MAX-PLANCK-INSTITUT FÜR INFORMATIK

Approximate Decision Algorithms for Point Set Congruence

Paul J. Heffernan Stefan Schirra

MPI–I–91–110 August 1991

Im Stadtwald
66123 Saarbrücken
Germany

Approximate Decision Algorithms for Point Set Congruence

Paul J. Heffernan*
School of Operations Research
and Industrial Engineering
Engineering & Theory Center
Cornell University
Ithaca, NY 14853
Stefan Schirra †
Max-Planck-Institut für Informatik
Im Stadtwald
W 6600 Saarbrücken
Germany

August 28, 1991

Abstract

This paper considers the computer vision problem of testing whether two equal cardinality point sets A and B in the plane are ε-congruent. We say that A and B are ε-congruent if there exists an isometry I and bijection $\ell : A \rightarrow B$ such that $dist(\ell(a), I(a)) \leq \varepsilon$, for all $a \in A$. Since known methods for this problem are expensive, we develop approximate decision algorithms that are considerably faster than the known decision algorithms, and have bounds on their imprecision. Our approach reduces the problem to that of computing maximum flows on a series of graphs with integral capacities.

1 Introduction

In this paper we address the following question: given two (equal cardinality) sets of points in the plane, what do we mean when we say that they are "congruent", and how do we

*Email: heff@orie.cornell.edu. Supported by an NSF graduate fellowship.

†Email: stschirr@cs.uni-sb.de. Supported by a grant from G.I.F., the German-Israeli Foundation for Scientific Research and Development

determine congruency? To answer this question, we must define congruent, and provide efficient algorithms that test for congruence. The practical motivation behind our study comes from computer vision, where an observed image is compared to a hypothesized model. Since the given problem is too difficult to permit efficient decision algorithms, we turn our attention to approximate decision algorithms with performance guarantees. Eventually, we reduce the problem to that of computing a maximum flow on each of a series of (s,t)-graphs. We apply known solution techniques, and a new idea of Feder and Motwani [FM] for speeding-up graph algorithms, which is well suited for our max-flow problems.

In pattern recognition, one often wishes to determine how well an observed image matches a model image. For us, the model and the observed image will be point sets A and B of equal cardinality. In order to make A match B as closely as possible, we perform an isometry on A, and then pair each point of A with one of B such that the points of each pair lie close to each other. An *isometry* is an affine mapping in the plane that preserves distances, and is composed of a translation, rotation, and possibly a reflection. Any isometry can be represented as

$$I(x) = MJx + t,$$

where $M = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}$ for some $\varphi \in [0, 2\pi)$, $J \in \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \right\}$, and $t \in \mathbb{R}^2$. Along with an isometry I, we must designate a bijection $\ell : A \to B$. A natural measure of our attempt to match the observed image A with the model image B is the maximum distance between any pair $(\ell(a), I(a))$, $a \in A$. Because of errors in measurement, it is unrealistic to expect an observed image to match exactly the model from which it came, so we will be interested in matchings that are *approximately congruent*. We say that A and B are *congruent with tolerance* ε, or *ε-congruent*, if there exist an isometry I and a bijection $\ell : A \to B$ such that $dist(\ell(a), I(a)) \leq \varepsilon$, for all $a \in A$, where $dist(\cdot,\cdot)$ is the distance function for our chosen metric. We are interested in approximate congruence under the L_2 metric, with various restrictions on the isometry, such as the translation-only and rotation-only cases.

Several researchers have studied approximate congruence, notably Baird [Ba] and Alt, et al. [AMWW]. The distinguishing feature of these algorithms is their high run-time: no known algorithm for the case of the isometry restricted to a translation is better than $O(n^6)$, and for the case of unrestricted isometry, the best known bound is $O(n^8)$ (this bound is tight for the algorithm of [AMWW], as an example in this paper shows). For models with a large number of points, such performance may be unacceptable. Faced with a problem whose solution seems to be too expensive, we look for a useful solution—we trade some of the exactness for improvements in run-time, and obtain fast algorithms that come with a performance bound. We obtain *approximate* decision algorithms for approximate congruence.

For a specified metric and isometry class, let $\varepsilon_{opt}(A,B)$ denote the minimum value of ε such that A and B are ε-congruent. Occasionally, we will designate the isometry class implied by this notation by writing $\varepsilon^{T}_{opt}(A,B)$, $\varepsilon^{\mathrm{Rd}}_{opt}(A,B)$, and $\varepsilon^{I}_{opt}(A,B)$, respectively, for the cases of translation, rotation about d, and general isometry. Intuitively, we would believe that it is more difficult to test for ε-congruence if ε is close to $\varepsilon_{opt}(A,B)$. We would be willing to accept

an algorithm that correctly answers queries for values of ε not near $\varepsilon_{opt}(A,B)$, but sometimes chooses not to answer when ε is near $\varepsilon_{opt}(A,B)$, if that algorithm provides substantial time savings over the complete decision algorithms, which always take a decision. We say that an algorithm that tests for ε-congruence is (α,β)-*approximate*, if, for any $\varepsilon \notin [\varepsilon_{opt}(A,B) - \alpha, \varepsilon_{opt}(A,B) + \beta]$, it correctly answers a query, and for $\varepsilon \in [\varepsilon_{opt}(A,B) - \alpha, \varepsilon_{opt}(A,B) + \beta]$, it either answers correctly or chooses not to answer. We call $[\varepsilon_{opt}(A,B) - \alpha, \varepsilon_{opt}(A,B) + \beta]$ the *indecision interval.* When we say that we have an "(α,β)-approximate algorithm for testing approximate congruence", we use the word approximate twice, but with two different meanings. The latter instance refers to the fact that we are trying to put the points of one set within the ε-balls of the other; the former refers to the indecision interval. An (α,β)-approximate algorithm has the desirable property that it will not return an incorrect answer; if it is not sure, it will simply say that it does not know the answer.

We present algorithms for three different classes of isometries—translation, rotation with fixed center, and general isometry. While we will assume that all distances in the paper represent the Euclidean metric, our algorithms apply equally well to any L_p metric. For the translation case we present a (γ,γ)-approximate algorithm for testing ε-congruence that runs in time $O(n^{1.5}(\varepsilon/\gamma)^4)$. For the case of a rotation with a fixed center d, we present two different (γ,γ)-approximate algorithms, which run in time $O(n^3(\varepsilon/\gamma)^2)$and $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^2)$, respectively, where $\delta = \max\{\mathrm{diam}(A), \mathrm{diam}(B)\}$. Finally, we show that any (γ,γ)-approximate algorithm for rotation with a fixed center can be converted to a (γ,γ)-approximate algorithm for general isometry with an additional factor of $(\varepsilon/\gamma)^2$ appearing in the time bound, giving us algorithms for general isometry of time $O(n^3(\varepsilon/\gamma)^4)$ and $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^4)$. We will show that for our translation algorithm and for one of our rotation algorithms (and therefore for our general isometry algorithm), it is possible to replace functions of n in the time bound by functions of (δ/γ). For example, the translation problem can be solved in time $O(n + (\delta/\gamma)^4(\varepsilon/\gamma)^4 \log(\delta/\gamma))$. These alternate time bounds imply that our algorithms are especially efficient on dense data.

It is possible to remove the indecision from our approximate decision algorithms, and obtain complete decision algorithms whose time-complexity is dependent on the difficulty of the problem instance. Specifically, if we think of $K_\varepsilon = |\varepsilon_{opt}(A,B) - \varepsilon|$ as the "difficulty parameter", then each of our approximate decision algorithms can be transformed into a complete decision algorithm, with K_ε replacing γ in the time bound. This gives decision algorithms for testing ε-congruence, with time bounds of $O(n^{1.5}(\varepsilon/K_\varepsilon)^4)$for the translation case, $O(n^3(\varepsilon/K_\varepsilon)^2)$and $O(n^{1.5}(\delta/K_\varepsilon)(\varepsilon/K_\varepsilon)^2)$for rotation, and $O(n^3(\varepsilon/K_\varepsilon)^4)$and $O(n^{1.5}(\delta/K_\varepsilon)(\varepsilon/K_\varepsilon)^4)$for the general case.

As mentioned earlier, the best known algorithm for approximate congruence has running time $O(n^8)$ [AMWW]. For approximate congruence enabled by translations, several $O(n^6)$ algorithms are known [AMWW, He, Iw]. Approximate congruence enabled by a rotation with a given center can be decided in time $O(n^4)$ [Iw, Sc1, Sc2], and approximate congruence enabled by a reflection in time $O(n^6)$ [Iw]. An algorithm in [AMWW] solves in time $O(n^6 \log n)$ the ε-congruence optimization problem under translation, which asks us to compute $\varepsilon^T_{opt}(A,B)$. A special case of the ε-congruence problem is that of *given correspon-*

Decision algorithms for approximate point set congruence without given correspondence

	known results				new results
	decision algorithms		approximate decision algorithms		
		disjoint ε-balls			
translations	$O(n^6)$ [AMWW, He, Iw]	$O(n \log n)$ [AMWW]		$O(n^{2.5})$ [Sc1, Sc2]	$O(n^{1.5})$
rotations	$O(n^4)$ [Iw, Sc1, Sc2]	$O(n^2)$ [AKMSW]			$O(n^3)$ $O(n^{1.5}(\delta/\varepsilon))$
isometries	$O(n^8)$ [AMWW]	$O(n^4 \log n)$ [AKMSW]	$O(n^2 \log n)$ [Be]	$O(n^4)$ [Sc1, Sc2]	$O(n^3)$ $O(n^{1.5}(\delta/\varepsilon))$

δ is the maximum of the diameters of the two point sets. The approximate decision algorithms are (γ, γ)-approximate algorithms, where γ is ε divided by some constant.

Table 1: new results compared to known algorithms

dence, where the bijection ℓ is given; this problem is studied in [AMWW, ISI, Iw]. In [AMZ] and [Sp], algorithms are given that generalize the approach of [AMWW] by considering sets A and B of unequal cardinality, and by generalizing the metric. Specifically, [AMZ] allows the "noise regions" (i.e. the ε-balls around the points of A) to be arbitrary nonconvex polygons; it also considers piecewise linear noise functions. In [AKMSW], algorithms are given for the decision problem for numerous metrics under various classes of isometries and under similarity (an isometry plus a change of scale), but with the assumption that the ε-balls around point set A are pairwise-disjoint or have limited overlap. In [AKMSW], [AMZ], and [Sp], combinatorial upper and lower bounds are given on the number of distinct bijections that can satisfy the decision problem. A problem related to ε-congruence is that of finding a translation that minimizes the Hausdorff distance between two point sets; this problem is studied in [HuKe] and [HKS]. Approximate decision algorithms have been studied by Schirra [Sc2], who gives (γ, γ)-approximate algorithms that test for ε-congruence under translation in $O(n^{2.5}(\varepsilon/\gamma)^2)$ time and under general isometry in $O(n^4(\varepsilon/\gamma)^2)$ time, and by Heffernan [He], who gives an $O(n^3(\varepsilon/\gamma)^6)$ time translation algorithm. Behrends [Be] considers approximate decision algorithms for pairwise disjoint ε-balls. We compare these

results with our new results in Table 1.

The remainder of the paper is organized as follows. Section 2 describes our translation results, and reveals much of our technique for constructing (γ, γ)-approximate algorithms. Section 3 describes two different approaches that test for ε-congruence under rotation around a fixed center, and Section 4 shows how to extend these results to the case of general isometry. In Section 5 we give an example that motivates the development of approximate decision algorithms for testing ε-congruent. For the decision algorithm for ε-congruence under general isometry of [AMWW], we show that the worst-case time bound of $O(n^8)$ is tight. Section 6 is the conclusion.

2 Translation

In this section we give our algorithms for the translation case. Most of the decision algorithms for approximate congruence without correspondence use the same scheme. They compute a finite set of candidates for the isometry in demand, and ask whether any candidate isometry enables approximate congruence for the planar point sets A and B with tolerance ε. The decision problem of whether a fixed isometry I enables approximate congruence is reduced to a maximum matching problem in a bipartite graph, which is reduced to a max-flow problem. Each point in A and B is represented by a node in a bipartite graph. The node representing point a_i and the node representing b_j are connected by an edge iff $dist(I(a_i), b_j) \leq \varepsilon$. This graph has a perfect matching iff there is a labeling ℓ such that I and ℓ enable ε-congruence [AMWW]. In this section we obtain fast, approximate decision algorithms for ε-congruence under translation, by replacing A and B with slightly perturbed, degenerate versions. Thereby we sacrify precision, but enjoy improved run-times by reducing the number of candidate translations and by limiting the number of edges in each max-flow problem.

We state first a lemma that we will use in constructing the set of candidate translations. Let c_A denote the centroid of point set A and c_B denote the centroid of point set B. We have [Sc2]

Lemma 1 *If I enables μ-congruence for A and B then $dist(I(c_A), c_B) \leq \mu$.*

The basis of our imposed structure is an orthogonal grid with width $\lambda = \sqrt{2}\gamma/5$ that we superimpose onto the plane. Note that every point of the plane is within distance $\gamma/5$ of the intersection of two grid lines, which we call a *grid point*. Therefore, if a translation T enables μ-congruence for A and B, then by Lemma 1 there exists a grid point g such that $dist(T(c_A), g) \leq \gamma/5$ and $dist(c_B, g) \leq \mu + \gamma/5$. Accordingly, the $O((\varepsilon/\gamma)^2)$ grid points within distance $\varepsilon + \gamma/5$ of c_B will form the set of candidate translations. For $g \in \mathbb{R}^2$ let $T_{c_A g}$ be the translation that maps c_A onto g.

Lemma 2 *If A and B are μ-congruent by a translation then there is grid point g with $dist(g, c_B) \leq \mu + \gamma/5$ such that $T_{c_A g}$ enables approximate congruence with tolerance $\mu + \gamma/5$ for A and B.*

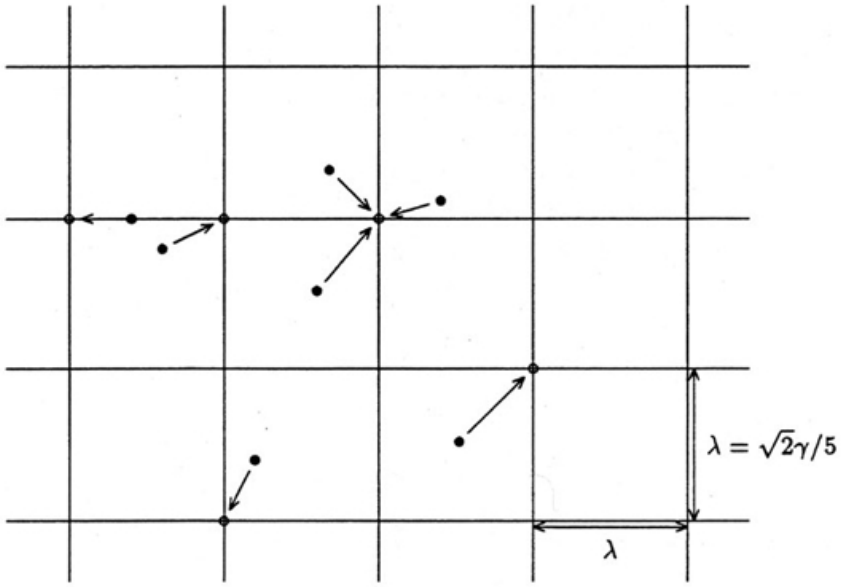

Figure 1: Moving points to grid points

We now use the grid to impose structure onto the point sets A and B. We move each point of A and B to its nearest grid point, with ties broken arbitrarily. Let $A^{\#} = \{a_1^{\#}, \ldots, a_n^{\#}\}$ and $B^{\#} = \{b_1^{\#}, \ldots, b_n^{\#}\}$ be the multisets corresponding to A and B, respectively, obtained in this manner, and let $A^{\circ} = \{a_1^{\circ}, \ldots, a_h^{\circ}\}$ and $B^{\circ} = \{b_1^{\circ}, \ldots, b_m^{\circ}\}$ be the sets of distinct points in $A^{\#}$ and $B^{\#}$ (see Figure 1).

Lemma 3 *If A and B are μ-congruent by a translation then $A^{\#}$ and $B^{\#}$ are approximately congruent with tolerance $\mu + 2\gamma/5$ by that translation.*

Proof. Each point is moved out of its place by a distance of $\gamma/5$ at most. □

Let $c_{A^{\#}}$ be the centroid of $A^{\#}$ and let $T_{c_{A^{\#}}g}$ be the translation that maps $c_{A^{\#}}$ onto g. Combining Lemma 2 and Lemma 3 we get

Lemma 4 *If A and B are μ-congruent by a translation then there is a grid point g with $dist(g, c_B) \leq \mu + \gamma/5$ such that $T_{c_{A^{\#}}g}$ enables approximate congruence with tolerance $\mu + 3\gamma/5$ for $A^{\#}$ and $B^{\#}$.*

Hence we can conclude that A and B are not ε-congruent if for all grid points g with $dist(g, c_B) \leq \varepsilon + \gamma/5$, the translation $T_{c_{A^{\#}}g}$ does not enable approximate congruence with tolerance $\varepsilon + 3\gamma/5$ for $A^{\#}$ and $B^{\#}$. On the other hand:

Lemma 5 *If $A^{\#}$ and $B^{\#}$ are approximately congruent with tolerance μ by $T_{c_{A^{\#}}g}$ for some gridpoint g then A and B are approximately congruent with tolerance $\mu + 2\gamma/5$ by that translation.*

Lemmas 4 and 5 form the basis of a (γ, γ)-approximate algorithm, which is given in Table 2. By Lemma 5, the point sets A and B are approximately congruent with tolerance ε by a translation if $A^\#$ and $B^\#$ are approximately congruent with tolerance $\varepsilon - 2\gamma/5$ by $T_{c_{A^\# g}}$ for some g; thus, any YES answer is correct. Lemma 4 states that if there is no grid point g among those considered such that $T_{c_{A^\# g}}$ enables congruence with tolerance $\varepsilon + 3\gamma/5$, then A and B are not ε-congruent; thus, a NO answer must be correct. We see that if the algorithm in Table 2 gives an answer, then that answer is correct.

[1] possible $\leftarrow$ **false**
[2] Compute $A^\#$ and $B^\#$;
[3] **for all** g with $dist(g, c_B) \leq \varepsilon + \gamma/5$ **do**
[4] **if** $A^\#$ and $B^\#$ are approximately congruent with tolerance $\varepsilon + 3\gamma/5$ by $T_{c_{A^\# g}}$ **then return** possible $\leftarrow$ **true fi**
[5] **if** $A^\#$ and $B^\#$ are approximately congruent with tolerance $\varepsilon - 2\gamma/5$ by $T_{c_{A^\# g}}$ **then return** YES **fi od**;
[6] **if** possible **then return** DON'T KNOW **else return** NO **fi**

Table 2: (γ, γ)-approximate algorithm for ε-congruence by translations

Lemma 6 *The algorithm in Table 2 is a (γ, γ)-approximate algorithm for approximate congruence with tolerance ε enabled by a translation.*

Proof. We argued above that if the algorithm returns an answer, then that answer is correct. We must show the algorithm returns an answer for any ε outside of the indecision interval. Let $\varepsilon < \varepsilon^T_{opt}(A, B) - \gamma$. Then A and B are not approximately congruent with tolerance $\varepsilon + \gamma$. Hence there is no grid point g such that $T_{c_{A^\# g}}$ enables approximate congruence with tolerance $\varepsilon + 3\gamma/5$ for $A^\#$ and $B^\#$, because this would imply that A and B are approximately congruent with tolerance $\varepsilon + 3\gamma/5 + 2\gamma/5$ by Lemma 5. Now let $\varepsilon \geq \varepsilon^T_{opt}(A, B) + \gamma$. Then A and B are approximately congruent with tolerance $\varepsilon - \gamma$. By Lemma 4 there is a grid point g such that $A^\#$ and $B^\#$ are approximately congruent with tolerance $\varepsilon - \gamma + 3\gamma/5 = \varepsilon - 2\gamma/5$ enabled by $T_{c_{A^\# g}}$. □

It remains to explain how to test whether $T_{c_{A^\# g}}$ enables μ-congruence for $A^\#$ and $B^\#$. To represent graph networks, we will use the notation $G = (V, E, c)$, where V and E are the vertex and edge sets, respectively, and $c : E \rightarrow \mathbb{N}$ gives the edge capacities. Consider the network

$$G(T_{c_{A^\# g}}, \mu, A^\#, B^\#) = (\{s, t\} \cup U \cup V, E, c)$$

where $U = \{u_1, \ldots, u_n\}$ represents the points in $A^\#$ and $V = \{v_1, \ldots, v_n\}$ represents the points in $B^\#$,

$$
\begin{aligned}
E &= \{(s,u_i) \mid 1 \le i \le n\} \\
&\cup \{(v_j,t) \mid 1 \le j \le n\} \\
&\cup \{(u_i,v_j) \mid dist(T_{c_{A^\# g}}(a_i^\#), b_j^\#) \le \mu\}, \\
\text{and } c(s,u_i) &= 1 \\
c(v_j,t) &= 1 \\
c(u_i,v_j) &= 1.
\end{aligned}
$$

If $G(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ has a max-flow of size n, then $T_{c_{A^\# g}}$ enables μ-congruence for $A^\#$ and $B^\#$. However, in the general case we have no better worst case bound than $O(n^2)$ for the number of edges. Using standard techniques [Me, Ta] this gives a $O(n^{2.5})$ worst case bound for each test.

Recently Feder and Motwani [FM] presented a method for speeding-up graph algorithms. They search for complete bipartite subgraphs and substitute them by simpler structures. The resulting graphs are called *compressed*. With their technique a max-flow for $G(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ can be computed in time $O(n^{2.5}/\log n)$.

Replacing A and B by a slightly degenerate version makes the idea of [FM] more fruitful for testing approximate congruence. Let $U_k^\circ = \{u_i \mid a_i \text{ is moved onto } a_k^\circ\} \subset U$ and $V_l^\circ = \{v_j \mid b_j \text{ is moved onto } b_l^\circ\} \subset V$. If $T_{c_{A^\# g}}$ moves gridpoint a_k° into the μ-neighborhood of b_l°, this generates a complete bipartite subgraph with node sets U_k° and V_l°. For every such bipartite clique, we remove the edges in $U_k^\circ \times V_l^\circ$, add a new node w, and add edges $\{(u_i,w) \mid u_i \in U_k^\circ\}$ and $\{(w,v_j) \mid v_j \in V_l^\circ\}$ with capacity 1 each. Thereby we replace $|U_k^\circ| \cdot |V_l^\circ|$ old edges by $|U_k^\circ| + |V_l^\circ|$ new edges. We call the resulting graph $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$.

A max-flow in $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ corresponds to a max-flow in $G(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$. With Dinic's algorithm a max-flow in $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ can be computed in $O(\sqrt{n})$ phases, each in time proportional to the new number of edges. (It is a well known fact that Dinic's algorithm takes $O(\sqrt{\text{number of nodes}})$ phases in a simple 0-1 network. A node is called simple, if it has indegree 1 or outdegree 1, and a network is called simple, if all nodes are simple. Note that the nodes in W are not simple in the compression of a graph. Analogously to the proof on the number of phases of Dinic's algorithm for simple 0-1 networks in [Me, page 81] it can be shown, that Dinic's algorithm takes $O(\sqrt{\text{number of simple nodes}})$ in a 0-1 network, if no edge connects non-simple nodes).

The μ-neighborhood of any point contains $O((\mu/\gamma)^2)$ grid points. Hence each point of $A^\# \cup B^\#$ contributes to $O((\mu/\gamma)^2)$ bipartite cliques. Hence the number of edges in graph $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ is $O(n(\mu/\gamma)^2)$.

For the construction of $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ we use range searching. For a set of n points a data structure of size $O(n)$ can be built in $O(n \log n)$ time for fixed radius disk queries, such that each query has time complexity $O(\log n + k)$, where k is the size of the output [LS]. For each point a° in A° we ask for the points of B° included in the disk with radius $\mu \in \{\varepsilon + 2\gamma/5, \varepsilon - 3\gamma/5\}$ and center $T_{c_{A^\# g}}(a^\circ)$. We have output size $k = O((\mu/\gamma)^2)$ for each query. Summing over all queries gives run-time $O(n(\log n + (\mu/\gamma)^2))$.

If space is not important, we could alternatively use standard range searching [PS] with squares of side length 2μ containing the query disks and test each reported point for inclusion in the query disk. This approach has the same time bounds for the two dimensional queries considered here, but space complexity $O(n \log n)$. However, it can be used in higher dimensions, too. For queries in $\mathbb{R}^d$, preprocessing time is $O(n \log^{d-1} n)$, query time $O(\log n + k)$, and space complexity $O(n \log^{d-1} n)$ [PS].

Theorem 1 *The algorithm in Table 2 has running time* $O(n^{1.5}(\varepsilon/\gamma)^4)$.

Proof. $A^\#$ and $B^\#$ can be constructed in $O(n)$ time, and there are $O((\varepsilon/\gamma)^2)$ grid points g for which $T_{c_{A^\# g}}$ is tested. Each $G^\#_{\text{comp}}(T_{c_{A^\# g}}, \mu, A^\#, B^\#)$ can be constructed in time $O(n \log n + n(\varepsilon/\gamma)^2)$ and a max-flow can be computed in time $O(\sqrt{n} \cdot n(\varepsilon/\gamma)^2)$. □

We now give an alternative for testing whether $T_{c_{A^\# g}}$ enables μ-congruence. We denote the number of points in A moved onto a_i° by n_i^A and the number of points in B moved to b_j° by n_j^B. We consider the following network

$$G^\circ(T_{c_{A^\# g}}, \mu, A^\circ, B^\circ) = (\{s,t\} \cup \{u_1, \ldots, u_h\} \cup \{v_1, \ldots, v_m\}, E^\circ, c)$$

where $\{u_1, \ldots, u_h\}$ represents the points in A° and $\{v_1, \ldots, v_m\}$ represents the points in B°,

$$\begin{aligned} E^\circ &= \{(s,u_i) \mid 1 \le i \le h\} \\ &\cup \{(v_j,t) \mid 1 \le j \le m\} \\ &\cup \{(u_i,v_j) \mid dist(T_{c_{A^\# g}}(a_i^\circ), b_j^\circ) \le \mu\}, \\ \text{and } c(s,u_i) &= n_i^A \\ c(v_j,t) &= n_j^B \\ c(u_i,v_j) &= \min(n_i^A, n_j^B). \end{aligned}$$

We claim that it suffices to compute a max-flow on $G(T_{c_{A^\# g}}, \mu, A^\circ, B^\circ)$:

Lemma 7 *$G(T_{c_{A^\# g}}, \mu, A^\circ, B^\circ)$ has a flow of capacity n iff the multisets $A^\#$ and $B^\#$ are μ-congruent by $T_{c_{A^\# g}}$.*

Proof. It is easy to see that a labeling implies a flow of capacity n. For the converse, consider the well-known fact that there exists an integral max-flow solution if all edge capacities are integral [Me, Ta]. Hence if there is a max-flow with capacity n, then there is a max-flow of capacity n with integer flow on each edge. Now it is easy to derive a labeling $\ell : A^\# \to B^\#$ from such a flow. Consider the edges one by one. If the max-flow pushes f units over edge (u_i, v_j) then ℓ maps f of the unused copies of a_i° to f unused copies of b_j°. Then these copies are marked as used and the next edge is considered. The capacities in the network guarantee that the number of unused copies is always sufficient. □

It remains to establish the time-complexity of computing a max-flow on a graph G° generated by the algorithm. There is a wealth of literature on the max-flow problem (for

an overview, see [AOT, CHM, Ta]), with numerous algorithms whose complexities depend principally on N and M, the number of nodes and edges, respectively, of G°. We will establish bounds on N and M, and state the complexities that we can obtain with max-flow algorithms. Again we use the fact that the ε-ball around a point of A° or B° contains only $O((\varepsilon/\gamma)^2)$ grid points, so a node of G° can be incident to only this many edges. If we use the trivial bound of $2n$ for N, then $M = O(n(\varepsilon/\gamma)^2)$. An alternate bound on N arises by considering the diameters of A and B. If $\delta = \max\{\text{diam}(A), \text{diam}(B)\}$, then there can be only $O((\delta/\gamma)^2)$ points of A° and B°, giving that N has this bound, and $M = O((\delta/\gamma)^2(\varepsilon/\gamma)^2)$.

The method of Sleator and Tarjan [Sl, ST] computes max-flows in time $O(NM \log N)$, and yields an efficient bound of $O((\delta/\gamma)^4(\varepsilon/\gamma)^2 \log(\delta/\gamma))$ when $(\delta/\gamma)^2$ is small compared to n (note that the graph G° can be built in $O((\delta/\gamma)^2(\varepsilon/\gamma)^2)$ time, since this is the number of potential edges that must be checked). This bound is better than that given above when $n = \Omega((\delta/\gamma)^{8/3} \log^{2/3}(\delta/\gamma))$. We summarize these in Table 3.

graph	total time	range
$G^{\#}_{\text{comp}}$	$O(n^{1.5}(\varepsilon/\gamma)^4)$	$n < (\delta/\gamma)^{8/3} \log^{2/3}(\delta/\gamma)$
G°	$O(n + (\delta/\gamma)^4(\varepsilon/\gamma)^4 \log(\delta/\gamma))$	$n > (\delta/\gamma)^{8/3} \log^{2/3}(\delta/\gamma)$

Table 3: Time-bounds for the (γ, γ)-approximate algorithms under translation

The max-flow methods mentioned here compute an integral maximum flow, if the input graph has integral capacities. If a max-flow of size n is computed, it is, therefore, an integral flow, and corresponds to a matching of the point sets A and B. This implies that we solve something more than the decision problem: if we determine that the point sets are ε-congruent, we actually return a bijection ℓ and a translation T that enable ε-congruence.

We mention here that our results for the translation case extend easily to higher dimensions. We showed in this section that a (γ, γ)-approximate algorithm can be obtained by considering a set of points representing candidate translations, such that no point within distance ε of c_B is more than distance $\gamma/5$ from a candidate point. If we cover $\mathbb{R}^d$ with orthogonal grid points at width λ, then no point of $\mathbb{R}^d$ is more than distance $\sqrt{d}\lambda/2$ from a grid point. By setting $\sqrt{d}\lambda/2 = \gamma/5$, we see that we should choose $\lambda = 2\gamma/(5\sqrt{d})$ (note that for $d = 2$, this gives the value $\lambda = 2\gamma/(5\sqrt{2})$ used in this section). By Lemma 2, we need only consider translations that map c_A to a grid point within distance $\varepsilon + \gamma/5$ of c_B. There are $O((\varepsilon/\lambda)^d)$ such points, which for $d = O(1)$ gives:

Theorem 2 *The algorithm in Table 2 is a (γ, γ)-approximate algorithm for ε-congruence under translation for point sets in $\mathbb{R}^d$. The algorithm runs in time $O(n^{1.5}(\varepsilon/\gamma)^{2d})$ resp. $O(n + (\delta/\gamma)^{2d}(\varepsilon/\gamma)^{2d} \log(\delta/\gamma))$.*

3 Fixed center Rotations

In this section we give a (γ,γ)-approximate algorithm for testing approximate congruence of two sets of n points enabled by a rotation with a fixed rotation center d. We give two approximate decision algorithms, which we name the *orthogonal*, $\mathcal{R}_{ortho}$, and the *polar*, $\mathcal{R}_{polar}$, based on the manner in which each perturbs the point sets. Analogously to the previous section, each algorithm perturbs the two point sets A and B a little bit, and for each rotation around d from a finite set, solves a max-flow problem on an appropriate graph to test whether the rotation enables approximate congruence. The best known algorithms for this problem have run time $O(n^4)$ [Iw, Sc1, Sc2], and the orthogonal algorithm is a modification of the decision algorithm of [Sc1, Sc2] with run-time $O(n^3(\varepsilon/\gamma)^2)$. The polar algorithm has versions which run in time $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^2)$and $O(n+(\delta/\gamma)^5(\varepsilon/\gamma)^2\log(\delta/\gamma))$, where $\delta=\max\{\mathrm{diam}(A),\mathrm{diam}(B)\}$. Our collection of algorithms allows one to choose one based on the relative sizes of n and (δ/γ).

3.1 The orthogonal rotation algorithm

We begin the orthogonal rotation algorithm by superimposing a grid with separation $\lambda=\gamma/(2\sqrt{2})$ onto the plane. Then we move each point in A and B to its nearest grid point, to form the multisets $A^{\#}$ and $B^{\#}$, and the sets of distinct points A° and B°. For $\varphi\in[0,2\pi)$ let R^{φ} be the counterclockwise rotation by angle φ with center d. For each pair $(a^{\circ},b^{\circ})\in A^{\circ}\times B^{\circ}$ there is a circular interval $I^{\mu}_{(a^{\circ},b^{\circ})}\subseteq[0,2\pi)$ such that $dist(R^{\varphi}(a^{\circ}),b^{\circ})\le\mu$ iff $\varphi\in I^{\mu}_{(a^{\circ},b^{\circ})}$. Let L be the set of interval endpoints of all such intervals. Since the intervals are closed we have

Lemma 8 *$A^{\#}$ and $B^{\#}$ are μ-congruent iff there is a $\varphi\in L$ such that $A^{\#}$ and $B^{\#}$ are μ-congruent by R^{φ}.*

Let $A^{\circ}=\{a^{\circ}_1,\ldots,a^{\circ}_h\}$ and $B^{\circ}=\{b^{\circ}_1,\ldots,b^{\circ}_m\}$. Let n^A_i be the number of points in A moved onto a°_i and n^B_j the number of points in B moved to b°_j. A network

$$G^{\circ}(R^{\varphi},\mu,A^{\circ},B^{\circ})=(\{s,t\}\cup\{u_1,\ldots,u_h\}\cup\{v_1,\ldots,v_m\},E^{\circ},c)$$

is defined analogously to Section 2 with $T_{c_{A^{\#}g}}$ replaced by R^{φ}, i.e. $\{u_1,\ldots,u_h\}$ represents the points in A°, $\{v_1,\ldots,v_m\}$ represents the points in B°, $E^{\circ}=\{(s,u_i)\mid 1\le i\le h\}\cup\{(v_j,t)\mid 1\le j\le m\}\cup\{(u_i,v_j)\mid dist(R^{\varphi}(a^{\circ}_i),b^{\circ}_j)\le\mu\}$ and $c(s,u_i)=n^A_i$, $c(v_j,t)=n^B_j$, and $c(u_i,v_j)=\min(n^A_i,n^B_j)$. Recall that $\varepsilon^{\mathrm{Rd}}_{opt}(A,B)$ is the minimum value where A and B are approximately congruent by a rotation with center d.

Lemma 9 *The algorithm in Table 4 is a (γ,γ)-approximate algorithm for approximate congruence enabled by a rotation with a fixed center d.*

Proof. If A and B are μ-congruent by a rotation with center d, then $A^{\#}$ and $B^{\#}$ are approximately congruent with tolerance $\mu+\gamma/2$ by a rotation with center d. If $A^{\#}$ and $B^{\#}$ are μ-congruent by a rotation with center d, then A and B are approximately congruent with

$\mathcal{R}_{ortho}$

```
[01] procedure fcROT(μ, d, A°, B°)
[02] begin
[03]   L ← ∅;
[04]   for each pair (a°, b°) ∈ A° × B° do
       Compute I^μ_(a°,b°); Add the endpoints to L od
[05]   Sort L into a circular list;
[06]   Let φ_o be an arbitrary element in L;
[07]   Compute G(R^φ_o, μ, A°, B°);
[08]   if G(R^φ_o, μ, A°, B°) has a max-flow of size n
       then return YES fi;
[09]   repeat φ ← succ(φ);
[10]   Update G(R^φ, μ, A°, B°) and compute a new max-flow;
[11]   if the max-flow has capacity n then return YES fi
[12]   until succ(φ) = φ_o;
[13]   return NO
[14] end;
[15] begin
[16] Compute A# and B#; Compute A° and B°;
[17] if fcROT(ε + γ/2, d, A°, B°) returns NO
     then return NO fi;
[18] if fcROT(ε − γ/2, d, A°, B°) returns YES
     then return YES fi;
[19] return DON'T KNOW
[20] end
```

Table 4: the orthogonal rotation algorithm

tolerance $\mu + \gamma/2$ by a rotation with center d, because the points are displaced to points which have distance $\gamma/4$ at most. Hence if the answer is YES or NO, this is correct. For $\varepsilon < \varepsilon^{\mathrm{Rd}}_{opt}(A, B) - \gamma$ the point sets $A^{\#}$ and $B^{\#}$ are not approximately congruent with tolerance $\varepsilon + \gamma/2$ by a rotation with center d, because A and B are not approximately congruent with tolerance $\varepsilon + \gamma$. So the output has to be NO. For $\varepsilon \geq \varepsilon^{\mathrm{Rd}}_{opt}(A, B) + \gamma$ the points sets A and B are approximately congruent with tolerance $\varepsilon - \gamma$ by a rotation with center d and hence the points sets $A^{\#}$ and $B^{\#}$ are approximately congruent with tolerance $\varepsilon - \gamma/2$ by a rotation with center d. □

The actual run-time of this algorithm depends upon the manner in which we compute the max-flows. In addition to the graph and the flow, we maintain the residual graph with respect to the present flow. We have a max-flow in $G_{old} = G(R^{pred(\varphi)}, \mu, A^{\circ}, B^{\circ})$ and we are looking for a max-flow in $G_{new} = G(R^{\varphi}, \mu, A^{\circ}, B^{\circ})$. First we delete the edges of G_{old} which are not in G_{new}. Then we insert the edges in G_{new} which are not in G_{old}. An edge is deleted

by decreasing its capacity in unit steps until the capacity becomes zero. Let (u_j, v_i) be the edge to be deleted. If, after a decrease of the capacity by one, the flow over this edge is larger then the capacity, we decrease the flow on the path $s - u_j - v_i - t$ by one. Since the difference between the capacity of a max-flow in the new network and the present flow is at most one, it is sufficient to search a path connecting s and t in the residual graph. This takes time proportional to the number of edges in the present graph. Similarly edges are inserted by increasing their capacity in unit steps.

Let M be a bound on the number of edges in one of the networks considered in the loop of procedure fcROT in Table 4. Let S_φ denote the sum of the capacities of the edges which are in $G(R^\varphi, \mu, A^\circ, B^\circ)$ and not in $G(R^{pred(\varphi)}, \mu, A^\circ, B^\circ)$ or vice versa. Then the time for the update is $O(S_\varphi M)$. Since each edge is inserted and deleted at most once and the sum of the capacities of all possible edges connecting nodes in $\{u_1, \ldots, u_h\}$ and $\{v_1, \ldots, v_m\}$ is n^2 at most, we have $\sum_{\varphi \in L} S_\varphi \leq 2n^2$. For each $\varphi \in [0, 2\pi)$ and arbitrary $a^\circ \in A^\circ$ the number of points in B° with distance at most μ to $R^\varphi(a^\circ)$ is $O((\mu/\gamma)^2)$. Hence $M = O(n(\mu/\gamma)^2)$ and the time for all updates is $O(n^3(\mu/\gamma)^2)$. The initial max-flow can be computed in time $O(n^2(\mu/\gamma)^2)$ [FF].

Theorem 3 *The (γ, γ)-approximate algorithm in Table 4 for approximate congruence by fixed center rotations has running time $O(n^3(\varepsilon/\gamma)^2)$.*

Proof. By discussion above. □

3.2 The polar rotation algorithm

We now give an alternate (γ, γ)-approximate algorithm for determining if two point sets A and B are ε-congruent under rotation around a fixed center d. The polar algorithm runs in time $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^2)$, where δ is the larger of the diameters of A and B. The polar rotation algorithm is sub-quadratic in n, as opposed to cubic for the orthogonal rotation algorithm, and is asymptotically superior when $\delta = o(n^{1.5}\gamma)$. The polar algorithm also provides an $O(n + (\delta/\gamma)^5(\varepsilon/\gamma)^2 \log(\delta/\gamma))$ time method for the rotation problem, thereby offering further improvements when (δ/γ) is small relative to n. Like the orthogonal rotation algorithm, this one extends easily to an algorithm for the general isometry case, with an additional $O((\varepsilon/\gamma)^2)$ factor in the run time.

The spirit of the algorithm is to discretize the interval of all possible rotations, where the number of subintervals depends on δ but not n. At first our discussion will assume that $dist(d, p) \leq \delta$ for all $p \in A \cup B$; later we will see that this assumption is not necessary, and that the time bound actually improves when d is not near the two point sets. We will cover the plane with two grids, one polar and one orthogonal. We describe first the polar grid, P. Let $\lambda = \gamma/(5\sqrt{2})$. Place concentric circles centered at d with radii $\lambda, 2\lambda, \ldots, k\lambda$, so that all points of $A \cup B$ are within the second-largest circle; we call the largest circle the *bounding circle*, and we denote it and its interior by $\mathcal{B}$. Now add a collection of evenly-spaced rays emanating from d, such that the sub-arc on the bounding circle between two neighboring

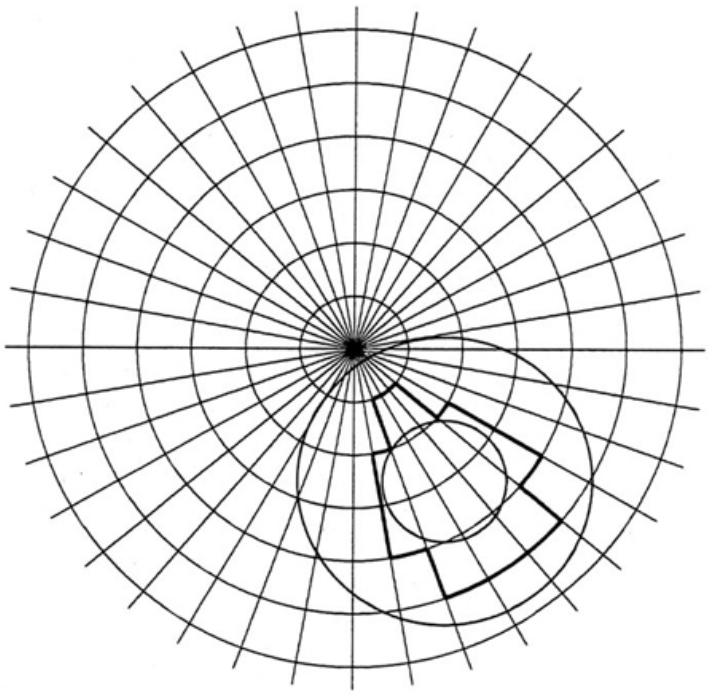

Figure 2: $b^{\#}(P, \mu)$

rays is of length approximately λ. Since $k \leq \lceil \delta/\lambda \rceil$, there are at most $2\pi\delta/\lambda$ rays. Together, the rays and circles form a polar grid P.

We also superimpose an orthogonal grid. This grid consists of horizontal and vertical lines spaced λ units apart. We denote by C the set of orthogonal grid points in $\mathcal{B}$ (we can ignore everything not in $\mathcal{B}$). We map each point $c \in C$ to the nearest grid point of the polar grid P to form the set C°. Note that no point $c \in C$ is more than $(\sqrt{2}/2)\lambda$ units from its corresponding point $c^{\circ} \in C^{\circ}$. We now map each point $p \in A \cup B$ to the nearest point of C°, to form the multiset $A^{\#}$ and $B^{\#}$, and the set of distinct points A° and B° resp.. Since each point $p \in A \cup B$ is within distance $(\sqrt{2}/2)\lambda$ of a point of the orthogonal grid C, it must be within distance $\sqrt{2}\lambda$ of a point of C°, giving $dist(p, p^{\#}) \leq \sqrt{2}\lambda = \gamma/5$ for all $p \in A \cup B$. Given a point p, let $\mathcal{U}(p, \mu)$ be the ball of points within distance μ of p. Note that $\mathcal{U}(p, \mu + (\sqrt{2}/2)\lambda)$ contains only $O((\mu/\lambda)^2)$ points of C, so only this many points of C°, and therefore of A°, can be in $\mathcal{U}(p, \mu)$. We summarize some properties:

- For every $p \in A \cup B$, $dist(p, p^{\#}) \leq \sqrt{2}\lambda = \gamma/5$.
- Each point $p^{\#} \in A^{\#} \cup B^{\#}$ is on a ray and a circle of P.
- A ball of radius μ contains only $O((\mu/\gamma)^2)$ elements of A° and B°.

Let $b(P, \mu)$ denote the smallest region containing $\mathcal{U}(b, \mu) \cap \mathcal{B}$ whose boundary lies entirely on the rays and circles of P. (see Figure 2). We claim that $b(P, \mu) \subset \mathcal{U}(b, \mu + \gamma/5)$, for

all $b \in B$. If this were not true, we would have a segment s in $\mathcal{B}$, with one endpoint on the boundary of $\mathcal{U}(b,\mu) \cap \mathcal{B}$ and the other on the boundary of $\mathcal{U}(b,\mu+\gamma/5)$, that does not intersect P. Since s lies in $\mathcal{B}$ and does not intersect P, it has length less than $\sqrt{2}\lambda = \gamma/5$. But s has endpoints on the boundaries of $\mathcal{U}(b,\mu)$ and $\mathcal{U}(b,\mu+\gamma/5)$, implying that its length is at least $\gamma/5$, a contradiction. Therefore we have

$$(\mathcal{U}(b,\mu) \cap \mathcal{B}) \subset b(P,\mu) \subset \mathcal{U}(b,\mu+\gamma/5).$$

We will say that $A^\#$ and $B^\#$ are *P-congruent with tolerance μ* if there exists a bijection $\ell : A^\# \to B^\#$ and a rotation R around d such that $R(a^\#) \in b^\#(P,\mu)$, where $b^\# = \ell(a^\#)$, for all $a^\# \in A^\#$.

Lemma 10 *If $A^\#$ and $B^\#$ are P-congruent with tolerance μ under bijection ℓ and rotation R, then A and B are congruent with tolerance $\mu + 3\gamma/5$ under ℓ and R.*

Proof. Suppose that $A^\#$ and $B^\#$ are P-congruent with tolerance μ under ℓ and R. We know that $b^\#(P,\mu) \subset \mathcal{U}(b^\#, \mu+\gamma/5)$ for all $b^\# \in B^\#$, implying that $A^\#$ and $B^\#$ are congruent with tolerance $\mu + \gamma/5$. Since $dist(b, b^\#) \leq \gamma/5$, we have $b^\#(P,\mu) \subset \mathcal{U}(b, \mu + 2\gamma/5)$. But for every $a \in A$, $dist(a, a^\#) \leq \gamma/5$, implying that $dist(R(a), R(a^\#)) \leq \gamma/5$, so A and B are $(\mu + 3\gamma/5)$-congruent. □

Lemma 11 *If A and B are congruent with tolerance μ under bijection ℓ and rotation R, then $A^\#$ and $B^\#$ are P-congruent with tolerance $\mu + 2\gamma/5$ under ℓ and R.*

Proof. Consider ℓ and R that admit μ-congruence for A and B. If $dist(\ell(a), R(a)) \leq \mu$, then $dist(\ell(a), R(a^\#)) \leq \mu + \gamma/5$. Since $\mathcal{U}(b, \mu+\gamma/5) \subset \mathcal{U}(b^\#, \mu+2\gamma/5) \subset b^\#(P, \mu+2\gamma/5)$, we have that $A^\#$ and $B^\#$ are P-congruent with tolerance $\mu + 2\gamma/5$ under ℓ and R. □

In Table 5, we give our approximate decision algorithm, which assumes that we have a sub-procedure TPc to test for P-congruence.

$\mathcal{R}_{polar}$

[1] Compute $A^\#$ and $b^\#(P, \varepsilon + 2\gamma/5)$, $b^\#(P, \varepsilon - 3\gamma/5)$ for all $b^\# \in B^\#$;
[2] **if** TPc($A^\#, B^\#, P, \varepsilon + 2\gamma/5$) returns NO
then return NO **fi**;
[3] **if** TPc($A^\#, B^\#, P, \varepsilon - 3\gamma/5$) returns YES with ($\ell$,$R$)
then return YES with (ℓ,R) **fi**;
[4] **return** DON'T KNOW

Table 5: the polar rotation algorithm

Lemma 12 *The procedure in Table 5 is a (γ,γ)-approximate algorithm for approximate congruence enabled by a rotation with a fixed center d.*

Proof. If the procedure returns a pair (ℓ, R), it is because ℓ and R enable P-congruence of $A^\#$ and $B^\#$ with tolerance $\varepsilon - 3\gamma/5$, so by Lemma 10, ℓ and R enable congruence of A and B with tolerance $(\varepsilon - 3\gamma/5) + 3\gamma/5 = \varepsilon$. If the procedure says that A and B are not ε-congruent, it is because $A^\#$ and $B^\#$ are not P-congruent with tolerance $\varepsilon + 2\gamma/5$; but by Lemma 11, this implies that A and B are not congruent with tolerance ε. Therefore, if the algorithm returns an answer, it is correct.

Suppose $\varepsilon > \varepsilon^{\mathrm{Rd}}_{opt}(A, B) + \gamma$. Then A and B are congruent with tolerance $\varepsilon - \gamma$, implying that $A^\#$ and $B^\#$ are P-congruent with tolerance $(\varepsilon - \gamma) + 2\gamma/5 = \varepsilon - 3\gamma/5$. Therefore, when we call $\mathrm{TPc}(A^\#, B^\#, P, \varepsilon - 3\gamma/5)$ we get ℓ and R that enable ε-congruence of A and B.

Suppose $\varepsilon < \varepsilon^{\mathrm{Rd}}_{opt}(A, B) - \gamma$. Then A and B are not congruent with tolerance $\varepsilon + \gamma$, implying that $A^\#$ and $B^\#$ cannot be P-congruent with tolerance $\varepsilon + 2\gamma/5$. Therefore, when we call $\mathrm{TPc}(A^\#, B^\#, P, \varepsilon + 2\gamma/5)$, we will conclude that A and B are not ε-congruent. □

All that remains is to discuss how we efficiently determine P-congruence. If we fix a ray of P as direction 0, then each ray of P represents a rotation of φ radians, where φ is the angle between the ray and the 0 direction. The boundary of a region $b^\#(P, \mu)$, $b^\# \in B^\#$, consists of two types of parts: pieces of rays and pieces of arcs. Because the rays are evenly spaced, and all points of $A^\#$ lie on a ray, points of $A^\#$ enter and leave a region $b^\#(P, \mu)$ only at rotations corresponding to rays. Since the regions $b^\#(P, \mu)$, $b^\# \in B^\#$, are closed, it suffices to consider only those directions φ corresponding to rays. We can define a max-flow graph for a rotation φ in the same style as in the previous sections. We let $A^\circ = \{a^\circ_1, \ldots, a^\circ_h\}$ and $B^\circ = \{b^\circ_1, \ldots, b^\circ_m\}$. Let A°_k be the set of points of A moved to a°_k and B°_l be the set of points of B moved to b°_l. We define the network

$$G^\#_{\mathrm{comp}}(R^\varphi, P, \mu, A^\#, B^\#) = (\{s, t\} \cup U \cup V \cup W, E, c)$$

where $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$ represent the points of $A^\#$ and the regions of $B^\#$, respectively, and

$$W = \{w^\otimes_{k,l} \mid R^\varphi(a^\circ_k) \in b^\circ_l(P, \mu)\}$$

represents complete bipartite subgraphs in $G(R^\varphi, P, \mu, A^\#, B^\#)$, and

$$\begin{aligned} E &= \{(s, u_i) \mid 1 \le i \le n\} \\ &\cup \{(v_j, t) \mid 1 \le j \le n\} \\ &\cup \{(u_i, w^\otimes_{k,l}) \mid a_i \in A^\circ_k \text{ and } R^\varphi(a^\circ_k) \in b^\circ_l(P, \mu)\}, \\ &\cup \{(w^\otimes_{k,l}, v_j) \mid b_j \in B^\circ_l \text{ and } R^\varphi(a^\circ_k) \in b^\circ_l(P, \mu)\}, \\ \text{and } c(e) &= 1 \text{ for all } e \in E \end{aligned}$$

The sets $A^\#$ and $B^\#$ are P-congruent with tolerance μ if and only if for some direction φ corresponding to a ray, the graph $G^\#_{\mathrm{comp}}(R^\varphi, P, \mu, A^\#, B^\#)$ has a max-flow of size n. We present the algorithm as procedure TPc in Table 6.

We have the following:

Theorem 4 *The (γ, γ)-approximate algorithm for approximate congruence enabled by rotation about a fixed center d given in Table 5 runs in time $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^2)$.*

[1] **procedure** TPc($A^{\#}, B^{\#}, P, \mu$)
[2] Let L be the set of directions represented by the rays of P;
[3] **repeat**
[4] Let φ be an arbitrary element of L;
[5] Compute a max-flow for $G^{\#}_{\text{comp}}(R^{\varphi}, P, \mu, A^{\#}, B^{\#})$;
[6] **if** $G^{\#}_{\text{comp}}(R^{\varphi}, P, \mu, A^{\#}, B^{\#})$ has a max-flow of n
engendered by R^{φ} and a bijection ℓ,
then return YES, with (ℓ, R^{φ});
[7] $L \leftarrow L \setminus \{\varphi\}$;
[8] **until** $L = \emptyset$;
[9] **return** NO

Table 6: Procedure for testing P-congruence

Proof. Since a region $b^{\circ}(P, \varepsilon)$ can contain $O((\varepsilon/\gamma)^2)$ points of A° and each $a^{\circ} \in A^{\circ}$ is contained in $O((\varepsilon/\gamma)^2)$ regions, we can bound the number of edges M of graph $G^{\#}_{\text{comp}}$ by $M = O(n(\varepsilon/\gamma)^2)$. □

Alternatively, in the same style as in Section 2, we can define graphs $G^{\circ}(R^{\varphi}, P, \mu, A^{\circ}, B^{\circ})$ and use the Sleator-Tarjan method. For the number of nodes N and the number of edges M of such a graph we have $N = O((\delta/\gamma)^2)$ and $M = O((\delta/\gamma)^2(\varepsilon/\gamma)^2)$. From this point onwards, our arguments are identical to those given in the translation section. The results are summarized in Table 7.

graph	total time	range
$G^{\#}_{\text{comp}}$	$O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^2)$	$n < (\delta/\gamma)^{8/3}\log^{2/3}(\delta/\gamma)$
G°	$O(n + (\delta/\gamma)^5(\varepsilon/\gamma)^2\log(\delta/\gamma))$	$n > (\delta/\gamma)^{8/3}\log^{2/3}(\delta/\gamma)$

Table 7: Time-bounds for the (γ, γ)-approximate algorithm $\mathcal{R}_{polar}$

The previous discussion assumes that d is within distance δ of every point of $A \cup B$, where $\delta = \max\{\text{diam}(A), \text{diam}(B)\}$. Clearly, if $\delta_d = \text{diam}(\{d\} \cup A \cup B) = O(\delta)$, the above time-bounds hold, since there still will be only $O(\delta/\gamma)$ rays of P. We describe now an improved method for cases where d is not close to $A \cup B$.

We construct $A^{\#}$ and $B^{\#}$ through the use of a polar and an orthogonal grid, just as for the regular polar algorithm. As before, we space the rays of the polar grid so that the largest circle is cut into arcs of length $\lambda = \gamma/(5\sqrt{2})$. We have balls $\mathcal{U}(A^{\#})$ and $\mathcal{U}(B^{\#})$ of diameter $O(\delta)$ that contain the sets $A^{\#}$ and $B^{\#}$, respectively. In testing for ε-congruence, we can reasonably assume that $\varepsilon < \delta$. Suppose d is in neither $\mathcal{U}(A^{\#})$ nor $\mathcal{U}(B^{\#})$, and let α and δ'' represent the distances from d to the closest and furthest points of $\mathcal{U}(A^{\#}) \cup \mathcal{U}(B^{\#})$. If we set $\delta' = \delta'' - \alpha$, and let $\mathcal{C}(d, \alpha)$ and $\mathcal{C}(d, \alpha + \delta')$ be the circles centered at d with radii

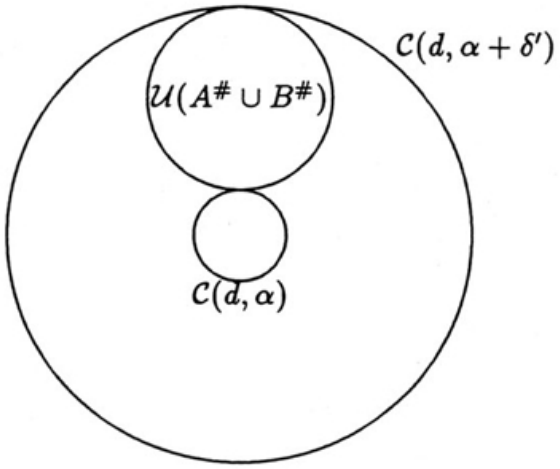

Figure 3: the ring between $C(d, \alpha)$ and $C(d, \alpha + \delta')$

α and $\alpha + \delta'$, respectively, then $\mathcal{U}(A^{\#}) \cup \mathcal{U}(B^{\#})$ lies entirely in the ring between $C(d, \alpha)$ and $C(d, \alpha + \delta')$ (see Figure 3). We assume that $\delta' = O(\delta)$, since otherwise A and B cannot be congruent under rotation around d for any $\varepsilon < \delta$.

Assume in the following discussion that $\varepsilon < \alpha$. We wish to show that, when testing $A^{\#}$ and $B^{\#}$ for P-congruence, it suffices to consider candidate rotation directions over only $O(\varepsilon/\alpha)$ radians, rather than the entire range of directions. Rotate A and construct a polar coordinate system centered at d so that the following is true: the ray $\theta = 0$ does not intersect $\mathcal{U}(A^{\#})$ or $\mathcal{U}(B^{\#})$; and the points a' and b' with maximum θ-coordinate in their respective sets $A^{\#}$ and $B^{\#}$ have equal θ-coordinate. In an intuitive sense, we realize that since both a' and b' are more than distance α from d, a rotation of $A^{\#}$ ($B^{\#}$) by $c\varepsilon/\alpha$ radians or more (for some constant c) will move a' (b') more than ε units away from any member of $B^{\#}$ ($A^{\#}$). We make this thought formal with the following argument. If τ is the line containing d and a', then τ is a supporting line to the convex hull of $A^{\#} \cup B^{\#}$. If we rotate $A^{\#}$ by a small positive rotation R^{φ}, then a' will move to the side of τ that does not contain $B^{\#}$, so the distance of $R^{\varphi}(a')$ to τ serves as a lower bound on the distance to the nearest point of $B^{\#}$. We make two observations. Firstly, a rotation $R^{\pi/2}$ of $\pi/2$ radians moves a' away from τ by α' units, where $\alpha' \geq \alpha$ is the distance of a' from d. Secondly, if $dist(R^{\varphi}(a'), \tau)$ is the distance between the point $R^{\varphi}(a')$ and the line τ, then the ratio $dist(R^{\varphi}(a'), \tau)/\varphi$ decreases monotonically with φ, for $\varphi \leq \pi/2$. These observations show that for any $\varphi \leq \pi/2$, we have $dist(R^{\varphi}(a'), \tau) > 2\alpha'\varphi/\pi \geq 2\alpha\varphi/\pi$. Therefore, if $\varphi \geq \varepsilon\pi/(2\alpha)$, then $dist(R^{\varphi}(a'), \tau) \geq \varepsilon$, and $R^{\varphi}(a')$ is not within distance ε of any point of $B^{\#}$. This shows that we need to consider only $O(\varepsilon/\alpha)$ radians of rotations in the positive direction. A similar argument gives the same bound for negative rotations.

The arcs of $C(d, \alpha + \delta')$ cut by the rays of the polar grid are of length $O(\gamma)$. Thus, sweeping $O(\varepsilon/\alpha)$ radians will cover an arc of length $O(\varepsilon(\delta' + \alpha)/\alpha) = O(\varepsilon(\delta + \alpha)/\alpha)$ on $C(d, \alpha + \delta')$, which contains $O(\varepsilon(\delta + \alpha)/(\alpha\gamma))$ rays. This, then, is the number of intervals into which we discretize the candidate rotations, and is therefore the number of max-flow

problems which we must solve.

We consider how this modified algorithm performs for various values of ε, δ, and α. If $\delta < \alpha$, then there are $O(\varepsilon/\gamma)$ max-flow intervals, which is less than the $O(\delta/\gamma)$ intervals for the algorithm that assumes $\delta_d = O(\delta)$. In other words, having the center of rotation d placed a reasonable distance away from A and B speeds up the algorithm, since the run-time is not dependent on the diameter of $A \cup B$. Rotating around a distant center d can be thought of as 1-dimensional translation; this is seen when one compares the $O(\varepsilon/\gamma)$ max-flow computations required for distant rotation to the $O((\varepsilon/\gamma)^2)$ computations required for general translation.

If $\alpha < \delta$, but we have a small value of ε such that $\varepsilon < \alpha$, then there are $O(\varepsilon\delta/(\alpha\gamma))$ intervals, which is inferior to the case where $\delta < \alpha$, but superior to the general $\delta_d = O(\delta)$ algorithm. Note that $\alpha < \delta$ implies $\delta_d = O(\delta)$, but being able to put balls around $A^{\#}$ and $B^{\#}$ that are at least ε away from d gains a slight improvement. For each of these three cases, we obtain the total time complexity by multiplying the given number of max-flow intervals by the time to compute a single max-flow.

4 General case

As is noted in [AMWW], a decision algorithm for approximate congruence enabled by rigid motions, i.e., isometries without reflection, is sufficient to decide approximate congruence enabled by an arbitrary isometry: one merely tests A and B as well as A and $-B$ for approximate congruence under a rigid motion. For approximate congruence enabled by rigid motions, the best known complete decision algorithm [AMWW] has running time $O(n^8)$. The time bound is asymptotically tight, i.e., the running time is $\Omega(n^8)$ in the worst case, as we shall see in the next section. So approximate decision algorithms are particular interesting in this case. In [Sc2] a (γ, γ)-approximate algorithm with running time $O(n^4(\varepsilon/\gamma)^2)$ has been presented. We improve this algorithm by the methods described in the last sections and give two (γ, γ)-approximate algorithms with run-time $O(n^3(\varepsilon/\gamma)^4)$ and $O(n^{1.5}(\delta/\gamma)(\varepsilon/\gamma)^4)$ where δ is the maximum of the diameter of A and the diameter of B. The algorithms are based on

Lemma 13 *Let I be an isometry and ℓ a labeling that enable approximate congruence with tolerance μ for A and B. Let $T_{c_A I(c_A)}$ be the translation that maps c_A onto $I(c_A)$. Then there is a rotation R with center $I(c_A)$ such that R and ℓ enable approximate congruence with tolerance μ for $T_{c_A I(c_A)}(A)$ and B.*

Proof. Every rigid motion can be composed of a rotation around an arbitrary center and a suitable translation [Ma]. □

The previous lemma immediately gives

Lemma 14 *Let I be an isometry and ℓ a labeling that enable approximate congruence with tolerance μ for A and B. Let $T_{c_A d}$ be the translation that maps c_A onto point d. There is a rotation centered at d such that R and ℓ enable approximate congruence with tolerance $\mu + dist(d, I(c_A))$ for $T_{c_A d}(A)$ and B.*

Let I_{opt} be a rigid motion and ℓ a labelling which enable ε-congruence for A and B. By Lemma 1, I_{opt} maps c_A into the ε-neighborhood of c_B. The algorithm in Table 8 inspects rotation centers in the ε-neighborhood of c_B and uses an (α,β)-approximate algorithm for testing approximate congruence under fixed center rotations. The set of rotation centers is chosen such that for each point in the ε-neighborhood of c_B there is rotation center within distance ν.

```
[1] Let R(A, B, c, ε) be an (α, β)-approximate algorithm
    for approximate congruence with tolerance ε
    under rotations with fixed center c (e.g. R_ortho or R_polar)
[2] possible ← false;
[3] Choose a set L of points such that each point in the ε neighborhood
    of c_B has distance ν at most to its nearest point in L;
[4] for all g ∈ L do
[5] if R(T_{c_A g}(A), B, g, ε) returns YES
      then return YES fi;
[6] if (R(T_{c_A g}(A), B, g, ε + ν) returns YES or DON'T KNOW
      then possible ← true fi;
    od;
[7] if possible
      then return DON'T KNOW
      else return NO
    fi;
```

Table 8: $(\alpha+\nu,\beta+\nu)$-approximate algorithm for ε-congruence by rigid motions

Lemma 15 *The algorithm in Table 8 is an $(\alpha+\nu,\beta+\nu)$-approximate algorithm for approximate congruence under rigid motions.*

Proof. First we prove the correctness of the algorithm. If $\mathcal{R}(T_{c_Ag}(A),B,g,\varepsilon)$ returns YES for some g then A and B are ε-congruent, too. If $\mathcal{R}(T_{c_Ag}(A),B,g,\varepsilon+\nu)$ returns NO for all $g\in L$ then A and B cannot be ε-congruent by Lemma 14.
Next we show that the algorithm is $(\alpha+\nu,\beta+\nu)$-approximate. Let $\varepsilon<\varepsilon^{\mathrm{I}}_{opt}(A,B)-(\alpha+\nu)$. For all $g\in L$ we have $\varepsilon^{\mathrm{Rg}}_{opt}(T_{c_Ag}(A),B)\geq\varepsilon^{\mathrm{I}}_{opt}(A,B)$. Hence $\varepsilon+\nu<\varepsilon^{\mathrm{Rg}}_{opt}(T_{c_Ag}(A),B)-\alpha$ for all g. So $\mathcal{R}(T_{c_Ag}(A),B,g,\varepsilon+\nu)$ returns NO for all g because it is $(\alpha,\cdot)$-approximate. Now let $\varepsilon\geq\varepsilon^{\mathrm{I}}_{opt}(A,B)+\beta+\nu$. There is a $g\in L$ such that $\varepsilon^{\mathrm{Rg}}_{opt}(T_{c_Ag}(A),B)\leq\varepsilon^{\mathrm{I}}_{opt}(A,B)+\nu$ by Lemma 1 and Lemma 14. Therefore $\varepsilon\geq\varepsilon^{\mathrm{Rg}}_{opt}(T_{c_Ag}(A),B)+\beta$ and $\mathcal{R}(T_{c_Ag}(A),B,g,\varepsilon)$ returns YES for this g because it is $(\cdot,\beta)$-approximate. Hence the algorithm above is $(\alpha+\nu,\beta+\nu)$-approximate if $\mathcal{R}$ is (α,β)-approximate. □

The run-time of the algorithm is $O(|L|\cdot$run-time of algorithm $\mathcal{R})$. L can be chosen such that $|L|=O((\varepsilon/\nu)^2)$. With $\mathcal{R}=\mathcal{R}_{ortho}$ or $\mathcal{R}=\mathcal{R}_{polar}$ and $\alpha=\beta=\nu=\gamma/2$ we get (γ,γ)-approximate algorithms for approximate congruence under rigid motions.

Theorem 5 *There is a (γ,γ)-approximate algorithm for approximate congruence enabled by rigid motions with running time $O(n^3(\varepsilon/\gamma)^4)$. Let $\delta = \max(\mathrm{diam}(A), \mathrm{diam}(B))$. There are (γ,γ)-approximate algorithm for approximate congruence enabled by rigid motions with running time $O(n^{1.5}(\varepsilon/\gamma)^4(\delta/\gamma))$ and $O((\varepsilon/\gamma)^2(n + (\delta/\gamma)^5(\varepsilon/\gamma)^2\log(\delta/\gamma)))$.*

Proof. By the discussion above and Theorems 3 and 4 of Section 3. □

5 Lower bound on the worst case running time of the decision algorithm of Alt et al.

We want to show that the algorithm of Alt et al. [AMWW] for deciding ε-congruence of two sets $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ of n points in the plane enabled by a rigid motion has worst case running time $\Theta(n^8)$. We start with a brief description of the algorithm.

If there is rigid motion that enables ε-congruence, then there is a rigid motion that enables ε-congruence and maps a_i and a_j onto circles C_k and C_l with radius ε and center b_k and b_l resp. for some $i,j,k,l \in [1..n]$. In the algorithm of [AMWW] such an isometry is searched for all combinations of i,j,k, and l. Let i,j,k,l be fixed. If a_i and a_j are simultanously moved on C_k and C_l resp., the other points of A move on algebraic curves [AMWW, Sc1] which possibly intersect the circles with radius ε centered at the points in B. If the motion of a_i and a_j on C_k and C_l is parametrized we get a set $I_{m,p}$ of parameter values for each pair $(m,p) \in [1..n]^2$ such that a_m has distance at most ε to b_p for the parameter values in $I_{m,p}$. Each $I_{m,p}$ consists of $O(1)$ intervals. The endpoints of these intervals are sorted and the isometric mapping corresponding to these paramater values are tested for enabling ε-congruence. For the test whether isometry I enables approximate congruence with tolerance ε, a maximum matching in $G_I = (U \cup V, E_I)$ is computed, where $U = \{u_1, \ldots, u_n\}$ represents the points in A, $V = \{v_1, \ldots, v_n\}$ represents the points in B, and $E_I = \{\{u_s, v_t\} \mid dist(I(a_s), b_t) \leq \varepsilon\}$. At first the graph and a maximum matching are computed for the isometry corresponding to the smallest parameter value. Graph and maximum matching for the other isometries are computed one by one, according to increasing parameter values, using graph and maximum matching of the preceeding isometry. The maximum matching is updated by a depth first search for an augmenting path in the residual graph started in the unmatched nodes. The algorithm stops if a perfect matching has been found.

For a lower bound on the worst case running time of this algorithm consider Fig. 4. Let $n/4$ of the points of B be in the circles $D_1, \ldots, D_4$ each. $n/4$ of the points of A are in E_1 and E_2 each. Furthermore $n/8$ points are in E_3 and $3n/8$ points are in E_4. We assume that the points are in general position, i.e. all distances between different point pairs are different. Since A and B are not ε-congruent, all combinations of i,j,k,l are taken into consideration. We concentrate on those $\Omega(n^4)$ combinations where a_i is in D_1, a_j in D_2, b_k in E_1, and b_l in E_2. When a_i and a_j are moved on C_k and C_l resp., each point of A which is in D_3 moves into the ε-neighborhood of each point of B which is in E_3. Consider the $\Omega(n^2)$ interval endpoints corresponding to these events. At each such event a new edge, say $\{u_x, v_y\}$, is added to the

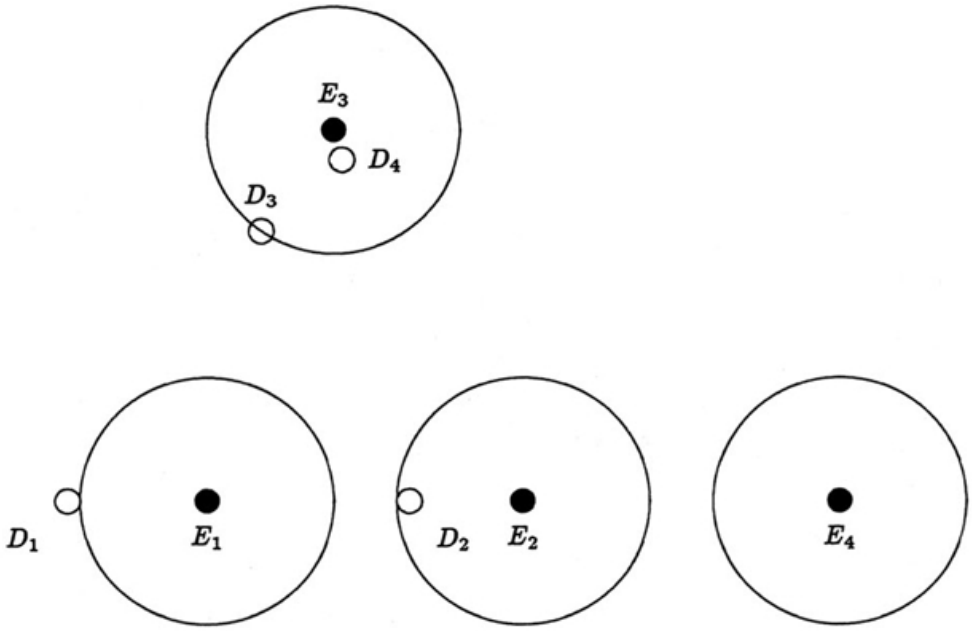

Figure 4: $n/4$ points are in $D_1, D_2, D_3, D_4, E_1, E_2$, $n/8$ points are in E_3 and $3n/8$ points are in E_4. The large circles are ε-balls centered at the midpoints of E_i.

present graph, where u_x corresponds to a point in D_3 and v_y corresponds to a point in E_3. The set of nodes reachable from u_x and v_y is contained in the set of nodes corresponding to the points in $D_3 \cup D_4 \cup E_3$. Furthermore, the $n/8$ nodes corresponding to the points in E_3 and the $n/4$ nodes corresponding to the points in D_4 form a complete bipartite subgraph. Since we maintain a maximum matching, all nodes corresponding to the points in E_3 are already matched, when the new edge is added. Hence the new edge cannot give rise to an augmenting path. Since $\Omega(n)$ of the nodes corresponding to the points in D_4 are unmatched, and each of these nodes has $\Omega(n)$ incident edges, DFS takes time $\Omega(n^2)$. Since there are $\Omega(n^2)$ such events for each of the $\Omega(n^4)$ combinations, the overall running time is $\Omega(n^8)$. Alt et al. have shown that the running time is $O(n^8)$, so the worst case running time is $\Theta(n^8)$.

The example above is a good example for the utility of the approximate decision algorithm in Table 8 using $\mathcal{R}_{ortho}$. Started with $\gamma = \varepsilon/4$ in $\mathcal{R}_{ortho}$ and $\nu = \varepsilon/4$, the algorithm has running time $O(n)$ for the example above: The points in A and B are moved to at most 32 different points. The computation of these points has time complexity $O(n)$. There are $O(1)$ grid points in the neighborhood of the centroid of B. For each such grid point, the number of interval endpoints and hence the number of graphs, which are considered, is $O(1)$. Each of these graphs is of size $O(1)$. Since edge capacities are bounded by n, there are $O(n)$ updates of edge capacities in unit steps by blocking-flow computations. Since a blocking-flow

computation takes constant time, the running time of all update steps together is $O(n)$. The initial max-flow computation has time complexity $O(1)$. So in this extreme example we have a speedup of n^7 compared to the complete decision algorithm of Alt et al.

6 Conclusion

In this paper we have addressed the question of testing whether two equal-cardinality point sets are ε-congruent, under the general L_p metric and a specified class of isometries. Our algorithms are (γ, γ)-approximate, which means that, for any $\varepsilon \notin [\varepsilon_{opt}(A,B)-\gamma, \varepsilon_{opt}(A,B)+\gamma]$, they correctly answer a query, and for $\varepsilon \in [\varepsilon_{opt}(A,B)-\gamma, \varepsilon_{opt}(A,B)+\gamma]$, they either answer correctly or choose not to answer. Our presentation culminated with an algorithm for the case where an arbitrary isometry may be performed on one of the sets, but we also obtained more efficient algorithms for the special cases of the isometry restricted to a translation only or a rotation only. In each case we gave an algorithm with running time dependent on δ/γ and linear in n in the worst-case, implying that these (γ, γ)-approximate algorithms are especially efficient on dense data.

A primary strength of a (γ, γ)-approximate algorithm is that an incorrect answer is never returned. Algorithms that work with perturbed data must have some imprecision, but our algorithms have the attractive feature of being prudent enough not to answer when the query is too difficult. The imprecision inherent in a (γ, γ)-approximate algorithm is exhibited in the indecision interval, $[\varepsilon_{opt}(A,B)-\gamma, \varepsilon_{opt}(A,B)+\gamma]$.

Another attraction of the (γ, γ)-approximate algorithms presented in this paper is that the user is offered a trade-off between precision and run-time. We mention a manner whereby this trade-off can be used to remove the indecision of an approximate decision algorithm. If we wish to test for ε-congruence (for any isometry class) for a given value ε, we likely will first choose to execute our algorithm with a relatively large value of γ, since for each of our algorithms, the run-time is inversely proportional to an exponent of γ. If ε lies in the indecision interval $[\varepsilon_{opt}(A,B)-\gamma, \varepsilon_{opt}(A,B)+\gamma]$, then we may discover that our choice of γ was too large, for we may not be given an answer. If we alternate setting $\gamma \leftarrow \gamma/2$ and repeating the test until receiving an answer, the cost of our procedure will be dominated by the final test. The effect of our approach is that the total run-time is within a constant factor of a single test which chooses for γ the "optimal" choice of $|\varepsilon_{opt}(A,B)-\varepsilon|$. This means that we can replace γ by $K_\varepsilon = |\varepsilon_{opt}(A,B)-\varepsilon|$ in the time bounds of our algorithms, and in the process obtain complete, not approximate, algorithms:

Theorem 6 *There exist decision algorithms for testing ε-congruence under translation with time-complexity $O(n^{1.5}(\varepsilon/K_\varepsilon)^4)$, under rotation with time-complexity $O(n^3(\varepsilon/K_\varepsilon)^2)$ and $O(n^{1.5}(\delta/K_\varepsilon)(\varepsilon/K_\varepsilon)^2)$, and under general isometry with time-complexity $O(n^3(\varepsilon/K_\varepsilon)^4)$ and $O(n^{1.5}(\delta/K_\varepsilon)(\varepsilon/K_\varepsilon)^4)$. Here, $K_\varepsilon = |\varepsilon_{opt}(A,B)-\varepsilon|$.*

These new expressions agree with our intuition, since we believe that testing for ε-congruence should be harder when ε is close to $\varepsilon_{opt}(A,B)$. We can also adapt our methods to estimate

$\varepsilon_{opt}(A, B)$ through a search procedure, where the tolerance of the estimate replaces γ in the time bound of the algorithm.

This means that we can replace γ by $|\varepsilon_{opt}(A, B) - \varepsilon|$ in the time bounds of our algorithms, and in the process obtain complete, not approximate, algorithms. These new expressions agree with our intuition, since we believe that testing for ε-congruence should be harder when ε is close to $\varepsilon_{opt}(A, B)$. We can also adapt our methods to estimate $\varepsilon_{opt}(A, B)$ through a search procedure, where the tolerance of the estimate replaces γ in the time bound of the algorithm.

Known decision algorithms for ε-congruence are expensive, and therefore unsuitable for many applications. Our response to this situation has been to develop approximate decision algorithms, which enjoy substantially improved time bounds by perturbing the point sets in order to impose structure. Our algorithms never return an incorrect answer, and have bounds on the query values for which they can choose to give no answer. We feel that this combination of speed, correctness, and bounded imprecision characterize our methods as practical algorithms for point matching.

References

[AOT] R.K. Ahuja, J.B. Orlin, R.E. Tarjan, "Improved Time Bounds for the Maximum Flow Problem", *SIAM Journal on Computing*, **18** (1989), pp. 939-954.

[AMWW] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl, "Congruence, Similarity, and Symmetries of Geometric Objects", *Discrete and Computational Geometry*, **3** (1988), pp. 237-256.

[AKMSW] E.M. Arkin, K. Kedem, J.S.B. Mitchell, J. Sprinzak, M. Werman, "Matching Points into Noise Regions: Combinatorial Bounds and Algorithms", in *2nd Symposium on Discrete Algorithms*, 1991.

[AMZ] E.M. Arkin, J.S.B. Mitchell, and K. Zikan, "Algorithms for Generalized Matching of Point Sets", Manuscript, 1988. Presented at the CORS/ORSA/TIMS National Meeting, Vancouver, B.C., May 1989.

[Ba] H.S. Baird, *Model-Based Image Matching Using Location*, MIT Press, 1984.

[CHM] J. Cheriyan, T. Hagerup, and K. Mehlhorn, "Can a maximum flow be computed in $o(nm)$ time ", *17th International Colloquium on Automata, Languages, and Programing* (1990), pp. 235-248.

[Be] B. Behrends, *Algorithmen zur Erkennung der ε-Kongruenz von Punktmengen und Polygonen*, Diplomarbeit, Freie Universität Berlin, Germany, 1990.

[FM] T. Feder and R. Motwani, "Clique Partitions, Graph Compression and Speeding-up Algorithms", in *Proc. of the 23rd ACM Symp. on Theory of Computing*, 1991.

[FF] L.R. Ford, Jr., and D.R. Fulkerson, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem", *Canadian Journal of Mathematics*, **9** (1957), pp. 210-218.

[He] P.J. Heffernan, "The Translation Square Map and Approximate Congruence", Tech. Report 940, School of ORIE, Cornell University, Ithaca, New York, 1990; to appear, *Information Processing Letters.*

[HoKa] J.E. Hopcroft and R.M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", *SIAM Journal of Computing*, **2** (1973), pp. 225-231.

[HuKe] D.P. Huttenlocher and K. Kedem, "Computing the Minimum Hausdorff Distance for Point Sets Under Translation", in *Proc. of 6th ACM Symp. on Computational Geometry* (1990), pp. 340-349.

[HKS] D.P. Huttenlocher, K. Kedem, and M. Sharir, "The Upper Envelope of Voronoi Surfaces: Theory and Applications", Tech. Report 200/91, Tel Aviv University, 1991.

[ISI] K. Imai, S. Sumino, and H. Imai, "Minimax geometric fitting of two corresponding sets of points", in *Proc. of 5th ACM Symp. on Computational Geometry* (1989), pp. 276-282.

[Iw] S. Iwanowski, *Approximate Congruence and Symmetry Detection in the Plane*, Ph.D. Thesis, Fachbereich Mathematik, Freie Universitat Berlin, 1990.

[LS] H.P. Lenhof and M. Smid, "An Optimal Construction Method for Generalized Convex Layers", Max-Planck-Institut für Informatik, Saarbrücken, Germany, in preparation.

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

[Me] K. Mehlhorn, *Data Structures and Algorithms 2: Graph Algorithms and NP-completeness*, Springer-Verlag, 1984.

[PS] F.P. Preparata and M.I. Shamos, *Computational Geometry — An Introduction*, Springer-Verlag, 1985.

[Sc1] S. Schirra, *Über die Bitkomplexität der ε-Kongruenz*, Diplomarbeit, Universität des Saarlandes, Saarbrücken, Germany, 1988.

[Sc2] S. Schirra, "Approximate Algorithms for Approximate Congruence", Tech. Report A21/90, Universität des Saarlandes, Saarbrücken, Germany, 1990.

[Sl] D.D. Sleator, "An $O(nm \log n)$ algorithm for maximum network flow", Tech. Report STAN-CS-80-831, Computer Science Dept., Stanford Univ., 1980.

[ST] D.D. Sleator and R.E. Tarjan, "A Data Structure for Dynamic Trees", Proc. 13th ACM Symp. on Theory of Computing, 1981, pp. 114-122.

[Sp] J. Sprinzak, " Master's Thesis ", Hebrew University, Dept. of Computer Science, 1990.

[Ta] R.E. Tarjan, *Data Structures and Network Algorithms*, SIAM, 1983, pp. 97-112.