Draft version December 1, 2016
Preprint typeset using LaTeX style AASTeX6 v. 1.0

C^3, A COMMAND-LINE CATALOG CROSS-MATCH TOOL FOR LARGE ASTROPHYSICAL CATALOGS

Giuseppe Riccio†, Massimo Brescia, Stefano Cavuoti, Amata Mercurio
INAF Astronomical Observatory of Capodimonte - via Moiariello 16, I-80131 Napoli, Italy

Anna Maria di Giorgio, Sergio Molinari
INAF Istituto di Astrofisica e Planetologia Spaziali - Via Fosso del Cavaliere 100, I-00133 Roma, Italy

(Dated: Received 2016 September 06; accepted: 2016 November 14; published 2016 MM DD)

ABSTRACT

Modern Astrophysics is based on multi-wavelength data organized into large and heterogeneous catalogs. Hence, the need for efficient, reliable and scalable catalog cross-matching methods plays a crucial role in the era of the petabyte scale. Furthermore, multi-band data have often very different angular resolution, requiring the highest generality of cross-matching features, mainly in terms of region shape and resolution. In this work we present C^3 (Command-line Catalog Cross-match), a multi-platform application designed to efficiently cross-match massive catalogs. It is based on a multi-core parallel processing paradigm and conceived to be executed as a stand-alone command-line process or integrated within any generic data reduction/analysis pipeline, providing the maximum flexibility to the end-user, in terms of portability, parameter configuration, catalog formats, angular resolution, region shapes, coordinate units and cross-matching types. Using real data, extracted from public surveys, we discuss the cross-matching capabilities and computing time efficiency also through a direct comparison with some publicly available tools, chosen among the most used within the community, and representative of different interface paradigms. We verified that the C^3 tool has excellent capabilities to perform an efficient and reliable cross-matching between large data sets. Although the elliptical cross-match and the parametric handling of angular orientation and offset are known concepts in the astrophysical context, their availability in the presented command-line tool makes C^3 competitive in the context of public astronomical tools.

Keywords: methods: data analysis – catalogs – techniques: miscellaneous – surveys

1. INTRODUCTION

In the last decade we entered the data-intensive era of astrophysics, where the size of data has rapidly increased, reaching in many cases dimensions overcoming the human possibility to handle them in an efficient and comprehensible way. In a very close future petabytes of data will be the standard and, to deal with such amount of information, also the data analysis techniques and facilities must quickly evolve. For example the current exploration of petabyte-scale, multi-disciplinary astronomy and Earth observation synergy, by taking the advantage from their similarities in data analytics, has issued the urgency to find and develop common strategies able to achieve solutions in the data mining algorithms, computer technologies, large scale distributed database management systems as well as parallel processing frameworks (Agrafioti 2012).

Astrophysics is one of the most involved research fields facing with this data explosion, where the data volumes from the ongoing and next generation multi-band and multi-epoch surveys are expected to be so huge that the ability of the astronomers to analyze, cross-correlate and extract knowledge from such data will represent a challenge for scientists and computer engineers. To quote just a few, the ESA Euclid space mission will acquire and process about 100 GBday$^-$1 over at least 6 years, collecting a minimum amount of about 200TB of data (Laureijs et al. 2014); Pan-STARRS (Kaiser 2004) is expected to produce more than 100TB of data; the GAIA space mission will build a $3D$ map of the Milky Way galaxy, by collecting about one petabyte of data in five years (Douglas et al. 2007); the Large Synoptic Survey Telescope (Ivezic 2009) will provide about 20TB/night of imaging data for ten years and petabytes/year of radio data products. Many other planned instruments

†giuseppe.riccio08@gmail.com

and already operative surveys will reach a huge scale during their operational lifetime, such as KiDS (Kilo-Degree Survey; de Jong et al. 2015), DES (Dark Energy Survey, Annis 2013), Herschel-ATLAS (Valiante 2015; Varga-Verebelyi et al. 2016), Hi-GAL (Molinari et al. 2016), SKA (Braun 2015) and E-ELT (Martins et al. 2014).

The growth and heterogeneity of data availability induce challenges on cross-correlation algorithms and methods. Most of the interesting research fields are in fact based on the capability and efficiency to cross-correlate information among different surveys. This poses the consequent problem of transferring large volumes of data from/to data centers, *de facto* making almost inoperable any cross-reference analysis, unless to change the perspective, by moving software to the data (Cavuoti et al. 2012).

Furthermore, observed data coming from different surveys, even if referred to a same sky region, are often archived and reduced by different systems and technologies. This implies that the resulting catalogs, containing billions of sources, may have very different formats, naming schemas, data structures and resolution, making the data analysis to be a not trivial challenge. Some past attempts have been explored to propose standard solutions to introduce the uniformity of astronomical data quantities description, such as in the case of the Uniform Content Descriptors of the Virtual Observatory (IVOA Recommendations 2005).

One of the most common techniques used in astrophysics and fundamental prerequisite for combining multi-band data, particularly sensible to the growing of the data sets dimensions, is the cross-match among heterogeneous catalogs, which consists in identifying and comparing sources belonging to different observations, performed at different wavelengths or under different conditions. This makes cross-matching one of the core steps of any standard modern pipeline of data reduction/analysis and one of the central components of the Virtual Observatory (Malkov et al. 2012).

The massive multi-band and multi-epoch information, foreseen to be available from the on-going and future surveys, will require efficient techniques and software solutions to be directly integrated into the reduction pipelines, making possible to cross-correlate in real time a large variety of parameters for billions of sky objects. Important astrophysical questions, such as the evolution of star forming regions, the galaxy formation, the distribution of dark matter and the nature of dark energy, could be addressed by monitoring and correlating fluxes at different wavelengths, morphological and structural parameters at different epochs, as well as by opportunely determining their cosmological distances and by identifying and classifying peculiar objects. In such context, an efficient, reliable and flexible cross-matching mechanism plays a crucial role. In this work we present C^3 (*Command-line Catalog Cross-match*[1], Riccio et al. 2016), a tool to perform efficient catalog cross-matching, based on the multi-thread paradigm, which can be easily integrated into an automatic data analysis pipeline and scientifically validated on some real case examples taken from public astronomical data archives. Furthermore, one of major features of this tool is the possibility to choose shape, orientation and size of the cross-matching area, respectively, between elliptical and rectangular, clockwise and counterclockwise, fixed and parametric. This makes the C^3 tool easily tailored on the specific user needs.

The paper is structured as follows: after a preliminary introduction, in Sec. 2 we perform a summary of main available techniques; in Sec. 3, the design and architecture of the C^3 tool is described; in sections 4 and 5, the procedure to correctly use C^3 is illustrated with particular reference to the optimization of its parameters; some tests performed in order to evaluate C^3 performance are shown in Sec. 6; finally, conclusions and future improvements are drawn in Sec. 7.

2. CROSS-MATCHING TECHNIQUES

Cross-match can be used to find detections surrounding a given source or to perform one-to-one matches in order to combine physical properties or to study the temporal evolution of a set of sources.

The primary criterion for cross-matching is the approximate coincidence of celestial coordinates (positional cross-match). There are also other kinds of approach, which make use of the positional mechanism supplemented by statistical analysis used to select best candidates, like the bayesian statistics (Budavári & Szalay 2008). In the positional cross-match, the only attributes under consideration are the spatial information. This kind of match is of fundamental importance in astronomy, due to the fact that the same object may have different coordinates in various catalogs, for several reasons: measurement errors, instrument sensitivities, calibration, physical constraints, etc.

In principle, at the base of any kind of catalog cross-match, each source of a first catalog should be compared with all counterparts contained in a second catalog. This procedure, if performed in the naive way, is extremely time consuming, due to the huge amount of sources. Therefore different solutions to this problem have been proposed, taking advantage of the progress in computer science in the field of multi-processing and high per-

[1] The C^3 tool and the user guide are available at the page http://dame.dsf.unina.it/c3.html.

forming techniques of sky partitioning. Two different strategies to implement cross-matching tools basically exist: web and stand-alone applications.

Web applications, like OpenSkyQuery (Nieto et al. 2006), or CDS-Xmatch (Pineau et al. 2011), offer a portal to the astronomers, allowing to cross-match large astronomical data sets, either mirrored from worldwide distributed data centers or directly uploadable from the user local machine, through an intuitive user interface. The end-user has not the need to know how the data are treated, delegating all the computational choices to the backend software, in particular for what is concerning the data handling for the concurrent parallelization mechanism. Other web applications, like ARCHES (Motch 2015), provide dedicated script languages which, on one hand, allow to perform complex cross-correlations while controlling the full process but, on the other hand, make experiment settings quite hard for an astronomer. Basically, main limitation of a web-based approach is the impossibility to directly use the cross-matching tool in an automatic pipeline of data reduction/analysis. In other words, with such a tool the user cannot design and implement a complete automatic procedure to deal with data. Moreover, the management of concurrent jobs and the number of simultaneous users can limit the scalability of the tool. For example, a registered user of CDS-Xmatch has only 500MB disk space available to store his own data (reduced to 100MB for unregistered users) and all jobs are aborted if the computation time exceeds 100 minutes (Boch et al. 2014). Finally, the choice of parameters and/or functional cases is often limited in order to guarantee a basic use by the end-users through short web forms (for instance, in CDS-Xmatch only equatorial coordinate system is allowed).

Stand-alone applications are generally command-line tools that can be run on the end-user machine as well as on a distributed computing environment. A stand-alone application generally makes use of APIs (Application Programming Interfaces), a set of routines, protocols and tools integrated in the code. There are several examples of available APIs, implementing astronomical facilities, such as STIL[2] (Taylor 2006), and astroML[3] (Vanderplas 2012), that can be integrated by an astronomer within its own source code. However, this requires the astronomer to be aware of strong programming skills. Moreover, when the tools are executed on any local machine, it is evident that such applications may be not able to exploit the power of distributed computing, limiting the performance and requiring the storage of the catalogs on the hosting machine, besides the problem of platform dependency.

On the contrary, a ready-to-use stand-alone tool, already conceived and implemented to embed the use of APIs in the best way, will result an off-the-shelf product that the end-user has only to run. A local command-line tool can be put in a pipeline through easy system calls, thus giving the possibility to the end-user to create a custom data analysis/reduction procedure without writing or modifying any source code. Moreover, being an all-in-one package, i.e including all the required libraries and routines, a stand-alone application can be easily used in a distributed computing environment, by simply uploading the code and the data on the working nodes of the available computing infrastructure.

One of the most used stand-alone tools is STILTS[4] (STIL Tool Set, Taylor 2006). It is not only a cross-matching software, but also a set of command-line tools based on the STIL libraries, to process tabular data. It is written in pure Java (almost platform independent) and contains a large number of facilities for table analysis, so being a very powerful instrument for the astronomers. On one hand, the general-purpose nature of STILTS has the drawback to make hard the syntax for the composition of the command line; on the other hand, it does not support the full range of cross-matching options provided by C^3. In order to provide a more user-friendly tool to the astronomers, it is also available its graphical counterpart, Tool for OPerations on Catalogs And Tables[5] (TOPCAT, Taylor 2005), an interactive graphical viewer and editor for tabular data, based on STIL APIs and implementing the STILTS functionalities, but with all the intrinsic limitations of the graphical tools, very similar to the web applications in terms of use.

Regardless the approach to cross-match the astronomical sources, the main problem is to minimize the computational time exploding with the increasing of the matching catalog size. In principle, the code can be designed according to multi-process and/or multi-thread paradigm, so exploiting the hosting machine features. For instance, Lee & Budavári (2013) evaluated to use a multi-GPU environment, designing and developing their own Xmatch tool, (Budavári & Lee 2013). Other studies are focused to efficiently cross-match large astronomical catalogs on clusters consisting of heterogeneous processors including both multi-core CPUs and GPUs, (Jia et al. 2015, Jia & Luo 2016). Furthermore, it is possible to

[2] http://www.star.bris.ac.uk/~mbt/stil/

[3] http://www.astroml.org/

[4] http://www.star.bris.ac.uk/~mbt/stilts/

[5] http://www.star.bristol.ac.uk/~mbt/topcat/

reduce the number of sources to be compared among catalogs, by opportunely partitioning the sky through indexing functions and determining only a specific area to be analyzed for each source. CDS-Xmatch and the tool described in Zhao et al. (2009, p. 604) use Hierarchical Equal Area isoLatitude Pixelisation (HEALPix, Gorski 2005), to create such sky partition. Du et al. (2014), instead, proposed a combined method to speed up the cross-match by using HTM (Hierarchical Triangle Mesh, Kunszt et al. 2001), in combination with HEALPix and by submitting the analysis to a pool of threads.

HEALPix is a genuinely curvilinear partition of the sphere into exactly equal area quadrilaterals of varying shape (see Fig. 3 in Gorski 2005). The base-resolution comprises twelve pixels in three rings around the poles and equator. Each pixel is partitioned into four smaller quadrilaterals in the next level. The strategy of HTM is the same of HEALPix. The difference between the two spatial-indexing functions is that HTM partitioning is based on triangles, starting with eight triangles, 4 on the Northern and 4 on the Southern hemisphere, each one partitioned into four smaller triangles at the next level (see also Fig. 2 in Du et al. 2014). By using one or both functions combined together, it is possible to reduce the number of comparisons among objects to ones lying in adjacent areas.

Finally OpenSkyQuery uses the *Zones* indexing algorithm to efficiently support spatial queries on the sphere, (Gray et al. 2006).

The basic idea behind the *Zones* method is to map the sphere into stripes of a certain height h, called zones. Each object with coordinates (ra, dec) is assigned to a zone by using the formula:

$$zoneID = dec + 90.0/h \quad (1)$$

A traditional B-tree index is then used to store objects within a zone, ordered by *zoneID* and right ascension. In this way, the spatial cross-matching can be performed by using bounding boxes (B-tree ranges) dynamically computed, thus reducing the number of comparisons (Fig. 1 in Nieto et al. 2006). Finally, an additional and expensive test allows to discard false positives.

All the cross-matching algorithms based on a sky partitioning have to deal with the so-called block-edge problem, illustrated in Fig. 1: the objects X and X' in different catalogs correspond to the same object but, falling in different pieces of the sky partition, the cross-matching algorithm is not able to identify the match. To solve this issue, it is necessary to add further steps to the pipeline, inevitably increasing the computational time. For example, the Zhao's tool, (Zhao et al. 2009, p. 604), expands a Healpix block with an opportunely dimensioned border; instead, the algorithm described by Du et al. (2014), combining Healpix and HTM virtual indexing function shapes, is able to reduce the block-edge problem, because the lost objects in a partition may be different from one to another.

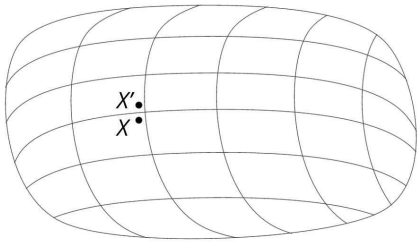

Figure 1. The *block-edge problem*. Objects X and X' in two catalogs. Even if corresponding to the same source, they can be discarded by the algorithm, since they belong to two different blocks of the sky partition.

3. C^3 DESIGN AND ARCHITECTURE

C^3 is a command-line open-source Python script, designed and developed to perform a wide range of cross-matching types among astrophysical catalogs. The tool is able to be easily executed as a stand-alone process or integrated within any generic data reduction/analysis pipeline. Based on a specialized sky partitioning function, its high-performance capability is ensured by making use of the multi-core parallel processing paradigm. It is designed to deal with massive catalogs in different formats, with the maximum flexibility given to the end-user, in terms of catalog parameters, file formats, coordinates and cross-matching functions.

In C^3 different functional cases and matching criteria have been implemented, as well as the most used join function types. It also works with the most common catalog formats, with or without header: Flexible Image Transport System (FITS, version tabular), American Standard Code for Information Interchange (ASCII, ordinary text, i.e. space separated values), Comma Separated Values (CSV), Virtual Observatory Table (VOTable, XML based) and with two kinds of coordinate system, equatorial and galactic, by using STILTS in combination with some standard Python libraries, namely *NumPy*[6] (Van Der Walt 2011), and *PyFITS*[7].

Despite the general purpose of the tool, reflected in a variety of possible functional cases, C^3 is easy to use and to configure through few lines in a single configuration file. Main features of C^3 are the following:

1. *Command line*: C^3 is a command-line tool. It

[6] http://www.numpy.org/

[7] PyFITS is a product of the Space Telescope Science Institute, which is operated by AURA for NASA. http://www.stsci.edu/institute/software_hardware/pyfits

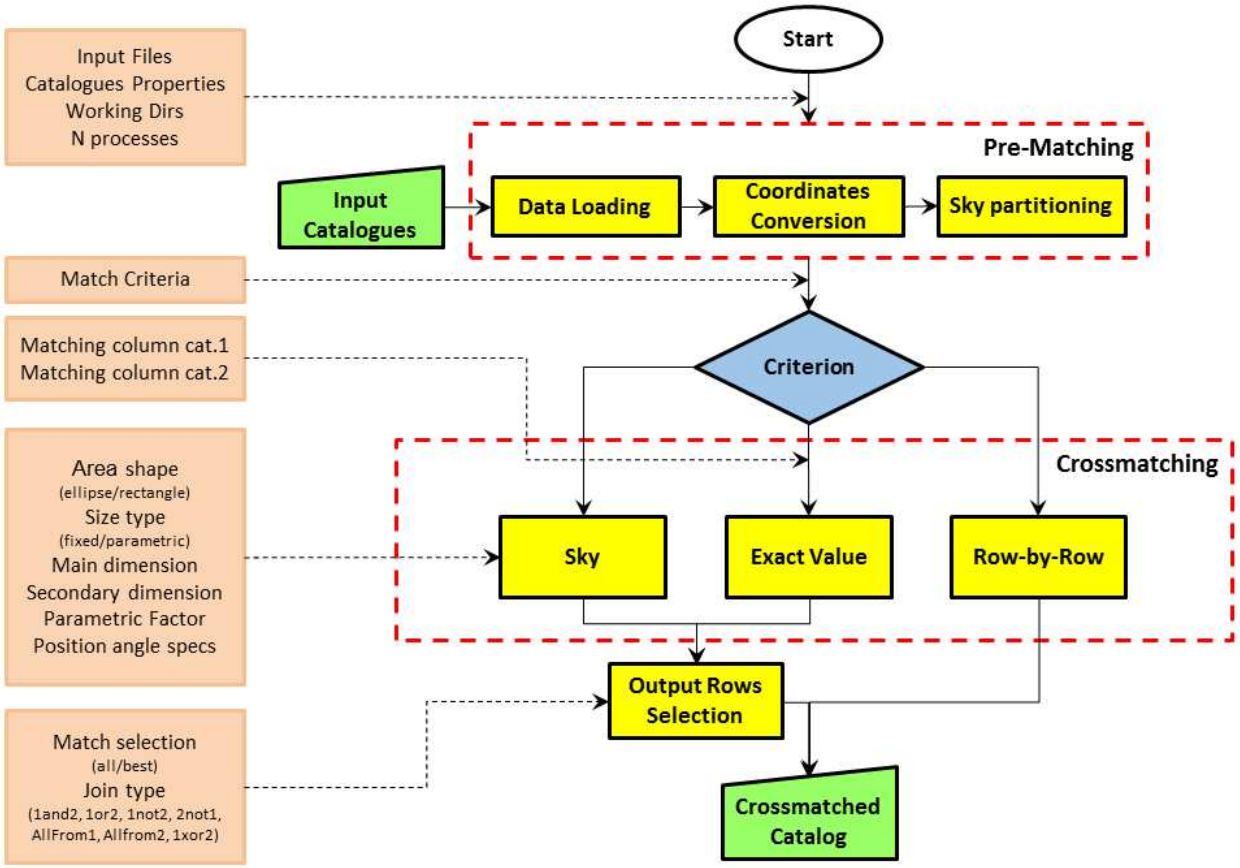

Figure 2. Flowchart of the C^3 tool. The configuration requires few parameters (square panels on the left), according to the chosen match criterion. Currently three different functional cases are available (*Sky*, *Exact Value*, *Row-by-Row*). The pipeline foresees a pre-matching step in order to prepare data for the multiprocess cross-matching phase.

can be used as stand-alone process or integrated within more complex pipelines;

2. *Python compatibility*: compatible with Python 2.7.x and 3.4.x (up to the latest version currently available, 3.5);

3. *Multi-platform*: C^3 has been tested on Ubuntu Linux 14.04, Windows 7 and 10, Mac OS and Fedora;

4. *Multi-process*: the cross-matching process has been developed to run by using a multi-core parallel processing paradigm;

5. *User-friendliness*: the tool is very simple to configure and to use; it requires only a configuration file, described in Sec. 4.

The internal cross-matching mechanism is based on the sky partitioning into cells, whose dimensions are determined by the parameters used to match the catalogs. The sky partitioning procedure is described in 3.3.1. The Fig. 2 shows the most relevant features of the C^3 processing flow and the user parameters available at each stage.

3.1. *Functional cases*

As mentioned before, the user can run C^3 to match two input catalogs by choosing among three different functional cases:

1. *Sky*: the cross-match is done within sky areas (elliptical or rectangular) defined by the celestial coordinates taken from catalog parameters;

2. *Exact Value*: two objects are matched if they have the same value for a pair of columns (one for each catalog) defined by the user;

3. *Row-by-Row*: match done on a same row-ID of the two catalogs. The only requirement here is that the input catalogs must have the same number of records.

The positional cross-match strategy of the C^3 method is based on the same concept of the Q-FULLTREE approach, an our tool introduced in Becciani et al. (2015)

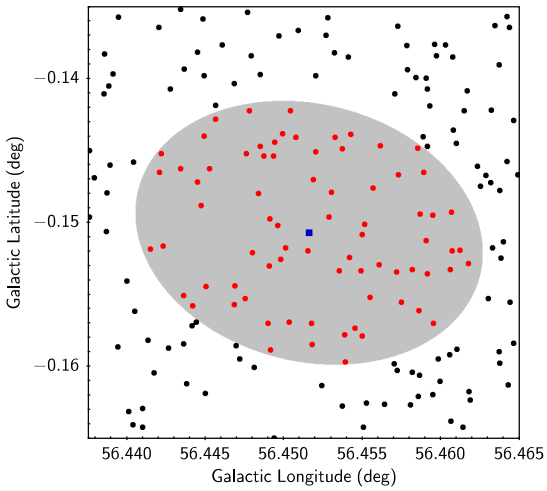

Figure 3. Graphical representation of an elliptical cross-match between two catalogs: the gray ellipse represents the matching region defined by the FWHMs referred to an object of first catalog (squared dot in the center of the ellipse); all other points (belonging to the second catalog), that fall into the region defined by the ellipse (red or light gray dots), are matching with the central object.

and Sciacca et al. (2016): for each object of the first input catalog, it is possible to define an elliptical, circular or rectangular region centered on its coordinates, whose dimensions are limited by a fixed value or defined by specific catalog parameters. For instance, the two Full Width at Half Maximum (FWHM) values in the catalog can define the two semi-axes of an ellipse or the couple width and height of a rectangular region. It is also possible to have a circular region, by defining an elliptical area having equal dimensions. Once defined the region of interest, the next step is to search for sources of the second catalog within such region, by comparing their distance from the central object and the limits of the area (for instance, in the elliptical cross-match the limits are defined by the analytical equation of the ellipse).

In the *Sky* functional case, the user can set additional parameters in order to characterize the matching region and the properties of the input catalogs. In particular, the user may define:

1. the shape (elliptical or rectangular) of the matching area, i.e. the region, centered on one of the matching sources, in which to search the objects of the second catalog;

2. the dimensions of the searching area. They can be defined by fixed values (in arcseconds) or by parametric values coming from the catalog. Moreover, the region can be rotated by a position angle (defined as fixed value or by a specific column present in the catalog);

3. the coordinate system for each catalog (galactic, icrs, fk4, fk5) and its units (degrees, radians, sexagesimal), as well as the columns containing information about position and designation of the sources.

An example of graphical representation of an elliptical cross-match is shown in Fig. 3.

In the *Exact Value* case, the user has to define only which columns (one for each input catalog) have to be matched, while in the most simple *Row-by-Row* case no particular configuration is needed.

3.2. *Match selection and join types*

C^3 produces a file containing the results of the cross-match, consisting into a series of rows, corresponding to the matching objects. In the case of *Exact value* and *Sky* options, the user can define the conditions to be satisfied by the matched rows to be stored in the output. First, it is possible to retrieve, for each source, all the matches or only the best pairs (in the sense of closest objects, according to the match selection criterion); then, the user can choose different join possibilities (in Fig. 4 the graphical representation of available joins is shown):

1 **and** 2**:** only rows having an entry in both input catalogs, (Fig. 4a);

1 **or** 2**:** all rows, matched and unmatched, from both input catalogs, (Fig. 4b);

All from 1 **(All from** 2**):** all matched rows from catalog 1 (or 2), together with the unmatched rows from catalog 1 (or 2), (Fig. 4c-d);

1 **not** 2 **(**2 **not** 1**):** all the rows of catalog 1 (or 2) without matches in the catalog 2 (or 1), (Fig. 4e-f);

1 **xor** 2**:** the "exclusive or" of the match - i.e. only rows from the catalog 1 not having matches in the catalog 2 and viceversa, (Fig. 4g).

3.3. *Execution phases*

Any experiment with the C^3 tool is based on two main phases (see Fig. 2):

1. *Pre-matching:* this is the first task performed by C^3 during execution. The tool manipulates input catalogs to extract the required information and prepare them to the further analysis;

2. *Matching:* after data preparation, C^3 performs the matching according to the criteria defined in the configuration file.

Finally, the results are stored in a file, according to the match criterion described in Sec. 3.2, and all the temporary data are automatically deleted.

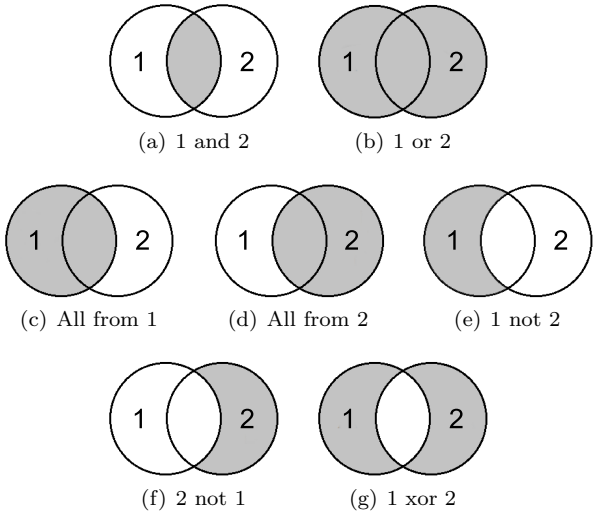

Figure 4. Join types available for C^3 output. Panel (a): rows both in catalog 1 and catalog 2 (1 and 2); (b) all rows of catalog 1 and catalog 2 (1 or 2); (c) all rows of catalog 1 (all from 1); (d) all rows of catalog 2 (all from 2); (e) rows in catalog 1 not matched with catalog 2 (1 not 2); (f) rows in catalog 2 not matched with catalog 1 (2 not 1); (g) rows from the catalog 1 not having matches in the catalog 2 and viceversa (1 xor 2).

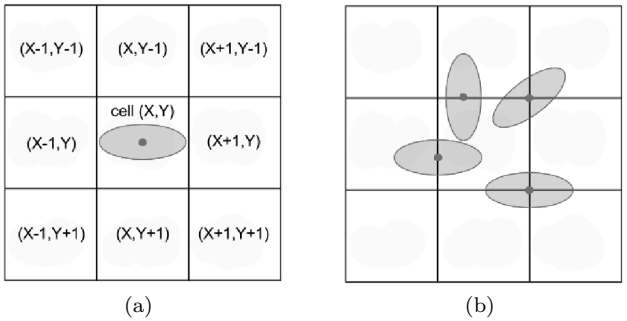

Figure 5. The C^3 sky partitioning method. The sky is partitioned in cells whose dimensions are determined by the maximum value assumed by the main dimension of the matching area or by the *minimum partition cell size* parameter (panel a). Each object of the second catalog is assigned to a cell: a match between a source and the ellipse defined by the first catalog object can happen only in the nine cells surrounding it (panel b).

3.3.1. *Pre-matching*

This is the preliminary task performed by C^3 execution. During the pre-matching phase, C^3 performs a series of preparatory manipulations on input data. First of all, a validity check of the configuration parameters and input files. Then it is necessary to split the data sets in order to parallelize the matching phase and improve the performance. In the *Exact Value* functional case only the first input catalog will be split, while in the *Sky* case both data sets will be partitioned in subsets. In the latter case, C^3 makes always use of galactic coordinates expressed in degrees, thus converting them accordingly if expressed in different format.

When required, the two catalogs are split in the following way: in the first catalog all the entries are divided in groups, whose number depends on the multi-processing settings (see Sec. 4), since each process is assigned to one group; in the second catalog the sky region defined by the data set is divided into square cells, by assigning a cell to each entry, according to its coordinates (Fig. 5).

We used the Python multiprocess module to overcome the GIL problem, by devoting particular care to the granularity of data to be handled in parallel. This implies that the concurrent processes do not need to share resources, since each process receives different files in input (group of object of the 1st catalog and cells) and produces its own output. Finally the results are merged to produce the final output.

The partitioning procedure on the second catalog is based on the dimensions of the matching areas: the size of the unit cell is defined by the maximum dimension that the elliptical matching regions can assume. If the "Size type" is "parametric", then the maximum value of the columns indicated in the configuration is used as cell size; in the case of "fixed" values, the size of the cell will be the maximum of the two values defined in the configuration (Fig. 5a). In order to optimize the performance, the size of the unit cell cannot be less than a threshold value, namely the *minimum partition cell size*, which the user has to set through the configuration file. The threshold on the cell size is required in order to avoid the risk to divide the sky in too many small areas (each one corresponding to a file stored on the disk), which could slow down the cross-matching phase performance. In Sec. 5 we illustrated a method to optimize such parameter as well as the number of processes to use, according to the hosting machine properties.

Once the partitioning is defined, each object of the second catalog is assigned to one cell, according to its coordinates. Having defined the cells, the boundaries of an elliptical region associated to an object can fall at maximum in the eight cells surrounding the one including the object, as shown in Fig. 5b. This prevents the block-edge problem previously introduced.

3.3.2. *Matching*

Once the data have been properly re-arranged, the cross-match analysis can start. In the *Row-by-Row* case, each row of the first catalog is simply merged with the corresponding row of the second data set through a serial procedure. In the other functional cases, the cross-matching procedure has been designed and implemented to run by using parallel processing, i.e. by assigning to each parallel process one group generated in the previous phase. In the *Exact Value* case, each object of the group is compared with all the records of the second cat-

alog and matched according to the conditions defined in the configuration file.

In the *Sky* functional case, the matching procedure is slightly more complex. As described in Sec. 3.1, the cross-match at the basis of the C^3 method is based on the relative position of two objects: for each object of the first input catalog, C^3 defines the elliptical/rectangular region centered on its coordinates and dimensions. Therefore a source of the second catalog is matched if it falls within such region.

In practice, as explained in the pre-matching phase, having identified a specific cell for each object of a group, this information is used to define the minimum region around the object used for the matching analysis. The described choice to set the dimensions of the cells ensures that, if a source matches with the object, it must lie in the nine cells surrounding the object (also known as Moore's neighborhood, Gray 2003, see also Fig. 5b). Therefore it is sufficient to cross-match an object of a group only with the sources falling in nine cells.

In the *Sky* functional case, C^3 performs a cross-matching of objects lying within an elliptical, circular or rectangular area, centered on the sources of the first input catalog. The matching area is characterized by 6 configuration parameters defining its shape, dimensions and orientation. In Fig. 6 is depicted a graphical representation of two matching areas (elliptical and rectangular) with the indication of its parameters.

In particular, to define the orientation of the matching area, C^3 requires two further parameters besides the offset and the value of the position angle, representing its orientation. The position angle, indeed, is referred, by default, to the greatest axis of the matching area with a clockwise orientation. The two additional parameters give the possibility to indicate, respectively, the correct orientation (clockwise/counterclockwise) and a shift angle (in degrees).

Finally, the results of the cross-matching are stored in a file, containing the concatenation of all the columns of the input catalogs referred to the matched rows. In the *Sky* functional case the column reporting the separation distance between the two matching objects is also included.

4. CONFIGURATION

The tool C^3 is interfaced with the user through a single configuration file, to be properly edited just before the execution of any experiment. If the catalogs do not contain the source's Designation/ID information, C^3 will automatically assign an incremental row-ID to each entry as object designation.

For the *Sky* functional case, assuming that both input catalogs contain the columns reporting the object coordinates, C^3 is able to work with galactic and equatorial (icrs, fk4, fk5) coordinate systems, expressed in the following units: degrees, radians or sexagesimal.

If the user wants to use catalog information to define the matching region (for instance, the FWHMs or a radius defined by the instrumental resolution), obviously the first input catalog must contain such data. The position angle value/column is, on the contrary, an optional information (default is 0^o, clockwise).

C^3 is conceived for a community as wide as possible, hence it has been designed in order to satisfy the requirement of user-friendliness. Therefore, the configuration phase is limited to the editing of a setup file,[8] containing all the information required to run C^3. This file is structured in sections, identified by square brackets: the first two are required, while the others depend on the particular use case. In particular, the user has to provide the following information:

1. the input files and their format (FITS, ASCII, CSV or VOTable);
2. the name and paths of the temporary, log and output files;
3. the match criterion, corresponding to one of the functional cases (*Sky, Exact Value, Row-by-Row*).

C^3 gives also the possibility to set the number of processes running in parallel, through an optional parameter which has as default the number of cores of the working machine (minus one left available for system auxiliary tasks).

4.1. *Sky functional case*

The configuration for the *Sky* functional case foresees the setup of specific parameters of the configuration file: those required to define the shape and dimensions of the matching area, the properties of the input catalogs already mentioned in Sec. 3.1, coordinate system, units as well as the column indexes for source coordinates and designation. In addition, a parameter characterizing the sky partitioning has to be set (see Sec. 3.3.1 for further information).

The parameters useful to characterize the matching area are the following:

Area shape: it can be elliptical or rectangular (circular is a special elliptical case);

Size type: the valid entries are *fixed* or *parametric*. In the first case, a fixed value will be used to determine the matching area; in the second, the dimen-

[8] C^3 can also automatically generate a dummy configuration file that could be used as template.

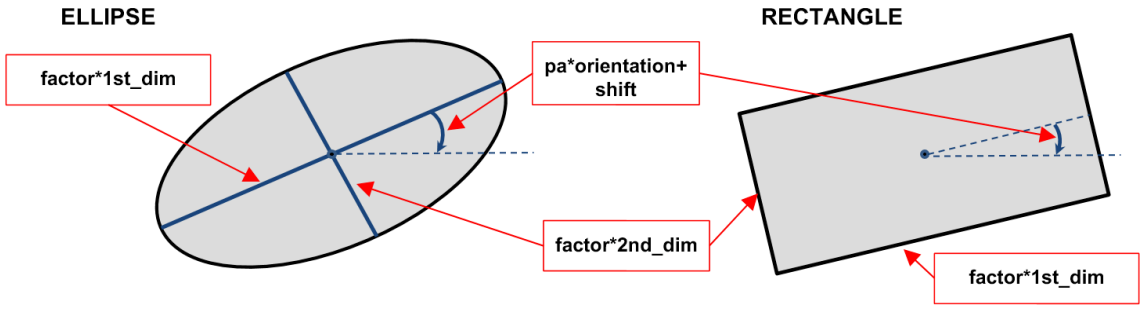

Figure 6. Configuration of C^3 Matching Area: it can be elliptical (circular as special case) or rectangular; its dimensions, defined in the configuration file as *matching area 1st and 2nd dimension*, represent the ellipse axes or width and height of the rectangle, multiplied, in the case of parametric size type, by a user defined *parametric factor*; the position angle is characterized by a value (in degree) and two additional parameters, respectively, *orientation* and *shift*.

sions and inclination of the matching area will be calculated by using catalog parameters;

First and second dimensions of matching area: the axes of the ellipse or width and height of the rectangular area. In case of fixed "Size type", they are decimal values (in arcsec), otherwise, they represent the index (integer) or name (string) of the columns containing the information to be used;

Parametric factor: it is required and used only in the case of parametric "Size type". It is a decimal number factor to be multiplied by the values used as dimensions, in order to increase or decrease the matching region, as well as useful to convert their format;

Pa column/value: it is the position angle value (in the "fixed" case, expressed in degrees) or the name/ID of the column containing the position angle information (in the "parametric" case);

Pa settings: the position angle, which in C^3 is referred, by default, to the main axis of the matching area (greatest) with a clockwise orientation. The two parameters defined here give the possibility to indicate the correct orientation (clockwise/counterclockwise) and a shift angle (in degrees).

The user has also to specify which rows must be included in the output file, by setting the two parameters indicating the match selection and the join type, as described in Sec. 3.2.

4.2. *Exact Value functional case*

For the *Exact value* functional case it is required to set the name or id of the columns used for the match for both input files. The user has also to specify which rows must be included in the output file, by setting the two parameters indicating the match selection and the join type, as described in Sec. 3.2.

4.3. *Row-by-Row functional case*

For the *Row-by-Row* functional case, no other settings are required. The only constrain is that both catalogs must have the same number of entries.

5. COMPUTATIONAL OPTIMIZATION PROCEDURE

As reflected from the description of C^3, the choice of the best values for its internal parameters (in particular the number of parallel processes and the minimum cell size, introduced in Sec. 3.3.1), is crucial to obtain the best computational efficiency. This section is dedicated to show the importance of this choice, directly depending on the features of the hosting machine. In the following tests we used a computer equipped with an Intel(R) Core(TM) $i5-4460$, with one $3.20GHz$, $4-core$ CPU, 32 GB of RAM and hosting Ubuntu Linux 14.04 as operative system (OS) on a standard Hard Disk Drive. We proceeded by performing two different kinds of tests:

1. a series of tests with a fixed value for the minimum cell size ($100''$) and different values of the number of parallel processes;
2. a second series by using the best value of number of parallel processes found at previous step and different values for the minimum cell size.

The configuration parameters used in this set of tests are reported in Table 1. The input data sets are two identical catalogs (CSV format) consisting of $100,000$ objects extracted from the UKIDSS GPS public data (Lucas et al. 2008), in the range of galactic coordinates $l \in [50, 60]$, $b \in [-1, 1]$. Each record is composed by 125 columns. The choice to cross-match a catalog with itself represent the worst case in terms of cross-matching computational time, since each object matches at least with itself.

Parameter	Value
Area Shape	Ellipse
Size Type	Fixed
1st dimension (Major axis)	$5''$
2nd dimension (Secondary axis)	$5''$
Position Angle settings	0^o
Coordinate System	Galactic (deg)
Match Selection	best
Join type	1 and 2

Table 1. C^3 settings in the first set of tests performed to evaluate the impact of the number of parallel processes and the minimum cell size configuration parameters on the execution time. The choice of same dimensions for the ellipse axes was due to perform a fair comparison with STILTS and CDS-Xmatch, which allow only circular cross-matching.

By setting "Match Selection" as best and "Join Type" as 1 and 2 (see Table 1), we obtained an output of $100,000$ objects matched with themselves as expected. We also performed all the tests by using a "random shuffled" version of the same input catalog, obtaining the same results. This demonstrates that the C^3 output is not affected by the particular order of data in the catalogs.

As expected, the number of parallel processes affects the partitioning of the first catalog. In particular, if a too large value is selected, it induces a negative impact on the computational efficiency, causing a bottleneck due to the higher frequency of disk access.

The results of these tests, shown in Table 2, confirm that the best choice of the concurrent processes is not the highest one. In fact, although the serial case ($N = 1$) is obviously the worst result, the computational time reaches the minimum with $N = 256$, from which it starts to increase. The overall *speedup* achieved in the best case is $\sim 7\times$ with respect to the serial case.

The computational time of the pre-matching phase appears almost constant in all tests, because this portion of the C^3 code is not parallel in the current version of the tool. The small time fluctuations of such phase are due to the unpredictable status of the hosting machine during the tests. The output creation phase, depending on the number of temporary files produced by each concurrent cross-matching process, reaches an almost constant value, mainly imposed by the serial nature of this phase.

Once the best number of concurrent processes has been chosen, we proceeded by looking for the value of the minimum cell size that provides the best result in terms of computational time. The number of subsets, in which the first input catalog has to be divided, depends on the number of parallel processes, while the minimum cell size determines the granularity of the second catalog, corresponding to the resolution of the sky partitioning. A too high cell size implies a partition with few large areas; a too small value causes the generation of a too large number of regions with very few objects.

The parameters used in this set of tests are the same of the previous step (see Table 1), with the number of concurrent processes fixed to $N = 256$. We decided to vary the cell size between $25''$ and $200''$. The results of the test are reported in Table 3.

In this case, the pre-matching phase is, as expected, slightly affected by the choice of the cell size, because the region has to be divided in a different number of cells. While the computational time of the output phase, on the contrary, is not affected by the cell size. The duration of the cross-matching phase reaches a minimum at $50''$, $75''$ and $100''$, where the minimum of the total computational time, and hence the best performance, is reached using $100''$ as minimum cell size.

In more general terms, the described example demonstrates that, in order to obtain the best computational performance, the configuration requires a series of heuristics to reach the best compromise between the granularity of the parallel processing and the scheduling management of the OS. As rule of thumb, the best results can be obtained by choosing the number of parallel processes limited between 10 and 100 times the number of cores of the hosting machine.

For what concerns the minimum cell size, in the previous example we considered 20 square degrees, with $100,000$ objects, thus a density of ~ 1 object in 2600 square arcsecs. Since the best results have been obtained with a cell size of $100''$, we obtained ~ 7 objects per cell. By extrapolating from our tests, the best cell size, conditioning the sky partitioning resolution, should be chosen between 2 and 10 sources per cell. Of course, this heuristic range depends on the specific density of the involved fields.

6. TESTING ON ASTROPHYSICAL DATA

In order to validate the results of C^3 and evaluate its performance in terms of cross-matching reliability and computational time efficiency, we performed several tests on real data. In particular, two kinds of tests have been executed, both using the most complex functional case *Sky*. The first set of tests (Sec. 6.1), has been performed to validate the C^3 results in terms of matching capability, through a comparison with other available cross-matching tools, for instance, STILTS and CDS-Xmatch. The second set of tests (Sec. 6.2), has been used to evaluate the computational time efficiency, by varying the dimensions of the input data sets (both in terms of rows and columns), again through a comparison with the other tools.

6.1. *Cross-matching validation tests*

TestID	N processes	Pre-matching time (s)	Cross-matching time (s)	Output creation time (s)	Total time (s)
NP1	1	29	11	412	452
NP4	4	28	3	108	139
NP8	8	28	3	72	102
NP16	16	28	3	54	85
NP20	20	28	3	50	81
NP32	32	28	3	45	76
NP64	64	28	3	40	71
NP100	100	28	3	40	71
NP128	128	28	3	39	70
NP256	256	28	4	37	69
NP512	512	28	4	38	70
NP1024	1024	28	5	38	72
NP2048	2048	28	8	39	74
NP2560	2560	28	11	38	77
NP3072	3072	28	13	38	79

Table 2. The computational time of the whole process (column 6) and of each phase of the tool execution (columns from 3 to 5), for experiments with the same configuration but different number of parallel processes (column 2). Here the minimum cell size is fixed to 100 arcsecs. The input data sets are two identical catalogs consisting of 100,000 objects extracted from the UKIDSS GPS public data. Each record is composed by 125 columns.

TestID	Cell size (arcseconds)	Pre-matching time (s)	Cross-matching time (s)	Output creation time (s)	Total time (s)
TH25	25	38	4	36	77
TH50	50	31	3	41	76
TH75	75	29	3	41	74
TH100	100	28	3	40	71
TH125	125	27	4	41	72
TH150	150	27	5	42	73
TH175	175	27	5	41	74
TH200	200	26	6	41	74

Table 3. The computational time of the whole process (column 6) and of each phase of the tool execution (columns from 3 to 5), for experiments with the same configuration but different minimum cell size (column 2). These tests have been done by fixing the number of parallel processes to $N = 256$. The input data sets are two identical catalogs consisting of 100,000 objects extracted from the UKIDSS GPS public data. Each record is composed by 125 columns.

In order to assess the reliability of the cross-matches produced by C^3, we performed an intensive test campaign. In this section we report the most significative examples which well represent the behavior of the tool. This set of tests has been performed by applying our tool on two data sets with variable number of objects and by comparing the results with those obtained by other applications representative of different paradigms: stand-alone command-line (STILTS, release 3.0-7), GUI (TOPCAT, release 4.2.3) and web application (CDS-XMatch[9]).

The first input catalog has been extracted by the UKIDSS GPS data in the range of galactic coordinates $l \in [40, 50]$, $b \in [-1, 1]$, while the second input catalog has been extracted by the GLIMPSE *Spitzer* Data, (Benjamin et al. 2003 and Churchwell et al. 2009), in the same range of coordinates. From each catalog, different subsets with variable number of objects have been extracted. In particular, data sets with, respectively, 1000, 10,000, 100,000, 1,000,000 and 10,000,000 objects have been created from the first catalog, while, from second catalog, data sets with 1000, 10,000, 100,000 and 1,000,000 rows have been extracted. Then, each subset of first catalog has been cross-matched with all the subsets of the second catalog. For uniformity of comparison, due to the limitations imposed by CDS-XMatch in terms of available disk space, it has been necessary to limit to only 3 the number of columns for all the subsets involved in the tests performed to compare C^3 and CDS-XMatch (for instance, ID and galactic coordinates). For the same reason, the data set with 10^7 rows has not been used in the comparison between C^3 and CDS-XMatch.

The common internal configuration used in these tests

[9] http://cdsxmatch.u-strasbg.fr/xmatch

ID	N_{input1}	N_{input2}	C^3, CDS-XMatch, STILTS/TOPCAT (all)	C^3, STILTS/TOPCAT (best)
T1	1000	1000	0	0
T2	1000	10,000	1	1
T3	1000	100,000	5	5
T4	1000	1,000,000	116	116
T5	10,000	1000	0	0
T6	10,000	10,000	14	14
T7	10,000	100,000	116	116
T8	10,000	1,000,000	1260	1248
T9	100,000	1000	12	12
T10	100,000	10,000	136	136
T11	100,000	100,000	1212	1211
T12	100,000	1,000,000	12,711	12,758
T13	1,000,000	1000	141	137
T14	1,000,000	10,000	1295	1267
T15	1,000,000	100,000	12,701	12,416
T16	1,000,000	1,000,000	126,965	123,261
T17	10,000,000	1000	191	169
T18	10,000,000	10,000	1995	1755
T19	10,000,000	100,000	19,717	17,235
T20	10,000,000	1,000,000	196,310	171,775

Table 4. Cross-matching results of C^3, STILTS/TOPCAT and CDS-Xmatch, for different dimensions of input catalogs (columns 2 and 3). Column 4 reports the number of matches of the three tools in the case of *all* matching selection criterion (tests T17-T20 have not been performed for CDS-Xmatch), while column 5 reports the matches found using the *best* criterion. In both cases all tools provided exactly the same number of matches in the whole set of tests.

is shown in Table 1, except for the "Match Selection" parameter. There was, in fact, the necessity to set it to *all* for uniformity of comparison with the CDS-Xmatch tool (which makes available only this option). Then the *best* type has been used to compare C^3 with STILTS and TOPCAT. Furthermore, in all the tests, the number of parallel processes was set to 256 and the minimum cell size to $100''$, corresponding to the best conditions found in the optimization process of C^3 (see Sec. 5). Finally, we chose same dimensions of the ellipse axes in order to be aligned with other tools, which allow only circular cross-matching areas.

Concerning the comparison among C^3 and the three mentioned tools, in the cases of both *all* and *best* types of matching selection, all tools provided exactly the same number of matches in the whole set of tests, thus confirming the reliability of C^3 with respect to other tools (Table 4).[10]

6.2. *Performance tests*

In terms of computational efficiency, C^3 has been evaluated by comparing the computational time of its cross-matching phase with the other tools. The pre-matching and output creation steps have been excluded from the comparison, because strongly dependent on the host computing infrastructure. The other configuration parameters have been left unchanged (Table 1). The complete setup for the described experiments is reported in the Appendix.

In Fig. 7 we show the computational time of the cross-matching phase for C^3 and STILTS, as function of the incremental number of rows (objects) in the first catalog, and by varying the size of the second catalog in four cases, spanning from 1000 to $1,000,000$ rows. In all diagrams, it appears evident the difference between the two tools, becoming particularly relevant with increasing amounts of data.

In the second set of tests performed on the C^3 cross-matching phase and STILTS, the computational time has been evaluated as function of the incremental number of columns of the first catalog (from the minimum required 3 up to 125, the maximum number of columns of catalog 1), and by fixing the number of columns of the second catalog in five cases, respectively, 3, 20, 40, 60 and 84, which is the maximum number of columns for catalog 2. In terms of number of rows, in all cases both catalogs were fixed to $1,000,000$ of entries. In Fig. 8 the results only for 3 and 84 columns of catalog 2 are reported, showing that C^3 is almost invariant to the increasing of columns, becoming indeed faster than STILTS from a certain amount of columns. Such trend is confirmed in all the other tests with different num-

[10] For uniformity of comparison, due to the limitations imposed by CDS-XMatch, the data set with 10^7 rows has not been used.

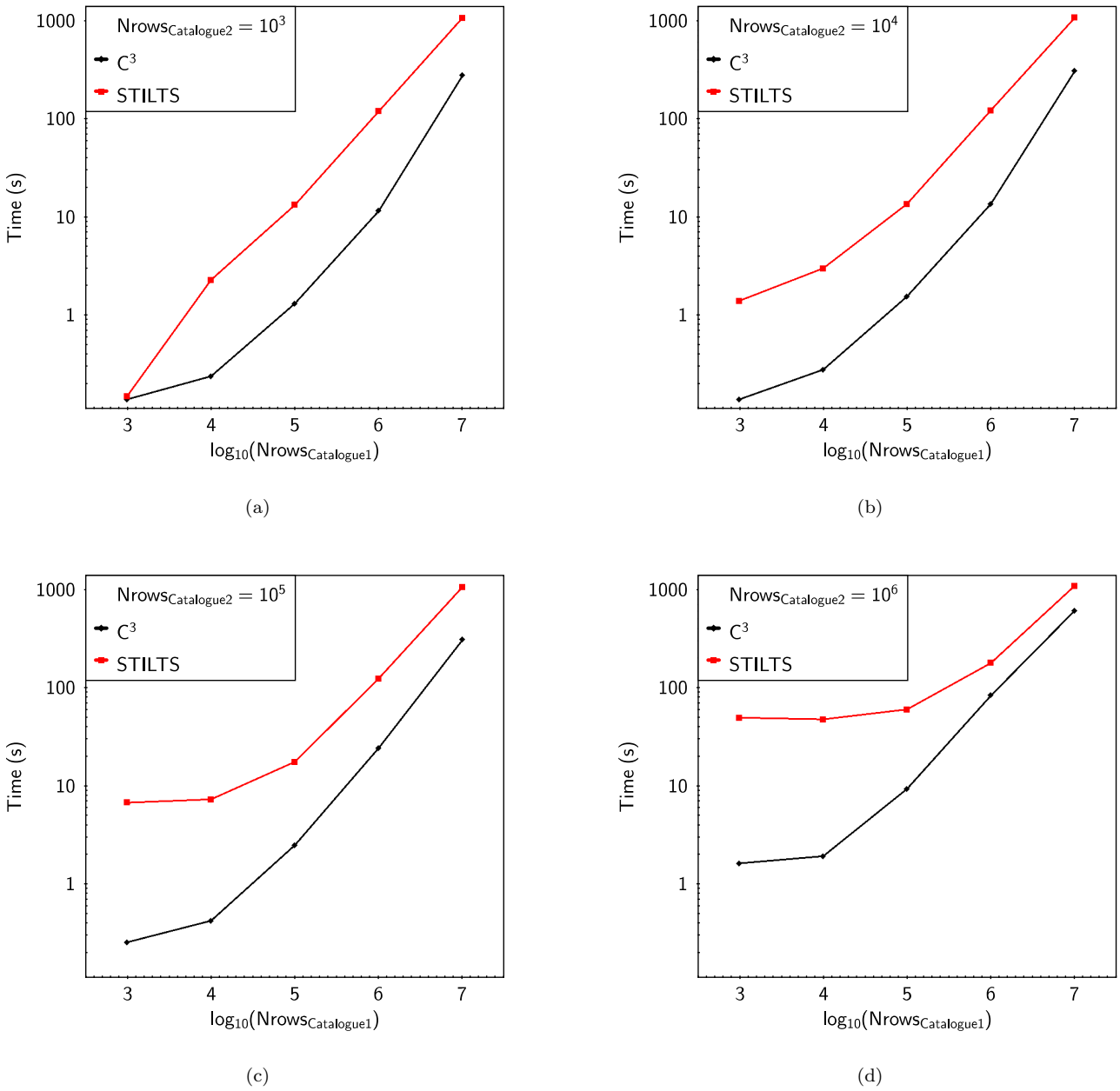

Figure 7. Computational time trends as function of the number of rows of the first input catalog for the C^3 cross-matching phase (black) and STILTS (red or gray) for four different dimensions of the second catalog: (a) 1000 rows, (b) 10,000 rows, (c) 100,000 rows, (d) 1,000,000 rows.

ber of columns of the second catalog. This behavior appears particularly suitable in the case of massive catalogs. Finally, in the case of two FITS input files instead of CSV files, STILTS computational time as function of the number of columns is constant and slightly faster than C^3.

In the last series of tests, we compared the computational efficiency between the C^3 cross-matching phase and CDS-Xmatch. In this case, due to the limitation of the catalog size imposed by CDS-Xmatch, the tests have been performed by varying only the number of rows from 1000 to 1,000,000 as in the analogous tests with STILTS (except the test with 10,000,000 rows), fixing the number of columns to 3. Moreover, in this case, the cross-matching phase of C^3 has been compared with the duration of the phase *execution* of the CDS-Xmatch experiment, thus ignoring latency time due to the job submission, strongly depending on the network status and the state of the job queue, but taking into account the whole job execution. The results, reported in Fig. 9,

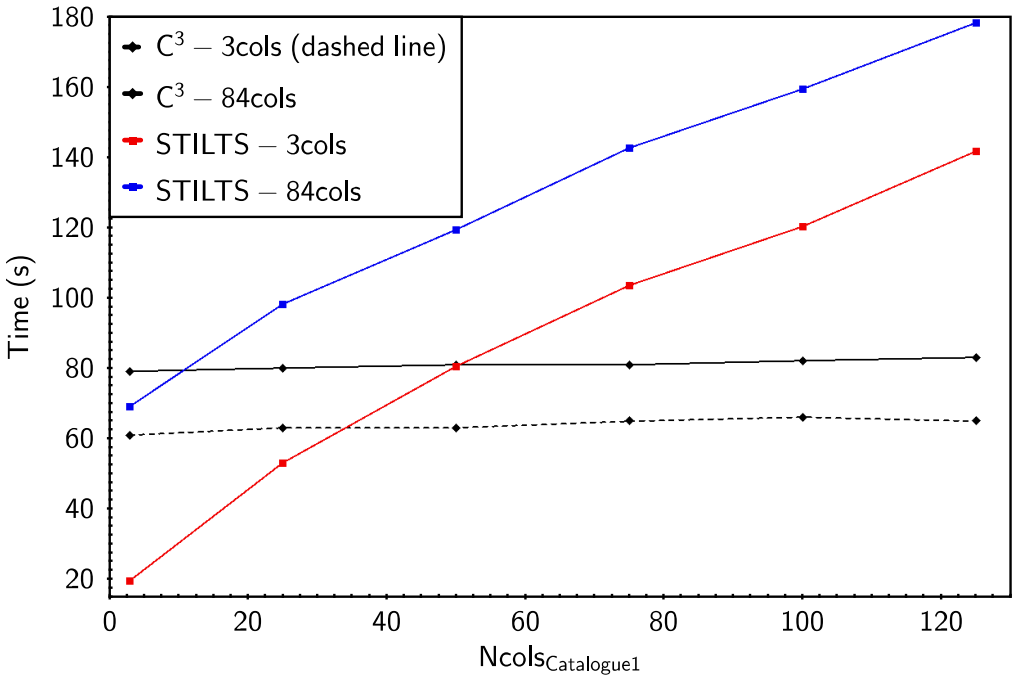

Figure 8. Computational time as function of the number of columns of the first input catalog for the C^3 cross-matching phase and STILTS, considering a second catalog with 3 (black dashed line for C^3, red or light gray line for STILTS) and 84 (black line for C^3 and blue or dark gray for STILTS) columns.

show a better performance of C^3, although less evident when both catalogs are highly increasing their dimensions, where the differences due to the different hardware features become more relevant.

At the end of the test campaign, two other kinds of tests have been performed: (i) the verification of the portability of C^3 on different OSs and (ii) an analysis of the impact of different disk technology on the computing time efficiency of the tool.

In the first case, we noted, as expected, a decreasing of C^3 overall time performance on the Windows versions (7 and 10), with respect to same tests executed on Linux versions (Ubuntu and Fedora) and MAC OS. On average C^3 execution was ~ 20 times more efficient on Linux and MAC OS than Windows. This is most probably due to the different strategy of disk handling among various OSs, particularly critical for applications, like cross-matching tools, which make an intensive use of disk accesses.

This analysis induced us to compare two disk technologies: HDD (Hard Disk Drive) vs SSD (Solid State Disk). Both kinds of disks have been used on a sample of the tests previously described, revealing on average a not negligible increasing of computing time performance in the SSD case of ~ 1.4 times with respect to HDD. For clarity, all test results presented in the previous sections have been performed on the same HDD.

7. CONCLUSIONS AND FUTURE DEVELOPMENTS

In this paper we have introduced C^3, a new scalable tool to cross-match astronomical data sets. It is a multi-platform command-line Python script, designed to provide the maximum flexibility to the end users in terms of choice about catalog properties (I/O formats and coordinates systems), shape and size of matching area and cross-matching type. Nevertheless, it is easy to configure, by compiling a single configuration file, and to execute as a stand-alone process or integrated within any generic data reduction/analysis pipeline.

In order to ensure the high-performance capability, the tool design has been based on the multi-core parallel processing paradigm and on a basic sky partitioning function to reduce the number of matches to check, thus decreasing the global computational time. Moreover, in order to reach the best performance, the user can tune on the specific needs the shape and orientation of the matching region, as well as tailor the tool configuration to the features of the hosting machine, by properly setting the number of concurrent processes and the resolution of sky partitioning. Although elliptical cross-match and the parametric handling of angular orientation and offset are known concepts in the astrophysical

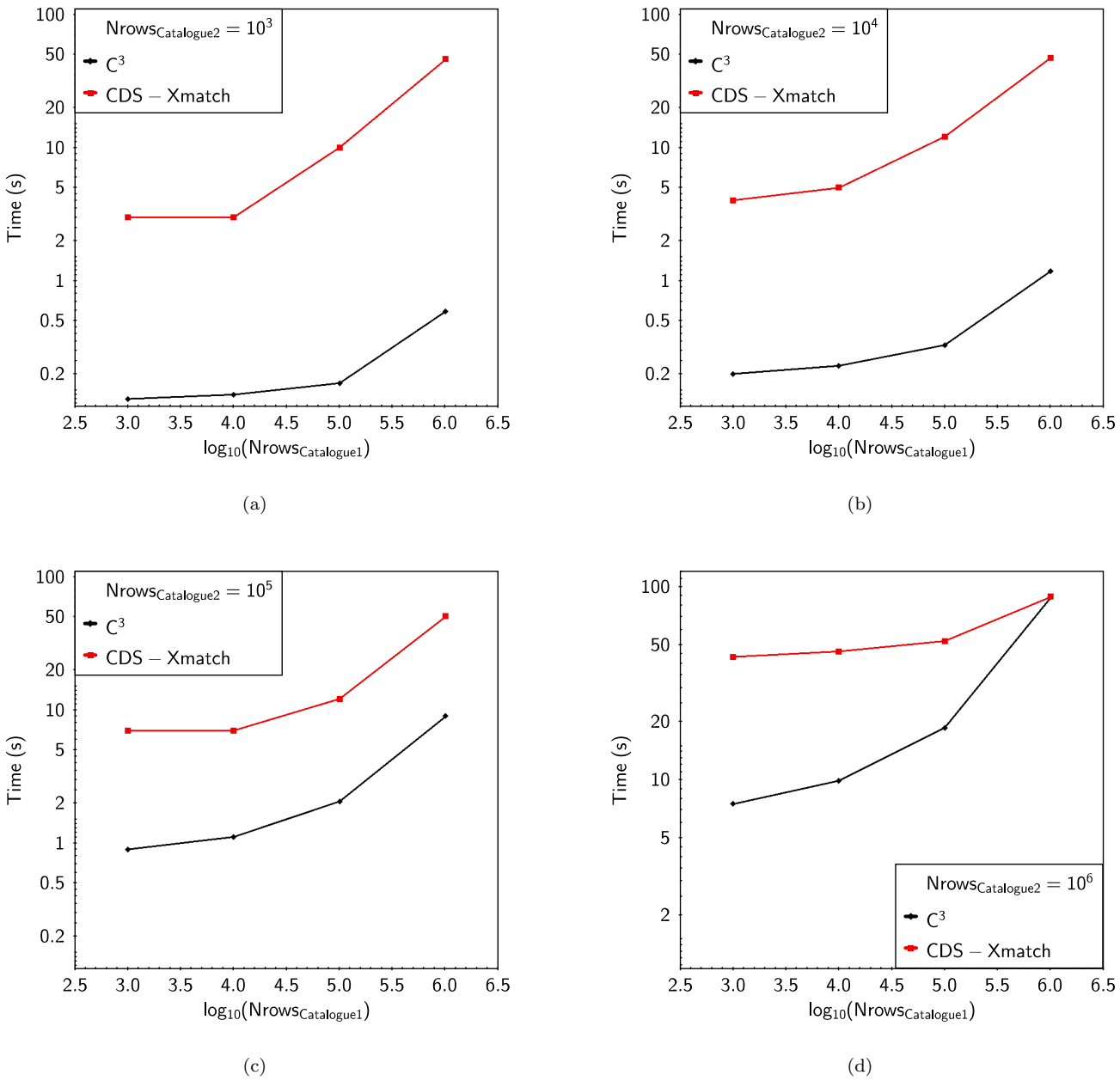

Figure 9. Computational time as function of the number of rows of the first input catalog for the C^3 cross-matching phase (black) and CDS-Xmatch (red or gray) for four different dimensions of the second catalog: (a) 1000 rows, (b) 10,000 rows, (c) 100,000 rows, (d) 1,000,000 rows. Considering only the cross-matching step of CDS-Xmatch, its performance are better than C^3.

context, their availability in the presented command-line tool makes C^3 competitive in the context of public astronomical tools.

A test campaign, done on real public data, has been performed to scientifically validate the C^3 tool, showing a perfect agreement with other publicly available tools. The computing time efficiency has been also measured by comparing our tool with other applications, representative of different paradigms, from stand-alone command-line (STILTS) and graphical user interface (TOPCAT) to web applications (CDS-Xmatch). Such tests revealed the full comparable performance, in particular when input catalogs increase their size and dimensions.

For the next release of the tool, the work will be mainly focused on the optimization of the pre-matching and output creation phases, by applying the parallel processing paradigm in a more intensive way. Moreover, we are evaluating the possibility to improve the sky partitioning efficiency by optimizing the calculation of the

minimum cell size, suitable also to avoid the block-edge problem.

The C^3 tool, (Riccio et al. 2016), and the user guide are available at the page http://dame.dsf.unina.it/c3.html.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referee for extremely valuable comments and suggestions. MB and SC acknowledge financial contribution from the agreement ASI/INAF I/023/12/1. MB, AM and GR acknowledge financial contribution from the 7th European Framework Programme for Research Grant FP7-SPACE-2013-1, *ViaLactea - The Milky Way as a Star Formation Engine*. MB and AM acknowledge the PRIN-INAF 2014 *Glittering kaleidoscopes in the sky: the multifaceted nature and role of Galaxy Clusters*.

REFERENCES

Agrafioti, I. 2012, From the Geosphere to the Cosmos, Synergies with Astroparticle Physics, Astroparticle Physics for Europe (ASPERA), Contributed Volume, http://www.aspera-eu.org

Annis, J. T., 2013, in American Astronomical Society Meeting Abstracts #221, DES Survey Strategy and Expectations for Early Science, 221, id.335.05

Becciani, U., Bandieramonte, M., Brescia, M., et al. 2015, in Proc. ADASS XXV Conf., Advanced Environment for Knowledge Discovery in the VIALACTEA Project, in press. (arXiv:1511.08619)

Benjamin, R. A., Churchwell, E., Babler, B.L., et al. 2003, PASP, 115, 953, doi: 10.1086/376696

Boch, T., Pineau, F. X., & Derriere, S. 2014, CDS xMatch service documentation, http://cdsxmatch.u-strasbg.fr/xmatch/doc/

Braun, R. 2015, in Proc. of "The many facets of extragalactic radio surveys: towards new scientific challenges" (EXTRA-RADSUR2015). 20-23 October 2015. Bologna, Italy. http://pos.sissa.it/cgi-bin/reader/conf.cgi?confid=267, id.34

Budavári, T., & Lee, M. A. 2013, Xmatch: Gpu enhanced astronomic catalog cross-matching, Astrophysics Source Code Library, record ascl:1303.021

Budavári, T., & Szalay, A. S. 2008, ApJ, 679, 301

Cavuoti, S., Brescia, M., Longo, G. 2012, Proc. SPIE, 8451, 845103, doi: 10.1117/12.925321

Churchwell, E., Babler, B.L., Meade, M.R. et al. 2009, PASP, 121, 213

de Jong, J. T. A., Verdoes Kleijn, G. A., Boxhoorn, D. R., et al. 2015, A&A, 582, A62

Douglas, J., de Bruijne, J., O'Flaherty, K., et al. 2007, ESA Bulletin, 132, 26

Du, P., Ren, J. J., Pan, J. C., Luo, A. 2014, SCPMA, 57, 577

Gorski, K. M., Hivon, E., Banday, A. J., et al. 2005, ApJ622, 759

Gray, J., Nieto-Santisteban, M. A. & Szalay, A. S. 2006, The Zones Algorithm for Finding Points-Near-a-Point or Cross-Matchin Spatial Datasetes, Microsoft Tech. Rep.: MSR-TR-2006-52

Gray, L. 2003, Not. Amer. Math. Soc. 50, 200

Ivezic, Z., 2009, in APS April Meeting Abstracts, LSST: the physics of the dark universe, 54, W4.003, http://adsabs.harvard.edu/abs/2009APS..APR.W4003I

IVOA Recommendation 2005, An IVOA Standard for Unified Content Decriptors Version 1.1 (http://adsabs.harvard.edu/abs/2005ivoa.spec.0819D)

Jia, X. & Luo, Q. 2016, in Proc. 28th Int. Conf. on Scientific and Statistical Database Management (SSDBM '16), ed. P. Baumann et al. (New York, NY, ACM), 12, doi: 10.1145/2949689.2949705

Jia, X., Luo, Q. & Fan, D. 2015, in Proc. IEEE XXI Int. Conf. on Parallel and Distributed Systems (ICPADS), 617, doi: 10.1109/ICPADS.2015.83

Kaiser, N., 2004, Proc. SPIE, 5489, 11

Kunszt, P. Z., Szalay, A. S., Thakar, A. R. in Proc. MPA/ESO/MPE Workshop, eds Banday, A. J., Zaroubi, S., Bartelmann, M. (Berlin: Springer), 631, doi: 10.1007/10849171_83

Laureijs, R., Racca, G., Stagnaro, L., et al. 2014, in Proc. SPIE, 9143, 91430H, doi: 10.1117/12.2054883

Lee, M. A., & Budavári, T., 2013, in ASP Conf. Ser. 475, Proc. Astronomical Data Analysis Software and Systems XXII Conf., Cross-Identification of Astronomical Catalogs on Multiple GPUs, ed. Friedel, D. N., (San Francisco, CA:ASP), 235

Lucas, P. W, Hoare, M. G, Longmore, A., et al. 2008, MNRAS, 391, 136

Malkov, O., Dluzhnevskaya, O., Karpov, S., et al. 2012, BaltA, 21, 319

Martins, C. J. A. P., Leite, A. C. O., Pedrosa, P. O. J., 2014, in Statistical Challenges in 21st Century Cosmology, Fundamental Cosmology with the E-ELT, Proc. of the International Astronomical Union, IAU Symposium, ed. Heavens, A., Starck, J.-L. & Krone-Martins, A., 306, 385-387, doi: 10.1017/S1743921314013441

Molinari, S., Schisano, E., Elia, D., et al. 2016, A&A 591, A149, doi: 10.1051/0004-6361/201526380

Motch, C., & Arches Consortium, 2015, in Astronomical Data Analysis Software and Systems XXIV, The ARCHES Project, ed. A. R. Taylor and E. Rosolowsky (San Francisco: Astronomical Society of the Pacific), 437

Nieto-Santisteban, M. A., Thakar, A. R., Szalay, A. S., 2006, Cross-Matching Very Large Datasets (Baltimore, MD: Johns Hopkins University)

Pineau, F. X., Boch, T., & Derriere, S. 2011, in ASP Conf. Ser. 442, Proc Astronomical Data Analysis Software and Systems XX, ed. Evans, I. N., Accomazzi, A., Mink, D. J., & Rots, A. H. (San Francisco, CA:ASP), 85

Riccio, G., Brescia, M., Cavuoti, S., Mercurio, A. 2016, C3: Command-line Catalog Crossmatch for modern astronomical surveys, Astrophysics Source Code Library, record ascl:1610.006

Sciacca, E., Vitello, F., Becciani, U., et al., 2016, Milky Way analysis through a Science Gateway: Workflows and Resource Monitoring, Proceedings of 8th International Workshop on Science Gateways, June 2016, Rome, Italy, submitted to CEUR-WS, http://ceur-ws.org, ISSN: 1613-0073.

Taylor, M. B., 2005, in ASP Conf. Ser. 347, Astronomical Data Analysis Software and Systems XIV, ed. P. Shopbell, M. Britton, & R. Ebert (San Francisco, CA:ASP), 29

Taylor, M. B., 2006, in ASP Conf. Ser. 351, Astronomical Data Analysis Software and Systems XV, ed. C. Gabriel et al. (San Francisco, CA:ASP), 666

Valiante, E. 2015, in IAU General Assembly, Meeting 29, The Herschel-ATLAS survey: main results and data release, 22, 2257414

Van Der Walt, S., Colbert, S. C. & Varoquaux, G., 2011, CSE, 13, 22

Vanderplas, J. T., Connolly, A. J., Ivezić, Ž & Gray, A. 2012, in Proc. Conf. on Intelligent Data Understanding (CIDU), Introduction to astroML: Machine learning for astrophysics, 47-54, doi: 10.1109/CIDU.2012.6382200

Varga-Verebelyi, E., Dobos, L., Budavari, T., 2016, in From Interstellar Clouds to Star-Forming Galaxies: Universal Processes?, IAU Symposium 315, Herschel Footprint Database and Service, eprint arXiv:1602.01050

Zhao, Q., Sun, J., Yu, C., et al., 2009, in Algorithms and Architectures for Parallel Processing, Proc. of 9th International Conference, ICA3PP 2009, A paralleled large-scale astronomical cross-matching function, eds. Arrems, H., Chang, S.,-L., 604-614, ISBN: 978-3-642-03095-6

APPENDIX

A. CONFIGURATION FILE EXAMPLE

This appendix reports the configuration file as used in the example described in Sec. 6.2. The text preceded by the semicolon is a comment.

```
[I/O Files]
Input catalog 1: ./input/ukidss.csv
Format catalog 1: csv ;csv, fits, votable or ascii
Input catalog 2: ./input/glimpse.csv
Format catalog 2: csv ;csv, fits, votable or ascii
Output: ./output/out.csv
Output format: csv ;csv, fits, votable or ascii
Log file: ./output/out.log
Stilts directory: ./libs
working directory: ./tmp ;temporary directory, removed when completed

[Match Criteria]
algorithm: sky ;sky, exact value, row-by-row

[Sky parameters]
area shape: ellipse ;ellipse or rectangle
size type: fixed ;parametric or fixed
matching area first dimension: 5 ;arcsec for fixed type - column name/number for parametric type
matching area second dimension: 5 ;arcsec for fixed type - column name/number for parametric type
parametric factor: 1 ;multiplicative factor for dimension columns - required for parametric type
pa column/value: 0 ;degrees for fixed type - column name/number for parametric type
pa settings: clock, 0 ;orientation (clock, counter), shift (degrees) -empty or default = clock,0
Catalog 2 minimum partition cell size: 100 ;arcsec

[Catalog 1 Properties]
coordinate system: galactic ;galactic, icrs, fk4, fk5
coordinate units: deg ;degrees (or deg), radians (or rad), sexagesimal (or sex)
glon/ra column: L ;column number or name - required for sky algorithm
glat/dec column: B ;column number or name - required for sky algorithm
designation column: SOURCEID ;column number or name - -1 for none

[Catalog 2 Properties]
coordinate system: galactic ;galactic, icrs, fk4, fk5
coordinate units: deg ;degrees (or deg), radians (or rad), sexagesimal (or sex)
glon/ra column: l ;column number or name - required for sky algorithm
glat/dec column: b ;column number or name - required for sky algorithm
designation column: designation ;column number or name, -1 for none

[Threads Properties]
```

```
thread limit: 256 ;maximum number of simultaneous threads (it depends on your machine)

[Output Rows]
Match selection: all ;all or best
Join type: 1 and 2 ;1 and 2, 1 or 2, all from 1, all from 2, 1 not 2, 2 not 1, 1 xor 2
```