A User's Guide to the IRAF Apphot Package

Lindsey Elspeth Davis

National Optical Astronomy Observatories*
Revised May 1989

ABSTRACT

The APPHOT package is a set of tasks for performing aperture photometry on uncrowded or moderately crowded stellar fields in either interactive or batch mode. The photometric technique employed is fractional pixel integration. Point spread function fitting techniques are not used and no knowledge of the point spread function is required for the computation of magnitudes. Separate tasks are provided for creating and modifying object lists, computing accurate centers and sky values for a list of objects, and performing photometry inside concentric circular or polygonal apertures.

This document describes the data preparation required to run APPHOT, how to set up the display and graphics devices, how to set the algorithm parameters, how to run the package tasks in interactive or batch mode and how to selectively examine the results. Detailed descriptions of the algorithms can be found in the document *Specifications for the APPHOT Package* by the author.

This document applies to APPHOT under IRAF version 2.8. APPHOT can be run under IRAF versions 2.5, 2.6 and 2.7 with only minor changes in the task setup. These differences are documented where appropriate in the text.

August 22, 1990

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

A User's Guide to the IRAF Apphot Package

Lindsey Elspeth Davis

National Optical Astronomy Observatories*
Revised May 1989

1. Introduction

The APPHOT package is a set of tasks for performing aperture photometry on uncrowded or moderately crowded fields. The photometric technique employed is fractional pixel integration. Point spread function techniques are not used and no knowledge of the point spread function is required for the computation of magnitudes.

The APPHOT package performs multi-aperture photometry on digitized starfields maintained as IRAF image files. Input to the package consists of an image file(s), an optional list(s) of object coordinates, numerous parameters controlling the analysis algorithms and, optionally, the graphics terminal and/or display. APPHOT output consists of successive records, where each record records the results of the analysis for a single object. Some tasks also produce graphics output in the form of plot metacode files.

Given starting coordinates and an IRAF image, the principal APPHOT task **phot** computes accurate centers, sky values and magnitudes for a list of objects. Separate IRAF callable tasks in APPHOT exist to create and modify object lists, to determine image characteristics such as the full width half maximum of the point spread function or standard deviation of the sky pixels, to compute accurate centers for a list of objects, to compute accurate local sky values for a list of objects, and to compute magnitudes inside a polygonal aperture.

The image data requirements of the APPHOT package are described in section 2. Section 3 describes various ways to tailor the IRAF environment to run the APPHOT package. Section 4 describes how to load the APPHOT tasks and how use the on-help facility. Section 5 describes how to examine and edit the APPHOT task and algorithm parameters. Several methods of creating and modifying APPHOT coordinate lists files are described in section 6. Sections 7 and 8 describe how to run the APPHOT tasks interactively without and with a coordinate list respectively. Batch mode reductions in APPHOT are described in section 9. Section 10 describes the format of the APPHOT output catalogue and plot files. Section 12 lists various APPHOT recipes for reducing common types of astronomical data with APPHOT.

2. Data Preparation and Requirements

APPHOT assumes that the images to be analyzed exist on disk in IRAF readable format. Facilities for reading and writing images exist elsewhere in IRAF in the DATAIO and MTLOCAL packages. None of the current APPHOT tasks alter the disk input images in any way.

APPHOT assumes that the pixel data is linear. The input images should be corrected for those instrumental defects which affect the intensity of a pixel prior to entering the APPHOT package. These defects include pixel to pixel variations in the bias values, pixel to pixel gain variations, cosmic rays, cosmetic defects, geometric distortion and detector non-linearities.

---

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

Users should be aware of the IRAF CCDRED package for reducing CCD data and the DTOI package for converting photographic density data to intensity data.

Extreme valued pixels should be removed from the images prior to entering the APPHOT package. These include pixel values at or near the data limits of the host machine as well as any host machine values such as INDEF, produced by divide by zero, and floating point underflows and overflows. Floating point operations involving such numbers may crash with arithmetic exception errors. For efficiency and portability reasons the APPHOT package and most IRAF routines do not test for these numbers. The **imreplace** and **imedit** tasks in the PROTO package can be used to replace extreme valued pixels. More general system facilities for handling bad pixels inside IRAF are planned for the near future.

In order to normalize the magnitude scales of a list of images to a common integration time, APPHOT requires that the image exposure times be stored in the image header. Similarly the correct computation of magnitude errors requires that the readout noise and gain parameters also be present in the image header or be supplied as constants in the APPHOT parameter files. The readout noise must be supplied in units of electrons and the gain is assumed to be in electrons per adu. The time units are arbitrary but must be consistent for a given set of images. APPHOT tasks access this information using a keyword and value scheme. The IMAGES package task **hedit** can be used to insert or edit this information prior to entering the APPHOT package. For example the following commands will enter the gain, readout noise and exposure time information into the headers of a list of images.

```
cl> hedit *.imh gain 14.0 add+ ver-
cl> hedit *.imh readout 20.0 add+ ver-
cl> hedit *.imh exptime add+ ver+
```

The point spread function is assumed to be constant for all regions of the image. This is critical in the case of faint objects for which small apertures which minimize the effects of crowding and sky noise in the aperture are used. The wings of the object will almost certainly extend beyond the aperture and good results will only be obtained if objects are consistently well centered and the shape and diameter of an object is constant throughout the object and invariant to magnitude.

The centering routines built into the APPHOT package assume that the images are close to circularly symmetric over the region to be used in centering although small deviations do not significantly affect the accuracy of the results. The user should be aware of the more sophisticated but less efficient centering routines in the **daofind** and **fitpsf** routines. Several choices of centering algorithm are available in APPHOT including no centering.

APPHOT assumes that the local sky background is approximately flat in the vicinity of the object being measured. This assumption is equivalent to assuming that the local sky region has a unique mode. Therefore any variations in the sky background which occur at the same scale as the sky region should be removed prior to entering the APPHOT package.

3. Setting up the IRAF Environment for APPHOT

3.1. Script, Compiled and Pset Tasks

IRAF supports three types of tasks: scripts, compiled tasks and pset tasks. In order to distinguish one type of task from another, APPHOT users should set the cl parameter **showtype** to yes as shown below.

```
cl> cl.showtype=yes
```

The cl command **?** or **?** *package* identifies script tasks by a terminating period and pset tasks by a terminating @. No trailing characters are added to the compiled task names. Pset tasks are

important to the APPHOT package and will be discussed further in section 5.

3.2. Other IRAF Packages

Before running APPHOT, users may wish to load the IRAF packages, DATAIO, IMAGES, TV and PLOT. Various input and output routines exist in the DATAIO package, including the fits reader and writer and the cardimage reader and writer. The IMAGES package contains routines for basic image arithmetic, for computing image statistics, for listing individual pixel values, and for examining and modifying the image headers. TV, a subpackage of IMAGES, contains the **display** task for loading images into the display device. The PLOT package contains tasks for plotting image data and for extracting and displaying individual plots from the plot metacode files produced by the APPHOT tasks.

Various useful tasks for manipulating and displaying the data produced by APPHOT can be found in the PROTO package under the NOAO package. In particular the user should be aware of the **fields**, **mkhistogram** and **tvmark** tasks.

3.3. The Image Cursor and Display Device

The APPHOT tasks are designed in interactive mode to read the image cursor and to perform various actions based on the position of the image cursor and the keystroke typed. The image cursor is directed to the display device defined by the environment variable **stdimage**. To check the value of the default display device, type the following command.

```
cl> show stdimage
imt512
```

In this example the default display device is the 512 pixel square SUN imtool window. All tasks which write images to the display access this device. For example the TV package **display** program will load an image onto this device.

In normal operation IRAF tasks which read the image cursor would read the hardware cursor from this device. In response to the user command

```
cl> =imcur
or
cl> show imcur
```

the cursor would come up on the image display ready to accept a keystroke command. At this point the user should check that the display is reading the correct image coordinates by moving the image cursor to the lower left hand corner of the display image and tapping any key. The coordinates should read x,y = (1,1) if the whole image was displayed.

Cursor readback is currently implemented under IRAF version 2.7 for the SUN workstations and the IIS model 70. Users with older versions of IRAF or other devices cannot run APPHOT tasks directly from the image display device and must redirect the image cursor. Two choices are available.

[1] The image cursor can be directed to accept commands from the standard input. This is the default setup under IRAF version 2.6. This setup can be checked by typing the following command.

```
cl> show stdimcur
text
```

If the value of **stdimcur** is not "text" the user can set this value by typing the following.

```
cl> set stdimcur = text
```

Each time the cursor is to be read the user will be prompted for image cursor mode text input. The form and syntax of this command are described in detail in section 7.3.

[2] Alternatively a contour plot of the image can be used in place of the image display and APPHOT tasks can be directed to read the graphics cursor. To direct the image cursor to the graphics device the user types

```
cl> set stdimcur = stdgraph
```

This usage permits interactive use of the APPHOT package for users with graphics terminals but no image display. This setup is most suitable for terminals which permit the text and graphics planes to be displayed simultaneously.

It is currently the responsibility of the user to ensure that the image displayed on the image display is the same as the image operated on by the APPHOT tasks.

4. Loading the Apphot Package

At this point the user has the local environment set up and is ready to load the APPHOT package. APPHOT resides in the DIGIPHOT package (IRAF version 2.8 and later) under the NOAO suite of packages. Assuming that the NOAO package is already loaded the user types the following.

```
cl> digiphot
cl> apphot
```

APPHOT can also be an add-on package (IRAF version 2.7 and earlier) installed under the LOCAL package. In this case the user types.

```
cl> local
cl> apphot
```

The following menu of tasks is displayed.

```
apselect      daofind     fitsky        photpars@   polyphot   wphot
center        datapars@   fitskypars@   polymark    qphot
centerpars@   fitpsf      phot          polypars@   radprof
```

The APPHOT package is now loaded and ready to run. A quick one line description of each APPHOT task can be obtained by typing the following command.

```
ap> help apphot
```

The following text appears.

```
digiphot.apphot:
       apselect - Extract select fields from apphot output files
         center - Compute accurate centers for a list of objects
     centerpars - Edit the centering parameters
        daofind - Find stars in an image using the DAO algorithm
       datapars - Edit the data dependent parameters
         fitpsf - Model the stellar psf with an analytic function
         fitsky - Compute sky values in a list of annular or circular
                  regions
     fitskypars - Edit the sky fitting parameters
           phot - Measure magnitudes for a list of stars
       photpars - Edit the photometry parameters
       polymark - Create polygon and coordinate lists for polyphot
       polyphot - Measure magnitudes inside a list of polygonal regions
       polypars - Edit the polyphot parameters
          qphot - Measure quick magnitudes for a list of stars
        radprof - Compute the stellar radial profile of a list of stars
          wphot - Measure magnitudes with weighting
```

For the remainder of this document we will use the principal APPHOT task **phot** as an example of how to setup the parameters in both interactive and batch mode. To get detailed help on the phot task the user types the following.

```
cl> help phot | lprint
```

The help page(s) for the **phot** task will appear on the local default printer.

5. Setting the APPHOT Photometry Task Parameters

The principal APPHOT task PHOT is described in sections 5.1 to 5.6. The quick photometry task QPHOT is described in section 5.7 and the polygonal aperture photometry task POLYPHOT is described in section 5.8.

5.1. The Task Parameters

The **phot** task parameter set specifies the required image, coordinate and output files, the graphics and display devices, the graphics and image cursor and the mode of use of the task, interactive or batch. To enter and edit the parameter set for the **phot** task the user types the following.

```
cl> epar phot
```

The parameter set for the **phot** task will appear on the terminal ready for editing as follows.

```
                              IRAF
                Image Reduction and Analysis Facility

   PACKAGE = apphot
   TASK = phot

   image   =                       Input image
   (datapar=                      ) Data dependent parameters
   (centerp=                      ) Centering parameters
```

```
(fitskyp=                    ) Sky fitting parameters
(photpar=                    ) Photometry parameters
(coords =                    ) Coordinate list
skyfile =                      Sky file
(output =              default) Results file
(plotfil=                    ) File of plot metacode
(graphic=             stdgraph) Graphics device
(display=             stdimage) Display device
(command=                    ) Image cursor: [x y wcs] key [cmd]
(cursor =                    ) Graphics cursor: [x y wcs] key [cmd]
(radplot=                  no) Plot the radial profiles
(interac=                 yes) Mode of use
(verify =                 yes) Verify critical parameters in non
                               interactive mode
(verbose=                  no) Print messages in non interactive mode
(mode   =                  ql)
```

The **phot** parameters can be edited in the usual fashion by successively moving the cursor to the line opposite the parameter name, entering the new value, followed by a carriage return, and finally typing a ^Z to exit the **epar** task and update the parameters. Some general points about the task parameter sets are summarized below. For more detailed descriptions of each parameter see the help pages for each task.

[1] **Image** specifies the list of input image(s) containing the stars to be measured. **Image** may be a list of images, an image template or a file containing a list of images. For example if we wish to measure stars in three images: m31U, m31B and m31V we could specify the **image** parameter in the following three ways.

```
image   =       m31B,m31U,m31V  Input image
 or
image   =             m31*.imh  Input image
 or
image   =              @imlist  Input image
```

"Imlist" is the name of a text file containing the list of images one image name per line. The image list file can easily be created with the cl package **files** task or the editor.

[2] Four parameter sets, henceforth psets, **datapars**, **centerpars**, **fitskypars** and **photpars** specify the algorithm parameters. They are described in detail in later sections.

[3] **Coords** specifies the name of the coordinate file(s) containing the initial positions of the stars to be measured. If **coords** = "", the current image cursor position is read and used as the initial position. The number of files specified by **coords** must be either one, in which case the same coordinate file is used for all the images, or equal in number to the set of input images. **Coords** can be a list of files, a file name template, or a file containing the list of x and y coordinates one per line. For example if we have three coordinate files "m31B.coo", "m31U.coo" and "m31V.coo" for the three images listed above, we could set the **coords** parameter in the following three ways.

```
(coords = m31B.coo,m31U.coo,m31V.coo) Coordinate list
 or
(coords =             m31*.coo) Coordinate list
 or
```

```
(coords =           @coordlist) Coordinate list
```

"Coordlist" is the name of a text file containing the names of the coordinate files in the desired order one per line.

[4] **Output** specifies the name of the results file(s). If **output** = "default" then a single output file is created for each input image and the root of the output file name is the name of the input image. In the case of the above example **phot** would create three output files called "m31B.mag.1", "m31U.mag.1" and "m31V.mag.1" assuming that this was the initial run of **phot** on these images. If the user sets the **output** parameter then the number of output files must be either one or equal to the number of input images. For example the user could set **output** to either

```
(output =              m31.out) Results
or
(output = m31b.out,m31u.out,m31v.out) Results
```

If the user sets **output** = "" then no output file is written.

[5] The parameters **graphics** and **display** specify the graphics and image display devices. In IRAF version 2.6 and later the APPHOT tasks which reference these parameters will in interactive mode issue a warning if they cannot open either of these devices and continue execution. In IRAF version 2.5 the **display** parameter must be set to "stdgraph" as listed below

```
(display=             stdgraph) Display device
```

or the following system error will be generated.

```
"cannot execute connected subprocess x_stdimage.e"
```

Most of the APPHOT tasks use IRAF graphics in interactive mode to allow users to set up their parameters and /or examine their results using radial profile plots. The **graphics** specifies which graphics device these plots will be written to. Similarly most IRAF tasks permit the user to optionally mark the star positions, apertures and sky annuli on the display device. The parameter **display** specifies which image display device this information will be written to. Currently IRAF does not support an image display kernel so the display marking features of APPHOT are not available unless the user chooses to run APPHOT interactively from a contour plot.

[6] If **plotfile** is not equal to "", then for each star written to **output** a radial profile plot is written to the plot metacode file **plotfile**. The **plotfile** is opened in append mode and succeeding executions of **phot** write to the end of the same file which may in the process become very large. *The user should be aware that writing radial profile plots to* **plotfile** *can significantly slow the execution of* **phot**. The variable **radplots** enables radial profile plotting in interactive mode. For each star measured a radial profile plot displaying the answers is plotted on the screen.

[7] The **interactive** parameter switches the task between interactive and batch mode. In interactive mode plots and text are written to the terminal as well as the output file and the user can show and set the parameters. In batch mode **phot** executes silently.

[8] The **verify** parameter allows the user to verify the critical task parameters in non interactive mode. It is normally set to yes but should be turned off when submitting jobs to background.

[9] The **verbose** switch permits the printing of results to the standard output in non interactive mode. It is normally turned off.

5.2. APPHOT Psets

APPHOT algorithm parameters have been gathered together into logical groups and stored in parameter files. The use of psets permits the user to store APPHOT parameters with their relevant datasets rather than in the uparm directory and allows APPHOT tasks to share common parameter sets. APPHOT presently supports 5 pset files: 1) **datapars** which contains the data dependent parameter 2) **centerpars** which contains the centering algorithm parameters 3) **fitskypars** which contains the sky fitting algorithm parameters 4) **photpars** which contains the multiaperture photometry parameters and 5) **polypars** which contains the polygonal aperture photometry parameters. The user should consult the manual page for each of the named pset files as well as the attached parameter set document, *External Parameter Sets in the CL and Related Revisions*, by Doug Tody.

The default mode of running APPHOT is to edit and store the pset parameter files in the uparm directory. For example to edit the **datapars** parameter set, the user types either

```
cl> epar datapars
or
cl> datapars
```

and edits the file as usual. All the top level tasks which reference this pset will pick up the changes from the uparm directory, assuming datapars = "".

Alternatively the user can edit the **phot** task and its psets all at once as follows using **epar**.

```
cl> epar phot
```

Move the cursor to the **datapars** parameter line and type **:e**. The menu for the **datapars** pset will appear and is ready for editing. Edit the desired parameters and type **:q**. **Epar** will return to the main **phot** parameter set. Follow the same procedure for the other three psets **centerpars**, **fitskypars** and **photpars** and exit the program in the usual manner.

Sometimes it is desirable to store a given pset along with the data. This provides a facility for keeping many different copies of say the **datapars** pset with the data. The example below shows how to write a pset out to a file in the same directory as the data. The user types

```
cl> epar phot
```

as before, enters the datapars menu with `:e` and edits the parameters. The command

```
:w data1.par
```

writes the parameter set to a file called "data1.par" and a **:q** returns to the main task menu. A file called "data1.par" containing the new **datapars** parameters will be written in the current directory. At this point the user is in the **phot** parameter set at the line opposite **datapars** and enters "data1.par" on the line opposite this parameter. The next time **phot** is run the parameters

will be read from "data1.par" not from the pset in the uparm directory. This procedure can be repeated for each data set which has distinct parameters, as in for example data taken on separate nights.

5.3. Datapars

All the data dependent parameters have been gathered together in one pset **datapars**. The idea behind this organization is to facilitate setting up the algorithm psets for data taken under different conditions. For example the user may have determined the optimal centering box size, sky annulus radius and width and aperture radii in terms of the current **fwhmpsf** and the rejection criteria in terms of the current background standard deviation **sigma**. In order to use the same setup on the next image the user need only reset the **scale** and background **sigma** parameters to the new values. The only pset which need be edited is **datapars**.

To examine and edit the **datapars** pset type

```
ap> datapars
```

and the following menu will appear on the screen.

```
                                IRAF
                Image Reduction and Analysis Facility

PACKAGE = apphot
TASK = datapars

(fwhmpsf=                  1.) FWHM of the PSF in scale units
(emissio=                 yes) Features are positive
(noise  =             poisson) Noise model
(thresho=                  0.) Detection threshold in counts above
                               background
(cthresh=                  0.) Centering threshold in counts above
                               background
(sigma  =              INDEF) Standard deviation of background in counts
(scale  =                 1.) Image scale in units per pixel
(ccdread=                   ) CCD readout noise image header keyword
(readnoi=              INDEF) CCD readout noise in electrons
(gain   =                   ) CCD gain image header keyword
(epadu  =                 1.) Gain in electrons per count
(exposur=                   ) Exposure time image header keyword
(itime  =              INDEF) Integration time
(datamin=              INDEF) Minimum good data pixel
(datamax=              INDEF) Maximum good data pixel
(mode   =                 ql)
($nargs =                  0)
```

The following is a brief description of the parameters and their function as well as some initial setup recommendations.

Scale is the image scale parameter in units per pixel, for example arc-seconds per pixel. All distance dependent parameters in the APPHOT package including the centering box width **cbox** in **centerpars**, the inner radius and width of the sky annulus, **annulus** and **dannulus** in

fitskypars, and the radii of the concentric circular apertures **apertures** in **photpars** are defined in scale units, for example arc seconds. This permits easy comparison with apertures published in the literature. Some other algorithm parameters such as **maxshift** in **centerpars** and the region growing radius **rgrow** in **fitskypars** are also defined in scale units. By default all distance parameters are defined in pixels.

Fwhmpsf is used as a first guess for modelling the psf in the **fitpsf** task, is important for the optimal use of the **daofind** algorithm, and critical for the centering algorithms "gauss" and "ofilter" as well as the **wphot** task. **Fwhmpsf** as well as the other distance dependent parameters can be set interactively from inside most of the APPHOT tasks.

Fwhmpsf and **scale** can be combined in an interesting way. **Scale** can be defined in units of half width half maximum of the psf per pixel in which case the value of **fwhmpsf** should be set to **2.0** by definition. By trial and error and use of the interactive setup menu optimal values of the remaining distance dependent parameters can be determined in units of **scale**. Once determined the same setup can be reused on another image by simply reseting the **scale** parameter.

APPHOT photometry routines permit measurement of both emission and absorption features. For the majority of applications including photometry of stars and galaxies all "objects" are emission "objects" and the **emission** parameter should be left at yes.

APPHOT currently supports two noise models "constant" and "poisson". If **noise** = "constant" the magnitude errors are computed from the Poisson noise in the sky background plus the readout noise. If **noise** = "poisson" the magnitude errors are computed on the basis of the Poisson noise in the constant sky background, Poisson noise in the object and readout noise. Most users with CCD data will wish to leave **noise** = "poisson". **Cthreshold** is a parameter required by the centering algorithms. If **cthreshold** > 0.0, pixels below the data minimum plus threshold in the centering subraster are not used by the centering algorithm. For difficult centering problems the user may wish to adjust this parameter. The **sigma** parameter specifies the standard deviation of the background in a single pixel. **Sigma** is used to estimate the signal to noise ratio in the centering subraster and to set the width and bin size of the histogram of sky pixels, the **khist** and **binsize** parameters in the pset **fitskypars**. Both **cthreshold** and **sigma** can be set interactively from inside the **phot** task.

APPHOT currently recognizes three image header keywords **ccdread**, **gain** and **exposure**. Knowledge of the instrument gain and readout noise is required for the correct computation of the magnitude errors but not required for the magnitude computation. The units of the gain and readout noise are assumed to be electrons per adu and electrons respectively. Exposure time information is required to normalize the magnitudes computed for a series of images to a common exposure time. The time units are arbitrary but must be consistent for a set of images. If this information is already in the image header the user can enter the appropriate header keywords. Otherwise the instrument constants gain and readout noise can be entered into the parameters **epadu** and **readnoise**. If the exposure time information is not present in the image header, the user can either edit it in with the **hedit** task or change the **itime** parameter for each image reduced. If both image header keywords and default parameter values are defined the image header keywords take precedence.

The two parameters **datamin** and **datamax** which define the upper and lower good data limits respectively are not currently implemented by the APPHOT tasks. They will be used in future versions of the package to, for example, set the limits over which a detector is linear.

After editing, the new **datapars** pset might look like the following. This user has chosen to wait and set **fwhmpsf**, **threshold**, **cthreshold**, and **sigma** interactively from inside **phot** but has decided to set the image header parameters **ccdread**, **gain** and **exposure**.

```
PACKAGE = apphot
TASK = datapars

(fwhmpsf=                1.) FWHM of the PSF in scale units
(emissio=              yes) Features are positive
(noise  =          poisson) Noise model
(thresho=                0.) Detection threshold in counts above
                             background
(cthresh=                0.) Centering threshold in counts above
                             background
(sigma  =            INDEF) Standard deviation of background in counts
(scale  =                1.) Image scale in units per pixel
(ccdread=          readout) CCD readout noise image header keyword
(readnoi=            INDEF) CCD readout noise in electrons
(gain   =             gain) CCD gain image header keyword
(epadu  =                1.) Gain in electrons per count
(exposur=          exptime) Exposure time image header keyword
(itime  =            INDEF) Integration time
(datamin=            INDEF) Minimum good data pixel
(datamax=            INDEF) Maximum good data pixel
(mode   =               ql)
($nargs =                0)
```

5.4. The Centering Parameters

The centering algorithm parameters have been grouped together in a single parameter set **centerpars**. To display and edit these parameters type the command.

```
ap> centerpars
```

The following menu will appear on the terminal.

```
                          IRAF
            Image Reduction and Analysis Facility

PACKAGE = apphot
TASK = centerpars

(calgori=          centroid) Centering algorithm
(cbox   =                5.) Centering box width in scale units
(maxshif=                1.) Maximum center shift in scale units
(minsnra=                1.) Minimum SNR ratio for centering
(cmaxite=                10) Maximum iterations for centering algorithm
(clean  =                no) Symmetry clean before centering
(rclean =                1.) Cleaning radius in scale units
(rclip  =                2.) Clipping radius in scale units
(kclean =                3.) K-sigma rejection criterion in skysigma
(mkcente=                no) Mark the computed center
(mode   =                ql)
($nargs =                 0)
```

APPHOT offers three choices for the centering algorithm: the default "centroid", "gauss" and "ofilter". The default centering algorithm does not depend on **fwhmpsf** but the remaining two do. For reasons of simplicity and efficiency the author recommends the default algorithm. In cases where there is significant crowding or the data is very noisy users may wish to experiment with the other algorithms. Centering can be disabled by setting **calgorithm** = "none". This option is useful if accurate centers have already been computed with the **daofind** or **fitpsf** tasks. More detailed information on the APPHOT centering algorithms can be found in the document, *Specifications for the Apphot Package* by Lindsey Davis.

The centering box **cbox** is defined in units of **scale**. Users should try to set **cbox** as small as possible to avoid adding noisy pixels to the centering subraster. **Cbox** can also be set interactively from inside the APPHOT **phot** task.

If the computed centers are more than **maxshift** / **scale** pixels from the initial centers or the signal-to-noise ratio in the centering subraster is less than **minsnratio** the new center will be computed but flagged with a warning message.

For stars which are crowded or contaminated by bad pixels the user may wish to enable the cleaning algorithm by setting **clean** = yes. Its use is complicated and not recommended for most data. The algorithm is described in the APPHOT specifications document.

If **mkcenter**=yes, **phot** tasks will mark the initial and final centers and draw a line between them on the default display device. At present this option only works if **display**="stdgraph".

In the above example we have elected to leave the **calgorithm** parameter at its default value and set **cbox** interactively from inside the **phot** task.

5.5. The Sky Fitting Parameters

The sky fitting algorithm parameters have been grouped together in a single parameter set **fitskypars**. To display and edit these parameters type the following command.

```
ap> fitskypars
```

The following menu will appear on the terminal.

```
                                IRAF
                Image Reduction and Analysis Facility

   PACKAGE = apphot
   TASK = fitskypars

   (salgori=                mode) Sky fitting algorithm
   (annulus=                 10.) Inner radius of sky annulus in scale units
   (dannulu=                 10.) Width of sky annulus in scale units
   (skyvalu=                  0.) User sky value
   (smaxite=                   10) Maximum number of sky fitting iterations
   (snrejec=                   50) Maximum number of sky fitting rejection
                                   iterations
   (skrejec=                  3.) K-sigma rejection limit in sky sigma
   (khist  =                  3.) Half width of histogram in sky sigma
   (binsize=                 0.1) Binsize of histogram in sky sigma
   (smooth =                   no) Lucy smooth the histogram
   (rgrow  =                  0.) Region growing radius in scale units
   (mksky  =                   no) Mark sky annuli on the display
```

```
(mode   =                    ql)
($nargs =                     0)
```

APPHOT offers ten sky fitting algorithms. The algorithms can be grouped into 4 categories 1) user supplied sky values including "constant" and "file" 2) sky pixel distribution algorithms including "median" and "mode", 3) sky pixel histogram algorithms including "centroid", "crosscor", "gauss" and "ofilter" 4) interactive algorithms including "radplot" and "histplot". The definitions of the mode used by APPHOT is the following.

```
mode = 3.0 * median - 2.0 * mean
```

Detailed descriptions of the algorithms can be found in the document, *Specifications for the Apphot Package* by the author. The author recommends "mode" the default, and one of the two histogram algorithms "centroid" and "crosscor".

The inner radius and width of the sky annulus in terms of **scale** are set by the parameters **annulus** and **dannulus**. These can easily be set interactively from within the **phot** task. Good statistics require several hundred sky pixels. The user should be aware that a circular sky region can be defined by setting **annulus** to 0.

The user should ensure that the parameter **sigma** in the **datapars** parameter set is defined if one of the histogram dependent sky fitting algorithms is selected. The extent and resolution of the sky pixel histogram is determined by **khist** and **binsize** and their relation to **sigma**. If **sigma** is undefined then the standard deviation of the local sky background is used to parameterise **khist** and **binsize** and the histograms of different stars can deviate widely in resolution.

The sky rejection algorithms are controlled by the parameters **skreject**, **snreject** and **rgrow**. It is strongly recommended that the user leave pixel rejection enabled. The user should experiment with the region growing radius if the local sky regions are severely crowded.

If **mksky** = yes, **phot** will mark the inner and outer sky annuli. At present this option will only work if **display** = "stdgraph".

5.6. The Photometry Parameters

The photometry algorithm parameters have been grouped together in a single parameter set **photpars**. To display and edit these parameters type.

```
ap> photpars
```

The following menu will appear on the terminal.

```
                          IRAF
              Image Reduction and Analysis Facility

PACKAGE = apphot
TASK = photpars

(weighti=             constant) Photometric weighting scheme for wphot
(apertur=                   3.) List of aperture radii in scale units
(zmag   =                  26.) Zero point of magnitude scale
(mkapert=                  no) Draw apertures on the display
(mode   =                  ql)
($nargs =                   0)
```

There are three weighting options inside APPHOT. The default is "constant". Inside the **phot**, **radprof** and **polyphot** tasks only constant weighting is used. Two other weighting schemes are available for the experimental **wphot** task, "gauss" and "cone". "Gauss" is the more highly recommended.

The aperture list is specified by **apert** in terms of **scale**. The apertures can be entered in any order but are sorted on output. **Apert** can be string or the name of a text file containing the list of apertures. Apertures can either be listed individually and separated by whitespace or commas or a ranges notation of the form apstart:apend:apstep can be used. These can be set interactively from within the **phot** task.

Examples of valid aperture strings are listed below.

```
1 2 3
1.0,2.0,3.0
1:10:1
```

An arbitrary zero point is applied to the magnitude scale with **zmag**. The user can accept the default or experiment with his/her data until a suitable value is found. The computation of the magnitude errors does not depend on the zero point.

If **mkapert** = yes, the **phot** task will draw the concentric apertures on the display. At present this option works only if **display** = "stdgraph".

5.7. The QPHOT Task

Qphot computes accurate centers, sky values and magnitudes for a list of objects using a restricted subset of the full **phot** parameter set. It is intended to be a quick look photometer suitable for use when observing on the mountain or in well behaved uncrowded stellar fields.

The user is automatically queried for the critical parameters **cbox**, **annulus**, **dannulus** and **apertures** which are defined in terms of pixels. The noise characteristics of the detector are assumed to obey Poisson statistics. **Qphot** computes centers for each object using the "centroid" centering algorithm. Sky values are calculated using the "mode" algorithm. The user can set the zero point of the magnitude scale, **zmag**, the gain of the detector, **epadu**, and exposure time image header keyword, **exposure**, but all the remaining parameters are set to their default values.

Qphot can be driven by the image cursor or a coordinate list in interactive mode or by a coordinate list in batch mode in exactly the same manner as the **phot** task.

5.8. The POLYPHOT Task

Polyphot computes accurate centers, sky values and magnitudes for a list of objects using polygonal shaped apertures. It is most suitable for measuring the magnitudes of large extended irregularly shaped objects.

Polyphot uses **datapars**, **centerpars**, and **fitskypars** in exactly the same manner as the **phot** task. In general users should set the task parameters in the same way as they set them in **phot**. However users who have defined their polygonal apertures interactively may wish to set the centering algorithm parameter **calgorithm** = "none", to avoid the task trying to center on the brightest feature in the aperture.

The apertures are defined either interactively with the image or graphics cursor or by two files **polygons** and **coords**. Polygons defines the shape of the polygonal aperture and coords defines the initial positions of the apertures. The polygons file may contain more than one

aperture and the flux through each aperture may be measured at more than one position. A detailed description of the file formats is given in section 6.4

6. Creating A Coordinate List

All APPHOT tasks operate on either lists of object coordinates or interactive cursor input. Lists are maintained as text files, one object per line with the x and y coordinates in columns one and two respectively. The coordinate and polygon files required by the **polyphot** task have a different format which is described below. List files may be created interactively with either the graphics or the image cursor, by a previously executed APPHOT task, by a previously executed IRAF task or by a user program. Various means of creating coordinate lists within IRAF are described below. Comments preceded by a # character and blank lines are ignored.

```
#Sample Coordinate List
   53.6    83.25
   100.0   35.8
   2.134   86.89
   ....    ....
```

6.1. Daofind

Daofind is an APPHOT task which detects stellar objects in an image automatically. The user sets the **fwhmpsf** of the psf for which the detection algorithm is to be optimized as well as an intensity threshold for detection. **Daofind** locates all the qualifying stars and writes their positions, rough magnitudes and shape characteristics to a file. This file can then be assigned to the **phot** task **coords** parameter and read directly.

For example if we have an image containing stars for which the **fwhmpsf** is 4.0 pixels and the sigma of the sky background is 10 we might run **daofind** as follows,

```
cl> daofind image fwhmpsf=4.0 threshold=30.0
```

where we have set our detection threshold at 3.0 * sigma.

6.2. Imtool On the SUN Machines

The SUN IRAF **imtool** facility supports both image world coordinate systems and output coordinate files. Coordinate lists can be created interactively by the users in the following way.

Display the IRAF image in the imtool window using the **display** task. Move the mouse to the top of the **imtool** window, press the right mouse button to enter the **imtool** menu, move the mouse to the setup option and release the mouse button. Press the return key until the black triangle is opposite the coordinate list file name parameter. Delete the default file name, enter the full host system path name of the desired coordinate file and press return. This name should now appear at the top of the imtool window. Move the mouse to the quit option and press the left mouse button to quit the setup window.

To enter the **imtool** cursor readout mode type the **F6** key. The x, y and intensity values at the cursor position are displayed in the lower right corner of the image. To mark stars and output their coordinates to the coordinate file, move the image cursor a star and press the left mouse button. A sequence number will appear on the display next to the marked position. The numbers can be changed from black to white and vice versa by toggling the **F5** key. The coordinate files are opened in append mode in order that stars may be added to an already existing list. **Imtool** coordinate files are directly readable by all APPHOT tasks.

6.3. Rgcursor and Rimcursor

The LISTS package tasks **rgcursor** and **rimcursor** can be used to generate coordinate lists interactively. For example a coordinate list can be created interactively using the display cursor and the image display.

```
cl> display image

... image appears on the display ...

cl> rimcursor > image.coo

... move display cursor to stars of interest and tape space bar ...

... type ^Z to terminate the list ...
```

Similarly a coordinate list can be created using the graphics cursor and a contour plot as shown below.

```
cl> contour image

... contour plot appears on the terminal ...

cl> rgcursor > image.coo

... move cursor to stars of interest and tap space bar ...

... type ^Z to terminate the list ...
```

The text file "image.coo" contains the x and y coordinates of the marked stars in image pixel units. The output of **rimcursor** or **rgcursor** can be read directly by the APPHOT **phot** task. **Rimcursor** is only available in IRAF versions 2.7 and later and only for selected devices.

6.4. The Polygon List

A utility routine **polymark** has been added to the APPHOT package to generate polygon and initial center lists for the **polyphot** task. The format of the polygon files is 1 vertex per line with the x and y coordinates of the vertex in columns 1 and 2 respectively. A line containing the single character ';' terminates the lists of vertices. There can be more than one polygon in a single polygon file.

```
Sample Polygon File

    1.0   1.0
    1.0   51.0
    51.0  51.0
    51.0  1.0
    ;
    80.0  80.0
    80.0  131.0
```

```
131.0 131.0
131.0 80.0
;
```

The accompanying coordinate file is optional. If no coordinate file is given the initial center for the polygon is the mean of its vertices in the polygon file. If a coordinate file is specified the initial center for the polygon is the position in the coordinate file. Each polygonal aperture may be moved to several positions.

```
Sample Polyphot Coords File

        50. 30.
        80. 100.
        60. 33.
        ;
        90. 50.
        55. 90.
        12. 122.
        ;
```

For example all the coordinates in group 1 will be measured using the aperture defined by polygon 1 and all the coordinates in group 2 will be measured with the aperture defined by polygon 2.

6.5. User Program

Obviously any user program which produces a text file with the coordinates listed 1 per line with x and y in columns 1 and 2 can be used to produce APPHOT coordinate files.

6.6. Modifying an Existing Coordinate List

The LISTS package routine **lintran** has been linked into the APPHOT package. It can be used to perform simple coordinate transformations on coordinate lists including shifts, magnifications, and rotations.

7. Running Apphot in Interactive Mode Without a Coordinate List

There are currently three ways to run the **phot** interactively without a coordinate list: 1) read image display cursor commands 2) read graphics cursor commands 3) read commands from the standard input. The three methods are briefly discussed below. Detailed examples of all three methods of operation can be found in the manual pages for each task.

7.1. Reading Image Cursor Commands

The default method of running APPHOT. The user loads an image onto the display, types **phot** and enters the image name. The image cursor appears on the display and the program is ready to accept user commands. This option is not available under IRAF version 2.6 and earlier because interactive image cursor readback was not available.

7.2. Redirecting the Image Cursor to the Graphics Cursor

Phot reads the graphics cursor and executes various keystroke commands. The environment variable **stdimcur** must be set to "stdgraph". For full access to all the graphics commands the parameter **display** must also be set to "stdgraph". The user creates a contour plot of the image on the graphics terminal with the **contour** task, types **phot** and answers the image name query. The graphics cursor appears on the contour plot ready for input. The user can move around the plot with the cursor successively marking stars.

7.3. Redirecting the Image Cursor to the Standard Input

The user can enter cursor commands directly on the standard input. The environment variable **stdimcur** must be set to "text". When the user types the task name **phot** and enters the image name, the following prompt appears.

```
Image cursor [xcoord ycoord wcs] key [cmd]:
```

Xcoord and *ycoord* are the coordinates of the object of interest, *wcs* is the current world coordinate system, always 1, *key* is a single character and *cmd* is an APPHOT task command. To perform the default action of the **phot** task the user responds as follows.

```
Image cursor [xcoord ycoord wcs] key [cmd]: 36. 42. 1
```

Phot measures the magnitude of the star near pixel coordinates x,y = (36.,42.) and writes the result to the output file. In IRAF version 2.5 all the cursor command fields must be typed. The square brackets indicate those fields which are optional under IRAF version 2.6 and later. Users with SUN workstations may wish to combine the IMTOOL coordinate list cursor readback facilities which generate coordinate lists with this mode of running APPHOT interactively.

7.4. The Interactive Keystroke Commands

A conscious effort has been made to keep the definitions of all the keystroke commands within the APPHOT package as similar as possible. The following are the most commonly used keystrokes in the APPHOT package.

7.4.1. The ? (Help) Keystroke Command

The ? key prints the help page describing the cursor keystroke and colon show commands for the specific APPHOT task. An abbreviated help page is typed by default when a user enters a task in interactive mode. The ? key can be typed at any point in the APPHOT task.

7.4.2. The :show (Set and Print parameter) Commands

Any APPHOT parameter can be displayed by typing :*parameter* command in interactive mode. For example to show the current value of the **fwhmpsf** parameter type the following command.

```
:fwhmpsf
```

To set any APPHOT parameter type :*parameter* "value". For example to set the **fwhmpsf** to 2.0 type.

```
:fwhmpsf 2.0
```

To display all the centering parameters type.

```
:show center
```

Similarly the sky fitting and photometry parameters can be displayed by typing.

```
:show sky
:show phot
```

All the parameters can be displayed with the following command.

```
:show
```

7.4.3. The i (Interactive Setup) Keystroke Command

This extremely useful key allows one to set up the principal APPHOT parameters interactively. To use this feature move the image cursor to a star on the display, or move the graphics cursor to a star on the contour plot and tap the i key, or enter the x and y coordinates, and the world coordinate system of the star and the i key manually. The program will query the user for the size of the extraction box and plot a radial profile of the star on the terminal. The user sets the **fwhmpsf**, the centering aperture **cbox**, the inner sky annulus **annulus** and **dannulus**, the list of apertures **aperts** and the data **sigma**, **threshold** and **cthreshold** using the graphics cursor and the radial profile plot. The cursor comes up on the plot at the position of the appropriate parameter. Typing return will preserve the old value. If the cursor is outside the range of values on the plot the old value is kept.

Setting **sigma**, **cthreshold**, and **threshold** interactively requires two keystrokes. In each case the measured parameter is the difference between the two y coordinates of the graphics cursor. It is recommended in general that the user leave **cthreshold** at zero.

7.4.4. The v (Verify) Keystroke Command

The v key verifies that the values of the critical task parameters currently in memory are the ones that the user wants. To each verify query the user either types CR to verify the current value or enters a new value.

7.4.5. The w (Write to Psets) Keystroke Command

The w key writes the current values of the parameters in memory to the appropriate psets. This feature is useful for saving values marked with the i key. On exiting APPHOT a prompt will remind the user that the current parameters in memory must be stored in the psets or lost.

7.4.6. The d (Radial Profile Plot) Keystroke Command

The d key plots a centered radial profile of a star on the graphics device.

7.4.7. The f (Fit) Keystroke Command

This key performs the default action of each APPHOT task without writing any results to the output file. In the **phot** task the f key will center, fit the local sky and compute the magnitudes for a star. This key allows the user to experiment interactively with the data, changing the default parameters, remeasuring magnitudes and so on before actually writing out any data.

7.4.8. The Space Bar (Fit and Write out Results) Keystroke Command

This key performs the default action of the task and writes the results to the output catalog.

8. Running Apphot In Interactive Mode From A Coordinate List

This is currently the best method for running APPHOT interactively for users without image cursor readback facilities. APPHOT tasks can pick stars out of the list sequentially or by number, measure stars sequentially or by number, rewind the coordinate lists and remeasure all the stars. Stars which are not in the coordinate list can still be measured and added to the output catalog.

[1] The :m (Move) Keystroke Command

This command moves the cursor to the next or a specified star in the coordinate list. If the hardware cursor on the device being read from is enabled the actual physical cursor will move to the requested star. For example a user might decide that star # 10 in the coordinate list is the best setup star. He/she simply types a :m 10 to move to the star in question followed by the i key to setup the parameters interactively.

[2] The :n (Next) Keystroke Command

This command moves to the next or specified star in the list, performs the default action of the task and writes the results to the output file. This key is particularly useful in examining the results of a large batch run. For example, a user measures the magnitudes of 500 stars using APPHOT in batch mode. He/she is suspicious about the results for twenty of the most crowded stars. By rerunning APPHOT in interactive mode using the original coordinate list, the user can selectively call up the stars in question, plot their radial profiles, and examine the results interactively.

[3] The l (List) Keystroke Command

This command measures all the stars in the coordinate list sequentially from the current position in the file.

[4] The r (Rewind) Keystroke Command

This command rewinds the coordinate list.

9. Running Apphot in Batch Mode

This is the simplest way to run APPHOT. Once the parameters are set and stored in the pset files the program can be run in batch by setting the parameter **interactive** = no. The program will read the coordinate list sequentially computing results for each star and writing them to the output file.

10. Apphot Output

APPHOT tasks write their output to text and/or plot files as well as to the standard output and graphics terminals.

10.1. Text Files

All APPHOT records are written to text files. The parameters for the task are listed at the beginning of each APPHOT output text file and identified with a #K string. The header record is not written until the record for the first star is to be written to the database. Parameter changes will generate a one line entry in the output text file. The data records follow the header record in the order in which they were computed. If the output file parameter **output** = "" no output file is written. This is the default action for the **radprof** task. If **output** = "default", the output file name is constructed using the image name.

10.2. Plot Files

Some APPHOT tasks can optionally produce graphics output. These files are maintained as plot metacode and may contain many individual plots. A directory of plots in each metacode file can be obtained with **gkidir**. Individual plots can be extracted with **gkiextract** and combined and plotted with **gkimosaic**.

10.3. Running Apselect on Apphot Output Files

Individual fields can be extracted from the APPHOT output files using the **apselect** task and a keyword and expression scheme. For example the user may wish to extract the x and y center coordinates, the sky value and the magnitudes from the APPHOT **phot** catalog. Using apselect they would type.

```
cl> apselect output xc,yc,msky,mag yes > magsfile
```

The selected fields would appear in the textfile "magsfile".

11. Apphot Recipes

In the following section three APPHOT reduction sessions which illustrate different methods of using the APPHOT package are described. In the first example the user wishes to compute magnitudes for a large number of stars in a single image. In the second example he/she wishes to measure the magnitude of a single standard star in each of a long list of images. Finally in the last example the user wishes to measure the magnitude of an elliptical galaxy through a list of apertures. Each example assumes that the user has started with the default set of package parameters.

11.1. Infrared photometry of a Star Field in Orion

An observer has an IRAF image on disk of a star forming region in Orion taken with the IR CCD camera. The Orion image is a composite formed from 64 separate IR images using the PROTO **irmosaic** and **iralign** tasks. The Orion image contains about 400 stars but is only moderately crowded. The observer decides to use the APPHOT package to reduce his data.

The observer decides to run the **daofind** routines to create a coordinate list of stars, to run **phot** in interactive mode with the image cursor directed to the standard input to setup and store the **phot** task parameters and finally, to run **phot** in batch mode to do the photometry.

To create the coordinate list the user needs to supply the full width half maximum of the pointspread function and an intensity threshold above background to the **daofind** program. Using the PLOT package task **implot** the user examines several stars in the image and decides that the **fwhmpsf** should be 3.0 pixels and that the standard deviation of the background should be 10.0 counts. The user decides to include all stars with a peak intensity greater than five standard deviations above local background in the coordinate list. The user runs **daofind** as

follows.

```
ap> daofind orion fwhmpsf=3.0 threshold=50.0
```

The x and y coordinates, a magnitude estimate and some statistics on image sharpness and roundness are output to the file "orion.coo.1". The user can obtain a printout of the coordinate list by typing.

```
ap> lprint orion.coo.1
```

Next the user decides to set up the parameters of the **phot** task. Using the **epar** task he enters the phot task parameter menu, types "orion" opposite the **image** parameter and "orion.coo.1" opposite the **coords** parameter. Next he moves the cursor opposite the **datapars** parameter and types **:e** to enter the **datapars** menu. He sets the gain parameter **epadu** to 5.0 electrons per adu and the readout noise **readnoise** to 5.0 electrons. He types **:q** to quit and save the **datapars** parameters and **^Z** to quit and save the **phot** parameters.

Now the user is ready to enter the **phot** task in interactive mode to set up the remaining data dependent parameters. The user types the following sequence of commands in response to the cursor prompt. Note that the example below assumes that the image cursor has been directed to the standard input. The image cursor comes up on the screen in the form of a prompt. This example could equally well be run from the image hardware cursor in which case the cursor would appear on the displayed image. The keystroke commands are identical in the two cases.

```
ap> phot orion
```

```
...
```
load the phot task ...

Image cursor [xcoord ycoord wcs] key [cmd]: ?

... print help page for phot task ...

Image cursor [xcoord ycoord wcs] key [cmd]: :radplots yes

... enable radial profile plotting ...

Image cursor [xcoord ycoord wcs] key [cmd]: :m 10

... move to star 10 in the coordinate list ...

Image cursor [xcoord ycoord wcs] key [cmd]: i

... enter interactive setup mode using star 10 ...

... mark fwhmpsf, cbox, sky annulus, apertures on the plot ...

... leave sigma and cthreshold at their default values ...

... check answers on radial profile plot of the results ...

Image cursor [xcoord ycoord wcs] key [cmd]: v

... verify the critical parameters ...

Image cursor [xcoord ycoord wcs] key [cmd]: :radplots no

... disable radial profile plotting ...

Image cursor [xcoord ycoord wcs] key [cmd]: w

... store new parameters in the psets ...

Image cursor [xcoord ycoord wcs] key [cmd]: q

... quit interactive cursor loop ...

q

... quit the phot task ...

The user decides he is happy with the current parameter set and decides to run the **phot** task in batch and submit it as a background task.

```
ap> phot orion inter- verify- &
```

The results will appear in the text file "orion.mag.1".

11.2. Landolt Standards

The user has 100 UBVRI images of 20 Landolt Standards taken on a single night. These frames have been taken at various short exposures. The user wishes to process all this data in batch mode with the output records going to a single file.

The observer decides to run the **daofind** task to create a coordinate list for each image, to run **phot** on a representative image in interactive mode with the image cursor directed to the standard input to set up and store the **phot** task parameters, and finally to run **phot** in batch mode on all the images to do the photometry.

Note that the example below assumes that the image cursor has been redirected to the standard input. The image cursor comes up on the screen in the form of a prompt. This example could equally well be run from the image hardware cursor in which case the cursor would appear on the displayed image. The keystroke commands are identical in the two cases.

To create the coordinate list the user needs to supply the full width half maximum of the point spread function and an intensity threshold above local background to the **daofind** task. Using the PLOT package task **implot** the user examines the Landolt standards in several images and decides that the average **fwhmpsf** is 4.3 pixels and that the standard deviation of the background is 15.0 counts. Note that if the user has access to an image display **fwhmpsf** and **threshold** can be determined interactively from inside the daofind task itself. The user decides to include all stars with a peak intensity greater than five standard deviations above local background in the coordinate list, which should include weak and spurious objects. The user runs **daofind** as follows.

```
ap> daofind lan*.imh fwhmpsf=4.3 threshold=75.0 verify- &
```

The x and y coordinates, an initial guess at the magnitude and some sharpness and roundness characteristics are output to the files "lan*.coo.1". For example the image lan100350.imh will now have a corresponding coordinate file lan100350.coo;1. The user may wish at this point to quickly check the coordinate files for spurious objects.

Next the user decides to set up the parameters of the phot task. Using the **epar** task he enters the phot task parameter set, enters "lan100350b" opposite the **image** parameter and "lan100350b.coo.1" opposite the **coords** parameter. Next he moves the cursor opposite the **datapars** parameter and types **:e** to enter the **datapars** menu. He sets the gain parameter **epadu** to 10 electrons per adu and the readout noise **readnoise** to 50.0 electrons. Finally in order to normalize all the magnitudes the user enters the image header exposure time keyword "itime" opposite the **exposure** parameter. He types **:q** to quit and save the **datapars** parameters and **^Z** to quit and save the **phot** parameters.

Now the user is ready to enter the **phot** task in interactive mode to set up the remaining data dependent parameters. The user types the following sequence of commands in response to the cursor prompt.

```
ap> phot lan100350.imh
```

`...` *load the phot task ...*

Image cursor [xcoord ycoord wcs] key [cmd]: ?

... print help page for phot task ...

Image cursor [xcoord ycoord wcs] key [cmd]: :radplots yes

... enable radial profile plotting ...

Image cursor [xcoord ycoord wcs] key [cmd]: :m #n

... move to star n in the coordinate list ...

Image cursor [xcoord ycoord wcs] key [cmd]: i

... enter interactive setup mode using star n ...

... mark fwhmpsf, cbox, sky annulus, apertures on the plot ...

... leave sigma and cthreshold at their default values ...

... check answers on radial profile plot of the results ...

Image cursor [xcoord ycoord wcs] key [cmd]: v

... verify the critical parameters ...

Image cursor [xcoord ycoord wcs] key [cmd]: :radplots no

... disable radial profile plotting ...

Image cursor [xcoord ycoord wcs] key [cmd]: w

... store new parameters in the psets ...

Image cursor [xcoord ycoord wcs] key [cmd]: q

... quit interactive cursor loop ...

q

... quit the phot task ...

Finally the user runs the **phot** task on the full list of images, with their corresponding parameter sets and dumps the output to a single text file named "output".

```
ap> phot lan*.imh coords="lan*.coo.*" output=output veri- inter- &
```

The results will appear in the text file "output".

11.3. Aperture Photometry of an Elliptical Galaxy

The user has a single image of the elliptical galaxy N315. She wishes to measure the magnitude of this galaxy through a list of apertures equal to those published in a well known catalogue of photoelectric photometry. Her data has been sky subtracted to give an average background value of 0.0 and the standard deviation of the sky background is 20.0 counts. From the published aperture photometry and the scale of her telescope she knows that she wishes to measure the galaxy through aperture radii of 10.5, 15.2, 20.8, and 25.6 arc seconds.

The user wishes to work in interactive mode using a contour plot of the image and the graphics cursor to enter commands to the **phot** task. She therefore sets the image cursor to "stdgraph" as follows.

```
ap> set stdimcur = stdgraph
```

Next she makes a contour plot of the image and writes it to a plot metacode file as follows.

```
ap> contour N315
```

`...` *contour plot appears on the terminal ...*

ap> =gcur

... enter cursor mode ...

:.write n315.plot

... write the plot to a file ...

q

... exit cursor mode ...

Now the user is ready to set up the parameters of the **phot** task. Since she already knows the values of the parameters she wishes to use she types

```
ap> epar phot
```

to enter the phot task menu. She positions the cursor opposite **image** parameter and types

"N315", opposite **dispay** and types "stdgraph". Next she moves the cursor opposite the **datapars** parameter and types **:e** to enter the **datapars** menu. She makes sure that **scale** = 0.75 arcseconds per pixel the scale of the telescope, sets the standard deviation of the sky background **sigma** to 20.0, sets the gain parameter **epadu** to 5 electrons per adu and the readout noise **readnoise** to 5.0 electrons. She types **:q** to quit and save the **datapars** parameters. She decides to leave the centering parameters in **centerpars** at their default values. **Cbox** in particular will be 5.0 arcseconds wide. The user has already removed a low order polynomial sky background from this image. She wishes to fix the sky value at 0.0. She moves the cursor opposite the **fitskypars** parameter and types **:e** to enter the sky fitting menu. She types "constant" opposite the **salgorithm** parameter, "0.0" opposite the **skyvalue** parameter and **:e** to exit the sky fitting parameter menu. Finally she enters the **photpars** menu and enters the aperture string "10.5,15.2,20.8,25.6" opposite the **apert** parameter.

To measure the galaxy she types

```
ap> phot N315
```

to enter the phot task, positions the cursor on the center of the galaxy in the contour plot and taps the space bar to make the measurement. The results are written to the file "N315.mag.1".