3D Polytope volume computation inspired on convex hull algorithm:
code development and performance evaluation

by Pascal CHEN.

date : 12th July 2019.

This work was developed in the *University of Las Palmas de Gran Canaria, SIANI research institute*, Spain, during an internship stage approved by: *Polytech Sorbonne of the Sorbonne Universite*, France.

Contents

1 ABSTRACT

This work was done in order to solve a simple problem: the problem of polytope volume. Firstly, it is important to know what inspired: the quickhull algorithm. Then the method will be described.

2 INTRODUCTION

2.1 Convex hull algorithm

A convex hull algorithm is an algorithm that construct a convex hull for a set of points.
For more details :
- https://en.wikipedia.org/wiki/Convex_hull_algorithms

2.2 2D Quickhull

Quickhull method is a convex hull algorithm, it uses a divide and conquer approach similar to that of quicksort, from which its name derives. Its average case complexity is considered to be $O(n \cdot \log(n))$.

For more details:
- https://en.wikipedia.org/wiki/Quickhull

2.3 3D Quickhull

In 3 dimensions, it is a bit different, the algorithm follow these steps:

1. Find three points instead of two, for example two points that are extremums and a random point.

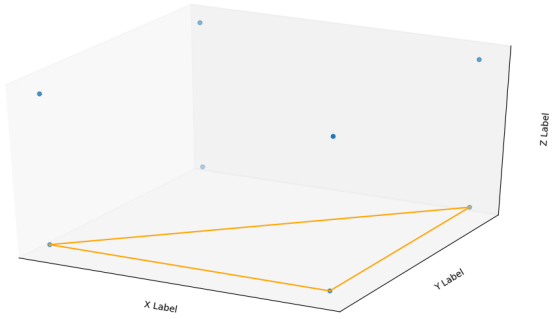

Fig. 1. Find three points.

2. Determine the points and the distances, on one side of the triangle.

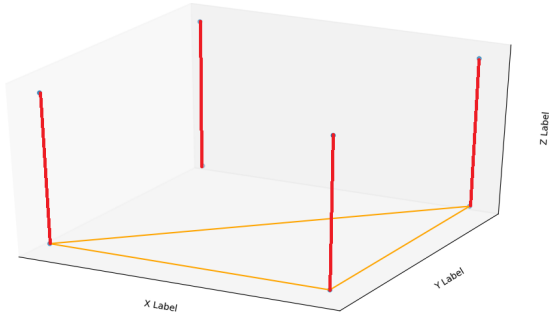

Fig. 2. Determine the distances.

3. Determine the point with the maximum distance from the plane. This point forms a tetrahedron.

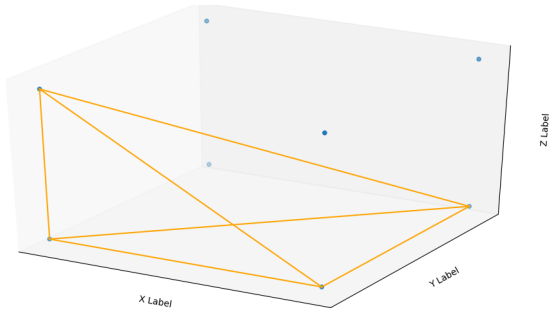

Fig. 3. First tetrahedron.

4. The points lying inside of that tetrahedron cannot be part of the convex hull and can therefore be ignored in the next steps.

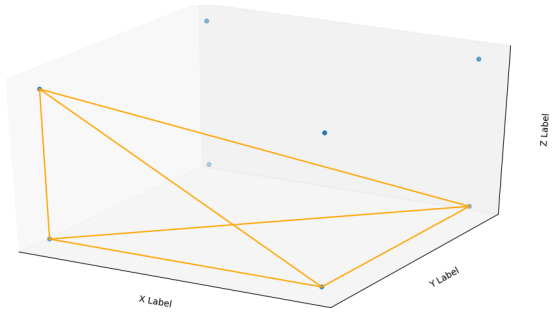

Fig. 4. No point inside.

5. Repeat the previous two steps on the four surfaces formed by the tetrahedron, and don't forget to merge if necessary.

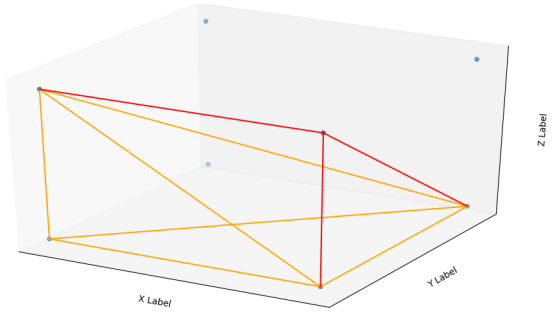

Fig. 5. Expand.

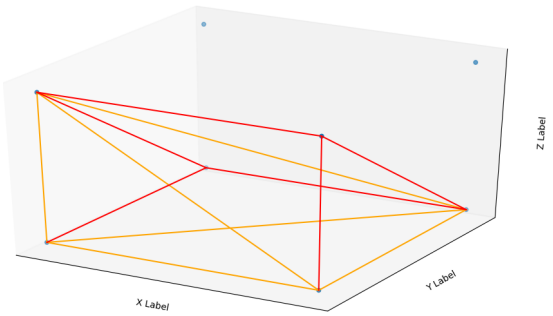

Fig. 6. Expand.

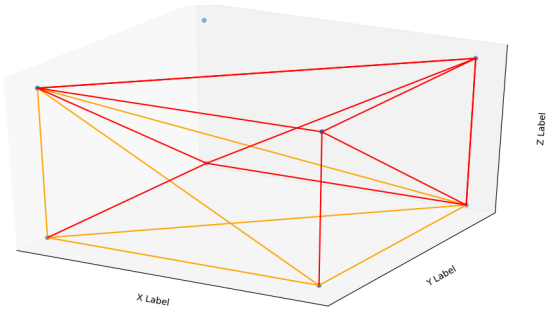

Fig. 7. Expand and merge.

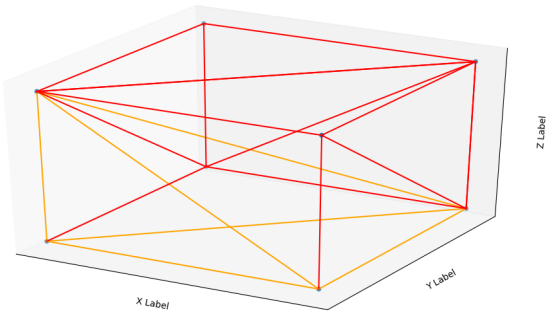

Fig. 8. Expand and merge.

6. Keep on doing so on until no more points are left, the recursion has come to an end and the points selected constitute the convex hull.

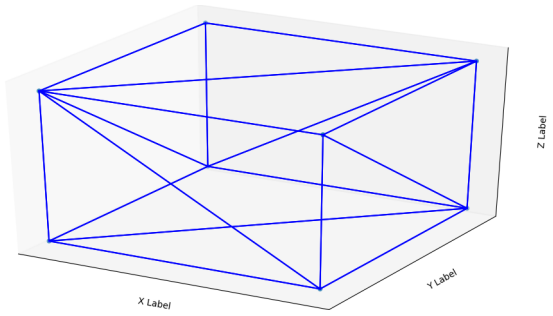

Fig. 9. Finish.

For more details:
- Quickhull pdf

2.4 Quickvolume

Quickvolume is the name I give to this quickhull-inspired algorithm. As explained, at each iteration, there is a tetrahedron that leads to a volume, the idea is simple.

At each iteration, the volume of the new tetrahedron is summed. To get the volume of tetrahedron see:
- https://en.wikipedia.org/wiki/Tetrahedron#General_properties

Finally we get the total volume, the convex hull, and the surface area.
To obtain the points of a plane, obtain the normal then multiply using the dot product with the vector $\vec{AB}$ where the point A is the 4th point of the tetrahedron and B one of the point of the plane, which makes it possible to obtain the good sign.

$$(\vec{n} \cdot \vec{AB}) \times \vec{n}.$$

3 RESULTS

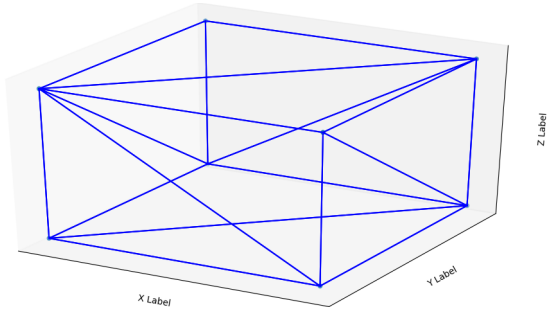

Fig. 10. Volume = 8.0.

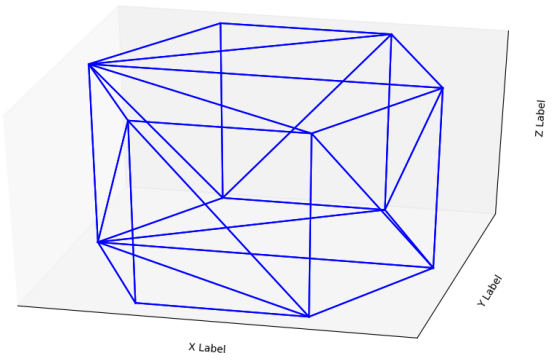

Fig. 11. Volume = 6.0.

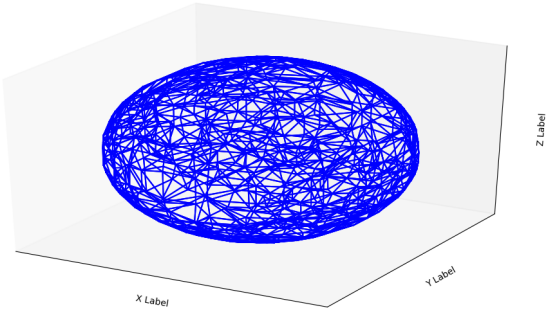

Fig. 12. Volume = [4.092202, 4.123257] for 10 trials and 1000 random points on the sphere.

For three real meshes:

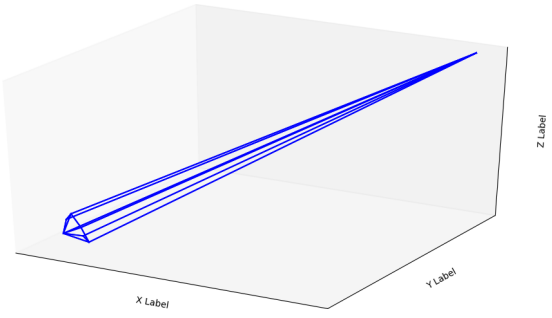

Fig. 13. Volume = 0.059721.

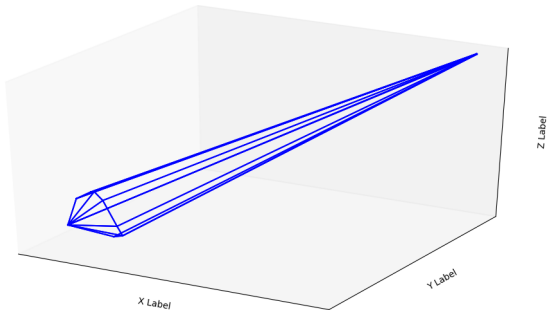

Fig. 14. Volume = 0.192816.

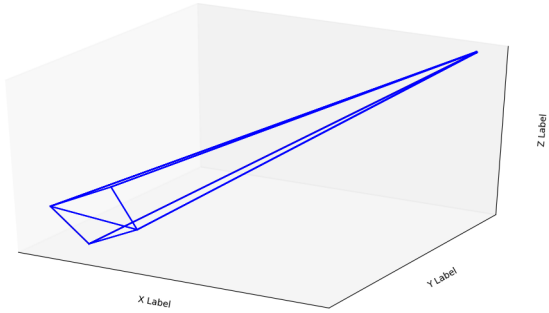

Fig. 15. Volume = 0.204380.

4 HOW TO USE

```
double volume(double * X, double * Y, double * Z, int nb_points);
```

Where :

- `X` is a vector of points of x-axis
- `Y` is a vector of points of y-axis
- `Z` is a vector of points of z-axis
- `nb_points` is the number of points (= size of X, Y, Z)

The output is a volume

5 CONCLUSIONS

In conclusion, this method is very simple and can be used with one line of code instead of trying to get a solution with current libraries for some applications as a data input for "ai" for example.

6 REFERENCES

- https://en.wikipedia.org/wiki/Convex_hull_algorithms
- https://en.wikipedia.org/wiki/Quickhull
- Quickhull pdf
- https://en.wikipedia.org/wiki/Tetrahedron#General_properties