A Robust Parallel Triangulation and Shelling Algorithm

Isabel Beichl and Francis Sullivan
Center for Computing and Applied Mathematics
National Institute of Standards and Technology

Abstract

This paper describes an efficient parallel algorithm for finding the Delaunay triangulation in shelling order [BM], [DK] for the convex hull of n vertices in m dimensional Euclidean space. A related purely sequential triangulation and shelling algorithm is described by Seidel in [S]. However, in [S] it is assumed that the initial set of vertices is non-degenerate, that is, every subset of $m+1$ vertices spans $\mathbf{R}^m$, and the algorithm relies on solving a linear program with $n-1$ constraints and $m-1$ variables to determine the next simplex in the shelling sequence. A probabilistic method for overcoming the non-degeneracy assumption is described in the paper. Instead of linear programming, we use Householder's QR decomposition of an $(m+1) \times m$ matrix for determining new simplices in the course of the triangulation. The QR decomposition [St] is known for its numerical stability.

The sequential version of our algorithm actually makes use of the shelling to reduce the amount of calculation. We describe how to parallelize our algorithm for the SIMD architecture of the Connection Machine. As always, the major issues in designing an efficient parallel algorithm are those of data motion. The datastructure that is used to check legality of the shelling turns out to be exactly the right basis for an efficient parallel algorithm. In all dimensions, the parallel complexity is $O(M(n)\Gamma(n))$, where $M(n)$ is the complexity for determining the min or max of n numbers and $\Gamma(n)$ is the number of simplices generated.

We are interested in applications of computational geometry to large-scale physical problems. This is what motivated our search for a method that is efficient, and robust in both its theoretical foundation and its numerical implementation. Before launching into an outline of the algorithm, we sketch one such application. In using molecular dynamics simulations of nucleation for detecting development of crystal-like cluster structure in $\mathbf{R}^3$ we need to calculate the distances among particles in the local neighborhood of a given particle (see Yang, Gould, Klein, and Mountain [YGKM] for a more detailed discussion). Our triangulation and shelling algorithm offers an efficient tool for attacking this problem. The idea is this: We use the coordinates of the positions of the particles as input vertices in the triangulation and shelling algorithm. Because of the shelling, we can easily identify the Delaunay tetrahedra containing a given vertex. The vertices of these tetrahedra are the nearest neighbors to the given vertex. The number of tetrahedra containing a given vertex is the same as the number of vertices in the Voronoi cell associated with that vertex because of the duality between Voronoi diagrams and the Delaunay triangulation. In Figure 1 we give a histogram of distances among nearest neighbor for points in a face-centered cubic lattice. This corresponds to some of the results described in [YGKM]. Figure 2 is a histogram of average number of tetrahedra surrounding a point for (a) the case of 1000 random points chosen uniformly in the unit cube and (b) a partial BCC lattice. Notice that the random curve is somewhat Gaussian with peak at the number of vertices for Voronoi cells in an infinite body-centered cubic lattice. Only even numbers occur because of Euler's formula.

Figure 1
Histogram of distances among nearest neighbors for FCC data.

Figure 2
Average count of surrounding Delaunay tetrahedra for (a) random data and (b) BCC data.

Here is an outline of our algorithm: To keep the basic geometric ideas clear, we describe the algorithm for the case in which the set of initial vertices lies in $\mathbf{R}^2$. However, the extension to higher dimensions is sketched. Place the origin at the centroid of the set of input points and then map the points to the surface of the paraboloid in $\mathbf{R}^3$. Assume that the all facets of the convex hull of the mapped points visible from below the paraboloid are triangles (i.e. the initial points are non degenerate). When projected back to $\mathbf{R}^2$, these facets are the Delaunay triangulation of the convex hull of the original data. Hence, we wish to identify the visible facets, which we will call the base triangles of a cap. A starting base triangle is found using gift-wrapping. The QR decomposition gives us a numerically reliable way to move from triangle to triangle, and the shelling concept defines an order for our moves so that all facets are found and enumerated in a consistent way.

Acknowledgements
We thank Chris Witzgall for reminding us that the triangulation of new vertices on the paraboloid gives the Delaunay triangulation of the initial vertices, and we thank Pete Stewart for proposing the use of the QR decomposition for computing rotations and Dianne O'Leary for proposing an implemetation of QR appropriate to the Connection Machine. We are grateful to Richard Cushman for first suggesting that our idea for a triangulation algorithm could, in fact, give a shelling, and we thank Jim Lawrence for helping us to understand this concept.

References

BM Bruggesser, H. and Mani P. 1971, "Shellable Decompositions of Cells and Spheres", **Math. Scand.** 29.

DK Danaraj, J. and Klee, V. 1978. "A representation of two-dimensional pseudo-manifolds and its use in the design of a linear time shelling algorithm." **Annals of Discrete Mathematics.** 2.

S Seidel, R. 1986 "Constructiong Higher Dimensional Convex Hulls at Logarithmic Cost per Face", **Proc. ACM Symp. on Theory of Computing.**

St Stewart, G. W., 1973 **Introduction to Matrix Computations** Academic Press, New York.

YGKM Yang, J., Gould, H., Klein, W., and Mountain, R. "Molecular Dynamics Investigation of Deeply Quenched Liquids", to appear.

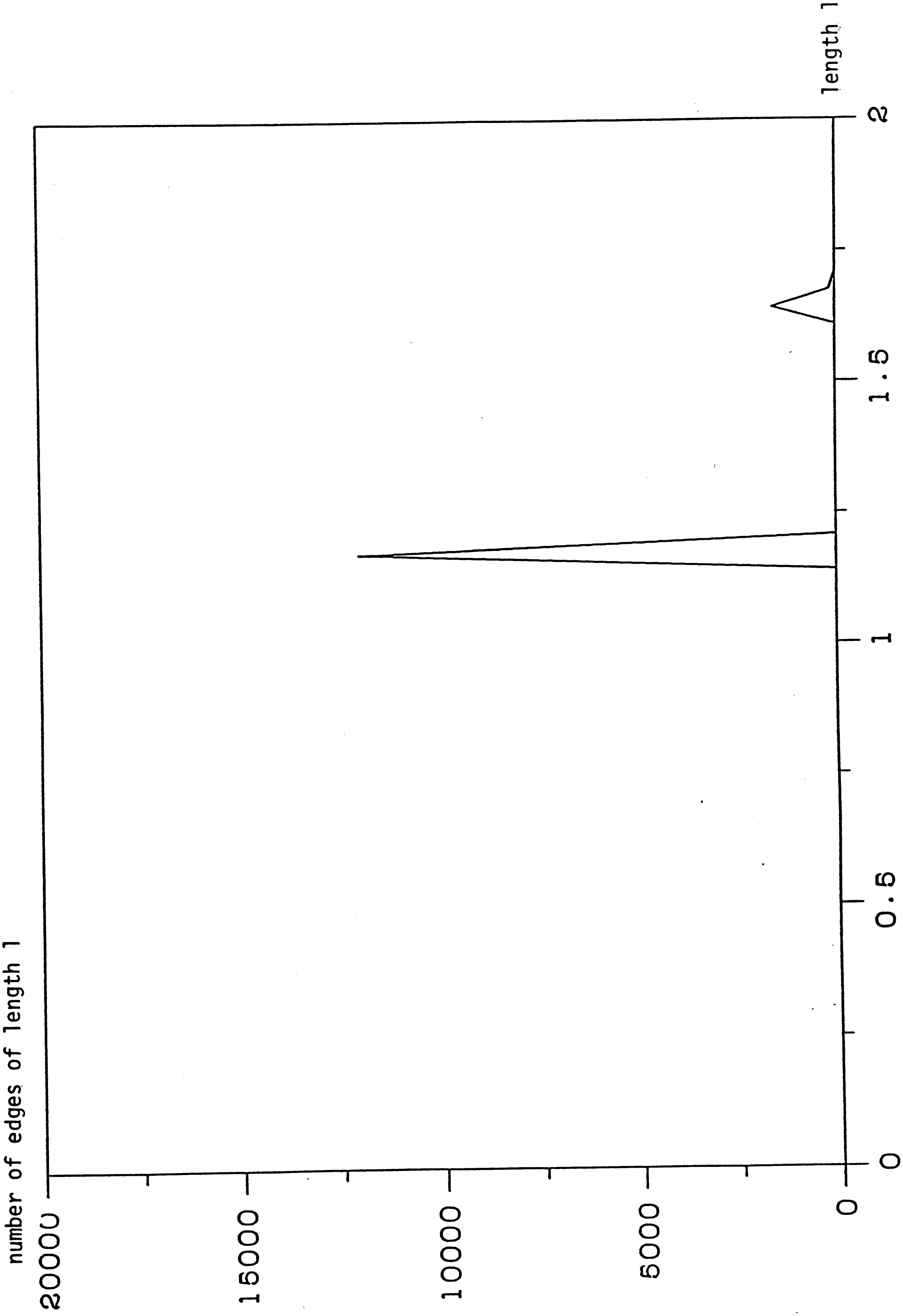
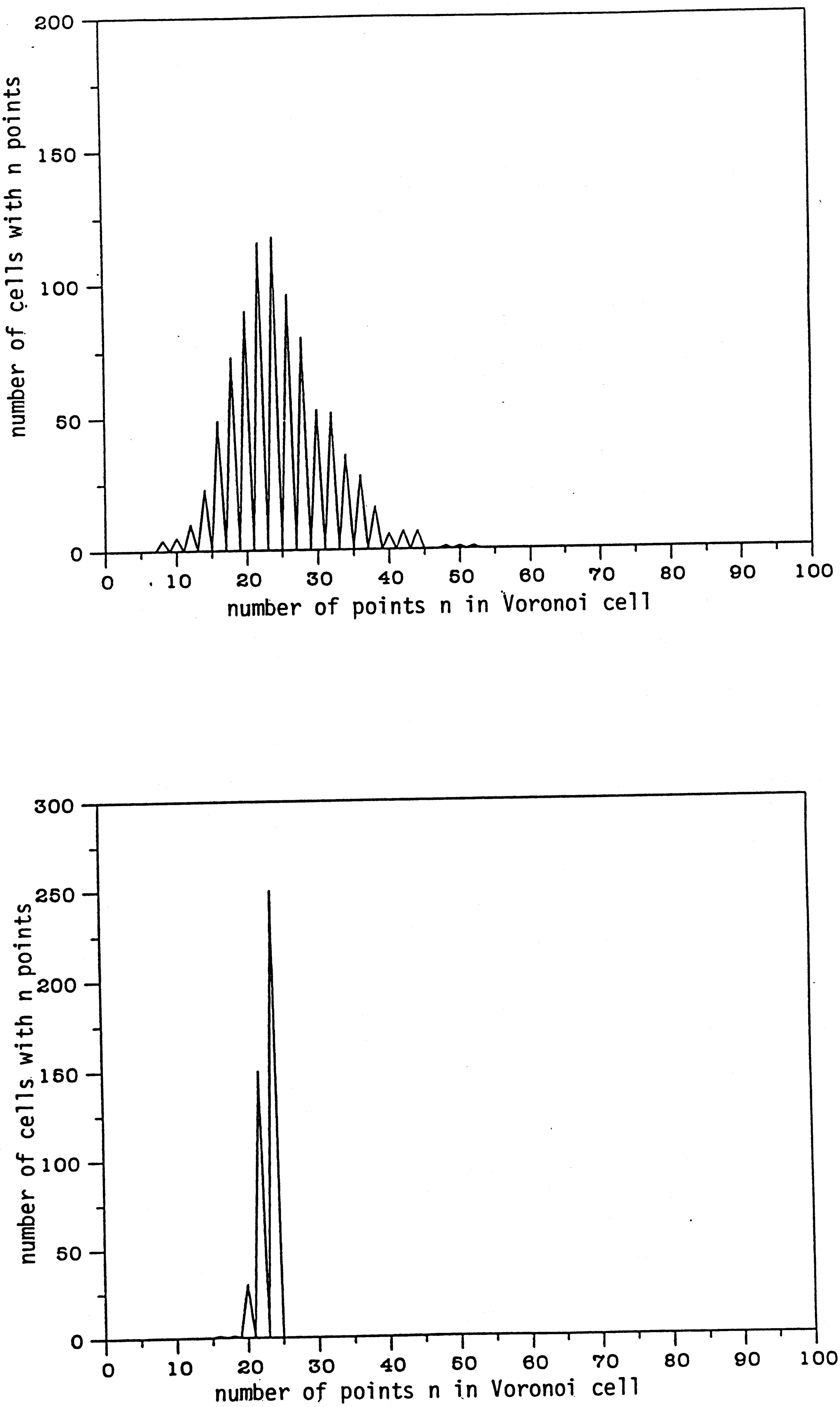

number of edges of length l
20000
15000
10000
5000
0
0
0.5
1
1.5
2
length l

Fig 1

(a)


200
150
100
50
0
number of cells with n points
0
10
20
30
40
50
60
70
80
90
100
number of points n in Voronoi cell
300
250
200
150
100
50
0
number of cells with n points
0
10
20
30
40
50
60
70
80
90
100
number of points n in Voronoi cell

(b)

Fig 2