On the ICP Algorithm[*]

Esther Ezra[†] Micha Sharir[‡] Alon Efrat[§]

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. The considered (standard) measure of resemblance that the algorithm attempts to optimize is the RMS (root mean squared distance). We show that 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, where n is the overall size of the input. We also present several structural geometric properties of the algorithm. 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.

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. A prominent one is the *sum of squared distances* or *root mean square* [4, 5], Under this measure, the cost function is $\Phi_2(A, B) = \frac{1}{m}\sum_{a \in A} \|a - N_B(a)\|^2$, where $N_B(a)$ denotes the nearest neighbor of a in B, and where $m = |A|$. 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 $\phi(A + t, B)$, 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. 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[1]. (ii) We then compute the new relative translation Δt_i that minimizes the cost function ϕ (with respect to the above fixed assignment). Specifically, we find the Δt_i that minimizes

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

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

[*]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, by Grant 155/05 from the Israel Science Fund, Israeli Academy of Sciences, 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-0348000) and an ITR/Collaborative Research grant (0312443).

[†]Department of Computer Science, University of Tel-Aviv, estere@post.tau.ac.il

[‡]Department of Computer Science, University of Tel-Aviv, michas@post.tau.ac.il

[§]Department of Computer Science, University of Arizona, alon@cs.arizona.edu

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

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[2].

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.

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 suppose that the ICP algorithm aligns A to B; that is, B is fixed and A is translated to best fit B.

Theorem 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 [6] 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 $d \geq 1$. $\square$

[2]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.

Corollary 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 3 *At each iteration $i \geq 2$ of the algorithm, the relative translation vector Δt_i satisfies*

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

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

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

$\square$

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 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 $\xi \in [0, 1]$.*

Proof: The function (see also (1))

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

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,

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

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

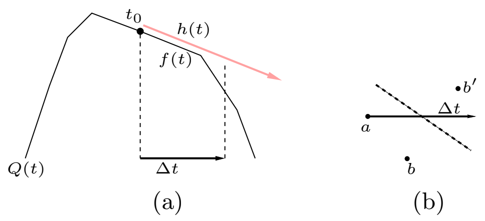


Figure 1: (a) Illustrating the proof that $RMS(t_0 + \xi\Delta t)$ is a strictly decreasing function of $\xi \in [0,1]$. (b) The new nearest neighbor lies ahead of the old one in the direction Δt

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 the boundary of a concave polyhedron (see also the proof of Theorem 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 1(a) for an illustration. Put $RMS_0(\xi) := \frac{1}{m}\sum_{a\in A}\|a + t_0 + \xi\Delta t - N_B(a+t_0)\|^2$. The graph of $RMS_0(\xi)$ is the image of the relative translation vector Δt on the paraboloid $\|t\|^2 + h(t)$. Since $Q(t) \le h(t)$, for any $t \in \mathbb{R}^d$, the concavity of $Q(t)$ implies that for any $0 \le \xi_1 < \xi_2 \le 1$, $Q(t_0 + \xi_1\Delta t) - Q(t_0 + \xi_2\Delta t) \ge h(t_0 + \xi_1\Delta t) - h(t_0 + \xi_2\Delta t)$. Since $\|t\|^2 + h(t)$ is (strictly) monotone decreasing[3] along Δt, we obtain

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

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

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

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

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 4 implies:

Theorem 5 *The ICP path π does not intersect itself.*

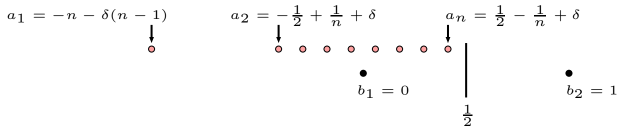
Corollary 6 (Monotonicity) *In the one-dimensional case, the ICP algorithm moves the points of A always in the same (left or right) direction. That is, either $\Delta t_i \ge 0$ for each $i \ge 0$, or $\Delta t_i \le 0$ for each $i \ge 0$.*

Corollary 7 *In any dimension $d \ge 1$, 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 3 we have

$$\Delta t_{k+1} = \frac{1}{m}\sum_{a\in A}\Big(N_B(a+t_k) - N_B(a+t_{k-1})\Big).$$

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



$a_1 = -n - \delta(n-1)$
$a_2 = -\frac{1}{2} + \frac{1}{n} + \delta$
$a_n = \frac{1}{2} - \frac{1}{n} + \delta$
$b_1 = 0$
$\frac{1}{2}$
$b_2 = 1$

Figure 2: The lower bound construction. Only the two leftmost cells of $\mathcal{V}(B)$ are depicted.

It is easy to see that (consult Figure 1(b))

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

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

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 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 8 *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 2.

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).

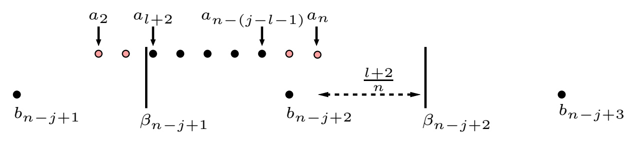


Figure 3: At the last iteration of round j, after shifting the points of A by $\Delta t = \frac{j}{n}$ to the right, the points $a_{l+2}, \ldots, a_{n-(j-l-1)}$ (represented in the figure as black bullets) still remain in $\mathcal{V}(b_{n-j+2})$.

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, as a consequence, show that property (iii) follows.

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 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 by $\frac{1}{n}$), 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 the boundary β_{n-j+2} (this is the first move in which this boundary is crossed at all); see Figure 3 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.[4]

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 $\lceil n/j \rceil$). Thus the overall number of iterations of the algorithm is $\Theta\left(\sum_{j=1}^{n} \left\lceil \frac{n}{j} \right\rceil\right) = \Theta(n \log n)$. $\square$

4 Concluding Remarks

A 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. 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, and we conjecture that the actual bound is subquadratic in this case, perhaps matching our lower bound, i.e., $\Theta(n \log n)$.

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 proof of Theorem 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, a preliminary version of the proof of Theorem 4 has been obtained during these discussions.

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

[6] V. Koltun, and M. Sharir. On overlays of minimization diagrams. Manuscript, 2005.

[4]Note that the induction step first establishes property (iii) of the preceding round, and then (i) and (ii) of the current round.