# One-to-one Point Set Matchings for Grid Map Layout*

David Eppstein† Marc van Kreveld‡ Bettina Speckmann§ Frank Staals‡


## Abstract

We study several one-to-one point set matching problems which are motivated by layout problems for grid maps. We are given two sets $A$ and $B$ of $n$ points in the plane, and we wish to compute an optimal one-to-one matching between $A$ and $B$. We consider two optimisation criteria: minimising the sum of the $L_1$-distances between matched points, and maximising the number of pairs of points in $A$ for which the matching preserves the directional relation. We show how to minimise the total $L_1$-distance under translation or scaling in $O(n^6 \log^3 n)$ time, and under both translation and scaling in $O(n^{10} \log^3 n)$ time. We further give a 4-approximation for preserving directional relations by computing a minimum $L_1$-distance matching in $O(n^2 \log^3 n)$ time.


## 1 Introduction

We study two point set matching problems motivated by geographic information visualization, specifically by the layout of *grid maps*. Grid maps are a special type of single-level spatial treemap [11]. The input is a set of geographic locations or regions (represented by their centroids) which are mapped one-to-one to a grid of equal-sized rectangles (see Fig. 1). Within the rectangle corresponding to each region or location, additional information can be displayed, see for example the London *BikeGrid* (`gicentre.org/bikegrid`). To aid the user in identifying the regions in the grid map it is essential that the (relative) positions of the input are preserved as much as possible.

We are now given a set of $n$ points $A$ (the geographic locations or region centroids) and a set of $n$ points $B$ (the centers of the grid cells) and we want to compute an optimal one-to-one matching $\phi$ between $A$ and $B$. We consider two optimisation criteria: $(i)$ minimising the sum of the $L_1$-distances between matched points under translation, scaling, and both scaling and translation of point set $A$, and $(ii)$ maximising the number of pairs of points in $A$ for which the matching preserves the *directional relation*. That is, if a point $a_2$ lies northwest of point $a_1$, then we would like $\phi(a_2)$ to lie northwest of $\phi(a_1)$ as well.

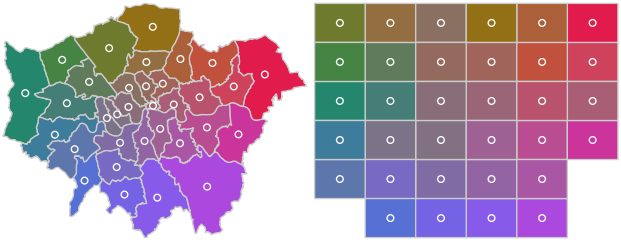

Figure 1: A grid map of the London boroughs.

**Related Work.** Point set matching problems have been studied extensively. We review some of the minimum distance matching methods that are most related to our work. A more extensive survey of existing methods is presented by Alt and Guibas [1], or more recently by Veltkamp and Hagedoorn [10].

Atkinson [4] presents an algorithm for computing a one-to-one matching in case the input point sets are assumed to be copies the same point set, possibly with affine transformations applied on them. His algorithm runs in $O(n \log n)$ time.

Hong and Tan [7] present a similar approach which can also be used when the points sets are not exact copies of each other. They allow a point $p$ to be matched to $q$ if $q$ lies in the *error area* $E(p)$ of $p$: a convex polygon for which the distance between $p$ and the closest point on the boundary of $E(p)$ and the distance between $p$ and the furthest point on the boundary differs by at most a constant factor. It is assumed that $p$ is contained in $E(p)$, and that for any point $q$ we can check if $E(p)$ contains $q$ in constant time. Furthermore, all error regions are pairwise disjoint. Sprinzak and Werman [8] extend Hong and Tan's method for point sets in arbitrary dimensions.

Alt et al. [2] present algorithms for several types of point set matching problems. Their algorithms can handle both the $L_2$-metric, and the $L_\infty$-metric. In case all points have disjoint error areas with radius $\epsilon$ (a disk with radius $\epsilon$ in the Euclidean case and a square with side lengths $2\epsilon$ in case of the $L_\infty$-metric) they present an $O(n \log n)$ time algorithm to compute the minimum distance matching, and the translation that yields this matching.

*B. Speckmann and F. Staals were supported by the Netherlands' Organisation for Scientific Research (NWO) under project no. 639.022.707 and 612.001.022, respectively.

†Department of Computer Science, University of California, Irvine, `eppstein@ics.uci.edu`.

‡Department of Information and Computing Sciences, Utrecht University, `m.j.vankreveld@uu.nl`, and `f.staals@uu.nl`.

§Department of Mathematics and Computer Science, TU Eindhoven, `speckman@win.tue.nl`.

In case the error regions are not given, but instead we should decide whether or not there is a translation with corresponding matching such that the distance between a pair of matched points is at most $\epsilon$, Alt et al. [2] present an $O(n^6)$ time algorithm. When we actually want to find the smallest such $\epsilon$ the running time increases to $O(n^6 \log n)$. Finally, Alt et al. [2] show that if we want to allow rotation and reflection as well we can solve the decision version of the problem in $O(n^8)$ time.

Vaidya [9] studies computing a minimum weight complete matching in a graph $G$ of $2n$ points in which the weights are given by the distance between the points. He shows that for the $L_1$-, $L_2$-, and $L_\infty$-distance an optimal matching can be computed in $O(n^{2.5} \log^4 n)$ time, or $O(n^{2.5} \log n)$ in case $G$ is bipartite. For the $L_1$- and $L_\infty$-metrics this can be improved to $O(n^2 \log^3 n)$. This is the algorithm we will use to compute basic one-to-one matchings without translation or scaling.

Efrat and Itai [6] investigate *bottleneck matching*. They show how to find a one-to-one matching that minimises the $L_\infty$-distance in $O(n^5 \log^2 n)$ time. This improves the results of Alt et al. [2] by almost a factor $n$. For the decision version of the problem their algorithm runs in $O(n^5 \log n)$ time.

There is also a point set matching approach by Cohen and Guibas [5] which uses the *Earth Mover's Distance*. In this setting each point has a certain weight. The amount of work to match a point $a \in A$ with a point $b \in B$ is determined by the distance between $a$ and $b$ and the weight of $a$ that is matched to $b$. The earth mover's distance expresses the minimum work required to match $A$ and $B$. When using the $L_2^2$-distance Cohen and Guibas [5] present an algorithm that computes an optimal transformation of $A$ and a minimum distance matching. For other distances their method yields only a locally optimal transformation and matching.

Alt et al. [3] propose a probabilistic method for matching planar regions. Their algorithm picks a random set of points $A$ in one region and a random set of points $B$ in the other. It then computes a transformation (consisting of a translation and a rotation) such that the area of overlap between the planar regions is close to maximal. They argue that it may be possible to extend their approach to compute a minimum distance matching under affine transformations.

**Organisation.** In the next section we consider minimum $L_1$-distance matchings. We first compute an optimal matching under translation, and then adapt our approach to compute a minimum distance matching under scaling, and both translation and scaling. In Section 3 we then consider matchings that preserve directional relations. Interestingly, we use an algorithm that computes a minimum $L_1$-distance matching to obtain a 4-approximation for this problem.

## 2 Minimising $L_1$-distance

We first define some notation. For a point $a = (a_x, a_y)$ and a translation $t = (t_x, t_y)$ we write $a + t = (a_x + t_x, a_y + t_y)$. We also use this notation for a set of points: $A + t = \{a + t \mid a \in A\}$. Similarly, for a scaling $\lambda = (\lambda_x, \lambda_y)$ we write $\lambda a = (\lambda_x \cdot a_x, \lambda_y \cdot a_y)$. A transformation (either translation or scaling) in which both components have the same value $c$ we denote by $\overline{c} = (c, c)$.

Let $\phi : A \to B$ be a one-to-one matching for the point sets $A$ and $B$, let $t$ be a translation and let $\lambda$ be a scaling. Then we define the *total distance* of matching $\phi$ with translation $t$ and scaling $\lambda$ as

$$D(\phi, t, \lambda) = \sum_{a \in A} d(\lambda a + t, \phi(a))$$

where $d(a, b)$ denotes the $L_1$-distance between $a$ and $b$. We can decompose $d$ into a horizontal and a vertical component: $d(a, b) = x(a, b) + y(a, b)$ with $x(a, b) = |a_x - b_x|$ and $y(a, b) = |a_y - b_y|$. We generalise this notion to $D$, which gives us $D(\phi, t, \lambda) = X(\phi, t, \lambda) + Y(\phi, t, \lambda)$.

Additionally, we define $D_{\mathcal{T}}(\phi, t) = D(\phi, t, \overline{1})$, $D_\Lambda(\phi, \lambda) = D(\phi, \overline{0}, \lambda)$ and $D_I(\phi) = D(\phi, \overline{0}, \overline{1})$. The functions $X_{\mathcal{T}}, Y_{\mathcal{T}}, X_\Lambda$, etc. are defined accordingly.

We now want to find a matching together with a translation and/or scaling that minimises the total distance. More formally, let $\Phi$ be the collection of all one-to-one matchings between $A$ and $B$, let $\mathcal{T}$ be the collection of all translations, and let $\Lambda$ be the collection of all scalings, then we try to find a matching $\phi^* \in \Phi$, a translation $t^* \in \mathcal{T}$, and a scaling $\lambda^* \in \Lambda$ such that

$$D(\phi^*, t^*, \lambda^*) = \min_{\phi \in \Phi, t \in \mathcal{T}, \lambda \in \Lambda} D(\phi, t, \lambda).$$

**Minimising $L_1$ under translation.** To find a minimum distance matching under translation, i.e. a matching that minimises $D_{\mathcal{T}}$, we identify a (finite) set of translations $T \subset \mathcal{T}$ that contains an optimal translation. We then use one of the existing one-to-one matching algorithms for each translation in $T$ to compute an optimal matching.

We say a translation $t$ is a *horizontal translation* if and only if $t = (c, 0)$ for some $c \in \mathbb{R}$. Two point sets $A$ and $B$ are *$x$-aligned* if (and only if) there is a point $a \in A$ and a point $b \in B$ with $a_x = b_x$. We define *vertical translation* and *$y$-aligned* symmetrically.

We now observe that for any matching $\phi$ between point sets $A$ and $B$ that are not $x$-aligned we can decrease $D_{\mathcal{T}}(\phi)$ by $x$-aligning $A$ and $B$ (Fig. 2). Hence:

**Lemma 1** *Let $A$ and $B$ be two non $x$-aligned sets of $n$ points in the plane, and let $\phi$ be any one-to-one*

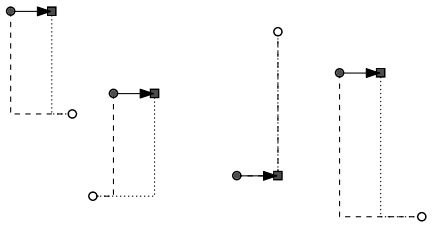

Figure 2: We can improve a matching between $A$ (grey) and $B$ (white) indicated by the dashed lines by $x$-aligning the point sets (the dotted lines).

*matching between $A$ and $B$. Then there is a horizontal translation $t^* \neq \overline{0}$ such that $A^* = A + t^*$ and $B$ are $x$-aligned and $D_{\mathcal{T}}(\phi, t^*) \leq D_I(\phi)$.*

Due to the lack of space, we omit the details of the proof. The crucial observation is that for a given $\phi$, $X'(t) = X_{\mathcal{T}}(\phi, t)$ is a piecewise linear function in $t$ that has its minimum at a breakpoint. Such a breakpoint corresponds to aligning the point sets. An analogous argument gives us that we can decrease $Y_{\mathcal{T}}$ by $y$-aligning two non $y$-aligned sets of points.

Consider the set $T$ of translations that both $x$-align and $y$-align $A$ and $B$. A translation $t \in T$ $x$-aligns a pair of points $(a, b)$, and independently $y$-aligns a pair of points $(a', b')$. This means $T$ contains at most $n^4$ translations.

From Lemma 1 (and its counterpart for $y$-aligning the point sets) it follows that $T$ contains an optimal translation $t^*$. Hence, we can find $\phi^*$ by computing a minimum distance matching for all translations in $T$. If we use the algorithm of Vaidya [9] to compute these point set matchings we obtain the following result:

**Theorem 2** *Given two sets $A$ and $B$ of $n$ points in the plane, a one-to-one matching $\phi^*$ and a translation $t^*$ that minimise $D_{\mathcal{T}}$ can be computed in $O(n^4 \cdot n^2 \log^3 n) = O(n^6 \log^3 n)$ time.*

The main difficulty in improving this result is that $X^*(t) = X(\phi_t^*, t)$, where $\phi_t^*$ denotes an optimal matching for horizontal translation $t$, is not unimodal. Therefore $X^*$ may have several local minima, which means we cannot use a binary search to find an optimal translation $t^*$. Instead, we have to compute a matching for all translations in $T$.

**Minimising $L_1$ under scaling.** For scaling we can use the same procedure as for translation: we prove that there is an optimal scaling that $x$-aligns and $y$-aligns $A$ and $B$ and does not increase the total distance. We again have a set of at most $n^4$ scalings that is guaranteed to contain an optimal scaling. Hence:

**Theorem 3** *Given two sets $A$ and $B$ of $n$ points in the plane, a one-to-one matching $\phi^*$, and a scaling $\lambda^*$ that minimise $D_\Lambda$ can be computed in $O(n^6 \log^3 n)$ time.*

**Minimising $L_1$ under both translation and scaling.** We can use same the approach, but now we $x$-align ($y$-align) two distinct pairs of points. We obtain:

**Theorem 4** *Given two sets $A$ and $B$ of $n$ points in the plane, a one-to-one matching $\phi^*$, a translation $t^*$, and a scaling $\lambda^*$ that minimise $D$ can be computed in $O(n^{10} \log^3 n)$ time.*

## 3 Preserving directional relations

The second criterion that we consider is preserving directional relations. Let $A$ and $B$ be two sets of $n$ points in which no two points have the same $x$- or $y$-coordinate, and let $dir(p, q)$ denote the directional relation of $q$ with respect to $p$ (see Fig. 3 (a)). The goal is now to find a matching $\phi^* : A \to B$ that maximises the number of pairs $(a_1, a_2) \in A \times A$ for which $dir(a_1, a_2) = dir(\phi^*(a_1), \phi^*(a_2))$. Stated differently, we are looking for a matching $\phi^*$ that minimises the number of *out-of-order* pairs $W$ defined as

$$W(\phi) = |\{(a_1, a_2) \mid (a_1, a_2) \in A \times A \wedge dir(a_1, a_2) \neq dir(\phi(a_1), \phi(a_2))\}|.$$

To avoid many nested brackets we will write $a' = \phi(a)$ from now on. Furthermore, we observe that translations and scalings do not influence $W$.

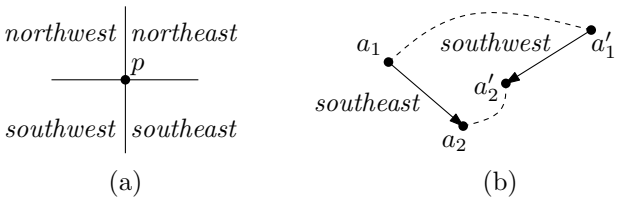


Figure 3: (a) The areas in the plane corresponding to each direction. (b) The directional relation between $a_1$ and $a_2$ is not preserved, $(a_1, a_2)$ is an $x$-inversion.

**A 4-approximation algorithm for minimising $W$.** We now describe an algorithm to compute a matching that approximately minimises $W$. Let $x\text{-}rank_P(p)$ denote the $x$-*rank* of point $p \in P$: that is, the number of points in $P$ to the left of $p$. For points $p \in P$ and $q \in Q$ we write $p \prec_x q$ for $x\text{-}rank_P(p) < x\text{-}rank_Q(q)$. The $y$-rank and $\prec_y$ are defined analogously.

For a given matching, $(a_1, a_2) \in A \times A$ is an $x$-*inversion* if (and only if) $a_1 \prec_x a_2$ and $a'_1 \succ_x a'_2$, or $a_2 \prec_x a_1$ and $a'_2 \succ_x a'_1$. See Fig. 3 (b). Similarly we define a $y$-*inversion*. An inversion is an $x$-inversion,

a $y$-inversion or both. We denote the number of $x$-inversions and the number of $y$-inversions of matching $\phi$ by $I_x(\phi)$ and $I_y(\phi)$, respectively. It is easy to see that there is a one-to-one correspondence between the number of out-of-order pairs $W(\phi)$ of matching $\phi$ and the number of inversions $I(\phi)$, i.e. $W(\phi) = I(\phi)$. Furthermore, we have $\max(I_x(\phi), I_y(\phi)) \leq I(\phi)$ and $I(\phi) \leq I_x(\phi) + I_y(\phi)$.

We define a distance measure $w$ between points $a \in A$ and $b \in B$:

$$w(a,b) = |x\text{-}rank_A(a) - x\text{-}rank_B(b)| + |y\text{-}rank_A(a) - y\text{-}rank_B(b)|.$$

We now compute a minimum distance matching $\phi$ with $w$ as distance measure. The distance measure $w$ is simply the $L_1$-distance on the ranks of the points, which means we can use Vaidya's algorithm [9] to compute $\phi$. We again denote the total distance of $\phi$ by $D(\phi)$, and decompose it into a separate $x$-component $X(\phi)$ and a $y$-component $Y(\phi)$.

The intuition behind our approach is as follows. Consider matching a point $a_r$ with $x\text{-}rank_A(a_r) = i$ to a point $b_r$ with $x\text{-}rank_B(b_r) = j > i$. By the pigeonhole principle there are at least $j - i$ points $\hat{a}$ with $x\text{-}rank_A(\hat{a}) > i$ that are matched to a point $\hat{b}$ with $x\text{-}rank_B(\hat{b}) < j$. Hence, matching $a_r$ to $b_r$ will result in at least $j - i = |x\text{-}rank_A(a_r) - x\text{-}rank_B(b_r)|$ $x$-inversions.

To prove this we order the points $a \in A$ by the rank of $\phi(a)$. For point $a_r$ we consider the number of points that have to "overtake" $a_r$ from left to right when sorting them by rank in $A$ (denoted $\hat{a}$ above). If there are $m$ such points, $a_r$ moves a distance of $2m + (j - i)$ in the sorting process. It follows that $I_x(\phi) = M + X(\phi)/2$, where $M = \sum_{a \in A} m_a$.

Additionally, we can show that there are also at least $m$ points that overtake $a_r$ from right to left, which results in $M \leq X(\phi)/2$. Hence $I_x(\phi) \leq X(\phi)$.

An easy argument as used in the analysis of bubblesort gives us an upper bound of $X(\phi) \leq 2I_x(\phi)$ for the number of inversions in terms of $X$. We conclude:

**Lemma 5** $I_x(\phi) \leq X(\phi) \leq 2I_x(\phi)$.

A symmetric argument holds for the $y$-rank and the number of $y$-inversions. This allows us to prove $W(\phi) = I(\phi) \leq D(\phi) \leq 4I(\phi) = 4W(\phi)$. The following theorem follows:

**Theorem 6** *Given two sets $A$ and $B$ of $n$ points in the plane, a one-to-one matching $\phi$ such that $W(\phi) \leq 4 \cdot \min_{\phi^* \in \Phi} W(\phi^*)$ can be computed in $O(n^2 \log^3 n)$ time.*

## 4 Concluding Remarks

We have seen how to compute a minimum distance matching with respect to the total $L_1$-distance under translation, scaling, and both translation and scaling. An implementation of our method shows that for small values of $n$ we can compute a minimum distance matching under translation or scaling. However, optimising under both is unfeasible in practice.

Furthermore, we can use an algorithm for a minimum $L_1$-distance matching to obtain a matching that approximately maximises the number of correct directional relations. An interesting remaining question is whether it is possible to devise an exact algorithm for this problem that runs in polynomial time.

## References


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

[2] H. Alt, K. Mehlhorn, H. Wagener, and E. Welzl. Congruence, similarity, and symmetries of geometric objects. *Discrete and Computational Geometry*, 3(1): 237–256, 1988.

[3] H. Alt, L. Scharf, and D. Schymura. Probabilistic matching of planar regions. *Computational Geometry*, 43(2):99–114, 2010.

[4] M. Atkinson. An optimal algorithm for geometrical congruence. *Journal of Algorithms*, 8(2):159–172, 1987.

[5] S. Cohen and L. Guibas. The earth mover's distance under transformation sets. In *Proc. 7th IEEE International Conference on Computer Vision*, volume 2, pages 1076–1083, 1999.

[6] A. Efrat and A. Itai. Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th annual ACM Symposium on Computational Geometry*, pages 301–310, 1996.

[7] J. Hong and X. Tan. A new approach to point pattern matching. In *Proc. 9th International Conference on Pattern Recognition*, volume 1, pages 82–84, 1988.

[8] J. Sprinzak and M. Werman. Affine point matching. *Pattern Recognition Letters*, 15(4):337–339, 1994.

[9] P. Vaidya. Geometry helps in matching. *SIAM Journal on Computing*, 18(6):1201–1225, 1989.

[10] R. Veltkamp and M. Hagedoorn. State of the Art in Shape Matching. In *Principles of Visual Information Retrieval*, chapter 4, pages 87–115. Springer-Verlag, 2001.

[11] J. Wood and J. Dykes. Spatially ordered treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1348–1355, 2008.