3-D Convex Hull

Milestone2 Project Report

Zhicheng Wang, Zhaowei Xu, Sujun Zhu
University of Southern California, Viterbi School
Los Angeles, USA
zhichenw@usc.edu, zhaoweix@usc.edu, sujunzhu@usc.edu

***Abstract*—We give a simple interpretation and a simple implementation of the classical divide-and-conquer algorithm for computing 3-d convex hulls (and in particular, 2-d Delaunay triangulations and Voronoi diagrams).**

Keywords—convex hull; divide-and-conquer; quick hull; deleting points; inserting points

I. Introduction and Motivation

Problem: Finding 3D convex hull, the smallest 3D surfaces that wrap all the given point in 3D space, with hardware implemented by Verilog on FPGA board.
Motivation: Building stronger skills and experience in Verilog coding and problem solving, possibly finding better solution to 3D convex hull, and understanding the graphics dealing.
Main Challenges:
-Solving 2-D convex hull problem and expanding situation into 3-D convex hull problem, and we have two algorithms: 1) Divide and Conquer; 2) Quick Hull.
-The challenge of Divide and Conquer Algorithm is to solve merging cases: we divided points into two parts recursively, and for two adjacent base parts, we need to merge them by deleting and/or remaining certain points.
-The challenge of Quick Hull is to find the certain facet: we need to find three certain points to make the initial facet and find the most remote points and making new facets.
-Visualizing the 3D solution: representing depth and shapes on multiple view-angle or generating 3D graph

II. Previous Work

A. Other Algorithms

"A Minimalist's Implementation of the 3-d Divide-and-Conquer Convex Hull Algorithm", by Timothy M. Chan

- The Gift-wrapping Algorithm [1] computes the convex hull in O(n^2) time by generating facets one at a time via an implicit breadth- or depth-first search.
- The Incremental Methods [2] maintain the convex hull as points are inserted one at a time.

B. Divide and Conquer

The Divide and Conquer method [3] [4] is earliest to achieve O(n log n) running time. Divide the point set into two hulls recursively and merge two adjacent hulls by using "bridge"[5].

C. Quick Hull

The Quick Hull [6] divides point set to two hulls by finding a certain facet and then find the most remote point to form new facets and find points recursively.

III. Approach and Concrete Experiments

A. The Problem:

We have various methods to compute convex hull in 2-D, but in 3-D, some methods are not efficient and even do not work. We want to use the most efficient one if possible.

B. The Possible Approaches and Challenges:

- The Divide and Conquer method: firstly project the 3D point set onto a 2D image by keeping points' x coordinate, changing y coordinate to z-t*y and deleting z coordinate. Then the problem changed into finding 2D convex hull for a set of 2D images where z-t*y vary from y to z; recursively divide the 2D point set into two parts based on their x value until every part only has 1 point; then by returning, every level except the lowest one has two complete convex hulls to merge; we build two "bridges" and move it until their curvature indicate that they are at the boundaries of the resulting convex hull. The two bridges and part of the original points from two input convex hulls not wrapped by the bridges can form the resulting convex hull.

  Challenges: the way to implement the method of turning 3D convex hull into 2D images is still questionable in our point of view.

- The 3D Quick Hull method is similar to the 2D one, in order to divide two hulls of points, we need to determine a facet at first to divide two hulls of points, and then in either space, we find the most remote point and form new facets; then we find points and facets recursively.

  Challenges: Quick Hull finds the shell's points randomly and determining what three points to form a surface is challenging.

C. *The Explanation of Approaches and Assumption:*

- For Divide and Conquer method, we are putting our bet on it since according to Chan's paper, it is proven to be the quickest 3D convex hull algorithm.

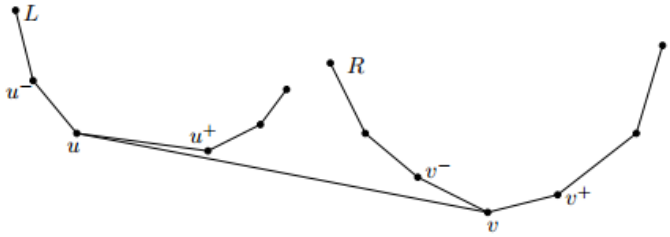

Figure1. Merging two 2-D lower hulls.

In the merging process, we connect rightmost point of L and leftmost point of R to form an initial bridge. Then we adjust this bridge's end points, u and v, by checking if u˙-u-v is counter-clockwise. This check can be done by examining if (u.x – u˙.x)*(u˙.y – v.y) – (u˙.y – u.y)*(v.x - u˙.x) is negative. If (u,v) is counter-clockwise then (u,v) is the outside bridge we are trying to find.

- For Quick Hull method, we can sort the points and find the closest and farthest points, and we just need one more point which is one of the convex points. Thus we can sort the points again but sort the y-axis value; then we can pick the farthest point on y-axis. These three points are definitely convex points, therefore we can use these three points to form a facet and find farthest point in space divided by the facet. Then we can form new facets and find the most remote points recursively.

D. *Stateflow chart of 2D-QuickHull:*

The following is an attempted QuickHull implementation by Stateflow. It takes a matrix of points and output as a matrix of edges (Figure3), 0 as none while 1 as an edge between those two points. Figure 2 is a more detailed algorithm. Two of the

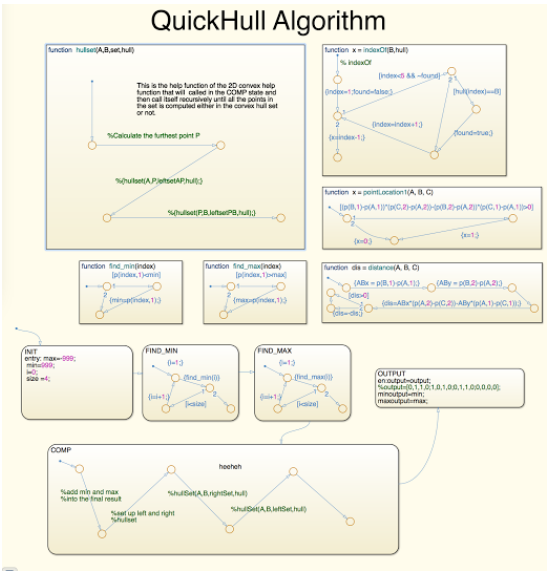

Figure2. Attempted 2D-QuickHull Algorithm flow chart1

main difficulties remain to be solved (with possible solution here): 1) the recursion transformation to iteration through buffer set-up and good recursion level prediction; 2) memory allocation and data structure set-up in Stateflow, where we need to find the best way to store and use the data.

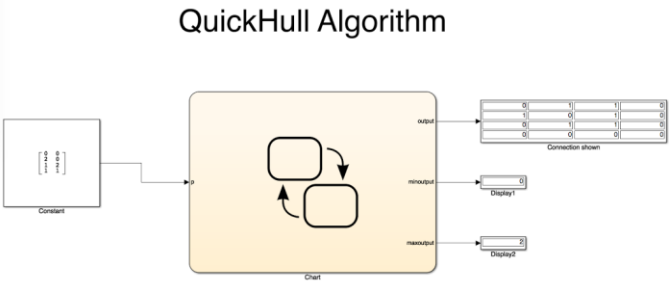

Figure3. QuickHull Algorithm flow chart 2

IV. Conclusion and Short-term Plan

A. *Conclusion:*

1) The main idea and findings:

To solve 3-D convex hull problem, we have two major algorithms, the Divide and Conquer method and the Quick Hull method. They have their own benefits and challenges.

2) The findings:

The Divide and Conquer method is the most efficient and suitable to the 3-D convex hull problem. We can use the parallelization when we take multiple projections at the same time and combine the results. But the conversion from 3D model to 2D images is difficult to implement.

The Quick Hull method is relatively easier to understand, but its running time is from O(nlog(n)) to O(n^2), which is a trade-off. We can still use the parallelization when dealing with the most remote point of each sides.

B. *Short-term Plan and Milestone3 topic:*

We will figure out which algorithm is suitable as soon as possible. Thus we will try to figure out the complexity in terms of hardware between the Divide and Conquer method and the Quick Hull method, and finally choose one.

In next presentation, we will determine our algorithm and conclude the reason and the progress we get on this algorithm, such as its sanity and use of the parallelization.

C. *Anticipated outcome:*

Since the time (t) variable in the D&C is really hard to manipulate, we may choose the Quick Hull and record the progress of using the Quick Hull.

References

[1] D. R. Chand and S. S. Kapur. An algorithm for convex polytopes. J. ACM, 17:78–86, 1970.

[2] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. Discrete Comput. Geom., 4:387–421, 1989.

[3] M. I. Shamos and D. Hoey. Closest-point problems. In Proc. 16th IEEE Sympos. Found. Comput. Sci., pages 151–162, 1975.

[4] F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. Commun. ACM, 20:87–93, 1977.

[5] Timothy M. Chan.A Minimalist's Implementation of the 3-d Divide-and-Conquer Convex Hull Algorithm.2003