DOLPHOT User's Guide

version 2.0

Andrew Dolphin

adolphin@raytheon.com

January 2013

http://purcell.as.arizona.edu/dolphot/

Contents

8   Changes to DOLPHOT   22

1 Introduction

DOLPHOT is a stellar photometry package that was adapted from HSTphot for general use. A number of utility programs are also included with the DOLPHOT distribution, including basic image reduction routines.

DOLPHOT has been referenced in the literature as "a modified version of HSTphot (Dolphin, A. E. 2000, PASP, 112, 1383)". The PASP paper emphasizes the algorithms used by HSTphot (and DOLPHOT) and tests of its reliability, and is thus useful reading for those who wish to know more about the details of DOLPHOT.

This manual is intended to describe installation and use of DOLPHOT. If you have any problems installing or using DOLPHOT or the provided utilities, please let me know.

Andrew Dolphin

2 Installation

To install DOLPHOT, you must decide on a location where you want the files to reside, and uncompress and unpack the distribution into that directory. A basic installation of DOLPHOT requires about 3.5 Mb for the sources and binaries.

2.1 Makefile

The Makefile contains several lines at the beginning which may be edited for your particular system.

The first two lines select the C and FORTRAN compiler options; the FORTRAN option is used solely for the purposes of linking against PGPLOT libraries (see below). For most modern systems, "gcc" suffices for both.

Note that behavior of compilers and make utilities can vary from system to system. It is strongly recommended that you use gcc as your compiler, and the Gnu make utility (gmake or gnumake) for compiling DOLPHOT. If "make" does not point to the appropriate gnu make binary, you will need to explicitly call "gmake" or "gnumake" from the command line. In addition, the definition of "MAKE" in the base makefile will need to be changed to whatever the appropriate name is.

BASE_DIR should be set to the directory containing Makefile if using gnu's make utility, but if not it can be set directly (for example, export `BASE_DIR = /data0/dolphot2.0`).

DOLPHOT contains some programs that can make use of PGPLOT, if it is installed on your system. (PGPLOT is graphing library written by Tim Pearson, and can be obtained at http://www.astro.caltech.edu/ tjp/pgplot/) If PGPLOT is installed, these lines should be uncommented and the appropriate directory locations used. The first definition (PGHEAD) should point to the location of cpgplot.h. The second must at minimum give the location of libpgplot and libcpgplot (as well as specifying those libraries with "-lpgplot" and "-lcpplot". More libraries and/or directories may be needed. (One can often copy the commands used to compile cpgdemo when installing pgplot.)

The default compilation flags are -O3 (for optimization) and -Wall (to show all normal warnings). Depending on your machine and compiler, you may wish to use different options. A common flag is -m64 to compile into 64-bit binaries if your system does not do this by default.

If you will be installing the ACS, WFC3, or WFPC2 modules, see the seperate documentation for information regarding the camera-specific compilation options. This also affects the Tiny Tim lines in Makefile.

If you will be using very large data sets, you may need to increase the maximum number of images and stars detected. These can be done by setting MAXNIMG and MAXNSTARS appropriately.

Finally are two debugging lines that will cause any floating point errors to either be printed to screen or cause the program to terminate. The first (NAN_PRINT) is enabled by default; the latter (NAN_CRASH) is not. It is recommended to keep the print statements since these are useful for knowing if there is a problem with your photometry.

Note that the compilers and make utilities on some machines do not conform to standards. Here are some problems that have been reported by users, as well as fixes

4

- Problem: make does not understand the "export" command, which ensures that definitions are used by a child make process as well as the current process. Solution: (1) delete the word "export" wherever it appears in Makefile, and (2) copy all definitions from Makefile into psfs/Makefile.

- Problem: compiler does not accept the -D flags, which sends definitions to the compiler. Solution: remove the -D flags from Makefile and insert the following substitute lines into every .c file where the definitions are used:

  ```
  #define BASEDIR ''/data0/dolphot''

  #define USEACS

  #define TTDIR ''/data0/tinytimv62''

  #define SWAP_BITS

  #define PGPLOT

  #define PGHEAD ''/usr/local/pgplot/cpgplot.h''
  ```

  As above, the directory names should be changed to the installation and Tiny Tim directories, respectively. To quickly determine which .c files require flags added, use grep, as in "`grep BASEDIR *.c */*.c`".

After editing the makefile, you should be able to type "make" to compile DOLPHOT without any error or warning messages. Should you need to later recompile the programs, typing "make clean" will remove the original compilations to ensure that all programs get recompiled.

All binaries are compiled into the "bin" directory within the DOLPHOT installation. It is recommended to add this directory to your path. (This is done by editing your .cshrc or .bashrc file in your home directory.)

2.2   Other Operating Systems

DOLPHOT has only been compiled and run on Unix systems (including OS X). You are welcome to attempt to port it to a VMS or Windows system, and I am interested to know if it is successful.

3 Image Reduction

An elementary CCD reduction package is included with DOLPHOT. Documentation about its tasks is extremely terse at this point.

3.1 *applymask*: Apply pixel mask

```
applymask <mask image> <data images>
```

All pixels that are zero in the mask image are set to be bad pixels in the data images. This is useful if you have used some other image reduction software that sets bad pixels equal to the mean sky value or something else that cannot be easily realized based on the image.

3.2 *calib*: Fit night solution

```
calib <fixed parameters>
```

Computes the transformation from standard to transformed magnitudes as a three-parameter fit: constant offset, linear airmass correction, and linear color correction. Any of those values can be set by the user via input parameters.

After starting *calib*, the user will be given an input promt. On each line, the following five values are needed: standard magnitude, airmass, color, instrumental magnitude, and instrumental magnitude uncertainty.

3.3 *ccdproc*: Process CCD image

```
ccdproc <image type> <FITS files>
```

This does full image processing: a pixel flipping (for the WIYN MIMO imager), bad pixel mask (also for the WIYN MIMO imager), overscan correction, bias correction, flatfield correction, trim (to the defined trimsec), and merge of amplifiers (for cases in which multiple FITS extensions have the same CCDNAME keyword).

Note that *ccdproc*'s default actions can be modified by ccdproc.param.

3.4 *crrej*: Clean images of cosmic rays

```
crrej ( <FITS file> <dx> <dy> <scale> <sky> ) per file
    <X size> <Y size> <sky inner> <sky outer> <sky step>
    <fudge factor> <sigma clip> <minimum>
```

Removes detected cosmic rays from images. The cleaned images overwrite the input images, so be sure to back up the images. Typing "crrej" with no arguments gives basic help.

3.5 *darkcombine*: Create dark image

`darkcombine <FITS files> <fudge factor> <sigma clip> <minimum>`

The name of the output dark file is determined by ccdproc.param.

3.6 *extract*: Extract section of a FITS file

`extract <input file> <output file> < chip >`

`extract <input file> <output file> < chip > < X0 > < X1 > < Y0 > < Y1 >`

`extract <input file> <output file> < chip > < X0 > < X1 > < Y0 > < Y1 > < mirror >`

The first option extracts the specified chip number in a single-extension, 3-dimensional FITS file. The second will extract just the image bounded by the coordinates. If the "mirror" value is specified, columns on the ouput image will be reversed.

3.7 *fillhead*: Insert missing FITS header cards

`fillhead <input file> <output file>`

3.8 *flatcombine*: Create flat image

`flatcombine <FITS files> <scale type> <fudge factor> <sigma clip> <minimum> <minimum value>`

3.9 *fringecombine*: Create fringe image

`fringecombine ( <FITS file> <scale> <sky> ) per file <scale type> <fudge factor> <low sigma clip> <high sigma clip> <pattern size> <minimum> <mask> <output file>`

3.10 *imarith*: Basic image manipulation

`imarith <FITS file 1> <operator> <FITS file 2> <output>`

`imarith <FITS file 1> <operator> <value> <output>`

Operators are +, -, *, and /.

3.11 *imcombine*: Combine images

`imcombine ( <FITS file> <dx> <dy> <scale> <sky> ) per file <X size> <Y size> <X offset> <Y offset> <sky inner> <sky outer> <sky step> <fudge factor> <sigma clip> <minimum> <average> <output file>`

3.12 *imhead*: Display header information

 imhead <FITS files>

 imhead -full <FITS files>

3.13 *imstat*: Display image statistics

 imstat <FITS file>

 imstat <FITS file> <section>

 imstat <FITS file> <section> <extension>

3.14 *makergb*: Create RGB image

 makergb (<FITS file> <min value> <max value>) for B,G,R <output>
     <brightness scale> <color scale>

3.15 *setbadpix*: Manually mask bad pixels

 setbadpix <FITS file> <extension> <chip> <x1> <y1> <x2> <y2>

3.16 *splitgroups*: Create single-extension FITS files

 setbadpix <FITS files>

3.17 *subfringe*: Fit and subtract fringe image

 subfringe <FITS file> <fringe image> <scale> <smoothing> <output>

3.18 *zerocombine*: Create zero image

 zerocombine <FITS files> <sigma clip> <minimum>

4 Preprocessing Steps

DOLPHOT runs from the command line, rather than from inside IRAF. No other software needs to be installed for DOLPHOT to run, although you will probably want an image display program (such as DS9) so that you can view the images you are working on.

DOLPHOT is able to handle most types of FITS data. If you have any issues with it unable to read data, please let me know the details.

4.1 *calcsky*: Calculate Sky for Image

`calcsky <fits base>` $< r_{in} > \quad < r_{out} > \quad < step > \quad < \sigma_{low} > \quad < \sigma_{high} >$

Calcsky creates a sky image, which depending on your DOLPHOT parameters can either provide an initial guess of the sky map or a definitive measurement. The sky computation is made by taking all pixels in an annulus around the pixel in question. The annulus extends from r_{in} to r_{out} pixels from the pixel whose value is being measured, and is sampled every *step* pixels. While $step = 1$ will always work, one can typically set *step* to the PSF size and achieve equally good results.

The algorithm is a mean with iterative rejection. Each iteration, the mean and standard deviation are calculated, and values falling more than σ_{low} below or σ_{high} above the mean are rejected. This procedure continues until no pixels are rejected.

After an initial sky map is calculated, it is boxcar smoothed with a kernel of size *step*.

Calcsky can run in an alternative mode, which is signaled by setting *step* to a negative number. This option will divide the image into squares of size $-step$ pixels on a side and run the iterative rejection algorithm on each square. The resulting value is assigned to the central pixel of the square, and the sky map is created by interpolation. When this option is used, r_{in} and r_{out} are ignored.

The resulting sky image is a FITS file named <fits base>.sky.fits.

5 Running DOLPHOT

5.1 DOLPHOT

The workhorse of the DOLPHOT package is the routine *dolphot*.

```
dolphot <output> <options>
```

The output file contains the photometry; there are also a number of <output>.* files created that will be described in the next section.

The options can be a combination of parameter files (-p<filename>) and parameter settings (<variable>=<value>¿). They are set in order, so that a variable set in a parameter file listed first can be overridden by a variable set manually in a later parameter.

5.2 DOLPHOT Parameters

DOLPHOT's behavior is controlled by the parameters set in the parameter files. Following is a list of parameters and default values. In the parameter definitions, integer and floating point parameters are differentiated by the absence or presence of a decimal point. (`Nimg = ''1''` signifies that it is an integer; `RAper = ''5.0''` signifies a floating point parameter.)

First are the image-specific parameters, which share a `img_` or `img#_` prefix. The former is used for any images for which the latter is not defined; the second will specify which image the parameter is defining. Image numbers start with 1, with the special value 0 used to specify the properties of a reference image. In the list below, the shorter `img_` version will be used.

`Nimg = 1`: Set the number of images to be photometered. An additional reference image is not counted in `Nimg`.

`img_file (undefined)`: Specify base filename of image (the filename minus the .fits suffix).

`img_shift = 0 0`: Set offset of image relative to reference. This value can be an initial guess that is later adjusted by DOLPHOT. Values are x and y on the image minus x and y on the reference image. Note that this parameter should not be set for the reference image.

`img_xform = 1 0 0`: Set the scale ratio, cubic distortion, and rotation of the image relative to the reference image. This value can be an initial guess that is later adjusted by DOLPHOT. Note that this parameter should not be set for the reference image.

`img_psfa = 3 0 0 0 0 0`: Set the PSF x-FWHM and linear and quadratic variations. This value can be an initial guess that is later adjusted by DOLPHOT.

`img_psfb = 3 0 0 0 0 0`: Set the PSF y-FWHM and linear and quadratic variations. This value can be an initial guess that is later adjusted by DOLPHOT.

`img_psfc = 0 0 0 0 0 0`: Set the PSF eccentricity and linear and quadratic variations. This value can be an initial guess that is later adjusted by DOLPHOT.

`img_RAper = 2.5`: Sets the size of the aperture within which photometry will be performed. For FitSky=0 or 1, this should include most of the light of the star. For FitSky=2, 3, or 4 options, this should also include significant "sky" area outside the star.

`img_RChi = -1`: Sets the size of the aperture within which the chi value will be calculated. This is used to determine object locations. This should generally include only the peak of the stellar PSF. RChi cannot be larger than RAper. If not defined (less than or equal to zero), RChi is set equal to RAper.

`img_RSky = 4.0 10.0`: Inner and outer radii for computing sky values, if FitSky=1 is being used. Also used in a few places if using FitSky = 2, 3, or 4, so should always be set. The inner radius (first number) should be outside the bulk of the light from the star; the outer (second) should be sufficiently large to compute an accurate sky.

`img_RPSF = 13`: Size of the PSF used for star subtraction. The rule of thumb is to make sure this is sufficiently large that significant unsubtracted star light is not seen beyond the subtracted regions in the residual image.

`RPSF`: Same as `img_RPSF`

`img_aprad = 20`: Set the aperture radius used for making aperture corrections.

`img_apsky = 30 50`: Set the inner and outer radii of the annulus used for calculating sky values for aperture corrections.

   Next are the parameters that affect the photometry.

`photsec = `: Set the section of the reference image to be photometered. By default, the entire image is photometered. If one wishes to restrict the solution, six values must be given: extension number (zero for the main image), Z (generally 1), minimum X and Y values, and maximum X and Y values.

`RCentroid = 1`: The centroid used for obtaining initial positions of stars is a square of size 2RCentroid + 1 on each side.

`SigFind = 2.5`: Sigma detection threshold. Stars above this limit will be kept in the photometry until the final output.

`SigFindMult = 0.85`: Multiple for sigma detection threshold in initial finding algorithm. This should be close to one for larger PSFs, and as low as 0.75 for badly undersampled PSFs.

`SigFinal = 3.5`: Sigma threshold for a star to be listed in the final photometry list. To get all stars, set `SigFinal` equal to `SigFind` .

`MaxIT = 25`: Maximum number of photometry iterations

`FPSF = G+L`: The functional form of the analytic PSF. Allowable options are "Gauss" (Gaussian), "Lorentz" (Lorentzian), "Lorentz2" (a squared Lorentzian), and "G+L" (sum of Gaussian and Lorentzian).

`PSFPhot = 1`: Type of photometry to be run. Options are 0 (aperture), 1 (standard PSF-fit), 2 (PSF-fit weighted for central pixels). Option 1 is suggested for most photometric needs, but option 0 can provide superior photometry if the signal-to-noise is high and the field is uncrowded.

`PSFPhotIt = 1`: Number of iterations on the PSF photometry solution, if PSFPhot is 1 or 2. This will refine the noise estimates on the pixels based on the model fit.

`FitSky = 1`: Sky-fitting setting. Options are 0 (use the sky map from calcsky), 1 (fit the sky normally prior to each photometry measurement), 2 (fit the sky inside the PSF region but outside the photometry aperture), 3 (fit the sky within the photometry aperture as a 2-parameter PSF fit), and 4 (fit the sky within the photometry aperture as a 4-parameter PSF fit). Options 1 and 3 are the suggested settings. Option 0 should be used only if the field is very uncrowded; option 2 can be used in extremely crowded fields; option 4 can help in fields with strong background gradients (though I have yet to see this be useful).

`SkipSky = 1`: Sampling of sky annulus; set to a number higher than 1 to gain speed at the expense of precision. This is only used if FitSky is set to 1. In general, this should never be larger than the FWHM of the PSF.

`SkySig = 2.25`: Sigma rejection threshold for sky fit; only used if FitSky is set to 1.

`ForceSameMag = 0`: When photometering a single star, run photometry assuming all measurements through a given HST camera/filter combination to be solved with the same count rate.

`NoiseMult = 0.05`: To allow for imperfect PSFs, the noise is increased by this value times the star brightness in the pixel when computing chi values.

`FSat = 0.999`: Fraction of nominal saturation for which pixesl are considered saturated.

`Zero = 25.0`: Zero point for a star of brightness 1 DN per second.

`PosStep = 0.25`: Typical stepsize in x and y during photometry iterations. Should be set to a factor of a few smaller than the PSF FHWM.

`dPosMax = 3.0`: Maximum position change of a star during a single photometry iteration. Note that this parameter is currently ignored.

`RCombine = 2.0`: Minimum separation of two stars (they will be combined if they become closer than this value). This value can generally be about 2/3 of the PSF FWHM, but setting below 1.4 will not always be effective.

`sigPSF = 10.0`: Minimum signal-to-noise for a PSF solution to be attempted on a star. Fainter detections are assigned type 2.

`PSFStep = 0.25`: Typical stepsize of FWHM during photometry iterations. Setting to zero will replace PSF solution with three-state solution in which a star will be very small, fit the stellar PSF, or very large.

`MinS = 0.65`: Minimum FWHM for a good star (type 1). This should be set to something like half the PSF FWHM.

`MaxS = 2.0`: Maximum FWHM for a good star (type 1). This needs to be set to something larger than the FWHM of the PSF.

`MaxE = 0.5`: Maximum ellipticity for a good star (type 1).

Last are extra options that turn on or off various features.

`UseWCS = 0`: Use WCS header information for alignment? Allowed values are 0 (no), 1 (use to estimate shift, scale, and rotation), or 2 (use to estimate a full distortion solution). Note that any shifts or rotations selected by `img_shift` and `img_xform` are applied in addition to what is determined by the WCS solution. If reducing HST data, selecting UseWCS=1 can eliminate the need for running the fitdistort utilities. UseWCS=2 generally is not recommended for HST data since the distortion coefficients provided with DOLPHOT provide higher-order corrections than do the WCS headers.

`Align = 1`: Align images to reference? Allowed values are 0 (no), 1 (x/y offsets only), 2 (x/y offsets plus scale difference), 3 (x/y offsets plus distortion), and 4 (full third-order polynomial fit)

`AlignIter = 1`: Number of passes on alignment. If the initial alignment guess is inaccurate, two or more iterations may be needed to provide a clean alignment (as indicated by small fit residuals). Must be a positive value.

`AlignTol = 0`: Tolerance on initial alignment solution. If greater than zero, DOLPHOT will search for matches out to the specified distance (in image pixels) from the initial guess. Must be a non-negative value.

`AlignStep = 1`: Step size for search in AlignTol, in image pixels. Must be a positive value. Should be set roughly to the FWHM.

`Rotate = 0`: Correct for rotation in alignment? Allowed values are 0 (no) and 1 (yes).

`SubResRef = 1`: Allows multiple detections per reference image pixe (an n x n grid). If half-pixel dithers were used when obtaining data, a value of 2 would be appropriate.

`SecondPass = 1`: Number of additional passes when finding stars to locate stars in the wings of brighter stars. Must be a non-negative value.

`SearchMode = 1`: Sets the minimization used to determine star positions. Allowed values are 0 (chi divided by SNR) and 1 (chi only). A value of one appears safe for all applications. A value of zero has been seen to fail if images of very different exposure times are used together.

`Force1 = 0`: Force all objects to be of class 1 or 2? Allowed values are 0 (no) and 1 (yes). For crowded stellar fields, this should be set to 1 and the χ and sharpness values used to discard extended objects.

`EPSF = 1`: Allow elliptical PSFs in parameter fits? Allowed values are 0 (no) and 1 (yes).

`PSFsol = 1`: Make analytic PSF solution? Allowed values are -1 (no), 0 (constant PSF), 1 (linear PSF variation), and 2 (quadratic PSF variation).

`PSFres = 1`: Solve for PSF residual image? Allowed values are 0 (no) and 1 (yes). Turning this feature off can create nonlinearities in the photometry unless PSFphot is also set to zero.

`psfstars = `: Specify coordinates of PSF stars. The file must contain extension, chip, X, and Y (the first four columns of DOLPHOT output).

`psfoff = 0.0`: Coordinate offset of PSF star list. Values equal the list coordinates minus the DOLPHOT coordinates, and would thus be 0.5 if using a DAOPHOT or IRAF star list.

`ApCor = 1`: Make aperture corrections? Allowed values are 0 (no) and 1 (yes). Default aperture corrections always have the potential for error, so it is strongly recommended that you manually examine the raw output from this process.

`SubPixel = 1`: The number of PSF calculations made per dimension per pixel. For Nyquist-sampled images, this can be set to 1, but very small PSFs require the extra precision.

`FakeStars = `: Run *DOLPHOT* in artificial star mode. The FakeStars parameter is the name of the text file containing the artificial star data. The file should contain the following information for each star, one star per line: extension (0 = main image), chip (usually 1), X, Y, and the number of counts on each image. If the warmstart option is being used, one also needs to specify the recovered X, Y, and object type values before the counts. Note that photometry must be run *first*; the photometry list, PSFs, etc. from *DOLPHOT* are used as inputs in the fake star routine.

`FakeMatch = 3.0`: Maximum allowable distance between input and recovered artificial star.

`FakePSF = 1.5`: Approximate FWHM of the image, used to determine which of two input stars a recovered star should be matched with.

`FakeStarPSF = 0`: Use PSF residual from initial photometry run. This should be left at zero, unless the PSF residuals are small and well-measured.

`FakePad = 0`: Define how far an artificial star must be inside the bounds of at least one of the images being used. Zero means that the star will be used if its center falls within any of the images. Negative values allow stars whose centers are outside all images to be used.

`RandomFake = 1`: Apply Poisson noise to fake stars when adding them. This should *always* be used, unless running fake star tests twice (once with and once without) to quantify photometric errors from crowding and background independently of the errors due to photon noise.

`UsePhot = `: Use the alignment, PSF solution, and aperture corrections from a previous photometry run. This allows re-running small patches of the image while having access to the more robust solutions obtained from an entire frame. The UsePhot parameter should equal the name of the original photometry run. Note that, if different photometry parameters are used, the aperture corrections may not be correct for the photometry. One will need to compare the original and re-run photometry to determine any systematic offset between the two sets and correct the re-run magnitudes accordingly.

`DiagPlotType = `: Generate diagnostic plots showing aperture corrections, PSF correction image, and alignment residuals. Options are PS, GIF, and PNG. Plots are generated only if PGPLOT is used.

`VerboseData = 0`: Generates a file named ".data" that includes all numbers output to the console while running (alignment, PSF solution, aperture corrections).

DOLPHOT includes a warmstart option, which allows you to predetermine a list of stars that will be photometered. The stars' position and types are fixed, and only brightnesses are measured. The most common use of this feature is variable star work, in which star positions are measured off a reference frame. That reference frame should be specified, and the approximate offsets of the image to be photometered should be calculated and set using the shift and xform options above. The format of the warmstart file is the extension, Z, X, Y, type, and signal-to-noise of each star. If taking from a DOLPHOT solution of the reference frame, these are the first four columns, the tenth, and the sixth. To run in this mode the warmstart file should be specified with the `xytfile` option set to the star list filename.

DOLPHOT can also compute photometry using difference imaging. In addition to the warmstart file (to which one must also add the counts and magnitude to every line), the PSF solution for the reference image photometry must be supplied. This is done with the `xytpsf` option, which should be set to the .fits file containing the PSF from the DOLPHOT solution of the reference image. One would also need to ensure that the `img0_psf` options are carefully set to equal whatever analytic PSF was determined for the reference image. The output photometry will be on the count and magnitude scale of

the reference image. This feature of DOLPHOT was described in Dolphin et al. (2004, AJ, 127, 875).

Screen output during the running of *DOLPHOT* will first give the information from the fits file (exposure time, gain, read noise, etc), and later the progress of the solution: the number of stars found in each search pass, the number of close stars eliminated, adjustment of the PSFs, etc.

5.3   DOLPHOT Output

DOLPHOT's output is quite complex, but breaks into simple sections. All information for one star is contained on one line to facilitate the use of text utilities such as awk. An output file with the suffix of `columns` identifies the specific contents of each column of the output data; what follows is a description of the values rather than a specific ordered list.

Positions are specified by extension number (zero if not in an extension), chip (for 3-D FITS images only), and X and Y position on the chip.

Fit quality can be estimated by several parameters given for the combined data as well as the per-chip data: χ, signal-to-noise, sharpness, roundness, major axis, crowding, and object type.

Because they depend very heavily on how well you have tuned the `NoiseMult` parameter, the χ values tend to be unreliable for measuring goodness of fit.

The sharpness is zero for a perfectly-fit star, positive for a star that is too sharp (perhaps a cosmic ray), and negative for a star that is too broad (perhaps a blend, cluster, or galaxy). In the uncrowded field, good stars should have sharpness values between -0.3 or 0.3. (If FitSky is 3, allowable sharpness vaues should be a tighter range.)

Roundness is usually not used for star discrimination, but can be used to discriminate between extended objects and diffraction spikes.

The crowding parameter is in magnitudes, and tells how much brighter the star would have been measured had nearby stars not been fit simultaneously. For an isolated star, the value is zero. High crowding values are also generally a sign of poorly-measured stars.

Object types are as follows:

- 1: good star

- 2: star too faint for PSF determination

- 3: elongated object

- 4: object too sharp

- 5: extended object

It goes without saying that anything fit as types 3, 4, or 5 should be excluded from your photometry list.

After this come the blocks of photometry, one per image. Each photometry block contains all of the goodness of fit values listed above, plus counts, background, count rate, count rate uncertainty, magnitude, magnitude uncertainty, FWHM, PSF ellipticity,

16

the three PSF parameters (a, b, and c), and error flag. Most of these are defined in the same way as described above.

If using any of the HST camera modules, a transformed VEGAMAG magnitude is provided if it can be computed based on the available filter set. Also, the five PSF parameters are not given.

An error flag of zero means the star was recovered extremely well in the image; other values added to the flag indicate the type of problem:

- 1: photometry aperture extends off chip

- 2: too many bad or saturated pixels

- 4: saturated at center of star

- 8: extreme case of one of the above

In general, star errors of types 1-3 are usable; if the error flag is 4 or higher, it is unusable. Star errors of types 4-7 may be usable, if charge from the saturated pixels does not leak into adjacent pixels. If precision photometry is required, only error flags of zero or perhaps one should be used.

There are a number of additional output files created. The most important is likely <output>.info, which contains the general information for the image, including a listing of the photometry blocks as well as image alignments and aperture corrections.

An image residual and PSF residual (*.res.fits and *.psf.fits) show the subtracted image and PSF alterations. These are ordinary FITS files and can be viewed.

Three other files are useful for troubleshooting: <output>.apcor, and <output>.psfs. The magnitude file contains individual-image fluxes with the standard PSF fit, with the standard small aperture fit, and with a plain small aperture fit (no weighting by PSF fraction within the aperture). The aperture correction file lists image number (starting from 0), chip, X, Y, PSF-fit magnitude, aperture correction to 0.5 arcsec, uncertainty, and weight. Finally the PSF star file lists the PSF stars, followed by values of 0 or 1 to tell whether they were (1) or were not (0) used for each image.

A word about the chip and position standards for DOLPHOT. DOLPHOT gives four values, extension, Z, X, and Y. The extension is zero if it is the root of the FITS file, or 1+ if an extension. Z is counted from 1 for 3-axis data, such as WFPC2 images; otherwise it is just 1. The X and Y positions are made such that the integer value means the star is centered in the lower left corner of the pixel. A star at (0.0,0.0) would be in the lower-left corner of the lower-left pixel. This will give positions identical to DoPHOT, but 0.5 lower in both X and Y than DAOPHOT.

5.4   Routines

The overall structure of the photometry program is that for each chip, a two-pass search for stars is made, followed by an iterative solution. Each search pass locates peaks in the image brightness, and attempts photometry on each that is found. Should two peaks correspond to one star, only one is put into the star list. The first pass is made on the

17

image minus the sky; the second is made with stars located in the first pass subtracted. To avoid false detections in the wings of bright stars, the second pass will only locate stars that have brightness peaks in the original frame as well as in the subtracted frame. Additionally, any pair of stars centered within 1.5 pixels of each other is combined into a single star.

The iterative solution is made for up to a user-defined number of iterations, depending on how long is needed for the solutions to converge. For each iteration, stars whose neighbors' photometry changed in the previous iteration will be remeasured. Each stars is measured with the sky and all other stars subtracted, meaning that in theory, the frame is blank except for the star being measured and stars below the detection threshold.

Once the solution is reasonably converged, DOLPHOT will attempt to determine a residual for the PSFs for a chip. This is only done if using PSF-fitting photometry (not if using small aperture photometry). It is observed that the PSFs are generally broadened at this time.

After all stars in a frame converge (or after the maximum number of iterations has been reached), a fine solution is made for all stars, providing improved astrometric accuracy.

6 Artificial Star Tests

Once the photometry has been run to your satisfaction, you will likely want to measure completeness, as well as photometric errors due to blending. This is done with fake star tests.

6.1 *fakelist*: Create artificial star list

```
acsfakelist <photometry> <Filter 1> <Filter 2> <Filter 1 minimum> <Filter
    1 maximum> <Color minimum> <Color maximum>
```

This utility creates a artificial star list that can is suitable for the *DOLPHOT* FakeStars parameter. The "photometry" parameter should be the original photometry from *DOLPHOT*. The two filters can either be both HST filters from the same camera or both $UBVRIJH$ (but not one HST and one $UBVRIJH$). Note that, for the colors to make sense, `Filter 1` should be bluer than `Filter 2`. The filters should be prefaced by the camera and underscore, e.g. WFPC2_F555W, ACS_F555W, and WFC3_F555W would reference the F555W filter on each of the three cameras.

Three additional flags can be sent to *fakelist*:

- `-usexy=`*filename*: Distribute artificial stars on chip according to photometry file. The file should be a text file with stars listed one per line with the following data: extension, chip, X, Y. By default, stars are distributed evenly.

- `-usecmd=`*filename*: Distribute artificial stars on CMD according to photometry file. The file should be a text file with stars listed one per line with the following data: extension, chip, X, Y, magnitude, and color. By default, stars are distributed evenly.

- `-nstar=50000`: Sets the approximate number of stars on the image.

Output is written to the console. In general, you will want to pipe the console output to a file, and use that file as your FakeStars parameter.

Note that, as of this writing, the generated results are only usable within DOLPHOT if all of the images being photometered are from HST.

6.2 Run Artificial Stars

In the same directory as *DOLPHOT* was originally run, you will now want to rerun *DOLPHOT*, while specifying the additional parameter, `FakeStars`, which is set to the filename created by *fakelist*. If you want the PSFs to include the PSF residuals, you should also use the parameter `FakeStarPSF=1` when re-running *DOLPHOT*.

Note that running the artificial star tests can take a very long time. Once completed, you will have a `.fake` file that lists the input values for every star, as well as the complete *DOLPHOT* photometry output.

7 Other Utility Reference

Additional utilities are included. They are listed here for completeness but presently undocmented.

7.1 *addstars*: Adds stars to the images

```
addstars <output> <options>
```

This is virtually identical to running *dolphot* in fake star mode, except that all stars in the fake star file are added to the images. Output FITS files have names ending in "mod.fits".

7.2 *apphot*: Simple Aperture Photometry

```
apphot <FITS base> <options>
```

The *apphot* utility performs simple aperture photometry, similarly to the like-named task within IRAF. The first parameter is the name of the FITS file to be photometered, minus the ".fits" suffix.

Options can be specified either on the command line, or in a file named `apphot.param`. The list of supported options and defaults is as follows

- `apsize = 20 50 100`: list of apertures to be used for photometry. The last aperture also serves as the inside of the sky annulus.

- `apsky = 20`: width of the sky annulus.

- `FitSky = 1`: if zero, sky is zero. If one, sky is estimated from the sky annulus. If two, a sky image (`<FITS base>.sky.fits`) is subtracted prior to sky estimation.

- `FSat = 0.999`: fraction of the nominal saturation limit to treat pixels as saturated.

- `infn = `: if specified, the list of positions is read from the file instead of console input.

- `Interact = 0`: if zero, photometry is output non-interactively. If one, the user can adjust the sky value interactively. If two, the user also identifies targets interactively (this requires linking with PGPLOT).

- `outfn = `: if specified, photometry is written to the file instead of to console.

- `RCentroid = 0`: if non-zero, *apphot* will refine the target position with a centroid algorithm before performing aperture photometry. The centroid box extends RCentroid pixels from the specified position.

- `SkySig = 2.25`: sigma clipping level for the sky measurement algorithm.

- `Zero = 25.0`: magnitude of a one count per second source.

The primary features above and beyond the IRAF task are interactive selection of objects and sky measurement, which are enabled with the "Interact" option. Interactive sky measurement should be coupled with a relatively large number of apsize specifications, and the sky can be set in such a way that magnitude asymptotically approaches a value as aperture increases.

7.3   *bintab2txt*: Save contents of a binary FITS table to a file

```
apphot <FITS file> <output> <output header>
```

Converts a binary FITS table contained in "FITS file" to data. The table is written to <output>; if the optional <output header> is specified, header information is written to that file.

7.4   *convertpos*: Convert positions between images

```
convertpos <DOLPHOT output> <position file> <DOLPHOT options>
```

This file converts object positions to other images in a DOLPHOT run. The "DOLPHOT output" should be the name of the DOLPHOT photometry file, and the "options" should include all parameters passed to DOLPHOT (aside from the output file).

Positions of interest should be included in the position file, one per line. The format is:

- Image number (zero = reference or photometry list, 1-N = the individual images)

- Extension

- Chip (usually 1)

- X

- Y

convertpos will provide equivalent positions for the reference frame and all of the individual photometry images. If the position does not trace to a location on one of the individual images, a position of -1,-1 will be reported.

7.5   *synthimg*: Create Synthetic Image

```
synthimg <star list> <output> <options>
```

8 Changes to DOLPHOT

8.1 September 20, 2013 release

1. Added "convertpos" utility

8.2 July 13, 2013 release

1. Added VerboseData option to ensure all numbers printed to console are output to a file.

2. Minor bug fixes

8.3 April 27, 2013 release

1. Added parameter SubResRef that allows dolphot to allow multiple stars to be photometered within a single pixel of the reference image.

2. Fake stars will be skipped rather than run if they do not fall within any of the images being photometered. A new parameter, FakePad has been added to require that the stars be a certain number of pixels inside the image.

3. Added capability to run dolphot only through determination of alignment.

4. Bug fixes to fakelist.

8.4 March 31, 2013 release

1. Added fakelist routine that will replace acsfakelist and wfc3fakelist. It does not require all images to be from the same camera.

2. Reduced number of simulteneously open files.

8.5 March 6, 2013 release

1. Improved output from imhead

8.6 January 27, 2013 release

1. Fixed bug in generation of multi-extension PSF and residual images

8.7 January 19, 2013 release

1. Improved alignment when using Align=4

8.8   January 11, 2013 release

1. Updated parameter files to match defaults listed in the manual.

8.9   January 2, 2013 release

1. Added utility "bintab2txt", which converts binary tables into plain text.

2. Bugs have been fixed regarding the maximum distance at which a star's photometry affects other stars. In older versions, images with coarser pixel scales than the reference would underestimate this value.

3. "AlignTol" and "AlignStep" parameters were added, along with capability to handle larger errors in initial alignment solution.

4. Error checking for RAper, RPSF, and RSky was improved.

5. "RSky = # #" and "img_rsky = # #" parameters added. These can replace specifying both "rsky0 = #" and "rsky1 = # #".

6. "ForceSameMag" parameter was added, which forces the count rate to be the same for images in the same HST filter.

8.10   October 14, 2012 release

1. Fixed bug in FITS binary table handling

2. Fixed bug that prevents WFPC2 library from adjusting RChi on a per-chip basis.

8.11   July 18, 2012 release

1. Fixed bug in FITS binary table handling

8.12   June 17, 2012 release

1. Fixed memory allocation bug for very large image sets

2. Added ability to read FITS binary tables

8.13   June 3, 2012 release

1. Restored capability for image-differencing photometry.

8.14 April 22, 2012 release

1. Added "DiagPlotType" parameter to dolphot. This generates diagnostic plots if compiling with PGPLOT.

2. A new .warnings file is generated. It will be blank unless there are potential problems with the photometry.

3. Added "AlignIter" parameter to dolphot; this provides a slightly greater effective search radius for alignment.

4. The .columns file now gives filename, exposure time, and for HST images, filter instead of the image number.

8.15 April 8, 2012 release

1. Added "UsePhot" parameter to dolphot; this allows a re-run of photometry using alignment, PSFs, and aperture corrections from a previous photometry run.

8.16 February 17, 2012 release

1. Added support for Anderson ACS and WFC3 PSFs

2. Made FITS reader more flexible, so that photometry keywords (exposure time, gain, etc.) can be given in a string format.

8.17 December 4, 2011 release

1. Fixed bug that caused photometry to be slightly different depending on order of images.

2. Removed 0.5 pixel error in distortion corrections.

8.18 October 15, 2011 release

1. Multi-camera HST mode is supported

2. RChi, and UseWCS options added

3. Image-specific RAper, RSky0, RSky1, and RPSF options added

4. addstars and applymask utilities added

5. Several bugs fixed

8.19   September 6, 2010 release

1. Added FitSky=4 option

2. Added ability to set SecondPass to values greater than one. This improves the
   efficiency of finding faint stars in crowded fields.

3. Allowing ACS/HRC drizzled reference frames to be used.

4. Initial WFC3 and WFPC2 photometry capabilities

8.20   February 16, 2007 release

1. The crowding parameter now works "as advertised" for FitSky=3

2. The PSF residual calculation now works correctly for FitSky=3

3. Several bugs in the fake star algorithms were fixed.

4. Some bug fixes were made involving ACS photometry obtained immediately after
   a PSF solution.

8.21   November 29, 2006 release

1. The initial background map (from getsky) is now used only a few places in DOLPHOT.
   It is used when making the signal-to-noise map used for star detection, as a Bayesian
   prior when FitSky is set to 3, and as the background when FitSky is set to 0. Sig-
   nificantly, the background map is no longer subtracted from the residuals.