Discrete Comput Geom 3:237-256 (1988)

Discrete & Computational Geometry


# Congruence, Similarity, and Symmetries of Geometric Objects*

Helmut Alt,[1] Kurt Mehlhorn,[2] Hubert Wagener,[3] and Emo Welzl[1,†]

[1] Fachbereich Mathematik, Freie Universität Berlin, Arnimallee 2-6, D-1000 Berlin 33, Federal Republic of Germany

[2] Fachbereich Informatik, Universität des Saarlandes, D-6600 Saarbrücken, Federal Republic of Germany

[3] Fachbereich Informatik, Technische Universität Berlin, Franklinstrasse 28/29, D-1000 Berlin 10, Federal Republic of Germany

**Abstract.** We consider the problem of computing geometric transformations (rotation, translation, reflexion) that map a point set $A$ exactly or approximately into a point set $B$. We derive efficient algorithms for various cases (Euclidean or maximum metric, translation or rotation, or general congruence).

## 1. Introduction

We consider the problem of deciding whether two geometric objects $A$, $B$ in $\mathbb{R}^d$ are congruent or similar and, if they are, of finding the geometric transformation (rotation, translation, reflection, stretch) that maps $B$ into $A$. Important practical applications are, for example, determining how to move an object from one given position into another one (robotics), recognition of patterns and of written text, and determining speed and direction of moving objects out of pictures taken at different times. The second problem we consider is finding all the symmetries of a given geometric object, i.e., its symmetry group. Important applications are within pattern recognition and some areas of biology and crystallography.

We consider only the basic case of $A$ and $B$ consisting of $n$ points in $\mathbb{R}^d$. In many cases the algorithms presented here can be extended to more general geometric objects, where the points are connected by lines, curves, or surfaces described, for example, by algebraic equations (see also [Ata1] and [Ata2]).

By a straightforward reduction of the set equality problem of real numbers it can be shown that each of the problems mentioned above has an $\Omega(n \log n)$

* Part of this research was supported by the DFG under Grants Me 620/6-1 and Al 253/1-1.

† Research associate of IIG, Institutes for Information Processing, Technical University of Graz, Austria.

lower bound even in the one-dimensional case [Atk], [H]. Optimal algorithms in the two-dimensional case have been found by Highnam [H] for symmetry and by Atallah [Ata1] for congruence, and in the three-dimensional case by Atkinson [Atk] for congruence. Here we first give an alternative optimal algorithm for three-dimensional congruence which then can be used to find the symmetry group of three-dimensional point sets in $O(n \log n)$ time. Then we extend the result to get $O(n^{d-2} \log n)$-time congruence algorithms for arbitrary dimension $d \geq 3$. These results are contained in Section 2.

Unfortunately, the problem of deciding congruence, similarity, and symmetry exactly is not realistic in practice. In fact, small perturbations in the input data, e.g., due to inaccuracies in physical measurements, will usually destroy these properties. We therefore introduce the notion of *approximate congruence*, i.e., congruence within a certain tolerance $\varepsilon > 0$. The problem of approximate congruence will be dealt with in Section 3.

Our model of computation is the Random Access Machine. In Section 2 we assume that the machine can represent arbitrary real numbers and can perform all the geometric computations involved (e.g., determining angles, distances, etc.) exactly without roundoff-errors. Since Section 2 is of mainly theoretical interest anyway, we feel that this standpoint, which is customary in computational geometry, is justified. In Section 3, on the approximate congruence, we are more careful. We only assume that the machines can represent integers of a length of at most $L$ bits, where $L$ is some constant, and can perform additions, subtractions, and multiplications on such integers in constant time. We refer to integers of length of at most $L$ bits as single precision integers. The input data are given in the form $p/q$ where $p$ and $q$ are single precision integers. All intermediate results of our algorithms result from constantly many applications of arithmetic operations to the input data and can without loss of generality therefore also be considered as single precision integers. Note that $L$ is treated as a constant and does not show up in the running time bounds. A thorough discussion of the case of variable $L$ can be found in [Sch].

We close with some notation which will be relevant for Sections 2 and 3. A *congruence* (isometric mapping) $M$ of $\mathbb{R}^d$ is any mapping which preserves Euclidean distances. Any congruence $M$ can be written in the form $M: x \mapsto Ax + t$ where $A$ is a $d \times d$ orthonormal real matrix, i.e., $A^T = A^{-1}$, and $t$ is any $d$-vector. A congruence is called of the *first* (*second*) *kind* if $\det A = +1$ $(-1)$. Any congruence $M$ of the second kind can be written as $x \mapsto AJx + t$ where $J(x_1, \ldots, x_d) = (-x_1, x_2, \ldots, x_d)$, $A^T = A^{-1}$ and $\det A = +1$. A mapping $x \mapsto \lambda x$ for some $\lambda \neq 0$ is called a *dilation* (*stretching*). A *similarity* is a congruence followed by a dilation. Two point sets $A$ and $B$ are congruent (similar) if there is a congruence (similarity) $M$ mapping $A$ onto $B$. The centroid $c_A$ (center of mass) of a finite point set $A$ is defined by $c_A = (\sum_{a \in A} a)/|A|$. Clearly, any similarity between $A$ and $B$ must map $c_A$ onto $c_B$.

## 2. The Exact Case

We first observe that the similarity problem for $n$-point sets in any dimension is reducible to the congruence problem in $O(n)$ time. In fact, in $O(n)$ time it is

possible to determine the centroids $c_A$, $c_B$ of the input sets $A$, $B$ and the maximum distances $m_A$, $m_B$ of $c_A$, $c_B$ to the points of $A$, $B$, respectively. Then $m_A/m_B$ is the factor by which $B$ is "stretched" around $c_B$. Clearly, the set $B'$ obtained this way is congruent to $A$ exactly if $B$ is similar to $A$. Therefore from now on we will not mention similarity anymore, but any result on congruence also holds for similarity.

These results are summarized in the following theorem:

**Theorem 1.**

(a) *For any $d \geq 3$ the congruence of two $n$-point sets in $R^d$ can be decided in $O(n^{d-2} \log n)$ time.*

(b) *The symmetry group of an $n$-point set in $\mathbb{R}^3$ can be determined in $O(n \log n)$ time.*

*Proof of* (a) (*Congruence*). In [Atk] an $O(n \log n)$ algorithm for $d = 3$ is given. We give an alternative one which can be applied to prove (b).

Given input sets $A$, $B$ our algorithm for determining a congruence of the first kind consists of the following steps; a congruence of the second kind can be determined by applying the algorithm to $A$ and $J(B)$:

**Algorithm 1.**

*Step 1.* Determine the centroids $c_A$, $c_B$. If $c_A \in A$, then check if $c_B \in B$. If not, give a negative answer, otherwise remove $c_A$, $c_B$ from $A$, $B$, respectively.

*Step 2.* Project all points of $A$ ($B$) onto the unit sphere using $c_A$ ($c_B$) as an origin. Mark the points in the sets $A'$ ($B'$) obtained this way with the distances of the original points from $c_A$ ($c_B$). Observe, that one point of the sphere can be the image of several points of $A$ ($B$) and, thus, be marked with several distances. In this case, sort these distances.

*Step 3.* Construct the convex hulls of $A'$, $B'$. Observe that all points in $A'$, $B'$ are extreme and therefore vertices of the convex hulls. Let $v$ be any such vertex, and let $v_0, v_1, \ldots, v_{k-1}$ be the vertices adjacent to $v$ listed in clockwise fashion about $v$. For $0 \leq i \leq k-1$ let $l_i$ be the length of the arc from $v$ to $v_i$ and $\varphi_i$ the angle on the sphere between $v_i$, $v$, and $v_{(i+1) \bmod k}$ (see Fig. 1). In addition to the

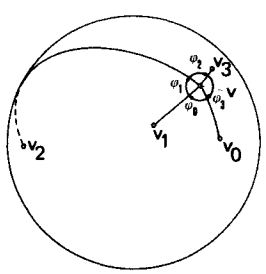



**Fig. 1**

information attached in step 2 mark $v$ with the lexicographically smallest cyclic shift of the sequence $(\varphi_0, l_0), \ldots, (\varphi_{k-1}, l_{k-1})$.

*Step 4.* The convex hulls together with the labels attached in steps 2 and 3 can be considered as labeled planar graphs [PS]. It is easy to see that the two input sets are exactly congruent of the first kind if the two labeled graphs are isomorphic. We decide this isomorphism using the $O(n \log n)$ partitioning algorithm of Hopcroft [AHU, Section 4.13] as follows. Consider a finite automaton whose states are the directed edges (each undirected edge of the planar graph gives rise to two directed edges) of the planar graph and which has input alphabet $\{a, b\}$. If edge $(v, w)$ is the current state and the input symbol is $a$ $(b)$ then the next state is $(v, z)$ where edge $(v, z)$ follows the edge $(v, w)$ in the clockwise order of edges about $v$ ($(z, w)$ where $(z, w)$ follows the edge $(v, w)$ in the clockwise order of edges about $w$). Finally, let $P$ be the following partition of the edges. Two edges $(v, w)$ and $(z, y)$ belong to the same block of $P$ iff $v$ and $x$ are labeled the same and $w$ and $y$ are labeled the same. The partitioning algorithm computes the coarsest refinement of $P$ which is compatible with the state transitions of the automaton. This partition is clearly identical to the automorphism partition of the union of the two graphs. Consequently the two graphs are exactly isomorphic if one of the classes in the partition contains edges of both.

*Analysis.* Step 1 clearly can be done in $O(n)$ time. Constructing the sets $A'$, $B'$ in step 2 takes time $O(n)$. However, sorting of the labels attached to one point may take time $O(n \log n)$. The construction of the convex hull in step 3 can be done in $O(n \log n)$ time [PS] [E].

For each vertex $v$ sorting the adjacent vertices $v_0, \ldots, v_{k-1}$ in a clockwise fashion takes time $O(k \log k)$. Also, in step 3 the lexicographically smallest cyclic shift can be computed in time $O(k \log k)$ as follows. We first sort the pairs $(\varphi_i, l_i)$, $0 \leq i < k$, in increasing order and then replace each pair by its rank. This yields a word $w_0 w_1 \cdots w_{k-1}$ with $0 \leq w_i < k$. Let $I = \{i_0 < i_1 < \cdots\}$ be the set of occurrences of letter 0. We next consider the pairs $(i_0, i_1)$, $(i_2, i_3), \ldots$ in turn. For each pair, say $(i_{2l}, i_{2l+1})$, we compare the subwords of length $i_{2l+1} - i_{2l}$ starting in $i_{2l}$ and $i_{2l+1}$, respectively. If the subword starting at index $i_{2l}$ is not larger lexicographically than the subword starting at $i_{2l+1}$ then we delete $i_{2l+1}$ from $I$ and we delete $i_{2l}$ otherwise. Thus in time $O(k)$ we can reduce the size of $I$ by half and the time bound follows. However, the sum of the run times for all vertices does not exceed $O(n \log n)$, since the total number of adjacent vertices considered is twice the number of edges which is $O(n)$.

**Algorithm 2.** If $d \leq 3$ Algorithm 1 is applied, otherwise the $d$-dimensional problem is reduced to $n$ problems of dimension $(d-1)$ in the following way:

*Step 1.* Construct the labeled sets $A'$, $B'$ as in step 2 of algorithm 1.

*Step 2.* For some point $a \in A'$ intersect the $d$-dimensional unit sphere $S_A$ around $c_A$ with some hyperplane which orthogonally bisects the line segment $\overline{c_A a}$. The intersection will be some $(d-1)$-dimensional sphere $S$ on the surface of $S_A$. Project each point $x \in A' - \{a\}$ onto $S$ along the arc from $x$ to $a$ on the surface

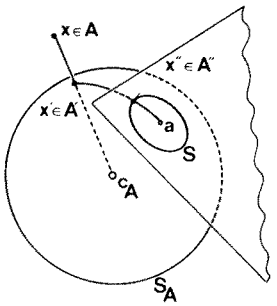



Fig. 2

of $S_A$. This yields a new set $A''$ (see Fig. 2). To each $x'' \in A''$ attach the following information: from each $x' \in A'$, which was projected onto $x''$, the label obtained in previous steps of the algorithm and, additionally, the length of the arc from $a$ to $x'$. If there are several points in $A'$ which are projected onto $x''$, list their labels sorted with respect to the lengths of the arcs. $A''$ still contains all the geometric information necessary to identify the original set $A$.

*Step 3.* Do the same construction as in step 2 for all points of $B$, yielding labeled point sets $B_1'', \ldots, B_n''$.

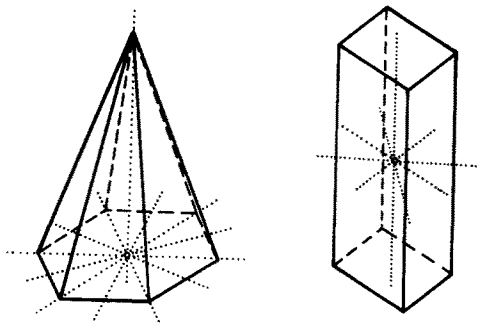
*Step 4.* The sets $A''$, $B_i''$ $(i = 1, \ldots, n)$ are subsets of $(d-1)$-dimensional hyperplanes. Transform them by an isometric mapping into subsets $A'''$, $B_i'''$ $(i = 1, \ldots, n)$ of $\mathbb{R}^{d-1}$ and apply this algorithm recursively to each pair $A'''$, $B_i'''$ $(i = 1, \ldots, n)$. It is easy to see that the original sets $A$, $B$ are exactly congruent if there is a label preserving congruence from $A'''$ into at least one of the sets $B_1''', \ldots, B_n'''$.

*Analysis.* Steps 1 and 2 may take $O(n \log n)$ time because of the sorting of labels. Step 3 takes time $O(n \log n)$ for each $B_i''$ $(i = 1, \ldots, n)$, i.e., $O(n^2 \log n)$ altogether. The transformations in step 4 take linear time, the recursive calls take $n$ times the run time of the $(d-1)$-dimensional problem. This is, certainly for $d \geq 4$, the most significant term and, since Algorithm 1 runs in time $O(n \log n)$ for $d = 3$, we get a total run time of $O(n^{d-2} \log n)$.

*Proof of* (b) (*Symmetry*). Our algorithm for determining the symmetry group of three-dimensional point sets makes use of the following theorem by Hessel (1830) which states that there are only finitely many types of such groups:

**Theorem 2** (Hessel's Theorem) (see [Ma]). *Any finite symmetry group for a subset of* $\mathbb{R}^3$ *is one of the following groups*:

(a) *The rotation groups T, C, I of the platonic solids tetrahedron, cube, and icosahedron, respectively.*
(b) *The cyclic groups* $C_n$, $n = 1, 2, \ldots$, *and the dihedral groups* $D_n$, $n = 2, 3, \ldots$.
(c) *The groups* $\bar{T}$, $\bar{C}$, $\bar{I}$; $\bar{C}_1$, $\bar{C}_2, \ldots$; $\bar{D}_2$, $\bar{D}_3, \ldots$, *where* $\bar{G}$ *means the group generated by G and a reflection at some point* (*inversion*).



(a) (b)

Fig. 3

(d) *The groups* $CT$, $C_{2n}C_n$, $D_{2n}D_n$, $D_nC_n$, $n = 2, 3, \ldots$, *where* $GH$ *means* $H \cup (G - H) \circ i$ *where* $i$ *is an inversion.*

For example, the sets in Fig. 3(a) and (b) have rotation groups $C_6$, $D_4$ and symmetry groups $D_6C_6$, $\bar{D}_4$, respectively.

In any case the actual set of transformations that map a set of points into itself is determined by the symmetry group and the set of rotation axes. If the symmetry group is not one of the finitely many derived from the platonic solids, there will be $n+1$ rotation axes for some $n \in \mathbb{N}$. As in Fig. 3 $n$ of those axes lie in a plane, the $(n+1)$st is perpendicular to that plane. Our algorithm to determine the symmetry group of a set $A$ of $n$ points proceeds as follows:

**Algorithm 3.**

*Step 1.* Apply Algorithm 1 to the input pairs $A$, $A$ and $A$, $J(A)$. Hopcroft's partitioning algorithm partitions the set of points into equivalence classes of "indistinguishable" ones, i.e., points which can be mapped into each other by some congruence (in this case symmetry) transformation. In other words, we get the orbits of all points under the symmetry group.

*Step 2.* First we check if the symmetry group is one of $T$, $C$, $I$, $\bar{T}$, $\bar{C}$, $\bar{I}$, $CT$. A detailed analysis shows that for each of these symmetry groups the size of the orbit of any point is bounded by some constant (e.g., 48 for $\bar{C}$). In addition the set of rotation axes can be determined uniquely from any nontrivial (i.e., non-singleton) orbit. Therefore, the algorithm for each of the above symmetry groups checks if the orbits satisfy the size bound. If that is the case the set of possible rotation axes is determined from one orbit. As mentioned before this determines the set of transformations. Each of these is applied to all points in $A$ and it is checked if the image set is $A$ again. If so, the symmetry group has been found.

*Step 3.* If step 2 has failed the symmetry group of $A$ must be one of $C_m$, $D_m$, $\bar{C}_m$, $\bar{D}_m$, $C_{2m}C_m$, $D_{2m}D_m$, or $D_mC_m$ for some $m \in \mathbb{N}$. It can be shown that in this

case each orbit has size 1, 2, $m$, $2m$, or $4m$. So, by inspecting one orbit of size $>2$ the value of $m$ and, thus, the number of symmetry groups is restricted to constantly many possibilities. The possible rotation axes are determined and all possibilities of groups are tested like in step 2.

*Analysis.* Step 1 takes $O(n \log n)$ time. Determining the possible sets of transformations by inspecting one orbit in step 2 takes constant time, the test to determine if this set of transformations really maps $A$ into itself takes time $O(n \log n)$ (comparison of two $n$-element sets). Determining the possible transformations in step 3 takes time $O(n)$, testing if they map $A$ into itself takes time $O(n \log n)$.

## 3. The Approximate Case

The major drawback of the congruence and symmetry problems considered in the previous section and previous papers [Ata1], [Ata2], [Atk], [H] is that they are ill-posed. In fact, arbitrary small perturbations in the input data will turn pairs of congruent sets into noncongruent ones and destroy symmetries. The symmetry problem even becomes trivial if we assume that input data are represented by, say, rational Cartesian coordinates. Namely, in this case only finitely many different symmetry groups are possible at all, consisting of rotations by multiples of 90° angles possibly combined with reflections.

This shows that the exact versions of these problems are unrealistic. We therefore define the *approximate congruence problem with tolerance* $\varepsilon$:

Given two sets $A$, $B$ of $n$ points each, decide if there exists a congruence which maps in a 1-to-1 fashion the points of $B$ into the $\varepsilon$-neighborhood of points of $A$, i.e., if there is a bijection $l: B \to A$ and a congruence $M$ such that $M(x) \in U_\varepsilon(l(x))$ for all $x \in B$. Here $U_\varepsilon(a)$ denotes the closed $\varepsilon$-neighborhood of $a$.

We believe that the approximate congruence problem is interesting by itself. Namely, many geometric objects occurring in practice (biology, crystallography) are not perfectly congruent or symmetric but only within a certain tolerance. In fact, the minimum possible tolerance can be considered as a quantitative measure for the degree of symmetry or congruence.

We next describe a series of results solving various cases of the approximate congruence problem in two dimensions. We classify these results according to the following criteria.

Type of congruence:

$T$—translation, i.e., $M: x \mapsto x + t$, for some $t \in \mathbb{R}^2$.

$R$—rotation around a center which is known, i.e., $M: x \mapsto A_\varphi(x - c) + c$, where $c$ is the center of the rotation, $\varphi$ is the angle of rotation, and

$$A_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$$

is the rotation matrix.

$I$—arbitrary isometric mapping, consisting of a combination of translation, rotation, and reflexion.

Knowledge of some bijective labeling $l: B \to A$ telling, for any point in $B$, into whose neighborhood it should be mapped:

$l$—known.
$u$—unknown, determine if such a labeling exists.

Specification of $\varepsilon$:

$D$—$\varepsilon$ is given (decision problem).
$E(\delta)$—$\varepsilon$ is given and the set $A$ is $\delta$-disjoint, i.e., $U_\delta(x) \cap U_\delta(y) = \varnothing$ for all $x, y \in A$ where $U_\delta(x)$ is the $\delta$-neighborhood of $x$ with respect to the underlying metric.
$O$—find the smallest $\varepsilon$ such that an approximative congruence exists (optimization problem).

Metric:

$e$—Euclidean metric.
$m$—maximum metric.

So, for example, *TlOe* means the problem: given two sets of $n$ points, $A$, $B \subseteq \mathbb{R}^2$, and a labeling $l: B \to A$, find the smallest $\varepsilon > 0$ such that there exists a translation $M: \mathbb{R}^2 \to \mathbb{R}^2$ with $M(x) \in U_\varepsilon(l(x))$ for all $x \in B$. We obtained the following upper bounds:

**Theorem 3.**

| *Problem* | *Solvable in time* | |
|---|---|---|
| $Tl\{D\{e, m\}, O\{e, m\}\}$ | $O(n)$ | |
| $TuD\{e, m\}$ | $O(n^6)$ | |
| $TuO\{e, m\}$ | $O(n^6 \log n)$ | |
| $TuE(\varepsilon)e$ | $O(n \log n)$ | |
| $TuE(a\varepsilon)m$ | $O(n \log n)$ | *for any fixed* $a > 1$ |
| $RlD\{e, m\}$ | $O(n \log n)$ | |
| $RlO\{e, m\}$ | $O(n^2)$ | |
| $IlD\{e, m\}$ | $O(n^3 \log n)$ | |
| $IuD\{e, m\}$ | $O(n^8)$ | |

*Proof.* Let $A=\{a_1,\ldots,a_n\}$, $B=\{b_1,\ldots,b_n\}$ be such that in the labeled cases $l(b_i)=a_i$ for $1\le i\le n$. We use $C_i$ to denote the circle of radius $\varepsilon$ (=boundary of $U_\varepsilon(a_i)$) around $a_i$, $1\le i\le n$.

$$Tl\begin{cases} D\begin{cases} e \\ \end{cases} \\ O\begin{cases} m \end{cases} \end{cases}$$

Let $r\in\mathbb{R}^2$ be some fixed point, called the reference point. Let $K_i$, $1\le i\le n$, be the set of images of $r$ under all translations mapping $b_i$ into $U_\varepsilon(a_i)$. It is easy to see that the decision problem has a positive answer, exactly if the intersection

$$K_1\cap\cdots\cap K_n\neq\varnothing.$$

In fact, each translation mapping the reference point into this intersection, will map each $b_i$ into $U_\varepsilon(a_i)$, $1\le i\le n$. In the Euclidean case each $K_i$ is in a closed disk, so the question whether the intersection is nonempty is equivalent to the question whether there exists a circle of the same radius, namely $\varepsilon$, enclosing the centers. The optimization problem is thus equivalent to finding the smallest enclosing circle of $n$ points, which can be solved in $O(n)$ time by an algorithm due to Megiddo [Me] (see also [PS]).

In the case of the maximum metric the $K_i$ ($1\le i\le n$) are squares (orthogonal to the axes) instead of disks and there is a straightforward $O(n)$ algorithm to find the smallest enclosing square of $n$ points. Thus we obtain $O(n)$ algorithms for the optimization and, consequently, the decision problem.

$$TuD\begin{cases} e \\ m \end{cases}$$

**Algorithm 4.** We present the algorithm for the Euclidean case but it works for the maximum metric, as well. It uses the following ideas:

*Step 1.* An easy geometric argument shows that if there is any solution at all, then there must be one where some point $b_j$ lies directly on the circle $C_i$ of radius $\varepsilon$ around the assigned point $a_i$ ($1\le i,j\le n$). We check this property for all pairs $i$, $j$.

For fixed $i$, $j$ describe any position of $b_j$ on the circle $C_i$ by the polar coordinate $\varphi$ using $a_i$ as the origin and, for example, the ray parallel to the $x$-axis in the positive direction through $a_i$ as the $\varphi=0$ ray (see Fig. 4).

*Step 2.* For any $l\neq j$ consider the circle $K_l$ onto which $b_l$ is mapped by translations which map $b_j$ onto $C_i$. For any $m\neq i$ we determine the set $I_{l,m}$ of angles $\varphi\in[0,2\pi[$ such that if the image of $b_j$ on $C_i$ has polar coordinate $\varphi$, then $b_l$ lies within $U_\varepsilon(a_m)$. For example, in Fig. 4 $I_{l,m}=[\varphi_1,\varphi_2]$. In any case $I_{l,m}$ is a (circular) interval of $[0,2\pi[$ whose endpoints are given by the cutpoints of $K_l$ and $C_m$. We determine all these intervals and sort their endpoints. (A detailed

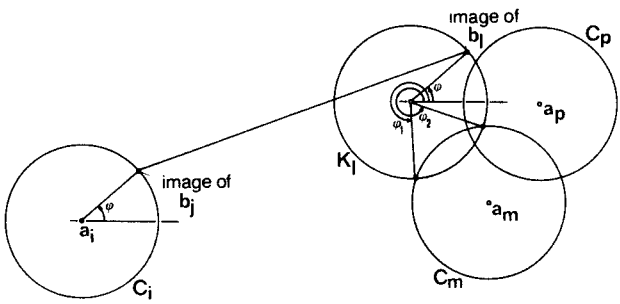



Fig. 4

description of how this is done is given in step 5 of this algorithm.) This gives a partition of the interval $[0, 2\pi[$ into subintervals, each of which is contained in some of the $I_{l,m}$ $(1 \le l, m \le n;\ l \ne j, m \ne i)$ and is disjoint from the others.

*Step 3.* We assign to each such subinterval $I$ a bipartite graph $G_I = (V, E_I)$, where $V$ is the disjoint union of two sets $\{u_1, \ldots, u_{i-1}, u_{i+1}, \ldots, u_n\}$ and $\{v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n\}$ of nodes and $E_I = \{(u_l, v_m);\ I \subset I_{l,m}\}$. Now suppose that the graph $G_I$ for some interval $I$ contains a perfect matching, namely edges $(u_{i_1}, v_{j_1}), \ldots, (u_{i_{n-1}}, v_{j_{n-1}})$. This means that $I \subset I_{i_1,j_1} \cap \cdots \cap I_{i_{n-1},j_{n-1}}$. Therefore any translation mapping $b_j$ onto $C_i$ such that the polar coordinate of the image of $b_j$ is in $I$ will map $b_{j_k}$ into $U_\varepsilon(a_{i_k})$, $1 \le k \le n-1$, i.e., there exists an assignment which shows that $A$ and $B$ are approximately congruent. First we determine the graph for the subinterval containing $\varphi = 0$. This can be done by checking, for each pair $l$, $m$, if the image of $b_l$ lies within $U_\varepsilon(a_m)$. We then determine the maximum matching of the graph (see [M]) and, if it is perfect, give a positive answer.

*Step 4.* Starting from the maximum matching of the graph for the first subinterval $I_1$ determined in the previous step we proceed to neighboring intervals using the sorted sequence of endpoints. Clearly, each time we proceed from one interval $I_k$ to its neighbor $I_{k+1}$, an edge has to be deleted from the corresponding graph or a new one has to be added. In the first case the maximum matching of $G_{I_{k+1}}$ can be determined from the one of $G_{I_k}$ in the following way: if the edge deleted was not part of the maximum matching then it stays the same for $G_{I_{k+1}}$. Otherwise, check if there is an augmenting path and, if so, use it to enlarge the matching. If $G_{I_{k+1}}$ results from $G_{I_k}$ by adding a new edge, determine if an augmenting path exists. If so, use it to enlarge the matching. There is an approximate congruence of $A$ and $B$ exactly if one of the maximum matchings is perfect.

*Step 5.* We now describe in detail how step 2 is realized. We start with the observation that expressions of the form $A + B\sqrt{C}$ can be compared using only elementary arithmetic operations. Note first that the sign of an expression $A + B\sqrt{C}$ is easily computed. Assume next that we want to compare $A + B\sqrt{C}$ and $D + E\sqrt{F}$. Then $A + B\sqrt{C} \le D + E\sqrt{F}$ iff $A - D + B\sqrt{C} \le E\sqrt{F}$. If the signs of the two sides are now different then we are done. If the signs are the same, say

positive, then $A+B\sqrt{C} \leq D+E\sqrt{F}$ iff $(A-D)^2+B^2C+2(A-D)B\sqrt{C} \leq E^2F$. This we already know how to check. We summarize in:

*Fact A.* Let $A$, $B$, $C$, $D$, $E$, and $F$ be elements of $\mathbb{Q}$ given as quotients of integers of bounded precision. Then the test $A+B\sqrt{C} = (<, \leq)D+E\sqrt{F}$ can be carried out using only $O(1)$ additions, subtractions, and multiplications on single precision integers.

We described step 2 for ease of description in terms of polar coordinates; the actual computations are better carried out in Cartesian coordinates. Let $(x, y)$ be the Cartesian coordinates of the image of $b_j$ with respect to a Cartesian coordinate system with origin $a_i$. Let $(x_l, y_l)$ be the vector $b_l - b_j$ and let $(u_m, v_m)$ be the Cartesian coordinates of $a_m$. Then $b_j$ is mapped onto $C_i$ and $b_l$ is mapped onto $C_m$ iff

$$x^2+y^2=\varepsilon^2$$

and

$$(x+x_l-u_m)^2+(y+y_l-v_m)^2=\varepsilon^2.$$

This system has solutions

$$x = R \pm \sqrt{Q},$$

where

$$R = -a/2,$$
$$Q = (4b^2\varepsilon^2 - b^2(a^2+b^2))/4(a^2+b^2),$$

and $a = x_l - u_m$ and $b = y_l - v_m$. If there are no real solutions for $x$ we are done, otherwise the two solutions for $x$ determine four points on the circle $C_i$; two of them are the endpoints of the circular interval we are looking for. We find these endpoints by checking which of the four points $(0, \varepsilon)$, $(R, \sqrt{\varepsilon^2-R^2})$, $(0, -\varepsilon)$, $(R, -\sqrt{\varepsilon^2-R^2})$ lie in the interval, i.e., if $b_j$ is mapped onto the point under consideration then $b_l$ is mapped into $U_\varepsilon(a_m)$. Note that the latter condition is easily checked by Fact A.

Let $p^1_{lm}$, $p^2_{lm}$, be the endpoints obtained in this way such that $[p^1_{lm}, p^2_{lm}]$ is the circular interval in counterclockwise traversal of $C_i$. We now sort these endpoints according to their circular order on $C_i$ using any $O(n \log n)$ sorting algorithm. The comparisons take $O(1)$ elementary arithmetic operations by Fact A. This finishes the detailed description of step 2.

*Analysis.* Step 1 states that steps 2–4 are repeated $n^2$ times for all pairs $i, j$, $1 \leq i, j \leq n$. In step 2 all cutpoints of circles $K_l$ and $C_m$, $1 \leq l, m \leq n$; $l \neq j$, $m \neq i$, are determined and sorted with respect to their angular polar coordinate. This takes time $O(n^2 \log n)$ by the discussion above. In step 3 determining the graph belonging to $\varphi = 0$ takes $O(n^2)$, determining its maximum matching $O(n^{2.5})$ time, (see [M, Section IV.9.2]). In step 4 we check, for each of the $O(n^2)$ subintervals,

if the corresponding graph together with the previous matching contains an augmenting path, which takes $O(n^2)$ time. So the total run time of one iteration of step 4 is $O(n^4)$, which asymptotically exceeds the run times of steps 2 and 3. The total run time of the algorithm is thus $O(n^6)$.

$TuO\begin{cases} e \\ m \end{cases}$

**Algorithm 5.**

*Step 1.* For $1 \le i_1, j_1, i_2, j_2, i_3, j_3 \le n$ let $\varepsilon(i_1, j_1, i_2, j_2, i_3, j_3)$ be the smallest $\varepsilon$ such that there is a translation mapping $b_{i_l}$ into $U_\varepsilon(a_{j_l})$, $1 \le l \le 3$. An easy geometric argument shows that the optimal value $\varepsilon_{\text{opt}}$ of $\varepsilon$ is equal to one of the $n^6$ values $\varepsilon(i_1, \ldots, j_3)$. Each such value can be computed in constant time as follows. Let $r$ denote the image of $b_{i_1}$. Then $\varepsilon(i_1, \ldots, j_3)$ is defined by

$$\begin{aligned} &\text{minimize} && \varepsilon \\ &\text{subject to} && |a_{j_l} - (b_{i_l} + (r - b_{i_1}))|^2 \le \varepsilon^2. \end{aligned}$$

Thus $r$ is the center and $\varepsilon$ is the radius of the smallest circle containing the three points $a_{j_l} - b_{i_l}$, $l = 1, 2, 3$. If the triangle formed by these three points is obtuse (this can be checked by computing the sign of three scalar products) then $r$ is the midpoint of the longest edge of the triangle and $\varepsilon$ is half the length of this edge and if the triangle is right or acute then $r$ is the center of the circumcircle and $\varepsilon$ is the radius of that circle. In the latter case $r$ is the intersection of the bisectors of the edges of the triangle and hence the coordinates of $r$ are quotients of linear functions in the coordinates of the endpoints. Thus in either case $\varepsilon = \sqrt{R}$ where $R$ is a rational function in the coordinates of $b_{i_l}$ and $a_{i_l}$, $1 \le l \le 3$.

*Step 2.* We sort the $n^6$ values computed in step 1 (time $O(n^6 \log n)$) and determine $\varepsilon_{\text{opt}}$ by binary search on these values. In each step of the binary search we invoke Algorithm 4 to decide whether a solution exists for the given values. (Note that Algorithm 4 uses only the value $\varepsilon^2$ in its computations; this is always a rational function of the inputs here.) The binary search takes time $O(n^6 \log n)$.

$TuE(\varepsilon)e$

**Algorithm 6.**

*Step 1.* We first want to argue that the labeling used in a solution (if there is any) is unique. Assume otherwise, say $l_i$ is a labeling under translation $T_i$: $x \mapsto x + t_i$, $i = 1, 2$. We may assume without loss of generality that $l_1^{-1}(a_j) = b_j$,

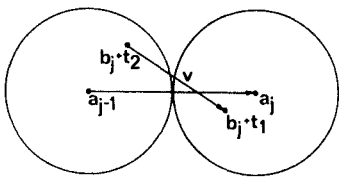



Fig. 5

$l_2^{-1}(a_j)=b_{j+1}$, and $l_2^{-1}(a_k)=b_1$ for $1\le j\le k$ and some $k$. Since $T_i$ is a solution with respect to labeling $l_i$, we have

$$\begin{aligned} &b_j+t_1\in U_\varepsilon(a_j) && \text{for } 1\le j\le k,\\ &b_j+t_2\in U_\varepsilon(a_{j-1}) && \text{for } 2\le j\le k,\\ &b_1+t_2\in U_\varepsilon(a_k). \end{aligned}$$

Let $v=t_1-t_2$. Then $(a_j-a_{j-1})\cdot v\ge 0$, as can be seen from Fig. 5. Since $\sum_{j=2}^{k}(a_j-a_{j-1})+(a_1-a_k)=0$ we conclude $v=0$. This contradicts the fact that $U_\varepsilon(a_j)\cap U_\varepsilon(a_{j-1})=\varnothing$ for all $j$. Thus there is at most one labeling which can be used in an approximate congruence.

*Step. 2.* It can easily be shown that any translation which is an approximate congruence between $B$ and $A$ must also map the centroid $c_B$ into $U_\varepsilon(c_A)$. We cover $U_\varepsilon(c_A)$ by circles which have sufficiently small radius $\varepsilon_0$ so that they cannot intersect more than two circles of radius $\varepsilon$ around points of $A$. Figure 6 shows that $\varepsilon_0=(\frac{2}{3}\sqrt{3}-1)\varepsilon$ will do.

For each small circle $C$ of this cover (there are constantly many of them) we decide if there exists a translation mapping $c_B$ into $C$ and satisfying the conditions for an approximate congruence between $A$ and $B$.

*Step 3.* Let $K_i$, $1\le i\le n$, be the set of images of $b_i$ under translations mapping $c_B$ into $C$, i.e., the circle of radius $\varepsilon_0$ around $b_i':=b_i+c-c_B$, where $c$ is the center of $C$. For the set $A$ we construct the first- and second-order Voronoi diagrams. (Note that the coordinates of the vertices of these diagrams are rational functions of bounded degree of the coordinates of the input and, therefore, cause no problems for our single precision model of computation.) Then we locate each

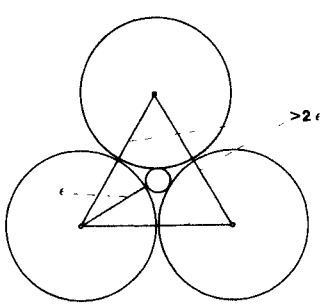



Fig. 6

$b_i'$ in order to determine the two points in $A$ closest to it. For the ones which are at a distance less than $\varepsilon + \varepsilon_0$ from $b_i'$, i.e., $K_i$ intersects their $\varepsilon$-neighborhoods, we introduce an edge in an undirected graph $G$ of the same form as in Algorithm 4.

*Step 4.* In the graph $G$ constructed in step 3 any node in the set $\{v_1, \ldots, v_n\}$ has degree $\leq 2$. We determine if $G$ has a perfect matching in the following way: if there is any node of degree 0, no perfect matching exists. Otherwise, if there is some node which has degree 1 we add the incident edge $e$ to the matching, and remove $e$, its endpoints, and all other edges incident to them from the graph. We repeat this until there are no more edges of degree 1 left. Now all remaining edges in $U$ and in $V$ must have degree 2 and hence each connected component is a cycle. For each cycle there are only two possible matchings. We check both matchings by the linear-time algorithm for the labeled case. By the argument at the beginning of the proof at most one of the labelings can yield an approximate congruence. Once we have checked each cycle only one labeling is left. We check this labeling by the linear-time algorithm.

*Analysis.* Step 2 states that steps 3–4 have to be executed for each circle in the cover, i.e., a constant number of times. Constructing the Voronoi diagrams and locating the points $b_i'$ in them takes $O(n \log n)$ time (see [PS] or [M]). Determining the matchings in step 3 takes $O(n)$, testing the two matchings for a cycle of length $c$ takes time $O(c)$ for a total of $O(n)$ over all cycles. The final test also takes time $O(n)$.

$TuE(a\varepsilon)m$

**Algorithm 7.** We proceed as in Algorithm 6. Here we have squares instead of circles. Let $\delta = 2a\varepsilon$, i.e., any two points in $A$ have a distance of at least $\delta$. In step 1 we cover the square of "radius" $\varepsilon$ and center $c_A$ by small squares of radius $\delta - 2\varepsilon$ which can intersect at most one of the $\varepsilon$-neighborhoods of the points of $A$. This makes the matching problem in step 4 even easier. The total run time is clearly $O(n \log n)$.

$RID\begin{cases} e \\ m \end{cases}$

**Algorithm 8.** Let $c$ be the center of the rotation and $I_i \subset [0, 2\pi[$ the set of angles under which a rotation around $c$ maps $b_i$ into $U_\varepsilon(a_i)$, $1 \leq i \leq n$.

In the case of the Euclidean metric each $I_i$ is an interval on the unit circle, which can be computed as follows. We may assume without loss of generality that $c$ is the origin of our underlying Cartesian coordinate system. Then $I_i$ has endpoints $\varphi_0 - \varphi_1, \varphi_o + \varphi_1$ where $\varphi_0$ and $\varphi_1$ are given by (see Fig. 7)

$$\cos \varphi_0 = A/\sqrt{C} \quad \text{and} \quad \cos \varphi_1 = B/\sqrt{C},$$

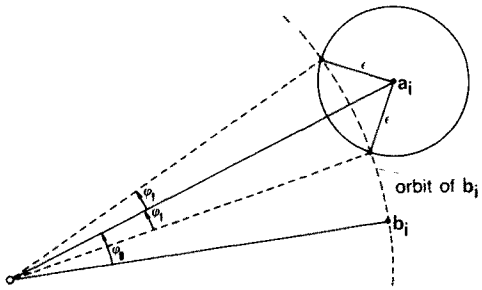



Fig. 7

where $A = a_i^T \cdot b_i$, $B = (\|a_i\|^2 + \|b_i\|^2 - \varepsilon^2)/2$, and $C = \|a_i\|^2\|b_i\|^2$. Here $a_i^T \cdot b_i$ is the scalar product of $a_i$ and $b_i$ and $\|x\|$ is the norm $\sqrt{x^T \cdot x}$ of $x$. Hence $\cos(\varphi_0 \pm \varphi_1) = (AB)/C \mp (1/C)\sqrt{|(C-A^2)(C-B^2)|}$ and we can compare the cosine values of the interval endpoints in time $O(1)$ by Fact A. Clearly, the problem instance has a positive answer exactly if $I_1 \cap \cdots \cap I_n \neq \emptyset$. We sort the boundaries of the intervals $I_i$, $i = 1, \ldots, n$ ($O(n \log n)$ time). Then we count how many $b_i$ are within $U_\varepsilon(a_i)$ (i.e., within the correct $\varepsilon$-neighborhood when a rotation of 0° is performed). We traverse the sequence of interval boundaries, and increment (decrement) our count by 1 for each left (right) interval boundary. Whenever the count becomes $n$, we have found a rotation of the desired form.

In case of the maximum metric each $I_i$ consists of at most four intervals, namely the set of intervals on a circle, which are within a given square. Again we sort the boundaries of all these intervals and proceed as in the Euclidean case.

The remaining algorithms are described for the Euclidean metric but they work for the maximum metric as well.

$$RIO\begin{cases} e \\ m \end{cases}$$

**Algorithm 9.** We assume again that the center $c$ of the rotation is the origin. Let $\varphi$ be the angle of a rotation satisfying approximate congruence for a minimum value of $\varepsilon$. An easy geometric argument shows that only two situations are possible:

1. The image of some $b_i$ has distance $\varepsilon_i$ from $a_i$ where $\varepsilon_i$ is the minimal possible distance of $b_i$ from $a_i$ under rotations.
2. There is a pair $i, j$ $(1 \leq i < j \leq n)$ such that $b_i$ and $b_j$ are mapped onto the circles $C_i$, $C_j$ respectively of radius $\varepsilon$ around $a_i$, $a_j$, see Fig. 8.

Situation 1 yields $n$ possible values of $\varepsilon$, namely $|\,\|a_i\| - \|b_i\|\,|$ for $1 \leq i \leq n$, and situation 2 yields $O(n^2)$ values of $\varepsilon$. This can be seen as follows. For fixed $i$ and $j$, situation 2 can be described by two equations in the unknowns $\varphi$ (= angle of rotation) and $\varepsilon$, namely $\|A_\varphi b_i - a_i\|^2 = \varepsilon^2$ and $\|A_\varphi b_j - a_j\|^2 = \varepsilon^2$. Solving these

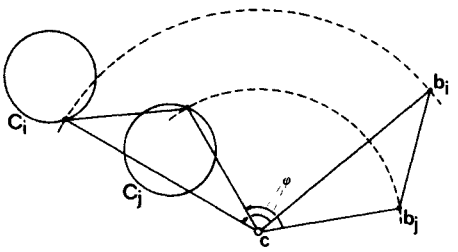



Fig. 8

equations for $\sin\varphi$ and $\cos\varphi$ yield $\sin\varphi$ and $\cos\varphi$ as linear functions in $\varepsilon^2$ and substitution into $\sin^2\varphi+\cos^2\varphi=1$ yields a quadratic equation $A\varepsilon^2+B\varepsilon+C=0$ for $\varepsilon$; here $A$, $B$, and $C$ are rational functions in the coordinates of $a_i$, $b_i$, $a_j$, and $b_j$. Thus each such equation yields at most two values of $\varepsilon$ and solutions of such equations can be compared by Fact A. We determine the set $E$ of all these values. The solution $\varepsilon$ of the optimization problem must be one of them, and we determine it by a variant of binary search. First we determine the median $\varepsilon_M$ of $E$ and apply Algorithm 8 to find out if there is a solution with tolerance $\varepsilon_M$.

Note that $\varepsilon_M$ is of the form $D+E\sqrt{F}$, where $D$, $E$, and $F$ are rational functions of the coordinates of four points. Thus the cosines of the interval endpoints in Algorithm 8 are of the form $G+H\sqrt{I+J\sqrt{K}}$, where $G$, $H$, $I$, $J$, $K$ are rational functions of the coordinates of six points. More precisely, $G, H, \ldots, K$ are quotients of $O(1)$ precision integers (recall that we assume that the coordinates are given as single precision integers). Expressions of this form cannot be compared on the basis of Fact A. However, Mignotte's [Mi] results imply that two such expressions can be compared by comparing numerical approximations of bounded precision. Such approximations can be computed in $O(1)$ time. If there is a solution with tolerance $\varepsilon_M$, we apply the algorithm recursively to the set $E_1$ of elements of $E$ less than $\varepsilon_M$, otherwise to the set $E_2$ of elements greater than $\varepsilon_M$ until the smallest element of $E$ for which a solution exists, is found. The time to compute the set $E$ is $O(n^2)$. The first step of the binary search takes time $O(n^2)$ to find $\varepsilon_M$ by fast median finding [M, Section II.4], $O(n^2)$ to split $E$ with respect to $\varepsilon_M$, and $O(n\log n)$ to apply Algorithm 8. It follows that the total run time is $O(n^2)$.

$$IID\begin{cases} e \\ m \end{cases}$$

**Algorithm 10.** Any isometric mapping consists either of a rigid motion, i.e., a combination of translation and rotation, or a reflexion (at some straight line in the two-dimensional case), which can be chosen arbitrarily, followed by a rigid motion. We give an algorithm which tests if $B$ can be mapped onto $A$ with tolerance $\varepsilon$ by a rigid motion. We consider both versions of isometric mappings

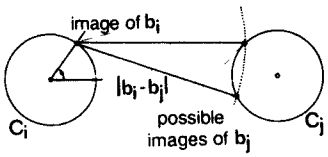



Fig. 9

by applying this algorithm once to the pair $A$, $B$ and once to $A$, $B'$, where $B'$ is obtained from $B$ by reflecting it at some arbitrary straight line.

A simple geometric argument shows that there exists a rigid motion mapping any $b_i$ into $U_\varepsilon(a_i)$, $1 \le i \le n$, exactly if there exists a rigid motion of that kind which maps at least two points $b_i$, $b_j$ of $B$ onto the borders of their corresponding $\varepsilon$-neighborhoods, i.e., the circles $C_i$, $C_j$. Our algorithm tests this property for all $O(n^2)$ possible pairs $(i, j)$. Given such a pair we denote by $(x_l, y_l)$ the coordinates of the image of $b_l$ and by $(a_l^x, a_l^y)$ the coordinates of $a_l$ with respect to a coordinate system with origin $a_i$. We check for the existence of the desired rigid motion by checking four cases separately: $y_i \gtrless 0$ and $a_j$ lies left (right) of the oriented line from $(x_i, y_i)$ to the image of $b_j$, i.e., $(y_j - y_i, x_i - x_j)^T \cdot (a_j^x - x_i, a_j^y - y_j) \gtrless 0$, see Fig. 9. We show how to treat the case where both quantities are positive, the other cases are symmetric. Note that the parameter $x_i$ completely determines the rigid motion in this case. Thus there exists a rigid motion of the desired form iff the following system of algebraic equalities and inequalities has a solution:

(1) The point $(x_i, y_i)$ lies on $C_i$ and $y_i \ge 0$:

$$x_i^2 + y_i^2 = \varepsilon^2, \qquad y_i \ge 0.$$

(2) The point $(x_j, y_j)$ lies on $C_j$, $(x_j, y_j)$ has distance $\|b_j - b_i\|$ from $(x_i, y_i)$, and $(x_j - x_i, y_j - y_i)^T \cdot (a_j^x - x_i, a_j^y - y_i) \ge 0$:

$$(x_j - a_j^x)^2 + (y_j - a_j^y)^2 = \varepsilon^2,$$
$$(x_j - x_i)^2 + (y_j - y_i)^2 = \|b_j - b_i\|^2,$$
$$(y_j - y_i, x_i - x_j)^T \cdot (a_j^x - x_i, a_j^y - y_i) \ge 0.$$

(3p) For all $p \ne i, j$ the image of $b_p$ lies in $U_\varepsilon(a_p)$, it has the correct distance from the image of $b_i$, and the angle defined by the images of $b_j$, $b_i$, and $b_p$ is the same as the angle defined by the original points $b_j$, $b_i$, and $b_p$:

$$(x_p - a_p^x)^2 + (y_p - a_p^y)^2 \le \varepsilon^2,$$
$$(x_p - x_i)^2 + (y_p - y_i)^2 = \|b_p - b_i\|^2,$$
$$(x_p - x_i, y_p - y_i)^T \cdot (x_j - x_i, y_j - y_i) = (b_p - b_i)^T \cdot (b_j - b_i)$$

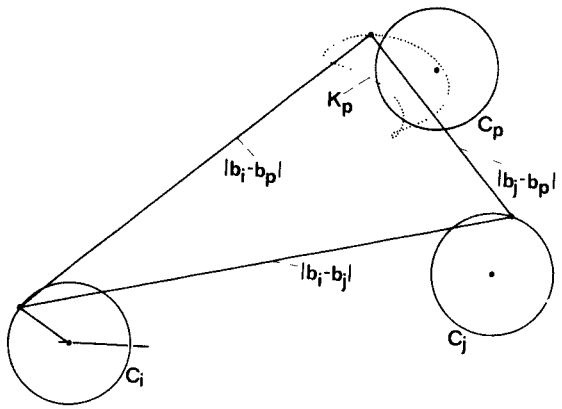



Fig. 10

We check the satisfiability of this system as follows. For $p \neq i, j$ let $S_p^{\leq}$ ($S_p^{=}$) be the system consisting of (1), (2), and (3p) ((3p) with $\leq$ replaced by $=$). The system $S_p^{=}$ has either all $x_i$'s, $-1 \leq x_i \leq 1$, as solutions or it has at most 12 different solutions. This follows from the fact that the image of $b_p$ moves on an algebraic curve $K_p$ of degree 6 as the image of $b_i$ moves on $C_i$ (see [W, p. 68]; Fig. 10 illustrates the curve $K_p$) and that this curve intersects $C_p$ in at most 12 points. We can determine the exact number by checking the validity of the following formulae using Collins's procedure [C] ($l = 1, 2, \ldots, 13$):

$$\exists Z_1, \ldots, Z_l: \quad -1 \leq Z_1 < Z_2 < \cdots < Z_l \leq 1 \wedge S_p^{=}(Z_1) \wedge \cdots \wedge S_p^{=}(Z_l).$$

This takes time $O(1)$ since the formulae have size $O(1)$. Let $l_p$ denote the number of distinct solutions found. If $l_p = 13$ then all $x_i$'s, $-1 \leq x_i \leq 1$, are solutions and we can drop $b_p$ from further consideration. If $l_p \leq 12$ then let $Z_1^p, \ldots, Z_{l_p}^p$ be variables for the different solutions. We next check whether $S_p^{\leq}(-1)$, $S_p^{\leq}(+1)$, $S_p^{\leq}((Z_j + Z_{j+1})/2)$, $1 \leq j < l_p$, hold. The latter is equivalent to

$$\exists Z_1, \ldots, Z_{l_p}: \quad -1 \leq Z_1 < \cdots < Z_{l_p} \leq 1 \wedge S_p^{=}(Z_1) \wedge \cdots \wedge S_p^{=}(Z_{l_p}) \wedge S_p^{\leq}((Z_j + Z_{j+1})/2).$$

We have now computed, for each $p \neq i, j$, a union $I_p$ of at most six intervals for $x_i$ within which $b_p$ is mapped into $U_\epsilon(a_p)$. We sort the endpoints of these intervals using any $O(n \log n)$ sorting algorithm. A comparison of $Z_k^r$ with $Z_l^s$ is performed in $O(1)$ time by evaluating

$$\exists Z_1^r, \ldots, Z_{l_r}^r, Z_1^s, \ldots, Z_{l_s}^s: \quad -1 \leq Z_1^r < \cdots < Z_{l_r}^r \leq 1 \wedge S_p^{=}(Z_1^r) \wedge \cdots \wedge S_r^{=}(Z_{l_r}^r) \wedge -1 \leq Z_1^s < \cdots < Z_{l_s}^s \leq 1 \wedge S_x^{=}(Z_1^s) \wedge \cdots \wedge S_s^{=}(Z_{l_s}^s) \wedge Z_k^r \leq Z_l^s.$$

Thus sorting takes $O(n \log n)$ time. It is now a simple matter to scan the endpoints and to determine whether the intersection of the $I_p$'s, $p \neq i, j$, is nonempty.

The algorithm takes time $O(n \log n)$ for each pair $i, j$, so the total run time is $O(n^3 \log n)$.

$$IuD\begin{cases} e \\ m \end{cases}$$

**Algorithm 11.** As in the labeled case, if there is a solution, there must be one which maps at least two points $b_i$, $b_j$ onto circles $C_k$, $C_l$. We test this property for all quadruples $i, j, k, l \in \{1, \ldots, n\}$. The coordinate $x_i$ is defined as in the labeled case, only that now the image of $b_i$ lies on $C_k$ instead of $C_i$. Let $I_{p,m}$, $1 \leq m \leq n$, be the set of values of $x_i$, such that the image of $b_p$ lies within $U_\epsilon(a_m)$. As before, $I_{p,m}$ is the union of at most six intervals. We sort the boundaries of all the intervals obtained this way and proceed exactly as in Algorithm 4. Whenever we find a perfect matching, the problem has a solution.

Steps 2–4 of Algorithm 4 are executed $O(n^4)$ times, one execution takes $O(n^4)$ time, so we get an upper bound of $O(n^8)$.

## 4. Conclusion

We presented algorithms for computing exact and approximate congruences and symmetries. We believe that the approximate version of the problem has more practical relevance than the exact version. Although we have found some answers we also left a large number of questions unresolved. Can our results on approximate congruence be transferred to higher dimensions? How do the time bounds change if we drop the assumption that the inputs are given as single precision rational numbers? [Sch] contains answers to that question. Can the running times of our algorithms be improved? Can they be improved if we allow our algorithms to make mistakes? A possible scenario is as follows: for point sets $A$ and $B$ let $\varepsilon_{\text{opt}}(A, B)$ be the minimal value of $\varepsilon$ for which an approximate congruence exists. Suppose now that we require a correct answer on input $(A, B, \varepsilon)$ only if $\varepsilon / \varepsilon_{\text{opt}}(A, B) \notin [\frac{1}{2}, 2]$ and do not care about the answer if $\varepsilon / \varepsilon_{\text{opt}}(A, B) \in [\frac{1}{2}, 2]$. This problem can be solved in time $O(n^{2.5})$ for translations and time $O(n^4)$ for general congruences [Sch].

## Acknowledgments

We thank Stefan Schirra, Sebastian Iwanowski, and Otfried Schwarzkopf for several helpful discussions.

## References

[AHU] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[Ata1] Atallah, M. J., Checking Similarity of Planar Figures, Internat. J. Comput. Inform. Science **13** (1984), pp. 279–290.

[Ata2] Atallah, M. J., On Symmetry Detection, *IEEE Trans. Comput.* **34** (1985), pp. 663–666.

[Atk] Atkinson, M. D., An Optimal Algorithm for Geometrical Congruence, *J. Algorithms* **8** (1987), pp. 159–172.

[C] Collins, G., *Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition*, Lecture Notes in Computer Science: Automata Theory and Formal Languages, Springer-Verlag, Berlin, 1975, pp. 134–183.

[E] Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, New York, 1987.

[H] Highnam, P. T., Optimal Algorithms for Finding the Symmetries of a Planar Point Set, *Inform. Process. Lett.* **22** (1986), pp. 219–222.

[M] Mehlhorn, K., *Data Structures and Algorithms*, Vols. 1, 2, 3, Springer-Verlag, Berlin, 1984.

[Ma] Martin, G. E., *Transformation Geometry*, Springer-Verlag, New York, 1982.

[Me] Megiddo, N., Linear-Time Algorithm for Linear Programming in $\mathbb{R}^3$ and Related Problems, *SIAM J. Comput.* **12** (1983), pp. 759–776.

[Mi] Mignotte, M., Identification of Algebraic Numbers, *J. Algorithms* **3** (1982), pp. 197–204.

[PS] Preparata, F. P. and M. I. Shamos, *Computational Geometry*, Springer-Verlag, New York, 1985.

[Sch] Schirra, St., Über die Bitkomplexität der $\varepsilon$-Kongruenz, Diplomarbeit, FB Informatik, Universität des Saarlandes, 1987.

[W] Wunderlich, W., *Ebene Kinematik*, Bibliographisches Institut, Mannheim, 1970.

*Received May 10, 1987, and in revised form November 10, 1987.*