FOCAS User's Guide

Francisco Valdes

IRAF Group - Central Computer Services
National Optical Astronomy Observatories††
September 1993

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. FOCAS may be used on any type of astronomical image but it was primarily designed for the optimal detection of small faint objects and their classification as stars, galaxies, or noise. It is largely automated for the measurement of large numbers of objects for various statistical studies and so the photometry, astrometry, and classification are of moderate accuracy. FOCAS is not intended for very crowded stellar fields where point spread function fitting methods are more appropriate, or for large bright objects where surface brightness methods apply.

There are roughly 100 FOCAS programs which provide a variety of image manipulation, catalog creation, display, and analysis functions. This guide limits itself to just the few programs which provide the basic processing from an image to a catalog or matched catalog and identifies some typical simple analysis operations. The user's guide is written as a cookbook showing typical examples. The example image used, in case one wants to follow along, is the "galfield" example produced by the IRAF artificial data task **mkexample**,

```
fo> mkexample galfield galfield
Creating example galfield in image galfield ...
fo> minmax galfield force+ update+    # Not required after V2.10.2
```

The examples are shown as being run from the IRAF command language. However the commands are generally identical if run from the Unix shell.

There are two configurations for FOCAS. One is independent with its own image format. The other uses some IRAF libraries which provide access to IRAF images (currently only the .imh format). In this case FOCAS is usually defined as part of the IRAF environment so that one may mix IRAF and FOCAS operations. However, the FOCAS programs are independent Unix programs which are simply known to IRAF as *foreign* tasks. Thus one must have the FOCAS commands available before starting the IRAF **cl** by defining the appropriate Unix path. As a quick check of this try the following from the Unix shell.

```
% setcat ^
Set or modify catalog header
  usage: setcat catalog
    Arguments: catalog - catalog to be set or modified
    Input: prompted with default values given in parenthesis.
    A < cr > accepts default value.
```

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

The other thing to check before starting if you are using a remote machine to run FOCAS and want to display images to your workstation console, provided that FOCAS has been configured to use IRAF networking, is that the `node` environment variable is defined. For example,

```
% printenv node
puppis
```

Note that either **imtool** or **SAOimage** may be used. The display server should be started before attempting to display an image.

2. The Data

The digital image data is usually created by a FITS reading program. In the standalone version one uses the FOCAS FITS reader called **rfits** and in the IRAF version one uses the IRAF FITS reader also called **rfits**. *It is important to note that FOCAS works internally only with integer pixel values even if the image is an IRAF real datatype image. This is for historical reasons. Real data is truncated upon input to FOCAS. If the data range is very small then the image data should be multiplied by a large enough factor so that the sky noise has a level of at least 1.*

The first thing to do is to display the image with the FOCAS display program **display** (standalone) or **fdisplay** (IRAF). While the IRAF **display** program may be used with IRAF images you still must use the FOCAS display program at least once. This is because this program not only displays the image but it sets the range of image intensities to map to the range of the display. It does this by asking for the display range in the image data and then storing the values in the image header. The other display oriented programs, **review** and **mreview**, automatically use these values. *If they are not set you are likely to get either an all white or all black display.*

The following example shows displaying an image and setting the display range.

```
fo> fdisplay
D* w galfield
Object: Example artificial galaxy field
Display range (0 255):
D* c
Averaging box size (1):
Position 209 237 , Average = 378 , Standard Deviation = 0
D* w galfield
Object: Example artificial galaxy field
Display range (0 255): 350 550
D* q
```

This program is generally interactive with the prompt `D*`. If one forgets and types the name of the image as a command line argument then an error will result. In the example above, the first time the display is loaded the default values were used. The display may appear totally white or black (depending on the greyscale settings and the data range). The 'c' command asks that the cursor be read and reports the data value and standard deviation. After a few checks one will have an idea of the sky level and another 'w' command will yield a good display. The idea is that you may iterate on this until a good display is obtained.

You will notice that the image is displayed with the first pixel at the top left of the display unlike the way IRAF displays things. This is the convention used by FOCAS and it cannot be changed. Also realize that FOCAS defines the first pixel as (0,0) rather than (1,1). All FOCAS image coordinates are *zero indexed*. Finally the display is commanded by FOCAS to be 800x800. If you are also using IRAF it is best to set

```
    fo> set stdimage=imt800
```

to avoid changing the display server frame buffer configurations.

The other thing to check when displaying the image is that the first line of the image does not contain any large or bright objects. This is because the initial estimate of the background when detecting objects is determined from the first few lines. One may work around this but the best results are when the first lines are mostly background. One may rotate or flip the image if needed.

Other information which is useful to determine are the minimum and maximum background in the first few lines and the sigma of the noise in the background sky over the image. This information can be found with the display program, graphing parts of the image, or listing some of the pixel values. The background limits can be used to constrain the initial sky in the first few lines. This is particularly needed if there is a large bright object there. It is useful to specify the sky sigma directly because the automatic determination may not be as good as that found by the user.

3. Creating a Catalog

The first step in creating a catalog is to run the **setcat** program. This program creates an initially empty catalog and the user interactively defines various parameters. The most important things that are set at this stage are the catalog name, the image associated with the catalog, the detection thresholds, and the detection filter. The sky sigma may also be set at this point. The following example shows the minimum needed for a typical image. A common convention is to name the catalog as the image name with the extension `cat`. Many of the parameters are simply left at the default values by typing a carriage return.

```
    fo> setcat galfield.cat

    Set catalog galfield.cat

    Set catalog header parameters? y

      Field image file (): galfield
      Field name (Example artificial galaxy field):
      Field epoch (01/01/90):
      Field passband ():
      Field coordinates in decimal (0 0):
      Exposure or integration (0):
      Observer ():
      Origin ():
      Saturation value (30000):
      Magnitude zero point (30):
      Catalog magnitude limit (100):
      Radius of fixed circular aperture (5):
      Sigma density above sky for detection (2.5):
      Sigma density below sky for detection (2.5): 10
      Sigma of sky (0 = automatic determination) (0):
      Minimum area for detection (6): 9
      Significance level for evaluation and splitting (-100):
      Area description filename (galfiel.ar):
      Sky updating constants (0.1 0.1):
      Comments:
      Fri Sep 17 18:36:33 1993: setcat galfield.cat
      Additional comments (Exit with < cr > or EOF):
```

```
Set detection filter? y
  Use builtin filter? y
Set coordinate transformation?
Set intensity relation?
Set point spread function?
Set classification rules?
```

One may iterate setting the detection parameters, detecting the objects, and reviewing the results until the desired level of detection is reached.

The **info** command may be used to list the header of a catalog which includes most of the parameters set and modified by **setcat**.

```
fo> info galfield.cat
    ...
```

4. Detecting the Objects

The detection of objects in the image is done by the program **detect**. Usually there is only one argument, the name of the catalog. Optional arguments allow specifying a restricted set of image lines (zero indexed) and the initial background limits.

```
fo> detect galfield.cat
fo> nobjects galfield.cat
Catalog: galfield.cat
Number of objects: Total = 492
                   Stars = 0
                   Galaxies = 0
                   SF = 0
                   Long = 0
                   Diffuse = 0
                   Multiple = 0
                   Noise = 0
                   Unknown = 492
```

At this point there are 492 objects detected. They are all unknown since they have yet to be evaluated or classified. Often at this point one will use the task **review** to check the detections.

```
fo> review galfield.cat
    ...
```

This will clear the terminal screen, display the image with the data range set by **[f]display**, and enter a keystroke driven mode. For the purpose of this guide simply try typing the key 'h' (no carriage return is needed). This will draw the detection isophotes around the objects. Then to exit type 'q' and answer no, or simply carriage return, to the question about saving changes to the catalog.

If FOCAS patch 1 has not been included in the FOCAS installation it is possible that the h key in **review** *can hang the image display. One must then kill the image display and restart it. Thereafter one would probably avoid this very useful h option.* Another quirk of SAOimage is that when the little cross and individual isophotes are drawn the main window is not automatically updated. A change to the main window, such a slight pan, will force this update.

The detection step also records the background determined against which the objects were detected. This is an image with 1/8th the resolution of the original image. It may be displayed and zoomed like any image.

```
fo> fdisplay
D* w galfield.sky
Object: Sky for Example artificial gala
Display range (350 550):
D* q
```

Some features which may appear are plumes starting from the first image line and slight elevations in the background near bright objects, especially galaxies. The plume features are due to bright objects on the edge of the image where the initial background is basically unknown. The plume shape occurs because as the processing proceeds line-by-line the background contributions from regions on the sides and from the new lines allows the background to slowly converges on the true background. The small elevations are simply due to faint extended light which was below the detection threshold. Usually while this is visible it is at a low level.

5. Sky Evaluation

The detection step generates a sky background. However this is only a crude estimate. A separate step, **sky**, determines the local sky for each object from a square annulus around the object. There are various options but generally the default sky is the one to use.

```
fo> sky galfield.cat
```

Some of the older documentation does not mention **sky**. Older versions of FOCAS performed the sky evaluation as part of the general evaluation. Because sky determination is so critical to the results it was separated into the separate step in order to provide additional options.

6. Evaluation

The evaluation step, performed by the program **evaluate**, determines most of the remaining catalog parameters for the objects except for classification. This includes photometry, astrometry, and shape measurements.

```
fo> evaluate galfield.cat
```

If one does not do the sky evaluation step then the object evaluation will fail. Because a diagnostic flag is uninitialized in the code what happens is indeterminate. Either the evaluation operation will appear to succeed but none of the objects will be successfully evaluated or a string of messages like the following will appear.

```
evlobj: Failed because nsbr = 0 < 10
```

7. Splitting of Merged Objects

There are often multiple objects which are so close that the detection isophote encloses them as one object. The degree to which this happens depends on the field, the detection thresholds, and, possibly, strong background gradients. In some cases it is possible that a large region of the image is detected as a single object. In any event, the program **splits** separates multiple objects into smaller component objects by finding isophotes which divide the parent object into two or more objects (satisfying the minimum area constraint as well as a few others). The original parent entry is not removed from the catalog but it is simply classed as ’m’ for multiple. The daughter objects are automatically evaluated and classified. The sky is not reevaluated and the parent object sky is used.

```
fo> splits galfield.cat
getmodels: Point spread function undefined.
```

This guide recommends that the splitting be done before classification though either order may be used. If the point spread function is not defined, as in the example, the classification step is not done and a normal warning is given.

8. Classification

The last processing step for a single image catalog is the classification of objects. Note that if classifications are not needed for the analysis this step may be skipped. The classification is done using an algorithm called *the resolution classifier*. This is described in detail [2]. Briefly, a sequence of classification templates is formed and the one which best fits the object defines the classification. Most of the templates are derived by spatially scaling the point spread function (PSF); in other words by broadening and narrowing the PSF. The classification information is then the scale factor from which ranges of scales are mapped into the astronomical classes of ’s’ star, ’g’ galaxy, and ’n’ noise. Other classes which may appear are ’sf’ for fuzzy star in which a template consisting of the PSF plus a small amount of broader component is the best fit, ’long’ as determined from the shape information, and ’d’ for diffuse where the light distribution is very broad. Objects which failed to be evaluated or are otherwise not classified have the initial default class of ’u’ for unclassified or unknown and the multiple objects with class ’m’ are also not considered by the classifier at all.

The classifier requires that a PSF be defined in the catalog header. There are various ways this may be accomplished. The two common ways are an automated method and one where the user selects PSF objects. The automated technique is performed by the program **autopsf**. Its arguments are the catalog, the minimum number of template objects to use, a sigma constraint in the scatter of template object sizes, and the size of the PSF template to be stored in the catalog. The following shows some typical recommended values.

```
fo> autopsf galfield.cat 10 .05 11
Objects = 100 <ir1> = 2.89 sigma = 1.51 sigma/<ir1> = 0.52
Objects = 87 <ir1> = 2.42 sigma = 0.78 sigma/<ir1> = 0.32
Objects = 72 <ir1> = 2.16 sigma = 0.54 sigma/<ir1> = 0.25
Objects = 57 <ir1> = 1.95 sigma = 0.38 sigma/<ir1> = 0.20
Objects = 47 <ir1> = 1.82 sigma = 0.29 sigma/<ir1> = 0.16
Objects = 37 <ir1> = 1.71 sigma = 0.22 sigma/<ir1> = 0.13
Objects = 31 <ir1> = 1.65 sigma = 0.18 sigma/<ir1> = 0.11
Objects = 26 <ir1> = 1.61 sigma = 0.16 sigma/<ir1> = 0.10
Objects = 20 <ir1> = 1.55 sigma = 0.14 sigma/<ir1> = 0.09
Objects = 17 <ir1> = 1.52 sigma = 0.13 sigma/<ir1> = 0.08
Objects = 14 <ir1> = 1.49 sigma = 0.12 sigma/<ir1> = 0.08
Objects = 11 <ir1> = 1.45 sigma = 0.10 sigma/<ir1> = 0.07
Final statistics:
Objects = 9 <ir1> = 1.42 sigma = 0.09 sigma/<ir1> = 0.06
ir1 = 1.27713
   1   1  -2  -0   6  -0  -1  -2   3  -4  -1
   1  -1  -0   1  -3  -0  -1  -3   1  -4  -4
  -1   0   2  -2   2   4   1  -1   9  -5  -3
  -4   1   0   6  21  23  14   7   6   0   1
  -4  -0   4  19  47  78  54  21   4  -0  -4
  -1   0   5  17  55 100  80  20   4   5   3
   0   2   1  13  35  62  49  16   2  -1  -3
   2  -1   2   2  11  23  12   6  -1   1   1
  -2  -6   0   3   2   0   1  -0   5   0   3
  -2   5  -4  -3  -0   3  -2  -0  -4  -4   3
   1   0  -2  -1  -3   2   2   3  -2  -3  -0
```

The reference to `ir1` is the first radial moment of the light distribution which is a measure of the image size. Based on the output above, or in the example below, one might wish to redo the PSF with a different template size. The goal is to have the template be as small as possible but with values falling to near zero at the edges.

The second way to determine the PSF is to interactively select the template objects. The suggested method is to use the **review** task. In the text window type '.' to initiate selection of an object. Move the image cursor to a desired PSF object, that is a relatively bright, unsaturated star, and then type any key such as the space bar. Check that the object selected is the one intended and then flag the object for later selection by typing 'b' to set flags, 'a' to set the ATTENTION flag, and 'q' to finish the flag setting. After selecting as many objects as possible, though even just one object may be used, exit with 'q' and respond with y or yes to save the changes (the changed flags) in the catalog. The PSF is then obtained by selecting the flagged objects from the catalog, extracting their images, combining them, and entering the PSF in the catalog. This is done as in the following example.

```
fo> review galfield.cat
fo> ffilter F A <galfield.cat | template 11 11 | setpsf galfield.cat
ir1 = 1.96439
 -0   1  -1  -1  -3   4  -8  -1   3  -0  -6
  1   8   0   7  -4   0   8   3   6   3  -5
  2  -0   0  -4   2   7   4   8  -1   8  -2
 -0  -0   6  15  26  33  25   8   0   1   7
  1   3   4  22  54  70  49  36   8   0   3
 -1   2   9  34  75 100  77  26  15   4  -3
 -3   8  10  27  46  57  43  19  11   1   5
  1   9   4  13  22  25  17  10  10   6   5
 -1   8  10  11   3   6   7   5  -5   1  -6
  2   5   4   5   4  -2  -0   0  -5   5  -5
  5   9   5  -4  -4  -5  -3   5  -3   1   4
```

Once the PSF is set the classification is done simply with the **resolution** program.

```
fo> resolution galfield.cat
fo> nobjects galfield.cat
Catalog: galfield.cat
Number of objects: Total = 608
                   Stars = 168
                   Galaxies = 347
                   SF = 8
                   Long = 5
                   Diffuse = 9
                   Multiple = 57
                   Noise = 14
                   Unknown = 0
```

The classifications work best when the PSF template is fairly narrow. If the image resolution is very high or the PSF is very broad better classifications (and detections) may often be achieved by block averaging the image.

9. Matching of Multiple Catalogs

If only single images are being cataloged then the previous steps complete the processing. Sometimes, however, one has multiple images of the same field either with the same filter or with multiple filters. In this case it may be desired to match the same objects from the different images in the different catalogs. This is done by transforming the measured image coordinates to a common coordinate system stored in the `ra` and `dec` catalog fields. These fields are not automatically computed so they must first be set before matching.

While any coordinate system might be used, the recommend one is the pixel coordinates of one image against which to transform the other images. To set the `ra` and `dec`c coordinates to the default catalog coordinates simply requires using the default identity transform. This is done as in the following example where all other options are turned down.

```
fo> setcat galfield.cat

Set catalog galfield.cat

Set catalog header parameters?
Set detection filter?
Set coordinate transformation? y
  Apply 4-meter flat field corrections? n
  Use catalog object reference points? n
  Enter reference points manually (if yes exit with EOF)? n
  Enter transform matrix? n
Set intensity relation?
Set point spread function?
Set classification rules?
```

If all the images were aligned prior to making the catalogs then the identity transformations would be set with **setcat** for all the catalogs and the matching can then proceed. If, however, the images are not aligned one has to determine coordinate transformations for the other catalogs. One may enter a transformation matrix manually. Though the matrix is shown as 3x3, currently only the first two rows are used; i.e. a 3x2 matrix. The transformation equations are:

```
 ra = m11 * x + m12 * y + m13
dec = m21 * x + m22 * y + m23
```

For the simple case of no rotation or scale change only the offset terms `m13` and `m23` need to be determine and the transformation matrix would be:

```
1 0 <x shift>
0 1 <y shift>
0 0 0
```

The third way to define the transformation and matching coordinates is to flag some objects in the catalog as reference points with **review** and let **setcat** determine the transformation. This is best done with an image display having multiple frame buffers; i.e. **imtool** rather than **SAOimage**. However one may also do it by writing down values on paper. The idea is to use **review** to locate objects in the catalog used as the standard coordinate system, note the `ra` and `dec`c values, and then enter these values in the `ra` and `dec` fields for the same object in the other catalogs using the ’u’ key (after selecting them with ’.’). With multiple frames one may load review with multiple catalogs and quickly switch between images and the process is relatively easy. With a single frame one would write the down the reference coordinates and remember the objects chosen as reference objects. When the ’u’ key is used a reference flag is set in the catalog. When exiting **review** be sure to save the catalog changes. Then in **setcat** simply say yes to using the catalog object reference points.

```
fo> setcat galfield.cat

Set catalog galfield.cat
Set catalog header parameters?
Set detection filter?
Set coordinate transformation? y
  Apply 4-meter flat field corrections? n
  Use catalog object reference points? y
Set intensity relation?
Set point spread function?
Set classification rules?
```

After setting the `ra` and `dec` coordinates in all the catalogs to be matched, a matched catalog is created using the program **match**. Note that multiple and unknown objects, classes 'm' and 'u', are excluded from the matching. Matched catalogs and single catalogs have different formats so one must use different programs in the analysis. To help distinguish the two use a convention such as the recommended one of using the extension `mcat` for the matched catalog. The matching is done as in the following example.

```
fo> match galfield.mcat galfield.cat galfield.cat ...
match: Creating new match file galfield.mcat from galfield.cat
match: adding file galfield.cat to match file galfield.mcat
    ...
```

This example uses the same catalog so that you might try it. Normally the fields would be different and might have names like a2390u.cat, a2390b.cat, a2390v.cat. Another idea sometimes used is to make an image which combines all the separate images. This "deep" image can then be processed and used as the reference and included in the matching.

If one forgets to set the `ra` *and* `dec` *coordinates then the following message will be returned from match.*

```
fo> match galfield.mcat galfield.cat galfield2.cat
match: Creating new match file galfield.mcat from galfield.cat
match: adding file galfield2.cat to match file galfield.mcat
WARNING:  insufficient storage for full matching
WARNING:  insufficient storage for full matching
    ...
```

This is because all objects will be at coordinate (0,0) and the buffering used to store objects with nearby `dec` *for matching will fill up.*

10. Using Common Isophotes

When there are multiple images it is sometimes desirable to do the photometry using identical isophotes. This may be done as follows. Select the image to define the isophotes. This would normally be the deepest or a composite of the individual images. This catalog is processed in the normal way. Then the catalog is copied to new catalog names, one for each image. **Setcat** is used to change the image name in each copy. However, set the area file name to the original name; this will require a double confirmation. Now you may run the programs **sky** and **evaluate**. In fact any program except **detect** and **splits** may be used so that one can redo the PSF and classifications if desired.

There is one new program that also needs to be used. Because the total magnitude is based on apportioning the original total light in split objects, this requires a program which is like **splits** but doesn't actually split objects. Instead it follows the splitting hierarchy to compute the total magnitudes. The program is called **totmag**.

```
fo> totmag galfield.cat
```

11. Other FOCAS Tools

This guide only covers the basic processing steps needed to create a single or matched catalog. In this section we mention some other useful FOCAS programs without much elaboration. The most common tools are the filtering programs **ffilter** (called **filter** in version before 1996) and **mfilter** and the listing programs **catlist**, **mcatlist**, and **matchlist**. The filter options described for **ffilter** are used in many places such as in some of the processing commands to limit the operation of the command and in **review**. The filter commands take an input catalog and produce an output catalog.

The listing tools extract information from a catalog and print it as a text stream. The commands are **catlist**, **mcatlist**, and **matchlist**. The text output can then be used in many programs for analysis and graphing. A template example of usage is,

`ffilter` *[options]* `<` *catalog* | `catlist` *[options]* | *analysis*

Some of the "analysis" tools which work on the text stream output from **catlist** are **opstrm** (a very powerful stream calculator), **plot.graph** (the IRAF graph routine), and **skip, statstrm, binstrm** (tools to manipulate and do statistics and histograms on input text).

As you have seen the **review** task is a very powerful one provided an image display server is available. The related one for matched catalogs if **mreview**. This requires a server with multiple frame buffers such as **imtool**.

Complete analysis often requires use of artificial data. The IRAF **artdata** package is useful in this regard. FOCAS provides the program **artcat** which enters objects in a catalog. The idea here is that **setcat** creates a catalog which points to an artificial field. The objects in the field are described by a text file (the same as produced by the IRAF tasks **starlist** and **gallist** and used by **mkobjects**) and this information is entered into the catalog by **artcat** (essentially replacing **detect** and usually the other processing steps). This only sets some of the fields but it does produce a dummy area file. This catalog may be reviewed and matched just like normal catalogs.

12. Summary

The basic steps are summarized by the following list.

```
rfits
display
setcat
detect
sky
evaluate
splits
autopsf
resolution
```

with `review` interspersed at any point after the objects are detected. The steps between detection and classifications can be done by the FOCAS script **autofocas**. If matched catalogs are desired then one needs to set the coordinate transformation with **setcat** and then do the matching **match**.

13. Getting Help

FOCAS is supported by the IRAF Group at the National Optical Astronomy Observatories. Help may be requested by e-mail to iraf@noao.edu. The author and maintainer of the software is Francisco Valdes whose e-mail is fvaldes@noao.edu.

References

A few FOCAS papers have been published but many more have not. These are available electronically from the anonymous FTP account on iraf.noao.edu in the directory iraf/docs/focas or by request to iraf@noao.edu (regular mail to NOAO/CCS). The FOCAS Manual is simply the collection of the manual pages which are part of the FOCAS distribution. They may be viewed on-line with the **focasman**. The manual pages are sometimes out-of-date. If one really wants a printed copy of the manual pages this may be obtained by request to iraf@noao.edu or NOAO/CCS.

Also available electronically is the software itself, installation instructions, a frequently asked questions list, and some standard test images. This are found in the FTP account in the iraf.old directory.

[1] Jarvis, J. F. and Tyson, J. A., **Astron. J. 86**, 476, 1981.

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

[3] Valdes, Francisco, *Faint Object Classification and Analysis System*, Central Computer Services, National Optical Astronomy Observatories, P.O. Box 26732, Tucson, AZ, 85725, USA, 1983.

[4] Valdes, Francisco, *FOCAS User's Manual*, Central Computer Services, National Optical Astronomy Observatories, P.O. Box 26732, Tucson, AZ, 85725, USA, 1983.

[5] Valdes, Francisco, *Faint Object Classification and Analysis System: Standard Test Image Results*, in **Proceedings of the 1st ESO/ST-ECF Data Analysis Workshop** (ESO Conference and Workshop Proceedings No. 31), ed. P.J. Grosbol, F. Murtagh, and R.H. Warmels, 1989.