D3.11.4: Report on the Preparation for Full-Scale DAC Matches

WP3.11: Cross-Matching and Astrometry at LSST Depths

Project Acronym	LUSC-B
Project Title	UK Involvement in the Legacy Survey of Space and Time
Document Number	LUSC-B-37

Submission date	31/JAN/2023
Version	1.0
Status	Final
Author(s) inc. institutional affiliation	Tom J Wilson (University of Exeter) Tim Naylor (University of Exeter)
Reviewer(s)	Bob Mann (UEDIN), Raphael Shirley (SOTON)

Dissemination level
Public

Version History

Version	Date	Comments, Changes, Status	Authors, Contributors, Reviewers
0.1	30/01/23	Initial draft	TJW
0.2	31/01/23	Minor updates, document number, SRD numbers	TJW
0.3	31/01/23	Grammar check, language change etc.	TN & TJW
0.4	14/03/23	Added extra details, expanded sections, grammar fixes	TJW
1.0	27/04/23	Approved by LSST:UK Executive Committee	TJW & TN

Table of Contents

List of Figures

List of Tables

1. Executive Summary

The goal of WP3.11, Cross-Matching and Astrometry at LSST Depths, is to create a cross-match tool through which common detections can be identified between LSST objects and sources in numerous other (non-LSST) datasets. Alongside identifying matches, the generated products should be able to report on the probability of such a match (to identify unreliable assignments for users), and provide key derived secondary information such as is necessary or useful for the astronomer.

Until now each deliverable has extended this cross-match functionality. This began with a "simple" Bayesian cross-match and we then added to the algorithms to include those needed to overcome the systematic effects such as the effects of hidden, unresolved contaminant objects on the positions of sources, or the inclusion of photometric information in the discernment of "false positive" matches. By contrast this deliverable, D3.11.4, reports the efforts of WP3.11 to accomplish the goal of performing matches, rather than making a match-capable software.

This report summarises those efforts, a three-pronged work. First, the incorporation of previous testing which identified the need to increase the scope of parallelisation within the codebase to reduce total runtimes. Second, the inclusion of additional software to ensure that the matches are both precise and accurate in their reported match likelihoods. Finally the beginning of "full scale" matches, running all-sky tests and simulating the wider workflow with the DAC team in which catalogues are generated, matched, and the consolidated results ingested into the UK DAC for access by in the community the UK RSP.

Where relevant, we will include reference to the Science Requirement Document items for WP3.11, R11.X.

2. Introduction

WP3.11 is charged with both the understanding of LSST's astrometry and the knowledge of how to robustly match (R11.1, R11.5), and the creation of software (R11.3, R11.6-7) to match, its data releases to a wide range of photometric catalogues at complementary dynamic ranges, wavelengths, angular resolutions etc. The purpose of performing these value-added calculations is that the science generated from combining both datasets should be greater than the sum from just the individual surveys.

This charge is then broken down into two main areas. First, WP3.11 built upon work previously done [1, 2, 3] to understand the effects of crowding on LSST objects – in which high densities of sources (relative to the size of the PSF) cause overlapping objects. Hence fainter objects "hide" within the footprint of brighter objects, causing second-order changes to measured positions and brightnesses of the objects as compared with the case of an isolated, single source. This was chiefly the focus of D3.11.1, extending the mathematics used to describe these astrometric "tugs" to key areas appropriate to faint LSST sources. Second, WP3.11 had to build the software involved in performing these matches. This was the focus of D3.11.2 and D3.11.3, building slowly from a simple Bayesian match, with no complications to the astrometry of sources, to a larger-scale, full match service appropriate to crowded LSST fields, both including the effects of hidden contaminant objects on the positions of objects but also using the magnitudes of sources in both catalogues to reject false "interloper" matches[1].

Therefore, as we began work on D3.11.4, we had an almost-complete software, but little proven confidence in its smooth operation, outside of limited trial runs and unit tests within the software itself. We began focusing on the overall workflow of the system, deploying the software on CSD3. Testing then quickly revealed that LSST-scale matches were not currently achievable within reasonable timescales with single-node compute; hence we implemented MPI parallelisation within the codebase, and extended the software to handle the split-and-join technique for distributed processing. Finally, we continued to improve and extend the cross-match software, the chief development of which is a parallel piece of software, outside of the main "cross-match" framework but crucial for its ability to report precise and accurate match likelihoods, accounting for potential systematic biases in reported astrometric uncertainties in the input catalogues (R11.5).

Surrounding all of this work, we have also begun ramping up efforts to engage with the wider community, chiefly the TVS and SMWLV Science Communities as the recipients of our In-Kind Contribution to LSST and the astronomers most affected by LSST crowding. Here we solicited feedback on the wider workflow that both the development team and the future team members from the UK DAC who will run the software during LSST operations will run. Largely a case of "which catalogues would you like to see cross-matched against LSST?," this dialogue ensures that in the case of limited HPC facility time we are optimising the scientific yield of our efforts. As this is a slow-burning project that is still in its early days, we limit discussion to the "future works" section at the end of the report.

2.1. Glossary of Acronyms

LSST - Legacy Survey of Space and Time
RSP - Rubin Science Platform
PSF - Point Spread Function
CSD3 - The Cambridge Service for Data Driven Discovery

[1]These are objects serendipitously close to an opposing catalogue source which may incorrectly be assigned as the match, but for the brightnesses being "wrong" as compared with the ensemble of potential matches.

AUF - Astrometric Uncertainty Function
MPI - the Message Passing Interface
DAC - Data Access Centre
DEV - LSST:UK Development work package
TVS - Transient and Variable Star
SMWLV - Stars, Milky Way, and Local Volume
SNR - signal-to-noise ratio

3. Run Timescales and MPI Parallelisation

We had run preliminary cross-matches of small-scale catalogues during previous testing[4]. Taking a small region of sky, both approximately 1% in sky area and source numbers, and profiling an earlier but fully-functioning version of `macauff` we examined the scaling relations of each of the four main stages in the matching process (additional AUF calculations, source grouping or "island" determination, photometric likelihood calculations, and best-match determination). Ultimately we concluded that the typical scaling was linear with total number of objects (for a fixed area) for matching steps 2-4, with the first step approximately flat in runtime (being simulation based and thus independent of source density for the most part).

However, with our code already slower than other methods due to needing complex algorithms and non-analytic solutions because of the crowding component of the AUF, we realised the software as it was would become intractable when scaled from *Gaia* dataset size (~ 2 billion objects in the full sky) to LSST (~ 30 billion objects in the southern sky). Despite the reasonable linear scaling relation, the 15–30-fold increase in source counts in the era of LSST meant that absolute runtimes would fail to meet our required several week turn-around timescales for publishing cross-matches for a new LSST Data Release. Hence we could no longer rely on our previous computational toolkit (Python's `multiprocessing` and Fortran's `OpenMP`) to run things in a "reasonable" time. We therefore had to change our previous vision for the scale of computation necessary for the LSST cross-matches, and implement an MPI-based, multi-node distribution of parallel matches.

With the assistance of Dominic Sloan-Murphy for an efficient and effective implementation, our codebase now accepts the so-called chunk "core" and "halo" model, in which a larger region is broken down into small parts and each processed separately (see Figure 1 for an example schematic). Here the sky is divided into small regions, each of which exactly divide the sky into (roughly even) patches of sky (red squares, Figure 1), with slightly larger squares, centered on the same central point as the "core" patch, adding additional objects (black squares, Figure 1). With the objects in one chunk's halo being in the core of another chunk, this does require a final step to the matching process where these duplicates are filtered out, but the advantages of being able to use distributed, multi-node HPC facility time to gain significant (of order 100-200x) speedup vastly outweigh this extra complication.

In addition, in the new paradigm where an all-sky match is broken up *before* match instead of *during* a match – i.e., in the creation of small chunks each of which are cross-matched separately, instead of attempting to load small parts of one giant all-sky "chunk" within the matching steps – we no longer require the sidestepping of memory issues and can run a match directly in memory, significantly improving bottlenecks from disk I/O. To create these chunks we have chosen to use the LSST partitioning software, which not only avoids unnecessary software development effort on our end but also conveniently means our cross-match patches can be on the same mapping as the pre-partitioned LSST data ingested as part of DAC ingestion into the RSP, which offers significant speedup advantages if data are correctly loaded into node memory. Performing subsequent scaling tests on the efficiency of the parallelism, we found that CPU idle time was limited across nodes, with 80-90% theoretical efficiency in most combinations of node number and task-per-node, with MPI handling the division of labour of each chunk to a node (or fraction of a node), and `OpenMP` still handling the intra-match parallelisation.

With this, our code ought to be capable of matching LSST and ancillary datasets within the timescale of a few days, much improved over both our initial estimate and our set goal.

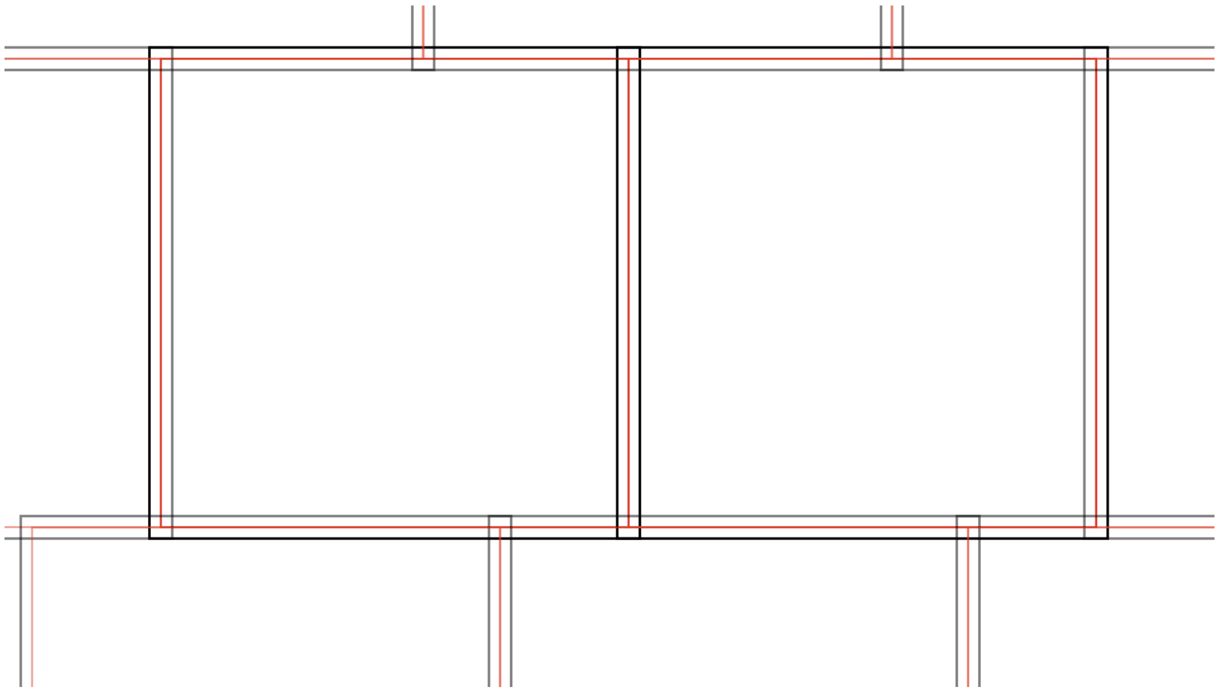


Figure 1: Schematic of MPI-based sky chunk decomposition. In the multi-node compute paradigm the sky is split up ahead of time, and each "chunk" runs a small part of the sky in parallel, each node free to use its own processors for smaller-scale parallelisation. Small overlaps are necessary to catch edge-case matches, using a "core" (red squares) and "halo" (black/grey squares, slightly larger than the red squares) model; this necessitates the reconciliation of duplicated objects (in one chunk's core and another's halo) to avoid conflicting reports of match vs non-match, match probability etc.

4. `macauff`

4.1. Code Extensions

Since D3.11.3 there have been numerous extensions to `macauff` (see Appendix A for details of its structure and format). In addition to various bugfixes and optimisations "under the hood," there have been several key advancements of the functionality of the software, described in more detail below. All of these improvements and extensions are available at https://github.com/Onoddil/macauff.

4.1.1. Galaxy Counts Model

A flexible model for galaxy counts has been added to the code, to allow for the simulation of extragalactic unresolved perturbers outside of the Galactic plane (R11.2-3).

It is a meta-model, effectively collating previous results and generalising the parameterisations. We approximate the spatial density of galaxies as a function of (absolute) magnitude with a simple Schechter function

$$\phi(M, z) = 0.4\ln(10)\phi^* \left[10^{-0.4(M-M^*)}\right]^{\alpha+1} \times \exp\left(-10^{-0.4(M-M^*)}\right), \tag{1}$$

with M^* the "characteristic" absolute magnitude, α the faint-end power-law slope, and ϕ^* a normalizing parameter, with each a function of redshift via further parameters P and Q. Literature data these five variables are then fit as a function of wavelength for both star-forming

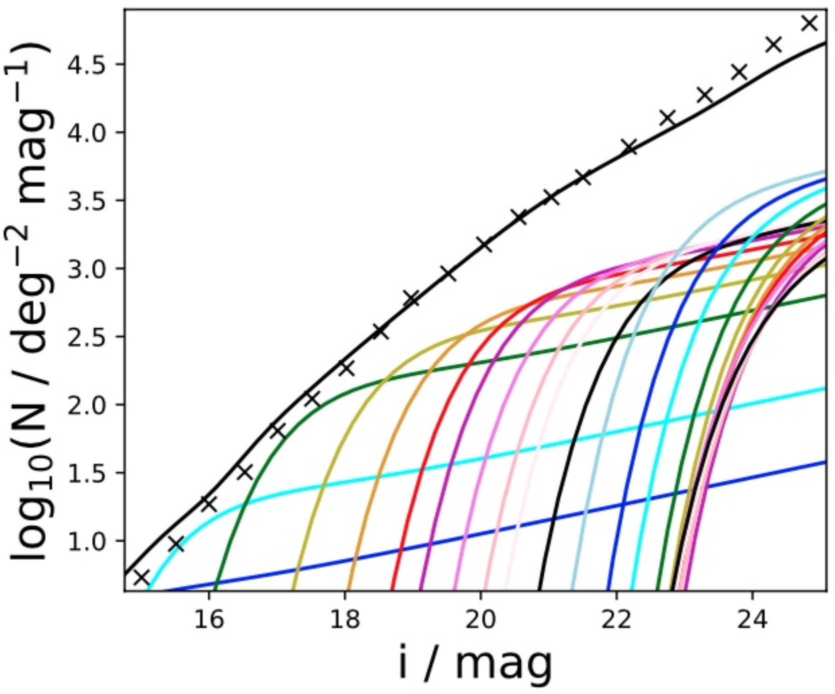


Figure 2: An example use of the flexible galaxy counts model used within `macauff`. Comparison between the model (solid black line) and example literature source density (crosses), with different redshift bin contributions shown in the solid colour lines below the data.

and quiescent ("blue" and "red") galaxies, such that differential galaxy counts can be derived in any chosen bandpass. An example, converted from spatial density to on-sky density, is shown in Figure 2, plotted against an observed distribution of galaxy counts, highlighting the components from different redshift bins in the lower lines of varying colours. The results were published[5] and the code implemented, adding to existing code that calculates Galactic source counts. This ensures that the source density of potential unresolved contaminant objects is as precise and accurate as possible within the codebase, making the perturbation component of the AUF as robust as possible at all sky coordinates.

4.1.2. Background-Dominated PSF Photometry Perturbations

The new methodology for calculating the effects of blended contaminant sources in PSF photometry for background-dominated detections was implemented. This follows the theoretical method previously laid out in D3.11.1. A new Class was added to derive its parameterisation and `perturbation_auf.py` extended to include its determination and weighting (R11.1, R11.3, R11.5).

Up to now, we have only been using what we refer to as a "flux-weighted" perturbation algorithm, in which both the bright central source and all hidden contaminants are treated as point sources. The position of the composite object is then the flux-weighted average of all point sources within the bright object's PSF. In PSF photometry, the astrometric effect of the contaminant sources should be weighted by the value of the PSF at *all* positions around *each* object, both contaminating and central, and hence the point-source approximation does not apply. Describing this effect is quite difficult analytically across the entire dynamic range of a catalogue, but at lower SNR the problem becomes tractable. This is because when the source is sufficiently faint the dominant source of noise in the astronomical image from which the photometric catalogues is built is from the sky and not the object, and hence all pixels in the image should have roughly

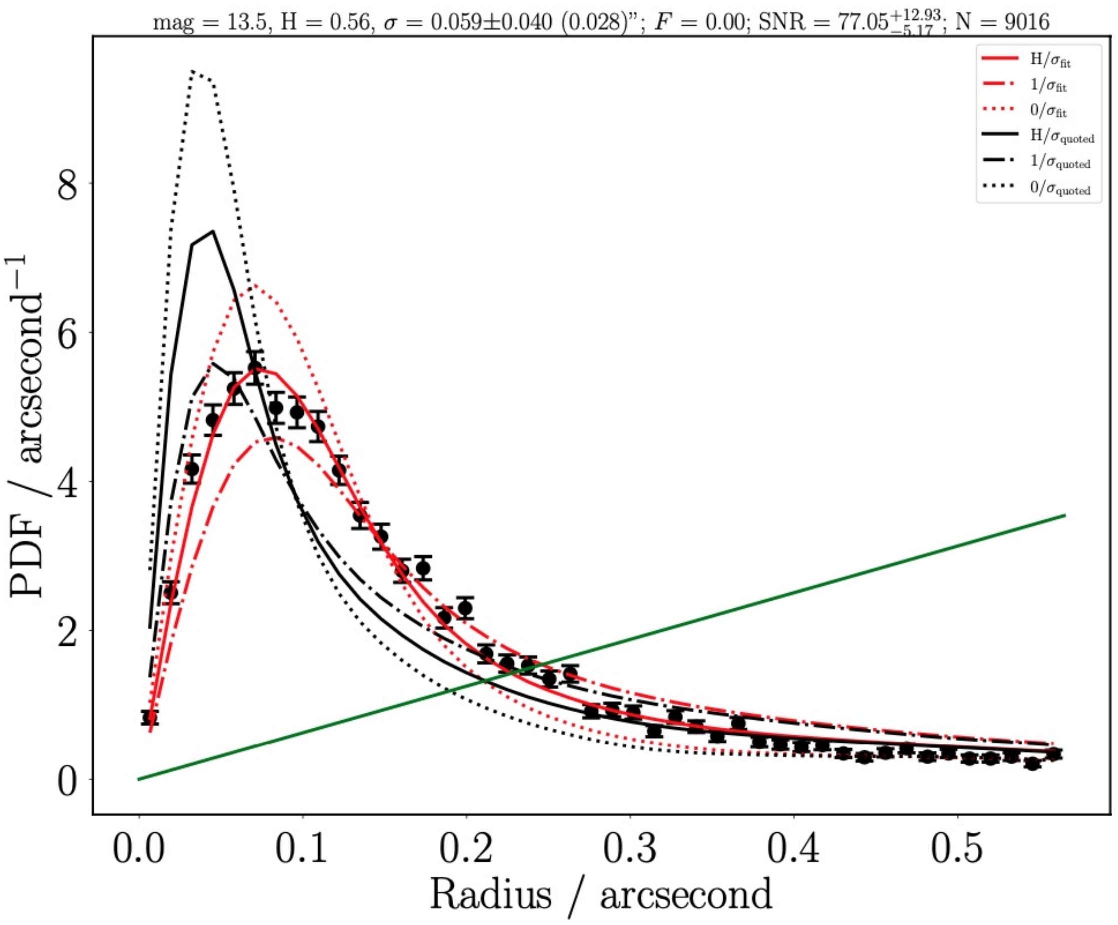


Figure 3: Implementation of new background-dominated PSF photometry algorithm and its weighting with a flux-weighted scheme. Black errorbars show an ensemble of separations between *Gaia* and CatWISE data, all for a small sky region and astrometric precision, effectively representing an empirical AUF. Six AUFs are shown overlaid. Dash-dot lines show the original flux-weighted algorithm AUF and dotted lines show the new PSF-fit algorithm AUF, with solid lines showing the weighting of the two valid for the magnitude of these sources. The three versions of the perturbation component of the AUF are combined with two respective astrometric centroid precisions (quoted catalogue value, black lines, and best-fit value determined through `AstrometricCorrections` in red). We desire the centroid precision that combines with the H-weighted perturbation AUF components to fit the data the best, and the solid red line is the only one of the six potential AUFs that fits the data, and hence represents the data-driven AUF. The green line shows the probability density for false matches, used to fit for false positive matches in cases of incorrect counterpart assignment in the creation of the cross-matched ensemble.

the same noise. This low SNR regime is frequently referred to as the "background-dominated" regime for this reason.

The "flux-weighted" model is primarily valid in the case of aperture photometry being used to determine the positions and brightnesses of objects in a catalogue, but we previously found it also applies in the case of bright, high SNR objects that were fit with a PSF photometry algorithm. Hence in cases where photometric catalogues were constructed using PSF photometry we have two valid algorithms, one that is valid at high SNRs and one that is valid at low SNRs. At intermediate SNRs, there will be some hand-off between these two algorithms.

Building off previous work[6], the PSF photometry method for determining the centre-of-light shift from an unresolved contaminant perturbing the measured position of a brighter source is based on fitting a composite two-PSF source with a single PSF model. Through a simple least-squares minimisation routine, in the limit – appropriate for most faint LSST sources, and assumed by the Rubin Pipeline[7] – that noise in the image is constant, we are essentially solving

$$\log \mathcal{L} = -\frac{1}{2} \times L \int_{-\infty}^{\infty} \left[\phi(\mathbf{r}) + f\phi(\mathbf{r}-\mathbf{d}) - (1+\Delta f)\phi(\mathbf{r}-\mathbf{\Delta d})\right]^2 \mathrm{d}^2 r. \qquad (2)$$

ϕ is the PSF model, and we wish to know what Δd, the perturbation due to the contaminating source, is (as well, potentially, as Δf, the *brightening* of the reported flux due to the hidden source).

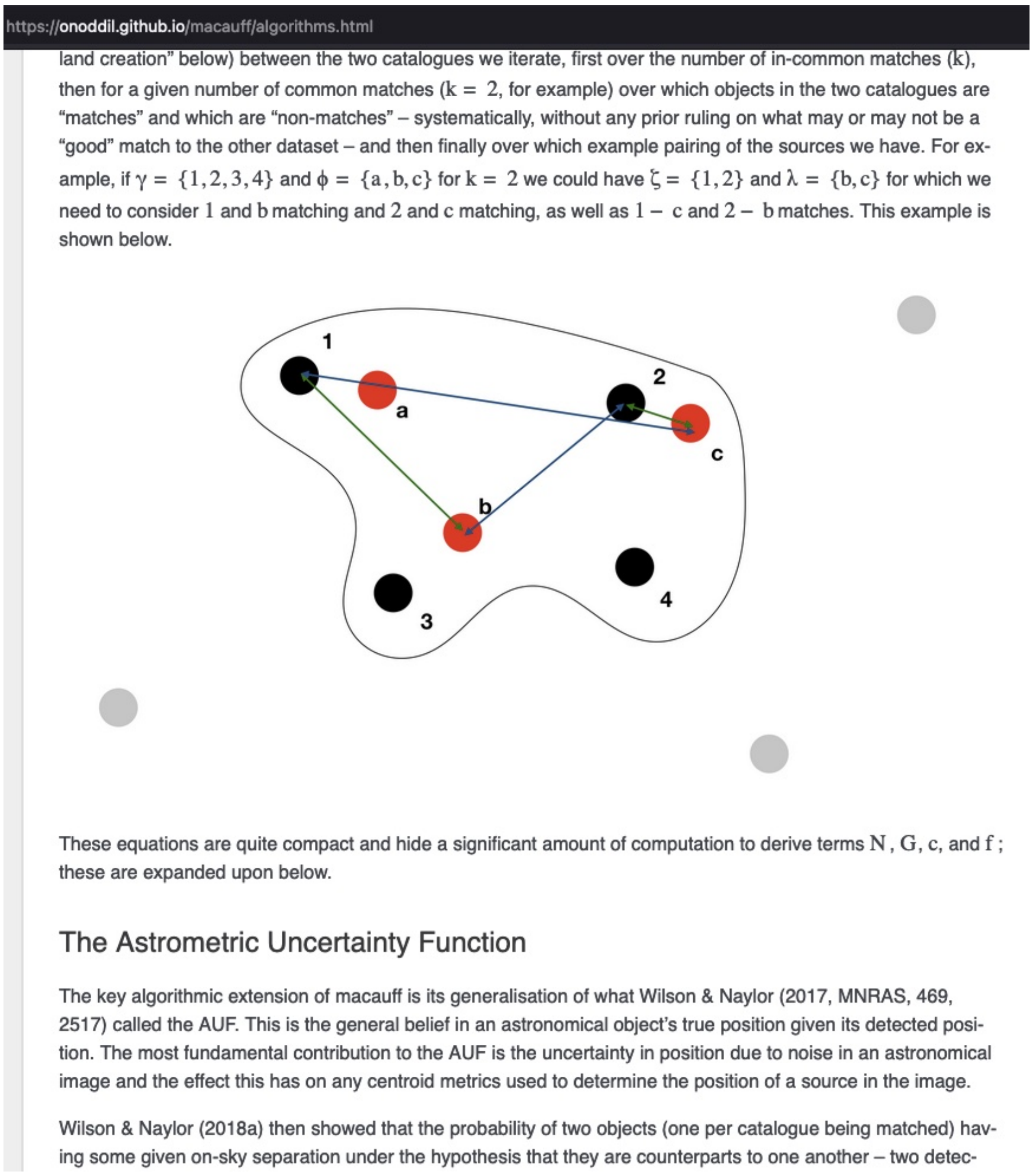
Expanding and solving the equation allows for an approximation in the limit of a faint perturber, as previously calculated, but here we extend the model to approximate the effect of objects of tens of percent the relative flux of the bright central source. Making the assumption that the perturbation from multiple objects can be split into a vector sum of their individual contributions, we modelled the astrometric perturbation and photometric brightening of the two-body composite source as a function of the contaminant's relative position and flux. We then eventually reduced the problem to a series of parametrised spline fits that feed into a skew-normal distribution.

Now armed with two methods for determining the perturbation to the position of an object due to hidden contaminant sources, we can better describe the AUF at all SNRs by a combination of the two methods. An example of this, within `AstrometricCorrections` (discussed in Section 4.2), is shown in Figure 3. Here H is the weighting between our original flux-weighted (naive assumption that all objects are point sources, or have infinite SNR) and new, PSF-fitting algorithms, with the former shown in dash-dot lines and the latter in dotted lines (for two different centroid precisions). The weighting between the two is used in intermediate SNR regimes, with the flux-weighted algorithm valid at effectively infinite SNR and the background-dominated case valid when the SNR tends to zero. When combined with the centroid[2] uncertainty of the objects (with all sources in Figure 3 having roughly the same positional precision) the total AUF should overlay the distribution of separations between our *WISE* and *Gaia* data – equivalent to looking at the deviations from "true" position for the *WISE* data.

4.1.3. Documentation

Documentation has been significantly expanded, now detailing all input parameters but also describing the mathematical algorithms used within the match process and including more information on the "before" and "after" stages of matching (R11.3).

[2]Here the "centroid" uncertainties are the precisions with which the given position of a source was determined on the detector.


https://onoddil.github.io/macauff/algorithms.html

land creation" below) between the two catalogues we iterate, first over the number of in-common matches (k), then for a given number of common matches (k = 2, for example) over which objects in the two catalogues are "matches" and which are "non-matches" – systematically, without any prior ruling on what may or may not be a "good" match to the other dataset – and then finally over which example pairing of the sources we have. For example, if $\gamma = \{1,2,3,4\}$ and $\phi = \{a,b,c\}$ for $k = 2$ we could have $\zeta = \{1,2\}$ and $\lambda = \{b,c\}$ for which we need to consider 1 and b matching and 2 and c matching, as well as 1 – c and 2 – b matches. This example is shown below.

These equations are quite compact and hide a significant amount of computation to derive terms N, G, c, and f; these are expanded upon below.

The Astrometric Uncertainty Function

The key algorithmic extension of macauff is its generalisation of what Wilson & Naylor (2017, MNRAS, 469, 2517) called the AUF. This is the general belief in an astronomical object's true position given its detected position. The most fundamental contribution to the AUF is the uncertainty in position due to noise in an astronomical image and the effect this has on any centroid metrics used to determine the position of a source in the image.

Wilson & Naylor (2018a) then showed that the probability of two objects (one per catalogue being matched) having some given on-sky separation under the hypothesis that they are counterparts to one another – two detec-

Figure 4: Example of new documentation available within `macauff` at https://onoddil.github.io/macauff/.

Expanding the previously written documentation – both improving what was written, and adding to existing documentation to make it clearer to understand – we

1. expanded the descriptions of installation of the software, both to make them easier to follow but also to keep up with upstream changes from dependencies and testing environments,
2. updated the "docstrings" of functions where APIs changed or descriptions were incomplete or misleading,
3. improved the quick-start guide for getting started with the code,
4. updated the descriptions of the parameters expected or optionally includeable within the configuration files `macauff` ingests.

However, we also added two new sections to the documentation to aid in understanding the matching process in more depth.

First, we added a page describing the mathematics behind the software, detailing the Bayesian framework we have constructed and the specifics of each step in the matching process. This page reflects the current approach to the complexities of cross-matching in crowded fields and will be maintained going forward if future developments change our algorithms. It describes past work[1, 2, 3], current developments[5, 7], and future works (such as the planned publication of the perturbation AUF component extensions described in Section 4.1.2). An example part of this page of the documentation can be seen in Figure 4.

Second, we included a "big picture" section, providing users with information on the stages that surround the actual cross-match run itself, and the necessary steps that need to be taken to provide catalogues in the correct format and combine datasets into a band-merged dataset once the matching process is complete. Here we describe

1. the currently expected format of `npy` binary files for the catalogues, split into astrometry and photometry in separate files, and how to use `parse_catalogue.py` to create them,
2. the core-halo chunk model, and the additional expected binary file which holds the flags for objects outside of a core region in each chunk patch,
3. `AstrometricCorrections`, the new functionality within `macauff` to determine centroid precisions from large-scale counterpart separation distributions between a particular catalogue and another with well-understood astrometry, for which more details are given in this report in Section 4.2,
4. an overview of `FitPSFPerturbations`, the code for determining the background-dominated, PSF photometry perturbation AUF component parameterisation, as described in Section 4.1.2,
5. the removal of potential duplicate sources as a result of the use of the core-halo model (i.e., the removal of halo objects that are in one chunk's halo and another's core),
6. and the final stage of the workflow of creating cross-matches between two photometric catalogues: the merging of their counterparts into a single dataset, and the reporting of non-matches, and any associated information derived during the cross-match process.

4.1.4. Unknown Proper Motions

A method for deriving, in a statistical sense, the likely range of possible proper motions a source may have, given its sky coordinates and brightness, was developed (R11.1-3, R11.5). Until LSST DR4-5 this will be key to avoiding reliance on using *Gaia* as a go-between for robust matches – especially since it will only overlap the brightest 10% of the LSST dataset or so. Even once LSST is able to report proper motions, this functionality will be key for objects below the single-visit detection limit, ensuring we do not miss matches due to this additional term of object counterpart separation. Thus, to that end, we wished to be able to include the *potential* proper motions for sources without measured ones.

The model we created[7] is relatively simple, but easy to compute and does not rely on any specifics, just requiring sky position and star distance, available through Galaxy model simulations. We extracted circular velocity distributions, and asymmetric drift velocity and velocity dispersions (purely theoretical orbit and the slight reduction in "pure" circular speed due to random interactions) from the literature. Combining models for the thin and thick disc and Galactic halo – with potential eventual need for a model for velocities within the Bulge – and converting from velocity to on-sky speed for all objects of a particular sky position and brightness we can create distributions of potential proper motions for sources without measured motions.

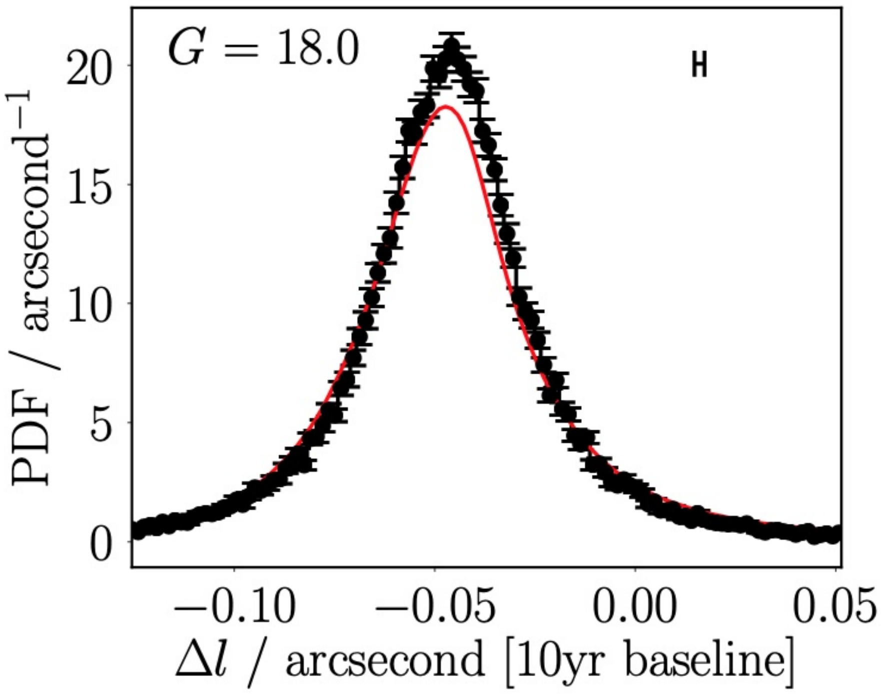


Figure 5: Potential proper motions of an object with particular coordinates and brightness but no measured sky motion. Comparison data taken from *Gaia*, of an ensemble of many sources' proper motions in the chosen sky region and $G \approx 18$, are given in the black errorbars, with our model in the solid red line. Units have been converted from proper motion in e.g. mas yr^{-1} to a separation drift across a decade, in arseconds.

These distributions fold in all manner of astrophysics – such as dwarf and giant stars at the same brightness, unknown velocity due to orbital interactions, or the Sun's motion around the Galaxy. An example of the model compared with *Gaia* data is given in Figure 5, with *Gaia* proper motions (converted to decade-long separation drift) in the black data compared with our model in red, for a small slice in sky position and brightness around $G = 18$.

4.2. Astrometry Bias Derivation

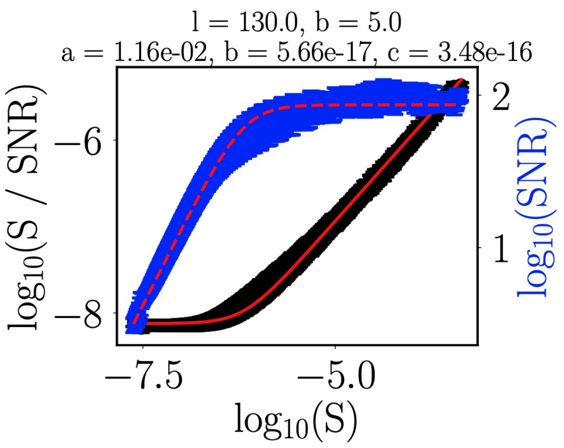
The software that we have created within WP3.11 is capable of performing reliable cross-matches – but only if the data provided to it are reliable. If the astrometric uncertainties provided in the catalogues[3] are not accurate, then the Bayesian matching algorithms we use will also not be accurate, and matches and corresponding probabilities no longer trustworthy. To ensure that this is not a factor in our matches, we created a parallel script within the `macauff` codebase to determine ensemble astrometric precisions from the data themselves, comparing and verifying the large-scale "correctness" of the catalogue positions and precisions through matches to high spatial resolution and high astrometric precision data.

In the absence of crowding, we could achieve the calibration or determination of astrometric uncertainties in the following way. If we had a large number of matches between our chosen catalogue and some other catalogue with significantly higher astrometric precision we could use the differences in position, or separation between sources in opposing catalogues, to determine our AUF. Practically, we would expect this to be dependent on the magnitude of the sources, and hence would likely fit sources in small magnitude ranges to derive the AUF for each magnitude

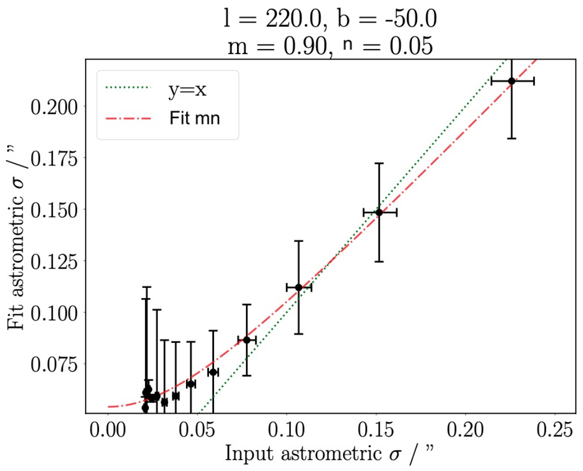
[3]Again, we refer to these as "centroid" uncertainties, which are the precisions with which the given position of a source was determined on the detector.

slice. Without being subject to the effects of crowding, these separations should be distributed with Gaussian shape, and hence we would fit the distribution of separations for a common Gaussian centroid uncertainty sigma, the ensemble astrometric uncertainty. Of course, in reality catalogues with high relative densities of sources such as LSST or *WISE* are subject to crowding and the effects of unresolved blended contaminants, so there is a second component of the AUF that we must take into account. We must first estimate this component, as accuractely as possible, and effectively combine it with the Gaussian centroid uncertainty distribution to model the AUF that we actually see. In this section we describe how we derive astrometric uncertainties as a function of "catalogue" astrometric uncertainty, as well as object magnitude. We show how we combine our two crowding models (as described in section 4.1.2) and the process by which the centroid uncertainties are fit for. Finally, we detail the relation between "input" and "output" uncertainties that describe the corrections to the astrometric precisions that may be necessary for accurate and reliable probabilistic cross-matches.

The code used to derive these corrective relationships in the astrometric uncertainties for a particular catalogue, `AstrometricCorrections`, is run on small sky regions – perhaps tens of square degrees. The code is agnostic to the size of any individual region – capable of being passed functions that generate small cutouts of larger catalogues or loading pre-made data files. However, scientifically we want the region to be as small as possible to avoid introducing biases in catalogue reduction (either telescope based or e.g. differential Galactic longitude-based crowding effects) but as large as possible to ensure good number statistics for brighter sources. For each sightline passed to `AstrometricCorrections`, three major steps are performed.



(a) SNR-magnitude scaling relation for a particular sky region from `AstrometricCorrections`. Using the magnitude and magnitude error of all sources in a small sky region we derive a flux (S) and SNR, shown in small log-flux bins (black errorbars, hiding extra structure that we currently ignore for this simple model). Our simple scaling relation between S and S/SNR (or, equivalently, one over the noise) is overlaid in the solid red line. We also show the equivalent SNR as a function of flux in blue errorbars, with the same model fit transformed to pure SNR in the dashed red line.



(b) Relation between quoted and derived astrometric uncertainties. Once a particular sightline has had data-driven centroid uncertainties derived from the distribution of separations to higher precision data, the relation is fit. Derived uncertainties are shown in black error bars, plotted against each data subset's average catalogue precision. This model (red dot-dash line) is a simple quadrature sum of a systematic (n) and quoted uncertainty scaled by a factor m. Green dotted line shows the relation if quoted astrometry described the distributions of separations from "truth" for the objects in the catalogue ($m = 1$, $n = 0$).

Figure 6: Figures showing aspects of `AstrometricCorrections`, along with Figure 3.

To calculate the hand-off between the two perturbation components of the AUF, valid in differing SNR limits, we actually use the photometric measurements and uncertainties, since we find that the photometric uncertainties tend to be more reliable than the astrometric ones. We first calculate SNR-magnitude relations which are used in the weighting between the photon-noise dominated, flux-weighted AUF algorithm and the background-dominated PSF photometry algorithm developed in D3.11.1 and described in Section 4.1.2. The weighting between the two algorithm AUFs used within `macauff` is determined with a quadratic weighting between the "plateau" SNR of the catalogue (where systematic effects such as PSF model uncertainty outweigh photon noise uncertainty and flux-weighting is applicable) and zero SNR (where the background-dominated algorithm is valid),

$$H = 1 - \sqrt{1 - \min(1,\ (a \times \mathrm{SNR})^2)}, \tag{3}$$

with fits to a simple SNR model calculated from the data,

$$\mathrm{SNR} = \frac{S}{\sqrt{cS + b + (aS)^2}}, \tag{4}$$

and hence

$$\frac{S}{\mathrm{SNR}} = \sqrt{cS + b + (aS)^2}, \tag{5}$$

where S is the flux of the source (and $\log_{10}(S)$ effectively an instrumental magnitude), a sets the "systematic" noise (linear with photon noise), c the relative level of photon noise in the images, and b the source-independent "background" noise level; see Figure 6a. In this and subsequent figures, we show *WISE* as the to-be-fit catalogue.

Next, in small magnitude slices in the to-be-fit catalogue, total AUFs are fit to the ensemble distribution of separations between catalogue objects and "truth" positions and these are used to fit for the centroid Gaussian uncertainty. The "truth" positions come from a catalogue with much higher astrometric precision than our catalogue for which we wish to derive astrometric corrections, to avoid confusing the contributions from both objects to position uncertainty in the separation between catalogue-catalogue matches. However, due to the second effect from crowding causing unresolved contaminants and bright-source positional shifts, we also require the second catalogue be of much higher *spatial* resolution, to remove the effect of crowding on the positions of these "truth" objects. Good choices for "ground truth" catalogue are therefore datasets like *Gaia* or the *Hubble* Source Catalog, with the examples detailed here using *Gaia*. As shown in Figure 3 above, this involves weighting each algorithm's perturbation AUF component by H, calculated from the SNR-magnitude relation, and fitting for the "centroid" precision. This requires the H-weighted perturbation component of the AUF to be convolved with the "centroid" component of the AUF, modelled as a Gaussian function with to-be-determined width σ_{fit}. The fitting routine also determines the fraction of false matches F, if any, to avoid false positives biasing the results. In Figure 3 the various combinations of our two perturbation AUF component algorithms and quoted and best-fit astrometric precisions are shown in red and black dotted or dash-dot lines, along with the H-weighted AUFs, with original (black solid lines) and "best-fit" (red solid line) uncertainty, with only the best-fit, H-weighted AUF giving good agreement to the data.

Finally, relations between derived and quoted uncertainties are fit (see Figure 6b). Here we use a relatively simple but physically motivated model. Best-fit uncertainties are fit as a sum-in-quadrature of a missing systematic uncertainty n – setting a minimum astrometric uncertainty – and scaled quoted uncertainties, padded by a factor m (for cases where e.g. missing systematics are corrected for in noisier data),

$$\sigma_{\mathrm{fit}} = \sqrt{(m \times \sigma_{\mathrm{quoted}})^2 + n^2}. \tag{6}$$

The code can be run separately from `CrossMatch` within `macauff`, or it can be called on a per-chunk basis within the MPI-based parallel framework for larger LSST-scale matches. If run separately, CrossMatch can use the grid of derived scaling relations to assign nearest-neighbour values for m and n to any matches performed using that catalogue. Alternatively, if run within a chunk-based parallel run, each chunk can effectively have its own scaling relations determined from the data in situ.

This new software is key for ensuring that ancillary datasets, such as CatWISE, have reliable astrometric precisions that don't bias our probabilistic matching process (R11.1, R11.3, R11.5), but it will also be key to our future Phase C work with LSST (see below).

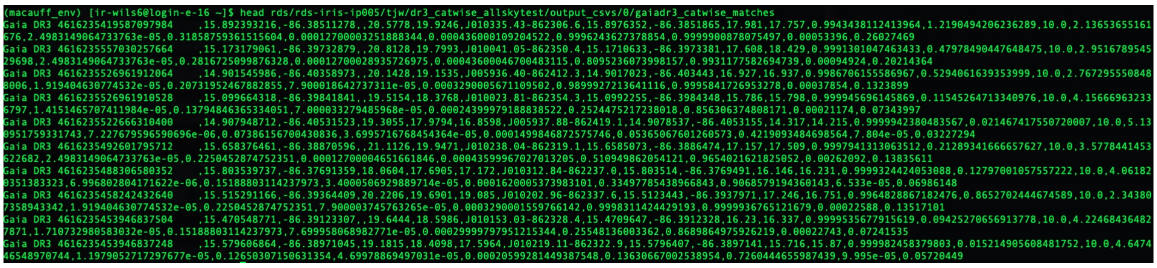
5. DAC-DEV Workflow and All-Sky Matches

The final, and most crucial, element of work undertaken within the remit of D3.11.4 is that of the actual running of the software. Previous testing, such as the "scaling relation" work[4] or the tests performed in D3.11.3, were relatively limited in scope; either they were small parts of the sky (in the case of D3.11.3) or reduced in scope to test specific parts of the code (in the case of the scaling relations). We therefore required, before "signing off" on the software, proof of its running on large scales, such as those we will see in production. This then tests our role in what we refer to as the "DAC-DEV Interface"[8], the "Cross-Match Processing" – the actual running of the algorithm and the determination of counterparts – and Interfaces 2 (Scheduling cross-matches), 4 (Software configuration), and 5 (Deployment of software). These aspects sit between the DAC's roles in the generating the reduced format "skinny tables" from catalogues like full LSST Data Releases and associated ancillary datasets, in this case *Gaia* and *WISE*, for use by macauff (Interface 3) and ingestion of final outputs (see Figure 7 for an example) back into the DAC (Interface 6). Combined with the eventual public-facing access of the data from the RSP (Interfaces 7 and 8), this test is designed to exercise all parts of the DAC-DEV workflow.

We have begun ramping up our interactions with the DAC team and performed an all-sky *Gaia*-CatWISE match (R11.6-7). As an all-sky test, we required the use of the partioning software to pre-generate sufficiently small patches of sky to run on a single node within a reasonable time; setting 25 square degree chunks resulted in 1558 separate cross-matches to perform and recombine. Individual runtimes for each chunk vary considerably due to the number of objects in each chunk, with *Gaia* varying from at smallest 64,000 sources up to 30,000,000 in the largest and *WISE* varying less (likely due to its inability to detect significant numbers of stars in crowded Galactic regions but also being more sensitive to extragalactic objects), from 680,000 to 4,000,000 sources. Due to a rate-limiting factor of an external API call to simulated data necessary for our cross-matching algorithms a total runtime is not verifiable at this stage, but individual chunks – working from largest to smallest to load balance nodes – ranged from longer, higher source density runs of an hour down to a more typical 20 minute runtime once this API call was removed.

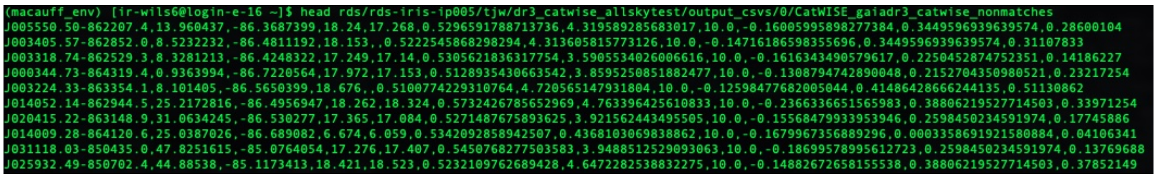
This test proved that the final code will run outside of those more limited cases we tested on previously. At the same time, the runs did reveal the *science* complexity of our cross-match algorithms, in which it is necessary to build a reasonable proxy of the brightness distribution of sources for any arbitrary sky coordinates in any filter. The tests showed us a weakness with our astronomical model used within the code in extreme cases such as within the very Galactic centre, where sources are systematically bright (due to reduced completeness limits from reddening or crowding) but our Galactic models push systematically faint. In such cases the Galactic extinction (the level to which objects are dimmed by intervening dust along the line of sight) was poorly chosen, incorrectly making all of our simulated objects much fainter than the real data. A more comprehensive model of differential line-of-sight dimming within each small region of sky brought our simulated source counts into agreement with observations once more, and avoided the issue of too-low number counts in these key regions of the sky.

Overall, however, the test was a success – if nothing else, the failure in the Galactic centre showed that the code written to gracefully handle individual chunk failures without impacting the runtime of any other chunk worked flawlessly! As we ramp up further in our testing with the DAC team, and begin scientific verification of the match results (R11.6-7), we will continue to investigate this potential issue, as it will impact LSST and the large fraction of Galactic plane science astronomers are planning with the Bulge.



```
(macauff_env) [ir-wils6@login-e-16 ~]$ head rds/rds-iris-ip005/tjw/dr3_catwise_allskytest/output_csvs/0/gaiadr3_catwise_matches
Gaia DR3 4616235419587097984    ,15.892393216,-86.38511278,,20.5778,19.9246,J010335.43-862306.6,15.8976352,-86.3851865,17.981,17.757,0.9943438112413964,1.2190494206236289,10.0,2.13653655161
676,2.4983149064733763e-05,0.31858759361515604,0.00012700003251888344,0.000436000109204522,0.9996243627378854,0.9999900878075497,0.00053396,0.26027469
Gaia DR3 4616235557030257664    ,15.173179061,-86.39732879,,20.8128,19.7993,J010041.05-862350.4,15.1710633,-86.3973381,17.608,18.429,0.9991301047463433,0.47978490447648475,10.0,2.9516789545
29698,2.4983149064733763e-05,0.2816725099876328,0.00012700028935726975,0.00043600046700483115,0.8095236073998157,0.9931177582694739,0.00094924,0.20214364
Gaia DR3 4616235526961912064    ,14.901545986,-86.40358973,,20.1428,19.1535,J005936.40-862412.3,14.9017023,-86.403443,16.927,16.937,0.9986706155586967,0.5294061639353999,10.0,2.767295550848
8006,1.919404630774532e-05,0.20731952467882855,7.900018642737311e-05,0.0003290005671109502,0.9899927213641116,0.9995841726953278,0.00037854,0.1323899
Gaia DR3 4616235526961910528    ,15.099664318,-86.39841841,,19.5154,18.3768,J010023.81-862354.3,15.0992255,-86.3984348,15.786,15.798,0.999945696145869,0.11545264713340976,10.0,4.15666963233
6797,1.4151465707411984e-05,0.13794846365334051,7.000033279485968e-05,0.00024399979188838522,0.2524475217238018,0.856306374808171,0.00021174,0.07343997
Gaia DR3 4616235522666310400    ,14.907948712,-86.40531523,19.3055,17.9794,16.8598,J005937.88-862419.1,14.9078537,-86.4053155,14.317,14.215,0.9999942380483567,0.021467417550720007,10.0,5.13
0951759331743,7.227679596590696e-06,0.07386156700430836,3.6995716768454364e-05,0.0001499846872575746,0.05365067601260573,0.4219093484698564,7.804e-05,0.03227294
Gaia DR3 4616235492601795712    ,15.658376461,-86.38870596,,21.1126,19.9471,J010238.04-862319.1,15.6585073,-86.3886474,17.157,17.509,0.9997941313063512,0.21289341666657627,10.0,3.5778441453
622682,2.4983149064733763e-05,0.2250452874752351,0.00012700004651661846,0.00043599967027013205,0.510949862054121,0.9654021621825052,0.00262092,0.13835611
Gaia DR3 4616235488306580352    ,15.803539737,-86.37691359,18.0604,17.6905,17.172,J010312.84-862237.0,15.803514,-86.3769491,16.146,16.231,0.9999324424053088,0.12797001057557222,10.0,4.06182
0351383323,6.996802804171622e-06,0.15188803114237973,3.4000506929889714e-05,0.0001620005373983101,0.33497785438966843,0.9068579194360143,6.533e-05,0.06986148
Gaia DR3 4616235458242432640    ,15.515291166,-86.39364409,20.2206,19.6901,19.085,J010202.96-862337.6,15.5123443,-86.3937971,17.246,16.751,0.9964828867182476,0.8652702444674589,10.0,2.34380
7358943342,1.919404630774532e-05,0.2250452874752351,7.900003745763265e-05,0.0003290001559766142,0.9998311424429193,0.9999936765121679,0.00022588,0.13517101
Gaia DR3 4616235453946837504    ,15.470548771,-86.39123307,,19.6444,18.5986,J010153.03-862328.4,15.4709647,-86.3912328,16.23,16.337,0.9999535677915619,0.09425270656913778,10.0,4.22468436482
7871,1.710732980583032e-05,0.15188803114237973,7.699958068982771e-05,0.00029999797951215344,0.25548136003362,0.8689864975926219,0.00022743,0.07241535
Gaia DR3 4616235453946837248    ,15.579606864,-86.38971045,19.1815,18.4098,17.5964,J010219.11-862322.9,15.5796407,-86.3897141,15.716,15.87,0.999982458379803,0.015214905608481752,10.0,4.6474
46548970744,1.1979052717297677e-05,0.12650307150631354,4.69978869497031e-05,0.00020599281449387548,0.13630667002538954,0.7260444655987439,9.995e-05,0.05720449
```

(a) An example of a match table between *Gaia* and *WISE*. The tables are headerless, with metadata handled separately, but the column names are G_ID, G_RA, G_DEC, BP, G, RP, W_ID, W_RA, W_DEC, W1, W2, MATCH_P, SEPARATION, ETA, XI, G_AVG_CONT, W_AVG_CONT, G_CONT_F1, G_CONT_F10, W_CONT_F1, W_CONT_F10, G_ASTRO_SIG_USED, and W_ASTRO_SIG_USED, which are explained in the text.



```
(macauff_env) [ir-wils6@login-e-16 ~]$ head rds/rds-iris-ip005/tjw/dr3_catwise_allskytest/output_csvs/0/CatWISE_gaiadr3_catwise_nonmatches
J005550.50-862207.4,13.960437,-86.3687399,18.24,17.268,0.5296591788713736,4.319589285683017,10.0,-0.16005995898277384,0.3449596939639574,0.28600104
J003405.57-862852.0,8.5232232,-86.4811192,18.153,,0.5222545868298294,4.313605815773126,10.0,-0.14716186598355696,0.3449596939639574,0.31107833
J003318.74-862529.3,8.3281213,-86.4248322,17.249,17.14,0.5305621836317754,3.5905534026006616,10.0,-0.1616343490579617,0.2250452874752351,0.14186227
J000344.73-864319.4,0.9363994,-86.7220564,17.972,17.153,0.5128935430663542,3.8595250851882477,10.0,-0.1308794742890048,0.2152704350980521,0.23217254
J003224.33-863354.1,8.101405,-86.5650399,18.676,,0.5100774229310764,4.720565147931804,10.0,-0.12598477682005044,0.41486428666244135,0.51130862
J014052.14-862944.5,25.2172816,-86.4956947,18.262,18.324,0.5732426785652969,4.763396425610833,10.0,-0.2366336651565983,0.38806219527714503,0.33971254
J020415.22-863148.9,31.0634245,-86.530277,17.365,17.084,0.5271487675893625,3.921562443495505,10.0,-0.15568479933953946,0.2598450234591974,0.17745886
J014009.28-864120.6,25.0387026,-86.689082,6.674,6.059,0.5342092858942507,0.4368103069838862,10.0,-0.1679967356889296,0.0003358691921580884,0.04106341
J031118.03-850435.0,47.8251615,-85.0764054,17.276,17.407,0.5450768277503583,3.9488512529093063,10.0,-0.18699578995612723,0.2598450234591974,0.13769688
J025932.49-850702.4,44.88538,-85.1173413,18.421,18.523,0.5232109762689428,4.6472282538832275,10.0,-0.14882672658155538,0.38806219527714503,0.37852149
```

(b) An example of a non-match table between *Gaia* and *WISE*, in this case the *WISE* non-matches. The tables are headerless, with metadata handled separately, but the column names are W_ID, W_RA, W_DEC, W1, W2, MATCH_P, NNM_SEPARATION, NNM_ETA, NNM_XI, W_AVG_CONT, and W_ASTRO_SIG_USED, which are explained in the text.

Figure 7: A few lines of outputs from match and non-match tables as produced by macauff.

An example of a successful output of these tables, ready for ingestion into the UK DAC and processing for serving through the RSP once combined with all of the other partitioned sky chunks, is given in Figure 7. Part of the DAC-DEV workflow was a preference for headerless data files, so in each case the headers are given in the caption. For both files, the prefix G_ or W_ refers to the catalogue (*Gaia* or *WISE* respectively), and the parts after that are detailed below.

1. ID: the designation of the source in the original catalogue.
2. RA: Right Ascension of source
3. DEC: Declination of source
4. BP, G, RP, W1, W2: magnitudes of source in catalogue
5. MATCH_P: Probability of detections being the same object, given by equation 26 of [2] (R11.6.1)
6. SEPARATION: On-sky separation of sources in arcseconds
7. ETA: The logarithmic ratio of match vs non-match only considering photometry; equation 37 of [2]; note for NNM_ETA this is the ratio for this source and the nearest non-matching object in the other catalogue (R11.6.3)

8. `XI`: The logarithmic ratio of match vs non-match only considering astrometry; equation 38 of [2]; note for `NNM_XI` this is the ratio for this source and the nearest non-matching object in the other catalogue (R11.6.2)

9. `AVG_CONT`: The average contamination within each source, as determined using the simulated perturbation component of the Astrometric Uncertainty Function within `perturbation_auf.py`

10. `CONT_F1`: Probability object contains a contaminating source above 1% relative flux

11. `CONT_F10`: Probability object contains a contaminating source above 10% relative flux

12. `ASTRO_SIG_USED`: The astrometric uncertainty as used within `macauff`, saved for cases where we may have updated the astrometric uncertainty using `AstrometricCorrections` (R11.5)

Note, however, that these are just the examples as provided within this test case; the software is capable of including extra columns from the original catalogues if desired, and other secondary match information can easily be added within `macauff` if requested by the community.

6. Future Work and Phase C

While the overall goal of D3.11.4 – the "proof-of-concept" of a large-scale cross-match – has been achieved, there are still various improvements that can be done as we close out Phase B and move into Phase C. First, the code will continue to be worked on, ironing out bugs, improving workflows, adding to documentation. Second, we will continue to interact with the wider LSST community and particularly the TVS and SMWLV Science Collaborations to ensure both that our In-Kind Contribution is satisfied but also to ensure that our efforts have the widest possible reach and impact. While the In-Kind Contribution we have created is software to perform cross-matches, this was largely a limitation of timelines, both of LSST:UK Phase B and Rubin construction; our ultimate deliverable is the LSST-catalogue cross-matches, so we will need to continue a dialogue to ensure that we maximise the potential value-added science achievable from our combined cross-match tables. This dialogue also satisfies the first DAC-DEV Workflow Interface[8], in which prioritisation of ancillary datasets is needed for cases where compute time is lower than required to perform all possible catalogue match combinations. Third, we have a couple of code extensions that could be implemented; the "unknown proper motion" code previously developed could be folded into the matching algorithms, or we could also extend to "real-time" cross-matching and investigate working with brokers such as Lasair to improve the ancillary catalogue collection required to identify transient objects (R11.3, R11.5, R11.8). Depending on community interest, this could conceivably include efforts to increase the useability of the code by community members for their own ends. At the moment, the code is written with a specific framework in mind, and it is envisaged that dedicated (UK) DAC team members will run the code on static LSST Data Releases to deliver its goals as promised, but given enough demand it could be possible to extend functionality to allow "on demand" user catalogue uploads or improvements to the codebase that lower potential barriers to running the code on local hardware. Fourth, the code created to determine the systematic biases of ancillary surveys within D3.11.4, `AstrometricCorrections`, will be used within Phase C as part of our efforts within the SIT-Com team to determine the reliability of LSST astrometry, where our models for the additional systematic astrometric uncertainties and positional offsets will be crucial in disentangling misrepresented *centroid* uncertainties from these other effects. Finally, as previously stated, we will continue to work with the DAC team to perform matches, with a particular eye on those catalogues that will be of use to astronomers currently, both to ensure the runs are scientifically useful but also to "recruit" beta testers (which satisfies the previously-deleted R11.4).

We also plan to submit a paper detailing the mathematical algorithmic extensions to the perturbation component of the AUF (see D3.11.1), publish the software through the Journal of Open Source Software, and publish a small Research Note of the AAS reporting our initial *Gaia*-CatWISE match which supersedes a previous version (R11.2, R11.7).

References

[1] Wilson & Naylor, 2017, MNRAS, 469, 2517

[2] Wilson & Naylor, 2018, MNRAS, 473, 5570

[3] Wilson & Naylor, 2018, MNRAS, 481, 2148

[4] Scaling Relations for WP3.11 Cross-Match Software macauff, accessed March 2023

[5] Wilson, 2022, RNAAS, 6, 60

[6] Plewa & Sari, 2018, MNRAS, 476, 4372

[7] Bosch et al., 2018, PASJ, 70, 5

[7] Wilson, 2023, RASTI, 2, 1

[8] WP3.11 DAC-DEV Interface Document, accessed March 2023

Annex A Software Architecture

The main physical deliverable for this work, as with D3.11.3, is the software, available online. As per the Project Management Plan, software development should be maintained in a version control repository. The main codebase is therefore located at https://github.com/Onoddil/macauff. It features a full test suite for validation, as well as functionality to generate test data to input into the cross-match algorithm for end-to-end verification. The folder structure is as follows.

- Top-level files.
  - CHANGES.rst: changelog file, itemising the updates to the codebase.
  - LICENSE and README.md: details of licensing of codebase, and a top-level overview description of the software.
  - MANIFEST.in, pyproject.toml, and setup.cfg: minor files that aid with the setup and installation of the Python package.
  - python-package.yaml: within the `.github/workflow` folder, this details the `GitHub` virtual environment setup and the running of the test suite.
  - setup.py: main installation file for the Python package, which describes the various dependencies for installing the package, and controls the compiling of `fortran` code.
  - tox.ini: configuration file describing setup of the `tox` environment for test purposes.
  - codecov.yml: configuration file for the code coverage banner available in the README.
- Documentation files, `docs/`.
  - conf.py: configuration file, for Python automated creation of documentation.
  - *.rst: files containing the raw text that makes up the documentation of the codebase.
- Code files, `macauff/`.
  - __init__.py: File used during the installation process, indicating which files and functions should be importable.
  - counterpart_pairing*: Python and Fortran code (`.py` and `_fortran.f90` respectively) to run the cross-match assignment functionality and calculate cross-match probabilities.
  - derive_psf_auf_params.py: Python code to generate the parameterisations of the effects of fitting two objects with a single PSF model in the limit that background noise dominates.
  - fit_astrometry.py: Python code to derive the systematic bias relations in quoted astrometric uncertainties for small regions of sky.
  - galactic_proper_motions.py: Model for unknown proper motions based on particular sky coordinates and source brightness.
  - galaxy_counts.py: Python code to derive differential galaxy count densities based on Galactic extinction and bandpass of observation.

- get_trilegal_wrapper.py: Wrapper for calls to the TRILEGAL simulations for generating Galactic source counts for a particular sightline and bandpass.
- group_sources*: Python and Fortran code used in the generation of "islands" of potential counterparts, independent of other sources in the respective catalogues.
- make_set_list.py: Python script used in group_sources.py to derive "sets" of source overlaps on astrometric considerations, based on lists of sources near to each object.
- matching.py: Python code handling the overall cross-match process.
- misc_functions*: Python and Fortran code that is used in multiple places throughout the cross-match process, and hence cannot be kept within any single script.
- parse_catalogues.py: Python script hosting convenience functions to convert input catalogues to binary `.npy` files, convert final outputs from the matching process into `.csv` files, and perform various tasks such as correct for systematic astrometry biases.
- perturbation_auf*: Python and Fortran code to handle the creation of the perturbation component of the Astrometric Uncertainty Function.
- photometric_likelihood*: Python and Fortran code to handle the creation of source match likelihoods on photometric grounds.
- shared_library.f90: Similar to misc_functions, contains subroutines that are needed across other Fortran modules, and are therefore accessible from those other files.
- Test files, `tests/`.
  - __init__.py: same as in the parent folder.
  - test_*.py: Unit test scripts, one per Python script in the folder above, to ensure consistency and accuracy of the main codebase.
  - test_full_match_process.py: An additional test script, which also includes functionality to generate a "dummy" dataset for testing the end-to-end capability of the codebase.
  - Data files, `data/`.
    - *.txt: example files of the configuration files used in the cross-match process.
    - *.npy: example versions of data generated via `derive_psf_auf_params.py`, usable within matching and used in testing.

In addition, to aid in the review, preliminary documentation – guides to installation and getting started with the codebase – is available at https://onoddil.github.io/macauff/. This documentation, formed from the raw files in `macauff/docs` in the repository, is structured as follows.

- Homepage: brief description of module and links to various starting pages.
- Installation: details the installation process of the module, its dependencies, and how to run the test suite to ensure successful installation.
- Quick Start: description of how to begin working with the package, describing the necessary files to run a cross-match, and giving examples of how to run matches between two catalogues.

- Inputs: more in-depth descriptions of each parameter that should be specified in the input configuration files used in the cross-match process.
- Documentation: collection of available functions within both the Python and Fortran code in the codebase, describing the inputs and outputs from each function or subroutine in detail.
- Algorithms: details of the mathematics behind the cross-matching code
- Pre- and Post-Process: information regarding the steps that happen before and after the “main” aspect of cross-matching, such as the creation of the input files, the core-halo “chunk” model, the removal of duplicated objects in both core and halo of differing chunks, and the creation of output merged cross-match data files.