Discrete Voronoi Diagrams and Post Office Query Structures without the InCircle Predicate

Timothy M. Chan * David L. Millman † Jack Snoeyink ‡

Abstract. We develop an algorithm to compute the Voronoi diagram (or distance transform) of n sites (feature pixels) in a $U \times U$ grid using double precision, instead of the usually 4-fold precision InCircle test. Our algorithm runs in $O(U^2 + n \log U)$ time.

1 Introduction

Geometric algorithms use numerical computations to perform geometric tests, so correct algorithms may produce erroneous results if they require more arithmetic precision than is available. Liotta *et al.* [5] suggested analyzing the precision of an algorithm by the maximum degree of its predicates. They demonstrated that a Voronoi diagram of sites on a grid could be reduced from a degree 6 to a degree 2 point location structure that correctly answers "Post Office" queries: find the nearest site to a query grid point. They still used degree 4 InCircle tests to compute the Voronoi before reducing it; Millman and Snoeyink [6] recently were able to start with a restricted Voronoi diagram computed with degree 3 predicates.

The distance transform of a $U \times U$ pixel image $\mathbb{I}$ can use the Voronoi diagram to label each pixel of $\mathbb{I}$ with its closest feature pixel; Breu *et al.* [1] showed that this could be done in $O(U^2)$ time. Their algorithm used InCircle (degree 4). Chan [2] generalized to compute the d-dimensional *discrete Voronoi diagram* in $O(U^d)$ time; his algorithm can be implemented using predicates of degree 3.

The distance transform can be computed using degree 2 computations by simply comparing squared distances to all feature pixels; in fact, Hoff and others [4] implement this idea in graphics hardware to compute 2D and 3D discrete Voronoi diagrams very quickly, albeit in $\Theta(nU^2)$ total work, and with the fixed resolution of the screen and precision of the Z-buffer.

We show how to compute the 2D discrete Voronoi diagram with double precision in $O(U^2)$ expected time with a scan line algorithm. The 2D discrete Voronoi diagram requires $O(U^2)$ space and answers post office queries in constant time. In some situations this memory requirement is too large and only $O(\log n)$ time post office queries are required. For these situations we describe an intermediate degree 2 data structure called the *Voronoi Polygon Set* that requires only $O(n \log U)$ space and adds no time complexity. Finally, we create a post office query structure from the Voronoi Polygon Set in $O(n \log n)$ expected time using $O(n)$ expected space that answers post office queries in $O(\log n)$ expected time with degree 2 predicates.

2 Geometric Preliminaries

Let $\mathbb{U} = \{1, \ldots, U\}$, and given a set of n feature pixels, called *sites*, $S = \{s_1, \ldots, s_n\}$ in $\mathbb{U}^2$ the *Voronoi Polygon Set* partitions $\mathbb{U}^2$ into n convex polygons $\{C(s_1), \ldots, C(s_n)\}$ where $C(s_i)$ is the convex hull of the grid points in closure of the Voronoi cell of s_i. We call each $C(s_i)$ a *Voronoi Polygon*, and each polygon stores a leftmost and rightmost grid point. Figure 1 shows that *gap* regions may remain between Voronoi Polygons; without higher degree predicates, it seems difficult to determine neighboring Voronoi cells across gaps.

For the remainder let h_i denote the horizontal line $y = i$ and ℓ_j denote the vertical line $x = j$.

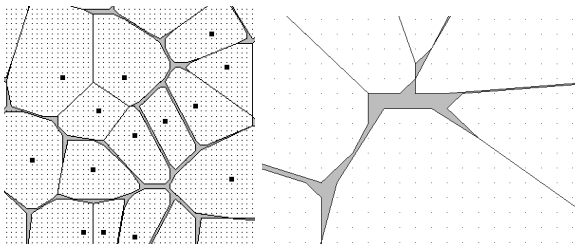

Figure 1: Left: The *Voronoi Polygon Set* of sites randomly chosen on a grid. To recover the Voronoi diagram's precise structure in the gray gaps seems to require higher degree predicates. Right: A close-up of a complex *gap*.

*David R. Cheriton School of Computer Science, University of Waterloo, tmchan@uwaterloo.ca

†Department of Computer Science, University of North Carolina at Chapel Hill, dave@cs.unc.edu

‡Department of Computer Science, University of North Carolina at Chapel Hill, snoeyink@cs.unc.edu

3 Discrete Voronoi Diagram

Given grid size U and a set of n sites S we specialize Chan's algorithm for computing the 2D discrete Voronoi diagram [2] in $O(U^2)$ expected time but using degree 2 predicates only.

Bin: First sort the sites of S by x breaking ties with y. Then partition S into U lists where $S_i = \{s \in S \mid s_x = i\}$, a list for each vertical grid line ℓ_i.

Preprocess: For each S_i solve the $1D$ problem of marking each grid point on ℓ_i with its closest neighbor in S_i. Let $\mathcal{R} = \{R_1, \ldots, R_U\}$ where list R_j contains $s \in S$ such that there exists a grid point q on h_j with $\text{mark}(q) = s$. Preprocessing constructs $\mathcal{R}$ where each R_i has at most U sites.

Scan line: For each vertical line h_j corresponding to R_j, label each grid point p on h_j with its closest site in S. The labels are stored in a list I_j of tuples (s, a, b) where grid points in the interval $[a, b]$ on h_j have site s as a label. This creates U lists $\{I_1, \ldots, I_U\}$ each with at most U intervals. The interval lists encode the discrete Voronoi diagram.

Space and time analysis omitted for abstract.

Bin compares degree 1 coordinate values, Preprocess and Scan line compare degree 2 squared distances. Therefore, we compute the Discrete Voronoi diagram in degree 2.

4 Post Office Query Structure

We describe an $O(U^2)$ space algorithm for clarity and point out that the Voronoi Polygon Set can be built one scan line at a time to reduce memory.

Given the interval set from the previous section, an alternative output is an expected $O(n)$ size data structure for solving post office queries in $O(\log n)$ expected time with degree 2 predicates. Computing this uses $O(U + n \log U)$ space and expected $O(U^2 + n \log n)$ time.

Compress: For each interval set I_j, merge the intervals of I_j into the Voronoi Polygon Set of rows $\{1, \ldots, j-1\}$. Update the leftmost and rightmost grid points of modified Voronoi Polygons.

Build: Represent each Voronoi Polygon $C(s_i)$ with a segment from a leftmost to rightmost grid point and a link to associated site s_i. Build the trapezoid graph [3] of the representative segments.

Space and time analysis omitted for abstract.

Compress and Build evaluate degree 2 orientation predicates on grid points and compare degree 1 x-coordinates. Thus, building the polygon set and trapezoid graph are degree 2.

Given a query point q, we solve post office queries on the trapezoid graph in $O(\log n)$ expected time by locating the trapezoid containing q [3]. The trapezoid is defined by at most two non-vertical segments each with an associated site. The post office query is answered by the site closer to q.

Point location uses the same predicates as building the trapezoid graph with the addition for comparing degree 2 squared distances. Therefore, point location is degree 2.

5 Conclusion

Our algorithm provides an exact method for computing the discrete Voronoi diagram and a post office query structure on a point set with single precision coordinates in double precision. The algorithm is simple to implement and degeneracies are easily handled.

We believe that our algorithm is a good candidate for parallelization. Each scan line step is independent; horizontal slabs can be processed concurrently and merged with a parallel reduce. We would like to investigate parallelizing our algorithm on graphical processing units since they show impressive results for Hoff *et al.* [4] and our algorithm does not require any libraries for robust computation.

We believe that our algorithm may generalize in dimension and apply to the discrete Power and furthest point Voronoi diagrams while maintaining degree 2 algebraic complexity. However, it is still unknown if a post office query structure is computable in subquadratic time with degree 2 predicates.

References

[1] H. Breu, J. Gil, D. Kirkpatrick, and M. Werman. Linear time Euclidean distance transform algorithms. *IEEE PAMI*, 17:529–533, 1995.

[2] T. M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Comput. Geom. Theory Appl.*, 35(1):20–35, 2006.

[3] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York, Inc., 3rd edition, 2008.

[4] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast computation of generalized Voronoi diagrams using graphics hardware. In *ACM SIGGRAPH*, pages 277–286, 1999.

[5] G. Liotta, F. P. Preparata, and R. Tamassia. Robust proximity queries: An illustration of degree-driven algorithm design. *SIAM J. Comput.*, 28(3):864–889, 1999.

[6] D. L. Millman and J. Snoeyink. Computing the implicit Voronoi diagram in triple precision. In *WADS*, volume 5664 of *LNCS*, pages 495–506. Springer, 2009.