FOCAS Automatic Catalog Matching Algorithms

Francisco G. Valdes

National Optical Astronomy Observatories[1], Box 26732, Tucson, AZ 85726, USA

Electronic mail: fvaldes@noao.edu

Luis E. Campusano

Departamento de Astronomía[2], Universidad de Chile, Casilla 36-D, Santiago, Chile

Electronic mail: lcampusa@das.uchile.cl

Juan D. Velásquez

Departamento de Ciencias de la Computación[2], Universidad de Chile, Casilla 2777, Santiago, Chile

Electronic mail: jvelasqu@dcc.uchile.cl

Peter B. Stetson

Dominion Astrophysical Observatory[3], 5071 West Sannich Rd., RR 5, Victoria, BC V8X 4M6, Canada

Electronic mail: stetson@dao.nrc.ca

Received ______________; accepted ______________

[1]National Optical Astronomy Observatories, operated by the Association of Universities for Research in Astronomy, Inc. (AURA) under cooperative agreement with the National Science Foundation

[2]Facultad de Ciencias Físicas y Matemáticas, U. de Chile

[3]Dominion Astrophysical Observatory, Herzberg Inst. of Astrophysics, National Research Council of Canada

ABSTRACT

This paper describes efficient algorithms that automatically take two or more catalogs of objects with instrumental coordinates and magnitudes and matches them. The challenges are that the instrumental coordinates may be only partially overlapping, at a different scale, rotated, or even inverted (flipped). The object magnitudes may be derived from different passbands so that the relative magnitudes of the objects differ. Also, the catalog may not contain all the same objects due to differences in separating close objects or to partial overlap between images. Finally, the catalog positions and magnitudes are subject to noise in the images from which they were derived.

The algorithms are applicable to any automated cataloging system. However, the implementation described here is part of the Faint Object Classification and Analysis System (FOCAS). FOCAS automatically produces catalogs of objects from digital images. The algorithms described here first take a subsample of the brightest objects from the catalogs and find a coordinate transformation between one catalog, the reference catalog, and other catalogs. Then all the objects in the catalogs are matched based on the transformed reference coordinates.

Subject headings: algorithms, catalogs, image processing, pattern matching

1. INTRODUCTION

The Faint Object Classification and Analysis System (FOCAS) is a suite of programs for automatically detecting, measuring, classifying, and cataloging objects in astronomical images (Jarvis and Tyson 1981, Valdes 1982a, Valdes 1982b, Valdes 1989, Valdes 1993). The principle product of FOCAS is a catalog of positions, magnitudes, and classifications. When there are multiple images of the same field in the same or different passbands the objects in the individual image catalogs may be matched and the catalog entries linked to form a *matched catalog*. The matching of the individual catalogs is not a trivial process because the images may not be accurately or even closely registered, the objects may not appear in all catalogs due to noise, crowding, or color differences, and when they do appear in all catalogs they will not necessarily have the same relative brightnesses. This paper describes the algorithms used to match FOCAS catalogs.

While the description, implementation, and examples of these algorithms are specific to FOCAS the algorithms are not. In the most general sense the algorithms deal with matching two or more sets of two dimensional coordinates which are not the same but related through some coordinate transformation. The algorithms described here are not quite this general in that a third quantity is used that provides a fuzzy ordering of the points. In astronomical applications this is the relative brightness of the objects. FOCAS generally deals with similar images (i.e. taken with the same telescope and detector) and the object coordinates are in the image coordinate system of pixels. However, the algorithms are not restricted to image coordinates or even coordinates with the same units and scales. For example, the algorithms allow for matching a set of objects with image coordinates against a reference catalog of right ascension and declination.

The matching of individual catalogs consists of the following steps. First a subset of the brightest objects in each catalog are compared to find matching objects; that is the same astronomical objects as seen in each image. This step assumes that while the relative brightness ordering may not be the same and all objects may not be present in this subset, in general a bright object in one image will be a bright object in the other image and 25% or more objects are in common. The image coordinates of the matched objects in each catalog are used to derive a coordinate transformation between the catalogs. One catalog is chosen as providing the coordinate reference system and the transformations convert the image coordinates of the other catalogs to the same reference system. All objects in the catalogs are then assigned coordinates in the common reference coordinate system using the derived transformations. Finally, objects are matched by near coincidence in the reference coordinate system. When there are more than two catalogs additional individual catalogs are successively added to the previous set of matched catalogs. The final result is a matched catalog which contains the objects from all of the individual image catalogs with links identifying which objects matched.

The reason for separating the catalog matching steps as described above is efficiency. The number of objects in a catalog can be extremely large. As discussed in this paper, determining the relationship between the coordinates in two catalogs is a pattern matching problem requiring the examination of many possible combinations of objects. This combinatoric aspect means that it is only practical to look at a small number of objects to find the unknown relationship between the image coordinates. Once the basic relationship is known then matching objects reduces to considering only the local neighborhood of each object in the common coordinate system, which can be done efficiently on a large number of objects.

The ability to produce matched catalogs from individual image catalogs has been a part of FOCAS for many years. However, the step of identifying common objects in the catalogs was not automated and required users to do this interactively with an image display or to input the coordinate transformations derived in some other way. The common bright object identification algorithm is the most recently implemented in FOCAS and involved the collaboration of all the authors of this paper. Using the algorithms described in this paper it is possible to entirely automate not only the matching of catalogs but all steps from initial detection of objects in a single image to forming a matched catalog of photometered and classified objects.

The following sections describe the algorithms used in each of the steps summarized above. Following the algorithm descriptions is an example of matching FOCAS catalogs with programs implementing these algorithms.

2. DETERMINING THE COORDINATE TRANSFORMATION

2.1. The Pattern Matching Problem

The first step is to determine the coordinate transformation between the coordinates in two catalogs (in FOCAS these are the image pixel coordinates). The transformation we seek may include a scale change, rotation, translation, and inversion (flip). This represents the typical relationship between astronomical images of the sky except for instrumental distortions. Minor distortions can be tolerated but we assume there are no severe distortions. Even when there are significant distortions, two images taken with the same instrument will have the same distortions which will then largely cancel out in most circumstances.

The coordinate transformation is obtained by finding a set of corresponding objects from the two catalogs and then computing the coefficients of the transformation by least squares. The first part is a general pattern matching problem between lists of points in a two dimensional space. A number of algorithms have been proposed for this problem (see references in Murtagh 1992). For our astronomical situation where we include the brightness of the objects, we use an intuitive approach mimicking what an astronomer does visually when matching images of the sky; namely to look for similar triangles among the brighter stars or galaxies. One might also include morphological similarity, such as using only classified stars or matching galaxies by appearance, but this is not done in our implementation.

This concept of finding corresponding objects using similar triangles has been described and implemented, independently, by Groth (1986) and Stetson (1989). Their algorithms are similar in many ways and we were guided by the work of both authors. Given that our starting definitions follow Stetson but that Groth describes his methods and results in more detail, we have organized this paper in sections similar to those of Groth in order to facilitate comparison of our algorithms with his. In our discussion we stress the significant new strategies as well as indicating areas of similarity.

The algorithm for finding a set of corresponding objects considers coordinate triangles formed from subsets of objects in each catalog. Objects in this case includes both stars and galaxies (though one could chose to select only stars if desired). The objects which are in common in the two catalogs will form many triangles which have the same shapes; shapes which are independent of scale, rotation, translation, and inversion. Non-common objects will only form a few chance triangles with the same shapes. Thus if one object in one catalog is part of many triangles that have the same shapes as triangles containing a particular object in the second catalog, the two objects will be identified as the same object regardless of the actual coordinate transformation between the catalogs.

One cannot actually look at all triangles from two catalogs because of the large combinatoric factors. Also the larger the subsets the greater the number of random triangle shape matches. So we make the assumption that the magnitudes of matching objects will be similar; i.e. the colors when using different passbands are not extreme in terms of many magnitudes so that a bright object in one image will also be a bright object in another image. Then we can take a subset of the brightest objects (which eliminates problems of spurious objects at the faint limits and numerous random triangle matches) and hope that within that set there will be a useful number of common objects. One parameter of the algorithm is N_{obj} which defines the number of the brightest objects to use from each catalog. Note that one can prefilter the catalogs to improve the odds of common objects based on some a priori knowledge. From this subset of objects all the triangle matches are found through the introduction of special coordinates that code the shape of the triangles.

The second part of solving for the coordinate transformation consists of fitting coefficients in the desired transformation equations by least squares. An important and novel component of our algorithm is that the fitting is iterated to reject misidentified object pairs leading to convergence to the true coordinate transformation and matching objects.

The algorithm has been implemented in FOCAS in the program `mktransform`. This program selects objects from FOCAS catalogs and computes the transformation from one catalog, called the *target catalog* to another catalog, called the *reference catalog*. The program then updates the target catalog object coordinates to the reference catalog coordinate system. Additional catalogs can be matched and transformed to the same reference coordinate system by using the same reference catalog.

2.2. Selecting The Points To Be Matched

From each catalog, the same number of the brightest objects is selected. In `mktransform` the brightest N_{obj} single objects (FOCAS detects and separates merged objects) are extracted. In our implementation, as in Groth's, we select the same number of objects from the two catalogs, though the catalogs generally have a different number objects (i.e. $N_1 \neq N_2$, with $N_1, N_2 > N_{obj}$). The algorithm does not require the two lists of coordinates to be the same length. For typical workstations the number of objects which can be reasonably handled is in the range 20-100 though beyond about 40 the gains are minimal.

As noted earlier, the program allows additional filters to be applied to the catalogs to enhance the number of common objects. Unlike Groth we do not eliminate objects which are close together in the same catalog. The reason for this is that we can tolerate some misidentifications and nearby objects will still lead to deriving a sensible coordinate transformation that can be used to sort out object mismatches and confusions between closely spaced objects.

Each catalog is read sequentially and objects which match a user specified filter are accepted; by default no filtering is applied. As each filtered object is read its instrumental brightness is compared against the N_{obj} previous brightest objects. If it is fainter than any of those it is discarded otherwise it inserted in the sorted list and the faintest one is discarded. Note that only the relative brightness within the same catalog are relevant. This selection and filtering during the input requires only one pass through each catalog and only six arrays of length N_{obj} for the two image coordinates and the brightness in each catalog. Thus the memory requirement for this step is $2 \times 3N_{obj} \times 4$ bytes $= 24N_{obj}$ bytes.

2.3. Generating Triangle Lists

The triangles formed from the two lists of selected objects are represented, following Stetson (1989), as points in a two dimensional *triangle space* (x_t, y_t) defined as:

$$x_t = b/a, \; y_t = c/a \tag{1}$$

where a, b, c are the lengths of the triangle sides in decreasing order. A triplet of common objects in the two catalogs will appear in triangle space as two nearby points, independent of scale, rotation, offset, and inversion between the original images. Fig. 1 shows schematically how the transformation to triangle coordinates is done starting from triplets of objects in the original images. The mapping of object triplets to triangle space is naturally restricted to the area inside the triangle shown in the figure.

The number of different triangles that can be generated from N_{obj} coordinates is $T = N_{obj}(N_{obj} - 1)(N_{obj} - 2)/6$. This can be a large number for even modest values of N_{obj} implying large memory requirements and many calculations. Thus, if the number of parameters describing each triangle and the number of triangles considered can be reduced, a substantial decrease in memory and execution time can be obtained.

The triangles formed by the set of objects can be minimally described by five quantities; the triangle space coordinates (x_t, y_t) and three indices identifying the three objects in the triangle. The use of indices allows linking to the array of image coordinates and brightnesses without reproducing these quantities for each of the many triangles containing a particular object.

The description of a triangle as a single point in triangle space and using two axes which are comparable (both are triangle length ratios) is simpler than Groth's description. Groth uses one length ratio and a vertex angle cosine as the primary description plus additional parameters dealing with position uncertainties, the handedness of the triangles, and the size of the triangles. His additional parameters allow rejection of mismatches at the expense of greater complexity and memory requirements.

The memory requirement in our description of the triangles requires only two four-byte words for the triangle length ratios and three two-byte words for the indices identifying the objects making up the triangle. Thus the total memory requirement, which includes the previously described object arrays, is $2 \times (12N_{obj} + 14T)$ bytes which is essentially $(28/6)N_{obj}^3$ bytes. This is at least a factor of three less than required by Groth's algorithm.

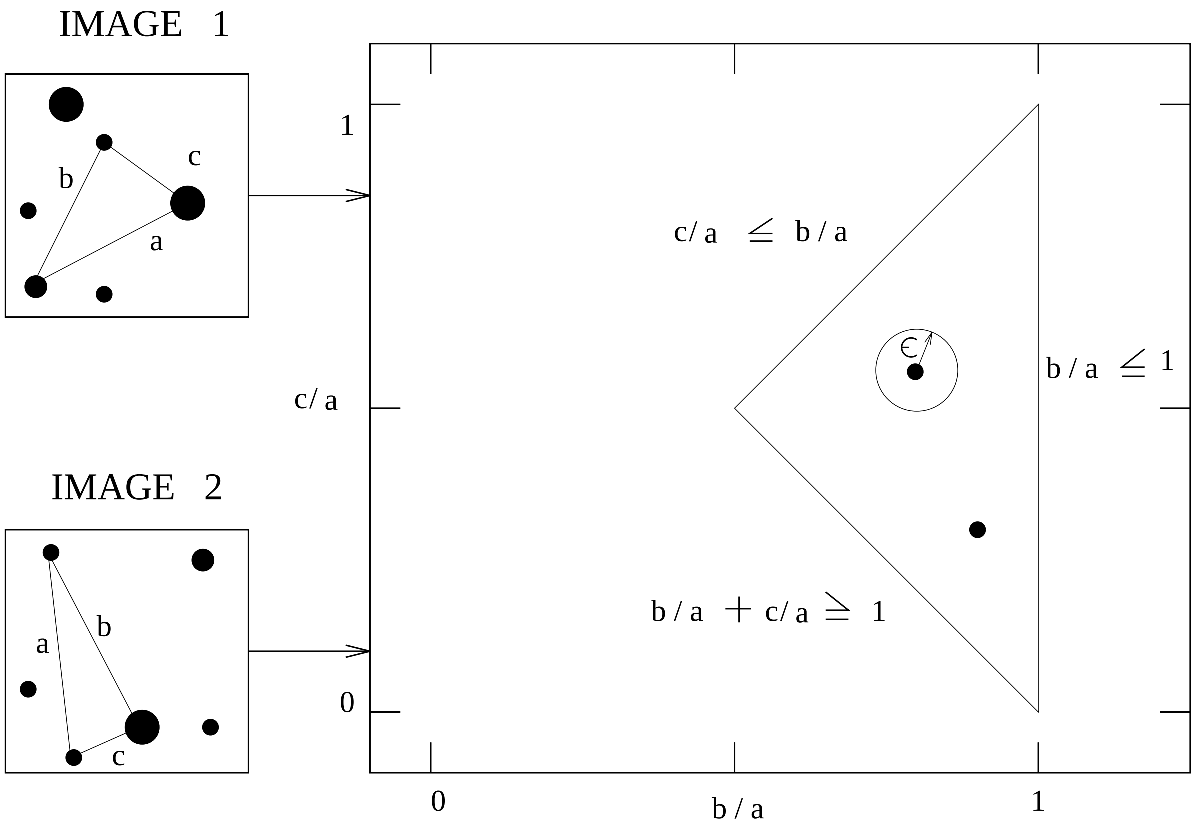

Fig. 1.— Representation of triangles in *triangle space*. The lengths of the triangle sides are used to form the ratios b/a and c/a and these ratios define a two dimensional space. Because of the ordering of the lengths the triangle coordinates will only occupy the indicated triangular region. Triangles from two images are matched when they are within a distance ϵ of each other in the triangle space.

An important optimization we found is to compute the distances between any two objects only once. Fig. 2 shows that two objects in a list define one of the sides of many triangles. By storing the lengths in an array indexed by the object indices, $l[i, j]$, the number of cartesian distances that must be computed is $\sim T^2$, much smaller than the $\sim T^3$ distances that would be needed otherwise. The dynamically allocated array of lengths is of size $N_{obj}(N_{obj} - 1)/2$. The lengths are only needed until the triangle space coordinates are computed for each triangle.

Following Groth, the triangles are restricted to those with the ratio $(b/a) < 0.9$ in order to reduce the number of false matches. This condition reduces the number of triangles considered for matching by about 1/4. This leads to slightly different numbers of triangles for each catalog; $T_1, T_2 \approx (3/4)T$.

2.4. Matching Triangles

Triangles are matched when their separation in triangle space is less than a specified distance ϵ (see figure 1) where ϵ is a parameter of the algorithm. This is a simple matching criterion. In Groth's algorithm the matching space consists of a side ratio (as in our algorithm) and the cosine of a vertex angle. What makes his algorithm different is that the matching distance is variable and is derived from propagation of errors given a position precision in the original coordinates (assumed the same for all objects).

Computing and comparing the distance between triangles in triangle space is the most time-consuming step of the algorithm. A naive search requires comparing every triangle in one list against every triangle in the other list. This would require $T_1 \times T_2 \approx 2 \times (3/4)T \approx N_{obj}^6/64$ comparisons where a comparison involves a cartesian distance computation. A very significant improvement is made by sorting one list by one of the triangle space coordinates (as also done by Groth). The `quicksort` (Gonnet &

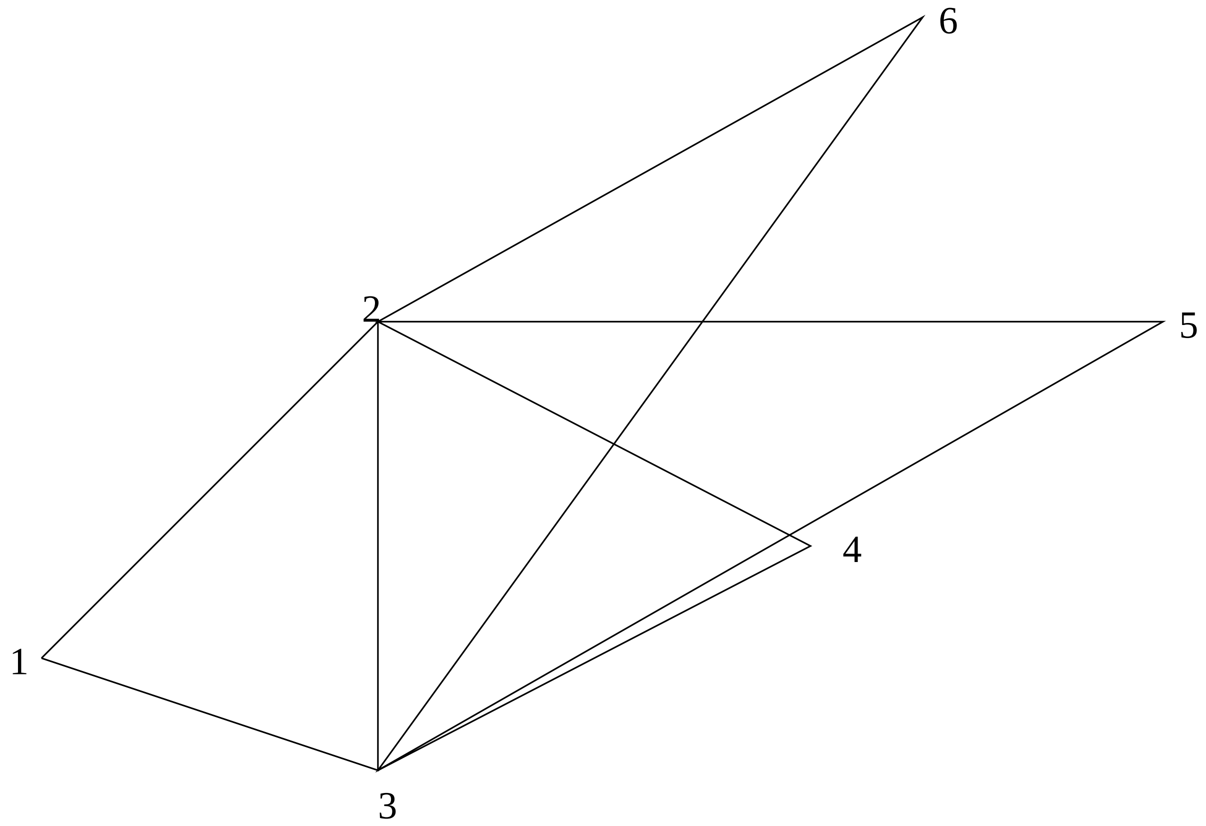

Fig. 2.— Two objects, 2 and 3 in this figure, define one side of many triangles with other objects. Rather than computing the sides of each triangle as needed, each distance between pairs of objects, such as between objects 2 and 3, are computed once and saved for later used in each of the triangles in which the two objects appear.

Baeza-Yates 1991) algorithm is used for this purpose.

A binary search can then quickly locate a subset $T_1'(\epsilon)$ of triangles within the one dimensional distance ϵ of a particular triangle in the second list. Only those triangles need to be checked for a two dimensional radius match. This means the number of two-dimensional comparisons is $T_1' \times T_2$. For a matching radius parameter of $\epsilon = 0.002$ we find $T_1' \approx 0.011(3/4)T$ which gives an improvement of about 100 in execution time over the naive approach. Note, however, that these optimization do not change the fact that the algorithm is $O(N_{obj}^6)$.

2.5. Reducing The Number of False Matches

One may extend the triangle space to eliminate mismatches. Two possible extensions are to include a normalized parameter related to the size of the triangle or brightnesses of the objects; say the triangle area, perimeter, or sum of brightnesses. The normalization might be the average of the quantity over all triangles in the list. This normalization makes the parameter scale independent. Similar triangles will then have similar values of the parameter. The idea is that these parameters will restrict matches such that larger triangles will only match with larger triangles and triangles of brighter objects will only match with triangles of brighter objects. This is intuitive and is certainly part of how a person recognizes matching triangles.

Though we mention these extensions, which we explored, we did not include them in our final algorithm for the following reasons. First this increases the dimensionality of the space in which the matching is done requiring more computations per triangle in this most time consuming step. It also requires additional memory per triangle and additional matching tolerance parameters. The benefits of this rejection of inappropriate triangle matches did not improve the final result since mismatches still occur by chance and the transformation computation step proved to be a powerful and sufficient way to reject mismatched object pairs.

As noted previously, Groth's algorithm also matches triangles in a two dimensional space of length ratio and vertex angle. Following the matching, mismatches are identified by finding a magnification and consistent handedness of the triangles. The magnification is found iteratively from the distribution of perimeter ratios and the sense from the relative numbers of same sense and opposite sense triangle matches. This greatly reduces triangle mismatches.

While Groth's algorithm does not determine and use a coordinate transformation between the two lists, his step of iteratively computing the magnification and handedness in order to reject false triangle matches is much like using a coordinate transformation; this essentially determines the scale and inversion of the transformation. So in comparing our algorithm with his we both use something like an iterative rejection based on defining a consistent coordinate transformation between the lists but Groth does this in rejecting triangles while we defer this to rejecting object pair assignments.

2.6. Assigning Matched Objects

Assigning candidate object matches between the two lists is based on the idea that when there are multiple common objects there will be an excess of matches between triangles containing corresponding objects. In other words an object from the first list will be part of many of the same matched triangles as that of the corresponding object in the second list.

This can be quantified as follows. In a list of N_{obj} objects, a particular object will be in $N_{obj}(N_{obj}-1)/2$ triangles. In an exactly matching second list there should be an object in that list which is in exactly that many matching triangles that contain the corresponding object from the first list. In practice the lists do not match exactly for various reasons and the actual number is less. However, it is hoped that the number will be significantly in excess of random triangle pairings. Groth showed that even with only 25% of the objects in common a valid matching of the objects could be obtained.

To count the number of triangles object i from list one has in common with object j from list two we define a vote array (as done by Groth and Stetson) of dimension $N_{obj} \times N_{obj}$ initialized to zero. For each triangle matched each of the three objects in the triangle casts a vote by incrementing the appropriate $[i, j]$ element in the vote array. At the end of the voting the vote array will, hopefully, have large numbers at the $[i, j]$ elements which correspond to matching objects. Table 1 shows an example of a vote array.

The vote array is the input for matching pairs of objects from the two lists. There are a number of ways in which this may be done. Clearly the higher the vote for a pair the more likely the two objects are a true match. One way is to apply a threshold to the number of votes. One would like to tie this threshold to the expectations for the number of random false votes and the number of true pairs. Estimating these numbers is hard and depends on many parameters such as the input number of objects, the expected errors in the coordinates, the tolerances, and the degree of magnitude disorder in the input lists.

We tried using a 10% factor from the theoretical perfect vote $(N_{obj}(N_{obj} - 1)/2)$ and a fraction (~ 0.5) of the highest actual vote cast for any pair. The latter is close to the criterion used by Groth. However, the number of matches produced varies greatly with variation of the parameters while the final results after rejecting points during the coordinate transformation step remains essentially identical.

Because the iterative rejection based on the coordinate transformation calculation worked so well we adopted the simple expedient of taking the top N_{obj} vote getters (which would be the maximum in a perfect case) out of the N_{obj}^2 combinations, without attempting to check for one object being assigned to multiple objects in the second list as done by Groth. This is another case of allowing some misidentifications from a simpler criterion to be rejected by the iterative coordinate transformation step. Extracting the sorted N_{obj} candidates is efficient since we only have to keep a small list sorted while checking every element in the array. To avoid a bias the actual number of candidates may be larger than N_{obj} when the lowest vote count occurs multiple times.

2.7. Rejecting False Matches and Computing the Coordinate Transformation

Our last step is to take the list of candidate object pairs and determine the image coordinate transformation that includes translation, rotation, scale change, and inversion. Since we are not guaranteed that the list has no mismatches and any mismatch is likely to be wildly wrong the algorithm must allow for rejection of such mismatches. The approach we adopted is to form a trial transformation to convert the coordinates from the target catalog to the reference catalog using the pairs with the highest likelihood of being correct. We select the six highest vote getters of the previous section; recall there are at least 20 pairs at this stage because $N_{obj} = 20$. Six is chosen since there are six coefficients to be determined requiring at least three coordinate pairs and we arbitrarily chose twice this minimum. The transformation is applied to the selected points from the target catalog and compared with the supposed identification in the reference catalog. The comparison is by cartesian distance using the square of the distance to avoid taking square roots. Those deviating by some amount are rejected and a new transformation is computed. This process is iterated until no more objects are rejected.

This is an *iterative sigma clipping* algorithm. It requires an estimate for the distribution of residuals, the positive valued square of the distances in our case, to define the clipping point. One could specify the expected sigma based on estimates of the catalog coordinate uncertainties. Rather than add another parameter we approximate this from the data. A common way to do this is to use the RMS of the residuals. Because the distribution of outliers in this application is extreme, i.e. a mismatch could be anywhere in the coordinate space, the RMS will be strongly influenced and so a different method is required. The distance squared residuals are sorted and the 60th percentile point is used as a sigma estimate for the random error distribution and all points greater than twice this (the clipping factor) are rejected.

The clipping algorithm currently used either accepts or rejects candidates at the derived clipping threshold. Stetson (1989) describes a variant algorithm that uses variable weights based on residuals to make the rejection criterion behave more smoothly; i.e. points are included with weights going continuously to zero at the threshold. Anyone implementing our algorithm might wish to consider this variant.

The form of the coordinate transformation sought is defined by the equations

$$x' = Ax + By + C \tag{2}$$

$$y' = Dx + Ey + F \tag{3}$$

where (x, y) are the target catalog image coordinates and (x', y') are the desired reference coordinates. This transformation is what has been used in FOCAS since it was developed. It might be more appropriate to use a transformation which is more specifically described in terms of a scale change, rotation, offset, and inversion and does not include a shear. The algorithm could be easily modified for any transformation equation desired.

The transformation coefficients, A to F, are determined by least squares fitting using the (x, y) image coordinates of objects in the target catalog which have been matched to objects in the reference catalog with (x', y') reference coordinates. The least-squares solution is obtained using the algorithm HFTI based on Householder transforms as described by

Lawson & Hanson(1974). This least-squares method was used because it is robust and the code is available from the authors in their book. Any equivalent method for solving (2) and (3) may be used.

Equations 2 and 3 provide a general transformation. If this transformation is assumed to be generated by a combination of rotation, translation, inversion, and a change of scale of one of the coordinate systems then the resulting coefficients can be used to derive the scale, rotation angle, and offset. This interpretation generates two estimates for the scale and the rotation angle. If the individual scales and rotations disagree significantly it would indicate a shear of some kind which is unlikely in astronomical images; especially when they come from the same instrument.

The solution allows us to detect and correct for inversion between the two images. If the derived rotation angles disagree by 180 degrees it is indicative of an inversion. The output of the program, an example is given in figure 4, includes the RMS of the transformation between the final list of matched objects, the transformation coefficients, and estimates of the offsets, scale change, rotation, and indication of any flip.

After the transformation converges the `mktransform` program goes through all objects in the input target catalog and computes reference coordinates from the image coordinates. These coordinates are then used by other programs, most notably the `match` program described below.

2.8. Program Requirements

The memory requirements were discussed previously. The amount of memory needed is dominated by the number of triangles and the number of quantities representing each triangle. For the default of $N_{obj} = 20$ objects per catalog the memory requirement is of order $(28/6)N_{obj}^3 = 32$ Kbytes which is quite modest. Larger numbers of objects rapidly increase the memory requirement.

The execution time for `mktransform` on a SUN Sparc 10 workstation is of order a second or two with the default selection of the 20 brightest catalog objects; 20 objects generally gives a good solution. With the optimizations described here even 50 objects can be used to compute a solution in under a minute.

This is to be compared to the non-automated approach for computing the transformation in FOCAS catalogs where the two images are displayed and the user selects three to ten common objects from each image with a cursor which are then located and flagged in the catalogs. Another program then uses the selected catalog objects to compute the transformation equations 2 and 3 and set the reference coordinates. This often is less satisfactory because the users selects a smaller number of common objects and is much slower, usually requiring ten minutes or more. Furthermore, the interactive approach is not suited to batch processing.

3. MATCHING CATALOGS BY REFERENCE COORDINATES

After applying `mktransform` all objects in the individual image catalogs have coordinates in a common reference coordinate system. This section describes how the objects are matched from the different catalogs based on proximity in the reference coordinate system. In outline, the cartesian distance between objects from different catalogs is computed and the ones with the smallest separation, within a user specified matching error radius (the default is three pixels), are linked together in a matched catalog. A matched catalog is a merging of the separate catalogs preserving all the individual information on each object and image but with links between the objects included. In the following the word "catalog" refers to a single image catalog and the merged catalog is referred to as a "matched catalog".

The challenges are to do the matching efficiently in terms of memory (since the catalogs may be quite large), to allow for errors in the coordinates due to noise and inaccuracies in the coordinate transformations, and to allow for missing objects or situations where two close objects are separated in one catalog but appear as a single object in another catalog. Also when there are more than two catalogs there may be various combinations of matches and non-matches.

A matched catalog is built up by successively adding new catalogs rather than simultaneously considering all objects. This minimizes the complexity and amount of memory required and also modularizes the program. The FOCAS program that implements this algorithm is called `match`.

3.1. Sorting the Input Catalogs

The first step before adding a catalog to a matched catalog is to sort the objects in the catalog by one of the coordinates. As discussed below the matched catalog is always in sort order. Sorting the catalog entries allows matching objects using a scrolling window of finite width. For FOCAS the y coordinate is used since objects are detected and cataloged by scanning through the image in this order. Thus, the input catalogs tend to be approximately ordered in this coordinate and the sorting can be done more efficiently.

The sorting is done by reading blocks of 3000 objects into a linked list which is continuously relinked in sort order as each object is read. The linked list is a list of object entries with a pointer from one object to the next object in the desired order. Since each object consists of a complex record of various measurements it is more efficient to change the pointers between object records than to physically sort the object records in memory.

As a new object is read the links are followed until the point where the new object has a coordinate between two previous objects is found and then the pointer from the first object is changed to the new object and the pointer from the new object is set to point to the second object. Because the input objects are approximately sorted the traversal of the links can be optimized by starting from the point at which the last entry was added rather than at the beginning.

The sorted blocks of objects are written to a sequence of temporary sorted subcatalogs. The subcatalogs are then merged by reading one object for each subcatalog and outputing the lowest in sort order to another temporary catalog and replacing the object just output by the next object from the appropriate subcatalog. Finally the temporary subcatalogs are deleted and the temporary sorted catalog is used as input for the matching. As this catalog is read during the matching step any merged objects are skipped; in FOCAS both the original merged objects and the separated objects are kept in the individual catalogs but not in the matched catalogs.

3.2. Matching the Input Objects into a Matched Catalog

When the first catalog is added to a new matched catalog it is simply a matter of copying the catalog header and objects. When further catalogs are added, the current matched catalog and the new catalog are used as input. Since the final result of adding a new catalog to the matched catalog updates the matched catalog, a temporary matched catalog is produced as output which then replaces the original matched catalog at the end of the matching.

A scrolling buffer of 1000 sorted entries from the matched catalog is read into memory and a check is made that the y range is greater than the error radius allowed for a pair of coordinates to match. The y range of objects will be limited since the matched catalog is maintained in a y sorted order by the nature of this algorithm and the input catalogs are sorted as described previously. As each object from the input single catalog is read all sets of matched objects in the buffer which have a maximum y (over all objects linked together as matches) less than the y of the object minus the matching distance can be written out as complete and the buffer elements replaced by further input from the current catalogs. The buffers are circular so that there is never any need to shift object entries in the buffers.

Each input object to be matched is compared with all objects in the matched catalog buffer to find the minimum distance to *any* object. Note that when a matched catalog entry consists of two or more objects all the objects in the entry are considered rather than some average position. The input object is linked with the minimum distance match found if it is within the allowed error distance otherwise the object is written out to a temporary *no match* catalog. If an object is linked to objects from previous catalogs this does not mean the match is finished since another input object may match with a smaller distance. If this happens then the previous object from the catalog being matched is unlinked and recompared with all other sets of matched objects for the next best match. After going through all objects in the input matched catalog and single image catalog the result is two temporary catalogs of matched and unmatched objects. These two are merged back to a complete matched catalog with the original input name and preserving the y sorting. All temporary files are finally deleted.

The matching of more than two catalogs in a cumulative manner raises difficult issues of dependencies on the order of adding catalogs and on identifying possible overlapping subgroupings. An important point is whether new objects are positionally compared to an average position for the previously matched objects or not. The former approach is strongly dependent on the order in which the catalogs are matched while our approach of considering all objects is much more weakly dependent. In particular, if we ignore cases where a new object has a choice between more than one previous matched group our method is order independent.

A limitation of the algorithm is that previously matched objects are never unlinked and rematched with the addition of a new catalog. This has the consequence that an object in one match may later be closer to an element of a different group than to any of the members of its current group. In this sense there can be some dependence on the order in which catalogs are added.

The algorithm given here will generate the largest single group for matched objects where subelements may not be within the matching distance but where all subelements are within that distance of at least one other object. Users may identify such possible loose groupings by a weight parameter which is calculated based on all the interobject distances.

In our experience the above concerns are not important in practice. Often the applications consists of only pairs of catalogs where they do not apply. The problems occur with three or more catalogs and become likely only in very crowded situations such as encountered in dealing with crowded stellar fields. FOCAS was not designed or intended for this type of data.

The matching of more than two catalogs, particularly with a much larger number than two, forms a separate field in computer science with complex algorithms. These are very intensive grouping or clustering methods that involve large combinatoric factors. For the purpose of FOCAS the rare possibilities of confusion does not warrant the much greater complexity of such algorithms.

3.3. Program Requirements

The `match` program actually implements the catalog sorting step described above by spawning a separate program. This allows the catalog sorting program, `catsort`, to be used separately if desired and to provides other sorting keys than the y coordinate.

The memory requirement for `catsort` is set by the size of the sorting blocks. For 3000 objects, where each FOCAS object record is 0.164 Kbytes, the memory needed is of order $3000 \times 0.164 = 492$ Kbytes. A maximum of 17 temporary files are provided which allows sorting catalogs with up to 51000 objects. These numbers can be adjusted if needed for even larger catalogs. The execution time depends on the number of objects in each catalog and disk access speed to read and write catalog files. On a Sparc 10 the execution time is of order three seconds for catalogs of 1000 objects.

The memory requirement for matching the objects is larger than one might first think with a buffer size of 1000 entries from the matched catalog and the input catalog being added. This is because each individual object record is of size 0.164 Kbytes and the matched catalog may contain entries which link up to eight objects from separate catalogs. Thus the total buffer memory requirement is $9 \times 1000 \times 0.164 = 1476$ Kbytes. Execution time depends on the number of objects in the individual catalogs and the number of catalogs being matched. On a Sun Sparc 10 matching just two catalogs the execution time is of order 15 seconds per 1000 catalog objects.

4. EXAMPLE

4.1. Two Images with Shift, Rotation, and Inversion

In this section we demonstrate the algorithms, as implemented in the FOCAS programs `mktransform` and `match`, using two very deep images of the South Galactic Pole. The images, shown in figure 3, are J and R band exposures taken with the CTIO 4-meter telescope (described further in *Physics Today*, 1987). They are part of a publicly available standard test image suite (Murtagh and Warmels, 1989). The archived images are registered, so for the purpose of this example the R band image was artificially flipped in Y, rotated by five degrees about a point slightly offset from the center of the image. The two images were then trimmed by the same amount in order to remove the portions of the shifted and rotated image for which there is no data. The resulting images are 195x421 pixels. Below we compare the introduced transformation against that recovered by `mktransform`.

4.2. MKTRANSFORM

The objects in the images are first cataloged by FOCAS to form two catalogs `sgp1.cat` and `sgp2.cat`. `Mktransform` is used to derive the coordinate transformation between the cataloged image (pixel) positions in the two images. We chose `sgp1.cat` to be the reference catalog and find the transformation that converts the image coordinates from `sgp2.cat` to this reference. Figure 4 shows the usage information (produced in all FOCAS commands with the caret argument) and the terminal output from applying the program to the two catalogs. For comparison 5 gives the expected output based on the known transformation introduced in the `sgp2` image.

The first step of selecting the 20 brightest objects in each catalog (the default value of N_{obj} for this program) is illustrated in the first two panels of figure 3. From these objects 896 triangles are formed from the first catalog and 867 from the second catalog; recall that triangles with $(b/a) > 0.9$ are eliminated. The matching triangles, as represented by points in triangle space, are found using the default distance $\epsilon = 0.002$. In this example 275 matching triangles are found.

Each triangle match suggests a correspondence between three pairs of stars. The vote array gives the number of times each star, as identified by the list index number ordered by brightness, in one catalog has a correspondence with a star in the second catalog. The

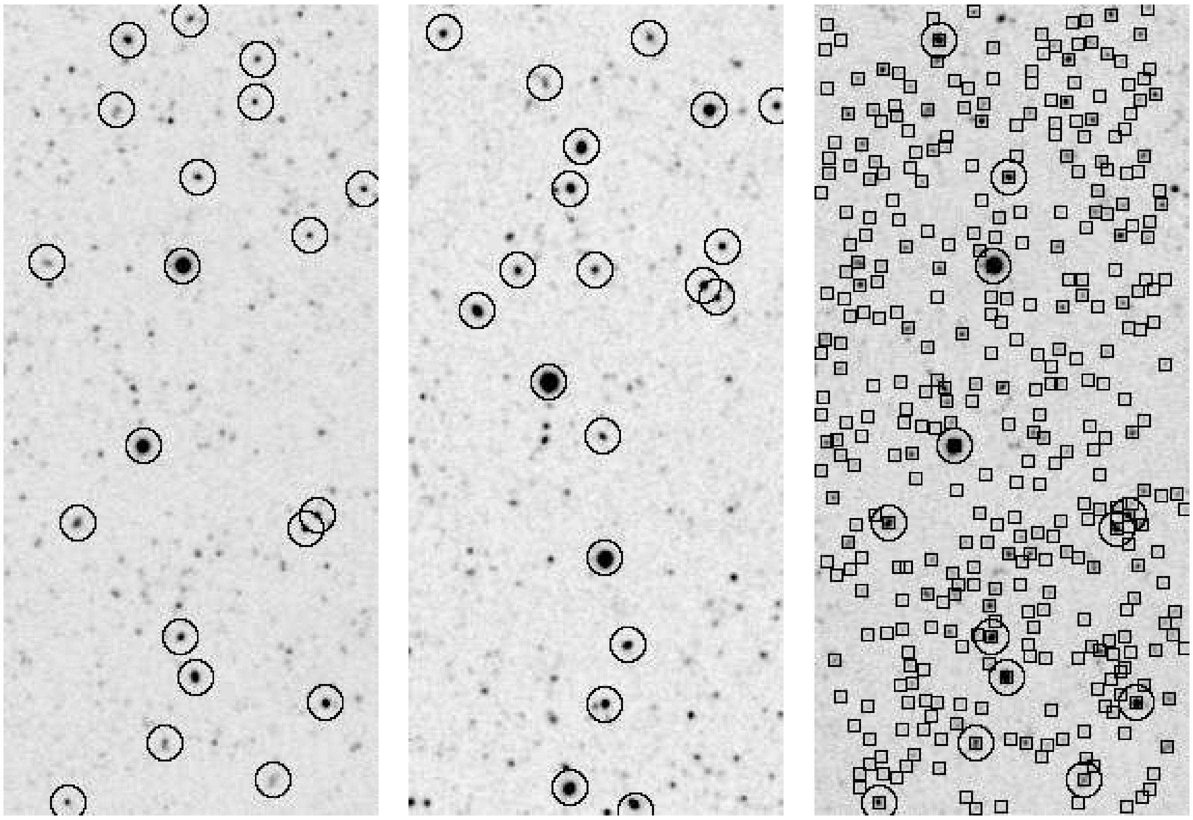

Fig. 3.— Example images consisting of J and R band exposures of the Southern Galactic Pole. The left figure shows the J band exposure with the 20 brightest objects from the FOCAS catalog of the image selected as reference coordinate input to the triangle matching algorithm. The middle figure shows the R band exposure with the 20 brightest objects from the FOCAS catalog of the image selected as target coordinate input to the triangle matching algorithm. The right figure shows the J band image with the large circles indicating the objects selected by the triangle algorithm to define the coordinate transformation from the target coordinates to the reference coordinates. The small squares indicate all the objects matched by the catalog matching algorithm between the catalogs of the two images after applying the coordinate transformation.

Fig. 4.— Example output from `mktransform`

```
% mktransform ^
 Finds the transformation matrix between two catalogs.
 Usage:    mktransform [-t tol] [-n nobj] target reference [filter]
 Argument: target    - catalog to find transformation and set coords
           reference - catalog giving reference coordinates
           -t  tol  - matching tolerance in triangle space (default 0.002)
           -n  nobj - number of objects to use (default 20)
           filter   - filter options
 Output:   The transformation matrix and reference coordinates are set
           in the target catalog.  The standard output gives the
           transformation coefficients and estimates of the offsets,
           scales, and rotations.
% mktransform sgp2.cat sgp1.cat
Number of matches = 13, RMS of fit =     0.39
Transformation coefficients:
            0.996879   0.087524   -19.994843
            0.087130  -0.995545   407.709351

                         Xtransform   Ytransform      Average
X Offset (pixels):                                     -19.99
Y Offset (pixels):                                     407.71
Scale:                       1.0007       0.9994       1.0000
Rotation (degrees):            5.02      -175.00         5.01
Flip:                                                     YES
```

Fig. 5.— Artificial transformation introduced to the data in the same format as given by `mktransform`.

```
Transformation coefficients:
            0.996195   0.087156   -19.936602
            0.087156  -0.996195   407.790354

                         Xtransform   Ytransform      Average
X Offset (pixels):                                     -19.94
Y Offset (pixels):                                     407.79
Scale:                       1.0000       1.0000       1.0000
Rotation (degrees):            5.00      -175.00         5.00
Flip:                                                     YES
```

array for this example is shown in table 1.

If the same 20 objects were in each list and only the triangles made from corresponding triplets of objects are matched, the number of triangles that have each of these stars in one of their vertices would be $(N_{obj}-1)(N_{obj}-2)/2 = 171$. However, in this example only 13 of the objects are in common between the two lists due to magnitude and edge differences. This would lead us to expect 66 votes for each pair of objects. Due to centering differences the actual number of matches, shown in bold type, range between 18 and 53. The point is that there is a clear difference between stars which match many times and a random level of chance matches. Because the star numbers are magnitude sorted one can also see a general diagonal trend but color differences are also apparent.

As discussed earlier, it is not always as clear as this example where to divide the vote array values between good candidate identifications and random matches. We chose to select the N_{obj} highest values as tentative object matches. In this case 21 object identifications are selected consisting of those with six or higher votes which are marked in the table by bold or italic fonts. There are 21 candidates selected because there are two cases with six votes.

The iterative rejection step uses the 12 best candidates to determine the initial coordinate transformation and then rejects and refits object pairs that do not transform near each other. The algorithm readily finds that only 13 of the 21 candidate pairings are consistent and the best transformation yields an RMS pixel position difference of 0.39 pixels. One should interpret this in light of the fact that the input coordinates used by the program are integer pixels only and are based on the center of the 3x3 box having the highest flux.

The transformation coefficients for equations 2 and 3 are shown in the output. The interpretation of this transformation in terms of origin shifts, scale change, rotation, and inversion yields the indicated values. Comparison of the transformation coefficients and

Table 1: Vote Array

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	4	**29**	2	0	0	2	1	3	3	2	1	0	0	1	0	0	2	0	0	1
2	**42**	*8*	1	2	1	1	1	2	2	1	1	1	1	0	4	0	0	1	0	*7*
3	2	3	**21**	0	1	0	0	1	0	0	3	0	0	2	2	0	2	1	0	2
4	3	0	2	0	**53**	0	1	1	2	1	0	2	2	2	1	0	2	3	2	0
5	3	1	0	**24**	2	0	1	0	4	2	2	0	0	1	2	0	1	0	0	0
6	0	1	3	2	1	2	0	1	0	2	*9*	3	0	0	2	2	3	1	1	0
7	0	2	1	1	0	**29**	1	1	1	0	0	0	2	0	0	0	0	2	0	0
8	1	1	1	2	1	1	1	2	0	1	3	2	1	0	3	1	1	0	1	1
9	2	0	0	0	0	4	1	**21**	1	0	2	0	0	0	2	1	1	2	1	1
10	3	1	0	1	1	1	**20**	2	1	*5*	3	1	1	0	1	1	1	0	0	0
11	2	2	1	1	2	*5*	2	3	**46**	1	0	0	3	2	2	1	2	2	2	0
12	1	0	0	1	0	1	1	1	1	1	2	2	0	0	1	1	1	0	2	*6*
13	2	0	1	2	1	0	1	0	0	2	1	3	0	1	0	0	0	0	1	0
14	0	2	3	0	0	0	0	2	0	1	3	3	0	1	**31**	2	2	3	1	0
15	0	0	*5*	1	4	0	0	1	1	0	4	0	1	1	0	2	0	0	0	0
16	1	1	2	3	2	1	2	0	1	0	1	**18**	0	0	2	2	1	2	0	0
17	0	1	2	2	1	1	0	0	2	**23**	1	0	1	0	0	1	1	0	0	0
18	0	3	0	1	1	0	2	1	1	3	0	0	2	1	1	0	1	1	1	0
19	0	0	0	0	0	0	1	3	1	1	0	4	2	0	2	2	0	**28**	3	1
20	*6*	4	0	4	0	4	1	1	1	1	1	0	1	0	2	0	1	0	0	0

interpreted scale, rotation, and flip with the known transformation introduced artificially in the target image shows excellent agreement. One should also keep in mind that the original images may not have been exactly registered and scales may be slightly different because of optical differences between the two bandpasses.

4.3. MATCH

The two catalogs are matched to form the matched catalog `sgp.mcat` using the command shown in figure 6. The figure also shows the usage information and the terminal output from running the program. The matching takes approximately nine seconds on a SUN Sparc 10 where the input catalogs contain 522 and 645 objects respectively.

Statistical measures of the matching are difficult to present and interpret because the large number of faint objects near the limit of the data and edge objects bias the matching percentages. Thus the results need to be presented by magnitude and spatial position. The eye is very good at this type of discrimination if the matched objects are marked on one of the images. One should see all the obvious objects marked with no pattern of matching density across the field. In figure 3 we see that the matching worked very well. The missing match in the bright galaxy in the field (the small box is missing) is due to a difference in centroid position in this extended galaxy which exceeds the default three pixel error radius.

5. CONCLUSION

We have presented algorithms that automate the task of matching objects from multiple FOCAS catalogs where the objects have similar though not identical brightnesses and possibly differing image coordinates; i.e. the catalogs are derived from unregistered though overlapping images. These algorithms have been implemented in FOCAS as illustrated in the example. However, the algorithms are general and may be used to good effect in any software dealing with the problem of matching lists of coordinates.

We studied and optimized the algorithms in terms of memory requirements and execution time. These algorithms work on catalogs of arbitrary size and produce matched catalogs with little effort and time. Two catalogs of roughly 1000 objects can be automatically matched in a few minutes on typical small workstations.

The triangle matching algorithm defining the coordinate transformation between the lists is the most novel one described. The concept and aspects of the algorithm are similar to those of Groth (1986) and Stetson (1989) in that they are based on the identification of similar triangles. However, our algorithm differs from the one offered by Groth and Stetson in several ways. In broadest terms our algorithm uses just a simple matching of triangles with two parameters and tolerates more mismatches in the list of candidate object identifications which are then weeded out in an iterative solution for the coordinate transformation between the two input lists. Groth's algorithm uses more parameters and greater complexity in deriving the set of candidate matched objects between the lists with the goal of eliminating mismatches before consideration of the coordinate relationship between the two lists or catalogs.

Though deriving a coordinate transformation was not an explicit part of Groth's algorithm (it was noted that the final list of matched objects could be used for this purpose), a step in his algorithm which eliminates many triangle mismatches can be considered, in a way, equivalent to using a iterative coordinate transformation and rejection. In either case it is an important element of each algorithm to reject mismatches derived from similar triangles based on coordinate consistency.

Another point to note about the differences with the Groth algorithm is his iteration of the derived set of matched objects back to the triangle generation step until all objects either remain matched or all points are rejected. We do not include this procedure since there are no rejection steps prior to the last step which, therefore, yields the same matched points that already survived the iterative transformation and rejection step.

The Faint Object Classification and Analysis System is distributed by the National Optical Astronomy Observatories (NOAO) at no fee. The software is currently available for Unix host computers. For information on obtaining FOCAS, which includes the algorithms described in this paper, send electronic mail to iraf@noao.edu or postal mail to the Central Computer Services of NOAO.

LEC and JDV were partially funded by FONDECYT grant No. 985/93. We thank useful discussions with Dr. Ricardo Baeza-Yates from the U. of Chile.

Fig. 6.— Example output from `mktransform`

```
% match ^
usage:  match [-D delta] mfile cfile1 cfile2 ...
  Create or add plate catalogs to a match catalog.
  Arguments: mfile - match catalog to be created or added to
             cfile - plate catalog files to be matched
                     Must be in common coordinate system
  Options: -D delta - maximum matching distance in RA/DEC units
                      default = 3
  Input: none
  Output: progress information
% match sgp.mcat sgp1.cat sgp2.cat
match: Creating new match file sgp.mcat from sgp1.cat
match: adding file sgp2.cat to match file sgp.mcat
```

REFERENCES

Gonnet, G.H. and Baeza-Yates, R. 1991, in Handbook of Algorithms and Data Structures, Addison-Wesley

Groth, E.J. 1986, AJ, 91, 1244

Jarvis, J. F. and Tyson, J. A. 1981, AJ, 86, 476

Lawson, C. L. and Hanson, R. J. 1974, in Solving Least Squares Problems, Prentice-Hall

Murtagh, F. and Warmels, R. H. 1989, *Test Image Descriptions*, in Proceedings of the 1st ESO/ST-ECF Data Analysis Workshop, ed. P.J. Grosbol, F. Murtagh, and R.H. Warmels, ESO. (The images are available electronically in `ftp://iraf.noao.edu/iraf/extern/focas.std.tar.Z`)

Murtagh, F. 1992, PASP, 104, 301

Physics Today, March 1987, p19

Stetson, P.B., 1989, in V Advanced School of Astrophysics, Image and Data Processing/Interstellar Dust, ed. B. Barbuy, E. Janot-Pacheco, A.M. Magalhães, and S.M. Viegas (São Paulo, Instituto Astrônomico e Geofísico).

Valdes, F. 1982a, *Faint Object Classification and Analysis System*, NOAO document (`ftp://iraf.noao.edu/iraf/docs/focas/focas.ps.Z`)

Valdes, F. 1982b, Proc. SPIE, 331, 465.

Valdes, F. 1989, ESO Proceedings 31, 35 (`ftp://iraf.noao.edu/iraf/docs/focas/standards.ps.Z`)

Valdes, F. 1993, *FOCAS Users's Guide*, NOAO document (`ftp://iraf/noao.edu/iraf/docs/focas/focasguide.ps.Z`)

This manuscript was prepared with the AAS LaTeX macros v3.0.