Abstract

The point-set registration problem seeks the transformation that optimally aligns a point set $A$ to a reference point set $B$, and has numerous applications in computer graphics, computer vision, and robotics. Proposed by Besl and McKay, the iterative closest point (ICP) algorithm has seen wide adoption as a heuristic solution to this problem, with good performance in practice. In this bachelor’s thesis we review and illustrate results from the literature about the convergence characteristics of ICP, focusing primarily on complexity bounds. We first explain in detail some of the results by Ezra, Sharir, and Efrat that bound the number of iterations performed by the algorithm, and then explore work by Arthur and Vassilvitskii that improves on the earlier results. In the process we build an intuition for geometric properties of the algorithm on which the presented ICP point configurations used to prove the complexity bounds rely. Finally, we propose a tool for exploratory analysis by creating convergence diagrams that visualise the final cost value the ICP algorithm converges on when run on $(A+t_0, B)$, for initial offsets $t_0$ on a grid of arbitrary resolution.
# Contents

1 Introduction 3

2 Point-set registration 3

3 The Iterative Closest Point Algorithm 4
   3.1 The optimal translation 5
   3.2 The momentum interpretation 6
   3.3 Convergence guarantee 7
   3.4 Angles between consecutive translations 9

4 Polynomial upper bound 10

5 Linearithmic lower bound on the line 11

6 Quadratic lower bound on the line 15
   6.1 Region-decomposition 17
   6.2 The Linear Shifter 18
   6.3 The Starter 18
   6.4 The Redirector and Booster 19
   6.5 Full construction and lower bound 21

7 Lower bound in higher dimensions 22
   7.1 Resetting ICP 23
   7.2 The Reset Widget 25
   7.3 Proving the reset theorem 28

8 Exploratory analysis with convergence diagrams 30

9 Discussion and open problems 31

References 33

A Errata 34
1 Introduction

The iterative closest point (ICP) algorithm proposed by Besl and McKay [2] is widely used to match and align a geometric object $A$ to a reference object $B$, a problem known in robotics, computer vision, and pattern recognition as point-set registration. Despite a wealth of empirical studies, with hundreds of published variations of the ICP algorithm [6, 7], rigorous theoretical analysis has been comparatively sparse given the significance and empirically proven effectiveness of ICP in applications. In this bachelor’s thesis we will review results from the literature, visualise geometric constructions therein, and lay out proofs and arguments in an expanded form that we hope aids intuitive understanding. Ezra, Sharir, and Efrat [3] laid important groundwork for the theoretical study of the algorithm, giving insights on geometric properties and convergence characteristics of ICP, as well as bounds on the number of iterations. We will review and explain in detail some of their results, including an $\Omega(n \log n)$ lower bound construction for the one-dimensional case. From this construction we move on to an elaborate $\Omega(n^2)$ lower bound construction by Arthur and Vassilvitskii [1] that improves on the earlier result, and is then built upon to prove a lower bound of $\Omega(n/d^{d+1})$ in $d$ dimensions. Shifting our view from the algorithm’s complexity to the quality of its output, we conclude by proposing our own method for exploratory analysis in the form of convergence diagrams that visualise the final value of the cost function for a grid of initial offsets of $A$.

2 Point-set registration

Given two finite subsets $A, B$ of a finite-dimensional real vector space $\mathbb{R}^d$, the rigid point-set registration problem seeks both the point correspondences $(a, b) \in A \times B$ and the spatial transformation $T$ that align $A$ most closely with $B$, where alignment is typically defined as minimising the distances between corresponding points $(a, b)$. This is formally stated as a continuous optimisation problem in which a cost function $\Phi$ is minimised globally over all possible point correspondence mappings $A \rightarrow B$ and all possible transformations $T$. A common choice for $\Phi$ is the root mean square cost function

$$\text{RMS}(T) = \frac{1}{|A|} \sum_{a \in A} \|T(a) - b\|^2,$$

where $b \in B$ is the point that corresponds to $a \in A$ and $\|\cdot\|$ denotes the Euclidean norm. Depending on the number of dimensions $d$ and the specific formulation of the problem, the spatial transformation $T(a) := s \cdot R(a) + t$ may be composed of translation $t$, rotation $R$, and scaling $s$, or use merely a subset of these. The original ICP algorithm proposal was inspired by the most common formulation which seeks to find the translation and rotation that best align two three-dimensional point sets, often referred to as point clouds, without taking scaling into account (i.e. implicitly assuming an optimal scaling factor of 1). Rigid point-set registration is used for a wide range of computer vision applications in areas such as (mobile) robotics, augmented reality, and medical imaging [9, 10], where it helps to align 3D scans of physical objects or environments. As an example, an autonomous vehicle can build up a coherent map of
its environment for navigation by continuously aligning partially overlapping LiDAR scans that update at a rate between 10 and 30 times per second [4].

The more difficult problem of non-rigid point-set registration additionally asks for a deformation of \( \mathcal{A} \), but in the context of ICP and this thesis we are interested in rigid transformations, and will further narrow our focus on the problem under translation only by assuming \( R = \text{Id} \) and \( s = 1 \).

3 The Iterative Closest Point Algorithm

The ICP algorithm is a heuristic solution to the rigid point-set registration problem that in each iteration greedily minimises its cost function, and eventually converges to a (local) minimum. Given two point clouds \( \mathcal{A}, \mathcal{B} \subset \mathbb{R}^d \), ICP optimises for the minimal average distance between corresponding point pairs \((a, b) \in \mathcal{A} \times \mathcal{B}\) by alternating between assigning each point \( a \) to its nearest neighbour \( b \in \mathcal{B} \), and translating the points of \( \mathcal{A} \) in a way that minimises the average distance between corresponding points. While the original formulation calculates both a translation vector \( t \) and a rotation matrix \( R \), the theoretical analyses in this thesis and the literature on which they are based use a simplified version in which \( \mathcal{A} \) and \( \mathcal{B} \) are aligned via translation only. We write \( \Delta t_i \) to denote the relative translation by which the points of \( \mathcal{A} \) are moved in iteration \( i \), and \( t_i = \sum_{j=1}^{i} \Delta t_j \) for the cumulative translation by which they have been translated at the end of iteration \( i \). In each iteration the ICP algorithm performs three steps to minimise the aforementioned root mean square cost function, which for the remainder of this thesis we restate as

\[
\text{RMS}(\Delta t_i) := \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \|a + t_{i-1} + \Delta t_i - N_B(a + t_{i-1})\|^2,
\]

where \( \| \cdot \| \) denotes the Euclidean norm as mentioned before, and \( N_B(a + t_{i-1}) \) represents the nearest neighbour \( b \in \mathcal{B} \) of the translated point \( a + t_{i-1} \). That is, point \( N_B(a + t_{i-1}) \) is the nearest neighbour of point \( a \) at the beginning of iteration \( i \) before it is translated further by \( \Delta t_i \) in the current iteration (for a cumulative translation of \( t_i \)). To achieve this minimisation, the algorithm optimises two quantities: the point correspondences \((a, b)\) and the relative translation \( \Delta t_i \), as follows.

Start with \( t_0 = 0 \). At each iteration \( i \):

1. For each point \( a \in \mathcal{A} \), set \( N_B(a + t_{i-1}) \) to the nearest neighbour of \( a + t_{i-1} \) in \( \mathcal{B} \). Cases of equidistance can be resolved either way, as long as the same principle is applied consistently.

2. Find the relative translation \( \Delta t_i \) that minimises the cost function for the current point correspondences:

\[
\Delta t_i = \arg \min_{\Delta t} \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \|a + t_{i-1} + \Delta t - N_B(a + t_{i-1})\|^2
\]

3. Translate all points of \( \mathcal{A} + t_{i-1} := \{a + t_{i-1} \mid a \in \mathcal{A}\} \) by the new relative translation \( \Delta t_i \) to obtain \( \mathcal{A} + t_i \).
3.1 The optimal translation

Algorithm 1 Iterative Closest Point (ICP) algorithm

Require: $A, B \subseteq \mathbb{R}^d$, $|A|, |B|, d < \infty$

t_0 \leftarrow 0
i \leftarrow 0
do

i \leftarrow i + 1

for each $a \in A$
do

$N_B(a + t_{i-1}) \leftarrow \arg\min_{b \in B} \|a + t_{i-1} - b\|
end for

$\Delta t_i \leftarrow \arg\min_{\Delta t} \frac{1}{|A|} \sum_{a \in A} \|a + t_{i-1} + \Delta t - N_B(a + t_{i-1})\|^2$

$t_i \leftarrow \sum_{j=1}^i \Delta t_j$
while $\Delta t_i \neq 0$
return $t_i$

The algorithm repeatedly performs these steps and terminates in iteration $T$ once a relative translation $\Delta t_T = 0$ has been determined, which we will show to be the first iteration in which no point in $A$ has changed its nearest neighbour in the previous iteration. The final cumulative translation $t_T = \sum_{i=1}^T \Delta t_i$ is the ICP result for aligning $A$ with $B$. See Algorithm 1 for a formal description in pseudocode.

Note that ICP only guarantees local optimality and does not necessarily produce a result that is optimal over all possible point correspondences. Since the correspondences between points $a$ and $b$ at each iteration are dictated by the nearest neighbour property, the algorithm is susceptible to local minima and the quality of its final result critically relies on the initial position of $A$ with respect to $B$. There exist variants of ICP that guarantee global optimality (at the expense of significant computation). [11]

3.1 The optimal translation

While determining the nearest-neighbour correspondences in step 1 can easily be achieved by either exhaustively checking all $|A| \cdot |B|$ pairings or more efficiently by using a $k$-d tree of $B$, it is not immediately obvious which relative translation $\Delta t_i$ should be chosen in step 2 to minimise the cost with respect to those correspondences.

To simplify the following calculation, let $y := a + t_{i-1} - N_B(a + t_{i-1})$. Note that $y$ does not depend on $\Delta t_i$. We can then write our RMS cost function as

\[
RMS(\Delta t_i) = \frac{1}{|A|} \sum_{a \in A} \|\Delta t_i + y\|^2
= \frac{1}{|A|} \sum_{a \in A} (\Delta t_i + y) \cdot (\Delta t_i + y)
= \frac{1}{|A|} \sum_{a \in A} (\Delta t_i^2 + 2y \cdot \Delta t_i + y^2).
\]
3.2 The momentum interpretation

We obtain the optimal relative translation $\Delta t_i$ under the convex RMS measure by taking the first derivative of this last form and setting it equal to zero:

$$0 = \frac{d}{d\Delta t_i} \text{RMS}(\Delta t_i)$$

$$\Rightarrow 0 = \frac{1}{|A|} \sum_{a \in A} (2\Delta t_i + 2y)$$

$$\Rightarrow 0 = 2\Delta t_i + \frac{2}{|A|} \sum_{a \in A} y$$

$$\Rightarrow 0 = 2\Delta t_i + \frac{2}{|A|} \sum_{a \in A} (a + t_{i-1} - N_B(a + t_{i-1}))$$

$$\Rightarrow \Delta t_i = \frac{1}{|A|} \sum_{a \in A} (N_B(a + t_{i-1}) - (a + t_{i-1})). \quad (1)$$

The optimal translation vector is therefore the mean difference between points of $B$ marked as nearest neighbours and their corresponding points in $A + t_{i-1}$. Equivalently, we can interpret it as the difference $b_0 - a_0$, where $b_0 = \frac{1}{|A|} \sum_{a \in A} N_B(a + t_{i-1})$ is the centroid (mean) of nearest neighbours, and $a_0 = \frac{1}{|A|} \sum_{a \in A} (a + t_{i-1})$ is the centroid of $A + t_{i-1}$. This is intuitive, as the contribution of any individual point $a \in A$ to the total cost becomes 0 if the relative translation perfectly aligns $a + t_{i-1}$ with its nearest neighbour, which is the case for $\Delta t_i = N_B(a + t_{i-1}) - (a + t_{i-1})$. The optimal relative translation for all points $a$, rather than for any individual point of $A$, is consequently the average of these vectors from $a + t_{i-1}$ to $N_B(a + t_{i-1})$.

3.2 The momentum interpretation

An alternative interpretation and derivation of $\Delta t_i$ in iterations $i \geq 2$ that shall prove useful for the subsequent complexity analyses was noted by Ezra, Sharir, and Efrat [3].

**Lemma 1.** At each iteration $i \geq 2$ of the algorithm, the relative translation vector $\Delta t_i$ satisfies

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

**Proof.** Using equation (1) for the optimal translation, we can derive the lemma with the following algebraic manipulation by starting with an index shift.

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

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

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

$$\Rightarrow \sum_{a \in A} N_B(a + t_{i-2}) = \sum_{a \in A} (a + t_{i-2}) + \sum_{a \in A} \Delta t_{i-1}$$

$$\Rightarrow \sum_{a \in A} N_B(a + t_{i-2}) = \sum_{a \in A} (a + t_{i-2} + \Delta t_{i-1})$$
3.3 Convergence guarantee

\[ \sum_{a \in A} N_B(a + t_{i-2}) - N_B(a + t_{i-1}) = \sum_{a \in A} (a + t_{i-1}) \]

\[ \Rightarrow \quad \frac{1}{|A|} \sum_{a \in A} (N_B(a + t_{i-1}) - N_B(a + t_{i-2})) = \frac{1}{|A|} \sum_{a \in A} (N_B(a + t_{i-1}) - (a + t_{i-1})) = \Delta t_i \]

From (1) we know that the right side is equal to \( \Delta t_i \), which confirms that the optimal relative translation for every iteration \( i \geq 2 \) is the mean difference between the current and previous nearest neighbour of each point in \( A \).

In a slight abuse of terminology we refer to definition (2) of the relative translation as the “momentum interpretation”, as it defines \( \Delta t_i \) not primarily in terms of the current relative positions of \( A \) and \( B \), as is the case for (1), but in terms of nearest-neighbour changes for points \( a \) that were moved far enough in the previous iteration to escape the “pull” of their nearest neighbour. This unveils a perhaps non-obvious understanding of which points contribute to the relative translation at any given iteration. Lemma 1 implies that any point \( a \) that did not change its nearest neighbour after the previous translation contributes a zero term to \( \Delta t_i \), meaning any \( a \) for which \( N_B(a + t_{i-1}) = N_B(a + t_{i-2}) \). The ICP algorithm terminates once no point \( a \) has changed its nearest neighbour, in which case \( \Delta t_i = 0 \). Figure 1 demonstrates this on a simple example. To understand this intuitively, recall that the optimal translation \( \Delta t_{i-1} \) moves the centroid \( a_0 \) of \( A \) on top of the centroid \( b_0 \) of all nearest neighbour points. If any point \( a \) changed its nearest neighbour after being translated by \( \Delta t_{i-1} \), the new optimal position for \( a_0 \) is the new centroid \( b_0 \) of the updated set of nearest neighbours. Each new neighbour \( b \) contributes its own position to \( b_0 \) divided by \( |A| \). The difference between the new and previous nearest-neighbour centroids is defined entirely by the contributions of the neighbours that changed. This implies that the optimal relative translation is equal to the average difference between the current and previous nearest neighbour of each point \( a \).

Note that a point \( a \) that has not changed its nearest neighbour in the previous iteration may still contribute to \( \Delta t_i \) with respect to (1), i.e. through its distance from its nearest neighbour, and that it always affects \( \Delta t_i \) by contributing to \( |A| \). However, equation (2) allows us to focus our attention solely on those points of \( A \) that have changed their neighbours, and to explain the relative translation fully as a result of these changes. It is worth internalising this observation, as it allows us to intuitively design point constructions \( (A, B) \) in which points of \( A \) change their nearest neighbour from a point \( b \) to a point \( b' \), triggering a contribution of \( (b' - b) / |A| \) to the next relative translation. By carefully constructing the initial placement of these points \( a, b, \) and \( b' \), we can control the sequence of relative translations in order to prove lower bounds on the algorithm’s complexity.

3.3 Convergence guarantee

In their original publication of the ICP algorithm, Besl and McKay [2] show that the ICP algorithm converges monotonically to a (local) minimum. Intuitively, this follows
3.3 Convergence guarantee

![Figure 1: Momentum interpretation. Points of $A$ are red, those of $B$ are violet. The centroid $a_0$ of $A$ is marked by the red star, the centroid $b_0 = \frac{1}{|A|} N_B(a)$ of all nearest neighbours is marked by the violet star. The optimal relative translation moves $a_0$ to $b_0$, which in turn moves when a nearest-neighbour change occurs. When no points change their neighbour, the algorithm terminates in the following iteration.](image)

from the fact that in each iteration, the algorithm performs two actions that each attempt to reduce the value of its cost function, but in any case cannot increase it. The nearest-neighbour assignment pairs each point $a \in A$ with its closest point $b \in B$ and the relative translation is derived directly from the cost function to minimise the average distance between corresponding points $(a, b)$. Based on the new positions of the points $a$ that have been moved closer to their respective nearest neighbour on average, the new nearest-neighbour assignment in the following iteration will necessarily lead to new point correspondences that are no further apart than the previous pairs (again, on average). The following explanations formalise these arguments.

**Theorem 1 (Convergence).** The ICP algorithm always converges monotonically to a local minimum with respect to the RMS cost function.

**Proof.** In each iteration $i \geq 1$, the ICP algorithm first determines the nearest-neighbour assignments

$$NNA_i(A, B) = \{(a, b = N_B(a + t_{i-1})) \mid a \in A\}$$

between point pairs $(a, b)$, where

$$N_B(a + t_{i-1}) = \arg \min_{b \in B} \|a + t_{i-1} - b\|$$

is the point $b \in B$ closest to the current position of $a$. Initially, the translation is $t_0 = 0$. 

8
The error for these correspondences before calculating and applying the next relative translation $\Delta t_i$ is
\[
e_1(i) = \frac{1}{|A|} \sum_{a \in A} \|a + t_{i-1} - N_B(a + t_{i-1})\|^2.
\]
The updated error after calculating and applying the optimal translation $\Delta t_i$ but before determining the new correspondences based on the new position of $A + t_{i-1}$ will be
\[
e_2(i) = \frac{1}{|A|} \sum_{a \in A} \|a + t_i - N_B(a + t_{i-1})\|^2.
\]
These values always satisfy $e_2(i) \leq e_1(i)$, because if it were the case that $e_2(i) > e_1(i)$, then $t_i = t_{i-1}$ would yield a smaller error $e_2'(i) < e_2(i)$, which contradicts the optimality of $t_i$. Based on $t_i$, the new nearest-neighbour correspondences $N_{NA_{i+1}}(A, B)$ are determined at the beginning of the next iteration $i + 1$. It is clear that
\[
\|a + t_i - N_B(a + t_j)\| \leq \|a + t_i - N_B(a + t_{i-1})\|
\]
for each point $a \in A$ because $N_B(a + t_{i-1})$ was the nearest-neighbour of $a + t_{i-1}$ in $B$ prior to translation $t_i$. If the distance between $N_B(a + t_{i-1})$ and $a + t_i$ were lower than that between $N_B(a + t_i)$ and $a + t_i$, this would directly contradict the nearest-neighbour property of $N_B$. From this and the fact that mean square errors are bounded from below by 0, it follows that for all $i$ the errors must satisfy
\[
0 \leq e_2(i + 1) \leq e_1(i + 1) \leq e_2(i) \leq e_1(i),
\]
and since this sequence of errors is non-increasing and bounded from below, the ICP algorithm must converge monotonically to a minimum cost value. \hfill \Box

### 3.4 Angles between consecutive translations

Another property that will be used in our later constructions, noted in [3], is the fact that angles between consecutive relative translation vectors $\Delta t_i$ and $\Delta t_{i+1}$ are always acute, and so the angles between adjacent edges in the polygonal path they form are obtuse. Let $b = N_B(a + t_i)$ and $b' = N_B(a + t_{i+1})$. Since in order for point $a$ to have changed its nearest neighbour from $b$ to $b'$ in iteration $i + 1$, the relative translation $\Delta t_i$ must have moved it across the bisector of $b$ and $b'$ from the side of $b$ to the side of $b'$, as depicted in Figure 2. Hence we have
\[
(b' - b) \cdot \Delta t_i = (N_B(a + t_{i+1}) - N_B(a + t_i)) \cdot \Delta t_i \geq 0,
\]
and since according to Lemma 1 the relative translation is the average difference between current and previous nearest neighbours, it follows that $\Delta t_{i+1} \cdot \Delta t_i \geq 0$ for each $i \geq 1$. If we concatenate all relative translation vectors $\Delta t_i$ to a polygonal path $\pi$ such that the tail of $\Delta t_{i+1}$ starts at the head of $\Delta t_i$, the angles between adjacent edges of $\pi$ are obtuse.

![Figure 2: Change of nearest neighbour. The vector $b' - b$ from the old to the new nearest neighbour of point $a$ has a positive inner product with the relative translation causing the neighbour change.](image-url)
4 Polynomial upper bound

Ezra, Sharir, and Efrat [3] begin their analysis of the ICP algorithm with the observation that the number of iterations is bounded from above by the number of possible nearest-neighbour assignments $\text{NNA} : \mathcal{A} \rightarrow \mathcal{B}$ that can occur in an ICP execution. While crucial details of the following arguments about the number of these assignments rely on a result by Koltun and Sharir [5] that exceeds the scope of this thesis (that is, the full understanding of its author), we include and illustrate this sketch of an upper bound proof to complement the later sections about lower bounds.

**Theorem 2.** Let $m = |\mathcal{A}|$ and $n = |\mathcal{B}|$ for the ICP configuration $(\mathcal{A}, \mathcal{B}) \subset \mathbb{R}^d \times \mathbb{R}^d$. The maximum possible overall number of nearest-neighbour assignments, over all translated copies of $\mathcal{A}$, is $\Theta(m^d n^d)$.

**Sketch of proof.** Let $\mathcal{V}(\mathcal{B}) = \{ \mathcal{V}(b) \mid b \in \mathcal{B} \}$ denote the Voronoi diagram of $\mathcal{B}$, where each cell $\mathcal{V}(b) = \{ p \in \mathbb{R}^d \mid b = \arg \min_{b' \in \mathcal{B}} \| p - b' \| \}$ contains the points $p$ to which no point of $\mathcal{B}$ is closer than $b$, as illustrated in Figure 3. Whenever the cumulative translation $t_i$ translates a point $a$ from one cell $\mathcal{V}(b)$ into another cell $\mathcal{V}(b')$, this point changes its nearest neighbour from $b$ to $b'$, thus changing the overall nearest-neighbour assignment (NNA). For each $a \in \mathcal{A}$ (before any translation), consider the shifted copy $\mathcal{V}(\mathcal{B}) - a$ of the Voronoi diagram. If the cumulative translation $t$ lies on the boundary between any two cells of $\mathcal{V}(\mathcal{B}) - a$, translating point $a$ may change its nearest neighbour, and therefore the overall NNA (see Figure 4). If we now consider the overlay $M(\mathcal{A}, \mathcal{B})$ of the $m$ shifted diagrams $\mathcal{V}(\mathcal{B}) - a$; for all $i \in \{1, ..., m\}$, we can see that each cell of this overlay consists of translations with a common NNA, and so the number of possible nearest-neighbour assignments is equal to the number of cells in this overlay $M(\mathcal{A}, \mathcal{B})$. A result from [5] implies that this number of cells is in $O(m^d n^d)$. \qed
We now demonstrate a construction that shows that this bound is tight for any $d \geq 1$. Let $B$ consist of $O(n)$ points such that $b_1$ is at the origin and on each of the $d$ coordinate axes $\lfloor \frac{n-1}{d} \rfloor$ points are lined up in intervals of uniform distance $\ell$. For $d = 2$ this resembles an L-shape. Let $\hat{b}_j$ denote the point in $B$ that is furthest from the origin along dimension $j$, i.e. at a distance of $\ell(n-1)/d$ from $b_1$. Then $A$ consists of points $\{a_1, ..., a_m\}$ with $a_1 = \hat{b}_1 - \sum_{j=2}^{d} \hat{b}_j$ and $a_i = i \cdot a_1$ for $i \geq 2$. Figure 5 illustrates this construction for the two-dimensional case. For each dimension, the Voronoi diagram $V(B)$ contains $(n-1)/d$ hyperplanes perpendicular to the axis of that dimension. In the two-dimensional case these are $(n-1)/2$ horizontal and vertical half lines, respectively. For clarity of exposition and without loss of generality we will illustrate the remaining details on the construction in two dimensions. For each point $a_i$, the Voronoi diagram $V(B) - a_i$ is moved up and to the left, creating $(n-1)/d$ new horizontal and vertical half lines, respectively (as well as other diagonal half lines and line segments). After $k$ such copies, their overlay $M(A, B)$ contains $k(n-1)/d$ horizontal half lines, and the next copy $V(B) - a_{k+1}$ will add $(n-1)/d$ new vertical half lines that will intersect the existing horizontal lines to create $k(n-1)^d/d^d$ new rectangular cells. The total number of cells is hereby $\sum_{k=1}^{m} k \left( \frac{n-1}{d} \right)^d \in \Omega(m^d n^d)$.

It follows from theorems 1 and 2 that the ICP algorithm reaches each NNA at most once, and so the number of iterations is at most $O(m^d n^d)$. While Ezra et al. state that it is unclear whether this bound is tight in the worst case, the $\Omega(n^2)$ lower bound construction we will explore in section 6 will resolve this question at least for $d = 1$.

## 5. Linearithmic lower bound on the line

Ezra, Sharir, and Efrat present an $\Omega(n \log n)$ lower bound ICP construction. In this section, we will begin to build an intuition for the behaviour of the ICP algorithm by exploring this construction, laying out the proof in greater detail than in their original
formulation, and fixing minor mistakes in its arguments (described in Appendix A). The section after this one will build on these intuitions to illustrate the improved $\Omega(n^2)$ lower bound construction by Arthur and Vassilvitskii. [1]

**Theorem 3.** There exist point sets $\mathcal{A}, \mathcal{B}$ of arbitrarily large common size $n$ on the real line, for which the number of iterations of the ICP algorithm is $\Theta(n \log n)$ under the RMS measure.

Let $\mathcal{A}, \mathcal{B} \subset \mathbb{R}$ be two finite point sets of equal cardinality $n$, such that $\mathcal{B}$ consists of the points $b_i = i - 1$ for $i \in \{1, \ldots, n\}$, and $\mathcal{A}$ consists of the points $a_i$, where

$$a_1 = -n - (n - 1)\delta,$$

$$a_i = \frac{b_i}{n} - \frac{1}{2} + \delta,$$

for $i \in \{2, \ldots, n\}$, and $\delta = o(1/n)$ is some sufficiently small offset. For many of the following arguments we can and will safely ignore the far left point $a_1$, so we define $A := \{a_2, \ldots, a_n\}$ to be the subset of equally spaced points in $\mathcal{A}$. Moreover, let the Voronoi cell $V(b_i)$ denote the set of points on the line to which no point in $\mathcal{B}$ is closer than $b_i$, which in our simple one-dimensional case with integer values $i - 1$ for each $b_i$ simply describes the open interval $V(b_i) = (b_i - 0.5, b_i + 0.5)$. Figure 6 illustrates this construction. Because the $n - 1$ points of $A$ are spaced at intervals of $1/n$ and so the extrema $a_2$ and $a_n$ are $\frac{n-1}{n} < 1$ apart, clearly the points of $A$ can be in at most two consecutive Voronoi cells at any given time. Two adjacent cells $V(b_i)$ and $V(b_{i+1})$ are separated by their common boundary $\beta_i = \frac{1}{2}(b_i + b_{i+1})$. Given $|\mathcal{B}| = n$, there exist $n - 1$ such boundaries. We define round $j$ as the sequence of iterations starting from the iteration in which boundary $\beta_{n-j+1}$ is crossed for the second time by any point, up to and including the last iteration in which that boundary is crossed for the last time. That is, round $j$ begins with the first iteration at the start of which $\beta_{n-j+1}$ has points of $A$ to its left and its right. Figures 7 and 8 illustrate the notion of a round on an example.

![Figure 7: First iteration of round $j = 3$ for $n = 8$. We show the point in time after calculating the relative translation $\Delta t_i$, drawn as a green arrow of length $3/n = j/n$, and before this translation is applied to $\mathcal{A}$. Since $a_n$ is already to the right of $\beta_6$, this boundary will be crossed for the second time in this iteration.](image1)

![Figure 8: Last iteration of round $j = 3$ for $n = 8$. Two iterations after the one depicted in Figure 7, the relative translation $\Delta t_i$ moves $a_n$ and $a_{n-1}$ across $\beta_7$, making this the first crossing of that boundary.](image2)
As point $a_1 \not\in A$ does not cross any boundary throughout the execution of ICP on this construction, we always refer to points of $A$ when talking about points crossing a boundary, and we call any occurrence of the relative translation moving points of $A$ across the boundary a crossing.

The goal of this proof is to show that over the full execution of ICP on this construction, the points of $A$ will all translate to the right of $b_n$, and that each boundary $\beta_{n-j+1}$ takes between $\lceil \frac{n-2}{j} \rceil + 1$ crossings. To do so, we introduce the following properties.

(i) In each iteration of round $j$ the relative translation is $\frac{j}{n}$.

(ii) In each iteration of round $j$, other than the last one, only boundary $\beta_{n-j+1}$ is crossed, and exactly $j$ points cross it.

(iii) In the last iteration of round $j$ one of three cases occurs:

(a) Exactly $j-1$ points cross a boundary, at least one of which crosses $\beta_{n-j+2}$.

(b) Exactly $j-1$ points cross $\beta_{n-j+1}$ and no other boundary is crossed.

(c) Exactly $j$ points cross $\beta_{n-j+1}$ and no other boundary is crossed.

In cases (b) and (c), the last iteration of round $j$ is followed by a single transition iteration that belongs to no round and in which $j-1$ points cross the next boundary $\beta_{n-j+2}$.

We will show that properties (i) and (ii) hold for all $j' \geq j$, and then show that property (iii) follows for $j$, and that (i) and (ii) hold for $j-1$. Initially, each point $a \in A$ has $b_1$ assigned as its nearest neighbour. The initial translation satisfies

$$\Delta t_1 = \frac{1}{n} \sum_{i=1}^{n} (N_B(a_i + t_{i-1}) - (a_i + t_{i-1}))$$

$$= \frac{1}{n} \sum_{i=1}^{n} (b_1 - a_i)$$

$$= \frac{1}{n} \left( b_1 - a_1 + \sum_{i=2}^{n} \left( b_1 - \frac{b_i}{n} + \frac{1}{2} - \delta \right) \right)$$

$$= \frac{1}{n} \left( n + (n-1)\delta - \frac{1}{n} \sum_{i=1}^{n-1} i + \sum_{i=2}^{n} \frac{1}{2} - \sum_{i=2}^{n} \delta \right)$$

$$= \frac{1}{n} \left( n + (n-1)\delta - \frac{n-1}{2} + \frac{n-1}{2} - (n-1)\delta \right)$$

$$= 1$$

and moves all points of $A$ across the first boundary $\beta_1$, changing the nearest neighbour for each point in $A$ to $b_2$. This also means that round $j = n$ consists of no iterations, and properties (i) - (iii) are vacuously true.

From the second iteration on, we can use Lemma 1 to determine the next relative translation. For this construction with distance 1 between adjacent points in $B$, this
implies that if a point $a$ crossed one boundary $\beta_i$ in the last iteration, the difference $N_B(a + t_{i-1}) - N_B(a + t_{i-2})$ between its current and previous neighbour is 1, and so point $a$ contributes $1/n$ to the relative translation. For iterations $i \geq 2$ the relative translation satisfies $\Delta t_i = k/n$, where $0 \leq k \leq n$ is the number of points in $A$ that crossed a boundary in the previous iteration. We may therefore frame the translation of $A$ as all points $a$ taking $k$ steps along a grid spaced in intervals of $1/n$ as visualized in Figure 6. Since in the first iteration, all $n - 1$ points of $A$ crossed $\beta_1$, we get $\Delta t_2 = \frac{n-1}{n}$, which moves all points of $A$ across the second boundary $\beta_2$ and shows round $j = n - 1$ to also be empty of iterations, trivially satisfying properties (i) - (iii) for the second and last time.

At the beginning of the third iteration, the distance between $a_2$ and the boundary $\beta_1$ to its left is

$$
(a_2 + t_2) - \beta_2 = \left( \frac{b_2}{n} - \frac{1}{2} + \delta \right) + \left( 1 + \frac{n-1}{n} \right) - 1.5
$$

$$
= \frac{1}{n} + \delta + \frac{n-1}{n} - 1
$$

$$
= \delta
$$

and the distance between $a_n$ and the boundary $\beta_3$ to its right is

$$
\beta_3 - (a_n + t_2) = 2.5 - \left( \left( \frac{n-1}{n} - \frac{1}{2} + \delta \right) + \left( 1 + \frac{n-1}{n} \right) \right)
$$

$$
= 2 - 2\frac{n-1}{n} - \delta
$$

$$
= \frac{2}{n} - \delta.
$$

Once again we had $n - 1$ points crossing a boundary in the previous iteration $i = 2$, from which it follows that $\Delta t_3 = \frac{n-1}{n}$, or $n - 1$ steps. The equal spacing of 1 between two adjacent boundaries means that point $a_2$ is $1 - \delta > \Delta t_3$ away from the next boundary $\beta_3$, and does not cross it in iteration $i = 3$. With that, $j = n - 2$ marks the first round that consists of at least one iteration.

Let us suppose now, that the induction hypothesis holds for all $j' \geq j + 1$, for some $3 \leq j + 1 \leq n - 1$, and consider the next round $j$. It then follows that for the first iteration $i$ of round $j$, the relative translation is $j/n$, which moves exactly $j$ points of $A$ across $\beta_{n-j+1}$, each time leading to the same relative translation in the following iteration. This continues up to (and excluding) the last iteration of round $j$, at the beginning of which all but $\ell$ points of $A$ have crossed $\beta_{n-j+1}$ in previous iterations, for some $1 \leq \ell \leq j$.

In the last iteration of round $j$, point $a_2$ is slightly less than $\ell$ steps away from its closest boundary $\beta_{n-j+1}$, i.e. its distance from the boundary is $\frac{\ell}{n} - \delta$. This in turn leads to a distance of $\frac{\ell^2}{n^2} - \delta$ between point $a_n$ and the next boundary $\beta_{n-j+2}$. Keeping in mind the relative translation of $j$ steps, what follows is one of three cases, depending on the value of $\ell$. 

14
Case (a) $1 \leq \ell < n - 1$: Boundary $\beta_{n-j+1}$ is crossed by the $\ell$ points to its left, and boundary $\beta_{n-j+2}$ is crossed by the $j-\ell-1$ rightmost points of $A$. This follows because $a_n$ is translated from a position $\frac{\ell+2}{n} - \delta$ on the left of the next boundary by $\frac{j}{n}$ to a position $\frac{\ell-2}{n} + \delta$ to the right of it, and that new distance fits the $j-\ell-1$ rightmost points of $A$, equally spaced apart by $1/n$. Overall, $j-1$ points cross a boundary, and the next iteration will be the first of round $j-1$, with a corresponding relative translation $\frac{\ell-1}{n}$.

Case (b) $\ell = j-1$: Boundary $\beta_{n-j+1}$ is crossed by the $j-1$ points to its left, but no point crosses another boundary as $a_n$ remains to the left of $\beta_{n-j+2}$ and merely reduces its distance to the next boundary from $\frac{j+1}{n} - \delta$ to $\frac{j}{n} - \delta$. The following transition iteration translates $j-1$ points across $\beta_{n-j+2}$, but belongs to neither round.

Case (c) $\ell = j$: This case is analogous to case (b) except that boundary $\beta_{n-j+1}$ is crossed by $j$ points and $a_n$ is $\frac{j}{n} - \delta$ to the left of $\beta_{n-j+2}$. In the following transition iteration this difference in distance is compensated for by the relative translation of $\frac{j}{n}$, and so $j-1$ points of $A$ cross $\beta_{n-j+2}$ as well.

These observations establish the inductive step. For an annotated visualisation of a complete ICP execution that demonstrates all three cases, see Figure 9.

The highest possible number of iterations for all points of $A$ to cross any boundary $\beta_{n-j+1}$ is $\left\lceil \frac{n-2}{j} \right\rceil + 1$, and occurs in the case that $\beta_{n-j+1}$ is first crossed by only a single point, after which the remaining $n-2$ points of $A$ cross it, $j$ at a time. The lowest possible number of crossing iterations occurs when during the first crossing of $\beta_{n-j+1}$, a full $j$ points cross it, so that the remaining $n-1-j$ points take $\left\lceil \frac{n-1-j}{j} \right\rceil = \left\lceil \frac{n-1}{j} \right\rceil - 1$ additional iterations. Leaving out the first and last iteration in which in addition to $\beta_{n-j+1}$ also the previous boundary $\beta_{n-j}$ or the next one $\beta_{n-j+2}$ could be crossed, we arrive at a minimum iteration count of $\left\lceil \frac{n-1}{j} \right\rceil - 2$ for each of the $n-1$ boundaries.

To obtain the asymptotic complexity, we ignore constants and arrive at $n \cdot H_n$ iterations, where $H_n = \sum_{x=1}^{n} \frac{1}{x}$ is the $n$th harmonic number. Given that

$$\ln(n) = \int_{1}^{n} \frac{1}{x} \, dx < H_n < 1 + \int_{1}^{n} \frac{1}{x} \, dx = \ln(n) + 1,$$

we see that $H_n \in \Theta(\ln n)$, and so for the total number of iterations the result $\Theta(n \ln n)$ follows. Figure 10 compares theoretical complexity to experimental measurements.

6 Quadratic lower bound on the line

Arthur and Vassilvitskii [1] improved on the result from theorem 3 with a creative composition of point configurations that we will explain and illustrate next. At its most fundamental, their construction consists of a combination of “widgets”, each of which is an ICP configuration $(A_j, B_j)$ engineered to cause a particular behaviour when the algorithm is executed on the overall construction $(\mathcal{A}, \mathcal{B}) = \left( \bigcup_j A_j, \bigcup_j B_j \right)$. The $\Omega(n^2)$ lower bound is then obtained by ensuring that $\Theta(n)$ points among those in $\mathcal{A}$ will each take $\Omega(n)$ iterations to move across a subset of $\mathcal{B}$.
6. Quadratic lower bound on the line

Figure 9: Full ICP execution for $n = 7, \delta = 1/n^2$. For better visual clarity the axis range was chosen to focus on $A = \{a_2, ..., a_n\}$ since $a_1$ never changes its nearest neighbour from the initial $N_B(a_1) = b_1$ and can be ignored for most of the presented arguments. Green arrows represent the relative translation $\Delta t_i$ at each iteration, annotated with the number of $1/n$-steps by which they translate $A$. Boundaries $\beta_1$ and $\beta_2$ are fully crossed by $A$ in iterations 1 and 2, respectively. Iteration 4 marks the last iteration of round $j = 5$ and an example of case (a) wherein boundary $\beta_{n-j+1} = \beta_3$ is crossed by $\ell = 1$ point and $\beta_{n-j+2} = \beta_4$ is crossed by 3 points for a total of $4 = j - 1$. Iteration 5 is the last iteration of round $j = 4$ and exemplifies case (b) with $\ell = j - 1 = 3$ points crossing $\beta_{n-j+1} = \beta_4$. This is followed by transition iteration 6, in which the relative translation moves $3 = j - 1$ points across $\beta_{n-j+2} = \beta_5$. Case (c) is demonstrated by iteration 7, the last iteration of round $j = 3$. Here we see $\ell = 3$ points cross $\beta_{n-j+1} = \beta_5$, and in the subsequent transition iteration 8, $2 = j - 1$ points cross $\beta_{n-j+2} = \beta_6$. The algorithm terminates in iteration 12 after no point has changed its nearest neighbour in the previous iteration.
6.1 Region-decomposition

**Definition 1** (Region decomposition). Let \( A = \bigcup A_j \) and \( B = \bigcup B_j \) be a partitioning of point sets. We refer to a pair \((A_j, B_j)\) as region \( j \), and to \( \{(A_j, B_j) \mid j \in \{1, \ldots, r\}\} \) as a region-decomposition if the shortest \((a, b)\) distance between any two points from distinct regions is greater than three times the longest \((a, b)\) distance

\[
\delta = \max_j \max_{a \in A_j, b \in B_j} \|a - b\|
\]

within any region, that is, greater than \(3\delta\). Intuitively, a partitioning of \( A, B \) is a region decomposition if all regions are sufficiently far apart from each other so that nearest neighbour \((a, b)\) correspondences stay within regions over the entirety of an ICP execution, as we will show next.

**Lemma 2.** Suppose \((A, B)\) has region-decomposition \( \{(A_j, B_j) \mid j \in \{1, \ldots, r\}\} \), and suppose the ICP algorithm is executed on \((A, B)\). If \( a \in A_j \), then \( N_B(a + t_i) \) remains in the corresponding \( B_j \) throughout the execution of ICP.

**Proof.** Let \( a \in A_j, b \in B_{j'} \) and \( b' \in B_k \) with \( k \neq j \). Consider an iteration \( i \) at the start of which the points of \( A \) have been translated by a cumulative translation \( t_{i-1} \) that satisfies \( \|t_{i-1}\| \leq \delta \). With \( \delta \) denoting the greatest \((a, b)\)-distance within any region, no point \( a + t_{i-1} \) can be more than \(2\delta\) from the farthest point \( b \) in its region. Moreover, \( t_{i-1} \) can have decreased the distance of \( a \) to the closest \( b' \) from a different region at most to \(3\delta + \epsilon - \delta = 2\delta + \epsilon\), for some \( \epsilon > 0\), which means that \( a + t_{i-1} \) is still farther from \( b' \) than from \( b \) and thus still has a point \( b \) from its own region as its nearest neighbour. Recall that \( t_i = t_{i-1} + \Delta t_i \) translates \( A \) to the optimal position with respect to the nearest-neighbour assignment (NNA) for iteration \( i \). Since \( N_B(a + t_{i-1}) \) for each point \( a \) stays in the corresponding \( B_j \), and the next cumulative translation \( t_i \) moves \( A \) to the optimal position with respect to the newest NNA, \( t_i \) also satisfies \( \|t_i\| \leq \delta \).
6.2 The Linear Shifter

The crucial relationship between \( a, b, \) and \( b' \) is formally described by the inequality

\[
\|a + t_{i-1} - b\| \leq \|a - b\| + \|t_{i-1}\|
\]

\[
\leq 2\delta
\]

\[
\leq \|a - b'\| - \|t_{i-1}\|
\]

\[
> 3\delta
\]

\[
\leq \|a + t_{i-1} - b'\|
\]

In the following constructions we assume that whenever an ICP configuration \((A, B)\) is augmented with new regions, these are added at a distance greater than \(3\delta\) from any other region so that the region-decomposition property is maintained. In doing so we ensure that the new regions can exert their intended influence on the global translation without ever directly interacting with points from other regions with regard to cross-region nearest-neighbour correspondence.

6.2 The Linear Shifter

The Linear Shifter is the first and most central widget in the construction. It is a point configuration \((A, B)\) (to be augmented later for an ICP construction \((A, B)\)) with \( A \) consisting of the single point \( a = 0 \), and \( B \) of \( b_0 = 0 \) and \( b_i = \sum_{j=0}^{i-1} \frac{1}{j!} \) for \( i \in \{1, \ldots, n\} \).

Note that this means each \( b_i = b_{i-1} + \frac{1}{k} \). See Figure 11 for an illustration. Assume the Linear Shifter is part of an ICP configuration such that \( k = |A| \) and we run ICP on it with a first translation \( \Delta t_1 = 1 \). In section 6.3 we will show how to achieve these starting conditions. In the first iteration point \( a \) moves to \( a + t_1 = 0 + 1 = b_1 \), which implies \( \Delta t_2 = \frac{1}{k}(N_B(a + t_{i-1}) - N_B(a + t_{i-2})) = \frac{1}{k}(b_1 - b_0) = \frac{1}{k} \).

This is the beginning of a sequence in which each iteration \( i \) translates point \( a \) by

\[
\Delta t_i = \frac{1}{k}(N_B(a + t_{i-1}) - N_B(a + t_{i-2}))
\]

\[
= \frac{1}{k}(b_{i-1} - b_{i-2})
\]

\[
= \frac{1}{k} \left( \sum_{j=0}^{i-2} \frac{1}{k^j} - \sum_{j=0}^{i-3} \frac{1}{k^j} \right)
\]

\[
= \frac{1}{k^{i-1}}
\]

from \( a + t_{i-1} = b_{i-1} \) to \( a + t_i = b_i \), changing its nearest neighbour accordingly. Point \( a \) altogether traverses each point in \( B \) of the Linear Shifter for a total of \( |B| + 1 \in \mathcal{O}(n) \) iterations. The algorithm terminates once point \( a \) has passed \( b_n \) without changing its nearest neighbour. We refer to this sequence of iterations in which a point \( a \) moves across each point \( b_i \) as the Linear Shifter sequence.

6.3 The Starter

On its own the Linear Shifter does not cause the ICP algorithm to move \( a \) at all. To initiate the sequence described above, we require an additional widget that induces an
6.4 The Redirector and Booster

So far, we have established the widgets required for an ICP construction of linear complexity. In order to achieve quadratic running time, we will augment the Linear Shifter’s $A$ with $n$ additional points that are spaced such that once the first point has completed the sequence in $\Omega(n)$ iterations, the next point begins its own Linear Shifter sequence of iterations.
6.4 The Redirector and Booster

Figure 13: Redirector / Booster. The points \( a, b_1, \) and \( b_2 \) make up Booster, a simplification of the Redirector in [1] that contains the additional region 2 (\( \{a', \} \{b' \} \)) drawn faintly on the left that turns out to be unnecessary for the one-dimensional construction. Point \( a \) is \( y \) to the left of the \( (b_1, b_2) \) midpoint, so that once it changes its nearest neighbour to \( b_2, \) it will contribute \( \frac{1}{k}(b_2 - b_1) = v \) to the next translation.

sequence. To facilitate this transition between Linear Shifter sequences, we introduce the following widget.

**Definition 2** (Redirector and Booster). The Redirector is a widget \((A, B)\) with two parameters: a shift \( v \) and a threshold \( y. \) Once the cumulative translation \( t_i \) exceeds \( y, \) the Redirector contributes an additional \( v \) to the next cumulative translation. It achieves this by ensuring that a point \( a \) changes its nearest neighbour to one that is positioned at \( kv \) from its initial neighbour. More precisely, the Redirector consists of the following two regions: region 1 consists of \( A_1 = \{a = \frac{1}{2}(b_1 + b_2) - y\} \) and \( B_1 = \{b_1, b_2 = b_1 + kv\}, \) whereas region 2 consists of \( A_2 = \{a'\} \) and \( B_2 = \{b' = a' + \frac{1}{2}kv - y\}. \)

For this one-dimensional construction, we will use a simplification of this widget that we refer to as the Booster. It differs from the Redirector by Arthur and Vassilvitskii insofar as it omits region 2 entirely. The purpose of region 2 is to balance out the initial pull of point \( a \) towards \( b_1, \) so that the Redirector contributes nothing to the first translation \( t_1. \) This turns out to be unnecessary because the Starter compensates for this pull either way, and always sets \( t_1 \) to 1. We consider the name Booster somewhat more instructive for the widget’s purpose in the one-dimensional construction, as it will be used to give the cumulative translation the necessary boost to ensure that, after one point \( a_i \) has completed the Linear Shifter sequence, the point \( a_{i+1} \) to its left will translate to the correct position to begin its own traversal of the Linear Shifter. See Figure 13 for an illustration of the Booster for the one-dimensional construction. We will later also use the Redirector for a construction in higher dimensions, shown in Figure 17.

**Lemma 3.** Suppose the ICP algorithm is run on an ICP configuration containing the Redirector/Booster. Once the cumulative translation \( t_i \) satisfies \( t_i \geq v, \) then the Redirector/Booster contributes \( v \) to the next relative translation \( \Delta t_{i+1}. \)

**Proof.** The result follows from Lemma 1 and the fact that point \( a \) is initially placed \( y \) to the left of the \((b_1, b_2)\) midpoint. Initially, \( N_B(a) = b_1 \) and once point \( a \) is translated to the right by \( y, \) it changes its nearest neighbour to \( b_2. \) This change triggers a contribution of \( \frac{1}{k}(b_2 - b_1) = v \) to the next translation. \( \square \)

The original formulation in [1] claims that the Redirector contributes nothing to the relative translation of any iteration except the one in which it triggers. This is techni-
cally incorrect, or might at least be somewhat misleading, as it suggests that an ICP configuration that contains the Redirector leads to the exact same sequence of relative translations as a configuration that does not contain the Redirector but is otherwise identical. For the first iteration with $t_{i-1} = t_0 = 0$ the “pull”
\[
\frac{1}{|A|} \left( (b_1 - (a + t_{i-1})) + (b' - (a' + t_{i-1})) \right) \\
= \frac{1}{|A|} \left( 0 - \left( \frac{1}{2} k v - y + t_{i-1} \right) + \left( a' + \frac{1}{2} k v - y - (a' + t_{i-1}) \right) \right) \\
= \frac{1}{|A|} \left( -\frac{1}{2} k v + y - t_{i-1} + \frac{1}{2} k v - y - t_{i-1} \right) \\
= -\frac{2 t_{i-1}}{|A|}
\]
of point $a$ and that of point $a'$ to their respective nearest neighbour balance out to $-2 t_0 / |A| = 0$ but in later iterations this is no longer the case. What holds true is that the points of the Redirector (and Booster) experience no nearest-neighbour changes except in the iteration in which the widget triggers, and since the proposed constructions are designed to control the translation by triggering specific nearest-neighbour changes (using Lemma 1), this minor inaccuracy does not affect the overall arguments.

6.5 Full construction and lower bound

Now that we have introduced all the necessary components, we can proceed to assemble them into the complete ICP construction $(A, B)$, on which the ICP algorithm will require $\Omega(n^2)$ iterations to converge. As remarked previously, whenever we place a new region in the construction, it is important that we ensure sufficient $(a, b)$-distance ($> 3\delta$) between regions to maintain the region-decomposition property. The finalised construction will satisfy $|A| = 2n + 2$, and this is the value $k = |A|$ for all widgets that get added in the assembly process, which goes as follows.

1. We begin with the Linear Shifter with $n$ points.

2. Let $\ell = \sum_{i=0}^{n} \frac{1}{2^i}$ be the position of a point $a \in A$ after it has completed the Linear Shifter sequence. The Linear Shifter has only $n + 1$ points in its $B$ but if we were to add a point $b_{n+2}$, according to the definition of $b_1$ in section 5.2, it would satisfy $b_{n+2} = \ell$. We now augment the set $A$ of the Linear Shifter with additional $n$ points by setting it to $A = \{a_0, ..., a_n \mid a_i = -2i\ell\}$. Note that, while for $B$ we have $b_i > b_{i-1}$, for $A$ we now have $a_i < a_{i-1}$.

3. Next we add a Booster with $y_i = (2i + 1)\ell$ and $v = \ell + 1$ for each $i \in \{0, ..., n - 1\}$. Each Booster will make sure that once point $a_i$ has completed its Linear Shifter sequence, the next point $a_{i+1}$ will begin its own such sequence.

4. Finally, we add the Starter to ensure $t_1 = 1$.

See Figure 14 for an illustration of the complete construction. Using this construction we can prove the following lower bound.
Theorem 4. There exist point sets $A, B \subset \mathbb{R}$ with $|A|, |B| = O(n)$ for which the ICP algorithm requires $\Omega(n^2)$ iterations.

Proof. Suppose we run ICP on the aforementioned construction. Initially, all Boosters are untriggered and each point $a_i$ of the Linear Shifter has $b_0$ assigned as its nearest neighbour. The Starter ensures $t_1 = 1$, which will cause $a_0$ to go through the Linear Shifter sequence. Whenever point $a_i$ finishes its Linear Shifter sequence after $n + 1$ iterations, it will have been translated to position $\ell$ by a cumulative translation of $t_n = (2i + 1)\ell$, and the next point $a_{i+1}$ will be at position $a_{i+1} + (2i + 1)\ell = -2(i + 1)\ell + (2i + 1)\ell = -\ell$ with $N_B(a_{i+1} + t_n) = b_0$. The current total cumulative translation of $(2i + 1)\ell$ meets the threshold $y_i$ for Booster $i$ to trigger and contribute its shift $v = \ell + 1$ that moves $a_{i+1}$ to $b_1 = 1$ and thus starts another Linear Shifter sequence that takes $n + 1$ iterations. This process repeats until all Boosters have triggered and all $n + 1$ points $a_i$ of the Linear Shifter have completed their Linear Shifter sequence. The algorithm then terminates after $(n + 1)^2 + 1 \in \Omega(n^2)$ iterations.

Figure 15 shows the relative translations $\Delta t_i$ at each iteration for $n = 4$. An important practical side note is that this construction relies on increasingly small distances between points $b_i$ and $b_{i-1}$ of the Linear Shifter that in practice would quickly lead to floating point precision issues. Let $F_1 = 2^{-1022}$ denote the smallest positive normalized IEEE-754 double precision (64) floating point number, and let $F_2 = 2^{-52} \cdot 2^{-1022}$ denote the smallest positive subnormal IEEE-754 double precision (64) floating point number. It is straightforward to verify that the smallest distance between points $b_n - b_{n-1} = \frac{1}{2^{n+1}} = (2n + 2)^{1-n}$ quickly becomes too small to be represented in standard software using IEEE-754 floating point arithmetic, namely for $n \geq 129$ in the case of $F_1$, and for $n \geq 134$ in the case of $F_2$. Unless a setup using arbitrary precision arithmetic is used, the specific conditions that induce $\Omega(n^2)$ running time in the demonstrated construction will not occur in applications of nontrivial problem size.

7 Lower bound in higher dimensions

With the widgets we introduced and the understanding gained from using them to produce the $\Omega(n^2)$ lower bound construction in the previous section, we can now build on this one-dimensional configuration to inductively prove the following exponential lower bound for point configurations in $d \geq 1$ dimensions.
7.1 Resetting ICP

Figure 15: Relative translations. We show $\Delta t_i$ for each iteration for a complete ICP run on the quadratic lower bound construction for $n = 4$. Note the logarithmic scale on the $\Delta t_i$ axis. Starting with $\Delta t_1 = 1$, as ensured by the Starter, point $a_0$ goes through the Linear Shifter sequence in $n+1 = 5$ iterations. Booster 0 triggers in iteration 6, starting the sequence for the next point of $A$. The algorithm terminates in iteration 26 when all points $a_0, ..., a_n$ have moved through the Linear Shifter sequence.

Theorem 5. There exist point sets $A, B \subset \mathbb{R}^d$ with $|A|, |B| = O(n)$ for which the ICP algorithm requires $\Omega(n/d)^{d+1}$ iterations.

Suppose that after a full ICP execution we could “reset” the position of $A$, and thus force ICP to go through the same sequence of translations a second time, doubling its run time. In this case we could repeatedly apply such a reset to achieve even greater run times. Recall that theorem 1 implies that a previous position of $A$, once abandoned for a more optimal one, cannot be recovered later in the same ICP execution. Geometrically, we can interpret this with the observation by Ezra, Sharir, and Efrat [3] that “the polygonal path $\pi$, obtained by concatenating all the relative translations that are computed during the execution of the algorithm, does not intersect itself.” While this rules out the possibility of the exact reset described just now, it does allow for a variation in which the $A$ is reset in $d-1$ dimensions but shifted to a new position in dimension $d$. This is the idea upon which Arthur and Vassilvitskii [1] build their proof for theorem 5, which we will now discuss.

7.1 Resetting ICP

Let $A, B \subset \mathbb{R}^{d-1}$ be two point sets of an ICP configuration on which the algorithm takes $T$ iterations. The idea is to lift this $(d-1)$-dimensional configuration to $\mathbb{R}^d$ and to augment it with $O(n/d)$ points that repeatedly reset the position of $A$ in all dimensions $< d$, resulting in a running time of $T n^2 / d$ iterations. Performing this augmentation repeatedly on the one-dimensional $\Omega(n^2)$ construction yields the theorem.

We denote $\mathbb{R}^{d-1} \times \{0\}$ as the “base space,” and $\{0, ..., 0\} \times \mathbb{R}$ as the “lift dimension.” A point or vector $t \in \mathbb{R}^{d-1}$ lies in the base space whereas $\vec{t} \in \mathbb{R}^d$ describes the same point or vector embedded in a higher dimensional space. When specifying vectors in terms of their components, we use the two-component notation $(x, y)$ to represent the concatenation of the $(d-1)$-dimensional base space translation vector $x$ with the coordinate $y$ in the lift dimension $d$.

The aim is to construct a Reset Widget that, once ICP has gone through $T$ iterations on the $(d-1)$-dimensional configuration, triggers a translation of the points of $A$ up into
7.1 Resetting ICP

dimension \(d\), and that also resets their position in all \(d-1\) lower dimensions to the initial state. This will lead to another \(T\) iterations in which the ICP algorithm repeats the base space movement of the first \(T\) iterations, merely offset in dimension \(d\). Figure 16 visualises the polygonal translation path \(\pi\) behind this reset and the following \(T\) iterations as a S-shaped movement.

To begin our construction, we embed \((A, B)\) from the base space into \(\mathbb{R}^d\) with lift coordinate 0 to get the lifted ICP configuration \((\overline{A}, \overline{B})\). Since this changes nothing about the relative positions of points, ICP clearly still requires \(T\) iterations on this lifted configuration, with \(\overline{\Delta t}_i = (\Delta t_i, 0)\), and thus also \(\overline{t}_i = (t_i, 0)\), for all \(i \in \{1, \ldots T\}\).

Suppose now that we could augment \((\overline{A}, \overline{B})\) with a Reset Widget that only in iteration \(T+1\) contributes \((-t_T, H)\) to the relative translation \(\overline{\Delta t}_{T+1}\), translating \(\overline{A}\) by \(-t_T\) in the base space (effectively negating all movement from the first \(T\) iterations), and by \(H\) in the lift dimension. The widget contributes nothing to the relative translation of other iterations.

As ICP is run on the augmented \((\overline{A}, \overline{B})\) that includes this Reset Widget, the points of \(\overline{A}\) will have been translated by \(\overline{t}_T = (t_T, 0)\) at the beginning of iteration \(T+1\). This triggers the widget to contribute its shift for a combined cumulative translation of \(\overline{t}_{T+1} = (0, H)\). The nearest-neighbour correspondences at the beginning of iteration \(T+2\) are then identical to those in the first iteration, since the points of \(\overline{A}\) initially had a lift coordinate of 0 before all being translated equally along the lift dimension. All points of \(\overline{B}\) of course stayed at rest, with lift coordinate 0. This results in \(\overline{\Delta t}_{T+2} = (\Delta t_i, 0)\), and subsequently \(\overline{\Delta t}_{T+1+i} = (\Delta t_i, 0)\) for \(i \geq 2\), so that ICP now takes \(2T+1\) iterations on this augmented reset configuration.

If we can find a widget with these properties, this will enable a doubling of the number of iterations required by an ICP configuration \((A, B)\), and the repeated use of this mechanism will allow us to prove theorem 5.

---

**Figure 16: The idea behind the Reset Widget.** Once the cumulative translation reaches the threshold \(t_T\) in the \((d-1)\)-dimensional base space, \(\overline{A}\) gets translated up \(H\) in dimension \(d\) and is reset by \(-t_T\) to its initial position in the base space. Since all points in \(\overline{A}\) and \(\overline{B}\) initially have lift coordinate 0, this reset recovers the nearest-neighbour correspondences from the first iteration between all points except those of the Reset Widget itself. Thus, \(T\) more iterations follow in which \(\overline{A}\) gets translated further by \((t_T, 0)\).
7.2 The Reset Widget

To construct such a Reset Widget, we will use the Redirector described in section 6.4. Recall that a Redirector triggers once its point $a$ changes its nearest neighbour from $b_1$ to $b_2$. This happens once the cumulative translation $t_i$ exceeds a threshold $y$, which is the case once $t_i$ satisfies $t_i \cdot v \geq y \cdot v$. The difference $kv = b_2 - b_1$ between the new and the previous nearest neighbour of $a$, averaged over $k = |A|$, is then contributed to the following iteration’s translation. The previously described behaviour of the Reset Widget would require point $a$ of its Redirector to change its nearest neighbour from $b_1$ to $b_2$ and trigger a shift of $v = (-t_T, H)$ once $A$ has been translated by a threshold $y = t_T$ in the base space. As it turns out, for a single Redirector configured with the required shift $v$ and threshold $y$, we would have $b_2 = b_1 + k(-t_T, H)$ and $a = \frac{1}{2}(b_1 + b_2) - (t_T, 0)$, and so $a$ would start out with $b_2$ as its nearest neighbour instead of the desired $b_1$. It would also be impossible for $a$ to be closer to $b_2$ than to $b_1$ after moving by $(t_T, 0)$. The corollary discussed in section 3.4 that states “In any dimension $d \geq 1$, the angle between any two consecutive edges of $\pi$ is obtuse.” [3] implies that an iteration with relative translation $\hat{t}_i = (-t_T, H)$ cannot directly follow an iteration with $\hat{t}_{i-1} = (t_{i-1}, 0)$ for positive $t_{i-1}$. While this prevents us from achieving the previously outlined reset within a single iteration, we can instead implement the Reset Widget using three Redirector widgets that trigger, one at a time, in three consecutive iterations. Figure 17 conveys the general shape of these Redirector regions for the two-dimensional case. Assume there exists a vector $v_0$ such that $t_i \cdot v_0 < t_T \cdot v_0$ for all iterations $i < T$. For the one-dimensional base case this holds for $v_0 = 1$. We construct the three Redirectors that make up the Reset Widget as follows.

Augment $(\overline{A}, \overline{B})$ with Redirector 1 with shift $v = (v_0, H)$ and threshold $y = (t_T, 0)$. It follows from our assumption about $v_0$ that $(t_i, 0) \cdot (v_0, H) < (t_T, 0) \cdot (v_0, H)$ for all $i < T$, and so Redirector 1 will first trigger in iteration $T + 1$. Ignoring the lift dimension shift of $H$, the base space shift $v_0$ induced by this Redirector could lead to changes in the nearest neighbour correspondences of $(\overline{A}, \overline{B})$, which would result in an additional base space shift $\Delta t'_{T+2}$ in the following iteration $T + 2$. Note that $\Delta t'_{T+2}$ denotes solely the part of the relative base space translation that is due to nearest-neighbour changes in iteration $T + 1$. The next Redirector we add will negate the cumulative translation $t'_{T+2} = t_{T+1} + \Delta t'_{T+2}$ to reset the base space position of $A$.

Augment $(\overline{A}, \overline{B})$ with Redirector 2 with $v = (-t'_{T+2}, H)$ and $y = (t_T + v_0, H)$. For sufficiently large $H$ we have

\[
(t_i, 0) \cdot v < y \cdot v
\]
\[
\Rightarrow (t_i, 0) \cdot (-t'_{T+2}, H) < (t_T + v_0, H) \cdot (-t'_{T+2}, H)
\]
\[
\Rightarrow -t_i t'_{T+2} < -t'_{T+2}(t_T + v_0) + H^2
\]
\[
\Rightarrow t'_{T+2}(t_T + v_0 - t_i) < H^2
\]

for all $i < T + 1$, and so Redirector 2 will trigger in iteration $T + 2$ and reset the cumulative translation to $\overline{t}_{T+2} = (0, 2H)$. At the end of iteration $T + 2$ (and the start of $T + 3$), all points of $(\overline{A}, \overline{B})$, except those in the Reset Widget, will have the same nearest-neighbour correspondences as they did initially (before any translation). Finally, we restart the base space movement with a third Redirector.
7.2 The Reset Widget

Figure 17: Reset Widget. Once point \( a \) of Redirector 1 has been translated by threshold \( (t_T, 0) \), it changes its nearest neighbour from \( b_1 \) to \( b_2 \), triggering a contribution of shift \( (v, H) \) to the next relative translation \( \Delta t_{T+1} \). This proceeds analogously for the other two Redirectors. The three of them trigger in sequence, for a total contribution of \( (t_1 - t_T, 3H) \) to the translation. When augmenting an ICP configuration with the Reset Widget, its six regions are positioned such that the region-decomposition property 6.1 is maintained.
Augment \((\overline{A}, \overline{B})\) with Redirector 3 with \(v = (t_1, H) = (\Delta t_1, H)\) and \(y = (0, 2H)\). This third Redirector will trigger in iteration \(T + 3\). Changing the cumulative base space translation from 0 to \(t_1\) causes the same changes in nearest-neighbour correspondences as \(t_1\) did in the first iteration, and thus initiates a repeat of the translation sequence of the initial \(T\) iterations in the base space, with \(\overline{A}t_{T+2+i} = (\Delta t_i, 0)\) for each \(2 \leq i \leq T\).

To ensure that no adverse translation in the lift dimension interferes with the following \(\Omega(n^2)\) iterations our Reset Widget has initiated, let us for check how the translation in the lift dimension unfolds next. Recall that each of the three Redirectors has two regions, each of which contributes one point to \(\overline{A}\) for a total contribution of 6 points to \(|\overline{A}|\). This leaves \(k - 6\) points in \(\overline{A}\) that are not part of the Reset Widget.

For the following considerations it is helpful to consult Figure 17 for an overview of the Reset Widget’s structure. At the beginning of iteration \(T + 4\), all points of \(\overline{A}\) will have moved by \(3H\) in the lift dimension, and since their respective nearest neighbours in \(\overline{B}\) still have lift coordinate 0, each of the \((k - 6)\) points of \(\overline{A}\) contribute a “pull” of \(3H\) “down” in the lift dimension towards their respective nearest neighbour. This is countered by the six points in \(\overline{A}\) that are part of the Reset Widget, each of which continues to “push up” in the lift dimension towards its respective nearest neighbour. Recall that each Redirector’s point \(a\) is initially assigned to its \(b_i\), but at iteration \(T + 4\) all three Redirectors have triggered because their respective point \(a\) changed its nearest neighbour to \(b_2\).

- Before Redirector 1 triggers in iteration \(T + 1\), both its points \(a\) and \(a’\) are at a lift dimension distance of \(kH/2\) from points \(b_2\) and \(b’\), respectively. At the beginning of iteration \(T + 4\), points \(b_2\) and \(b’\) have become the respective nearest neighbour of \(a\) and \(a’\), which will by then have moved up by a total lift translation of \(3H\) so that the lift dimension distances have been reduced to \((k/2 - 3)H\) each.

- Points \(a\) and \(a’\) of Redirector 2 are initially at \((k/2 + 1)H\) and \((k/2 - 1)H\) lift distance from their respective nearest neighbour \(b_2\) and \(b’\). Moving by \(3H\) reduces these distances to \((k/2 - 2)H\) and \((k/2 - 4)H\) at the beginning of iteration \(T + 4\).

- Similarly, points \(a\) and \(a’\) of Redirector 3 start out at lift dimension distances of \((k/2 + 2)H\) and \((k/2 - 2)H\), and are translated by \(3H\), for reduced lift distances of \((k/2 - 1)H\) and \((k/2 - 3)H\) from \(b_2\) and \(b’\) by the start of iteration \(T + 4\).

Together with the \(k - 6\) points that are not part of the Reset Widget, and are each pulling along a vector of \(-3H\) in the lift dimension towards their nearest neighbour, the sum of lift dimension components of all vectors \(N_B(a + t_{T+3}) - (a + t_{T+3})\) is

\[
(\frac{k - 6}{2})(-3H) = \left(18 - 3k + \frac{6k}{2} - 18\right)H = 0.
\]
7.3 Proving the reset theorem

Proof. We prove theorem 5 inductively by repeatedly augmenting (A, B) with Reset Widgets. This augmentation requires the existence of a vector $v_0$ such that $t_i \cdot v_0 < t_T \cdot v_0$ for all $i < T$. As mentioned before, this holds for $v_0 = 1$ in the one-dimensional base case. Suppose ICP takes $T'$ iterations on an augmented ICP configuration $(\mathcal{A}', \mathcal{B})$ for which such a $v_0$ exists, and let $\overline{t_{T'}}$ denote the final cumulative translation when ICP is run on $(\mathcal{A}', \mathcal{B})$. Then $\overline{t_i} \cdot (v_0, H) < \overline{t_{T'}} \cdot (v_0, H)$ for all $i < T'$. Therefore, after augmenting an ICP configuration with a Reset Widget, we can augment it again in arbitrarily many lift dimensions to obtain a lower bound of $\Omega(2^{d-1}n^2)$. This follows from the fact that adding one Reset Widget in each of the $d - 1$ possible lift dimensions doubles the number of iterations $d - 1$ times. Instead of merely using a single Reset Widget per lift dimension, we will add $\Omega(n/d)$ Reset Widgets per lift dimension.

This implies a lift dimension component of 0 for $\Delta t_{T+4}$ and is consistent with the momentum interpretation from Lemma 1, since in iteration $T + 3$ the only nearest-neighbour changes occur among points with lift component 0, i.e. points other than those of the Reset Widget. Since all three Redirectors of the Reset Widget remain triggered (because their point $a$ keeps $b_2$ as its nearest neighbour), we arrive at $\overline{t_{T+3+i}} = \overline{t_i} = (t_i, 0)$ for $i \geq 2$, and thus repeat the base space movement of the first $T$ iterations. This shows that the Reset Widget successfully resets the ICP configuration to increase the number of iterations from $T$ to $2T + 2$. Figure 18 conveys the translation path of a single reset.

Figure 18: Reset path. Redirector $i$ of the Reset Widget triggers in iteration $T + i$ for $1 \leq i \leq 3$. The translation path of a single reset describes an S-curve.

Figure 19: Reset in $d$ dimensions. Example for $d = 3$ with four Reset Widgets in each lift dimension. $H$ is not to scale.
In a single lift dimension we can add \( m \) Reset Widgets that trigger in sequence to reset \((\mathcal{A}, \mathcal{B})\) but will not reset each other. We achieve this by configuring widget \( i \) to trigger when the global translation reaches a lift coordinate of \( 3H \). Doing so yields a running time of at least \( mT \) iterations for the augmented configuration. Figure 19 illustrates the polygonal translation path \( \pi \) for \( d = 3 \) and \( m = 4 \) in the case that we perform \( m \) resets in each lift dimension. By choosing \( m = \Theta(n/d) \) we induce \( \Omega(n/d) \) resets in each of the \( d - 1 \) lift dimensions. Adding \( \Theta(n/d) \cdot (d - 1) \) Reset Widgets maintains \(|\mathcal{A}|, |\mathcal{B}| \in \mathcal{O}(n)\), and so the total running time becomes

\[
\Omega(n^2) \cdot \frac{n}{d} \cdot \ldots \cdot \frac{n}{d} = \Omega(n^{d+1} \frac{d}{d-1}) = \Omega(n/d)^{d+1},
\]

from which theorem 5 follows immediately.

We conclude the main part of this thesis dedicated to complexity bounds with this result that significantly tightens the gap between the lower bound and the upper bound from section 4 and offer a brief look at the effects of initialisation on convergence next.

![Convergence diagrams](image-url)
8 Exploratory analysis with convergence diagrams

Whether the ICP algorithm converges to a local or global minimum depends on the initial position of $A$ relative to $B$. In this final section we propose and demonstrate a visualisation method for exploratory analysis that may aid further study of these initialisation effects. The following ICP convergence diagrams for $d = 2$ assume that $A$ is a shifted copy of $B$, that is, $a_i = b_i + t_0$, where $t_0 \in \mathbb{R}^d$ is an offset. Moreover, we reduce $B$ by its mean so that the centroid $b_0 = \frac{1}{n} \sum_{b \in B} b$ lies on the origin. For each initial offset $t_0$ on a point grid of arbitrary resolution, we run the ICP algorithm on the configuration $(A + t_0, B)$. The resulting convergence diagram is a pixel grid of the chosen resolution, where each pixel with centre $t_0$ is drawn in a colour that represents the final value of the cost function that ICP converges on. This means that each offset (and pixel centre) $t_0$ represents the centroid $a_0$ of $A$ at the start of the first ICP iteration. Each region drawn in a single colour on the diagram depicts initial offsets $t_0$ from which the algorithm’s cost function converges to the same value. It is notable that these regions are often quite irregular and do not follow an immediately obvious pattern, nor do they seem to display a clear visual connection to the Voronoi diagram of $B$. In Figure 20 we show four examples for arbitrary point clouds $B$ that were generated at random from a normal distribution. A higher resolution does not generally coincide with a more gradual transition of colours (i.e. final cost values) even when $n$ is increased considerably above the values depicted here.

Some point clouds $B$ result in highly irregular and noisy diagrams, as depicted in Figure 21. In these cases, even the largest cost values (drawn as a red pixel) tends to be miniscule, which suggests that these patterns can be explained by floating point imprecision. These noisy diagrams occur more frequently for higher values of $n$.

Figure 22 shows that already for the simple case $n = 2$, a seemingly minor change of $t_0$ can lead to substantial differences in the convergence behaviour even when the initial nearest-neighbour correspondences are identical for both initialisations. It is therefore unclear whether a (non-trivial) initial offset $t_0$ that guarantees convergence to a global minimum can be determined visually.

Figure 21: Precision noise. For $A = B + t_0$, higher values of $n$ increase the likelihood of reaching a global minimum, and that of noisy diagrams. Even red pixels correspond to very small cost values here.
Even though these diagrams did unfortunately not lead us to immediate, generalisable insights about the dynamics of the ICP algorithm, we include them in this thesis in the hope that they be of interest, and potentially influence further visual exploration of the ICP algorithm’s geometric properties and behaviour.

9 Discussion and open problems

In this thesis we have explored convergence characteristics of the ICP algorithm, with a primary focus on upper and lower bound constructions. The polynomial bounds shown here seem somewhat at odds with the considerable popularity of the ICP algorithm in applications that are both data-intensive and performance-critical. This leads Arthur and Vassilvitskii to ask “What can be said when an algorithm is known
to be fast in practice but slow in the worst case?” [1] In their effort to reconcile their \( \Omega(n/d)^{d+1} \) lower bound with the observed speed of ICP, they perform a smoothed analysis following the work by Spielman and Teng [8], and achieve a smoothed complexity of

\[
O \left( |A|^3 |B|^8 \cdot d \left( \frac{D}{\sigma} \right)^2 p^{-2/d} \right)
\]

with probability at least \( 1 - 2p \), when all points of \( A \) and \( B \) have been independently perturbed by a \( d \)-dimensional normal distribution with variance \( \sigma^2 \), and \( D \) is the maximum diameter of these sets \( A \) and \( B \). While this smoothed complexity is polynomial independent of the dimensionality \( d \) of the data, Arthur and Vassilvitskii remark that the large exponent “could use improvement.” This is true especially in light of the fact that many applications of the ICP algorithm use low-dimensional data for which the non-smoothed upper bound is lower than the smoothed one. The arguments and constructions presented here examine a simplified version of the original ICP algorithm. In addition to finding the translation that aligns \( A \) optimally to \( B \), the full ICP algorithm and its numerous variations also take into account rotation. Based on the observations in section 4 that the number of ICP iterations is bounded from above by the number of possible nearest-neighbour assignments, we would expect the addition of \( d - 1 \) degrees of freedom (in the form of rotation) to raise the upper bound in the worst case noticeably. Despite its deceptive simplicity, the ICP algorithm displays many fascinating and subtle properties, and this thesis discussed merely a fraction of the known results. We hope these explanations may illuminate and clarify what we know, and inspire further study of both its complexity and geometric properties.
References


A. Errata

The following excerpts contain minor inaccuracies we found in [3]. They are remedied in section 5. The $\Omega(n \log n)$ lower bound proof defines round $j$ and the properties used for induction as follows.

Assume that the points of $\mathcal{A}$, except for the leftmost one, are assigned to $b_{n-j+1}$ and $b_{n-j+2}$, for some $1 \leq j \leq n \ldots$, and consider all iterations of the algorithm, in which some points of $\mathcal{A}$ cross the common Voronoi boundary $\beta_{n-j+1}$ of the cells $V(b_{n-j+1}), V(b_{n-j+2})$. We call the sequence of these iterations round $j$ of the algorithm. Then, \ldots (iii) at the last iteration of the round, the overall number of points of $\mathcal{A}$ that cross either $\beta_{n-j+1}$ or $\beta_{n-j+2}$ is exactly $j - 1$.

This definition of round $j$ is equivalent to the one we give except for the minor detail that for $j = 1$ the point $b_{n-j+2} = b_{n+1}$ does not exist. Instead of $1 \leq j \leq n$ the authors likely meant to write $2 \leq j \leq n$. More importantly however, their property (iii) fails to take into account what we refer to as case (c) in section 5, where in the last iteration of round $j$, exactly $j$ rather than $j - 1$ points of $\mathcal{A}$ cross a boundary. Refer to iteration $i = 7$ shown in Figure 9 for an example case in which the authors’ property (iii) does not hold. We introduced the notion of a transition iteration between such rounds to cover this case and ensure the inductive step holds for all possible cases.

Let us now consider the last such iteration [of round $j$]. In this case, all the points of $\mathcal{A}$, except $\ell$ of them, for some $0 \leq \ell < j$ (and the leftmost point, which we ignore), have crossed $\beta_{n-j+1}$ in previous iterations.

The case $0 = \ell$ would contradict the fact that the iteration in question is the last iteration of round $j$, and since $\ell = j$ does occur in case (c) the inequality in the excerpt above needs to be corrected to $0 < \ell \leq j$.

It now follows, using the above properties, that the number of iterations for all the points of $\mathcal{A}$ to cross $\beta_{n-j+1}$ is $\lceil \frac{n}{j} \rceil$, where in the first (last) such iteration some of the points may cross $\beta_{n-j} (\beta_{n-j+2})$ as well.

The fact that $\beta_{n-1}$ is crossed in a single iteration ($i = 2$) serves as a counterexample to the first claim in this excerpt, given that $\left\lceil \frac{n}{n-1} \right\rceil \neq 1$ for $n \in \mathbb{N}$.

A small mistake in the following excerpt from [1] is remedied in section 6.2.

Define $B = \{b_0, b_1, \ldots, b_m\}$ by setting $b_0 = 0$ and $b_i = 1 + \frac{1}{k} + \cdots + \frac{1}{k^i}$ for $i > 0$.

This definition for point set $B$ of the Linear Shifter accidentally adds a term of 1 that implies $b_1 = 2$, and so the initial translation of $t_1 = 1$, induced by what we refer to as the Starter in section 6.3, does not lead to the desired property that $a + t_1 = b_1$. Omitting this initial term of 1 and defining $b_i := \sum_{j=0}^{i-1} \frac{1}{k^j}$ corrects this.

In section 6.4 we remark on a potentially misleading claim in the original formulation of Lemma 3 after the corresponding proof.