Testing the congruence of d-dimensional point sets

Peter Braß* and Christian Knauer*

Revision: 1.7

23. November 1999

Abstract

This paper presents an algorithm that tests the congruence of two sets of n points in d-dimensional space in $O(n^{\lceil \frac{1}{3}d \rceil} \log n)$ time. This improves the previous best algorithm for dimensions $d \geq 6$.

Keywords: Computational geometry, Shape matching, Congruence, Symmetry.

1 Introduction

Geometric pattern matching problems have been studied by many papers in computational geometry, see [9] survey. The simplest pattern matching problem is to decide whether two patterns are actually the same, in the model considered here: whether two given point sets are congruent. This problem is well-understood in dimensions at most three, where there are good algorithms which reach the asymptotic lower bound of $\Omega(n \log n)$ [2, 3, 4, 10, 11, 13]. For higher dimensions, however, the situation is different: only dimension reduction methods are known, which lead to an $O(n^{d-2} \log n)$ algorithm by Alt et al. [2], a Monte-Carlo $O(n^{\lfloor \frac{1}{2}d \rfloor} \log n)$ algorithm by Akutsu [1], a deterministic algorithm of the same complexity by Matousek [1], and the $O(n^{\lceil \frac{1}{3}d \rceil} \log n)$ algorithm described below. The exponential dependence on d is rather unsatisfactory, since the only lower bound is the $\Omega(n \log n)$ bound which already holds in dimension one. Although some dimension-dependence is unavoidable, since the congruence testing problem without dimension restriction is at least as difficult as graph isomorphism, the correct asymptotics is probably $O(n \log n)$ for fixed dimension d, with a multiplicative constant exponentially dependent on d. This large gap makes it an interesting problem for further work, especially since it seems related to characterization problems of highly symmetric point sets, which are an important research area in connection with aperiodic tilings.

2 Previous algorithms

Given two point sets A and B, of n points each, in $\mathbb{R}^d$, we want to test whether a congruence exists that maps A on B. This congruence consists of two parts: a translation and a rotation. The translational part is easy to determine, since the image of the centroid $c(A)$ of A under that congruence has to be the centroid $c(B)$ of B, and the centroid can be determined fast (in $O(n)$). So each algorithm preprocesses the sets by translating A

*Institut für Informatik, Freie Universität Berlin, Takustraße 9, D–14195 Berlin, Germany. {brass, knauer}@inf.fu-berlin.de

to $A - c(A)$ and B to $B - c(B)$, and searches for a rotation around 0 that maps one set on the other. Also the distances of the points to the centroid have to be preserved, so we can replace each point by a point on the unit sphere which carries the distance (or an ordered list of distances) of the point in that direction as a label. So in the following we have two sets A', B', each consisting of n labeled points (counting multiplicities) on the unit sphere with center 0, and we are looking for a label-preserving rotation that maps A' on B'.

In the two-dimensional case it is easy to reduce this problem to the classical substring matching problem: The points of A' are cyclically ordered on the circle, and described by the pair of their label and the angle to the next point on the circle (which together is just some symbol in an alphabet). So starting at an arbitrary point and going around the circle once, we can encode A' as a string in that alphabet; and going around the circle twice for B', we encode that set in another string. Then A and B are congruent if and only if the string of A' is a substring of the string for B', which can be tested in $O(n \log n)$ time.

The algorithm of Alt et al. [2] for the threedimensional case (similar also Sugihara [13]) involves again the creation of a combinatorial model and solution of the problem on that: Each of our sets consists of n labeled points on the unit sphere, so their convex hull is some polyhedron, which we can describe by its edge graph augmented by edge labels carrying the length of the edge and the angular distance to the next edge around that vertex. But the isomorphism of (labeled) polyhedral graphs (threeconnected and planar) can be tested in $O(n \log n)$ time.

For higher dimensions no such direct method is known, but it is possible to reduce one higher-dimensional problem to a number of alternative lower-dimensional ones. The underlying idea is that if we know the correct image point $b \in B'$ for some point $a \in A'$, then the subspace through a is mapped on the subspace through b and the same holds for their orthogonal complements. Thus we can orthogonally decompose each point of A' and each point of B' into a pair of points (one on a line, one on the orthogonal hyperplane), and these have to be mapped on each other by the congruence. But the point on the line is uniquely identified by its signed distance to 0 (a, b positive), so we can just append this number to the label of the point on the hyperplane (additionally to all previous labels of the original point), and have reduced the original problem to a problem of points with longer labels which lie in a hyperplane. Thus if we know the correct images of k linearly independent points, we can reduce the dimension of the space by k, appending k additional labels (coordinates) to each point.

If we do not know the correct image of any point, the simplest way is to take a fixed $a \in A'$ and try each of the potential image points $b \in B'$ in turn: if we are successful in one of the lower-dimensional problems, we can extend the solution to a solution of the original problem (by removing that additional label and adding a (or b) times that label to each point); if we are not successful for any choice of b, then the sets A and B are not congruent. This is the algorithm of Alt et al. [2] which runs in time $O(n^{d-2} \log n)$, reducing one d-dimensional problem of n points to n alternative $d{-}1$-dimensional problems in each step.

If we use the fact that any closest pair in A' has to be mapped on a closest pair in B', and that the number of closest pairs is only linear in n (for fixed dimension d: since the degree of the graph of closest pairs is bounded by a constant, the 'kissing number' or 'Newton number' which grows exponentially in d [12, 14]), we can reduce the problem in each step by two dimensions to $O(n)$ alternative $d-2$-dimensional problems. This is the algorithm of Matousek [1] which runs in $O(n^{\lfloor \frac{1}{2} d \rfloor} \log n)$ time.

At this point a difficulty arises which is caused by the inner symmetries of our point set: since the point-k-tuples which define the alternative reductions are themselves defined by metric properties, the set of all these alternative reductions will be closed under inner isometries of the point set. But for dimension $d \geq 4$ this set can be quite big, since rotation subgroups in orthogonal planes are possible. So e.g. in dimension four the set consisting of two regular $\frac{n}{2}$-gons with common center 0, but in orthogonal planes, allows $\frac{n^2}{4}$ rotations, and any fulldimensional triple of points (that could be used to reduce the dimension by three) will generate an orbit of $\Omega(n^2)$ other possible reductions. Similar constructions work also in higher dimensions. This big number of alternative reductions is not really needed, since they all give the same result, but they have to be recognized.

3 The new algorithm

Our solution to this problem is to recognize whenever the point set lies in orthogonal subspaces (which can be matched independently), and use an extended version of the smallest distance graph in the other cases, in such a way that we get a dimension reduction of three with a linear number of alternatives in each step.

For this purpose we add for each point $a \in A'$ the antipodal point $-a$ labeled as 'new', if it is not already in the set A' (in the following always the same for B'). Then the smallest distance occuring between two antipodal pairs $p, -p, q, -q$ on the unit sphere is at most $\sqrt{2}$, and $\sqrt{2}$ is reached if and only if p is orthogonal to q. We can extend this from two antipodal pairs to larger point sets by constructing a graph with the extended set as vertices, which has in the beginning the antipodal pairs as connected components. Any graph containing this graph as subgraph has the property that the smallest distance occuring between points of distinct connected components is at most $\sqrt{2}$, and reaches this value if and only if the connected components lie in orthogonal subspaces. So we start from the antipodal pairs graph (in the following, edges between two antipodal points will be called *antipodal edges*, whereas the remaining edges will be called *strong edges*; two points sharing such an edge will be called *strongly adjacent* or *strong neighbours*), and extend in each step the graph by all those edges between points of distinct components that are of minimum length among all such point pairs. Then we either find after some steps a connected component which spans a subspace of dimension at least three, which allows a dimension reduction, or we find that all the connected components lie in orthogonal subspaces, which can be treated independently. The algorithm is show in figure 1.

Note that — for the sake of simplicity — we omitted several sanity checks that are performed in the course of the algorithm, e.g. testing whether the two graphs have the same number of connected components all the time, etc. If any of these checks fail, the algorithm gives a negative answer.

3.1 Correctness

The correctness of our algorithm is obvious if $d \leq 3$; also the claimed time bound follows immediately in that case. To prove the correctness as well as the time complexity for the general case, we will show that the following invariants hold throughout the algorithm, just before $d_{\min}$ is recomputed and edges are added (and components are merged) as long as no dimension reduction is found and $d_{\min}$ is smaller than $\sqrt{2}$:

1. Each connected component is planar, i.e. it consists of points distributed on a great cycle of the unit sphere.
2. Each point has at most two strong neighbours in its component and one antipodal neighbour; i.e. a connected component consists of two paths of the same length (i.e. number of vertices) and each vertex of a path is connected to its antipodal vertex on the other path via an antipodal edge.

Algorithm *Congruencetest*(A, B, d)
Input: Two sets A, B, of n labeled points each, in $\mathbb{R}^d$.
Output: Decides whether there is a labelpreserving congruence that maps A on B.

$\triangleright$If the dimension is at most three, use one of the known $O(n \log n)$-algorithms to decide the problem.
$\triangleright$Preprocess A and B to A' and B' by moving the centroid to 0 and projecting each point on the unit sphere around 0, with the projection distance appended as an additional label.
$\triangleright$Construct the extended sets A'' and B'' by adding for each point $a \in A'$ and each point $b \in B'$ the antipodal points $-a$, $-b$, labeled as 'new', if they were not already contained in the set. Construct the initial graphs for both sets by joining each point to its antipodal point.
repeat
    if there is only one component left
      then (* $\dim A'' = \dim B'' = 2$ *)
          $\triangleright$Compute for A'' and B'' the coordinates of the points in the linearly subspaces spanned by the sets, and apply one of the known $O(n \log n)$-algorithms to decide the twodimensional problem.
      else
          $\triangleright$Determine the smallest distance $d_{\min}$ between two points of distinct connected components.
          if $d_{\min} = \sqrt{2}$
            then
                $\triangleright$Decompose each A'' and B'' in their connected components, computing the coordinates of the points in the linear subspaces spanned by the components.
                for each matching of the A''-components to the B''-components
                  do
                    for each component pair (α, β) in the matching
                      do
                        $\triangleright$Apply one of the known $O(n \log n)$-algorithms for the twodimensional case to decide whether α and β are congruent.
                if there is a matching such that each matched pair is congruent
                  then
                      A and B are congruent
                  else
                      they are not congruent.
            else (* $d_{\min} < \sqrt{2}$ *)
                $\triangleright$Add all edges of length $d_{\min}$ that join points in distinct connected components to the graph.
                if there is a triple $(\text{point}, \text{neighbour}_1, \text{neighbour}_2)$ of A'' that spans a linear subspace of dimension three
                  then
                      for all triples $(\text{point}, \text{neighbour}_1, \text{neighbour}_2)$ of B'' that span a linear subspace of dimension three
                        do
                          $\triangleright$Perform the dimension reduction, and call recursively *Congruencetest* for the $d-3$-dimensional problem.
                      if one of the potential image triples from B'' gives congruent sets
                        then
                            A and B are congruent
                        else
                            they are not congruent.
until a reduction is found, or the set is decomposed in orthogonal subproblems.

Figure 1: Algorithm *Congruencetest*

The proof proceeds by induction: In the beginning the invariants obviously do hold. Now assume that each connected component is planar, $d_{\min}$ is computed, $d_{\min} < \sqrt{2}$ and we add all minimum length edges between vertices in distinct connected components. We show that either a dimension reduction is found (i.e. the graph now contains three points spanning a threedimensional subspace) or the invariants also do hold after the merging step.

We will look at two connected components c_1 and c_2 that are merged in this step[1]. The two great circles C_1 and C_2 containing these components either do coincide or they share exactly two vertices.

If they coincide, then the resulting point set is clearly planar. Since the minimal distance between the two merged components is larger than the distance between any two strongly adjacent vertices, the new edges can only join vertices of degree one. Therefore the maximal degree remains two.

If the great circles differ, we will either establish a strong adjacency between c_1 and a point $p \in c_2 - C_1$ or all the points in c_2 minimizing the distance to c_1 do lie on C_1. In the first case, the new graph contains three points spanning a three dimensional subspace. In the second case the new edges join the points $\{p, -p\} = C_2 \cap C_1 \subseteq c_2$ to c_1. Now either $c_2 = \{p, -p\}$ and we can argue as before to see that the resulting point set is planar and the maximal degree remains two[2], or $\#c_2 > 2$ and p has at least one strong neighbour $q \in c_2$, since we assumed that $C_2 \neq C_1$. In that case it will span a threedimensional subspace together with q and its new neighbour from c_1.

The correctness of the algorithm immediately follows from these invariants. If we find a dimension reduction at some point, then the correctness follows by induction over the dimension, as was argued in the introduction. If we end up with a single connected component, the problem is actually only twodimensional, and if we end up with a set of connected components of pairwise distance $\sqrt{2}$ we have decomposed the original problem into twodimensional orthogonal subproblems that can be solved independently.

3.2 Analysis

To facilitate the analysis of the algorithm, we have to specify the implementation in some more detail. We store the edges of the complete graph on the point set (except the edges connecting antipodal vertices) in a minimum heap $\mathcal{H}$, using the distance between two points as the key. The initialization time for this structure is $O(n^2 \log n)$. Furthermore we use a union-find data structure $\mathcal{C}$ to store the points in each connected component. This structure is initialized with n sets, each containing two points (namely a point along with its antipodal point, i.e. $\mathcal{C} = \{C_p | p \in A\}$ and $C_p = \{p, -p\}$); this initial step takes $O(n)$ time. Finally we keep the $n \times n$ adjacency matrix G of the smallest distance graph. It can be set up in $O(n^2)$ time as $G_{p,q} = [q = -p]$.

Throughout the algorithm C_p will contain exactly those points connected to p, and $\mathcal{H}$ will contain all non-edges of length more than d_{min}, i.e. $e = (p,q) \in \mathcal{H} \iff G_{p,q} = 0$ and $|p - q| > d_{min}$.

In order to determine the new minimal intercomponent distance in step 9., we find the smallest key of $\mathcal{H}$. This can be done in $O(\log n)$ time. If $d_{min} = \sqrt{2}$ we have decomposed the original problem in orthogonal and our data structures need not be updated anymore. Otherwise we have to make sure that our invariants do hold in the next step. To do so, we extract all edges of length d_{min} from $\mathcal{H}$ and store them in a list E_{min}. If l denotes the length of this list, the time for this step is bounded by $O(l \log n)$. For all

[1]Note that due to the antipodal vertices, the process is symmetric, i.e. if vertex u becomes adjacent to vertex v in the merging process, then also the antipodal vertices u' and v' become adjacent in the same step.

[2]If $\#c_2 = 2$, the great circle containing c_2 is not uniquely determined. Since we assumed $c_2 \subseteq C_1$ we can take $C_2 = C_1$ in that case.

edges $(p,q) \in E_{min}$ we first check whether $C_p = C_q$, i.e. if they join points in the same connected component *before edges are added.* If not, we make them adjacent in G. After that we update $\mathcal{C}$ by merging two different components in case we just added edges between them. The procedure we just described requires at most l find queries to $\mathcal{C}$ and at most l union operations on $\mathcal{C}$, so the overall time required is of order $O(l \log^* n)$.

Now, although l can be as large as $O(n)$ (see below), we see that the amortized cost for determining the minimal intercomponent distances in step 9. and maintaining the data structures $\mathcal{H}$ and $\mathcal{C}$ in step 24. is of order $O(n^2 \log n)$, since each edge is 'touched' at most once in the course of the algorithm.

The task of projecting a set of points to the linear subspace it spans and computing a basis for that space, along with the appropriate coordinate computation can be done in linear time.

The parts of the algorithm that handle the case $d \le 3$ (step 1.), the case of all points lying on plane (step 7.) and the case of orthogonal subproblems (steps 12.–22.), respectively, are called at most once during the execution of the algorithm. Step 1. as well as step 7. takes $O(n \log n)$ time. In steps 12.–22. there are at most d components in each set, since the components are mutually orthogonal, so we have to try at most $d! = O(1)$ possibilities for the matching, and for each matching we have to solve at most $d = O(1)$ twodimensional subproblems; the total time required for these steps is therefore $O(n \log n)$, too.

The repeat loop is executed $O(n^2)$ times and in each step $O(n)$ edges are added (see below). But since each edge will be used only once in this process, the overall time spent for finding minimum length edges and merging components is bounded by $O(n^2 \log n)$, as argued above.

The number of recursive calls of the algorithm in step 34. is bounded by the number of triples in the nearest neighbour graph. To prove the $O(n^{\lceil \frac{1}{3} d \rceil} \log n)$ time bound, we have to show that not too many recursive calls are generated. To be more precise, we prove that steps 24.–34. generate only $O(n)$ possible reductions. For this we look at the last time that $d_{\min}$ was recomputed and edges were added. We know that before the edges were added, each connected component was planar, i.e. the points of the component were distributed on a great cycle of the unit sphere, and each point had at most two strong neighbours in its component. Now consider the graph that has the connected components as its vertices and that has an edge between two components iff they are merged in this step. This is a smallest distance graph, and therefore its degree is bounded by the kissing number k_d of dimension d, which is less than $3^d = O(1)$. This implies that a vertex from a component can only have new neighbours in at most $k_d = O(1)$ components. Furthermore it has at most two neighbours in each of these components, since a sphere around that point intersects the great circle containing the other connected component in at most two points, unless the center of the sphere is in the subspace orthogonal to the plane spanned by the circle. But this is not possible, since $d_{\min} = \sqrt{2}$ in that case, which we excluded in steps 12.–22. We see that at most $O(n)$ edges were added in this merging step and that each vertex has degree at most $O(1)$ and therefore the number of triples in the nearest neighbour graph is bounded by $O(n)$.

The time $T(n,d)$ that algorithm *Congruencetest* needs to decide the congruence of a d-dimensional n-point set, can be estimated as $T(n,d) \le O(n^2 \log n) + O(n)T(n,d-3)$, where the first term accounts for the time spent in the loop and for the code parts that handle the initialization and the base cases. We see that $T(n,d) = O(n^{\lceil d/3 \rceil} \log n)$, which proves our claim.

4 Final Remarks

We presented a new dimension reduction technique which allowed to test the congruence of two d-dimensional n-point sets in $O(n^{\lceil \frac{1}{3}d \rceil} \log n)$ time, essentially by factoring out possible symmetries which otherwise prevent dimension reduction beyond two by causing many spurious alternative reductions. To reach a dimension reduction of four by the same method, one will have to overcome the problem that the degree in the constructed graph is not anymore bounded by a constant, and we do not doubt that a dimension reduction by five, although possible, will present further technical difficulties. So although the algorithm presented here is the currently fastest, we do not believe that a refinement of the dimension reduction techniques will lead to what we ultimately believe possible, an $O(n \log n)$-algorithm for each fixed dimension d.

Instead we suggest further attention to a different line of attack, which is based on the reduction of the underlying point set, as it was used in the three-dimensional algorithm by Atkinson [4]. For if we can distinguish different 'sorts' of points, based on some local criterion, so that any point of A has to be matched to a point of the same sort of B, when we can replace A and B by subsets of at most half as many points which have to be matched to each other, and whose matching we can extend to the original sets. Repeating this procedure, we finally stop with sets which are undistinguishable by our local rule. If we could show for some easy criterion (that can be checked in $O(n \log n)$) that this local undistinguishability already implies a global transitive symmetry (with at most some simply classified exceptions), then we had an $O(n \log n)$ algorithm for testing congruence in that dimension. This same problem, criteria for 'local' symmetry implying global transitive symmetry, is also very interesting in the study of geometric crystallography and aperiodic tilings [7], but the criterion given in [6] can unfortunately not be applied here. Counterexamples, i.e. sets on the sphere in which some big neighbourhood of each point looks the same, but the set is not transitively symmetric, would be a kind of 'spherical aperiodic tiling' which is not known up to now. This is again probably a difficult problem, since even a threedimensional version, whether it is possible to tile a sphere by congruent pieces of arbitrarily small diameter, is an old open problem of Ruziewicz (see [5], problem C8). Another related open problem is the classification of metrically homogeneous sets [8], i.e. sets in which for each point the same multiset of distances to other points arises: this would imply the existence of an $O(n^2 \log n)$ time algorithm, but this is only known in dimension two where metrically homogeneous sets are sets with a transitive symmetry. So this problem is related to a number of interesting open problems in discrete geometry.

References

[1] T. Akutsu. On determining the congruence of point sets in d dimensions. *Comput. Geom. Theory Appl.*, 9(4):247–256, 1998.

[2] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, 3:237–256, 1988.

[3] M. J. Atallah. On symmetry detection. *IEEE Trans. Comput.*, 34:663–666, 1985.

[4] M. D. Atkinson. An optimal algorithm for geometrical congruence. *J. Algorithms*, 8:159–172, 1987.

[5] H. P. Croft, K. J. Falconer, and R. K. Guy. *Unsolved Problems in Geometry*. Springer-Verlag, 1991.

[6] B. Delone, N. Dolbilin, M. Stogrin, and R. Galiulin. A local criterion for regularity of a system of points. *Sov. Math. Dokl*, 17:319–322, 1976.

[7] N. Dolbilin, J. Lagarias, and M. Senechal. Multiregular point systems. *Discrete Comput. Geom.*, 20:477–498, 1998.

[8] B. Grünbaum and L. Kelly. Metrically homogeneous sets. *Israel J. Math.*, 6:183–197, 1968. Corrigendum in *Israel J. Math.*, 8:93–95, 1970.

[9] L. J. Guibas and H. Alt. Discrete geometric shapes: Matching, interpolation, and approximation. Technical Report Report B 96-11, Institut für Informatik, Freie Universität Berlin, 1996.

[10] P. T. Highnam. Optimal algorithms for finding the symmetries of a planar point set. *Inform. Process. Lett.*, 22:219–222, 1986.

[11] G. K. Manacher. An application of pattern matching to a problem in geometrical complexity. *Inform. Process. Lett.*, 5:6–7, 1976.

[12] C. Shannon. Probability of error for optimal codes in a gaussian channel. *Bell System Tech. J.*, 38:611–656, 1959.

[13] K. Sugihara. An $n \log n$ algorithm for determining the congruity of polyhedra. *J. Comput. Syst. Sci.*, 29:36–47, 1984.

[14] A. D. Wyner. Capabilities of bounded discrepancy decoding. *AT&T Tech. J.*, 44:1061–1122, 1965.