CS 274 Computational Geometry
Final Project Report

Ankush Gupta
gupta@berkeley.edu

The task assigned was to implement an algorithm for computing delaunay triangulations of two dimensional points.

I implemented the *divide-and-conquer* algorithm from Stolfi and Guibas. The correctness of the algorithm was validated by visually comparing its output with that of `Triangle`[1] on the same data set.

Compiling/Running the Code

This code was tested on a machine running Ubuntu 12.04. The only external libraries it needs are the `boost` libraries. The source contains `tuxfamily Eigen`[2] used internally to represent points. The code can be compiled by making a `build` directory and then running cmake, followed by make, as following:

```
cd build
cmake ${PATH TO PROJECT DIRECTORY}
make delaunay
```

The code is executed as following (assuming we are in the build directory):

```
./bin/delaunay -i input_filename [-o output_filename] [-V or -A][-T]
```

For an explanation of the various flags, run `./bin/delaunay -h`.

Timing

The implementation works correctly on small data-sets. Unfortunately, it did not terminate on the data-set `ttimeu100000.node` (100k points), even after 30 minutes. It does not get stuck in any infinte-loop as it is continually hitting the base-cases. The implementation was

[1] Jonathan Richard Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator

[2] http://eigen.tuxfamily.org/index.php

not done with an eye towards efficiency, yet it is surprsing that it takes so long.

Following are the times it took on the `ttimeu10000.node` (10k points) data set; it does not include the file i/o time.

	Alternating Cuts	Vertical Cuts
10k points	17.62 seconds	31.14 seconds

As my implementation did not terminate within 30 minutes on the 100k data-points input, I did not try it on the 1 million points data-set. However, its output on the `spiral.node` data is attached below. More pictures are present in the `data` directory.

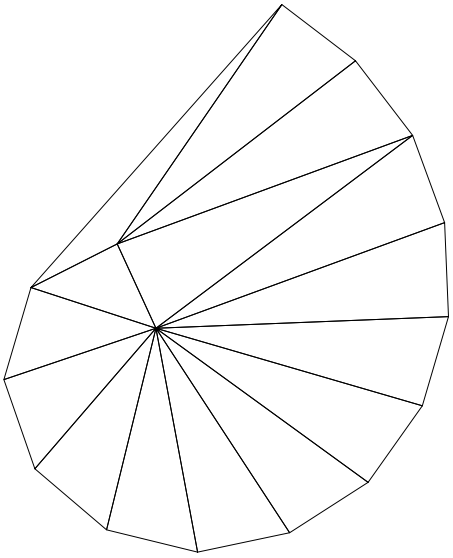

Figure 1: Output of the program on `spiral.node` data-set. Generated using `ShowMe`.

Alternating Vs. Vertical Cuts

If we have a data-set which has a large horizontal spread and very-small vertical spread, then it would take alternating cuts algorithm significantly longer than vertical cuts only algorithm. This is because then, it will encounter almost collinear points, which would mean that the geometric predicates need to calculate more bits to correctly evaluate the sign of the determinants.
