Pattern Matching for Spatial Point Sets*

David E. Cardoze
Leonard J. Schulman
College of Computing
Georgia Institute of Technology
Atlanta GA 30332-0280
{cardoze, schulman}@cc.gatech.edu

Abstract

Two sets of points in d-dimensional space are given: a data set D consisting of N points, and a pattern set or probe P consisting of k points. We address the problem of determining whether there is a transformation, among a specified group of transformations of the space, carrying P into or near (meaning at a small directed Hausdorff distance of) D. The groups we consider are translations and rigid motions. Runtimes of approximately $O(n \log n)$ and $O(n^d \log n)$ respectively are obtained (letting $n = \max\{N, k\}$ and omitting the effects of several secondary parameters). For translations, a runtime of approximately $O(n(ak+1)\log^2 n)$ is obtained for the case that a constant fraction $a < 1$ of the points of the probe is allowed to fail to match.

1 Introduction

Two sets of points in d-dimensional space are given: a *data set* D consisting of N points, and a *pattern set* or *probe* P consisting of k points. One wishes to determine whether there is a transformation, among a specified group of tranformations of the space, carrying P into or near D.

B. Chazelle has called this the "constellation" problem. You are given a diagram of a constellation of k stars, and you wish to locate the constellation in the night sky or in a star chart. How would you do this?

The problem has been considered in the literature in a plethora of variations. The first parameter to consider is the *dimension* of the space. Second is the group of transformations: the groups considered have been either *Translations*, or *Rigid Motions* (translations and rotations), or *Rigid Motions with Scaling*. Third is whether, in case there are any satisfactory matches, the algorithm is required to provide just *one*, or *list all* of them. Fourth is what qualifies as a satisfactory match: only *Exact Matches*, carrying P into D; or all matches for which the directed Hausdorff distance from P to D is below a specified *Threshold* δ; or *Nearly Exact Matches*, in which the threshold is small enough compared to interpoint distances that every point of P matches to a unique and distinct point of D; or *Best Matches*, achieving the minimum possible directed Hausdorff distance from P to D; or *Approximate Best Matches*, approximating the true minimum. Fifth is whether *all* probe points are required to match, or whether *Point Failures* are allowed, i.e. the suitability of the match is judged after dropping the r worst-matching points of the probe.

A sixth distinction which is less important but will be useful in our work is whether the problem is *real-valued* or *integer*, i.e. whether the underlying space is $\mathbb{R}^d$ or $\mathbb{Z}^d$. In the former case there is generally some limit on the precision with which the points are known (in case they represent measured data); and even if in principle they can be known exactly, there is generally a limit on the precision with which the points are represented, if, as is usual, the data has been stored in floating point. Thus the most natural formulations of the real case are the Nearly Exact, Approximate Best Match, and Threshold formulations. The use of a threshold δ also allows for flexibility in use of the algorithm. By setting δ appropriately, the algorithm can identify matchings up to any specified tolerance for distortion, and if δ is chosen small, the only mappings of the probe into the data that are allowed are $1-1$, and distortion-free to within the precision with which the data are known or represented.

For Translations, in the Integer case, our method works for Exact, Threshold, Nearly Exact and Approximate Best Match. In the Real case, the method works for Threshold, Nearly Exact and Approximate Best Match. In all cases, any specified number of point failures can be tolerated.

For Rigid Motions, in the Real case (Integer considered as a special case), our method works for Threshold, Nearly Exact and Approximate Best Match.

*An earlier version of this manuscript appeared in the Proceedings of FOCS'98, 8-11 November 1998 in Palo Alto, CA, Copyright 1998 IEEE.

For each of these versions of the problem, the algorithm works for any dimension, and outputs the full list of matches.

In all cases, the method improves upon the best known algorithms by a factor of about k (up to log terms) in the runtime.

In order to do this we will allow ourselves a few kinds of leeway. (a) We will use randomization. (b) We will allow rare "false positives": matches may be announced which do not in fact satisfy the stated criteria. Any error probability p can be guaranteed, however, with only a $\log(1/p)$ multiplicative effect on the runtime. Importantly, the algorithms have no false negatives: no satisfactory match is ever overlooked. Note also that a typical probe and data pair is likely to have rather few matches announced by the algorithm, so that a simple explicit verification of the output of the algorithm will usually quickly produce a guaranteed complete answer to the given task. However, there *are* (probe,data) pairs having so many matches that simply verifying exhaustively the outputs of a correct run of our algorithm, would take far longer than the algorithm did. It is curious that the computational bottleneck of this problem appears to be verification. (c) Except for the Integer Exact Translations case, we will settle for Nearly Exact, rather than Exact Matches. But the precision can be as high as is desired, with modest computational cost.

Our runtimes depend on a "space size" parameter of the problem, s (essentially the diameter of the point sets), and on the desired quality of the solution as determined by the precision parameter ε and the accuracy (or threshold) parameter δ. Both s and δ have negligible impacts on the runtime of the algorithm.

We do not formally address the Real Exact formulation but, since our runtimes are faster than existing runtimes for that formulation, it is likely that the best heuristic way to solve it is a "generate and test" method, using the Nearly Exact version of our algorithm to generate matches in which the destinations of probe points in the data are uniquely identified, then testing whether these matches can be perturbed into exact matches.

Applications: The problems presented here, and some of their variants, have applications in such diverse fields as machine vision, document processing, computational biology and computational chemistry. In these disciplines they have been used in *pharmacophore identification* [11], *protein structure alignment* [1], *image registration* [24] and *model-based object recognition* [14, 26].

Our results and prior work: The problem is interesting already in one dimension. Up to now nothing better than $O(Nk)$ time algorithms were known. (For Exact Match one can simply test each possibility exhaustively; for Best Match one can "slide" the probe past the data, using an event list keyed on probe points crossing data points and Voronoi nodes.) Throughout this paper we use n to denote the maximum of N and k, the sizes of the probe and data sets; normally $n = N$. The first result of our paper is a randomized algorithm for Exact Match in the integer case running in time $O(n \log n + \log^{O(1)} s)$, where s is the diameter of the data set. We apply this result to obtain an algorithm for Threshold for the real case running in time $O(n\varepsilon^{-1} \log(n\varepsilon^{-1}) + \log^{O(1)}(\frac{s}{\varepsilon\delta}))$, where δ is the distance threshold and ε the precision parameter. (Nearly Exact and Approximate Best Match algorithms with similar runtimes are consequences.) As noted earlier, we allow a small probability (polynomial in n for these runtimes) of false-positive errors. In general for an error probability of n^{-b} the integer and real case runtimes are $O(bn \log n + \log^{O(1)} s)$ and $O(bn\varepsilon^{-1} \log(n\varepsilon^{-1}) + \log^{O(1)}(\frac{s}{\varepsilon\delta}))$ respectively.

The method extends to solve the Translation problem in higher dimensions. We solve the Integer Exact problem in time $O(bn \log n + \log^{O(1)}(sn))$, allowing an error probability of n^{-b}. The best previous result for Approximate Best Match was a runtime of $O(Nk \log N)$ in two dimensions, achieving approximation factor 2; and the same runtime in higher dimensions, achieving approximation factor of $2 + \varepsilon$ for any fixed $\varepsilon > 0$, both due to Goodrich, Mitchell, and Orletsky[13]. For Real inputs we improve this to a runtime of $O(bn\varepsilon^{-O(d)} \log(n\varepsilon^{-1}) + \log^{O(1)}(\frac{s}{\varepsilon\delta}))$, for Threshold Matches, with approximation factor $1 + \varepsilon$; the same runtime follows for Nearly Exact match with an appropriate value of δ, and a runtime slower by a factor of $\log\log(s\delta^{-1}) + \log\varepsilon^{-1}$ follows for Approximate Best Match. (Our precision parameter becomes the approximation factor for the case of Approximate Best Match.)

(Throughout this paper $b, \delta, \varepsilon, k, N, n, r, s$ are regarded as variable in the O notation, and d as a constant.)

In all of the above cases our method extends to enable us to detect all matches that have up to a specified number of point failures, r. In all cases the runtime (excluding the $\log^{O(1)} s$ term which remains unaffected) is multiplied by a factor of $(r + 1) \log n$, provided r is bounded by a constant fraction of the probe size.

We address the rigid motion problem in arbitrary dimension d, providing an $O(bn^d\varepsilon^{-O(d)} \log(n\varepsilon^{-1}) + \log^{O(1)} \frac{s}{\varepsilon\delta})$ time algorithm which, with error probability n^{-b}, identifies all rigid motion matches achieving Threshold δ to within precision ε. As before, the same runtime follows for Nearly Exact Match (with appropriate δ); Approximate Best Match follows with a further factor of $\log\log(s\delta^{-1}) + \log\epsilon^{-1}$. This improves (in the runtime and in addressing arbitrary ε) on the Approximate Best Match result provided in [13], which achieves approximation factor 4 in the plane in time $O(N^2k \log N)$, and approximation factor $8 + \varepsilon$ (for any fixed $\varepsilon > 0$) in $\mathbb{R}^3$ in time $O(N^3k \log N)$. De Rezende and Lee described an $O(N^dk)$ time solution to the Exact Rigid

Motion problem.

Recently and independently of our work, Indyk, Motwani and Venkatasubramanian [19] studied the rigid motions problem in its Threshold (hence also Nearly Exact or Approximate Best Match) formulation in two and three dimensions, with and without point failures. Their runtimes (omitting log terms as well as the dependence on a precision parameter and on the value of the threshold) are: in two dimensions, without point failures, $O(\min\{k(N^4 s)^{1/3}, N(s+N)\})$, and with point failures, $O(sNk)$; in three dimensions, without point failures, $O(\min\{k\max\{N^{2.25}s^{1/2}, N^{2.5}\}, Ns(s^2+N), N^2(N+s)\})$, and with point failures, $O(s^3 Nk)$.

Irani and Raghavan [20] provided a randomized algorithm for Nearly Exact Rigid Motions with Scaling in the plane, with point failures. Their runtime is $O(N^2 k \log N)$, which is slower than our $O(n^2 \log n)$ Nearly Exact Rigid Motions method for the plane, and solves a harder problem. They allow rare "false negatives" and no false positives, the opposite of our failure mode.

Other Work: Best Match for Rigid Motion is a more difficult problem than Exact Match. An $O(N^2 k^3 \log^2 N)$ time algorithm for the Best Match problem in the plane was given by Chew, Goodrich, Huttenlocher, Kedem, Kleinberg and Kravets [9].

The spatial pattern matching problem changes character considerably when the two sets play symmetric roles. For Exact Translations on the real line, Rote gave an $O(n \log n)$ time algorithm. For Exact Rigid Motions, with $k = N$, Atallah [3] provided an $O(N \log N)$ time algorithm in dimension 2 for planar figures. Atkinson [4] obtained the same runtime in dimension 3. Alt, Mehlhorn, Wagener and Welzl [2] obtained runtime $O(N^{d-2} \log N)$ in dimensions $d > 3$. The undirected Hausdorff distance between two sets is the maximum of each of the directed distances. Computation of the least such distance obtainable under a given group of transformations, is quite a different problem from computation of the least possible directed distance. Some of the known results are those of [15, 17] for translations, and [16] for rigid motions.

There is an obvious similarity between the one dimensional problem, and string matching problems. There are a variety of well-known and exceptionally efficient string matching methods [23, 22, 21, 8, 12]. The most important distinction between the problems is the sparsity of the data in the spatial point set problem. The spatial problem can be reduced by discretization to a string search problem, but the result is an instance of size s, which is typically far greater than n. Indeed, consider the constellations: the night sky is dark — the visible stars are sparse.

2 Algorithm: One Dimension

The one-dimensional problem has four variants, depending upon whether the points are reals or integers, and whether they lie on the line or on the circle. "Integers", in the case of the line, means that there is a length dividing all distances $|x-y|$, for $(x, y) \in D^2 \cup P^2$. In the case of the circle, this length must also divide the circumference of the circle.

The real case is handled by reduction to the integer case. We defer discussion of it to section 2.4.

To formalize the integer problem it is convenient to represent the probe and data by their characteristic functions $\overline{P}$ and $\overline{D}$ from $\mathbb{Z}$ or $\mathbb{Z}/s$ (s being the size of the circle) to $\{0, 1\}$.

The *space size* of a problem P, D is, for inputs on a circle, its size s; for inputs P, D on the line, the space size is $s(P, D) = \max\{|x-y| : x, y \in P^2 \cup D^2\} + 1$.

For integer t we let $P + t$ denote the set which contains x if and only if P contains $x - t$ (i.e. $P + t$ is the Minkowski sum of P and $\{t\}$); $D + t$ is defined similarly. Formally, our task is now:

Problem: Identify the set of "valid displacements," defined for the line by $M_{D,P} = \{t \in \mathbb{Z} : \forall x, \overline{P}(x-t) \leq \overline{D}(x)\}$ and for the circle by $M_{D,P} = \{t \in \mathbb{Z}/s : \forall x, \overline{P}(x-t) \leq \overline{D}(x)\}$.

Our main result for the integer case is:

Theorem 1 *Let integer pattern P and integer data D be given. Let $s = s(P, D)$. Choose any $b > 0$. Our algorithm runs in time $O(bn \log n + \log^{O(1)} s)$ and with probability at least $1 - n^{-b}$ produces the correct list $M_{D,P}$.*

Note: As is customary in computational geometry [27, 25], we adopt the real Random-Access Machine as our computational model. More concretely, our runtimes count FLOPs on a machine whose word length is large enough to hold the inputs (i.e. proportional to $\log s$.)

Proof: First we note mutual reductions between the line and circle cases. Given probe P and data D on the circle, reduce these to a line instance with space size $2s(P, D)$ as follows. Let $\text{Rep}_s : \mathbb{Z}/s \to \mathbb{Z}$ be the function that assigns to every $x \in \mathbb{Z}/s$ the integer $0 \leq y < s$ such that $x \equiv y \bmod s$. Map P to $L(P) = \cup_{x \in P}\{\text{Rep}_s(x)\}$. Map D to $L(D) = \cup_{x \in D}\{\text{Rep}_s(x), \text{Rep}_s(x) + s\}$. Note that t is a valid displacement for (D, P) iff $\text{Rep}_s(t)$ is a valid displacement for $(L(D), L(P))$. Thus we can "pull back" the valid translations for D and P from those for $L(D)$ and $L(P)$. The cost of the reduction is a doubling of the data set and of the space size.

The line problem is essentially a special case of the circle problem: given probe P and data D on the line, reduce these to a circle instance with space size $u \geq 2s(P, D)$ as follows.

Map P to $C(P) = \{x \in \mathbb{Z}/u : \exists p \in P$ such that $x = p \bmod u\}$. Similarly map D to $C(D) = \{x \in \mathbb{Z}/u : \exists d \in D$ such that $x = d \bmod u\}$. Note that t is a valid displacement for (D, P) iff t mod u is a valid displacement for $(C(D), C(P))$, and $d_{\min} - p_{\max} \leq t \leq d_{\max} - p_{\min}$. Thus we can pull back the valid translations for D and P from those for $C(D)$ and $C(P)$. The cost of the reduction is a doubling of the space size. For future reference note that both line-circle reductions hold in higher dimension (read now unbounded-torus); costs increase by 2^d rather than doubling.

Outline of the method: The original, sparse problem is subjected to several different randomizations. Each of these is transformed by a space reduction procedure to a dense problem, which is then solved using a Fast Fourier Transform. In each of the randomizations, false positive matches may appear due to the space reduction and, depending on how it is performed, the Fourier Transform. However, due to statistical independence of the trials, when the different randomized solutions are combined there is only a small probability of any false positive remaining.

We begin with pseudocode for the main procedure calls of the algorithm, and then explain each of the procedures. Thanks to the above reduction we can assume that the input to the algorithm is a line problem.

Translations(P, D)

1. If n is less than some threshold run the naive algorithm, otherwise continue.
2. $u :=$ SelectPrime$(2s)$
3. $(P_1, D_1) := C(P, D, u)$
4. Repeat for $i \in \{1, ..., (b+1)\log_{\frac{4}{3}} n\}$

   $\mu_i :=$ FindTranslations$(P_1, D_1, 2^{\lceil \lg(20n) \rceil})$.
5. Return the pullbacks μ of $\cap_i \mu_i$ to (P, D).

FindTranslations(P, D, h)

1. $(P_1, D_1) :=$ RandomMultiply(P, D).
2. $(P_2, D_2) := L(P_1, D_1)$.
3. $(P_3, D_3) :=$ ReduceSpace(P_2, D_2, h).
4. $\nu :=$ FindMatches(P_3, D_3)
5. Return the pullback of ν to (P, D).

We now describe each of the underlying procedures.

2.1 SelectPrime

The procedure SelectPrime(n) returns a prime between n and $2n$. This can be done in time $\log^{O(1)} n$ by choosing integers at random within the given range and testing for primality (see [5]).

2.2 FindTranslations

The procedure FindTranslations(P, D, h) takes three arguments: a probe set P, a data set D, both in $\mathbb{Z}/u$ for some prime u; and an integer h. FindTranslations returns a superset of $M_{D,P}$ which is not likely to be too much larger than $M_{D,P}$.

2.2.1 RandomMultiply

The arguments to RandomMultiply(P, D) are a probe set P and data set D, both in $\mathbb{Z}/u$ for some prime u. The procedure first chooses an integer q uniformly at random from $\{1, 2, \ldots, u-1\}$. The circle $\mathbb{Z}/u$ is then carried bijectively to itself under $R_{u,q}(x) = xq$ mod u, yielding a new probe $R_{u,q}(P) = \cup_{x \in P}\{R_{u,q}(x)\}$ and a new data set $R_{u,q}(D) = \cup_{x \in D}\{R_{u,q}(x)\}$; the valid displacements are transformed bijectively, t to $R_{u,q}(t)$.

2.2.2 Space Reduction

The arguments to ReduceSpace(P, D, h) are a probe P, a data set D in $\mathbb{Z}$, and a target space size h. A map $H_h : \mathbb{Z} \to \mathbb{Z}/h$ is defined by $H_h(x) = x$ mod h, and new probe and data sets are defined on the circle by $H_h(P) = \cup_{x \in P}\{H_h(x)\}$ and $H_h(D) = \cup_{x \in D}\{H_h(x)\}$. We explicitly store the preimages $H^{-1}_{h,P}$ of every new probe point, and $H^{-1}_{h,D}$ of every new data point.

Let p_0 be an arbitrarily chosen element of P. The pullbacks of any valid displacement τ that is identified for the problem $(H_h(P), H_h(D))$ are defined to be the displacements $H^{-1}_{h,D}(H_h(p_0) + \tau) - p_0$.[1] We denote the set of all pulled-back displacements by $H_h^{-1}(M_{H_h(D), H_h(P)})$.

Every match of P into D is carried to a match of $H_h(P)$ into $H_h(D)$. The converse, however, may fail: new matches can be created by this process. (I.e. a pullback may not be a valid displacement.)

Proposition 1 $M_{D,P} \subseteq H_h^{-1}(M_{H_h(D), H_h(P)})$.

Proof: The point is H_h is a homomorphism. Specifically, let $t \in M_{D,P}$. Then for any $p \in P$, $p + t$ equals some $d \in D$; and $H_h(p) + H_h(t) = H_h(p+t) = H_h(d) \in H_h(D)$. Therefore $H_h(t) \in M_{H_h(D), H_h(P)}$. Let $d_0 \in D$ be s.t. $d_0 = p_0 + t$: then $d_0 \in H^{-1}_{h,D}(H_h(p_0) + H_h(t))$, and $t \in H^{-1}_{h,D}(H_h(p_0) + H_h(t)) - p_0$. □

[1] There may be several such, if $|H^{-1}_{h,D}(H_h(p_0) + \tau)| > 1$.

2.2.3 Finding Matches

The procedure FindMatches(P, D) takes two arguments: a probe P and a data set D on a circle of size h (a power of 2). Following an idea introduced by Fischer and Paterson [12], the procedure works as follows. Let $\tilde{D}$ be the complement of D. Note that $P + t \subseteq D$ if and only if $\overline{P + t} \cdot \overline{\tilde{D}} = 0$ (boolean dot product). The list of all these dot products is the convolution of $\overline{\tilde{D}}$ with the function whose value at x is $\overline{P}(-x)$; this convolution can be computed in $O(h \log h)$ time using an FFT. Better yet, a randomized approximation to this convolution can be obtained in time $O(h)$ using the following result:

Theorem (Indyk)[2] [18]: Let $u, v \in \{0,1\}^{\mathbb{Z}/h}$. There exists a randomized algorithm which in $O(h)$ time computes an approximation of the boolean convolution of u and v in the sense that it outputs a vector $w \in \{0,1\}^{\mathbb{Z}/h}$ such that for any i:

1. *If $(u * v)[i] = 0$ then $w[i] = 0$,*
2. *If $(u * v)[i] = 1$ then $Pr[w[i] = 1] = \frac{1}{2}$.*

After computing either the exact or randomized-approximate convolution, FindMatches returns all translations t for which the coefficient in w is 0.

2.3 Overall Analysis

We begin with an upper bound on the probability that FindTranslations outputs a fixed invalid translation t for D and P, the inputs to FindTranslations. Let $N_{D,P}$ be the set of invalid translations for D and P, that is $N_{D,P} = \{t \in \mathbb{Z}/u : P + t \not\subseteq D\}$. For each $t \in N_{D,P}$ fix an element of P, call it $A(t)$, such that $A(t) + t \notin D$.

Lemma 1 *Let $P, D \subseteq \mathbb{Z}/u$ be the inputs to FindTranslations. Let $p \in P$ and $d \in D$. If t is such that $p + t \neq d$ then the probability that either*

(i) Rep$_u(R_{u,q}(d)) \equiv$ Rep$_u(R_{u,q}(p))$ + Rep$_u(R_{u,q}(t))$ $\pmod h$, or

(ii) Rep$_u(R_{u,q}(d)) + u \equiv$ Rep$_u(R_{u,q}(p))$ + Rep$_u(R_{u,q}(t))$ $\pmod h$

after step 2 of FindTranslations, is at most $\frac{8}{h}\frac{u}{u-1}$.

Proof: Consider case (i). It implies that there is a nonzero r satisfying $-\frac{2u}{h} \leq r \leq \frac{2u}{h}$ such that

$$\text{Rep}_u(qd) - \text{Rep}_u(qt) - \text{Rep}_u(qp) = rh$$

and therefore

$$\begin{aligned} qd - qt - qp &= rh \pmod u \\ q &= (d - t - p)^{-1} rh \pmod u \end{aligned}$$

For any fixed r, the probability that $q \equiv rh \cdot (d-t-p)^{-1} \pmod u$ is $\frac{1}{u-1}$. Thus the probability of case (i) occurring is at most $\frac{4}{h}\frac{u}{u-1}$. A similar argument applies to case (ii). □

Lemma 2 *Let $t \in N_{D,P}$, where D and P are the inputs to FindTranslations. The probability that Rep$_u(R_{u,q}(t)) \in H_h^{-1}(M_{H_h(D_2),H_h(P_2)})$ after step 2 of FindTranslations is at most $n\frac{8}{h}\frac{u}{u-1}$.*

Proof: Suppose Rep$_u(R_{u,q}(t)) \in H_h^{-1}(M_{H_h(D_2),H_h(P_2)})$ after step 2 of FindTranslations. Then there exists $d \in D$ such that either:

(i) Rep$_u(R_{u,q}(d)) \equiv$ Rep$_u(R_{u,q}(A(t)))$ + Rep$_u(R_{u,q}(t))$ $\pmod h$, or

(ii) Rep$_u(R_{u,q}(d)) + u \equiv$ Rep$_u(R_{u,q}(A(t)))$ + Rep$_u(R_{u,q}(t))$ $\pmod h$

From the previous lemma the probability of this occuring is at most $\frac{8}{h}\frac{u}{u-1}$. Thus the probability that Rep$_u(R_{u,q}(t)) \in H_h^{-1}(M_{H_h(D_2),H_h(P_2)})$ is at most $n\frac{8}{h}\frac{u}{u-1}$. □

Corollary 1 *Let $t \in N_{D,P}$, where (P, D) are the inputs to FindTranslations. Then $Pr(t \in \mu_i) \leq \frac{3}{4}$.*

Proof: Let $t \in N$. Let F be the event $[t \in \mu_i]$, and C the event $\left[\text{Rep}_u(R_{u,q}(t)) \in H_h^{-1}(M_{H_h(D_2),H_h(P_2)})\right]$. Note that $Pr(C) \leq \frac{1}{2}$ since $h \geq 20n$. Therefore

$$\begin{aligned} Pr\{F\} &\leq Pr\{C\} + Pr\{\neg C\} \cdot Pr\{F|\neg C\} \\ &\leq \frac{1}{2} + \frac{1}{2}P\{F|\neg C\} \leq \frac{3}{4} \end{aligned}$$

□

We construct the list of matches ν by identifying all the shifts t for which the convolution in FindMatches obtains a value of 0. Each such match $t \in \nu$ is pulled back, as explained in sections 2.2.1 and 2.2.2, to a candidate displacement for (P, D) and included in μ_i.

We now bound the probability of failure of the Translations algorithm.

Lemma 3 *Let D and P be the inputs to the Translations algorithm, and μ its output. The probability that $\mu \neq M_{D,P}$ is at most n^{-b}.*

[2] Indyk's result is stated for convolution on $\mathbb{Z}$ but the translation to $\mathbb{Z}/h$ is immediate.

Proof: Recall that $M_{D_1,P_1} \subseteq \cap_i \mu_i$. Now let $t \in N_{D_1,P_1}$. We know that the probability that $t \in \mu_i$ for $i \in \{1, \ldots, (b+1)\log_{\frac{4}{3}} n\}$ is at most $\frac{3}{4}$. Thus the probability that $t \in \cap_i \mu_i$ is at most $\frac{1}{n^{b+1}}$. If $t \notin D_1 - p_0$, it cannot appear in any μ_i. Therefore by a union bound, $Pr(M_{D_1,P_1} \neq \cap_i \mu_i) \leq n^{-b}$. The map from $M_{D,P}$ to M_{D_1,P_1} is bijective and the lemma follows. □

Finally consider the runtime of the algorithm. Each step in FindTranslations takes time $O(n)$. Therefore the total time spent on all calls to FindTranslations is $O(bn \log n)$. SelectPrime takes time $O(\log^{O(1)} s)$, therefore the total running time of the algorithm is $O(bn \log n + \log^{O(1)} s)$. This completes the proof of theorem 1. □

2.4 Real-Valued Inputs

Recall that the directed Hausdorff distance $h(X, Y)$ from a set X to a set Y is $\sup_{x \in X} \inf_{y \in Y} \rho(x, y)$ (where ρ is the underlying metric, in this case simply distance in one dimension). This "distance" is not symmetric but satisfies the triangle inequality.

We focus on the Threshold version of the problem. Given a parameter δ, we wish to identify all shifts which carry the probe to a position in which $h(P, D) < \delta$. Let

$$M^{\delta}_{D,P} = \{t \in \mathbb{R} : h(P + t, D) < \delta\}$$

on the line; and let

$$M^{\delta}_{D,P} = \{t \in \mathbb{R}/s : h(P + t, D) < \delta\}$$

on a circle of circumference s.

When not empty, $M^{\delta}_{D,P}$ is an infinite set, so the task of our algorithm is to find suitable representatives from this set. We use a secondary parameter ε, indicating the precision with which this task is performed. Specifically, we require that the Threshold algorithm solve the following:

Problem: Identify a set of "valid displacements," $\mu^{\delta,\varepsilon}_{D,P}$, satisfying the "$(\delta, \varepsilon)$ conditions":

1. $h(M^{\delta}_{D,P}, \mu^{\delta,\varepsilon}_{D,P}) < \varepsilon\delta$.
2. $\mu^{\delta,\varepsilon}_{D,P} \subseteq M^{\delta(1+\varepsilon)}_{D,P}$.

A solution to this problem identifies, for every shift achieving probe-data Hasudorff distance δ, a nearby shift of almost the same quality.

Theorem 2 *Given real input (P, D) and a parameter δ, we can identify a set of displacements $\mu^{\delta,\varepsilon}_{D,P}$, satisfying the (δ, ε) conditions with probability at least $1 - n^{-b}$, in time $O(bn\varepsilon^{-1} \log(n\varepsilon^{-1}) + \log^{O(1)} \frac{s}{\varepsilon\delta})$, where $s = s(P, D) = \max\{\rho(x, y) : (x, y) \in P^2 \cup D^2\} + 1$.*

Proof: We begin by discretizing the probe and the data as follows. Set $a = \frac{2\delta\varepsilon}{5d^{1/2}}$. (In the case of the circle, slight adjustment is necessary so that a divides the circumference.) For a point q let $\hat{q}$ be the point satisfying $\hat{q} = ma, m \in \mathbb{Z}$ where $q \in [\hat{q} - \frac{a}{2}, \hat{q} - \frac{a}{2})$, i.e. a nearest multiple of a. Discretize the probe by rounding each point p to $\hat{p}$. Let the discretized probe P' be the union of the discretizations of all the probe points. Let the discretized data D' be the union of the discretizations of all the data points, together with all integer multiples of a that, for some $d \in D$, are within distance $\delta + \frac{3\delta\varepsilon}{5}$ of $\hat{d}$.

The set of all valid displacements of this discretized version satisfy the (δ, ε) conditions. The data size has increased by a factor of $O(\varepsilon^{-1})$, and the theorem follows by reduction to theorem 1. □

Nearly Exact Matches: If δ, ε are chosen so that $(2 + 8\varepsilon/5)\delta < \min\{|x - y| : (x, y) \in P^2 \cup D^2\}$ then a match satisfying the (δ, ε) conditions will have the properties (a) every probe point has exactly one data point within distance δ, and (b) this mapping from probe points to data points is injective.

3 Translations in Dimension Two and Above

We now address the following problem. A probe P and data set D are given, both finite point sets in the space $V = \mathbb{Z}^d$ (for the integer, unbounded case) or $V = \mathbb{R}^d$ (real, unbounded case) or $V = \prod_1^d (\mathbb{Z}/s_i)$ (integer torus) or $V = \prod_1^d (\mathbb{R}/s_i)$ (real torus). The problem is to identify all "valid translations", i.e. elements $t \in V$ such that $P + t$ (Minkowski sum) is (or is near) a subset of D. In the threshold case, just as for one dimension, the (δ, ε) conditions are required, with Euclidean distance as the underlying metric. Thanks to the reduction noted earlier we may assume the problem is unbounded, at a possible cost of a factor of 2^d in the size of the data set.

For the Threshold version of the problem, with parameters δ and ε, discretize following the same description as for the one dimensional case: now each coordinate is rounded to the nearest multiple of a, and discretized data points are replicated throughout a metric ball. Proceed as for Integer Exact Match. For the Nearly Exact version of the problem, set $\delta = \min \rho(x, y)$ for $(x, y) \in D^2 \cup P^2$. (δ can be computed in time $O(n \log n)$, see [6, 7, 25].) Continue as for Threshold. In all cases call the new probe and data P_1, D_1. It remains only to describe the procedure for Integer Exact Match.

Choose a random vector $u \in \mathbb{R}^d$ from the spherically symmetric Gaussian distribution with total variance 1 (each coordinate has variance $d^{-1/2}$). Form a new probe P_2 in one dimension by mapping each point x of P_1 to the number

$u \cdot x$ (dot product in $\mathbb{R}^d$); form a one dimensional data set D_2 similarly.

As before, fix a point $p_0 \in P_1$ and define N to be the set of vectors $t \in \mathbb{R}^d$ such that $p_0+t \in D_1$ but $P_1+t \not\subseteq D_1$. For each $t \in N$ fix a point $A(t) \in P_1$ such that $A(t) + t \notin D_1$.

The probability that there exist $t \in N$ and $y \in D_1$ such that $|(u \cdot (A(t) + t)) - (u \cdot y)| < \frac{1}{(2\pi d)^{1/2} n^{b+2}}$ is at most n^{-b}. If this happens, this run of the algorithm is considered to fail.

Discretize the line at a scale of $\frac{1}{5(2\pi d)^{1/2} n^{b+2}}$. Form a new probe P_3 by rounding each probe point in P_2 to the nearest "integer"; form a new data set D_3 by repeating each data point in D_2 at the four nearest "integers". Now solve the Exact Match problem for P_3 and D_3.

Any Exact Match of P_1 and D_1 is carried to an Exact Match of P_3 and D_3. On the other hand if the failure event described above has not occurred, then every such match of (P_3, D_3) will pull back to a match (of the desired type) of (P, D). Thus we have:

Theorem 3 *Let pattern P and data D be given in $\mathbb{Z}^d$ or $\mathbb{R}^d$. Let $s = s(P, D) = \max\{\rho(x, y) : (x, y) \in P^2 \cup D^2\} + 1$. In the Threshold case let δ be the threshold; in the Nearly Exact case let $\delta = \min\{\rho(x, y) : (x, y) \in P^2 \cup D^2\}$. Choose any $b > 0$. Our algorithm runs in time $O(bn \log n + \log^{O(1)} s)$ in the Integer Exact case, or $O(b\varepsilon^{-O(d)} n \log(n\varepsilon^{-1}) + \log^{O(1)} \frac{s}{\varepsilon\delta})$ in the Threshold or Nearly Exact cases. With probability at least $1 - n^{-b}$ it produces, in the Integer Exact case, the correct list of valid translations; and in the remaining cases, a set of translations satisfying the (δ, ε) conditions.* □

4 Point Failures

Given two sets D and P of integers, and a number $t \in \mathbb{Z}$, let the number of point failures of a translation by t be $f(t) = |\{x \in P : x + t \notin D\}|$. We want to find the set $M_{D,P,r} = \{t \in \mathbb{Z} : f(t) \leq r\}$ of translations that match all but at most r elements of P to elements in D, or "r-matches". Our method extends also to solve this problem for a given directed Hausdorff distance δ, and for translations in higher dimensions. We first address the case of integer P and D in one dimension. Let $a \in (0, 1)$ be any constant.

Theorem 4 *Let integer pattern P and integer data D be given. For any $r \leq ak$ our algorithm runs in time $O(bn(r + 1)\log^2 n + \log^{O(1)} s)$ and with probability at least $1 - n^b$ produces the correct list $M_{D,P,r}$.*

Proof: The basic form of the algorithm will be the same as for 0-matches. First note that the mutual line-circle reductions preserve r-matches; hence we assume a line input. Constants c, v and T will be specified below.

PartialTranslations(P, D, r)

1. If n is less than some threshold run the naive algorithm, otherwise continue.
2. $u :=$ SelectPrime$(2s)$.
3. $(P_1, D_1) := C(P, D, u)$.
4. Repeat for $i \in \{1, \ldots, v \log n\}$

   $\mu_i :=$ FindPartialTrans$(P_1, D_1, c(r + 1)n, r)$.
5. Set $c_t := |\{i \in \{1, \ldots, v \log n\} : t \in \mu_i\}|$.
6. Return the pullbacks μ of $\{t \in \cup_i \mu_i : c_t \geq T\}$.

FindPartialTrans(P, D, h, r)

1. $(P_1, D_1) :=$ RandomMultiply(P, D).
2. $(P_2, D_2) := L(P_1, D_1)$.
3. $(P_3, D_3) :=$ ReduceSpace(P_2, D_2, h).
4. $\nu :=$ FindPartialMatches(P_3, D_3, r).
5. Pick p_0 uniformly at random from P, and return the pullback of ν to (P, D) with respect to p_0. (Translations for which p_0 is not matched are omitted.)

The only new procedure is FindPartialMatches(P, D, r), which we describe below.

First we observe how r-matches are transformed under ReduceSpace. For any $p \in P$, the pullback with respect to p of an r-match τ for $(H_h(D), H_h(P))$ is defined to be the set of displacements $H^{-1}_{h,D}(H_h(p) + \tau) - p$. Further define $H^{-1}_h(M_{H_h(D),H_h(P),r})(p)$ to be the union of $H^{-1}_{h,D}(H_h(p)+ \tau) - p$ over all r-matches τ in $M_{H_h(D),H_h(P),r}$. We now have:

Proposition 2 *If $t \in M_{D,P,r}$ and $p \in P$ is such that $p+t \in D$ then $t \in H^{-1}_h(M_{H_h(D),H_h(P),r})(p)$.*

4.1 Find Partial Matches

FindPartialMatches(P, D, r) takes three arguments: a probe P and a data set D (on a circle of size h), and an integer r. Note that t is an r-match if and only if $\overline{P + t} \cdot \overline{D} \geq |P| - r$ (dot product over $\mathbb{Z}$). The list of all these dot products is the convolution of $\overline{D}$ with the function whose value at x is $\overline{P}(-x)$; using the fast Fourier transform (over $\mathbb{Z}/h$), FindPartialMatches computes this in time $O(h \log h)$. (Recall that we have assumed a word length proportional to $\log s \geq \log n$. This precision in the computation of the transform suffices so that rounding errors do not affect the computation of any value $\overline{P+t\cdot D}$ by more than $1/4$, hence after a final rounding to the nearest integer, the list of dot products is correct.)

4.2 Overall Analysis

We begin with an upper bound on the probability that FindPartialTrans outputs a fixed invalid r-match t for D and P (the inputs to FindPartialTrans). For the remainder of this section let $v = (b+3)(\frac{4}{3}\log\frac{4}{3}-\frac{1}{3})^{-1}\frac{2}{1-a}$, let $T = \frac{2}{3}(1-a)v$, and let $c = \frac{17}{1-a}$.

Lemma 4 *If t is an invalid r-match for D and P, the probability that $t \in \mu_i$ is at most $\frac{8}{c}\frac{u}{u-1}$.*

Proof: Let $S_t = \{p \in P : p + t \notin D\}$: then $|S_t| = r + l$ for some $l > 0$. Now let $\tilde{S}_t$ be a subset of $|S_t|$ of size $r+1$. FindPartialTrans will output t if there is an element p of $\tilde{S}_t$ and an element $d \in D$ such that

(i) $\text{Rep}_u(R_{u,q}(d)) \equiv \text{Rep}_u(R_{u,q}(p)) + \text{Rep}_u(R_{u,q}(t)) \pmod h$, or

(ii) $\text{Rep}_u(R_{u,q}(d)) + u \equiv \text{Rep}_u(R_{u,q}(p)) + \text{Rep}_u(R_{u,q}(t)) \pmod h$.

The probability of this event is at most $(r+1)n\frac{8}{h}\frac{u}{u-1}$. Since $h \geq c(r+1)n$,

Lemma 5 *If $t \in M_{D,P,r}$ the probability that FindPartialTrans outputs t is at least $\frac{k-r}{k} = 1-a$.*

Proof: If t is not output by FindPartialTrans then p_0 is such that $p_0 + t \notin D$. Since there are at most $r \leq ak$ such elements the claim follows. □

Applying a Chernoff bound we obtain that $Pr\{c_t \geq T\log n\} \leq e^{(\frac{1}{3}-\frac{4}{3}\log\frac{4}{3})\frac{(1-a)}{2}v\log n}$ if t is an invalid r-match, and $Pr\{c_t \leq T\log n\} \leq e^{-\frac{(1-a)}{18}v\log n}$ if $t \in M_{D,P,r}$. Thus the probability of $t \in M_{D,P,r}$ not being reported is at most $\frac{1}{n^{b+3}}$ and the probability of $t \notin M_{D,P,r}$ being reported is at most $\frac{1}{n^{b+3}}$. Since the size of $M_{D,P,r}$ is at most n^2 and the set of invalid r-matches that can potentially be reported is also at most n^2, we obtain that the probability of failure is at most $\frac{1}{n^b}$.

Computing u can be done in time $O(\log^{O(1)} s)$ as before. FindPartialTrans takes $O((r+1)n\log((r+1)n))$ time, and is called $O(b\log n)$ times. The c_t's can be computed in $O(bn\log n)$ time. Therefore (and since $r < n$) the total running time is $O(b(r+1)n\log^2 n + \log^{O(1)} s)$.

Note: If we let $r = 0$, then we obtain an $O(bn\log^2 n + \log^{O(1)} s)$ algorithm for the case of no point failures. Whether it is preferable to use the asymptotically slightly faster result stated earlier, will depend on the efficiency with which Indyk's procedure can be implemented.

5 Approximate Best Match

We have focussed on the Threshold formulation of the Real problem. Much of the previous literature is stated in terms of the *optimization* version of the problem: given P and D, find the smallest δ such that P can be carried to within a directed Hausdorff distance δ of D. Depending on whether the answer is required to be exact, this is either the Best Match or Approximate Best Match formulation. For reasons given earlier, having to do with finite precision in real-valued data, we believe that the approximate formulation is more natural (at least, if the approximation factor can be made $1+\varepsilon\ \forall\varepsilon > 0$).

Our algorithm can be used to solve this version. First perform a "doubling search" on $\lg\delta$: run the Threshold matching algorithm starting with $\delta = s/2^{2^0}$, then $\delta = s/2^{2^1}$, and in general $\delta = s/2^{2^i}$ until an i_0 is reached for which no matches are found. Then perform a regular binary search on $\lg(s/\delta)$ between 2^{i_0-1} and 2^{i_0}. Then run the algorithm another few times, decreasing the "precision parameter" ε, until the best match value δ has been identified to within the desired approximation factor $1+\varepsilon$. The runtime of this procedure depends on the actual value of the best match δ, and is $(\log\log(s/\delta) + \log\varepsilon^{-1})$ times the runtime of the Threshold algorithm with precision ε.

6 Rigid Motions

We sketch how our method applies to solving the Rigid Motions problem in any dimension d. (As will be apparent from the method, in dimension $d > 2$ Scaling can also be allowed without substantial adverse effect on the analysis.) We solve the Threshold, Nearly Exact and Approximate Best Match formulations; the latter two are reduced to the first, just as before, by either judicious choice of, or binary search for, an appropriate threshold.

Notation: Given a point x and an affine subspace A, let $[x \to A]$ denote the shortest vector from x to a point in A. Given a set of points $\{p_1, ..., p_j\}$, let $A_{\{p_1,\ldots,p_j\}}$ denote the affine subspace they span (by which we mean the lowest dimension affine subspace containing all the points).

Let $f : \mathbb{N} \to \mathbb{R}^+$ and let $||r||$ be the Euclidean norm of vector r. Define a "frame" for the probe, relative to the function f, to be a set $F = \{q_1, ..., q_d\} \subseteq P$ which provides a numerically stable way of "gripping" the probe, in the following sense: if the positions of each of $\{q_1, ..., q_j\}$ $(1 \leq j \leq d)$ are perturbed by a small amount α, forming a new set of points $\{p_1, ..., p_j\}$ s.t. $\rho(p_i, q_i) \leq \alpha\ \forall 1 \leq i \leq j$, then for any point of the probe x, $||[x \to A_{\{p_1,\ldots,p_j\}}] - [x \to A_{\{q_1,\ldots,q_j\}}]|| \leq \alpha f(j)$. The first $d-1$ points of F will be referred to as its $(d-1)$-subframe F'.

We will shortly outline an argument showing that for $f(j) = (3^j - 1)/2$, frames do exist; and a frame can be quickly found.

First however we show how a frame F allows us to solve the rigid motions problem in $\mathbb{R}^d$. Discretize the data by introducing a scalar grid of length scale $O(\varepsilon/(f(d)d^{1/2}))$, and duplicating each data point at all points of the grid within distance $\delta(1+O(\varepsilon))$. Then range over all n^{d-1} ways in which the points of the $(d-1)$-subframe F' can be assigned to data points. If, for such an assigment, there is a rigid motion carrying the subframe to within directed Hausdorff distance $\varepsilon/f(d)$ of the chosen data points, then, for any such rigid motion, any remaining point of the probe is restricted to lie in a "solid ring" in space (annulus, in the plane) whose cross-section has radius $O(\varepsilon)$. Upon finding that a satisfactory rigid motion exists for a particular assignment, choose one such rigid motion; then each remaining point of the probe is constrained to lie on a circle. Form a one-dimensional Integer problem on that circle by discretizing it at a scale of $O(\varepsilon)$, and placing a data point at each "integer" lying within distance $O(\varepsilon)$ of any of the duplicated data points in $\mathbb{R}^d$. Solve all the circle problems and intersect their solutions (allowing for rounding because of the slightly differing discretizations on each circle).

The total number of probe points in all the one dimensional Integer Exact problems is the same as the original probe (minus the frame points); the total number of data points in all these problems has increased by a factor that depends only on the dimension. Since the runtime of the one-dimensional integer exact problem (as given in theorem 1, and noting that the parameters b and s are fixed in this discussion) is convex, the total time required to solve all instances at hand is no more than it would be solve one problem containing all the probe and data points, hence $O(n\log n)$. Combining the solutions requires only intersecting all candidate translations (allowing for rounding due to the differing discretizations of the various circles), and can be done in time $O(n\log n)$. Hence the total runtime for the algorithm is $O(n^d\log n)$.

Quality of the solution: We phrase the (δ,ε) conditions for the rigid motion problem by over-representing any particular rigid motion as a point in $(\mathbb{R}^d)^k$, the i'th "$\mathbb{R}^d$-coordinate" being the image of the i'th probe point. The set of rigid motions satisfying the δ threshold condition is

$$M^{\delta}_{D,P} \;=\; \{X\in(\mathbb{R}^d)^k : h(X,D)<\delta \text{ and X is the image of } P \text{ under a rigid motion}\}.$$

We require that the threshold algorithm solve:

Problem: Obtain a set $\mu^{\delta,\varepsilon}_{D,P}$ of "valid rigid motions" of the probe, satisfying the following "(δ,ε) conditions":

1. Every point in $\mu^{\delta,\varepsilon}_{D,P}$ is the image of P under a rigid motion.
2. $h(M^{\delta}_{D,P},\mu^{\delta,\varepsilon}_{D,P})<\varepsilon\delta$.
3. $\mu^{\delta,\varepsilon}_{D,P}\subseteq M^{\delta(1+\varepsilon)}_{D,P}$.

To summarize these results (and including the effect of the secondary parameters on the runtime), we have:

Theorem 5 *Let pattern P and data D be given in $\mathbb{R}^d$, along with a threshold δ, a precision parameter ε, and a reliability parameter b. Let s be the space size of the problem. Then in time $O(bn^d\varepsilon^{-O(d)}\log(n\varepsilon^{-1})+\log^{O(1)}\frac{s}{\varepsilon\delta})$ the above algorithm identifies a set of matches $\mu^{\delta,\varepsilon}_{D,P}$, which with probability at least $1-n^{-b}$ satisfies the (δ,ε) conditions above.*

Frame Selection: In any fixed dimension d a frame can be selected in time $O(k)$ as follows: begin by selecting any point. Next, choose the point furthest from the first; in general, choose the point furthest from the affine subspace spanned by the existing points. (This is facilitated by applying a Gram-Schmidt process each time a new point is adjoined, thus obtaining an orthogonal basis for the subspace. The square of the distance of each point from the subspace can be updated by subtracting the square of the projection on the newest dimension.)

Lemma 6 *This process yields a frame with $f(j)=(3^j-1)/2$.*

Proof: By induction on j. The case $j=1$ is immediate. Let $j>1$; let $\{q_1,\ldots,q_j\}$ be the first j points selected by the process; and let $\{p_1,\ldots,p_j\}$ be points s.t. $\rho(p_i,q_i)<\alpha\ \forall i$. Let x be any probe point. Let $w=[x\to A_{\{q_1,\ldots,q_{j-1}\}}]$, let $w'=[x\to A_{\{p_1,\ldots,p_{j-1}\}}]$, let $v=[q_j\to A_{\{q_1,\ldots,q_{j-1}\}}]$ and let $v'=[p_j\to A_{\{p_1,\ldots,p_{j-1}\}}]$. By construction, $||w||\le ||v||$.

Given two vectors y,z, let z_y denote the projection of y on z, namely $z_y=\frac{y\cdot z}{z\cdot z}z$.

Now $[x\to A_{\{p_1,\ldots,p_j\}}]-[x\to A_{\{q_1,\ldots,q_j\}}]=(w'-v'_{w'})-(w-v_w)=(w'-w)+(v_w-v'_{w'})$. The inductive hypothesis implies that $||w'-w||\le\alpha f(j-1)$, while, as we next show, the inductive hypothesis and the fact that $||w||\le||v||$ together imply that $||v_w-v'_{w'}||\le\alpha(2f(j-1)+1)$; the bound $||[x\to A_{\{p_1,\ldots,p_j\}}]-[x\to A_{\{q_1,\ldots,q_j\}}]||\le\alpha(3f(j-1)+1)$ follows and implies the lemma.

Lemma 7 $||z_y-z_{y+r}||\le||r||$.

Lemma 8 $||z_y-(z+r)_y||\le\frac{||r||\cdot||y||}{||z||}$.

Lemma 7 is immediate.

Proof of lemma 8: Fixing y and letting w vary, the vectors w_y lie on the sphere for which the segment $\overline{0y}$ is a diameter. With this in mind, the case $||r||\ge||z||$ is immediate. If $||r||<||z||$, set β to be the angle between z and $z+r$; $\beta\le\arcsin(||r||/||z||)$. The central angle in the above sphere between the points z_y and $(z+r)_y$, i.e. the angle between the vectors $z_y - y/2$ and $(z+r)_y - y/2$, is 2β. The distance between z_y and $(z+r)_y$ is therefore bounded by $||y|| \sin\beta$ and hence by $||y|| \cdot ||r||/||z||$. □

To continue with the proof of lemma 6, write $v_w - v'_{w'} = (v_w - v'_w) + (v'_w - v'_{w'})$. By lemma 7 $||v'_w - v'_{w'}|| \leq ||w - w'|| \leq ||\alpha f(j-1)||$. By lemma 8 $||v_w - v'_w|| \leq ||v' - v|| \cdot ||w||/||v|| \leq ||v' - v||$. Write $v' - v = ([p_j \rightarrow A_{\{p_1,\ldots,p_{j-1}\}}] - [q_j \rightarrow A_{\{p_1,\ldots,p_{j-1}\}}]) + ([q_j \rightarrow A_{\{p_1,\ldots,p_{j-1}\}}] - [q_j \rightarrow A_{\{q_1,\ldots,q_{j-1}\}}])$. The fact that $||p_j - q_j|| \leq \alpha$ immediately implies the same of the first term; the inductive hypothesis implies that the second term is of norm at most $\alpha f(j-1)$. In combination, then, $||v_w - v'_{w'}|| \leq \alpha(2f(j-1)+1)$. □

7 Parallelizability

It is easy to check that the algorithms can, for any fixed dimension, be parallelized to run in polylogarithmic time with a polylogarithmic work overhead.

8 Acknowledgments

Thanks to Amihood Amir, Piotr Indyk and Suresh Venkatasubramanian for helpful communications.

References

[1] T. Akutsu. Protein structure alignment using dynamic programming and iterative improvement. *IEICE Trans. on Information and Systems*, Vol. E78-D No. 0:1-8, 1996.

[2] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3:237-256, 1988.

[3] M. J. Atallah. Checking similarity of planar figures. *Internat. J. Comput. Inform. Science* 13:279-290, 1984.

[4] M. D. Atkinson. An optimal algorithm for geometrical congruence. *J. Algorithms*, 8:159-172, 1987

[5] E. Bach, J. Shallit. *Algorithmic Number Theory, Volume 1: Efficient Algorithms*. MIT Press, 1996.

[6] J. L. Bentley and M. I. Shamos. Divide-and-Conquer in Multidimensional Space. *Proc. 8'th Symp. on Theory of Comput.,* 220-230, 1976.

[7] J. L. Bentley. Multidimensional divide and conquer. *Comm. ACM* 23(4):214-229, 1980.

[8] R. S. Boyer and J. S. Moore. A fast string-searching algorithm. *Comm. ACM*, 20(10):62-72, 1977.

[9] L. P. Chew, M. Goodrich, D. Huttenlocher, K. Kedem, J. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. *Proc. 5'th Canadian Conf. Comp. Geom.*, 151-156, 1993.

[10] P. J. de Rezende, D. T. Lee. Point Set Pattern Matching in d-Dimensions. *Algorithmica* 13:387-404, 1995.

[11] P. W. Finn, L. E. Kavraki, J-C. Latombe, R. Motwani, C. Shelton, S. Venktasubramanian, A. Yao. RAPID: Randomized Pharmacophore Identification for Drug Design. *Proc. 13'th ACM Symp. Comp. Geom.*, 324-333, 1997.

[12] M. J. Fischer and M. S. Paterson, String Matching and Other Products, pages 113-125. In R. M. Karp, editor, *Complexity of Computation*. SIAM-AMS 1974.

[13] M. T. Goodrich, J. S. B. Mitchell, and M. W. Orletsky. Practical methods for approximate geometric pattern matching under rigid motions. *Proc. 10'th ACM Symp. Comp. Geom.*, 103-112, 1994.

[14] J. E. Hopcroft and D. P. Huttenlocher. Affine invariants for model-based recognition, in *Geometric Invariance on Computer Vision*, J. Mundy and A. Zisserman, eds., 354-374, 1992.

[15] D. P. Huttenlocher and K. Kedem. Efficiently computing the Hausdorff distance for point sets under translation. *Proc. 6'th ACM Symp. Comp. Geom.*, 340-349, 1990.

[16] D. P. Huttenlocher, K. Kedem, and J. M. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. *Proc. 8'th ACM Symp. Comp. Geom.*, 110-120, 1992.

[17] D. P. Huttenlocher, K. Kedem and M. Sharir. The upper envelope of Voronoi Surfaces and its Applications. *Proc. 7'th ACM Symp. Comp. Geom.*, 194-203, 1991.

[18] P. Indyk, Faster algorithms for string matching problems: matching the convolution bound. *These Proceedings.*

[19] P. Indyk, R. Motwani and S. Venkatasubramanian, Geometric Matching Under Noise: Combinatorial Bounds and Algorithms. To appear in Proc. 10'th SIAM-ACM Symp. Discr. Alg., 1999.

[20] S. Irani and P. Raghavan. Combinatorial and Experimental Results for Randomized Point Matching Algorithms. *Proc. 12'th ACM Symp. Comp. Geom.*, 68-77, 1996.

[21] R. M. Karp and M. O. Rabin. Efficient Randomized Pattern-Matching Algorithms. *Technical Report TR-31-81*, Aiken Computation Laboratory, Harvard University, 1981.

[22] D. E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM J. Comp.*, 6(2):323-350, 1977.

[23] J. H. Morris and V. R. Pratt. A linear pattern matching algorithm. *Technical Report No. 40*, Computing Center, University of California, Berkeley, 1970.

[24] D. M. Mount, N. S. Netanyahu, J. Le Moigne. Improved Algorithms for Robust Point Pattern Matching and Applications to Image Registration. *Proc. 14'th ACM Symp. Comp. Geom.*, 1998.

[25] F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer Verlag, 1985.

[26] W. Rucklidge. *Efficient Visual Recognition Using the Hausdorff Distance*. Springer Verlag, 1996.

[27] F. F. Yao, Computational Geometry. In J. van Leuwen, ed., *Handbook of Theoretical Computer Science, vol. A: Algorithms and Complexity*, 343-390, 1991.