Delaunay Tetrahedralization Constructions and Smooth Boundary Mesh Rendering

Quan Yuan Yu Qu

Nov. 30, 2017

1 PROBLEM DESCRIPTION

In this project, we create a tetrahedralized structure between two clouds of points on a pair of paralleled planes respectively, then grow the structure by some radius, and finally create a high-resolution water-tight triangle mesh that approximates the boundary of the grown structure.

The two paralleled planes where two clouds of points (a.k.a. “site”) on are named “floor” and “ceiling” respectively. This project is divided into two tasks:

(1) Construct the Delaunay Tetrahedralization structure of the union of the clouds of points on two paralleled planes.

(2) Compute the high-resolution water-tight triangle mesh that approximates the boundary of the structure in (1) growing by a radius r and render the mesh using smooth shading with visible and hidden silhouettes drawn.

2 SOLUTION OUTLINE

The whole solution is divided into following steps:

1. Delaunay tetrahedralization between two clouds of points on a pair of paralleled plane.
    a) Compute the centers of the circumscribing circles and spheres, with visualization in order to make it easier for further debugging and validations.
    b) With known centers, compute the bulge of the sphere generated from a triangle. Similarly, make it visible for further tasks.
    c) Compute the naive Delaunay Tetrahedralization for three conditions:
        i. 3 balls on floor and 1 ball on ceiling;
        ii. 1 ball on floor and 3 balls on ceiling;
        iii. 2 balls on floor and 2 balls on ceiling (1 edge on floor and 1 edge on ceiling).
    d) Find and remove the duplicated edges. Counting the number of site, tetrahedra and beam on floor and ceiling and display the result on canvas. Different key selections, colors and additional shapes are adopted to show these results.
2. Tetrahedralized structure triangularization.
    a) Sample vertices from triangle.

b) Sample vertices from sphere surface.

c) Sample vertices from right circular cylinder surface.

d) Use ball pivoting to compute a triangular mesh for all vertices.

3 DETAILS AND IMPLEMENTATION

3.1.1 Delaunay Tetrahedralization

3.1.1.1 Compute the centers of the circumscribing circles

Given three points A, B and C that are not on the same line, the center of circumcircle P can be calculated by barycentric coordinates $P = (1 - s - t)A + sB + tC$, where s and t can be solved by the equations

$$\begin{cases} AP^2 = BP^2 \\ AP^2 = CP^2 \end{cases}$$

This gives

$$\begin{cases} s = (pq - qr)n \\ t = (pq - pr)n \end{cases}$$

where

$$\begin{cases} p = AB^2 \\ q = AC^2 \\ r = AB \cdot AC \\ n = \dfrac{1}{2(pq - r^2)} \end{cases}$$

The pros in this way is that we don't need to worry about the sequence (or orientation) of points.

3.1.1.2 Compute the centers of the circumscribing spheres

Given four points A, B, C and D that are not on the same plane, and any three of them are not on the same line, the center of circumcircle P can be calculated by barycentric coordinates $P = (1 - s - t - u)A + sB + tC + uD$, where s, t and u can be solved by the equations

$$\begin{cases} AP^2 = BP^2 \\ AP^2 = CP^2 \\ AP^2 = DP^2 \end{cases}$$

This gives

$$\begin{bmatrix} a & d & e \\ d & b & f \\ e & f & c \end{bmatrix} \begin{bmatrix} s \\ t \\ u \end{bmatrix} = \frac{1}{2} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

where

$$\begin{cases} a = AB^2 \\ b = AC^2 \\ c = AD^2 \\ d = AB \cdot AC \\ e = AB \cdot AD \\ f = AC \cdot AD \end{cases}$$

3.1.1.3 Compute the bulge of a sphere generated from a triangle

Given four points A, B, C and D that has a circumscribing sphere with center O and radius r, the circumcircle of points A, B and C is with center O_1 and radius r_1, then the bulge d for point D is given as

$$d = r + \text{sign}(O_1D \cdot O_1O)|O_1O|$$

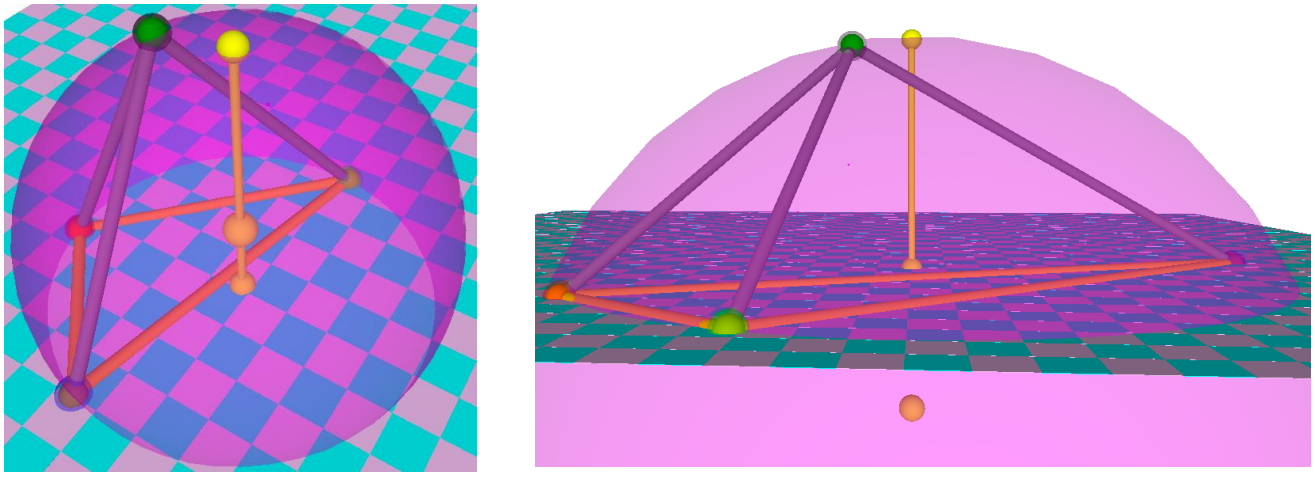


Figure 1 Bulges in two different configurations

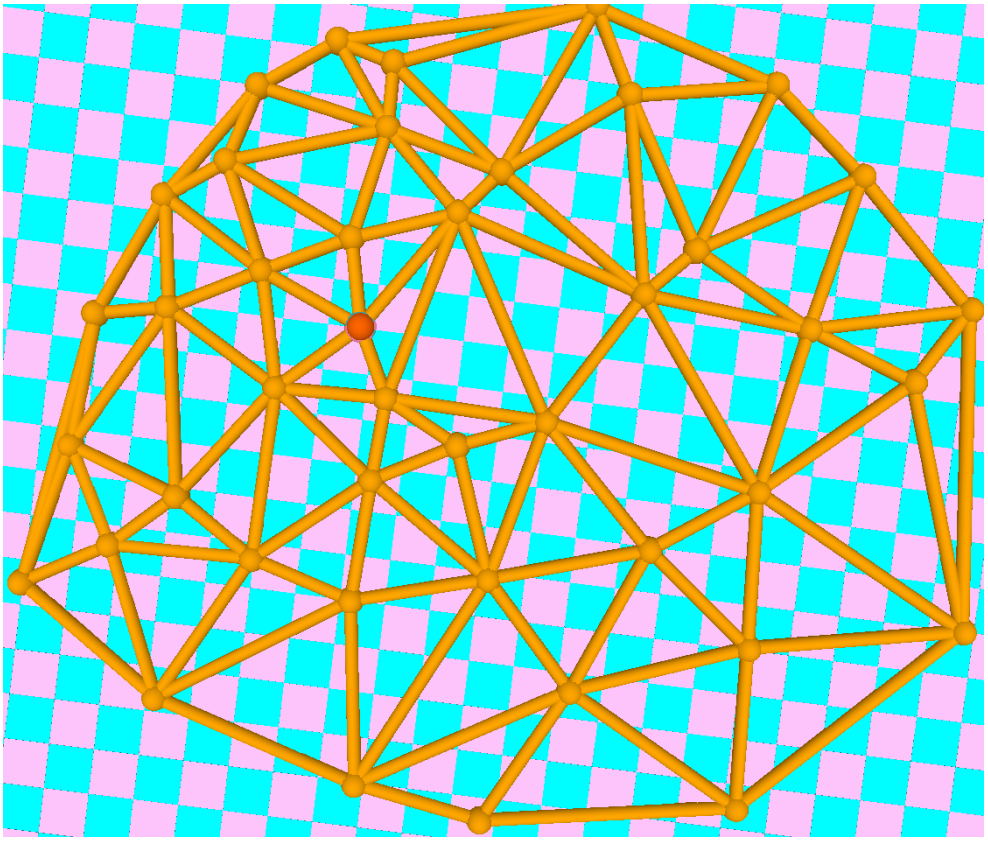
3.1.1.4 Compute the Delaunay Triangulation on each plane

Here we use the naïve method to compute the Delaunay triangulation. The algorithm is list in Table 1. Time complexity is $O(n^4)$.

Table 1 naïve Delaunay triangulation

```
for each triplet t of all points on the plane
  calculate the circumcircle c(O, r) from t
  if anyOtherPointInCircle(c, t)
    continue
  add three edges to edge list
  add triangle to triangle list
```

The result looks like below.



3.1.1.5 Compute the Delaunay Tetrahedralization

There are three and only three different kinds of tetrahedra between two clouds of points on a pair of paralleled planes: 3 balls on floor and 1 ball on ceiling, 1 balls on floor and 3 balls on ceiling, and 2 balls on floor and 2 balls on ceiling. To compute the Delaunay tetrahedralization, we first create Delaunay triangulations on each plane. Then use these two triangle meshes to create tetrahedralized structure.

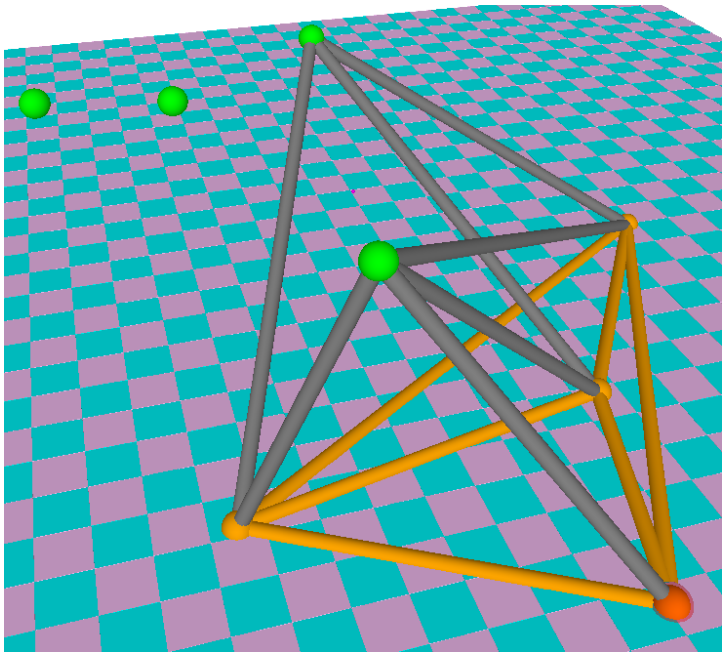
3.1.1.5.1 3 balls on floor and 1 ball on ceiling

Here we use the naïve method. The algorithm is list in Table 2. Time complexity is $O(nm)$.

Table 2 algorithm for 3 balls on floor and 1 ball on ceiling

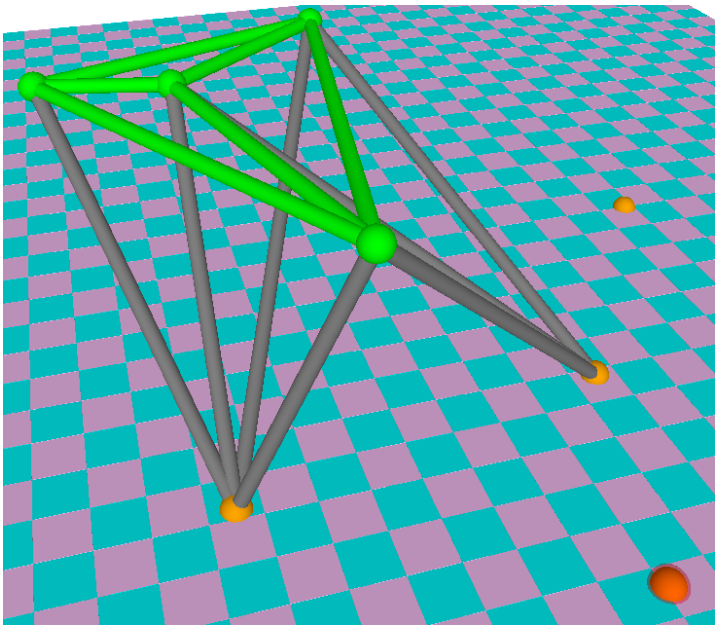
```
add edges from floor triangle mesh to floor edge list
add triangles from floor triangle mesh to floor triangle list
for each triangle t on the floor triangle mesh
  find the point with the minimum bulge to t for each point p in ceiling
  add 3 new edges to mixed edge list (3 edges of the floor triangle are already added)
  add the tetrahedron to floor tetrahedron list
```

The result looks like below.



3.1.1.5.2 1 balls on floor and 3 balls on ceiling

This case is almost the same as the first one. The result looks like below.



3.1.1.5.3 2 balls on floor and 2 balls on ceiling

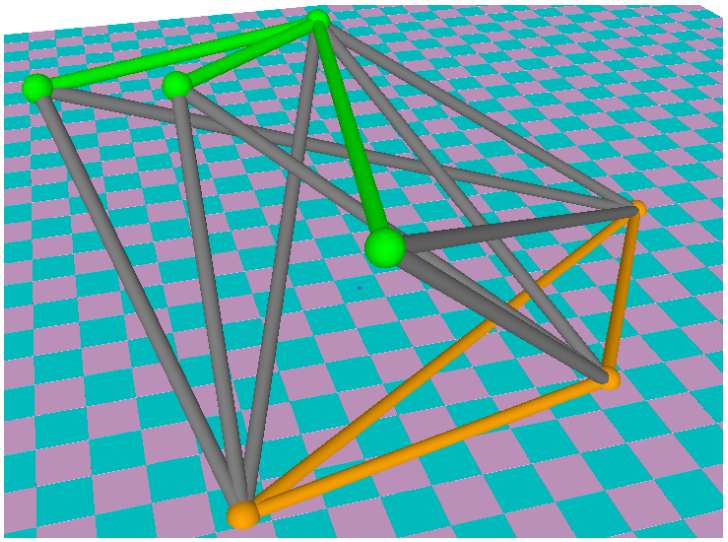
In this case, we could not pick any two points on a plane, because the edge between them may not be a valid edge in Delaunay triangulation on that plane. Thus, we should pick 1 valid edge on floor and 1 valid edge on ceiling. The algorithm is list in Table 3. Time complexity is $O(nm(n+m))$.

Table 3 algorithm for 2 balls on floor and 2 balls on ceiling

```
for each edge ef of all edges in the floor triangle mesh
  for each edge ec of all edges in the ceiling triangle mesh
    calculate the circumsphere s(O, r) from four vertices of ef and ec
```

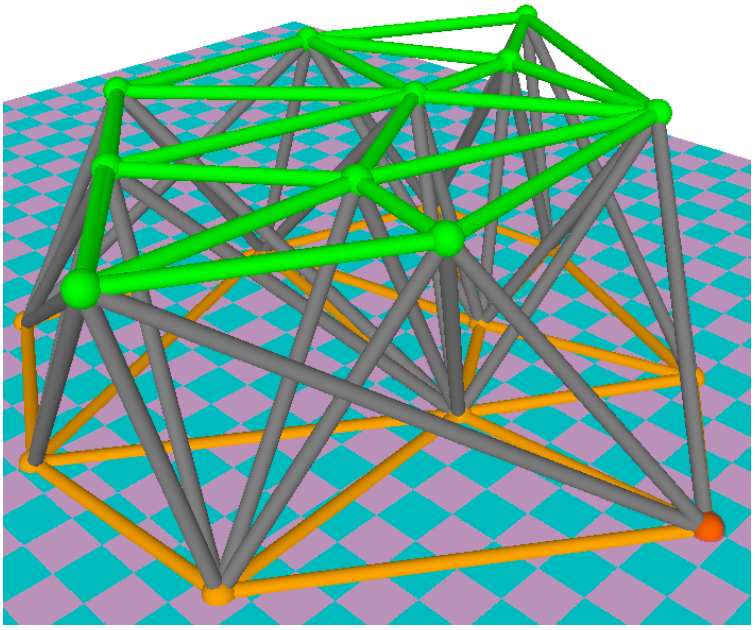
```
    if anyOtherPointInSphere(s, ef, ec)
      continue
    add 4 edges to mixed edge list
    add 1 edge to floor edge list
    add 1 edge to ceiling edge list
    add the tetrahedron to mixed tetrahedron list
    break
```

The result looks like below.



3.1.1.6 Find and remove the duplicated edges

Counting the number of site, tetrahedra and beam on floor and ceiling and display the result on canvas. All the edges are stored as a pair of node IDs where $ID_1<ID_2$, then we just naively loop throw all edges and remove the duplicated ones. The result looks like below.



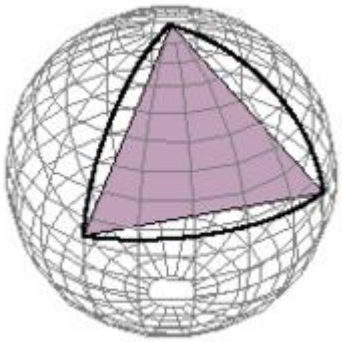
3.1.2 Tetrahedralized structure triangularization

3.1.2.1 Sample vertices from triangle

To uniformly sample vertices from a triangle, we just subdivide each edge by its midpoint and connect all three midpoints to divide a triangle into four smaller triangles. Then repeat the same process for each smaller triangle until the level of details we want.

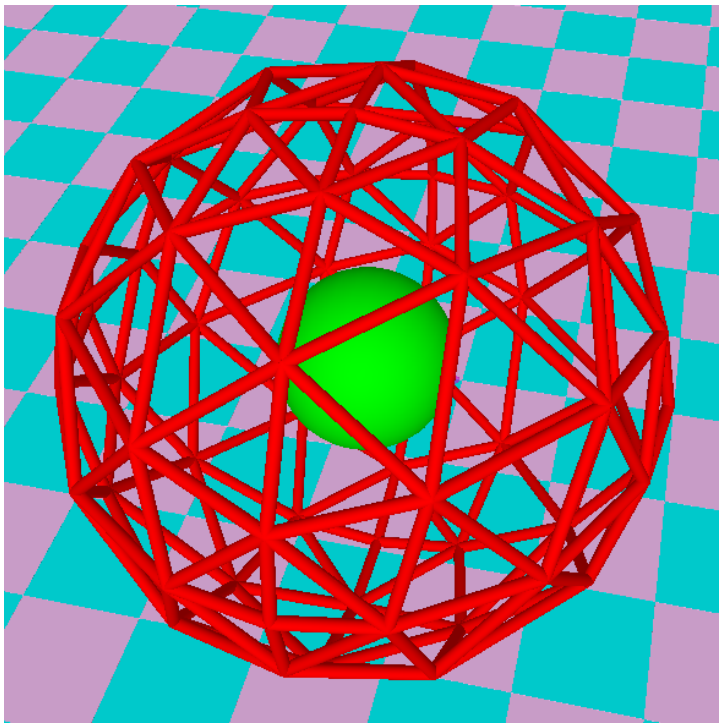
3.1.2.2 Sample vertices from sphere surface

We divide the sphere surface into 8 same shape parts (sphere triangle). For each sphere triangle, we map the sample points on the plane triangle to sphere triangle by pushing points along the radius of the sphere.



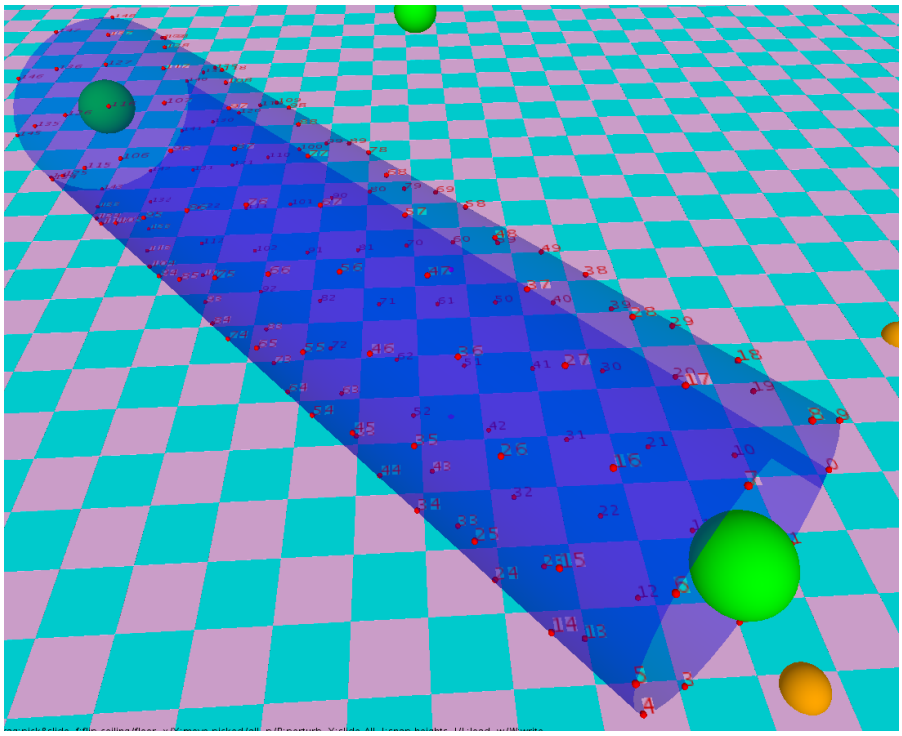
http://mathstat.slu.edu/escher/index.php/File:Flat-under-spherical.png

Then the shape looks like figure below.



3.1.2.3 Sample vertices from right circular cylinder surface

We divide the cylinder surface into several equally spaced layers, where each layer is a circle that perpendicular to the axis of the cylinder with the center on the axis. Then we just uniformly sample points from each layer. See figure below.



3.1.2.4 Use ball pivoting to compute a triangular mesh for all vertices

The ball pivoting algorithm (Digne, 2014) is an algorithm to reconstruct surface from a set of 3D points given their coordinates and normals. There are two main steps in the algorithm, the first is finding a seed triangle, the second is expanding triangulation form the seed triangle. Since we assume to have only one connected component for the mesh, we only need to find one seed triangle.

3.1.2.4.1 Find seed triangle

We use a naïve way to find the seed triangle. We just loop throw all combinations of 3 vertices and check if their normal is point to same half space and if any other vertices are inside the circumsphere of them with a given radius.

3.1.2.4.2 Expand triangulation

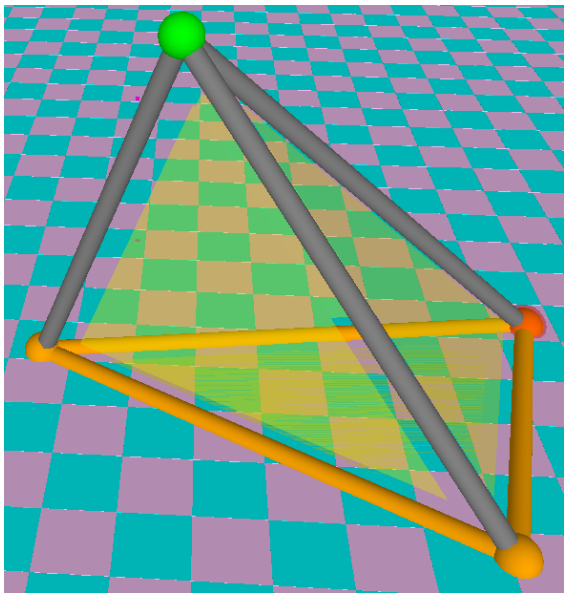
Then we start from the seed triangle and expand the mesh. We maintain a list of front edges that we may expand new triangles from.

4 RESULTS

Some results are shown in the above figures, below are more results. Also see the video for animated views.

4.1 DELAUNAY TETRAHEDRALIZATION

We have implemented the construction of Delaunay Tetrahedralization and make the key selections and auxiliary graphics to visualize and validate the computation. For instance, each face of the tetrahedron constructed by balls on floor and ceiling can be colored as yellow for the easiness of recognition and debugging. See figure below.



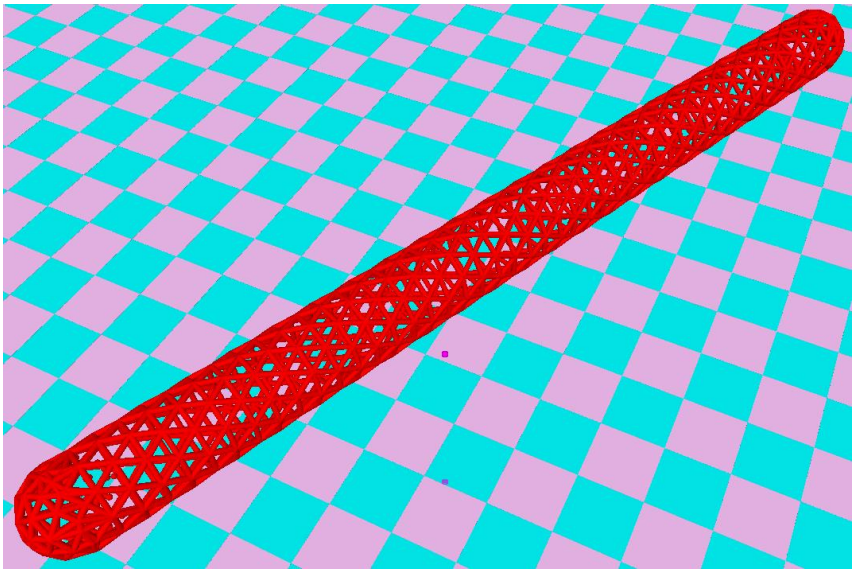
Moreover, the bulge and the center of the triangle and the center of the sphere can be displayed as well.

Additionally, with different shortcuts (**1, 2, 3, 4, 5, 6, 7, 9, 0**), user can choose to either show or hide floor tubes, ceiling tubes and mixed tubes, as well as tetrahedra.

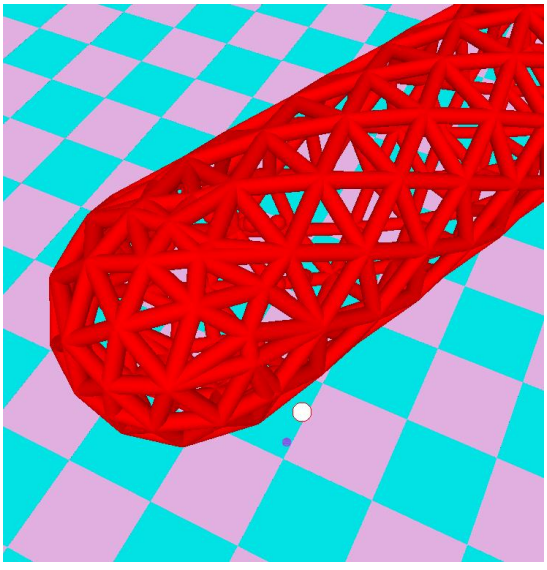
4.2 TETRAHEDRALIZED STRUCTURE TRIANGULARIZATION

When being satisfied with the tetrahedralized structure, user can press '**c**' to calculate the triangular mesh for the structure. It usually takes several seconds to several minutes to get the result depending on the complexity of the structure. During the computation, info messages will be outputted to the console. After the calculation finished, user can press '**j**' to show all sampled vertices, press '**k**' to show the triangular mesh. Notice that the program will become lagged because there are too many tubes (edge) and spheres (vertex) in the scene (usually tens of thousands).

4.2.1 Tube triangulation



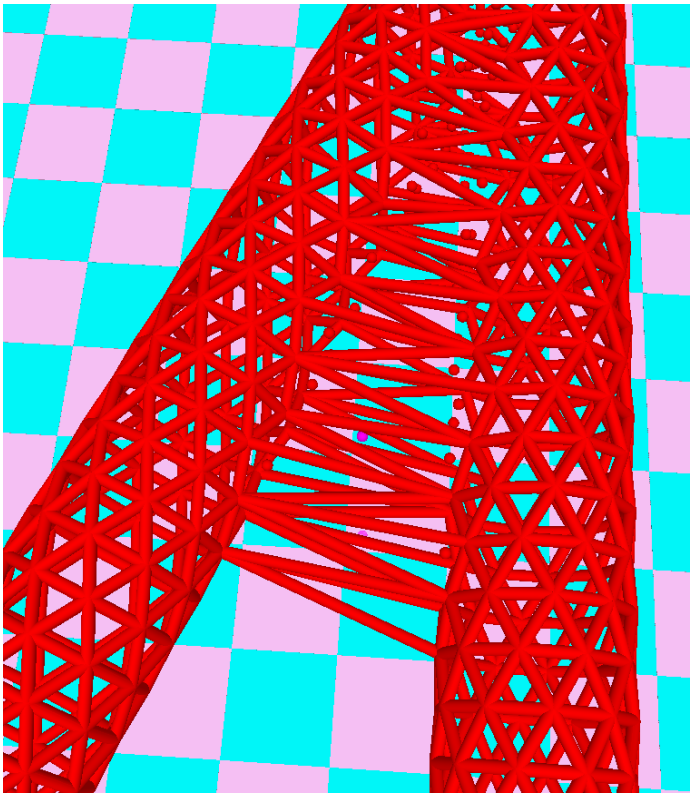
Tube triangulation



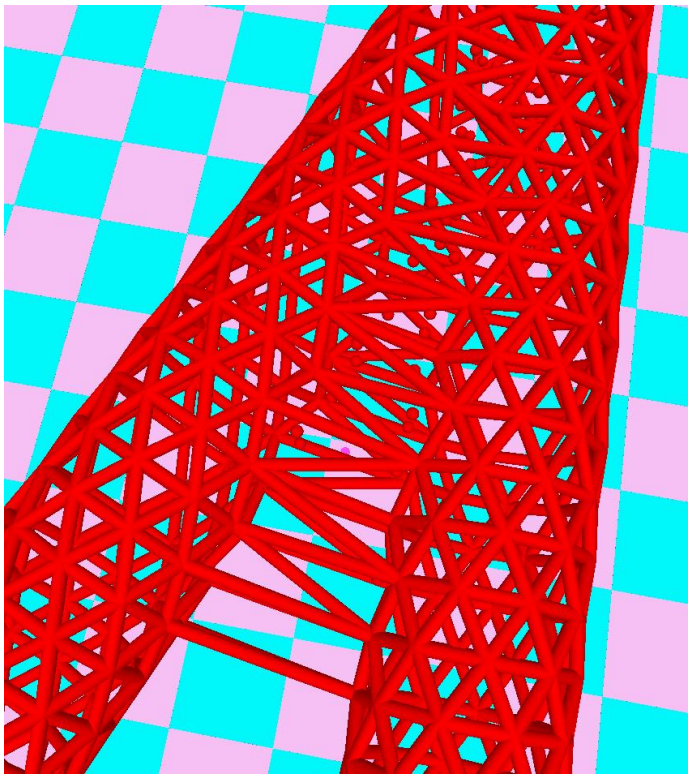
The connection between sphere and tube

4.2.2 Connection between two tubes

As the ball used for pivoting become smaller, the concave part is represented better.



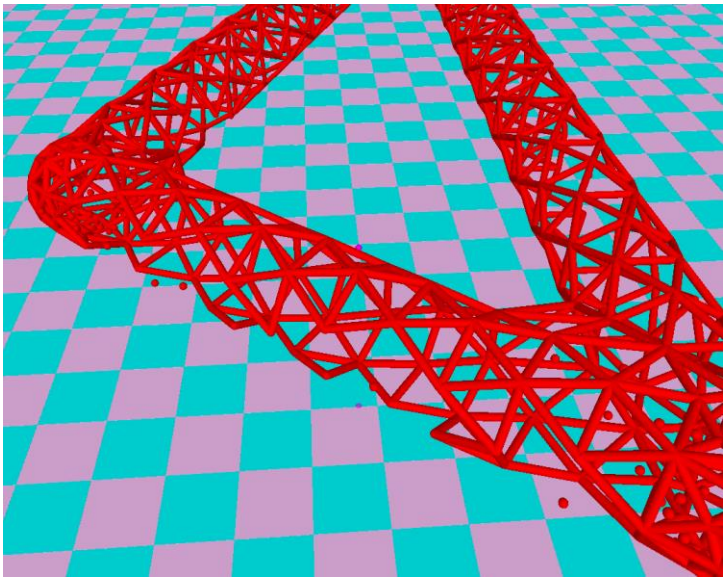
Big ball



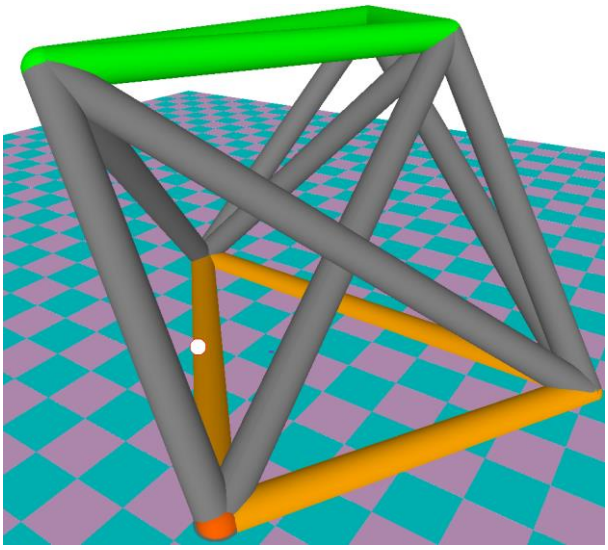
Small ball

4.2.3 Sparse point cloud

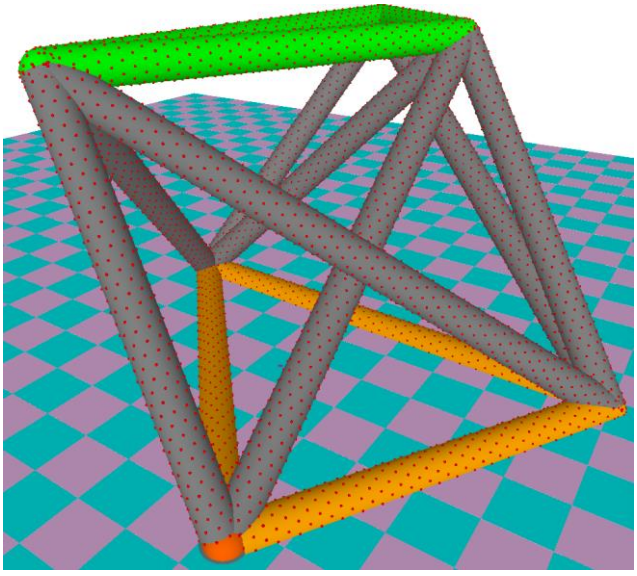
If the ball cloud is too sparse, i.e. not enough sample points, there will be holes on the mesh.



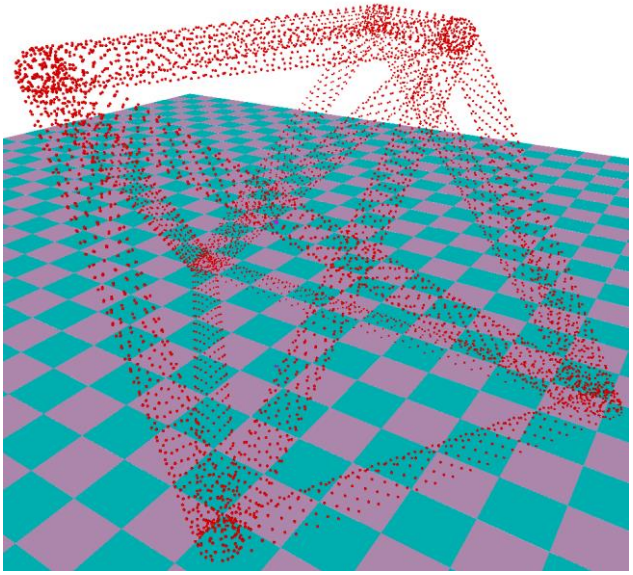
4.2.4 Triangle mesh



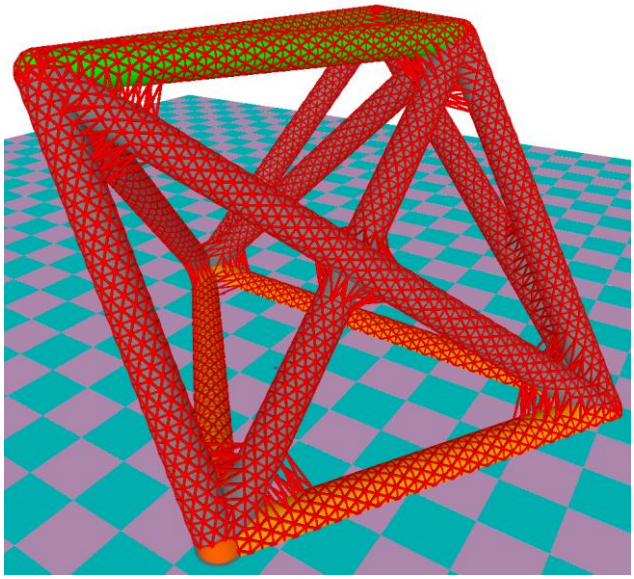
The tetrahedralization structure



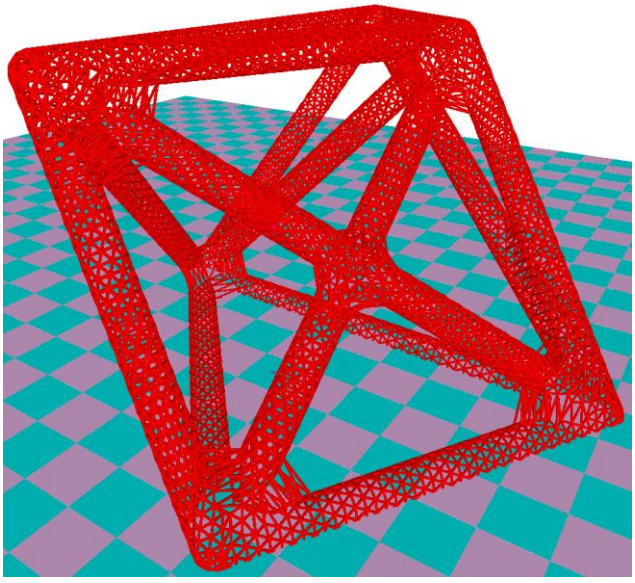
Structure + Sample points



Sample points



Structure + Triangle mesh



Triangle mesh

5 REFERENCE

Circumcenter:

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

http://www.cdsimpson.net/2014/10/barycentric-coordinates.html

Circumsphere:

http://realtimecollisiondetection.net/blog/?p=20

Delaunay triangulations:

https://en.wikipedia.org/wiki/Delaunay_triangulation

Ball Pivoting:

http://www.ipol.im/pub/art/2014/81/article_lr.pdf

https://vgc.poly.edu/~csilva/papers/tvcg99.pdf

Digne, J. (2014). An analysis and implementation of a parallel Ball pivoting algorithm. *Image Processing On Line*, pp. 149-168.