Specification Document: Tying CSC Astrometry to the *Gaia* Reference Frame. Translation-only approach

Rafael Martínez-Galarza (CfA) Date: September 6, 2022

1 Motivation

This document describes how to achieve fine stack astrometry corrections for release 2.1 of the Chandra Source Catalog, by anchoring the coordinates of stack-level detections for each stack to the *Gaia* reference frame, using a translation-only approach. The reason why we adopt a translation-only approach is that analysis of Chandra aspect solutions predict that the corrections to coordinates due to aspect uncertainties related to rotation and scaling are small. We have in fact corroborated that any additional corrections introduced by rotation and scaling are only a fraction (about 30% or less) of the PSF size at off-axis angles up to 10 arcmins. Therefore, `SKY (X,Y)` positions **will not change**, only the mapping to (RA, Dec). Doing otherwise would imply a major change to the physical coordinate system as defined in the data systems, and we have therefore decided not to include them. We have nevertheless also produced a specification document for the full transformation, including rotation and scaling. We intend to include the full transformation in the next full release of the catalog.

This document first provides an overview of the scientific requirements, and then describes the selected algorithmic approach to achieve these requirements. We also describe main output of the algorithm, i.e., the resulting stack-specific translations that transform CSC detection coordinates to absolute coordinates. We thoroughly test this approach by applying it to existing CSC2.0 stacks. The resulting approach is applicable to the currently existing ~ 7300 CSC2 stacks, as well as new stacks that will be the result of adding newer observations (post-2014) to the catalog database, although we will need manual Quality Assurance for a fraction of the stacks. For CSC 2.1, we expect to have a total of about 10,000 stacks.

2 General goal

The specific goal of the algorithm described here is to tie the CSC astrometry to the *Gaia* reference frame, which will be assumed to be an absolute reference. The catalog astrometry is currently based on telemetry information provided by the aspect camera during the observations and is refined during processing by cross-matching sources from different observations in a stack. However, because of the telescope's spatial resolution, the effect of off-axis observations, and the effect of optical axis tilting over time, the systematic errors in the source positions can be as large as a few arcseconds. *Gaia* astrometry is a suitable absolute reference, both because of its high accuracy and for its all-sky coverage. We therefore require an algorithm capable of linearly translating the CSC coordinates to match the selected absolute reference frame. This should be done at the stack level. As a result, the inter-stack coordinate differences for a given source will be minimized, as they currently can reach several arcseconds.

3 Requirements

The following is a list of scientific requirements for the coordinate update:

- The required linear translation needs to be performed at the individual stack level, i.e., a single translation of coordinates is performed for each stack. The translation can also be applied at the observation level, if desired.
- The approach adopted needs to be applicable to both existing CSC2.0 stacks, as well as to new stacks that will be generated by the addition of new observations.
- The translation requires identifying matching detections in both the source catalog (CSC) an the target catalog (either *Gaia* DR2 or the ALLWISE catalog). Stack fine astrometry must be executed prior to the master match pipeline, a stage at which "sources" and their associated properties do not yet exist. Therefore the coordinates to be used are stack (or obsid) detection coordinates.

Even when this pipeline is applied to existing CSC2.0 data, it is important to always use the coordinates of CSC stack-level detections. Master source coordinates are averages over the stack-level detection coordinates in the uncorrected reference frames, and are therefore not appropriate for this analysis.

- PSF degradation far from the optical axis affects our capability to find suitable matching sources at large off-axis angles. Therefore, we need to impose a limit on the off-axis angle considered for the sources. In the present document we use a limit of 10 arc minutes, which in general gives enough matches for the majority of the stacks while still keeping the PSF relatively small. In combination with this radius, error weighting of the coordinates can be performed, as described below, in order to give less importance in the calculation to detections that have larger errors.

- The stack fine astrometry step should be executed prior to master match step, and it should improve our ability to identify linkages between detections and master sources. As a test of this capability, currently overlapping CSC2 stacks with detections corresponding to the same master source should have their coordinate differences reduced as a result of the stack fine astrometry transformation.

- The systematic offset between the updated CSC coordinates and the absolute reference (*Gaia*) coordinates resulting from the translation need to be smaller or equal than the current systematic offset, and within the limit imposed by the current CSC2 astrometric errors. A good estimate of this error for a particular stack comes from computing the standard deviation of the difference between the target (*Gaia*) and the source (CSC) detection matches.

  Once the stack fine astrometry transformation has been applied, one could leverage this information to estimate the absolute astrometric uncertainty of a source by adding this standard deviation (for the stack) to the relative uncertainty derived for each source from the position error draws. In order to do so, the standard deviation of the difference between the reference *Gaia* coordinates and the corresponding transformed CSC coordinates for each stack should be stored. In § 5.5.7 we describe how to do this.

- The number of stacks requiring manual intervention to achieve the coordinate transformation should be kept at a minimum. Ideally they should be less than 600 stacks. This includes stacks with too few reference sources to perform a meaningful transformation, and automatically transformed stacks where source crowding or source scarcity make the transform unreliable.

- Relevant proper motion considerations need to be take into account. Specifically, sources that could have moved in the sky during the duration of the Chandra Mission by more than the typical astrometric error need to either be corrected for, or ignored in the analysis. Here we recommend to ignore them, as specified below.

4 Weighted Least Squares

In order to perform the translation-only transformations, we will use a least squares approach (i.e., we will compute the translation that minimizes the Euclidean distance between source and target coordinate pairings), but we will add the option -and strongly advice to use this option- to weight the contribution of each pairing to the least squares sum according to the total position error for that pairing.

In the standard least squares approach, the goal is to adjust the parameters of a model function to best fit the data. For the purposes of this specification document, the model function is a linear translation of coordinates between the source data set (CSC coordinates) and the target data set (*Gaia* or ALLWISE coordinates), and the data in question are the target coordinates. The statistic used to evaluate the goodnes of this fit is the sum of the squares of the differences between the transformed source coordinates and the target coordinates.

For a given stack, the data set consists of the standard coordinates of the detections for both the source and target catalogs $(\xi_i, \eta_i)_{\rm CSC}$ and $(\xi_i, \eta_i)_{Gaia}$, where the latter take the role of the dependent variable. The model function takes the form $f(\xi_i^{\rm CSC}, \eta_i^{\rm CSC}, \beta)$, where β is a vector containing the function parameters, in this case the translation in each of the two dimensions, i.e., $\beta = \{\Delta\xi, \Delta\eta\}$. For a given

point i in the data set, its *residual* is given by the Euclidean distance between the transformed coordinates and the target coordinates:

$$r_i = \sqrt{(\xi_i^{Gaia} - \xi_i^{\rm trans})^2 + (\eta_i^{Gaia} - \eta_i^{\rm trans})^2} \tag{1}$$

where $\xi^{\rm trans} = \xi^{\rm CSC} + \Delta\xi$ and $\eta^{\rm trans} = \eta^{\rm CSC} + \Delta\eta$.

For a total of n suitable pairings in a given stack, we therefore attempt to minimize:

$$S = \sum_{i=1}^{n} r_i^2 \tag{2}$$

by adjusting the elements of β.

The position errors for stack detections are not uniform. They depend on the number of counts for a given detection, and also on its off-axis angle. Therefore, detections with large error could bias the translation towards larger values of the coordinate shifts. In order to account for that, we can introduce a weight w_i to each coordinate paring in the sum, so that detections with larger errors will contribute less to the sum:

$$S = \sum_{i=1}^{n} w_i r_i^2 \tag{3}$$

Where $w_i = 1/\sigma^2$, σ being the square root of the product between the semi-major and semi-minor axis of the error ellipses derived from MLE for the corresponding pairing. It has been demonstrated (Aitken, 1935) that if the weights are the reciprocal of the variance of the measurement ($1/\sigma^2$), then the derived parameters of the transformation are a best linear unbiased estimator.

This weighted least-squares (WLS) approach can be easily implemented using Python's `scipy.optmize.least_square` tool, as described in § 5.5.

5 Implementation for CSC fine stack astrometry

In this section we describe how to use weighted least squares to perform an update of the CSC coordinates, using *Gaia* coordinates as the target absolute reference. We first describe how to prepare the input data. We then describe the parametrization of the WLS method that best suits our purposes. Depending on the number of CSC sources within a stack, and depending on the number of *Gaia* matches found for that set of sources, the coordinate correction can be performed either directly, or using an intermediate translation involving another catalog of sources that has been independently matched to *Gaia*. Below we describe how to perform both approaches, using the *ALLWISE* catalog as an intermediate catalog for those stacks requiring a two-step approach.

5.1 Preparing the datasets

The input to this new CSC stack astrometry algorithm needs to be a list of detection coordinates and their respective postion errors (semi-minor and semi-major axis of the error ellipses) for each stack (the source list), accompanied by a list of source coordinates for *Gaia* sources in a similar area of the sky. No position errors are required for the *Gaia* set.

Therefore, for each stack a list of coordinates and position errors for detections needs to be prepared. These coordinates should be in units of arcseconds, and should be defined on the tangent plane to the stack in question, i.e., standard coordinates (ξ, η) should be used[1]. The standard coordinates are defined as:

$$\begin{aligned} \xi &= \frac{\cos\delta \sin(\alpha - \alpha_0)}{\sin\delta \sin\delta_0 + \cos\delta \cos\delta_0 \cos(\alpha - \alpha_0)} \\ \eta &= \frac{\sin\delta \cos\delta_0 - \cos\delta \sin\delta_0 \cos(\alpha - \alpha_0)}{\sin\delta \sin\delta_0 + \cos\delta \cos\delta_0 \cos(\alpha - \alpha_0)} \end{aligned} \tag{4}$$

[1] Alternatively, physical X and Y coordinates can be used.

where (α_0, δ_0) refer to the coordinates of the tangent plane reference point. The inverse transformations are obtained as:

$$\begin{aligned} \tan(\alpha - \alpha_0) &= \frac{\xi}{\cos\delta_0 - \eta \sin\delta_0} \\ \sin\delta &= \frac{\sin\delta_0 + \eta\cos\delta_0}{\sqrt{1+\xi^2+\eta^2}} \end{aligned} \tag{5}$$

Since the stack fine astrometry step will happen before master sources are created, the coordinates to be used here are the detection coordinates. For each existing or new stack that has gone through the source detection pipeline, this list can be constructed by first identifying the stack centroid, defined by the stack event list file header keywords `ra_stack` and `dec_stack`, and then performing a cone search of all stack detections that have been classified as TRUE or MARGINAL[2] within a certain radius of this centroids. A search radius of 10 arcminutes strikes a good compromise between the potential number of control sources and the undesirable effects of PSF degradation too far from the optical axis. Convex hulls should be excluded from the list of detections for given stacks. Detections that are flagged for saturation, i.e. cratered detections, (`sat_src_flag=TRUE`) should be excluded. Specifically, detections for which `o.sat_src_flag = 1` for any of the contributing observations should also be excluded.

Next is to construct the target list of sources. Since many more *Gaia* sources are expected in any particular region than CSC sources, this list can be constructed for each stack by performing a cross-match between the list of detection coordinates for the stack, and the *Gaia* DR2 catalog (or source catalog) of sources. For this cross match, a reasonable search radius is 3 arcseconds, as the largest between-stack offsets that we have observed in CSC2 is about 2.5 arcseconds. It is recommended to get all matches for both *Gaia* and ALLWISE for all stacks that are determined to be suitable for processing, and store them locally prior to running the WLS translation. Additionally, the density of *Gaia* and *ALLWISE* sources within 10 arcminutes of each stack centroid should be recorded prior to running the pipeline. This is because, as specified later, stacks with very high density of either *Gaia* or *ALLWISE* sources are more prone to error due to the increased likelihood of serendipitous matches that are not real.

In addition, the list of *Gaia*-ALLWISE source associations is needed for each stack. This is to support the two-step transformation described in § 5.5. As part of the second data release of *Gaia* (DR2), the *Gaia* source catalog has been independently matched with several external catalogs, including ALLWISE. The matches table is called `gaiadr2.allwise_best_neighbors` and only includes the detection identifiers for both catalogs. A procedure is detailed in § 5.5 describing how to search for the relevant. *Gaia*/ALLWISE matches.

The input needed is therefore constructed from:

- The source list of coordinates (with positon errors)
- The target list of coordinates for each stack constructed as described above.
- The auxiliary list of *Gaia*-ALLWISE matches is all the input we need.
- The density of *Gaia* sources within $r_{\rm stack}$ arcminutes of each processed stack, for example, in units of sources per square arcminute.
- The density of *ALLWISE* sources within $r_{\rm stack}$ arcminutes of each processed stack, for example, in units of sources per square arcminute.

In Figure 1 we show the distribution of the number of CSC2 detections within a stack. There are 223 stacks with a single CSC2 detection, 221 with 2 detections, and 204 with 3 detections. When cross-matched with *Gaia* and ALLWISE within 3" of each detection, this results in a total of 977 stacks (out of 7243, i.e., the 13%) with few enough matches to justify manual QA (see 5.5). The number of

[2]Although MARGINAL detections typically have lower signal to noise ratios, and subsequently larger position error ellipses, their position error distributions do not dramatically depart from the distribution for TRUE sources. We therefore include them here, as the benefit of having more matches outweights the larger position errors, specially since astroalign deals well with outliers.

stacks requiring manual QA will ultimately increase, because there will be additional stacks with enough matches, but that are too crowded with *Gaia* or ALLWISE sources and will require inspection. Overall, it is expected that the total fraction of stacks that will require QA is about 15% (or about 1500 stacks) for CSC2.1.

5.2 Proper motion considerations

Some of the *Gaia* DR2 sources have large radial proper motion vectors, and their recorded position in both catalogs corresponds to a particular observation epoch, or to an average over epochs. If the proper motion vector results in a change of coordinates larger than the relative astrometric error in CSC, which averages around 0.5 arcseconds (this is the case for proper motions larger than about 25 mas/yr), then there is a chance that the position entries for the source in both source and target catalogs correspond to different epochs, as the source is not observed at the same time for both surveys.

The fraction of *Gaia* sources with recorded radial proper motions that fall in this high proper motion category is less than 2%, and less than 1% if we extend the tolerance to 50 mas/yr. Since sources with radial proper motion between 25 and 50 mas/yr are unlikely to significantly affect the fine astrometry, while providing additional sources in less crowded fields, in what follows we ignore those sources sources with radial proper motion larger than 50 mas/yr for the analysis. In order to exclude all matches that include *Gaia* sources with radial proper motions larger than 50 mas/yr, it is necessary to filter using columns *pmra* and *pmdec* from the *Gaia* DR2 source catalog, as described in § 5.5.

Gaia sources with null proper motions in both RA and DEC should be included as well, as they correspond to extragalactic sources with very small proper motions.

Note that if there are corresponding *ALLWISE* sources to *Gaia* sources with high proper motions, those corresponding *ALLWISE* sources should also be excluded from the *ALLWISE* list. Therefore,

5.3 Parametrization and implementation of the WLS translation

The `scipy` implementation of the least squares method solves the problem of minimizing S (Eq. 3) by setting the gradient of the model function with respect to the parameters to zero:

$$\frac{\partial S}{\partial \Delta\xi} = 2\sum_i w_i r_i \frac{\partial r_i}{\partial \Delta\xi} = 0, \qquad \frac{\partial S}{\partial \Delta\eta} = 2\sum_i w_i r_i \frac{\partial r_i}{\partial \Delta\eta} = 0 \tag{6}$$

In order to make sure that the function has a derivative everywhere and is robust for optimization, prior to the minimization the method also passes the residual as an input to a sub-linear loss function $\rho(z)$ that will allow the differentiation and will add robustness against outliers[3]. There are several options for this loss function, and our test indicates that for the problem at hand, which is a linear problem, the Cauchy loss is the best option to deal with strong outliers.

The derivatives are solved numerically starting from some starting point, which by default can be set to 0 (no translation). All that needs to be passed to `scipy.optimize.least_squares` is a function f that evaluates the residual (and multiplies it by the weights), the starting point or initial guess for the parameters, the type of loss function that will be applied, the bounds for the parameters (i.e., the maximum and minimum values they can adopt), and the arguments to be passed to the function f, namely the source coordinates, the target coordinates, and the errors in the source coordinates (the CSC position errors). We adopt bounds of ± 5 arcseconds for each of the model parameters, as we do not expect shifts in coordinates larger than 3 arcseconds, but we leave some additional wiggle space in case there are extreme outliers. A call to the WLS method then looks like this:

```
# A function to perform the transformation as a matrix multiplication
def model(pars, X):
    return np.matmul(np.array([[1.0,0.0,pars[0]],
    [0.0,1.0,pars[1]],[0.0,0.0,1.0]]),[X[0], X[1], 1.0])

# A function to estimate the residual, multiplied by the weights
```

[3]See `https://scipy-cookbook.readthedocs.io/items/robust_regression.html` for a discussion on the loss function.

```
def fun(pars, X, Y, X_err):
    return np.dot(1.0/(X_err*X_err),
    np.sqrt((model(pars,X)[0]-Y[0])**2 +  (model(pars,X)[1]-Y[1])**2))

# Set starting point
x0 = (0.0,0.0)

# The call to least_squares
res = least_squares(fun, x0, loss='cauchy',
bounds=(-5, 5), args=([xi_csc2,eta_csc2],[xi_gaia,eta_gaia],csc2_err), verbose=1)
```

The use of the Cauchy loss function also allows for most robustness in the case of crowed fields, where it is most likely to have ambiguous matches between CSC and either *Gaia* or ALLWISE.

An additional consideration: although the cross matching is performed to find all *Gaia* (or ALLWISE) matches within 3" of each stack detection, for a given stack we can typically find enough of these matches that are within 1" of the stack detection. Therefore in the algorithm below we start by first counting the number of matches in the three ranges (0" − 1"), (1" − 2")m and (2" − 3"), and then selecting the range that has at least 4 matches and that also has the smallest standard deviation of the separation between matches. Only sources in the selected range will be used to perform the transformation. This approach ensures that there will be less serendipitous matches, while at the same time it will also allow us to identify stacks for which the shift is larger than 1".

To estimate the probability of finding serendipitous matches in crowded fields, let us look at stack `acisfJ1115042m611530_001`, in NGC 3603, which is the CSC 2.0 stack with the largest number of CSC/*Gaia* matches (2848). For this stack, in the most crowded region, the typical separation between CSC detections is about 2", whereas the typical separation between *Gaia* sources is about 1", but the typical distance between matching CSC/*Gaia* detections is close to 0.2". Given a good set of matching detections between the two surveys, what is the probability of finding exactly the same matches (within a similar tolerance of 0.2") if a translation of more than 1" between surveys is performed? This probability would be close to 1 if the separation between *Gaia* sources (r_G) in the field was close to the 0.2" tolerance, and decreases as $1/r_G^2$ as the separation between sources increases. So, since there is a factor of 5 between separation (1") and tolerance (0.2"), we estimate this probability to be of less than 0.04.

5.4 Two-step transformation

Depending on the environment of a stack (for example, on whether the stack in question corresponds to a galactic or extragalactic field), one can find more valid matches for the CSC detections in the ALLWISE catalog than in the *Gaia* catalog. For those stacks that have more valid matches with ALLWISE than they have *Gaia*, a two-step translation allows to find a transformation between CSC and *Gaia* coordinates via an intermediate dataset, i.e., the ALLWISE point source catalog. The basic idea is simple: the resulting translation is found by combining two independent translations: one between *Gaia* and ALLWISE coordinates ($\mathbf{T_1}$) and one between ALLWISE and CSC coordinates ($\mathbf{T_2}$). $\mathbf{T_1}$ and $\mathbf{T_2}$ are found using the same WLS formulation described in the previous sections. The only difference for $\mathbf{T_1}$ is that since an independent matching has been performed between *Gaia* and ALLWISE sources as part of DR2, no cross-matching is needed to find potential corresponding pairs. The coordinates are obtained directly from the respective tables, after checking against the `gaiadr2.allwise_best_neighbors` table.

For a given stack, a $\mathbf{T_1}$ translation is found by running the WLS approach with the default parameters (Cauchy loss, error weighting) using the ALLWISE coordinates as the source list and the *Gaia* coordinates as a target list. Because no significant PSF smearing happens for WISE, we are not constrained to limit the search for *Gaia*/ALLWISE matches to within 10 arcminutes of the stack centroid. This transformation is likely to be possible for almost all of the CSC stacks, as there is a large enough number of matches between *Gaia* and ALLWISE all over the sky, as shown in Fig. 2.

Similarly, a $\mathbf{T_2}$ translation is found by running the WLS approach with the default parameters using the CSC coordinates as the source list, and the ALLWISE coordinates as the target list. The final

transformation $\mathbf{T}$ is found by multiplying these two translations:

$$\begin{bmatrix} \alpha_{\rm corr} \\ \delta_{\rm corr} \\ 1 \end{bmatrix} = \mathbf{T_2 T_1} \begin{bmatrix} \alpha_0 \\ \delta_0 \\ 1 \end{bmatrix} \tag{7}$$

5.5 Algorithm

The end-to-end workflow for the fine astrometry correction is as follows. We separate the algorithm into a *Gaia* alone section, and a *Gaia*+ALLWISE two-step transformation section. The decision on whether a two-step translation is required should be done based on a comparison of the total number of valid matches between CSC and *Gaia*, and between CSC and ALLWISE.

5.5.1 Input

The pipeline is run per stack. For each stack, the inputs to the pipeline are:

- *csc-table*: For each stack, a table of stack detections within a radius $r_{\rm stack}$ of the stack centroid containing the detection identifiers, coordinates, position errors, and all flags for each detection. The coordinates of the tangent plane reference point (`ra_stack` and `dec_stack`) should also be available. We default $r_{\rm stack} = 10$ arcmin, but the specific value should be a pipeline parameter. Detections that are flagged for saturation (`sat_src_flag=TRUE`) should be excluded. Specifically, detections for which `o.sat_src_flag = 1` for any of the contributing observations should be excluded. Stack detections corresponding to convex hull centroids should also be excluded. Manually included detections (`man_inc=True`) can be included as long as for the detection `src_quality` indicates that the detection or true or marginal (no false detections should be included). The table should include the following columns for each detection: `detect_stack_id`, `s.region_id`, `ra_stack`, `dec_stack`, `s.ra`, `s.dec`, `s.err_ellipse_r0`, `s.err_ellipse_r1`, `s.err_ellipse_ang`.

- *Gaia source density:* For each stack, the density $n_{\rm Gaia}$ of *Gaia* sources within $r_{\rm stack}$ should be computed, and given in unit of sources per square arcminute.

- *gaia-table*: For each stack, a table with all the *Gaia* sources (from table `gaiadr2.gaia_source`) within $r_{\rm stack}$ of the stack centroid, including the source identifier (`source_id`), coordinates (`ra, dec`), and proper motions in RA and DEC (`pmra, pmdec`). *Gaia* sources with radial proper motions larger than 50 mas/yr should be excluded. To calculate the radial proper motion, `pmra` and `pmdec` should be added in quadrature. The *Gaia* source catalog can be dowloaded at: `http://cdn.gea.esac.esa.int/Gaia/gdr2/gaia_source/`

- *allwise-table*: For each stack, a table with all the ALLWISE sources within $r_{\rm stack}$ of the stack centroid, including the source identifier (`allwise_oid`), coordinates (`ra, dec`), coordinate errors (`sigra, sigdec`), and contamination and confusion flag (`cc_flags`). Only sources with `cc_flags=0` should be included. Sources whose corresponding match in the *gaia-wise-table* association table (`gaiadr2.allwise_best_neighbors`, se below) is a high proper motion source (> 50 mas/yr) should also be excluded. Information for a bulk download of the *ALLWISE* tables can be found here: `http://wise2.ipac.caltech.edu/docs/release/allwise/expsup/sec1_5.html#bulk`. Column descriptions are found here: `https://wise2.ipac.caltech.edu/docs/release/allwise/expsup/sec2_1a.html`

- *ALLWISE source density:* For each stack, the density $n_{\rm ALLWISE}$ of *ALLWISE* sources within $r_{\rm stack}$ should be computed, and given in unit of sources per square arcminute.

- *gaia-wise-table*: The full table of *Gaia*/ALLWISE associations (table `gaiadr2.allwise_best_neighbors` from the DR2). It can be found here: `https://gaia.aip.de/metadata/gdr2/allwise_best_neighbour/`

5.5.2 Deciding whether a two-step approach is preferred

.

The first decision to be made is whether the transformation will be done using the *Gaia* catalog alone, or if the intermediate step using ALLWISE is required. So, for a given stack, determine this by counting the total number of valid matches (within 3") between the CSC detections of that stack, and both the *Gaia* and ALLWISE catalogs.

1. Consider the *csc-table* and *gaia-table* tables. Perform a cross-match between the two tables to find all valid *Gaia* sources within 3 arcseconds of each valid CSC stack detection. The total number of matches found is $n_{\mathrm{match}^{\mathrm{Gaia}}}$.

2. Now consider the *csc-table* and *wise-table* tables. Perform a cross-match between the two tables to find all valid ALLWISE sources within 3 arcseconds of each valid CSC stack detection. The total number of matches found is $n_{\mathrm{match}^{\mathrm{WISE}}}$.

3. If ($n_{\mathrm{match}^{\mathrm{Gaia}}} \geq n_{\mathrm{match}^{\mathrm{WISE}}}$) use the direct approach of §5.5.3. Otherwise, use the two-step approach of §5.5.4.

5.5.3 Translation using *Gaia* alone

1. For each stack, consider the *csc-table* and *gaia-table* tables. Perform a cross-match between the two tables to find all valid *Gaia* sources within 3 arcseconds of each valid CSC stack detection. Include all matches, even if they are ambiguos (i.e., even if more than one *Gaia* source matches the relevant CSC detection).

2. Divide the matches in three groups according to their separation r_{match}:

   - Group 1: r_{match} such that 0" $< r_{\mathrm{match}} \leq$ 1".
   - Group 2: r_{match} such that 1" $< r_{\mathrm{match}} \leq$ 2".
   - Group 3: r_{match} such that 2" $< r_{\mathrm{match}} \leq$ 3".

3. For each group i:

   - Define $n_{\mathrm{match,i}}$ as the total number of CSC detections with at least one match.
   - For each match pair j in group i, consider the component separation (in sky coordinates) in both spatial dimensions Δx_j and Δy_j.
   - Calculate the standard deviation of Δx_j and of Δy_j over all pairs in group i. These are $\sigma_{\mathrm{Gaia,x,i}}$ and $\sigma_{\mathrm{Gaia,y,i}}$.
   - Estimate the quantity $\sigma_{\mathrm{Gaia,i}} = \sqrt{\sigma_{\mathrm{Gaia,x,i}} * \sigma_{\mathrm{Gaia,y,i}}}$

4. For the next steps, use only the group i that has at least three matches ($n_{\mathrm{match,i}} \geq 3$) and that has the lowest $\sigma_{\mathrm{Gaia,i}}$. If none of the groups has at least 3 matches, then use the one with the largest $n_{\mathrm{match,i}}$, regardless of the standard deviation values, unless there is a tie, in which case add the condition that the group should have the smallest $\sigma_{\mathrm{Gaia,i}}$.

5. Coordinates should be input in units of arcseconds, not degrees or hour angles, and should be the standard coordinates (ξ, η) resulting from reprojecting (RA, DEC) onto the plane tangent to the stack reference position `(ra_stack, dec_stack)`. Starting from the coordinates (RA, DEC) for the detection in decimal degrees, first perform the following transformations for both the coordinates and the coordinate errors:

```
c1 = SkyCoord(ra=ra_csc2*u.deg, dec=dec_csc2*u.deg, frame='icrs')

xi_csc2_1 = (cos(c1.dec.rad)*sin(c1.ra.rad - ra_stack_rad))
            /(sin(c1.dec.rad)*sin(dec_stack_rad)
            +cos(c1.dec.rad)*cos(dec_stack_rad)
```

```
            *cos(c1.ra.rad - ra_stack_rad))

xi_csc2 = 3600.*xi_csc2_1*(180./pi)

eta_csc2_1 = (sin(c1.dec.rad)*cos(dec_stack_rad)
            -cos(c1.dec.rad)*sin(dec_stack_rad)
            *cos(c1.ra.rad - ra_stack_rad))
            /(sin(c1.dec.rad)*sin(dec_stack_rad)
            +cos(c1.dec.rad)*cos(dec_stack_rad)
            *cos(c1.ra.rad - ra_stack_rad))

eta_csc2 = 3600.*eta_csc2_1*(180./pi)

c2 = SkyCoord(ra=ra_gaia*u.deg, dec=dec_gaia*u.deg, frame='icrs')

xi_gaia_1 = (cos(c2.dec.rad)*sin(c2.ra.rad - ra_stack_rad))
            /(sin(c2.dec.rad)*sin(dec_stack_rad)
            +cos(c2.dec.rad)*cos(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))

xi_gaia = 3600.*xi_gaia_1*(180./pi)

eta_gaia_1 = (sin(c2.dec.rad)*cos(dec_stack_rad)
            -cos(c2.dec.rad)*sin(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))
            /(sin(c2.dec.rad)*sin(dec_stack_rad)
            +cos(c2.dec.rad)*cos(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))

eta_gaia = 3600.*eta_gaia_1*(180./pi)
```

where `ra_stack_rad` and `dec_stack_rad` are the stack's tangent plane reference coordinates, in radians.

If the stack contains coordinates that cross the 24 hour angle line (i.e., if it contains detections in both sides of this line), then prior to this step add 360.0 to the RA in decimal degrees for all sources with RA $< 12h$. This should be done for all CSC detections, as well as all *Gaia* and ALLWISE sources.

6. The resulting array of *CSC* coordinates is the "source" list for in the WLS approach, while the resulting array of *Gaia* coordinates is the "target" list.

7. The coordinates should be stored in individual arrays, namely `xi_csc2`, `eta_csc2`, `xi_gaia`, `eta_gaia2`.

8. Compute the circular approximation of the CSC position errors to use as the vector of weights for WLS:
$$\mathbf{w} = [1/(r_0 r_1), 1/(r_0 r_1)]$$
where r_0 and r_1 are the semi-major and semi-minor axes of the position error ellipses. Save the resulting vector as variable `w`.

9. Run the WLS approach with the resulting coordinates and errors:

```
# A function to perform the transformation as a matrix multiplication
def model(pars, X):
```

```
    return np.matmul(np.array([[1.0,0.0,pars[0]],
    [0.0,1.0,pars[1]],[0.0,0.0,1.0]]),[X[0], X[1], 1.0])

# A function to estimate the residual, multiplied by the weights
def fun(pars, X, Y, weights):
    return np.dot(weights, np.sqrt((model(pars,X)[0]-Y[0])**2
        + (model(pars,X)[1]-Y[1])**2))

# Set starting point
x0 = (0.0,0.0)

# The call to least_squares
res = least_squares(fun, x0, loss='cauchy',
bounds=(-5, 5), args=([xi_csc2,eta_csc2], [xi_gaia,eta_gaia], w),
    verbose=1)
```

10. The translations in ξ and η are stored in `res.x` and `res.y`. Apply the transformation to the input CSC coordinates by multiplying the resulting translation matrix $\mathbf{T}$ with the vector of coordinates `(ra_csc_arcsec,dec_csc_arcsec, 1)` for each detection, and store the translation parameters for the stack.

11. If $n_{\rm match,i} < 10$, and the density of *Gaia* sources $n_{\rm Gaia}$ in the current stack is larger than 100 sources per square arcminute, then flag the stack for manual QA and continue.

5.5.4 Two-step translation

1. For each stack, consider the *csc-table* and *allwise-table* tables. Perform a cross-match between the two tables to find all valid ALLWISE sources within 3 arcseconds of each valid CSC stack detection. Include all matches, even if they are ambiguos (i.e., even if more than one ALLWISE source matches the relevant CSC detection)

2. Divide the matches in three groups according to their separation $r_{\rm match}$:

   - Group 1: $r_{\rm match}$ such that 0" $< r_{\rm match} \leq$ 1".
   - Group 2: $r_{\rm match}$ such that 1" $< r_{\rm match} \leq$ 2".
   - Group 3: $r_{\rm match}$ such that 2" $< r_{\rm match} \leq$ 3".

3. For each group i:

   - Define $n_{\rm match,i}$ as the total number of CSC detections with at least one match.
   - For each match pair j in group i, consider the component separation (in sky coordinates) in both spatial dimensions Δx_j and Δy_j.
   - Calculate the standard deviation of Δx_j and of Δy_j over all pairs in group i. These are $\sigma_{\rm WISE,x,i}$ and $\sigma_{\rm WISE,y,i}$.
   - Estimate the quantity $\sigma_{\rm WISE,i} = \sqrt{\sigma_{\rm WISE,x,i} * \sigma_{\rm WISE,y,i}}$

4. For the next steps, use only the group that has at least three matches ($n_{\rm match,i} \geq 3$) and that has the lowest $\sigma_{\rm WISE,i}$. If none of the groups has at least 3 matches, then use the one with the largest $n_{\rm match,i}$, regardless of the standard deviation values, unless there is a tie, in which case add the condition that the group should have the smallest $\sigma_{\rm WISE,i}$.

5. Coordinates should be input in units of arcseconds, not degrees or hour angles, and should be the standard coordinates (ξ, η) resulting from reprojecting (RA, DEC) onto the plane tangent to the stack reference position `(ra_stack, dec_stack)`. Starting from the coordinates (RA, DEC) for the detection in decimal degrees, first perform the following transformations for both the coordinates and the coordinate errors:

```
c1 = SkyCoord(ra=ra_csc2*u.deg, dec=dec_csc2*u.deg, frame='icrs')

xi_csc2_1 = (cos(c1.dec.rad)*sin(c1.ra.rad - ra_stack_rad))
            /(sin(c1.dec.rad)*sin(dec_stack_rad)
            +cos(c1.dec.rad)*cos(dec_stack_rad)
            *cos(c1.ra.rad - ra_stack_rad))

xi_csc2 = 3600.*xi_csc2_1*(180./pi)

eta_csc2_1 = (sin(c1.dec.rad)*cos(dec_stack_rad)
            -cos(c1.dec.rad)*sin(dec_stack_rad)
            *cos(c1.ra.rad - ra_stack_rad))
            /(sin(c1.dec.rad)*sin(dec_stack_rad)
            +cos(c1.dec.rad)*cos(dec_stack_rad)
            *cos(c1.ra.rad - ra_stack_rad))

eta_csc2 = 3600.*eta_csc2_1*(180./pi)

c2 = SkyCoord(ra=ra_wise*u.deg, dec=dec_wise*u.deg, frame='icrs')

xi_wise_1 = (cos(c2.dec.rad)*sin(c2.ra.rad - ra_stack_rad))
            /(sin(c2.dec.rad)*sin(dec_stack_rad)
            +cos(c2.dec.rad)*cos(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))

xi_wise = 3600.*xi_wise_1*(180./pi)

eta_wise_1 = (sin(c2.dec.rad)*cos(dec_stack_rad)
            -cos(c2.dec.rad)*sin(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))
            /(sin(c2.dec.rad)*sin(dec_stack_rad)
            +cos(c2.dec.rad)*cos(dec_stack_rad)
            *cos(c2.ra.rad - ra_stack_rad))

eta_wise = 3600.*eta_wise_1*(180./pi)
```

where `ra_stack_rad` and `dec_stack_rad` are the stack's tangent plane reference coordinates, in radians.

If the stack contains coordinates that cross the 24 hour angle line (i.e., if it contains detections in both sides of this line), then prior to this step add 360.0 to the RA in decimal degrees for all sources with RA $< 12h$. This should be done for all CSC detections, as well as all *Gaia* and ALLWISE sources.

6. The resulting array of *CSC* coordinates is the "source" list for the WLS transformation, while the resulting array of ALLWISE coordinates is the "target" list.

7. The coordinates should be stored in individual arrays, namely `xi_csc2`, `eta_csc2`, `xi_wise`, `eta_wise2`.

8. Compute the circular approximation of the CSC position errors:

$$\sigma_{\rm CSC} = r_0 \times r_1$$

where r_0 and r_1 are the semi-major and semi-minor axes of the position error ellipses.

9. Compute the circular approximation of the ALLWISE position errors (also in arcseconds):

$$\sigma_{\rm WISE} = \mathrm{sigra} \times \mathrm{sigdec}$$

where *sigra* and *sigdec* are the errors for the detection from the ALLWISE table.

10. Add these two errors in quadrature:

$$\sigma_{\rm tot} = \sqrt{\sigma^2_{\rm CSC} + \sigma^2_{\rm WISE}}$$

11. Compute the final vector of weights as:

$$\mathbf{w} = [1/\sigma_{\rm tot}, 1/\sigma_{\rm tot}]$$

where the operation is computed element-wise. Save the resulting vector of weigths as `w`.

12. Run the WLS approach with the resulting coordinates and erros:

```
# A function to perform the transformation as a matrix multiplication
def model(pars, X):
    return np.matmul(np.array([[1.0,0.0,pars[0]],
    [0.0,1.0,pars[1]],[0.0,0.0,1.0]]),[X[0], X[1], 1.0])

# A function to estimate the residual, multiplied by the weights
def fun(pars, X, Y, weights):
    return np.dot(weights, np.sqrt((model(pars,X)[0]-Y[0])**2
        +  (model(pars,X)[1]-Y[1])**2))

# Set starting point
x0 = (0.0,0.0)

# The call to least_squares
res = least_squares(fun, x0, loss='cauchy',
bounds=(-5, 5), args=([xi_csc2,eta_csc2], [xi_wise,eta_wise], w), verbose=1)
```

13. The translations in ξ and η are stored in `res.x` and `res.y`. Apply the transformation to the input CSC coordinates by multiplying the resulting translation matrix $\mathbf{T_1}$ with the vector of coordinates `(ra_csc_arcsec,dec_csc_arcsec, 1)` for each detection, and store the translation parameters ($\mathbf{T_1}$) for the stack.

14. If $n_{\rm match,i} < 10$, and the density of ALLWISE sources $n_{\rm WISE}$ in the current stack is larger than 100 sources per square arcminute, then flag the stack for manual QA and continue.

15. Now independently compute the *Gaia*/ALLWISE transformation. The `gaiadr2.allwise_best_neighbors` table will be needed.

16. Identify all ALLWISE source identifiers and coordinates in *allwise-table*, which should contain sources within 10 arcminutes of the stack centroid.

17. Find the *Gaia* identifier associated with each ALLWISE identifier in table `gaiadr2.allwise_best_neighbors`. If more than one *Gaia* source is listed as a best neighbor, then ignore the association. This is, keep only one-to-one associations.

18. Retrieve the coordinates and position errors of each associated ALLWISE source from the ALLWISE source catalog (*allwise-table*)

19. Run the WLS approach with the resulting coordinates and errors (in this case, the ALLWISE errors only):

```
# A function to perform the transformation as a matrix multiplication
def model(pars, X):
    return np.matmul(np.array([[1.0,0.0,pars[0]],
    [0.0,1.0,pars[1]],[0.0,0.0,1.0]]),[X[0], X[1], 1.0])

# A function to estimate the residual, multiplied by the weights
def fun(pars, X, Y, weights):
    return np.dot(weights, np.sqrt((model(pars,X)[0]-Y[0])**2
        +  (model(pars,X)[1]-Y[1])**2))

# Set starting point
x0 = (0.0,0.0)

# The call to least_squares
res = least_squares(fun, x0, loss='cauchy',
bounds=(-5, 5), args=([xi_wise,eta_wise], [xi_gaia,eta_gaia], w_WISE), verbose=1)
```

Store the resulting translation parameters in matrix form in $\mathbf{T_2}$

20. Find the final solution translation by multiplying $\mathbf{T} = \mathbf{T2T1}$, and store the result as the solution for the current stack.

5.5.5 Stacks with no matches

In the unlikely event of a stack having no *Gaia* or ALLWISE matches for any of the CSC detections, no correction is possible. Flag the stack for manual QA.

5.5.6 Transform the coordinates back to decimal degrees

After the solution translation has been applied to the CSC coordinates for each stack, the updated coordinates in arcseconds should be transformed back to decimal degrees. In order to do so, the inverse transformations in Eq. 5 should be applied a this point.

5.5.7 Estimating source absolute position uncertainties

.

The astrometric corrections derived from the stack astrometry pipeline will be used to update the *master level* absolute errors for CSC sources once all source positions have been corrected as part of this stack astrometry pipeline. At the stack level, the MLE-derived errors will be kept, without modification. The reason is that for a significant fraction of the stacks, we will not have enough *Gaia*-CSC or *ALLWISE*-CSC matches to estimate a reliable astrometric error at the stack level. However, an statistical analysis over all master level sources should provide enough information to estimate the updated astrometry uncertainty that comes from errors in the aspect solution, calibration, determination of guide star positions, etc.

In order to determine the total absolute astrometric error for CSC sources, the same procedure that was used in CSC 2.0 by cross-matching CSC 2.0 master sources with SDSS sources will be employed here, but using *Gaia* as the reference positions instead of SDSS. The procedure is described here:

`https://cxc.harvard.edu/csc/memos/files/Rots_CSC2AstrometricError.pdf`

This procedure will be performed by the science team and will result in a single astrometric error value added in quadrature to the MLE-derived errors for all master level sources. The uncertainties will be updated in the database with a post-processing migration.

5.6 Storing translation matrices in the archive

Transformation matrices (which in the case of translation-only transformation are diagonal matrices) should be archived in the asol file for each stack. A way to do this is by storing the matrices in a xfm3 file, which can be added as an HDU extension of the stack-level asol file. For those stacks requiring two transformations, both matrices should be stored.

After the transform has been created and stored, then it has to be applied to the stacks. How this is done in CSC 2.1 processing will be different depending on whether we are processing a new stack or updating existing data from release 2.0. The way in which this will be done is currently TBD, but in the implementation it should happen at this stage of the pipeline.

Below is a list of files that need to be updated with the resulting updated WCS coordinates. Note that, because the transformation will be translation only, we do not expect a change in the ROLL angles.

```
Per-ObsId products
   EVT3            - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DE
   ECORRIMG    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TA
   ECORRIMG_JPG    - Inherit from updated ECORRIMG
   BKGIMG    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG
   EXPMAP    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG
   ASPSOL3    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if present
         ASPSOL:ra,dec; add the applied transform (i.e., translation only) as a new HDU STKXFM
     following the XFM HDU and using the same format
   BADPIX3    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if present
   FOV3    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (
   PIXMASK    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TAR
   POLY3    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if present)
   SEXPMAP    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TAR

Per-ObsId source region data products
   REGEVT3         - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DE
   REGIMG    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG
   REGIMG_JPG    - Inherit from updated REGIMG
   PSF/BINPSF    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_
   PSF_JPG    - Inherit from updated PSF
   REGEXPMAP    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T
   SPECTRUM    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TA
   ARF    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if present)
   RMF    - Nothing needs to be updated
   LIGHTCURVE    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if pres
   ECF90REG<BAND>  - Update WCS (EQSRC), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DE
         ***ECF90REG<BAND> should have an EQSRC descriptor populated with WCS information***
   DRAWS    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if present),
         SRCDRAWS:ra,dec
   APERPHOT    - Nothing needs to be updated
         ***Headers do not include standard coordinate information (RA*/DEC*/ROLL* keywords are miss

Stack products
   STKEVT3       - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_
   STKECORRIMG     - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DE
   STKECORRIMG_JPG - Inherit from updated STKECORRIMG
   STKBKGIMG    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T
   STKEXPMAP    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T
   STKFOV3    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TAR
   SENSITY    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TAR
   MRGSRC    - Update WCS (EQSRC), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG
         SRCLIST:ra,dec, SRCFIT:detect_{ra|dec},src_{ra|dec}_<band>, ext_{ra|dec}_<band>,
```

```
     WAVSRC:RA,DEC,wav<n><band>_{ra|dec},MPNTSRC:RA,DEC,MEXTSRC:CENTER_{RA|DEC},
     NO_DETECT:RA,DEC
   THETAMEAN    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T

Stack source region data products
   STKREGEVT     - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_
   STKREGIMG    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T
   STKREGIMG_JPG   - Inherit from updated STKREGIMG
   STKREGIMG3_JPG  - Inherit from updated STKREGIMG
   STKREGEXP    - Update WCS (EQPOS), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_T
   SRCREG/BKGREG   - Update WCS (EQSRC), all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DE
         ***EQSRC is not correctly populated with WCS information***
   STKDRAWS    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if presen
         SRCDRAWS:ra,dec
   STKAPERPHOT    - Nothing needs to be updated
         ***Headers do not include standard coordinate information (RA*/DEC*/ROLL* keywords are miss

Master source data products
   BAYESBLKS       - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if pr
         ***Needs a header scrub***
   SRCAPERPHOT    - Update all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if pre
         ***Needs a header scrub***
   SRCPOLY3    - Update WCS, all RA_*/DEC_*/ROLL_* header keywords as well as RA_TARG/DEC_TARG (if p
         ***Needs a header scrub***
```

5.7 QA cases and procedure

QA should be triggered in the following three situations:

- *Density too high*: If the density of reference sources in the current stack is larger than 100 sources per square arcminute.
- *Shift too large*: If the resulting total shift of coordinates is larger than 1 arcsecond.
- *Too few matches*: If there are less than 3 matches, as long as any of the following conditions is also met:
  - There are no CSC sources with an off-axis angle smaller than 5 arcminutes.
  - There are no sources with more than 15 counts.
  - The total shift applied is more than 0.5 arcsec.

5.7.1 Procedure

The operator should be able to visualize in DS9 post stamp images of all CSC stack detections with *Gaia* matches, with a crosshair mark centered on the *Gaia* source positions. For crowded fields, only a maximum of 20 detections should be visualized. The operator should be able to blink between the image before and after the correction (with the crosshair mark fixed), to see if the translation did reduce the distance between CSC detection and *Gaia* sources for most of the detections. If the operator determines that this is not the case, the operator must select the matches that he/she considers most reliable and the WLS approach should be repeated with the selected matches. The process should allow for several iterations.

5.7.2 Flags

Sources in stacks that are processed in QA should be assigned an additional flag in the database. Other flags will be needed in the database. These are currently TBD.

6 Tests

We have thoroughly tested the approach proposed here using all of the $\sim$ 7200 stacks currently contained in version 2.0 of the CSC. We prepared the input tables as described in § 5.1, except for reducing the number of matches in very crowded stacks, and ran the algorithm following the steps described in the previous section. Here we show the results of these tests and evaluate them based on the following criteria, as per the scientific requirements:

- The distributions of RA and DEC shifts resulting from the transformation matrices should not be much larger than 1-2 arcseconds, as the relative error in position for each stack is smaller than that amount, and the larger between-stack error is of the order of 2.5 arcseconds.
- Overall, the coordinate differences of stack detections with the *Gaia* reference frame should be smaller or stay the same after the conversion.
- The between-stack differences in coordinates of stacks should be reduced, in particular for the outlying cases.
- The approach should be able to process most of the stacks automatically. No more than about 600 stacks should require manual processing.

6.1 Translation with *Gaia* only

We ran the WLS approach with all stacks that had at least 5 *Gaia* matches within 1" of the CSC detections. The results for the coordinate shifts are shown in Figure 3. The distribution is symmetric and has a mean close to -0.15". A few stacks have translations that are beyond the 2σ level, and those should be inspected manually.

6.2 Between-stack correction

A number of CSC2 stacks have significant differences in coordinates despite the fact that they overlap with each other. We should ensure that the between-stack differences in coordinates are significantly reduced. We identified a number of these stacks and checked their coordinates both before and after the correction. For the majority of cases, the between-stack differences were reduced. For a fraction of them no statistically significant improvement was achieved, but those are typically stacks that already agree between them at the level of the CSC2 accuracy. Only a very small fraction of these cases resulted in a larger between-stack coordinate difference. The latter are typically related with faulty transformations that give large rotations or scale factors, or with stacks having too few sources. These will all be sent to a manual quality accuracy control process.

In Fig. 4 we show a few examples of the between-stack correction. Not surprisingly the ones with larger differences prior to the correction tend to be in crowded *Chandra* fields that have many overlapping stacks.

7 Code

Below is a transcription of the code used for tests. This code as well as other dependencies to run the steps described in § 5.5 is located at:

```
\data\L3\rafael\stack_astrometry\
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from astropy.table import Table
from astropy.io import fits
from astropy import units as u
from astropy.coordinates import SkyCoord
```

```
from scipy.optimize import least_squares

# Read table of sources within 10 arcmins of each stack center
dat = Table.read('/Users/juan/L3/fine_astrometry/data/matches_new_v1.fits', format='fits')

# Turn into dataframe
df = dat.to_pandas()

# List of all stacks
stacks = np.unique(df['detect_stack_id'].values)

# Read tables of GAIA and ALLWISE matches for those sources within 3 arcsecs
gaia_matches = Table.read('/Users/juan/L3/fine_astrometry/data/gaia_new.fits', format='fits')
wise_matches = Table.read('/Users/juan/L3/fine_astrometry/data/wise_new.fits', format='fits')


# Select matches within 3" and convert to dataframes
df_gaia = gaia_matches.to_pandas()
df_gaia = df_gaia[df_gaia['Separation']<3.0*0.0002777777]
df_gaia = df_gaia[df_gaia['pmra']>-50]
df_gaia = df_gaia[df_gaia['pmra']<50]
df_gaia = df_gaia[df_gaia['pmdec']>-50]
df_gaia = df_gaia[df_gaia['pmdec']<50]

df_wise = wise_matches.to_pandas()


# Group by stack id
g = df.groupby(['detect_stack_id'])

# Placeholders
xi_shifts = []
eta_shifts = []
shifts = []
diffs_xi_0 = []
diffs_eta_0 = []
diffs_xi_1 = []
diffs_eta_1 = []

for st in stacks:
    print('Attempting ',st)

    # Select stacks with 5 matches or more
    if len(g.get_group(st)) >= 5:
        df_sub = g.get_group(st)
        df_merged = pd.merge(df_gaia, df_sub, on=['name'], how='inner')


        if (len(df_merged)>=5):

            # Tangent plane reference coordinates
            ra_stack_rad = df_merged[df_merged['detect_stack_id_x']==st]['ra_stack'].values*(np.pi/1
            dec_stack_rad = df_merged[df_merged['detect_stack_id_x']==st]['dec_stack'].values*(np.pi


            # The coordinates are input as arrays in units of arcseconds
```

```
ra_csc2 = df_merged[df_merged['detect_stack_id_x']==st]['ra_STACK_1'].values
dec_csc2 = df_merged[df_merged['detect_stack_id_x']==st]['dec_STACK_1'].values

ra_gaia = df_merged[df_merged['detect_stack_id_x']==st]['ra_cone'].values
dec_gaia = df_merged[df_merged['detect_stack_id_x']==st]['dec_cone'].values

c1 = SkyCoord(ra=ra_csc2*u.deg, dec=dec_csc2*u.deg, frame='icrs')


# Get standard coordinates
xi_csc2 = (np.cos(c1.dec.rad)*np.sin(c1.ra.rad - ra_stack_rad))/(np.sin(c1.dec.rad)*np.s
                                                                 + np.cos(c1.dec.rad)
                                                                 *np.cos(c1.ra.rad -

eta_csc2 = (np.sin(c1.dec.rad)*np.cos(dec_stack_rad)
            - np.cos(c1.dec.rad)*np.sin(dec_stack_rad)*np.cos(c1.ra.rad - ra_stack_rad))
            + np.cos(c1.dec.rad)*np.cos(dec_stack_rad)*np.cos(c1.ra.rad - ra_stack_rad))


xi_csc2 = 3600.*xi_csc2*(180./np.pi)
eta_csc2 = 3600.*eta_csc2*(180./np.pi)

# Same for Gaia
c2 = SkyCoord(ra=ra_gaia*u.deg, dec=dec_gaia*u.deg, frame='icrs')

# Get standard coordinates
xi_gaia = (np.cos(c2.dec.rad)*np.sin(c2.ra.rad - ra_stack_rad))/(np.sin(c2.dec.rad)*np.s
                                                                 + np.cos(c2.dec.rad)
                                                                 *np.cos(c1.ra.rad -

eta_gaia = (np.sin(c2.dec.rad)*np.cos(dec_stack_rad)
            - np.cos(c2.dec.rad)*np.sin(dec_stack_rad)*np.cos(c2.ra.rad - ra_stack_rad))
            + np.cos(c2.dec.rad)*np.cos(dec_stack_rad)*np.cos(c2.ra.rad - ra_stack_rad))

xi_gaia = 3600.*xi_gaia*(180./np.pi)
eta_gaia = 3600.*eta_gaia*(180./np.pi)

# Issue a warning if there are less than 5 matches
if (len(xi_csc2)<=5):
    print('WARNING: this stack has 5 or less matches: ', st)
    few_srcs.append(st)

# Get arrays of coordinates for source coordinates and target coordinates
positions_src = []
for pos in zip(xi_csc2, eta_csc2):
    positions_src.append(pos)
positions_src = list(positions_src)

positions_trgt = []
for pos in zip(xi_gaia, eta_gaia):
    positions_trgt.append(pos)
positions_trgt = list(positions_trgt)

def model(pars, X):
    return np.matmul(np.array([[1.0,0.0,pars[0]],[0.0,1.0,pars[1]],[0.0,0.0,1.0]]),[X[0]
```

```
def fun(pars, X, Y, X_err):
    return np.dot(1.0, np.sqrt((model(pars,X)[0]-Y[0])**2 +  (model(pars,X)[1]-Y[1])**2)

# initial guess
x0 = (0.0,0.0)

csc2_err = np.random.normal(0, 0.2, size=(2, len(xi_gaia)))

res = least_squares(fun, x0, loss='cauchy', bounds=(-10, 10), args=([xi_csc2,eta_csc2],[

print (" linear fit ",res.x)

xi_shifts.append(res.x[0])
eta_shifts.append(res.x[1])

new_xi = xi_csc2 + res.x[0]
new_eta = eta_csc2 + res.x[1]

diff_xi_0 = xi_gaia - xi_csc2
diff_eta_0 = eta_gaia - eta_csc2

diff_xi_1 = xi_gaia - new_xi
diff_eta_1 = eta_gaia - new_eta

diffs_xi_0.append(diff_xi_0)
diffs_eta_0.append(diff_eta_0)

diffs_xi_1.append(diff_xi_1)
diffs_eta_1.append(diff_eta_1)
```

List of Figures

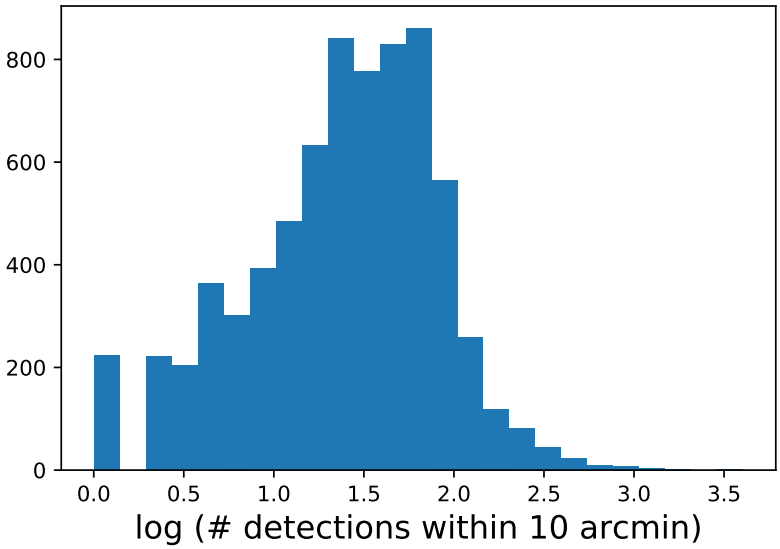

Figure 1: The distribution of the number of stack detections for all of the CSC 2.0 stacks

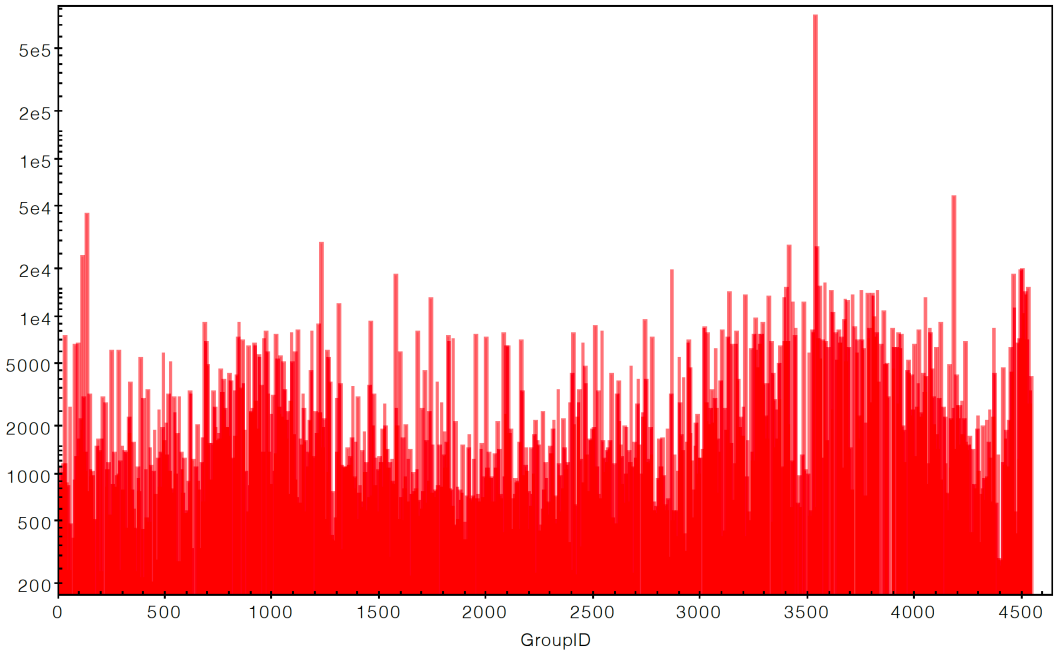

Figure 2: A histogram of *Gaia*/ALLWISE matches for a large subset of CSC stacks. In all of the cases, at least 200 matches are available for the transformation, within 10 arcminutes of the stack centroid.

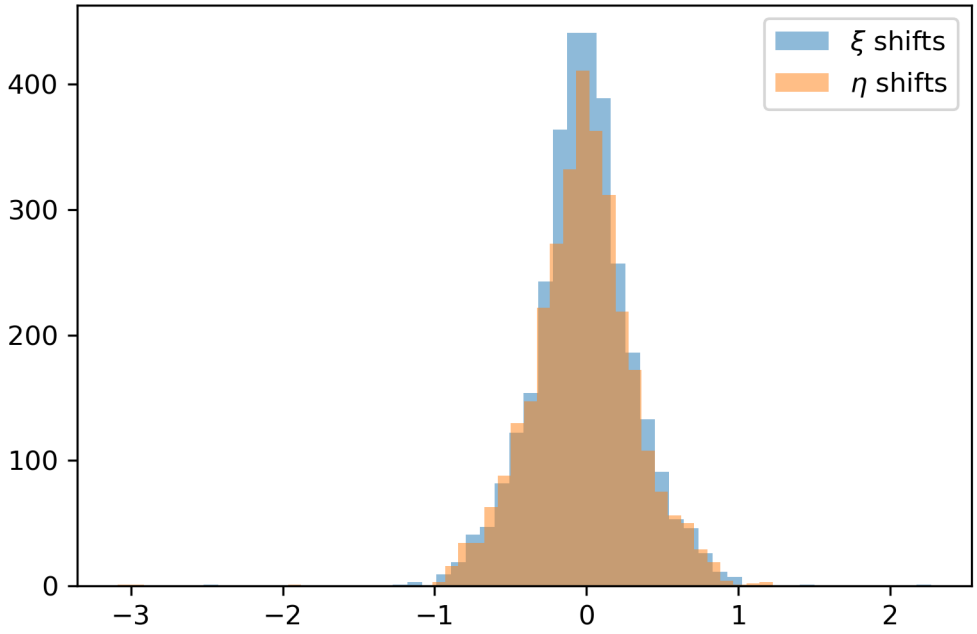

Figure 3: Corrections in ξ and η (in arcseconds) for a sub sample of CSC2 stacks for which the WLS translation approach was used.

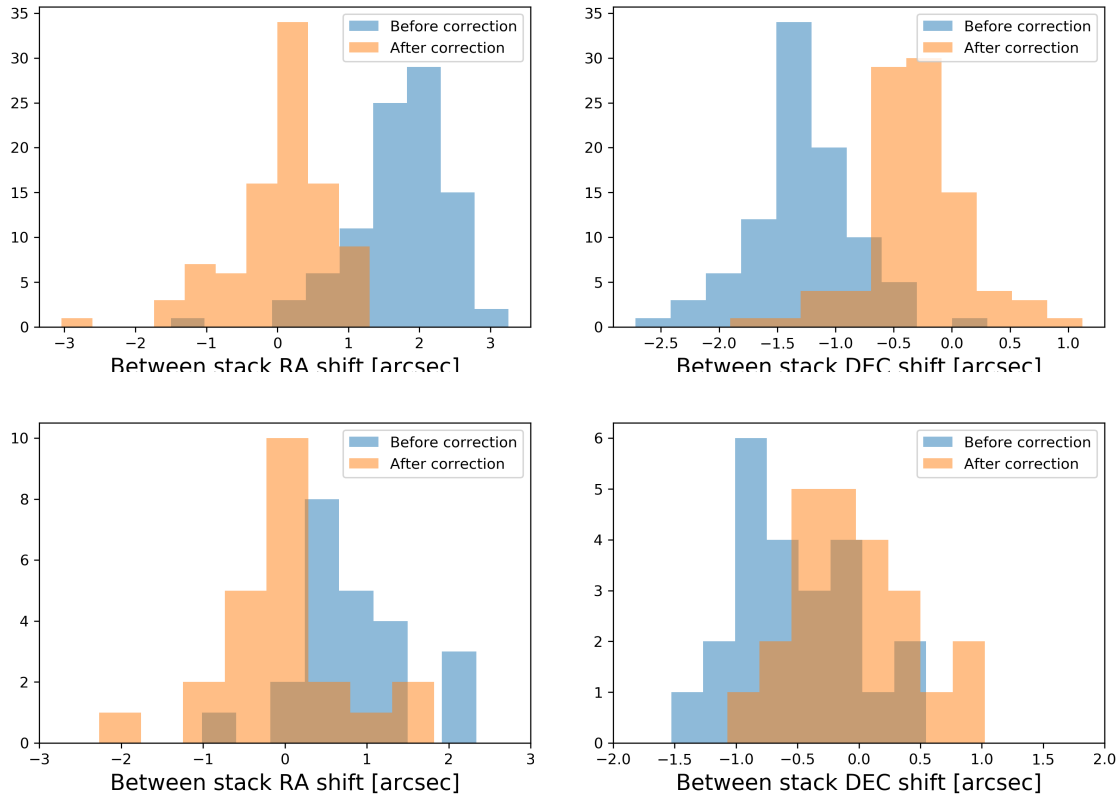

Figure 4: Correction in the coordinates of stacks that overlap, after the triangulation transformation has been applied. Shown are stacks acisfJ0658299m555730/acisfJ0658351m555811 (top), and stacks acisfJ0658299m555730/acisfJ0659003m555413 (bottom). They are all in the region of the cluster of galaxies ClG 0657-56.