Computing Largest Common Point Sets under Approximate Congruence

Christoph Ambühl[1], Samarjit Chakraborty[2], and Bernd Gärtner[1]

[1] Institut für Theoretische Informatik
[2] Institut für Technische Informatik und Kommunikationsnetze
ETH Zürich, ETH-Zentrum, CH-8092 Zürich, Switzerland
E-mail: ambuehl@inf.ethz.ch, samarjit@tik.ee.ethz.ch, gaertner@inf.ethz.ch

Abstract. The problem of computing a *largest common point set* (LCP) between two point sets under ε-congruence with the bottleneck matching metric has recently been a subject of extensive study. Although polynomial time solutions are known for the planar case and for restricted sets of transformations and metrics (like translations and the Hausdorff-metric under L_∞-norm), no complexity results are formally known for the general problem. In this paper we give polynomial time algorithms for this problem under different classes of transformations and metrics for any fixed dimension, and establish NP-hardness for unbounded dimensions. Any solution to this (or related) problem, especially in higher dimensions, is generally believed to involve implementation difficulties because they rely on the computation of intersections between algebraic surfaces. We show that (contrary to intuitive expectations) this problem can be solved under a rational arithmetic model in a straightforward manner if the set of transformations is *extended* to general affine transformations under the L_∞-norm (difficulty of this problem is generally expected to be in the order: translations $<$ rotation $<$ isometry $<$ more general). To the best of our knowledge this is also the first paper which deals with the LCP-problem under such a general class of transformations.

1 Introduction

Let $\varepsilon \geq 0$ be a real number, $\mathcal{G}$ be a transformation group (such as translations, rotations, isometry, or linear transformations), and A and B be two d-dimensional point sets. A *largest common point set* (LCP) [2, 3] between A and B under ε-congruence is a subset A' of A, having the largest possible cardinality, for which there exists a transformation $g \in \mathcal{G}$ such that the *distance* between the sets $g(A')$ and B' is less than ε, where B' is some subset of B, and the distance is measured using some appropriate metric. Related geometric problems for determining the similarity between two point sets have been extensively studied. Two commonly used metrics for quantifying the notion of similarity have been the *Hausdorff distance* [11, 12, 18] which is defined as the maximum distance between a point in one set and its nearest neighbor in the other set, and the *bottleneck matching metric* [14] seeks a perfect bipartite matching between two equal cardinality

point sets such that the maximum distance between any two matched points is minimized, and it returns this distance.

A systematic study of these problems was initiated by Alt *et al.* [5]. They presented algorithms for several versions of the problem for planar point sets under the bottleneck matching metric. In particular, they proposed an $O(n^8)$ decision algorithm to determine if there exists an isometric transformation using which two equal cardinality planar point sets can be brought to within ε distance of each other under the Euclidean bottleneck matching metric. Although they do not mention it explicitly, it is straightforward to adapt this algorithm to compute the LCP of two planar point sets (not necessarily of equal cardinality), without incurring any increase in running time.

Problems involving the *exact matching metric*, where two points are said to match only when the underlying geometric transformation takes one of them exactly on to the other have been studied in [1, 3, 13, 21]. Given two point sets, the problem of computing the minimum Hausdorff distance between them has been studied with different parameters such as only translation and general Euclidean motion, planar and d-dimensional point sets, and the underlying metrics being L_∞, L_1 and L_2 [11, 12, 18]. In an effort to improve the running time, various approximation algorithms for either the Hausdorff or the bottleneck metric for point sets in two, three, and in general d-dimensions have been presented in [9, 10, 15–17, 19, 20, 22].

Pattern matching using bottleneck metric It should be noted that most of the known exact algorithms, especially those involving three and higher dimensional point sets, are restricted to either the exact or the Hausdorff metric. While the exact metric is ill-posed for many practical applications, many problems also demand a one-to-one matching between the two point sets, thereby rendering the Hausdorff metric unsuitable in such situations [14]. This motivates the study of the problem using the bottleneck matching metric. However, in contrast to the Hausdorff metric where the distance between two point sets is in essence determined by the distance between two points each of which belongs to one of the sets, the existence of a bottleneck matching is a global property of the point sets and therefore complicates the problem. As a result, it is not apparent how the algorithms concerned with the Hausdorff metric can be adapted for computing the bottleneck matching. Neither do the algorithms of [5] extend from the planar case to work in three or higher dimensions in any simple way. Very recently, a new paradigm for point set pattern matching based on algebraic convolutions was proposed in [9] and [19]. This reduced the complexity of the problem under Hausdorff metric to nearly quadratic time. However, as noted in [20], "the one-to-one restriction imposed by bottleneck matching distance seems not to fit well within the rigid framework of algebraic convolutions".

Our results In this paper we present polynomial time exact (in contrast to approximation) algorithms for computing the LCP between two d-dimensional point sets under ε-congruence with the bottleneck matching metric under the L_2- and the L_∞-norms and various classes of transformations. All of our algorithms

are based on the general framework of traversing an arrangement of surfaces or hyperplanes in some high dimensional space, based on the dimensionality of the point sets. This can be considered as a generalization of the concepts in the original algorithms for bottleneck matching presented by Alt *et al.* in [5] and in those proposed more recently for the Hausdorff metric by Chew *et al.* in [11] and [12].

We prove that for unbounded dimensions the problem is NP-hard. All previous hardness results pertain to the exact matching metric and show that *subset matching* [3] is NP-hard for unbounded dimensions [1] and computing the LCP of an unbounded number of point sets even in one dimension is hard [2]. Polynomial time algorithms for d-dimensional point sets (fixed d) are known either for the exact metric [1] or for Hausdorff metric under the L_∞-norm and restricted set of transformations [11].

Many geometric pattern matching algorithms tend to suffer from implementation difficulties because of their reliance on algebraic techniques. As noted by Alt and Guibas [4] these algorithms "are probably difficult to implement and numerically unstable due to the necessary computation of intersection points of algebraic surfaces". We show that (surprisingly) if we extend our transformation group to include general affine transformations (isometries are a special case of this) then under the L_∞-norm this problem can be solved using a realistic rational arithmetic model. To the best of our knowledge this is the first time that the LCP-problem is being considered under such general transformations. All of the previous algorithms have considered either translations, rotations, isometries, or at most scaling [21]. It was claimed that the algorithms in [21] generalize to broader classes of transformations, but these were not dealt with explicitly.

Before proceeding further we first give a formal definition of our problem. A point set S is ε-congruent under the bottleneck matching metric and a transformation group $\mathcal{G} : \mathbb{R}^d \to \mathbb{R}^d$, to a point set S' if there exists a transformation $g \in \mathcal{G}$ and a bijective mapping $l : S \to S'$ such that for each point $s \in S$, $\delta(g(s), l(s)) < \varepsilon$, where $\delta(\cdot, \cdot)$ denotes some metric such as L_2 or L_∞ (we will only treat these two in the sequel). Given two point sets A and B, and a real number ε, the LCP between A and B is the maximum cardinality subset $A' \subseteq A$ which is ε-congruent to some subset of B. In what follows, ε-congruence is always assumed to be under the bottleneck matching metric.

We outline our basic method in Section 2. In Section 3 we present the two-dimensional realization of the general scheme, followed by the d-dimensional case in Section 4. In Section 5 we establish the NP-hardness result.

2 Computing the LCP – the general scheme

Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_m\}$ be two point sets in d-dimensional real space $\mathbb{R}^d$, and fix any metric δ on $\mathbb{R}^d$. For any transformation $L : \mathbb{R}^d \to \mathbb{R}^d$ and given indices $i \in [n], j \in [m]$, we define $\mathcal{T}_{ij}(L)$ as the set of translations that map the point $L(a_i)$ into the ε-neighborhood of b_j under the metric δ. Identifying

a translation with its defining vector v, we can write this set as

$$\mathcal{T}_{ij}(L) := \{v \in \mathbb{R}^d \mid \delta(L(a_i) + v, b_j) < \varepsilon\}.$$

If both, the transformation L and the translation vector v are fixed, then the *distance* between the point sets $L(A)+v$ and B can be computed by constructing a bipartite graph $G = (A \cup B, E)$ and computing the maximum matching in it. Here, for $a_i \in A$ and $b_j \in B$, the edge $(a_i, b_j) \in E$ if $\delta(L(a_i) + v, b_j) < \varepsilon$. Computing the LCP between A and B under ε-congruence essentially amounts to finding the optimal transformation L and the translation vector v under which the graph G has the largest maximum matching. For isometric transformations, we would restrict the transformation group from which L is chosen to consist of only pure rotation. This restricted definition of an isometry does not result in any loss of generality because isometry including mirror image just increases the computation time of any of our algorithm by only a constant factor. An algorithm now has to be run once with A and B, and then with A and a mirror image of B on a plane that can be chosen arbitrarily.

LCP under translation Here, the transformation L is fixed, and we are looking for the LCP between the point sets $L(A) = \{L(a_i) \mid i \in [n]\}$ and B under some translation v. The 'overlay' of all the possible sets of translations $\mathcal{T}_{ij}(L)$ for all indices $i \in [n]$ and $j \in [m]$ forms an *arrangement* $\mathcal{A}(L)$, inducing a decomposition of the space $\mathbb{R}^d$ into a number of *faces*. A face of $\mathcal{A}(L)$ is a maximal set of vectors with the property that any two v, v' in the set have the same relation to all $\mathcal{T}_{ij}(L)$, meaning that v lies inside (on the boundary of, outside) $\mathcal{T}_{ij}(L)$ if and only if the same is true for v'. Cells are faces not contained in any boundary of some $\mathcal{T}_{ij}(L)$.

If the metric δ is induced by the L_∞-norm or the L_2-norm, the sets $\mathcal{T}_{ij}(L)$ are balls of radius ε which are cubes in the former case and Euclidean-balls in the later. In the L_∞-case, the cells of the arrangement $\mathcal{A}(L)$ are then simply axis-aligned boxes, while they are bounded by spherical surface patches in the L_2-case. Also, the following property clearly holds.

Property 1. Let $v_{opt} \in \mathbb{R}^d$ be some translation that enables in computing the LCP between $L(A)$ and B. Then v_{opt} lies in some cell of $\mathcal{A}(L)$, and all vectors v' in that cell are optimal translations as well, meaning that they could also be used in computing the LCP.

This property results in the formulation of a discrete version of the problem since the process of finding v_{opt} now reduces to examining just one vector in every cell of $\mathcal{A}(L)$ and computing the maximum bipartite matching in the resulting graph. Any vector in the cell for which the graph has the largest matching serves as an optimal translation.

LCP under general transformations Now we turn to the more general problem of computing the LCP between A and B under any transformation L from some group (such as rotation, scaling, or linear transformations), followed

by a translation v. It is clear that as L varies, the LCP between $L(A)$ and B under translation remains invariant as long as the arrangement $\mathcal{A}(L)$ does not undergo any combinatorial change, meaning that a cell does not disappear and no new cells appear. The different possible combinatorial structures arising from the arrangement $\mathcal{A}(L)$ as L varies, partitions the space $\mathcal{L}$ of all transformations L under consideration into a number of cells. Each cell in this case is defined as the maximal set of transformations generating combinatorially equivalent arrangements $\mathcal{A}(L)$. Therefore the LCP-problem is reduced to a cell enumeration problem in an arrangement in the space $\mathcal{L}$. We are required to find a transformation L in every cell of this arrangement, compute $\mathcal{A}(L)$ and compute an optimal translation for that given L by solving the LCP-problem under translation.

3 LCP in 2-d under isometry with the L_2-norm

Let us assume without any loss of generality that the point set A is being rotated about the origin followed by a translation of the rotated set. Since the point sets involved are planar, the angle of rotation can be parametrized by a single parameter $\theta \in [0, 2\pi)$, the transformation L as described in Section 2 is therefore rotations and the space $\mathcal{L}$ is $[0, 2\pi)$. Since the underlying norm is L_2, for each pair of points $a_i \in A$ and $b_j \in B$, and a fixed angle θ, the set $\mathcal{T}_{ij}(\theta)$ of translations that take $R_\theta(a_i)$ (which is the point resulting out of rotating a_i about the origin by θ) into the ε-ball around the point b_j is a circular disk of radius ε in the space of possible translations. Overlaying all such $O(mn)$ circular disks for $i \in [n]$ and $j \in [m]$ forms the arrangement $\mathcal{A}(\theta)$. Corresponding to each cell c of this arrangement we can construct a bipartite graph $G_c = (A \cup B, E)$ where for each pair of points $a_i \in A$ and $b_j \in B$, the edge $(a_i, b_j) \in E$ if the cell c lies within the disk $\mathcal{T}_{ij}(\theta)$. We are interested in finding the graph G_c with the largest maximum bipartite matching over all the possible graphs corresponding to the different cells of $\mathcal{A}(\theta)$ arising from all possible values of θ.

As θ varies, the combinatorial structure of the arrangement changes as any of the following two conditions are fulfilled: two circular disks of $\mathcal{A}(\theta)$ touching and three disks meeting at a point. The first condition results in a linear equation in $\sin\theta$ and $\cos\theta$ and therefore can be transformed into a quadratic equation in either $\sin\theta$ or $\cos\theta$. The second condition results in a cubic equation in $\sin\theta$ and $\cos\theta$ and can be transformed into an algebraic equation of degree six in either of them. Since there are $O(mn)$ disks in the arrangement $\mathcal{A}(\theta)$, it gives rise to a collection of $O(m^3n^3)$ univariate algebraic equations of at most degree six.

The solutions to the equations result in partitioning the space $[0, 2\pi)$ into $O(m^3n^3)$ intervals and each interval corresponds to a set of combinatorially equivalent arrangements arising out of the angles θ lying within the interval. It may be observed that as we move from one interval to its adjacent one, the combinatorial structure of an arrangement $\mathcal{A}(\theta)$ (for any θ belonging to the former interval) changes with the introduction of a new cell or the disappearance of an existing one.

Our algorithm traverses the $O(m^3n^3)$ intervals of $[0, 2\pi)$ and at each interval if a new cell appears in the arrangement of disks then it constructs the bipartite graph corresponding to this cell and computes the maximum matching in it, which takes $O(mn\sqrt{m+n})$ time. The worst case overall running time of the algorithm is therefore $O(m^4n^4\sqrt{m+n})$.

4 LCP in d dimensions

This section gives polynomial-time algorithms for computing the LCP of two d-dimensional point sets, under isometries and general affine transformations, and under the L_2- as well as the L_∞-norm. The case of affine transformations with the L_∞-norm is special in the sense that it can be solved with rational arithmetic in a straightforward manner. In the other cases, one needs to resort to algebraic techniques even in two dimensions [5].

4.1 Affine transformations and the L_∞-norm

Following the general scheme given in Section 2, we consider the arrangement $\mathcal{A}(L)$ for any fixed linear transformation L. For any set $\mathcal{T}_{ij}(L) = (x_1, x_1 + \varepsilon) \times \ldots \times (x_d, x_d + \varepsilon)$, the values $x_t, x_t + \varepsilon$ are the first and the second *coordinate* of $\mathcal{T}_{ij}(L)$ in *direction* t. As L varies, the combinatorial structure of $\mathcal{A}(L)$ can only change at points where for some i, j, k, ℓ, $\mathcal{T}_{ij}(L)$ and $\mathcal{T}_{k\ell}(L)$ share a coordinate in some direction t. It is easy to see that

$$\mathcal{T}_{ij}(L) = \mathcal{T}_{ij}(id) + a_i - L(a_i) := \{v + a_i - L(a_i) \mid v \in \mathcal{T}_{ij}(id)\}, \tag{1}$$

where id denotes the identity transformation. Let x_t^{ij} and $x_t^{k\ell}$ be the coordinates of $\mathcal{T}_{ij}(id)$ and $\mathcal{T}_{k\ell}(id)$ in direction t.

According to (1), $\mathcal{T}_{ij}(L)$ and $\mathcal{T}_{k\ell}(L)$ then share a coordinate in direction t if and only if

$$x_t^{ij} - x_t^{k\ell} + (a_i - a_k - L(a_i - a_k))_t \in \{-\varepsilon, 0, \varepsilon\}. \tag{2}$$

Considering L as a $d \times d$-matrix with variable coefficients $x_{11}, \ldots x_{dd}$, we see that conditions (2) are linear equality constraints involving the variables $x_{t1}, \ldots x_{td}$.

In other words, all combinatorial changes in $\mathcal{A}(L)$ can be characterized by hyperplanes in d^2-dimensional space. Any cell in the arrangement $\mathcal{H}$ of those hyperplanes corresponds to a set of linear transformations L that generate combinatorially equivalent arrangements $\mathcal{A}(L)$.

To find the LCP of A and B under affine transformations, we therefore proceed as follows.

1. Traverse all cells of $\mathcal{H}$. Because $\mathcal{H}$ is defined by $O(m^2n^2)$ hyperplanes in dimension d^2, the number of cells is $(mn)^{2d^2}$ in the worst case, and all cells can be traversed in time proportional to their number, using *reverse search* with $O(m^2n^2)$ space [6, 23].

2. In each cell of $\mathcal{H}$, choose a representative transformation L (the reverse search algorithms [6, 23] used to traverse $\mathcal{H}$ actually generate such 'points' L, so there is no extra effort involved here). Traverse all cells in the arrangement $\mathcal{A}(L)$, which is an arrangement of unit cubes. The reverse search paradigm can also be adapted to this case within the space bounds of step 1 and with time $(mn)^{O(d)}$. For this, one can consider $\mathcal{A}(L)$ as a subset of cells of the arrangement induced by the facet-defining hyperplanes of all the cubes contributing to $\mathcal{A}(L)$.
3. In each cell of $\mathcal{A}(L)$, perform the graph matching. This can be done in $O((mn)^{O(1)})$ time.
4. Among all the matchings that have been computed, return the one with maximum cardinality.

The overall runtime of the algorithm is $(mn)^{2d^2+O(d)}$ which is polynomial for any fixed d. The space complexity is $O(m^2n^2)$, thus independent from d.

Note that step 3 can be improved by only considering the cells of $\mathcal{A}(L)$ which have been created in the combinatorial change encountered by going from $\mathcal{A}(L')$ to $\mathcal{A}(L)$, where L' is a representative transformation in some neighboring cell of $\mathcal{H}$ that has previously been processed. Typically, there are only few 'new' cells and they are easier to compute than the whole arrangement $\mathcal{A}(L)$. This approach is possible, because the reverse search traverses $\mathcal{H}$ along a tree of cells whose edges correspond to neighboring cells. However, as the complexity of this step is dominated by step 1 of the algorithm, we will not elaborate on this.

4.2 Incorporating isometries and the L_2-norm

First let us consider the case of general affine transformations under the L_2-norm. In this case, $\mathcal{A}(L)$ is an arrangement of Euclidean ε-balls, whose combinatorial structure changes if and only if some circumscribed ball of k ball centers ($2 \leq k \leq d+1$) attains radius ε. If $q_1, \ldots, q_k$ are those ball centers, it can be shown that their circumcenter c is given by $c = q_1 + \sum_{\ell=2}^{k} \lambda_\ell \mathbf{q}_\ell$, where $\mathbf{q}_\ell = q_\ell - q_1$, and the λ_ℓ are obtained from solving the system of linear equations

$$M \begin{pmatrix} \lambda_2 \\ \vdots \\ \lambda_k \end{pmatrix} = \begin{pmatrix} \mathbf{q}_2^T \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_k^T \mathbf{q}_k \end{pmatrix}, \quad M := \begin{pmatrix} 2\mathbf{q}_2^T \mathbf{q}_2 & \cdots & 2\mathbf{q}_2^T \mathbf{q}_k \\ \vdots & & \vdots \\ 2\mathbf{q}_2^T \mathbf{q}_k & \cdots & 2\mathbf{q}_k^T \mathbf{q}_k \end{pmatrix}. \tag{3}$$

The ball centers $\mathbf{q}_\ell$ depend linearly on the entries of L (cf. equation (1)); Cramer's rule then implies that the entries of $\det(M)M^{-1}$ are polynomials in the entries of L ($\det(M)$ itself is a polynomial too, of course). This again means that the condition

$$\| \sum_{\ell=2}^{k} \lambda_\ell \mathbf{q}_\ell \|^2 = \varepsilon^2 \tag{4}$$

for a combinatorial change can be written in the form $f(x) = 0$, where f is a constant-degree polynomial (the constant being in $O(d)$) in the entries

$x_{11}, \ldots, x_{dd}$ of the matrix L. Our strategy will be as before: we consider the set of all the $(mn)^{O(d)}$ polynomials f coming from the conditions on any k ($2 \leq k \leq d+1$) out of the mn ε-balls for the pairs of points in $A \times B$; these polynomials define an arrangement $\mathcal{H}$, whose cells correspond to combinatorial equivalence classes of arrangements $\mathcal{A}(L)$. By traversing the cells of $\mathcal{H}$, we can generate all the combinatorial types that $\mathcal{A}(L)$ may assume; if we perform for each type the graph matching in all the cells, the LCP will be found. We use the following result to 'compute' $\mathcal{H}$.

Theorem 1 (Basu, Pollack, Roy [7]). *Let $\mathcal{P} = \{f_1, \ldots, f_r\}$ be a set of p-variate polynomials with rational coefficients and maximum algebraic degree s ($p < r$). Two points $q, q' \in \mathbb{R}^p$ are* equivalent *if $\operatorname{sign}(f_\ell(q)) = \operatorname{sign}(f_\ell(q'))$ for $\ell = 1, \ldots, r$. The vector $(\operatorname{sign}(f_1(q)), \ldots, \operatorname{sign}(f_\ell(q))$ is a* sign condition *of $\mathcal{P}$. In $O(r(r/p)^p s^{O(p)})$ arithmetic operations, one can compute all sign conditions determined by $\mathcal{P}$.*

Applied to our setting with $r = (mn)^{O(d)}, p = d^2$, and $s = O(d)$, we can obtain all sign conditions in $(mn)^{O(d^3)}$ time. The 'full-dimensional' sign conditions (the ones containing no zero sign) correspond to the combinatorial equivalence classes of linear transformations L we are interested in. Therefore, this solves our problem, once we can obtain a representative transformation L from every such class.

Although full-dimensional sign conditions are always attained by suitable transformations L with rational coordinates, such transformations might be difficult to find. However, all we need is the combinatorial structure of $\mathcal{A}(L)$, and this can be derived directly from the sign condition. More precisely, for every subset $\mathcal{B}$ of ε-balls, we can check whether $\mathcal{A}(L)$ contains a point in all the balls of $\mathcal{B}$, by checking whether all $d+1$-element subsets of $\mathcal{B}$ have a common point (Helly's Theorem). The latter information is given by the sign condition, though. Processing sets $\mathcal{B}$ by increasing size in a dynamic programming fashion, we can construct all sets $\mathcal{B}$ that define cells of $\mathcal{A}(L)$ in time proportional to their number (which is $(mn)^{O(d)}$), incurring only some polynomial overhead.

The overall runtime (including all graph matchings) is then bounded by $(mn)^{O(d^3)}$

In order to handle isometries, we exploit the fact that L defines an isometry if and only if $L^{-1} = L^T$ (and $\det(L) = 1$, in case we want to limit ourselves to rotations). These conditions give rise to $O(d^2)$ additional polynomial equations which we add to the ones already obtained from the combinatorial change conditions. As before, Theorem 1 can be used to deal with this case in $(mn)^{O(d^3)}$ time.

4.3 L_2-norm and rotations in 3-space

The goal of this subsection is to give a concrete bound on the exponent entering the runtime in case of a three-dimensional LCP problem. Here it is crucial to use the fact that a rotation in 3-space has only 3 degrees of freedom. More precisely, a rotation in 3-space can be parametrized by three parameters ϕ_1, ϕ_2, ϕ_3,

and the resulting transformation matrix L has entries which are polynomials in $\sin\phi_i, \cos\phi_i, i = 1, 2, 3$. The combinatorial change conditions of type (4) mentioned in Section 4.2, resulting from all 2-, 3- and 4-tuples of balls give rise to $O(m^4n^4)$ polynomial equations of some constant maximum degree in six variables, say r_i and s_i, $i = 1, 2, 3$, where $r_i = \sin\phi_i$ and $s_i = \cos\phi_i$.

Consider a family of r polynomials $\mathcal{P}$ in p variables, each of degree at most s, and an algebraic variety $\mathcal{V}$ of real dimension p' which is defined as the zero set of a polynomial of degree at most s. Using $r^{p'+1}s^{O(p)}$ arithmetic operations it is possible to compute a set of points in each non-empty semi-algebraically connected component of $\mathcal{P}$ over $\mathcal{V}$, along with the signs of all the polynomials of $\mathcal{P}$ at each of these points [8]. The number of such points is $r^{p'}s^{O(p)}$. Applied to our case with the set $\mathcal{P}$ as the $O(m^4n^4)$ polynomials in the six variables r_i and s_i, $i = 1, 2, 3$, and the variety $\mathcal{V}$ as the zero set of the polynomial $\sum_{i=1,2,3}(r_i^2 + s_i^2 - 1)^2$ we can obtain in $O(m^{16}n^{16})$ time the sign conditions of $O(m^{12}n^{12})$ cells of the arrangement defined by $\mathcal{P}$. Given a fixed sign condition the corresponding arrangement $\mathcal{A}(L)$ consists of $O(m^3n^3)$ cells, (combinatorial descriptions of) which can be computed by dynamic programming as indicated before in $O(m^4n^4)$ time (for each cell we get an overhead which is at most linear in the number of balls). The graph matching for each cell takes $O(mn\sqrt{m+n})$ time, therefore resulting in an algorithm with total complexity $O(m^{16}n^{16}\sqrt{m+n})$.

5 NP-hardness of LCP

In this section we prove that the LCP problem under approximate congruence is (not surprisingly) NP-hard, even when the transformation group is restricted to only translations.[1] However, the proof establishes hardness also if we allow in addition general linear transformations, or isometries. The proof is a reduction from SAT. Suppose we are given a SAT formula ϕ with m clauses over n variables.

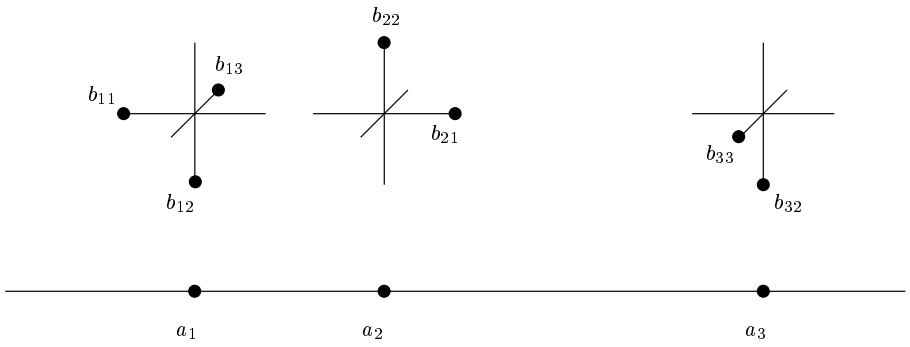

Fig. 1. Point sets for $\phi = (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3)$

[1] David Eppstein (personal communication) independently found another proof of this.

We transform ϕ into a pair of point sets $A = A(\phi)$, $B = B(\phi) \subseteq \mathbb{R}^n$ such that $|A| = m$, $|B| = k$, $k \geq m$ the number of literals in ϕ, and with the property that ϕ is satisfiable if and only if the LCP of A and B has the largest possible size m. This shows that LCP is NP-hard.

For every clause C_i, $i < m$ we create a point $a_i := (i, 0, \ldots, 0)$. For C_m, we define $a_m := (m+1, \ldots, 0)$. The collection of all the a_is forms the set A.

To get the set B, we proceed as follows. For every variable x_j occurring in clause C_i (in form of x_j or $\bar{x}_j$) we introduce a point b_{ij}, with coordinates

$$b_{ij} = a_i \pm \varepsilon' e_j,$$

where we choose the plus sign if the variable occurs non-negated, and the minus sign otherwise. e_j is the j-th unit vector, and ε' is chosen slightly larger than ε (we will see below what that exactly means). The collection of all the b_{ij}s forms the set B. Fig. 1 gives an example for $m = n = 3$ (for better readability, the set B has been shifted vertically; this does not change the LCP, of course, if translations are among the allowed transformations).

Theorem 2. *The formula ϕ is satisfiable if and only if the ε-LCP of $A(\phi)$ and $B(\phi)$ has size m.*

Proof. Let us first assume ϕ is satisfiable, and let $\tilde{x}_i, i = 1 \ldots, n$ with $\tilde{x}_i \in \{\texttt{true}, \texttt{false}\}$ be a satisfying assignment. Define an n-vector v by

$$v_i = \begin{cases} \varepsilon'/n, \text{ if } \tilde{x}_i = \texttt{true} \\ -\varepsilon'/n, \text{ otherwise.} \end{cases}$$

The claim is that v defines a translation which maps every a_i into the ε-neighborhood of some b_{ij}, thus giving rise to a ε-LCP of size m. To see this, consider a fixed a_i and let x_j be a variable whose value $\tilde{x}_j$ is responsible for satisfying clause C_i. If x_j occurs in C_i unnegated, we have

$$a_i + v - b_{ij} = v - \varepsilon' e_j = \begin{pmatrix} \pm\varepsilon'/n \\ \vdots \\ \pm\varepsilon'/n \\ -(n-1)\varepsilon'/n \\ \pm\varepsilon'/n \\ \vdots \\ \pm\varepsilon'/n \end{pmatrix}.$$

This implies

$$\|a_i + v - b_{ij}\|_2 = \varepsilon' \sqrt{1 - \frac{1}{n}} \leq \varepsilon,$$

if ε' is suitably chosen, so a_i is indeed approximately matched with b_{ij}. The case where x_j occurs negated in C_i is similar.

This argument actually holds for any L_p-norm with $p \geq 2$, in particular for the L_∞-norm.

For the other direction, let us assume that a ε-LCP of size m exists, i.e. all a_i are matched. We first observe that for any i, a_i must have been matched to some b_{ij}. This holds by construction, if ε is sufficiently small ($\varepsilon \ll 1$), which we assume for the rest of the proof. Under translations, this is quite obvious, and under general affine transformations, the gap between a_{n-1} and a_n prevents the only alternative matching $a_i \leftrightarrow b_{n+1-i,j}, i = 1 \ldots n$.

Now we define an assignment of truth values to variables as follows: we set x_j to `true` (`false`) if some a_i is matched to b_{ij}, where x_j occurs non-negated (negated) in C_i. If none of the two cases occur, the value of x_j is arbitrary. This is well-defined, because if x_j occurs negated in C_i and non-negated in C_k, then $\|(a_i - a_k) - (b_{ij} - b_{kj})\| = 2\varepsilon'$, which means that one of $\|a_i - b_{ij}\|$ and $\|a_k - b_{kj}\|$ must be at least $\varepsilon' > \varepsilon$. This shows that if some a_i is matched to a point b_{ij} coming from a non-negated (negated) instance of x_j, then this holds for all a_k that are matched to b_{kj}. This also easily implies that the defined truth-assignment satisfies one literal in every clause.

6 Concluding remarks

The main aim of this paper was to establish polynomial time bounds for solving the LCP-problem in bounded dimensions under bottleneck matching. As a consequence the time bounds presented here are not the tightest possible ones. We believe that there is scope for reasonable improvements in this direction. For example, the algorithm for planar point sets presented in Section 3, which is a straightforward realization of our general scheme, runs in time $O(n^{8.5})$ compared to $O(n^8)$ of [5]. Both the algorithms require solutions to algebraic equations of degree six. However, the number of events in which three disks touch at a point can be bounded by $O(m^2n^3)$ instead of $O(m^3n^3)$, using the fact that the dynamic Voronoi diagram of k sets of rigidly moving points on a plane with each set having m points has a complexity $O(m^2k^2 \log^* k)$ [12]. This reduces the worst case time complexity of our algorithm to $O(n^{7.5})$. For realistic point sets it would be faster since the number of graph matchings depend on the number of combinatorial changes in the arrangement of disks as the angle of rotation varies, which in general would be much smaller.

References

1. T. Akutsu. On determining the congruence of point sets in d dimensions. *Computational Geometry: Theory and Applications*, 9:247–256, 1998.
2. T. Akutsu and M.M. Halldórsson. On approximation of largest common subtrees and largest common point sets. *Theoretical Computer Science*, 233(1-2):33–50, 2000.
3. T. Akutsu, H. Tamaki, and T. Tokuyama. Distribution of distances and triangles in a point set and algorithms for computing the largest common point sets. *Discrete and Computational Geometry*, 20:307–331, 1998.
4. H. Alt and L. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 121–153. Elsevier Science Publishers B.V. North-Holland, 1999.

5. H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237–256, 1988.
6. D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete and Applied Mathematics*, 65:21–46, 1996.
7. S. Basu, R. Pollack, and M.-F. Roy. A new algorithm to find a point in every cell defined by a family of polynomials. In B.F. Caviness and J. Johnson, editors, *Proc. Symp. on Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer Verlag, 1995.
8. S. Basu, R. Pollack, and M.-F. Roy. On computing a set of points meeting every semi-algebraically connected component of a family of polynomials on a variety. *Journal of Complexity*, 13:28–37, 1997.
9. D.E. Cardoze and L.J. Schulman. Pattern matching for spatial point sets. In *Proc. 39th Annual Symposium on Foundations of Computer Science*, pages 156–165, 1998.
10. S. Chakraborty and S. Biswas. Approximation algorithms for 3-D common substructure identification in drug and protein molecules. In *Proc. 6th. International Workshop on Algorithms and Data Structures*, LNCS 1663, pages 253–264, 1999.
11. L.P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. *Discrete and Computational Geometry*, 21:257–274, 1999.
12. P. Chew, M. Goodrich, D. Huttenlocher, K. Kedem, J. Kleinberg, and D. Kravets. Geometric pattern matching under eucledian motion. *Computational Geometry: Theory and Applications*, 7:113–124, 1997.
13. P.J. de Rezende and D.T. Lee. Point set pattern matching in d-dimensions. *Algorithmica*, 13:387–404, 1995.
14. A. Efrat, A. Itai, and M. Katz. Geometry helps in bottleneck matching and related problems. To appear in Algorithmica.
15. P.J. Heffernan. The translation square map and approximate congruence. *Information Processing Letters*, 39:153–159, 1991.
16. P.J. Heffernan. Generalized approximate algorithms for point set congruence. In *Proc. 3rd. Workshop on Algorithms and Data Structures*, LNCS 709, pages 373–384, Montréal, Canada, 1993.
17. P.J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. In *Proc. 8th. Annual ACM Symp. on Computational Geometry*, pages 93–101, 1992.
18. D.P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete and Computational Geometry*, 9:267–291, 1993.
19. P. Indyk, R. Motwani, and S. Venkatasubramanian. Geometric matching under noise: Combinatorial bounds and algorithms. In *Proc. 10th. Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 457–465, 1999.
20. P. Indyk and S. Venkatasubramanian. Approximate congruence in nearly linear time. In *Proc. 11th. Annual ACM-SIAM Symp. on Discrete Algorithms*, 2000.
21. S. Irani and P. Raghavan. Combinatorial and experimental results for randomized point matching algorithms. *Computational Geometry: Theory and Applications*, 12:17–31, 1999.
22. S. Schirra. Approximate decision algorithms for approximate congruence. *Information Processing Letters*, 43:29–34, 1992.
23. N. Sleumer. Output-sensitive cell enumeration in hyperplane arrangements. In *Proc. 6th. Scandinavian Workshop on Algorithm Theory*, LNCS 1432, pages 300–309, 1998.