Approximate Algorithms for Approximate Congruence

by

Stefan Schirra

A 21/90

Fachbereich 14 – Informatik
Universität des Saarlandes
6600 Saarbrücken
Germany

This research was supported by the DFG SPP Datenstrukturen und Algorithmen
Grant Me 620/6
Approximate Algorithms for Approximate Congruence

Stefan Schirra*
Fachbereich 14 Informatik
Universität des Saarlandes
6600 Saarbrücken
Germany

Abstract

We study the decision problem whether two sets of $n$ points in the plane are approximately congruent with a given tolerance $\varepsilon$. Approximate algorithm means that the algorithm is not guaranteed to take a decision for all tolerance values. For point sets $A$ and $B$ let $\varepsilon_{opt}(A,B)$ be the smallest tolerance value permitting approximate congruence of $A$ and $B$. An algorithm for approximate congruence is said to be $(\alpha,\beta)$-approximate for $\alpha,\beta \geq 0$ if the algorithm is guaranteed to give an answer for test values outside the interval $[\varepsilon_{opt}(A,B) - \alpha, \varepsilon_{opt}(A,B) + \beta]$. For tolerance values in $[\varepsilon_{opt}(A,B) - \alpha, \varepsilon_{opt}(A,B) + \beta]$, the algorithm may give an correct answer or may report that an answer cannot be found. We give an $(\frac{1}{2}\varepsilon_{opt}(A,B), \varepsilon_{opt}(A,B))$-approximate algorithm with time complexity $O(n^4)$. We use an additional input parameter $\gamma$ for tradeoff between running time and size of the uncertainty interval and give an $(\gamma,\gamma)$-approximate algorithm with running time $O((\xi)^{2.5}n^4)$. Moreover we give an $((\frac{1}{2}\varepsilon_{opt}(A,B), \varepsilon_{opt}(A,B))$-approximate algorithm with run time $O(n^{2.5})$ for approximate congruence enabled by a translation and an $(\gamma,\gamma)$-approximate algorithm for this case with running time $O((\xi)^{2}n^{2.5})$. Here $\varepsilon_{opt}(A,B)$ is the smallest tolerance value permitting approximate congruence of $A$ and $B$ enabled by a translation.

1 Introduction

Two sets $A$ and $B$ of points in the plane are called congruent if there is an isometry which maps the points in $A$ onto the points in $B$, where a map $I: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ is called isometry if $\text{dist}_2(a, b) = \text{dist}_2(I(a), I(b))$ for all points $a$ and $b$. Here $\text{dist}_2$ denotes Euclidian metric.

*This work was supported by the DFG Schwerpunktprogramm Datenstrukturen und Algorithmen Grant Me 620/6
Although there are several optimal $O(n \log n)$ algorithms [1,2] for testing congruence, testing congruence is difficult in practice. Clearly, every algorithm for testing congruence is highly sensitive to inexact computations and often, the input data are inaccurate. Therefore Alt et al. [1] considered approximate congruence.

**Definition 1 (Alt et al.)** Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ be two sets of points in the plane and let $\varepsilon$ be a non-negative real number. The point sets $A$ and $B$ are called approximately congruent with tolerance $\varepsilon$ (or shortly $\varepsilon$-congruent) if an isometric mapping $I$ and a bijective labeling $\ell: A \rightarrow B$ exist s.t. $\text{dist}(I(a_i) - \ell(a_i)) \leq \varepsilon$ for all $i, 1 \leq i \leq n$. We say that $I$ and $\ell$ enable approximate congruence for $A$ and $B$.

Note that approximate congruence as defined above depends on the chosen metric. Alt et al. studied two kinds of problems in connection with approximate congruence:

1. **The decision problem:** For a given tolerance value $\varepsilon$ decide whether $A$ and $B$ are $\varepsilon$-congruent.

2. **The optimization problem:** Compute the smallest tolerance value $\varepsilon_{opt}(A, B)$ permitting approximate congruence for $A$ and $B$.

They gave an algorithm for the general decision problem and algorithms for some restricted decision and optimization problems where the approximate congruence must be enabled by special isometries e.g. translations, rotations around a fixed center,... Moreover they considered different metrics. The decision algorithm of Alt et al. for the general case with Euclidian metric has running time $O(n^8)$ (in contrast to several optimal $O(n \log n)$ algorithms [1,2] for testing exact congruence. In the case of exact congruence the isometry fixes the correspondence between the points. In the approximate case however, the correspondence between the points, the labeling, has to be found by the algorithm. Therefore the problem is easier if a labeling is fixed in advance: Alt et al. [1] and Iwanowski [5] gave $O(n \log n)$ time algorithms for some restricted decision problems and Imai et al. [4] presented optimization algorithms with time complexity $O(n^3 \log n)$ for Euclidian metric and $O(n)$ for maximum metric. In this paper we consider only the case that a labeling is not given and use Euclidian metric.

Since the running times of the algorithms for testing approximate congruence are far from being practical, approximation algorithms are interesting. The basic idea of our algorithm for approximate congruence is known from daily life. If one compares distances (e.g. between cities on a map) most of the time it is obvious which of two distances is smaller. A tape measure is necessary only if the distances are nearly equal. Nearly the same applies to approximate congruence. It is not necessary to have a complicated algorithm that solves every problem instance. There are simpler and faster strategies that work correctly if the testvalue $\varepsilon$ is evidently large enough for permitting approximate congruence or evidently to small.

More precisely, we call an algorithm an $(\alpha, \beta)$-approximate algorithm for approximate congruence if it solves the decision problem for all $\varepsilon$ outside the interval
\[ \varepsilon_{opt}(A, B) - \alpha, \varepsilon_{opt}(A, B) + \beta, \] where \( \varepsilon_{opt}(A, B) \) is the smallest tolerance value permitting approximate congruence for \( A \) and \( B \). Similarly let \( \varepsilon_{opt}^T(A, B) \) be the smallest tolerance value permitting approximate congruence for \( A \) and \( B \) when the approximate congruence has to be enabled by a translation. Analogously an algorithm for testing approximate congruence under translations is called \((\alpha, \beta)\)-approximate if it solves the decision problem for all \( \varepsilon \) outside the interval \([\varepsilon_{opt}(A, B) - \alpha, \varepsilon_{opt}^T(A, B) + \beta]\). Furthermore we require that an \((\alpha, \beta)\)-approximate algorithm reports "DON'T KNOW" if a decision can not be taken. Hence an \((\alpha, \beta)\)-approximate algorithm always terminates and works reliable, i.e. all "YES" and "NO" answers are correct.

In the next section we derive an \((\frac{1}{2}\varepsilon_{opt}^T(A, B), \varepsilon_{opt}^T(A, B))\)-approximate algorithm for approximate congruence under translation with running time \( O(n^{2.5}) \) and an \((\frac{1}{2}\varepsilon_{opt}(A, B), \varepsilon_{opt}(A, B))\)-approximate algorithm with running time \( O(n^4) \) for the general case. In Section 3 an additional input parameter is used as tradeoff between running time and size of the uncertainty interval. We give an \((\gamma, \gamma)\)-approximate algorithm with running time \( O((\frac{\gamma}{2})^3 n^4) \) for the general case. The comparable exact \((0,0)\)-approximate algorithm of Alt et al. [1] has running time \( O(n^8) \). Furthermore we give an \((\gamma, \gamma)\)-approximate algorithm with running time \( O((\frac{\gamma}{2})^3 n^{2.5}) \) for testing approximate congruence under translation. Here the comparable exact algorithm of Alt et al. has running time \( O(n^6) \). In Section 4 discuss briefly the machine model used in the time analysis of the previous sections, the Real-RAM [9], and its relation to practice in connection with approximate congruence. Section 5 offers some concluding remarks.

2 First Approximate Algorithms

First of all we derive an \((\frac{1}{2}\varepsilon_{opt}(A, B), \varepsilon_{opt}(A, B))\)-approximate algorithm for testing approximate congruence with tolerance \( \varepsilon \) of two point sets \( A = \{a_1, \ldots, a_n\} \) and \( B = \{b_1, \ldots, b_n\} \). Then we give an \((\frac{1}{2}\varepsilon_{opt}(A, B), \varepsilon_{opt}(A, B))\)-approximate algorithm for the general case. Both algorithms are based on the fact, that an isometry which enables approximate congruence maps the centroid \( c_A \) of \( A \) near to the centroid \( c_B \) of \( B \). Let \( T_{c_A c_B} \) be the translation that maps \( c_A \) onto point \( c_B \). In the sequel we sometimes define the labeling \( \ell \) on the point sets implicitly by a permutation \( \pi \) on the index set. In this case \( \ell(a_i) = b_{\pi(i)} \).

Lemma 1 Every isometry \( T \) that enables approximate congruence with tolerance \( \varepsilon \) maps \( c_A \) into the ball with radius \( \varepsilon \) around \( c_B \).

Proof: Let \( \pi \) be the permutation that defines the labeling. We have

\[ I(c_A) - c_B = I(\frac{1}{n} \sum a_i - \frac{1}{n} \sum b_i = \frac{1}{n} \sum (I(a_i) - b_{\pi(i)}) \leq \frac{1}{n} \sum_{i=1}^{n} \varepsilon = \varepsilon \]

Lemma 2 Let \( T \) be a translation and \( \ell \) a labeling that enable \( \varepsilon \)-congruence for \( A \) and \( B \). Let \( T_{c_A c_B} \) be the translation that maps \( c_A \) onto point \( c_B \). \( T_{c_A c_B} \) enables approximate congruence for \( A \) and \( B \) with tolerance \( \varepsilon + \| T(c_A) - c_B \| \).
Figure 1: $A$ and $B$

Figure 2: $B$ with $\varepsilon$-balls and $I(A)$, where $I$ is an isometry which enables $\varepsilon$-congruence

Figure 3: $B$ with $\varepsilon$- and $2\varepsilon$-balls and $T_{\varepsilon A}B(A)$ rotated by $90^\circ$
Proof: $T$ is the composition of $T_{\epsilon A'B'}$ and the translation which maps $c_B$ onto $T(c_A)$.

Lemma 2 gives an approximate algorithm for testing approximate congruence enabled by a translation. Consider the following algorithm.

(01) Construct $G_{\varepsilon} = (V, E)$ where
(02) $V = \{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\}$ and
(03) $E = \{\{u_i, v_k\} \mid T_{\epsilon A'B'}$ maps $a_i$ into the $\varepsilon$-ball with center $b_k\};$
(04) if $G_{\varepsilon}$ has a perfect matching then return YES; $
i$
(05) Construct $G_{2\varepsilon} = (V, E)$ where
(06) $V = \{u_1, \ldots, u_n\} \cup \{v_1, \ldots, v_n\}$ and
(07) $E = \{\{u_i, v_k\} \mid T_{\epsilon A'B'}$ maps $a_i$ into the $2\varepsilon$-ball with center $b_k\};$
(08) if $G_{2\varepsilon}$ has not a perfect matching then return NO; $
i$
(09) return DON'T KNOW;

Theorem 1 The algorithm above is an $(\frac{1}{2}\varepsilon_{opt}(A, B), \varepsilon_{opt}(A, B))$-approximate algorithm for approximate congruence by translation with time complexity $O(n^{2.5})$.

Proof: A labeling which together with $T_{\epsilon A'B'}$ enables $\varepsilon$-congruence corresponds to a matching of the bipartite graph $G_{\varepsilon}$ of maximum cardinality. Hence $T_{\epsilon A'B'}$ enables approximate congruence with tolerance $\varepsilon$ if $G_{\varepsilon}$ has a perfect matching. So all "YES" answers are reliable. If there is a translation and a labeling $\ell$ that enable $\varepsilon$-congruence, $T_{\epsilon A'B'}$ and $\ell$ enable approximate congruence with tolerance $2\varepsilon$. Hence, if $G_{2\varepsilon}$ has not a perfect matching, $A$ and $B$ are not $\varepsilon$-congruent by a translation.

For the time analysis note first that the centroids of $A$ and $B$, and $T_{\epsilon A'B'}(A)$ can be computed in time $O(n)$. $G_{\varepsilon}$ and $G_{2\varepsilon}$ can be constructed in time $O(n^2)$. Testing for a perfect matching can be done by computing a maximum matching in time $O(n^{2.5})$ [8]. Figure 4 shows the dependence of the algorithm on $\varepsilon$ and $\varepsilon_{opt}(A, B)$. For the general case the following basic observation [1] is helpful: It is sufficient to have an algorithm that searches for a composition of a translation and a rotation that enables approximate congruence since every other isometry is a composition of a reflection at a line that can be chosen arbitrarily, a translation and a rotation. Such an algorithm can then be applied first to $A$ and $B$ and then to $A$ and the image of $B$ obtained by a reflection of $B$ at an arbitrary line e.g. a coordinate axis. Therefore we restrict our attention to compositions of translations and rotations, which are called even isometries or rigid motions [7].

Lemma 3 Let $I$ be an even isometry and $\ell$ a labeling that enable $\varepsilon$-congruence for $A$ and $B$. Let $T_{de}$ be the translation that maps point $d$ onto point $e$. There is a rotation $R$ with center $e$, s.t. $R$ and $\ell$ enable approximate congruence with tolerance $\varepsilon + \|I(d) - e\|$ for $T_{de}(A)$ and $B$.

Proof: Every even isometry is a composition of a translation and a rotation whose center can be chosen arbitarily [7]. So we have $I = \hat{T} \circ \hat{R}$, where $\hat{T}$ is the translation that maps $d$ onto $I(d)$ and $\hat{R}$ is a rotation with center $d$. We show that the rotation
Figure 4: Output in relation to $\epsilon_{opt}^T(A, B)$

\[ R = T_{de} \circ \hat{R} \circ T_{de}^{-1} \] with center $e$ enables approximate congruence for $T_{de}(A)$ and $B$ with tolerance $\epsilon + \| I(d) - e \|$. Let $\pi$ be the permutation that defines the labeling $\ell$ and let $\hat{T}$ be the translation that maps $I(d)$ onto $e$. We have

\[
\| R(T_{de}(a_i)) - b_{\pi(i)} \| = \| T_{de} \circ \hat{R} \circ T_{de}^{-1} \circ T_{de}(a_i) - b_{\pi(i)} \|
= \| \hat{T} \circ \hat{T} \circ \hat{R}(a_i) - b_{\pi(i)} \|
= \| \hat{T}(I(a_i)) - b_{\pi(i)} \|
\leq \| I(a_i) - b_{\pi(i)} \| + \| I(d) - e \|
\leq \epsilon + \| I(d) - e \|.
\]

Combining Lemma 1 and Lemma 3 we have

**Lemma 4** Let $I$ be an even isometry and $\ell$ a labeling that enable $\epsilon$-congruence for $A$ and $B$. There is a rotation $R$ with center $c_B$ s.t. $R$ and $\ell$ enable approximate congruence with tolerance $\epsilon + \| I(c_A) - c_B \| \leq 2\epsilon$ for $T_{c_Ac_B}(A)$ and $B$.

Now we need an algorithm for testing approximate congruence of two point sets $D = \{d_1, \ldots, d_n\}$ and $E = \{e_1, \ldots, e_n\}$ enabled by a rotation around a given center $c$ when no labeling is given. Alt et al. [1] studied this restricted problem, but only the case that a labeling is given. However, it is straightforward to combine their algorithm with the methods used in other algorithms of [1] to design an algorithm for the case that no labeling is known.
We identify the rotations with center \( c \) with the points on the unit sphere \( S^1 \). The set of rotations around the fixed center \( c \) that map \( d_i \) into the \( \epsilon \)-ball around \( c_k \) form a circular interval \( I_{i,k} \) on \( S^1 \). We assign a bipartite graph \( G_\alpha = (V, E_\alpha) \) to each point \( \alpha \in S^1 \), where \( V = \{ u_1, \ldots, u_n \} \cup \{ v_1, \ldots, v_n \} \) and \( E_\alpha = \{ \{ u_1, v_1 \}, \ldots, \{ u_n, v_n \} \} ; \alpha \in I_{i,k} \). The rotation corresponding to \( \alpha \) enables approximate congruence iff \( G_\alpha \) has a perfect matching. Since for every \( \alpha \) there is an endpoint \( \beta \) of an interval s.t. \( E_\alpha \subset E_\beta \), it suffices to compute maximum matchings for the graphs assigned to the endpoints of the intervals and to test if one of them is perfect. A maximum matching can be computed in time \( O(n^{2.5}) \) \([8]\). Since there are \( O(n^2) \) interval endpoints that can be computed in constant time each, we have an algorithm that solves our problem in time \( O(n^{4.5}) \).

This can be done better. We sort the endpoints in time \( O(n^2 \log n) \) and test the graphs for a perfect matching in the computed order. In general (if all interval endpoints are different) the graphs associated to two contiguous endpoints differ only by one edge that is deleted or added. We start with an arbitrary endpoint \( \beta_1 \). We build the graph \( G_{\beta_1} \) and compute a maximum matching. Each time we proceed from interval endpoint \( \alpha_1 \) to the successor \( \alpha_2 \), one edge is added or deleted. If an edge is added, we try to compute an augmenting path. The computation of one augmenting path is sufficient because the cardinality of the matching can increase only by one. If an edge is deleted which is not contained in the maximum matching of \( G_{\alpha_1} \), the maximum matching of \( G_{\alpha_2} \) is a maximum matching for \( G_{\alpha_1} \), too. If the deleted edge was in the matching, it suffices again to compute an augmenting path. The procedure can easily be modified for degenerate cases. Since an augmenting path can be computed in time \( O(n^2) \), we have an algorithm for testing approximate congruence of two point sets of cardinality \( n \) enabled by a rotation around a given center with running time \( O(n^4) \).

Consider the following outline of an algorithm.

\[
\begin{align*}
(01) & \text{ if } T_{c_{AB}}(A) \text{ and } B \text{ are } \epsilon\text{-congruent by a rotation with center } c_B \\
& \quad \text{ then return YES fi;}
(02) & \text{ if } T_{c_{AB}}(A) \text{ and } B \text{ are not } 2\epsilon\text{-congruent by a rotation with center } c_B \\
& \quad \text{ then return NO fi;}
(03) & \text{ return DON'T KNOW}
\end{align*}
\]

**Theorem 2** The algorithm above is an \( (\epsilon_{opt}(A,B), 2\epsilon_{opt}(A,B)) \)-approximate algorithm for approximate congruence with time complexity \( O(n^4) \).

**Proof:** By Lemma 4, \( T_{c_{AB}}(A) \) and \( B \) have to be approximately congruent with tolerance \( 2\epsilon \) by a rotation with center \( c_B \) if \( A \) and \( B \) are approximately congruent with tolerance \( \epsilon \). Thus if the test in (03) is positive, \( A \) and \( B \) cannot be approximately congruent with tolerance \( \epsilon \). Further, \( A \) and \( B \) are \( \epsilon\text{-congruent if } T_{c_{AB}}(A) \) and \( B \) are \( \epsilon\text{-congruent, because } T_{c_{AB}} \) is an isometry.

By Lemma 1 there is an isometry \( I_a \) s.t. \( \| I_a(c_B) - c_B \| \leq \epsilon_{opt}(A,B) \). Hence a rotation around \( c_B \) exists s.t. \( T_{c_{AB}}(A) \) and \( B \) are approximately congruent with tolerance
Clearly \( T_{c \epsilon c_B}(A) \) and \( B \) cannot be \( 2\epsilon \)-congruent if \( \epsilon < \frac{1}{2} \epsilon_{\text{opt}}(A, B) \). Hence the algorithm may be unable to give an answer only if the tolerance value lies in the interval \([\frac{1}{2} \epsilon_{\text{opt}}(A, B), 2\epsilon_{\text{opt}}(A, B)]\).

The computation of \( T_{c \epsilon c_B}(A) \) takes time \( O(n) \). Lines (02) and (03) use the algorithm described above for testing approximate congruence by rotation around a given center. They have time complexity \( O(n^4) \) each.

Two remarks:

1. If the algorithm reports "DON'T KNOW", we may conclude that the test value is near \( \epsilon_{\text{opt}}(A, B) \) without any knowledge of the exact value of \( \epsilon_{\text{opt}}(A, B) \).

2. The algorithm may give a correct answer even if the test value lies in the interval \([\frac{1}{2} \epsilon_{\text{opt}}(A, B), 2\epsilon_{\text{opt}}(A, B)]\).

3 Time versus Uncertainty

The uncertainty interval can be made arbitrarily small at the expense of running time. Let \( \gamma \) be a positive real number smaller than the test value \( \epsilon \). Cover the ball with center \( c_B \) and radius \( \epsilon \) with balls of radius \( \gamma \). Let \( U_i \) be the \( i \)-th ball of the covering, \( c_i \) be it's center, and \( T_{c \epsilon c_i} \) the translation that maps \( c_A \) onto \( c_i \). Consider the following outline of an algorithm.

\begin{verbatim}
(01) possible ← false;
(02) for each ball of the covering do
(03)  Compute \( T_{c \epsilon c_i}(A) \);
(04)  if \( T_{c \epsilon c_i}(A) \) and \( B \) are \( \epsilon \)-congruent by a rotation around \( c_i \)
(05)    then return YES fi;
(06)  if \( T_{c \epsilon c_i}(A) \) and \( B \) are \((\epsilon + \gamma)\)-congruent by a rotation around \( c_i \)
(07)    then possible ← true fi
(08) od;
(09) if possible
(10)  then return DON'T KNOW
(11) else return NO
(12) fi;
\end{verbatim}

Theorem 3 The algorithm above is an \((\gamma, \gamma)\)-approximate algorithm for approximate congruence with time complexity \( O((\gamma)^2 n^4) \).

Proof: If \( j \) exists s.t. \( T_{c \epsilon c_j}(A) \) and \( B \) are \( \epsilon \)-congruent, then \( A \) and \( B \) are \( \epsilon \)-congruent, too. On the other hand, every isometry that enables \( \epsilon \)-congruence for \( A \) and \( B \) maps \( c_A \) in \( U_{\epsilon}(c_B) \) and hence in some \( U_k \) of the covering. Therefore by Lemma 3 (with \( d = c_A \) and \( e = c_B \)) \( T_{c \epsilon c_k}(A) \) and \( B \) are \((\epsilon + \gamma)\)-congruent by a rotation around \( c_k \). Hence if there is no such \( k \), then \( A \) and \( B \) cannot be \( \epsilon \)-congruent. This shows the correctness of the algorithm.

8
If \( \varepsilon \) is less than \( \varepsilon_{opt}(A, B) - \gamma \), then \( T_{c_A c_B}(A) \) and \( B \) cannot be \( (\varepsilon + \gamma) \)-congruent. By Lemma 1 there is a ball, say \( U_j \), that contains the image of \( c_A \) under an isometry that enables congruence with tolerance \( \varepsilon_{opt}(A, B) \). Hence by Lemma 3 a rotation around \( c_j \) exists that enables approximate congruence for \( T_{c_A c_j}(A) \) and \( B \) with tolerance \( \varepsilon \) if \( \varepsilon \geq \varepsilon_{opt}(A, B) + \gamma \). So the interval where the algorithm may be unable to give an answer is \( [\varepsilon_{opt}(A, B) - \gamma, \varepsilon_{opt}(A, B) + \gamma] \). Since \( \frac{4\delta^2}{\gamma^2} \) balls of radius \( \gamma \) are sufficient to cover the \( \varepsilon \)-ball with center \( c_B \), there are \( O(\frac{\varepsilon^2}{\gamma^2}) \) iterations of the loop. As in the proof of Theorem 2, each execution of the loop has time complexity \( O(n^4) \).

Analogously an \( (\gamma, \gamma) \)-approximate algorithm for testing approximate congruence enabled by a translation can be constructed. Again, the \( \varepsilon \)-ball with center \( c_B \) is covered by \( \gamma \)-balls. For each center \( c_i \) we test whether the translation which maps \( c_A \) onto \( c_i \) enables \( \varepsilon \)-congruence or \( (\varepsilon + \gamma) \)-congruence. We get

**Theorem 4** There is an \( (\gamma, \gamma) \)-approximate algorithm for approximate congruence enabled by a translation with time complexity \( O((\frac{\varepsilon}{\gamma})^2 n^{2.5}) \).

### 4 Bit Complexity

The analysis of the algorithms of the previous sections is based on the Real-RAM model [9]. This means we have a RAM that can store reals and compute exactly with reals. From a theoretical point of view this model is an adequate simplification for developing algorithms in computational geometry, but from a practical point of view it is not. Real computer cannot handle real numbers. In practice, floating point arithmetic is used, and rounding and cancellation errors often cause many problems in robustness of the implemented algorithms. Theoretically correct algorithms compute wrong answers or no answers at all.

Since exact congruence is destroyed by perturbations of the data, it is evident that algorithms for deciding exact congruence have to be highly sensitive to rounding errors. Although rounding errors and inaccurate input data were the motivation for Alt et al. to investigate approximate congruence, their algorithms [1] are also highly sensitive to rounding errors. This is also true for the algorithm of Section 2 for approximate congruence enabled by a rotation around a given center. Rounding errors might lead to wrong answers, e.g. if the interval endpoints are not sorted in the right order. Moreover we may not assume that the centroids are correctly computed if floating point arithmetic is used. Hence the analysis of the uncertainty set becomes wrong.

Designing algorithms that use floating point arithmetic and can be guaranteed to produce an output that is the correct output for a slightly perturbed input is not useful for congruence problems. Since there is always an arbitrary small perturbation of the input s.t. the point sets are not congruent, a trivial algorithm that always decides that the point sets are not congruent without regard to the input obviously satisfies the above condition on the output.

If we want reliable answers for the actual input data, we have to avoid rounding errors. In the sequel we consider rational numbers and use rational arithmetic
with arbitrary precision. Since floating point numbers are rational, this is not a restriction. A rational number is stored as a pair of integers representing nominator and denominator. We call the maximum of the lengths of the binary representations of nominator and denominator the length of the represented rational number. In contrast to floating point arithmetic, space and time needed for the elementary arithmetic operations are not bounded by a constant, hence it is no longer fair to assume unit costs for arithmetic computations. In the remaining part of this section we redo the time analysis of Section 3 and base it on the bit complexity model where every manipulation of a single bit has unit costs. We assume that all input data have length $L$ at most.

The algorithm for deciding $\epsilon$-congruence by a rotation with a fixed center can be executed exactly with arbitrary precision rational arithmetic. Let us assume that the coordinates of the points in the two sets, the coordinates of the fixed center, and $\epsilon$ are rational numbers of length $\leq K$. In [1] it is shown that the cosinus values of the interval endpoints have the form $u + v\sqrt{2}$, where $u, v, z$ are rational numbers of length $O(K)$. Since two such numbers can be compared in time $O_B(K^{1+\delta})$, $\delta > 0$, the endpoints of the intervals $I_{l,k}$ can be sorted in time $O_B(n^2 \log n K^{1+\delta})$. Here index $B$ indicates that the bit complexity model is used for the time analysis. The matchings can be computed in time $O_B(n^4 \log n)$ where the additional $\log n$ factor in the time bound arises from computations with non-negative integers $\leq n$. Hence in the bit complexity model the algorithm has time complexity $O_B(n^4 \log n + n^2 \log n K^{1+\delta})$, where $K$ is the length of the rational coordinates of the input points.

Computing the centroid of $n$ points exactly with rational coordinates is quite expensive, because the length of a sum of two rational numbers may be twice as large as the length of the terms of the sum. With the fast integer multiplication algorithm of Schönhage and Strassen [11] the sum of two rational numbers of length $K$ can be computed in time $O_B(K \log K \log \log K) = O(K^{1+\delta})$, $\delta > 0$. Using a binary scheme for addition, the sum of $n$ coordinates can be computed in time $O_B((nL)^{1+\delta})$ (in the first step $n/2$ additions of rational numbers of length $L$ at most have to be done, in the second step $n/4$ additions of rational numbers of length $2L$ at most, ... and in the last step one addition of two numbers of length $n/2 L$ at most.) Since multiplication with $1/n$ can be done in time $O_B(nL \log n)$, the centroids $c_A$ and $c_B$ can be computed in time $O_B((nL)^{1+\delta})$. Their coordinates have length $O(nL)$.

Let $H$ be the length of $\gamma$. This implies $\gamma > 1/2^H$. If the error in the computation of the centroids is bounded by $\gamma/2$, every isometry that enables $\epsilon$-congruence for $A$ and $B$ maps the approximation point for $c_A$ into the the ball with radius $\epsilon + \gamma$ centered at the approximation point for $c_B$. We compute the exact value of $c_A$ in each coordinate using rational arithmetic and then compute the nearest point $z_A$ with integers coordinates to $2^H z_A$ and use $z_A/2^H$ as approximation for $c_A$. In the same way we compute an approximation $z_B$ to $c_B$. Then $z_A/2^H - c_A < \gamma/4$.

Since the coordinates of the input points are bounded by $2^L$, the coordinates of the centroids are also bounded by $2^L$. Hence the approximation points have coordinates of length $O(L + H)$ and can be computed in time $O_B(nL + H)$.
It remains to determine the centers of the covering balls. Let \((c_x, c_y)\) be the coordinates of the approximation for \(c_B\) and let \(t = \lceil \frac{\xi}{\gamma} \rceil\). We choose the points
\[
(c_x \pm k\gamma, c_y \pm j\gamma) \quad 0 \leq k, j \leq t + 1
\]
as centers of the covering balls. \(\lceil \frac{\xi}{\gamma} \rceil\) is an integer of length \(L + H + 1\) at most. Therefore each coordinate of a center has length in \(O(L + H)\). Also the points in \(T(A)\) have length in \(O(L + H)\). Hence each test for approximate congruence with tolerance \(\varepsilon\) or \(\varepsilon + \gamma\) can be done in time \(O_B(n^4 \log n + n^2 \log n(L + H)^{1+\varepsilon})\). We have

**Theorem 5** If all coordinates of the input points and \(\varepsilon\) are rational numbers of length \(L\) at most and \(\gamma\) is a rational number of length \(H\) at most, the \((\gamma, \gamma)\)-approximate algorithm described in Section 3 has time complexity
\[
O_B((\frac{\varepsilon}{\gamma})^2(n^4 \log n + n^2 \log n(L + H)^{1+\varepsilon}))
\]
in the bit complexity model.

Rational arithmetic is not sufficient for the general decision algorithm of [1] with rational input data. Non-rational real algebraic numbers may occur in some computations during an execution of the algorithm and a method for exact computations with real algebraic numbers is necessary. Using the representation of real algebraic numbers by isolating intervals and defining polynomials [6], the algorithm of Alt et al. has time complexity \(O_B(n^8 \log n + n^6 \log nL^{2+\varepsilon})\) [10]. There appears no dependence on the degree of the defining polynomials in the time bound because all polynomials computed by the algorithm have small degree bounded by some constant. The time bound improves if real algebraic numbers are represented by defining polynomials and sign sequences à la Thom [3]. Since coding à la Thom requires no computation of approximations like isolating intervals, computations with integers of length \(O(L)\) suffice and the running time is \(O_B(n^8 \log n + n^6 \log nL^{1+\varepsilon})\).

## 5 Conclusions

We have presented an \(O(n^4)\) algorithm for the decision problem for approximate congruence of two point sets in the plane which is guaranteed to give an answer if \(\varepsilon \notin [\frac{1}{2}\varepsilon_{opt}(A, B), 2\varepsilon_{opt}(A, B)]\), where \(\varepsilon_{opt}(A, B)\) is the smallest tolerance value permitting approximate congruence for \(A\) and \(B\). By an additional input parameter \(\gamma\) we can reduce the uncertainty interval to \(\varepsilon - \gamma, \varepsilon + \gamma\). The second algorithm has running time \(O((\frac{\xi}{\gamma})^2 n^4)\). The best known algorithm which always finds an answer has time complexity \(O(n^8)\). If the input coordinates and \(\varepsilon\) are rational numbers of length \(L\) at most and \(\gamma\) is a rational number of length \(H\) at most the algorithm has time complexity \(O_B((\frac{\xi}{\gamma})^2(n^4 \log n + n^2 \log n(L + H)^{1+\varepsilon}))\) in the bit complexity model if arbitrary precision rational arithmetic is used. Also we have presented approximate algorithms for the case that approximate congruence has to be enabled
by a translation. It is important to note that the algorithms are reliable, i.e. if an
decision is taken, it is correct.

We have used $\gamma$ as an additional parameter for tradeoff between length of the
uncertainty interval and running time. If we first test $\varepsilon$-congruence with $\gamma = \frac{\varepsilon}{2}$,
then with $\gamma = \frac{\varepsilon}{4}$, and so on until an answer is found, the procedure will terminate if
$\varepsilon \neq \varepsilon_{opt}(A, B)$. This algorithm has running time

$$O\left(\frac{\varepsilon}{\varepsilon_{opt}} \frac{1}{\varepsilon_{opt} - \varepsilon} \frac{1}{2^4 n^4}\right) = O\left(\frac{1}{\varepsilon_{opt}} \frac{1}{\varepsilon} \frac{1}{2^4 n^4}\right)$$

since the execution of the algorithm with the smallest value for $\gamma$ dominates the
running time. The algorithm is better than the algorithm of Alt et al. if

$$|\varepsilon - \varepsilon_{opt}| = \Omega\left(\frac{1}{n^2 \varepsilon}\right)$$

In order to ensure termination a new parameter $\mu$ can be used as a threshold value,
s.t. the algorithm stops if the actual value of $\gamma$ is less than $\mu$.

Lemma 1, Lemma 2, and Lemma 3 generalize directly to higher dimensions. Hence
the approximate algorithms for testing approximate congruence under translation
generalize to higher dimensions, too. The generalization of the algorithms for the
general case depends on the availability of an algorithm for deciding approximate
congruence for two point sets in $\mathbb{R}^d$ enabled by an isometry that fixes one point (a
rotation in three dimensional space).

Acknowledgement: The author thanks Kurt Mehlhorn for several helpful discus-
sions on the topic.

References

[1] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and
symmetries of geometric objects. Discrete and Computational Geometry, 3:237–


[3] M. Coste and M.F. Roy. Thom’s lemma, the coding of real algebraic numbers
and the computation of the topology of semi-algebraic sets. Journal of Symbolic

sponding sets of points. In Proc. of the 5th ACM Symp. on Comp. Geometry,


