On the ICP Algorithm*

Esther Ezra
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
estere@post.tau.ac.il

Micha Sharir
School of Computer Science
Tel Aviv University
Tel Aviv 69978, Israel
and
Courant Institute of
Mathematical Science
New York University
New York, NY 10012, USA
michas@post.tau.ac.il

Alon Efrat
School of Computer Science
University of Arizona
Tucson, Arizona 85721, USA
alon@CS.Arizona.EDU

ABSTRACT

We present upper and lower bounds for the number of iterations performed by the Iterative Closest Point (ICP) algorithm. This algorithm has been proposed by Besl and McKay [4] as a successful heuristics for pattern matching under translation, where the input consists of two point sets in d-space, for $d \geq 1$, but so far it seems not to have been rigorously analyzed. We consider two standard measures of resemblance that the algorithm attempts to optimize: The RMS (root mean squared distance) and the (one-directional) Hausdorff distance. We show that in both cases the number of iterations performed by the algorithm is polynomial in the number of input points. In particular, this bound is quadratic in the one-dimensional problem, for which we present a lower bound construction of $\Omega(n \log n)$ iterations under the RMS measure, where n is the overall size of the input. Under the Hausdorff measure, this bound is only $O(n)$ for input point sets whose *spread* is polynomial in n, and this is tight in the worst case.

We also present several structural geometric properties of the algorithm under both measures. For the RMS measure, we show that at each iteration of the algorithm the cost function monotonically and strictly decreases along the vector Δt of the *relative translation*. As a result, we conclude that the polygonal path π, obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself. In particular, in the one-dimensional problem all the relative translations of the ICP algorithm are in the same (left or right) direction. For the Hausdorff measure, some of these properties continue to hold (such as monotonicity in one dimension), whereas others do not.

*Work on this paper by the first two authors has been supported by NSF Grants CCR-00-98246 and CCF-05-14079, by a grant from the U.S.-Israeli Binational Science Foundation, work by the second author was also supported by Grant 155/05 from the Israel Science Fund, and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. Work on this paper by the last author has been partially supported by an NSF CAREER award (CCR-03-48000) and an ITR/Collaborative Research grant (03-12443).

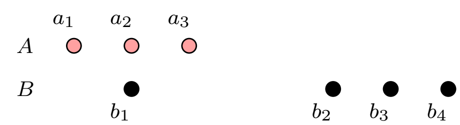

Figure 1: A local minimum in $\mathbb{R}^1$ of the ICP measures. The global minimum is attained when a_1, a_2, a_3 are aligned on top of b_2, b_3, b_4, respectively.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: ANALYSIS OF ALGORITHMS AND PROBLEM COMPLEXITY Nonnumerical Algorithms and Problems [computations on discrete structures, geometrical problems and computations]

General Terms

Algorithms, Theory

Keywords

ICP, RMS, Hausdorff distance, Pattern matching, Voronoi diagram

1. INTRODUCTION

The matching and analysis of geometric patterns and shapes is an important problem that arises in various application areas, in particular in computer vision and pattern recognition [3]. In a typical scenario, we are given two objects A and B, and we wish to determine how much they *resemble* each other. Usually one of the objects may undergo certain transformations, like translation, rotation and/or scaling, in order to be matched with the other object as well as possible. In many cases, the objects are represented as finite sets of (sampled) points in two or three dimensions (they are then referred to as "point patterns" or "shapes"). In order to measure "resemblance", various *cost functions* have been used. Two prominent ones among them are the (one-directional) *Hausdorff distance* [3], and the *sum of squared distances* or *root mean square* [4, 6]. Under the first measure, the cost function is $\Phi_\infty(A,B) = \max_{a\in A} \|a - N_B(a)\|$, and under the second measure, it is $\Phi_2(A,B) = \frac{1}{m}\sum_{a\in A} \|a - N_B(a)\|^2$, where $\|\cdot\|$ denotes the Euclidean norm[1], $N_B(a)$ denotes the nearest neighbor of a in B, and $m = |A|$. In what follows, we also use the (slightly abused) notation

$$RMS(t) := \frac{1}{m}\sum_{a\in A} \|a + t - N_B(a+t)\|^2. \qquad (1)$$

A heuristic matching algorithm that is widely used, due to its simplicity (and its good performance in practice), is the *Iterative Closest Point* algorithm, or the *ICP* algorithm for short, of Besl and McKay [4]. Given two point sets A and B in $\mathbb{R}^d$ (also referred to as the *data shape* and the *model shape*, respectively), we wish to minimize a cost function[2] $\phi(A+t,B)$, over all *translations* t of A relative to B. The algorithm starts with an arbitrary translation that aligns A to B (suboptimally), and then repeatedly performs local improvements that keep re-aligning A to B, while decreasing the given cost function $\phi(A+t,B)$, until a convergence is reached[3]. This is done as follows.

At the i-th iteration of the ICP algorithm, the set A has already been translated by some vector t_{i-1}, where $t_0 = \overrightarrow{0}$. We then apply the following two steps:

(i) We assign each (translated) point $a + t_{i-1} \in A + t_{i-1}$ to its nearest neighbor $b = N_B(a + t_{i-1}) \in B$ under the Euclidean distance. (ii) We then compute the new relative translation Δt_i that minimizes the cost function ϕ (with respect to the above fixed assignment). Specifically, under the one-directional Hausdorff distance, we find the Δt_i that minimizes

$$\phi_\infty(A + t_{i-1}, \Delta t_i, B) = \max_{a\in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|,$$

and under the sum of squared distances, we minimize

$$\phi_2(A + t_{i-1}, \Delta_{t_i}, B) = \frac{1}{m}\sum_{a\in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|^2.$$

We then align the points of A to B by translating them by Δt_i, so the new (overall) translation is $t_i = t_{i-1} + \Delta t_i$.

The ICP algorithm performs these two steps repeatedly and stops when the value of the cost function does not decrease with respect to the previous step (as a matter of fact, the ICP algorithm in its original presentation stops when the difference in the cost function falls below a given threshold $\tau > 0$; however, in our analysis, we assume that $\tau = 0$). It is shown by Besl and McKay [4] that, when $\phi(\cdot,\cdot)$ measures the sum of squared distances, this algorithm always converges monotonically to a local minimum, and that the value of the cost function decreases at each iteration[4]. An easy variant of their proof (noted below) establishes convergence also when the cost function measures the (one-directional) Hausdorff distance.

In other words, in stage (i) of each iteration of the ICP algorithm we assign the points in (the current translated copy of) A to their respective nearest neighbors in B, and in stage (ii) we translate the points of A in order to minimize the value of the cost function with respect to the assignment computed in stage (i). This in turn may cause some of the points in the new translated copy of A to acquire new nearest neighbors in B, which causes the algorithm to perform further iterations. If no point of A changes its nearest neighbor in B, the value of the cost function does not change in the next iteration (in fact, the next relative translation equals $\overrightarrow{0}$) and, as a consequence, the algorithm terminates. Note that the pattern matching performed by the algorithm is *one-directional*, that is, it aims to find a translation of A that places the points of A near points of B, but not necessarily the other way around.

Since the value of the cost function is strictly reduced at each iteration of the algorithm, it follows that no nearest-neighbor assignment arises more than once during the course of the algorithm, and thus it is sufficient to bound the overall number of nearest-neighbor assignments (or, NNA's, for short) that the algorithm reaches in order to bound the number of its iterations.

The convergence of the algorithm to either local or global minimum heavily relies on the initial position of the input points (see [4] for details and for a heuristics that "helps" the algorithm to converge in practice to the global minimum). There are simple constructions, such as the one depicted in Figure 1, that show that the algorithm may terminate at a local minimum that is quite different (and far) from the global one, under either of the resemblance measures that we use. (Nevertheless, as reported by many practical experimentations, the convergence to the (possibly local) minimum is rather fast in practice [4, 5, 8, 9].) Still, this is a disadvantage of the algorithm from a theoretical "worst-case" point of view, and the potential convergence to a local minimum raises several interesting questions. The most obvious question is to obtain sharp upper and lower bounds on the maximum possible number of local minima that the function can attain. Another is to analyze the decomposition of space into "influence regions" of the local minima, where each such region consists of all the translations from which the algorithm converges to a fixed local minimum.

Our results. In the next section we first note a (probably weak) upper bound of $O\left(m^d n^d\right)$ on the number of iterations of the algorithm in $\mathbb{R}^d$ under any of the two measures, for any $d \geq 1$. We then present several structural geometric properties of the algorithm under the RMS measure. Specifically, we show that at each iteration of the algorithm the (real) cost function monotonically and strictly decreases, in a continuous manner, along the vector Δt of the relative translation[5]. As a result, we conclude that the polygonal

[1] Of course, other distances can also be considered, but this paper treats only the Euclidean case.

[2] The only cost function used in the original version of the ICP algorithm is the sum of squared distances (see [4, 5, 8, 9, 11]). In this paper we also consider the (one-directional) Hausdorff distance cost function, as defined above.

[3] In the original version of the algorithm [4], the points of A can also be rotated in order to be matched with the points of B, though in this paper we analyze it only under translations.

[4] We definitely decrease it with respect to the present nearest-neighbor assignment, and the revised nearest-neighbor assignment at the new placement can only decrease it further.

[5] This is a much stronger property than the originally noted one, that the value at the end of the translation is smaller

path π obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself. In particular, for $d = 1$, the ICP algorithm is *monotone* — all its translations are in the same (left or right) direction. Next, in Section 3 we present a lower bound construction of $\Omega(n \log n)$ iterations for the one-dimensional problem under the RMS measure (assuming $m \approx n$). The upper bound is quadratic, and closing the substantial gap between the bounds remains a major open problem. In Section 4 we discuss the problem under the (one-directional) Hausdorff distance measure. In particular, we present for the one-dimensional problem the upper bound $O\left((m+n)\log \delta_B / \log n\right)$ on the number of iterations of the algorithm, where δ_B is the *spread* of the input point set B (i.e., the ratio between the diameter of the set and the distance between its closest pair of points). We then present a tight lower bound construction with $\Theta(n)$ moves, for the case where the spread of B is polynomial in n. We also study the problem under the Hausdorff measure in two and higher dimensions, and show that some of the structural properties of the algorithm that hold for the RMS measure do not hold in this case. We give concluding remarks and present open problems in Section 5.

Why study the ICP algorithm. The pattern matching problem is a central and important problem that arises in many applications, ranging from surveillance to structural bioinformatics, and the ICP algorithm has been identified and used as a practical heuristic solution over the past fifteen years. Many experimental reports on its performance, including additional heuristic enhancements of it (e.g., in finding a good initial translation and using various techniques for sampling points from the input model) have been published [4, 5, 9, 11]. Still, to the best of our knowledge, this technique has never before been subject to a serious and rigorous analysis of its worst-case behavior, which it definitely deserves. Another motivation, which has unfolded as work on the paper progressed, is that the problem possesses a beautiful geometric structure, and has many surprising and subtle features.

The present work, though revealing many of these features, is only an initial step towards a fully comprehensive understanding of the algorithm. We hope that it will trigger additional work that will successfully tackle the remaining open problems.

2. GENERAL STRUCTURAL PROPERTIES OF THE ICP ALGORITHM UNDER THE RMS MEASURE

Let $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ be two point sets in d-space, for $d \geq 1$, and, as above, suppose that the ICP algorithm aligns A to B; that is, B is fixed and A is translated to best fit B.

Theorem 2.1. *The maximum possible overall number of nearest-neighbor assignments, over all translated copies of A, is $\Theta\left(m^d n^d\right)$.*

Sketch of proof: Let $\mathcal{V}(B)$ denote the *Voronoi diagram* of B, that is, the partition of $\mathbb{R}^d$ into d-dimensional cells $\mathcal{V}(b_i)$, for $i = 1, \ldots, n$, such that each point $p \in \mathcal{V}(b_i)$ satisfies $\|p - b_i\| \leq \|p - b_j\|$, for each $j \neq i$.

The global NNA changes at critical values of the translation t, in which the nearest-neighbor assignment of some point $a + t$ of the translated copy of A is changed; that is, a crosses into a new Voronoi cell of $\mathcal{V}(B)$. For each $a \in A$ (this denotes the *initial* location of this point) consider the shifted copy $\mathcal{V}(B) - a = \mathcal{V}(B - a)$ of $\mathcal{V}(B)$; i.e., the Voronoi diagram of $B - a = \{b - a \mid b \in B\}$. Then a critical event that involves the point a_i occurs when the translation t lies on the boundary of some Voronoi cell of $\mathcal{V}(B - a_i)$, for $i = 1, \ldots, m$. Hence we need to consider the *overlay* $M(A, B)$ of the m shifted diagrams $\mathcal{V}(B - a_1), \ldots, \mathcal{V}(B - a_m)$. Each cell of the overlay consists of translations with a common NNA, and the number of assignments is in fact equal to the number of cells in the overlay $M(A, B)$. A recent result of Koltun and Sharir [7] implies that the complexity of the overlay is $O(m^d n^d)$. It is straightforward to give constructions that show that this bound is tight in the worst case, for any $d \geq 1$. □

Corollary 2.2. *For any cost function that guarantees convergence (in the sense that the algorithm does not reach the same NNA more than once), the ICP algorithm terminates after $O(m^d n^d)$ iterations.*

Remark: A major open problem is to determine whether this bound is tight in the worst case. So far we have been unable to settle this question (under the RMS measure) even for $d = 1$; see below for details. In other words, while there can be many NNA's, we suspect that the ICP algorithm cannot step through many of them in a single execution.

We next present a simple but crucial property of the relative translations that the algorithm generates.

Lemma 2.3. *At each iteration $i \geq 2$ of the algorithm, the relative translation vector Δt_i satisfies*

$$\Delta t_i = \frac{1}{m} \sum_{a \in A} \Big(N_B(a + t_{i-1}) - N_B(a + t_{i-2}) \Big), \qquad (2)$$

where $t_j = \sum_{k=1}^{j} \Delta t_k$.

Proof: Follows using easy algebraic manipulations, based on the obvious equality that follows by construction

$$\Delta t_i = \frac{1}{m} \sum_{a \in A} \Big(N_B(a + t_{i-1}) - (a + t_{i-1}) \Big). \qquad (3)$$

(See [6, Lemma 5.2] for similar considerations.) □

Remark: The expression in (2) only involves differences between points of B. More precisely, the next relative translation is the average of the differences between the new B-nearest neighbor and the old B-nearest neighbor of each point of (the current and preceding translations of) A. This property does not hold for the *first* relative translation of the algorithm.

Theorem 2.4. *Let Δt be a move of the ICP algorithm from translation t_0 to $t_0 + \Delta t$. Then $RMS(t_0 + \xi \Delta t)$ is a strictly decreasing function of $\xi \in [0, 1]$.*

First Proof: We present two (related) proofs. In the first proof, put

$$RMS_0(\xi) := \frac{1}{m} \sum_{a \in A} \|a + t_0 + \xi \Delta t - N_B(a + t_0)\|^2.$$

than that at the beginning.

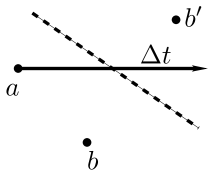

Figure 2: The new nearest neighbor lies ahead of the old one in the direction Δt.

Note that, by the definition of the ICP algorithm, the graph of $RMS_0(\xi)$ is a parabola that attains its minimum at $\xi = 1$. Hence, its derivative is negative for $\xi \in [0, 1)$. That is,

$$\frac{1}{2}RMS_0'(\xi) = \frac{1}{m}\sum_{a\in A}\Big(a + t_0 + \xi\Delta t - N_B(a + t_0)\Big) \cdot \Delta t < 0,$$

or

$$\frac{1}{2}RMS_0'(\xi) = \xi\|\Delta t\|^2 + \frac{1}{m}\sum_{a\in A}\Big(a + t_0 - N_B(a + t_0)\Big) \cdot \Delta t < 0,$$

where the last term is the sum of the inner products of the respective pairs of vectors. On the other hand, for any $\xi \in [0, 1]$, the function

$$RMS_1(\xi) := RMS(t_0 + \xi\Delta t) =$$

$$\frac{1}{m}\sum_{a\in A}\|a + t_0 + \xi\Delta t - N_B(a + t_0 + \xi\Delta t)\|^2$$

is the real RMS-distance from A to B at the translation $t_0 + \xi\Delta t$. Our goal is to show that $RMS_1'(\xi) < 0$, for any $\xi \in [0, 1]$, in which the function $RMS_1(\xi)$ is smooth (note that $RMS_1(\xi)$ is non-smooth exactly at points where some a changes its nearest neighbor in B). As above, we have, at points ξ where $RMS_1(\xi)$ is smooth,

$$\frac{1}{2}RMS_1'(\xi) = \xi\|\Delta t\|^2 + \frac{1}{m}\sum_{a\in A}\Big(a + t_0 - N_B(a + t_0 + \xi\Delta t)\Big) \cdot \Delta t.$$

It follows that

$$RMS_0'(\xi) - RMS_1'(\xi) =$$

$$\frac{2}{m}\sum_{a\in A}\Big(N_B(a + t_0 + \xi\Delta t) - N_B(a + t_0)\Big) \cdot \Delta t.$$

We claim that each of the terms in the latter sum is non-negative. Indeed, consider a fixed point a. When a changes its nearest neighbor from some b to another b', it has to cross the bisector of b and b' from the side of b to the side of b'. This is easily seen to imply that (see also Figure 2)

$$(b' - b) \cdot \Delta t \geq 0.$$

Adding up all these inequalities that arise at bisector crossings during the motion of a, we obtain the claimed inequality. Hence $RMS_0'(\xi) - RMS_1'(\xi)$ is non-negative throughout the motion, and since $RMS_0'(\xi)$ is negative, so must be $RMS_1'(\xi)$.

Second Proof: (This can be regarded as a geometric interpretation of the first proof.) The function

$$RMS(t) = \frac{1}{m}\sum_{a\in A}\|a + t - N_B(a + t)\|^2 =$$

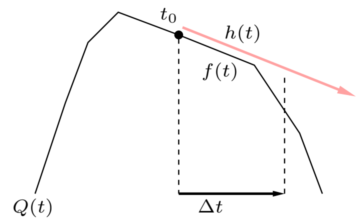

Figure 3: Illustrating the proof that $RMS(t_0 + \xi\Delta t)$ is a strictly decreasing function of $\xi \in [0, 1]$.

$$\frac{1}{m}\sum_{a\in A}\Big(\|t\|^2 + 2t \cdot (a - N_B(a + t)) + \|a - N_B(a + t)\|^2\Big)$$

is the average of m Voronoi surfaces $\mathcal{S}_{B-a}(t)$, whose respective minimization diagrams are $\mathcal{V}(B - a)$, for each $a \in A$. That is,

$$\mathcal{S}_{B-a}(t) = \min_{b\in B}\|a+t-b\|^2 = \min_{b\in B}\Big(\|t\|^2 + 2t\cdot(a-b) + \|a-b\|^2\Big),$$

for each $a \in A$. Subtracting the term $\|t\|^2$, we obtain that each resulting Voronoi surface $\mathcal{S}_{B-a}(t) - \|t\|^2$ is the lower envelope of n hyperplanes, and is thus the boundary of a concave polyhedron. Hence $Q(t) := RMS(t) - \|t\|^2$ is equal to the average of these concave polyhedral functions, and is thus itself the boundary of a concave polyhedron (see also the proof of Theorem 2.1).

Consider the NNA that corresponds to the translation t_0. It defines a facet $f(t)$ of $Q(t)$, which contains the point $(t_0, Q(t_0))$. We now replace $f(t)$ by the hyperplane $h(t)$ containing it, and note that $h(t)$ is tangent to the polyhedron $Q(t)$ at t_0; see Figure 3 for an illustration. The graph of $RMS_0(\xi)$, as defined above, is the image of the relative translation vector Δt on the paraboloid $\|t\|^2 + h(t)$. Since $Q(t) \leq h(t)$, for any $t \in \mathbb{R}^d$, the concavity of $Q(t)$ implies that for any $0 \leq \xi_1 < \xi_2 \leq 1$, $Q(t_0 + \xi_1\Delta t) - Q(t_0 + \xi_2\Delta t) \geq h(t_0 + \xi_1\Delta t) - h(t_0 + \xi_2\Delta t)$. Since $\|t\|^2 + h(t)$ is (strictly) monotone decreasing[6] along Δt, we obtain

$$RMS(t_0 + \xi_1\Delta t) - RMS(t_0 + \xi_2\Delta t) =$$

$$\|t_0 + \xi_1\Delta t\|^2 + Q(t_0 + \xi_1\Delta t) - \|t_0 + \xi_2\Delta t\|^2 - Q(t_0 + \xi_2\Delta t) \geq$$

$$\|t_0 + \xi_1\Delta t\|^2 - \|t_0 + \xi_2\Delta t\|^2 + h(t_0 + \xi_1\Delta t) - h(t_0 + \xi_2\Delta t) > 0,$$

which implies that $RMS(t_0 + \xi\Delta t)$ is a strictly decreasing function of $\xi \in [0, 1]$. $\square$

Let π be the connected polygonal path obtained by concatenating the ICP relative translations Δt_j. That is, π starts at the origin and its j-th edge is the vector Δt_j. Theorem 2.4 implies:

Theorem 2.5. *The ICP path π does not intersect itself.*

In particular, Theorem 2.5 implies that, on the line, the points of A are always translated in the same direction at each iteration of the algorithm. We thus obtain:

Corollary 2.6 (Monotonicity). *In the one dimensional case, the ICP algorithm moves the points of A always in the same (left or right) direction. That is, either $\Delta t_i \geq 0$ for each $i \geq 0$, or $\Delta t_i \leq 0$ for each $i \geq 0$.*

[6] By definition, Δt moves from t_0 to the minimum of the fixed paraboloid $\|t\|^2 + h(t)$, whence the claim.

COROLLARY 2.7. *In any dimension $d \geq 1$, the angle between any two consecutive edges of π is obtuse.*

Proof: Consider two consecutive edges Δt_k, Δt_{k+1} of π. Using Lemma 2.3 we have $\Delta t_{k+1} = \frac{1}{m}\sum_{a\in A}\Big(N_B(a+t_k) - N_B(a+t_{k-1})\Big)$. As follows from the first proof of Theorem 2.4,

$$\Big(N_B(a+t_k) - N_B(a+t_{k-1})\Big)\cdot \Delta t_k \geq 0,$$

for each $k \geq 1$, where equally holds if and only if a does not change its B-nearest neighbor. Hence $\Delta t_{k+1}\cdot\Delta t_k \geq 0$. It is easily checked that equality is possible only after the last step (where $\Delta t_{k+1} = 0$). $\Box$

LEMMA 2.8. *At each iteration $i \geq 1$ of the algorithm $RMS(t_{i-1}) - RMS(t_i) \geq \|\Delta t_i\|^2$.*

Proof: By (3), we have $\Delta t_i = \frac{1}{m}\sum_{a\in A}(N_B(a+t_{i-1}) - a - t_{i-1})$. Hence

$$RMS(t_{i-1}) - RMS(t_i) = RMS(t_{i-1}) - RMS(t_{i-1}+\Delta t_i) =$$

$$\frac{1}{m}\sum_{a\in A}\|a+t_{i-1} - N_B(a+t_{i-1})\|^2 -$$

$$\frac{1}{m}\sum_{a\in A}\|a+t_{i-1}+\Delta t_i - N_B(a+t_{i-1}+\Delta t_i)\|^2 =$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|a+t_{i-1}-N_B(a+t_{i-1})\|^2 - \|a+t_{i-1}+\Delta t_i - N_B(a+t_{i-1})\|^2\Big) +$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|a+t_{i-1}+\Delta t_i - N_B(a+t_{i-1})\|^2 -$$

$$\|a+t_{i-1}+\Delta t_i - N_B(a+t_{i-1}+\Delta t_i)\|^2\Big) \geq$$

$$\frac{1}{m}\sum_{a\in A} -\Big((2(a+t_{i-1}) + \Delta t_i - 2N_B(a+t_{i-1}))\cdot\Delta t_i\Big),$$

because each term in the second sum is non-negative, by definition of nearest neighbors. The latter expression is $\frac{1}{m}\sum_{a\in A} 2(N_B(a+t_{i-1}) - a - t_{i-1})\cdot\Delta t_i - \|\Delta t_i\|^2$, which, by definition of Δt_i, is equal to $\|\Delta t_i\|^2$. $\Box$

COROLLARY 2.9. *If the relative translations computed by the algorithm are $\Delta t_1, \ldots, \Delta t_k$, then*

$$\frac{1}{k}\left(\sum_{i=1}^{k}\Delta t_i\right)^2 \leq \sum_{i=1}^{k}\|\Delta t_i\|^2 \leq RMS(0) - RMS(t_k). \quad (4)$$

Proof: Use the Cauchy-Schwarz inequality. $\Box$

LEMMA 2.10. *At each iteration $i \geq 1$ of the algorithm*

$$RMS(0) - RMS(t_i) \leq \|t_{i+1}\|^2 - \|\Delta t_{i+1}\|^2. \quad (5)$$

Proof: We have

$$RMS(t_i) - RMS(0) = \frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i)-a-t_i\|^2 - \|N_B(a)-a\|^2\Big) =$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i) - a - t_i\|^2 - \|N_B(a+t_i) - a\|^2\Big) +$$

$$\frac{1}{m}\sum_{a\in A}\Big(\|N_B(a+t_i) - a\|^2 - \|N_B(a) - a\|^2\Big).$$

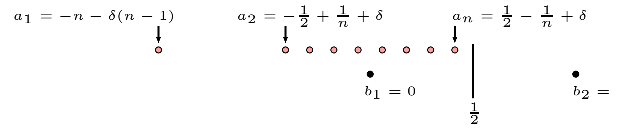

Figure 4: The lower bound construction. Only the two leftmost cells of $\mathcal{V}(B)$ are depicted.

As in the proof of Lemma 2.8, the second sum is non-negative, and the first sum is

$$\frac{1}{m}\sum_{a\in A}\Big(-t_i\cdot(2(N_B(a+t_i)-a-t_i)+t_i)\Big) = -\|t_i\|^2 - 2t_i\cdot\Delta t_{i+1},$$

by definition of Δt_{i+1}. That is, we have

$$RMS(t_i) - RMS(0) \geq -\|t_i\|^2 - 2t_i\cdot\Delta t_{i+1} =$$

$$-\|t_i+\Delta t_{i+1}\|^2 + \|\Delta t_{i+1}\|^2 = -\|t_{i+1}\|^2 + \|\Delta t_{i+1}\|^2,$$

as asserted. $\Box$

Remarks: (1) Combining inequalities (4) and (5), we obtain, for any $k \geq 1$,

$$\sum_{i=1}^{k}\|\Delta t_i\|^2 \leq RMS(0) - RMS(t_k) \leq \|t_{k+1}\|^2 - \|\Delta t_{k+1}\|^2.$$

In particular, we have, replacing $k+1$ by k for simplicity, $\sum_{i=1}^{k}\|\Delta t_i\|^2 \leq \|t_k\|^2$. Note that, for $d = 1$, this inequality is trivial (and weak), due to the monotonicity of the ICP translations. For $d \geq 2$, the inequality means, informally, that as the ICP is rambling around, the path π that it traces does not get too close to itself. In particular, if each Δt_i is of length at least δ then, after k steps, the distance between the initial and final endpoints of the ICP path is at least $\delta\sqrt{k}$.

(2) Specializing Remark (1) to the case $k = 1$, we obtain $\|\Delta t_1\|^2 \leq RMS(0) - RMS(t_1) \leq \|t_2\|^2 - \|\Delta t_2\|^2$. This provides an alternative proof that the angle between Δt_1 and Δt_2 is non-acute. Moreover, the closer this angle is to $\pi/2$ the sharper is the estimate on the decrease in the RMS function.

3. THE ICP ALGORITHM ON THE LINE UNDER THE RMS MEASURE

In this section we consider the special case $d = 1$, and analyze the performance of the ICP algorithm on the line under the RMS measure. Theorem 2.1 implies that in this case the number of NNA's, and thus the number of iterations of the algorithm, is $O(mn)$. In general, we do not know whether this bound is sharp in the worst case (we strongly believe that it is not). However, in the worst case, the number of iterations can be superlinear:

THEOREM 3.1. *There exist point sets A, B on the real line of arbitrarily large common size n, for which the number of iterations of the ICP algorithm (under the RMS measure) is $\Theta(n\log n)$.*

Proof: We construct two point sets A, B on the real line, where $|A| = |B| = n$. The set A consists of the points $a_1 < \cdots < a_n$, where $a_1 = -n - \delta(n-1)$, $a_i = \frac{2(i-1)-n}{2n} + \delta$, for $i = 2, \ldots, n$, and $\delta = o\left(\frac{1}{n}\right)$ is some sufficiently small parameter. The set B consists of the points $b_i = i - 1$, for $i = 1, \ldots, n$. See Figure 4.

Initially, all the points of A are assigned to b_1. As the algorithm progresses, it keeps translating A to the right. The first translation satisfies

$$\Delta t_1 = \frac{1}{n}\sum_{i=1}^{n}(b_1 - a_i) = \frac{1}{n}(b_1 - a_1) - \frac{n-1}{n}\delta = 1,$$

which implies that after the first iteration of the algorithm all the points of A, except for its leftmost point, are assigned to b_2. Using (2), we have $\Delta t_2 = \frac{1}{n}\sum_{i=1}^{n-1}(b_2 - b_1) = \frac{n-1}{n}$, which implies that the $n-1$ rightmost points of A move to the next Voronoi cell $\mathcal{V}(b_3)$ after the second iteration, so that the distance between the new position of a_n from the right boundary of $\mathcal{V}(b_3)$ is $\frac{2}{n} - \delta$, and the distance between the new position of a_2 and the left boundary of $\mathcal{V}(b_3)$ is δ, as is easily verified.

In the next iteration $\Delta t_3 = \frac{n-1}{n}$ (arguing as above). However, due to the current position of the points of A in $\mathcal{V}(b_3)$, only the $n-2$ rightmost points of A cross the right Voronoi boundary of $\mathcal{V}(b_3)$ (into $\mathcal{V}(b_4)$), the nearest neighbor of a_2 remains unchanged (equal to b_3).

We next show, using induction on the number of Voronoi cells the points of A have crossed so far, the following property. Assume that the points of A, except for the leftmost one, are assigned to b_{n-j+1} and b_{n-j+2}, for some $1 \le j \le n$ (clearly, these assignments can involve only two consecutive Voronoi cells), and consider all iterations of the algorithm, in which some points of A cross the common Voronoi boundary β_{n-j+1} of the cells $\mathcal{V}(b_{n-j+1})$, $\mathcal{V}(b_{n-j+2})$. Then, (i) at each such iteration the relative translation is $\frac{j}{n}$, (ii) at each iteration, other than the last one, the overall number of points of A that cross β_{n-j+1} is exactly j, and no point crosses any other boundary, and (iii) at the last iteration of the round, the overall number of points of A that cross either β_{n-j+1} or β_{n-j+2} is exactly $j-1$. In fact, in the induction step we assume that properties (i), (ii) hold, and then show that property (iii) follows, for j, and that (i) and (ii) hold for $j-1$.

To prove this property, we first note, using (2), that the relative translation at each iteration of the algorithm is $\frac{k}{n}$, for some integer $1 \le k \le n$. The preceding discussion shows vacuously that the induction hypothesis holds for $j = n$ and $j = n-1$. Suppose that it holds for all $j' \ge j$, for some $2 \le j \le n-1$, and consider round $(j-1)$ of the algorithm, during which points of A cross β_{n-j+2} (that is, we consider all iterations with that property). Thus, at each iteration of round j (except for the last one), in which there are points of A that remain in the cell $\mathcal{V}(b_{n-j+1})$, the j rightmost points of A (among those contained in $\mathcal{V}(b_{n-j+1})$) cross β_{n-j+1}. Let us now consider the last such iteration. In this case, all the points of A, except l of them, for some $0 \le l < j$ (and the leftmost point, which we ignore), have crossed β_{n-j+1} (in previous iterations). The key observation is that the distance from the current position of a_n to the next Voronoi boundary β_{n-j+2} is $\frac{l+2}{n} - \delta$ (this follows since we shift in total $n-1$ points of A that are equally spaced apart $\frac{1}{n}$), and since the next translation Δt satisfies $\Delta t = \frac{j}{n}$ (using the induction hypothesis and (2)), it follows that only $j-1$ points of A cross a Voronoi boundary in the next iteration. Moreover, the points $a_2, \ldots, a_{l+1}$ cross the boundary β_{n-j+1}, and the points $a_{n-(j-l-2)}, \ldots, a_n$ cross

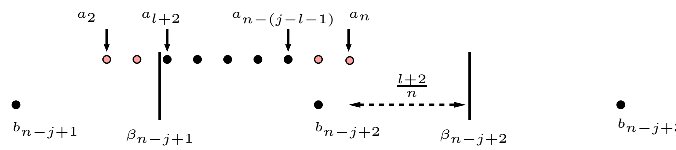

Figure 5: At the last iteration of round j, after shifting the points of A by $\Delta t = \frac{j}{n}$ to the right, the points $a_{l+2}, \ldots, a_{n-(j-l-1)}$ (represented in the figure as black bullets) still remain in $\mathcal{V}(b_{n-j+2})$.

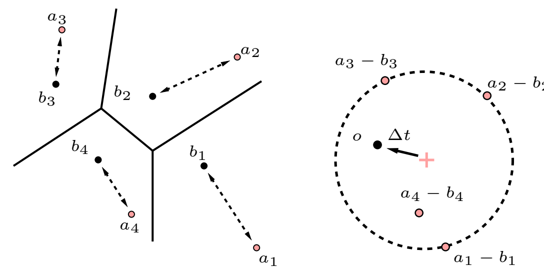

Figure 6: Proof of Lemma 4.2.

.

the boundary β_{n-j+2} (this is the first move in which this boundary is crossed at all); see Figure 5 for an illustration.

Thus, at the next iteration, since only $j-1$ points have just crossed between Voronoi cells, (2) implies that the next translation is $\frac{j-1}{n}$, and, as is easily verified, at each further iteration, as long as there are at least $j-1$ points of A to the left of β_{n-j+2}, this property must continue to hold, and thus $j-1$ points will cross β_{n-j+2}. This establishes the induction step.

It now follows, using the above properties, that the number of iterations required for all the points of A to cross β_{n-j+1} is $\lceil \frac{n}{j} \rceil$, where in the first (last) such iteration some of the points may cross β_{n-j} (β_{n-j+2}) as well. This implies that the number of such iterations, in which the points of A cross only β_{n-j+1} (and none of the two neighboring Voronoi boundaries), is at least $\left\lceil \frac{n}{j} \right\rceil - 2$ (but not more than $\left\lceil \frac{n}{j} \right\rceil$). Thus the overall number of iterations of the algorithm is $\Theta\left(\sum_{j=1}^{n} \left\lceil \frac{n}{j} \right\rceil\right) = \Theta(n \log n)$. □

4. THE PROBLEM UNDER THE HAUSDORFF MEASURE

4.1 General Structural Properties of the ICP Algorithm

Lemma 4.1. *The ICP algorithm converges under the (one-directional) Hausdorff measure, in at most $O(m^d n^d)$ steps.*

Proof: At each iteration i, we compute Δt_i that minimizes $\max_{a \in A} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|$. Since $\|a + t_i - N_B(a + t_i)\| \le \|a + t_i - N_B(a + t_{i-1})\|$, for each $a \in A$, the cost function decreases after each iteration. The lemma then follows from Corollary 2.2. □

The following lemma provides a simple tool to compute the relative translations that the algorithm executes.

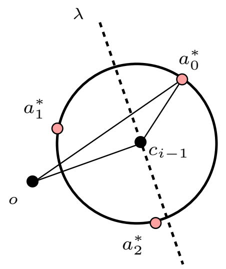

Figure 7: The angle $\measuredangle a_0^* c_{i-1} o$ is obtuse.

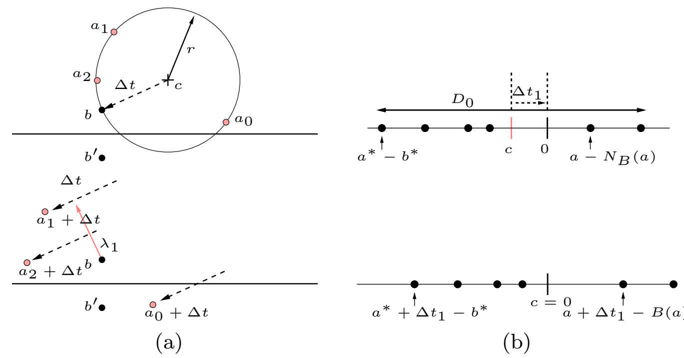

Figure 8: (a) Proof of Lemma 4.3. The point b is placed at the origin, the center of the minimum enclosing disc of the points a_0, a_1, a_2 is c, and its radius is r. Initially, $\|a_0 - b\| = \max \|a_i - b\| > r$, for $i = 0, 1, 2$ (top), and after translating by Δt, $\|a_0 + \Delta t - b'\| < r$ (bottom). (b) Proof of Lemma 4.6. The points $a - N_B(a)$, for $a \in A$, before translating by Δt_1 (top), and after the translation (bottom).

Lemma 4.2. *Let D_{i-1} be the smallest enclosing ball of the points $\{a + t_{i-1} - N_B(a + t_{i-1}) \mid a \in A\}$. Then the next relative translation Δt_i of the ICP algorithm moves the center of D_{i-1} to the origin.*

Proof: The proof follows from the (easy) observation that since D_{i-1} is a minimum enclosing ball, all points appearing on its boundary are not contained in the same halfspace bounded by a hyperplane that passes through its center, and thus any further infinitesimal translation of the points $a + t_{i-1} + \Delta t_i$ from their current position causes at least one of the points on the boundary of (the translated ball) $D_{i-1} + \Delta t_i$ to get further from the origin (which is also the center of $D_{i-1} + \Delta t_i$); see Figure 6 for an illustration. $\square$

In contrast with Theorem 2.4, we have:

Lemma 4.3. *Put $H(t) = \max_{a \in A} \|a + t - N_B(a + t)\|$. In two and higher dimensions, the cost function $H(t_0 + \xi \Delta t)$ of $\xi \in [0, 1]$ does not necessarily decrease monotonically along the relative translation vector Δt that the algorithm executes from translation t_0.*

Proof: A planar example (which can be lifted to any dimension $d \geq 3$) is depicted in Figure 8(a). Initially, all three points a_0, a_1, a_2, are closer to b. By Lemma 4.2, the translation Δt moves the center c of the circumcircle of $\Delta a_0 a_1 a_2$ to b, so the final distance of all three a_i's from b is equal to the radius r of this circle. As we translate each of them by Δt, a_0 crosses into $\mathcal{V}(b')$, its distance to its nearest neighbor (first b and then b') keeps decreasing, and its final value is strictly smaller than r. In contrast, the distances of a_1, a_2 from b (their nearest neighbor throughout the translation) both *increase* towards the end of the translation, and their final values are both r. Hence, towards the end of the translation $H(t_0 + \xi \Delta t)$ is *increasing*. $\square$

Lemma 4.4. *Let $H(t)$ be as above. At each iteration $i \geq 1$ of the algorithm*

$$H(t_{i-1})^2 - H(t_i)^2 \geq \|\Delta t_i\|^2.$$

Proof: Using Lemma 4.2, the next relative translation Δt_i is the vector $c_{i-1}o$, where c_{i-1} is the center of the minimum enclosing ball D_{i-1} of the set $A^* = \{a + t_{i-1} - N_B(a + t_{i-1}) \mid a \in A\}$, and o is the origin.

By Lemma 4.2, the cost $H(t_i)$ (obtained after the relative translation by Δt_i) is the radius of D_{i-1}. Let A_0^* denote the set of all points $a^* \in A^*$ that appear on ∂D_{i-1}, and let a_0^* be the point of A_0^* farthest from the origin.

As above, since D_{i-1} is a minimum enclosing ball, it follows that all points of A_0^* cannot be contained in the same halfspace bounded by a hyperplane through c_{i-1}, which, in particular, implies that a_0^* and o are separated by the hyperplane λ, perpendicular to the segment connecting c_{i-1} and o, and passing through c_{i-1}; see Figure 7 for an illustration. Clearly, the cost $H(t_{i-1})$ is at least $\|a_0^*\|$ (the maximum distance may be obtained by another point of A^* that lies in the interior of D_{i-1}). Hence the angle $\measuredangle a_0^* c_{i-1} o$ is at least $\pi/2$, and thus

$$H(t_{i-1})^2 - H(t_i)^2 \geq \|a_0^*\|^2 - \|a_0^* - c_{i-1}\|^2 \geq \|c_{i-1} - o\|^2 = \|\Delta t_i\|^2.$$

$\square$

Corollary 4.5. *If the relative translations computed by the algorithm are $\Delta t_1, \ldots, \Delta t_k$, then*

$$\frac{1}{k} \left(\sum_{i=1}^{k} \Delta t_i \right)^2 \leq \sum_{i=1}^{k} \|\Delta t_i\|^2 \leq H(0)^2 - H(t_k)^2. \quad (6)$$

4.2 The one-dimensional problem

Let A, B be two point sets on the real line, with $|A| = m$, $|B| = n$.

Lemma 4.6 (Monotonicity). *The points of A are always translated in the same direction, over all iterations of the algorithm. That is, either $\Delta t_i \geq 0$ for each $i \geq 1$, or $\Delta t_i \leq 0$ for each $i \geq 1$.*

Proof: Let $a^* \in A$, $b^* = N_B(a^*)$, be the pair (which is unique if we assume initial general position) that satisfies initially $\xi = |b^* - a^*| = \max_{a \in A} |N_B(a) - a|$. Suppose without loss of generality that $a^* < b^*$. By Lemma 4.2, the initial "ball" (i.e., interval) D_0 has $a^* - b^* = -\xi$ as its left endpoint, and its right endpoint is smaller than ξ (otherwise, the algorithm terminates). Hence the center (midpoint) of D_0 is *negative*, so the first translation Δt_1 of the algorithm is to the right. See Figure 8(b).

After translating, $a^* + \Delta t_1$ is still to the left of b^* (since $\Delta t_1 < \xi$) and is *closer* to b^*, so b^* is still the nearest neighbor of $a^* + \Delta t_1$, and $|a^* + \Delta t_1 - b^*| = \max_{a \in A}\{|a + \Delta t_1 - N_B(a)|\} \geq \max_{a \in A}\{|a + \Delta t_1 - N_B(a + \Delta t_1)|\}$, as is easily verified. Thus $a^* + \Delta t_1 - b^*$ is still the left endpoint of the new interval D_1, whose right endpoint is *closer* to the origin (or at the same distance, in which case the algorithm terminates). Hence, the preceding argument implies that Δt_2 will also be to the right, and, using induction, the lemma follows. □

Remarks: (1) The proof implies that the pair a^*, b^*, which attains the maximum value of the cost function at the initial position of A continues to do so over *all* iterations of the algorithm. The point a^* gets closer to b^*, and can never exit its cell $\mathcal{V}(b^*)$ (actually, it never passes over b^*).
(2) The relative translation Δt_i is always determined by a^*, b^*, and by another pair of points a', b', which determine the other endpoint of D_{i-1}. Note that in the next iteration $N_B(a')$ must change, or else the algorithm terminates.
(3) While monotonicity holds in $\mathbb{R}^1$, we do not know (in view of Lemma 4.3) whether the analog of Theorem 2.5 holds for the Hausdorff measure in two (and higher) dimensions.

Our main result on the ICP algorithm under the Hausdorff measure is given in the following theorem.

THEOREM 4.7. *Let A and B be two point sets on the real line, with $|A| = m$, $|B| = n$, and let δ_B be the spread of B. Then the number of iterations that the ICP algorithm executes is $O\left((m+n)\log\delta_B/\log n\right)$.*

Proof: Let the elements of A be $a_1 < a_2 < \ldots < a_m$, and those of B be $b_1 < b_2 < \ldots < b_n$. Put $\Delta_A = a_m - a_1$, $\Delta_B = b_n - b_1$. Assume, without loss of generality, that, initially, all the points of A lie to the left of all the points of B, and that a_m, b_1 coincide. Then $b_1 - a_1 = \max_{a\in A}|N_B(a) - a| = \Delta_A$, and the initial interval D_0 (in the above notation) is $[a_1 - b_1, 0]$. As shown in Lemma 4.6, all translations will be to the right, and a_1 will stay to the left of b_1. Thus the overall length of all translations is at most $b_1 - a_1 = \Delta_A$. Put $I_{k-1} = b_1 - (a_1 + t_{k-1})$, for each iteration $k \geq 1$ of the algorithm.

A relative translation Δt_k, computed at the k-th iteration of the algorithm, for $k \geq 0$, is said to be *short* if $\Delta t_k < \frac{I_{k-1}}{2n/\log n}$, otherwise, Δt_k is *long*. We first claim that the overall number of (short and long) relative translations that the algorithm executes is $O\left(m\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$.

We say that a pair (a', b') of points, $a' \in A$, $b' \in B$, $a' \neq a_1$, is a *configuration* of the algorithm, if, at some iteration k, $a' - b'$ is the right endpoint of D_{k-1} (so (a_1, b_1), (a', b') determine the k-th relative translation of the algorithm). Due to monotonicity, each configuration can arise at most once, and thus an upper bound on the overall number of such configurations also applies to the actual number of iterations performed by the algorithm.

The idea of the proof is as follows. The overall number of long relative translations is relatively small, since, after performing each of them, the distance between (the translated copy of) a_1 and b_1 (which measures the cost function) significantly decreases. As to the number of short relative translations, if there are at least two configurations involving the *same* point $a' \neq a_1$ in A, which determine short relative translations, then the cost function must significantly decrease (since a' has changed its nearest neighbor, and becomes significantly further from its previous nearest neighbor), and, as a result, each such point a' cannot be involved in too many configurations that determine short relative translations.

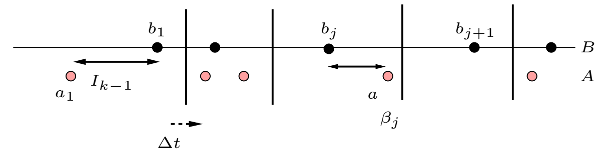

Figure 9: Proof of Theorem 4.7.

Let $\mathcal{S}$ be the sequence of all configurations produced by the algorithm (sorted by the "chronological" order of their creation), which determine short relative translations. We next bound the number of *a-configurations* in $\mathcal{S}$, namely, those that involve the same point $a \in A$.

Fix some $a \neq a_1 \in A$. Let (a, b_j), (a, b_l), $1 \leq j \neq l \leq n$, be two consecutive configurations in $\mathcal{S}$, so each configuration that appears between (a, b_j), (a, b_l) does not involve a. Due to the monotonicity of the relative translations, we must have $j < l$. Suppose that (a, b_j) arises at the k-th iteration, and (a, b_l) arises at the k'-th iteration ($k' > k$). Since (a, b_j) determines a short relative translation, (the translated copy of) a must lie to the right of b_j before the k-th step, for otherwise Δt_k would be at least $\frac{I_{k-1}}{2}$, and thus would not be short. Furthermore, we have, by construction,

$$\Delta t_k = \frac{1}{2}\left(I_{k-1} + (b_j - (a + t_{k-1}))\right) < \frac{I_{k-1}}{2n/\log n},$$

and thus

$$|b_j - (a + t_{k-1})| \geq I_{k-1} - \frac{I_{k-1}}{n/\log n}.$$

Hence

$$|b_l - b_j| \geq |b_{j+1} - b_j| \geq 2\left(I_{k-1} - \frac{I_{k-1}}{n/\log n}\right),$$

and, in particular, $|b_{j+1} - (a + t_{k-1})| \geq I_{k-1} - \frac{I_{k-1}}{n/\log n}$. Thus a can pass over b_{j+1} only if we further translate it by at least $\left(I_{k-1} - \frac{I_{k-1}}{n/\log n}\right)$; see Figure 9 for an illustration. Since (a, b_l) determines a short relative translation at the k'-th iteration (and thus lies to the right of b_l at that time), it follows that $\sum_{r=k}^{k'-1}\Delta t_r > I_{k-1} - \frac{I_{k-1}}{n/\log n}$. But then, $|b_1 - (a_1 + t_{k'-1})| < \frac{I_{k-1}}{n/\log n}$. Thus the cost function is reduced by a factor of at least $n/\log n$ between each two consecutive configurations of $\mathcal{S}$ that involve the same point $a \neq a_1$ of A.

We now show that the overall number of such configurations is $O\left(\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$, for a fixed point $a \neq a_1 \in A$. Let C_B be the distance between the closest pair in B; that is, $C_B = \frac{\Delta_B}{\delta_B}$. We claim that when I_{k-1} becomes smaller than $\frac{C_B}{4}$ (at some iteration $k \geq 1$), the algorithm terminates. Indeed, since $I_{k-1} = \max_{a\in A}\{|N_B(a + t_{k-1}) - (a + t_{k-1})|\}$, this implies that the next relative translation satisfies $|\Delta t_k| < \frac{C_B}{4}$. On the other hand, the distance between each (translated) point $a + t_{k-1}$, $a \in A$, to its nearest Voronoi boundary is at least $\frac{C_B}{4}$ (since the distance between any $b \in B$ and the (left or right) boundary of its Voronoi cell $\mathcal{V}(b)$ is at least $\frac{C_B}{2}$), and thus, after shifting the points by Δt_k, the nearest-neighbor assignments do not change. This easily implies that the overall number of iterations, in which I_0 is reduced by a factor of at least $n/\log n$ until it becomes

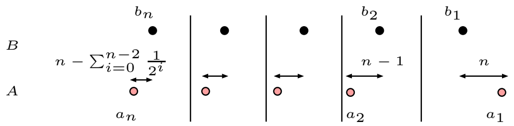

Figure 10: The lower bound construction.

smaller than $\frac{C_B}{4}$, is

$$O(\log_n(\Delta_A) - \log_n(C_B)) = O\left(\log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right),$$

as asserted. Thus the overall number of iterations of the algorithm that involve short relative translations, over all points of A, is $O\left(m \log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$.

We next show that the overall number of long relative translations is $O\left(n \log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$. A long relative translation Δt_k reduces I_{k-1} by a factor of at least $(1 - \frac{1}{2n/\log n})$, so if j long relative translations occur before the k-th iteration then $I_{k-1} \le \Delta_A\left(1 - \frac{1}{2n/\log n}\right)^j$. Arguing as above, the largest value of j for which $\Delta_A\left(1 - \frac{\log n}{2n}\right)^j \ge \frac{C_B}{4}$ satisfies (using the fact that $(1-x) < e^{-x}$, for $0 < x < 1$) $j = O\left(n \log\left(\frac{\Delta_A}{\Delta_B}\delta_B\right)/\log n\right)$.

In order to remove the factor $\log\frac{\Delta_A}{\Delta_B}$ from the bound, we argue that when $\Delta_A \ge 4\Delta_B$, the algorithm terminates after at most two iterations. Indeed, the first relative translation Δt_1 satisfies $|\Delta t_1| \ge \frac{\Delta_A}{2} \ge 2\Delta_B$ (assuming all the points of A lie initially to the left of the points in B). This implies that after the first iteration of the algorithm, the next relative translation is determined by (a_1, b_1), (a_m, b_n), and these two pairs of points maintain this property in any further iteration, so the algorithm will terminate at the next iteration, as claimed. This is easily shown to hold also by any other initial placement of A. Hence, the actual bound on the overall number of iterations is $O\left((m+n)\log\delta_B/\log n\right)$, which completes the proof of the theorem. □

Corollary 4.8. *The number of iterations of the ICP algorithm is $O(m+n)$ when the spread of the point set B is polynomial in n, where the constant of proportionality is linear in the degree of that polynomial bound.*

Our second main result of this section is a matching linear lower bound construction, for the case where the spread of B is polynomial (actually linear) in n.

Theorem 4.9. *There exist point sets A, B of arbitrarily large common size n, such that the spread of B is linear, for which the number of iterations of the algorithm is $\Theta(n)$.*

Proof: We construct two point sets A, B on the real line, with $|A| = |B| = n$. For simplicity of the analysis, we implicitly define the two point sets by the following relations: (i) $a_1 = 0$, (ii) $a_1 - b_1 = n$, (iii) $a_j - b_j = -\left(n - \sum_{k=0}^{j-2}\frac{1}{2^k}\right)$, for each $2 \le j \le n$, and (iv) $a_1 - \frac{b_1+b_2}{2} = 2n$, $a_j - \frac{b_j+b_{j+1}}{2} = \sum_{k=1}^{j-1}\frac{1}{2^k} - \varepsilon$, for each $2 \le j \le n-1$, where $\varepsilon = o\left(\frac{1}{2^n}\right)$. It is easy to verify that the above conditions determine uniquely the sets A and B, and that $2(n-1) < |b_{j+1} - b_j| \le 2n$, for each $j = 1, \ldots, n-1$, and thus the spread of B is $O(n)$. Note that in this construction each point $a_j \in A$ is initially located in the respective Voronoi cell $\mathcal{V}(b_j)$, for $j = 1, \ldots, n$; see Figure 10 for an illustration.

We now claim, using induction on the number of iterations of the algorithm, that the relative translation at the i-th iteration Δt_i is $-\frac{1}{2^i}$, for $i \ge 1$. As a consequence, each point $a_j \in A$ progresses to the left towards $\mathcal{V}(b_{j+1})$, and, in particular, a_{i+1} crosses the Voronoi boundary common to $\mathcal{V}(b_{i+1})$ and $\mathcal{V}(b_{i+2})$ due to property (iv), for $i = 1, \ldots, n-2$. In addition, all the remaining NNA's remain the same (at that iteration), and the nearest neighbor of a_{i+1} remains b_{i+2} at any further iteration — see below. This would imply that the overall number of iterations is $n-2$, which asserts our bound.

The pair a_1, b_1 satisfies $b_1 = N_B(a_1)$ and $|a_1 - b_1| = \max_{a \in A}|a - N_B(a)|$, as is easily verified, and, by Lemma 4.6, this pair attains the maximum value of the cost function at each subsequent iteration of the algorithm. Thus (at the first iteration of the algorithm) $(a_1 - b_1)$ is the right endpoint of the interval D_0, and, $(a_2 - b_2)$ is its left endpoint. Hence

$$\Delta t_1 = \frac{(b_1 - a_1) + (b_2 - a_2)}{2} = -\frac{1}{2},$$

and, as a consequence, all the points of A move to the left, and, due to property (iv) of the construction, the nearest neighbor of a_2 becomes b_3, and the NNA's of all the remaining points do not change. Suppose now, for the induction hypothesis, that at the $(i-1)$-th iteration $\Delta t_{i-1} = -\frac{1}{2^{i-1}}$, and, as a consequence, the overall computed translation t_{i-1} is $-\sum_{j=1}^{i-1}\frac{1}{2^j}$. It can be easily verified, using property (iv), that each point a_j, $j = 2, \ldots, i$, has exchanged its nearest neighbor in B from b_j to b_{j+1}, and that a_j is located to the right of b_{j+1}. We next claim, using properties (iii) and (iv), that each of these points satisfies

$$(a_j + t_{i-1} - b_{j+1}) = n - 2\varepsilon - \sum_{k=1}^{i-1}\frac{1}{2^k} < \tag{7}$$

$$(a_1 + t_{i-1} - b_1) = n - \sum_{k=1}^{i-1}\frac{1}{2^k},$$

as is easily verified. In addition, due to property (iii) $(a_{i+1} + t_{i-1} - b_{i+1}) = -(n-1) < (a_j + t_{i-1} - b_j)$, for each $j = i+2, \ldots, n$. That is, $(a_{i+1} + t_{i-1} - b_{i+1})$ is the left endpoint of the "ball" (i.e., interval) D_{i-1}. Thus, at the i-th step we have

$$\Delta t_i = \frac{(b_1 - (a_1 + t_{i-1})) + (b_{i+1} - (a_{i+1} + t_{i-1}))}{2} =$$

$$-\frac{1 - \sum_{k=1}^{i-1}\frac{1}{2^k}}{2} = -\frac{1}{2^i},$$

as asserted, which, using property (iv), implies that the new nearest neighbor of a_{i+1} is b_{i+2}. Note that it can be easily verified, using (7) and properties (iii), (iv), that all the remaining points remain in their previously computed cells, and, in particular, that none of the points a_j, for $j = 2, \ldots, i$ may exit the cell $\mathcal{V}(b_{j+1})$ in any further iteration (since the overall translation length is less than 1). This completes the induction step. Note that, the NNA's of the points a_1, a_n do not change during the execution of the algorithm, and thus the overall number of iterations is $n-2$, as asserted. □

Remark: In the above construction, the number of bits that is required in order to represent each input point is

$\Theta(n)$. We are not aware of any construction in which this number is $O(\log n)$ and the number of iterations is $\Omega(n)$. We would therefore like to conjecture that in the latter case the overall number of iterations that the algorithm performs is sublinear.

5. CONCLUDING REMARKS

One major open problem that this paper raises is to improve the upper bound, or, alternatively, present a tight lower bound construction, on the number of iterations performed by the algorithm under each of the above measures. This problem is challenging even in the one-dimensional case. So far, we have not managed to obtain a construction that yields $\Omega(n^2)$ iterations (under the RMS measure), and we conjecture that the actual bound is subquadratic in this case, perhaps matching our lower bound, i.e., $\Theta(n \log n)$.

Another problem concerns the running time of the algorithm. The algorithm has to reassign the points in A to their (new) nearest neighbors in B at each iteration. This can be done by searching with each point of A in $\mathcal{V}(B)$, but this will take time that is more than linear in m for each iteration. Thus, for points in $\mathbb{R}^1$, when the number of iterations is linear or super-linear, we face a super-quadratic running time. The irony is that we can solve the pattern matching problem (for the RMS measure) directly, without using the ICP algorithm, in $O(mn \log m)$ time, as follows. (i) Compute the overlay $M(A, B)$ of the Voronoi diagrams $\mathcal{V}(B - a)$, for $a \in A$, in $O(mn \log m)$ time. (ii) Process the intervals of $M(A, B)$ from left to right. (iii) For each interval I, compute the corresponding NNA by updating, in $O(1)$ time, the NNA of the previous interval (only one point changes its nearest neighbor). (iv) Obtain, in $O(1)$ time, the minimizing translation of this NNA, using (3) for the leftmost interval, and (2) for any sequential interval, and the corresponding value of the cost function. (v) Collect those I for which the minimizing translation lies in I; these are the *local minima* of the cost function. (vi) Output the global minimum from among those minima. The problem can also be solved for the Hausdorff measure in $O(mn \log m)$ time, by computing the upper envelope of the m Voronoi surfaces $\mathcal{S}(B - a)$, for $a \in A$, and reporting its global minimum (see, e.g., [10]).

Of course, in practice the ICP algorithm tends to perform much fewer steps, so it performs much faster than this worst case bound. We remark that a variant of the preceding algorithm (for points in $\mathbb{R}^1$) can be employed in the ICP algorithm, so that the overall cost of updating the NNA's remains $O(mn \log n)$, regardless of how many iterations it performs. Many interesting open problems arise in this connection, such as finding a faster procedure to handle the NNA updates, analyzing the performance under the Hausdorff distance and in higher dimensions, and so on.

Moreover, inspired by a comment of D. Kozlow, if we contend ourselves with finding a *local minimum* of the cost function, this can be found in near-linear time, using binary search over the intervals of $M(A, B)$, which we keep *implicit*. Details are omitted here.

Clearly, one expects the algorithm to converge faster (say, under the RMS measure) when the initial placement of A is sufficiently close to B, in the sense that $RMS(0)$ is small. Attempts to exploit such heuristics in practice are reported in [5, 9]. It would be interesting to quantify this "belief", and show that when $RMS(0)$ is smaller than some threshold that depends on the layout of B, the algorithm converges after very few iterations.

Finally, we note that some of the results given in this paper were supported and verified by running experimentation. Our implementation is based on the CGAL [1] and LEDA [2] libraries.

Acknowledgments.. The authors wish to thank Boris Aronov for useful discussions concerning the problem. In particular, during these discussions, the second proof of Theorem 2.4 has been obtained, and the parametrization given in the construction presented in Section 3 has been simplified (from the original construction given by the authors). The authors also wish to thank Leo Guibas, Pankaj Agarwal, Jie Gao and Vladlen Koltun for helpful discussions concerning this problem. In particular, the first proof of Theorem 2.4 has been obtained during these discussions. We also thank Emo Welzl, Dmitri Kozlow and Sariel Har-Peled for helpful comments on the problem.

6. REFERENCES

[1] The CGAL project homepage. `http://www.cgal.org/`.

[2] The LEDA homepage. `http://www.algorithmic-solutions.com/enleda.htm`.

[3] H. Alt and L. Guibas. Discrete geometric shapes: matching, interpolation, and approximation. In *Handbook of Computational Geometry.* J.-R. Sack and J. Urrutia, eds. Elsevier, Amsterdam, pages 121–153, 1999.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

[5] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 260–267, Oct. 2003.

[6] S. Har-Peled and B. Sadri. How fast is the k-means method? *Algorithmica*, 41(3):185–202, 2005.

[7] V. Koltun, and M. Sharir. On overlays and minimization diagrams. These proceedings.

[8] H. Pottmann, Q.-X. Huang, Y.-L. Yang, and S.-M. Hu. Geometry and convergence analysis of algorithms for registration of 3D shapes. Technical Report 117, Geometry Preprint Series, TU Wien, June 2004.

[9] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Third Internat. Conf. 3D Digital Imag. Model. (3DIM)*, pages 145–152, June 2001.

[10] M. Sharir and P. Agarwal. *Davenport-Schinzel Sequences and their Geometric Applications.* Cambridge University Press, New York, 1995.

[11] G. C. Sharp, S. W. Lee and D. K. Wehe. ICP registration using invariant features, *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(1):90–102, 2002.