J Glob Optim (2018) 72:55–70
https://doi.org/10.1007/s10898-018-0653-y

A new algorithm for the small-field astrometric point-pattern matching problem

Cláudio P. Santiago[1] · Carlile Lavor[2] · Sérgio Assunção Monteiro[3] · Alberto Kroner-Martins[4]

Received: 30 May 2017 / Accepted: 10 April 2018 / Published online: 16 April 2018

Abstract The small-field astrometric point-pattern matching problem is a fundamental problem in astronomy, that although considered a long time ago still lacks a formal definition. It can be textually stated as: given two lists of positions of stars, find the common stars between these lists, taking into account rotation, translation, reflection and scaling operations. It is expected that there might be missing stars between the lists. In this work, we give the astronomical context, review some heuristic methods adopted in the literature, present distance matrix formulations for the problem and propose a new algorithm to solve it.

Keywords Distance matrix · Astronomy · Astrometry · Point-pattern matching

1 Introduction

One of the most fundamental aims of small-field astrometry is to study the variations in the time of the relative positions of stars in small field observations of typically some arc min in length (in astronomical observations, distances are expressed in angular units as they

✉ Sérgio Assunção Monteiro
sergio.amonteiro@dataprev.gov.br

Cláudio P. Santiago
prata@llnl.gov

Carlile Lavor
clavor@ime.unicamp.br

Alberto Kroner-Martins
algol@sim.ul.pt

1 Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94550, USA

2 University of Campinas (IMECC - UNICAMP), Campinas, SP 13083-859, Brazil

3 UFRJ - Federal University of Rio de Janeiro, Rio de Janeiro, RJ 23035-380, Brazil

4 SIM - Universidade de Lisboa, 1749-016 Lisboa, Portugal

represent spherical distances). This study is of great importance because it is from these relative motions between stars that we are able to infer the distances between stars and the sun. There are two different motions to consider. One of them, called proper motion, manifests itself as a linear motion through time, and is intrinsic to the star. In fact, it is the projection of the space motion of the star in the tangent plane of a unitary sphere centered on the solar system. The other motion, called parallax, is an apparent effect created from the movement of the Earth around the Sun. This motion is periodic and correlates strongly with the time of the year. A very important property of the parallax is that the further the star is from the sun, the smaller it is. Thus, by measuring this effect, known as the annual parallax of the stars, it is possible to obtain a purely geometric measurement of the distance between these stars and the sun, assuming that the orbit of the Earth around the Sun is well-known.

To solve this problem one needs to determine the relative angular distances between the stars at each observation during the night, and then combine all these measurements resulting into the most probable distances between the pairs of stars and their associated error distributions. Then, it is necessary to determine how these angular distances vary in time. However, first the fundamental problem of matching the stars between all the different observations performed at a same date as well as between different dates (usually called epochs) must be solved. Even in cases where the astronomer is not interested in studying the positions of the stars per se, the identification of the stars among different observations is necessary in most astronomical studies.

The available methods in the literature to solve this problem are heuristic, unable to guarantee a solution for a given precision. Furthermore, they are either based in some form of triangulation or constrained to two dimensions, without an evident efficient extension to higher dimensions. Usually, a method based on triangle matching strategies is adopted [2,9], with $O(n^6)$ to $O(n^{4.5})$ complexity, where n is the number of stars. A short overview of some classic strategies for solving this problem can be found in [7]. That paper presents an alternative method to triangle matching that is based on coordinate couples, their distances and angles relative to an arbitrary axis, with $O(n^2)$ complexity. However, because this method relies on sorting angles, its applicability to higher dimensions is not clear. Further works on this problem adopt variations of the triangle matching method, simplifying it, and thus improving its robustness [11], its execution time [10] (optimistic pattern matching), or enabling its application to wide-field imaging [8] and to the identification of astronomical images with incorrect or missing metadata via a precomputed geometric hashing of the sky [4]. However, to the best of our knowledge, this problem was not yet formally stated.

We present a distance matrix formulation [5] for the astrometric point-pattern matching problem and propose a new way to solve it.

2 The problem

Before formally describing the problem, let us define the following sets for some integer $s > 0$:

$$\begin{aligned} I^s &= \{1, \ldots, s\}, \\ \hat{\mathcal{J}}^s &= \left\{(u, u) : u \in I^s\right\}, \\ \mathcal{J}^s &= I^s \times I^s - \hat{\mathcal{J}}^s. \end{aligned}$$

Given a set of points $p_i = (x_i, y_i) \in I\!R^2$, $i \in I^n$, for some integer $n > 0$, the associated distance matrix D is given by:

$$D_{i,j} = \|p_i - p_j\|_2, i, j \in I^n.$$

Given two lists of stars taken from two different images, we wish to find which stars in one list correspond to the ones in another. One crucial assumption that is used in our approach is that one of the lists is contained in the other. It is worth mentioning that the input lists may have the same size. Although it is possible to extend the results here to the general case of mutual containment of two given lists, such extension using the approach presented in this paper needs considerable changes in order to be efficient. Throughout this paper, we will use the term point to denote a star.

Problem 1 Given two sets $\mathcal{A} = \{p_i \in I\!R^2 : i \in I^m\}$ and $\mathcal{B} = \{q_i \in I\!R^2 : i \in I^n\}$, such that $m \le n$, we wish to find an injective function $f : I^m \to I^n$ such that there exists a sequence of k operations (rotations and scaling) $\mathcal{M}_j$, $j \in I^k$, where

$$p_i = \prod_{j \in I^k} \mathcal{M}_j q_{f(i)}, i \in I^m, f(i) \in I^n.$$

One can define an inner product on $I\!R^{n \times n}$ by setting

$$\langle A, B \rangle = Tr\left(AB^t\right) = \sum_{i,j}^{n} A_{i,j} B_{i,j}.$$

This defines the Frobenius norm on $I\!R^{n \times n}$ by setting $\|A\| = \sqrt{\langle A, A \rangle}$.

If D is a distance matrix, we will say that $\frac{D}{\|D\|}$ is as a normalized distance matrix, where $\|\cdot\|$ is the Frobenius norm. All matrix norms used in this article are to be understood as the Frobenius norm.

Since normalized distance matrices are invariant under rotations and scaling operations (see [6]), our approach to solve Problem 1 will be finding a subset of I^n whose corresponding normalized distance matrix is equal (within some specified tolerance) to the distance matrix associated with I^m, formally described as follows:

Problem 2 Given two distance matrices $D^1 \in I\!R^{m \times m}$ and $D^2 \in I\!R^{n \times n}$, with $m \le n$, and a positive tolerance $\epsilon \in I\!R$, find an injective function $f : I^m \to I^n$ such that

$$\left| \frac{D^1_{x,y}}{\|D^1\|} - \frac{D^2_{f(x),f(y)}}{\left\| D^2_{[i \in f(I^m), j \in f(I^m)]} \right\|} \right| \le \epsilon, \quad \forall (x, y) \in I^m \times I^m, \tag{1}$$

where $D^2_{[i \in f(I^m), j \in f(I^m)]}$ is the submatrix of D^2 formed by the rows and columns with indices in $f(I^m)$.

Notice that Problems 1 and 2 are not equivalent. A solution for Problem 1 is also a solution for Problem 2, but the converse is clearly not true, because of the error (tolerance) allowed in Relation (1).

Given a distance matrix D, we will denote by D_j the jth row of D.

3 The algorithm

Since we want to find an injective function $f : I^m \to I^n$ that satisfies (1), we cannot use such relation, unless we test it for every injective function possible, which would lead to an algorithm of factorial complexity. Instead, for each pair $(x, x+1)$, $x \in I^m$, and a $\delta > 0$, we find a pair $(\hat{x}, u)$, $\hat{x}, u \in I^n$, such that

$$\left| \frac{D^1_{x,y}}{D^1_{x,x+1}} - \frac{D^2_{\hat{x},\hat{y}}}{D^2_{\hat{x},u}} \right| \leq \delta, \tag{2}$$

$\forall y \in I^m$, $\forall \hat{y} \in \tilde{I}$, for some $\tilde{I} \subset I^n$ with $|\tilde{I}| = |I^m|$. Note that $\tilde{I}$ is introduced existentially here. We shall show how to construct it using Algorithm 2 presented later in this section. So, whenever a pair $(\hat{x}, u)$ satisfies the above relation, an injective function f such that $f(x) = \hat{x}$, $f(x+1) = u$ and $f(y) = \hat{y}$ is a possible solution for Problem 2. Since more than one injective function is possible, we will use a bipartite graph to represent all the injective functions that can be constructed from Relation (2). Such graph $\mathcal{G} = (V, E)$ can be defined as:

$$V(\mathcal{G}) = I^m \cup \tilde{I}, \tag{3}$$

$$E(\mathcal{G}) = \left\{ (a, b) : a \in I^m, b \in \tilde{I}, \left| \frac{D^1_{x,a}}{D^1_{x,x+1}} - \frac{D^2_{\hat{x},b}}{D^2_{\hat{x},u}} \right| \leq \delta \right\}. \tag{4}$$

The set of edges of $\mathcal{G}$ can be described as follows:

$$\mathcal{G}_j = \left\{ b : b \in \tilde{I}, \left| \frac{D^1_{x,j}}{D^1_{x,x+1}} - \frac{D^2_{\hat{x},b}}{D^2_{\hat{x},u}} \right| \leq \delta \right\}. \tag{5}$$

So, if a graph $\mathcal{G}$ is such that for each $j \in V(\mathcal{G})$, $\mathcal{G}_j$ is built using Relation (5), then

$$s \in \mathcal{G}_j \iff (i, s) \in E(\mathcal{G}).$$

Thus, an injective function f will correspond to a perfect matching of $\mathcal{G}$. For a graph $\tilde{\mathcal{G}}$ constructed as described by Relations (3, 4), we denote by $\mathcal{M}^*(\tilde{\mathcal{G}})$ the set of all perfect matchings of $\tilde{\mathcal{G}}$.

Relation (2) is easier to check, as the whole submatrix $D^2_{f(I^m)}$ of D^2 is not taken into account, as in Relation (1), which requires the value $\|D^2_{f(I^m)}\|$.

If we are to use Relation (2), we need to find the "correct" value of δ from the value of ϵ, since the input value for our algorithm will be ϵ, not δ. Therefore, we must find bounds on δ_ϵ (where δ is obtained as a function of ϵ) in order to guarantee that if Relation (2) is satisfied for δ, then Relation (1) will be satisfied for ϵ (without loss of generality, we will consider matrices D^1, D^2 to be of equal dimension).

Given a distance matrix $D \in I\!R^{m \times m}$, let us define the following quantity:

$$\mu_D = \min\{D_{ij} : (i, j) \in \mathcal{J}^m\}.$$

Lemma 1 *Given two distance matrices $D^1, D^2 \in I\!R^{m \times m}$, define*

$$\mu^*_{D^1,D^2} = \min\left\{ \frac{\|D^1\|_{max}}{\mu_{D^1}}, \frac{\|D^2\|_{max}}{\mu_{D^2}} \right\}.$$

If $\delta > 0$ is such that

$$\left| \frac{D^1_{r,s}}{D^1_{u,v}} - \frac{D^2_{r,s}}{D^2_{u,v}} \right| \leq \delta, \quad \forall (r, s) \in \mathcal{J}^m, \tag{6}$$

for some $(u, v) \in \mathcal{J}^m$*, then*

$$\left| \frac{D^1_{r,s}}{\|D^1\|} - \frac{D^2_{r,s}}{\|D^2\|} \right| \leq \delta \left(1 + \mu^*_{D^1,D^2} \right), \quad \forall (r, s) \in \mathcal{J}^m. \tag{7}$$

Proof Since (7) must be valid for $\forall (r, s) \in \mathcal{J}^m$, we will first check it for $r = u$ and $s = v$.

$$\begin{aligned}
\left| \frac{D^1_{u,v}}{\|D^1\|} - \frac{D^2_{u,v}}{\|D^2\|} \right| &= \left| \frac{\|D^1\|}{D^1_{u,v}} - \frac{\|D^2\|}{D^2_{u,v}} \right| \left| \frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right| \\
&= \left| \frac{\|D^1\|}{D^1_{u,v}} - \frac{\|D^2\|}{D^2_{u,v}} \right| \frac{\left| \frac{\|D^1\|}{D^1_{u,v}} + \frac{\|D^2\|}{D^2_{u,v}} \right|}{\left| \frac{\|D^1\|}{D^1_{u,v}} + \frac{\|D^2\|}{D^2_{u,v}} \right|} \left| \frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right| \\
&= \left| \frac{ \frac{\sum_{(x,y)\in\mathcal{J}^m} \left(D^1_{x,y}\right)^2}{\left(D^1_{u,v}\right)^2} - \frac{\sum_{(x,y)\in\mathcal{J}^m} \left(D^2_{x,y}\right)^2}{\left(D^2_{u,v}\right)^2} }{ \frac{\|D^1\|}{D^1_{u,v}} + \frac{\|D^2\|}{D^2_{u,v}} } \right| \left| \frac{D^1_{x,y}}{\|D^1\|} \frac{D^2_{x,y}}{\|D^2\|} \right| \\
&= \left| \frac{\sum_{(x,y)\in\mathcal{J}^m} \left(D^1_{x,y}\right)^2}{\left(D^1_{u,v}\right)^2} - \frac{\sum_{(x,y)\in\mathcal{J}^m} \left(D^2_{x,y}\right)^2}{\left(D^2_{u,v}\right)^2} \right| \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{D^1_{u,v}}{\|D^1\|} + \frac{D^2_{u,v}}{\|D^2\|} \right|} \\
&\leq \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{D^1_{u,v}}{\|D^1\|} + \frac{D^2_{u,v}}{\|D^2\|} \right|} \sum_{(x,y)\in\mathcal{J}^m} \left| \frac{\left(D^1_{x,y}\right)^2}{\left(D^1_{u,v}\right)^2} - \frac{\left(D^2_{x,y}\right)^2}{\left(D^2_{u,v}\right)^2} \right| \\
&= \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{(D^1_{u,v})}{\|D^1\|} + \frac{(D^2_{u,v})}{\|D^2\|} \right|} \sum_{(x,y)\in\mathcal{J}^m} \left| \frac{D^1_{x,y}}{D^1_{u,v}} - \frac{D^2_{x,y}}{D^2_{u,v}} \right| \left| \frac{D^1_{x,y}}{D^1_{u,v}} + \frac{D^2_{x,y}}{D^2_{u,v}} \right|.
\end{aligned}$$

Using (6):

$$\begin{aligned}
&\leq \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{D^1_{u,v}}{\|D^1\|} + \frac{D^2_{u,v}}{\|D^2\|} \right|} \delta \sum_{(x,y)\in\mathcal{J}^m} \left| \frac{D^1_{x,y}}{D^1_{u,v}} + \frac{D^2_{x,y}}{D^2_{u,v}} \right| \\
&\leq \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{D^1_{u,v}}{\|D^1\|} + \frac{D^2_{u,v}}{\|D^2\|} \right|} \delta \left| \frac{\|D^1\|}{D^1_{u,v}} + \frac{\|D^2\|}{D^2_{u,v}} \right| \\
&= \delta \frac{\left(\frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \right)^2}{\left| \frac{D^1_{u,v}}{\|D^1\|} + \frac{D^2_{u,v}}{\|D^2\|} \right|} \left| \frac{\|D^1\|}{D^1_{u,v}} + \frac{\|D^2\|}{D^2_{u,v}} \right| \\
&\leq \delta \frac{D^1_{u,v}}{\|D^1\|} \frac{D^2_{u,v}}{\|D^2\|} \leq \delta.
\end{aligned} \tag{8}$$

Let $\eta_1 = \frac{D^1_{u,v}}{\|D^1\|}$ and $\eta_2 = \frac{D^2_{u,v}}{\|D^2\|}$. Then

$$|\eta_1 - \eta_2| = \left| \frac{D^1_{u,v}}{\|D^1\|} - \frac{D^2_{u,v}}{\|D^2\|} \right|. \tag{9}$$

Thus, using (9) and (8):

$$||\eta_2| - |\eta_1|| \leq |\eta_1 - \eta_2| \leq \delta \Rightarrow |\eta_1| \leq |\eta_2| + \delta. \tag{10}$$

Therefore, $\forall (r,s) \in \mathcal{J}^m - \{(u,v)\}$:

$$\left| \frac{D^1_{r,s}}{\|D^1\|} - \frac{D^2_{r,s}}{\|D^2\|} \right| = \left| \frac{D^1_{r,s}}{\frac{D^1_{u,v}}{\eta_1}} - \frac{D^2_{r,s}}{\frac{D^2_{u,v}}{\eta_2}} \right| = \left| \eta_1 \frac{D^1_{r,s}}{D^1_{u,v}} - \eta_2 \frac{D^2_{r,s}}{D^2_{u,v}} \right| \tag{11}$$

Using (10) in (11):

$$\begin{aligned}
\left| \frac{D^1_{r,s}}{\|D^1\|} - \frac{D^2_{r,s}}{\|D^2\|} \right| &\leq \left| (|\eta_2| + \delta) \frac{D^1_{r,s}}{D^1_{u,v}} - \eta_2) \frac{D^2_{r,s}}{D^2_{u,v}} \right| \\
&= \left| \eta_2 \frac{D^1_{r,s}}{D^1_{u,v}} - \eta_2 \frac{D^2_{r,s}}{D^2_{u,v}} \right| + \delta \frac{D^1_{r,s}}{D^1_{u,v}} \\
&= \eta_1 \left| \frac{D^1_{r,s}}{D^1_{u,v}} - \frac{D^2_{r,s}}{D^2_{u,v}} \right| + \delta \frac{D^1_{r,s}}{D^1_{u,v}} \\
&\leq \frac{D^1_{u,v}}{\|D^1\|_{max}} \delta + \delta \frac{\|D^1\|_{max}}{\mu_{D^1}} \leq \delta \left(1 + \frac{\|D^1\|_{max}}{\mu_{D^1}} \right).
\end{aligned} \tag{12}$$

Similar to (10) we can bound η_2:

$$|\eta_2| \leq |\eta_1| + \delta. \tag{13}$$

Using (13) and following the same reasoning as before we can write

$$\left| \frac{D^1_{r,s}}{\|D^1\|} - \frac{D^2_{r,s}}{\|D^2\|} \right| \leq \delta \left(1 + \frac{\|D^2\|_{max}}{\mu_{D^2}} \right). \tag{14}$$

From (12) and (14) we can deduct the following bound:

$$\left| \frac{D^1_{r,s}}{\|D^1\|} - \frac{D^2_{r,s}}{\|D^2\|} \right| \leq \delta \left(1 + \min \left\{ \frac{\|D^1\|_{max}}{\mu_{D^1}}, \frac{\|D^2\|_{max}}{\mu_{D^2}} \right\} \right) = \delta (1 + \mu^*_{D^1,D^2}). \tag{15}$$

Since $(1 + \mu^*_{D^1,D^2}) \geq 1$, bound (15) is superior to bound (8) and the result follows. $\square$

3.1 Understanding the algorithm

Since Relation (2) is not sufficient to find a solution to Problem 2, we need to try several values of ϵ. Therefore, Algorithm 1 consists of trying different values of ϵ [Loop (4–7)] in order to find a small enough tolerance for which Relation (2) holds. At each iteration of the loop, if no injective function is found (Line 4), a higher value of ϵ is considered (Line 6) and Algorithm 2 is called (Line 5). Algorithm 2 consists of eliminating impossible injective functions. We refer to each pass of the loop defined by Lines 5–33 as iterations of Algorithm 2. For each value of ϵ considered, the bound on delta in Line 3 is taken from Lemma 1. We have shown that f is a solution to Problem 2 if there exists an injective function such that Relation (2) holds. Thus, for every row k in D^1, we need to find a row r in D^2 such that

Relation (2) holds. Since r and s are candidates, $D^2_{r,s}$ will be used to normalize row r (Line 9). For every row considered in Line 5, we try to find a row r and an element $D^2_{r,s}$ (Line 8) such that Relation (2) holds for $x = k$, $f(x) = r$ and $f(x+1) = s$. Thus, a graph $\mathcal{G}$ that contains all possible bijections such that $f(k) = r$ and $f(k+1) = s$ is constructed (Lines 9–12) using Relations (3, 4).

If there is at least one possible injective function that can be extracted from the graph (perfect matching) (Line 13), then the current graph $\mathcal{G}$ will be considered (Lines 14–22).

Line 16 checks the consistency of the current injective function with the previously found ones, just after the first iteration (Line 14), thereby removing the inconsistent mappings (by removing edges) from the current graph $\hat{\mathcal{F}}$. If the first iteration is being executed, then there are no graphs stored, so the current graph is simply stored (Line 22).

If no pair (r, s) that satisfies Relation (2) (Line 26) can be found, then there is no f for which Relation (2) holds, i.e., the problem cannot be solved for the current value of δ (Line 27). If there is a unique pair (r, s) that satisfies Relation (2) (Line 30), then we no longer need to compare any more rows of matrix D^1 and loop 5 does not need to be executed (Line 31).

Lines 34–39 check if an injective function can be extracted from the graphs stored in $\mathcal{F}^*$. Algorithm 3 checks whether a graph $\tilde{\mathcal{G}}$ contains a perfect matching f for which Relation (2) is satisfied.

In the following lemma, we use the expressions perfect matching and bijective function interchangeably, since it is easy to see that a bijective function induces a perfect matching and vice versa.

Lemma 2 *Given a tolerance $\epsilon > 0$, Algorithm 2 returns the solution for Problem 2.*

Proof We will prove that Algorithm 2 returns a bijection f if and only if f is a solution to Problem 2. We will assume that, for such ϵ and each pair $(k, k+1)$, $k \in \mathcal{I}^{m-1}$, there exists a pair $(r, s) \in \mathcal{J}^m$ and δ (computed from ϵ using Lemma 1) such that Relation (6) is valid. Otherwise, Algorithm 2 is called with another ϵ as parameter (Algorithm 1). Sufficiency is trivial, i.e., if Algorithm 2 returns a bijection f, then f solves Problem 2, since Algorithm 3 checks whether a given assignment (bijection) satisfies Relation (1). In order to prove necessity, given a tolerance factor $\epsilon > 0$, $\exists f$ and δ such that Relation (6) is valid for $(k, k+1)$ and $(f(k), f(k+1))$, $\forall k \in \mathcal{I}^{m-1}$. We need to prove that a bijective function f (a perfect matching) can be extracted from a graph $g \in \mathcal{F}^*$ in Line 34 in the last iteration of loop Lines 5–33. We will prove the result by contradiction. Let $\tilde{k}$ be the smallest value of k such that $f \notin \mathcal{F}^*$ in loop between Lines 5–33.

In order to simplify the proof, if a bijective function f (or a perfect matching) can be extracted from a graph $g \in \mathcal{F}^*$, we will say that $f \in \mathcal{F}^*$, since the set of edges in a matching induces a function and can be thought of as graph and all graphs stored in $\mathcal{F}^*$ have at least one perfect matching (Line 17).

Clearly, $\tilde{k} > 1$, since if $\tilde{k} = 1$, $f \in \mathcal{G}$ (Lines 10–12), then $f \in \mathcal{F}$ (Line 22) and, therefore, $f \in \mathcal{F}^*$ (Line 29). Since $f \notin \mathcal{F}$ at the end of this iteration, we must have that $f \notin \hat{\mathcal{F}}$ for every iteration of loop between Lines 15–20. That is only possible if $f \notin \tilde{\mathcal{G}} \cap \mathcal{G}, \forall \tilde{\mathcal{G}} \in \mathcal{F}^*$ (Line 16). Because of the choice of $\tilde{k}$, $f \in \tilde{\mathcal{G}}$, for some $\tilde{\mathcal{G}} \in \mathcal{F}^*$. From our assumption, for the given value of ϵ, there exists (r, s) and δ (from Lemma 1) such that Relation (6) is valid for $(k, k+1)$ and $(f(k), f(k+1))$, where $f(\tilde{k}) = r$ and $f(\tilde{k}+1) = s$. So $f \in \mathcal{G}$. Thus $f \in \tilde{\mathcal{G}} \cap \mathcal{G}$. Therefore $f \in \hat{\mathcal{F}}$ (Line 16) and $f \in \mathcal{F}$ (Line 18), implying that $f \in \mathcal{F}^*$, contradicting our hypothesis, thereby proving the result. □

Algorithm 1 FindAssignment

```
1: Input: two distance matrices $D^1 \in I\!R^{m\times m}$ and $D^2 \in I\!R^{n\times n}$, with $m \leq n$, the desired tolerance $\epsilon$.
2: Output: an injective function $f : I^m \to I^n$ as described in Problem 2 or $\emptyset$, if there is no solution.
3: $f \leftarrow \emptyset$;
4: while $f = \emptyset$ do
5:     $f \leftarrow$ FindAssignment($D^1$,$D^2$,$\epsilon$);
6:     $\epsilon \leftarrow \epsilon * 2$;
7: end while
8: return $f$;
```

Algorithm 2

```
1: Input: two distance matrices $D^1 \in I\!R^{m\times m}$ and $D^2 \in I\!R^{n\times n}$, with $m \leq n$ and the desired tolerance $\epsilon$.
2: Output: an injective function $f : I^m \to I^n$ as described in Problem 2 or $\emptyset$, if there is no solution.
3: $\delta = \frac{1}{1+\mu^*_{D^1,D^2}}\epsilon$;
4: $\mathcal{F}^* \leftarrow \emptyset$;
5: for $k \in I^{m-1}$ do
6:     $\alpha_i \leftarrow \frac{D^1_{k,i}}{D^1_{k,k+1}}, \forall i \in I^m$;
7:     $\mathcal{F} \leftarrow \emptyset$;
8:     for $(r, s) \in \mathcal{J}^n$ do
9:         $\beta_i \leftarrow \frac{D^2_{r,i}}{D^2_{r,s}}, \forall i \in I^n$;
10:        $\mathcal{G}_k \leftarrow \{r\}$;
11:        $\mathcal{G}_{k+1} \leftarrow \{s\}$;
12:        $\forall i \in I^m - \{k, k+1\}, \mathcal{G}_i \leftarrow \{t : |\beta_t - \alpha_i| \leq \delta, t \in I^n\}$;
13:        if $|\mathcal{M}^*(\mathcal{G})| \geq 1$ then
14:            if $k > 1$ then
15:                for $\tilde{\mathcal{G}} \in \mathcal{F}^*$ do
16:                    $\hat{\mathcal{F}} \leftarrow \tilde{\mathcal{G}} \cap \mathcal{G}$;
17:                    if $|\mathcal{M}^*(\hat{\mathcal{F}})| \geq 1$ then
18:                        $\mathcal{F} \leftarrow \mathcal{F} \cup \hat{\mathcal{F}}$;
19:                    end if
20:                end for
21:            else
22:                $\mathcal{F} \leftarrow \mathcal{F} \cup \mathcal{G}$;
23:            end if
24:        end if
25:    end for
26:    if $\mathcal{F} = \emptyset$ then
27:        return $\emptyset$;
28:    end if
29:    $\mathcal{F}^* \leftarrow \mathcal{F}$;
30:    if $|\mathcal{F}^*| = 1$ then
31:        break;
32:    end if
33: end for
34: for $\tilde{\mathcal{G}} \in \mathcal{F}^*$ do
35:    $f \leftarrow CheckAssignment(D^1, D^2, \tilde{\mathcal{G}}, \epsilon)$;
36:    if $f \neq \emptyset$ then
37:        return $f$;
38:    end if
39: end for
40: return $\emptyset$;
```

Algorithm 3 CheckAssignment

```
1: Input: two distance matrices $D^1 \in I\!R^{m\times m}$ and $D^2 \in I\!R^{n\times n}$, with $m \le n$, the desired tolerance $\epsilon$ and a graph $\tilde{\mathcal{G}}$.
2: Output: an injective function $\tilde{f}$ or $\emptyset$, if there is no such function.
3: for extract a function $f \in \tilde{\mathcal{G}}$ do
4:     $V \leftarrow true$;
5:     for $(x, y) \in \mathcal{J}^n$ do
6:         if $|\frac{D^1_{x,y}}{\|D^1\|} - \frac{D^2_{f(x),f(y)}}{\|D^2_{f(I^m)}\|}| > \epsilon$ then
7:             $V \leftarrow false$;
8:             break;
9:         end if
10:    end for
11:    if $V$ then
12:        return $f$;
13:    end if
14: end for
15: return $\emptyset$;
```

4 An example and computational results

4.1 An example

In this subsection, we will present an example in order to illustrate how Algorithm 1 works. For the two lists of points in $I\!R^2$, $\left\{(2,1),(1,3),(3,\frac{3}{2})\right\}$ and $\{(7,7),(5,1),(4,7),(3,1),(3,3)\}$, we obtain the following distance matrices ($\epsilon = 10^{-5}$):

$$D^1 = \begin{pmatrix} 0 & \sqrt{5} & \frac{\sqrt{5}}{2} \\ \sqrt{5} & 0 & \frac{5}{2} \\ \frac{\sqrt{5}}{2} & \frac{5}{2} & 0 \end{pmatrix}$$

and

$$D^2 = \begin{pmatrix} 0 & 2\sqrt{10} & 3 & 2\sqrt{13} & 4\sqrt{2} \\ 2\sqrt{10} & 0 & \sqrt{37} & 2 & 2\sqrt{2} \\ 3 & \sqrt{37} & 0 & \sqrt{37} & \sqrt{17} \\ 2\sqrt{13} & 2 & \sqrt{37} & 0 & 2 \\ 4\sqrt{2} & 2\sqrt{2} & \sqrt{17} & 2 & 0 \end{pmatrix},$$

where $I^m = \{1,2,3\}$ and $I^n = \{1,2,3,4,5\}$.

Since $\mu_{D^1} = \frac{\sqrt{5}}{2}$ and $\|D^1\|_{max} = \frac{5}{2}$, $\frac{\|D^1\|_{max}}{\mu_{D^1}} = \frac{\frac{5}{2}}{\frac{\sqrt{5}}{2}} = \frac{5}{\sqrt{5}}$. Since $\mu_{D^2} = 2$ and $\|D^2\|_{max} = 2\sqrt{13}$, $\frac{\|D^2\|_{max}}{\mu_{D^1}} = \frac{2\sqrt{13}}{2} = \sqrt{13}$. Therefore $\mu^*_{D^1,D^2} = \min\{\frac{5}{\sqrt{5}}, \sqrt{13}\} = \frac{5}{\sqrt{5}}$. Since $\epsilon = 10^{-5}$, $\delta = \frac{1}{1+\mu^*_{D^1,D^2}}\epsilon = \frac{1}{1+\frac{5}{\sqrt{5}}}10^{-5} = \frac{\sqrt{5}}{5+\sqrt{5}}10^{-5}$.

Initially, we set $k = 1$ ($k \in I^{m-1} = \{1,2\}$). Thus, $\alpha_1 = \frac{0}{\sqrt{5}} = 0$, $\alpha_2 = \frac{\sqrt{5}}{\sqrt{5}} = 1$ and $\alpha_3 = \frac{\frac{\sqrt{5}}{2}}{\sqrt{5}} = \frac{1}{2}$.

We will now try different pairs (r, s) (see loop in Line 8) that belong to $\mathcal{J}^n$, i.e.,

$$(r,s) \in \{(1,2),(1,3),(1,4),(1,5),(2,1),(2,3),(2,4),(2,5),(3,1),$$

$$(3,2), (3,4), (3,5), (4,1), (4,2), (4,3), (4,5), (5,1), (5,2), (5,3), (5,4)\}.$$

For $r = 1$ and $s = 2$, we have that $\beta_1 = \frac{0}{2\sqrt{10}} = 0$, $\beta_2 = \frac{2\sqrt{10}}{2\sqrt{10}} = 1$, $\beta_3 = \frac{3}{2\sqrt{10}}$, $\beta_4 = \frac{2\sqrt{13}}{2\sqrt{10}} = \sqrt{\frac{13}{10}}$ and $\beta_5 = \frac{4\sqrt{2}}{2\sqrt{10}} = \frac{2}{\sqrt{5}}$.

From the choice of r, s, we have that $\mathcal{G}_1 = \{1\}$, $\mathcal{G}_2 = \{2\}$ and previous calculations imply that

$$\begin{aligned}
|\beta_1 - \alpha_3| &= \frac{1}{2} > \frac{1}{10} > \delta,\\
|\beta_2 - \alpha_3| &= \frac{1}{2} > \frac{1}{10} > \delta,\\
|\beta_3 - \alpha_3| &= |\frac{3}{2\sqrt{10}} - \frac{1}{2}| > \frac{1}{100} > \delta,\\
|\beta_4 - \alpha_3| &= |\sqrt{\frac{13}{10}} - \frac{1}{2}| > \frac{1}{10} > \delta,\\
|\beta_5 - \alpha_3| &= |\frac{2}{\sqrt{5}} - \frac{1}{2}| > \frac{1}{10}.
\end{aligned}$$

Thus, $\mathcal{G}_3 = \emptyset$ and vertex 3 is not connected to any other vertex in $\mathcal{G}$, implying that $\mathcal{G}$ does not have a perfect matching. Thus, $|\mathcal{M}^*(\mathcal{G})| = 0$ and we can infer that for this choice of r, s there is no injective function that solves the problem.

The same happens for $r = 1, 2, 3, 4$ and $s = 1, 2, 3, 4$. However, for $r = 5$ and $s = 1$, we can find the following injective function:

$$f(1) = 5, \quad f(2) = 1, \quad f(3) = 2.$$

We have found a solution for $k = 1$. Therefore, only one iteration of Algorithm 2 was required.

4.2 Computational results

Our experiments were performed both with synthetic and real data. We present first tests performed on randomly generated instances. For each combination of parameters, we ran 50 instances. For these random instances, all results are the average over these 50 instances.

We performed several experiments for different values of m, n and for points in $\mathbb{R}^2$ and $\mathbb{R}^3$. We generated 8 test sets:

– *Set1A*: dimension = 2, $n = 100$, $m = (\frac{x}{100})n$, where $x = 5, 10, \ldots, 50$;
– *Set1B*: dimension = 2, $n = 200$, $m = (\frac{x}{100})n$, where $x = 5, 10, \ldots, 50$;
– *Set1C*: dimension = 2, $n = 50, 55, \ldots, 200$, $m = 0.1 * n$;
– *Set1D*: dimension = 2, $n = 50, 55, \ldots, 200$, $m = 0.5 * n$;
– *Set2A*: dimension = 3, $n = 100$, $m = (\frac{x}{100})n$, where $x = 5, 10, \ldots, 50$;
– *Set2B*: dimension = 3, $n = 200$, $m = (\frac{x}{100})n$, where $x = 5, 10, \ldots, 50$;
– *Set2C*: dimension = 3, $n = 50, 55, \ldots, 200$, $m = 0.1 * n$;
– *Set2D*: dimension = 3, $n = 50, 55, \ldots, 200$, $m = 0.5 * n$.

In order to consider different types of experiments, we need additional notation. We will refer to sets *Set1A*, *Set1B*, *Set1C* and *Set1D* as sets *Set1X*. Similarly, will refer to sets *Set2A*, *Set2B*, *Set2C* and *Set2D* as sets *Set2X*. We will also refer to sets *Set1A* and *Set2A* as *SetXA*. Similarly, we will use the labels *SetXB*, *SetXC* and *SetXD* for all other pairs.

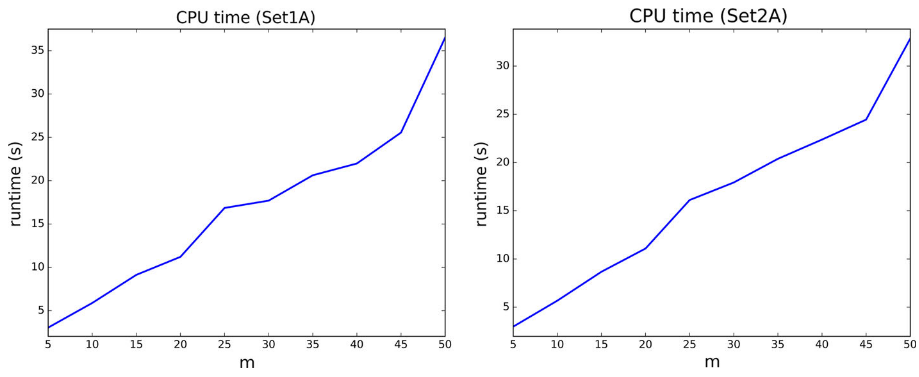

Fig. 1 Runtime as a function of m in $\mathbb{R}^2$ and $\mathbb{R}^3$

Sets *Set1X* are experiments in which the dimension of the points is 2. For sets *Set2X*, the dimension of the points is 3.

Sets *SetXA* are experiments in which n is fixed ($n = 100$) and m (the size of the smaller list of points) varies. For these sets, we show two graphs: runtime as a function of n and runtime as a function of the ratio n/m.

Sets *SetXB* are experiments in which n is fixed ($n = 200$) and m (the size of the smaller list of points) varies. For these sets, we show two graphs: runtime as a function of n and runtime as a function of the ratio n/m.

Sets *SetXC* are experiments in which n varies and the m is obtained from n such that the ratio is constant $\frac{n}{m} = 10$. For these sets, we show two graphs: runtime as a function of n.

Sets *SetXD* are experiments in which n varies and the m is obtained from n such that the ratio is constant $\frac{n}{m} = 2$. For these sets, we show two graphs: runtime as a function of n.

In all experiments, Algorithm 1 found the solution with $\epsilon = 10^{-5}$, which was the initial ϵ used, which means that Algorithm 2 was called only once from Algorithm 1.

Figure 1 shows the runtime as a function of m for sets *Set1A* and *Set2A*, respectively. Figure 2 shows the runtime as a function of m for sets *Set1B* and *Set2B*, respectively. These experiments show that the runtime increases linearly with m for the different values of n considered.

Figure 3 shows the runtime as a function of the ratio $\frac{n}{m}$ for sets *Set1A* and *Set2A*, respectively. Figure 4 shows the runtime as a function of the ratio $\frac{n}{m}$ for sets *Set1B* and *Set2B*, respectively. These experiments show that the runtime decreases with $\frac{n}{m}$ for the different values of the ratio considered.

Figures 5 and 6 show the runtime for experiments *SetXC* and *SetXD*, respectively. We can see that the runtime increases as n increases. For these experiments, the runtime can be bounded by a fourth degree polynomial.

Nearly all experiments for sets *SetXA* and *SetXB* finished in one or two iterations (see Figs. 7, 8, 9, 10). A significant number of instances from sets *SetXC* and *SetXD* required two or more iterations. The number of iterations required by these experiments can be seen on Figs. 11 and 12, respectively. The term iteration here is applied to denote the same as defined in Sect. 3.1.

We will now present experiments done with real data. We ran our algorithm using data of a region of the sky containing a star *Groombridge 34* (see [1]), around which there is one of the closest exoplanets found (see [3]).

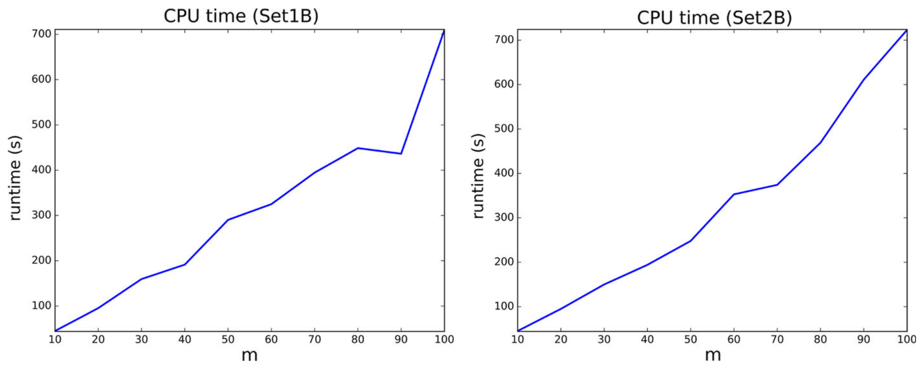

Fig. 2 Runtime as a function of m in $I\!R^2$ and $I\!R^3$

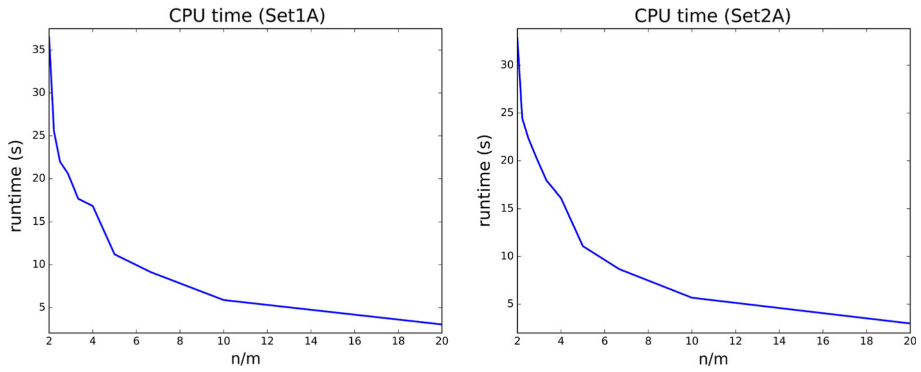

Fig. 3 Runtime as a function of the ratio $\frac{n}{m}$ in $I\!R^2$ and $I\!R^3$

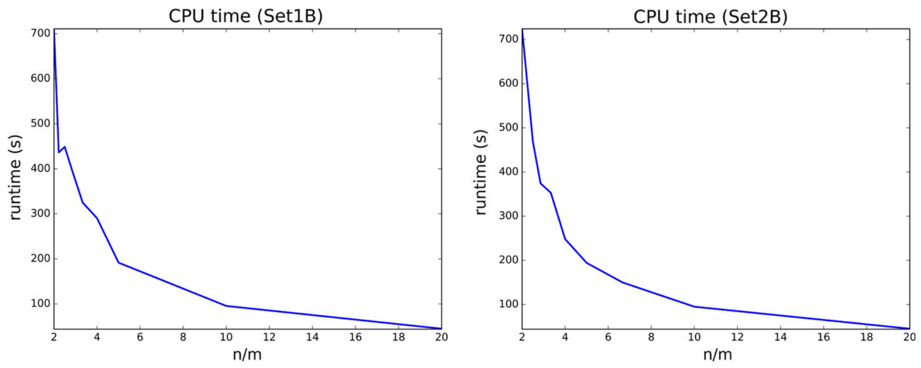

Fig. 4 Runtime as a function of the ratio $\frac{n}{m}$ in $I\!R^2$ and $I\!R^3$

12 sets were run using the same reference catalogue with 202 stars, i.e., they differ only on the observation data. Table 1 shows the results. The columns *catalogue* and *observed* show the value of n and m for each set, respectively.

For both sets, Algorithm 1 found the solution with $\epsilon = 10^{-5}$, which was also the initial ϵ used and only one iteration was required.

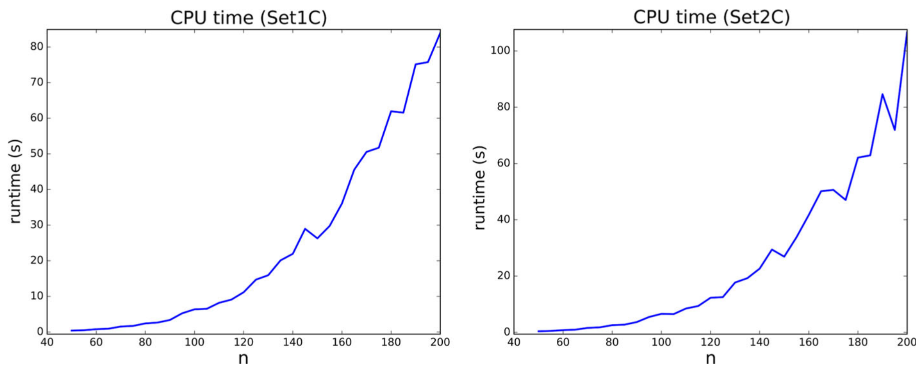

Fig. 5 Runtime as a function of n for $\frac{n}{m} = 10$ in $I\!R^2$ and $I\!R^3$

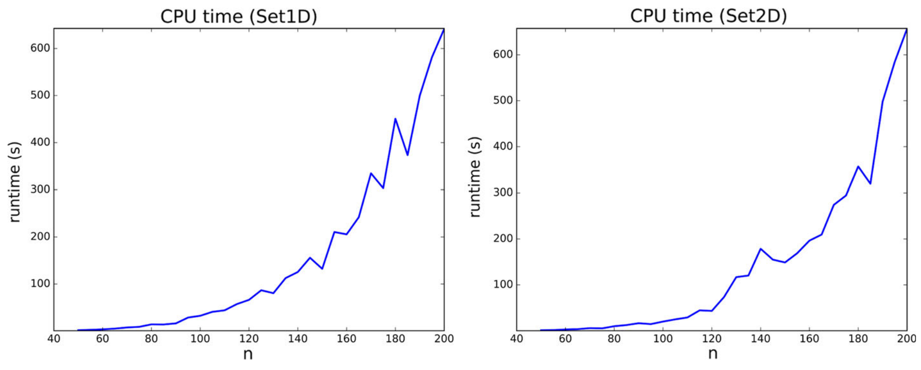

Fig. 6 Runtime as a function of n for $\frac{n}{m} = 2$ in $I\!R^2$ and $I\!R^3$

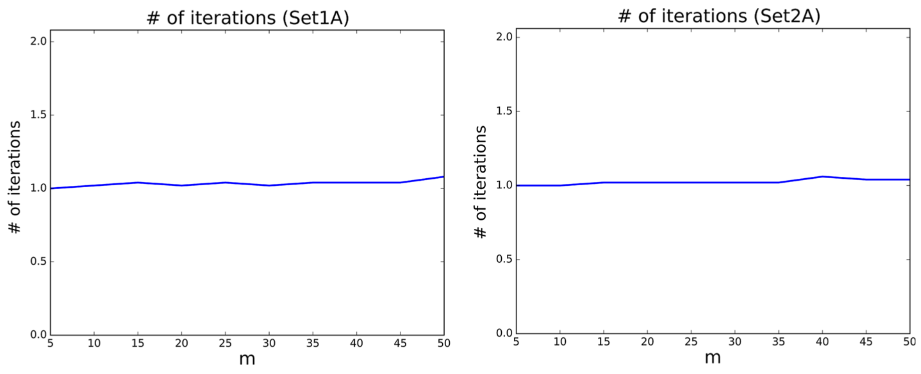

Fig. 7 # of iterations as a function of m in $I\!R^2$ and $I\!R^3$

5 Conclusion

Besides providing a clear and formal definition of the small-field astrometric point-pattern matching problem, we developed a new algorithm to solve it. Unlike the methods developed in the literature, most of which based on triangulation (see [2,10,11]), our method guarantees to find an injective function with a given precision, if such function exists. Additionally, our

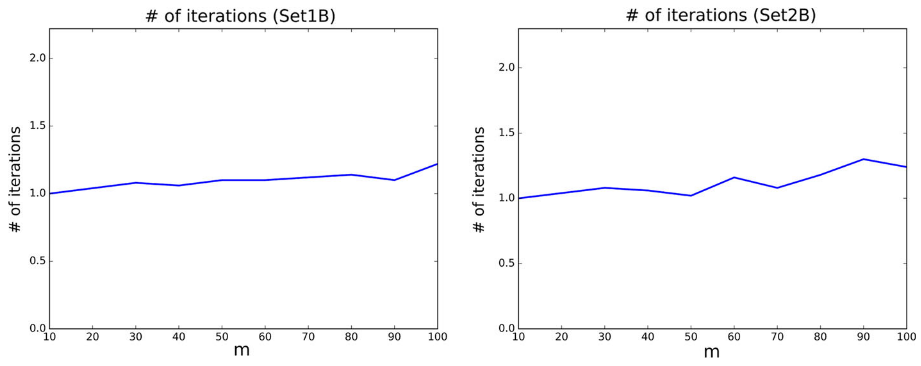

Fig. 8 # of iterations as a function of m in $\mathbb{R}^2$ and $\mathbb{R}^3$

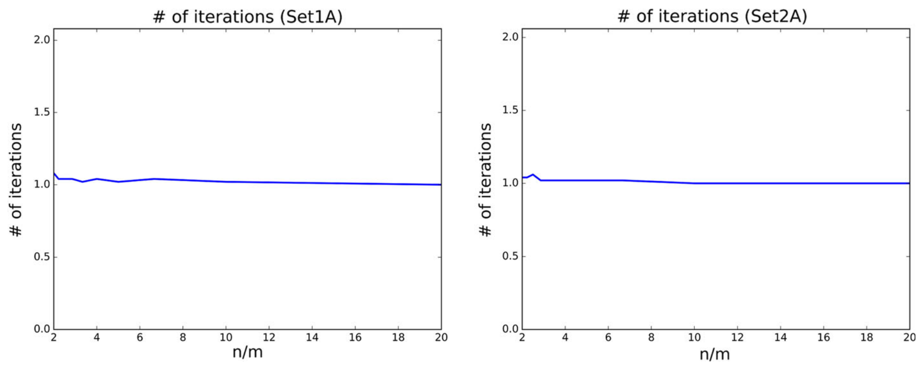

Fig. 9 # of iterations as a function of the ratio $\frac{n}{m}$ in $\mathbb{R}^2$ and $\mathbb{R}^3$

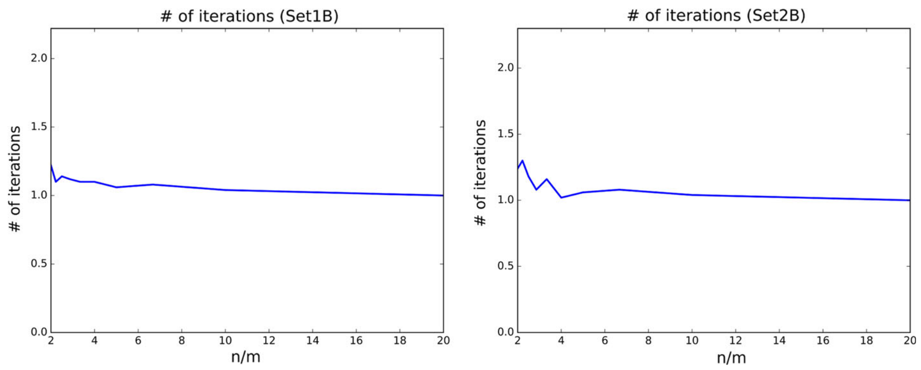

Fig. 10 # of iterations as a function of the ratio $\frac{n}{m}$ in $\mathbb{R}^2$ and $\mathbb{R}^3$

method can be used to solve problems in any dimension, unlike the methods we found to solve this problem in the literature (see [2,7,10,11]).

Our algorithm solved all randomly generated instances very efficiently, both for points in $\mathbb{R}^2$ and $\mathbb{R}^3$, since it required an average of less than two iterations. The growth of the number of operations executed was inferior to a polynomial of fourth degree, pointing to a fairly low complexity of our algorithm in practice. Moreover, our approach was tested in a

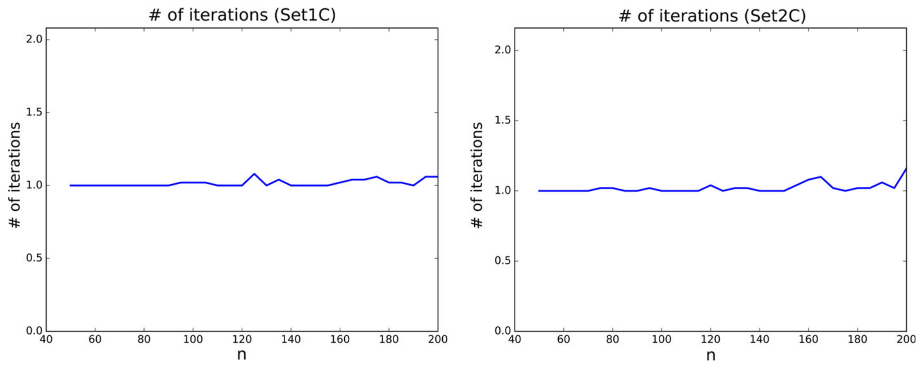

Fig. 11 # of iterations as a function of n for $\frac{n}{m} = 10$ in $I\!R^2$ and $I\!R^3$

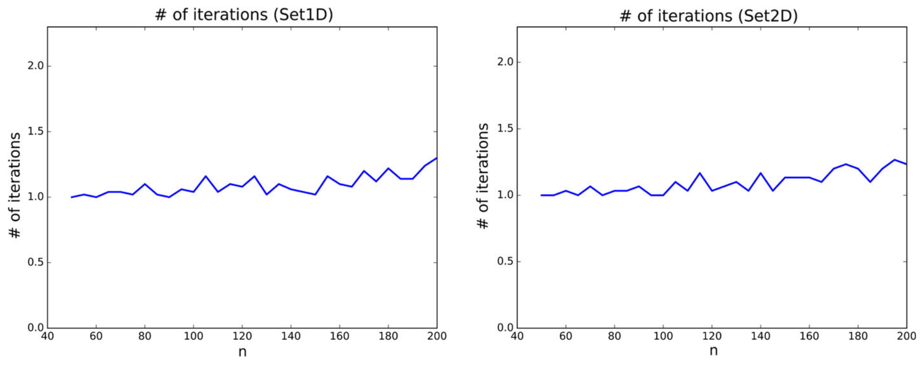

Fig. 12 # of iterations as a function of n for $\frac{n}{m} = 2$ in $I\!R^2$ and $I\!R^3$

Table 1 Groombridge 34

Set	Runtime (s)	Catalogue	Observed
Groombridge34-1	159.8	202	22
Groombridge34-2	124.7	202	16
Groombridge34-3	154.0	202	21
Groombridge34-4	182.0	202	24
Groombridge34-5	190.3	202	26
Groombridge34-6	221.8	202	30
Groombridge34-7	148.0	202	21
Groombridge34-8	165.5	202	23
Groombridge34-9	156.7	202	22
Groombridge34-10	134.5	202	18
Groombridge34-11	235.5	202	38
Groombridge34-12	785.0	202	128

real instance (see [1]). We considered a catalogue with 202 stars for a number of observed data sets. Our algorithm solved these instances with a tolerance of $\epsilon = 10^{-5}$.

Our approach is very flexible, allowing for the incorporation of observation errors that will be considered, including larger instances, in a future work. Furthermore, our approach is independent of the dimension of the space in which the points lie.

Another possible extension of our work would be to find the injective function over the largest domain that solves Problem 2. As an application of this extension, one could find points that were not in the original list. As regards celestial bodies, one could use this approach to detect new objects in the sky, as well as objects that are moving with respect to the background stars.

Acknowledgements This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. We also thank the support of the Brazilian Agencies CNPq, FAPESP, and FAPERJ. We would like to thank the anonymous reviewers for the helpful comments provided on earlier drafts of the manuscript.

References

1. Groombridge, S., Airy, G.B.: A Catalogue of Circumpolar Stars. BiblioBazaar, Charleston (2009)
2. Groth, E.J.: A pattern-matching algorithm for two-dimensional coordinate lists. Astron. J. **91**(5), 1244–1248 (1986)
3. Howard, A.W., Marcy, G.W., Fischer, D.A., Isaacson, H., Muirhead, P.S., Henry, G.W., Boyajian, T.S., Braun, K., Becker, J.C., Wright, J.T.: The NASA-UC-UH ETA-earth program. IV. A low-mass planet orbiting an M dwarf 3.6 PC from earth. Astrophys. J. **794**(1), 51–59 (2014)
4. Lang, D., Hogg, D.W., Mierle, K., Blanton, M., Roweis, S.: Astrometry.net: blind astrometric calibration of arbitrary astronomical images. Astron. J. **139**, 1782–1800 (2010)
5. Liberti, L., Lavor, C., Maculan, N., Mucherino, A.: Euclidean distance geometry and applications. SIAM Rev. **56**(1), 3–69 (2014)
6. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia (2000)
7. Murtagh, F.: A new approach to point-pattern matching. Publ. Astron. Soc. Pac. **104**, 301–307 (1992)
8. Pal, A., Bakos, G.: Astrometry in wide-field surveys. Publ. Astron. Soc. Pac. **118**(848), 1474–1483 (2006)
9. Stetson, P.B.: The techniques of least squares and stellar photometry with CCDs. Vth IAG-USP Advanced School of Astrophysics (1989). http://nedwww.ipac.caltech.edu/level5/Stetson/Stetson5_2.html. Accessed Feb 2017
10. Tabur, V.: Fast algorithms for matching CCD images to a stellar catalogue. Publ. Astron. Soc. Austral. **24**, 189–198 (2007)
11. Valdes, F.G., Campusano, L.E., Velasquez, J.D., Stetson, P.B.: FOCAS automatic catalog matching algorithm. Publ. Astron. Soc. Pac. **107**, 1119–1128 (1995)