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

by Pascal CHEN.

date : 10th 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. In this document, you will find some explications of the work.

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 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 $O(n \cdot \log(n))$. Under average circumstances the algorithm works quite well, but processing usually becomes slow in cases of high symmetry or points lying on the circumference of a circle. In the planar case, the algorithm can be broken down to the following steps:

In 2 dimensions:
1. Find the points with minimum and maximum x coordinates, as these will always be part of the convex hull.
2. Use the line formed by the two points to divide the set in two subsets of points, which will be processed recursively.
3. Determine the point, on one side of the line, with the maximum distance from the line. This point forms a triangle with those of the line.
4. The points lying inside of that triangle cannot be part of the convex hull and can therefore be ignored in the next steps.
5. Repeat the previous two steps on the two lines formed by the triangle (not the initial line).
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.
For more details:
- https://en.wikipedia.org/wiki/Quickhull
In 3 dimensions, it is a bit different:
1. Find three points instead of two.
2. Use the plane formed by the three points to divide the set in two subset of points.
3. Determine the point, on one side of the triangle, with the maximum distance from the plane. This point forms a tetrahedron.
4. It is the same, but inside the tetrahedron.
5. Repeat the previous two steps on the four surfaces formed by the tetrahedron, and don't forget to merge.
6. It is the same.
For more details:
- http://media.steampowered.com/apps/valve/2014/DirkGregorius_ImplementingQuickHu pdf

2.3 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. The input is a set of points and the output is a volume.

3 ALGORITHM EXPLAINED

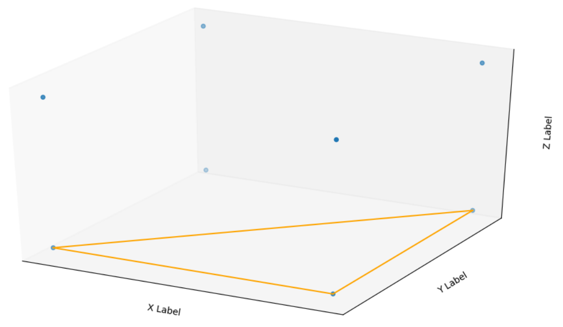

Fig. 1. Find three points.

Step 1: Find three points.

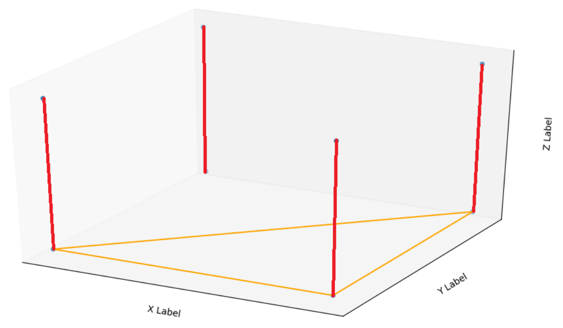

Fig. 2. Determine the distance.

Step 2: Determine the distance of each point of the polytope.
Step 3: Determine the point with the maximum distance from the plane.

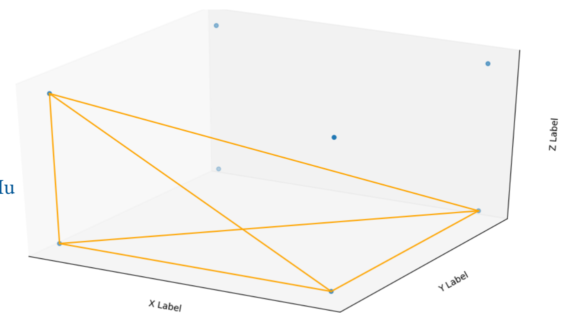

Fig. 3. First tetrahedron.

Step 4: Construct the first tetrahedron and sum the volume created.

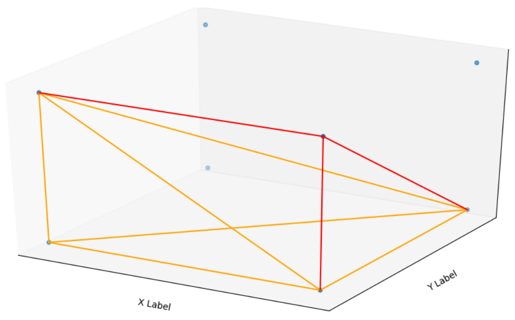

Fig. 4. Expand.

Step 5: Repeat the previous steps and merge if necessary.

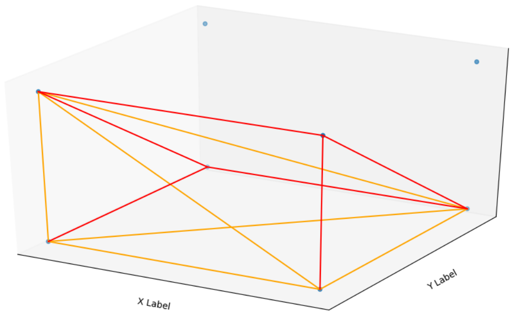

Fig. 5. Expand.

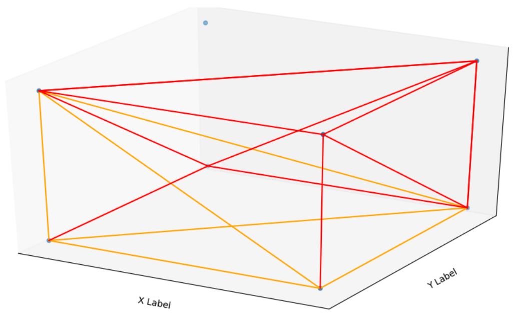

Fig. 6. Expand and merge.

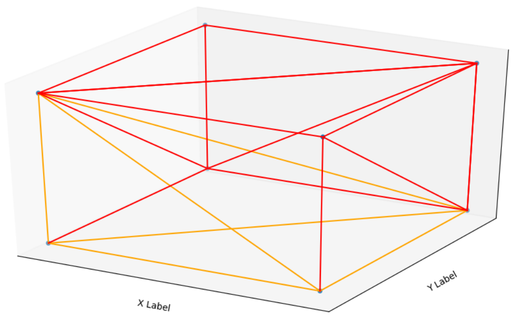

Fig. 7. Expand and merge.

Step 6: Stop when there is no point outside of the current convex hull.

At each iteration, we sum the volume of the new tetrahedron.
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.

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

4 RESULTS

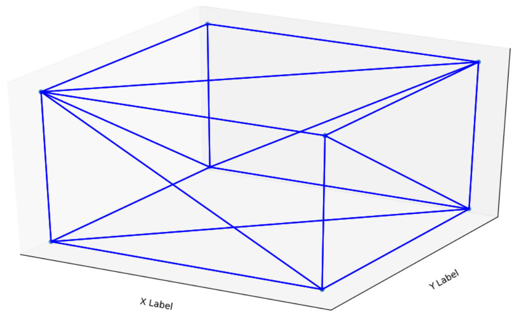

Fig. 8. Volume = 8.0.

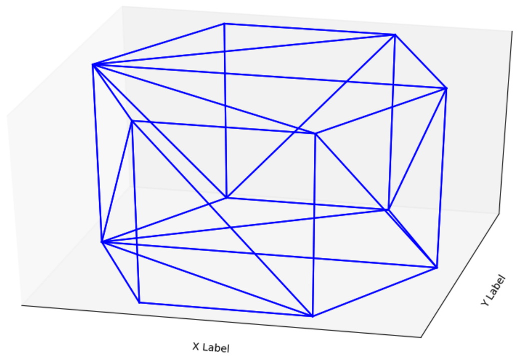

Fig. 9. Volume = 6.0.

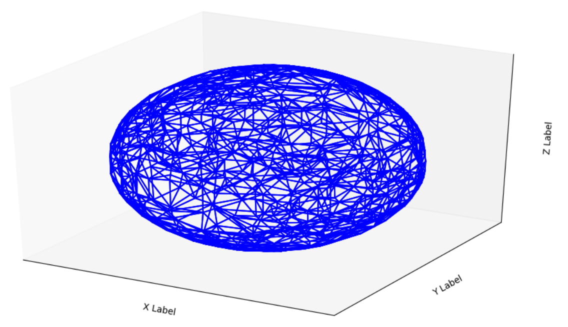

Fig. 10. Volume = [4.092202, 4.123257] for 10 trials and 1000 random points on the sphere.

For three real meshes:

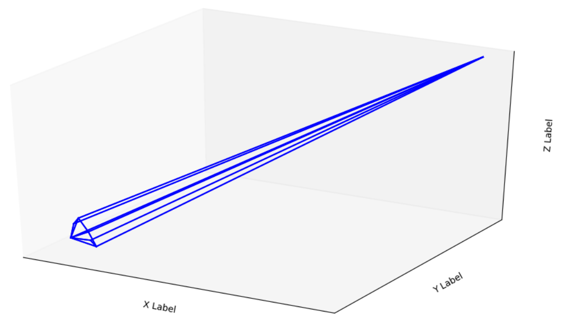

Fig. 11. Volume = 0.059721.

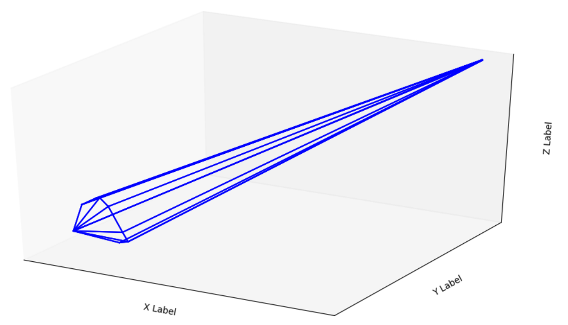

Fig. 12. Volume = 0.192816.

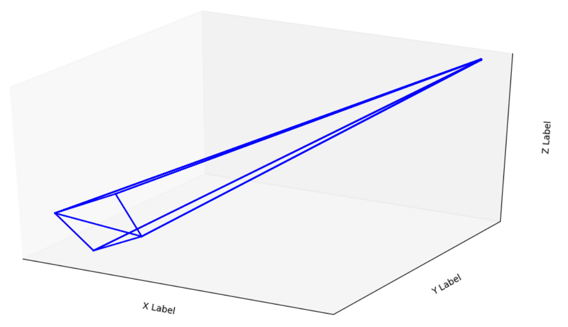

Fig. 13. Volume = 0.204380.

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
- http://media.steampowered.com/apps/valve/2014/DirkGregorius_ImplementingQuickHull.pdf - https://en.wikipedia.org/wiki/Tetrahedron#General_properties