Testing the congruence of d-dimensional point sets

Extended Abstract

Peter Braß* and Christian Knauer*

Revision: 1.7

30. November 1999

1 Introduction

Geometric pattern matching problems have been studied by many papers in computational geometry, see [5] for a 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, 6, 7, 9]. 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.

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 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 and a similar approach works for the threedimensional case; the algorithm of Alt et al. [2] involves again the creation of a combinatorial model and solution of the problem on that. In both cases congruence can be tested in $O(n \log n)$ time.

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

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 [8, 10]), 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 $\sqrt{2}$,

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.

1. $\triangleright$If the dimension is at most three, use one of the known $O(n \log n)$-algorithms to decide the problem.
2. $\triangleright$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.
3. $\triangleright$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.
4. **repeat**
5. **if** there is only one component left
6. **then** (* $\dim A'' = \dim B'' = 2$ *)
7. $\triangleright$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.
8. **else**
9. $\triangleright$Determine the smallest distance $d_{\min}$ between two points of distinct connected components.
10. **if** $d_{\min} = \sqrt{2}$
11. **then**
12. $\triangleright$Decompose each A'' and B'' in their connected components, computing the coordinates of the points in the linear subspaces spanned by the components.
13. **for** each matching of the A''-components to the B''-components
14. **do**
15. **for** each component pair (α, β) in the matching
16. **do**
17. $\triangleright$Apply one of the known $O(n \log n)$-algorithms for the twodimensional case to decide whether α and β are congruent.
18. **if** there is a matching such that each matched pair is congruent
19. **then**
20. A and B are congruent
21. **else**
22. they are not congruent.
23. **else** (* $d_{\min} < \sqrt{2}$ *)
24. $\triangleright$Add all edges of length $d_{\min}$ that join points in distinct connected components to the graph.
25. **if** there is a triple $(\text{point}, \text{neighbour}_1, \text{neighbour}_2)$ of A'' that spans a linear subspace of dimension three
26. **then**
27. **for** all triples $(\text{point}, \text{neighbour}_1, \text{neighbour}_2)$ of B'' that span a linear subspace of dimension three
28. **do**
29. $\triangleright$Perform the dimension reduction, and call recursively *Congruencetest* for the $d-3$-dimensional problem.
30. **if** one of the potential image triples from B'' gives congruent sets
31. **then**
32. A and B are congruent
33. **else**
34. they are not congruent.
35. **until** a reduction is found, or the set is decomposed in orthogonal subproblems.

Figure 1: Algorithm *Congruencetest*

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.

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] 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.

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

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

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

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

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