Using simplicial partitions to determine a closest point to a query line

Asish Mukhopadhyay*
School of Computer Science
University of Windsor
Windsor, Canada

Abstract

In this note we show that simplicial partitions can be used to answer the following queries efficiently: given a point set in the plane, determine a point that is closest to an arbitrary query line.
Keywords: Computational geometry; pattern classification; closest point; query line; simplicial partitions

1 Introduction

The problem of preprocessing a set, S, of n points in the plane to determine efficiently the closest point to a query line, l, was initially addressed by Cole and Yap [2], who obtained a solution with preprocessing time and space in $O(n^2)$ and query time in $O(\log n)$. Lee and Ching [3] obtained the same result, using a duality-based approach. Mitra [4] obtained a solution with preprocessing time and space in $O(n \log n)$ and query time in $O(n^{0.695})$. Subsequently, Mitra and Chaudhuri [5] improved the preprocessing space complexity to $O(n)$. In this paper we show that a simplicial partition of the point set can be used to obtain a solution whose preprocessing time and space are in $O(n^{1+\epsilon})$ and $O(n \log n)$ respectively and query time in $O(n^{\frac{1}{2}+\epsilon})$, for arbitrary $\epsilon > 0$.

The practical applications that motivate problems of this genre come from the areas of Pattern Recognition and Image Processing. For example, if the query line separates two classes of patterns, then an answer to the above query determines the pattern closest to the query line and provides an idea about the stability of classification.

2 Geometric Insights

The most trivial solution is to avoid all preprocessing, look at each point in turn and determine the closest. The space and query times of this naive method are both in $O(n)$. Ways of improving this naive solution come from the insight that we should preprocess the point set in a way that allows us to look at well-structured groups of points sets that are disjoint from the query line instead of a point at a time.

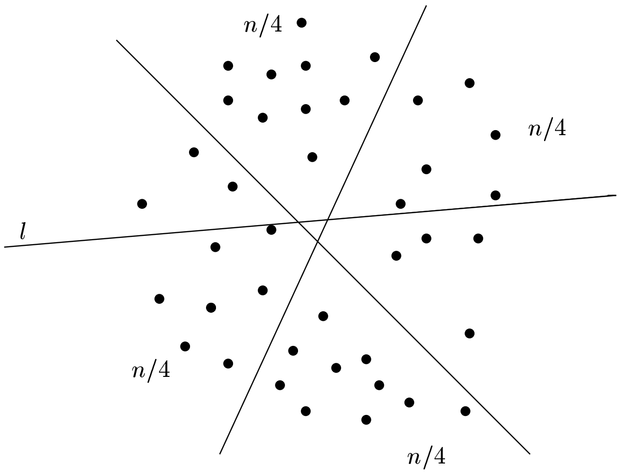

Figure 1: Line l cuts a hamsandwich partition

An arbitrary query line can intersect at most three of the four regions into which a plane is partitioned by two intersecting lines. Mitra [5] uses this insight to partition a point set into 4 groups each of size $n/4$ using ham-sandwich cuts such that the query line is disjoint from at least one of the four groups (see Fig. 1). The recursive hamsandwich partitions of the groups intersected by the query line allows one to identify smaller subgroups of points disjoint from the query line at each level of the recursion. Each group of points disjoint from the query line is queried with respect to its convex hull since a closest point is on the hull boundary.

If $Q(n)$ is the time to resolve a query on n points then it satisfies the recurrence

$$Q(n) \leq Q(n/2) + Q(n/4) + O(\log n) \quad (1)$$

where $O(\log n)$ is the time required to resolve a query with respect to $n/4$ points that are provably disjoint from the query line.

*Research supported by an NSERC Operating Grant 227693

The interesting question here is whether there are other partitions of the point sets that yield better query times and, if so, what are the trade-offs involved vis-a-vis processing time and space.

3 Simplicial Partitions

Simplicial partitions have been used in the context of range-searching [7]. In this paper we explore its applicability to the problem at hand.

A *simplicial partition* of S is a collection of pairs

$$\Psi(S) \quad = \quad \{(S_1, t_1), (S_2, t_2), \ldots, (S_r, t_r)\} \qquad (2)$$

where the S_i's are mutually disjoint subsets of S whose union is S, and each t_i is a triangle that contains S_i (see Fig. 2).

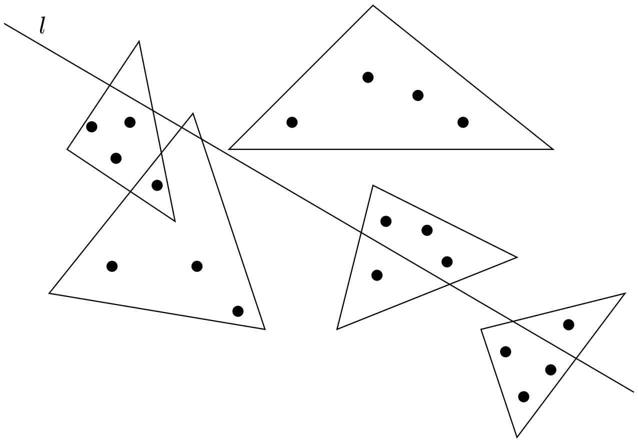

Figure 2: Line l cuts a simplicial partition

A simplicial partition is *fine* if $|S_i| \leq 2 * n/r$ for each i. For a given simplicial partition $\Psi(S)$, the *crossing number* of a line l is the number of triangles of $\Psi(S)$ that the line properly intersects. The crossing number of $\Psi(S)$ is the maximum crossing number over all lines that intersect the simplicial partition.

The following important result by Matousek [6] shows how to construct a fine simplicial partition with a low crossing number.

Theorem 1 *For any given set S of n points and a parameter r, $1 \leq r \leq n$, a fine simplicial partition of size r and crossing number $O(\sqrt{r})$ exists. Further, for any given $\epsilon > 0$, such a simplicial partition can be constructed in time in $O(n^{1+\epsilon})$.*

A simplicial partition is stored in an r-way tree, called a *partition tree*. At each of the r children of the root of a partition tree we store the simplex that encloses the corresponding point set, as well as any secondary information related to the point set. Each child-node is in turn the root of a partition tree on the point set that belongs to this node.

4 Algorithmic Details

The constructibility of a fine simplicial partition with a low crossing number implies that the point sets enclosed by $r - O(\sqrt{r})$ simplexes are not intersected by a query line l and the convex hulls of these can be queried for the closest point to the query line. For the remaining $O(\sqrt{r})$ simplices that are intersected by the query line, we proceed recursively on their respective simplicial partitions.

In the analysis below we appeal to the following theorems:

Theorem 2 *([1]) The minimum distance of a line l from a convex polygon on n vertices can be determined in time in $O(\log n)$.*

Theorem 3 *[8] The convex hull of a set of n points in the plane can be constructed in $O(n \log n)$ time.*

4.1 Processing time

The processing time required to build the partition tree is $O(n^{1+\epsilon})$ by Theorem 1 above. After building the partition tree, we compute the convex hull of the points associated with each node v. If $T(n)$ be the time required to compute this information on a partition tree for n nodes then it satisfies the recurrence inequality:

$$T(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ O(n \log n) + \sum_1^r T(n_v) & \text{if } n > 1 \end{cases}$$

Since $n_v \leq 2n/r$, for $r > 2$, $n_v < n$. Then as $\sum n_v = n$ and r is a constant, we conclude that $T(n) = O(n \log^2 n)$.

Since $\log^2 n = o(n^\epsilon)$ for arbitrary $\epsilon > 0$, the time needed to construct the partition tree dominates the preprocessing time.

4.2 Processing Space

If $S(n)$ be the space required in a partition tree on n nodes, then it satisfies the recurrence inequality:

$$S(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ n + r + \sum_1^r S(n_v) & \text{if } n > 1 \end{cases}$$

where n_v is the number of points at node v. Since $n_v \leq 2n/r$, for $r > 2$, $n_v < n$. Then as $\sum n_v = n$ and r is a constant, we conclude that $S(n) = O(n \log n)$.

4.3 Query time

If we let $Q(n)$ denote the query time on a point set of size n then it satisfies the following inequality:

$$Q(n) \leq \begin{cases} 1 & \text{if } n = 1 \\ r + \sum O(\log(n_v)) + \sum Q(n_v) & \text{if } n > 1 \end{cases}$$

since the nearest point query is done on the convex hull of at most $r - c\sqrt{r}$ sets S_i, and we recur on at most $c\sqrt{r}$ subsets S_i that are intersected by the query line l. The sum $\sum Q(n_v)$ is precisely taken over these latter subsets, while the sum $\sum O(\log(n_v)$ is taken over the remaining subsets whose convex hulls are queried for the closest point to the query line l.

For a given $\epsilon > 0$ and $c > 0$, if we set $r = \lceil 2(c\sqrt{2})^{\frac{1}{\epsilon}} \rceil$ then r becomes independent of n and it is easily shown that

$$Q(n) \quad = \quad O(n^{1/2+\epsilon}) \tag{3}$$

5 Conclusions

In this note we have shown that the results of Mitra and Chaudhuri [5] can be generalized to simplicial partitions. We have been able to improve the query time at the expense of increasing storage by a logarithmic factor.

An open question is whether this problem can be solved so that the processing time and space are in $O(n \log n)$ and $O(n)$ respectively and the query time in $O(polylog(n))$.

6 Acknowledgements

Constructive feedback from Dr. Pinaki Mitra on an initial draft of this paper are gratefully acknowledged.

References

[1] B. Chazelle, Comnputational Geometry and Comvexity, *Ph. D thesis, CMU*, 1980.

[2] R. Cole and C. Yap, Geometric retrieval problems, *Proc. IEEE 24th Symp. on Foundations of Computer Science*, pp 112 -121, 1983.

[3] D. T. Lee and Y. T. Ching, The power of geometric duality revisited, *IPL*, Vol. 21, pp 117 - 122, 1985.

[4] P. Mitra, Finding the closest point to a query line, in *Snapshots in Computational Geometry (Editor G. Toussaint)*, Vol II, pp 53-63, 1992.

[5] P. Mitra and B. B. Chaudhuri, Efficiently computing tthe closest point to a query line, *Pattern Recognition Letters*, Vol 19, pp 1027-1035, 1998.

[6] J. Matoušek, Efficient partition trees, *Discrete and Computational Geometry*, Vol. 8, pp 315-334, 1992.

[7] Mark de Berg, Marc van Kreveld, Mark Overmars, Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Second edition, Springer Verlag, 2001.

[8] Franco P. Preparata, M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 2nd edition, 1990