ECE465 Final Project Proposal

Jonathan Lam & Henry Son

March 1, 2021

Motivation for changing our algorithm

Over the course of the first project, we realized that our previous “distributed” algorithm (graph coloring) induced too high of a communication cost and was not very amenable to speedup over networked nodes. As a result, we decided to change our focus to a problem that is more cleanly divisible into parts (i.e., one with low communication costs) for sub-processing between nodes.

What is convex hull?

Given a set of points P, the convex hull of P is the smallest convex shape that contains all the points within P. It has numerous uses in geometric modeling, statistics, and etc. Practical use cases include collision avoidance, smallest bounding box, and shape analysis.

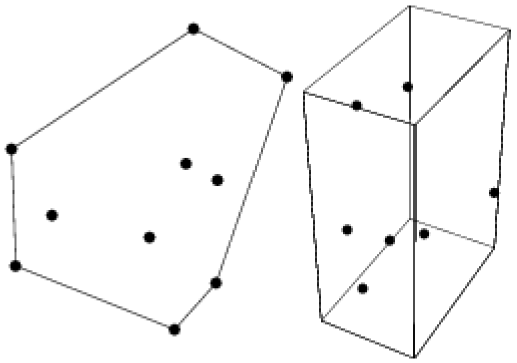

Figure 1: 2D, 3D examples of Convex Hulls. As you can see from the diagram, the pointset is contained within the convex hull, denoted as the area/volume enclosed by the solid lines.

Quickhull: an algorithm for finding the convex hull

Quickhull is a divide-and-conquer algorithm and can be used for arbitrary N-dimensional space. A good explanation, as well as pseudocode for the 2-D case, can be found on its Wikipedia page. The input pointset will first be divided into non-overlapping regions to be distributed among compute nodes, which will each perform Quickhull independently. These disjoint convex hulls will then be joined on the coordinator node.

What it'll look like

The final project might look like an API that has endpoints to receive 2-D or 3-D models (or arbitrary N-D models) and calculate their convex hulls. For example, a user could input a file generated by a 3-D CAD program and receive a "gift-wrapped" version of it (its convex hull). Time permitting, there may also be additional endpoints related to convex hulls, such as calculating diameter of a convex hull, finding the union or intersection of convex hulls, distance between convex hulls, etc.

Infrastructure, environment, and tooling

The infrastructure would be an AWS Lambda[1] compute to reduce costs and easily scale. The algorithm will again be written in Go and use the default Go build tools.

[1] Neither of us have used Lambda before nor know exactly how it works, so it may end up being on regular EC2 instances.