Order type invariant labeling and comparison of point sets

Greg Aloupis[*] Muriel Dulieu[†] John Iacono[†] Stefan Langerman[*] Özgür Özkan[†]
Suneeta Ramaswami[‡] Stefanie Wuhrer[§]

Abstract

We consider the problem of computing an order type invariant labeling for a given set of n points. In $2D$, such a labeling can be constructed in $O(hn^2)$ time, where h is the size of the smallest convex layer. In $3D$ the time complexity is $O(n^3 \log n)$ if the point set is in general position[1].

This is useful to test if two point sets have the same order type within the same time bounds. It can also be used as preprocessing for any order type invariant algorithm, such as triangulation/tetrahedralization, or polygonization.

1 Introduction

When an algorithm computes a structure such as a triangulation or polygonization, and many solutions are possible, sometimes it is desirable that the same solution is always returned for two combinatorially equivalent point sets. With this in mind, we consider the problem of computing an order type invariant labeling for a point set S.

In the plane, the *order type* of S is determined by assigning to every ordered triple of points p_i, p_j, p_k an orientation depending on their relative positions: clockwise, counter-clockwise, or collinear. In 3D, order type labels are assigned to ordered subsets of four points, depending on their arrangement (collinear, coplanar, left/right-handed tetrahedra). An order type representation (OTR) of S is any (possibly implicit) encoding of this information.

Two unlabeled point sets S_1 and S_2 are *combinatorially equivalent* if and only if they have the same order type, or in other words if they have the same OTR for some labeling. If the point sets are labeled, then there must be a bijection between S_1 and S_2, such that any triple in S_1 has the same orientation as its corresponding triple in S_2. In other words, the OTR of S_1 will be identical to the OTR for some permutation (or relabeling) of S_2. Note that the bijection is not necessarily unique.

Our goal is to provide an order type invariant (re)-labeling of a given point set S (this labeling of S, $p'_1, p'_2, \ldots, p'_n$ is just a permutation of the given points). This means that for any combinatorially equivalent set S' labeled by the algorithm as $q'_1, q'_2, \ldots, q'_n$, we will have the property that OTR$(p'_1, p'_2, \ldots, p'_n)$ = OTR$(q'_1, q'_2, \ldots, q'_n)$. The implication is that any algorithm relying only on combinatorial structure will produce the same output regardless of the initial labeling of S, if our relabeling is used as preprocessing.

We show that in $2D$ an order type invariant labeling can be computed in $O(hn^2)$ time, where h is the size of the smallest convex layer of the given point set. The factor h actually represents the size of the smallest subset of labelings (out of all $n!$ labelings of S) that can be formed "quickly" so that any labeling in the subset has a distinguishable OTR compared to any labeling left out. It seems that producing such a subset is not harder than producing a subset of "representative points" of S. So, equivalently one could think of our first phase as finding h "special" points among S. We first focus on how to report h labelings, in Section 3. These are further processed to report a unique labeling, by comparing all OTRs, in Section 3. The factor of $O(n^2)$ in the above term represents the smallest known encoding of an OTR in $2D$.

In $3D$, for general position, we can quickly reduce S to a subset consisting of a constant number of points (equivalently, we obtain a constant number of labelings). Our order type invariant labeling is done in $O(n^3 \log n)$ time; see Section 4.

Our labeling permits the comparison of two point sets, to see if they have the same order type. See Section 5.

2 Related work

Goodman and Pollack [7] showed that the number of order types on n points is at least $n^{4n+O(n/\log n)}$ and at most n^{6n}. They improved the upper bound later to $\left(\frac{n}{2}\right)^{4n(1+O(1/\log(n/2)))}$ [8]. Aichholzer et al. [1] enumerated all order types for up to 10 points, and thus established that any two such point sets in general position with the same number of hull points have isomorphic triangulations. Aichholzer and Krasser [2] extended this to sets of 11 points. Goodman et al. [10] showed that the coordinate representation of order type requires exponential storage.

[*]Département d'Informatique, Université Libre de Bruxelles, aloupis.greg@gmail.com, stefan.langerman@ulb.ac.be

[†]Department of Computer Science and Engineering, Polytechnic Institute of New York University, mdulieu@gmail.com, jiacono@poly.edu, ozgurozkan@gmail.com

[‡]Department of Computer Science, Rutgers University, rsuneeta@camden.rutgers.edu

[§]Cluster of Excellence MMCI, Saarland University, swuhrer@mmci.uni-saarland.de

[1]Our original accepted submission contained claims about higher dimensions and non-general position. After realizing that there was a flaw in our proof, we have since needed to revoke that portion of our work, as we work on a suitable correction.

The most related work is that of Goodman and Pollack [9], in which an efficient OTR was defined for any dimension d. Their OTR in 2D stores the number of points to the left of the vector through every ordered pair of points. In 3D, for each ordered triple of points, it stores the number of points in the halfspace bounded by the halfplane through the triple, with normal vector equal to the cross product of the triple. This extends easily to any dimension d, where the OTR has size $O(n^d)$. Note that there is no assumption on general position. Interestingly, they showed that this is sufficient to determine exactly which points are in any particular halfspace. The time to compute their OTR was given as $O(n^d \log n)$. Note that it has been established elsewhere that in 2D the time complexity is quadratic [6, 4].

Goodman and Pollack also defined *canonical orderings*, which are essentially a subset of all $n!$ orderings (labelings) that can be checked to determine if two sets are combinatorially equivalent. For 2D they gave h canonical orderings, one for each convex hull point p. For each p, the remaining points can be labeled by a clockwise sort around p, starting from any direction that does not intersect the convex hull. It is easy to see that if two point sets S_1,S_2 have the same order type, then the OTR of S_1 will match the OTR for at least one of the canonical orderings of S_2. This was a great improvement over the $n!$ OTR comparisons that would have to be made using brute force. Since two OTRs can be compared in quadratic time, two point sets can be compared in $O(hn^2)$ time. Note that one does not need the improvement of [6, 4] to avoid spending $\Theta(n^2 \log n)$ time computing each of the h OTRs. Since each of these OTRs represents the same point set, once one is computed the others can be obtained by a permutation, i.e. in quadratic time each. This is why quadratic time is spent for each of the h comparisons.

In 3D the canonical orderings were based on the directed edges of the convex hull (at most $6n - 12$ in total, which can be reported in $O(n \log n)$ time). For each such edge, a labeling of S could be obtained via a clockwise plane-sweep starting from a tangent plane containing the edge. The OTR corresponding to one of these labelings can be computed in $O(n^3 \log n)$ time, and each of the remaining takes cubic time, by a permutation on the first. So, two point sets can be compared, by in turn comparing an OTR of one to $\leq 6n-12$ OTRs of the other. Each such OTR comparison takes cubic time. Therefore two sets can be compared in $O(n^4)$ time.

Such sweeps can be extended to any fixed dimension d, according to [9]. There are $O(n^{\lfloor d/2 \rfloor})$ canonical orderings. Each corresponds to a "face flag", which can be thought of as a directed simplex. Each ordering can be computed in $O(n \log n)$ time once its face flag is known. Finding all face flags takes $O(n^d)$ time for general position; otherwise in $O(n^{(d(d+3)/2)})$ time.

For each ordering, an OTR is produced: the first in $O(n^d \log n)$ time, and the rest in $O(n^d)$ time. Therefore, for general position, it takes $O(n^{\lfloor 3d/2 \rfloor})$ time to compare two point sets (comparing $O(n^{\lfloor d/2 \rfloor})$ orderings, all of which are constructed in $O(n^{\lfloor 3d/2 \rfloor})$ time, and where each comparison takes $O(n^d)$ time). For non-general position the comparison is dominated by the time to compute all face flags: $O(n^{(d(d+3)/2)})$.

3 Order type invariant labeling in $2D$

We use "canonical" labelings, similarly to Goodman and Pollack [9]. Instead of using the convex hull and letting h represent its size, we set h to be the number of points on the convex layer c^* with smallest size. It takes $O(n \log n)$ time to compute all convex layers [3].

Assume for now that c^* contains more than one point. We select the starting vector for any point p on c^* to be the one through its clockwise neighbor on c^*. This is an order type invariant choice. Still, we can end up with $h = O(n)$ canonical labelings. For point sets with more than a constant number of convex layers, $h = o(n)$. For random point sets, h is sub-linear as well; see [5]. Note that the worst case is not the point set in convex position, for this is trivial to label (there are n possible distinct sequential labelings of a cycle, but they are all symmetric and combinatorially equivalent). It is worse to have a constant number of layers, each with a linear number of points.

Finally, if p is the only point on the smallest layer, then it is defined as the first point of our order type invariant labeling of S. We remove it from S and repeat. We will repeat at most once because even if we end up with a single point for a second time, the two points permit us to uniquely sort the rest.

The convex layer idea is simply a heuristic way of narrowing down the candidate labelings, from all permutations. The general idea is to apply any combinatorial test that easily ranks points, and keep only those points that have highest rank. If several points achieve the highest rank, we can apply another combinatorial test, and so on. So far, we have followed this idea by saying that the points of highest rank are those on a particular convex hull layer. Of course, we cannot apply an endless series of tests, for this will eventually become computationally expensive. The challenge is to select the most efficient tests possible. We have tried several combinations of heuristics (see full paper). Curiously, none so far have resulted in a worst-case sub-linear set of labelings, within $O(n^3)$ time.

By now we have chosen $O(h) = O(n)$ order type invariant labelings. We compute the OTR for each labeling $p'_1, \ldots, p'_n$, in quadratic time. Recall that this is a concatenation of $n(n-1)$ variables, each of which represents the number of points to the left of the vector from one point to another. The order of the variables is given by a lexicographic description of the vector index. This implicitly encodes any combinatorial differences missed by the simple geometric tests of the preceding section.

The rest is simple: we choose the largest lexicographic OTR as our solution. Since each OTR is a list of quadratic size, this takes $O(hn^2)$ time. If several lists tie for the highest rank, we choose arbitrarily among them. Clearly any of these lists will provide the same combinatorial output if given to a combinatorial algorithm (for instance, for triangulation). Furthermore, if we are given two point sets, comparing their respective lexicographically selected OTRs will determine if they have the same order type. In 2D this is just a variant of the method of Goodman and Pollack, described in Section 2. Instead of comparing h OTR's from one set to one arbitrary OTR of the other, we compare h OTRs among themselves for each set, and then compare the winners.

4 Labeling in $\mathbb{R}^3$

Recall that in $3D$ the order type of S depends on the spatial relation among every quadruple of points.

In $2D$ we used only those points on the smallest convex layer, to determine a subset of labelings. The nice thing about $2D$ is that it is easy to obtain one labeling per point. In $3D$ it still makes sense (although only heuristically) to use only those points on the smallest convex layer, so we begin by discarding all other points. The objective is to obtain a labeled (ordered) triple of points, in an order type invariant way. From these, it is possible to label all remaining points with a sweep.

From the smallest convex layer, we will iteratively keep removing points, and in fact we will be satisfied if we can reduce S to any constant number of candidate points. However we want at least three non-collinear points, or possibly two if they happen to be convex hull neighbors. As in $2D$, if we remove too many points, we can store the few remaining and re-iterate on the removed set. Even if we end up with a collinear triple after three iterations of discarding, it is easy to ensure that the next iteration will give a non-collinear point; simply exclude all points collinear to the triple from that iteration. So this part of the algorithm does not rely on general position.

Once we have our desired constant number of points, for every ordered non-collinear triple among them, we can carry out a uniquely directed plane sweep to relabel S. By [9], the OTR for each such labeling can be computed in $O(n^3 \log n)$ time. Again, the overall solution will be the lexicographically largest OTR. For a constant number of candidate points, the number of triples, and thus the number of OTRs to compare, is constant. Each representation has size $O(n^3)$, so it takes cubic time to choose lexicographically. Therefore the overall run-time is also $O(n^3 \log n)$. What remains is to show how to remove all but a constant number of points, in an order type invariant way.

Let L be the points of the smallest convex layer. Let $CH(L)$ denote the convex hull of L. The following simple procedure does just what we need; it identifies a constant number of points from S. Let $P = L$. Repeat the following steps until no points are removed from P in step 2.

1. Compute $CH(P)$. Remove all interior points from P, including non-extreme collinear points.
2. Remove from P all points except those that have the lowest vertex degree on $CH(P)$.

If more than three points survive, $CH(P)$ must be a polyhedron with all vertex degrees being equal. If S is in general position then every face of $CH(P)$ must be a triangle. It can be shown, using Euler's formula, that $CH(P)$ must be combinatorially equivalent to a tetrahedron, octahedron, or icosahedron. Each has only a constant number of vertices.

Step 1 takes $O(n \log n)$ time. Step 2 takes linear time since the sum of vertex degrees is linear. There are fewer than n iterations since at least one point is removed each time. Thus this procedure takes $O(n^2 \log n)$ time, which does not affect the time complexity dominated by OTR computations.

5 Comparing the order type of two sets

As mentioned, the method of Goodman and Pollack [9] compares order types of three-dimensional point sets in general position in $O(n^4)$ time. The evaluation is done by comparing the OTR for each canonical labeling of S_1 to one arbitrary OTR from S_2.

We have ongoing research to improve OTR representation and computation. For now, in Section 4 we have shown that the number of canonical labelings can be reduced to a constant. It costs $O(n^3 \log n)$ time to compute the OTR for each, and with a lexicographic sort we can select one representative OTR for any point set. Then S_1 and S_2 are compared by matching their OTRs.

6 Final notes

Given an order-type invariant labeling of S, we can solve the following problems in an order-type invariant way. Consider any problem involving building a graph by adding edges to S, based on combinatorial comparisons, i.e. based on order type. Examples of such problems include triangulation/tetrahedralization, polygonization, finding a non-crossing matching, finding a non-crossing spanning tree, or computing a halving line or a ham-sandwich cut. An order type invariant labeling can be constructed as preprocessing, so that any combinatorially equivalent input will produce the same output.

We ask whether an OTR can be computed in $o(n^3 \log n)$ time in $3D$, and whether any worst-case sub-cubic algorithm exists for the planar case. For the latter, we specifically ask whether a sub-linear number of canonical labelings can be computed in sub-cubic time. Of course, the other possibility also exists (if the number of labelings remains linear): that of reporting an OTR in sub-quadratic time. This is unlikely, as it implies the existence of an OTR that can be stored in sub-quadratic space. Whether this can be done was an open problem posed in [8].

Note that without general position, in $3D$ it is possible to follow the vertex-degree pruning step with another test for each vertex. That is, we can enumerate the size of surrounding faces, and keep only those vertices that qualify lexicographically. This would result in a polyhedron where every vertex appears identical, with respect to degree and surrounding face sizes. A topological proof of Archimedes' theorem (see [11, 12]) tells us that such a polyhedron is isomorphic to a semi-regular polyhedron (i.e., one of the Platonic or Archimedean solids, prisms, or antiprisms), or to the pseudorhombicuboctahedron. Among these, only the prisms and antiprisms may have more than constant size, and this is why general position is assumed. Our ongoing work involves resolving the case where pruning results in a prism of more than constant size (and where repetition of our algorithm on the discarded set of points recursively yields such prisms as well).

We thank the other participants of the 2011 Mid-Winter Workshop on Computational Geometry for providing a stimulating research environment.

References

[1] O. Aichholzer, F. Aurenhammer, and H. Krasser. Enumerating order types for small point sets with applications. *Order*, 19(3):265–281, 2002.

[2] O. Aichholzer and H. Krasser. Abstract order type extension and new results on the rectilinear crossing number. In *Symposium on Computational Geometry*, pages 91–98, 2005.

[3] B. Chazelle. On the convex layers of a planar point set. *IEEE Transactions on Information Theory*, 31(4):509–517, 1985.

[4] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. In *IEEE Symposium on Foundations of Computer Science*, pages 217–225. IEEE, 1983.

[5] K. Dalal. Counting the onion. Master's thesis, McGill University, 2004.

[6] H. Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal of Computing*, 15(2):341–363, 1986.

[7] J. Goodman and R. Pollack. Upper bounds for configurations and polytopes in R^d. *Discrete & Computational Geometry*, 1:219–227, 1986.

[8] J. Goodman and R. Pollack. The complexity of point configurations. *Discrete Applied Mathematics*, 31:167–180, 1991.

[9] J. E. Goodman and R. Pollack. Multidimensional sorting. *SIAM Journal on Computing*, 12(3):484–507, 1983.

[10] J. E. Goodman, R. Pollack, and B. Sturmfels. Coordinate representation of order types requires exponential storage. In *ACM Symposium on Theory of Computing*, 1989.

[11] M. Villarino. On the Archimedean or semiregular polyhedra. Technical Report arXiv:math/0505488v1, 2005.

[12] T. Walsh. Characterizing the vertex neighbourhoods of semi-regular polyhedra. *Geometriae Dedicata*, 1:117–123, 1972.