Approximate one-to-one point pattern matching

Marc Benkert[a], Joachim Gudmundsson[b], Damian Merrick[a], Thomas Wolle[c,*]

[a] *XP Software, Australia*
[b] *University of Sydney and NICTA, Australia*
[c] *Arclight Investments Pty. Ltd., Sydney, Australia*

Abstract

Given a set $A = \{a_1, \ldots, a_n\}$ of n image points and a set $B = \{b_1, \ldots, b_n\}$ of n model points, the problem is to find a transformation matching (a one-to-one mapping) each point $a \in A$ to some point $b \in B$ such that the length of the longest edge in the matching is minimized (so-called bottleneck distance). The geometric transformations we allow are translation, rotation, reflexion and scaling. In this paper, we give $(1+\varepsilon)$-approximation algorithms for the case when the points are given in $\mathbb{R}^2$, two of which run in $O(\frac{n^{3.5}}{\varepsilon^4} \log n)$ and $O(\frac{n^{2.5}}{\varepsilon^4} \log n \log \frac{diam(B)}{d_{opt}})$ time, respectively, where $diam(B)$ is the diameter of B and d_{opt} is the bottleneck distance in an optimal matching.

Keywords: point pattern matching, similarity transformations, bottleneck distance, approximation algorithm

1. Introduction

The matching and analysis of geometric patterns and shapes is of importance in various application areas, in particular computer vision and pattern recognition, but also in other areas concerned with the form of objects such as GIS and computer animation. A recent application stems from detecting 'formations' among animals, military units or athletes. For example, many birds, including the Canadian Goose, fly in a $\vee$ formation when migrating to save energy [20]. The shape of the formation is believed to depend on wingspan and weather conditions. Similarly, military units use formation movement to preserve security among its units during military operations. Different formations are also used for different purposes; for example, defending, searching or attacking.

A natural approach to detect 'fixed' formations among moving objects is to extract the problem to the well-known problem of detecting a geometric pattern. Let $A = \{a_1, \ldots, a_n\}$ be a set of image points and $B = \{b_1, \ldots, b_n\}$ a set of model points in d dimensions. A matching between A and B is a one-to-one mapping between the points in A and the points in B, and to each matching, we can associate a distance between A and B with respect to this matching (e.g. the maximum distance between any two matched points). The problem is to find a transformation T that transforms A into $T(A)$, such that the distance between $T(A)$ and B is minimised, according to some distance measure. The geometric transformations we allow are translation, rotation, reflexion and scaling; together they are called *similarity transformations*. In this paper we restrict ourselves to the case when A and B are point sets in the plane.

The general problem of mapping one point set into another point set has received a lot of attention both in the computer vision community [19] and computational geometry community [3], and many different distance functions have been proposed. Perhaps the most common is the Hausdorff distance, for which many algorithms have been suggested [2, 10, 11]. When only translation is allowed Huttenlocher et al. [15, 16, 17] find the minimum Hausdorff distance for points in the

*Corresponding author
Email addresses: marc.benkert@xpsoftware.com (Marc Benkert), joachim.gudmundsson@gmail.com (Joachim Gudmundsson), damian merrick@xpsoftware.com (Damian Merrick), thomas.wolle@gmail.com (Thomas Wolle)

plane in $O(n^3 \log n)$ time. Chew et al. [11] gave an $O(n^5 \log n)$ time algorithm when rigid transformation is allowed. *Rigid transformations* are also called rigid motions or isometries and allow translation, rotation and reflexion. This complexity may be unacceptably high for applications involving large point sets. For this reason approximation algorithms have been considered. Alt et al. [1] show a $(1 + \pi/4)$-approximation algorithms for rigid motion and similarity transformation having running time $O(n^2 \log n \log^* n)$. Other approximation algorithms were given by Goodrich et al. [13] who show a 2-approximation algorithm with running time $O(n^2 \log n)$ for translations and a 3-approximation algorithm with running time $O(n^3 \log n)$ for rigid transformations. Despite its popularity, the Hausdorff distance suffers from the drawback that the mapping defined by associating each object in B to its closest neighbour in A is not necessarily a bijection.

The distance measure that we will focus on from now on is the *bottleneck distance* suggested by Alt et al. [4]:

Definition 1. *Let A and B be two point sets in $\mathbb{R}^d$ of the same cardinality. Let M be a one-to-one matching between A and B. By $d(M) = d(M(A, B))$ we denote the maximum (Euclidean) distance between two matched points of M. By M_{opt} we denote the matching that minimizes this distance among all matchings, $d(M_{\text{opt}}(A, B)) = d(A, B)$ is called* bottleneck distance *of A and B.*

In the case when only translation of the set A is allowed, Efrat et al. [12] show that an optimal translation can be computed in time $O(n^5 \log^2 n)$. They also provided a $(1 + \varepsilon)$-approximation algorithm with running time $O(\frac{1}{\varepsilon^4} n^{3/2} \log n \log \frac{1}{\varepsilon})$. For a rigid transformation, Alt et al. [4] gave an $O(n^8)$ time algorithm for the decision problem, i.e. given a positive constant ρ is there a rigid transformation of A such that the bottleneck distance is at most ρ.

For rigid motion Heffernan and Schirra [14] take an alternative approach to reduce the complexity of the decision problem. They only require the decision algorithm to give a correct answer when the given tolerance ε is not too close to the optimal solution, i.e. it has to lie outside the interval $[d_{opt} - \alpha, d_{opt} + \beta]$ for fixed $\alpha, \beta > 0$. Using network flow methods, the resulting algorithm requires $O(n^{2.5})$ time. This algorithm can be modified for the optimisation problem by using binary search on α and β, thus obtaining a $(1 + \varepsilon)$-approximation algorithm with running time $O(n^{2.5} \log \frac{1}{d_{opt}})$ (see also [14]), where d_{opt} is the minimum distance between $T(A)$ and B over all transformations.[1]

A slightly different setting was considered by Arkin et al. [5]. Find a transformation of a set of n points in the plane such that each transformed point lies in one of n given pairwise-disjoint 'noise regions'. This is similar to the above decision problem: given a $\rho > 0$ place a disk of radius ρ with centre at each point in B. Is there a transformation of A such that each transformed point in A lies in one of the regions of B? For equally sized disks, Alt et al. [4] show that the decision version can be solved in $O(n \log n)$ time for translations, later Arkin et al. [5] show an $O(n^4 \log n)$ time algorithm for rigid transformations and an $O(n^5 \log n)$ time algorithm for similarity transformations. An overview of the results for matching using the bottleneck distance can be found in Fig. 1.

Not much is known about the hardness of these types of problems. Deciding whether two n-point sets $A, B \in \mathbb{R}^d$ are congruent is a fundamental problem in geometric pattern matching. If the dimension d is unbounded, the problem is equivalent to graph isomorphism and is conjectured to be in FPT (Fixed-parameter Tractable). When $|A| = m < |B| = n$, the problem becomes that of deciding whether A is congruent to a subset of B and is known to be NP-complete. Cabello et al. [7] show that point subset congruence, with d as a parameter, is $W[1]$-hard, and that it cannot be solved in $O(mn^{o(d)})$-time, unless SNP$\subset$DTIME$(2^{o(n)})$. This shows that, unless FPT$= W[1]$, the problem of finding a rigid transformation of A that minimizes its distance to B, is not in FPT.

In this paper, we present a $(1 + \varepsilon)$-approximation algorithm for a similarity transformation T that transforms A into $T(A)$ such that $d(T(A), B)$ is minimised, where we use the bottleneck

[1] A reviewer suggested that the same technique as in [12] could be used to obtain a running time of $O(n^{2.5} \log n)$. However, it is not immediately clear to the authors that the technique is applicable to this case.

Bottleneck distance			
Translation	$O(n^5 \log^2 n)$	[12]	optimal
	$O(\frac{1}{\varepsilon^4} n^{3/2} \log n \log \frac{1}{\varepsilon})$	[12]	$(1+\varepsilon)$-approximation
	$O(n \log n)$	[4]	pairwise disjoint disks
Rigid	$O(n^{2.5} \frac{\varepsilon}{\lvert d_{opt} - \varepsilon \rvert} \log \frac{1}{d_{opt}})$	[14]	$(1+\varepsilon)$-approximation
	$O(n^8)$	[4]	decision version
	$O(n^4 \log n)$	[5]	pairwise disjoint disks
Similarity	$O(\frac{1}{\varepsilon^4} n^{2.5} \log n \log \frac{1}{d_{opt}})$	This paper	$(1+\varepsilon)$-approximation
	$O(n^5 \log n)$	[5]	pairwise disjoint disks

Table 1: Summarising previous results for point-pattern-matching using the bottleneck distance.

distance as defined above. We will start with a trivial approximation algorithm with run time $O(\frac{n^{5.5}}{\varepsilon^4} \log n)$ (Section 2.1). We then improve this to obtain a second approximation algorithm that runs in $O(\frac{n^{3.5}}{\varepsilon^4} \log n)$ time (Section 2.2). With a further improvement, we obtain our main result of this paper, which is a $(1+\varepsilon)$-approximation algorithm running in $O(\frac{n^{2.5}}{\varepsilon^4} \log n \log \frac{1}{d_{opt}})$ time, where d_{opt} is the minimum distance between $T(A)$ and B over all transformations, and B is scaled to have diameter 1 (Section 3).

2. Approximation algorithms

The problem of matching two point sets is a very common problem and appears in many applications. Existing literature has focussed on finding *exact* solutions with the drawback that the time complexity suffers. Here we will present approximation algorithms that give trade-offs between the quality of the matching and the running time.

Consider an optimal similarity transformation $T_{opt}(A)$ of A and let d_{opt} denote the bottleneck distance between $T_{opt}(A)$ and B. A similarity transformation $T(A)$ is said to be a *k-approximate transformation* if the bottleneck distance between $T(A)$ and B is at most $k \cdot d_{opt}$, and an algorithm that returns a k-approximate similarity transformation for any set A and B is called a *k-approximation algorithm*.

As a tool we will use the algorithm by Efrat et al. [12] that, given two point sets in the plane each containing n points, computes the minimum bottleneck distance between the two sets in $L(n) = O(n^{\frac{3}{2}} \log n)$ time. The main idea of our approach is to test a number of transformations, compute the bottleneck distance for each transformation, and then report the best one. Thus, the running time will be the number of tested transformations multiplied by $L(n)$.

2.1. A first approximation algorithm

As a first approach we test $\binom{n}{2} \cdot \binom{n}{2}$ possible transformations. This works as follows: we select two points of A and two points of B. Then we transform A into $T(A)$ such that, after the transformation, the two chosen points of A coincide with the two chosen points of B. There are $\binom{n}{2} \cdot \binom{n}{2}$ ways to choose the points, each of them specifying a transformation. The transformation that generates the best solution is reported. We call this approach the *naïve* algorithm.

Theorem 1. *The naïve algorithm is a 3-approximation algorithm with running time $O(n^4 \cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

The running time follows trivially from the above description, and the approximation bound follows from the proof of Theorem 2.

The above approximation bound can be improved by adding Steiner points. That is, run the naïve algorithm that finds a 3-approximate transformation with bottleneck distance $d \leq 3 \cdot$

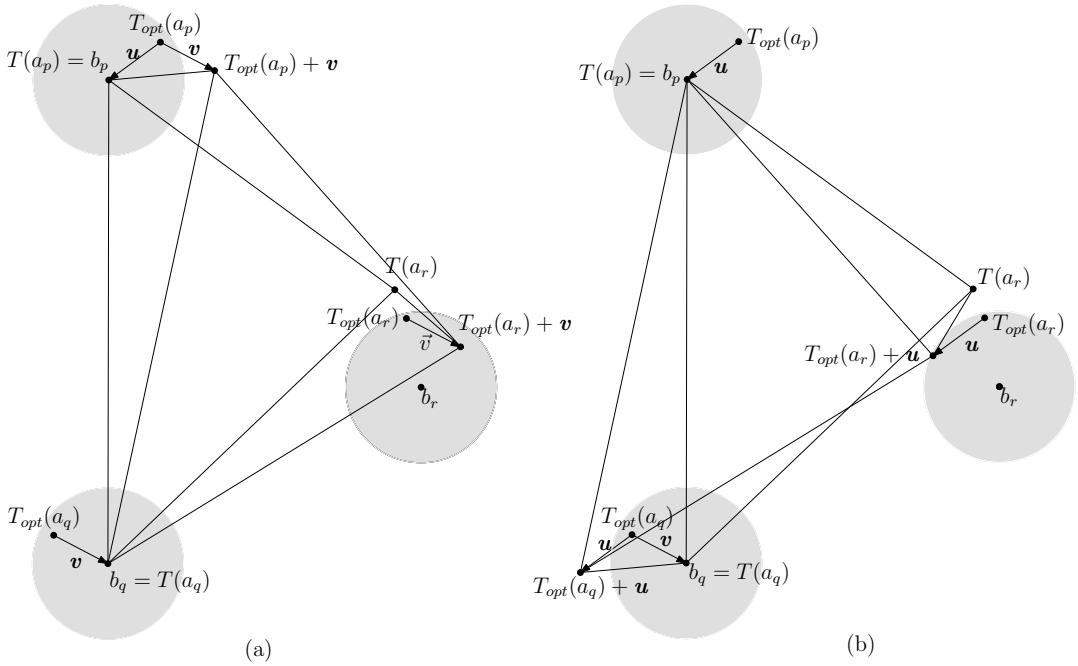

Figure 1: Illustration for the proof of Theorem 2. The grey disks have radius d_{opt}.

d_{opt}. Using the just computed distance d, we place $O(\frac{1}{\varepsilon^2})$ points, for each point $b \in B$, on the intersections of a grid centered at b, where every cell in the grid has side length $\frac{\varepsilon \cdot d}{6}$. The extended set of points containing points in B and the Steiner points is denoted B'. By slightly modifying the naïve algorithm, such that each transformation T takes two points in A and makes them coincide with two points in B' we obtain the following lemma.

Lemma 1. *The modified naïve algorithm is a $(1+\varepsilon)$-approximation algorithm with running time $O(\frac{n^4}{\varepsilon^4} \cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

2.2. A second approximation algorithm

Here we consider an improvement to speed-up the algorithm. Instead of testing *every* pair of points in A with *every* pair of points in B, we will only test *one* pair of points a_p, a_q in A with *every* pair in B. The points a_p, a_q in A are chosen such that they are the *furthest pair* in A, that is the distance between them is the largest distance over all pairs of points in A. As above we report the similarity transformation T with the smallest bottleneck distance. We call this approach the *improved* algorithm. Thus, we get:

Theorem 2. *The improved algorithm is a 3-approximation algorithm with running time $O(n^2 \cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

Proof. The running time follows trivially from the above description, hence we focus on the approximation bound.

Consider an optimal transformation T_{opt} and consider the similarity transformation T reported by the algorithm. Note that the matching between $T(A)$ and B does not have to be identical to the matching between $T_{opt}(A)$ and B. However, to prove the theorem, it suffices to prove that there exists a matching between $T(A)$ and B with small bottleneck distance. Thus, we will now fix the matching between $T(A)$ and B to be the same as the matching between $T_{opt}(A)$ and B; and

let $M(A, B)$ denote this one-to-one matching. Let a_p and a_q be two points in A with maximum distance, and let $b_p = M(a_p)$ and $b_q = M(a_q)$.

Recall that we will consider four point sets in this proof: A, B, $T_{opt}(A)$ and $T(A)$. And from the algorithm we have $b_p = T(a_p)$ and $b_q = T(a_q)$. To prove the theorem it suffices to prove that for any point a_r in A the distance between $T(a_r)$ and $b_r = M(a_r)$ is at most $3 \cdot d_{opt}$.

Because length relations and angle relations are preserved when performing uniform scaling, rotation and translation, it holds that

$$\frac{|T_{opt}(a_r), T_{opt}(a_q)|}{|T_{opt}(a_r), T_{opt}(a_p)|} = \frac{|T(a_r), T(a_q)|}{|T(a_r), T(a_p)|}$$

and

$$\angle(T_{opt}(a_p), T_{opt}(a_q), T_{opt}(a_r)) = \angle(T(a_p), T(a_q), T(a_r)),$$

as illustrated in Fig. 1(a). Therefore, the two triangles

$$\triangle(b_q, b_p, T(a_r)) = \triangle(T(a_q), T(a_p), T(a_r)) \quad \text{and} \quad \triangle(T_{opt}(a_q), T_{opt}(a_p), T_{opt}(a_r))$$

are *similar*. Let $\vec{v} = \overrightarrow{T_{opt}(a_q), b_q}$ and $\vec{u} = \overrightarrow{T_{opt}(a_p), b_p}$, and note that $|\vec{v}| \leq d_{opt}$ and $|\vec{u}| \leq d_{opt}$. If we translate one of the triangles by $\vec{v}$, the similarity is still preserved. Hence, also the triangles

$$\triangle(b_q, b_p, T(a_r)) \quad \text{and} \quad \triangle(T_{opt}(a_q) + \vec{v}, T_{opt}(a_p) + \vec{v}, T_{opt}(a_r) + \vec{v})$$

are similar, see Fig. 1(a). And so are the two triangles

$$\triangle(b_p, b_q, T_{opt}(a_p) + \vec{v}) \quad \text{and} \quad \triangle\,(T(a_r), b_q, T_{opt}(a_r) + \vec{v}),$$

see Fig. 1(a). Note that $T_{opt}(a_q) + \vec{v} = b_q$. From this and the assumption that a_p and a_q have maximum distance in A, we also know that $|T(a_r), T_{opt}(a_r) + \vec{v}| \leq |T(a_p), T_{opt}(a_p) + \vec{v}| = ||\vec{v} - \vec{u}|| \leq 2 \cdot d_{opt}$. See Fig. 1(a) for an illustration.

Consider exactly the same arguments as above, but this time swapping q and p, see Fig. 1(b). Similar to above we can then prove that $|T(a_r), T_{opt}(a_r) + \vec{u}| \leq |T(a_q), T_{opt}(a_q) + \vec{u}| = ||\vec{u} - \vec{v}|| \leq 2 \cdot d_{opt}$. From above we have $|T(a_r), T_{opt}(a_r) + \vec{v}| \leq |T(a_p), T_{opt}(a_p) + \vec{v}| = ||\vec{v} - \vec{u}|| \leq 2 \cdot d_{opt}$. Putting these two inequalities together, we get the scenario as illustrated in Fig. 3(a). Taking all indicated constraints into account, we analyse this scenario to determine how large the distance between $T(a_r)$ and $T_{opt}(a_r)$ can get.

Using the notations in the figure, one can prove (see Appendix A.2 for details) that the distance f between $T(a_r)$ and $T_{opt}(a_r)$ is maximised when $\ell_1 = \ell_2$ and $\ell_3 = \ell_4$, as illustrated in Fig. 3(b). Note that $\ell_1, \ell_2 \leq e = ||\vec{u} - \vec{v}||$. Straight-forward trigonometry gives:

$$|T_{opt}(a_r), x| \leq d_{opt} \cos(\alpha)$$

and since ℓ_1 and ℓ_2 are bounded by $|T_{opt}(a_r) + \vec{u}, T_{opt}(a_r) + \vec{v}| \leq 2 \cdot d_{opt} \cdot \sin\alpha$, we get

$$|T(a_r), x| \leq \sqrt{3} \cdot d_{opt} \sin\alpha.$$

Together, we obtain:

$$\begin{aligned} |T(a_r), T_{opt}(a_r)| &\leq |T(a_r), x| + |x, T_{opt}(a_r)| \\ &\leq \sqrt{3} \cdot d_{opt} \sin\alpha + d_{opt} \cos\alpha \\ &\leq 2 \cdot d_{opt} \end{aligned}$$

The last inequality holds since the maximum is attained for $\alpha = \frac{\pi}{3}$. As a last step, we calculate:

$$|T(a_r), b_r| \leq |T(a_r), T_{opt}(a_r)| + |T_{opt}(a_r), b_r| \leq 2 \cdot d_{opt} + d_{opt} = 3 \cdot d_{opt}$$

□

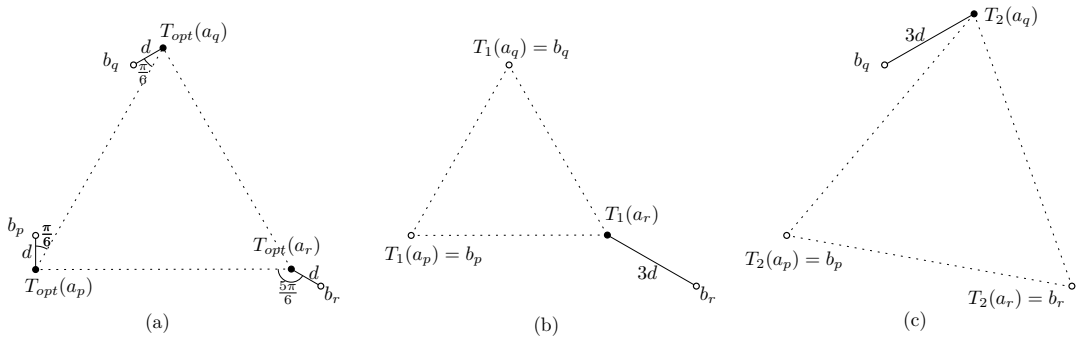

Figure 2: Illustrating the proof of Lemma 2. Points of set A are shown as small black circles, while points of B are small circles with a white centre.

At first it may seems as the improved algorithm performs worse than the naïve algorithm, since much less transformations are considered. However, the next lemma tells us that even though the naïve algorithm performs considerable more work, its solution is not guaranteed to be better than the solution computed by the improved algorithm.

Lemma 2. *There exist two point sets A and B, such that the naïve algorithm reports a 3-approximation.*

Proof. Set A consists of three pairwise equidistant points, $A = \{a_p, a_q, a_r\}$. The positions of the points of set $B = \{b_p, b_q, b_r\}$ are shown in Fig. 2(a), together with an optimal transformation T_{opt} of A. The bottleneck distance between $T_{opt}(A)$ and B is exactly d. Recall that the naïve algorithm will consider many different transformations, keeping track of the best one. In this proof, however, we only consider two of those, because all other transformations are symmetric due to the symmetry inherent in A. These two transformations T_1 and T_2 are depicted in Fig. 2(b) and Fig. 2(c), respectively.

One can consider all the points in $T_{opt}(A) \cup B$ in a coordinate system. Using basic mathematics and trigonometry, one can then compute (see Appendix A.1 for details) the coordinates of the point $T_1(a_r)$ and also the distance between $T_1(a_r)$ and b_r, see Fig. 2(b). This distance is exactly $3 \cdot d$. In a similar, but more complicated computation involving suitable trigonometric identities, one can compute (see Appendix A.1 for details) the coordinates of the point $T_2(a_q)$ and also the distance between $T_2(a_q)$ and b_q, see Fig. 2(c). Also this distance is exactly $3 \cdot d$.

Thus, all transformations T considered by the naïve algorithm give rise to a bottleneck distance of $3d$ between $T(A)$ and B. □

As in the previous section the approximation bound can be improved by adding Steiner points. That is, run the algorithm that finds a 3-approximate transformation with bottleneck distance $d \leq 3 \cdot d_{opt}$. Next, for each point $b \in B$ place $O(\frac{1}{\varepsilon^2})$ points on the intersections of a grid centered at b, where every cell in the grid has side length $\frac{\varepsilon \cdot d}{6}$. The extended set of points containing points in B and the Steiner points is denoted B'. By modifying the above approach, as before, such that each transformation $T(A)$ takes a_q and a_p in A and matches them with two points in B', we obtain the following lemma.

Lemma 3. *The modified improved algorithm is a $(1+\varepsilon)$-approximation algorithm with running time $O(\frac{n^2}{\varepsilon^4} \cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

3. A third approximation algorithm

Above we proved that testing only $O(n^2)$ transformations is sufficient to obtain a good approximation (cf. Theorem 2). In this section we will show how this bound can be improved to

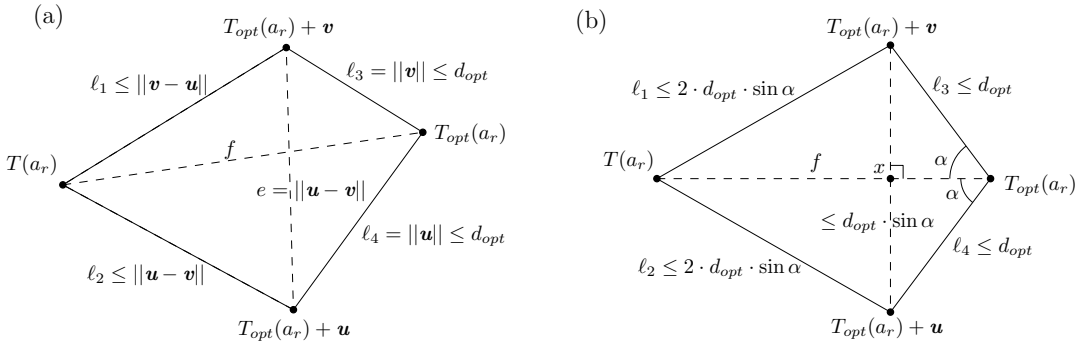

Figure 3: Illustration for the proof of Theorem 2.

$O(\frac{n}{\varepsilon^2} \log_{(1+\varepsilon)} \frac{1}{d_{opt}})$, where $0 < \varepsilon < \frac{1}{10}$ is a given constant. The main approach is as follows. Let b and b' be the furthest pair in B and assume $|b, b'| = 1$. For each point $a \in A$ we identify a' as a furthest point from a, and then we test if there exists a good approximate similarity transformation with the property that a and b coincide, and a' is placed on a few locations 'near' b'. That is, we compute transformations by placing a' on the vertices of a 'circular grid'. Then we iterate the entire process where we construct smaller and finer grids, keeping track of the best transformation until we reach a good approximation.

More formally, for $a \in A$, let $T_{opt}^{a=b}$ be the similarity transformation, such that $d_{opt}^{a=b} := d(T_{opt}^{a=b}(A), B)$ is minimal under the restrictions that $T_{opt}^{a=b}(a) = b$ and b is a point of the furthest pair in B. To see how this relates to $T_{opt}(A)$, we make the following straight-forward observation.

Observation 1.

$$d_{opt} = d(T_{opt}(A), B) \leq \min\{d_{opt}^{a=b} | a \in A\} \leq 2 \cdot d_{opt}$$

From now on, we fix a point $a \in A$, and let a' be a point in A furthest from a. We will show how one can compute a $(1+\varepsilon)$-approximation of $d_{opt}^{a=b} = d(T_{opt}^{a=b}(A), B)$ by only testing $O(\frac{1}{\varepsilon^2} \log_{(1+\varepsilon)} \frac{1}{d_{opt}^{a=b}})$ transformations. Our algorithm starts with **Step 0**, which can be considered as a preprocessing step; having **Step 0** simplifies the algorithm and its analysis.

Step 0: (Handling the case $d_{opt}^{a=b} \geq \frac{1}{10}$.)

Compute a minimum axis-aligned bounding square $\mathcal{S}$ of B in linear time. Note that the side length of $\mathcal{S}$ is at most 1. Construct a $\frac{10\sqrt{2}}{\varepsilon} \times \frac{10\sqrt{2}}{\varepsilon}$ grid G in $\mathcal{S}$, where each cell has side length at most $\frac{\varepsilon}{10\sqrt{2}}$. For each vertex v of the grid G, perform the transformation $T'(A)$ with $T'(a) = b$ and $T'(a') = v$, and compute the bottleneck distance between the two sets. Among all the $O(\frac{1}{\varepsilon^2})$ transformations tested, let T be the one with the smallest bottleneck distance. If $d(T(A), B) > (1+\varepsilon) \cdot \frac{1}{10}$, then $T(A)$ is a $(1+\varepsilon)$-approximation; we report $T(A)$ and terminate the algorithm (see Lemma 4). Otherwise we continue.

Step 1: (All steps from now handle the case $d_{opt}^{a=b} < \frac{1}{10}$.)

We define λ_0 as the annulus with center at b, inner radius $\frac{3}{4}$ and outer radius $\frac{5}{4}$. Recall that the furthest pair in B, b and b', has distance 1. For the ease of presentation, we assume w.l.o.g. that B is such that all points in $B \cap \lambda_0$ lie below and to the right of b. This is always possible, because otherwise there would be a pair of points in B, specifically in $B \cap \lambda_0$, with inter point distance greater than 1. Define Γ_0 to be the sector of λ_0 between the two half lines with origin at b and with slopes $\frac{1}{5}$ and $-\left(\frac{\pi}{2} + \frac{1}{5}\right)$, as shown in Fig. 4(a). Note that $b' \in \Gamma_0$. Initialise variables keeping track of the best distance and the loop counter: $d_{-1} \leftarrow 1$ and $i \leftarrow 0$.

Step 2: Construct a grid in Γ_i based on concentric circles and lines radiating from b, as illustrated in Fig. 4(b), such that we have $O(\frac{1}{\varepsilon^2})$ grid cells with side length at most $\frac{\varepsilon}{32} \cdot d_{i-1}$.

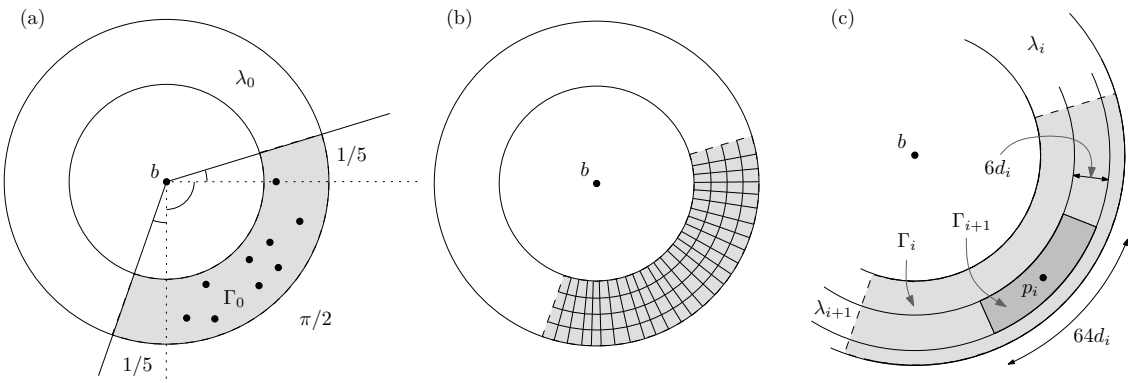

Figure 4: Illustrating the constructions: (a) λ_0, Γ_0 in grey and with centre angle $\frac{\pi}{2} + \frac{2}{5}$, (b) the grid in Γ_i, (c) λ_{i+1} with width $6d_i$ and Γ_{i+1} in dark grey, with centre angle $64d_i$ and p_i in the middle.

Step 3: For each grid vertex v in Γ_i perform a similarity transformation $T'(A)$ such that $T'(a) = b$ and $T'(a') = v$. Calculate the bottleneck distance between B and $T'(A)$. Among all the $O(\frac{1}{\varepsilon^2})$ transformations tested, let T_i be the one with the smallest bottleneck distance. Let d_i be the smallest bottleneck distance, and let p_i be the grid vertex that gave rise to T_i.

Step 4: If $\frac{d_{i-1}}{d_i} \leq (1+\varepsilon)$, then report $T_i(A)$ and terminate the algorithm (see Lemma 6). Otherwise consider the annulus λ_{i+1} with center at b, inner radius $(|b, p_i| - 3 \cdot d_i)$, outer radius $(|b, p_i| + 3 \cdot d_i)$. Set Γ_{i+1} to be the sector of this annulus with centre angle $64 \cdot d_i$ and p_i in the middle of this sector, as shown in Fig. 4(c). Set $i \leftarrow i + 1$ and goto **Step 2**.

We will consider the correctness of the algorithm by proving a series of lemmas. First, we note that the algorithm correctly handles the case where $d_{opt}^{a=b} \geq \frac{1}{10}$.

Lemma 4. *If $d(T(A), B) > (1+\varepsilon) \cdot \frac{1}{10}$ in* **Step 0**, *then $d(T(A), B) \leq d_{opt}^{a=b} \cdot (1+\varepsilon)$.*

Proof. From the definition of T and the grid construction, it follows that $d(T(A), B) \leq d_{opt}^{a=b} + \frac{\varepsilon}{10}$. And with the antecedent of the lemma, we have that $(1+\varepsilon) \cdot \frac{1}{10} < d(T(A), B) \leq d_{opt}^{a=b} + \frac{\varepsilon}{10}$. From this, it follows that $\frac{1}{10} \leq d_{opt}^{a=b}$, which then implies that $d(T(A), B) \leq d_{opt}^{a=b} + \frac{1}{10} \cdot \varepsilon \leq d_{opt}^{a=b} + d_{opt}^{a=b} \cdot \varepsilon = d_{opt}^{a=b} \cdot (1+\varepsilon)$. □

The other case, where $d_{opt}^{a=b} < \frac{1}{10}$, is considered in the following lemmas. We observe that in each iteration, we get closer to the optimal solution. Note that the proof of the next lemma is long and technical; it can be found in Section 3.1.

Lemma 5. *For every $i \geq 0$ we have that $T_{opt}^{a=b}(a')$ lies within Γ_i.*

Assume we just finished the ith iteration, for some $i > 1$. From Lemma 5, we know that $T_{opt}^{a=b}(a')$ lies within Γ_i. Let v be the grid vertex in Γ_i closest to $T_{opt}^{a=b}(a')$. The transformation $T'(A)$ with $T'(a) = b$ and $T'(a') = v$ has a bottleneck distance of at most $d_{opt}^{a=b} + \sqrt{2}\frac{\varepsilon}{32} \cdot d_{i-1}$, hence:

$$d_i \quad \leq \quad d_{opt}^{a=b} + \frac{\sqrt{2}}{32} \cdot \varepsilon \cdot d_{i-1} \tag{1}$$

Lemma 6. *If $\frac{d_{i-1}}{d_i} \leq (1+\varepsilon)$ in* **Step 4**, *then $d_i \leq d_{opt}^{a=b} \cdot (1+\varepsilon)$.*

Proof. Substituting the antecedent of the lemma (namely $d_{i-1} \leq d_i \cdot (1+\varepsilon)$) into (1) gives us

$$d_i \quad \leq \quad d_{opt}^{a=b} + \frac{\sqrt{2}}{32}\varepsilon \cdot d_i \cdot (1+\varepsilon) \leq d_{opt}^{a=b} \cdot (1+\varepsilon)$$

Here, the last step follows from our requirements on ε: recall that $0 < \varepsilon < \frac{1}{10}$. □

Therefore, whenever our algorithm terminates, it returns a $(1+\varepsilon)$-approximation of $d_{opt}^{a=b}$. What is left to do is to show *that* our algorithm terminates.

Lemma 7. *The algorithm terminates at the latest when the iteration counter i reaches the value of $j := \lceil \log_{(1+\varepsilon)} \frac{1}{d_{opt}^{a=b}} - 1 \rceil$.*

Proof. Consider the execution of the algorithm. If it terminates when $i < j$, then the lemma holds.

If the algorithm iterated past $i = j - 1$, then we have that

$$\frac{d_{i-1}}{d_i} > (1+\varepsilon), \text{ for all } i, 0 \leq i \leq j-1.$$

Rewritten, this is (recall that $d_{-1} = 1$): $(1+\varepsilon)^{(i+1)} d_i < d_{-1} = 1$, and therefore,

$$(1+\varepsilon)^j \cdot d_{j-1} < 1.$$

Note that $\log_{(1+\varepsilon)} \frac{1}{d_{opt}^{a=b}} - 1 \leq j$. Combining this and the last inequality, we get:

$$(1+\varepsilon)^{\log_{(1+\varepsilon)} \frac{1}{d_{opt}^{a=b}} - 1} \cdot d_{j-1} \leq (1+\varepsilon)^j \cdot d_{j-1} < 1,$$

and hence:

$$\frac{d_{j-1}}{(1+\varepsilon) \cdot d_{opt}^{a=b}} < 1.$$

With $d_{opt}^{a=b} \leq d_x$, for all x, we conclude with:

$$d_{j-1} < (1+\varepsilon) \cdot d_{opt}^{a=b} \leq (1+\varepsilon) \cdot d_j.$$

Therefore,

$$\frac{d_{j-1}}{d_j} \leq (1+\varepsilon),$$

and the algorithm will terminate in **Step 4**, in the iteration where $i = j$. □

We summarise this section with the following theorem.

Theorem 3. *The algorithm is a $2(1+\varepsilon)$-approximation algorithm with running time $O(\frac{n}{\varepsilon^2} \cdot \log_{(1+\varepsilon)} \frac{1}{d_{opt}} \cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

Proof. We first prove the approximation bound. Looking at Lemmas 4 and 6, we know that, for a fixed $a \in A$, the algorithm gives us a similarity transformation T with

$$d(T(A), B) \leq d_{opt}^{a=b} \cdot (1+\varepsilon).$$

After having performed the algorithm for all $a \in A$ and having kept track of the best transformation T', we obtain together with Observation 1:

$$d(T'(A), B) \leq \min\{d_{opt}^{a=b} | a \in A\} \cdot (1+\varepsilon) \leq 2 \cdot d_{opt} \cdot (1+\varepsilon).$$

For the running time, recall that we perform a run of the algorithm for each $a \in A$. In each run, we iterate at most $j = \lceil \log_{(1+\varepsilon)} \frac{1}{d_{opt}^{a=b}} - 1 \rceil = O(\log_{(1+\varepsilon)} \frac{1}{d_{opt}})$ times (cf. Lemma 7), where in each iteration, we test $O(\frac{1}{\varepsilon^2})$ similarity transformations. Altogether, this results in the running time as claimed in the theorem. □

As in previous sections, also the third algorithm can be modified by adding $O(\frac{1}{\varepsilon^2})$ Steiner points around each chosen a, and running the algorithm for each of the Steiner points instead of using a. Doing this increases the running time to $O(\frac{n}{\varepsilon^4}\log_{(1+\varepsilon)}\frac{1}{d_{opt}}\cdot L(n))$.

Lemma 8. *The modified version of the third algorithm is a $(1+\varepsilon)$-approximation algorithm with running time $O(\frac{n}{\varepsilon^4}\log_{(1+\varepsilon)}\frac{1}{d_{opt}}\cdot L(n))$, where $L(n)$ is the time needed to compute the bottleneck distance between two sets of points, each of size n.*

Remark 1. Readers familiar with the well-separated pair decomposition (WSPD) [8] might consider using the WSPD to get an $O(n\log n)$ time bound. However, this would only work in the case when $\frac{1}{d_{opt}}$ is bounded by a constant. In the above approach we get an $O(n\log n)$ time bound as long as $\frac{1}{d_{opt}}$ is polynomially bounded, that is $\frac{1}{d_{opt}} = n^{O(1)}$.

3.1. Proof of Lemma 5

We will prove this by distinguishing $i = 0$ and $i \geq 1$, where we devote the next two sections to those cases. Note that as we consider **Step 1** to **Step 4** of the algorithm, we know that $d_{opt}^{a=b} < \frac{1}{10}$.

$i = 0$

We will not only prove $T_{opt}^{a=b}(a') \in \Gamma_0$, but also $T_j(a') \in \Gamma_0$ for all $j \geq 0$, which is needed later on. Recall that T_j is the best transformation found in **Step 3** in the iteration when $i = j$, and $T_j(a) = T_{opt}^{a=b}(a) = b$.

By the way the algorithm works, we have that $T_0(a') \in \Gamma_0$. After the first step, we also have that $d_0 \leq d_{opt}^{a=b} + \frac{\varepsilon}{32}\cdot d_{-1} = d_{opt}^{a=b} + \frac{\varepsilon}{32} < \frac{1}{10} + \frac{1}{320} = \frac{33}{320}$ (here we used $d_{opt}^{a=b} < \frac{1}{10}$ and $\varepsilon < \frac{1}{10}$). Note that d_j is a strictly decreasing series of values, for increasing $j = 0, 1, 2, \ldots$ until the algorithm terminates. Recall that a' is a point furthest from a, and that b, b' is a furthest pair in B of distance 1. Let j be a fixed non-negative integer. We will now consider the locations of $T_{opt}^{a=b}(a')$ and $T_j(a')$ in three different cases (similar to the generic cases in Fig. 5(a)-(c)).

If $T_{opt}^{a=b}(a')$ (or $T_j(a')$) has a distance of less than $1 - \frac{33}{320}$ from b, then the distance between b' and any point in $T_{opt}^{a=b}(A)$ (or $T_j(A)$) is bigger than $\frac{33}{320} > d_0 > d_j \geq d_{opt}^{a=b}$; hence a contradiction.

Similarly, if $T_{opt}^{a=b}(a')$ (or $T_j(a')$) has a distance of more than $1 + \frac{33}{320}$ from b, then the distance between $T_{opt}^{a=b}(a')$ (or $T_j(a')$) and any point in B is bigger than $\frac{33}{320} > d_0 > d_j \geq d_{opt}^{a=b}$. Therefore, $T_{opt}^{a=b}(a')$ and $T_j(a')$ are inside an annulus λ' of width $\frac{66}{320}$, and λ' is 'in the middle of', 'thinner than' and concentric with λ_0, see Fig. 5(d).

It remains to consider the case that $T_{opt}^{a=b}(a')$ (or $T_j(a')$) is inside λ' and outside Γ_0, i.e. $T_{opt}^{a=b}(a') \in \lambda' \setminus \Gamma_0$ (or $T_j(a') \in \lambda' \setminus \Gamma_0$). In this case, $T_{opt}^{a=b}(a')$ (or $T_j(a')$) must be matched to a point b'' in B, and the distance between b'' and $T_{opt}^{a=b}(a')$ (or $T_j(a')$) must be at most $\frac{33}{320}$. Therefore, $b'' \in \lambda_0$, since $T_{opt}^{a=b}(a')$ and $T_j(a')$ are in λ'. Recall **Step 1**, where (w.l.o.g.) we assumed B such that all points in $B \cap \lambda_0$ are below and to the right of b, i.e. they are in the lower right quadrant of the coordinate system with origin at b. Now with straight-forward trigonometry, one can show that the distance between any point in $B \cap \lambda_0$ (e.g. b'') and any point in $\lambda' \setminus \Gamma_0$ (e.g. $T_{opt}^{a=b}(a')$ or $T_j(a')$) is at least $2 \cdot \frac{3}{4} \cdot \sin\frac{1}{10} \approx 0.149 > d_0 > d_j \geq d_{opt}^{a=b}$. Thus a contradiction, and we conclude with $T_{opt}^{a=b}(a') \in \Gamma_0$ and $T_j(a') \in \Gamma_0$.

$i \geq 1$

Also in this case, we will assume the opposite and prove contradictions. So we assume there exists a region Γ_i such that $T_{opt}^{a=b}(a') \notin \Gamma_i$ and $i \geq 1$. Without loss of generality we assume i is the smallest positive integer with $T_{opt}^{a=b}(a') \notin \Gamma_i$; and we consider an optimal transformation $T_{opt}^{a=b}$ giving rise to the optimal bottleneck distance $d_{opt}^{a=b}$. Similarly as above, we will have a few cases to consider as indicated in Figure 5:

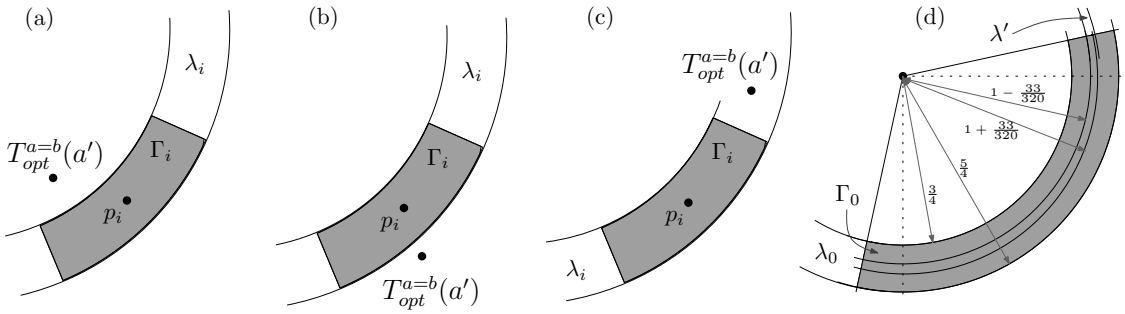

Figure 5: (a)-(c) cases for the location of $T_{opt}^{a=b}(a')$ for $i \geq 1$, (d) illustrating λ'.

3.1.1. Case 1: $T_{opt}^{a=b}(a')$ lies inside the inner boundary of λ_i
(see Fig. 5(a))
Recall from the algorithm that $T_{i-1}(a') = p_{i-1}$ is in the middle of the annulus of λ_i, and $T_{i-1}(a')$ is matched to a point $b^* \in B$ and $|b^*, T_{i-1}(a')| \leq d_{i-1}$. Also recall that a' is a point furthest from a. Since the distance from a point outside λ_i to p_{i-1} is at least $3 \cdot d_{i-1}$ it follows that the distance between b^* and any point in $T_{opt}^{a=b}(A)$ is at least $2 \cdot d_{i-1} > d_{opt}^{a=b}$, thus a contradiction.

3.1.2. Case 2: $T_{opt}^{a=b}(a')$ lies outside the outer boundary of λ_i
(see Fig. 5(b))
Since a' is furthest from a it follows that $|b, b'| \leq |a, T_{i-1}(a')| + d_{i-1}$. If $T_{opt}^{a=b}(a')$ lies outside the outer boundary of λ_i then $|a, T_{opt}^{a=b}(a')| \geq |a, T_{i-1}(a')| + 3 \cdot d_{i-1}$. Hence, the distance between $T_{opt}^{a=b}(a')$ and any point in B is at least $2 \cdot d_{i-1} > d_{opt}^{a=b}$, thus a contradiction.

3.1.3. Case 3: $T_{opt}^{a=b}(a')$ lies outside Γ_i but inside λ_i
(see Fig. 5(c))
Without loss of generality we may assume $T_{opt}^{a=b}(a')$ lies above Γ_i, as shown in Fig. 5(c). The other case, where $T_{opt}^{a=b}(a')$ lies on the other side of Γ_i is symmetric.

First, we consider the point set $T_{opt}^{a=b}(A)$ and observe that $T_{opt}^{a=b}(a')$ must be matched to a point b^* in B within distance at most $d_{opt}^{a=b} \leq d_{i-1}$ from $T_{opt}^{a=b}(a')$, i.e. b^* must be inside a disc centred at $T_{opt}^{a=b}(a')$ of radius at most d_{i-1} (the small grey disc in Fig. 6(a)).

Now, we switch to $T_{i-1}(A)$, i.e. we consider $T_{i-1}(A)$ and b^*. We see that b^* must be matched to a point $T_{i-1}(a^*)$ in $T_{i-1}(A)$, such that $|b^*, T_{i-1}(a^*)| \leq d_{i-1}$. Note that $T_{i-1}(a^*)$ is inside the disc $\mathcal{D}^*$ which is the disc centred at $T_{opt}^{a=b}(a')$ of radius $2 \cdot d_{i-1}$ (see Fig. 6(a)).

When switching between $T_{opt}^{a=b}(A)$ and $T_{i-1}(A)$, we can think of this as a rotation around the point $T_{i-1}(a) = T_{opt}^{a=b}(a) = b$ by the angle $\alpha := \angle(T_{i-1}(a'), b, T_{opt}^{a=b}(a'))$. We call this a (counter-)clockwise *rotation step*.

Now, we switch back and consider $T_{opt}^{a=b}(A)$ again; more specifically, we look at $T_{opt}^{a=b}(a^*)$. We observe that $T_{opt}^{a=b}(a^*)$ is contained in a disc $\mathcal{D}'$ that can be obtained from $\mathcal{D}^*$ by one counter-clockwise rotation step. $T_{opt}^{a=b}(a^*)$ must be matched to a point b^{**} inside a disc that is concentric with $\mathcal{D}'$ and has radius at most $3 \cdot d_{i-1}$.

We switch back to $T_{i-1}(A)$. The point b^{**} must be matched to a point $T_{i-1}(a^{**})$, where this point is inside the disc $\mathcal{D}^{**}$ defined as follows: $\mathcal{D}^{**}$ is concentric with $\mathcal{D}'$ and has radius $4 \cdot d_{i-1}$ (see Fig. 6(a)).

We can repeat the same argument, and by switching back and forth between T_{i-1} and $T_{opt}^{a=b}$, we can construct more discs concentric with other discs, and discs emerging from other discs after one counter-clockwise rotation step. These discs contain points of $T_{i-1}(A)$, $T_{opt}^{a=b}(A)$ and/or B. In what follows, however, we will only focus on the series of discs $\mathcal{D}^*, \mathcal{D}^{**}, \mathcal{D}^{***}, \ldots$, and we use the notation $\mathcal{D}^{k*}$ for a disc $\mathcal{D}^{*\cdots*}$ with k asterisks as superscript. These discs are defined recursively by repeating the construction as described above.

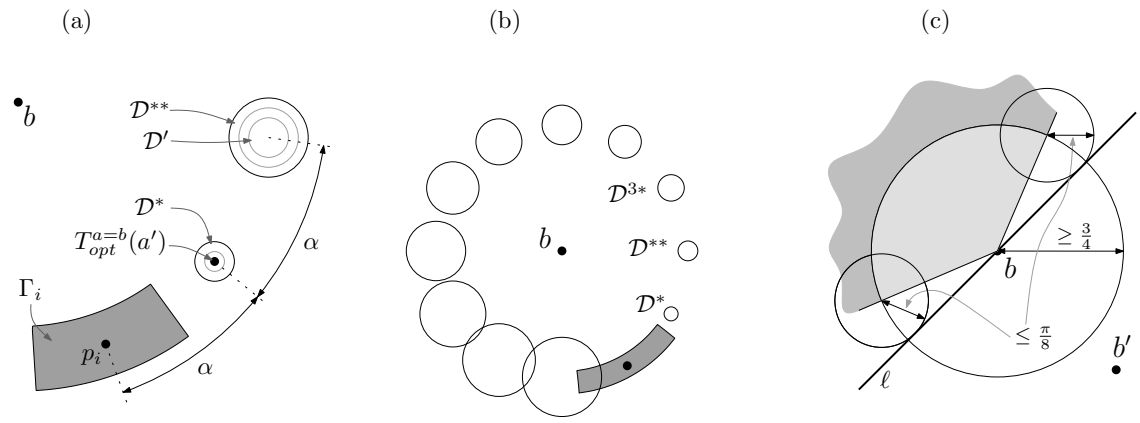

Figure 6: Illustrating: (a) the construction of $\mathcal{D}^*$ centred at $T_{opt}^{a=b}(a')$, $\mathcal{D}'$ obtained from $\mathcal{D}^*$ after rotation around b by angle α, $\mathcal{D}^{**}$ concentric with $\mathcal{D}'$, (b) the series of discs $\mathcal{D}^*, \mathcal{D}^{**}, \mathcal{D}^{***}, \ldots$ with growing radii, (c) the sector (light grey) in the proof of Lemma 10, the dark grey area has to contain a disc centre of a disc $\mathcal{D}^{*\cdots*}$ that has empty intersection with the lower right halfplane.

The goal is to prove the existence of a disc $\mathcal{D}^{*\cdots*}$ that is located 'opposite' of b', such that the point $b^{*\cdots*} \in B$ that is contained in $\mathcal{D}^{*\cdots*}$ is further away from b' than b. This contradicts that b and b' is a furthest pair in B.

To this end, we consider properties of $\mathcal{D}^{k*}$, for $k = 1, 2, 3, \ldots$. We first observe that the radius of $\mathcal{D}^{k*}$ is $2k \cdot d_{i-1}$, and each of those discs contains a point in B. Furthermore, the centres of these discs are located on a circle that is centred at b and has radius $|b, T_{opt}^{a=b}(a')| \geq \frac{3}{4}$, as $T_{opt}^{a=b}(a') \in \Gamma_o$. Along this circle, we have that the angular difference of consecutive disc-centres corresponds exactly to one rotation step.

Lemma 9. *Let α be the angle of one rotation step, i.e. $\alpha = \angle(T_{i-1}(a'), b, T_{opt}^{a=b}(a'))$. Then*

$$32 \cdot d_{i-1} \leq \alpha \leq \frac{\pi}{2} + \frac{2}{5}$$

Proof. From the construction of the sector Γ_i, we know that Γ_i has centre angle $64 \cdot d_{i-1}$. As $T_{opt}^{a=b}(a') \notin \Gamma_i$ and $T_{i-1}(a')$ is in the middle of Γ_i, we conclude that $\alpha \geq 32 \cdot d_{i-1}$.

From above, we have that $T_{opt}^{a=b}(a') \in \Gamma_0$ and $T_{i-1}(a') \in \Gamma_0$. Hence, the angle of a rotation step is bounded from above by the centre angle of Γ_0, which is $\frac{\pi}{2} + \frac{2}{5}$. □

Switching back and forth between $T_{i-1}(A)$ and $T_{opt}^{a=b}(A)$ as often as needed to comprise a full circle (namely $\lfloor \frac{2\pi}{\alpha} \rfloor$ times), we obtain that the last disc with largest radius has radius at most

$$\frac{2\pi}{\alpha} \cdot 2 \cdot d_{i-1} \leq \frac{2\pi}{32 \cdot d_{i-1}} \cdot 2 \cdot d_{i-1} \leq \frac{\pi}{8}$$

Now we know that there are at most $\lfloor \frac{2\pi}{\alpha} \rfloor$ discs $\mathcal{D}^*, \mathcal{D}^{2*}, \ldots$, of radius at most $\frac{\pi}{8}$ arranged on a circle around b (see Fig. 6(b)). And this circle has radius $|b, T_{opt}^{a=b}(a')| \geq \frac{3}{4}$ (recall the construction of Γ_0 in **Step 1** and $T_{opt}^{a=b}(a') \in \Gamma_0$). To show a contradiction, we need to formalise what we mean by 'opposite' of b'. Recall that b' is to the right and below of b. Consider the line ℓ through b that is perpendicular to $\overline{b, b'}$. The line ℓ partitions the underlying plane into two halfplanes (see Fig. 6(c)).

Lemma 10. *There exists a disc $\mathcal{D}^{k*}$, for some $k \in \{1, 2, \ldots, \lfloor \frac{2\pi}{\alpha} \rfloor\}$, that has empty intersection with the halfplane that contains b'.*

Proof. Either of the two halfplanes can be seen as a sector with centre angle π and centre at b. Since the angle of a rotation step is at most $\frac{\pi}{2} + \frac{2}{5} < \pi$, it follows that the centre of at least one

of the discs $\mathcal{D}^{k*}$, for $k = 1, 2, \ldots, \lfloor \frac{2\pi}{\alpha} \rfloor$, is inside the halfplane that does not contain b'. However, this is not enough as we have to show that the entire disc is contained in that halfplane, not only its centre. For this, we use that the centres of the discs are on a circle with radius at least $\frac{3}{4}$, and that the radius of the discs is at most $\frac{\pi}{8}$. To avoid an intersection of such a disc with the halfplane containing b', the centre of that disc has to be inside a sector that is located 'opposite' of b' and has some centre angle less than π (see Fig. 6(c)). With straight-forward trigonometry, we obtain that the centre angle of this sector is at least $\pi - 2 \cdot (\arcsin(\frac{\pi}{8} \cdot \frac{4}{3})) \approx 2.039$. This is bigger than the angle α of any rotation step, $\alpha \leq \frac{\pi}{2} + \frac{2}{5} \approx 1.971$, see Lemma 9; and hence, the lemma follows. □

In other words, there exists a disc $\mathcal{D}^{k*}$ that contains a point $b^{k*} \in B$, such that b^{k*} is not contained in the halfplane that contains b', i.e. b^{k*} is 'opposite' of b'. Hence, the distance between b^{k*} and b' must be larger than the distance between b and b'. This contradicts the fact that b and b' comprise a furthest pair in B, and therefore, the assumption that $T_{opt}^{a=b}(a') \notin \Gamma_i$ must be wrong. This concludes the proof of Lemma 5.

4. Concluding Remarks

We considered approximation algorithms for computing the minimum bottleneck distance between two point sets, where we allow to translate, rotate, scale or reflect (mirror) a point set before calculating the distance. In a series of improvements, we arrived at an algorithm for this problem that is substantially faster than known or trivial solutions. In our considerations, we did not explicitly take reflexion into account. This can be done by simply mirroring one point set and repeating the algorithm with the reflected points. As this doubles the running time, it does not influence the asymptotic behaviour.

The algorithms of Section 2 and 3 use, as a plugin, a method to compute the bottleneck distance between two point sets. Hence, any improvement in running time for this problem would immediately also improve the running time of the algorithms in the present paper. In the analysis, we assumed $0 < \varepsilon < \frac{1}{10}$. The upper bound on ε could possibly be increased to a larger value. But not too large, as otherwise certain arguments in the proofs do not work anymore. In a practical situation, where we are given any $\varepsilon > 0$, we could always run the algorithm with a fixed small enough ε to satisfy the bounds above. Also this does not change the running times, as for $\varepsilon \geq \frac{1}{10}$, the running time actually is independent of ε. We can stop our algorithm as soon as the current bottleneck distance reaches a threshold determined by the user, application or resolution of the data. This might speed up the process in a practice.

The algorithm can be extended to higher dimensions by adapting the annulus grid construction to a high-dimensional spherical shell. The constants used for the construction of the grid will have to be correspondingly changed. For a fixed dimension d we conjecture the running time will increase with an additional factor of $\varepsilon^{-O(d)}$.

Acknowledgements

Thanks to Bojan Djordjevic for many discussions and for the construction used in Lemma 2, and to the thorough reviewers of a previous version of this paper. Thanks also to the attentive reviewer who observed that the approximation algorithm described in Section 3 easily can be extended to higher dimensions. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[1] O. Aichholzer, H. Alt and G. Rote. Matching Shapes with a Reference Point. International Journal on Computational Geometry and Applications, 7(4):349–363, 1997.

[2] D. Aiger and K. Kedem. Exact and Approximate Geometric Pattern Matching for Point Sets in the Plane under Similarity Transformations. Canadian Coference on Computational Geometry, pages 181-184, 2007

[3] H. Alt and L. J. Guibas. Discrete geometric shapes: matching, interpolation, and approximation. In Handbook of Computational Geometry, pages 121–153, Elsevier, 1996.

[4] H. Alt, K. Mehlhorn, H. Wagener and E. Welzl. Congruence, Similarity, and Symmetries of Geometric Objects. Discrete & Computational Geometry, 3:237–256, 1988.

[5] E. M. Arkin, K. Kedem, J. S. B. Mitchell, J. Sprinzak and M. Werman. Matching Points into Noise Regions: Combinatorial Bounds and Algorithms. In proceedings of the 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA 1991), pages 42–51, 1991.

[6] M. Benkert, J. Gudmundsson, F. Hübner and T. Wolle. Reporting flock patterns. Computational Geometry - Theory and Applications, 41(3):11-125, 2008.

[7] S. Cabello, P. Giannopoulos and C. Knauer. On the parameterized complexity of d-dimensional point set pattern matching. Information Processing Letters, 105(2):73–77, 2008.

[8] P. B. Callahan and S. R. Kosaraju. A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields. Jorunal of the ACM, 42(1):67–90, 1995.

[9] J. Casey. A Sequel to the First Six Books of the Elements of Euclid, Containing an Easy Introduction to Modern Geometry with Numerous Examples. 5th ed., Dublin: Hodges, Figgis, & Co., 1888.

[10] L. P. Chew and K. Kedem. Improvements on Geometric Pattern Matching Problems. In proceedings of the 3rd Scandinavian Workshop on Algorithm Theory (SWAT 1992), pages 318–325, 1992.

[11] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg and D. Kravets. Geometric Pattern Matching Under Euclidean Motion. Computational Geometry - Theory & Applications, 7:113–124, 1997.

[12] A. Efrat, A. Itai and M. Katz. Geometry Helps in Bottleneck Matching and Related Problems. Algorithmica 31(1):1–28, 2001.

[13] M. T. Goodrich, J. S. B. Mitchell and M. W. Orletsky. Approximate geometric pattern matching under rigid motions. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21:371–379, 1999.

[14] P. J. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. Computational Geometry - Theory and Applications, 4:137–156, 1994.

[15] D. P. Huttenlocher and K. Kedem. Computing the Minimum Hausdorff Distance for Point Sets Under Translation. Symposium on Computational Geometry, pages 340–349, 1990.

[16] D. P. Huttenlocher, K. Kedem and M. Sharir. The Upper Envelope of Voronoi Surfaces and Its Applications. Symposium on Computational Geometry, pages 194–203, 1991.

[17] D. P. Huttenlocher, K. Kedem and J. M. Kleinberg. On Dynamic Voronoi Diagrams and the Minimum Hausdorff Distance for Point Sets Under Euclidean Motion in the Plane. Symposium on Computational Geometry, pages 110–119, 1992.

[18] D. Mumford. Mathematical theories of shape: do they model perception? In Proceedings of the International Society for Optical Engineering Symposium (SPIE), Geometric methods in computer vision, 1570:2–10, 1991.

[19] R. C. Veltkamp and M. Hagedoorn. State-of-the-art in shape matching. In Principles of Visual Information Retrieval, M. Lew (ed.), Springer, 2001

[20] H. Wiemerskirch, J. Martin, Y. Clerquin, P. Alexandre and S. Jiraskova. Energy saving in flight formation Pelicans flying in a 'V' can glide for extended periods using the other birds' air streams. Nature 413:697–698, 2001.

Appendix A. More details on Proofs

Appendix A.1. Details of the Proof of Lemma 2

In the proof of Lemma 2, we omitted the details that show that the distances between $T_1(a_r)$ and b_r, and between $T_2(a_q)$ and b_q are exactly $3 \cdot d$. We will elaborate on these details here.

For a point p, let $x(p)$ denote its coordinate on the x-axis and $y(p)$ denote its coordinate on the y-axis. Without loss of generality, we consider the points in a coordinate system defined in the following way:

$$x(T_{opt}(a_p)) = 0 \quad (A.1)$$
$$y(T_{opt}(a_p)) = 0 \quad (A.2)$$
$$x(T_{opt}(a_r)) = \ell \quad (A.3)$$
$$y(T_{opt}(a_r)) = 0 \quad (A.4)$$

All other coordinates follow from this definition, see Fig. A.7. To compute the distance between

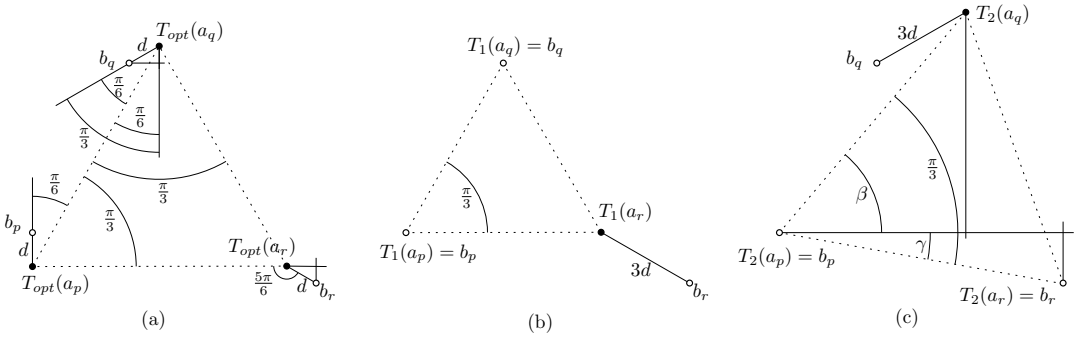

Figure A.7: More details for the proof of Lemma 2.

$T_1(a_r)$ and b_r, we first calculate their coordinates. The coordinates of b_r are computed as follows (using equations (A.3), (A.4); see Fig. A.7(a)):

$$x(b_r) = x(T_{opt}(a_r)) + d \cdot \cos(\frac{\pi}{6}) = \ell + d \cdot \frac{\sqrt{3}}{2} \quad (A.5)$$
$$y(b_r) = y(T_{opt}(a_r)) - d \cdot \sin(\frac{\pi}{6}) = -\frac{d}{2} \quad (A.6)$$

To compute the x-coordinate of $T_1(a_r)$, we first observe that this is equal to the distance between b_p and b_q; and we also note that $|T_{opt}(a_p), T_{opt}(a_q)| = |T_{opt}(a_p), T_{opt}(a_r)| = \ell$, see Fig. A.7(b).

$$|b_p, b_q| = |T_{opt}(a_p), T_{opt}(a_q)| - 2 \cdot d \cdot \cos(\frac{\pi}{6}) = \ell - \sqrt{3} \cdot d \quad (A.7)$$

Now, the coordinates of $T_1(a_r)$ are as follows (using equation (A.7)):

$$x(T_1(a_r)) = |b_p, b_q| = \ell - \sqrt{3} \cdot d \quad (A.8)$$
$$y(T_1(a_r)) = d \quad (A.9)$$

We are now ready to compute the distance between $T_1(a_r)$ and b_r (using equations (A.5), (A.6),

(A.8) and (A.9)):

$$|T_1(a_r), b_r| = \sqrt{\Big(x(T_1(a_r)) - x(b_r)\Big)^2 + \Big(y(T_1(a_r)) - y(b_r)\Big)^2} \quad (A.10)$$

$$= \sqrt{\Big((\ell - \sqrt{3}\cdot d) - (\ell + d\cdot\frac{\sqrt{3}}{2})\Big)^2 + \Big((d) - (-\frac{d}{2})\Big)^2} \quad (A.11)$$

$$= \sqrt{\Big(\ell - \sqrt{3}\cdot d - \ell - d\cdot\frac{\sqrt{3}}{2}\Big)^2 + \Big(d + \frac{d}{2}\Big)^2} \quad (A.12)$$

$$= \sqrt{\Big(-d\cdot\frac{3\cdot\sqrt{3}}{2}\Big)^2 + \Big(\frac{3\cdot d}{2}\Big)^2} \quad (A.13)$$

$$= \sqrt{d^2\cdot\frac{27}{4} + \frac{9\cdot d^2}{4}} = \sqrt{d^2\cdot\frac{36}{4}} = \sqrt{d^2\cdot 9} \quad (A.14)$$

$$= 3\cdot d \quad (A.15)$$

It remains to compute the distance between $T_2(a_q)$ and b_q. First, we consider the distance between b_p and b_r. However, we do not explicitly compute this distance, as it will be cancelled out in later calculations anyway. This is also the distance between $T_2(a_q)$ and b_p, see Fig. A.7(c).

$$|b_p, b_r| = |T_2(a_q), b_p| = \ell' \quad (A.16)$$

Next, we compute the angle β between the x-axis and the line through $T_2(a_q)$ and b_p, see Fig. A.7(c):

$$\beta = \frac{\pi}{3} - \gamma \quad (A.17)$$

Here γ is the angle by which we rotated to obtain $\triangle(T_2(a_p), T_2(a_r), T_2(a_q))$. The angle γ can be computed in two ways (see Fig. A.7(c)):

$$\sin\gamma = \frac{|y(b_p) - y(b_r)|}{\ell'} = \frac{3\cdot d}{2\cdot\ell'} \quad (A.18)$$

$$\cos\gamma = \frac{|x(b_p) - x(b_r)|}{\ell'} = \frac{\ell + d\cdot\frac{\sqrt{3}}{2}}{\ell'} \quad (A.19)$$

Now, using the trigonometric identities

$$\sin(\theta_1 \pm \theta_2) = \sin\theta_1\cdot\cos\theta_2 \pm \cos\theta_1\cdot\sin\theta_2 \quad (A.20)$$

$$\cos(\theta_1 \pm \theta_2) = \cos\theta_1\cdot\cos\theta_2 \mp \sin\theta_1\cdot\sin\theta_2 \quad (A.21)$$

we can compute the following (using equations (A.17)-(A.21)):

$$\cos\beta = \cos(\frac{\pi}{3} - \gamma) = \cos\frac{\pi}{3}\cdot\cos\gamma + \sin\frac{\pi}{3}\cdot\sin\gamma \quad (A.22)$$

$$= \frac{\ell + d\cdot\frac{\sqrt{3}}{2}}{2\cdot\ell'} + \frac{\sqrt{3}\cdot 3\cdot d}{2\cdot 2\cdot\ell'} = \frac{2\cdot\ell + d\cdot\sqrt{3}}{2\cdot 2\cdot\ell'} + \frac{\sqrt{3}\cdot 3\cdot d}{2\cdot 2\cdot\ell'} \quad (A.23)$$

$$= \frac{2\cdot\ell + d\cdot 4\cdot\sqrt{3}}{2\cdot 2\cdot\ell'} = \frac{\ell + d\cdot 2\cdot\sqrt{3}}{2\cdot\ell'} \quad (A.24)$$

$$\sin\beta = \sin(\frac{\pi}{3} - \gamma) = \sin\frac{\pi}{3}\cdot\cos\gamma + \cos\frac{\pi}{3}\cdot\sin\gamma \quad (A.25)$$

$$= \frac{\sqrt{3}\cdot\ell + d\cdot\frac{3}{2}}{2\cdot\ell'} - \frac{3\cdot d}{2\cdot 2\cdot\ell'} = \frac{2\cdot\sqrt{3}\cdot\ell + d\cdot 3}{2\cdot 2\cdot\ell'} - \frac{3\cdot d}{2\cdot 2\cdot\ell'} \quad (A.26)$$

$$= \frac{\sqrt{3}\cdot\ell}{2\cdot\ell'} \quad (A.27)$$

We are now ready to compute the coordinates of $T_2(a_q)$ (see Fig. A.7(c), using (A.24) and (A.27)):

$$x(T_2(a_q)) = \ell' \cdot \cos\beta = \ell' \cdot \frac{\ell + d \cdot 2 \cdot \sqrt{3}}{2 \cdot \ell'} = \frac{\ell + d \cdot 2 \cdot \sqrt{3}}{2} \quad (A.28)$$

$$y(T_2(a_q)) = \ell' \cdot \sin\beta + d = \ell' \cdot \frac{\sqrt{3} \cdot \ell}{2 \cdot \ell'} + d = \frac{\sqrt{3} \cdot \ell}{2} + d \quad (A.29)$$

Now we need the coordinates of b_q (see Fig. A.7(a)):

$$x(b_q) = x(T_{opt}(a_q)) - d \cdot \sin\frac{\pi}{3} = \frac{\ell}{2} - \frac{\sqrt{3} \cdot d}{2} = \frac{\ell - \sqrt{3} \cdot d}{2} \quad (A.30)$$

$$y(b_q) = y(T_{opt}(a_q)) - d \cdot \cos\frac{\pi}{3} = \ell \cdot \sin\frac{\pi}{3} - d \cdot \cos\frac{\pi}{3} = \frac{\ell \cdot \sqrt{3}}{2} - \frac{d}{2} = \frac{\ell \cdot \sqrt{3} - d}{2} \quad (A.31)$$

Having the coordinates of $T_2(a_q)$ and b_q, we can compute the distance between them (using equations (A.28)-(A.31)):

$$|T_2(a_q), b_q| = \sqrt{\Big(x(b_q) - x(T_2(a_q))\Big)^2 + \Big(y(b_q) - y(T_2(a_q))\Big)^2} \quad (A.32)$$

$$= \sqrt{\Big(\frac{\ell - \sqrt{3} \cdot d}{2} - \frac{\ell + d \cdot 2 \cdot \sqrt{3}}{2}\Big)^2 + \Big(\frac{\ell \cdot \sqrt{3} - d}{2} - \frac{\sqrt{3} \cdot \ell}{2} - \frac{2 \cdot d}{2}\Big)^2} \quad (A.33)$$

$$= \sqrt{\Big(\frac{\ell - \sqrt{3} \cdot d - \ell - d \cdot 2 \cdot \sqrt{3}}{2}\Big)^2 + \Big(\frac{\ell \cdot \sqrt{3} - d - \sqrt{3} \cdot \ell - 2 \cdot d}{2}\Big)^2} \quad (A.34)$$

$$= \sqrt{\Big(\frac{-\sqrt{3} \cdot d - d \cdot 2 \cdot \sqrt{3}}{2}\Big)^2 + \Big(\frac{-d - 2 \cdot d}{2}\Big)^2} \quad (A.35)$$

$$= \sqrt{\Big(\frac{-3 \cdot \sqrt{3} \cdot d}{2}\Big)^2 + \Big(\frac{-3 \cdot d}{2}\Big)^2} \quad (A.36)$$

$$= \sqrt{\frac{27 \cdot d^2}{4} + \frac{9 \cdot d^2}{4}} = \sqrt{\frac{36 \cdot d^2}{4}} = \sqrt{9 \cdot d^2} \quad (A.37)$$

$$= 3 \cdot d \quad (A.38)$$

Appendix A.2. Details of the Proof of Theorem 2

In the proof of Theorem 2, we claimed that the distance between $T(a_r)$ and $T_{opt}(a_r)$ is at most $2 \cdot d_{opt}$, and that a scenario as shown in Fig. 3(b) maximises the distance between $T(a_r)$ and $T_{opt}(a_r)$. To see this, we analyse the situation in the following way.

Let us consider the points $T_{opt}(a_r) + \vec{v}$ and $T_{opt}(a_r) + \vec{u}$, as shown in Fig. A.8(a). The distance between those points is $e = ||\vec{u} - \vec{v}|| \leq 2 \cdot d_{opt}$. Let us now consider the region R of intersection of two disks of radius e centred at $T_{opt}(a_r) + \vec{v}$ and $T_{opt}(a_r) + \vec{u}$, respectively, as indicated in Fig. A.8(a) with dotted lines.

The point $T(a_r)$ cannot be outside this region R, because otherwise one of the constraints

$$|T(a_r), T_{opt}(a_r) + \vec{v}| \leq e \quad (A.39)$$

$$|T(a_r), T_{opt}(a_r) + \vec{u}| \leq e \quad (A.40)$$

would not be fulfilled. Recall that $|\vec{u}| \leq d_{opt}$ and $|\vec{v}| \leq d_{opt}$, and hence

$$|T_{opt}(a_r), T_{opt}(a_r) + \vec{u}| \leq d_{opt} \quad (A.41)$$

$$|T_{opt}(a_r), T_{opt}(a_r) + \vec{v}| \leq d_{opt} \quad (A.42)$$

Let us consider the case that the point $T_{opt}(a_r)$ is outside R. Using the triangle inequality and equations (A.41) and (A.42) it then follows that

$$e \leq d_{opt} \quad (A.43)$$

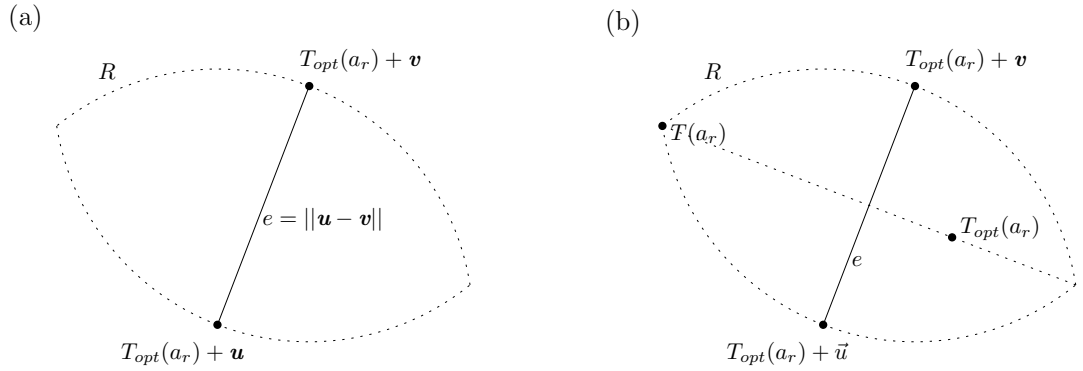

Figure A.8: More details for the proof of Theorem 2.

Taking equations (A.39)-(A.43) together yields:

$$|T(a_r), T_{opt}(a_r)| \leq 2 \cdot d_{opt} \tag{A.44}$$

It remains to consider the case, where the point $T_{opt}(a_r)$ is inside R. Clearly, if both $T_{opt}(a_r)$ and $T(a_r)$ are in R and on the same side of the line through $T_{opt}(a_r) + \vec{u}$ and $T_{opt}(a_r) + \vec{v}$, then

$$|T(a_r), T_{opt}(a_r)| \leq e \leq 2 \cdot d_{opt} \tag{A.45}$$

Hence, the subcase where both $T_{opt}(a_r)$ and $T(a_r)$ are in R, but on different sides of the line through $T_{opt}(a_r) + \vec{u}$ and $T_{opt}(a_r) + \vec{v}$, is the only case left to consider. Note that no matter where $T_{opt}(a_r)$ is located inside R, $|T_{opt}(a_r), T(a_r)|$ is maximised when the point $T(a_r)$ is placed at the corresponding vertex of R, as is shown in Fig. A.8(b). Symmetrically, $|T_{opt}(a_r), T(a_r)|$ is maximised (keeping the constraints (A.41) and (A.41) fulfilled), when the point $T_{opt}(a_r)$ is placed at maximal distance from $T_{opt}(a_r)+\vec{u}$ and $T_{opt}(a_r)+\vec{v}$, which is d_{opt}. Thus, we obtain the scenario as in Fig. 3(b).