A User's Guide to Stellar CCD Photometry with IRAF

Philip Massey Lindsey E. Davis

March 29, 1990

Abstract

This document is intended to guide you through the steps for obtaining stellar photometry from CCD data using IRAF. It deals both with the case that the frames are relatively uncrowded (in which case simple aperture photometry may suffice) and with the case that the frames are crowded and require more sophisticated point-spread-function fitting methods (i.e., **daophot**). In addition we show how one goes about obtaining photometric solutions for the standard stars, and applying these transformations to instrumental magnitudes.

Contents

1 Introduction

This user's guide deals with both the "relatively simple" case of isolated stars on a CCD frame (standard stars, say, or uncrowded program stars) and the horrendously more complicated case of crowded field photometry. We describe here all the steps needed to obtain instrumental magnitudes and to do the transformation to the standard system. There are, of course, many possible paths to this goal, and IRAF provides no lack of options. We have chosen to illuminate a straight road, but many side trails are yours for the taking, and we will occasionally point these out ("let many flowers bloom"). This Guide is *not* intended as a reference manual; for that, you have available (a) the various "help pages" for the routines described herein, (b) "A User's Guide to the IRAF APPHOT Package" by Lindsey Davis, and (c) "A Reference Guide to the IRAF/DAOPHOT Package" by Lindsey Davis. (For the "philosophy" and algorithms of DAOPHOT, see Stetson 1987 *PASP* **99**, 111.) What *this* manual is intended to be is a real "user's guide", in which we go through all of the steps necessary to go from CCD frames to publishable photometry. (N.B.: as of this writing the IRAF routines for determining the standard transformations and applying those transformations are still being written.) We outline a temporary kludge that will work with Peter Stetson's CCDRED VMS Fortran package. Hopefully the PHOTRED package currently under development at Cerro Tololo will be available by Spring 1990, and this manual will then be revised.

The general steps involved are as follows: (1) fixing the header information to reflect accurate exposure times and airmasses, (2) determining and cataloging the characteristics of your data (e.g., noise, seeing, etc.), (3) obtaining instrumental magnitudes for all the standard stars using aperture photometry, (4) obtaining instrumental magnitudes for your program stars using IRAF/daophot, (5) determining the aperture correction for your program stars, (6) computing the transformation equations for the standard star photometry, and (7) applying these transformations to your program photometry. We choose to illustrate these reductions using *UBV* CCD data obtained with an RCA chip on the 0.9-m telescope at Cerro Tololo, but the techniques are applicable to data taken with any detector whose noise characteristics mimic those of a CCD.

If you are a brand-new IRAF user we strongly recommend first reading the document "A User's Introduction to the IRAF Command Language" by Shames and Tody, which can be found in Volume 1A of the 4 blue binders that compose the IRAF documentation. (Actually if you are a brand-new IRAF user one of us recommends that you find some simpler task to work on before you tackle digital stellar photometry!) The procedures described here will work on any system supported by IRAF; for the purposes of discussion, however, we will assume that you are using the image display capabilities of a SUN. If this is true you then may also want to familiarize yourself with the ins and outs of using the SUN Imtool window; the best description is to be found in "IRAF on the SUN".

We assume that your data has been read onto disk, and that the basic instrumental signature has been removed; i.e., that you are ready to do some photometry. If you haven't processed your data this far yet, we refer you to "A User's Guide to Reducing CCD Data with IRAF" by Phil Massey.

2 Getting Started

2.1 Fixing your headers

You're going to have to this some time or another; why not now? There are two specific things we may need to fix at this point: (a) Add any missing header words if you are reducing non-NOAO data, (b) correct the exposure time for any shutter opening/closing time, and (c) correct the airmass to the effective middle of the exposure.

Two things that will be useful to have in your headers are the exposure time and the airmass. If you are reducing NOAO data then you will already have the exposure time (although this may need to be corrected as described in the next paragraph) and enough information for the **setairmass** task described below to compute the effective airmass of the observation. You can skip to the "Correcting the exposure time" section below. If you are reducing non-NOAO data you should examine your header with a

imhead imagename**l+ | page**

and see exactly what information *is* there. If you are lacking the exposure time you can add this by doing an

hedit imagename**"ITIME"** value **add+ up+ ver- show+**

If you know the effective airmasses you can add an "AIRMASS" keyword in the same manner, or if you want to compute the effective airmass (corrected to mid-exposure) using **setairmass** as described below, you will need to have the celestial coordinates key words "RA" and "DEC", as well as the siderial-time ("ST"), and preferably the coordinate "EPOCH" and the date-of-observation ("DATE-OBS"), all of which should have the form shown in Figure 1.

You may want to take this opportunity to review the filter numbers in the headers, and fix any that are wrong. If you are lacking filter numbers you may want to add them at this point.

2.1.1 Correcting the exposure time

The CTIO 0.9-m has an effective exposure time that is 25-30ms longer than the requested exposure time (Massey et al. 1989 *A.J.* **97**, 107; Walker 1988 *NOAO Newsletter* **No. 13**, 20). First see what "keyword" in your header gives the exposure time:

imhead imagename**.imh l+ | page**

will produce a listing such as given in Figure 1. The exposure time keyword in this header is "ITIME". In this case we wish to add a new exposure time to each of the headers; we will call this corrected exposure time EXPTIME, and make it 25 ms larger than whatever value is listed as ITIME. To do this we use the **hedit** command as follows:

hedit *.imh EXPTIME "(ITIME+0.025)" ver- show+ add+.

An inspection of the headers will now show a new keyword EXPTIME. (Walker lists a similar correction for the CTIO 1.5-m shutter, but the CTIO 4-m P/F shutters have a negligible correction. The direct CCD shutters on the Kitt Peak CCD cameras give an additional 3.5ms of integration on the edges but 13.0ms in the center [e.g., Massey 1985 *KPNO Newsletter* **36**, p. 6]; if you have any 1 second exposures you had best correct these by 10ms or so if you are

```
n602alu.imh[316,500][short]: u h68
    No bad pixels, no histogram, min=-11., max=17421.
    Line storage mode, physdim [512,500], length of user area 649 s.u.
    Created Sun 14:35:46 12-Nov-89, Last modified Sun 14:35:47 12-Nov-89
    Pixel file 'tofu!/data/massey/pixels/n602alu.pix' [ok]
    'KPNO-MTN'            / institution of origin
    DATA-TYP= '    object (  0 )'   / object,dark,comp,etc.
    ITIME   =                  700  / integration time secs
    UT      = ' 1:54:12.00'  /  universal time
    ZD      = ' 0:38:46.00'  /  zenith distance
    DATE-OBS= '28/11/85'            / dd/mm/yy observation
    ST      = ' 1:38:46.00'  /  sidereal time
    RA      = ' 1:28:52.00'  /  right ascension
    DEC     = '-73:40:50.00'  /  declination
    EPOCH   =                1975.  / epoch of RA and DEC
    CAM-TEMP=                  .00  / camera temperature, deg C
    DEW-TEMP=                  .00  / dewar temp, deg C
    HISTORY1= 'bt=   120 bp=     0 cr=     0 dk=     0 '
    HISTORY2= 'ff=   903 fg=     0 sc=   .000  bi=  900  '
    COMMENT = 'u h68'
    F1POS   =                    5  / filter bolt I position
    F2POS   =                    0  / filter bolt II position
```

Figure 1: Header information for image n602alu.imh

.

```
   images = "*.imh"          Input images
  (intype = "beginning")     Input keyword time stamp
 (outtype = "effective")     Output airmass time stamp\n
    (date = "date-obs")      Observation date keyword
(exposure = "exptime")       Exposure time keyword (seconds)
 (airmass = "airmass")       Airmass keyword (output)
(utmiddle = "utmiddle")      Mid-observation UT keyword (output)
(latitude = -30.1652)        Observatory latitude\n
    (show = yes)             Print the airmasses and mid-UT?
  (update = yes)             Update the image header?
(override = yes)             Override previous assignments?
    (mode = "ql")
```

Figure 2: The parameter file for **setairmass**.

interested in 1% photometry.)

2.1.2 Computing the effective airmass

The task **setairmass** in the **astutil** package will compute the effective airmass of your exposure, using the header values of RA, DEC, ST, EPOCH, and DATE-OBS, and whatever you specify for the observatory latitude. An example is shown in Figure 2. The default for the latitude is usually the IRAF variable **observatory.latitude**. To by-pass this "feature", simply put the correct latitude in the parameter file (e.g., −30.1652 for CTIO, +31.963 for KPNO; +19.827 for Mauna Kea.).

2.2 imexamine: A Useful Tool

The **proto** package task **imexamine** is a powerful and versatile task which can be used to interactively examine image data at all stages of the photometric reduction process. In this section we discuss and illustrate those aspects of **imexamine** which are most useful to photometrists with emphasis on three different applications of the task: 1) examining the image, for example plotting lines and columns 2) deriving image characteristics, for example computing the FWHM of the point-spread function 3) comparing the same region in different images.

The task **imexamine** lives within the **proto** package, and you will also need to load **images** and **tv**. Then **display** the image, and type **imexamine**. When the task is ready to accept input the image cursor will begin blinking in the display window, and the user can begin executing various keystroke and colon commands. The most useful data examining commands are summarized below. The column, contour, histogram, line and surface plotting commands each have their own parameter sets which set the region to be plotted and control

the various plotting parameters. All can be examined and edited interactively from within the **imexamine** task using the appropriate **:epar** command.

c - Plot the column nearest the image cursor

e - Make a contour plot of a region around the image cursor

h - Plot the histogram of a region around the image cursor

l - Plot the line nearest the image cursor

s - Make a surface plot of a region around the image cursor

:c N - Plot column N

:l N - Plot line N

x - Print the x, y, z values of the pixel nearest the image cursor

z - Print a 10 by 10 grid of pixels around the image cursor

o - Overplot

g - Activate the graphics cursor

i - Activate the image cursor

? - Print help

q - Quit **imexamine**

:epar c - Edit the column plot parameters

:epar e - Edit the contour plot parameters

:epar h - Edit the histogram plot parameters

:epar l - Edit the line plot parameters

:epar s - Edit the surface plot parameters

Example 1 below shows how a user can interactively make and make hardcopies of image line plots using **imexamine** and at the same time illustrates many of the general features of the task.

The **imexamine** task also has some elementary image analysis capability, including the capacity to do simple aperture photometry, compute image statistics and fit radial profiles. The most useful image analysis commands are listed below.

h - Plot the histogram of a region around the cursor

r - Plot the radial profile of a region around the cursor

m - Plot the statistics of a region around the cursor

:epar h - Edit the histogram parameters

:epar r - Edit the radial profile fitting parameters

Example 2 shows how a photometrist might use **imexamine** and the above commands to estimate the following image characteristics: 1) the full width at half maximum (FWHM) of the point-spread function, 2) the background sky level 3) the standard deviation of the background level 4) and the radius at which the light from the brightest star of interest disappears into the noise (this will be used to specify the size of the point-spread-function, e.g.,PSFRAD).

Finally **imexamine** can be used to compare images. Example 3 shows how to compare regions in the original image and in the same image with all the fitted stars subtracted out. The example assumes that the target image display device supports multiple frame buffers, i.e. the user can load at least two images into the display device at once.

The **imexamine** task offers even more features than are discussed here and the user should refer to the manual page for more details.

Example 1: Plot and make hardcopies of image lines within **imexamine**.

- **display** the image and then type **imexamine**.
- move the image cursor to a star and tap **l** to plot the image line nearest the cursor
- tap the **g** key to activate the graphics cursor
- type **:.snap** to make a hardcopy of the plot on your default device
- expand a region of interest by first moving the graphics cursor to the lower left corner of the region and typing **E**, and then moving the graphics cursor to the upper right corner of the region and typing anything
- type **:.snap** to make a hardcopy of the new plot
- tap the **i** key to return to the image cursor menu
- type **:epar l** to enter the line plot parameter set, change the value of the logy parameter to yes and type **CNTL-Z** to exit and save the change
- repeat the previous line plotting commands
- type **q** to quit **imexamine**

Example 2: Compute some elementary image characteristics using **imexamine**.

- **display** the image and then type **imexamine**.
- move to a bright star and tap the **r** key
- examine the resulting radial profile plot and note the final number on the status line which is the FWHM of the best fitting Gaussian
- repeat this procedure for several stars to estimate a good average value for the FWHM
- set the parameters of the statistics box ncstat and nlstat from 5 and 5 to 21 and 21 with **:ncstat 21** and **:nlstat 21** commands so that the sizes of the statistics and histogram regions will be identical
- move to a region of blank sky and tap the **m** key to get an estimate of the mean, median and standard deviation of the sky pixels in a region 21 by 21 pixels in size around the image cursor
- leave the cursor at the same position and tap the **h** key to get a plot of the histogram of the pixels in the same region
- tap the **g** key to activate the graphics cursor, move the cursor to the peak of the histogram and type **C** to print out the cursor's value. The "x" value then gives you a good estimate of the sky. Similarly, you can move the cursor to the half-power point of the histogram and type **C** to estimate the standard deviation of the sky pixels. Tap the **i** key to return to the image cursor menu
- compare the results of the h and m keys
- repeat the measurements for several blank sky regions and note the results
- move to a bright unsaturated star and turn up the zoom and contrast of the display device as much as possible
- using the **x** key mark the point on either side of the center where the light from the star disappears into the noise and estimate PSFRAD
- type **:epar r** to edit the radial profile fitting parameters and set rplot to something a few pixels larger than PSFRAD and tap the **r** key
- note the radius where the light levels off and compare with the eyeball estimate
- repeat for a few stars to check for consistency
- type **q** to quit imexamine

Example 3: Overplot lines from two different images.

- **imexamine image1,image2**

- move the image cursor to a star and type **z** to print the pixel values near the cursor
- tap the **n** key to display the second image followed by **z** to look at the values of the same pixels in the second image
- tap the **p** key to return to the first image
- tap **l** to plot a line near the center of the star and tap the **o** key to overlay the next plot
- tap the **p** key to return to the second image and without moving the image cursor tap the l key again to overplot the line
- type **q** to quit imexamine

2.3 Dealing with Parameter Files (Wheels within Wheels)

The **daophot** (and **apphot**) packages are unique in IRAF in that they obtain pertinent information out of separate "parameter files" that can be shared between tasks. As anyone that has used IRAF knows, each IRAF command has its own parameter file that can be viewed by doing an **lpar** *command* or edited by doing an **epar** *command*. However, in **daophot** and **apphot** there are "wheels within wheels"—some of the parameters are in fact parameter files themselves. For instance, the aperture photometry routine **phot** does not explicitly show you the methods and details of the sky fitting in its parameter file. However, if you do an **lpar phot** you will see a parameter called "fitskypars" which contains, among many other things, the radii of the annulus to be used in determining the sky value. You will also find listed "datapars" (which specifies the properties of your data, such as photons per ADU and read-noise), "centerpars" (which specifies the centering algorithm to be used), and "photpars" (which gives the size of the digital apertures and the zero-point magnitude). The contents of any of these parameter files can be altered either by **epar**ing them on their own or by typing a ":e" while on that line of the main parameter file. If you do the latter, a control-z or a ":q" will bring you back. For example, to examine or edit **fitskypars**, you can do an explicit **lpar fitskypars** or **epar fitskypars**, or you can do an **epar phot**, move the cursor down to the "fitskypars" line, and then type a **:e** to edit (see Figure 3). Confusing? You bet! But once you are used to it, it is a convenient and powerful way to specify a whole bunch of things that are used by several different commands—i.e., you are guaranteed of using the same parameters in several different tasks. If there is only one thing that you want to change in a parameter file you *can* enter it on the command line when you run the command, just as if it were a "normal" (hidden) parameter, i.e., **phot imagename dannulus=8.** does the same as running **epar fitskypars** and changing the "width of sky annulus" **dannulus** to 8.0.

Mostly these things are kept out of the way ("very hidden" parameters) because you *don't* want to be changing them, once you have set them up for your data. There are exceptions, such as changing the PSF radius in making a point-spread function in a crowded field (Sec. 4.6). However, you are well protected here if you leave the **verify** switch on. A task will then give you an opportunity to take one last look at anything that you really care about when you run the task. For instance, if we had simply run **phot** on an image (we'll see how to do

```
                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = phot

image   =                         Input image
coords  =              default  Coordinate list (default: image.coo.?)
output  =              default  Results file (default: image.mag.?)"
(datapar=                     ) Data dependent parameters
(centerp=                     ) Centering parameters
(fitskyp=                     ) Sky fitting parameters
(photpar=                     ) Photometry parameters
skyfile =                       Sky 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=                    no) Mode of use
(verify =                   yes) Verify critical parameters in non interactive mo
(update =                   yes) Update critical parameters in non interactive mo
(verbose=                   yes) Print messages in non interactive mode
(mode   =                    ql)

:e

                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = fitskypars

(salgori=                  mode) Sky fitting algorithm
(annulus=                  20.) Inner radius of sky annulus in scale units
(dannulu=                   3.) 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 iteratio
(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)
```

Figure 3: Changing the Sky Annulus in **fitskypars**.

this shortly) it would have said "Width of sky annulus (10.)", at which point we could either have hit [CR] to have accepted the 10., or we could have entered a new value.

3 Aperture Photometry on your Standards

Standard stars provide a good example of relatively uncrowded photometry, and in this section we will describe how to obtain instrumental magnitudes for your standards using **phot**. The basic steps are

- Decide what aperture size you wish to use for measuring your standards **(this should be the same for all the frames).** At the same time we will pick a sky annulus.
- Set up the various parameter files (**datapars, centerpars, fitskypars, photpars**) to have the correct values.
- For each frame:
    1. Identify the standard star(s) either interactively using a cursor or by using the automatic star finding algorithm **daofind**.
    2. Run the aperture photometry program **phot** on each of your standard star frames.

Although the routines you will need to use are available both in the **daophot** and **apphot** packages, we strongly advise you to run them from the **daophot** package: the default setup is somewhat different, and the two packages each have their own data parameter files.

3.1 Picking an Aperture Size

Unfortunately, there are not good tools available with IRAF to do this yet, and we will restrict our discussion here to some of the considerations before telling you to just go ahead and use a radius that is something like 4 or 5 times the FWHM of a stellar image; e.g., 12 or 15 pixels as a radius, assuming you have the usual sort of "nearly undersampled" FWHM$\approx$ 3 data. You might naively expect (as I did) that you wish to pick an aperture size that will "contain all the light" from your standard stars, but in fact this is impossible: the wings of a star's profile extend much further than you imagine at a "significant" level. King (1971 *Publ. A.S.P.* **83**, 199) and Kormendy (1973 *A.J.* **78**, 255) discuss the fact that on photographic plates the profile of a star extends out to *arcminutes* at an intensity level far exceeding the diffraction profile; Kormendy attributes this to scattering off of dust and surface irregularities on the optical surfaces. Massey *et al.* (1989 **97**, 107) discusses this in regards to CCD's and standard star solutions using the very data we are using here as an example (which is not exactly a coincidence). Although the intensity profile falls off rapidly, the increase in area with radius increases rapidly, and in practical terms Massey *et al.* found that in cases where the FWHM was typically small (2.5-3 pixels), increasing the digital aperture size from a diameter of 18 pixels to one of 20 pixels resulted in an additional 1-2% increase in light for a well-exposed

star, and that this increase continues for larger apertures until masked by the photometric errors.

Given that you presumably want 1% photometry or better, what should you do? Well, the fact that photoelectric photometery through fixed apertures in fact does work suggests that there is some radius beyond which the same fraction of light is excluded, despite variations in the seeing and guiding. You do not want to choose a gigantic aperture ($>$ 20 pixels, say) because the probability of your having a bad pixel or two goes up with the area. But you do not want to choose too small an aperture ($<$10 pixels, say) or you will find yourself at the mercy of the seeing and guiding. Most photoelectric photometrists will use an aperture of at least 10 arcseconds in diameter, but remember you have one advantage over them: you are not sensitive to centering errors, since any digital aperture can be exactly centered. If you have enough standard star observations (I used about 300 obtained over a 10 night run) you can compute magnitude differences between a large aperture (20 pixels), and a series of smaller apertures (8, 10, 12, 15, 18) for each filter, and then see for which radius the difference (in magnitudes) becomes constant. Unfortunately, there are no tools currently available within IRAF for taking the differences between two apertures, or for conveniently plotting these differences, so you are on your own. My recommendation would be that if you have typical data with a FWHM of $\leq$ 4 pixels, that you use something like an aperture of 12 to 15 pixels in radius for your standard stars. **You can save yourself a lot of trouble if you simply adopt a single radius for all the standards from all the nights for all filters.**

3.2 Setting Things Up

As discussed in "Dealing with Parameter Files" (Section 2.1) we must setup the parameter files from which **phot** will get the details of what it is going to do. The easiest way to do this is to simply **epar phot**, and on each of the four parameter lists to do a **:e**. Mostly we will leave the defaults alone, but in fact you will have to change at least one thing in each of the four files.

In **datapars** (Figure 4) we need to specify both the FWHM of a star image (*fwhmpsf*) and the threshold value above sky (*threshold*) if we are going to use the automatic star-finding routine **daofind**; the choices for these are discussed further below. In order to have realistic error estimates for our aperture photometry we need to specify the CCD readnoise *readnoise* in electrons and the gain (photons per ADU) for the CCD *epadu*. In order to correct the results for the exposure time we need the exposure time keyword *exposure*. Do an

imhead *imagename* **l+ | page**

to see a listing of all the header information (Figure 5). By specifying the (effective) airmass and filter keywords, these can be carried along in the photometry file for use when we do the standards solution (*airmass* and *filter*). Finally we use *datamin* and *datamax* so we will know if we exceeded the linearity of the CCD in the exposure, or whether there is some anomalously low valued pixel on which our star is sitting. Since the value of the sky on our standard exposures is probably nearly zero, *datamin* should be set to a negative value about three times the size of the readnoise in *ADU's*; e.g., $-3 \times 65. \div 2.25 \approx -90$ in this example. Note that although we will later argue that the shape of the PSF changes a little about 20000

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = datapars

(fwhmpsf=                  3.) FWHM of the PSF in scale units
(emissio=                 yes) Features are positive?
(noise  =             poisson) Noise model
(thresho=                500.) 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=                 65.) CCD readout noise in electrons
(gain   =                   ) CCD gain image header keyword
(epadu  =                2.25) Gain in electrons per count
(exposur=             EXPTIME) Exposure time image header keyword
(itime  =              INDEF) Exposure time
(airmass=             AIRMASS) Airmass image header keyword
(xairmas=              INDEF) Airmass
(filter =              F1POS) Filter image header keyword
(ifilter=              INDEF) Filter
(datamin=                -90.) Minimum good data value
(datamax=              32000.) Maximum good data value
(mode   =                 ql)
```

Figure 4: Parameters for **datapars**.

```
std159[316,500][short]: 98-653 v
    No bad pixels, no histogram, min=-312., max=30550.
    Line storage mode, physdim [512,500], length of user area 772 s.u.
    Created Sun 20:25:21 28-Jan-90, Last modified Sun 20:25:21 28-Jan-90
    Pixel file 'tofu!/data/massey/pixels/std159.pix' [ok]
    'KPNO-MTN'            / institution of origin
    DATA-TYP= '    object (  0 )'   / object,dark,comp,etc.
    ITIME   =                    1  / integration time secs
    UT      = ' 4:11:18.00'  /  universal time
    ZD      = ' 0:56:14.00'  /  zenith distance
    DATE-OBS= '28/11/85'            / dd/mm/yy observation
    ST      = ' 3:56:14.00'  /  sidereal time
    RA      = ' 6:50:45.00'  /  right ascension
    DEC     = ' 0:16:07.00'  /  declination
    EPOCH   =               1975.0  / epoch of RA and DEC
    CAM-TEMP=                  .00  / camera temperature, deg C
    DEW-TEMP=                  .00  / dewar temp, deg C
    HISTORY1= 'bt=   119 bp=     0 cr=     0 dk=     0 '
    HISTORY2= 'ff=   902 fg=     0 sc=   .000  bi=  900  '
    COMMENT = '98-653 v'
    F1POS   =                    3  / filter bolt I position
    F2POS   =                    0  / filter bolt II position
    EXPTIME =                1.025
    AIRMASS =             1.605717
    UTMIDDLE= '4:11:18.5'
```

Figure 5: Header information for std159.imh

```
PACKAGE = daophot
   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)
```

Figure 6: Parameters for **centerpars**.

ADU's (presumably due to some sort of charge-transfer problem), for the purposes of simple aperture photometry we are happy using 32000 ADU's as the maximum good data value. (We do not really want to use 32767 as afterall the overscan bias was probably at a level of several hundred.)

In **centerpars** (Figure 6) we need to change the centering algorithm *calgorithm* from the default value of "none" to "centroid". If the FWHM of your frames are unusually large (> 4, say, you would also do well to up the size of **cbox** to assure that the centering works well; make it something like twice the FWHM. In this case the FWHM is 3 pixels or a bit smaller, and we are content to leave it a the default setting of 5 pixels.

In **fitskypars** (Figure 7) the only things we must specify are the size and location of the annulus in which the modal value of the sky will be determined. If you are going to use a value of 15 for your photometry aperture, you probably want to start the sky around pixel 20. Keeping the width of the annulus large (5 pixels is plenty) assures you of good sampling, but making it too large increases the chances of getting some bad pixels in the sky.

In **photpars** (Figure 8) we merely need to specify the size (radius) of the aperture we wish to use in measuring our standards.

3.3 Doing It

There are basically two ways of proceeding in running photometry on the standard stars, depending upon how you are going to identify the relevant star(s) on each frame. If you have only one (or two) standard stars on each frame, and it is always one of the brightest stars present, then you can avoid a lot of the work and use the automatic star-finding program **daofind** to find all your standards and the whole thing can be done fairly non-interactively.

```
                                    I R A F
                        Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = fitskypars

(salgori=                     mode) Sky fitting algorithm
(annulus=                     20.) Inner radius of sky annulus in scale units
(dannulu=                      5.) 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 iteratio
(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)
```

Figure 7: Parameters for **fitskypars**.

```
                                    I R A F
                        Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = photpars

(weighti=                  constant) Photometric weighting scheme for wphot
(apertur=                      15.) List of aperture radii in scale units
(zmag   =                      25.) Zero point of magnitude scale
(mkapert=                       no) Draw apertures on the display
(mode   =                       ql)
```

Figure 8: Parameters for **photpars**.

However, if you are one of the believers in cluster field standards, then you may actually want to identify the standards in each field using the cursor on the image display so that the numbering scheme makes sense. We describe below each of the two methods.

3.3.1 Automatic star finding

First let's put the name of each frame containing standard stars into a file; if you've put the standard star exposures into a separate directory this can be done simply by a **files *.imh > stands**. This will leave us with funny default output file names for a while (we advise against including the ".imh" extension when we discuss crowded field photometry in the next section), but this will only be true for a short intermediate stage.

We want to run **daofind** in such a way that it finds only the brightest star or two (presumably your standard was one of the brightest stars in the field; if not, you are going to have to do this stuff as outlined below in the "Photometry by eye" section). We will delve more fully into the nitty-gritty of **daofind** in the crowded-field photometry section, but here we are content if we can simply find the brightest few stars. Thus the choice of the detection threshold is a critical one. If you make it too low you will find all sorts of junk; if you make it too high then you may not find any stars. You may need to run **imexamine** on a few of your images: first **display** the image, and then **imexamine**, using the "r" cursor key to produce a radial profile plot. Things to note are the typical full-width-half-maximum and the peak value. If your sky is really around zero for your standard exposures, then using a value that is, say, twenty times the readnoise (in ADU's) is nearly guaranteed to find only the brightest few stars; do your radial plots in **imexamine** show this to be a reasonable value? In the example here we have decided to use 500 ADUs as the threshold ($20 \times 65 \div 2.25 \approx 500$).

Now **epar daofind** so it resembles that of Figure 9. Go ahead and execute it (Figure 10). Note that since *verify* is on that you will be given a chance to revise the FWHM and detection threshold. By turning verbose on you will see how many stars are detected on each frame. Make a note of any cases where no stars were found; these you will have to go back and do with a lower threshold.

The run of **daofind** produced one output file named *imagename.imh.coo.1* for each input file. If you **page** one of these you will find that it resembles that of Figure 11. The file contains many lines of header, followed by the x and y center values, the magnitudes above the threshold value, the "sharpness" and "roundness" values, and finally an ID number. In the example shown here in Figure 11 two stars were found: one 2.9 mags brighter than our detection threshold, and one about 0.4 mag brighter than our detection threshold.

In a few cases we doubtlessly found more than one star; this is a good time to get rid of the uninteresting non-standards in each field. If things went by too fast on the screen for you to take careful notes while running **daofind** we can find these cases now: do a

txdump *coo* image,id,x,y yes

to get a listing of the location and number of stars found on each image. If you have cases where there were lots of detections (a dozen, say) you may find it easier to first **sort *.coo* mag** in order to resort the stars in each file by how bright they are. Of course, your standard may not be the brightest star in each field; you may want to keep an eye on the x and y

```
                                    I R A F
                      Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = daofind

image   =              @stands  Input image
output  =              default  Results file (default: image.coo.?)
(convolu=                    ) Output convolved image
(datapar=                    ) Data dependent parameters (fwhmpsf, threshold, e
(ratio  =                   1.) Ratio of sigmay to sigmax of Gaussian kernel
(theta  =                   0.) Position angle of major axis of Gaussian kernel
(nsigma =                  1.5) Width of convolution kernel in sigma
(sharplo=                  0.2) Lower bound on sharpness for feature detection
(sharphi=                   1.) Upper bound on sharpness for  feature detection
(roundlo=                  -1.) Lower bound on roundness for feature detection
(roundhi=                   1.) Upper bound on roundness for feature detection
(boundar=              nearest) Boundary extension (constant, nearest, reflect,
(constan=                   0.) Constant for boundary extension
(graphic=             stdgraph) Graphics device
(display=             stdimage) Display device
(command=                    ) Image cursor: [x y wcs] key [cmd]
(cursor =                    ) Graphics cursor: [x y wcs] key [cmd]
(mkdetec=                   no) Mark detected stars on the display
(interac=                   no) Interactive mode
(verify =                  yes) Verify critical parameters in non interactive mo
(update =                   no) Update critical parameters in non interactive mo
(verbose=                  yes) Print messages in non interactive mode
(mode   =                   ql)
```

Figure 9: Parameter file for **daofind**.

```
da> daofind @stands

FWHM of features in scale units (3.):
        New FWHM of features: 3. scale units  3. pixels
Detection threshold in counts above background (500.):
        New detection threshold: 500. counts


Image: std145.imh  fwhmpsf: 3.  ratio: 1.  theta: 0.  nsigma: 1.5

  116.87   262.16   -3.024   0.516  -0.031     1

1 stars detected threshold: 500.  0.2 <= sharp <= 1.  -1. <= round <= 1.


Image: std146.imh  fwhmpsf: 3.  ratio: 1.  theta: 0.  nsigma: 1.5

  116.99   265.17   -2.307   0.502  -0.258     1

1 stars detected threshold: 500.  0.2 <= sharp <= 1.  -1. <= round <= 1.


Image: std147.imh  fwhmpsf: 3.  ratio: 1.  theta: 0.  nsigma: 1.5

  114.80   260.13   -2.841   0.558  -0.043     1

1 stars detected threshold: 500.  0.2 <= sharp <= 1.  -1. <= round <= 1.


Image: std148.imh  fwhmpsf: 3.  ratio: 1.  theta: 0.  nsigma: 1.5

   58.91   183.13   -0.351   0.546   0.083     1
  135.09   252.93   -2.928   0.522  -0.049     2

2 stars detected threshold: 500.  0.2 <= sharp <= 1.  -1. <= round <= 1.
```

Figure 10: Screen output from a **daofind** run.

```
#K IRAF       = NOAO/IRAF V2.8Eversion   %-15s
#K USER       = massey         name      %-15s
#K HOST       = tofu           computer  %-15s
#K DATE       = 01-29-90       mm-dd-yr  %-15s
#K TIME       = 10:54:49       hh:mm:ss  %-15s
#K PACKAGE    = apphot         name      %-15s
#K TASK       = daofind        name      %-15s
#
#K SCALE      = 1.             units     %-15.7g
#K FWHMPSF    = 3.             scaleunit %-15.7g
#K EMISSION   = yes            switch    %-15b
#K DATAMIN    = -90.           counts    %-15.7g
#K DATAMAX    = 32000.         counts    %-15.7g
#K EXPOSURE   = EXPTIME        keyword   %-15s
#K AIRMASS    = AIRMASS        keyword   %-15s
#K FILTER     = F1POS          keyword   %-15s
#
#K NOISE      = poisson        model     %-15s
#K THRESHOLD  = 500.           counts    %-15.7g
#K CTHRESHOLD = 0.             counts    %-15.7g
#K SIGMA      = INDEF          counts    %-15.7g
#K GAIN       = ""             keyword   %-15s
#K EPADU      = 2.25           e-/adu    %-15.7g
#K CCDREAD    = ""             keyword   %-15s
#K READNOISE  = 65.            e-        %-15.7g
#
#K IMAGE      = std148.imh     imagename %-15s
#K FWHMPSF    = 3.             scaleunit %-15.7g
#K NSIGMA     = 1.5            sigma     %-15.7g
#K RATIO      = 1.             number    %-15.7g
#K THETA      = 0.             degrees   %-15.7g
#
#K SHARPLO    = 0.2            number    %-15.7g
#K SHARPHI    = 1.             number    %-15.7g
#K ROUNDLO    = -1.            number    %-15.7g
#K ROUNDHI    = 1.             number    %-15.7g
#
#N XCENTER  YCENTER  MAG      SHARP   ROUND   ID                              \
#U pixels   pixels   #        #       #       #                               \
#F %-12.2f  %-9.2f   %-9.3f   %-8.3f  %-8.3f  %-6d                            \
#
   58.91    183.13   -0.351   0.546   0.083   1
   135.09   252.93   -2.928   0.522   -0.049  2
```

Figure 11: Output file from **daofind**.

```
                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = phot

image   =              @stands  Input image
coords  =              default  Coordinate list (default: image.coo.?)
output  =           standstuff  Results file (default: image.mag.?)"
(datapar=                     ) Data dependent parameters
(centerp=                     ) Centering parameters
(fitskyp=                     ) Sky fitting parameters
(photpar=                     ) Photometry parameters
skyfile =                       Sky 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=                    no) Mode of use
(verify =                   yes) Verify critical parameters in non interactive mo
(update =                   yes) Update critical parameters in non interactive mo
(verbose=                   yes) Print messages in non interactive mode
(mode   =                    ql)
```

Figure 12: The parameter file for a run of **phot**.

values to see if it is the star you thought you were putting in the middle! To get rid of the spurious stars you will need to **edit** each of the output files (e.g., **edit std148.imh.coo.1**) and simply delete the extras.

Finally we can run aperture photometry on these frames, using the ".coo" files to locate the standard star in each frame. **epar phot** until it resembles that of Figure 12. Note that we are specifying a *single* output file name ("standstuff" in this example); *all* the photometry output will be dumped into this single file, including things like the airmass and filter number. Go ahead and execute **phot**. You should see something much like that of Figure 13 on the screen. We will discuss the output below under "Examining the results".

3.3.2 Photometry by Eye

In this section we will discuss the case of selecting stars *without* running the automatic star-finding program, using the image display window and the cursor. The first step is to **epar**

```
da> phot @stands
Coordinate list (default: image.coo.?) (default):
Results file (default: image.mag.?)" (standstuff):

Centering algorithm (centroid):
        New centering algorithm: centroid
Centering box width in scale units (5.):
        New centering box width: 5. scale units  5. pixels
Centering threshold in counts above background (0.):
        New centering threshold: 0. counts
Sky fitting algorithm (mode):
        Sky fitting algorithm: mode
Inner radius of sky annulus in scale units (20.):
        New inner radius of sky annulus: 20. scale units 20. pixels
Width of the sky annulus in scale units (5.):
        New width of the sky annulus: 5. scale units 5. pixels
Standard deviation of background in counts (INDEF):
        New standard deviation of background: INDEF counts
File/list of aperture radii in scale units (15.):
        Aperture radius 1: 15. scale units 15. pixels

std145.imh x: 116.65 y: 262.34 s: 27.38 m: 13.801 e: ok
std146.imh x: 116.93 y: 265.30 s: 16.78 m: 16.047 e: ok
std147.imh x: 114.63 y: 260.23 s: 24.40 m: 13.910 e: ok
std148.imh x: 135.25 y: 252.78 s: 79.27 m: 14.820 e: ok
std149.imh x: 139.84 y: 254.18 s: 85.78 m: 14.613 e: ok
std150.imh x: 140.51 y: 255.30 s: 52.30 m: 16.963 e: ok
std151.imh x: 94.90 y: 158.68 s: 168.09 m: 14.916 e: ok
std152.imh x: 91.65 y: 153.92 s: 170.32 m: 15.118 e: ok
std153.imh x: 95.41 y: 156.13 s: 86.20 m: 16.997 e: ok
std154.imh x: 66.67 y: 101.90 s: 40.02 m: 13.429 e: ok
std155.imh x: 64.40 y: 101.57 s: 39.24 m: 13.301 e: ok
std156.imh x: 67.51 y: 101.49 s: 47.02 m: 16.560 e: ok
std157.imh x: 143.00 y: 220.31 s: 27.96 m: 15.180 e: ok
std158.imh x: 142.84 y: 220.40 s: 19.67 m: 12.307 e: ok
std159.imh x: 140.34 y: 219.43 s: 17.06 m: 12.313 e: ok
```

Figure 13: Running **phot** non-interactively on the standard stars.

```
                                    I R A F
                        Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = phot

image   =                         Input image
coords  =                         Coordinate list (default: image.coo.?)
output  =                default  Results file (default: image.mag.?)"
(datapar=                       ) Data dependent parameters
(centerp=                       ) Centering parameters
(fitskyp=                       ) Sky fitting parameters
(photpar=                       ) Photometry parameters
skyfile =                         Sky 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 mo
(update =                    yes) Update critical parameters in non interactive mo
(verbose=                    yes) Print messages in non interactive mode
(mode   =                     ql)
```

Figure 14: Parameter file for **phot** when stars will be selected interactively.

phot so it resembles that of Figure 14. Note that we have replaced the **coords** coordinate list with the null string (two adjacent double-quotes) and turned "interactive" on.

We need to display the frame we are going to work on in the imtool window:

display std145 1

will display image **std145.imh** in the first frame buffer.

Now let's run **phot**. We are not likely to be *too* accurate with where we place the cursor, so to be generous we will increase the allowable center shift to 3 pixels; otherwise we will get error messages saying that the "shift was too large":

phot std145 maxshift=3.

(Note that even though **maxshift** is a parameter of **centerpars** we can change it on the command line for **phot**.) Also note that we left off the "**.imh**" extension for a reason: we are going to take the default names for the output files, and they will be given names such as **std145.mag.1** and so on. If we had included the **.imh** extension would would now be getting **std145.imh.mag.1** names.

At this point I get a flashing circle in my **imtool** window; I don't know what you get (it depends upon how your defaults are set up) but there should be some sort of obvious marker on top of your image. Put it on the first star you wish to measure and hit the space bar. The coordinates and magnitude should appear in the **gterm** window, and you are ready to measure the next star on this frame. Proceed until all the stars on this frame are measured, and then type a "q" followed by another "q". Display the next frame, and run **phot** on it.

When you get done you will have kerjillions of files.

3.4 Examining the Results: the power of txdump

Depending upon which of the two methods you selected you will either have a single file **standstuff** containing the results of all your aperture photometry, or you will have a file for each frame (**stand145.mag.1**, **stand146.mag.1** ...)containing the stars on each frame. In either event the file will pretty much resemble that shown in Figure 15. The file begins with a large header describing the parameters in force at the time that **phot** was run. There is, however, a real subtlety to this statement. If you had changed a parameter in **datapars**, say, (or any of the other parameters) between running **daofind** and **phot**, the header in **phot** will reflect only the setting that was in force at the time that **phot** was run—in other words, it does not take the values of what was used for the **threshold** from the coordinate file and retain these, but instead simply copies what value of **thresh** happens to be in **datapars** at the time that **phot** is run. To those used to the "self-documenting" feature of VMS DAOPHOT this is a major change!

Once we get past the header information we find that there are 5 lines per star measured. The "key" to these five lines of information are found directly above the measurement of the first star. On the first line we have "general information" such as the image name, the beginning x and y values, the id, and the coordinate file. On the next line we have all the centering information: the computed x and y centers, the x and y shift, and any centering errors. On the third line of the file we have information about the sky. On the fourth line we have some information out of the image header: what was the integration time, what was the airmass, and what was the filter. Note that **phot** has used that integration time in producing the magnitude—the exposures are now normalized to a 1.0 sec exposure. The fifth line gives the actual photometry, including the size of the measuring aperture, the total number of counts within the aperture, the area of the aperture, and the output magnitude, photometric error, and any problems encountered (such as a bad pixel within the aperture).

We can extract particular fields from this file (or files) by using the **txdump** command. For instance, are there any cases where there there were problems in the photometry? We can see those by saying

txdump standstuff image,id,perror

(If you did "Photometry by eye" you can substitute ***mag*** for **standstuff**.) When it queries you for the "boolean expression" type

perror!="No_error"

The "!=" construction is IRAF-ese for "not equal to"; therefore, this will select out anything for which there was some problem in the photometry.

```
#K IRAF       = NOAO/IRAF V2.8Eversion   %-15s
#K USER       = massey         name      %-15s
#K HOST       = tofu           computer  %-15s
#K DATE       = 01-29-90       mm-dd-yr  %-15s
#K TIME       = 10:47:08       hh:mm:ss  %-15s
#K PACKAGE    = apphot         name      %-15s
#K TASK       = phot           name      %-15s
#
#K SCALE      = 1.             units     %-15.7g
#K FWHMPSF    = 3.             scaleunit %-15.7g
#K EMISSION   = yes            switch    %-15b
#K DATAMIN    = -90.           counts    %-15.7g
#K DATAMAX    = 32000.         counts    %-15.7g
#K EXPOSURE   = EXPTIME        keyword   %-15s
#K AIRMASS    = AIRMASS        keyword   %-15s
#K FILTER     = F1POS          keyword   %-15s
#
#K NOISE      = poisson        model     %-15s
#K THRESHOLD  = 500.           counts    %-15.7g
    .
    .
    .
#K SMOOTH     = no             switch    %-15b
#K RGROW      = 0.             scaleunit %-15.7g
#K SKYVALUE   = 0.             counts    %-15.7g
#
#K WEIGHTING  = constant       model     %-15s
#K APERTURES  = 15.            scaleunit %-15s
#K ZMAG       = 25.            zeropoint %-15.7g
#
#N IMAGE               XINIT     YINIT     ID   COORDS              LID        \
#U imagename           pixels    pixels    ##   filename            ##         \
#F %-23s               %-10.2f   %-10.2f   %-5d %-20s               %-5d
#
#N XCENTER  YCENTER  XSHIFT YSHIFT XERR   YERR   CIER CERROR                   \
#U pixels   pixels   pixels pixels pixels pixels ##   cerrors                  \
#F %-12.2f  %-9.2f   %-7.2f %-7.2f %-7.2f %-7.2f %-5d %-13s
#
#N MSKY           STDEV          SSKEW          NSKY   NSREJ SIER SERROR       \
#U counts         counts         counts         npix   npix  ##   serrors     \
#F %-18.7g        %-15.7g        %-15.7g        %-7d   %-6d  %-5d %-13s
#
#N ITIME          XAIRMASS       IFILTER                                       \
#U timeunit       number         name                                          \
#F %-18.7g        %-15.7g        %-23s
#
#N RAPERT   SUM            AREA           MAG    MERR    PIER PERROR          \
#U scale    counts         pixels         mag    mag     ##   perrors         \
#F %-12.2f  %-15.7f        %-15.7f        %-7.3f %-6.3f  %-5d %-13s
#
std145.imh             116.87    262.16    1    std145.imh.coo.1    1         \
   116.65   262.34   -0.22  0.18   0.00   0.00   0    No_error                \
   27.38264       18.17514       -5.188609      703    2     0    No_error     \
   3.025          1.359431       2                                           \
   15.00    110639.3       707.074        13.801 0.008 0    No_error
std146.imh             116.99    265.17    2    std146.imh.coo.1    1         \
   116.93   265.30   -0.06  0.13   0.02   0.01   0    No_error                \
```

Figure 15: Output file from **phot**.

We can create a single file at this point containing just the interesting results from the photometry file(s): do a

txdump standstuff image,id,ifilt,xair,mag,merr yes > standsout

to dump the image name, id-number, filter, airmass, magnitude, and magnitude error into a file **standsout**. (Again, if you did "Photometry by Eye" substitute ***mag*** for **standstuff**). Unfortunately, what you do with this file is up to you right now until the standard reductions routines become available. In the example shown here we have selected the fields in the same order as used in Peter Stetson's VMS CCDCAL software, and at the end of this manual we will describe a (painful) kludge that nevertheless *will* let you use these numbers with that software.

4 Crowded Field Photometry: IRAF/daophot

4.1 Historical Summary

In the beginning (roughly 1979) astronomers interested in obtaining photometry from stars in "relatively" crowded fields would make the journey to Tucson in order to use Doug Tody's RICHFLD program which ran on the IPPS display system. RICHFLD allowed the user to define a point-spread-function (PSF), and then fit this PSF to the brightest star in a group, subtract off this star, and then proceed to the next brightest star, etc. This represented a giant qualitative improvement over the possibilities of aperture photometry, and allowed stars separated by a few FWHM's to be accurately measured.

Beginning in 1983, a group of RICHFLD users at the DAO (including Ed Olszewski and Linda Stryker) began modifications to the "poorman" program of Jeremy Mould. This was largely motivated by the implementation of the "Kitt Peak CCD" at the prime-focus of the Tololo 4-m, and the idea was to design a crowded-field photometry program that (a) allowed simultaneous PSF-fitting, (b) made use of the *known noise characteristics of a CCD* to do the fitting in a statistically correct manner (i.e., to make "optimal" use of the data), and (c) to be largely batch oriented. In mid-1983 Peter Stetson arrived at the DAO, and took over the effort. The result was DAOPHOT, which did all these things and more. By 1986 DAOPHOT was well distributed within the astronomical community. The basic algorithms and philosophy can be found in Stetson 1987 (PASP **99**, 111).

DAOPHOT (and its companion program ALLSTAR) were not part of a photometry package; they were instead stand-alone Fortran programs which did not deal in any way with the issue of image display or what to do with the instrumental magnitudes once you had them. They were also only supported on VMS, although several "frozen" versions were translated into UNIX by interested parties around the country. There was therefore much to be gained from integrating the algorithms of daophot with IRAF in order to make use of the image display capabilities and general tools for manipulating images. Also, since many astronomers were now reducing their CCD data with IRAF, it avoided the necessity of translating the IRAF files into the special format needed by VMS DAOPHOT. Dennis Crabtree began this translation program while at the DAO; it was taken over by Lindsey Davis of the IRAF group

in early 1989, and taken to completion in early 1990. Pedro Gigoux of CTIO is currently hard at work on the photometry reduction package, scheduled for completion sometime during the spring.

4.2 daophot Overview

The steps involved in running daophot are certainly more involved than in simple aperture photometry, but they are relatively straightforward. The following sections will lead you through the necessary procedures. Alternative routes will be noted at some points, and more may be gleaned from reading the various "help" pages. A general outline is given here so that you have some overview in mind; a detailed step-by-step summary is provided at the end of this section.

- Before you reduce the first frame, **imexamine** your data to determine FWHM's and the radius at which the brightest star you wish to reduce blends into the sky. Run **imhead** to find the "key-words" in your data headers for exposure times, filter number, and airmass. Enter these, along with the characteristics of your chip (read-noise, photons per ADU, maximum good data value) into the parameter sets **datapars** and **daopars**.
- Use **daofind** and **tvmark** to produce a list of x and y positions of most stars on the frame.
- Use **phot** to perform aperture photometry on the identified stars. This photometry will be the basis of the zero-point of your frame via the PSF stars. This is also the only point where sky values are determined for your stars.
- Use **psf** to define the PSF for your frame. If your PSF stars are crowded this will require some iteration using the routines **nstar** and **substar**.
- Use **allstar** to do simultaneous PSF-fitting for all the stars found on your frame, and to produce a subtracted frame.
- Use **daofind** on the subtracted frame to identify stars that had been previously hidden.
- Run **phot** *on the original frame* to obtain aperture photometry and sky values for the stars on the new list.
- Use **append** to merge the two aperture photometry lists.
- Run **allstar** again on the merged list.

When you have done this for your *U, B,* and *V* frames it is then time to

- Use **txdump**, **tvmark**, and the image display capabilities to come up with a consistent matching between the frames. If there are additions or deletions then you will need to re-run **phot** and **allstar** one more time.

Finally you will need to

- Determine the aperture correction for each frame by subtracting all but the brightest few isolated stars on your frames and then running **phot** to determine the light lost between your zero-point aperture and the large aperture you used on your standard stars.

4.3 How Big Is A Star: A Few Useful Definitions

The parameter files **datapars** and **daopars** contain three "size-like" variables, and although this document is not intended as a reference guide, there is bound to be confusion over these three parameters, particularly among those new to DAOPHOT. In the hopes of un-muddying the waters, we present the following.

fwhmpsf This is the full-width at half-maximum of a stellar object (point-spread function, or psf). The value for **fwhmpsf** gets used only by the automatic star-finding algorithm **daophot**, unless you do something very bad like setting **scale** to non-unity.

psfrad This is the "radius" of the PSF. When you construct a PSF, the PSF will consist of an array that is

$$(2 \times psfrad + 1) \times (2 \times psfrad + 1)$$

on a side. The idea here is that "nearly all" of the light of the brightest star you care about will be contained within this box. If you were to construct a PSF with some large value of **psfrad** and then run **nstar** or **allstar** specifying a smaller value of **psfrad**, the smaller value would be used. Making the **psfrad** big enough is necessary to insure that the wings of some nearby bright star are properly accounted for when fitting a faint star.

fitrad This is how much of the psf is used in making the fit to a star. The "best" photometry will be obtained (under most circumstances) if this radius is set to something like the value for the fwhm.

4.4 Setting up the parameter files "daopars" and "datapars"

The first step in using IRAF/daophot is to determine and store the characteristics of your data in two parameter files called "datapars" and "daopars"; these will be used by the various daophot commands. In Section 1 we discussed how to deal with parameter files, and in Section 2 we went through setting up "datapars" for the standard star solutions; at the risk of repeating ourselves, we will go through this again as the emphasis is now a little different.

First inspect your headers by doing an **imhead** imagename **long+ | page**. This will produce a listing similar to that shown in Figure 16. The things to note here are (a) what the filter keyword is (we can see from Figure 16 that the answer is F1POS; while there is an F2POS also listed, the second filter bolt was not used and was always in position "zero"), (b)

```
n602alu.imh[316,500][short]: u h68
    No bad pixels, no histogram, min=-11., max=17421.
    Line storage mode, physdim [512,500], length of user area 770 s.u.
    Created Sun 14:35:46 12-Nov-89, Last modified Sun 14:35:47 12-Nov-89
    Pixel file 'tofu!/data/massey/pixels/n602alu.pix' [ok]
    'KPNO-MTN'            / institution of origin
    DATA-TYP= '    object (  0 )'   / object,dark,comp,etc.
    ITIME   =                  700  / integration time secs
    UT      = ' 1:54:12.00'  /  universal time
    ZD      = ' 0:38:46.00'  /  zenith distance
    DATE-OBS= '28/11/85'            / dd/mm/yy observation
    ST      = ' 1:38:46.00'  /  sidereal time
    RA      = ' 1:28:52.00'  /  right ascension
    DEC     = '-73:40:50.00'  /  declination
    EPOCH   =                1975.  / epoch of RA and DEC
    CAM-TEMP=                  .00  / camera temperature, deg C
    DEW-TEMP=                  .00  / dewar temp, deg C
    HISTORY1= 'bt=   120 bp=     0 cr=     0 dk=     0 '
    HISTORY2= 'ff=   903 fg=     0 sc=   .000  bi=  900  '
    COMMENT = 'u h68'
    F1POS   =                    5  / filter bolt I position
    F2POS   =                    0  / filter bolt II position
    EXPTIME =              700.025
    AIRMASS =             1.377947
    UTMIDDLE= '2:00:02.0'
```

Figure 16: Header for image n602alu.imh.

what the effective exposure time keyword is (EXPTIME in this example), and (c) what the effective airmass keyword is (AIRMASS in this example).

Next you need to examine some "typical" frames in order to determine the FWHM (**fwhmpsf**) and the radius of the brightest star for which you plan to do photometry (**psfrad**). First **display** an image, and use the middle button of the mouse (or whatever you need to do on your image display) to zoom on a few bright stars. On the SUN the "F6" key will let you see x and y values. The "default" PSF radius is 11 pixels: are your stars bigger than 23 pixels($23 = 2 \times 11 + 1$) pixels from one side to the other? The FWHM is undoubtably variable from frame to frame, but unless these change by drastic amounts (factors of two, say) using a "typical" value will doubtless suffice. You can use the **imexamine** routine to get some idea of the FWHM; do **imexamine** filename and then strike the "r" key (for radial profile) after centering the cursor on a bright (but unsaturated) star. The last number on the plot is the FWHM of the best-fit Gaussian.

We are now ready to do an **epar datapars**. This parameter file contains information which is data-specific. We set **fwhmpsf** to the FWHM determined above, and we enter the names of the keywords determined from the header inspection above. The "gain" and "readnoise" are values you have either determined at the telescope (using the Tololo routines) or which are carved in stone for your chip. Choosing the value for datamax, the "Maximum good data value", (in ADU's, NOT electrons) is a little bit trickier. In the case of aperture photometry we were satisfied to take the nominal value for the chip, but point-spread-function fitting is a bit more demanding in what's "linear". The data obtained here was taken with an RCA chip, and we all know that RCA chips are linear well past 100,000 e-. Thus, naively, we would expect that with a gain of 2.25 that the chip was still linear when we hit the digitization limit of 32,767 ADU's. Subtract off 500 for the likely bias, and we *might* think that we were safe up to 32,200. However, we would be wrong. Experience with PSF fitting on these data shows that something (presumably in those little silver VEB's) has resulted in these data being non-linear above 20,000 ADU's. My suggestion here is to start with the nominal value but be prepared to lower it if the residuals from PSF fitting appear to be magnitude dependent (more on this later). The value for **datamin**, the "Minimum good data value", will be different for each frame (depending what the sky level is) and there is not much point in entering a value for that yet. Similarly the value we will use for threshold will change from frame to frame depending upon what the sky level is. When you are done your **datapars** should resemble that of Figure 17.

Next we will **epar daopars**. This parameter file contains information specific to what you want **daophot** to do. The only things here we might want to change at this point are the "Radius of the psf" **psfrad** (if your experiment above showed it should be increased somewhat), and you might want to change the fitting radius **fitrad**. Leaving the fitting radius to "something like" the FWHM results in the best SNR (you can work this out for yourself for a few different regimes if you like to do integrals). The "standard values" are shown in Figure 18. .

```
                                    I R A F
                      Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = datapars

(fwhmpsf=                  3.) 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=                 65.) CCD readout noise in electrons
(gain   =                    ) CCD gain image header keyword
(epadu  =                2.25) Gain in electrons per count
(exposur=             EXPTIME) Exposure time image header keyword
(itime  =              INDEF) Exposure time
(airmass=             AIRMASS) Airmass image header keyword
(xairmas=              INDEF) Airmass
(filter =              F1POS) Filter image header keyword
(ifilter=              INDEF) Filter
(datamin=              INDEF) Minimum good data value
(datamax=              20000.) Maximum good data value
(mode   =                 ql)
```

Figure 17: A sample **datapars** is shown.

```
                                    I R A F
                        Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = daopars

(varpsf =                    no) Variable psf across image?
(psfrad =                  11.) Radius of the psf
(fitrad =                   2.5) Fitting radius in pixels
(matchra=                   1.5) Search radius for match with the PHOT results
(critove=                   0.2) Critical overlap group membership
(maxiter=                    50) Maximum number of iterations
(maxgrou=                    60) Maximum number of stars per group
(maxnsta=                  3000) Maximum number of stars to fit
(recente=                   yes) Recenter stars during fit (allstar only)
(clipran=                   2.5) Clipping range in standard deviations (allstar
(clipexp=                     6) Clipping exponent (allstar only)
(text   =                   yes) Text file on output?
(mode   =                    ql)
```

Figure 18: A sample **daopars** is shown.

4.5 Finding stars: daofind and tvmark

The automatic star finder **daofind** convolves a Gaussian of width FWHM with the image, and looks for peaks greater than some threshold in the smoothed image. It then keeps only the ones that are within certain roundness and sharpness criteria in order to reject non-stellar objects (cosmic rays, background galaxies, bad columns, fingerprints). We have already entered a reasonable value for the FWHM into **datapars**, but what should we use as a threshold? We expect some random fluctuations due to the photon statistics of the sky and to the read-noise of the chip. You can calculate this easily by first measuring the sky value on your frame by using **imexamine** and the "h" key to produce a histogram of the data (**implot** and the "s" key is another way). In the example shown in Figure 19 we see that the sky value is roughly 150. In general, if s is the sky value in ADU, p is the number of photons per ADU, and r is the read-noise in units of electrons, then the expected 1σ variance in the sky will be

$$\left(\sqrt{s \times p + r^2}\right)/p$$

in units of ADU's. For the example here we expect $1\sigma = \left(\sqrt{150. \times 2.25 + 65^2}\right)/2.25 = 30$ ADU's. Of course, if you have averaged N frames in producing your image, then you should be using $N \times p$ as the gain both here and in the value entered in **datapars**; similarly the readnoise is really just $r \times \sqrt{N}$. If instead you summed N frames then the gain is just p and

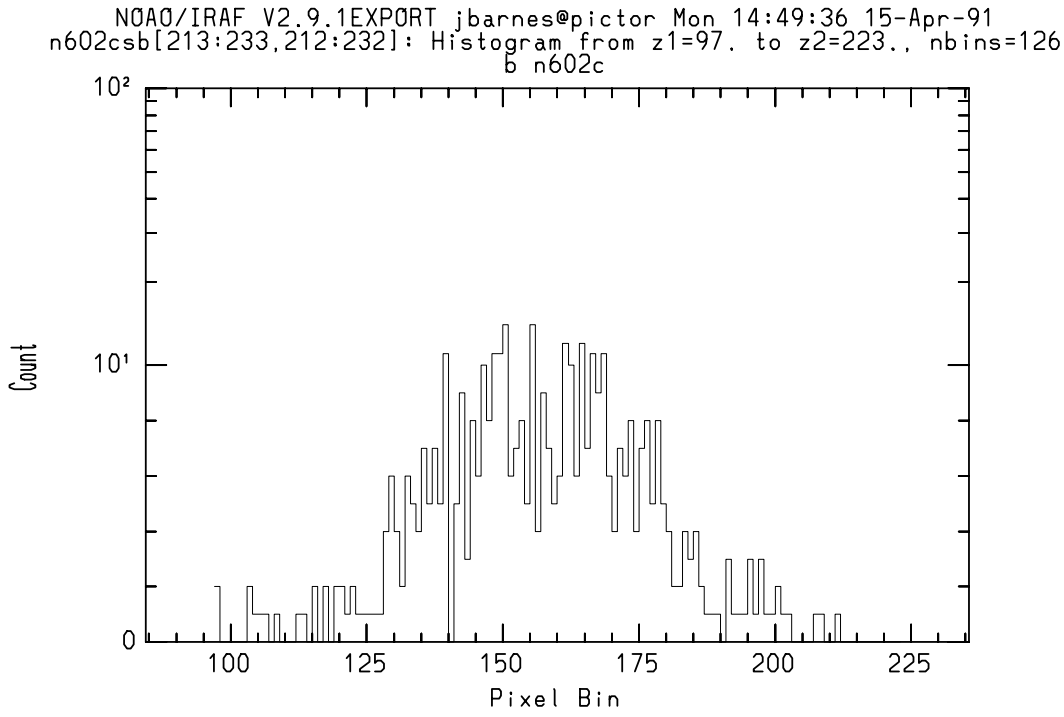

Figure 19: The **imexamine** histogram ("h" key) indicates that the sky value is roughly 150.

```
                                  I R A F
                       Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = datapars

(fwhmpsf=                   3.) FWHM of the PSF in scale units
(emissio=                  yes) Features are positive?
(noise  =              poisson) Noise model
(thresho=                 105.) 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=                  65.) CCD readout noise in electrons
(gain   =                     ) CCD gain image header keyword
(epadu  =                 2.25) Gain in electrons per count
(exposur=              EXPTIME) Exposure time image header keyword
(itime  =                INDEF) Exposure time
(airmass=              AIRMASS) Airmass image header keyword
(xairmas=                INDEF) Airmass
(filter =                F1POS) Filter image header keyword
(ifilter=                INDEF) Filter
(datamin=                  60.) Minimum good data value
(datamax=                20000.) Maximum good data value
(mode   =                   ql)
```

Figure 20: Datapars with **threshold** and **datamin** entered.

the readnoise is still $r \times \sqrt{N}$.

In the example shown here the expected 1σ variation of the sky is 30 ADU's; we might therefore want to set our star detection threshold to 3.5 times that amount. That won't guarantee that every last star we find is real, nor will it find every last real star, but it should do pretty close to that!

We should use this opportunity to set datamin in **datapars** to some value like $s - 3\sigma$. In this case we will set it to 60. This is not currently used by **daofind** but will be used by all the photometry routines. Figure 20 shows the data parameters with the appropriate values of threshold and datamin now entered.

We now can **epar daofind** so it resembles that of Figure 21. Note that although nothing appears to be listed under **datapars** the default name is "datapars"; you could instead have created a separate data parameter file for each "type" of data you have and have called them separate names (you could do this by doing an **epar datapars** and then exiting with a ":w

```
da> lpar daofind
        image = "n602csb"       Input image
       output = "default"       Results file (default: image.coo.?)
 (convolution = "")             Output convolved image
    (datapars = "")             Data dependent parameters (fwhmpsf, threshold,
       (ratio = 1.)             Ratio of sigmay to sigmax of Gaussian kernel
       (theta = 0.)             Position angle of major axis of Gaussian kernel
      (nsigma = 1.5)            Width of convolution kernel in sigma
     (sharplo = 0.2)            Lower bound on sharpness for feature detection
     (sharphi = 1.)             Upper bound on sharpness for  feature detection
     (roundlo = -1.)            Lower bound on roundness for feature detection
     (roundhi = 1.)             Upper bound on roundness for feature detection
    (boundary = "nearest")      Boundary extension (constant, nearest, reflect,
    (constant = 0.)             Constant for boundary extension
    (graphics = "stdgraph")     Graphics device
     (display = "stdimage")     Display device
    (commands = "")             Image cursor: [x y wcs] key [cmd]
      (cursor = "")             Graphics cursor: [x y wcs] key [cmd]
(mkdetections = no)             Mark detected stars on the display
 (interactive = no)             Interactive mode
      (verify = yes)            Verify critical parameters in non interactive m
      (update = no)             Update critical parameters in non interactive m
     (verbose = no)            Print messages in non interactive mode
        (mode = "ql")
pr> daofind
Input image (n602csb):

FWHM of features in scale units (3.):
        New FWHM of features: 3. scale units  3. pixels
Detection threshold in counts above background (105.):
        New detection threshold: 105. counts
```

Figure 21: Parameters for **daofind**.

newnamepar"). This might be handy if all your U frames were averages, say, but your B and V frames were single exposures; that way you could keep track of the separate effective gain and readnoise values. In that case you would enter the appropriate data parameter name under **datapars**. As explained earlier, you could also do a ":e" on the **datapars** line and essentially do the **epar datapars** from within the **epar daofind**. For normal star images, the various numerical values listed are best kept exactly the way they are; if you have only football shaped images, then read the help page for **daofind** for hints how best to find footballs.

We can now run **daofind** by simply typing **daofind**. As shown in Figure 21 that we were asked for the FWHM and threshold values; this is a due to having turned "verify" on in the parameter set. This safeguards to a large extent over having forgotten to set something correctly. A [CR] simply takes the default value listed.

Running **daofind** produced an output file with the (default) filename of **n602csb.coo.1**. (Do *not* give the **.imh** extension when specifying the image name, or the default naming process will get very confused!) We can page through that and see the x and y centers, the number of magnitudes brighter than the cutoff, the sharpness and roundness values, and the star number. However, of more immediate use is to use this file to mark the found stars on the image display and see how we did. If we have already displayed the frame in frame 1, then we can **epar tvmark** to make it resemble Figure 22. This will put red dots on top of each star found.

We can see from Figure 23 that **daofind** did a pretty nice job. If we didn't like what we saw at this point we could rerun **daofind** with a slightly higher or slightly lower threshold—try varying the threshold by half a sigma or so if you are almost right. As you may have guessed, subsequent runs will produce output files with the names n602csb.coo.2, n602csb.coo.3,... If you are using a very slow computer, or are exceedingly impatient, you could have saved some time by putting a "c" (say) under "convolv" in your first run of **daofind**—this would have saved the smoothed image as cn602csb.imh, and would drastically reduce the number of cpu cycles needed to rerun **daofind** with a different threshold value. If you really very happy with what **daofind** did but you just want to add one or two stars at this point, you can in fact do that quite readily using **tvmark**. Set the parameters as in Figure 22, but turn interactive on. Position the cursor on top of the star you wish to add and strike the "a" key. Note that this will "disturb" the format of the file, but we really don't care; it will still work just fine as the input to **phot**.

Note that it is fairly important that you do a good job at this stage. If you have used too low a threshold, and have a lot of junk marked as stars, these fictitious objects are likely to wander around during the PSF-fittings until they find something to latch onto—*not* a good idea. However, you also do not want the threshold to be so high that you are missing faint stars. Even if you are not planning to publish photometry of these faint guys, you need to have included them in the list of objects if they are near enough to affect the photometry of stars for which you do have some interest. If you find that varying the threshold level does not result in a good list, then something is wrong—probably you have badly over- or under-estimated the FWHM. When you are close to the "perfect" value of the threshold, changing its value by as little as half a sigma will make a substantial difference between getting junk

```
pr> lpar tvmark
        frame = 1                Default frame number for display
       coords = "n602csb.coo.1" Input coordinate list
     (logfile = "")              Output log file
     (autolog = no)              Automatically log each marking command
    (outimage = "")              Output snapped image
   (deletions = "")              Output coordinate deletions list
    (commands = "")              Image cursor: [x y wcs] key [cmd]
        (mark = "point")         The mark type
       (radii = "0")             Radii in image pixels of concentric circles
     (lengths = "0")             Lengths and width in image pixels of concentric
        (font = "raster")        Default font
       (color = 204)             Gray level of marks to be drawn
       (label = no)              Label the marked coordinates
      (number = no)              Number the marked coordinates
    (nxoffset = 0)               X offset in display pixels of number
    (nyoffset = 0)               Y offset in display pixels of number
   (pointsize = 1)               Size of dot in display pixels
      (txsize = 1)               Size of text and numbers in font units
   (tolerance = 1.5)             Tolerance for deleting coordinates in image pix
 (interactive = no)              Mode of use
        (mode = "ql")
```

Figure 22: Parameter file for **tvmark**.

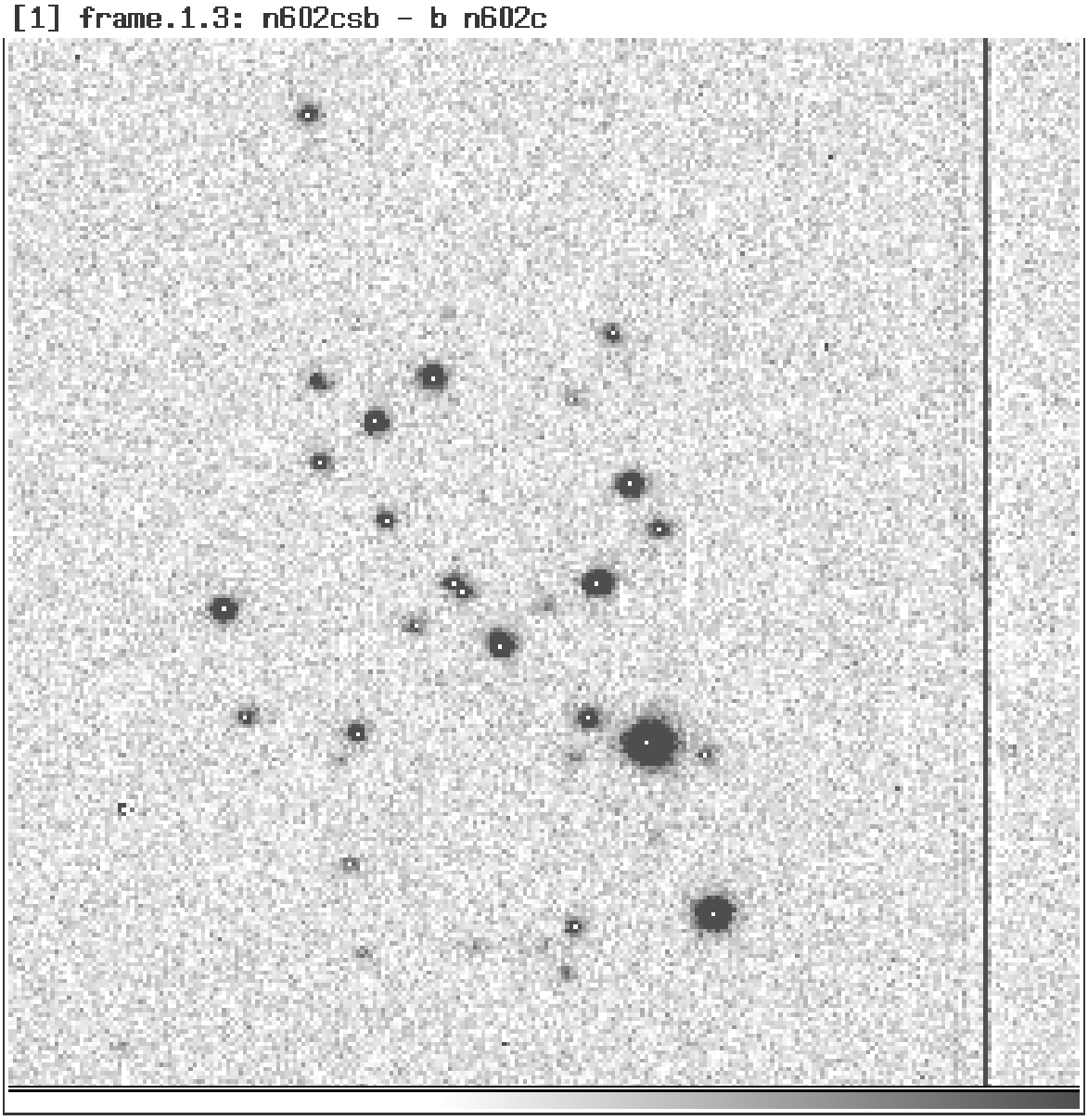

Figure 23: Stars found with **daofind** and marked with **tvmark**.

and real stars.

4.6 Aperture Photometry with phot

The next step is to do simple aperture photometry for each of the stars that have been found. These values will be used as starting points in doing the PSF fitting, and this is the only time that sky values will be determined.

One of the few ways of "crash landing" in the current implementation of the software is to forget to reset "datamin" in the datapars file before running phot on a new frame. It is the only critical parameter which is not queried when verify is turned on. Therefore, this is a good time to check to see that "datamin" is really set to several sigma lower than the sky value of this particular frame.

The aperture photometry routine **phot** has more parameters than all the others put together: there are the parameter files **centerpars**, **fitskypars**, and **photpars**. Fortunately the "verify" option frees you from having to look at these, and helps prevent you from making a mistake. If this is your first pass through DAOPHOT it is worth your while to do the following:

unlearn centerpars
unlearn fitskypars
unlearn photpars

If you have used **phot** for measuring standard stars, then this will reset the defaults to reasonable values for crowded-field photometry; in particular, we want to make sure that the centering algorithm in **centerpars** is set to "none". Do an **epar phot** and make it look like that of Figure 24. Since we have the "verify" switch turned on, we can be happy, not worry, and simply type **phot**. **phot** will then prompt you as shown in Figure 24. Note that the answers were particularly simple: we told it the name of the frame we wished to work with, we accepted the default for the coordinate list (it will take the highest "version" of image.coo.NUMBER) and the default for the output photometry list (n602csb.mag.1 will be produced in this case.) We accepted the centers from **daofind** as being "good enough" to not have to recenter (they are good to about one-third of a pixel, plenty good enough for aperture sizes of 2.5 pixels and bigger; when we run this routine later on the second pass we would make a Big Mistake by turning centering on here, so leave it off). The sky values will be taken from an annulus extending from a radius of 10 pixels to a radius of 20 pixels, and it will determine the standard deviation of the sky from the actual data. Note that this is probably a lot closer in than you used on your standard stars; in crowded regions of variable background keeping this annulus relatively close in will help. Finally, we used a measuring aperture of 3 pixels. The number of counts within this aperture will be what defines the zero-point of your frame, as we will see in Section 4.9, and keeping this value *fixed* to some value like your typical FWHM will keep you safe.

```
                                  I R A F
                     Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = phot

image   =               n602csb  Input image
coords  =               default  Coordinate list (default: image.coo.?)
output  =               default  Results file (default: image.mag.?)"
(datapar=                      ) Data dependent parameters
(centerp=                      ) Centering parameters
(fitskyp=                      ) Sky fitting parameters
(photpar=                      ) Photometry parameters
skyfile =                        Sky 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=                     no) Mode of use
(verify =                    yes) Verify critical parameters in non interactive mo
(update =                    yes) Update critical parameters in non interactive mo
(verbose=                    yes) Print messages in non interactive mode
(mode   =                     ql)

da> phot
Input image (n602csb):
Coordinate list (default: image.coo.?) (default):
Results file (default: image.mag.?)" (default):

Centering algorithm (none):
        New centering algorithm: none
Sky fitting algorithm (mode):
        Sky fitting algorithm: mode
Inner radius of sky annulus in scale units (10.):
        New inner radius of sky annulus: 10. scale units 10. pixels
Width of the sky annulus in scale units (10.):
        New width of the sky annulus: 10. scale units 10. pixels
Standard deviation of background in counts (INDEF):
        New standard deviation of background: INDEF counts
File/list of aperture radii in scale units (3.):
        Aperture radius 1: 3. scale units 3. pixels
```

Figure 24: Questions and answers with **phot**.

4.7 Making the PSF with psf

If you are used to the VMS version of DAOPHOT, you are in for a pleasant surprise when it comes to making a PSF within the IRAF version. Nevertheless, just because it's easy doesn't mean that you shouldn't be careful.

What constitutes a good PSF star? Stetson recommends that a good PSF star meets the following criteria:

1. No other star at all contributes any light within one fitting radius of the center of the candidate star. (The fitting radius will be something like the FWHM.)

2. Such stars as lie near the candidate star are significantly fainter. ("Near" being defined as, say, 1.5 times the radius of the brightest star you are going to measure.)

3. There are no bad columns or rows near the candidate star; there should also be no bad pixels near the candidate star.

In making a PSF, you wish to construct a PSF which is free from bumps and wiggles (unless those bumps and wiggles are really what a single isolated star would look like.) First off, does it matter if we get the PSF "right"? If we had only isolated stars, then the answer would be no—any old approximation to the PSF would give you good relative magnitudes, and there are programs in the literature which do exactly this. However, if your stars are relatively isolated you are not going to gain anything by PSF-fitting over aperture photometry anyway, so why bother? If you are dealing with crowded images, then the PSF has to be right *even in the wings*, and for that reason we construct a PSF empirically using the brightest and least crowded stars in our frame. If you are very, very lucky you will find that your brightest, unsaturated star is well isolated, and has no neighbors about it—if that's the case, use that one and forget about the rest. Usually, however, you will find that it isn't quite that easy, and it will be necessary to construct the PSF interatively. The steps involved will be

1. Select the brightest, least-crowded stars for the zeroth-order PSF.

2. Decrease the size of the PSF radius and fit these stars with their neighbors using **nstar**.

3. Subtract off the PSF stars and their neighbors using **substar** to see if any of the PSF stars are "funny"; if so, go back to the step 1 and start over.

4. Edit the **nstar** results file (**imagename.nst.N**) and delete the entries for the PSF stars. You are left with a file containing the magnitudes and positions of just the neighbors.

5. Subtract off just the neighbors using this file as input to **substar**. Display the results, and examine the region around each PSF star. Are the neighbors cleanly removed?

6. Increase the PSF radius back to the original value. Construct an improved PSF using the new frame (the one with the neighbors gone.)

```
PACKAGE = daophot
   TASK = psf

image   =              n602csb  Image for which to build PSF
photfile=              default  Aperture photometry file (default: image.mag.?)
psfimage=              default  Output PSF image (default: image.psf.?)
groupfil=              default  Output PSF group file (default: image.psg.?)
(datapar=                     ) Data dependent parameters
(daopars=                     ) Daophot fitting parameters
(showplo=                  yes) Show plots of PSF stars and fit residuals?
(plottyp=                 mesh) Default plot type (mesh|contour)
(plotfil=                     ) Name of output plot metacode file
(graphic=              stdgraph) Graphics device
(display=              stdimage) Display device
(command=                     ) Image cursor: [x y wcs] key [cmd]
(cursor =                     ) Graphics cursor: [x y wcs] key [cmd]
(verify =                  yes) Verify critical PSF parameters
(mode   =                   ql)
```

Figure 25: Parameter file for **psf**

7. Run **nstar** on the PSF stars and their neighbors again, and again subtract these using **substar**. Examine the results. If you are happy, proceed; otherwise, if the neighbors need to be removed a bit more cleanly go back to step 4.

First **display** the frame, and put dots on all the stars you've found using **tvmark** as discussed above. Next **epar psf** and make sure it looks like that of Figure 25. We have set this up so we can choose the stars interactively from the display window.

Next run **psf**. The defaults that you will be asked to **verify** are probably fine, but pay particular attention to **psf radius** and **fitting radius**. The **psf radius** should be as large as you determined above (11 usually works well on "typical" CCD frames whose star images have FWHM's ≈ 3). The "fitting radius" should be relatively generous here—maybe even larger than what you want to use on your program stars. A reasonable choice is approximately that of the FWHM.

You will find that the cursor has turned into a circle and is sitting on your image in the display window. Position it on a likely looking PSF star, and strike the "a" key. You will be confronted with a mesh plot that shows the star and it surroundings. To find out more about the star (such as what the peak data value is you can type an "s" while looking at the mesh plot. To reject the star type an "x", to accept the star type an "o". In the latter case, you will next see a mesh plot that shows you the star with a two-dimensional Gaussian fit removed from the star. Again, exit this with a "o". If you don't find these mesh plots particularly

useful, you can avoid them by setting **showplot=no** in the **psf** parameters (see Figure 25). At this point you will be told what the star number was, what the magnitude was, and what the minimum and maximum data values within the PSF were. (If you picked a star whose peak intensity was greater than "datamax" it will tell you this and not let you use this star.) When you are done selecting stars, type a "w" (to write the PSF to disk) followed by a "q".

If in making the PSF you noticed that there were stars you could have used but didn't because they had faint neighbors not found in the earlier step of star finding, you can add these by hand by simply running **tvmark** interactively and marking the extra stars. First **epar tvmark** so it resembles that of Figure 22. Then:

display n602csb 1
tvmark 1 n602csb.coo.1 interactive+

Striking the "l" key will mark the stars it already knows about onto the display (as red dots this time around); positioning the cursor on the first star you wish to add and type an "a". When you are done adding stars exit with a "q" and re-run **phot**.

Now that you have made your preliminary PSF, do a **directory**. You'll notice that in addition to the image **n602csb.psf.1.imh** that the **psf** routine has also added a text file **n602csb.psg.1**. If you **page** this file you will see something like that of Figure 26. This contains the aperture photometry of each PSF star plus its neighbors, with each set constituting a "group". Running the psf-fitting photometry routine **nstar** will fit PSF's to each of the stars within a group simultaneously.

Before we run **nstar**, however, we must decide what psf radius to use. Why not simply keep it set to the value found above (e.g., something like 11 pixels)? The answer to this is a bit subtle, but understanding it will help you diagnose what is going wrong when you find a PSF going awry (and don't worry, you will). Let's consider the case that you construct a PSF from a single star with one neighbor whose center is 12 pixels away from the center of the PSF star, and let's have the PSF radius be 11 and the PSF fitting radius be 3. The PSF looks something like that of Figure 27. The light from the neighbor star "spills over" into the PSF.

What happens when you try to fit two PSF's simultaneously? The bump from the PSF of the brighter star sits within the fitting radius of the fainter star, and it is the sum of the PSF's which are being fit to each star (that's what "simultaneous" means). Thus there is an "implicit subtraction" of the fainter star simply from fitting the bumpy PSF to the brighter star, and the brightness of the fainter star will be underestimated. The way to avoid this is to see that the PSF of the brighter star does not come within the fitting radius of the fainter star, and *that* we can accomplish easily by truncating the PSF size to something like the separation of the two stars minus the fitting radius. Thus in the example here we would want to fit the two stars using PSF's that were only $(12 - 3 = 9)$ pixels in radius. It's true that there may still be light of the PSF star beyond this radius, but that will matter only if the PSF star is still going strong when you get within the *fitting radius* of the fainter star.

Now that we understand all that, run **nstar**. Specify the appropriate image name for "image corresponding to photometry" and give it the ".psg" file **n602csb.psg.1** for the "input group file". Remember to decrease the **psf radius** when it tries to verify that number. **nstar**

```
#K IRAF       = NOAO/IRAF V2.8Eversion   %-15s
#K USER       = massey         name      %-15s
#K HOST       = tofu           computer  %-15s
#K DATE       = 12-02-89       mm-dd-yr  %-15s
#K TIME       = 15:05:40       hh:mm:ss  %-15s
#K PACKAGE    = daophot        name      %-15s
#K TASK       = psf            name      %-15s
#K IMAGE      = n602csb        imagename %-15s
#K APFILE     = n602csb.mag.1  filename  %-15s
#K PSFIMAGE   = n602csb.psf.1  imagename %-15s
#K GRPSFILE   = n602csb.psg.1  filename  %-15s
#K LOWBAD     = 50.            counts    %-15.7g
#K HIGHBAD    = 20000.         counts    %-15.7g
#K PSFRAD     = 11.            pixels    %-15.7g
#K FITRAD     = 2.5            pixels    %-15.7g
#K PSFMAG     = 13.874         magnitude %-15.7g
#
#N ID    GROUP XCENTER   YCENTER   MAG         MSKY                          \
#U ##    ##    pixels    pixels    magnitudes  counts                        \
#F %-9d  %-6d  %-10.2f   %-10.2f   %-12.3f     %-14.3f
#
8        1     184.20    198.94    13.874      162.252
7        1     198.14    196.06    18.311      158.061
11       1     170.11    204.81    17.160      158.099
5        2     199.97    158.04    15.031      158.074
```

Figure 26: The "point spread function group" file **n602csb.psg.1**

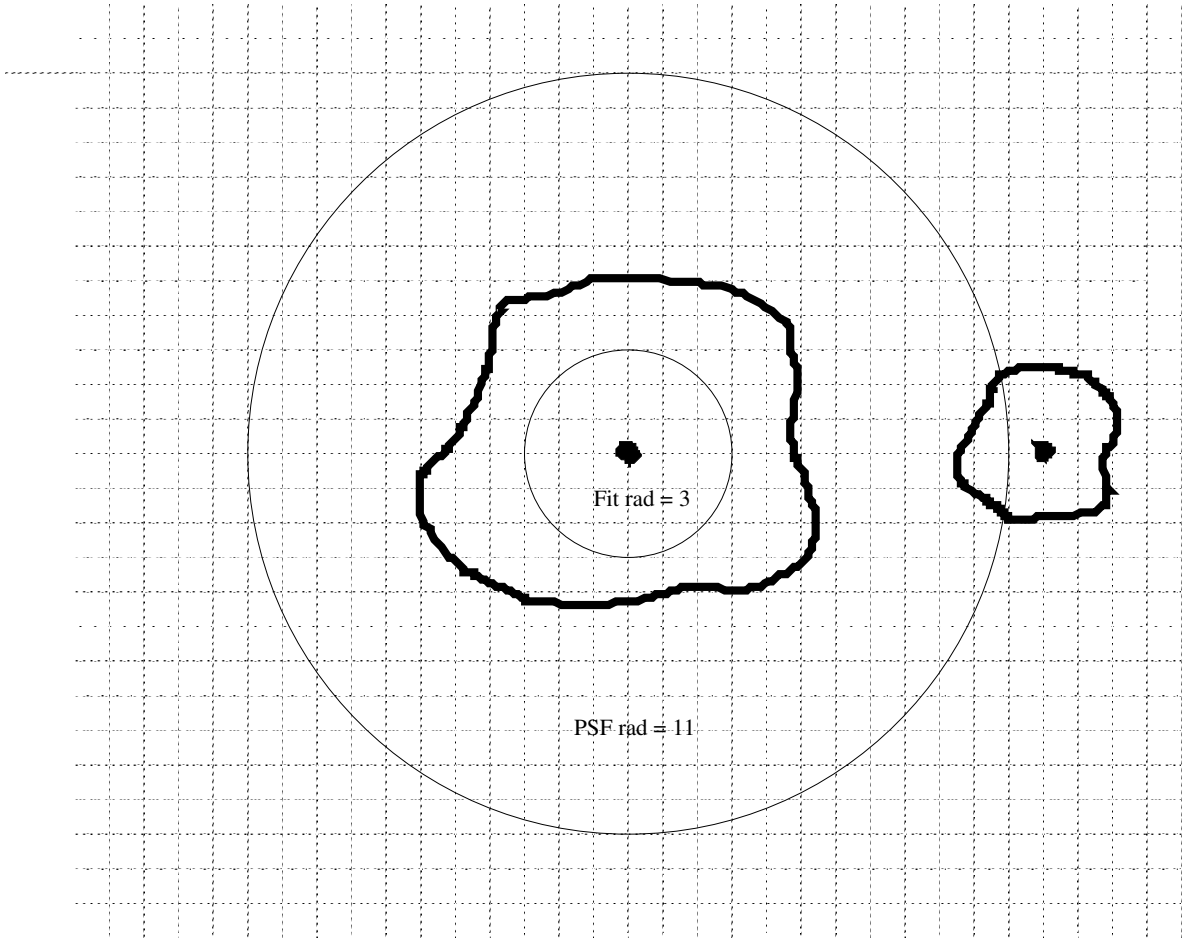

Figure 27: The zeroth order PSF of a star with a neighbor 12 pixels away.

will produce a photometry output file **n60csb.nst.1**. You can subtract the fitted PSF's from these stars now by running **substar**. Again, **verify** the PSF radius to the smaller value. When the routine finishes, **display** the resultant frame **n60csb.sub.1.imh** and take a look at the PSF stars...or rather, where the PSF stars (and their neighbors) were. Are they subtracted cleanly? Does one of the PSF stars have residuals that look the reverse of the residuals of the others? If so, it would be best to reconstruct the PSF at this point throwing out that star—possibly it has a neighbor hidden underneath it, or has something else wrong with it. Are the variations in the cores of the subtracted image consistent with photon statistics? To answer this you may want to play around with **imexamine** on both the original and subtracted images, but if the stars have cleanly disappeared and you can't even tell where they were, you are doing fine.

The worst thing to find at this point is that there is a systematic pattern with position on the chip. This would indicate that the PSF is variable. There is the option for making a variable PSF, but the assumption is that the PSF varies smoothly in x and y; usually this is not the case. (In the case of the non-flat TI chips the variations are due to the potato-chip like shape.) If you *do* decide the PSF is variable, be sure to use plenty of stars in making the PSF. As it says in the "help page", twenty-five to thirty is then not an unreasonable number. If that doesn't scare you off, nothing will.

If the brightest stars have residuals that are systematically different than those of the fainter stars, maybe that chip wasn't quite as linear as you thought, or perhaps there are charge transfer problems. This proved to be the case for the RCA CCD data being reduced here. In Figure 28 we show the residuals that result when we based our PSF on a star whose peak counts were 30000 ADUs. Empirically we found that stars with peaks of 18K ADUs (a mere 40K electrons) were safe to use, with the result that the dynamic range of our data was simply not quite as advertised. Although the PSF function broke down above 18K, the chip remained "linear" in the sense that aperture photometry continued to give good results—the total number of counts continued to scale right up to the A/D limit of 32,767 ADUs (72K electrons after bias is allowed for). This appears to be a subtle charge transfer problem.

We will assume that you have gotten the PSF to the point where the cores of the stars disappear cleanly, although there may be residuals present due to the neighbors. Our next step is to get rid of these neighbors so that you can make a cleaner PSF. Edit the **nstar** output file **n602csb.nst.1** and delete the lines associated with the PSF stars, leaving only the neighbors behind. You can recognize the PSF stars, as they are the first entry in each group. When you are done with this editing job, re-run **substar**, using the edited ".nst" file as the photometry file. Again in running **substar** make sure you **verify** the PSF radius to the smaller value you decided above. Examine the results on the image display. Now the PSF stars should be there but the neighbors should be cleanly subtracted. Are they? If so, you are ready to proceed. If not, re-read the above and keep at it until you get those neighbors reasonably well out of the frame.

We can now run **psf** on the subtracted frame—the one with only the neighbors gone. We have added some noise by doing the subtraction, and so we should reset **datamin** to several sigma below the previously used value. We are going to have to do more typing this time

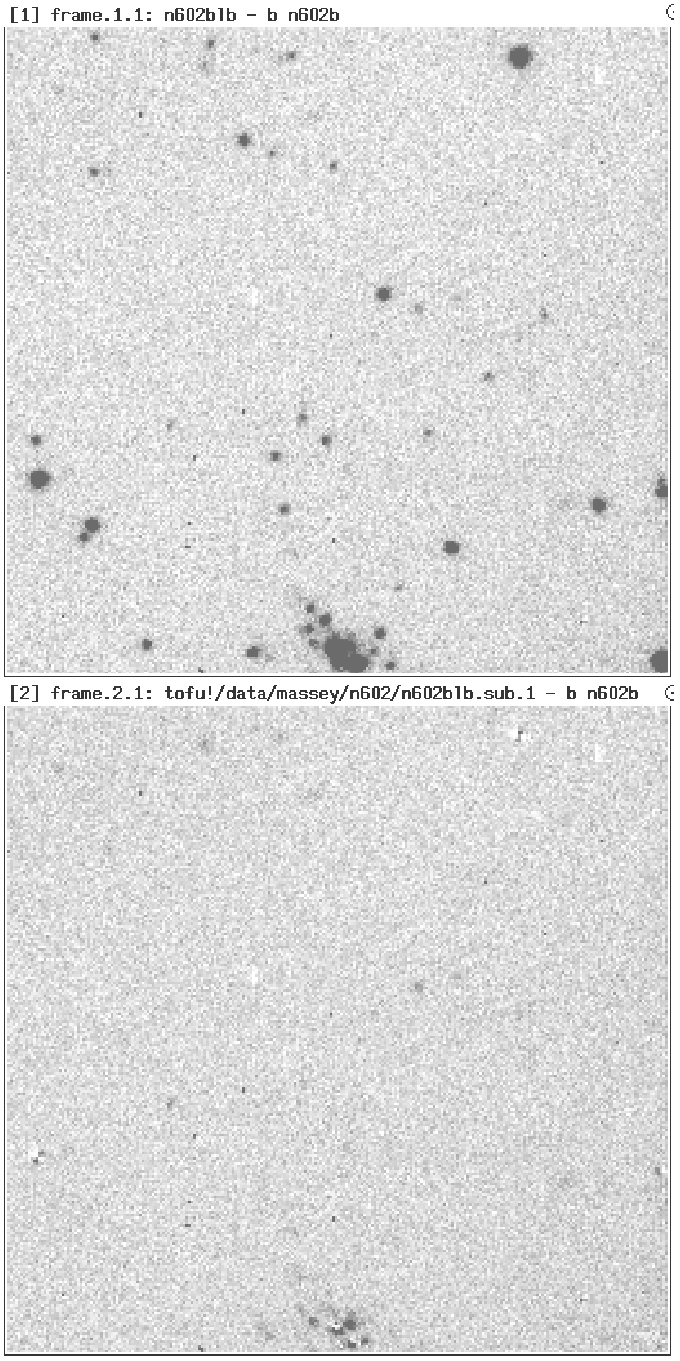

Figure 28: A "before" and "after" pair of images, where the PSF was constructed with a star that was too bright. Note the systematic residuals for the two bright stars. A "bad" PSF star would result in a similar effect; however, in these data we found that there was always a systematic effect if the PSF stars were about 18000 ADU.

when we run it, as the defaults for things will get very confused when we tell it that the "Image for which to build PSF" is actually **n60csb.sub.1**. For the "Aperture photometry file" we can tell it the original photometry file **n602csb.mag.1** if we want, or even the old ".psg" file **n602csb.psg.1** since every star that we are concerned about (PSF star plus neighbor) is there. Go ahead and give it the next 'version" number for the "Output psf image" **n602csb.psf.2** and for the "Output psf group file" **n602csb.psg.2**. We can of course do this all on the command line:

psf n602csb.sub.1 n602csb.mag.1 n602csb.psf.2 n602csb.psg.2 datamin=-150.

An example is shown in Figure 29. *This time make sure you take the large psf radius.* Make a new PSF using the cursor as before.

How good is this revised PSF? There's only one way to find out: run **nstar** on the original frame, this time keeping the psf radius large. Then do **substar** and examine the frame with both the PSF stars and neighbors subtracted. Does this show a substantial improvement over the first version? Now that you have a cleaner PSF it may be necessary to repeat this procedure (edit the **n602csb.nst.2** file, remove the PSF stars, run **substar** using this edited file to produce a frame with the just the neighbors subtracted this time using a better PSF, run **psf** on this improved subtracted frame) but probably not.

4.8 Doing the psf-fitting: allstar.

The next step is to go ahead and run simultaneous PSF-fitting on all your stars, and produce a subtracted frame with these stars removed. To do both these things you need only run **allstar**. The defaults are likely to be right: see Figure 30. As you may imagine, **allstar** produces a photometry file **n602csb.als.1**, and another subtracted image: **imagename.sub.N**.

Display the subtracted frame, and blink it against the original. Has IRAF/daophot done a nice job? If the stars are clearly gone with a few hidden ones now revealed, you can be proud of yourself—if the results are disappointing, there is only one place to look, and that is in the making of the PSF. Assuming that all is well, it is now time to add those previously hidden stars into the photometry. The easiest way to do this is to run **daofind** on the subtracted image. Set the value of **datamin** to a value several sigma lower than what you had used earlier in case the subtraction process generated some spuriously small values, and you will want to *increase* the value of threshold by 1 or 2 sigma above what you used previously. Why? Because the subtraction process has certainly added noise to the frame, and if you don't do this you will be mainly adding spurious detections. Use **tvmark** as before to examine the results of **daofind**; remember that the coordinate file name will be **imagename.sub.N.coo.1** this time around. If you are really close, but want to add a couple of stars, re-run **tvmark** on this file using **interactive+**; this will allow you to add (and delete) coordinates from the file.

Now run **phot** using this new coordinate file as the input list. However, you do want to use the *original* frame for this photometry; otherwise the sky values for the newly found stars will be very messed up owing to the many subtracted images. A new aperture photometry file **n602csb.mag.2** will have been produced. Use **append** to concatenate these two files: **append n602csb.mag.1,n602csb.mag.2 n602csb.mag.3**. You can now re-run **allstar**

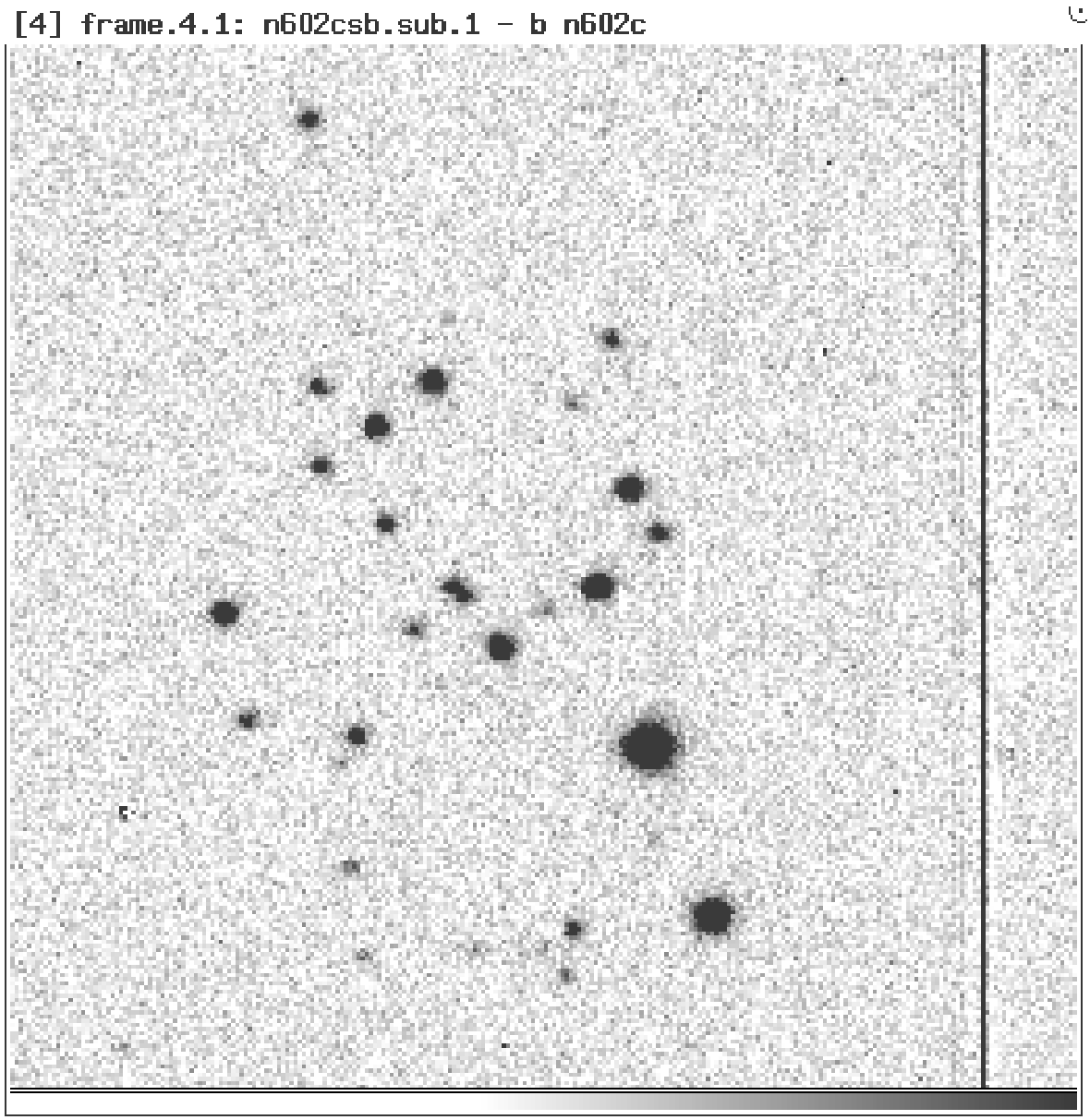

Figure 29: Making the first revision PSF using the frames with the neighbors subtracted. Compare this to Figure 23, which shows the same region before the neighbors have been removed.

```
                                    I R A F
                      Image Reduction and Analysis Facility
PACKAGE = daophot
   TASK = allstar

image   =              n602csb  Image corresponding to photometry
photfile=              default  Input photometry file (default: image.mag.?)
psffile =              default  PSF image (default: image.psf.?)
allstarf=              default  Output photometry file (default: image.als.?)
subimage=              default  Subtracted image (default: image.sub.?)
(cache  =                   no) Cache the data in memory ?
(datapar=                    ) Data dependent parameters
(daopars=                    ) DAOPHOT parameters
(verbose=                   no) Print messages
(verify =                  yes) Verify critical ALLSTAR parameters
(mode   =                   ql)

:go

Psf radius in pixels (11.):
        New psf radius: 11. pixels
Fitting radius in pixels (2.5):
        New fitting radius: 2.5 pixels
Maximum group size in number of stars (60):
        New maximum group size: 60 stars
```

Figure 30: Running **allstar**.

using this combined photometry file as the input.

4.9 Matching the frames

In the example here we have been reducing the *B* frame of a set of *UBV*. Once all three frames have been reduced it is often necessary to do a little fiddling. Have the same stars been identified in each group? In many cases you don't want the same stars to have been identified in each clump—afterall, some stars are red, some are blue (that's presumably why you are doing this afterall, right?), but in some cases you may find that a clump was identified as three objects on the *U* and the *V* frames and clearly should have been three on the *B* frame but instead is four or two. What to do?

Using **tvmark** it is relatively easy to set this right. First we need to use **txdump** to produce a file for each frame that can be displayed. Do something like an

txdump n602csu.als.2 > tvu

followed by an

txdump n602csb.als.2 > tvb

and a

txdump n602csv.als.2 > tvv

In each case select **xc,yc** and use **MAG!=INDEF** as a selection criteria. Thus you will then have three text files that contain only the x's and y's of the stars with photometry.

Next display the three frames (**display n602csu 1**, **display n602csb 2**, **display n602csv 3**) and put colored dots up to denote the different allstar stars:

tvmark 1 tvu color=204 inter-,
tvmark 2 tvb color=205 inter-,

and

tvmark 3 tvv color=206 inter-

will give pleasing results. Zoom, pan, register, and blink around the frames until you are convinced that you really do want to add or delete a star here or there. If you want to add or delete a star to the *U* frame list, do a

tvmark 1 tvu color=203 inter+

You are now in interactive mode, and centering the cursor on the star you want to add and striking the "a" key will append the x and y value of the cursor the tvu list. Similarly, striking the "u" key will delete a star from the list if you are using IRAF v2.9 or later. (For earlier versions you are just going to have to do a little editing by hand, good luck!) The star you add or delete will have a white dot appear on top of it. If you need to switch to a different coordinate file, simply exit the interactive **tvmark** with a "q" and re-execute it specifying, for example, **tvmark 3 tvv color=203 inter+**.

When you are done with adding and deleting stars, then it is time to redo the photometry. Do a **phot n602csu coords=tvv datamin=100** in order to generate new aperture photometry and sky values. These can then be run through **allstar**, and the procedure repeated for each of the frames.

4.10 Determining the Aperture Correction

The zero-point of your magnitudes have been set as follows. When you ran **phot** using a small aperture (3 pixels in the example above) magnitudes were defined as -2.5 * log(Counts above sky)/(Exposure time) + Const. (The constant Const was hidden away in **photpars** and is the magnitude assigned to a star that had a total of one ADU per second within the measuring aperture you used.) When you defined your PSF the magnitudes of the PSF stars determined from the aperture photometry were then used to set the zero-point of the PSF. However, your standard stars were presumably measured (if you did things right) through a much larger aperture, and what we must do now is measure how much brighter the PSF would have been had its zero-point been tied to the same size aperture used for the standard stars.

We need to determine the aperture correction from the brightest, unsaturated stars (so there will still be reasonable signal above sky at the size of the large aperture); if you can pick out stars that are reasonably well isolated, so much the better. If this sounds vaguely familiar to you, you're right—this is basically what you did for selecting PSF stars, and these would be a good starting point for selecting stars for determining the aperture correction. Ideally you would like to use at least five such stars, but since when is data reduction ideal? Nevertheless, it is in the determination of the aperture correction the largest uncertainty enters in doing CCD photometry on crowded fields.

We will first need to pick out the brightest, isolated stars and then to subtract off any stars that might affect their being measured through the large "standard star" aperture (e.g., something like 15 pixels). To do this we need good photometry of any of these neighbor stars, and we describe two ways to do this (1) the very long complicated way, and (2) the very short easy way:

1. **Method 1: Using the image display** We can also use **tvmark** to mark the stars that we wish to use for aperture photometry. First we should remind ourselves what are multiple stars and what aren't: **display** the image, and then use **tvmark** to mark the stars with **allstar** photometry:

   display n602csb 1
   txdump n602csb.als.2 xc,yc yes > tvb
   tvmark 1 tvb color=204 interact-

   Now go through and mark the stars you want to use as the aperture correction stars *plus any neighbors that might contribute light to a large aperture centered on the bright stars:*

   tvmark 1 bapstars color=203 interact+

   Use the "a" key to generate a list (**bapstars**) of the approximate x and y positions of these stars. Next run this list through **phot** to generate improved centers and good sky values:

   phot n602csb bapstars bapphot calgor="centroid"

   Next run the photometry output file **bapphot** through **group**:

```
ap> apselect n602csb.nst.3 id,group.mag,yes
8  1  13.877
7  1  18.376
10  1  17.156
5  2  15.037
13  3  16.520
16  3  18.214
31  3  19.336
17  3  17.788
```

Figure 31: The three PSF stars and their groups.

group n602csb bapphot default default crit=0.2

This will have generated a "group" file **n602csb.grp.1**.

Finally (!) run this group file through **nstar**:

nstar n602csb default default default

2. **Method 2: Using the ".psg" files** If you used a goodly number ($> 3-5$, say) stars in making the PSF, then we will simply use these stars as the aperture correction stars. Your last **nstar** run should have produced an "**.nst**" file that contains good photometry for the PSF stars *and* their neighbors. (If you don't remember if you did this, run **nstar** using the "**.psg**" as the input group file.) Note that this method relies upon the assumption that the sum of the psf radius and psf fitting radius is about as large as the size of the large aperture you will use, so that all the important neighbors have been included in the point-spread-function group, but this is probably a reasonable assumption.

Now that we are done with the preliminaries (!!), we now want to produce two files: one of them containing only the neighbors that we wish to subtract off, and another containing only the bright isolated stars which we want to use in computing the aperture correction. To do this we will use **group** to divide up the "**.nst**" file (we could simply use the editor but that would be a lot of work). First we will use **txdump** on the **nstar** file to see the magnitude range covered by the PSF stars and their neighbors: hopefully there won't be any overlap. To do this try

txdump n602csb.nst.3 id,group,mag yes

In the example shown in Figure 31 we see that the PSF stars have magnitudes of 13.9, 15.0, and 16.5 in the three groups; all the neighbor stars are fainter than 17.0. Thus we can use **select** to create a file containing the photometry of the faint stars:

select n602csb.nst.3 n602csbsub

and answer **MAG>17.0** when you are queried for the "Boolean expression". This will put the photometry of the stars you wish to get rid of into the file **n602csbsub**. Next do an

```
da> phot n602csb.sub.2 n602csbap n602csbapresults apertures=3,15 ann=20 dan=5

Centering algorithm (none):
        New centering algorithm: none
Sky fitting algorithm (mode):
        Sky fitting algorithm: mode
Inner radius of sky annulus in scale units (20.):
        New inner radius of sky annulus: 20. scale units 20. pixels
Width of the sky annulus in scale units (5.):
        New width of the sky annulus: 5. scale units 5. pixels
Standard deviation of background in counts (INDEF):
        New standard deviation of background: INDEF counts
File/list of aperture radii in scale units (3,15):
        Aperture radius 1: 3. scale units 3. pixels
        Aperture radius 2: 15. scale units 15. pixels

n602csb.sub.2 x: 184.20 y: 198.94 s: 157.49 m: 15.872 15.501 e: ok
n602csb.sub.2 x: 199.90 y: 158.08 s: 157.94 m: 17.031 16.681 e: ok
n602csb.sub.2 x: 149.08 y: 222.37 s: 157.44 m: 18.508 18.115 e: ok
```

Figure 32: The aperture correction run of **phot**.

txdump n602csb.nst.3 xc,yc > n602csbap

and answer **MAG<17.0** in response to "Boolean expression". This will put the x and y values of the stars we wish to use for the aperture correction into the file **n602csbap**. Next subtract the stars in the first file:

substar n602csb n602csbsub

and accept the defaults. This will result in the subtracted image **n602csb.sub.N**. It is this file on which we wish to run the aperture photometry to determine the aperture correction:

phot n602csb.sub.N n602csbap n602csbapresults apertures=3.,15. annulus=20. dannu=5.

You will see something like Figure 32 on your terminal. In this example we've made the assumption that the aperture size that set your zero-point in making the PSF was 3 pixels (i.e., what you used with **phot** Way Back When), and that the aperture size used on your standard stars was 15 pixels. It is time to drag out your hand calculator. Using all three stars we find an average aperture correction of -0.371 with a standard deviation of the mean of 0.012 mag; given the large range in magnitude, I might have been tempted to ignore the two fainter stars and keep the aperture correction based only upon the brightest star (the frame is sparsely populated, and there isn't a whole heck of a lot else we can do). By an amazing coincidence, the aperture correction based just on the brightest star is also -0.371.

4.11 daophot summary

- Set up **datapars** and **daopars**.

  1. Do an **imhead** on some image and note the keywords for the filter position, the effective exposure time, and the effective airmass.
  2. Use **display** and **imexamine** on a few frames to determine the typical full-width-half-max of stars and what would be a good value to use for the radius of the psf (i.e., what radius will contain the brightest star for which you wish to do photometry.)
  3. Enter these into **daopars** (psfrad) and **datapars** (header key words, fwhm). Also check that the correct values are entered in **datapars** for the gain (photons per ADU) and read-noise (in electrons), as well as the "maximum good data value".

- Find stars.

  1. Do an **implot** or **imexamine** to determine the sky level on your frame. Calculate the expected 1σ error.
  2. Enter the sky value minus 3σ as your value for **datamin** in **datapars**.
  3. Run **daofind** using as a threshold value 3 to 5 σ.
  4. Use **tvmark** to mark the stars found (**imagename.coo.1**). If you need to, rerun **daofind** with a larger or small threshold.

- Run aperture photometry using **phot**.

- Generate a PSF. Run **psf** and add stars using the "a" key. Try to select bright, uncrowded stars. Then:

  1. Run **nstar** using the file **imagename.psg.1** as the "input photometry group" file. If there are neighbors, be sure to decrease the psf radius as explained above. Run **substar** (also using the smaller sized psf radius) and display the resultant subtracted frame **imagename.sub.1**. Do the residuals of the PSF stars look consistent, or is one of them funny? If need be, start over.
  2. Remove any neighbor stars by editing the PSF stars out of the ".nst" file, and rerunning **substar**. Run **psf** on the subtracted file, using the normal psf radius again. You will have to over-ride the defaults for the input and output file names now that you are using the subtracted image. Rerun **nstar** on the original frame using the normal psf radius and the revised PSF. Run **substar** and display the results. Are the PSF stars nicely removed, and do the areas around the PSF stars look clean? It may be necessary to remove neighbors again using this revised PSF.

- Run **allstar**. Display the subtracted frame and see if your stars have been nicely subtracted off.

- Run **daofind** on the subtracted frame, using a value for **threshold** which is another σ or two larger than before, and a value for **datamin** which is several σ lower than before. Use **tvmark** to examine the results, and if need be run **tvmark** interactively so that you may add any extra stars.
- Run aperture photometry using **phot** *on the original frame*, using the new coordinate list produced above.
- **append** the two aperture photometry files.
- Run **allstar** using the combine photometry file.
- Repeat all of the above for each frame in your "set" (e.g., all short and long exposures in each filter of a single field, say.
- Use **txdump** to select the stars from the allstar files which have magnitudes not equal to "INDEF". Mark these stars using **tvmark**, and then use the capabilities of the image display and **tvmark** to match stars consistently from frame to frame. Rerun **phot** and **allstar** on the final coordinate lists.
- Determine the aperture corrections.
- Transform to the standard system (see the next section) and then publish the results.

5 Transforming to the Standard System

This section will eventually tell you how to easily and painless obtain the transformation equations for going from your instrumental magnitudes to the standard system, and how to apply these transformation equations to your program fields. Unfortunately, the IRAF routines for doing this are still under construction. In the meanwhile, we are providing here a kludge solution that can be used by initiates of Stetson's VMS CCDCAL routines. If you haven't been made a member of the club yet, and don't feel like waiting until the IRAF routines are become available before you get results, then I would recommend getting a hold of the good Dr. Stetson and bribing him until he offers to send you a copy of CCDCAL. There is an excellent manual that comes along with it, and we will not attempt to repeat any of that material here.

5.1 Standard Star Solution

First we will describe how to get output good enough to fool the CCDCAL software into believing the photometry was produced by CCDOBS (for the standard magnitudes), and what modifications need to be made to CCDSTD.FOR

On the standard file do a **txdump standstuff lid,ifilt,xair,mag,merr > foolit** to dump the star number, filter number, airmass, and instrumental magnitudes and errors into the file **foolit**. Unfortunately, you are now going to have to edit this file and stick in the star name

(in what ever form you have it in creating the library of standard stars with CCDLIB) in place of the image name and star ID. (These were simply placed in the file to help guide you). While you are at it, line up the filter numbers, airmasses, and magnitudes into nice, neat columns. When you get done, stick in a line at the top that gives the number of instrumental magnitudes and their names, using a i1,13x,n(6x,a6) format. For instance, in the case shown here there are 3 instrumental magnitudes, U, B, and V. Finally, the filter numbers have to be edited so they agree with these (e.g., they must denote instrumental magnitude 1, 2, and 3...now aren't you sorry you didn't decide to wait until the IRAF routines were finished?). In Figure 33 we show an example of the "before" and "after" file.

CCDOBS.FOR itself now needs to be modified. Search for line statement "1120" (which will say JSTAR=JSTAR+1). Add a line that sets the integration time to 1 (tint=1.). Modify the READ statement as shown in Figure 34, and finally modify the 213 FORMAT statement so it actually matches your data file. You should now be able to compile, link, and run this modified version of CCDOBS and have it work on your standard star data.

5.2 Program Stars

The work required for faking "CCDCAL" is actually a lot less. The data files are easily produced. Do a

txdump n602csu.als.2 id,xc,yc,mag,merr,nit,chi > csu
txdump n602csb.als.2 id,xc,yc,mag,merr,nit,chi > csb
txdump n602csv.als.2 id,xc,yc,mag,merr,nit,chi > csv

answering **MAG!=INDEF** to "boolean expression" each time. These three files (**csu**, **csb**, **csv** can be used with CCDCAL once a single modification is made to CCDCAL.FOR: on statement number 2020 change the format to "free format", e.g., 2020 IF(NL(IOBS).NE.2) READ(2,*,END=2040). When CCDCAL queries you for an integration time, be sure to tell it 1.0, as your data have already been corrected for exposure times.

6 Acknowledgements

We are grateful to Jeannette Barnes and Carol Neese for critical readings of this document, although final blame for style and content of course rests with the authors.

```
std145.imh  1  2  1.359431  13.801  0.008
std146.imh  1  5  1.364883  16.047  0.010
std147.imh  1  3  1.370639  13.910  0.009
std148.imh  1  3  1.573533  14.820  0.010
std149.imh  1  2  1.580884  14.613  0.008
std150.imh  1  5  1.589008  16.963  0.010
std151.imh  1  2  1.262856  14.916  0.006
std152.imh  1  3  1.26529  15.118  0.008
std153.imh  1  5  1.268781  16.997  0.006
std154.imh  1  2  1.202828  13.429  0.010
std155.imh  1  3  1.201648  13.301  0.009
std156.imh  1  5  1.200005  16.560  0.010
std157.imh  1  5  1.625146  15.180  0.005
std158.imh  1  2  1.616683  12.307  0.006
std159.imh  1  3  1.605717  12.313  0.006

000000000111111111122222222233333333334444444444
123456789012345678901234567890123456789012345678
3                  U          B          V
(this line intensionally left blank)
bd-11162  2  1.359431  13.801  0.008
bd-11162  1  1.364883  16.047  0.010
bd-11162  3  1.370639  13.910  0.009
f11       3  1.573533  14.820  0.010
f11       2  1.580884  14.613  0.008
f11       1  1.589008  16.963  0.010
f24       2  1.262856  14.916  0.006
f24       3  1.26529   15.118  0.008
f24       1  1.268781  16.997  0.006
96-36     2  1.202828  13.429  0.010
96-36     3  1.201648  13.301  0.009
96-36     1  1.200005  16.560  0.010
98-653    1  1.625146  15.180  0.005
98-653    2  1.616683  12.307  0.006
98-653    3  1.605717  12.313  0.006
```

Figure 33: The output of **txdump** and the final file ready for **ccdstd**. Note the switching of the filter number "5" with "1".

```
C
C Read in the next star.
C
 1120 JSTAR=JSTAR+1
        tint=1.
      READ (2,213,END=1900)STARID(JSTAR),IMAG(JSTAR),
     .       X(JSTAR),OBSMAG(JSTAR),OBSERR(JSTAR)
  213 FORMAT(A10,i1,2x,f8.6,1x,f7.3,1x,f6.3)
      T(JSTAR)=THRS=60.*TMIN
      OBSMAG(JSTAR)=OBSMAG(JSTAR)+2.5*ALOG10(TINT)
      OBSERR(JSTAR)=OBSERR(JSTAR)**2
```

Figure 34: Modifications to CCDOBS.FOR