Publications of the Astronomical Society of the Pacific, 132:054503 (10pp), 2020 May https://doi.org/10.1088/1538-3873/ab7ee8

An Algorithm for Coordinate Matching in World Coordinate Solutions

Joseph E. Postma and Denis Leahy
Dept. of Physics & Astronomy University of Calgary, 2500 University Dr NW Calgary, Alberta, T2N 1N4, Canada; jpostma@ucalgary.ca, joepostma@live.ca, leahy@ucalgary.ca
Received 2020 January 13; accepted 2020 March 11; published 2020 April 8

Abstract

Algorithms for point source extraction and catalog-to-image coordinate matching for world coordinate solutions are presented. In particular the coordinate matching algorithm is lightweight, simple to understand, easy to code, and solves orders of magnitude more quickly than existing solutions to this common astrometric problem.

Key words: Astronomical coordinate systems – Astrometry – Astronomy software

Online material: color figure

1. Background

The Flexible Image Transport System or FITS file has been developed as a digital image storage format within astronomy since 1981 (Wells et al. 1981). Significant periods of development of the FITS format have been ongoing (Greisen & Harten 1981; Grosbøl et al. 1988; Harten et al. 1988; Cotton et al. 1995). Standardization for the representation of image pixel coordinates as sky coordinates has also been developed for the FITS format (Calabretta & Greisen 2002; Greisen & Calabretta 2002; Greisen et al. 2006).

After the launch of UVIT (Kumar et al. 2012), CCDLAB (Postma & Leahy 2017) was developed into a data-reduction pipeline for the mission as described in the CCDLAB paper. The in-orbit nature of the ASTROSAT (Rao et al. 2009) satellite carrying UVIT is such that the UVIT telescopes are randomly rotationally oriented relative to the sky for any given observation. It then becomes necessary to form world coordinate solutions (WCSs, but sometimes World Coordinate System) so that objects in the final science images can be given accurate sky coordinates, and so that the centroid data can be de-rotated and axially aligned with sky coordinates. The WCS solver found at astrometry.net (Lang et al. 2010) was most-often found to not work for solving WCS for the far-UV UVIT images, likely because the detection wavelengths of UVIT FUV image data are sufficiently incommensurate with the catalogs used by that package. When we contacted relevant developers for discussion on the methods for performing WCS solutions, one response was: "Only two or three people in the world truly understand this process." Thus, we have sought to make the process more clear in the current manuscript, and we provide the algorithm and code for a reasonable and simple automated solution.

The unique detector system of UVIT required extensive interactive development during the engineering phases of the system build (Hutchings et al. 2007; Postma et al. 2011). It was necessary to test the UVIT detector system's field-programmable-gate-array implementation of point-source-extraction (PSE) centroiding against a software implementation of the same algorithm. Thus, it was necessary to implement the PSE algorithm into CCDLAB so as to operate on integration-mode images from UVIT. The PSE algorithm will be described at length given that coordinate extraction from images is the one of the inputs to a WCS solution; it is simple, fast, reliable, and parallelizable (although it was not parallelized in the UVIT FPGA code), and of course functions on any 2D image data array.

The algorithms are parallelizable and thus scale with increasing thread count, are fast, and will be discussed at length. They are currently implemented in CCDLAB and as of writing are being coded into a stand-alone dynamic-link-library for Microsoft.Net, which is utilizable by Python. The library along with all Visual C++ code should eventually appear on the NASA FITS Support Office (https://fits.gsfc.nasa.gov/) under the FITS I/O libraries link as "JPFits." Contact the author for code samples or questions about implementation, etc.

2. Point Source Extraction

The algorithm for PSE from full-frame UVIT image data has been discussed previously (Postma et al. 2011), but we shall go over it again in more detail and with some improvements which are possible in software code rather than in an FPGA implementation, such as parallelization. There are many other existing implementations of source extraction, for example *sextractor* (https://www.astromatic.net/software/sextractor) or IRAF (http://ast.noao.edu/data/software), however we have continued with the algorithm as developed for UVIT given its simplicity and its speed and because it already existed in CCDLAB to build off of to then use for the WCS algorithm.

We would like to scan an image for point sources, and return the centroids and other metadata of said sources in a list. We begin with two parameters of the source separation radius (SSR) and the centroiding kernel radius (CKR), where the minimum of the SSR is the CKR (SSR $\geqslant$ CKR). We thus begin scanning the image at pixel index [$x = y =$ SSR], on a pixel-by-pixel basis and thus in a nested for loop across each dimension which can thus be parallelized. Although we call the CKR a radius, the kernel is in fact a square and so a CKR equal to 1 gives a 3×3 pixel sample, 2 gives a 5×5 pixel sample, etc. A CKR equal to 0 gives a single pixel, and although a single pixel may be tested for threshold value, it cannot be centroided to sub-pixel accuracy. The CKR should be chosen so that it covers most of the point-spread-function (PSF) of the sources. Undersampled PSF's which require sampling with only a 3×3 kernel result in a systematic bias of the centroid toward the center pixel which is called "fixed-pattern-noise" as discussed in Hutchings et al. (2007), and so keep in mind whether you are centroiding such sources and also require sub-pixel centroid accuracy to better than one-third of a pixel, in which case you will need to calibrate and correct said bias. Otherwise, PSF's which can be sampled with a 5×5 centroid kernel (or larger) have negligible centroid bias. By "centroiding" or "centroid" we mean a weighted-average coordinate position given the weighting of pixel coordinates by the pixel values.

We require a pixel value threshold (PVT) to test whether or not a pixel is a possible source. We also add a total kernel value threshold (KVT) to test whether the PSF sampled by the centroid kernel is a possible source. These thresholds can be in either pixel count native to the image, or, signal-to-noise which allows for more generality and independence from the image bit range, etc.

We thus also require the background to be subtracted either from the image as a whole, or locally from each centroid kernel. If an image is sufficiently background-subtracted then the background is zero and does not require subtraction from any pixel values; however if the background must be determined locally for each centroid kernel, due to gradient in the background or some existing offset from zero, then the local background can be determined as, for example, the second-minimum of the four corners of a square formed by the SSR. That is, gather the pixel values at [$x -$ SSR, y], [$x +$ SSR, y], [x, $y -$ SSR], [x, $y +$ SSR], and determine the value of the second-minimum as the local background. The idea here is that the minimum pixel value will be systematically low, the maximum pixel value will be high due to an adjacent source or noise, the second-highest pixel value will likewise be high, and the remaining second-minimum should have the highest probability of representing the local background. It is of course helpful here to have the SSR extend beyond the PSF. Of course, variations upon this method of local background determination may be utilized by any developer, such as using the median of an annulus around the source peak at some distance, or the median or minimum of the SSR kernel itself, etc. We use the corner method simply because it is trivial and was found to be accurate to within 1% of a more robust determination, which is sufficient for accurate weighted-mean centroiding. With background determined or determinable, PVT and KVT thresholds set, and SSR and CKR set, then one simply scans through the image checking for locations which satisfy the threshold criteria as will be discussed below.

It is helpful to create an integer-base source-index map of identical dimensions to the image being analyzed, so that locations in the image for which a source is found can be given the integer index value for where the source is located in the source list. That is, as sources are found, their metadata are added to storage in an array list at some increasing index i therein. The source index map should be initialized to -1, and then when a source is found at some pixel location [x, y] in the source image, which is then added to the source metadata list at index i, all pixels within the (circular radius) CKR of [x, y] in the source index map should be set to value i. This way, when some arbitrary location [a, b] in the source image must be checked to determine if a source had been located there by the PSE, then location [a, b] in the index map will have value -1 if no source was located there, and otherwise will have some value $i \geqslant 0$ (for zero-based indexing code) where i can then be used to index into the source list returned by the PSE in order to gather the metadata (such as the centroid) for that source.

This algorithm functions well for images which do not suffer from pixel saturation, such as the binned centroid-count images from UVIT. However, many CCD imagers on telescopes are direct-exposure cameras and bright sources in the detection field can cause saturation of the well-depth of the pixels. For example, the Baker-Nunn telescope at the University of Calgary's Rothney Astrophysical Observatory (Milone & Clark 1996) has a $4\overset{\circ}{.}4 \times 4\overset{\circ}{.}4$ field of view imaged onto a 4096×4096 pixel CCD and is used for searching for transient objects such as asteroids and comets (Cardinal et al. 2008), and operating with its normal sixty-second integration time leaves thousands of the brightest stars saturating the pixel wells just below the maximum 2^{16} bit ADC value. In this case the saturated brightest objects no longer form peaked profiles, but form island plateaus (see Figure 1), and thus to properly detect these plateaus and estimate their centroids requires a different approach. Thus, with the expectation that the saturation levels of the image are known, one can utilize a saturation threshold value (STV) in combination with a recursive island-mapping algorithm to map the saturated sources.

When we map the saturation islands we wish to record the maximum and minimum axes coordinate values for each one so that a kernel may then be formed around the island for centroiding, and it is helpful to either set the saturation level or extend those axes limits to include some of the wings of the island for centroiding. The island-mapping algorithm

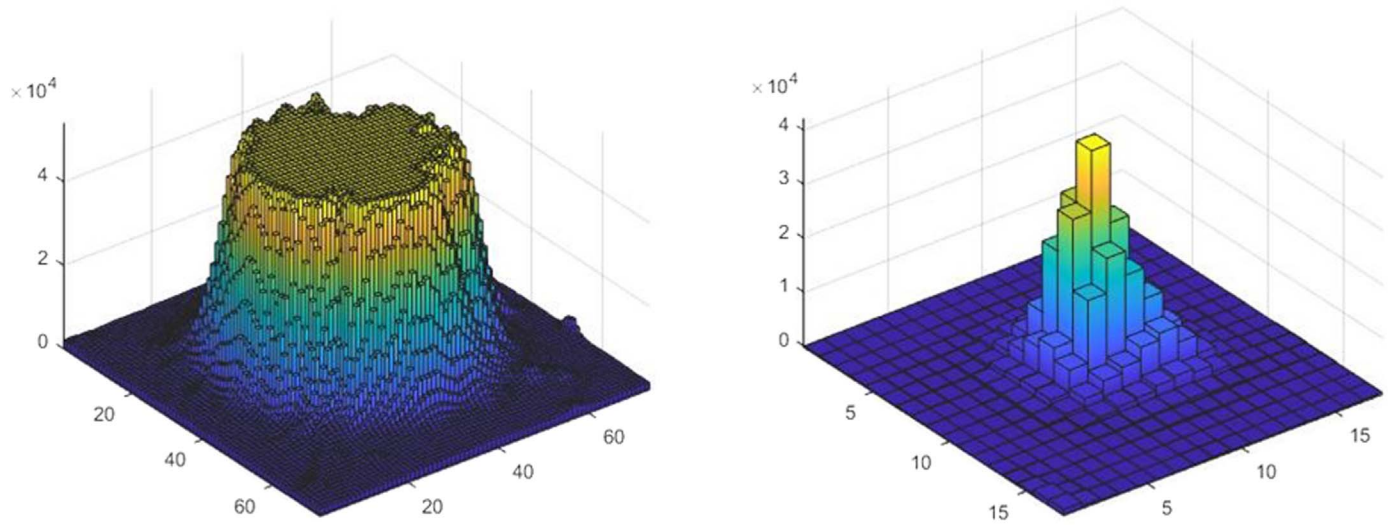

Figure 1. Saturated sources form islands (left), rather than unsaturated sources which form peaks (right). These require different algorithms to detect and accurately centroid.
(A color version of this figure is available in the online journal.)

implemented in highly readable Visual C++ code is presented is as follows:

```
void MAPSATURATIONISLAND (int X, int Y, int sourceindex, int &xmin,
  int &xmax, int &ymin, int &ymax) {

      SOURCE_INDEX_MAP[X, Y] = sourceindex;
      if (X < xmin)
            xmin = X;
      if (X > xmax)
            xmax = X;
      if (Y < ymin)
            ymin = Y;
      if (Y > ymax)
            ymax = Y;
      for (int x = X − 1; x < = X + 1; x++)
            for (int y = Y − 1; y < = Y + 1; y++)
                  if (SAFETOMAPSATURATION(x, y))
                        MAPSATURATIONISLAND(x, y, sourceindex,
  xmin, xmax, ymin, ymax);}

bool SAFETOMAPSATURATION (int x, int y) {
      return (x > = 0) && (x < IMAGEWIDTH) && (y > = 0) &&
  (y < IMAGEHEIGHT) && (IMAGE[x, y] > PIX_SAT) && (SOUR-
  CE_INDEX_MAP[x, y]==−1);}
```

The MAPSATURATIONISLAND function would be called for any pixel which exceeds the STV while scanning through all pixels in an image.

Summary of the PSE Algorithm:

1. Create an integer-base source-index-map of equal dimension to the source image to be scanned for point sources. Initialize this map to -1 for all array values, where -1 indicates no source.
2. If the image may have saturated sources, then *first* scan the entire image checking for pixels which are above the pixel STV.
    1. If a saturated pixel is detected, use recursive island-mapping to find all adjacent saturated pixels in the vicinity.
    2. Set the source-index-map of the currently found saturated pixels to the current source index during mapping.
    3. Keep track of the minimum and maximum row and column indices mapped out, and use these after the mapping is finished to form a square kernel around the saturated pixel group to then centroid the saturated source. It is likely good to extend the range of the minimum and maximum row and column indices beyond the saturation threshold, such that the wings of the saturated source are included in the centroid calculation.
    4. The scan of the image can be parallelized with appropriate use of for example a *critical* directive (as in the OpenMP nomenclature) and in order to assign centroid metadata to lists which exist outside of the parallelized loop.
        1. For example, for the i'th source found, the new i'th index in the source metadata lists might contain the x-centroid, the y-centroid, the central pixel amplitude, the total kernel amplitude, the local background estimate, etc., and, all pixels in the source index map which are currently mapped by the island-finding algorithm in the source image should be set to i.

5. Now proceed with the point-source extraction algorithm.

3. Choose an integer-valued SSR, an integer-valued CKR, a PVT, a KVT, and a Boolean to indicate whether local background determination is required.
    1. One may also utilize maximum-value thresholds so that sources which are above a specified brightness are excluded from the scan results.
4. Begin scanning the source image at $[x = y = \mathrm{SSR}]$ in a nested for loop, which can be parallelized.
5. If required, determine the local background at each pixel.
6. Check if the $[x, y]$ pixel amplitude (above background) meets the pixel threshold; if not then continue to the next iteration in the loop, i.e., the next pixel.
7. Check if any other pixels within the SSR circle about $[x, y]$ are greater in value than the pixel at $[x, y]$; if any are, then continue to the next iteration in the loop, i.e., the next pixel. This way only the peak of the profile, or the brightest source within the source separation circle, will be a potential source.
    1. At the same, one may also check whether any sources have already been detected at that location within half of the SSR, by checking if the source-index-map has any non -1 values within said circle; if there are, then continue to the next iteration.
8. Check if the total centroiding kernel (either square or circular can be used, but circular typically matches a PSF) sum (background subtracted) is above the total KVT; if not then continue to the next iteration in the loop, i.e., the next pixel.
9. If we have made it here then we have an i'th local maximum source in the image within the SSR, and so centroid the kernel centered on $[x, y]$.
    1. Here, the centroid calculation *must* be upon a square centroiding kernel so that all rows and columns are equally sampled among each other.
    2. Use here for example a *critical* directive (as in the OpenMP nomenclature) in order to assign centroid metadata to lists which exist outside of the parallelized loop.
        1. For example, for the i'th source found, the i'th index in the source metadata lists might contain the x-centroid, the y-centroid, the pixel amplitude, the total kernel amplitude, the local background estimate, etc.
    3. Assign the integer i to all pixels in the source-index map within the circular kernel radius of pixel $[x, y]$.
    4. Optionally, one might wish to save the centroid kernel of the current point source as an independent file.
10. Finished.

3. World Coordinate Solutions

The history of the FITS standards for WCSs (or system, but generally, WCS) can be found in the papers referenced in the introduction. We will remain consistent with that standard here. What we shall discuss however is an algorithm for determining the WCS solution when one has two lists of points which should correspond but which differ due to a respective field transformation, item entry position (sorting) in the list, and a partially incommensurate population of items in the list. That is, when performing a WCS, one will have a list of coordinates from the image in pixel coordinates, and one will have a list of coordinates from some catalog in sky or world coordinates (for example R.A. and decl.). Typically, brightness or magnitude is part of each entry's metadata in such lists. The image pixel coordinates would likely be extracted from PSE of sources in the image, while the catalog could be from, say, the Gaia Data Release 2 (Brown et al. 2018).

The perfect scenario would be where the n image coordinate list entries match on a one-to-one basis the n entries in the catalog, when, say, both lists are sorted by brightness. In this case the list of image coordinates and the catalog list can be directly fitted for the transformation parameters in a WCS routine. Of course, this never happens. The hopeless scenario is where none of the image sources match to anything in the catalog, and this can happen often.

Practical considerations are that the list of image point sources and the catalog list will generally not be sorted in a matching entry-wise basis, even when sorted by brightness. At the very least, the detection wavelengths for the image, and the catalog, are likely going to be different, and different wavelengths see different and dissimilar objects on the sky, different structures in the same objects, and different brightnesses of the same objects, and the extracted entries from the catalog may not correspond exactly to the area of sky captured in an image. For example, one might expect extreme or total non-correspondence between sources in a visual image and a radio source catalog. And so, the first two items which generate non-identity between an image source list and a catalog list are dissimilar sorting of the entries, even when sorted by brightness, and, for n brightest sources in an image, only some value less than n may actually correspond with the relevant brightest entries in the catalog. This was particularly a problem for UVIT and its far-UV channel, which sees things at 120 nm that most existing catalogs do not. Along with detector characteristics there is obviously also an astrophysical dependence upon the degree to which different wavelengths see different brightnesses, thus having an effect upon which sources get sorted to the top of the list when sorted by brightness. It is of course the brightest objects which are most useful for this particular problem of determining corresponding sources, and the degree to which a wavelength differential between a catalog and image affects the sorting is of course a very general concern.

This being said, what we fundamentally require is some degree of correspondence between the entries of the image source list, and the catalog. Thus, similarity between the wavelengths detected in the image and the catalog is a first criteria. Second, the central position and area of the sky in the image should be known to some degree of accuracy, such that only the relevant entries in the catalog can be extracted and used for comparison to the image sources. The goal here is that for the first n brightest sources in the image, there should be some correspondence to the first n brightest sources in the catalog being used. The greater the correspondence, the better.

The intent here is an algorithm which can quickly determine a WCS solution for modern scientific instruments. As such, although the WCS standard uses a general transformation matrix which allows for skew, rotation, scale, and reflection (parity), modern scientific detectors are likely designed so that the pixels subtend a rectangular area on the sky (no or little skew and distortion), and the transformation matrix is mostly only a function of scale, pointing position, pointing rotation, and parity. Scale is typically well known at design time, and can be determined precisely from the first few images after first light. These simplify the parameter space greatly, but the algorithm which we will discuss will still allow for some degree of skew on the image field. Perhaps a detector could be designed with image transformation parameters which amount to some vast encryption problem, but typically in science our intent is to design detectors which correspond on a one-to-one basis to the spatial dimensions which they image, aside from the practical considerations of field projection and field transformation to sky coordinates, etc. Nonlinear distortions due to engineering limitations, and skew, are typically at the unit pixel scale.

The core step of solving of a WCS is a least-squares fit, via a transformation matrix, utilizing *intermediate* catalog coordinates, of a corresponding set of image pixel positions and catalog positions. The purpose of the intermediate coordinates are to project spherical sky catalog coordinates onto a plane, which can then be scaled, shifted, rotated and inverted via the CD matrix to the planar image pixel coordinates. The task, then, is to determine which coordinates in the image correspond with which coordinates in the catalog so that the least squares routine can then function on those corresponding values.

The WCS process is that the catalog positions are first transformed into intermediate coordinates, and then the image pixels are fit via a 2D transformation matrix (the CD matrix) to those intermediate coordinates. This means that if there is no or little skew, then the rectilinear relations among intermediate coordinates should be correspondent with the same relations among matching image pixel coordinates, aside from rotation, scale, and reference point. That is, the only thing which needs to be solved for is rotation, scale, and reference position, and so this requires only three points for an initial solution. Let us refer to the WCS transformation matrix from the FITS WCS Paper I, in Equation (1) in the current manuscript.

$$x_i = \sum_{j=1}^{N} \mathrm{CD}_{ij}(p_j - r_j). \tag{1}$$

We shall discuss Equation (1) in detail just so that we are clear on what the WCS transformation process is and how we are going to use it to build an automatic WCS solver. First the term x_i is an *intermediate coordinate* in units of degree, which in our case of wishing to solve the relevant terms of the equation, are computed directly from the catalog data given a chosen coordinate projection, such as for example the common Gnomic or Tangent-Plane projection. The Tangent-Plane intermediate coordinates for right-ascension (α) and decl. (δ) are computed as in Equation (2) (where the intermediate coordinate terms x_1 and x_2 from Equation (1) are now written as X, Y). For completeness, since we will eventually wish to transform intermediate coordinates computed from image pixels via a solved WCS CD matrix to sky coordinates, then the inverse of Equation (2) is presented in Equation (3).

$$\begin{aligned} X &= \frac{\cos(\delta)\sin(\alpha - \alpha_0)}{\cos(\delta_0)\cos(\delta)\cos(\alpha - \alpha_0) + \sin(\delta_0)\sin(\delta)} \\ Y &= \frac{\cos(\delta_0)\sin(\delta) - \cos(\delta)\sin(\delta_0)\cos(\alpha - \alpha_0)}{\cos(\delta_0)\cos(\delta)\cos(\alpha - \alpha_0) + \sin(\delta_0)\sin(\delta)} \end{aligned} \tag{2}$$

$$\begin{aligned} \alpha &= \alpha_0 + \tan^{-1}\left(\frac{X}{\cos(\delta_0) - Y\sin(\delta_0)}\right) \\ \delta &= \frac{\sin^{-1}(\sin(\delta_0) + Y\cos(\delta_0))}{\sqrt{1 + X^2 + Y^2}}. \end{aligned} \tag{3}$$

The reference positions α_0 and δ_0 are the CRVAL1 and CRVAL2 values, respectively, as per the FITS WCS keyword standard. Typically these would be thought to be the sky coordinates at center of the image, but since this is not known before solving the WCS, then they can simply be taken as the means of the R.A. and decl. coordinates extracted from the catalog. That is, for n sky coordinates extracted from a catalog which correspond to n pixel positions from sources extracted from an image, the coordinate reference positions can be the means of the coordinates.

In Equation (1), the (p_j–r_j) term is the difference between the source image pixel coordinates (p) and the reference pixel coordinates (r). That is, r_1 and r_2 are the coordinate reference pixel values CRPIX1 and CRPIX2, respectively. CRPIX1 and CRPIX2 should correspond with the CRVAL1 and CRVAL2, however, the actual pixel location is not known before the WCS is solved; that is, the CRVAL are parameters which must be solved, and so they can be given an initial guess of the means of the source image pixel [x, y] coordinates and then bounded by the extremities of such in the least squares fit.

The CRPIX1 and CRPIX2 are two parameters which must be solved, but the CD matrix (in units of degrees per pixel)

contains additional parameters which must also be solved for. Writing Equation (1) out, we have Equation (4) where again $x_1 = X$ and $x_2 = Y$, and x and y are used here to denote the image pixel coordinates of a given source corresponding to the intermediate coordinates X and Y. The inverse is also provided in Equation (5) so that intermediate coordinates may be converted to pixel values.

$$\begin{aligned} X &= \mathrm{CD}_{1,1}(x - \mathrm{CRPIX1}) + \mathrm{CD}_{1,2}(y - \mathrm{CRPIX2}) \\ Y &= \mathrm{CD}_{2,1}(x - \mathrm{CRPIX1}) + \mathrm{CD}_{2,2}(y - \mathrm{CRPIX2}) \end{aligned} \tag{4}$$

$$\begin{aligned} x &= \mathrm{CD}^{-1}_{1,1} \cdot X + \mathrm{CD}^{-1}_{1,2} \cdot Y + \mathrm{CRPIX1} \\ y &= \mathrm{CD}^{-1}_{2,1} \cdot X + \mathrm{CD}^{-1}_{2,2} \cdot Y + \mathrm{CRPIX2} \\ &\text{where} \\ \mathrm{CD}^{-1}_{1,1} &= \det \cdot \mathrm{CD}_{2,2} \\ \mathrm{CD}^{-1}_{2,1} &= -\det \cdot \mathrm{CD}_{2,1} \\ \mathrm{CD}^{-1}_{1,2} &= -\det \cdot \mathrm{CD}_{1,2} \\ \mathrm{CD}^{-1}_{2,2} &= \det \cdot \mathrm{CD}_{1,1} \\ &\text{and} \\ \det &= \frac{1}{\mathrm{CD}_{1,1} \cdot \mathrm{CD}_{2,2} - \mathrm{CD}_{1,2} \cdot \mathrm{CD}_{2,1}}. \end{aligned} \tag{5}$$

The CD matrix transformation of Equation (1) will allow for transformation between any 2D planar coordinate systems. Such coordinate systems transformed through this equation may be arbitrarily inverted (parity), and they may be arbitrarily scaled, rotated, shifted and skewed, along one or both dimensions relative to each other. However, for scientific instrumentation the parity of the image dimensions with respect to sky coordinates are typically known and these can be trivially corrected to correspond to the directionality of sky coordinates; that is, for example, if an image were derotated, decl. should likely increase "upwards" in an image and R.A. increase "leftwards" in the image, at least when imaging in the Northern Hemisphere. In cases where the parity cannot be pre-aligned, it would be trivial to code for the algorithm to test all four parity relation possibilities, but the implementation of this algorithm in CCDLAB assumes that the parity is known and corrected (we may update the code to allow for parity variability in the future). At the same time, image skew is likely only at the unit pixel scale, if it is in fact not negligible across the entire field.

Thus, a solution to Equation (4) can be reduced to first solving a solution for only rotation, scale, and reference position, which thus requires only three corresponding points between the catalog coordinates (converted to intermediate coordinates) and image source pixel coordinates. Thus, we must fit triangles formed by the intermediate catalog coordinates to triangles formed by the image source coordinates to each other via only a scale (S), rotation (ϕ), and reference position (CRPIX1,2) transformation, as in Equation (6). Using a least squares fit allows for tolerance due to skew and distortion, and this tolerance will be carried through the automatic WCS solution to the final solution where the CD matrix can then capture it (if it is systematic). We will eventually need to invert Equation (6) when transforming intermediate catalog coordinates to pixel locations when testing if the initial solution is valid, as in Equation (7).

$$\begin{aligned} X &= S[\cos(\phi)(x - \mathrm{CRPIX1}) - \sin(\phi)(y - \mathrm{CRPIX2})] \\ Y &= S[\sin(\phi)(x - \mathrm{CRPIX1}) + \cos(\phi)(y - \mathrm{CRPIX2})] \end{aligned} \tag{6}$$

$$\begin{aligned} x &= \mathrm{CRPIX1} + \frac{X\cos(\phi) + Y\sin(\phi)}{S} \\ y &= \mathrm{CRPIX2} + \frac{-X\sin(\phi) + Y\cos(\phi)}{S}. \end{aligned} \tag{7}$$

Thus, for a list of the brightest n sources in an image, and the brightest n entries in the catalog, we create a table of all unique triangles which can be formed from the respective n coordinates. Practical values of n will be discussed later, as the number of unique triangles which can be formed from n points is an n-choose-r problem with r equal to 3. It is thus helpful to create a triangle *class* in software which holds the points and other metadata about the triangles' properties.

The crucial function here is to organize the points of each triangle so that independent of scale, rotation, and reference position, the three points of an intermediate coordinate triangle will translate directly to the three points of an image coordinate triangle if the points are in fact correspondent. Thus, the three side lengths of the triangle should be computed, and the points should then be sorted based upon a simple analysis of the side lengths. That is, the first point of the triangle should be made such that it is the vertex of the shortest two side lengths, the second point such that it is the other point of the shortest side length, and the remaining point is determined. This way, when a match is found between an intermediate coordinate triangle and an image coordinate triangle, the respective points can then be used directly for a linear least squares solution with respect to each other. One should also calculate the triangle vertex angles via the Cosine Law for the triangle-point vertices as they have just been sorted as these will be used to test whether or not two triangles are correspondent.

After the triangles have been created and their point-order formatted as such, then the task is to search for similar triangles between the intermediate catalog coordinates and the image pixel source coordinates. That is, we do not wish to least-squares fit every possible intermediate coordinate triangle to every possible PSE triangle, given the computational expense of performing least-squares. We wish to only fit triangles which look like they should be a match given a much more computationally inexpensive preselection. The vertex angles of the triangles are useful here, and have already been sorted by the previous scheme of sorting the point vector based on side lengths, and so one can check for similar vertex angles to within some tolerance. That is, vertex angle 1 of the

intermediate coordinate triangle should match within some tolerance to vertex angle 1 of the PSE triangle, and so on to vertex angles 2 and 3. This is a fast check which allows for an extremely large number of triangles to be compared very quickly. However, when the number of triangles is very large then there is a growing likelihood of triangles having similar vertices but whose points are not actually corresponding coordinates.

Thus if all of the vertices survive the tolerance test, a secondary check can be performed where the side lengths of the image coordinate triangles are compared to the side lengths of the intermediate coordinate triangles, given an upper and lower bound of the transformation scale. Due to the way we sorted the triangle-points based on side-lengths, the side-lengths also correspond on a one-to-one basis between intermediate coordinates and PSE coordinates just as the triangle vertices did. That is, given an upper and lower bound for the scale, the intermediate coordinate triangle side lengths should fall within the upper and lower bounds of the image coordinate triangle side lengths if the triangles comprise truly correspondent coordinates. To allow some tolerance even when the lower and upper scale bounds are set equal to the initial scale estimate, the width of the centroid kernel used to determine the image source coordinate in the PSE can be respectively subtracted from and added to the lower and upper bounds of the image triangle side lengths. This allows for a degree of distortion and/or skew to be present in the image. Or instead of using the centroid kernel width from the PSE, one could specify the tolerance to use here.

At this point the probability of false-positive matches should be greatly reduced, while the probability for true-positive matches should be relatively increased. And so if an intermediate coordinate and image coordinate triangle pairing has survived these checks, then a least-squares solution can now be solved for the rotation, scale, and reference point to transform the given image triangle coordinates to the given intermediate triangle coordinates. If this is a truly correspondent triplet of coordinates, then this solution should also successfully transform, by inverse, other catalog intermediate points to image coordinate locations, while if the triangles were not actually correspondent coordinates but only similar triangles then their false-positive transformation solution will not result in successful matches between points for the remaining coordinates. Parity inversions do not affect the vertex angles or the side-lengths, but here is where image coordinates can be inverted to test for the four possible variations of parity relations.

Thus, for the n coordinates of image sources and catalog entries, one can transform the intermediate catalog coordinates to image pixel coordinates via an inverse transform utilizing the parameters of the initial solution just found. This is where it is useful to have the source index map from the PSE, so that the computed image coordinates from the catalog intermediate coordinates can be directly checked to see if they fall upon sources detected in the image, and if they do, to gather the source indexes in the image point source metadata list.

For the n coordinates from the catalog, at this point there are already three which will fall on image sources given the initial solution. If this is a false-positive solution, however, then the success rate for transforming additional catalog entries, via their intermediate coordinates, and via the inverse transform of the initial solution, to the previously determined image source locations, will be quite poor. For a true-positive solution, then catalog coordinates will transform through to image source locations with a high success rate. This establishes stopping criteria for the algorithm in that if, say, an additional three points are successfully transformed, or, twenty-five percent of the n coordinates are successfully transformed, then one likely has a good true-positive solution. These stopping criteria can be modified as needed. We do not assume that all n catalog entries must or will fall onto n image sources, given that structural non-correspondence between the image source detections and the catalog may exist due to differences in detection wavelengths, and extracted field-overlap, etc.

If the stopping criteria are met, then the matching subset from n of corresponding image coordinates and catalog intermediate coordinates can be used to determine the general transformation through the full CD matrix and provide the initial WCS solution. However, the solution can likely be improved because the initial number of brightest sources and entries n was likely small (the magnitude order of ten), while the number of sources in the image may be in the hundreds and the number of catalog entries for the field area may be in the thousands. Thus, if the field allows for it, one may reduce the thresholds in the PSE so that an order of one-hundred brightest sources are determined, and then two-or-three hundred brightest source from the catalog for that region should be scanned to check for entries which fall onto those image sources given the inverse of the WCS solution. This should populate most of the field with a scattered sampling of matching catalog coordinates to image source coordinates, and then the WCS can be resolved to give a final solution. Even with a solution with $\sim 10^2$ points scattered across the field, one may wish to explore further possible refinement with options provided by other existing packages.

Summary of the Automatic WCS Algorithm:

1. Perform PSE on an image to gather the pixel coordinate centroids of the brightest n sources.
    1. The PSE class should contain an integer image map, of equal dimension to the source image, denoting the list position index of the corresponding centroid for that location. For example, for a centroid at location $[x,\ y]$ in the image (where decimal-valued $[x,\ y]$ centroids are rounded to integer indices), all pixels of the integer map within a distance of the kernel radius

to that location should be set equal to the integer index value at which the centroid appears in the PSE source list. In this way, an arbitrary coordinate $[a, b]$ can be directly indexed into the integer map to gather the list index of the source at that location in the PSE source list. The integer map should be initialized to -1 so as to avoid conflict with zero-based indexing code, and this value also forms a redundant Boolean map given that all locations in the integer map with value -1 indicate that no source is located there.

2. Ensure that you are aware of the parity of the source image, and remove or account for coordinate inversions as necessary. For example, if the image had no field rotation, then "left" should be increasing R.A. and "up" should be increasing decl.; field rotation can of course change those relations, but, this is only a problem of field rotation. If beam splitters or other optical train effects invert the coordinate relations, then this must be known and accounted for, although it would be trivial to code for all possibilities in the case where the user is not aware of such meta-properties of the image.
3. The PSE routine can be iterated upon a changing threshold value until n sources are found, with some iteration limit in cases where n sources are not available.

2. Gather the brightest n sources from a catalog.
   1. The catalog sources should be extracted from a region correspondent with the source image. Ideally the sources extracted from the catalog correspond directly to the area in the image used for PSE. In practice, however, the pointing of the center of the image is typically not precisely what is reported in the image header—hopefully the difference is not too large.
   2. Beware of NaN's in the catalog magnitude values as these will typically sort to the start of the list when sorting the catalog list by brightness. Exclude these.
   3. The catalog and the image should have sufficiently correspondent wavelengths used for source detection, given that different wavelengths see different things in the sky.
3. Form intermediate coordinates out of the catalog list gathered in point 2.
   1. The reference values required in computing intermediate coordinates, i.e., the CRVALn values, can be the means of the catalog coordinates. For example, CRVAL1 is the mean of the R.A. values, and CRVAL2 is the mean of the decl. values.
4. Form all the unique triangles given the n sources for the image coordinates and the catalog intermediate coordinates.
   1. A triangle class is helpful which contains:
      1. Three points, where the points have x and y position values
      2. The vertex angles of the three points.
      3. Distance magnitudes between the three points.
   2. The points of each triangle should be sorted such that two corresponding triangles will have an identical point-order, so that the triangle coordinates match to each other when utilized in a least squares transformation.
      1. The common vertex of the shortest two side-lengths can be the first point; the other point of the shortest side length can be the second point; and the third point is then determined.
      2. The distance magnitudes are that of the first vertex to the second, the first to the third, and the second to the third.
5. In a nested "for loop" (which can be parallelized), compare PSE triangles to intermediate coordinate triangles.
   1. If the corresponding vertex angles exceed a matching tolerance, say $\pm 0^\circ.5$, then continue the for loop to the next triangle pair comparison.
   2. If the corresponding side lengths of the intermediate coordinate triangles are outside of the bounds of the PSE triangle side lengths, given application of the scale range (upper and lower bounds), and given the tolerance of the PSE kernel width, then continue the for loop to the next triangle pair comparison.
6. At this point we have a potential valid pairing of coordinate triangles between the image PSE coordinates and the catalog intermediate coordinates, given that the three corresponding triangle vertices are within tolerance, and the side lengths are within tolerance given the scale plus kernel width range. However, it needs to be confirmed that these are actually correspondent coordinates given that it is possible to have a false-positive match of two similar triangles.
   1. Perform a least squares solution of the PSE triangle coordinates to intermediate coordinates, via a scale, rotation, and reference position transformation.
      1. For most instruments, the scale should already be known and can be constrained in the least squares fit to, say, plus or minus five percent, although this range can be larger if needed.
      2. For most instruments, the field rotation may already be known and can likely be constrained to a few degrees, although this can range from -180° to $+180^\circ$ with an initial guess of 0°.
      3. For the initial guess of the pixel reference, the means of the PSE coordinates can be used, and their range can be constrained by their maximum and minimum values.

4. One must ensure to remain aware of the parity of image coordinates with respect to catalog coordinates, given that we are only rotating and scaling and shifting the PSE triangle coordinates in the least squares fit.

2. The least squares transform will minimize wherever it needs to, but to check if the solution actually resulted in the respective triangle points falling onto one another, one can use the inverse of the solution to check if the catalog intermediate points inverse-transform onto the actual PSE coordinates given the PSE source index map. It may be possible that two similar triangles are not actually correspondent coordinates, and yet the transform of the intermediate coordinates of the intermediate triangle fall onto another triplet of PSE points. Thus, checking the indices serves the purpose of both performing a Boolean check of point-overlay, and checking that the indices match, thus eliminating such false positive solutions. If the inverse-transformed three intermediate coordinate triangle points fall onto the same indices of the current PSE triangle points, then this is a potential solution; if not then it was a false positive, and so continue the for loop to the next comparison iteration.
3. At this point we know that the solution resulted in a pair of triangles having points which fall on each other via transformation within the tolerance of the centroid kernel. However it is still possible that these are only similar triangles but not actually correspondent coordinates, given sufficient shifting of the reference position and rotation and scaling the two triangles were able to be fit to each other. If this is still a false-positive solution, then the probability that this solution will successfully transform additional intermediate catalog coordinates to other image source PSE coordinates is low, whereas if it is a true-positive solution then it should successfully transform additional intermediate coordinates to other PSE coordinates.
    1. We already have three pairs of points from the two triangles which fall on each other through the transformation solution, and so now check all other *n–3* intermediate catalog coordinates for additional successful inverse-transforms onto PSE coordinates via the PSE source index map. Here is where the confidence-criteria or stopping criteria can halt the nested for loop and report a solution.
    2. If an additional, say, three points of the catalog intermediate coordinates inverse-transform to PSE source locations, then likely this is a good solution.
    3. Or, if at least 25% of the n catalog intermediate coordinates inverse-transform to PSE source locations, then likely this is a good solution.
    4. If neither the previous two criteria are satisfied, then continue the loop to the next comparison iteration.
4. At this point we are confident that we have a solution. If parallelized, use for example a *critical* directive (as in the OpenMP nomenclature) to assign the solution to variables outside of the parallelized nested for loop.
5. Use all matching points determined in 6.3 to compute a full WCS solution via the standard CD matrix scheme.
6. Refine the solution further by finding now, say, n^*3 brightest image sources via PSE, and compare these against, say, the n^*6 brightest catalog entries using the existing WCS solution to find matching coordinates, and re-solve for a new WCS solution which now has points sampling more of the field.
7. Finished.

On a quad-core eight-threaded desktop PC running at 3.5 GHz on a 7th generation Intel processor, the algorithm is capable of performing approximately 500 million (5×10^8) triangle comparisons per second, when the scale and rotation are constrained to five percent. This rate decreases an order of magnitude when the rotation is constrained to only $\pm 180°$, given that the least-squares solver has more work to do here. Perhaps writing an optimized solver for only scale, rotation, and reference would improve things, given that we currently use a generic least-squares solver from AlgLib (https://www.alglib.net).

This finally returns us to a discussion of n, the number of brightest points used to form triangles of catalog intermediate coordinates and PSE coordinates, which triangles are then compared to one another. The number of unique triangles which can be formed from n points is $n!/(n–3)!3!$, or $n(n–1)(n–2)/6$. If one is confident of excellent correspondence between the source image and the catalog extraction, then perhaps only $n = 10$ is sufficient which results in 14,400 triangle comparisons which thus requires only a fraction a second to examine every single comparison, with a solution being found well-before all such comparisons need to made in this case given that the 10 respective points have good correspondence.

However, arbitrary images and catalogs may generally have an unknown correspondence. We certainly must use the correct catalog for a given image and its attendant wavelength detection range. And so for an example of $n = 100$, then there is a total of 26×10^9 possible triangle comparisons, which on our example system would require 52 s to process the entire lot; $n = 200$ would require an hour. If there is good correspondence though then of course a solution will still be found much more quickly than that, and core and thread count is increasing

rapidly on modern CPU's which massively benefits to parallelization.

For our test cases, the Baker-Nunn $4.^{\circ}4 \times 4.^{\circ}4$ field imaged upon a 4k × 4k pixel CCD through an *R*-band filer typically requires only $n = 10$ and solves in a few milliseconds when utilizing the GAIA DR2 "Gmag" catalog. The UVIT $0.^{\circ}5 \times 0.^{\circ}5$ field imaged on a 4k × 4k array typically requires $n = 25$ and also solves in a few milliseconds, when comparing near-UV sources to the bluer GAIA DR2 "BPmag" catalog. For UVIT far-UV images we typically require $n = 50$ with solutions in under one second utilizing the previous catalog, but one must be careful to avoid nebular regions and restrict the extraction to regions with point sources only during the PSE. The WCS standard errors on the UVIT images are typically $\sim 0.''2$ with an image PSF of $\sim 1.''2$ (full-width-half-maximum, FWHM) at a scale of $0.''416$ per pixel, while for the Baker-Nunn images they are $\sim 3.''5$ with an image PSF of $\sim 12''$ (FWHM) at a scale of $3.''9$ per pixel. In comparison with the same images and given identical field constraints, astrometry.net timed out after ten minutes for both of the UVIT images, and solved the Baker-Nunn image in 245 s of user time and 1.24 s of system time.

4. Conclusion

It is not actually typically the case that we must solve a completely arbitrary and unknown image against a completely arbitrary and unknown catalog... and nothing is likely capable of really doing that in any case, without making some assumptions about what the image is looking at. We are usually able to exploit correspondence and constraint. We may certainly invent situations which are entirely unsolvable, but for scientific instruments we generally do have the preliminary knowledge required in order to exploit correspondence between the brightest sources in our image and the brightest entries in an appropriate catalog, and to exploit constraint in the field scale and rotation and parity relations, etc. That being said, by restricting the analysis to only n brightest points and by working in intermediate coordinate space and comparing vertex angles, this algorithm would still function well in cases where source image metadata such as scale, rotation, pointing, etc., are not well known—one would simply need to expand or otherwise iterate upon the catalog search area and the parameter space. The algorithms presented here provide a robust, fast, and reliable method for "automatically" determining WCSs, where of course the critical first step is to determine which coordinates from a catalog match which coordinates in an image.

This project was undertaken with the financial support of the Canadian Space Agency in its support of the Indian Space Research Organization's UVIT telescope onboard the ASTROSAT spacecraft. We thank the referee for their review.

References

Brown, A. G. A., Vallenari, A., Prusti, T., et al. 2018, A&A, 616, 22
Calabretta, M. R., & Greisen, E. W. 2002, A&A, 395, 1077
Cardinal, R. D., Ryan, W. H., Hug, G., Ikari, Y., & Young, J. 2008, IAUC, 8993, 1
Cotton, W. D., Tody, D., & Pence, W. D. 1995, A&AS, 113, 159
Greisen, E. W., & Calabretta, M. R. 2002, A&A, 395, 1061
Greisen, E. W., Calabretta, M. R., Valdes, F. G., & Allen, S. L. 2006, A&A, 446, 747
Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 371
Grosbøl, P., Harten, R. H., Greisen, E. W., & Wells, D. C. 1988, A&AS, 73, 359
Harten, R. H., Grosbøl, P., Greisen, E. W., & Wells, D. C. 1988, A&AS, 73, 365
Hutchings, J. B., Postma, J., Asquin, D., & Leahy, D. 2007, PASP, 119, 1152
Kumar, A., Ghosh, S. K., Hutchings, J., et al. 2012, Proc. SPIE, 8443, 84431N
Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S. 2010, AJ, 139, 1782
Milone, E. F., & Clark, T. A. 1996, BAAS, 28, 1272
Postma, J., Hutchings, J. B., & Leahy, D. 2011, PASP, 123, 833
Postma, J. E., & Leahy, D. 2017, PASP, 129, 189
Rao, V. K., Agrawalb, P. C., Sreekumara, P., & Thyagarajan, K. 2009, AcAau, 65, 6
Wells, D. C., Greisen, E. W., & Harten, R. H. 1981, A&AS, 44, 363