Faint Object Classification and Analysis System

Francisco Valdes
National Optical Astronomy Observatories††
Tucson, Arizona 85726
October 1982

ABSTRACT

A general introduction to FOCAS is given. Following the introduction are detailed sections describing the data structures of the FOCAS catalogs, image files, and area description files and algorithms for detection, evaluation, classification, splitting, and matching. This manual makes reference to the **FOCAS User's Manual**.

1. Introduction

The Faint Object Classification and Analysis System, called FOCAS, is a set of programs for creating and manipulating catalogs of objects from digital astronomical images. The creation of a catalog is accomplished by an automatic threshold detector where the threshold is measured relative to a simultaneously determined background. The manipulation of catalogs includes the separation of merged objects, the measurement of various position, shape, and photometric parameters, the astronomical classification of the objects, and the display and analysis of the catalogs. This paper describes much of the workings of FOCAS with reference to the FOCAS User's Manual.

The programs of Version 3 FOCAS are designed around the UNIX operating environment. This means that many of the normal UNIX system programs and operating features are used to give great flexibility in data manipulation and programming. The most powerful feature is the use of shell scripts which make the tools of FOCAS a programming language. Many of the routines which are part of FOCAS are actually shell scripts.

An important first step for a user is to realize what FOCAS cannot do. It is not a precision photometry or astrometry program. It is also not infallible in detecting objects, separating merged objects, and making classifications. For these reasons it is not greatly useful for studying a few objects which can be done directly by the astronomer. The original motivating purpose for FOCAS was the measurement of large numbers of objects to obtain statistical results. These qualifications are not actual limitations. The limits are imposed by the absence of programs to do precision photometry or astrometry. Such programs will be added as time and need allows. The catalogs have the flexibility to record any data, such as for specialized projects, and to drive routines to make additional measurements from the digital image.

The remainder of this introduction is devoted to an outline of the various steps and stages in using FOCAS. This outline is not a general recipe. There is no linear way to describe the steps one goes through in analyzing a particular digital image. For example, different steps and programs are used for photographic data as opposed to CCD data. The image data may have a previous history of processing by other image processing systems such as IPPS and the CCD reduction programs. Finally, there is no recipe at all for the analysis of the catalog or matched

††Operated by the Association of Universities for Research in Astronomy, Inc. under cooperative agreement with the National Science Foundation.

catalog since every astronomer will have a different project and goal in mind.

The first step is the astronomer's which is to obtain a digital astronomical image and record it on tape. The two common forms are digitized photographic plates and CCD images. To get the image(s) into the computer FOCAS has programs to read PDS tapes (**rpds(1)**), CAMERA format tapes (**rcam(1)**), and FITS tapes (**rfits(1)**). The tape density and track limits are placed by the available tape drives which are 9-track, 1600 BPI or 800 BPI. The image data is integer and can be read into 8, 16, and 32 bit disk files. If the data is digitized photographic data with sensitometer spots the spots are read as an image and used to determine the density-to-intensity curve.

FOCAS operates from the image as a disk resident file. Thus, images are limited by available disk memory. The available memory will be set by policy, the number of users requiring storage, and the addition of disk drives to the system. Until the KPNO/B UNIX Vax 11/750 computer (where FOCAS currently resides) becomes a general data reduction facility it is impossible to give a firm figure for the memory limitations. A ballpark figure for initial users is 30-40 Mbytes. Bigger images could be processed in pieces. However, this size is also near the limit for reasonable processing times of order 10 hours. (Detection is only a small part of this figure.)

The usual next step is to display the image or a piece of the image. The program to display an image is **display(5)**. In order to understand the mechanics of image display one must know how FOCAS interprets the image pixel intensities. There are four types of pixel values, some or all of which may be identical; raw integer, image data, intensity, and display. The raw integer pixel values are the actual values in the disk image file. The image data is obtained by a scaling and offset to the raw integer values. The scale and offset values are in the image header and are called **bscale** and **bzero**. These values are normally supplied by entries of the same name in the FITS image header. This option is used to allow floating point data to be used by FOCAS and is the common result of writing FITS tapes from IPPS rasters. The intensity values are obtained by a table lookup from the the image data values. The lookup table is set by the user in the catalog header. Finally, the display pixel values are 8-bit obtained from the raw integer pixels by a scale and offset in the header analogous to the conversion to image data. The header values are called **dscale** and **dzero** and are set by the user to produce a display to his/her taste. These display values are set by the user using either **chimhdr(1)** (change image header) or with the general image display program **display(5)**.

The **display(5)** program automatically decimates (reduces the resolution) of an image to fit the 512 x 512 IIS display. To display parts of a big image or to extract small fields to run tests one uses **exfld(1)** (extract field).

The next step is generating a catalog of objects. The catalog for a field is initialized by **setcat(2)**. This program is a general interactive (prompting) routine for manipulating the catalog header and supplying information to various of the processing routines. (A detailed description of the catalog header is given in the next section of this manual.) Not all of the options need be set initially. In fact, in the initial setting of a catalog for object detection the only required (non-defaulted) input is the field image filename. There are, however, three important parameters which the user should consider for detection; the detection threshold sigma, the minimum detection size, and the detection filter.

The FOCAS automatic detector (**detect(2) -** see section 5 for a more detailed description) begins by examining the first few lines of the image to determine the background mean and sigma. Note the detector works entirely with the raw integer pixels. The sigma is assumed to be constant throughout the image though the background level can vary smoothly. The detector uses a detection filter which assigns a value to a pixel by weighting it with the neighboring pixels. The detection filter weights are set by **setcat(2)** which has a default of unit weight for the central pixel and zero everywhere else; i.e. there is no filter and each pixel is examined individually. The mean and sigma of the background refer to the filtered values. A pixel is detected as possibly being part of an object when its filtered value is a specified number of sigma away from the background. The number of sigma for detection is set in the header by **setcat(2)** and defaults to 2.5 for pixels above the background and 5 below the background. The detection of dark objects is used for the removal of holes in CCD images. Setting the sigma below 2.5 will produce a large number of spurious detections and is not generally useful. Larger values will give only the detection of real objects. Values near 2.5 push the image data to the limits of detection. The only test an object must pass to be included in a catalog by the detector is a minimum size (the number of contiguous pixels including diagonals). The combination of detection filter, threshold sigma, and minimum size determines the limits of the detector.

After the detector has been run the catalog will contain some information about the objects detected in the field. This information is minimal. The photometric data about the object is determined by the program **evaluate(2)**. This program accesses the area description file to determine the detection area and then measures the object from the image file. Many additional parameters are used from the catalog header. Foremost of these is the data-to-intensity relation. The default relation is linear; image data value = intensity. With **setcat(2)** the relation can be entered in a piecewise linear form or calibration spot data can be used to generate a fitted curve. Another photometric parameter is the magnitude zero point;

$$\text{magnitude} = \text{magzero} - 2.5\log_{10}(\text{ Instrumental Intensity })$$

The resulting detection magnitude can be used to limit the objects evaluated using the catalog magnitude limit in the header. Earlier versions of FOCAS used this magnitude limit but normally, for complete detection, the magnitude limit is set very large so that this test is not used.

In addition to the minimum area and magnitude limit **evaluate(2)** uses a number of test to determine if the object is likely to be real. First, dark objects are not evaluated. The various moments of the intensity-position distribution are calculated and checked to see if the values are sensible; for example, if the centroid lies outside the bounds of the field. A final test is called the significance of the object. It checks to see if the average of the brightest nine pixels in a 3x3 grid (the core) is brighter than than the average pixel on the detection isophote by a specified number of sky sigma. Typical values of this significance limit (set in the header) are .1 or .2 though -100 could be used to avoid this test or 0 to form a very weak test. The advantage of a test like this is that it does not depend on the magnitude zero point (being purely instrumental), and behaves similarly for objects detected next to bright objects and for isolated objects.

The remaining steps in making a completed catalog are classification and separating merged objects. Classification is performed by **resolution(2)** using the resolution classifier method described in section 8 and in Valdes (1982). The resolution classifier requires a point spread function (PSF) in the header. The PSF is set in the catalog by **setcat(2)**. It can be entered either manually or from a file. Generally, however, the PSF is not known a priori and must be determined from the star images in the catalog. An automatic method based on the small spread is first moment radii (equation 3f) in bright stars is provided by the program **autopsf(2S)**. Alternatively, the user can manually review the detections in the catalog (**review(2)**), make a catalog of the stars alone (**filter(2)**), form an average star template from the star catalog (**template(2)**), and then enter the template using **setcat(2)** or **setpsf(2S)**.

Another component of the classification system is to define the classification rules which relate the astronomical classification to the resolution parameters **scale** and **frac**. The classification rules are also a part of the catalog header and, therefore, can be set and modified using **setcat(2)**. There is a default set of rules. It is not critical to set the classification rules exactly since any time the classification rules are changed the classifications in the catalog are automatically adjusted (except for those whose classification has been explicitly set by the user) and requires very little time.

The final processing step is the splitting of merged objects using **splits(2)**. Objects can also be split by manual direction in **review(2)**. The splitting algorithm is simply a variant of the original detection. The detector is repeatedly run over an image as the threshold raw integer pixel minimum is raised. The process terminates when the evaluater (the same as in **evaluate(2)**) detects more than one real object (real meaning minimum size, significance level, moment test, etc.) or when the peak of the object is reached. For complete details see section 7. This splitting is repeated on each split fragment for a specified number of levels in the splitting tree (original parent - daughter - daughter - ...). The resolution classifier is automatically run on the split objects.

The catalog has been fully processed at this stage and various analysis programs and manual reviewing of the catalog can be performed. (Note that the manual review and analysis can take place in parallel with the evaluation, classification, and splitting steps.) It is recommended that the brighter multiple, noise, long, and diffuse objects be examined using **review(2)** since these are the objects which often cause problems. Subcatalogs of selected objects can be extracted from the full catalog using **filter(2)**.

One higher level step is available for image data of the same field in multiple passbands or even the same passband. The catalogs from each image file can be matched to form what is called a matched catalog. The matched catalogs have very similar operators (manual review, filter and listing programs) and can produce colors and more reliable discrimination of real objects at the faint limit of the image material.

There are really no specific analysis programs in FOCAS. Analysis of the objects in a catalog is accomplished using either user written programs or, more commonly, using the various stream analysis programs. It is not obvious to a new user how powerful the stream operator tools can be. The idea is to use **filter(2)** to select objects to be studied, **catlist(2)** to extract a stream of ascii data for each object, **opstrm(4)** to apply algebraic transformations, and then either plot the data using the various plotting tools, list the data in tabular form, or apply statistical operations on the stream. The various analysis tools available and examples are given in section 4 of the FOCAS User's Manual.

To save the FOCAS processing data, the catalogs, area description files, etc., tapes can be written using the UNIX archiving programs. In addition, a user can write the catalog data in a machine independent, transportable format using ascii card image records with **tapecat(2S)**.

2. Catalogs

The term catalog as used in this description refers to a data file containing information about the objects in a single image file. There are currently available two catalog formats, a full catalog and a short catalog. The short catalog is simply an abbreviated and compacted version of a full catalog and is used to save space. It will only be of use to users with a lot of data. Another variant of the basic catalog is the matched catalog. A matched catalog is a simple merging of single field catalogs. The format of a matched catalog will be given at the end of this section.

A catalog has two basic parts; header information and object data records. The header is further divided into several parts as described below.

2.1. Catalog Header

The first step in creating a catalog is to supply the minimum information about the image field to be cataloged and the processing parameters required to carry out the detection phase. This information is part of the header. Subsequently, additional information is entered into the header for and by the various FOCAS routines. Some of the header information is entirely internal, defined during the years of development of the system. These quantities will not be discussed here (see the FOCAS User's Manual **sgparm(6)**).

The header information is broken down into the following categories:

Catalog Type
Field information
Detection parameters
Evaluation parameters
The Point Spread Function
Classification rules
Comments and history

All user interaction with the catalog header is through the program **setcat(2)**.

2.1.1. Catalog Type

The first byte of the header is a integer indicating the catalog type. The intention of this is that as FOCAS develops, earlier catalogs can be read in a transparent fashion by the same programs. There are currently two types of catalogs:

Type 1 has the header and full object records described in 2.2.1.

Type 2 has the header and short object records described in 2.2.2.

2.1.2. Field Information

The field information consists of a name or description, an epoch, a two character band (i.e. V), coordinates (in decimal), exposure or integration time, observer, origin or instrument, digitization size in pixels, the digitization origin, and the image field filename. In **setcat(2)** when the field filename is entered some of this information is taken from the image file header. It can then be modified. The only fundamentally important parameter is the filename which all programs accessing the image data need. The rest of the information is for record keeping.

2.1.3. Detection Parameters

There are a number of parameters which control the operation of the automatic detector. Most of these parameters are for tweaking the behavior of the detector and are now essentially never changed. The only parameters the user needs to be aware of are the detection filter, the detection threshold parameter, the minimum detection area, and the sky sigma. As described in the introduction and in the section 5 the detector uses a moving filter to enhance sensitivity to real objects. The detection filter is set by **setcat(2)** and consists of an array of maximum dimensions 5 x 5 which give the weights of the pixels. Thus, for each pixel a filter value is determined as the sum of the surrounding pixels times the detection filter weight.

The detection threshold specifies how many times the sky sigma (determined automatically by the detector from the first few lines of the image) above or below the local sky background a pixel must be to be taken as a possible part of an object. Note that this refers to the pixel values convolved with a detection filter as described below. The values of this parameter would normally be in the range 2.5 (to detect objects at the limits of the image noise) to 5 (to detect only clearly present images, though possibly lint, cosmic rays, etc). A default value of 2.5 above sky and 5 below sky is supplied by **setcat(2)**. The role of the below sky detection is to allow the cleaning of holes or regions of poor sensitivity and to avoid biasing the detector sky determination.

The minimum area is just what it implies. After no further contiguous pixels have been found a candidate object is rejected if the total number of pixels is less that the minimum area and accepted if it is equal to or greater than the minimum area. For a small detection threshold the bigger the minimum area the fewer extraneous noise fluctuations will be detected though large scale enhancements of the background may still be detected. A minimum of 1 pixel with a large threshold can be used to detect bright pixels for the purpose of removing them in a cleaning operation. A default value of 15 pixels is supplied by **setcat(2)** but the actual value to be used will clearly depend on the digitization scale relative to seeing.

The sky sigma is not a user set parameter. It is set by **detect(2)**. This sigma is not the same as the one used in calculating the threshold since it is the pixel to pixel sigma without any filtering. It is used in determining the more accurate sky at detected objects by **evaluate(2)** and in **splits(2)** to select the threshold incrementing level. For determining sky it sets the histogram bin width using the header parameter **skybw** = .1 which is the fraction of the sky sigma per bin. Similarly, in splitting merged objects the parameter **sptbw** = .2 is the fraction of the sky sigma by which the rising threshold is incremented on each pass.

2.1.4. Evaluation Parameters

The evaluation of an object is performed in the intensity domain. The parameters in the header governing the evaluation are the intensity relation, the saturation value, magnitude zero point, magnitude limit, the radius of a circular aperture, a significance threshold, and a transformation matrix. The intensity relation is a piecewise linear lookup table. If there is no table (the default) then the intensity is set equal to the image data value (the raw integer pixel values scaled by **bscale** and **bzero**). There are three option in setting the intensity relation; to enter a set of image data points and their intensity, to specify a file which has the image data points and intensities, or to use calibration spots contained in an image file.

The saturation value is simply a cutoff raw integer pixel value (not intensity) at which a flag is set in an object if it contains pixels greater than this value. Such objects are treated differently by the classifier. This value also terminates the splitting. Though the saturation value can be used as a general flag it is most useful to mark where the intensities become nonlinear, such as caused by saturation.

The magnitude zero point is not critical except, of course,in the data analysis, and controls the conversion of instrumental luminosities into cataloged magnitudes. It has the form

$$\text{magnitude} = \text{magzero} - 2.5\log_{10}(\text{ Instrumental Intensity }) \qquad (1)$$

The zero point is used in conjunction with the catalog magnitude limit to control the evaluation of objects. If the object is below the magnitude limit it is not evaluated. The magnitude zero point can be changed at any time with **setcat(2)** and all subsequent use of magnitudes will incorporate the new magnitude system.

The radius of a fixed aperture, **rfca**, is also an obvious parameter. It is in units of pixels (default = 5.41 pixels). The evaluator will compute the instrumental luminosity of pixels within this distance from the object center (see section 2.2 for the definition of center). There is no use of fractional pixels in this evaluation.

The significance limit governs the minimum significance of a detected object to be evaluated or accepted as a split object. The significance of an object is the ratio of the average core intensity above the average detection isophote intensity to the sky intensity sigma. Normal values of this parameter are .1 or .2 (the default). To eliminate this test use a value of -100.

A transformation relation between the scan coordinates and some other coordinate system can be specified. This transformation sets the values of the **ra** and **dec** entries in the object record from the **xc** and **yc** entries and is initially an identity matrix. It is used to produce astronomical coordinates, to provide a common coordinate system for catalog matching, and to average images (**addfields(2)**). If none of these purposes is required then it can be ignored. The transformation relation is given by

$$\begin{bmatrix} t1 & t2 & t3 \\ t4 & t5 & t6 \\ t7 & t8 & t9 \end{bmatrix} \begin{bmatrix} \mathbf{xc} \\ \mathbf{yc} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{ra}\cdot scale \\ \mathbf{dec}\cdot scale \\ scale \end{bmatrix} \tag{2}$$

The transformation matrix can be set in three ways; using objects labeled as reference points in the catalog (see **review(2)**), specifying four reference points, or entering the matrix.

2.1.5. The Point Spread Function

The PSF is an array of maximum dimensions 15 x 15 giving the relative intensities of an unresolved image. It can be entered into the catalog header by **setcat(2)** and by shell scripts which call it (**autopsf(2S)** and **setpsf(2S)**). The PSF is stored line by line (a line is constant y). The PSF is used by the resolution classifier.

2.1.6. The Classification Rules

The classification rules convert the resolution classifier values of **scale** and **frac** to 5 character class names. An entry consists of the range of magnitudes, the range of scales, and a range of fractions followed by a character string. The number of such rules is arbitrary.

2.1.7. Comments and History

Most of the FOCAS programs which modify the catalog add a comment line to the header. Also **setcat(2)** allows the user to add comments.

2.2. Object Records

The FOCAS automatic detector **detect(2)** extracts regions of contiguous pixels in an image with intensities above some threshold measured relative to sky. These detections are called objects. Such objects may consist of more than one astronomical object. They are separated by **splits(2)**. Both the original detection and objects formed by recursive subdivision are kept in the catalog as objects; objects which are split to smaller objects are generally assigned a class 'm' for multiple.

Each object in the catalog has a data record of identical format. The form of this object record will undoubtedly evolve (as it has from the earlier versions of FOCAS). Hopefully, the catalog manipulation software will decode new types using the first word of the header, the catalog type. There are currently two catalog types; the full object record and the short object record. It is clearly important for the user to know what information about an object is provided by FOCAS in planning to use the system.

2.2.1. Full Object Record

Catalog type 1 consists of the header described earlier and the following object records.

entnum is a sequential entry number assign by the detector.

subent is a subentry number assigned by **splits(2)**. Each place in the number starting from the left is a level in splitting. The usual form of an object entry number printed by FOCAS routines is **entnum**.**subent** For example 3.12 is the second object split from the first object split from the third detection.

class is a 5 character string used to assign a classification to the object. The strings are arbitrary but FOCAS does use certain conventions. In particular, until an object is classified it is called 'u'. An object which is successfully split is called 'm'. The classifier automatically labels saturated objects 's' and certain objects with a large asymmetry 'long' (see section 8). The default classification rules define classes 's' for stars, 'n' for noise, 'g' for galaxies, and 'd' for very diffuse objects. Each user will develop his own set of names for objects as the need arises.

eflgs is a 16 bit word whose bits represent various flags. The currently defined flags are given in table 1. The first column is the variable name and the second column gives the filter option identifier (see **filter(2)**).

Table 1: Object Flags

ATTN	A	General user flag to signal special attention
EDGE	B	Object touches the edge of the field
CLSF	C	The resolution classifier had problems
DARK	D	The object is below sky
EVAL	E	The object was successfully evaluated by **evaluate(2)**
FRCD	F	The object has had a forced classification
SIZE	L	Object exceeds the current limits of FOCAS
PEAK	P	There are saturated pixels in the object
REFP	R	This object is a coordinate reference point
SNGL	S	The object not split at any level by **splits(2)**
BCOR	-	Bright star correction applied to fix saturation
ADDO	-	Artificial object added to the field

sbr, **nsbr** and **ssbr** are the evaluated instrumental sky brightness at the object, the number of pixels used in forming the sky value, and the instrumental intensity sky sigma.

area and **tarea** are the detection or isophotal area and the grown area (see Section 6.2) in pixels.

ispht is the average intensity along the detection boundary.

xavg and **yavg** are the average x and y widths.

xc and **xy** are the midpoint of the maximum x and y dimensions of the detection area before evaluation and the center of the 3 x 3 grid with the greatest luminosity within the detection area (called the core luminosity) after evaluation.

ra and **dec** are the coordinates produced by the transform matrix in the header.

Li, **Lc**, **Lfca**, and **Ltotal** are the various integrated luminosities of the image. They are, respectively, the isophotal luminosity (luminosity within the detection area), core luminosity (maximum luminosity of a 3 x 3 grid within the detection area), the luminosity with the circular aperture defined by **rfca** in the header, and the luminosity within the grown area. The total luminosity is not actually a total luminosity. The luminosities are in instrumental units.

mag is the magnitude corresponding to **Li**. It is calculated with the magnitude zero point as defined in equation 1.

icx, **icy**, **ixx**, **iyy**, **ixy**, **ir1**, **ir3**, and **ir4** are the various intensity weighted moments of the detection area. They are defined by the following equations:

$$\textbf{icx} = \sum x I(x,y) \ / \ \sum I(x,y) \tag{3a}$$
$$\textbf{icy} = \sum y I(x,y) \ / \ \sum I(x,y) \tag{3b}$$
$$\textbf{ixx} = \sum x^2 I(x,y) \ / \ \sum I(x,y) \tag{3c}$$
$$\textbf{iyy} = \sum y^2 I(x,y) \ / \ \sum I(x,y) \tag{3d}$$
$$\textbf{ixy} = \sum xy I(x,y) \ / \ \sum I(x,y) \tag{3e}$$
$$\textbf{ir1} = \sum r I(x,y) \ / \ \sum I(x,y) \tag{3f}$$
$$\textbf{ir3} = \sum r^3 I(x,y) \ / \ \sum I(x,y) \tag{3g}$$
$$\textbf{ir4} = \sum r^4 I(x,y) \ / \ \sum I(x,y) \tag{3h}$$

where $I(x,y)$ is the intensity above sky (i.e. the raw intensity - **sbr**).

cx, **cy**, **xx**, **yy**, **xy**, and **ir1** are the unweighted moments of the detection area defined by:

$$\textbf{cx} = \sum x \ / \ area \tag{4a}$$
$$\textbf{cy} = \sum y \ / \ area \tag{4b}$$
$$\textbf{xx} = \sum x^2 \ / \ area \tag{4c}$$
$$\textbf{yy} = \sum y^2 \ / \ area \tag{4d}$$
$$\textbf{xy} = \sum xy \ / \ area \tag{4e}$$
$$\textbf{r1} = \sum r \ / \ area \tag{4f}$$

fitxc, **fityc**, **scale**, **frac** and **prob** are parameters relating to the resolution classifier. **fitxc** and **fityc** are the x and y position in the 3 x 3 area about **xc** and **yc** which give the best maximum-likelihood fit of the PSF to the image. **scale** and **frac** are the scale and fraction parameters of the best fit resolution template (see Section 8). The last parameter **prob** is not currently used but could be used to contain the classification probability which can be obtained from the resolution classifier.

arpos is the offset in the area description file for the object area description.

2.2.2. Short Records

To reduce the size of the catalogs for analysis the rarely used object parameters are removed and some of the floating point quantities are packed into short 16 bit integers. The program to do this is called **shortcat(2)**. All programs reading the catalogs internally convert the short records back to long records (with corresponding truncation inaccuracy) and act as if the catalogs have full records. Thus, **catlist(2)** will print any entry in the full record but the value may be meaningless if it is not present in the short record. The short record has the following entries.

entnum and **subent** have the same meaning as the long record except that **subent** is truncated to 16 bits. Thus, some split objects may give odd subentry numbers.

class and **eflgs** are as in the long record.

xc and **yc** are short integers of the long record form and cannot be subpixels.

magi, **magc**, **magfca**, and **magtotal** are short integers equal to 250 $\log_{10}$(instrumental luminosity) where the luminosities are the detection luminosity **Li**, the core luminosity **Lc**, the circular aperture luminosity **Lfca**, and the total luminosity **Ltotal** described earlier.

icx, **icy**, **ixx**, **iyy**, **ixy**, **ir1**, **cx**, **cy**, **xx**, **yy**, **xy**, and **r1** are not packed and are exactly the same as the long records.

scale and **frac** are the resolution parameters multiplied by 100 and packed into short integers.

2.3. Matched Catalog Header and Records

The program **match(3)** creates a composite catalog from 1 or more single field catalogs with objects being matched by **ra** and **dec**. The matched catalog data structure is a simple variation of the single field catalogs. The header begins with a short integer giving the number of catalogs forming the matched catalog. Following this are that many single catalog headers in exactly the same form as that described above. The following object records begin with a two byte structure, the first byte uses the 8 bits to indicate which catalogs have objects which matched. There will be at least one bit set. The second byte gives the matching weight (see Section 9). Following these two bytes there are as many object records, in exactly the same form as the single catalog entries, as the number of bits set in the matching indicator byte.

It is possible that someday the distinction between single and matched catalogs will vanish.

2.4. Derived Object Parameters

There are a number of useful quantities which can be derived from the entries in the object record. Some of these, notably the magnitudes corresponding to the instrumental luminosities and the radial moments, can be simply extracted using **catlist(2)**. The radial moments extracted by **catlist(2)** differ from the preceding definitions in that the appropriate roots are taken. Thus,

$$\mathbf{ir2} = (\mathbf{ixx}+\mathbf{iyy})^{1/2} \tag{5a}$$

$$\mathbf{r2} = (\mathbf{xx}+\mathbf{yy})^{1/2} \tag{5b}$$

$$\mathbf{ir3}_{\text{extracted}} = \mathbf{ir3}^{1/3}_{\text{cataloged}} \tag{5c}$$

$$\mathbf{ir4}_{\text{extracted}} = \mathbf{ir4}^{1/4}_{\text{cataloged}} \tag{5d}$$

Other quantities can be obtained using **opstrm(4)**. The following derived parameters deal with the shape of the image and are derived from the second moments. These shape parameters are the position angle and asymmetry of the the objects. The position angle is given by

$$\text{P.A.} = \tan^{-1}(2\mathbf{ixy}/(\mathbf{iyy}-\mathbf{ixx})) \tag{6a}$$

and the asymmetry by

$$\text{asymmetry} = ((\mathbf{iyy}-\mathbf{ixx})^2+(2\mathbf{ixy})^2)^{1/2}/(\mathbf{ixx}+\mathbf{iyy})\ . \tag{6b}$$

The same relations also hold for the unweighted moments using **xx**, **yy**, and **xy**. The asymmetry is used in classifying objects as described in section 8.

3. Image Files

An image file is associated with each catalog. FOCAS has routines for creating, maintaining, and modifying image files. These files have a standard structure which is described in this section. This structure has two parts; a header and the image data.

3.1. Image Header

The image header is a 512 byte record. This size is limiting in comment space and may change later. The philosophy of the header structure is strongly modeled on the FITS keyword system. The FITS system is described in *FITS: A Flexible Image Transport System* available through the Computer Support Office. The header parameters are as follows.

blksiz reflects the FITS tape block size. The FITS standard is 2880.

bitpix is the number of bits per pixel.

bytepix is the number of 8 bit bytes per pixel.

naxis is the number of dimensions in the image. This must be 2 in FOCAS.

naxis1 is the number of pixels in the x dimension.

naxis2 is the number of lines in the y dimension.

dflgs and **maptyp** are display flags and display map type. They are not currently used.

bunit, **ctype1** and **ctype2** are integer codes to the pixel intensity and dimension units. They are not currently used.

maxpxl and **minpxl** are the maximum and minimum pixel values in the image.

bscale and **bzero** are the scale and offset to be applied to the integer pixel values to convert them to data values.

dscale and **dzero** are a scale and offset used to convert the integer pixel values to 8 bit display values.

crpix1, **crval1**, **cdelt1**, **crota1**, **crpix2**, **crval2**, **cdelt2**, and **crot2** are the coordinate reference pixel location (rpix1 and rpix2), the coordinate values at the reference pixel (rval1 and rval2), the coordinate pixel spacing in the two dimensions (delt1 and delt2), and the rotation angles of the two dimensions to the standard coordinate system given by **cytpe1** and **ctype2** (rota1 and rota2).

blank is the pixel value assigned to pixels with indefinite values.

simple is a two character string used to identify the type of pixel data type. The standard value is ’T’ to denote integer pixels and has the same meaning as in the FITS format definition.

object is a 32 character string giving the object or field name.

origin is a 32 character string giving the origin of the image.

date is a 32 character string giving a date for the image.

observer is a 32 character string giving the observer or owner of the image.

nextpic is a 32 character string giving the filename of another image in a sequence.

comment is a 256 character string containing history or comment information.

3.2. Image Data

The raw integer pixel values follow the 512 byte header. The pixel data types allowed are 8 bit unsigned integers and 16 and 32 bit signed integers. It is also possible to represent floating point data but it is only used for the picture arithmetic program. The pixel data is stored in x,y form where x is defined to be the most rapidly varying index, i.e. **naxis1**.

4. Area Description File

With each catalog and image file there is an associated area description file containing one entry per object. The entries contain detailed information about the location and shape of the detection area. All references to the image data for an object require this information. It is also used to generate the isophote and the total area boundary. The area file is created by **detect(2)** and added to by **splits(2)**. The user never addresses this file directly, the filename is contained in the catalog header. The structure of this file is of no relevance to the user and will not be described in the FOCAS documentation.

5. Automatic Detection

This section describes the automatic detection algorithm in detail. Variable names will be used in the description and are those actually used in the FOCAS detector. There are two parts to the type of automatic detection used by FOCAS; determining the local background in a fast and scan line oriented fashion and, therefore, defining what is not background and assembling the individual pixels which are not background into contiguous objects. This algorithm works entirely in the pixel data space whether that be density, intensity, counts, etc. The discussion will refer only to the detection of objects with greater pixel values than the background. There is a complementary process proceeding in parallel to detect objects below the background though with an independent threshold value.

5.1. Background Determination

The basic assumptions in the automatic detector are that the background varies smoothly without abrupt level changes over a scale which is many tens of pixels and that the sigma of the background is constant over the entire image being studied. If the first assumption is violated then false detections will be made to the point that an abrupt level increase will cause the remaining part of the image to be considered one object. If the second assumption is violated then the detection sensitivity, which is defined in terms of the sky sigma, will vary.

A user defined moving filter is used to increase the sensitivity to extended objects and minimize the pixel to pixel sigma. This signal filtering technique gives maximum sensitivity to images having the shape of the filter. The filter is defined in **setcat(2)** by specifying the size of the filter **fx** and **fy** (maximum dimensions 5 x 5) and the array weights **dfltr**[y][x]. The filter weights are floating point quantities though usually weights are simple integers. The dimensions of the array must be odd for a symmetric filter. The pixel data can be used without smoothing by specifying a filter of dimensions 1 x 1 which is the initial filter set by **setcat(2)**. Whenever, the detection routine calls for a pixel value it calls a routine **pixel** which multiplies the neighboring pixels by the filter weights and sums the products to produce a final filter value. The pixel values are not normalized by the sum of the weights. For the remainder of this discussion, except when noted, the pixel values refer to this weighted sum.

For filters of dimensions greater than 1 the detection ignores a band at the edge of the image of width equal to half of the filter size. In other words, the filter never attempts to use pixels outside the image data. The effect of this restriction is seen when looking at the isophotes of EDGE objects.

The cost of allowing a general detection filter specified by the user is computation speed. The calculation of the filter pixel values is by far the major computation step since there are **fx*****fy***(number pixels in the image) operations. If very large images (and many of them) are to be processed it can be advantageous to make a special detection program in which the desired filter is put into the code in an optimally efficient manner.

The first step after file initialization and memory allocation is to determine the initial background values and background sigma from the first **fy** lines of the image. There is one background value for every SKY pixels along a line where SKY has a value of 8. The **intsky** routine first computes the average of every pixel in the initial line and sets all the background values to this average. It then iterates to remove pixels which are part of objects and to get the variation in the background on the SKY pixel scale. The iteration proceeds by lowering a threshold value from the maximum pixel in the image in 100 equal divisions of size max = (maximum pixel - background average)/100. At each iteration i and for each background point l the threshold is a0 = sky[l] + th0 + max * (100-i). The threshold constant th0 is determined at the end of each iteration from the sky sigma described below (th0 = 0 for the first iteration). Pixels above the threshold are ignored and pixels below the threshold are used to update the sky value.

The sky updating algorithm is the same as used in the detection process. After the SKY pixels have been evaluated the background is updated by equation A if the number of pixels found to have been background is greater than **nupdte** = 3 with average value <s> and by equation B otherwise.

$$\text{sky}[l](new) = \text{sky}[l](old) + \mathbf{sa} * (<s> - \text{sky}[l](old)) +$$
$$2 * \mathbf{sb} * ((\text{sky}[l+1] + \text{sky}[l-1])/2 - \text{sky}[l](old)) \tag{7a}$$

$$\text{sky}[l](new) = \text{sky}[l](old) +$$
$$2 * \mathbf{sb} * ((\text{sky}[l+1] + \text{sky}[l-1])/2 - \text{sky}[l](old)) \tag{7b}$$

where **sa** = 0.06 and **sb** = 0.02. The boundary values of sky used in the update are given by boundary extension; sky[0]=sky[1] and sky[max+1]= sky[max] where sky[1] and sky[max] are the sky values at the limits of the image.

At the end of each iteration the detection threshold value th0 is updated to be **thrln0** * (the sigma of the detected background points about the local background). The parameter **thrln0** is the primary user supplied value for determining the sensitivity of the detector. As can be seen it determines the threshold in terms of the sigma of the background and normally has values between 2.5 and 5.

The final iteration is i = 100 so that a0 = sky[l] + th0. At this point the value of **skyhw** = sigma / (square root of the total weight of the filter) is set in the catalog header. This value is the unfiltered background sigma and is used in later processing steps. A comment line is written in the catalog header giving the detection filter convolved sigma used in the detector as well as the thresholds.

After determining the initial background levels the detection process proceeds line by line to the end of the image. For each line the detection filter is moved along the line and a threshold is determined by a0 = sky[l] + th0. If a pixel value is greater than this value and if the previous pixel was not above the threshold, the position of the pixel along the line marks the start or left end of a run length code (rlc). As long as the succeeding pixels are above the threshold the right end of the rlc is incremented. When a pixel is found no longer to be above sky or the end of the line is reached the rlc is passed to the object assembling routine.

In parallel with this process the sky is updated using equations 7a and 7b above. After every SKY line the background sky values sky[l] are divided by the total weight of the filter and written to a sky image file which the user can review later with **display(5)**.

5.2. Object Assembly

The input to the object assembly routine are the run length codes of pixels found to be above the detection threshold. During the course of processing a line the rlc is added to a linked list if it overlaps a rlc from a previous line. An overlap is defined to include diagonally adjacent pixels. If no previous rlc overlaps the new rlc then a new entry is created in the set of linked lists describing the growing object. At the end of a line the linked lists are searched to see if any did not have a rlc from the current line added to them. If this is the case then the object is complete and it is passed to an area evaluation routine.

The area evaluation makes the area description data structure stored in the area description file. The area description data structure contains information about the x extents of the object, its starting y value, the number of lines in the object, and the rlcs describing the detection isophote. A rectangular region containing the object is defined by putting a buffer region around the maximum x and y points of the object. The buffer width is given by **buf** = 10 pixels. There are, naturally, restrictions due to the edge of the image and if the object borders the edge of the image an EDGE flag is set. Also if the object exceeds the maximum number of rlcs (MXRLC = 3200) then the SIZE flag is set and the area description is truncated after than many rlcs. Further restrictions are the maximum x or y lengths of the rectangular region (MXLN = 800) and the total area of the rectangle (MXSZ = 400 x 400). If the length limits are exceeded the oversize dimension of the rectangle is set to MXLN and if the final area limit is exceeded the sides of the rectangle are reduced in proportion to give an area smaller than MXSZ. The SIZE flag is again set for such objects and a diagnostic message is printed.

For very large objects the rectangle limitations may actually exclude significant parts of the object. No object, however, has ever been found to exceed the MXRLC degree of complexity.

The last step after the bounding rectangle has been defined is to measure the detection area of the object, i.e the number of pixels within the rlcs. If the area is less than the user specified minimum area **minarea** the object is discarded. If the object has a greater area then it is accepted as an entry in the catalog. A sequential entry number **entnum** is assigned, a position **xc** and **yc** is given by the average of the maximum x and y extents of the object description, the detection area **area** and total area **tarea** (see next paragraph) is recorded, the object type is set to ’u’, and the object record is written to the catalog while the area description is written to the area description file.

The total area is determined by first filling in any x or y concavities in the isophote shape and then adding rings (following the isophote) around the object until the area exceeds the detection area by a specified fraction **grow** = 2.

Once all the objects have been either rejected or recorded in the catalog from a given ending line the object assembly routine returns to the detection phase for the next line.

6. Evaluation

The catalog entries describing the objects found by **detect(2)** have only a limited amount of information. In particular, there are no photometric quantities or a physically reasonable position. In this section the various algorithms used to evaluate an object are described.

6.1. Sky Determination

The sky background determined by the the detector is not highly accurate and a more sophisticated method is used when a object is evaluated. The bounding rectangle containing the object is read from the image file. The bounding rectangle includes a region of at least **buf** = 10 pixels around the object detection boundary. The outer **skwdth** = 6 pixels around the rectangle are used to determine the mean sky level.

First the mean value of all the sky pixels (in raw data units) is determined. Next a histogram of 512 bins of bin width = **skybw** * **skyhw** is formed about the mean. The quantity **skyhw** is the raw background sigma determined by the detector. The bin width parameter is **skybw** = 0.1 so that the bin width of the sky histogram is 10% of the sky sigma. The peak of the histogram (in raw pixel values) is found and the number of pixels **nsbr**, the mean **sbr**, and the standard deviation **sbr** of the intensities of the pixels within 2 * **skyhw** of this peak are calculated. If the number of sky pixels is less than 10 the evaluation of the object fails.

6.2. Positions, Shapes, and Luminosities

If the sky determination is successful then the first two x and y intensity weighted and unweighted moments of the object within the detection isophote are determined. The moments are defined in section 2.2.1. The pixel intensities are sky subtracted and the moments are normalized by the isophotal luminosity, which is the zeroth moment. If the isophotal luminosity **Li** is negative the evaluation fails at this point. Also during this step if any pixel's raw value exceed the user specified saturation level, **satr**, the saturation flag PEAK is set.

The core of the object is found by examining all pixels within the detection area using a 3 x 3 box filter; i.e. the sum of intensities within a 3 x 3 grid centered on the pixel in question. The core is that point which has the maximum luminosity **Lc** and its position is recorded as **xc** and **yc**. This position is taken as the object center in all further evaluations rather than the moment centroids. The second central moments are determined relative to **xc** and **yc**.

The total luminosity **Ltotal** within the total area **tarea** is measured next. The total area is determined by first filling in any x or y concavities in the isophote shape and then adding rings around the object until the area exceeds the detection area by a specified fraction **grow** = 2. If the total luminosity is negative then the evaluation fails. Also if both the isophotal magnitude and total magnitude exceeds the user specified magnitude limit **maglim** then the evaluation fails.

The first four intensity weighted radial moments and the first two unweighted radial moments are determined relative to the peak position. If any of the moments yield negative values then the evaluation fails.

An aperture luminosity **Lfca** is measured centered on the peak. If a pixel center lies within **rfca** of the peak its intensity is included in the total luminosity.

The mean intensity (sky subtracted) of the pixels lying on the detection area boundary is measured next. This value is recorded as **ispht** and is used in the significance test. The significance test compares the difference between the mean peak intensity **Lc**/9 and the isophotal intensity **ispht** to the sky sigma of the average of 9 pixels **sbr**/3. Thus, the significance of an object is

$$\text{significance} = (\mathbf{Lc}/9 - \mathbf{ispht})) \;/\; (\mathbf{sbr}/3). \tag{8}$$

This significance is tested against the significance limit **sig** defined by the user. If the significance is less than **sig** the evaluation fails.

The average x width and y width of continuous segments of the detection area in x and y form the quantities **xavg** and **yavg**. These segments are just like run length codes.

The final evaluation step is to set the reference coordinate values **ra** and **dec** from the peak position **xc** and **yc** using the transformation matrix in the catalog header. Initially this matrix is the identity matrix. The evaluation flag EVAL is then set and a successful evaluation is returned.

7. Splitting

The separation of objects which are merged at the detection isophote is very straightforward. Simply, the object is examined at brighter isophotes for a separation into two or more components. How this is done is described in greater detail in this section.

Only objects with the evaluation flag EVAL and without the large flag SIZE are examined. The image of the object is read from the image file using a rectangle which is just large enough to contain the detection area, thus minimizing the number of pixels the detector has to examine. The pixels exterior to the detection area are set to a value less than the darkest pixel in the object. Like the detector the splitting algorithm works entirely with the raw image pixels. A step size for increasing the isophote is determined by taking the pixel to pixel sigma determined by **detect(2)** and recorded in the header as **skyhw** and multiplying it by **sptbw** which is set at 0.2.

A loop is entered incrementing a detection threshold level. Note that this differs from **detect(2)** in that there is no filtering or reference to sky. The threshold level is explicitly given as a raw image pixel value. Otherwise the detection algorithm is the same; at each line run length codes of pixels above the threshold are found and passed to the object assembly routine described in section 5. The assembly takes the rlcs which are then added to existing linked lists if they overlap or, otherwise, a new object list is created.

The object assembly in **detect(2)** produces an object if the area is greater than **minarea** and the object is recorded in the catalog. In the splitting routine called by **splits(2)** and **review(2)** there are additional steps at this point. If the detected object is the first one then nothing is done. If the end of the original object image is reached by the detector with only one object found then the next threshold value is calculated and the next pass in the loop begun. However, if a second object is found then the one with the smaller area is evaluated as described in section 6. If the object fails the evaluator (which includes the significance test) then it is discarded. In addition to discarding the object area definition the pixels forming the object are set to the background level. This eliminates the redetection of the remaining pixels in succeeding passes.

As more objects are found all but the one with the largest area is evaluated. The reason for not evaluating each object immediately is that the evaluation requires time and if only the original object is found at each new detection level then repeatedly evaluating the object can be extremely time consuming. If, when the end of the image is reached and more than one object has been found, the unevaluated object is evaluated. If there is still more than one object which is considered to be real by the minimum area and evaluation criteria then the objects are recorded as components of the original object. The original object is reclassified as 'm'. In **review(2)** the objects are placed at the end of the working catalog and are unclassified. In **splits(2)** the new objects are placed after the parent and the resolution classifier is run on the new object images.

The threshold incrementing loop terminates when the object is split or the threshold exceeds the saturation value given by **satr** as set in the catalog header by the user with **setcat(2)**. If the object was not split at this point then the single object flag SNGL is set. In **splits(2)** the new objects are immediately searched for splitting again if the generation level of the objects is less than that set by the user as an argument to the program. In **review(2)** the user can repeatedly split an object as desired. *The splitting technique is visually demonstrated by saying yes to the display query when splitting is performed by* **review(2)***.*

8. Classification

Classification is accomplished using the Resolution Classification method described by Valdes (1982). The paper is also available through the Computer Support Office. Briefly, the Resolution Classifier fits to each object a series of templates derived from the image point spread function by broadening to various scales. The scale of the best fitting template is then a measure of the degree of resolution of the object and the classification is made from this scale value; scales near that of the PSF are stars, larger resolved scales are galaxies, and very large scales are diffuse sky enhancements detected as objects. Other templates have a stellar component plus a broader component. The fraction of broad component is also used as part of the classification scheme. This template fitting approach is based on Bayesian classification meaning that on the basis of the statistics of the pixels in the object the fit measures the likelihood that the observed image with noise closely resembles the noise-free model template. One then choses the template with the greatest likelihood of agreeing with the observation. Bayesian classification is optimal for the templates examined and makes full use of the image data. The resolution classifier uses the additional criterion that even if a template is not a true model of a galaxy's shape, if it is resolved (does not agree with the PSF better than a broader PSF) then it cannot be a star and must be a galaxy.

There are two components to a Bayesian classifier; the noise-free photon counts of a model image and the statistical noise properties of the detector. The Resolution Classifier takes as a model for the ideal image

$$\langle N_i \rangle = N_o t_i + N_s \tag{9}$$

where N_o is a luminosity scale factor, N_s is the mean sky background, and the quantities t_i form the template defining the shape of the object. The templates, as mentioned earlier, are based on the PSF template s_i. The templates are parameterized by

$$t(r_i) = \beta s(r_i/\alpha) + (1-\beta)s(r_i) \tag{10}$$

where r_i is the position of pixel i, α is a broadening scale parameter (or sharpening parameter if $\alpha<1$), and β is the fraction of broadened PSF. Thus, the templates consist of a stellar core and a fraction β of a broader component of scale α. *Note that these models assume a magnitude independent shape for unresolved objects. Thus, the classifier is not applied to objects whose saturation flag PEAK has been set. These objects are automatically classified stars.*

The model for the photon counting noise depends on the detector. Specifically, the detective quantum efficiency (*DQE*), which is the ratio of the output signal-to-noise to the input photon signal-to-noise as a function of the signal, characterizes the detector. In Valdes (1982) a model for photographic detectors was used. For the purpose of developing an algorithm this dependence on the detector is ignored. It is only important for objects whose intensities span a large range (for faint objects the *DQE* can be considered a constant) and the absolute level of the *DQE* determines the absolute probabilities. The resolution classifier only determines the template of maximum probability (or maximum likelihood) and does not determine the classification probability. The point of this discussion is that the the likelihood function just be approximately based on the Poisson nature of the noise. The function derived in Valdes (1982) is

$$\lambda = \ln P(N_i \mid t_i) = \sum_i \left[N_i/(N_0 t_i + N_s)(1 - ln N_i/(N_o t_i + N_s)) - 1 \right] \tag{11}$$

where λ (called the likelihood) is the log of the probability that the observed N_i come from an object corresponding to template t_i.

The classification algorithm in **resolution(2)** finds the template t_i parameterized by α and β (called **scale** and **frac** in the catalog and hereafter) which maximizes the likelihood. The remaining part of this section describes the steps of the algorithm. When the resolution templates are first required the point spread function template is read from the catalog header. If this template is not set then the classifier stops. The template can be any size up to 15 x 15. The resolution templates are then formed.

It is clearly impossible to test all possible values of **scale** and **frac**. Instead the resolution classifier forms the set of templates given in table 2.

Table 2: Resolution Templates Used in FOCAS

alpha	beta	
p1	----	Single pixel
p2	----	Five pixels of uniform intensity
p3	----	Nine pixels of uniform intensity
0.7-2.0	1	Broadened stars in steps of 0.02 in **scale**
2.0	0-1	Stellar nucleus plus fraction broadened star in steps of 0.05 in **frac**
2.5-9.5	1	Broadened stars in steps of 0.5 in **scale**
100.0	1	Uniform enhancement of background sky

The first three templates of table 2 are explicit tests for noise objects. Though the FOCAS detector generally requires an image to be of a minimum size it is often possible for bright pixels to be surrounded by an enhanced background sufficient to create an object of the necessary area. In the catalog the templates p1, p2 and p3 are assigned the **scale** values of 0.1, 0.2, and 0.3 respectively. The last template consisting of the PSF broadened by a factor of 100 is a test for noise objects of a different kind; broad patches of enhanced sky. Most objects have scales in the range 0.9<**scale**<2 which encompasses the transition from unresolved to clearly resolved. Thus, this range of **scale** is investigated with a fine grid of templates. The two component templates only use a **scale** of 2. Study of a full two parameter space showed that there was no gain in discrimination and the interpretation of templates with a broad component of scale nearly that of the PSF was difficult. Finally, the transition between truly broad and diffuse galaxies and large scale sky fluctuations is covered coarsely with the broad templates between 2.5 and 9.5.

These templates have dimensions equal to that in the PSF. The templates are fit to the intersection of this area and the object detection area. This avoids use of the poor signal-to-noise parts of the image lying outside the detection isophote in faint and small objects. It is important in fitting the templates to have the correct centering of the template on the object image. The fitting center is found by maximizing λ in the 3 x 3 are around the core center **xc** and **yc** for the template of **scale** = 1. This is like finding a center by fitting the PSF and gives the best (integer pixel) center for stars. The value of this fitting center is recorded in the catalog as **fitxc** and **fityc**.

For each fit the value of λ is maximized with respect to the luminosity scale factor N_o. This is done using a Newton's method to find the zero of the derivative of equation 11. Failure to converge in this step results in an error message. It is common for one of the sharp noise templates (p1-p3) to fail to converge so the classifier continues with the other templates. When one of the other template fits fail, however, the classifier leaves the value of **scale** and **frac** unset and proceeds to the next object. The search through **scale-frac** space uses a binary search rather than fitting all of the templates. After the noise templates are fitted the **frac**=1 models ordered by **scale** are searched by the binary tree method for a maximum λ. Finally, the **scale**=2 two component models ordered in **frac** are searched. The maximum likelihood **scale** and **frac** are then recorded.

The binary search technique significantly speeds up the template fitting but depends on there being a smooth relation between λ and **scale** or **frac**. However, when the λ values are found to be poorly behaved because of gridding effects, nonmonotonic light profiles, or very faint images, an exhaustive search of the templates is made for the maximum λ. In this case the CLSF flag of the object is set.

Once the values of **scale** and **frac** are determined a class name is assigned using the classification rules in the header and certain empirical rules. These rules are, in order of precedence:

1. Objects with a forced (FRCD flag) classification are not reclassified.
2. Objects classed as 'm' multiple are not reclassified.
3. Saturated objects are classed as 's' stars.
4. Objects with asymmetries > 0.9 (see section 2.4, equation 6b) are classed 'long'.
5. The classification rules in the header are used.
6. If **scale** and **frac** are not set the classification is unchanged.

The classification rules define the ranges of magnitude, **scale**, and **frac** for a given class name. There can be as many such rules as desired. As a guide the typical scatter in star scales is 0.8 < **scale** < 1.2. A plot of **scale** vs. magnitude can be instructive for **scale** between 0 and 2.

9. Matching

The structure of a matched catalog was described in section 2.3. The matching algorithm in **match(3)** is described in this section. The matching is done by **ra** and **dec** coordinate values in the object records of the single field catalogs. The first step in the matching procedure is to order the the catalog object records in **dec** using a system call to **catsort(2)**. The first input catalog to **match(3)**, after sorting, is simply copied to the matched catalog while removing the multiple class objects. Subsequent input catalogs go through the following matching procedure.

The candidates for matching in the matched catalog are kept in a wrap-around buffer of 250 entries. Each object in the single (**dec** ordered) catalog which is not classed multiple is read. If the **dec** value is less than **delta** = 20 units from the first matched catalog entry in the buffer it is immediately recorded as unmatched. All matched catalog entries in the buffer with **dec** values less than **delta** relative to the single object are recorded unchanged and new entries are read into the buffer. If the last object in the matched catalog buffer is not greater than **delta** relative to the single object then a diagnostic warning message is printed that there is insufficient buffer room for full matching.

The single object is matched against all matched catalog entries which have **dec** values in the interval **-delta** to +**delta** relative to the single object **dec**. The minimum separation relative to all components of all the matched catalog entries is determined and if the minimum separation is less than **delta** a possible match is found. If the matched entry already has a component from the single catalog and if the new object has a smaller separation then the new object is added to the matched entry and the other object from the single catalog is retested to see if matches any other matched catalog entry. This insures that the closest objects are matched even when several objects are nearby. The new matched entry is finalized when objects from the single catalog advance in **dec** beyond **delta** and the buffer is updated.

Whenever a new object is added to a matched entry the matching weight of the entry is reevaluated. The matching weight is defined as:

$$\text{weight} = 254\,(1-d)/delta + 1 \qquad (12)$$

where d is the average separation between all pairs of components. The weight has a maximum of 255 (8 bits) and unmatched objects have a weight of 0.

There is a caution to note about the matching process. When bright objects in overlapping fields turn up as unmatched this means either the objects do not fall in the overlap or the splitting of merged objects was not the same in both catalogs. For example consider a close binary star with two bright components. If in one catalog the objects were split and in the other the split failed then one of the split components, though bright, will be unmatched. Use of **mreview(3)** will generally solve the mystery of unmatched bright objects.

10. References

1. **FOCAS User's Manual**, 1982, Kitt Peak National Observatory, Computer Support Office, Tucson, Az.

2. Valdes, F., 1982, *The Resolution Classifier*, in **Instrumentation in Astronomy IV**, S.P.I.E. Proceedings, Vol. 331.

3. Wells, D. C. and Greisen, E. W., 1979, *FITS: A Flexible Image Transport System*, Kitt Peak National Observatory, Computer Support Office, Tucson, Az.