Cross-matching Gaia objects

L. Lindegren

GAIA–LL–060 (V1, 9 August 2005)

Abstract. The cross-matching problem for Gaia is reviewed in the context of standard statistical procedures for classification and cluster analysis. Classification is the appropriate method when an input catalogue of sources is available. Observations not cross-matched against a catalogue should instead be subject to cluster analysis to identify new sources. A procedure for cluster analysis that takes proper motion into account is developed and demonstrated. A coherent procedure for cross-matching according to these principles is outlined.

1 Introduction

Cross-matching is a central algorithm in the Gaia data analysis because very little sensible processing can be done to the elementary observations before they have been cross-matched to a source or to each other – thus forming a source. In this document, the function of the cross-matching is analyzed and suitable algorithms identified and discussed. The present work is directly relevant for Work Packages C3–9 and C3–11 of [5].

Because of the way the Gaia instruments are operating, cross-matching applies to objects created through the on-board detection process, including confirmation, using the skymappers. The assumption is that the observations on subsequent CCDs (in the AF, MBP or RVS) implicitly refer to the same source, so that they do not have to be considered separately.

The following terminology will be used:

- An *observation* is the result of a single confirmed detection by one of the skymappers and subsequent CCDs. For cross-matching, the data needed from the observation are: the time and two-dimensional position of the detection, the magnitude, and (for solar system objects) an instantaneous proper motion vector. Uncertainties may also be required. To obtain these data requires some pre-processing to take into account current FOV, calibration and attitude information, and correct for aberration (satellite velocity). Each observation has a unique identifier O.
- A *source* is the entity on the sky assumed to generate observations. A source is described by a set of astrometric and photometric parameters. Each source has a unique identifier S.
- A *cluster* is a set of observations tentatively associated with same source. It is described by the membership list $C = \{O_1, O_2, \dots O_n\}$, where $n \geq 1$ is the number of observations in the cluster.
- An *object* can be either an observation, a cluster, or a source.

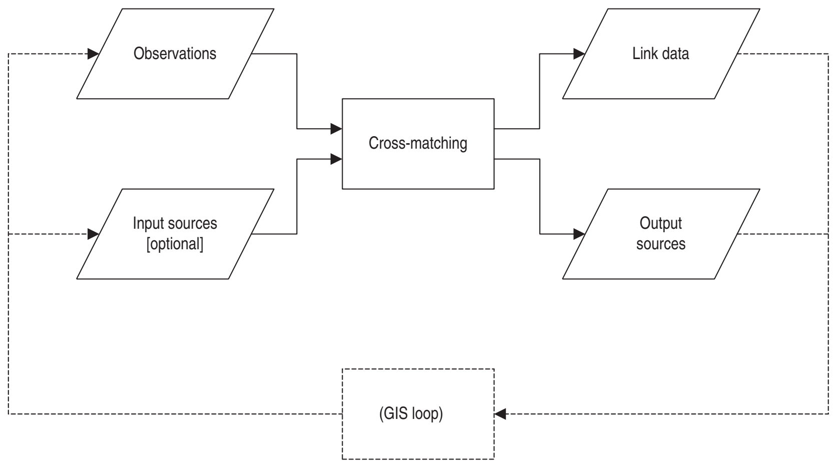

FIGURE 1: Functional diagram of the cross-matching algorithm.

The purpose of the cross-matching is to assign exactly one source to each observation. It corresponds to dividing all the observations into mutually exclusive clusters and making a one-to-one association between clusters and sources.

The function of the cross-matching algorithm is further illustrated by the simplified data flow diagram in Fig. 1. The input consists of a set of observations and, optionally, a set of input sources. The output consists of a set of output sources and link data associating every observation with exactly one output source. The output sources are of three kinds: (1) new, linked sources; (2) old, linked sources; and (3) old, unlinked sources. If no input sources are given, there will be no output sources of kind 2 and 3. The cross-matching may be part of a larger iteration loop (e.g., the GIS) in which both the input catalogue and the observations are successively improved, the latter through the improved attitude and instrument calibration.

The cross-matching algorithm used in GDAAS (see [3] and the present Appendix) matches a list of observations to a given list of sources based purely on positional coincidence. Thus it performs part of the function described in Fig. 1. We wish to consider a generalized algorithm that can also match observations without an input source list and take into account proper motions and possibly more complex source models as well.

2 Cross-matching as a statistical problem

Cross-matching is closely related to two well-known statistical procedures, namely *classification* (assigning cases to one of a fixed number of possible classes) and *cluster analysis* (partitioning cases into subsets of similar cases).[1] The data case is, in our terminology, the observation; the class is the source; and the subset of similar cases is the cluster. The

[1]There is a great deal of variation in terminology depending on the area of application (classical multivariate data analysis, machine learning, data mining, ...). For example, classification is sometimes regarded as the more general procedure, with cluster analysis a form of unsupervised classification.

cross-matching of observations to a given set of input sources is an example of classification, while the cross-matching of observations without input sources is an example of cluster analysis. Let us therefore discuss some classification and cluster analysis algorithms and see how they can be adapted to the cross-matching.[2]

2.1 Classification

Simple classification algorithms that appear suitable for cross-matching with given input sources are the nearest-neighbour algorithm and Bayesian classification.

The nearest-neighbour algorithm links each observation to the nearest source. Formally, it requires that a distance measure $D(O, S)$ can be computed for every possible combination of observation (O) and source (S); then, for each O, a link is established to the S with the smallest $D(O, S)$. The distance measure could in the simplest case be just the angular distance between the observation and source, but it could also take into account a mismatch in magnitude (Sect. 3.5.3), or any other data, through some more general metric, e.g.,

$$D(O, S) = \sum_k w_k \left[x_k(O) - x_k(O|S)\right]^2 \tag{1}$$

Here, $x_k(O)$ ($k = 1 \dots K$) are the components of the observed data vector $\boldsymbol{x}(O)$, of dimension K, and $x_k(O|S)$ are the corresponding data predicted on the assumption that observation O was produced by source S. w_k are pre-assigned weights making it possible, for example, to compare a mismatch in magnitude against that in position. We shall write (1) more concisely

$$D(O, S) = \left\|\boldsymbol{x}(O) - \boldsymbol{x}(O|S)\right\|^2 \tag{2}$$

where the weights $\boldsymbol{w}$ (if required) are implied in the norm. We may also refer to $\boldsymbol{x}(O)$ as the observed 'coordinates', even though the vector could include non-positional data.

Bayesian classification uses Bayes' rule to compute for each source the probability $p(S|O)$ that this particular source was responsible for producing the observed coordinates; then the most probable source is selected. According to Bayes' rule we have

$$p(S|O) = \frac{p(S)p(O|S)}{\sum_{S'} p(S')p(O|S')} \tag{3}$$

where $p(S)$ is the prior probability of S and $p(O|S)$ is the probability density of the observed data on the assumption that they were generated by S (i.e., the likelihood of S for the given data). Since the denominator in (3) is independent of S we only need to consider the numerator to select the most probable source. If we assume that all the input sources are bright enough to be detected with high probability, there is no *a priori* reason why a particular source should be considered more probable than any other; thus $p(S)$ will be the same for all sources. Moreover, if the probability density model for $p(O|S)$ is gaussian with uncorrelated variables x_k of standard deviation σ_k, and if we choose $w_k = \sigma_k^{-2}$ in (1), then

$$p(S|O) \propto \exp\left[-\frac{1}{2}D(O, S)\right] \tag{4}$$

[2]The literature on these techniques is enormous but much of it irrelevant for the present, relatively straightforward applications. See, for example, [2] for a good general reference and introduction.

where the constant of proportionality only depends on O. This means that the source with the smallest $D(S|O)$ also gives the highest posterior probability $p(S|O)$. Thus, under assumptions that are reasonable enough for the cross-matching problem, the nearest-neighbour algorithm is practically equivalent to Bayesian classification. [The argument provides some guidance for the choice of weights w_k in (1): they should be inversely proportional to the variances, taking into account, for example, variability in the case of a magnitude variable.] Consequently, the nearest-neighbour criterion is subsequently adopted and Bayesian classification not further considered below.

2.2 Cluster analysis

Cluster analysis is based on some measure of dissimilarity $\Delta(C_i, C_j)$ between two disjoint clusters C_i and C_j.[3] Since a cluster may consist of a single observation ($n = 1$), we can also measure the dissimilarity between two observations, or between an observation and a cluster. We shall return later to the exact definition of the dissimilarity measure for cross-matching.

The problem is to find the optimum partitioning $C_1 \cup C_2 \cup \cdots \cup C_M$ of the complete set of observations, in the sense of maximizing the inter-cluster dissimilarities $\Delta(C_i, C_j)$ $(i \neq j)$ while minimizing the intrinsic dissimilarities of each cluster, $\Delta(O_k, C_i \setminus O_k)$ $(\forall O_k \in C_i)$. As can be expected, the result depends on how much emphasis is put on the inter-cluster dissimilarities versus the intrinsic dissimilarities. At one extreme all observations are put in a single cluster, making the inter-cluster dissimilarity zero; at the other extreme each observation forms its own cluster, making the intrinsic dissimilarities zero. The optimum solution is somewhere in between these extreme cases. A reasonable way to achieve this for the cross-matching is to set an upper limit on the intrinsic dissimilarity of any cluster.

Of the many methods available for cluster analysis (hierarchical, partitioning, graph methods, ...), only hierarchical agglomerative algorithms are considered here, because one variant of it, the minimum variance method discussed below, appears particularly well adapted for the cross-matching.

The basic idea of hierarchical agglomeration is very simple and can be described in the following steps (*Algorithm HA*):

1. Starting with N observations, make one cluster for each observation. The number of clusters is $M = N$.
2. Compute the $M(M-1)/2$ dissimilarities among the M clusters.
3. Find the pair (C_i, C_j) with the smallest dissimilarity.
4. Agglomerate C_i and C_j into a single cluster, decreasing M by 1.
5. Repeat steps 2 through 4 until $M = 1$.

[3] We distinguish here between the dissimilarity measure, which applies to two objects of the same kind and obeys the symmetry relation $\Delta(C_i, C_j) = \Delta(C_j, C_i)$, and the distance measure which is asymmetric as shown by (2).

The scheme of successive agglomerations can be represented by a tree or dendrogram, which explains why the algorithm is called hierarchical.

Standard agglomerative algorithms differ in principle mainly in how the dissimilarity between the agglomerated cluster $C_i \cup C_j$ and another cluster C_k is computed. Two common choices are the so-called 'single link' and 'complete link' methods,

$$\Delta(C_i \cup C_j, C_k) = \min\left[\Delta(C_i, C_k), \Delta(C_j, C_k)\right] \qquad \text{(single link)} \tag{5a}$$

$$\Delta(C_i \cup C_j, C_k) = \max\left[\Delta(C_i, C_k), \Delta(C_j, C_k)\right] \qquad \text{(complete link)} \tag{5b}$$

Single link allows the formation of elongated clusters (since only the dissimilarity with the nearest neighbour counts), while complete link favours compact clusters (since only the dissimilarity with the most distant member counts). Neither property appears particularly attractive for the cross-matching problem.

Much more promising is *Ward's minimum variance method* [8, 6, 7], especially when generalized as discussed in Sect. 3.4. In this method[4] the intrinsic dissimilarity of a cluster is measured by the sum of squared residuals (SSR) with respect to the cluster centre,

$$R(C) = \sum_{O \in C} \left\| \boldsymbol{x}(O) - \boldsymbol{x}(C) \right\|^2 \tag{6}$$

[Weight factors may be implied in the above expression as in (2).] The coordinates for the cluster centre, $\boldsymbol{x}(C)$, are chosen to minimize the SSR. In the linear case they are simply given by the centre of gravity of the member coordinates:

$$\boldsymbol{x}(C) = \frac{1}{n(C)} \sum_{O \in C} \boldsymbol{x}(O) \tag{7}$$

where $n(C)$ is the number of observations in C.

It is readily seen that the agglomeration of two disjoint clusters C_i and C_j results in a cluster $C = C_i \cup C_j$ with coordinates

$$\boldsymbol{x}(C) = \frac{n(C_i)\boldsymbol{x}(C_i) + n(C_j)\boldsymbol{x}(C_j)}{n(C_i) + n(C_j)} \tag{8}$$

If the original clusters have SSR $R(C_i)$ and $R(C_j)$, respectively, it can be shown (cf. Sect. 3.4.1) that the SSR for the agglomerated cluster is

$$R(C) = R(C_i) + R(C_j) + \frac{n(C_i)n(C_j)}{n(C_i) + n(C_j)} \left\| \boldsymbol{x}(C_i) - \boldsymbol{x}(C_j) \right\|^2 \tag{9}$$

The third term on the right-hand side is the penalty, in terms of the SSR, for agglomerating C_i and C_j. Only if the cluster centres coincide will there be no penalty. In the minimum variance method, this term is taken as the measure of dissimilarity between the clusters:

$$\Delta(C_i, C_j) = \frac{n(C_i)n(C_j)}{n(C_i) + n(C_j)} \left\| \boldsymbol{x}(C_i) - \boldsymbol{x}(C_j) \right\|^2 \tag{10}$$

[4] A description of the algorithm is found in Chapter 3 of [6]. Fortran listings are given in that reference and on the web [7]. Note, however, that the discussion in [6] of the mathematical properties of the method confuses the variance with the sum of squared residuals.

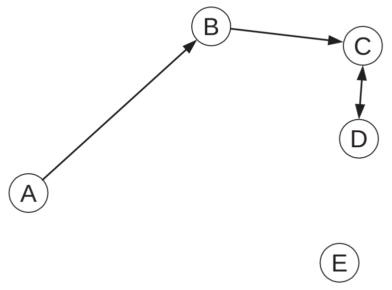

FIGURE 2: Illustrating the nearest-neighbour chain algorithm: starting from the arbitrary point A, its nearest neighbour $\mathrm{NN}(A) = B$ is found, then $\mathrm{NN}(B) = C$ and $\mathrm{NN}(C) = D$. The chain ends here since $\mathrm{NN}(D) = C$. Thus C and D are mutual nearest neighbours and may be agglomerated. (Figure based on [6].)

In the general Algorithm HA described above, the agglomeration is carried all the way to the point where all observations are in a single cluster ($M = 1$). However, for the cross-matching this makes little sense. It is more natural to stop agglomerating once the dispersion of residuals within the clusters have reached a certain limit corresponding to the estimated uncertainties in the observations, attitude, etc. The minimum variance method allows to define such a stopping rule in a simple way. We note that a cluster containing only one observation has $R = 0$. As clusters are built up by agglomeration, their R values increase by accumulation of the corresponding dissimilarities. Thus we can easily keep track of $R(C)$ and $n(C)$ as the agglomeration proceeds. We can then introduce the rule that an agglomeration is only allowed if the resulting internal variance $R(C)/n(C)$ of the agglomerated cluster is below a given limit. A cluster becomes non-agglomerable when it is not allowed to agglomerate with any other cluster. The hierarchical agglomeration stops when all clusters are non-agglomerable, or when $M = 1$.

A Fortran implementation (`hcon2.f`) of the HA algorithm using the minimum variance criterion was given by F. Murtagh [7]. In Sect. 4 we describe numerical experiments performed with a routine based on `hcon2.f` but modified to use the internal variance of a cluster as the stopping criterion, as well as some other refinements to be discussed below. In a number of test cases the resulting algorithm appears to function as well as can be expected.

One problem with clustering algorithms is that they may require the calculation of very many dissimilarities, in the worst case for all possible pairs of coordinates. This tends to give computing times that increase quadratically with the number of observations. For example, in Algorithm HA, the clusters to be agglomerated at each step are the ones with the smallest dissimilarity. To find this pair may require that all $M(M-1)/2$ dissimilarities are computed, where M is the current number of clusters. It is difficult to avoid the quadratic behaviour altogether but various tricks can be used to mitigate the problem.

Murtagh's implementation uses the 'nearest-neighbour-chain' (NNC) algorithm to reduce the number of tests required. This device takes advantage of the circumstance – valid under conditions that hold for the minimum variance method – that two clusters may be agglomerated if they are *mutual nearest neighbours*, independent of the size of their dissimilarity. That is, the hierarchy of clusters obtained will be the same as if agglomerations were made strictly in the sequence of increasing dissimilarity. The NNC algorithm builds a chain of nearest neighbours, starting from an arbitrary (agglomerable) cluster, until a pair of mutual nearest neighbours has been found (Fig. 2). This pair is then agglomerated, the chain is correspondingly modified, and the procedure continues until there is only one cluster in the chain. Then a new chain is initiated, and so on. In our case, where we have a definite upper limit on the internal variance of a cluster, it is furthermore possible to limit the search for the nearest neighbour to a certain radius.

3 Generalized cross-matching

In this section we attempt to formulate a generalized cross-matching algorithm based on the concepts of classification and cluster analysis. In particular, the generalized method should be able to take into account proper motions. For, although the fraction of high-proper motion stars that Gaia will observe is small, their absolute number is not. As the acceptance criteria for the cross-matching are successively sharpened during the GIS process (Fig. 1), an increasing number of stars will be problematic for the cross-matching, unless their proper motions can be taken into account.

3.1 Applicable (linear) source models

The inclusion of proper motions, or indeed of arbitrarily complex source models, is in principle straightforward for the classification problem. This can be seen from (2), where the distance measure $D(O, S)$ is readily computed from the coordinates $\boldsymbol{x}(O|S)$ of source S at the known epoch of observation O. The situation is not so simple for the cluster analysis problem. However, it will be shown in Sect. 3.4 that the cluster analysis algorithm using the minimum variance criterion can be generalized to any *linear* source model. Let us first clarify exactly what this means.

The motion of a source on the sky may be described by a two-dimensional model in the tangent plane, $[\xi(t), \eta(t)]$, or by the coordinates (direction cosines) $[x(t), y(t), z(t)]$ on the unit sphere in three-dimensional space – the choice of coordinates is discussed in Sect. 3.5.2. We note that the distance measure for classification and the dissimilarity measure for cluster analysis are both defined as sums over the different components of the coordinate vector. Thus it is sufficient to consider a single component of the coordinates.

Accordingly, let $u(t)$ be any of the functions $\xi(t)$, $\eta(t)$, $x(t)$, $y(t)$, or $z(t)$. A source model is *linear* if $u(t) = \sum_k a_k f_k(t)$, where $f_k(t)$ are known functions and a_k the (initially unknown) astrometric source parameters.

Among possible linear models the following three are most relevant:

$$u(t) = u_0 \qquad \text{(order 0)} \qquad (11a)$$
$$u(t) = u_0 + u_1 t \qquad \text{(order 1)} \qquad (11b)$$
$$u(t) = u_0 + u_1 t + u_2 p_u(t) \qquad \text{(order 1.5)} \qquad (11c)$$

u_0 is the mean position (or position at epoch $t = 0$); u_1 is the proper motion, and u_2 the parallax. $p_u(t)$ is the known parallax factor in u. Time t should be reckoned from an origin close to or during the mission. (The coordinate components are not independent when parallax is included, since u_2 must be the same for each coordinate.)

The cross-matching discussed in previous sections did not allow for proper motion or parallax, and therefore corresponds to the zeroth-order model (11a). The general case of a linear model of arbitrary order >1 will be considered in following sections.

3.2 Information model

The inclusion of proper motions in the cluster analysis poses an interesting problem. We must be able to compute the dissimilarity $\Delta(C_i, C_j)$ between arbitrary (disjoint) clusters. But the clusters may initially consist of just a single observation each. How can we compute the dissimilarity between two positional observations at different epochs? If proper motion is allowed, it will be possible to match *any* two non-simultaneous observations perfectly, i.e., with zero dissimilarity. This shows that cluster analysis is impossible without an *a priori* constraint on the magnitude of proper motions.

It is natural, therefore, to take a Bayesian approach and introduce a prior probability density for the proper motion component u_1. A reasonable choice is to assume $u_1 = 0$ with an uncertainty large enough to accommodate high-proper motion stars. Adding observations to the cluster eventually leads to an improved estimate of u_1, in the end converging to a value not far from the true proper motion.

Since the two parameters u_0 and u_1 are coupled to each other for every epoch $t \neq 0$, it is necessary to consider the joint probability density of u_0 and u_1. Using a gaussian model, the probability density of $\boldsymbol{u} = (u_0, u_1)$ may be specified by its current estimated value $\widehat{\boldsymbol{u}}$ and covariance matrix $\boldsymbol{V}$. There are however several equivalent representations of the gaussian information to choose from. For example, the normal matrix $\boldsymbol{N} = \boldsymbol{V}^{-1}$ can be used instead of $\boldsymbol{V}$, and the right-hand side of the normal equations $\boldsymbol{N}\widehat{\boldsymbol{u}}$ can be used instead of $\widehat{\boldsymbol{u}}$. Yet another choice is discussed in Sect. 3.4.2.

Presently we adopt the normal matrix $\boldsymbol{N}$ and estimate $\widehat{\boldsymbol{u}}$ to represent the state of knowledge of $\boldsymbol{u}$ for any observation, cluster or source. Furthermore, we will use an *unscaled* version of the normal matrix, where the upper-right element is simply the number of observations, $N_{00} = n(C)$, because this offers the most direct analogy with the zeroth-order treatment in Sect. 2.2. Thus $\boldsymbol{V} = \sigma^2\boldsymbol{N}^{-1}$, where σ is the standard error of each observation. There is another, more fundamental reason for using the unscaled version of the normal matrix instead of $\boldsymbol{V}^{-1}$. The current state of knowledge $[\boldsymbol{N}\ \widehat{\boldsymbol{u}}]$ can be regarded as the result of the least-squares estimation problem $\min_u \|\boldsymbol{A}\boldsymbol{u} - \boldsymbol{b}\|^2$, namely, $\widehat{\boldsymbol{u}} = (\boldsymbol{A}'\boldsymbol{A})^{-1}\boldsymbol{A}'\boldsymbol{b}$ with $\boldsymbol{N} = \boldsymbol{A}'\boldsymbol{A}$. This is consistent with the use of $R = \|\boldsymbol{A}\widehat{\boldsymbol{u}} - \boldsymbol{b}\|^2$ as a measure of the internal dispersion of the cluster, expressed in a physical unit (angle). This would not be possible if the data equations were scaled by their uncertainties, except in the trivial case when a constant standard deviation σ were assumed for all observations (and in that case it would merely represent a change of unit).

The general formulae for agglomeration and dissimilarity in terms of $\boldsymbol{N}$, $\widehat{\boldsymbol{u}}$ and R are derived in Sect. 3.4.1. It will be seen that they are exactly analogous to the zeroth-order formulae in Sect. 2.2.

3.3 Specification of the proper motion prior

Consider a single observation in u, obtained a epoch t with a precision σ_u that will include current attitude errors, etc. Adopting the linear model (11b), with source parameter vector $\boldsymbol{u} = \begin{bmatrix} u_0 & u_1 \end{bmatrix}$, the data equation is

$$\begin{bmatrix} 1 & t \end{bmatrix} \boldsymbol{u} \cong u \quad (\pm\sigma_u) \tag{12}$$

Recall that we wish to work with unscaled normal matrices for the clustering algorithm; thus the relevant matrix for this observation is

$$\boldsymbol{N} = \begin{bmatrix} 1 & t \\ t & t^2 \end{bmatrix} \tag{13}$$

independent of σ_u. Now suppose we wish to add prior knowledge of proper motion corresponding to the data equation

$$\begin{bmatrix} 0 & 1 \end{bmatrix} \boldsymbol{u} \cong \widetilde{u}_1 \quad (\pm\widetilde{\sigma}_1) \tag{14}$$

(typically we would use $\widetilde{u}_1 = 0$ with a fairly large $\widetilde{\sigma}_1$). How can this be expressed as 'unscaled' normal equations, i.e., compatible with (13)? The solution is to multiply the data equation with $L = \sigma_u/\widetilde{\sigma}_1$ (having dimension of time), yielding

$$\begin{bmatrix} 0 & L \end{bmatrix} \boldsymbol{u} \cong L\widetilde{u}_1 \quad (\pm\sigma_u) \tag{15}$$

Since this has the same statistical weight as (12), they can be combined without scaling to give the normal equations

$$\begin{bmatrix} 1 & t \\ t & t^2 + L^2 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} u \\ tu + L^2\widehat{u}_1 \end{bmatrix} \tag{16}$$

The normal matrix to be used is therefore

$$\boldsymbol{N} = \begin{bmatrix} 1 & t \\ t & t^2 + L^2 \end{bmatrix} \tag{17}$$

together with the estimate obtained by solving (16),

$$\widehat{\boldsymbol{u}} = \begin{bmatrix} u - t\widehat{u}_1 \\ \widehat{u}_1 \end{bmatrix} \tag{18}$$

3.4 Cluster analysis and the least-squares criterion

The least-squares formalism is clearly central for the definition of dissimilarity when using the minimum-variance criterion. (Not surprising, since 'minimum variance' is just another expression for ‘least squares’.) Indeed we can formulate the agglomeration and dissimilarity directly in terms of the data equations $[\boldsymbol{A}_i\ \boldsymbol{b}_i]$ and $[\boldsymbol{A}_j\ \boldsymbol{b}_j]$ for two clusters C_i, C_j. The parameters of the agglomerated cluster are obtained by minimizing the total SSR

$$R = \left\|\boldsymbol{A}_i\boldsymbol{u} - \boldsymbol{b}_i\right\|^2 + \left\|\boldsymbol{A}_j\boldsymbol{u} - \boldsymbol{b}_j\right\|^2 \tag{19}$$

The dissimilarity is simply the increase in R when using a common $\boldsymbol{u}$ in (19), compared to the value obtained when the two terms are separately minimized.

One way to solve least-squares problems is by means of normal equations, as will be done in Sect. 3.4.1. However, textbooks on numerical methods (e.g., [4]) generally discourage the use of normal equations for least-squares problems, because this method is more susceptible to roundoff errors than alternative algorithms operating directly on the data equations. In Sect. 3.4.2 we consider a different formalism that takes this aspect into account, at the expense of more computations.

3.4.1 Cluster analysis using normals

The observations for the two disjoint clusters C_i and C_j give the data equations

$$\boldsymbol{A}_k\boldsymbol{u} \cong \boldsymbol{b}_k \qquad (k = i, j) \tag{20}$$

which can be solved separately by the method of least-squares. The normal equations are

$$\boldsymbol{N}_k\boldsymbol{u} = \boldsymbol{h}_k \qquad (k = i, j) \tag{21}$$

with $\boldsymbol{N}_k = \boldsymbol{A}'_k\boldsymbol{A}_k$ and $\boldsymbol{h}_k = \boldsymbol{A}'_k\boldsymbol{b}_k$, from which follow the least-squares estimates

$$\widehat{\boldsymbol{u}}_k = \boldsymbol{N}_k^{-1}\boldsymbol{h}_k \qquad (k = i, j) \tag{22}$$

and finally the sums of squared residuals

$$R_k = \left\|\boldsymbol{b}_k - \boldsymbol{A}_k\widehat{\boldsymbol{u}}_k\right\|^2 = \left\|\boldsymbol{b}_k\right\|^2 - \widehat{\boldsymbol{u}}'_k\boldsymbol{N}_k\widehat{\boldsymbol{u}}_k \qquad (k = i, j) \tag{23}$$

In practice we do not keep the complete data equations (20) for each cluster, but only $\boldsymbol{N}_k$, $\widehat{\boldsymbol{u}}_k$, and R_k as explained previously. We note that the right-hand sides of the normal equations can be computed from these data as $\boldsymbol{h}_k = \boldsymbol{N}_k\widehat{\boldsymbol{u}}_k$.

Now if we consider the agglomerated cluster $C = C_i \cup C_j$, its normal equations are, of course,

$$\begin{aligned}(\boldsymbol{N}_i + \boldsymbol{N}_j)\boldsymbol{u} &= \boldsymbol{h}_i + \boldsymbol{h}_j \\ &= \boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \boldsymbol{N}_j\widehat{\boldsymbol{u}}_j\end{aligned} \tag{24}$$

This gives the following rule for agglomerating coordinates:

$$\widehat{\boldsymbol{u}} = (\boldsymbol{N}_i + \boldsymbol{N}_j)^{-1}(\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \boldsymbol{N}_j\widehat{\boldsymbol{u}}_j) \tag{25}$$

which is a direct generalization of (8). The use of prior information on the proper motions will guarantee that the inverse exists. The normal matrix for the agglomerated cluster is of course

$$\boldsymbol{N} = \boldsymbol{N}_i + \boldsymbol{N}_j \tag{26}$$

(see, however, Eq. 31 below), and its SSR is

$$R = \left\|\boldsymbol{b}_i\right\|^2 + \left\|\boldsymbol{b}_j\right\|^2 - \widehat{\boldsymbol{u}}'\boldsymbol{N}\widehat{\boldsymbol{u}} \tag{27}$$

Defining the dissimilarity in analogy with Sect. 2.2, i.e., as the penalty in the SSR when agglomerating C_i and C_j, we find

$$\begin{aligned}
\Delta(C_i, C_j) &= R - R_i - R_j && \text{(28a)}\\
&= \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\widehat{\boldsymbol{u}}_j - \widehat{\boldsymbol{u}}'\boldsymbol{N}\widehat{\boldsymbol{u}} && \text{(28b)}\\
&= \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\widehat{\boldsymbol{u}}_j - (\widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i + \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j)\boldsymbol{N}^{-1}(\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \boldsymbol{N}_j\widehat{\boldsymbol{u}}_j) && \text{(28c)}\\
&= \widehat{\boldsymbol{u}}_i'(\boldsymbol{N}_i - \boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_i)\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j\widehat{\boldsymbol{u}}_j \\
&\qquad - \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \widehat{\boldsymbol{u}}_j'(\boldsymbol{N}_j - \boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_j)\widehat{\boldsymbol{u}}_j && \text{(28d)}\\
&= \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j\widehat{\boldsymbol{u}}_j \\
&\qquad - \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_i\widehat{\boldsymbol{u}}_i + \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_i\widehat{\boldsymbol{u}}_j && \text{(28e)}\\
&= \widehat{\boldsymbol{u}}_i'\boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j(\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_j) - \widehat{\boldsymbol{u}}_j'\boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_i(\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_j) && \text{(28f)}
\end{aligned}$$

In going from (28d) to (28e) the expansions $\boldsymbol{N}_i = \boldsymbol{N}_i\boldsymbol{N}^{-1}(\boldsymbol{N}_i + \boldsymbol{N}_j) = \boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_i + \boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j$ (etc) have been used to simplify the expressions in parentheses. Moreover,

$$\begin{aligned}
\boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_j &= \boldsymbol{N}_i\boldsymbol{N}^{-1}(\boldsymbol{N} - \boldsymbol{N}_j)\\
&= \boldsymbol{N}_i - \boldsymbol{N}_i\boldsymbol{N}^{-1}\boldsymbol{N}_i\\
&= \boldsymbol{N}_i - (\boldsymbol{N} - \boldsymbol{N}_j)\boldsymbol{N}^{-1}\boldsymbol{N}_i\\
&= \boldsymbol{N}_j\boldsymbol{N}^{-1}\boldsymbol{N}_i
\end{aligned} \tag{29}$$

Since normal matrices are symmetric and non-negative definite, it follows that (29) is also symmetric and non-negative definite. Thus,

$$\Delta(C_i, C_j) = (\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_j)'\boldsymbol{N}_i(\boldsymbol{N}_i + \boldsymbol{N}_j)^{-1}\boldsymbol{N}_j(\widehat{\boldsymbol{u}}_i - \widehat{\boldsymbol{u}}_j) \geq 0 \tag{30}$$

in complete analogy with (10). In fact, for the zeroth-order model, where the normal matrices are of dimension 1×1, the above equation reduces to (10), which has therefore been proved.

There is however one small problem with these formulae when used together with the prior information in proper motion proposed in Sect. 3.3. As more and more observations agglomerate, the weight of the prior increases in proportion to the number of observations. This is unreasonable, since the prior information on a particular source cannot depend on the number of times the source was observed. No unexceptionable solution has been found to this problem, but the following *ad hoc* procedure it proposed: after agglomerating the data according to (25), the normal matrix for the new cluster is computed as

$$\boldsymbol{N} = \boldsymbol{N}_i + \boldsymbol{N}_j - \begin{bmatrix} 0 & 0 \\ 0 & L^2 \end{bmatrix} \tag{31}$$

instead of (26). The last term prevents the weight of the prior information to increase during agglomeration. It will however introduce a small error in R as it is built up through accumulation of dissimilarities.

3.4.2 Cluster analysis using orthogonal transformations

As an alternative to the use of normal equations, we consider briefly a formalism for the agglomeration and dissimilarity calculations based on orthogonal transformations applied to the data equations. This should be the preferred method if higher-order source models are used (e.g., including parallax), resulting in normal matrices of dimension greater than 2×2. However, it is not likely that this is really required for cross-matching.

The method is based on the principle that the SSR is preserved by any orthogonal transformation $\boldsymbol{Q}$ applied to the data equations:

$$R \equiv \left\|\boldsymbol{A}\boldsymbol{u} - \boldsymbol{b}\right\|^2 = \left\|\boldsymbol{Q}\boldsymbol{A}\boldsymbol{u} - \boldsymbol{Q}\boldsymbol{b}\right\|^2 \tag{32}$$

With n denoting the number of data equations (number of observations in the cluster) and m the number of unknowns, the dimensions are: $\boldsymbol{A}[n \times m]$, $\boldsymbol{u}[m]$, $\boldsymbol{b}[n]$, $\boldsymbol{Q}[n \times n]$.

It is always possible to find an orthogonal transformation such that $\boldsymbol{Q}\boldsymbol{A}$ is zero below the diagonal; thus

$$\boldsymbol{Q}\boldsymbol{A} = \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \begin{matrix} \}\ m \\ \}\ n-m \end{matrix} \tag{33}$$

where $\boldsymbol{R}$ is an upper-triangular matrix of dimension $m \times m$. With a corresponding partitioning of the right-hand side,

$$\boldsymbol{Q}\boldsymbol{b} = \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{e} \end{bmatrix} \begin{matrix} \}\ m \\ \}\ n-m \end{matrix} \tag{34}$$

the SSR can be written

$$R = \left\| \begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{u} - \begin{bmatrix} \boldsymbol{z} \\ \boldsymbol{e} \end{bmatrix} \right\|^2 = \left\|\boldsymbol{R}\boldsymbol{u} - \boldsymbol{z}\right\|^2 + \left\|\boldsymbol{e}\right\|^2 \tag{35}$$

which is minimized for

$$\widehat{\boldsymbol{u}} = \boldsymbol{R}^{-1}\boldsymbol{z} \tag{36}$$

yielding $R = \|\boldsymbol{e}\|^2$. The 'square root information' array $\begin{bmatrix}\boldsymbol{R} & \boldsymbol{z}\end{bmatrix}$ clearly holds the same information as the normal equations array $\begin{bmatrix}\boldsymbol{N} & \boldsymbol{h}\end{bmatrix}$ in (21), since $\boldsymbol{N} = \boldsymbol{R}'\boldsymbol{R}$ and $\boldsymbol{h} = \boldsymbol{R}'\boldsymbol{z}$, while R has the same meaning as before; therefore we can use $\boldsymbol{R}$, $\boldsymbol{z}$ and R to describe the current knowledge of $\boldsymbol{u}$.

The agglomeration of two clusters with square root information arrays $\begin{bmatrix}\boldsymbol{R}_i & \boldsymbol{z}_i\end{bmatrix}$ and $\begin{bmatrix}\boldsymbol{R}_j & \boldsymbol{z}_j\end{bmatrix}$ is obtained by minimizing

$$R = \left\|\boldsymbol{R}_i\boldsymbol{u} - \boldsymbol{z}_i\right\|^2 + \left\|\boldsymbol{e}_i\right\|^2 + \left\|\boldsymbol{R}_j\boldsymbol{u} - \boldsymbol{z}_j\right\|^2 + \left\|\boldsymbol{e}_j\right\|^2 = R_i + R_j + \left\| \begin{bmatrix} \boldsymbol{R}_i \\ \boldsymbol{R}_j \end{bmatrix} \boldsymbol{u} - \begin{bmatrix} \boldsymbol{z}_i \\ \boldsymbol{z}_j \end{bmatrix} \right\|^2 \tag{37}$$

Applying an orthogonal transformation $\boldsymbol{Q}$ ($2m \times 2m$) to the augmented information array such that

$$\boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}_i & \boldsymbol{z}_i \\ \boldsymbol{R}_j & \boldsymbol{z}_j\end{bmatrix} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{z} \\ \boldsymbol{0} & \boldsymbol{e}\end{bmatrix} \tag{38}$$

with upper triangular $\boldsymbol{R}$, yields

$$R = R_i + R_j + \left\|\boldsymbol{R}\boldsymbol{u} - \boldsymbol{z}\right\|^2 + \left\|\boldsymbol{e}\right\|^2 \tag{39}$$

which is minimized for $\widehat{\boldsymbol{u}} = \boldsymbol{R}^{-1}\boldsymbol{z}$. From the last equation we also conclude that

$$\Delta(C_i, C_j) = \left\|\boldsymbol{e}\right\|^2 \tag{40}$$

One way to compute the dissimilarity between $\begin{bmatrix}\boldsymbol{R}_i & \boldsymbol{z}_i\end{bmatrix}$ and $\begin{bmatrix}\boldsymbol{R}_j & \boldsymbol{z}_j\end{bmatrix}$ is therefore to triangularize the first m columns of the augmented information matrix, of dimension $2m \times (m+1)$, by orthogonal transformation.[5] A compact and efficient algorithm for the partial triangularization of a matrix, using Householder transformations, is given in Appendix VII.B of [1]. The implementation of this routine requires $(20m^3{+}45m^2{+}37m)/6$ floating-point operations[6] to triangularize the first m columns of an $2m \times (m+1)$ matrix. To this should be added $2m-1$ operations to compute $\|\boldsymbol{e}\|^2$. Thus for $m = 2, 3, 4, 5$, the number of operations needed is 72, 181, 365, and 644. The count for $m = 2$ is about twice what a straightforward implementation of (30) requires, while the present method may be competitive, and certainly more accurate, for higher m.

In practice the transformation in (38) is needed for each component of the spatial coordinate. If only proper motion is considered (not parallax), the left-hand side of the data equations is the same for each component; thus the same orthogonal transformation $\boldsymbol{Q}$ applies. This is efficiently implemented by augmenting (38) with one set of vectors $\boldsymbol{z}_i$, $\boldsymbol{z}_j$ per component. The dissimilarity is the sum of $\|\boldsymbol{e}\|^2$ over the components.

3.5 Additional considerations

3.5.1 Choice of batch size

Cluster analysis is intrinsically an $O(N^2)$ process, where N is the number of observations. This behaviour can be somewhat mitigated by various tricks, such as the NNC algorithm described in Sect. 2.2 and by restricting the search for the nearest neighbour among sorted data, but not completely eliminated. Thus it is preferable to do the cross-matching in batches with small N. The batches cannot be made arbitrarily small, though: obviously they must correspond to areas of the sky that are at least several times the positional uncertainty per observation, including proper motion effects. Thus it is hardly reasonable to consider areas smaller than of order an arcmin. In high-density regions such an area would contain hundreds of stars, in low-density regions only a few stars. However, the use of small areas incurs another penalty: it will be necessary to consider also all observations in a border area around the chosen patch, thus effectively doing the cross-matching in

[5] It would seem possible to construct a more efficient way to compute Δ, by taking into account that $\boldsymbol{R}_i$ and $\boldsymbol{R}_j$ are already in square root form, but I have not found any such algorithm in the literature.

[6] All operations $+$, $-$, $\times$, $/$, $\surd$ counted with equal weight.

slightly overlapping areas. The penalty comes from having to treat some of the data at least twice, because of the overlap, plus some administration to keep track of the status of data in the overlap areas. The relative amount of overhead due to the overlap will decrease as (area)$^{-0.5}$, while the amount of computation per source for the cluster analysis scales as (area), so there should be an optimum size possibly depending on the star density. It is difficult to estimate the optimum size without a detailed implementation of the whole process, including the data access, but it seem unlikely to be much greater than several arcmin. (Data retrieval could be made in larger batches that are then subdivided for the cross-matching.) Thus we may assume that the cross-matching will be made in areas that cover only a very small part of the sky.

3.5.2 Choice of rectangular coordinates

As mentioned in Sect. 3.1, the cross-matching could use any cartesian coordinate system allowing the expansion of stellar motion in a linear model like (11). The two most obvious choices are

- Standard coordinates (ξ, η): these are local rectangular coordinates in the tangent plane of the unit sphere, valid (to sufficient accuracy) over areas that may extend up to a few degrees.
- Three-dimensional coordinates (x, y, z) on the unit sphere: these are global coordinates valid over the whole sphere.

The advantage of standard coordinates is the economy of computation in using them, once they have been computed; the disadvantage is that they are local and perhaps unique for each cross-matching area (as discussed in Sect. 3.5.1). In overlap areas the same data may thus be represented by different standard coordinates, which is clearly inconvenient. These complications, as well as the need to specify tangent point coordinates and making the transformations, are avoided if three-dimensional coordinates are used; on the whole these seem to allow the simpler and safer treatment. The disadvantages of the three-dimensional coordinates are that more space is needed for their storage and more arithmetic operations are required, e.g., for computing dissimilarities (these being quadratic sums over three instead of two coordinate components).

It is not *a priori* obvious which coordinate representation should be chosen. On the other hand, as long as the source model does not go beyond first order (i.e., including position and proper motion, but not parallax), the detailed implementation of the classification and cluster analysis algorithms could be completely general with respect to this choice. It is therefore recommended that both options are retained for the time being.

3.5.3 Use of magnitude criterion

Both the distance measure for classification and the dissimilarity measure for cluster analysis could take into account the magnitude difference between the source and observation, or between two clusters. The extension if straightforward, and only requires the specification of a scale factor to make a magnitude error comparable with an error in position. In uncrowded areas there may not be any need to use the magnitude criterion, but in crowded areas it could significantly improve the cross-matching (cf. Sect. 4.3).

Including the magnitude criterion will of course create problems for variable stars. What may happen is that several sources are created at approximately the same position but with different mean magnitudes. Such cases could however be detected by proper post-processing, cf. Sect. 3.5.5.

The cross-matching algorithm should be flexible enough to permit a manual (or automatic) choice on whether to use the magnitude criterion.

3.5.4 Solar system objects

Many solar system objects move fast enough that their proper motions are detected already with the skymapper/confirmation observation. Until the special cross-matching for solar-system objects has been fully defined, it is not clear if such observations would be filtered out already before the cross-matching procedure considered here. It should be noted however that the proper motion formalism developed in Sect. 3.3 and the subsequent cluster analysis could be applied to solar system objects as well, to the extent that their motions on the sky are uniform over a certain time interval. For example, this might identify successive transits of a fast-moving object on consecutive scans.

3.5.5 Breaking and joining clusters

In addition to the classification and cluster analysis algorithms, the full cross-matching procedure must include criteria and methods to break and join clusters as part of the post-analysis of the clustered observations. One important reason for breaking up a cluster into two (or more) clusters is that it contains quasi-simultaneous observations, i.e., more than one detection made on the same FOV crossing: by definition, a source can only be observed once per FOV transit. (Alternatively, the epoch condition might be built into the classification/clustering algorithms.)

There could be many reasons for joining clusters into a single one. Objects representing distinct sources in an earlier cross-matching may later turn out to be similar enough to be considered the same source. If the magnitude criterion is used, variable stars may produce multiple sources at the same position.

The post-processing algorithms needed to detect and correct these and other cases are at the moment completely undefined.

3.6 Synthesis

The general cross-matching algorithm is a combination of classification and cluster analysis. A natural division between the two procedures could be as follows: first classify as many of the observations as possible (i.e., matching them to known sources); then apply cluster analysis on the remaining (unmatched) observations. The classification must include a special class holding the unmatched observations that will be fed to the cluster analysis. The scheme must be complemented with procedures for pre- and post-processing of the data, handling of overlap areas (Sect. 3.5.1), and for the joining and breaking of clusters (Sect. 3.5.5).

The complete cross-matching could then consist of the following steps. It is assumed that observations and sources (for the input catalogue) are stored in the database by position, e.g., according to the HTM system.

1. Set parameters for the cross-matching, in particular the estimated σ_u and the upper limit on the internal variance of a cluster, whether the cluster analysis should take proper motions into account and if so the relative prior weight (L) of the proper motions
2. Select the region of the sky to be cross-matched (e.g., whole sky)
3. If the selected region is larger than the optimum batch area, subdivide it into batch areas of suitable size. Steps 4–9 are performed for one batch area at a time.
4. Add a margin area along the border, defining the extended batch area
5. Extract all sources in the batch area and all observations in the extended batch area
6. Do the classification of observations against sources
7. For each source, check that all the matched observations were made at distinct epochs; if not, release the more distant observation
8. For all remaining (unmatched) observations, do the cluster analysis
9. For each cluster, check that the observations were made at distinct epochs; if not, release the more dissimilar observation
10. Store the results for the sources/clusters whose positions (at the reference epoch) fall inside the batch area, and mark the corresponding observations as matched; all other clusters are dissolved and the corresponding observations marked as unmatched. The computed source parameters constitute an updated input catalogue.
11. Check the whole region for unmatched observations (except along the border of the region); if there are unmatched observations, goto 6; otherwise stop.

4 Explorative tests of the cluster algorithm

The cluster algorithms described in Sect. 2.2 (without proper motions) and Sect. 3.4.1 (including proper motions) were briefly tested on ad hoc simulated data. The purpose was not to thoroughly test the algorithm on realistic data, but merely to demonstrate that it works *in principle* as expected. There are many details of the implementation that remain to be worked out and very little has been done to optimize the algorithm in terms of the several parameters that may influence the performance (limits for the internal dispersion, weight of prior data, search radius, etc).

Primarily two aspects of the cluster analysis algorithm were tested: the statistical correctness of the cluster assignment as function of stellar density, and the convergence of the first-order model in the presence of a high-proper motion star. The simplicity of the simulated data reflects the limited scope of these tests.

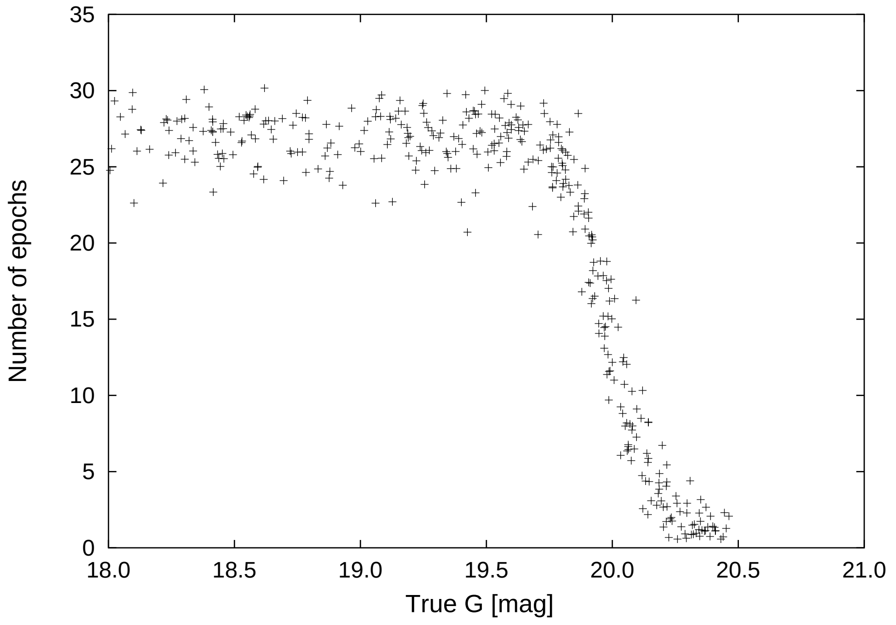

FIGURE 3: Simulation of a fuzzy detection limit: The diagram shows the number of observations (epochs) per star as function of its true magnitude (G). The region was assumed to be scanned 30 times, which was thus the maximum number of observations per star. (The ordinates were randomized to within ± 0.5 units.)

4.1 Simulated data

Data were simulated in two steps: first a random field of sources (stars) was generated, then their observations at specific epochs. Each source was specified by its position x, y (in arcsec) and magnitude G; observations of the same quantities were obtained by adding gaussian noise. A fuzzy detection limit was implemented as described below.

The rectangular stellar field was specified by the limits in x, y and G, and by the stellar density law, which was of the form $\Sigma(G) = \Sigma(20)\exp[\alpha(G-20)]$ with $\alpha = 0.7$.

The epochs of scans across the region were assumed to be equidistant in time – not quite realistic, but sufficient for a first test – with 6 epochs per year during 5 years. Thus, at most 30 observations could be obtained per star. However, whether or not a certain star was observed at a certain epoch was determined by the random number generator, in such a way that the probability of observation was 0.9 for stars well above the detection limit. For fainter stars a fuzzy detection limit was implemented in the following simplistic manner. Given the true magnitude G, the standard error in magnitude per observation was computed as $\sigma_G(G) = \sigma(20)\exp[\beta(G-20)]$, with $\sigma_G(20) = 0.15$ mag and $\beta = 0.6$. For every potential observation a 'detection magnitude' was generated as $G_{\rm det} = G + \sigma_G(G)N(0,1)$, and the star was assumed to be observed only if $G_{\rm det} \leq 20$. In the subsequent analysis, all detected observations were considered, even when a star only received one observation. Figure 3 shows the resulting number of observations as function of the true magnitude for a field with $\sim$400 observed stars.

For each observation, 'observed' values of x, y and G were generated by adding gaussian errors. A fixed standard deviation σ (usually 1 or 0.5 arcsec) was assumed in x, y, while the $\sigma_G(G)$ law given previously was used for the observed magnitude.

4.2 Defining the success rate

For subsequent experiments it is desirable to quantify the success of the cluster analysis in a single number. This can then be used not only to characterize the performance in a given situation, but to tune the parameters of the cluster analysis in order to optimize the performance. It is not immediately obvious how to define such a number.

From the simulations we know of course the correct assignment of every observation to a unique source. Having performed the cluster analysis, each observation has been assigned to a cluster. In the ideal case every cluster corresponds to exactly one source, and in that case it is straightforward to decide for each observation whether it was assigned to the correct source or not. But the more typical result is that there is no one-to-one correspondence between the true sources and derived clusters.

A solution then is to look at observation *pairs*. Among N observations there are $N(N-1)/2$ distinct pairs (i, j), i.e., for $1 \leq i < j \leq N$. For any i, j it is easy to decide (a) if they were generated by the same source in the simulations; and (b) if they were assigned to the same cluster in the analysis. The clustering was completely successful if and only if, for every pair, (a) gives the same answer (true or false) as (b). Thus, define for each pair

$$A_{ij} = \begin{cases} 1 & \text{if } i,\ j \text{ are generated by the same source,} \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

$$B_{ij} = \begin{cases} 1 & \text{if } i,\ j \text{ are in the same cluster,} \\ 0 & \text{otherwise} \end{cases} \tag{42}$$

Then count the number of pairs with different combinations of the two outcomes,

$$M_{00} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (1 - A_{ij})(1 - B_{ij}) \qquad \text{(correctly unpaired)} \tag{43a}$$

$$M_{01} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} (1 - A_{ij})B_{ij} \qquad \text{(incorrectly paired)} \tag{43b}$$

$$M_{10} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} A_{ij}(1 - B_{ij}) \qquad \text{(incorrectly unpaired} \equiv \text{missed)} \tag{43c}$$

$$M_{11} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} A_{ij}B_{ij} \qquad \text{(correctly paired)} \tag{43d}$$

with $M_{00} + M_{01} + M_{10} + M_{11} = N(N-1)/2$. Finally the success rate is calculated as

$$F = \frac{M_{11}}{M_{11} + M_{10} + M_{01}} \tag{44}$$

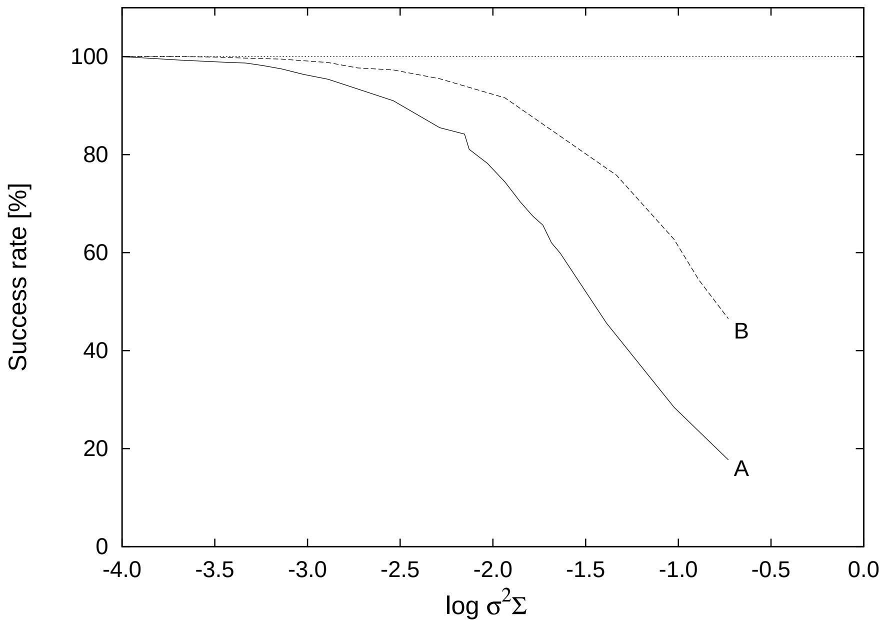

FIGURE 4: Success rate F as function of the dimensionless quantity $\sigma^2\Sigma$, where σ is the positional error per observation and Σ the density of the observed stars. The data were generated without proper motions and analyzed by the zeroth-order model. Curve A was obtained without taking the observed magnitudes into account; for Curve B the dissimilarity included the magnitude mismatch, with a 0.25 mag error equivalent to a positional error of the assumed σ.

A success rate $F = 1$ means that all true pairings were found, and no false pairing. Evaluation of F in some other (extreme or typical) cases suggests that F always gives a fair impression of the success of the cluster analysis. For example, if all but one out of N observations were assigned to the 'correct' cluster, we will get $F \simeq 1 - 4/N$ if N is large. Assigning one cluster to each observation gives $F = 0$, while assigning all observations to the same cluster gives $F \simeq 1/m$, if m is the true number of sources. Finally, if the sources are clumped together two and two in the clusters, so that only half the actual number of sources are found, we would get $F \simeq 0.5$ in the limit of large N.

4.3 Success rate as function of star density

In this first test, a random star field with ∼500 sources was generated (without proper motions) and the density Σ was then varied by scaling the size of the field. The positional uncertainty was fixed at $\sigma = 1$ arcsec. It is clear that, other factors remaining unchanged, the success of the cross-matching should only depend on the dimensionless quantity $\sigma^2\Sigma$, i.e., the expected number of sources in a square of side length σ.

Figure 4 shows the run of the success rate F as function of $\log \sigma^2\Sigma$ (Curve A). A high success rate seems to require that $\sigma^2\Sigma$ is of order 0.001 or less. However, if the magnitude criterion is used (Sect. 3.5.3), a somewhat higher density can be handled (Curve B).

The maximum expected star density is $\Sigma = 3 \times 10^6$ deg^{-2} = 0.23 arcsec^{-2}. It is clear that such a density cannot be successfully analyzed with a positional error of $\sigma = 1$ arcsec (for which $\log \sigma^2\Sigma = -0.64$), but may be possible with $\sigma = 0.1$ arcsec ($\log \sigma^2\Sigma = -2.64$), especially if the magnitude criterion can be used.

4.4 Tests including proper motion

The purpose of these tests was to demonstrate the possibility, at least in principle, to perform cluster analysis on position and proper motion data as described in Sect. 3.4.1. More precisely, we wish to show that the algorithm converges to the correct solution (success rate $F = 1$) as the positional standard error σ is reduced. To this end, a very small field (40×40 arcsec^2) was considered and a moderate star density, 81 000 deg^{-1}, giving only 10 observed sources in the field. The magnitude criterion was not used in these tests. No proper motions were simulated, except for one of the stars (at $G = 18.6$, with 28 observations), which was sometimes given a high proper motion of $\simeq$1.3 arcsec yr^{-1}.

Figures 5–10 show the simulations and results plotted with different symbols in the field. The true position of a source is marked with a big plus sign (+). The observations are marked with small crosses ($\times$), except those of the high-proper motion star (in Figs. 7–10 only), which are marked with small squares ($\Box$). The clusters derived by the cluster analysis are shown as circles of radius $(R/n)^{1/2}$ centred on the cluster centroids.

4.4.1 Tests with the zeroth-order model

For the experiments shown in Figs. 5–8, the cluster analysis was made with the zeroth-order model, i.e., not taking into account proper motion. In Figs. 5–6 the stars did not have any proper motion, and the results converged as expected to the correct pairings: the success rate increased from $F = 0.981$ for $\sigma = 1$ arcsec (Fig. 5) to $F = 1$ for $\sigma = 0.5$ arcsec (Fig. 6).

In Figs. 7–8, the star at $(x, y) \simeq (13, 24)$ arcsec was given a proper motion of (0.6, 1.2) arcsec yr^{-1}. In this case the zeroth-order cluster analysis did not converge to the correct pairings as the positional errors decreased; on the contrary, the data for the high-proper motion star were increasingly fragmented into smaller clusters, as could be expected. This shows that a high-proper motion star could be a problem for the simplest (zeroth-order) clustering algorithm.

4.4.2 Tests with the first-order model

Figure 9 shows the same data as in Fig. 7 but analyzed with the first-order model allowing proper motion. The parameter for the prior information was set to $L = 0.1$ yr. The algorithm now correctly recognized the high-proper motion star and gave a significantly improved success rate ($F = 0.898$ versus 0.810 in Fig. 7). Reducing the positional error to 0.5 arcsec retrieved the correct pairings, $F = 1$ (Fig. 10).

In Figs. 9–10 the clusters are shown elongated by the derived proper motion. It can be noted that the cluster results for the non-proper motion objects were not much deteriorated by the additional degrees of freedom introduced by the first-order model: the derived proper motions were small and did not disturb the cross-matching. Of course, such benign behaviour cannot be expected in severely crowded regions. Nevertheless, the test shows that cluster analysis based on the first-order model may work even when the motions are comparable to the distances between sources.

5 Conclusions

We have shown that the cross-matching problem for Gaia can be solved by application of well-known statistical procedures, namely classification and cluster analysis. Suitable algorithms for this have been identified, modified to accommodate proper motions, and tested on simulated data. Results are promising in terms of performance, but the cluster analysis algorithm in particular may be significantly more demanding in terms of computation than previous algorithms. Much work remains on fitting together the different elements, as outlined in Sect. 3.6, and defining various post-processing algorithms.

References

[1] G.J. Bierman, *Factorization Methods for Discrete Sequential Estimation* (Mathematics in Science and Engineering, Vol. 128), Academic Press, 1977

[2] W.J. Krzanowski, *Principles of Multivariate Analysis: A User's Perspective*, 2nd ed., Oxford University Press, 2000

[3] M.G. Lattanzi, X. Luri, A. Spagna, J. Torra, C. Jordi, F. Figueras, R. Morbidelli, A. Volpicelli, *Cross-matching implementation in the GDAAS context*, GDAAS–TN–005, V1.2, 24 June 2001

[4] C. Lawson & R. Hanson, *Solving Least Squares Problems*, Prentice-Hall, 1974

[5] F. Mignard, C. Bailer-Jones, *A Model for the Gaia Data Analysis Consortium*, DACC–CHAIR–002, V2.0, 26 July 2005

[6] F. Murtagh & A. Heck, *Multivariate Data Analysis*, Astrophysics and Space Science Library, Reidel, 1987

[7] F. Murtagh, *Fionn Murtagh's Multivariate Data Analysis Software and Resources Page*, `http://astro.u-strasbg.fr/∼fmurtagh/mda-sw/`

[8] J.H. Ward, *Hierarchical Grouping to Optimize an Objective Function*, J. Am. Stat. Assoc., 58, 236, 1963

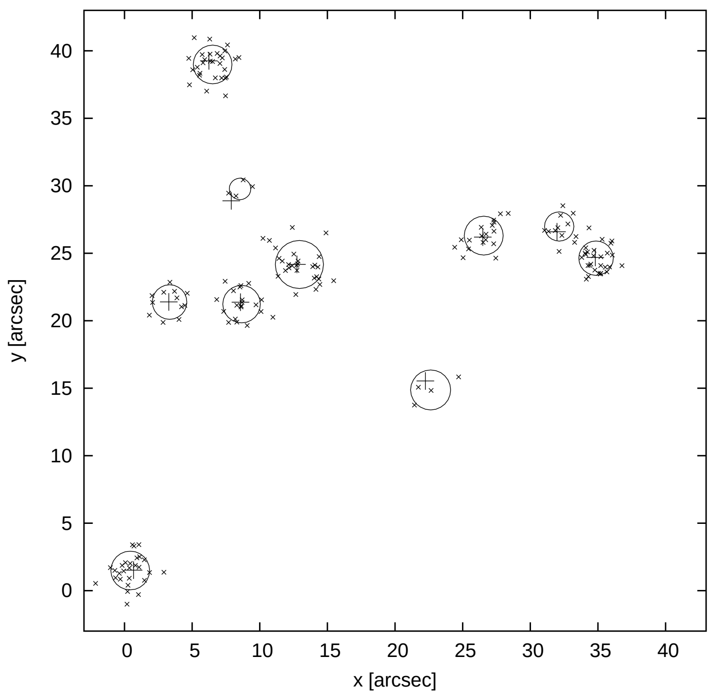

FIGURE 5: Test of the 0th order model on data without proper motion. Positional standard error $\sigma = 1$ arcsec, success rate $F = 0.981$. See text for explanation of symbols.

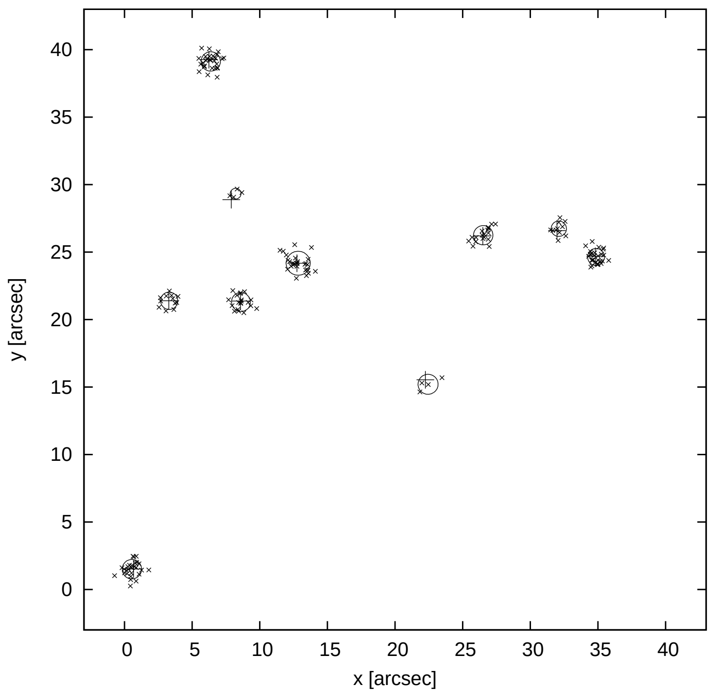

FIGURE 6: Test of the 0th order model on data without proper motion. Positional standard error $\sigma = 0.5$ arcsec, success rate $F = 1$.

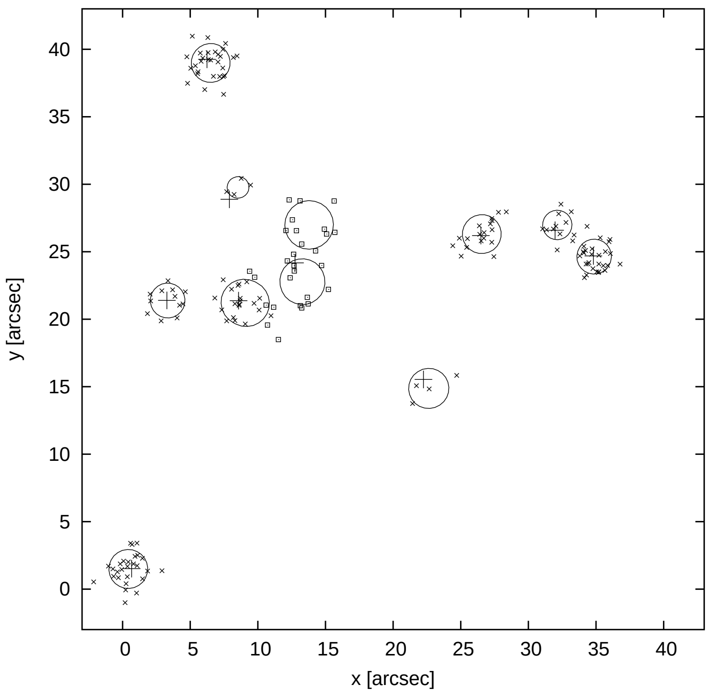

FIGURE 7: Test of the 0th order model on data with proper motion on one star (small squares). Positional standard error $\sigma = 1$ arcsec, success rate $F = 0.810$.

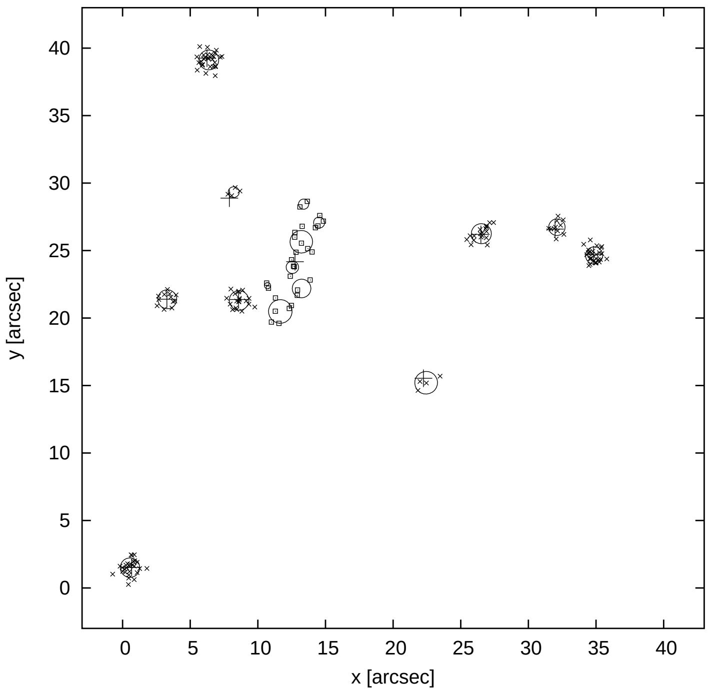

FIGURE 8: Test of the 0th order model on data with proper motion on one star (small squares). Positional standard error $\sigma = 0.5$ arcsec, success rate $F = 0.834$.

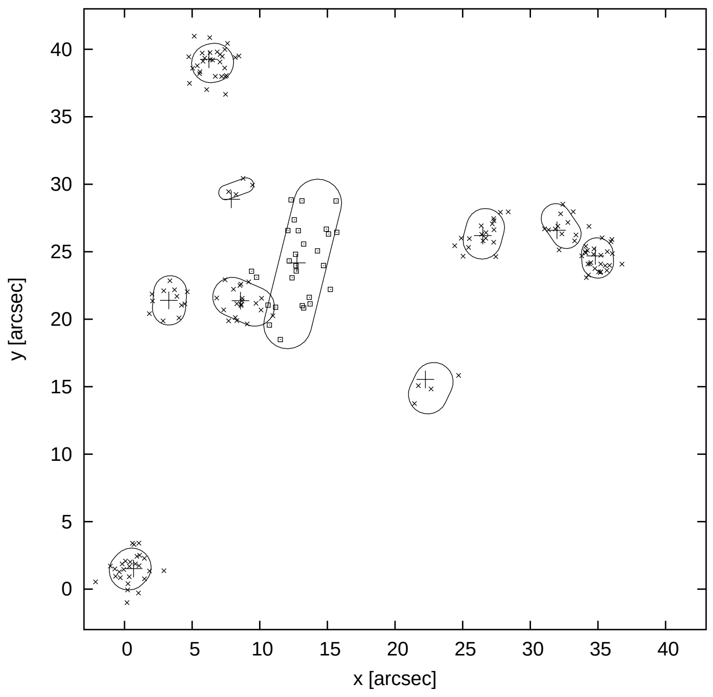

Figure 9: Test of the 1st order model on data with proper motion on one star (same data as in Fig. 7). Positional standard error $\sigma = 1$ arcsec, success rate $F = 0.898$.

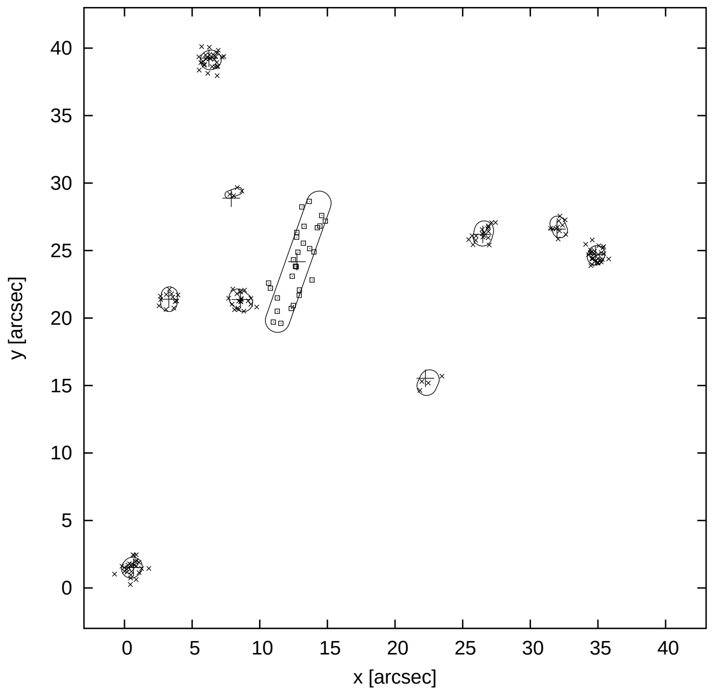

Figure 10: Test of the 1st order model on data with proper motion on one star (same data as in Fig. 8). Positional standard error $\sigma = 0.5$ arcsec, success rate $F = 1$.

Appendix: The Torino Object Matching Algorithm

For reference, the object matching (OM) algorithm currently used in GDAAS [3] is briefly described in this Appendix. It is based on the Fortran routines `posmat` and `match_solver` provided by the Torino group (V1.3, October 2003). The present description is not complete but intended to give a general idea about how the algorithm works.

Basically, the OM algorithm compares two lists of objects to find the best positional matches between them. One list contains *observations*, the other *sources*. The goal is that every observation is assigned to (at most) one source.

For each observation, potentially matching sources are looked for within a given search radius. If there are more than one source within this radius, then the nearest source is selected. However, since such an assignment must be considered uncertain, a record of the remaining potential matches is kept.

Next, the uncertain assignments are reviewed by means of the subroutine `match_solver`. This is invoked for each observation in turn. It checks if there were previous observations assigned to the same sources, and in that case decides on the optimum assignment. If the assignment of a previous observation were changed by this decision, then it is necessary to review the assignment of that observation again. This is done by recursively calling `match_solver`.

A flowchart of the main OM algorithm (`posmat`) is shown in Fig. 11. The `match_solver` subroutine is shown separately in Fig. 12.

The cross-matching is made in standard (tangential plane) coordinates (ξ, η), thus both observations and source data are first transformed to this system. The sources are sorted by η to avoid going through the whole list of sources for every observation. D(obs,src) is the distance between an observation and a source, cf. (2).

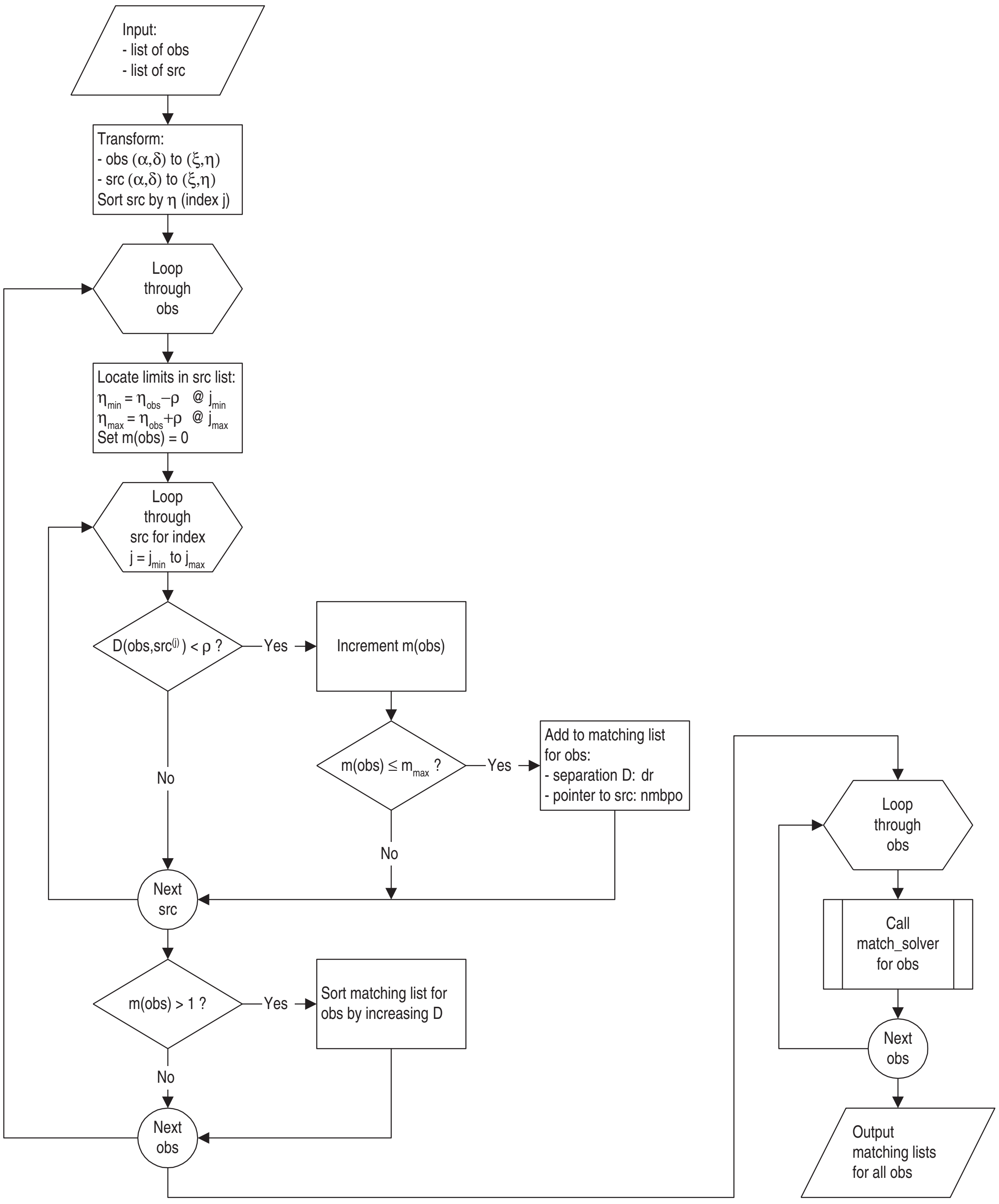
Input:
- list of obs
- list of src
Transform:
- obs (α,δ) to (ξ,η)
- src (α,δ) to (ξ,η)
Sort src by η (index j)
Loop through obs
Locate limits in src list:
$\eta_{min} = \eta_{obs} - \rho$ @ j_{min}
$\eta_{max} = \eta_{obs} + \rho$ @ j_{max}
Set m(obs) = 0
Loop through src for index $j = j_{min}$ to j_{max}
$D(obs, src^{(j)}) < \rho$?
Yes
Increment m(obs)
No
$m(obs) \le m_{max}$?
Yes
Add to matching list for obs:
- separation D: dr
- pointer to src: nmbpo
No
Next src
m(obs) > 1 ?
Yes
Sort matching list for obs by increasing D
No
Next obs
Loop through obs
Call match_solver for obs
Next obs
Output matching lists for all obs

FIGURE 11: Simplified flow chart for the main part of the Torino object matching algorithm, `posmat`. The subroutine `match_solver` is shown in Fig. 12.

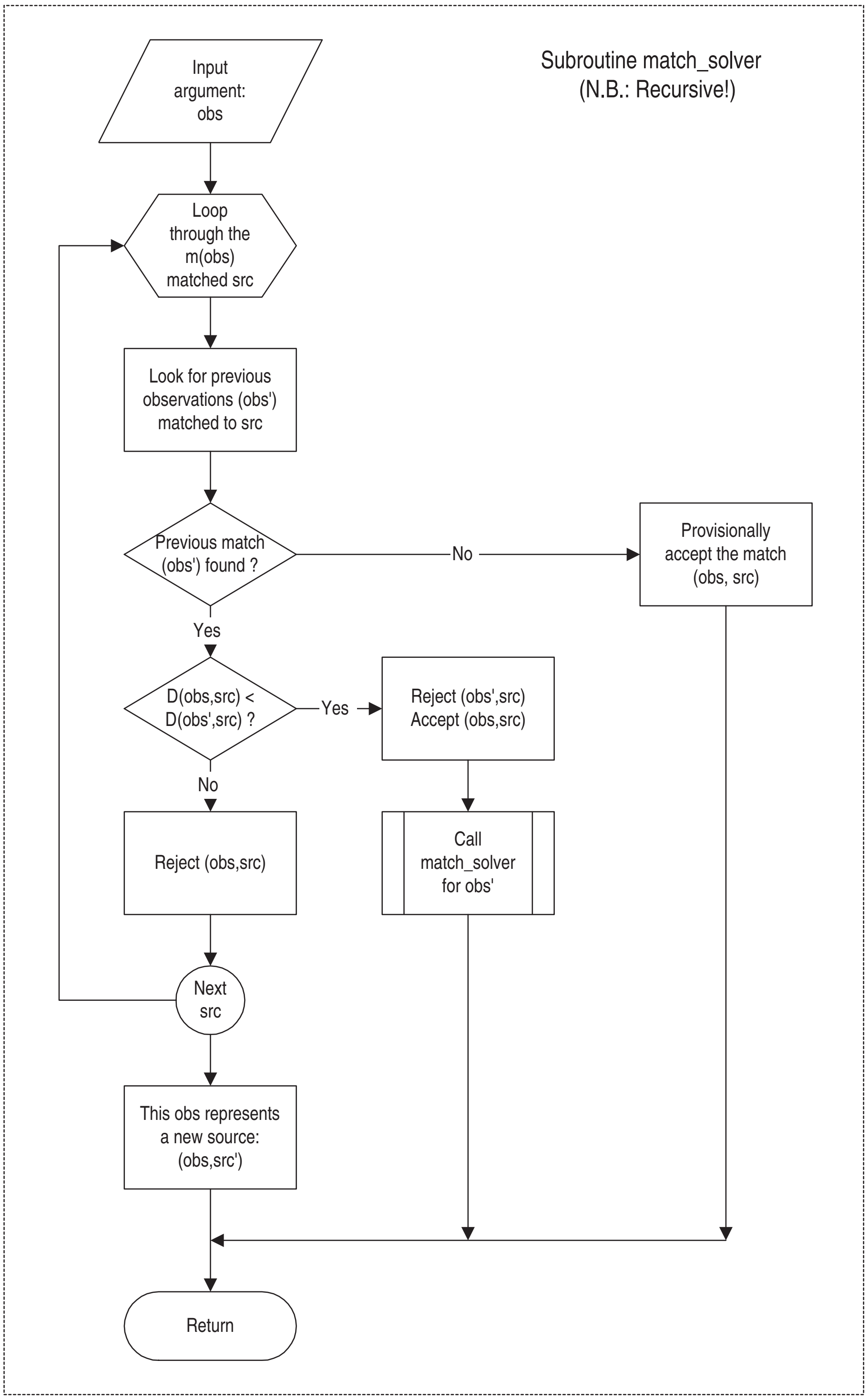
Subroutine match_solver
(N.B.: Recursive!)
Input argument: obs
Loop through the m(obs) matched src
Look for previous observations (obs') matched to src
Previous match (obs') found ?
No
Provisionally accept the match (obs, src)
Yes
D(obs,src) < D(obs',src) ?
Yes
Reject (obs',src) Accept (obs,src)
No
Reject (obs,src)
Call match_solver for obs'
Next src
This obs represents a new source: (obs,src')
Return

FIGURE 12: Simplified flow chart for the recursive `match_solver` subroutine used by the Torino object matching algorithm in Fig. 11.