A User's Guide to Stellar CCD Photometry with IRAF

Philip Massey Lindsey E. Davis

April 15, 1992

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 **1**

2 Getting Started **2**
2.1 Fixing your headers 2
2.1.1 Correcting the exposure time 2
2.1.2 Computing the effective airmass 4
2.2 Dealing with Parameter Files (Wheels within Wheels) 4

3 Stanard Star Photometry and Reductions **5**
3.1 Obtaining Aperture Photometry of Your Standards 5
3.2 Picking an Aperture Size 7
3.3 Setting Things Up 7
3.4 Doing It: Aperture Photometry at Last 9
3.4.1 Automatic star finding 12
3.4.2 Photometry by Eye 14
3.5 Examining the Results: the power of **txdump** 16
3.6 The Standard Star Solutions 20
3.7 Making the Standard Star Catalog 20
3.8 Making the Standard Star Observations File 21
3.9 Defining the Transformations 24
3.10 Solving Those Transformation Equations 29

4 Crowded Field Photometry: IRAF/daophot **34**
4.1 Historical Summary 34
4.2 **daophot** Overview 34
4.3 How Big Is A Star: A Few Useful Definitions 36
4.4 Setting up the parameter files "daopars" and "datapars" 36
4.5 Finding stars: **daofind** and **tvmark** 38
4.6 Aperture Photometry with **phot** 45
4.7 Making the PSF with **psf** 47
4.8 Doing the psf-fitting: **allstar**. 55
4.9 Matching the frames 56
4.10 Determining the Aperture Correction 57
4.11 **daophot** summary 59

5 Transforming to the Standard System **61**

A imexamine: A Useful Tool **66**

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. The first version of this manual became available two years ago in Jan 1990; the current version includes revisions to V2.10 of IRAF, new examples, and descriptions of the the PHOTCAL package for doing the standard star solutions and transformations.

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) computing the transformations from instrumental to standard magnitudes and indices, (5) obtaining instrumental magnitudes for your program stars using IRAF/daophot, (6) determining the aperture correction for your program stars,and (7) applying the transformations to your program photometry. We choose to illustrate these reductions using *UBV* CCD data obtained with a TI chip at the prime-focus of the 4m 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 recommend you first study the *A Beginner's Guide to Using IRAF* by Jeannette Barnes. (Actually, if you are a brand-new IRAF user, we recommend that you find some simpler task to work on before proceeding to digital stelalr 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 windows and IRAF; the best description is to be found in *A Quick Look at SunIRAF on the Tucson Sun Network* by Jeannette Barnes.

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.

Three things that will be useful to have in your headers are the exposure time, the airmass (or information sufficient for computing the airmass), and the filter identification. If you are reducing NOAO data then all these data are probably there, and you can proceed to Sec. 2.1.1. Otherwise, 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**"EXPTIME"** 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", and "EPOCH", as well as the siderial-time ("ST"), 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 4m P/F shutter has an effective exposure time that is insignificantly different than the requested exposure time (Walker 1988 N̂OAO Newsletter **No. 13**, 20); the correction is +0.004 seconds. However, some shutters both at Tololo and Kitt Peak have very significant shutter corrections, some as much as a tenth of a second! Therefore we will go through the exercise of correcting the exposure time although in truth it does not matter for the current example.

First see what "keyword" in your header gives the exposure time:

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

will produce a listing such as given in Figure 1. The exposure time keyword in this header is "EXPTIME". In this case we wish to add a new exposure time to each of the headers; we will call this corrected exposure time CEXPTIME, and make it 4 ms larger than whatever value is listed as EXPTIME. To do this we use the **hedit** command as follows:

hedit *.imh CEXPTIME "(EXPTIME+0.004)" ver- show+ add+.

An inspection of the headers will now show a new keyword CEXPTIME.

```
obj003[797,797][real]: V G93-48
    No bad pixels, no histogram, min=-1570.715, max=32540.18
    Line storage mode, physdim [896,797], length of user area 1580 s.u.
    Created Fri 09:51:10 06-Dec-91, Last modified Fri 09:51:10 06-Dec-91
    Pixel file 'tofu!/data1/massey/pixels/obj003.pix' [ok]
    'KPNO-IRAF'  /
    '06-04-89'  /
    'KPNO-IRAF'  /
    '12-10-88'  /
    New copy of obj003.imh
    IRAF-MAX=          3.254018E4  /  DATA MAX
    IRAF-MIN=         -1.570715E3  /  DATA MIN
    IRAF-B/P=                  32  /  DATA BITS/PIXEL
    IRAFTYPE= 'FLOATING'  /
    IRAF-MAX=          3.254018E4  /  DATA MAX
    IRAF-MIN=         -1.570716E3  /  DATA MIN
    IRAF-B/P=                  32  /  DATA BITS/PIXEL
    IRAFTYPE= 'FLOATING'  /
    OBSERVAT= 'CTIO              ' / ORIGIN OF DATA
    CCDPICNO=                  3  / ORIGINAL CCD PICTURE NUMBER
    EXPTIME =                  1  / ACTUAL INTEGRATION TIME (SECONDS)
    DARKTIME=                  1  / TOTAL ELAPSED TIME (SECONDS)
    OTIME   =                  1  / SHUTTER OPEN TIME (SECONDS)
    IMAGETYP= 'OBJECT            ' / OBJECT,DARK,BIAS,ETC.
    DATE-OBS= '12/10/88          ' / DATE (DD/MM/YY) OF OBSERVATION
    RA      = '21:51:59.00'  /  right ascension
    DEC     = ' 2:33:31.00'  /  declination
    EPOCH   =              1985.  / EPOCH OF RA AND DEC
    ZD      = ' 0:00:00.00'  /  zenith distance
    HA      = '00:00:00          ' / HOUR ANGLE (H-M-S)
    UT      = ' 0:07:59.00'  /  universal time
    ST      = '20:47:55.00'  /  sidereal time
    AIRMASS =               1.23  / AIRMASS
    DETECTOR= 'TI1               ' / DETECTOR (CCD TYPE, PHOTON COUNTER, ETC)
    CAMTEMP =                 0.  / CAMERA TEMPERATURE, DEG C
    DEWTEMP =                 0.  / DEWAR TEMPERATURE, DEG C
    FILTERS = '2 0               ' / FILTER BOLT POSITIONS
    PREFLASH= 1.000000000000000E-4  / PREFLASH TIME (SECONDS)
    VEBGAIN =                 38  / VEB GAIN SETTING
    SECPPIX =              0.292  / SEC OF ARC PER PIXEL
    TRIM    = 'Oct 12  4:09 Trim data section is [4:800,4:800]'
    OVERSCAN= 'Oct 12  4:09 Overscan section is [815:833,4:800] with mean=413.7
    ZEROCOR = 'Oct 12  4:09 Zero level correction image is Zero2.imh'
    FLATCOR = 'Oct 12  4:09 Flat field image is flat902 with scale=11735.'
    CCDSEC  = '[4:800,4:800]'
    CCDPROC = 'Oct 12  4:10 CCD processing done'
```

Figure 1: Header information for image obj003.

```
as> lpar setairmass
       images = "*.imh"         Input images
 (observatory = )_.observatory) Observatory for 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)\n
        (show = yes)            Print the airmasses and mid-UT?
      (update = yes)            Update the image header?
    (override = yes)            Override previous assignments?
        (mode = "ql")
```

Figure 2: Running **setairmass**.

2.1.2 Computing the effective airmass

Chances are any airmass in your header is the airmass at the *beginning* of your exposure. What you really want is the "effective airmass"—the airmass at mid-exposure. 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. You may have noticed in Fig. 1 that the hour-angle (HA) and zenith-distance (ZD) keywords in fact are wrong; the Tololo computers were returning 0.0 in order to keep the astronomer on his/her toes. Fortunately, **setairmass** ignores these two keywords; you could, if you wish, have deleted them with **hedit *.imh ZD,HA delete+**.

The task **setairmass** relies upon the header word **OBSERSVAT** (see Fig. 1) to identify the obseratory where the observations were made, and hence what latitude to use in computing the airmass. This keyword will be found in all NOAO headers, but if your data comes from somewhere else, be sure to first run the **observatory** task to specify this.

A sample run of **setairmass** is show in Figure 2.

2.2 Dealing with Parameter Files (Wheels within Wheels)

The photometry packages **daophot** and **apphot** both contain 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 (control-d on some machines) 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). 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. 4.7). 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 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 Stanard Star Photometry and Reductions

3.1 Obtaining Aperture Photometry of 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 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 routines are actually identicial, and the only difference in this instance is that the two packages have their own data parameter files.

```
                                 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.?)"
skyfile =                       Sky file
(datapar=                     ) Data dependent parameters
(centerp=                     ) Centering parameters
(fitskyp=                         ) Sky fitting parameters
(photpar=                     ) Photometry parameters
(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=                  10.) Inner radius of sky annulus in scale units
(dannulu=                  10.) Width of sky annulus in scale units
(skyvalu=                   0.) User sky value
(smaxite=                   10) Maximum number of sky fitting iterations
(snrejec=                   50) Maximum number of sky fitting rejection 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
(mkskey =                   no) Mark sky annuli on the display
(mode   =                   ql)
```

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

3.2 Picking an Aperture Size

Unfortunately, there are no 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 data. 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 photometry 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.**

3.3 Setting Things Up

As discussed in Sec. 2.2 we must set up 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.

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

(scale  =                   1.) Image scale in units per pixel
(fwhmpsf=                  4.0) FWHM of the PSF in scale units
(emissio=                  yes) Features are positive ?
(sigma  =                INDEF) Standard deviation of background in counts
(datamin=                 -30.) Minimum good data value
(datamax=               32000.) Maximum good data value
(thresho=                  50.) Detection threshold in counts above background
(cthresh=                   0.) Centering threshold in counts above background
(noise  =              poisson) Noise model
(ccdread=                     ) CCD readout noise image header keyword
(gain   =                     ) CCD gain image header keyword
(readnoi=                  10.) CCD readout noise in electrons
(epadu  =                  1.1) Gain in electrons per count
(exposur=             CEXPTIME) Exposure time image header keyword
(airmass=              airmass) Airmass image header keyword
(filter =              FILTERS) Filter image header keyword
(obstime=             UTMIDDLE) Time of observation image header keyword
(itime  =                INDEF) Exposure time
(xairmas=                INDEF) Airmass
(ifilter=                INDEF) Filter
(otime  =                INDEF) Time of observation
(mode   =                   ql)
```

Figure 4: Parameters for **datapars**.

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., $-3 \times 10. \div 1.1 \approx -30$ in this example. We will consider when we do the PSF-fitting is if the chip is really and truly linear up to the value quoted; 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 after all the overscan bias was at a level of about 400; furthermore, the flat-field was not identically 1.000)

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 4.0 pixels and so we have changed the default cbox from 5 to 8.

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 need to specify the size (radius) of the aperture we wish to use in measuring our standards.

3.4 Doing It: Aperture Photometry at Last

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. 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.

```
obj287[797,797][real]: U F16
    No bad pixels, no histogram, min=-199.1429, max=6163.634
    Line storage mode, physdim [896,797], length of user area 1661 s.u.
    Created Fri 10:04:58 06-Dec-91, Last modified Fri 10:04:58 06-Dec-91
    Pixel file 'tofu!/data1/massey/pixels/obj287.pix' [ok]
    'KPNO-IRAF'  /
    '06-04-89'  /
    'KPNO-IRAF'  /
    '13-10-88'  /
    New copy of obj287.imh
    IRAF-MAX=           6.163634E3  /  DATA MAX
    IRAF-MIN=          -1.991429E2  /  DATA MIN
    IRAF-B/P=                   32  /  DATA BITS/PIXEL
    IRAFTYPE= 'FLOATING'  /
    IRAF-MAX=           6.163634E3  /  DATA MAX
    IRAF-MIN=          -1.991430E2  /  DATA MIN
    IRAF-B/P=                   32  /  DATA BITS/PIXEL
    IRAFTYPE= 'FLOATING'  /
    OBSERVAT= 'CTIO             ' / ORIGIN OF DATA
    CCDPICNO=                 287  / ORIGINAL CCD PICTURE NUMBER
    EXPTIME =                   2  / ACTUAL INTEGRATION TIME (SECONDS)
    DARKTIME=                   2  / TOTAL ELAPSED TIME (SECONDS)
    OTIME   =                   2  / SHUTTER OPEN TIME (SECONDS)
    IMAGETYP= 'OBJECT           ' / OBJECT,DARK,BIAS,ETC.
    DATE-OBS= '12/10/88         ' / DATE (DD/MM/YY) OF OBSERVATION
    RA      = ' 1:52:29.00'  /  right ascension
    DEC     = '-7:00:31.00'  /  declination
    EPOCH   =               1950.  / EPOCH OF RA AND DEC
    UT      = ' 9:24:47.00'  /  universal time
    ST      = ' 6:06:16.00'  /  sidereal time
    AIRMASS =             2.217149 / AIRMASS
    DETECTOR= 'TI1              ' / DETECTOR (CCD TYPE, PHOTON COUNTER, ETC)
    CAMTEMP =                  0.  / CAMERA TEMPERATURE, DEG C
    DEWTEMP =                  0.  / DEWAR TEMPERATURE, DEG C
    FILTERS = '0 0              ' / FILTER BOLT POSITIONS
    PREFLASH= 1.00000000000000E-4  / PREFLASH TIME (SECONDS)
    VEBGAIN =                  38  / VEB GAIN SETTING
    SECPPIX =               0.292  / SEC OF ARC PER PIXEL
    TRIM    = 'Oct 12  6:36 Trim data section is [4:800,4:800]'
    OVERSCAN= 'Oct 12  6:36 Overscan section is [815:833,4:800] with mean=407.4'
    ZEROCOR = 'Oct 12  6:36 Zero level correction image is Zero2.imh'
    FLATCOR = 'Oct 12  6:36 Flat field image is Uskytotal with scale=24015.'
    CCDSEC  = '[4:800,4:800]'
    CCDPROC = 'Oct 12  6:36 CCD processing done'
    CEXPTIME=                2.004
    UTMIDDLE= '9:24:48.0'
```

Figure 5: Header information for obj287.imh

```
PACKAGE = daophot
   TASK = centerpars

(calgori=            centroid) Centering algorithm
(cbox   =                  8.) 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**.

```
                              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**.

3.4.1 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. (More on **imexamine** can be found in App. A.) 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, fifty 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 ($50 \times 10. \div 1.1 \approx 500$).

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 about 3.1 mag brighter than the threshold, and the other 1.0 mag brighter than the 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,mag 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 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 obj002.imh.coo.1**) and simply delete the extras.

```
                                   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
Results file (default: image.coo.?) (default):

FWHM of features in scale units (4.) (CR or value): 4.2
        New FWHM of features: 4.2 scale units  4.2 pixels
Detection threshold in counts above background (50.) (CR or value): 500.
        New detection threshold: 500. counts


Image: obj001.imh  fwhmpsf: 4.2  ratio: 1.  theta: 0.  nsigma: 1.5

   57.97   202.42   -0.395   0.293   0.226     1
  303.97   334.75   -3.872   0.441  -0.003     2
  386.92   547.80   -2.440   0.409   0.101     3

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


Image: obj002.imh  fwhmpsf: 4.2  ratio: 1.  theta: 0.  nsigma: 1.5

  306.94   334.02   -3.147   0.367  -0.058     1
  390.02   546.96   -1.000   0.351  -0.031     2

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

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

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".

3.4.2 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 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 obj279 1

will display image **obj279.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 obj279 maxshift=3.

(Note that even though **maxshift** is a parameter of **centerpars** we can change it on the

```
#K IRAF       = NOAO/IRAFV2.10BETA     version    %-23s
#K USER       = massey                 name       %-23s
#K HOST       = tofu                   computer   %-23s
#K DATE       = 12-27-91               mm-dd-yr   %-23s
#K TIME       = 09:40:48               hh:mm:ss   %-23s
#K PACKAGE    = apphot                 name       %-23s
#K TASK       = daofind                name       %-23s
#
#K SCALE      = 1.                     units      %-23.7g
#K FWHMPSF    = 4.2                    scaleunit  %-23.7g
#K EMISSION   = yes                    switch     %-23b
#K DATAMIN    = -30.                   counts     %-23.7g
#K DATAMAX    = 30000.                 counts     %-23.7g
#K EXPOSURE   = CEXPTIME               keyword    %-23s
#K AIRMASS    = airmass                keyword    %-23s
#K FILTER     = FILTERS                keyword    %-23s
#K OBSTIME    = UTMIDDLE               keyword    %-23s
#
#K NOISE      = poisson                model      %-23s
#K THRESHOLD  = 500.                   counts     %-23.7g
#K CTHRESHOLD = 0.                     counts     %-23.7g
#K SIGMA      = INDEF                  counts     %-23.7g
#K GAIN       = ""                     keyword    %-23s
#K EPADU      = 1.1                    e-/adu     %-23.7g
#K CCDREAD    = ""                     keyword    %-23s
#K READNOISE  = 10.                    e-         %-23.7g
#
#K IMAGE      = obj002.imh             imagename  %-23s
#K FWHMPSF    = 4.2                    scaleunit  %-23.7g
#K NSIGMA     = 1.5                    sigma      %-23.7g
#K RATIO      = 1.                     number     %-23.7g
#K THETA      = 0.                     degrees    %-23.7g
#
#K SHARPLO    = 0.2                    number     %-23.7g
#K SHARPHI    = 1.                     number     %-23.7g
#K ROUNDLO    = -1.                    number     %-23.7g
#K ROUNDHI    = 1.                     number     %-23.7g
#
#N XCENTER  YCENTER  MAG     SHARPNESS  ROUNDNESS  ID

#U pixels   pixels   #       #          #          #
#F %-12.2f  %-9.2f   %-9.3f  %-12.3f    %-12.3f    %-6d
#
   306.94   334.02   -3.147  0.367      -0.058     1
   390.02   546.96   -1.000  0.351      -0.031     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**.

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 **obj279.mag.1** and so on. If we had included the **.imh** extension would would now be getting **obj279.imh.mag.1** names.

At this point you should get a flashing circle on your **imtool** window. 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, each with names like *image*.mag.1. It will help if you keep track of the order in which you have measured your standard stars so that you can easily asign them the appropriate names later on.

3.5 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 (**obj279.mag.1**, **obj280.mag.1** ...). 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)

```
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
Minimum good data value (-30.) (CR or value):
        New minimum good data value: -30. counts
Maximum good data value (32000.) (CR or value):

obj001.imh x: 304.01 y: 334.64 s: 28.46 m: 13.206 e: ok
obj002.imh x: 306.98 y: 334.11 s: 7.66 m: 10.771 e: ok
obj003.imh x: 300.97 y: 334.47 s: 16.96 m: INDEF e: err
obj004.imh x: 340.70 y: 486.29 s: 13.05 m: 11.786 e: ok
obj005.imh x: 342.30 y: 482.71 s: 5.14 m: 11.623 e: ok
obj006.imh x: 340.81 y: 480.49 s: 9.55 m: 12.152 e: ok
obj007.imh x: 413.60 y: 390.61 s: 8.69 m: 12.096 e: ok
obj008.imh x: 413.35 y: 387.60 s: 5.81 m: 11.495 e: ok
obj009.imh x: 409.42 y: 390.32 s: 10.48 m: 11.590 e: ok
obj225.imh x: 303.40 y: 472.97 s: 9.52 m: 12.090 e: ok
obj226.imh x: 309.15 y: 474.69 s: 5.65 m: 11.590 e: ok
obj227.imh x: 307.64 y: 477.24 s: 13.73 m: 11.754 e: ok
obj228.imh x: 389.62 y: 381.63 s: 14.18 m: 11.587 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.

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

```
#K IRAF       = NOAO/IRAFV2.10BETA      version    %-23s
#K USER       = massey                  name       %-23s
(more)
#K PACKAGE    = apphot                  name       %-23s
#K TASK       = phot                    name       %-23s
#
#K SCALE      = 1.                      units      %-23.7g
#K FWHMPSF    = 4.                      scaleunit  %-23.7g
#K EMISSION   = yes                     switch     %-23b
#K DATAMIN    = -30.                    counts     %-23.7g
#K DATAMAX    = 32000.                  counts     %-23.7g
#K EXPOSURE   = CEXPTIME                keyword    %-23s
#K AIRMASS    = airmass                 keyword    %-23s
#K FILTER     = FILTERS                 keyword    %-23s
#K OBSTIME    = UTMIDDLE                keyword    %-23s
#
#K NOISE      = poisson                 model      %-23s
#K THRESHOLD  = 50.                     counts     %-23.7g
#K CTHRESHOLD = 0.                      counts     %-23.7g
#K SIGMA      = INDEF                   counts     %-23.7g
#K GAIN       = ""                      keyword    %-23s
#K EPADU      = 1.1                     e-/adu     %-23.7g
#K CCDREAD    = ""                      keyword    %-23s
#K READNOISE  = 10.                     e-         %-23.7g
#
#K CALGORITHM = centroid                algorithm  %-23s
#K CBOXWIDTH  = 8.                      scaleunit  %-23.7g
#K MAXSHIFT   = 1.                      scaleunit  %-23.7g
(more)
#K KCLEAN     = 3.                      sigma      %-23.7g
#
#K SALGORITHM = mode                    algorithm  %-23s
#K ANNULUS    = 20.                     scaleunit  %-23.7g
#K DANNULUS   = 5.                      scaleunit  %-23.7g
#K SMAXITER   = 10                      number     %-23d
#K SKREJECT   = 3.                      sigma      %-23.7g
#K SNREJECT   = 50                      number     %-23d
#K KHIST      = 3.                      sigma      %-23.7g
 (more)
#K SKYVALUE   = 0.                      counts     %-23.7g
#
#K WEIGHTING  = constant                model      %-23s
#K APERTURES  = 15.                     scaleunit  %-23s
#K ZMAG       = 25.                     zeropoint  %-23.7g
#
#N IMAGE               XINIT     YINIT     ID    COORDS                LID    \
#U imagename           pixels    pixels    ##    filename              ##     \
#F %-23s               %-10.2f   %-10.2f   %-5d  %-23s                 %-5d
#
#N XCENTER    YCENTER    XSHIFT  YSHIFT  XERR    YERR    CIER CERROR        \
#U pixels     pixels     pixels  pixels  pixels  pixels  ##   cerrors       \
#F %-12.2f    %-11.2f    %-8.2f  %-8.2f  %-8.2f  %-8.2f  %-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                 OTIME              \
#U timeunit         number           name                    timeunit           \
#F %-18.7g          %-15.7g          %-23s                   %-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
#
obj001.imh            303.97    334.75    1     obj001.imh.coo.1      1      \
   304.01   334.64     0.04    -0.11   0.00    0.00    0    NoError         \
   28.46224         8.600574         4.441579         706    2     0    NoError \
   10.004           1.246568         00                      0:02:55.0          \
   15.00    542160.3      707.2919      13.206  0.002  0    NoError
obj002.imh            306.94    334.02    2     obj002.imh.coo.1      1      \
(more)
```

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

(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.

3.6 The Standard Star Solutions

Now that we have instrumental magnitudes for the standard stars, we need to use these to find the transformation equations that will allow us to put our observations on the standard system. The routines for doing this are found in the **photcal** package. There are *many* routes to the same end-point, and the most straight-forward way depends upon what sort of standard observations you made—you would want to do one thing if you have one standard star per field as we do in this example (i.e., isolated Landolt stars), while you would want to do something else if you instead had lots of standard stars in each field (i.e, the "VideoCam M92 field"). We will follow the one path through this section, but try to point out other choices you may wish to make; there is also excellent on-line help available by typing **help pcintro**.

The steps we need to go through for finding the transformation equations go something like this:

1. Create a catalog that contains the magnitudes and color indices of the standard stars on the standard system. A suitable catalog of the Landolt *UBVRI* standards (i.e., Landolt 1983 *Astronom Journ* **88**, 439) is already available, and other "standard" standards catalogs may come on-line in the future; otherwise, the user will have to create his/her own.

2. Create a "standard star observations" file that consists of the airmass, instrumental magnitudes and errors for each set of observations (u,b, and v, say).

3. Create a file containing the algebraic form of the transformation equations, and references to which columns in the tables contain which information. A sample template is available for making transformations using the Landolt catalog.

4. Fit the transformation equations, throwing out points to suit your fancy. This step is done interactively, with lots of cool graphics, and should find it deeply satisfying.

After you have reduced your program stars, you will need to then create a "program star observations file" that consists of the relevant instrumental magnitudes, airmasses, etc for each set of observations, and finally you will then apply the transformations to these data. Details of this will will given in the final section (Sec. 5).

3.7 Making the Standard Star Catalog

The standard star catalog consists of the the magnitudes and indices for the standard stars on the standard system; i.e., what you might type in directly from one of Landolt's papers. If you are using Landolt (1983) *UBVRI* standards you do not have to do anything; this catalog already is available on-line in the correct format. Other catalogs may also be available; do a **page photcal$catalogs/README** to see what else may be there. Any other catalog can be created by the user by simply running **mkcatalog**; first read the "help" page for this task. Note that there are actually two files associated with each "catalog"—the data itself, and a file describing the format of this catalog.

3.8 Making the Standard Star Observations File

We need to create a file containing the instrumental magnitudes for our standard stars—as in the case of the standard star catalog, there will actually be two files, a file containing the data, and a file containing the format of the file.

In the example we are working through here, there is exactly one standard star observation per file, and so it is pretty easy to determine what "u" value goes with what "b" value. (Following normal nomenclature, we are designating magnitudes on the standard system with capital letters (i.e., *UBV*) and magnitudes on the instrumental system as lower case letters (i.e., *ubv*.)

We start out by using the editor to create a file showing what image names define a set. If we were to list all our standard star files, we might do this a **imhead @stands** and find a list like that shown in Fig. 16

We need to use the editor to create a file resembling that of Fig. 17, that contain matched observations.

In this example there are several points of interest. (1) The image names have to be given with the ".imh" extension, because this is how the files happen to be listed in our photometry file *standstuff*. (See Fig. 15.) This silliness traces back to the original way we ran **daofind** on the list created by **files**. (2) In listing the files, we ignored the order within any set; a comparison between Figs. 17 and 16 shows that the order is sometimes *u,b,v* and sometimes *v,b,u*. Note also that there was no *v* exposure of for standard star 95_132 (we realized at the telescope that a 1 sec exposure in *V* saturated this star, and so we kept only the *U* and *B* images). (3) We have labeled each set with the standard star name from the on-line Landolt catalog; note the prevalence of the underscore (_) character.

We are now ready to create the file of standard star observations using **mknobsfile** (note the "n"; **mkobsfile** [without the "n"] does things a little differently). Edit the parameters for **mknobsfile** so they resemble that of Fig. 18.

Running mknobsfile will then produce output that resembles that of Fig. 19.

All **mknobsfile** did was create a file (*standobs* in this example) that contains the standard star observations in some "matched" way. We see immediately that there is *some* problem—it found two stanards in the last set of observations of F_16 rather than the single one we had. Let's look at the contents of the file created by **mknobsfile**, *standobs* (Fig. 20).

We see that the problem was that the *U, B* and *V* exposures of the last F_16 observation were not well lined up: we had a matching tolerance of 10 pixels in Fig. 18 and although the *B* and *V* exposures were that close together, the *U* exposure was not well aligned. If we had

```
ph> imhead @stands
obj001.imh[797,797][real]: G93-48 U
obj002.imh[797,797][real]: B G93-48
obj003.imh[797,797][real]: V G93-48
obj004.imh[797,797][real]: GD246 V
obj005.imh[797,797][real]: B GD246
obj006.imh[797,797][real]: U GD246
obj007.imh[797,797][real]: F108 U
obj008.imh[797,797][real]: F108 B
obj009.imh[797,797][real]: V F108
obj225.imh[797,797][real]: G246 U
obj226.imh[797,797][real]: B G246
obj227.imh[797,797][real]: G246 V
obj228.imh[797,797][real]: F108
obj229.imh[797,797][real]: B F108
obj230.imh[797,797][real]: F108
obj231.imh[797,797][real]: F16 U
obj232.imh[797,797][real]: F16 B
obj233.imh[797,797][real]: F16 V
obj234.imh[797,797][real]: F22 V
obj235.imh[797,797][real]: B F22
obj236.imh[797,797][real]: U F22
obj237.imh[797,797][real]: U F24
obj238.imh[797,797][real]: F24 B
obj239.imh[797,797][real]: F24 V
obj240.imh[797,797][real]: 95-132 U (only?)
obj241.imh[797,797][real]: B 95-132
obj279.imh[797,797][real]: GD71 V (one of these is bound to be it...)
obj280.imh[797,797][real]: GD71
obj281.imh[797,797][real]: U GD71
obj282.imh[797,797][real]: U F22
obj283.imh[797,797][real]: B F22
obj284.imh[797,797][real]: V F22
obj285.imh[797,797][real]: V F16
obj286.imh[797,797][real]: B f16
obj287.imh[797,797][real]: U F16
```

Figure 16: The images containing standard star observations.

```
edit standstars
G93_48 : obj001.imh obj002.imh obj003.imh
GD_246 : obj004.imh obj005.imh obj006.imh
F_108 : obj007.imh obj008.imh obj009.imh
GD_246 : obj225.imh obj226.imh obj227.imh
F_108 : obj228.imh obj229.imh obj230.imh
F_16 : obj231.imh obj232.imh obj233.imh
F_22 : obj234.imh obj235.imh obj236.imh
F_24 : obj237.imh obj238.imh obj239.imh
95_132 : obj240.imh obj241.imh
GD_71 : obj279.imh obj280.imh obj281.imh
F_22 : obj282.imh obj283.imh obj284.imh
F_16 : obj285.imh obj286.imh obj287.imh
```

Figure 17: An example of matching observations of standard stars. Note in this example the form of the standard star names; this is how the stars are labeled in the on-line Landolt catalog. The file shown above *standstar* was simply created with the editor.

```
                                   I R A F
                    Image Reduction and Analysis Facility
PACKAGE = photcal
   TASK = mknobsfile

photfile=           standstuff  The input list of APPHOT/DAOPHOT databases
idfilter=             00,10,20  The list of filter ids
imsets  =           standstars  The input image set file
observat=             standobs  The output observations file
(obspara=                     ) The input observing parameters file
(obscolu=                2,3,4) The format of obsparams
(shifts =                     ) The input x and y coordinate shifts file
(apercor=                     ) The input aperture corrections file
(apertur=                    1) The aperture number of the extracted magnitude
(toleran=                  10.) The tolerance in pixels for position matching
(allfilt=                   no) Output only objects matched in all filters
(verify =                   no) Verify interactive user input ?
(verbose=                  yes) Print status, warning and error messages ?
(mode   =                   ql)
```

Figure 18: The parameter file for **mknobsfile**. The file *standstuff* is the file containing all the instrumental magnitudes; if instead you followed the instructions in Sec. 3.4.2 above and have multiple files named obj223.mag.1, etc., then substitute *mag* here. The filter list should be in some logical order (such as "u", "b", "v"); we realized what the filters were called by inspecting our photometry file(s). The "imsets" file is the file created that have the matched set of observations.

```
ph> mknobsfile
The input list of APPHOT/DAOPHOT databases (standstuff):
The list of filter ids (00,10,20):
The input image set file (standstars):
The output observations file (standobs):
Warning: Image set 9 name  95_132 is incomplete

Observations file: standobs
        Image set: G93_48  1 stars written to the observations file
        Image set: GD_246  1 stars written to the observations file
        Image set: F_108  1 stars written to the observations file
        Image set: GD_246  1 stars written to the observations file
        Image set: F_108  1 stars written to the observations file
        Image set: F_16  1 stars written to the observations file
        Image set: F_22  1 stars written to the observations file
        Image set: F_24  1 stars written to the observations file
        Image set: 95_132  1 stars written to the observations file
        Image set: GD_71  1 stars written to the observations file
        Image set: F_22  1 stars written to the observations file
        Image set: F_16  2 stars written to the observations file
```

Figure 19: Running the **mknobsfile** routine.

set a matching tolerance of 15 this would have worked. Instead, we now simply have to use the editor to delete the three "INDEF" values shown for F_16.

This problem does suggest how we would treat data for which our fields contained several standard stars, rather than just one each. In that case we would have been better off labeling our image sets (Fig. 17 not with the name of some individual standard star, but with something like the field name: "SA95", say, or "M92". Running **mknobsfile** would then have created names like "SA95-1", "SA95-2", "SA95-3" for each star it found within the photometry file corresponding to the images listed. At this point you would have to use the editor to edit in the "real" name of each standard star so that the correct standard indices could be found in the catalog during the next step, solving the transformation equations. We might also have wanted to decrease the matching tolerance to something like 5 pixels.

As well as this output file *standobs*, we also generated a second file called *fstandobs.dat* when we ran **mknobsfile**. This second file is a "format" file and describes the format of the observations file. The contents of this format file are needed in setting up the transformation equations; the contents are shown in Fig. 21

3.9 Defining the Transformations

The first step in solving the transformation equations is to generate a text file (called a "configuration file" in PHOTCAL-ese) containing (a) descriptions of the "catalog" (basically what columns in what file contain the standard star indices), (b) descriptions of the "observations" (what columns in what file contain the standard star airmasses, instrumental magnitudes and errors, and (c) the algebraic form of the transformation equations. If you are dealing with

```
# FIELD             FILTER     AIRMASS XCENTER  YCENTER  MAG        MERR

G93_48              00         1.247   304.01   334.64   13.206     0.002
*                   10         1.242   306.98   334.11   10.771     0.002
*                   20         1.238   300.97   334.47   INDEF      INDEF
GD_246              00         1.606   340.81   480.49   12.152     0.002
*                   10         1.625   342.30   482.71   11.623     0.002
*                   20         1.638   340.70   486.29   11.786     0.003
F_108               00         1.339   413.60   390.61   12.096     0.002
*                   10         1.329   413.35   387.60   11.495     0.002
*                   20         1.322   409.42   390.32   11.590     0.003
GD_246              00         1.542   303.40   472.97   12.090     0.002
*                   10         1.551   309.15   474.69   11.590     0.002
*                   20         1.558   307.64   477.24   11.754     0.003
F_108               00         1.311   386.50   381.62   12.050     0.002
*                   10         1.306   389.59   380.58   11.471     0.002
*                   20         1.302   389.62   381.63   11.587     0.003
F_16                00         1.100   414.61   393.30   12.624     0.003
*                   10         1.097   412.59   391.48   11.093     0.002
*                   20         1.096   411.01   392.24   11.015     0.002
F_22                00         1.268   447.01   413.95   12.298     0.002
*                   10         1.271   448.92   414.32   11.489     0.002
*                   20         1.273   450.04   417.19   11.428     0.002
F_24                00         1.246   398.57   408.95   11.367     0.001
*                   10         1.243   400.12   407.36   10.930     0.002
*                   20         1.241   394.83   408.86   INDEF      INDEF
95_132              00         1.376   445.91   422.60   13.173     0.007
*                   10         1.366   444.60   420.62   11.255     0.002
*                   20         INDEF   INDEF    INDEF    INDEF      INDEF
GD_71               00         1.440   335.29   527.67   12.116     0.002
*                   10         1.440   341.35   528.02   11.562     0.003
*                   20         1.440   340.05   530.37   11.669     0.003
F_22                00         2.063   377.16   372.50   12.721     0.003
*                   20         2.106   379.72   372.40   11.538     0.004
*                   10         2.083   383.03   371.85   11.701     0.004
F_16                00         2.217   331.97   355.77   13.193     0.007
*                   10         INDEF   INDEF    INDEF    INDEF      INDEF
*                   20         INDEF   INDEF    INDEF    INDEF      INDEF
F_16-2              00         INDEF   INDEF    INDEF    INDEF      INDEF
*                   10         2.192   341.91   358.41   11.375     0.003
*                   20         2.167   341.67   362.88   11.172     0.003
```

Figure 20: The output file from **mknobsfile** before correction.

```
# Declare the observations file variables

observations

X00         3           # airmass in filter 00
x00         4           # x coordinate in filter 00
y00         5           # y coordinate in filter 00
m00         6           # instrumental magnitude in filter 00
error(m00)  7           # magnitude error in filter 00

X10         9           # airmass in filter 10
x10         10          # x coordinate in filter 10
y10         11          # y coordinate in filter 10
m10         12          # instrumental magnitude in filter 10
error(m10)  13          # magnitude error in filter 10

X20         15          # airmass in filter 20
x20         16          # x coordinate in filter 20
y20         17          # y coordinate in filter 20
m20         18          # instrumental magnitude in filter 20
error(m20)  19          # magnitude error in filter 20
```

Figure 21: The contents of the "format" file *fstandobs.dat* corresponding to that file *standobs*.

simple *UBVRI* stuff then most of this is handled for you, and you merely need to do a little editing; in the worse case, you can do a **help config** and attempt to figure out the syntax of this very versatile task.

In our case we wish to solve transformation equations that are of this form:

$$u = U + Const + Color\ Term \times (U - B) + Extinction \times Airmass$$

$$b = B + Const + Color\ Term \times (B - V) + Extinction \times Airmass$$

$$v = V + Const + Color\ Term \times (B - V) + Extinction \times Airmass$$

Note that the instrumental magnitudes are on the left, and the standard magnitudes and indices are on the right; we are after the "best" values (in a least-squares sense) of each zero-point constant "Const", the color term coefficient, and the extinction coefficient. A few wrinkles to keep in mind: the airmasses are a little bit different for the "u", "b", and "v" exposures, and as you will infer from Fig. 21 are called "X00", "X10", and "X20" in this example. The instrumental magnitudes are similarly going to be "m00", "m10", and "m20". Also, the Landolt catalog does not explicitly contain the the standard *U* magnitude; instead, it contains *V*, *B-V*, and *U-B*, and it is the sum of these three that will give the standard *U*. Finally, formating considerations result in in the indices *U-B* being listed not as *U-B* but as *UB*. Yes, this is confusing, but apparently it cannot be helped.

Why do we choose to solve the equations in this form? Most photoelectric photometrists are used to equations written in terms of the color indices themselves (rather than *U* or *B*), and with the order inverted than that shown above, i.e., with the standard indices on the left and the instrumental magnitudes on the right. Why do it differently with CCD data?

```
                                 I R A F
                   Image Reduction and Analysis Facility
PACKAGE = photcal
   TASK = mkconfig

config  =            ctio.cfg  The new configuration file
catalog =             landolt  The source of the catalog format specification
observat=            standobs  The source of the observations file format
transfor=             landolt  The source of the transformation equations
(templat=                   ) An existing template configuration file
(catdir =           )_.catdir) The standard star catalog directory
(verify =                  no) Verify each new entry
(edit   =                 yes) Edit the new configuration file
(check  =                 yes) Check the configuration file
(verbose=                 yes) Verbose output
(mode   =                  ql)
```

Figure 22: The **mkconfig** task.

The answer is that while photoelectric observers were usually able to cycle through different filters quite rapidly, CCD observers can't—sometimes we even have to put up with 4-minute read-down times. Similarly, *UBVRI* photoelectric observations are often rapid enough that variations in transparency were circumvented. In addition, centering errors (from which analysis of CCD data is essentially immune!) might be similar in all filters if one observed rapidly enough. These, and other factors, resulted in the *colors* usually being better determined than the individual magnitudes. With CCD data the measurements through each filter are much more "independent" than with photoelectric photometry. A single airmass usually suffices for all the filters in photoelectric *UBVRI* work, while in extreme cases CCD exposures through each filters might be made on separate nights!

Edit the task **mkconfig** until it resembles that of Fig. 22, substituting in whatever name you would like for the configuration file name (*ctio.cfg* in this example) and making sure that the "observation" file corresponds to whatever you generated with **mknobsfile**. Running the task will generate the file *ctio.cfg* and immediately dump you into the editor to give you a chance to modify the file. The unmodified file appears in Fig. 23.

So far so good! The first part of the file (labeled "catalog") is simply the description of the standard star catalog; i.e., what the variables are called and what columns they are in. The second part of the file (labeled "observations") is simply the content of the "format file" shown in Fig. 21. The third part of the file ("transformation") is a set of sample transformation equations. The coefficients that we will solve for are called "u1", "u2", and so forth, and reasonable initial values" are supplied. The coefficient under "fit" in each equation will be solved for, while those listed under "const" will be left alone, but you can change this interactively when you do the fit. Note that second-order color terms are explicitly included in these equations (u4 * UB * XU) but are implicitly set to zero by setting "u4 = const = 0.0".

We must (a) get rid of the two equations we don't want (we are not fitting R or I), and

```
#  Declare the Landolt UBVRI standards catalog variables

catalog

V       4               # the V magnitude
BV      5               # the (B-V) color
UB      6               # the (U-B) color
VR      7               # the (V-R) color
RI      8               # the (R-I) color
VI      9               # the (V-I) color

error(V)  12            # the V magnitude error
error(BV) 13            # the (B-V) color error
error(UB) 14            # the (U-B) color error
error(VR) 15            # the (V-R) color error
error(RI) 16            # the (R-I) color error
error(VI) 17            # the (V-I) color error
# Declare the observations file variables

observations

X00           3              # airmass in filter 00
x00           4              # x coordinate in filter 00
y00           5              # y coordinate in filter 00
m00           6              # instrumental magnitude in filter 00
error(m00)    7              # magnitude error in filter 00

X10           9              # airmass in filter 10
x10           10             # x coordinate in filter 10
y10           11             # y coordinate in filter 10
m10           12             # instrumental magnitude in filter 10
error(m10)    13             # magnitude error in filter 10

X20           15             # airmass in filter 20
x20           16             # x coordinate in filter 20
y20           17             # y coordinate in filter 20
m20           18             # instrumental magnitude in filter 20
error(m20)    19             # magnitude error in filter 20
# Sample transformation section for the Landolt UBVRI system

transformation

fit   u1=0.0, u2=0.65, u3=0.000
const u4=0.0
UFIT : mU = (UB + BV + V) + u1 + u2 * XU + u3 * UB + u4 * UB * XU

fit   b1=0.0, b2=0.35, b3=0.000
const b4=0.0
BFIT : mB = (BV + V) + b1 + b2 * XB + b3 * BV + b4 * BV * XB

fit   v1=0.0, v2=0.17, v3=0.000
const v4=0.0
VFIT : mV = V + v1 + v2 * XV + v3 * BV + v4 * BV * XV

fit   r1=0.0, r2=0.08, r3=0.000
const r4=0.0
RFIT : mR = (V - VR)  + r1 + r2 * XR + r3 * VR + r4 * VR * XR
(more)
```

Figure 23: Running **mkconfig** dumps you in the editor confronting this file.

```
transformation

fit   u1=0.0, u2=0.65, u3=0.000
const u4=0.0
UFIT : m00 = (UB + BV + V) + u1 + u2 * X00 + u3 * UB + u4 * UB * X00

fit   b1=0.0, b2=0.35, b3=0.000
const b4=0.0
BFIT : m10 = (BV + V) + b1 + b2 * X10 + b3 * BV + b4 * BV * X10

fit   v1=0.0, v2=0.17, v3=0.000
const v4=0.0
VFIT : m20 = V + v1 + v2 * X20 + v3 * BV + v4 * BV * X20
```

Figure 24: The edited form of the transformation equations.

(b) we must change the name of the observational variables "mU" "XU", "mB","XB",'mV", and "XV" to be "m00","X00", and so on, as listed in the "observations" section. When we are done, the "transformations" section should resemble that of Fig. 24.

When you are done, exit the editor in the normal way (**:wq** for me). You will hopefully see a listing showing you the number of catalog variables, observational variables, and "parameters" (read "coefficients"), along with with a written guarantee that there is no funny stuff: "Warnings = 0" and "Errors = 0". After this, you are ready for a real treat—the interactive transformation equation solver!

3.10 Solving Those Transformation Equations

The task that actually solves the transformation equations is called **fitparams**. Edit the parameter file of this task until it resembles that of Fig. 25. Upon running this task, you will be confronted with a plot like that of Fig. 26.

Clearly one point is just plain crazy: however, because we set *nreject* to some positive value, **fitparams** was smart enough to figure this out. We could simple delete this point by hitting "d" and following with an "f" for a new fit, but there is no need to, either. We can easily rescale the plot: type a **w** (for "window") followed by a "y" to expand the y-axis. Do it a few more times to really be able to see those points. What are the values of the coefficients, and what are the individual residuals like? You can type an ":vshow" in the graphics window, and get something that resembles that of of Fig. 27.

Of the remaining error, how much of it is due to non-linear color terms? We can plot the residuals against *any* variable by hitting a "g". It will ask us what "graph key [is] to be defined, to which we can answer "k" We will then be asked to "Set graph axis types". We could answer at this point "UB, residuals" (or "X00, residuals", or anything else you might want to see). Then type a "k" to see the plot. Does there appear to be something systematic with *U-B* color not allowed for in the linear term? If the points simply scatter, as they do in Fig. 28 then leave well-enough alone!

```
                                  I R A F
                    Image Reduction and Analysis Facility
PACKAGE = photcal
   TASK = fitparams

observat=            standobs  List of observations files
catalogs=             landolt  List of standard catalog files
config  =            ctio.cfg  Configuration file
paramete=            ctio.ans  Output parameters file
(weighti=         photometric) Weighting type (uniform,photometric,equations)
(addscat=                 yes) Add a scatter term to the weights ?
(toleran=   3.0000000000000E-5) Fit convergence tolerance
(maxiter=                  15) Maximum number of fit iterations
(nreject=                   2) Number of rejection iterations
(low_rej=                  3.) Low sigma rejection factor
(high_re=                  3.) High sigma rejection factor
(grow   =                  0.) Rejection growing radius
(interac=                 yes) Solve fit interactively ?
(logfile=             testlog) Output log file
(log_unm=                 yes) Log any unmatched stars ?
(log_fit=                 yes) Log the fit parameters and statistics ?
(log_res=                 yes) Log the results ?
(catdir =           )_.catdir) The standard star catalog directory
(graphic=            stdgraph) Output graphics device
(cursor =                    ) Graphics cursor input
(mode   =                  ql)
```

Figure 25: The parameter set for the first pass through **fitparams**.

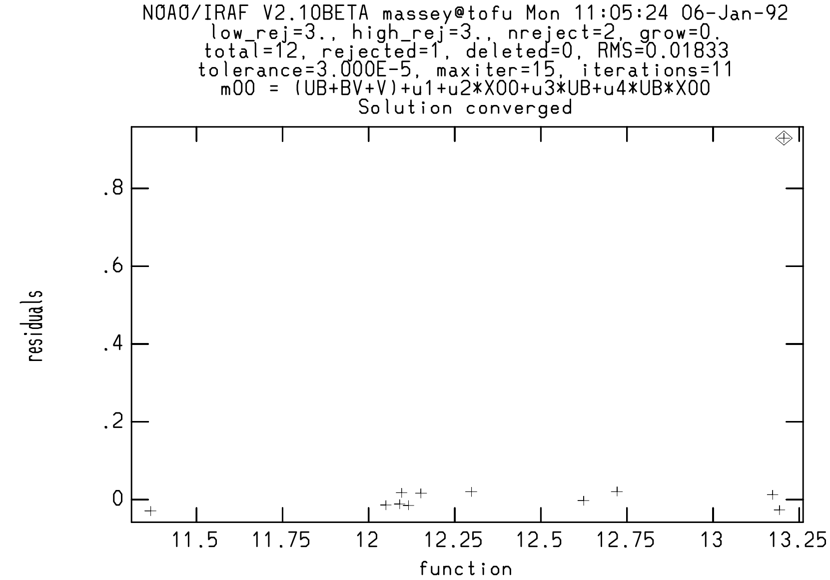

Figure 26: The residuals as a function of the fit (U mag). Note that the one wildly discrepant point at (13.2, .7) was automatically rejected.

Are you curious about whether we needed that color times airmass term was significant or not? Remember, we set "u4=0.0" and left it a constant. We could at this point see the effect of including it by typing **:fit u4 0.0**. Doing this lowers the RMS from 0.018 to 0.014. Well... we're not convinced this is significant (a **:vshow** reveals that the value of u4 is -0.07 ± 0.13), and so we decide to change it back to a constant by doing a **:const u4 0.0**. When we are happy we type a "q" to go on to the next equation.

Occasionally you will find that the solution fails to converge—you'll notice this because (a) the "residuals" plots will be blank, (b) the RMS will be "INDEF" at the top of the plot, and (c) the number of iterations will equal the maximum number of iterations (maxiter). Simply change the tolerance from its default value of 3.000e-5 to something just a mite large... 7e-5, say. You can do this by a **:tolerance 7e-5**. Do a new fit (**f**) and all is likely to be well.

When we are all done we can page through the log file we specified (see Fig. 25. In this case we find that all three observations of the star G93_48 are bad—tenths and tenths of magnitudes off. This must have been the wrong star! Looking back at the image and the finding chart reveals the telescope was pointed about half a degree too far north! Oh, well, it *was* the start of a run!

A summary of useful **fitparams** commands follows:

? - Print options (these and many more!)

d - Delete point nearest cursor

```
low_reject    3.
high_reject   3.
nreject       2
grow          0.
tol           3.000000E-5
maxiter       15


niterations        11
total_points       12
rejected           1
deleted            0
standard deviation 0.02149688
reduced chi          1.001827
average error      0.02145767
average scatter    0.01743245
RMS                 0.0183326

parameter           value           error
u1              -0.3669052       0.0340730    (fit)
u2               0.5310039       0.0204199    (fit)
u3              -0.0544222       0.0128918    (fit)
u4               0.0000000       0.0000000    (constant)


#      objectid       function           fit       residuals         sigma
         G93_48          INDEF         INDEF           INDEF         INDEF
         GD_246         12.152      12.13554      0.01646042    0.01897077
          F_108         12.096      12.07869      0.01731205    0.01897077
         GD_246          12.09      12.10156     -0.01155567    0.01897077
          F_108          12.05      12.06382     -0.01381969    0.01897077
           F_16         12.624      12.62638    -0.002383232    0.02264708
           F_22         12.298      12.27782      0.02018356    0.02099739
           F_24         11.367      11.39651     -0.02950859    0.02273522
         95_132         13.173      13.16032       0.0126791    0.02477681
          GD_71         12.116      12.13055     -0.01455021    0.02310606
           F_22         12.721      12.69996      0.02103519    0.02111612
           F_16         13.193      13.21951     -0.02651405    0.02351362
```

Figure 27: The results of doing a **:vshow** while in **fitparams**. Note that the rejected point must be G93_48. The RMS is about 0.02 mag, the value of chi is nearly 1., and all is therefore right witht eh world. The values of the fitting coefficients, and their errors, are also shown.

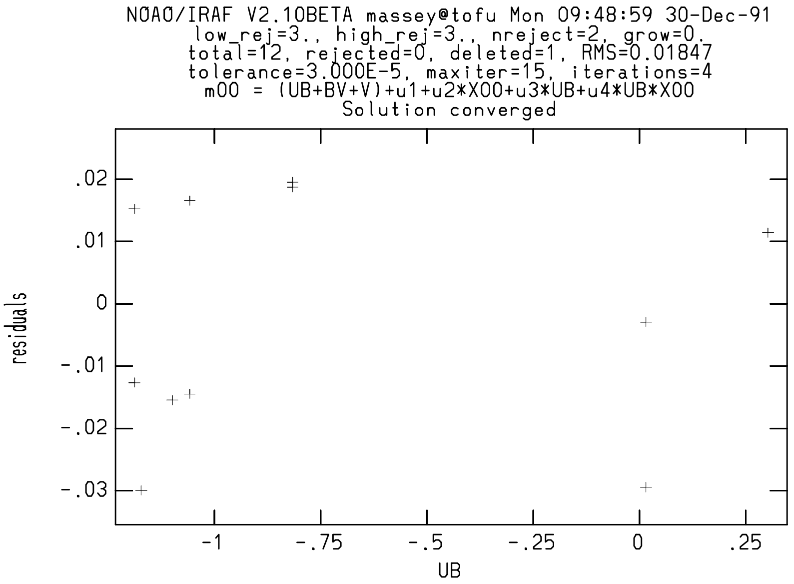

Figure 28: No second-order color term appears to be present in these data.

u - Undelete point nearest cursor

f - Do the fit again and resraw or overplot the graph

g - Redefine some key as a graph key, with user-specified axis

i - Plot residuals versus fit

w - Window the plot (follow the "w" with a "?" to see options

q - Quit, and go onto the next equation

:vshow - Show the error and results of the fit

:fit param [value] Change a constant coefficient to a fitted coefficient with a specified initial value.

:const param [value] Change a fitted coefficient to a constant with a specified value.

:tolerance [value] Change the value of tolerance (useful if the solution fails to converge).

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 wrote the first version of the PHOTCAL routines used in determining the transformations from the instrumental to standard system, and has reamined working on this package collaboratively with Lindsey Davis.

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 **pappend** 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 of 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 Sec. 2.2 we discussed how to deal with parameter files, and in Sec. 3.3 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 29. If you have not yet corrected your headers (see Sec. 2.1), now's the time. The things to note here are (a) what the filter keyword is (we can see from Figure 29 that the answer is FILTERS, (b) what the effective exposure time keyword is (CEXPTIME in this example), and (c) what the effective airmass keyword is (AIRMASS in this example). These latter were added or fixed by the procedure described in Sec. 2.1.

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

```
obj211[797,797][real]: Short B field 1
    No bad pixels, no histogram, min=-93.55627, max=25415.95
    Line storage mode, physdim [896,797], length of user area 1580 s.u.
    Created Fri 14:10:04 06-Dec-91, Last modified Fri 14:10:04 06-Dec-91
    Pixel file 'tofu!/data1/massey/pixels/obj211.pix' [ok]
    'KPNO-IRAF'  /
    '06-04-89'  /
    'KPNO-IRAF'  /
    '12-10-88'  /
    New copy of obj211.imh
  (more)
    OBSERVAT= 'CTIO              ' / ORIGIN OF DATA
    CCDPICNO=                  211  / ORIGINAL CCD PICTURE NUMBER
    EXPTIME =                   30  / ACTUAL INTEGRATION TIME (SECONDS)
    DARKTIME=                   30  / TOTAL ELAPSED TIME (SECONDS)
    OTIME   =                   30  / SHUTTER OPEN TIME (SECONDS)
    IMAGETYP= 'OBJECT            ' / OBJECT,DARK,BIAS,ETC.
    DATE-OBS= '12/10/88          ' / DATE (DD/MM/YY) OF OBSERVATION
    RA      = ' 0:52:30.00'  /  right ascension
    DEC     = '-37:50:42.00'  /  declination
    EPOCH   =                1950.  / EPOCH OF RA AND DEC
    UT      = ' 3:11:18.00'  /  universal time
    ST      = '23:51:45.00'  /  sidereal time
    AIRMASS =             1.034776 / AIRMASS
    DETECTOR= 'TI1               ' / DETECTOR (CCD TYPE, PHOTON COUNTER, ETC)
    CAMTEMP =                   0.  / CAMERA TEMPERATURE, DEG C
    DEWTEMP =                   0.  / DEWAR TEMPERATURE, DEG C
    FILTERS = '1 0               ' / FILTER BOLT POSITIONS
    TILTPOS =                  234  / TILT POSITION
    TELFOCUS=                 2282  / TELESCOPE FOCUS
  (more)
    CEXPTIME=               30.004
    UTMIDDLE= '3:11:33.0'
```

Figure 29: Header for image obj211

bigger than 23 pixels($23 = 2 \times 11 + 1$) pixels from one side to the other? In our data we find that the "diameter of the biggest star" is something like 27 pixels, and therefore we need to set the PSF radisu to be 13. The FWHM is undoubtably variable from frame to frame, but unless these change by substantial amounts (50%?) using a "typical" value will probably 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 cubic-spline.

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 "read-noise" appropriate for your chip are also entered. 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 a TI chip, and we all know to get nervous when the number of electrons get near 30,000 with this chip. We will leave this at 32000 for now, but keep our eyes open for indications of non-linear behavior—easy enough to see when doing PSF-fitting! **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 30.

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). We will modify "fitrad" to be 4., but otherwise leave the other parmaeters alone for now (Fig. 31). .

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 32 we see that the sky value is roughly 115. In general, if s is the sky value in ADU, p is the number of photons per ADU, and r is

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

(scale  =                   1.) Image scale in units per pixel
(fwhmpsf=                  3.8) FWHM of the PSF in scale units
(emissio=                  yes) Features are positive ?
(sigma  =                INDEF) Standard deviation of background in counts
(datamin=                INDEF) Minimum good data value
(datamax=               32000.) Maximum good data value
(thresho=                   0.) Detection threshold in counts above background
(cthresh=                   0.) Centering threshold in counts above background
(noise  =              poisson) Noise model
(ccdread=                     ) CCD readout noise image header keyword
(gain   =                     ) CCD gain image header keyword
(readnoi=                  10.) CCD readout noise in electrons
(epadu  =                  1.1) Gain in electrons per count
(exposur=             CEXPTIME) Exposure time image header keyword
(airmass=              AIRMASS) Airmass image header keyword
(filter =              FILTERS) Filter image header keyword
(obstime=             UTMIDDLE) Time of observation image header keyword
(itime  =                INDEF) Exposure time
(xairmas=                INDEF) Airmass
(ifilter=                INDEF) Filter
(otime  =                INDEF) Time of observation
(mode   =                   ql)
```

Figure 30: 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 =                  13.) Radius of the psf
(fitrad =                   4.) 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)
(mode   =                   ql)
```

Figure 31: A sample **daopars** is shown.

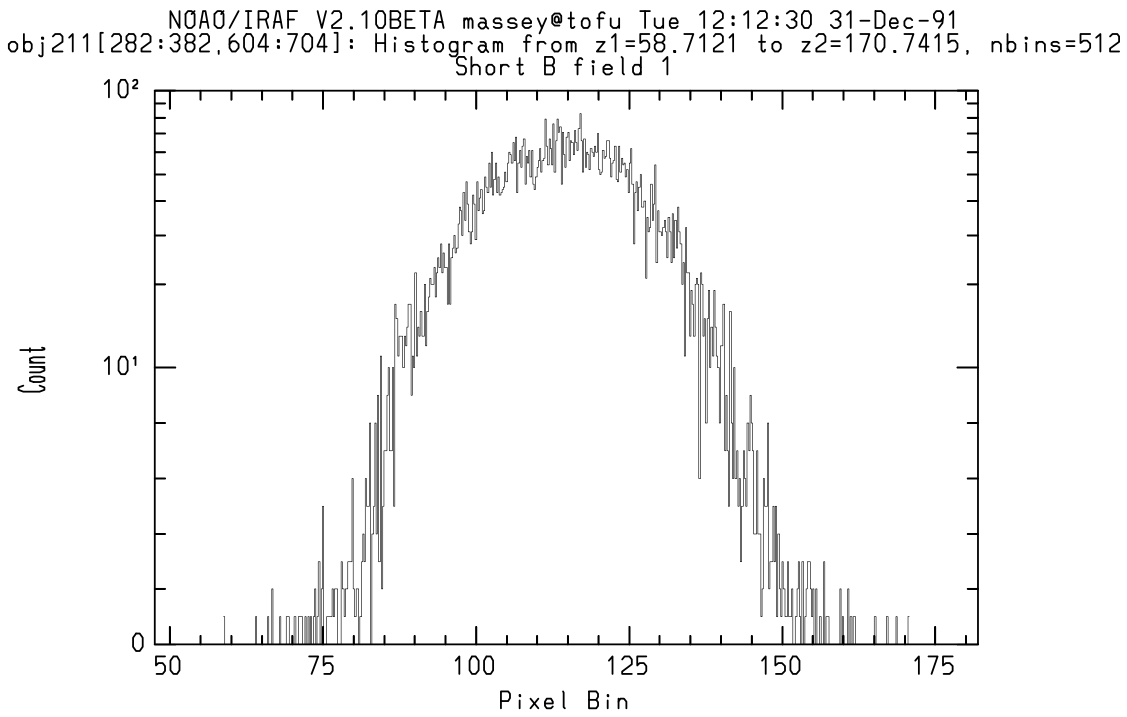

Figure 32: The **imexamine** histogram ("h" key) indicates that the sky value is roughly 115.

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

(scale  =                    1.) Image scale in units per pixel
(fwhmpsf=                   3.8) FWHM of the PSF in scale units
(emissio=                   yes) Features are positive ?
(sigma  =                 INDEF) Standard deviation of background in counts
(datamin=                   80.) Minimum good data value
(datamax=                32000.) Maximum good data value
(thresho=                   50.) Detection threshold in counts above background
(cthresh=                    0.) Centering threshold in counts above background
(noise  =               poisson) Noise model
(ccdread=                      ) CCD readout noise image header keyword
(gain   =                      ) CCD gain image header keyword
(readnoi=                   10.) CCD readout noise in electrons
(epadu  =                   1.1) Gain in electrons per count
(exposur=              CEXPTIME) Exposure time image header keyword
(airmass=               AIRMASS) Airmass image header keyword
(filter =               FILTERS) Filter image header keyword
(obstime=              UTMIDDLE) Time of observation image header keyword
(itime  =                 INDEF) Exposure time
(xairmas=                 INDEF) Airmass
(ifilter=                 INDEF) Filter
(otime  =                 INDEF) Time of observation
(mode   =                    ql)
```

Figure 33: Datapars with **threshold** and **datamin** entered.

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 $1\sigma = \left(\sqrt{115. \times 1.10 + 10^2}\right)/1.10 = 13.7$ 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 the readnoise is still $r \times \sqrt{N}$.

In the example shown here the expected 1σ variation of the sky is 13.7 ADU's; we might therefore want to set our star detection threshold to 3.5 times that amount (roughly 50 in this case). 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 80. This is not used by **daofind** but will be used by all the photometry routines. Figure 33 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 34. Note that although nothing appears to be listed under **datapars** the default name is "datapars"; you could instead have

```
                                                                                          da> lpar daofind
        image = "obj211"        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 = yes)            Print messages in non interactive mode
        (mode = "ql")
da> daofind
Input image (obj211):
Results file (default: image.coo.?) (default):

FWHM of features in scale units (3.8) (CR or value):
        New FWHM of features: 3.8 scale units  3.8 pixels
Detection threshold in counts above background (50.) (CR or value):
        New detection threshold: 50. counts
```

Figure 34: Setting up and running **daofind**.

```
pr> lpar tvmark
        frame = 1               Default frame number for display
       coords = "obj211.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 35: Parameter file for **tvmark**.

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 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 34 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 **obj211.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 35. This will put red dots on top of each star found.

We can see from Figure 36 that **daofind** did a pretty nice job. If we didn't like what we

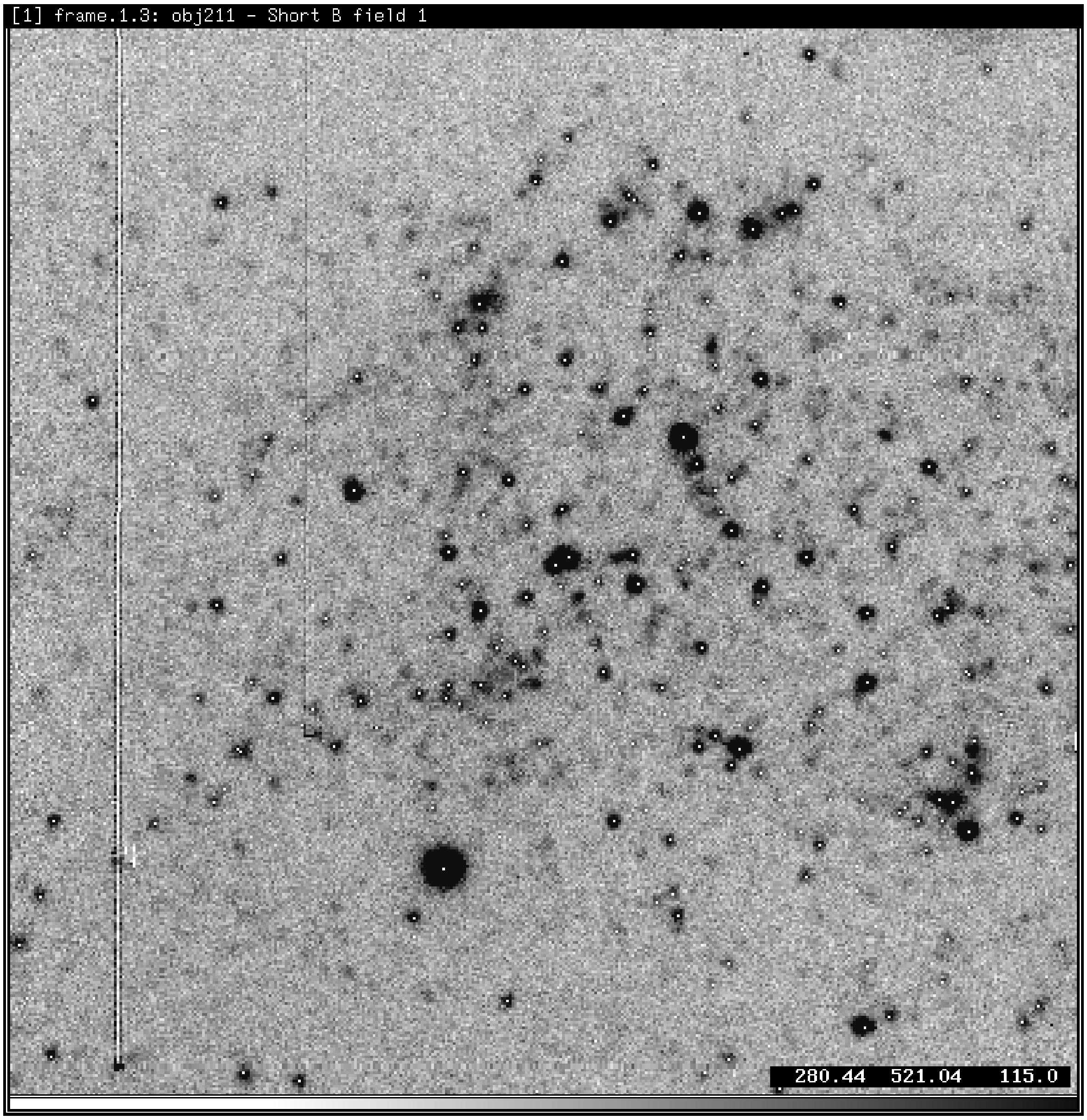

Figure 36: Stars found with **daofind** and marked with **tvmark**.

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 obj211.coo.2, obj211.coo.3, obj211.coo.4, ...If you are using a very slow computer, or a very large chip, 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 cobj211.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 35, 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 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.

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 37. 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 37. Note that the answers were particularly simple: we told it the name of the frame we wished to work

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

image   =              obj211  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 obj211
Coordinate list (default: image.coo.?) (default):
Results file (default: image.mag.?)" (default):

Centering algorithm (none) (CR or value):
        New centering algorithm: none
Sky fitting algorithm (mode) (CR or value):
        Sky fitting algorithm: mode
Inner radius of sky annulus in scale units (10.) (CR or value):
        New inner radius of sky annulus: 10. scale units 10. pixels
Width of the sky annulus in scale units (10.) (CR or value):
        New width of the sky annulus: 10. scale units 10. pixels
Standard deviation of background in counts (INDEF) (CR or value):
        New standard deviation of background: INDEF counts
File/list of aperture radii in scale units (3.) (CR or value): 4.
        Aperture radius 1: 4. scale units 4. pixels
Minimum good data value (80.) (CR or value):
        New minimum good data value: 80. counts
Maximum good data value (32000.) (CR or value):
```

Figure 37: Questions and answers with **phot**.

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 (obj211.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 4.0 pixels. The number of counts within this aperture will be what defines the zero-point of your frame, as we will see in Sec. 4.10 and keeping this value *fixed* to some value like your typical FWHM will keep you safe.

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.

```
PACKAGE = daophot
   TASK = psf

image   =               obj211  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 38: Parameter file for **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.)
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 38. 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); here we are using 13 (the images here are somewhat oversampled, with typical FWHM ≈ 4). 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 38). 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 35. Then:

display obj211 1

tvmark 1 obj211.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 **obj211.psf.1.imh** that the **psf** routine has also added a text file **obj211.psg.1**. If you **page** this file you will see something like that of Figure 39. 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 40. 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

```
#K IRAF       = NOAO/IRAFV2.10BETA      version    %-23s
#K USER       = massey                  name       %-23s
#K HOST       = tofu                    computer   %-23s
#K DATE       = 01-02-92                mm-dd-yr   %-23s
#K TIME       = 13:36:10                hh:mm:ss   %-23s
#K PACKAGE    = daophot                 name       %-23s
#K TASK       = psf                     name       %-23s
#K IMAGE      = obj211                  imagename  %-23s
#K APFILE     = obj211.mag.1            filename   %-23s
#K PSFIMAGE   = obj211.psf.1            imagename  %-23s
#K GRPSFILE   = obj211.psg.1            filename   %-23s
#K SCALE      = 1.                      units/pix  %-23.7g
#K OTIME      = 3:11:33.0               timeunit   %-23s
#K IFILTER    = 10                      filter     %-23s
#K XAIRMASS   = 1.034776                number     %-23.7g
#K PSFRAD     = 13.                     scaleunit  %-23.7g
#K FITRAD     = 4.                      scaleunit  %-23.7g
#
#N ID    GROUP XCENTER   YCENTER   MAG         MSKY

#U ##    ##    pixels    pixels    magnitudes  counts
#F %-9d  %-6d  %-10.2f   %-10.2f   %-12.3f     %-14.3f

145      1     239.06    207.60    14.677      137.598
122      1     227.71    189.31    20.140      131.225
110      1     227.94    182.25    23.740      130.841
645      2     718.10    477.73    17.228      115.288
757      2     713.06    489.04    20.391      115.959
758      2     722.53    472.35    19.810      113.978
746      3     416.02    745.53    18.464      114.790
```

Figure 39: The "point spread function group" file **obj211.psg.1**

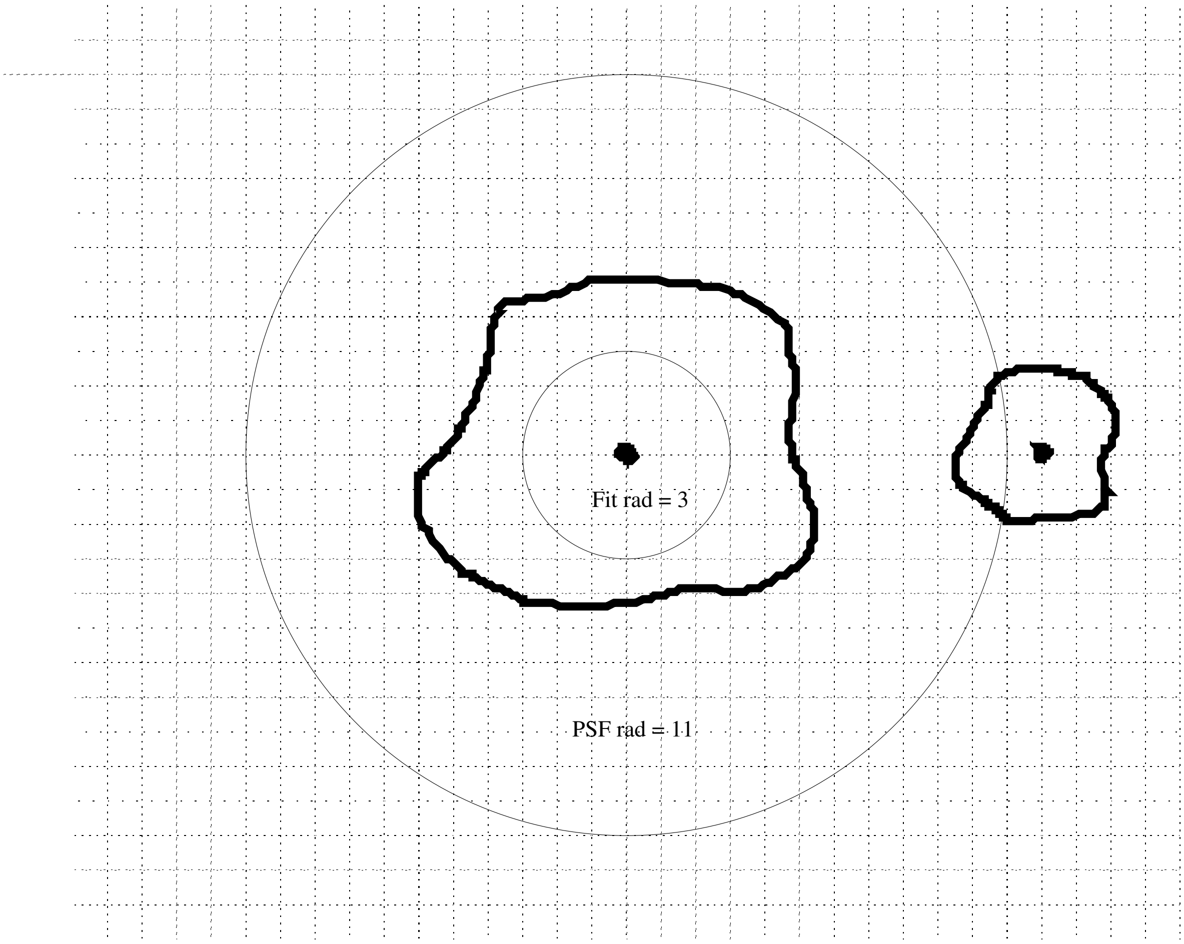

Figure 40: The zeroth order PSF of a star with a neighbor 12 pixels away.

```
da> nstar
Image corresponding to photometry: obj211
Input group file (image.grp.?) (default): obj211.psg.1
PSF image (default: image.psf.?) (default):
Output photometry file (default: image.nst.?) (default):

Psf radius in scale units (13.): 7.
        New psf radius: 7. scale units 7. pixels
Fitting radius in scale units (4.):
        New fitting radius: 4. scale units 4. pixels
Maximum group size in number of stars (60):
        New maximum group size: 60 stars
Minimum good data value (80.) (CR or value):
        New minimum good data value: 80. counts
Maximum good data value (32000.) (CR or value):
        New maximum good data value: 32000. counts

da> substar
Image corresponding to photometry file: obj211
Photometry file (default: image.nst.?) (default):
PSF image (default: image.psf.?) (default):
Subtracted image (default: image.sub.?) (default):

Psf radius in scale units (13.): 7.
        New psf radius: 7. scale units 7. pixels
Minimum good data value (80.) (CR or value):
        New minimum good data value: 80. counts
Maximum good data value (32000.) (CR or value):
        New maximum good data value: 32000. counts
```

Figure 41: Running **nstar** and **substar** with a small psf radius.

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 **obj211.psg.1** for the "input group file". Remember to decrease the **psf radius** when it tries to verify that number. **nstar** will produce a photometry output file **obj211.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. An example of the **nstar** and **substar** steps is show in Fig. 41.

When the routine finishes, **display** the resultant frame **obj211.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.

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 **obj211.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? (Compare Fig. 36 with Fig. 42.) 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 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 **obj211.sub.1**. For the "Aperture photometry file" we can tell it the original photometry file **obj211.mag.1** if we want, or even the old ".psg" file **obj211.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" **obj211.psf.2** and for the "Output psf group file" **obj211.psg.2**. We can of course do this all on the command line:

psf obj211.sub.1 obj211.mag.1 obj211.psf.2 obj211.psg.2 datamin=50.

Make a new PSF using the cursor as before. *This time make sure you take the large psf radius.*

How good is this revised PSF? One way to find out, if you have multiple PSF stars, is to 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 **obj211.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. If you have only a single good PSF star (as we concluded with this frame), you will have to wait until you have done photometry over the entire frame to evaluate how good a PSF you

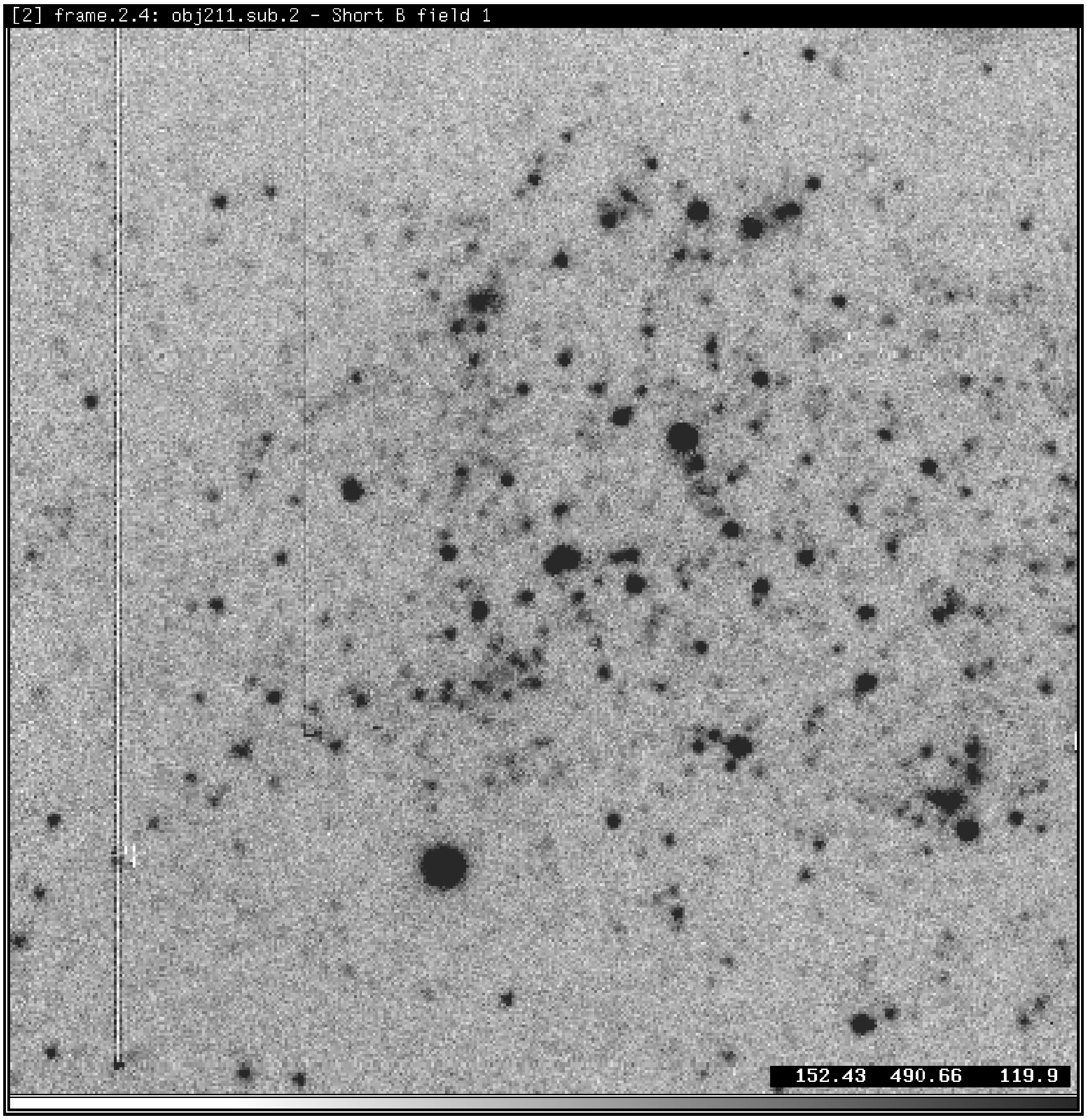

Figure 42: Making the first revision PSF using the frames with the neighbors subtracted. Compare this to Fig. 36, and note that the neighbors around the bright star (center bottom) have been removed.

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

image   =              obj211  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  =                 yes) 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 (13.):
        New psf radius: 13. 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
Minimum good data value (80.) (CR or value):
        New minimum good data value: 80. counts
Maximum good data value (32000.) (CR or value):
```

Figure 43: Running **allstar**.

created.

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 43. As you may imagine, **allstar** produces a photometry file **obj211.als.1**, and another subtracted image: **imagename.sub.N**.

How long will this take? On a SPARCstation 2 with lots of memory, and this TI 800×800 image with ≈ 750 stars—some of which are *very* crowded!—the **allstar** run took less than 8 minutes. On slower machines, or if you cannot fit the entire frame into memory, it may take considerably longer, so be patient.

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 **obj211.mag.2** will have been produced. Use **pappend** to concatenate these two files: **pappend obj211.mag.1,obj211.mag.2 obj211.mag.3**. You can now re-run **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—after all, some stars are red, some are blue (that's presumably why you are doing this after all, 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. In this example the image *obj210* is the U exposure, *obj211* is the B exposure, and *obj212* is the V exposure. To match these three frames we will need to do something like this:

txdump obj210.als.2 > tvu
txdump obj211.als.2 > tvb
txdump obj212.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 obj210 1**, **display obj211 2**, **display obj212 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=209 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. 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 obj213 tvv default 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 obj211 1
   txdump obj211.als.2 xc,yc yes > tvb
   tvmark 1 tvb color=204 interact-

```
ap> txdump *nst* image,id,group.mag,yes
obj210  43  1  16.589
obj210  36  1  21.385

obj211  62  1  14.677
obj211  51  1  20.249

obj212  245  1  16.616
obj212  270  2  17.898
obj212  54  3  17.207
```

Figure 44: The PSF stars and their neighbors from the three images.

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 obj211 bapstars bapphot calgor="centroid"

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

group obj211 bapphot default default crit=0.2

This will have generated a "group" file **obj211.grp.1**.

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

nstar obj211 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 we 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. (Uing the editor would be faster in this case, but imagine that we have many more PSF stars and neighbors than we do in this example.) 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 *nst* image,id,group,mag yes

In the example shown in Figure 44 we see that the PSF stars have magnitudes of 14.7-17.2, and that the neighbors all have magnitudes greater than 20.2. Thus we can use **pselect** to create a file containing the photometry of the faint stars:

pselect obj210.nst.1,obj211.nst.1,obj212.nst.1 obj210sub,obj211sub,obj212sub

and answer **MAG>20.0** when you are queried for the "Boolean expression". This will put the photometry of the stars you wish to get rid of into the three files *obj210sub,obj211sub,* and *obj212sub*. Note from the example shown in Fig. 44 that the thrid of these files will in fact be blank—none of the three stars had any neighbors.

Next do an

txdump obj210.nst.1 xc,yc > obj210ap

and answer **MAG<20.0** 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 **obj210ap**. Do this for the other two files as well.

Next subtract the stars in the first file:

substar obj210 obj210sub

and accept the defaults. This will result in the subtracted image *obj210.sub.N*. It is this file on which we wish to run the aperture photometry to determine the aperture correction:

phot obj210.sub.N obj210ap obj210res aper=4.,15. annulus=20 dannu=5 verb+

You will see something like Figure 45 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 4 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 one of the PSF stars we find an average aperture correction of -0.222. It would have been far better to have more stars to base the aperture correction on, but this frame has few bright isolated stars.

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.)

```
ph> phot obj210.sub.5 obj210ap obj210res aper=4.,15. ann=20 dann=5 verb+

Centering algorithm (none) (CR or value):
        New centering algorithm: none
Sky fitting algorithm (mode) (CR or value):
        Sky fitting algorithm: mode
Inner radius of sky annulus in scale units (20.) (CR or value):
        New inner radius of sky annulus: 20. scale units 20. pixels
Width of the sky annulus in scale units (5.) (CR or value):
        New width of the sky annulus: 5. scale units 5. pixels
Standard deviation of background in counts (INDEF) (CR or value):
        New standard deviation of background: INDEF counts
File/list of aperture radii in scale units (4.,15.) (CR or value):
        Aperture radius 1: 4. scale units 4. pixels
        Aperture radius 2: 15. scale units 15. pixels
Minimum good data value (0.) (CR or value):
        New minimum good data value: 0. counts
Maximum good data value (32000.) (CR or value):
        New maximum good data value: 32000. counts

obj210.sub.5 x: 235.09 y: 210.14 s: 65.55 m: 16.588 16.366 e: ok
```

Figure 45: The aperture correction run of **phot**.

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.
- **pappend** 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

Now that we have a good set of instrumental magnitudes in all three filters (*obj210..als.3*, *obj211.als.3*, and *obj212.als.6* in this example) it is time to transform these to the standard system using the transformation equations and coefficients determined in Sec. 3.6. The first step is to use **mkobsfile** (no "n"!) to prepare a a file containing a matched set of instrumental magnitudes for each object that was found on all three frames. (If an object *wasn't* found on all three frames, we have no idea how to get its color, and hence how to apply any color term.) It is **mkobsfile** which will apply the aperture correction, and we will also have to be prepared to tell **mkobsfile** what shift to apply to the x and y values on each frame to register them to the other frames.

```
                              I R A F
                Image Reduction and Analysis Facility
PACKAGE = photcal
   TASK = mkobsfile

photfile=                       The input list of APPHOT/DAOPHOT databases
idfilter=             00,10,20  The list of filter ids
observat=                       The output observations file
(imsets =                STDIN) The input image set file
(obspara=                     ) The observing parameters file
(obscolu=                2 3 4) The format of obsparams
(shifts =                STDIN) The x and y coordinate shifts file
(apercor=                STDIN) The aperture corrections file
(apertur=                    1) The aperture number of the extracted magnitude
(toleran=                   3.) The tolerance in pixels for position matching
(allfilt=                  yes) Output only objects matched in all filters
(verify =                   no) Verify interactive user input ?
(verbose=                  yes) Print status, warning and error messages ?
(mode   =                   ql)
```

Figure 46: The parameter file for **mkobsfile**.

You may first want to think a little about how you want the stars numbered. Do you want them in magnitude order? In x order? In y order? In random order? If the latter is your preference proceed; otherwise, take the allstar output file corresponding to the "first" frame (i.e., *U* inthis example) and use **psort** to sort on some meaningful quantity. For instance

psort obj210.als.3 ycenter

will sort on the file on "y" value.

To deal with a single set of *UBV* exposures, it is easiest to answer issues such as the shift, aperture corrections, and image set definition directly from the terminal. Edit the parameter file for **mkobsfile** so it resembles that of Fig. 46.

Run **mkobsfile** and answer the questions as shown in Fig. 47. If you don't know the shifts between frames, use **imexamine** and the "r" key to figure this out. The shifts are in the sense that they are what you need to *add* to the x and y values from the B exposure to make them like the x and y of the U exposure, i.e., $x_U - x_B$ and $u_U - y_B$. When it asks you for the "name of image set" be sure to keep this specific but short—all your stars will have this name as the root! Also be sure to give these shifts separated by a space, and not a comma!

The output file from this will strongly remind you of the output file from your standard stars (Fig. 20); the results from the program stars here are shown in Fig. 48.

Finally, you are ready to calibrate these program stars. If you set up your equations the way we discussed in Sec. 3.6, with the instrumental magnitudes on the left and the standard indices on the right then you need to use **invertfit**. (If you decided that equations such as a photoelectric photometrist might use, with instrumental magnitudes on the right and standard indices on the left, were more appropriate to your data, then you will want to use **evalfit** instead.)

Edit the parameter file of **invertfit** so it resembles that of Fig. 49

```
ph> mkobsfile obj210.als.3,obj211.als.3,obj212.als.6
The list of filter ids (00,10,20):
The output observations file: n300field1sobs

Enter name of image set 1 (name, <EOF>=quit entry): f1s
    Enter image name 1 (name, <CR>=INDEF): obj210
    Enter image name 2 (name, <CR>=INDEF): obj211
    Enter image name 3 (name, <CR>=INDEF): obj212
Enter name of image set 2 (name, <EOF>=quit entry): ^Z


Image set 1 (f1s):  Enter the shift in x and y
    Image obj210 (xshift  yshift, <CR>=0.0 0.0, <EOF>=quit entry):
    Image obj211 (xshift  yshift, <CR>=0.0 0.0, <EOF>=quit entry): -3.6 2.8
    Image obj212 (xshift  yshift, <CR>=0.0 0.0, <EOF>=quit entry): 1.1 1.4


Image set 1 (f1s):  Enter the aperture correction
    Image obj210 (magnitude, <CR>=0.0, <EOF>=quit entry): -.222
    Image obj211 (magnitude, <CR>=0.0, <EOF>=quit entry): -.165
    Image obj212 (magnitude, <CR>=0.0, <EOF>=quit entry): -.280


Observations file: n300field1sobs
        Image set: f1s  134 stars written to the observations file
```

Figure 47: Running **mkobsfile** and inserting the shifts and aperture corrections interactively.

Finally, you are ready to run **invertfit** and obtain photometry for all your program stars! The output file will resemble that of Fig. 50.
This output file is "self-documenting". Notice that by including "x00" and "y00" in the parameter **print** of **invertfit**, we have the positions for these stars as well.

It is a long, hard road you have traveled, but you do have calibrated photometry at this point! It's time to really get to work!

```
# FIELD             FILTER      AIRMASS XCENTER  YCENTER  MAG        MERR

f1s-1               00          1.037   686.59   11.37    21.532     0.119
*                   10          1.035   686.96   11.54    20.129     0.065
*                   20          1.033   687.24   11.18    20.004     0.065
f1s-2               00          1.037   530.70   64.68    22.119     0.199
*                   10          1.035   530.34   65.37    20.672     0.078
*                   20          1.033   530.66   64.84    20.488     0.091
f1s-3               00          1.037   168.41   87.29    20.607     0.065
*                   10          1.035   168.92   87.80    19.461     0.027
*                   20          1.033   169.81   87.34    19.199     0.063
f1s-4               00          1.037   57.95    103.22   20.997     0.095
*                   10          1.035   58.48    103.41   20.077     0.059
*                   20          1.033   58.58    102.94   19.944     0.062
f1s-5               00          1.037   522.39   106.22   20.304     0.055
*                   10          1.035   522.72   106.27   18.844     0.029
*                   20          1.033   523.20   106.03   18.759     0.065
f1s-6               00          1.037   333.61   113.31   20.279     0.054
*                   10          1.035   334.25   113.47   19.734     0.034
*                   20          1.033   334.51   113.00   19.824     0.041
f1s-7               00          1.037   160.48   134.03   20.707     0.066
*                   10          1.035   160.92   134.16   20.103     0.053
*                   20          1.033   161.19   133.86   20.115     0.072
f1s-8               00          1.037   89.12    140.98   21.183     0.099
*                   10          1.035   89.35    141.07   20.232     0.062
*                   20          1.033   89.85    140.87   20.209     0.073
f1s-9               00          1.037   391.61   151.42   19.147     0.026
*                   10          1.035   391.62   151.54   18.763     0.152
*                   20          1.033   392.28   151.05   18.200     0.056
f1s-10              00          1.037   401.43   155.58   20.753     0.061
*                   10          1.035   402.03   155.62   20.265     0.058
*                   20          1.033   402.69   154.90   20.264     0.078
(more)
```

Figure 48: The output from running **mkobfile**. A careful comparison with your allstar output files will reveal that the xcenter and ycenters have been modified to include the shift you gave, and that the magnitudes now include the aperture corrections.

```
                              I R A F
                    Image Reduction and Analysis Facility
PACKAGE = photcal
   TASK = invertfit

observat=      n300field1sobs  List of observations files
config  =            ctio.cfg  Configuration file
paramete=            ctio.ans  Fitted parameters file
calib   =         n300field1s  Output calibrated standard indices file
(catalog=                    ) List of standard catalog files
(errors =           obserrors) Error computation type (undefined,obserrors
(objects=                 all) Objects to be fit (all,program,standards)
(print  =             x00,y00) Optional list of variables to print
(format =                    ) Optional output format string
(append =                  no) Append output to an existing file ?
(catdir =           )_.catdir) The standard star catalog directory
(mode   =                  ql)
```

Figure 49: The parameter file for invertfit.

```
 Wed 13:57:21 08-Jan-92
# List of observations files:
#               n300field1sobs
# Config:       ctio.cfg
# Parameters:   ctio.ans
#
# Computed indices for program and standard objects
#
# Columns:
#       1       object id
#       2       x00
#       3       y00
#       4       V
#       5       error(V)
#       6       BV
#       7       error(BV)
#       8       UB
#       9       error(UB)


f1s-1      686.590 11.370  21.408  0.067   0.051   0.098   -0.117  0.144
f1s-2      530.700 64.680  21.890  0.094   0.113   0.127   -0.072  0.227
f1s-3      168.410 87.290  20.598  0.065   0.196   0.073   -0.393  0.075
f1s-4      57.950  103.220 21.348  0.064   0.059   0.091   -0.628  0.119
f1s-5      522.390 106.220 20.164  0.067   0.008   0.076   -0.055  0.066
f1s-6      333.610 113.310 21.235  0.042   -0.178  0.057   -1.017  0.068
f1s-7      160.480 134.030 21.524  0.074   -0.095  0.095   -0.957  0.091
f1s-8      89.120  140.980 21.616  0.076   -0.058  0.102   -0.591  0.125
f1s-9      391.610 151.420 19.589  0.058   0.516   0.172   -1.208  0.168
f1s-10     401.430 155.580 21.672  0.081   -0.081  0.103   -1.080  0.090
```

Figure 50: The final output file, ready for you to *really* start to work doing the analysis!

A imexamine: A Useful Tool

In the **images tv** package there is a powerful and versatile task called **imaxamine** 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.

You can run **imexamine** either by simpling typing **imexamine** (if you have already **display**ed the image, or by tying **imexamine** *imagename*. 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