点集合の距離重複度列のノルムと最大部分集合問題

阿久津達也[†] 玉木 久夫[‡] 徳山 豪[‡]
[†] 東京大学医科学研究所ヒトゲノム解析センター
[‡] 日本アイ・ビー・エム東京基礎研究所

Abstract. ユークリッド平面内の二つの点集合 P,Q に対して、各々の部分集合で、互いに合同なものを共通部分集合という。この論文では点数最大の共通部分集合を計算する問題 (LCP) を考察する。そのために、距離重複度ベクトルの内積の概念を導入し、その解析を行なう。さらに、高次元の場合に、この内積の概念を拡張する。内積の値に対する上界を用いて、ＬＣＰの高速解法を設計する事が出来る。この講演原稿では、ページ制限の都合で４次元以上の場合の結果の解説は省く。

Distribution of Distances and Triangles in a Point Set and Algorithms for Computing the Largest Common Point Set

Tatsuya Akutsu[†] [1], Hisao Tamaki[‡] and Takeshi Tokuyama[‡]
[†] Human Genome Center, Institute of Medical Science, University of Tokyo, Tokyo 108, JAPAN
[‡] IBM Tokyo Research Laboratory,Yamato 242, JAPAN

Abstract. This paper considers the following problem which we call the largest common point set problem (LCP): given two point sets P and Q in the Euclidean plane, find a subset of P with the maximum cardinality which is congruent to some subset of Q. We introduce a combinatorial-geometric quantity $\lambda(P, Q)$, which we call the inner product of the distance-multiplicity vectors of P and Q, show its relevance to the complexity of of various algorithms for LCP, and give a non-trivial upper bound on $\lambda(P, Q)$. We generalize this notion to higher dimensions, give some upper bounds on the quantity, and apply them to algorithms for LCP in higher dimensions. Along the way, we prove a new upper bound on the number of congruent triangles in a point set in the four-dimensional space (which we omit in this abstract). Table 1 summarizes the algorithmic results of this paper, where we omit logarithmic factors and K denotes the size of the largest common point set. We have included some results on the largest similar point set problem (LSP), congruent copy detection (CCD) and the similar copy detection problem (SCD), where $n = |P|$, $m = |Q|$.

Dimensions	2	3	4	≥ 5
LCP	$n^{1.43}m^{1.77} + n^2$	$n^{1.8}m^{2.8} + n^3$	$n^{2.83}m^4 + n^4$	$n^{d-1}m^d + n^d$
LSP	n^2m^2(trivial)	$n^{7/3}m^{8/3} + n^3$	$n^3m^4 + n^4$	$n^{d-1}m^d + n^d$
LCP(Rand'zd)	LCP/K	LCP/K^2	NA	NA
ϵ-approx	LCP/K^2	LCP/K^3	NA	NA
CCD	$\min\{n^{1.43}m^{0.77}, n^{4/3}m\}$	$n^{1.8}m^{0.8} + \min\{n^{2.5}, n^3m^{-2}\}$	$n^{3.83}$	n^d
SCD	nm^2 [18]	$n^{3/7}m^{2/3} + n^3m^{-2}$	n^4	n^d

Table 1: Time complexities for LCP problems (logarithmic terms are omitted)

[1] Partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Informatics", of the Ministry of Education, Science and Culture of Japan.

1 Introduction

Matching of points sets in the plane or space is an important problem that arises, in various forms, in computer vision [14] and computational biology [2, 22, 17]. In the simplest form of the problem, we are asked if two finite sets of points P and Q are congruent, i.e., there is a transformation T consisting of translation, rotation, and possibly reflection such that $T(P) = Q$ [5, 19, 4]. A generalization of this question is to ask if Q is congruent to some subset of P (*congruent copy detection* or CCD for short)[7, 12, 18]. We consider in this paper yet further generalization: we ask for a set R of the maximum cardinality that is congruent to some subset of P and to some subset of Q at the same time. We call such R the *common point set* between P and Q and this problem the *largest common point set problem* or LCP. A version of LCP, in which congruence is replaced by similarity (i.e., we allow scaling as well) is previously considered by [15]. We set $n = |P|$, $m = |Q|$ and assume without loss of generality that $n \geq m$.

To solve LCP, we consider two standard schemes for point matching problems, namely *voting* and *alignment.* We consider both deterministic and randomized algorithms based on these schemes. Since the randomization techniques we use are also rather standard, our focus is on combinatorial-geometric analyses of the performance of those algorithms. In particular, we study a quantity which we call the *inner product of distance-multiplicity vectors* of two point sets. This quantity is naturally related to the running time of the algorithms for LCP we consider. We derive a new upper bound on this quantity and hence new upper bounds on the running times of those algorithms.

Given a set P of n points in the plane and a positive real l, we define the *multiplicity* of distance l in P, denoted by $H_P(l)$, to be the number of ordered pairs $(\boldsymbol{p}, \boldsymbol{q})$ of P such that the distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ is l. The *distance-multiplicity vector* of P, denoted by $M(P)$, is the vector $(H_P(l_1), \ldots, H_P(l_\nu))$ where $l_1, \ldots, l_\nu$ is the list of distinct distances with positive multiplicities, sorted in a non-decreasing order of their multiplicities in P. Erdös [9] asked for theoretical bounds on $\nu = \nu(P)$, the number of distinct distances in P, and $H_P[l_1]$, the largest multiplicity of a distance in P; currently, the best known upper bound on $H_P[l_1]$ is $O(n^{4/3})$ and the best known lower bound on $\nu(P)$ is $\Omega(n^{4/5})$ [20, 21].

Given two point sets P and Q, we define $\lambda(P, Q) = \sum_l H_P[l] H_Q[l]$, where the summation is taken over all distances occurring in P or in Q. We call $\lambda(P, Q)$ the "inner product" of the distance-multiplicity vectors $M(P)$ and $M(Q)$, a natural terminology if we regard $M(P)$ as a finite representation of a vector in $(\mathbf{Z}^+)^{\mathbf{R}}$. Since $\lambda(P, Q)$ represents the number of matches between the distances in P and those in Q, it is not difficult to imagine that this quantity may play an essential role in the complexity analysis of algorithms based on distance comparisons — the more distance matches we have, the more potential congruences we have to consider. We will see that this is indeed the case for both voting- and alignment-based algorithms.

Let $\lambda(n, m) = \max_{|P|=n,|Q|=m} \lambda(P, Q)$. The trivial upper bound on $\lambda(n, m)$ is $O(n^2 m^2)$, while a result of Erdös[9] implies that $\lambda(n, m) \geq \Omega(nm^2\sqrt{\log m})$. The above known bound of $O(n^{4/3})$ on the maximum multiplicity of a distance immediately implies that $\lambda(n, m) = O(n^{4/3} m^2)$. Our result is a stronger upper bound of $O(n^{1.43} m^{1.77})$. Note that his is an improvement for all values of $n \geq m$. This bound is proved by extending the technique of Székely, who has recently developed simplified proofs of the $O(n^{4/3})$ bound on the maximum multiplicity and the $\Omega(n^{4/5})$ bound on the number of distinct distances. $\lambda(n, m)$. We remark that further improving our upper bound, especially the special case of $O(n^{3.2})$ on $\lambda(n, n)$, is arguably difficult, because it would mean improving the best known lower bound $\Omega(n^{4/5})$ on $\nu(P)$, a long standing open problem. Indeed, if there is a point set P with only $O(n^{4/5})$ distinct distances, then $\lambda(P, P)$ is $\Omega(n^{6/5}(n^{4/5})^2) = \Omega(n^{3.2})$.

Similarly to the importance of the distance multiplicities in the plane, the multiplicity of distances and triangles plays an important role in the analysis of LCP algorithms in the three-dimensional space. Here, the multiplicity of a triangle ABC (not necessarily degenerate) in a point set P is defined to be the number of ordered triples (p, q, r) of P such that pqr is congruent to ABC. Triangle-multiplicity vectors and their inner-products are defined analogously to distance-multiplicity vectors. Given point sets P and Q in the three-dimensional space, let $\lambda^{(3,1)}(P, Q)$ ($\lambda^{(3,2)}(P, Q)$, resp.) denote the inner-product of the distance-multiplicity (triangle multiplicity, resp.) vectors of P and Q. Let $\lambda^{(3,1)}(n, m)$ and $\lambda^{(3,2)}(n, m)$ be defined as before, taking the maximum over P and Q with $|P| = n$ and $|Q| = m$. Our upper bounds are $\lambda^{(3,1)}(n, m) = n^{3/2}m^2(\log^* n)^{O(1)}$ and $\lambda^{(3,2)}(n, m) = n^{1.8}m^{2.8}(\log^* n)^{O(1)}$. These notions naturally carry over to higher dimensions: for a points set in $\mathbf{R}^d$.

From a practical point of view, one might question the relevance of the running time upper bounds of the LCP algorithms obtained from our combinatorial-geometric analyses, because they depend heavily on the exactness of the input and computation. Because of the inevitable errors in the input and computation, several authors formulate point matching problems as that of approximate matching[4, 7, 12]. For example, CCD is formulated as asking for a transformation T that minimizes the directed Hausdorff distance from $T(Q)$ to P, i.e., $\max_{q \in Q} \min_{p \in P} dist(T(q), p)$ [7]. Although this is a mathematically clean formulation and adopted by many authors, the time complexity of solving CCD in this model is rather high (the best known upper bound for the 2-dimensional case is $O(n^2m^3 \log^2 n)$ [7]). Goodrich *et al.* [12] propose to give up exact minimization and to look for a transformation that achieves a Hausdorff distance that is within some constant times the minimum possible, obtaining an algorithm with a reduced running time of $O(n^2m \log^2 n)$. It is not clear, however, if this approach can be extended to general LCP to produce algorithms with a practical running time. In fact, it is not even clear how we formulate LCP in this model, since we have two quantities to optimize (the subset size and the Hausdorff distance between the subsets).

A more heuristic approach for approximate matching is to take an algorithm for the exact matching model and use its approximate version, where approximate equality is used instead of exact equality to test matches. This is the approach taken by [15] and probably the one preferred in practice. The generalized Hough transformation method[6] commonly used in computer vision may be interpreted as belonging to such an approach. Our position is that it is meaningful to analyze idealized algorithms as long as their approximate versions are used in practice. The hope that such an analysis will tell us something about the performance of the approximate versions is only heuristic, as much as the hope that such approximate versions produce answers of any well-defined correctness.

Because of page limitation, we omit most of proofs in this version.

2 Algorithms

The goal of this section is to motivate our combinatorial-geometric analysis of the inner product of the multiplicity vectors by showing how it is related to the analysis of several algorithms for LCP. We will consider algorithms based on a voting scheme and algorithms based on an alignment scheme. In the description of the following algorithms, we will consider congruence transformations that consist only of translations and rotations, since reflections can be taken care of by running the algorithm once for P and Q and once for P and the mirror image of Q. We will refer to such congruence transformations simply as transformations. We will use K to always denote the size of the largest common point set between P and Q.

2.1 Voting scheme

In this and the next subsection, we assume that the point sets P and Q are in the plane. Let $dist(\boldsymbol{x}, \boldsymbol{y})$ be the distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$. Consider pairs $\boldsymbol{p}, \boldsymbol{p}'$ and $\boldsymbol{q}, \boldsymbol{q}'$ of points of P and Q, respectively. If $dist(\boldsymbol{p}, \boldsymbol{p}') = dist(\boldsymbol{q}, \boldsymbol{q}')$, these four points determine a unique transformation T such that $T(\boldsymbol{p}) = \boldsymbol{q}$ and $T(\boldsymbol{p}') = \boldsymbol{q}'$. This transformation is denoted by $T[\boldsymbol{p}, \boldsymbol{p}'; \boldsymbol{q}, \boldsymbol{q}']$.

For each pair $(\boldsymbol{p}, \boldsymbol{q})$ of points $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$, and each transformation T, we define $mult_{\boldsymbol{p},\boldsymbol{q}}(T)$ to be the number of pairs $\boldsymbol{x} \in P$ and $\boldsymbol{y} \in Q$ satisfying that $T[\boldsymbol{p}, \boldsymbol{x}; \boldsymbol{q}, \boldsymbol{y}] = T$. If $|T(P) \cap Q| = k$, it is easy to see that $mult_{\boldsymbol{p},\boldsymbol{q}}(T) = k-1$ for any $\boldsymbol{q} \in T(P) \cap Q$ and $\boldsymbol{p} = T^{-1}\boldsymbol{q}$. Hence, $\max_{\boldsymbol{p},\boldsymbol{q}}(\max_T mult_{\boldsymbol{p},\boldsymbol{q}}(T)) = K-1$ and the transformation achieving this maximum is the transformation that gives the largest common point set.

For each pair $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$, $\max_T mult_{\boldsymbol{p},\boldsymbol{q}}(T)$ can be computed by considering all matching pairs of edges $(\boldsymbol{px}, \boldsymbol{qy})$, $\boldsymbol{x} \in P$, $\boldsymbol{y} \in Q$, and letting each pair cast a vote to the transformation $T[\boldsymbol{p}, \boldsymbol{x}; \boldsymbol{q}, \boldsymbol{y}]$. We call this process the *local voting* for pair $(\boldsymbol{p}, \boldsymbol{q})$.

A similar voting idea is used in the generalized Hough transformation [6], where a vote is cast to a bucket in the transformation space rather than to an individual transformation as in our case. In other words, our scheme may be viewed as a limit of a version of the generalized Hough transformation as the bucket size goes to zero.

In the deterministic version of the algorithm, we need to execute the local voting process for each pair $(\boldsymbol{p}, \boldsymbol{q})$, with $\boldsymbol{p} \in P$ and $\boldsymbol{q} \in Q$, and take the best result. The total number of votes cast in the algorithm execution is $\lambda(P, Q)$, because each matching pair $(\boldsymbol{pp}', \boldsymbol{qq}')$ of edges contributes exactly one vote, namely in the local voting for the pair $(\boldsymbol{p}, \boldsymbol{q})$. Thus, $\lambda(n, m)$ bounds the essential term in the running time, with a multiplicative factor of $O(\log n)$ that accounts for the table maintenance for counting a vote.

Theorem 1 *In an efficient implementation, the deterministic voting algorithm computes the two-dimensional LCP in $O((\lambda(n, m) + n^2) \log n)$ time and $O(n)$ space.*

Proof: For each $\boldsymbol{p} \in P$, we first sort the set $\{(\boldsymbol{p}, \boldsymbol{x}) | \boldsymbol{x} \in P\}$ of pairs in terms of $dist(\boldsymbol{p}, \boldsymbol{x})$ and store it in a binary search data structure $D_{\boldsymbol{p}}(P)$. This needs $O(n \log n)$ time for each $\boldsymbol{p} \in P$. In the local voting for a fixed pair $(\boldsymbol{p}, \boldsymbol{q})$, we search $dist(\boldsymbol{q}, \boldsymbol{y})$, for each $\boldsymbol{y} \in Q$, in the data structure $D_{\boldsymbol{p}}(P)$ in $O(\log n)$ time to obtain the matches in P of the edge $\boldsymbol{qy}$. This needs $O(m \log n)$ time for each of nm local voting processes. The rest of the running time is at most $O(\log n)$ per vote. Thus, the overall running time is $O((\lambda(n, m) + nm^2 + n^2) \log n)$ but we have $\lambda(n, m) = \Omega(nm^2)$ as noted earlier. We omit how to reduce the space complexity to $O(n)$. □

Observe that, in the above deterministic voting algorithm, the optimal transformation T receives the maximum vote of $K-1$ in the voting process for each $(\boldsymbol{p}_i, T(\boldsymbol{p}_i))$, $i = 1, \ldots, K$, where $\{\boldsymbol{p}_i\}$ is the largest common point set. This is a redundancy which is difficult to avoid deterministically. We can reduce the redundancy, however, at the cost of a small failure probability using a standard random sampling procedure: sample a random subset R of P with size cn/K, for some sufficiently large c, and do local voting only for pairs in $R \times Q$. Here, once we fix a pair $\boldsymbol{p} \in R$ and $\boldsymbol{q} \in R$, the local voting for (p, q) is done exactly as before, scanning the entire sets P and Q (not restricting to R). This speeds up the algorithm by a factor of $|P|/|R| = c/K$: the terms $n^2 \log n$ and $nm^2 \log n$ of the deterministic time becomes $|R| n \log n$ and $|R| m \log n$, respectively, and the expected total number of votes cast is $(c/K)\lambda(P, Q)$. The algorithm makes an error only when R is disjoint from the largest common point set. The probability of this is at most e^{-c}.

So far, we have assumed that we know the value of K, which is not the case in general. However, we can efficiently and with high probability estimate K within a factor of, say, 2, using the randomized approximation scheme described below.

Theorem 2 *LCP can be solved in $O(c(\lambda(n,m)+n^2)K^{-1}\log n)$ time with probability $1-2^{-c}$, if $K > \log^3 n$.*

If we can be satisfied with an approximately optimal solution, that has size at least $(1-\epsilon)$ times the optimal, we can further reduce the running time by a different sampling strategy. We sample $\tilde{P} \subset P$ with αn points and $\tilde{Q} \subset Q$ with βm points, where $\alpha\beta = c\log n/K$ for some sufficiently large constant c, and compute the largest common set between $\tilde{P}$ and $\tilde{Q}$.

Theorem 3 *For any fixed $\epsilon > 0$, we can find, with high probability, a common point set between P and Q of size at least $(1-\epsilon)$ times the largest in expected $O((\lambda(n,m)+n^2)K^{-2}\log^3 n)$ time.*

2.2 Alignment scheme

In the *alignment* scheme, we consider each pair of edges $\boldsymbol{pp'}$ from P and $\boldsymbol{qq'}$ of Q with equal length, align $\boldsymbol{pp'}$ with $\boldsymbol{qq'}$, and count the number of points of P that now coincide with a point of Q. In other words, we count the size of the intersection $T(P) \cap Q$, where $T = T[\boldsymbol{p},\boldsymbol{p'};\boldsymbol{q},\boldsymbol{q'}]$. In the deterministic version, we need to consider $\lambda(P,Q)$ alignments and the counting for each alignment takes $O(m\log n)$ time. Alignments can be enumerated in a manner similar to the voting-based algorithm. Thus, we have a running time bound of $O((\lambda(n,m)m+n^2)\log n)$. This is worse than the voting-based algorithm. However, when we apply random sampling (for exact optimization), the rate of reduction in running time is larger in this case. Sample a subset R of P with size cn/K, with sufficiently large c so that R contains at least 2 points of the largest common point set with high probability. Then, we need only to consider each edge $\boldsymbol{pp'}$ in R for alignment. This is in contrast with the voting case, where we need to consider each edge $\boldsymbol{pp'}$ with $\boldsymbol{p} \in R$ and $\boldsymbol{p'} \in P$ for voting. Thus, the first term of the running time is reduced by a factor of c^2/K^2 and, when K is close to m, is comparable with the voting scheme. Irani and Raghavan [15] apply random sampling to an alignment scheme for LSP; this is reasonable because K is linear in m in the applications they intend.

For CCD, where our goal is to determine if $K = m$, the alignment scheme has an advantage of giving a fast Las Vegas algorithm (i.e. an algorithm that does not risk an error), in contrast to the voting scheme that gives a Monte Carlo randomized algorithm (i.e. an algorithm that has some probability of making an error). The idea is to sample from Q rather than from P. The reason we sample from P for the general LCP is because we want to reduce the $n^2\log n$ term in the deterministic time, which accounts for the preprocessing of distances in P . For CCD, this preprocessing can be avoided and sampling from Q turns out advantageous.

Theorem 4 *CCD can be solved in $O(n^{4/3}m\log n + n^{4/3}\log^{8/3} n)$ deterministic time and in $O(\min\{\lambda(n,m)m^{-1}\log n, n^{4/3}m\log n\} + n^{4/3}\log^{8/3} n)$ Las Vegas expected time.*

Proof: We randomly sample a pair $\boldsymbol{q},\boldsymbol{q'}$ of points from Q. Using the the methods of Agarwal et al. [1], we find all occurrences of the distance $dist(\boldsymbol{q},\boldsymbol{q'})$ in P in $O(n^{4/3}\log^{8/3} n + k\log n)$ time, where k is the number of occurrences of the distance. In the worst case, we have $k = O(n^{4/3})$, from the bound on the repeated distances, and the expected value of k is $\lambda(P,Q)/m^2$. For each such occurrence $\boldsymbol{pp'}$, we align $\boldsymbol{qq'}$ with $\boldsymbol{pp'}$ and test if the entire Q is matched into P, using $O(m\log n)$ time. The total running time is $O(km\log n + n^{4/3}\log^{8/3} n)$, from which the claim follows by plugging in the bounds on k. □

2.3 Higher dimensions

Both the voting and alignment schemes easily generalize to higher dimensions. We only discuss the voting scheme in three dimensional case here, because of page limitation. The generalization of the alignment scheme (and especially its specialization to CCD) can be done similarly.

Let the point sets P and Q be in the three-dimensional space. The local voting is now done fixing a pair of edges $\boldsymbol{pp'}$ from P and $\boldsymbol{qq'}$ from Q of equal length. Each pair of points $\boldsymbol{x} \in P$ and $\boldsymbol{y} \in Q$ such that the triangles $\boldsymbol{pp'x}$ and $\boldsymbol{qq'y}$ are congruent casts a vote to the transformation that matches the triangles. When these triangles are degenerate, i.e., $\boldsymbol{x}$ lies on line $\boldsymbol{pp'}$ and $\boldsymbol{y}$ on line $\boldsymbol{qq'}$, and hence does not determine a transformation, the vote becomes "public", i.e., it conceptually adds a count to every candidate in this particular local voting process. Public votes are counted separately and added to the maximum vote count in the end of the local voting. We do the local voting for every matching pair $(\boldsymbol{pp'}, \boldsymbol{qq'})$ of edges and take the transformation with the maximum vote count. It is clear that the maximum vote count is $K-2$.

The number of local voting processes to be performed is bounded by $\lambda^{(3,1)}(n,m)$ and the number of total votes cast by $\lambda^{(3,2)}(n,m)$. To enumerate matching pairs of edges for which to perform local voting, we use a sorted list of distances in P as we did in the planar case: $O(n^2 \log n)$ preprocessing time and $O(nm^2 \log n)$ query time. To enumerate matching triangles that cast votes, we similarly use a sorted list of triangles: $O(n^3 \log n)$ preprocessing time and $O(\lambda^{(3,1)}(n,m) m \log n)$ query time. Thus, we have:

Theorem 5 *LCP in the three-dimensional space can be solved in time* $O((\lambda^{(3,2)}(n,m) + \lambda^{(3,1)}(n,m)m + n^3) \log n)$.

When we use our current bounds on $\lambda^{(3,2)}(n,m)$ and $\lambda^{(3,1)}(n,m)$ that are roughly $O(n^{1.8}m^{2.8})$ and $O(n^{1.5}m^2)$ respectively, it turns out that the second term is subsumed by the first. The random sampling approaches also work for the three-dimensional space and gives a speed up of roughly K^2 for exact optimization and K^3 for approximate optimization. We omit the details.

3 Bounding $\lambda(n,m)$

We now turn to the combinatorial-geometric analysis of the inner-product of multiplicity vectors. We start with the planar case.

Theorem 6 $\lambda(n,m) = O(n^{1.43}m^{1.77})$.

We need some lemmas due to Székely [21, 16] (Lemmas 1 and 2 below), which he used in proving the $\Omega(n^{4/5})$ bound on the number of distinct distances in an arbitrary set of n points.

A crossing number of a drawing of a graph G on a plane is the number of intersections of drawn edges. Let $cr(G)$ denote the minimum of the crossing numbers over all drawings of G.

Lemma 1 *In the graph G with n vertices and t edges, if each edge has at most y edges parallel to it, then* $cr(G) = \Omega(t^3/yn^2) - O(y^2 n)$

Recall the notation in the introduction: $\nu(P)$ is the number of distinct distances in P, $\mathcal{D}(P) = \{l_1, ..., l_{\nu(P)}\}$ is the set of such distances listed in a non-increasing order of multiplicity, and $H_P[l]$ is the multiplicity of the distance l in P. Let $1 \le k \le \nu(P)$. We consider the subset $\{l_1, ..., l_k\}$ of $\mathcal{D}(P)$, consisting of the first k entries, and let $f(k) = \sum_{i=1}^{k} H_P(l_i)$. We create k concentric circles of radii l_1,..,l_k around each point of P. Hence, we have nk circles in total: let $C(k)$ denote the set of these circles.

The circles in $\mathcal{C}(k)$ have $2f(k)$ incidences with the point set P. We delete circles containing at most two points of P and let $\mathcal{C}'(k)$ denote the set of remaining circles. Each circle of $\mathcal{C}'(k)$ contains at least three points of P and these points cut the circle into at least three arcs. If we let A denote the set of these arcs coming from all the circles of $\mathcal{C}'(k)$, then $|A|$ is equal to the number of incidences between the circles of $\mathcal{C}'(k)$ and the points of P, and hence, is at least $2f(k) - 2kn$. Define a multigraph $G(k)$ whose vertices are points of P, and whose edges are arcs of A. Since each pair of at most kn circles of $\mathcal{C}'(k)$ intersect at most twice, the crossing number $cr(G(k))$ of $G(k)$ is at most k^2n^2.

The following lemma is due (in a slightly different form) to Székely.

Lemma 2 *For each y, the number of edges in $G(k)$ having at least y edges parallel to them is at most $O(n^2k/y^2 + kn\log n)$.*

Proposition 1 $f(k) = O(\min\{n^{10/7}k^{5/7}, n^2\})$.

Proof: $O(n^2)$ is a trivial bound. The $O(n^{10/7}k^{5/7})$ bound needs to be proved for $k < n^{4/5}$, where we can assume that $f(k) \gg kn\log n$.

We construct $G(k)$, set $y = \sqrt{Cn^2k/f(k)}$ for some constant C, and delete all edges of multiplicity larger than y. From Lemma 2, at most $f(k)$ edges are removed, if we set C sufficiently large. Then, we have a graph G with $\Omega(f(k))$ edges, whose edge multiplicity is at most y. From Lemma 1, $k^2n^2 > cr(G) = \Omega(f(k)^3/yn^2) - O(y^2n)$. Hence, $k^2n^2 = \Omega(f(k)^{3.5}/n^3k^{0.5})$ (the $O(y^2n)$ term is negligible). It follows that $f(k)^{3.5} = O(k^{2.5}n^5)$ and hence $f(k) = O(n^{10/7}k^{5/7})$. □

It is an easy excercise to obtain Theorem 6 from the above lemma.

4 Bounding $\lambda^{(3,1)}(n,m)$ and $\lambda^{(3,2)}(n,m)$.

We now turn to the three-dimensional space. In the following, $O'(f(n))$ will abbreviate $(\log^* n)^{O(1)}f(n)$, i.e., it hides polynomial factors of $\log^* n$. It is known that the number of occurrence of a given distance in a set of n points in the three dimensional space is $O'(n^{3/2})$ [8]. This trivially implies that $\lambda^{(3,1)}(n,m) = O'(n^{3/2}m^2)$. In the following, we derive an upper bound on $\lambda^{(3,2)}(n,m)$.

For each triangle Δ, let $H_P(\Delta)$ denote the multiplicity of Δ in P (i.e., the number of ordered triples of P forming an occurrence of Δ) and recall that $\lambda^{(3,2)}(P,Q) = \sum_\Delta H_P(\Delta)H_Q(\Delta)$. An upper bound on $H_P(\Delta)$ can be obtained by considering the following circle-point incidence problem. Let $\Delta = ABZ$ be an arbitrary triangle and h be the distance from Z to the line AB. For each pair $\boldsymbol{p}, \boldsymbol{q}$ of points of P with $dist(\boldsymbol{p}, \boldsymbol{q}) = dist(A,B)$, the trajectory of the vertex Z of a triangle congruent to Δ locating A and B at $\boldsymbol{p}$ and $\boldsymbol{q}$, respectively, forms a circle of radius h. Thus, if $N = O'(n^{3/2})$ is the multiplicity of distance $dist(A,B)$ in P, the multiplicity of Δ is at most the number of incidences between N circles and n points.

Lemma 3 *Given a set of N circles and n points in the d-dimensional space for $d \geq 3$, there are $O(N + n + \min\{N^{2/3}n, Nn^{1/2}, N^{4/5}n^{3/5}\log^* N\})$ circle-point incidences.*

Proof: Consider the bipartite graph $H = (V, W, E)$, where V corresponds to the set of circles, W corresponds to the set of points, and E is the edge set, where a vertex of V and a vertex of W are connected with an edge if the corresponding circle and point are incident. Since two circles intersect at most two points (in any dimensional space), graph does not contain $K_{2,3}$. Thus, we have $O(N + n + \min\{N^{2/3}n, Nn^{1/2}\})$ bound (a Canham-like bound). By projecting the figure to a two-dimensional plane, so that no pair of points overlap, we have an arrangement of ellipses. Now, similarly to [8], we can use cutting theory of arrangements, and obtain the $O(N^{4/5}n^{3/5}\log^* N)$ bound. □

From this lemma, it immediately follows that $H_P(\Delta) = O'((n^{3/2})^{4/5}n^{3/5}) = O'(n^{1.8})$ and hence $\lambda(n,m) = O'(n^{1.8}m^3)$. We show below that this bound can be slightly improved, using an idea similar to the one used in the analysis of $\lambda(n,m)$.

Theorem 7 $\lambda^{(3,2)}(n,m) = O'(n^{1.8}m^{2.8})$.

Note: Erdös and Purdy [11] gave an $O(n^{3-1/3})$ bound for the number of triangles with a given volume in the space, and asked a question about the number of pairwise congruent triangles, for which we have an $O'(n^{1.8})$ bound as a byproduct of the above Theorem (precisely, as a corollary of Lemma 3). We can similarly obtain an $O'(n^{2.2})$ bound on the number of pairwise similar triangles in the space.

参考文献

[1] P. Agarwal, B. Aronov, M. Sharir, and S. Suri, "Selecting Distances in the plane" *Algorithmica* vol 9, pp. 495-514, 1993.

[2] T. Akutsu, "Substructure search and alignment algorithms for three-dimensional protein structures," Proc. SIGAL-41, pp. 1–8, IPSJ, 1994.

[3] T. Akutsu, "On determining the congruency of point sets in higher dimensions," *Proc. ISAAC'94 (LNCS 834)*, pp. 38-55, 1994.

[4] H. Alt, K. Melhorn, H. Wagener and E. Welzl. "Congruence, similarity, and symmetries of geometric objects," *Discrete & Comput. Geom.*, vol. 3, pp. 237-256, 1988.

[5] M. Atkinson, "An Optimal Algorithm for Geometric Congruence" *J. Algorithms* vol. 8 pp. 159-172, 1987.

[6] D. H. Ballard, "Generalizing the Hough Transformation to Detect Arbitrary Shapes", Pattern Recognition, 13-2(1981), pp. 111-122.

[7] P. Chew, M. Goodrich, D. Huttenlocker, K. Kedem, J. Kleinberg, and D. Kravet, "Geometric Pattern Matching under Euclidean Motion" Proc. 5th CCCG pp. 151-156 (1993)

[8] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir and E. Welzl, "Combinatorial complexity bounds for arrangements of curves and spheres," *Discrete & Comput. Geom.*, vol. 5, pp. 99–160, 1990.

[9] P. Erdös, "On the set of distances of n points", *Amer. Math. Monthly* vol 53, pp.248-250 (1946).

[10] P. Erdös, "On the set of distances of n points in Euclidean space", *Magyar Tud. Akad. Mat. Kut. Int. Közl.* 5 (1960) pp.165–169. Also in *Paul Erdös: The Art of Counting (Selected Writing)* (J. Spencer, ed.) pp. 676-679, MIT Press, 1973.

[11] P. Erdös and G. Purdy, "Some extremal problems in geometry", *J. Comb. Theory* vol 10-3, pp. 246-252 (1971).

[12] M. T. Goodrich, J. S. B. Mitchell and M. W. Orletsky, "Practical methods for approximate geometric pattern matching under rigid motions," *Proc. 10th ACM SCG*, pp. 103–112, 1994.

[13] P. J. Heffernan. "Generalized approximate algorithms for point sets congruence," *Proc. Workshop on Algorithms and Data Structures*, pp. 373–384, 1993.

[14] J. E. Hopcroft and D. P. Huttenlocher, Geometric invariance on computer vision, chapter 18, pp.354-374. MIT Press, 1992.

[15] S. Irani and P. Raghavan, "Combinatorial and experimental results for randomized point matching algorithms" *Proc. 12th ACM SCG*, pp. 68-77, 1996.

[16] J. Matoušek, "Combinatorial and algorithmic geometry", Unpublished lecture notes.

[17] R. Motowani et al., "Hashing Molecules for Common Substructures" Working paper, 1996.

[18] P. J. de Rezende and D. T. Lee, "Point set pattern matching in d-dimensions," *Algorithmica*, vol. 13, pp. 387–404, 1995.

[19] K. Sugihara, "An $n \log n$ algorithm for determining the congruity of polyhedra" *J. Comput. and Sys. Sci.*, vol. 29, pp. 36–47, 1984.

[20] E. Szemerédi and W.T. Trotter, "Extremal problems in discrete geometry," *Combinatorica*, vol. 3, pp. 381–392, 1983.

[21] L. Székely, "Crossing numbers and hard Erdös problems in discrete geometry" Manuscript, 1996.

[22] G. Vriend and C. Sander, "Detection of common three-dimensional substructures in proteins," *PROTEINS: Structure, Function, and Genetics*, vol. 11, pp. 52-58, 1991.