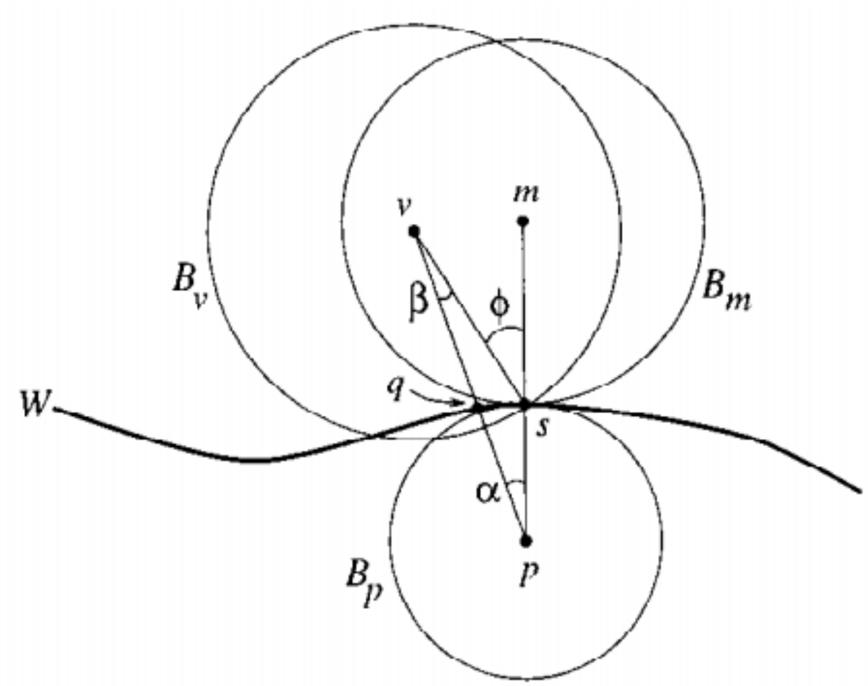
- However, we can still clearly have extra triangles in the result, as there is no guarantee that the normals at each triangle are close to the actual surface normals.

Additional filtering

- The next step in the algorithm is to filter:
  - The bad triangles we want to remove are nearly perpendicular to the underlying surface.
  - However, we don't know the underlying surface!

Using the poles

- Instead, we go back to the poles: we can prove that the line from a sample point to each of its pole is nearly orthogonal to the surface, given a sufficiently dense sample.

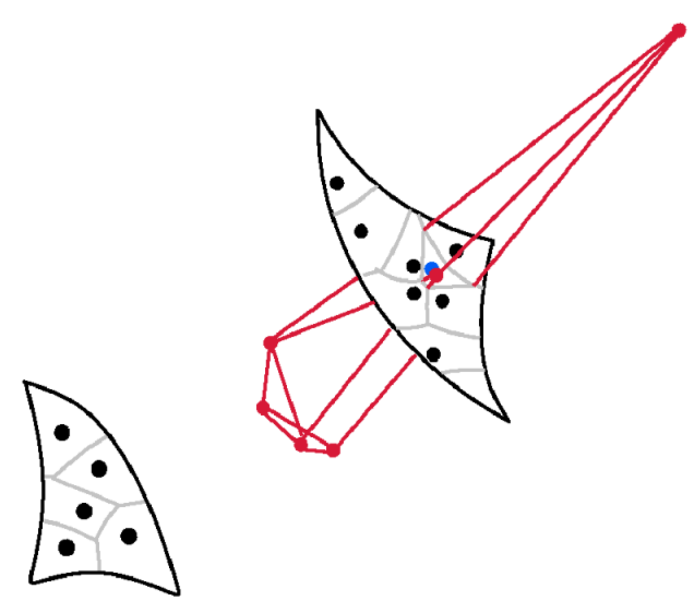


Next step in the algorithm:

- Remove any triangle T for which the normal to T and the vector to the pole at a vertex of the triangle are too large.
    - Greater than θ for the largest angle vertex of T, and greater than $3\theta/2$ for all others.
    - θ is another input parameter, which they set to be 4ε to get good practical results, but this can also be varied to find a “nice” output.

Theoretical guarantee

- More precisely: Take an ε-sample, and set θ=4ε. Let T be a triangle of the crust, trimmed as described on last slide, and take any point t∈T.
  Then the angle between T's normal and the normal to the actual underlying surface at the point closet to t measures O(√ε).



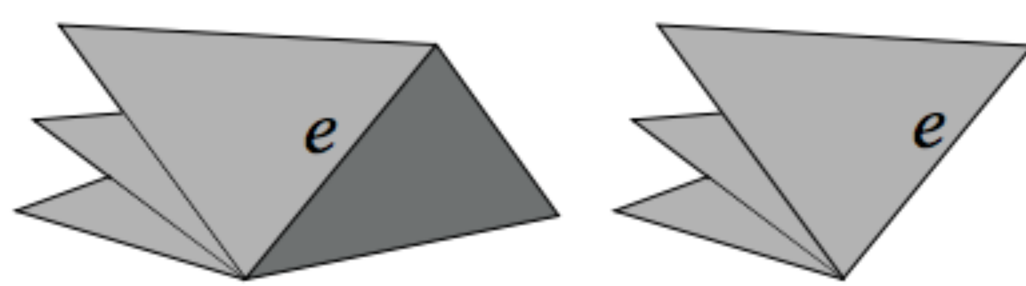
Final cleanup

- After filtering by normals, remaining triangles are roughly parallel to the original surface.
  - Can prove that this set of triangles still contains a piece-wise linear surface homeomorphic to F.
- However, we don't necessarily have a surface, since there could be small remaining triangles that enclose pockets:
  - All 4 faces of a very flat tetrahedra may make it past the filtering step.

Sharp edges

- Define a sharp edge as one which has a dihedral angle greater than $3\pi/2$ between a successive pair of incident triangles in the cyclic order around the edge.
    - In other words, an edge is sharp if all incident triangles are in a small wedge.
    - If only one incident triangle, then automatically sharp.

not sharp



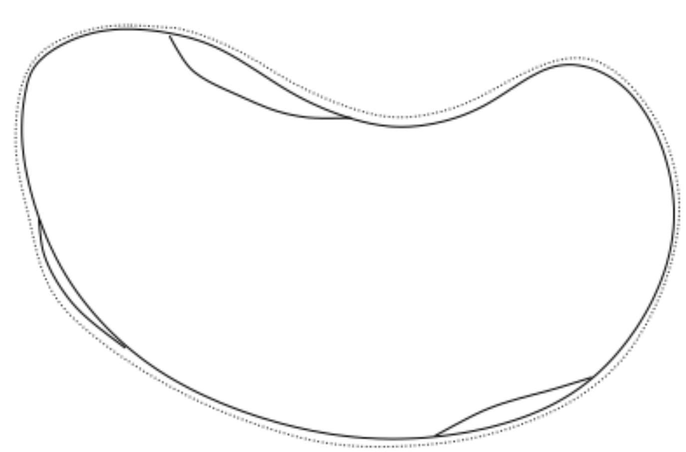
sharp

Dey 2006

Cleaning up

- Can use the wedges to prove that no Delaunay triangle we need to keep will have a sharp edge - so can trim without losing anything important.

- Even with sharp edges trimmed, we still may have “pockets”, where we kept both an inner and outer layer.



Dey 2006

Final trimming

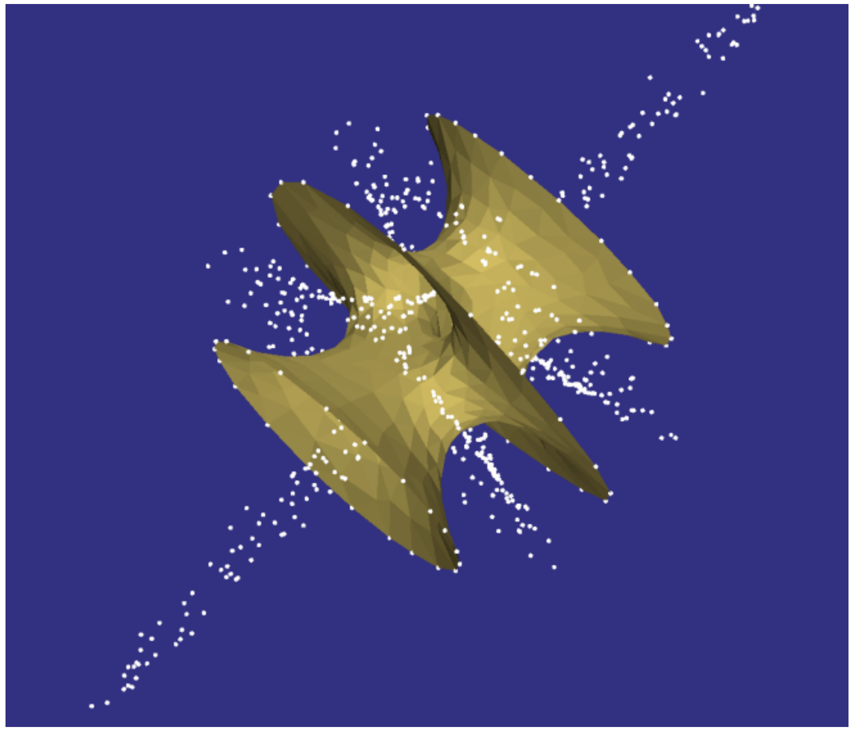
- The final step:
  - orients triangles and poles consistently
  - greedily remove triangles with sharp edges
  - take the “outside” of remaining triangles (which makes sense since we oriented things)

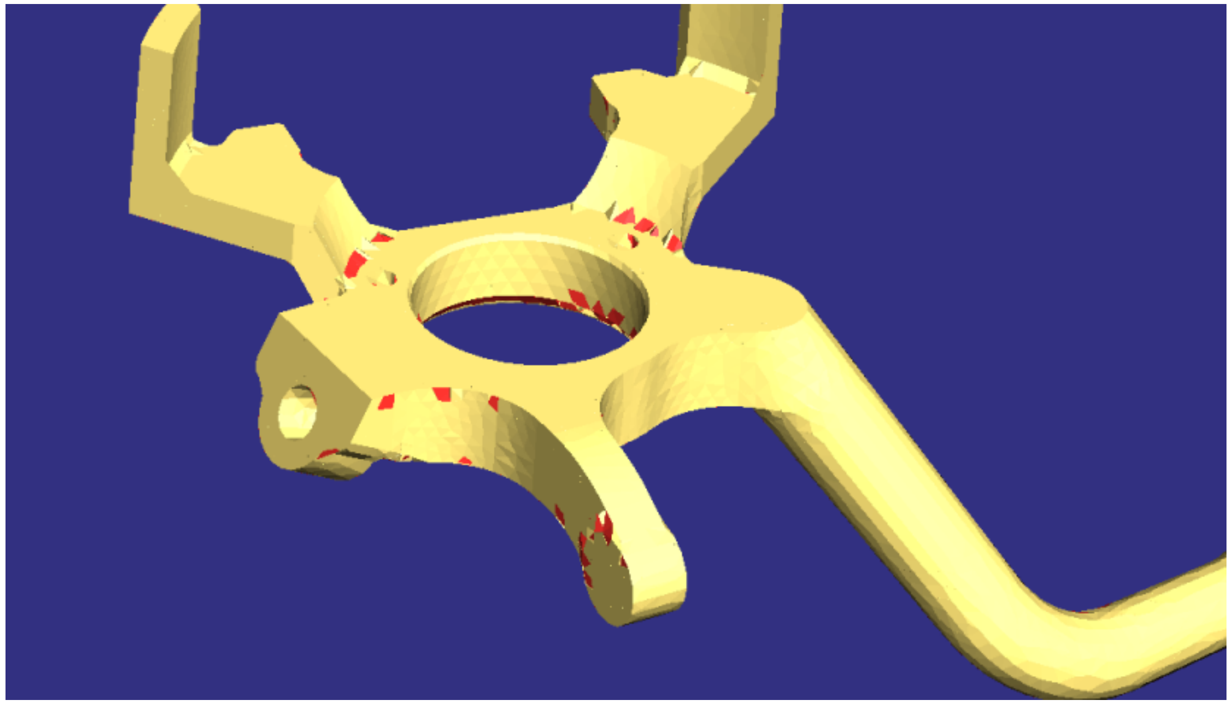
Crust: takeaway

- This was the first algorithm with good, provable guarantees on the quality of the reconstruction.

- The main drawback is ε-samples: it's hard to guarantee a good enough approximation.

- It is also only good for smooth inputs: anything with sharp edges can have holes

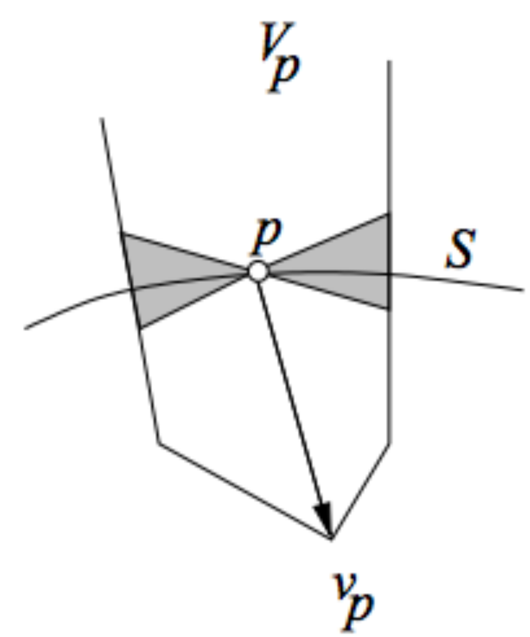


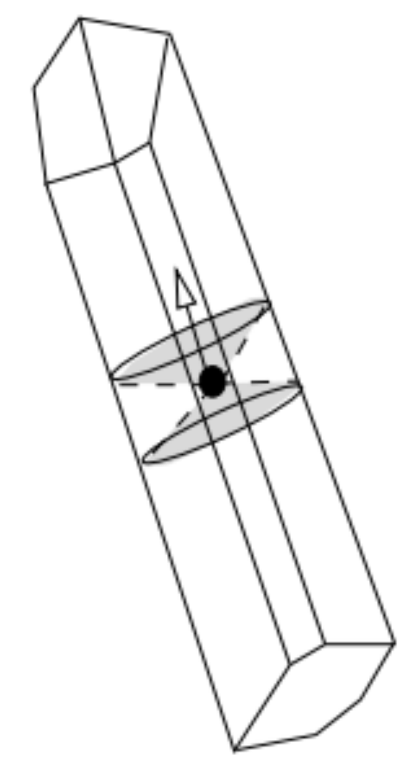


Extension: cocone

- The Cocone algorithm uses the poles from the crust algorithm in order to enumerate a set of triangles that will contain a good reconstruction:

We find any Voronoi edges that intersect the “cocone”, and take triangles from the Delaunay triangulation that are dual to one of these edges.





Cocone result

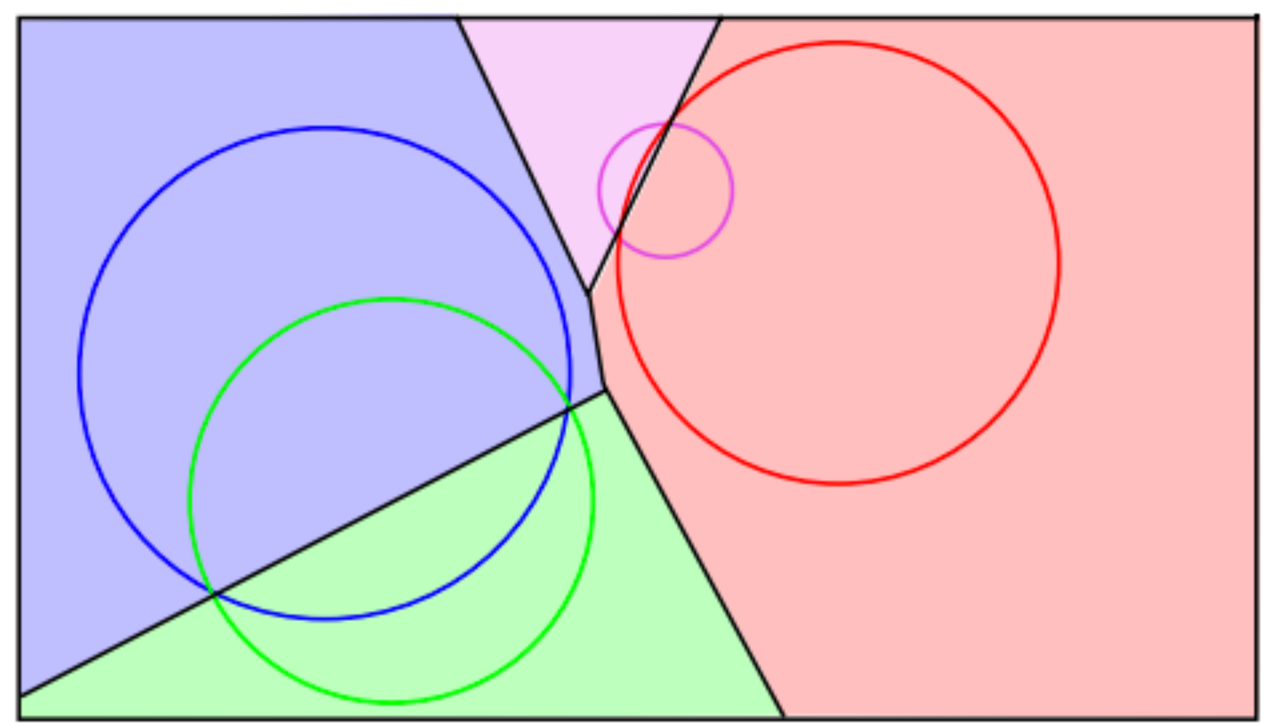
- In the end, the output of cocone is homeomorphic to the original surface, for $\varepsilon \leq .05$.
- In addition, they are also isotopic.
- (Really, same guarantees as in crust, but much simpler to prove and faster to implement.)

Extension: power crust

- The power crust algorithm computes a weighted Voronoi diagram:

- Think of a point c with weight ρ^2 as a ball $B_{c,\rho}$.

- Then the power distance between a point x and a ball $B_{c,\rho}$ as $d^2(c,x)-\rho^2$



Power crust

- The power crust algorithm then just uses the pole vertices (and their Voronoi balls)
  - It computes the power diagram of these polar balls, and does a similar filtering as the normal crust algorithm afterwards.
- It does do better on poorly sampled inputs and things with sharp corners, in practice.
  - However, the known theoretical guarantees are similar to crust.

More recent trends

- It is difficult to know the “correct” value of ϵ
- Also, ϵ may not be needed everywhere: really, the key value is local feature size
  - Near finer features with higher curvature, you need to sample more
  - On flatter features, want to sample less
- The witness complex attempts to balance this by selecting a subset of the points to keep.

Witness complex

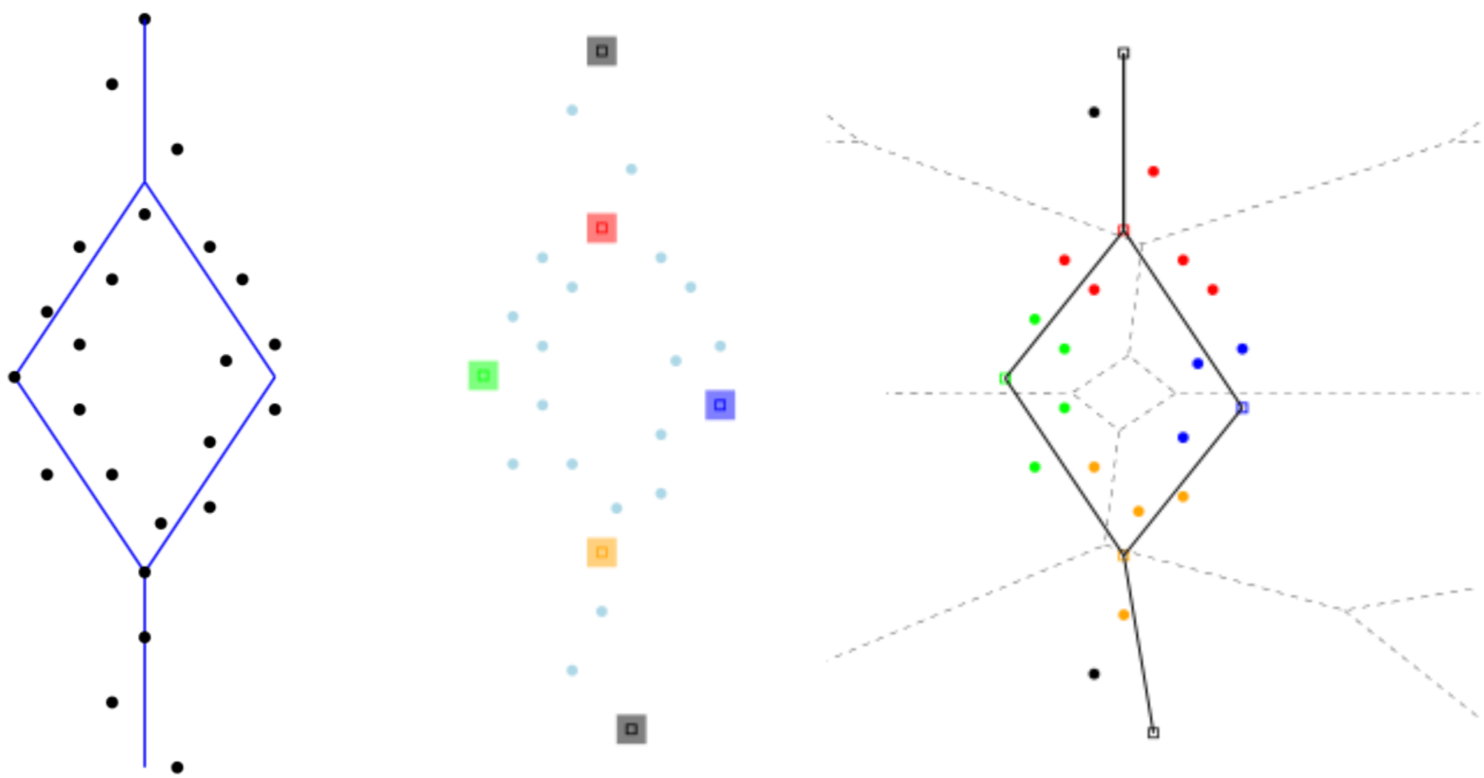
- Introduced by de Silva in 2003:

$$\text{A simplex } \sigma = \{q_0, q_1, \ldots, q_k\} \text{ is weakly witnessed by a point } x \text{ if } \forall i \in [0,k], q \in Q \setminus \{q_0, \ldots, q_k\}, d(q_i) \leq d(q,x)$$

- Given Q a subset of P, the witness complex W(Q,P) is the collection of simplices with vertices from **Q** whose all subsimplices are weakly witnessed by a point in **P**.

- Why use it? Allows reduction in number of points, and can work well.

Witness complex



P L W(L,P)

- This leads towards a notion they call ϵ sparsity

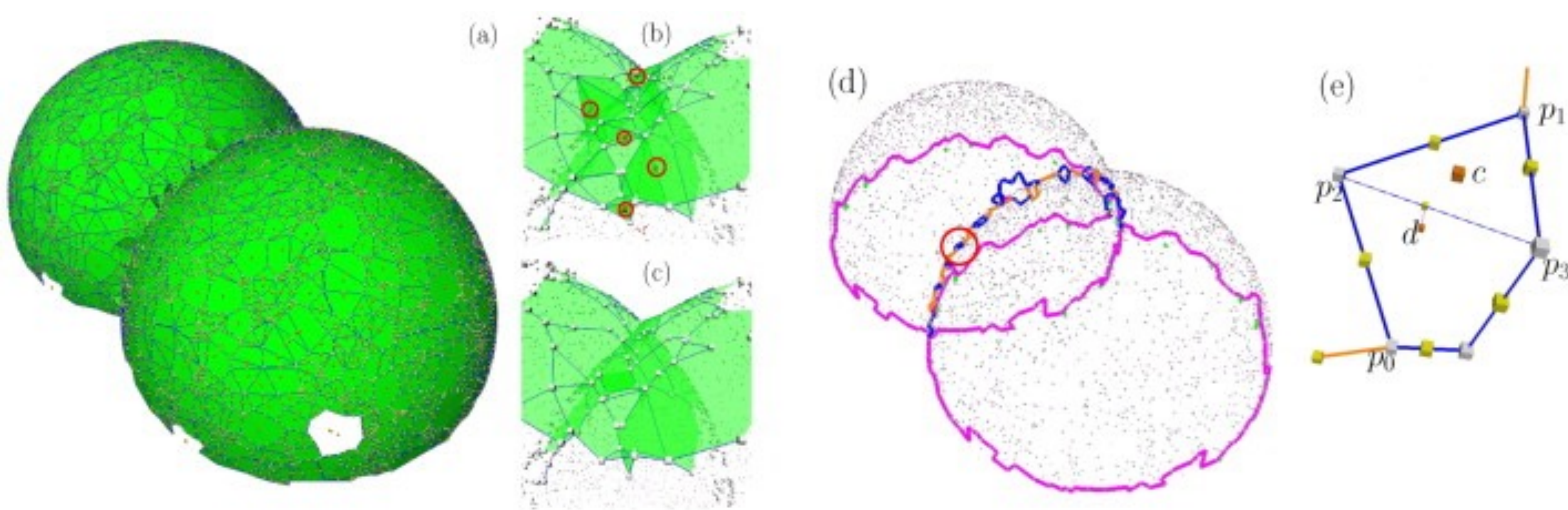
Example application

- Example algorithm [Chazal-Oudot 2007]:
    - Compute a witness complex
    - Iteratively add points from the witness complex, computing their persistent homology as you go
    - At some point along the way (assuming a “nice” sample), the prove that you will find a pair of Rips complexes that contain a complex with the homology of X in between them
    - (Recall that $\check{C}(X,r) \subseteq VR(X,r) \subseteq \check{C}(X,2r)$)

Dealing with singularities

- Newer algorithms have tried various ways to deal with corners or non-manifold shapes:

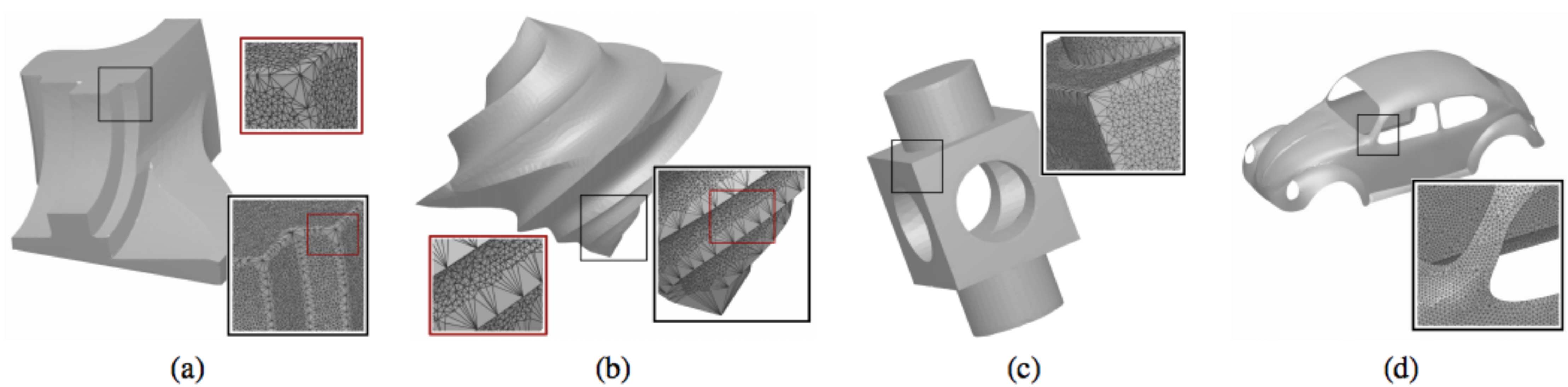
- Some incorporate persistence to detect “big” features, and use offsets instead of Voronoi diagrams to compute representations:



Cazals and Cohen-Steiner 2012

More singularities

- Other algorithms used have used tools like Reeb graphs and Laplacians in order to detect the different patches or irregularities, and recover smooth surfaces in between



Dey et al 2012

The list continues

- I've focused on Delaunay-based methods in this talk, but there are also offset surfaces, implicit surfaces, and numerous other types of algorithms to explore - often your data even constrains you.