A plane sweep algorithm for the all nearest neighbors problem for a set of convex planar objects

THORSTEN GRAF AND KLAUS HINRICHS*

December 16, 1992

1 Introduction

We consider the two-dimensional *all nearest neighbors problem* (ANN problem) for a set of non-intersecting convex objects with respect to arbitrary L^t-metrics d_t $(1 \leq t \leq \infty)$:

Given a set S of n non-intersecting compact convex objects in the plane, find a nearest neighbor of each with respect to d_t.

The Minkowski-distance d_t of two objects $s_1, s_2 \in S$ is defined as follows:

$$\begin{aligned} d_t(s_1,s_2) &:= \min\{(|p.x - q.x|^t + |p.y - q.y|^t)^{\frac{1}{t}} : p \in s_1, q \in s_2\} \quad 1 \leq t < \infty \\ d_\infty(s_1,s_2) &:= \min\{\max\{|p.x - q.x|, |p.y - q.y|\} : p \in s_1, q \in s_2\} \end{aligned}$$

If S consists of n points $\Omega(n \log n)$ is a well known tight lower bound for this problem in the algebraic decision tree model of computation. Optimal algorithms for this simple version of the problem are given in [Sh 75], [Va 89], [HNS 92]; a survey is given in [FKP 92]. However, all these algorithms do not apply to sets of more complex objects, e.g. convex polygons.

[BH 92] presents a plane sweep algorithm to solve the closest pair problem for a set of convex planar objects with respect to an arbitrary Minkowski-metric. This algorithm tests whether a new object encountered during the left-to-right sweep forms a new closest pair with any of those objects seen so far. Among these objects only those intersecting the δ-slice to the left of the sweep line have to be considered, where δ denotes the minimal object distance found so far. A total ordering on these *active* objects is determined by their intersections with the δ-slice. During the sweep the algorithm maintains the set of active objects and the minimal distance δ between any pair of active objects which become neighbors with respect to this total ordering. New neighbor pairs are obtained either by encountering a new object or by deactivating an object which does not intersect the δ-slice any more. It seems to be surprising that this algorithm finds the correct result by just testing pairs of active objects which become neighbors during the sweep. [BH 92] proves that this is sufficient for both an intersection free configuration and a configuration with intersecting objects, a more intuitive proof is given in [BGH 92].

The technique used in the above algorithm cannot be applied to the ANN problem for convex objects. Since we search for a nearest neighbor of each object in our configuration, we do not have a global δ for all objects but an individual δ for each object. This makes the technique used in [BH 92] unsuitable for the ANN problem since the deactivation events do not occur in a predefined order and therefore have to be rearranged dynamically. Even worse, the individual δ-values emphasize the local nature of the problem, and therefore it is not necessarily true that at any time during the left-to-right sweep or the right-to-left sweep an object and any of its nearest neighbors become neighbors in the y-table, i.e. with respect to the total ordering in [BH 92].

Our algorithm PSANN (=Plane Sweep All Nearest Neighbors) uses four sweeps, from left to right, from right to left, from top to bottom and from bottom to top. The algorithm also works correctly if we weaken the condition of disjointness to the conditon that the objects in our configuration are *diagonal disjoint*, which means that for any object the x-diagonal connecting certain x-extremal and the y-diagonal connecting certain y-extremal points do not intersect any other object of S. For configurations consisting of n line segments, circular discs or convex polygons whose number of edges is treated as a constant, PSANN runs in asymptotically optimal time $O(n \log n)$. Let S be a set of n convex polygons with a total of N edges. If each polygon is given by its vertices in cyclic order then PSANN finds nearest neighbors with respect to the Euclidean metric in time $O(n \log N)$. This runtime is achieved by employing an optimal $O(\log m)$ algorithm [CD 87] for detecting whether two convex polygons with at most m edges intersect, and an optimal $O(\log m)$ algorithm ([E 85], [CW 83]) for computing the minimal Euclidean distance between two such non-intersecting convex polygons.

*Institut für numerische und instrumentelle Mathematik -INFORMATIK -, Universität Münster, Einsteinstr. 62, D-4400 Münster, e-mail: thorsten@math.uni-muenster.de, hinrichs@math.uni-muenster.de

2 Plane-sweep applied to the all nearest neighbors problem

Our algorithm PSANN is based on the well known plane-sweep method. Plane-sweep's name is derived from the image of sweeping the plane from left to right with a vertical line (front, or cross section), stopping at every transition point (event) of a geometric configuration to update the cross section, i.e. to maintain the problem-dependent ***sweep invariants*** which have to hold for the objects being encountered so far. All processing is done at this moving front, without any backtracking, with a look-ahead of only one object.
The sweep invariants determine the events contained in the x-queue (***event queue***) and the status of the sweep which is maintained by the y-table (***sweep line structure***). In the slice between two events the properties of the geometric configuration detected so far do not change, and therefore no invariant has to be maintained and the y-table does not have to be updated. The skeleton of a plane-sweep algorithm is as follows: The procedure 'transition' is the

```
initialize x-queue;
initialize y-table;
while not empty(x-queue) do
    {e := next(x-queue); transition(e)}
```

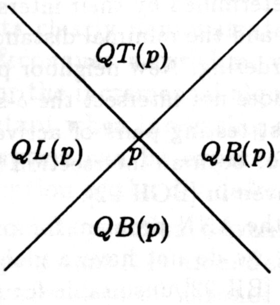
advancing mechanism of the plane-sweep. It embodies all the work to be done when a new event is encountered; it moves the front from the slice to the left of an event e to the slice immediately to the right of e.
In this paper we apply the plane-sweep principle to solve the all nearest neighbors problem in a set S consisting of n convex compact objects with respect to any L^t-metric d_t $(1 \leq t \leq \infty)$. For each object $s \in S$ let $s[L]$ denote the smallest and $s[R]$ the largest point in s according to the lexicographic order '$\leq^x$':

$$\forall p, q \in E^2, p \leq^x q :\Longleftrightarrow (p.x < q.x) \vee [(p.x = q.x) \wedge (p.y \leq q.y)].$$

Throughout this paper we write $p \leq_x q$ and $p \leq_y q$ instead of $p.x \leq q.x$ and $p.y \leq q.y$, respectively. These notations are extended to point sets as follows

$$P \leq_x Q :\Longleftrightarrow \forall p \in P\ \forall q \in Q : p \leq_x q$$

For a point $p \in E^2$ the two diagonal lines (with slopes ± 1) through p subdivide the plane into four quadrants. Let us denote these quadrants as follows:



$$QR(p) := \{q >_x p : |p.x - q.x| \geq |p.y - q.y|\}$$
$$QL(p) := \{q <_x p : |p.x - q.x| \geq |p.y - q.y|\}$$
$$QB(p) := \{q <_y p : |p.x - q.x| < |p.y - q.y|\}$$
$$QT(p) := \{q >_y p : |p.x - q.x| < |p.y - q.y|\}$$

For an object $s \in S$ denote by $nn(s)$ a nearest neighbor found so far and by $\delta(s)$ the distance between s and $nn(s)$. Furthermore let $NN(s)$ be the set of all nearest neighbors of s in S and $\Delta(s)$ their distance to s.
Our algorithm PSANN uses four sweeps: from left to right, from right to left, from top to bottom and from bottom to top. We only describe the left-to-right sweep, the other sweeps work similarly. In the left-to-right sweep we find a nearest neighbor $r \in NN(s)$ for all those objects $s \in S$ for which there exist points $p \in s$ and $q \in r \cap QR(p)$ such that $d_t(p,q) = d_t(s,r)$. In the three remaining sweeps we solve the partial nearest neighbors problems with the corresponding points $p \in s$ and $q \in r \cap QL(p)$, $q \in r \cap QB(p)$ and $q \in r \cap QT(p)$. During the sweep PSANN maintains for each object $s \in S$ the smallest distance $\delta(s)$ to another object detected so far. The y-table stores the ***active*** objects $s \in S$ which have been encountered by the sweep line SL and intersect their individual $\delta(s)$-slice $D_{\delta(s)}$ to the left of SL, i.e. the objects for which $s[R].x + \delta(s)$ is to the right of SL. These are exactly those objects of S encountered so far by SL which can have a nearest neighbor closer than $\delta(s)$ among the objects of S lying completely to the right of SL.
The x-queue is initialized with two sets of events: ***insertion events*** and ***deletion events***. Insertion events are given by the left end points: an object s is inserted into the y-table when the sweep line encounters its left end point $s[L]$. Deletion events are determined by the right end points and the smallest distance to another object detected so far: an object s is removed from the y-table as soon as it no longer intersects its $\delta(s)$-slice $D_{\delta(s)}$, i.e. if the position of the sweep line is at or to the right of $s[R].x + \delta(s)$. This may happen either if the sweep line proceeds to

the right or if an object's $\delta(s)$-slice shrinks because $\delta(s)$ becomes smaller. Since $\delta(s)$ can be different for different active objects, the deactivation events can not be processed in a predefined order as in [BH 92]. Hence we have to deal with a dynamic processing of deactivation events. Repeated shrinking of $\delta(s)$ for an active object s requires left shifts of its deactivation event. We only know that the sweep line is at or to the right of $s[R].x + \delta(s)$ when a deletion event for s is executed.
In order to allow an efficient processing of the objects contained in the y-table it is necessary to find a canonical total order on these objects. Such a total order will be defined in the following, it is the same as in [BH 92]. For each object $s \in S$ let (Fig. 1)

$$R(s) := \{(x,y) : x \geq s[R].x,\ y = s[R].y\} \qquad Q(s) := \overline{s[L], s[R]} \qquad \overline{Q}(s) := Q(s) \cup R(s)$$

For the moment we assume that there are no intersecting pairs of objects in S; we will show later that the left-to-right sweep works correctly even if there are intersecting pairs of objects with the restriction that S is ***x-diagonal disjoint***, i.e. for any two objects $s_1, s_2 \in S$ we have $Q(s_1) \cap s_2 = s_1 \cap Q(s_2) = \emptyset$. Let $s[SL] := \overline{Q}(s) \cap SL$ denote the ***representation point*** of an active object $s \in S$ with respect to a sweep line SL. For objects $s_1, s_2 \in S$ and a sweep line SL we define

$$s_1 \sqsubseteq s_2 :\Longleftrightarrow s_1[SL] <_y s_2[SL]$$

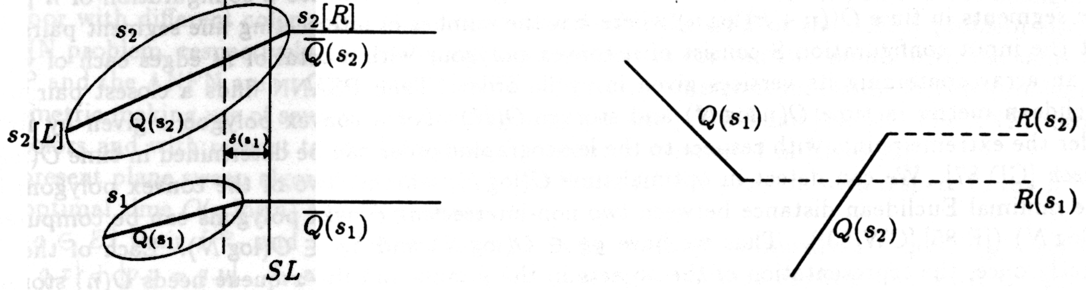


Figure 1: a) Q and $\overline{Q}$ for two objects $s_1 \sqsubseteq s_2$ b) Do we have to exchange s_1 and s_2?

During the sweep the active objects are stored in the y-table with respect to $\sqsubseteq$. It seems to be necessary to exchange objects in the y-table when the sweep line reaches an intersection point of $R(s_1)$ and $Q(s_2)$ for two active objects $s_1, s_2 \in S$ (Fig. 1 b)).
However, this never happens since s_1 has been deactivated when the intersection point of $R(s_1)$ and $Q(s_2)$ is reached by SL: If $\tilde{p} := R(s_1) \cap Q(s_2)$ is such an intersection point with smallest x-coordinate and s_1 and s_2 are both active when $\tilde{p}$ is reached by SL then obviously s_1 and s_2 are neighbors in the y-table with respect to $\sqsubseteq$. Now s_1 has been deactivated since $\delta(s_1) \leq d_t(s_1, s_2) \leq \tilde{p}.x - s_1[R].x$. The same argument applies to the next such intersection point if we consider the set $S \setminus s_1$ instead of S, and similarly to all further intersection points. Hence active objects do not change their relative order with respect to $\sqsubseteq$ while SL is moving to the right. Since furthermore $s_1[SL] \neq_y s_2[SL]$ for two active objects $s_1 \neq s_2$ the relation $\sqsubseteq$ is a total ordering on the objects in the y-table.
During the sweep from left to right we maintain the following *sweep invariants*:

1) Each object s in the y-table intersects its individual $\delta(s)$-slice $D_{\delta(s)}$ to the left of SL.
2) For s_1, s_2 neighbored in the y-table with respect to $\sqsubseteq$ we have $d(s_1, s_2) \geq \delta(s_1)$ and $d(s_1, s_2) \geq \delta(s_2)$.

The first invariant is maintained by removing objects from the y-table that no longer intersect their $\delta(s)$-slice. In order to maintain the second invariant we have to compute distances of objects which become neighbors with respect to $\sqsubseteq$ in the y-table and update their δ-values, if necessary. We obtain such new neighbor pairs after inserting a new object into the y-table or after removing an object from the y-table.
It seems to be surprising that PSANN finds the correct result by just testing pairs of objects which become neighbors in the y-table with respect to $\sqsubseteq$. In the full version of the paper ([GH 92]) we prove that this is sufficient both for intersection free and for diagonal disjoint configurations.

3 Analysis

Let S be a set of n convex planar diagonal disjoint objects. Let $M(s)$ denote the storage needed by algorithm PSANN for an object $s \in S$. Then obviously PSANN needs a total of $M(S) := \sum_{s \in S} M(s)$ storage, which is in $O(n)$ if each object uses $O(1)$ storage as it is true for configurations consisting of points, line segments or disks, and $O(N)$ for a configuration of n convex polygons with a total of N edges. Let the input configuration S be subset of a specific class C of compact, convex point sets of a certain type in the plane, e.g. the class of points, line segments, disks or the class of convex polygons. We assume that there exist functions g_C and h_C which depend on the underlying class C of objects such that:

- For each $s \in C$ the points $s[L]$ and $s[R]$ can be computed by at most $O(g_C)$ operations.
- For any $s_1, s_2 \in C$ their minimal distance $d_t(s_1, s_2)$ with respect to any L^t-metric d_t $(1 \leq t \leq \infty)$ can be computed by performing at most $O(h_C)$ operations.

For computing all points $s[L]$ and $s[R]$, $s \in S$, we need $O(n\, g_C)$ operations. The initialization of the x-queue, i.e. sorting the objects with respect to $s[L].x$ and $s[R].x$, can be accomplished in $O(n \log n)$ worst-case time. Each comparison of two objects with respect to $\sqsubseteq$ costs constant time; therefore the cost for all operations performed on the y-table during the plane-sweeps, i.e. inserting, deleting objects and finding neighbors, sums up to $O(n \log n)$ time in the worst case. PSANN computes the distance of at most $3(n-2)+1 = 3n-5$ pairs of objects $(n \geq 3)$, implying that all these distance computations take $O(n\, h_C)$ time. Each distance computation may result in shifting of two deactivation events which costs $O(\log n)$ (see appendix) and therefore in total $O(n \log n)$. Hence the plane sweep algorithm PSANN solves the all nearest neighbors problem for a configuration S as defined above in time $O(n \log n + n(g_C + h_C))$. It is obvious that PSANN solves the all nearest neighbors problem for a configuration S of n objects in time $O(n \log n)$ and storage $O(n)$ if the underlying class C is the class of points, the class of line segments, the class of disks or another class for which we have $g_C, h_C \in O(1)$.
By using the same technique as in the Bentley-Ottman algorithm [BO 79] for detecting line segment intersections PSANN can be modified to solve the all-nearest-neighbors problem for a configuration of n possibly intersecting line segments in time $O((n+k)\log n)$ where k is the number of intersecting line segment pairs.
Let the input configuration S consist of n convex polygons with a total of N edges each of which is represented by an array containing its vertices given in cyclic order. Then PSANN finds a closest pair with respect to the Euclidean metric in time $O(n \log N)$ and storage $O(N)$: For a convex polygon s given by its vertices in cyclic order the extreme points with respect to the lexicographic order can be determined in time $O(\log N)$ by Fibonacci search [CD 87]. We can detect in optimal time $O(\log N)$ whether two of the convex polygons intersect [CD 87]. The minimal Euclidean distance between two non-intersecting convex polygons can be computed in optimal time $O(\log N)$ ([E 85],[CW 83]). Thus we have $g_C \in O(\log N)$ and $h_C \in O(\log N)$. Each of the N edges is stored exactly once, the representation of the objects in the y-table and the x-queue needs $O(n)$ storage which sums up to a storage complexity of $O(N+n) = O(N)$.

References

[BGH 92] F. Bartling, Th. Graf, K. Hinrichs: A plane sweep algorithm for finding a closest pair among convex planar objects, *internal report*

[BH 92] F. Bartling, K. Hinrichs: A plane-sweep algorithm for finding a closest pair among convex planar objects, A. Finkel, M. Jantzen (eds.), *STACS 92, 9th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 577, 221-232, Springer-Verlag, Berlin, 1992

[BO 79] J.L. Bentley and T. Ottmann, Algorithms for reporting and counting intersections, *IEEE Transactions on Computers C28*, 643-647 (1979).

[CD 87] B. Chazelle, D. P. Dobkin: Intersection of convex objects in two and three dimensions, *Journal of the ACM*, 34(1), 1-27 (1987).

[CW 83] F. Chin, C. A. Wang: Optimal algorithms for the intersection and the minimum distance problems between planar polygons, *IEEE Trans. Comput.* 32(12), 1203-1207 (1983).

[E 85] H. Edelsbrunner: Computing the extreme distances between two convex polygons, *Journal of Algorithms* 6, 213-224 (1985).

[FKP 92] P.-O. Fjällström. J.Katajainen, J. Petersson: Algorithms for the all-nearest-neighbors problem, *Report 92/2*, Dept. of Computer Science, University of Copenhagen, Denmark 1992.

[GH 92] Th. Graf, K. Hinrichs: A plane sweep algorithm for the all nearest neighbors problem for a set of convex planar objects, *Preprints "Angewandte Mathematik und Informatik"*, Nr. 17/92-I, Westfälische Wilhelms-Universität Münster, 1992

[HNS 92] K.Hinrichs, J.Nievergelt, P.Schorn: An all-round algorithm for 2-dimensional nearest-neighbor problems, *Acta Informatica*, 29(4), 383-394 (1992)

[Sch 91] P. Schorn: Robust algorithms in a program library for geometric computation, PhD Dissertation No. 9519, ETH Zürich, Switzerland, 1991

[Sh 75] M.Shamos, D.Hoey: Closest-point problems, *16th Annual IEEE Symposium on Foundation of Computer Science*, 151-162 (1975)

[Va 89] P.Vaidya: An $O(n \log n)$ algorithm for the all-nearest-neighbours-problem, *Discrete & Computational Geometry*, 4, 101-115 (1989)