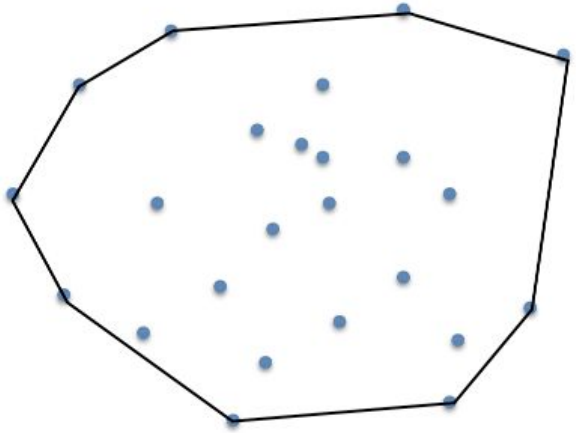
Construct Convex Hull in 3-Dimensional space

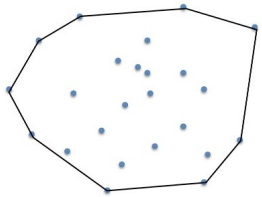
Tianying Tina Zhang
tz1416@nyu.edu

With algorithms and visualization



2D

3D



2D

3D

Project goal

Fully implement different algorithms by myself from blank to construct convex hulls in 3d space and visualize them.

1. Native: Brute Force
2. Incremental Construction:
   Simple Incremental Construction
   Advanced Incremental Construction
3. QuickHull
4. Gift Wrapping

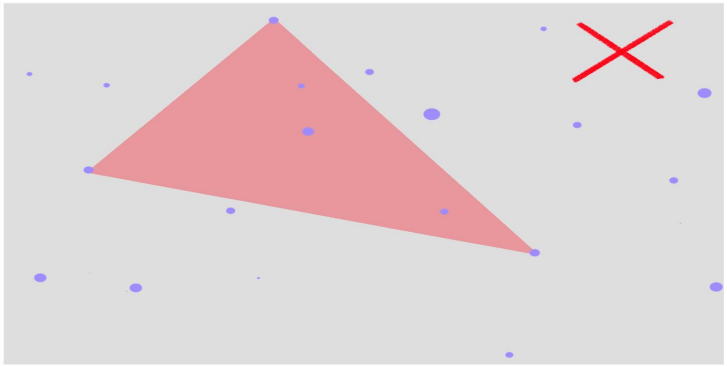
Native: Brute Force

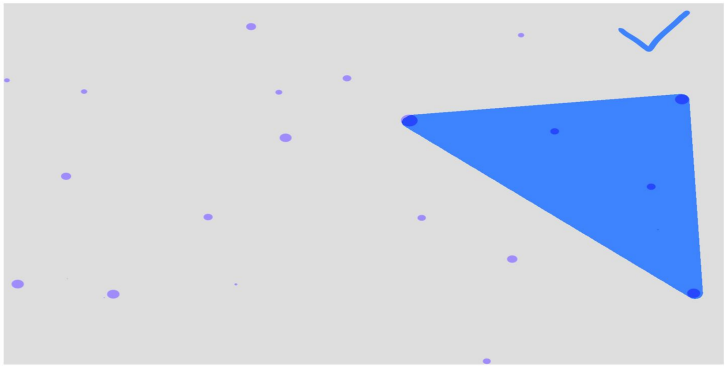
Time Complexity:

O(n^4)

Data Structure:

1. An array to store the points.
2. An array to store the correct convex hull triangles/faces.





Incremental Construction(1)

Construct the initial tetrahedron

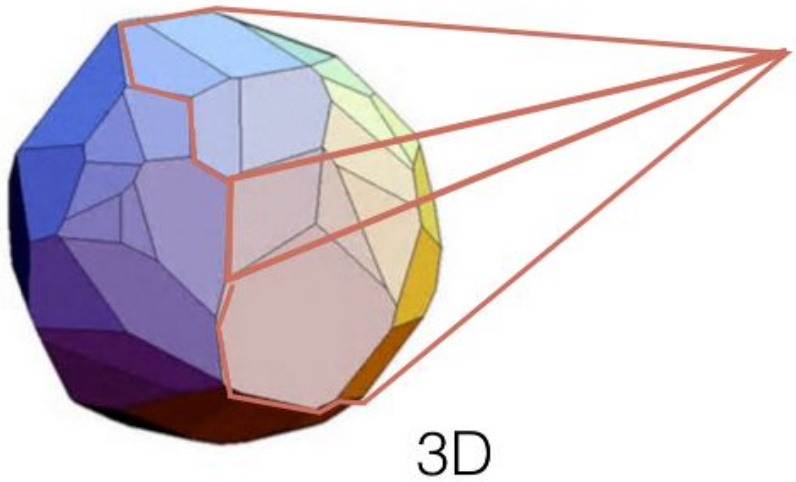
Add each vertex incrementally

Time Complexity:

O(n^2)

Data Structure:

1. An array to store the points.
2. An array to store the correct convex hull triangles/faces.
3. An 2D array to store the edges.



Incremental Construction(2)

Construct the initial tetrahedron

Add each vertex incrementally.

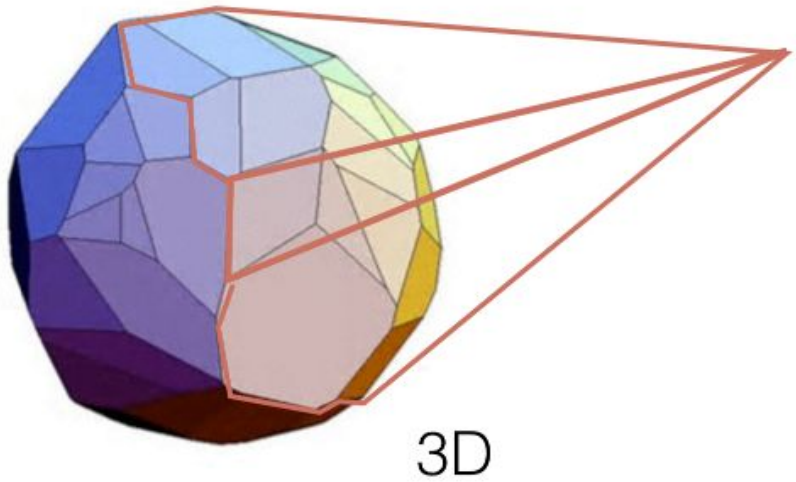
Run DFS to find all of the faces.

Time Complexity:

O(n^2)

Data Structure:

1. An array to store the points.
2. An array to store the correct convex hull triangles/faces.
3. An 2D array to store the edges



Quick Hull(not REAL) but better than before

Construct the initial tetrahedron: construct a face that dividing the points in two halves P1 and P2. And continuously find the farthest points for the two sides of the face.

Add each vertex incrementally.

Run DFS to find all of the faces. This is not a good way to implement this but for small size of points, it still works.

I met problem on merging the polygons so after initialize the first triangle face, I still use the dfs way. In the algorithm side, it seems nothing improved, but in the visualization side, it saves a lot of rendering process.

Time Complexity:

O(n^2) but if a real quickhull can be O(nlogn)

Data Structure:

1. An array to store the points.
2. An array to store the correct convex hull triangles/faces.
3. An 2D array to store the edges

Gift wrapping

Find the first face on the convex hull

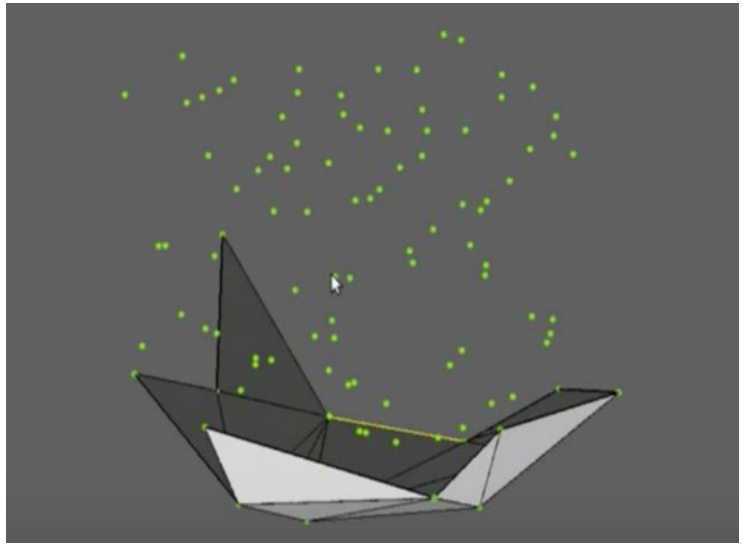
"Wrap" gift

Time Complexity:

O(n^F)

Data Structure:

1. An array to store the points.
2. An array to store the correct convex hull triangles/faces.
3. A queue to store the triangles which needed to be wrapped/not entirely closed.



Citations

Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. Computational Geometry: Algorithms and Applications (3rd ed.). Springer-Verlag TELOS, Santa Clara, CA, USA.

Gregorius, Dirk. Implementing QuickHull, 2014, media.steampowered.com/apps/valve/2014/DirkGregorius_ImplementingQuickHull.pdf

Thank you!