JUST Do IT!

A DoPHOT User's Manual
now in C!

Mario L. Mateo[1], **Abhijit Saha**[2], **Paul L. Schechter**[3]
and Rebecca Levinson[3]

[1] Department of Astronomy, University of Michigan, 834 Dennison Bldg., Ann Arbor, MI 48109

[2] Space Telescope Science Institute, 3700 San Martin Drive, Baltimore, MD 21218

[3] Physics Department, Massachusetts Institute of Technology, Cambridge, MA 02139

CONTENTS

C. Summary of Object Types

1. PREFACE

DoPHOT is a computer program designed to search for "objects" on a digital image of the sky and to produce positions, magnitudes and crude classifications for those objects. Digital images, like the research programs for which they are obtained, vary enormously, and no single computer program is likely to handle all cases equally well. The particular application for which DoPHOT was designed, and to an extent, optimized, involved large numbers of poorly sampled, low signal-to-noise images for which a fast, hands-off approach seemed desirable. The scientific program for which it was written required photometry of, what at the time, seemed like a large number (10^5) of relatively bright stars, with an accuracy of 0.03 mag or better.

Anyone considering using DoPHOT would do well to think carefully about using some other program (see, for example, the various programs described in the proceedings of the 1st ESO/ST-ECF Data Analysis Workshop edited by Grøsbol, Murtaugh, and Warmels) or writing new code designed specifically for the application at hand. The user who chooses DoPHOT may sacrifice completeness in the identification of objects, and accuracy in their positions, magnitudes and classifications. This is especially likely if DoPHOT is used on data sets (or for research programs) which are very different from those for which it was originally designed. Some users may, despite these losses, find it advantageous to use DoPHOT.

DoPHOT was written and modified by individuals more interested in expeditiously carrying out their own research than in producing a black box requiring minimal understanding of its contents. While some attempt has been made to make the present version less user-hostile than earlier versions, the program is not designed to check whether the input data are "reasonable", or whether the output is "reasonable". The necessary "reason" must be supplied by the user, giving careful thought to the input data and careful attention to the output results. The greater the user's understanding of the program, the more likely he or she will find it useful. Through this manual the authors try to convey a thorough understanding of the program, suggesting ways of monitoring results and pointing out pitfalls to be avoided.

A second consequence of DoPHOT's origin as a program intended for use only by

its authors warrants mention. In the course of working with different data sets a variety of unanticipated and frustrating difficulties were encountered for which ad hoc solutions were implemented. Many of these "quick fixes" used the programmer's equivalent of baling wire and duct tape. They are not pretty. Moreover, while many of the fixes adopted were adequate for the immediate problem they were intended to address, they are far from optimum. New data sets are likely to reveal the shortcomings of many of these fixes.

DoPHOT is a free-standing program, not part of any "package," "facility" or "system." The user is therefore responsible for the needed preprocessing (bias subtraction, "flat fielding," and perhaps the editing of unwanted parts of an image), and for the display of input and output images. Users who find that DoPHOT has been incorporated into some local "system" should recognize that it can be run outside that system, as a non-interactive, batch job, which might avoid some of the overheads endemic to such systems. However, in its current version in C, DoPHOT does now require the NASA HEASARC CFITSIO library for reading and writing fits files. We extend our apologies to those who detest library dependences. We just didn't want to reinvent the wheel on data io.

DoPHOT is composed of a great many relatively short modules, allowing for their quick replacement with improved versions. Since no two data sets present exactly the same challenges, the modules have evolved, and we expect them to continue to evolve, as new circumstances are encountered. The user is *encouraged* to improve upon and to customize these routines and to advise the authors of improvements which they have implemented and the results they have obtained. The user is *cautioned* to preserve a version of the code as originally distributed, to guard against accidental corruption in the implementation of genuine improvements. It is the authors' experience that while many of the difficulties encountered by users are the result of genuine shortcomings of the program, many others are the result of local modifications which have unanticipated repercussions. To help guard against such accidental corruption, a test images and its associated input/output files are distributed with the program, so that the user can check whether the version he or she is using produces results *identical*, not merely close, to those sup-

plied.

DoPHOT's authors believe that astronomers are only beginning to take advantage of the potential of digital images, and that there is much to be had in the way of improvement. Users of DoPHOT are therefore urged to think carefully about which aspects of DoPHOT might be worth adopting in a next generation reduction program and which aspects ought to be abandoned for something better.

Having issued these discouraging warnings, some words of reassurance are in order. DoPHOT has been used with modest success by a great many users, on traditional "stare" mode images, on time-delay-integration images, on Hartmann and Shack-Hartmann test images and even on the comparison arcs of echelle spectrograms. The paper describing DoPHOT has been cited over 500 times. Customized versions have been written which allow (separately) for a point spread function which varies with position within an image, for the classification of objects as asteroid trails, and for taking proper account of the noise associated with the subtraction a smooth background (specifically an elliptical galaxy). Photometry has been obtained for many tens of millions of stellar images, with relative photometry which on occasion reproduces to better than 0.005 mag for a single measurement of a single star. At least some of DoPHOT's users have found that the time invested in learning to use it has yielded a satisfactory return on their investment.

DoPHOT and its capabilities have been described in a paper: Schechter, Mateo, & Saha (1993), PASP 105, 1342. This manual acquaints the new user with running DoPHOT, and serves as a reference for all users alike.

In the sections which immediately follow, DoPHOT is described conceptually, with as little reference as possible to variable names and subroutine names. More detailed descriptions of the parameter names and data formats are given in the two Appendices following the main body of the manual.

The present manual describes DoPHOT 4.0, which is not much different from the manual for DoPHOT 3.0, except in the "Details for Connoisseurs" section. This is by design. DoPHOT 4.0 in C was built modularly from the DoPHOT 3.0 routines in Fortran 77, and until intentional improvements were made, produced identical photometry. Although we have tried to change all instances of 1.0, 2.0, and 3.0 to 4.0, some cases

may have slipped by the editors. It is safe to assume that any references to DoPHOT 1.0, 2.0, or 3.0 really mean DoPHOT 4.0, and any file names containing **1_0**, **2_0**, or **3_0**, should be read to contain **4_0**.

2. GENERAL DESCRIPTION

A central feature of DoPHOT is the adoption of a *model* for every kind of object which one seeks to identify within the image. The model for a cosmic ray is a single high pixel. The model for a star might typically be an elliptical gaussian. The model for a galaxy might *also* be an elliptical gaussian, but one which is signficantly bigger than those associated with stars. The model for a double-star is composed of two single stars. To classify an object, one chooses the model which fits the object best.

With the exception of the model for a cosmic ray, these models are usually specified in terms of analytic functions with free parameters. The output is therefore a list of object classifications and parameters for those objects. Six parameters are associated with the simplest models: x and y position within the image, total (or central) intensity, and three shape parameters (length, width and tilt). While one can obtain better models with more parameters, DoPHOT was written on the gamble (hunch) that one might do reasonably accurate photometry and classification using just these six parameters.

"Object" is a somewhat slippery concept. Operationally, DoPHOT enters an object in its output list if it encounters an array of pixels that are significantly brighter than the surrounding sky when "filtered" through a model stellar profile, or "mask" as it is called later in this manual. The degree of significance and the size of that array must be specified by the user (although DoPHOT supplies defaults for these, and for all other user-specifiable inputs).

"Stars" are a special kind of object (celestial point sources) for which the three shape parameters are, to first order, the same throughout the image. In obtaining magnitudes for stars, DoPHOT assumes that all stars have the same "typical" shape. DoPHOT determines that "typical" shape by fitting the brighter stars for the shape parameters and taking a suitably weighted average. In some versions of the code, those shape parameters are allowed to vary smoothly across the image.

If DoPHOT started out with no idea whatsoever of what the "typical" shape of a star was, then it would be unable to decide whether the first object it encounters is a cosmic ray, a galaxy or a star. The user must therfore supply an estimate of the FWHM in pixels of a stellar image. DoPHOT takes that estimate and multiplies it by 1.2 to in-

sure that none of the first stars encountered are accidentally classified as galaxies, but the user must not be too sloppy about the initial estimate of the FWHM.

The models for various types of objects are fit to subrasters. The sizes of these subrasters are specified by the user or, alternatively, determined by DoPHOT itself as some factor times the typical FWHM of stellar objects. In fitting the object models the background sky level is treated as an additonal free parameter. Fitting to a uniform background corresponds to taking a straight average of the pixels in the subraster in the limit of an infinitely faint object.

This seemingly innocuous choice raises several interesting questions. If the subraster contains objects other than the one being fitted, the parameters derived for the fitted object will err systematically. DoPHOT therefore *removes* objects from the working image as they are identified, subtracting the best current model for the object. This procedure is less likely to give poor results if one first identifies the brighter objects, and then, after removing them, identifies the fainter objects, much in the spirit of the CLEAN algorithm (Högbom 1974, *A. A. Suppl.*, **15**, 417) used with aperture synthesis data.

DoPHOT makes successive passes over the data, identifying progressively fainter objects at threshold levels specified by the user. After each search the shape parameters for all objects are redetermined, new "typical" values are determined, and the stars are fit once again, now with the new "typical" values. Since the objects have been subtracted from the image, they must be temporarily added back, fitted, and then subtracted again. All this is done at considerable expense in computation, in the hope that the newly derived parameters will be more accurate.

DoPHOT constructs and maintains a *noise image* which provides weights for each pixel used in its non-linear least square fitting subroutine. It also produces, as part of its output, a star-subtracted image. The failings of the analytic model for stellar objects are most obvious near bright stars, where one often sees a consistent pattern of residuals from one star to the next. These positive residuals might be expected to trigger the false identification of spurious objects.

To avoid such "false hits," DoPHOT adds extra noise to the noise array every time it subtracts a star from the image. This is taken to be some fraction of the subtracted

light, by default 30% (for the analytic model PSF) unless changed by the user. Moreover, for the purpose of calculating this extra noise, the linear size of each image is taken to be bigger by some factor, by default 1.3 (for the analytic model only), than its actual linear size. The result of this extra noise is that potential "objects" found in the neighborhoods of previously identified objects are less significant than they otherwise would be, and are therefore less likely to be included in the output object list. [The current version of DoPHOT implements an empirical PSF: since this is a much better match to the real stellar images, the extra noise added by default (can be changed by user) is only 3%].

While this avoids "false hits," it has the negative consequence of making it more difficult to identify faint stars near brighter stars. The extra noise associated with each object is temporarily subtracted from the noise array when that object is temporarily added back to the star-subtracted image for refitting.

Fitting models to subrasters offers the advantage that one can solve for the parameters associated with a star even when some pixels are missing, as might be the case if there were a bad column, or if an object lies close to the edge of an image. DoPHOT also removes pixels from further consideration whenever a cosmic ray is detected. DoPHOT will attempt to obtain magnitudes and shapes for objects as long as some minimum number of pixels (which the user can adjust) are still contained in the subraster.

Given the systematic pattern of residuals in the star-subtracted images, one would expect that the total fluxes derived from fitting the model point spread function (PSF) to the data will likewise suffer systematic errors. But to first order, one would expect to make the same systematic error for all stars. As an aid in correcting for such systematic errors, DoPHOT calculates total fluxes in a rectangular aperture of a size specified by the user.

Aperture magnitudes are very much more uncertain than fit magnitudes because there is much more "shot" noise from the sky inside the aperture than there is under the model profile. One must compromise between making the aperture large enough to collect most of the light from an object and yet small enough to minimize the noise contribution from the sky background. While DoPHOT reports both fit and aperture

magnitudes, it is left to the user to compare these two and to decide whether and how to correct the fit magnitudes. For some purposes (e.g. looking for variable stars) the uncorrected fit magnitudes may be sufficient.

When the footprint of an object (the half intensity contour, which in the present version is modelled as an ellipse) is bigger than that of the "typical" star, one faces the choice of classifying the object as a single star, a multiple star or a galaxy. DoPHOT does not do a very thorough job of addressing this question. If an object has shape parameters which make it significantly bigger than a star (by an amount which the user can adjust), it is declared to be big and DoPHOT attempts to fit 2 "typical" stars to it.

The goodness of fit parameters for the fit of one big object and two "typical" stars are then compared, and a decision is made (based on a user adjustable parameter) as to whether the object should be classified as a galaxy or "split" into two stars. Stars which are "split" may be further split, allowing at least in principle, for higher multiples. If the default value of this user adjustable parameter is used, the user will find spurious "galaxies" appearing in globular cluster images and spurious "split" stars in images with large numbers of galaxies.

An as yet unrectified shortcoming of DoPHOT is that it does not maintain pointers linking such sibling objects. The photometry for these objects is obtained using the single point spread function. It is more uncertain than it would be if the siblings were fit as a pair because of the extra noise added after the subtraction of the nearby sibling.

DoPHOT allows for a "warmstart" in which it reads a list of objects and determines typical parameters from those objects. It allows the user to force object types for the objects on the input list. A special object type which is sometimes useful is an "obliteration," a rectangular portion of the image which the program ignores. While one might prefer to deal with such regions by blocking them out in pre-processing, including them in the object list gives one a record of the areas not considered. The program automatically "obliterates" stars brighter than a user-adjustable threshold.

3. CRYPTO-DoPHOT

The following is a schematic version of DoPHOT in an imaginary computer language. The main tasks are identified and their relative locations in the code are correctly illustrated. Reference to this listing of "CRYPTO-DoPHOT" may be useful in understanding some of the detailed descriptions in the following sections of this manual.

```
crypto_dophot
  INPUT_PARAMETERS
  READ_IMAGE
  MAKE_NOISE ARRAY
  {  if warmstart then
     READ_OBJECT_LIST
     DETERMINE_TYPICAL_SHAPE_PARAMETERS }
  {  for threshold = highest to lowest
     [ SEARCH_FOR_PIXELS_ABOVE_THRESHOLD
       ( if high_pixel then
     CHECK_FLUX_THROUGH_STAR_MASK )
       ( if significant then
     CLASSIFY_STAR/COSMIC/BADLY_SATURATED
           ADD_OBJECT_TO_LIST ) ]
     [ for all objects in list
 DETERMINE_OBJECT_SHAPE_PARAMETERS
       ( if object_is_big then
           CLASSIFY_GALAXY/DOUBLE_STAR )]
     DETERMINE_NEW_TYPICAL_SHAPE_PARAMETERS
     [ for all objects in list
       ( if star then
          DETERMINE_FIT_MAGNITUDE
          DETERMINE_APERTURE_MAGNITUDE )
       DETERMINE_EXTENDEDNESS ]
     WRITE_OBJECT_LIST_TO_DISK
     WRITE_OBJECT_SUBTRACTED_IMAGE_TO_DISK }
  END
```

4. CONTENTS OF VERSION 4.0 AND HOW TO USE THEM

The program and data files provided to you have been arranged into a subdirectory structure, which you are strongly urged to maintain. In versions 3.0 and prior, the program was arranged into a more redundant subdirectory structure with the underlying intent to maintain always a pristine ORIGINAL copy of the code as reference, and to have an easy facility to check that code that has been modified produces IDENTICAL results when run on the supplied test images with the supplied input data and parameters. We believe that with modern source control the old subdirectory structure may be cumbersome to users rather than helpful, so it has been dropped in 4.0. We do still encourage that users keep and maintain a pristine copy of DoPHOT at all times on their machines. As with previous versions, all modified code should produce IDENTICAL results when run on the supplied test images with the supplied input data and parameters. Whether the user chooses to use GIT and make their working copy a branch off of a pristine master or simply download DoPHOT twice, one copy for modifications one for verification is at his or her discretion.

4.1 CFITSIO precursor

DoPHOT 4.0 requires the CFITSIO library which is not included with DoPHOT. It can be obtained at http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html and installed by following the guidelines on that page. Once it is installed (and working), edit **Makefile** in the top level, MAIN directory supplied with this package to reflect the path of the CFITSIO installation. Unfortunately, you will NEED TO MODIFY THE MAKEFILE, EVEN IN THE 'PRISTINE' DoPHOT COPY as well as the working copy, however this is the ONLY FILE YOU SHOULD EVER MODIFY IN THE PRISTINE COPY.

4.2 UNIX Systems

The tar supplied to you, should, when properly read, produce a main-level directory (which can be a sub-directory in your general subdirectory tree), which will be referred to as MAIN. MAIN contains

1) **README** containing general information and instruction file(s) that you should read before proceeding.

2) subdirectory **manual** containing a TEXable version of this manual.

3) a **Makefile**

4) the original Version4_0 code as .c files.

5) subdirectory **headers** containing headers for most of the source routines

6) subdirectory **structs** containing additional header files with struct information

7) subdirectory **verif_data** containing test files

8) subdirectory **working_data** which is empty.

Run 'make test' to populate the **verif_data** subdirectory with a dophot executable. Alternately, run 'make dophot' from MAIN to populate **working_data** with a dophot executable.

In either case, the **working_source** subdirectory will populate with .o files and any errors in your modified Makefile or CFITSIO installation will now appear.

On a pristine copy of DoPHOT version4_0, the **working_data** subdirectory is empty. By running the 'make' described above, an executable image of **dophot** is created here. You may choose to use this as your work area for DoPHOT; we urge that you do. Before you can do so, you will need proper parameter files and other optional input files (as described elsewhere in this manual) in addition to the image on which you wish to run DoPHOT.

The **verif_data** subdirectory comes with the test image **J0159.8.chip_01.fits**, the default parameter file **param_default_c**, a file containing the modified parameters **pm**, and a list of input objects **obj_in_J0159.8.chip_01** for the field. Additionally **verif_data** contains a subdirectory **compare_out** containing files intended for verification that the supplied program or any modification you make is executing correctly. By running 'make test' in **working_to_compilable**, **verif_data** will be populated with a dophot executable.

Let us walk through and see how to run DoPHOT on the galaxy cluster image called **J0159.8.chip_01.fits**. Type './dophot' at the command line (or equivalent for your machine). You should be prompted for a parameter modification file. Type in 'pm', which contains the non-default parameters for this test file, and hit return. 'pm' also informs DoPHOT as to which file contains the default parameters, namely **param_default_c**,

which should never be modified.

DoPHOT should now execute in a number of seconds and report 6 output files matching the ones in the **compare_out** subdirectory. Feel free to examine all of them, but in particular diff **obj_out_J0159.8.chip_01** with the **compare_out** file of the same name. If there are any discrepancies, something has gone wrong or you are not working with a pristine copy of DoPHOT.

5. IMAGE FORMATS

All image reading and writing in DoPHOT 4.0 is handled by CFITSIO, and the only accepted file types are FITS format images. CFITSIO can handle almost all standard types of data, but DoPHOT 4.0 will process the pixel data as 32 bit integers regardless. DoPHOT veterans may recall that versions 3.0 and earlier only accepted 16 bit signed integer data and processed it as the same – this has been modified in DoPHOT 4.0. As DoPHOT 4.0 processes data as 32 bit integers, it outputs FITS files of the same type. DoPHOT will preserve all header information from input files in the output files if the input files have 32 bits per pixel (BITPIX = 32 or -32). Barring this matching file size for input and output files, a minimal header is created for the output files.

6. INSTRUCTIONS FOR NOVICES

This section assumes that you have successfully compiled DoPHOT and would now like to try running it. Naturally you would like to try running it on your own data. *Avoid* that temptation, and instead, try running it on one of the test images which are distributed with DoPHOT. This will be very much easier and will teach you some things about the program

6.1 Running DoPHOT on the Test Images

In addition to the executable file which resulted from compiling and linking DoPHOT, you will need up to four additional files (all of which are supplied with the test images):

1) an input image
2) a default parameter file
3) an image-specific parameters modification file
4) an input object list

The test images are written in "FITS" format, the parameter files are written in a format which looks very much like FITS header format (but isn't!). The object list is an ASCII table. An input object list is only needed for "warmstarts."

The command needed to run the executable file will be different for different computers, but likely './dophot'. When DoPHOT is finished, it produces up to six output files:

1) an object-subtracted output image
2) an output object list
3) an parameter report
4) a diagnostic log
5) a shadow file
6) an error file

Before you do anything else, run the program on the first of the test images. You can do this by executing DoPHOT in the 'verif_data' subdirectory, and supplying 'pm' as the image-specific parameters modification file.

Most computers have some means of comparing files, looking for differences. Under both the UNIX and VMS operating systems, that command is "diff". Compare the

output object file with the output list in the 'compare_out' subdirectory supplied with the test images. These should be *identical*, not merely similar. If they are not, there is a very good chance that you are not working with the original version of DoPHOT.

Look at the output object list file. Each row gives data for a different object. If you have chosen DoPHOT's external format for the output photometry file (*i.e.* `OBJTYPE_OUT` = COMPLETE; see Appendix A), then the columns are interpreted as follows:

1) sequential object number, with the objects in the input list at the beginning
2) object classification, with 1 $\Leftrightarrow$ a single star, 2 $\Leftrightarrow$ galaxy, 3 $\Leftrightarrow$ one sibling of a multiple, 7 $\Leftrightarrow$ a faint object for which the shape was not determined and 8 $\Leftrightarrow$ an object which could not be modeled and has, instead been excised. Other object classifications are described in Section 7.
3) x position (x=0.5 $\Leftrightarrow$ centered on first pixel)
4) y position (y=0.5 $\Leftrightarrow$ centered on first pixel)
5) "fit" instrumental apparent magnitude, computed from the model of the object $-2.5 \times \log_{10}$[total intensity in DN]).
6) formal uncertainty in the fit magnitude
7) sky level (in Digital Numbers, DN)
8) a, the long dimension (FWHM) of the footprint
9) b, the short dimension (FWHM) of the footprint
10) θ, position angle of the footprint in degrees
11) "extendedness" parameter
12) "aperture" instrumental magnitude
13) uncertainty in the aperture magnitude
14) aperture sky value
15) difference between the aperture and fit magnitudes

The objects in the input list (if one was used) should be the first objects on the output list. The difference between the aperture and fit magnitudes, while redundant, is a useful diagnostic. This number should be the same (to within the quoted uncertainties) for all stars. The scatter in this quantity should be small for the brightest stars, but grows rapidly for the fainter stars. While the fit magnitudes have smaller uncertain-

ties, and give good relative magnitudes for objects within the image, an *average* correction to aperture magnitudes is needed to compare measurements obtained from different images and to calibrated frames using external standards.

The values for the footprint ellipses are different for each of the galaxies, but are identical for all of the stars. There are versions of DoPHOT which allow the typical parameters to vary in a smooth fashion across the image, in which case the footprint parameters would not be the same. For the case of "obliterated" objects the FWHM refer to the sizes of the x and y sides of the corresponding rectangular obliteration boxes.

The parameter report file should be nearly identical to the default parameter file, with the only differences being the parameters specified by the parameter modification file, or those changed when prompted by DoPHOT due to invalid inputs in the parameter modification file.

The output, object-subtracted image file is a 32 bit integer FITS format file. Regions which have been excluded from consideration because of bright stars or blemishes will appear black if stars are displayed as bright on a dark background.

If the galaxy model and point spread function were perfect, one would see only noise in the object-subtracted image. Since these are only approximate, even when using the empirical PSF, one sees a characteristic pattern in the residuals at the positions of the objects. If the user is interested in seeing the comparative improvement of the empirical PSF over the elliptical PSF, he should refit the images after commenting out THRESHEMP, EMP_STAR_(X,Y,Z), and EMP_SUBRAS_OUT in the parameter modifications file. The residuals on the star objects will now be larger as an analytic model is subtracted.

An interesting exercise is to subtract the object-subtracted image from the input image. This generates a synthetic image which contains only the information in the object list, and is noise-free. This is often a useful diagnostic, since it can indicate, for example, when "objects" have been incorrectly detected on the diffraction spikes of bright stars.

The shadow file and error file contain the fitted shape parameter information for each of the objects. Unlike the output object file, the shadow file contains the unique

shape information from the fit of each star object rather than the average star shape subtracted from the image. The error file reflects the same. Details on the row information can be found in Appendix A.

6.2 Running DoPHOT

If the user has two images of the same field, taken through the same filter, he or she would be well advised (other things being equal) to reduce these first, and to compare in detail the completeness and the relative accuracy of the magnitudes and positions obtained for the two images.

Before running DoPHOT, an image should have had any electronic bias level subtracted and should have been corrected for pixel-to-pixel photometric variations across the image. Any regions (bad columns, overscan, bloomed images) for which the data are "obviously" a problem should be masked off, either by setting the data values in such regions equal to the lowest allowable value consistent with the data format (often -32768) or by entering such regions as "objects" in the "input object" file as regions to be obliterated. If one is too stingy about masking off such regions, DoPHOT will consume a great many CPU cycles finding spurious "objects" in the wings of objects that extend beyond the masked region.

The "modified parameters" file will at the bare minimum need a specification of the full width at half maximum (FWHM) of the typical stellar image. Better classifications will be obtained on the first pass through the program if an approximate sky level, good to 10%, is also specified.

The user may specify file names for the input and output image files, the input and output object lists, and input and output parameter files and the output diagnostic log. The very first thing that DoPHOT does is ask the user to specify the name of the parameter modification file. A valid file *must* be specified because it is from the modified parameters file that DoPHOT determines where the default parameters file exists.

Two user-specified numbers are used to compute the noise array: the read noise (in photons) per pixel and the number of photons associated with each digital number (DN) in the image. The number of photons per DN can be estimated by looking at the pixel-to-pixel variations in high intensity flat field images. The read noise can be estimated by

looking at the pixel-to-pixel variations in low intensity dark or bias images.

There are *two ways* in which the user controls the faintest objects identified by DoPHOT. Arrays of pixels in the vicinity of potential objects are filtered through a model stellar "mask". Entries are added to the object list only when the signal through this mask is significantly greater than the surrounding sky. This minimum signal-to-noise ratio, measured in terms of Gaussian sigma, is user specifiable. Relatively few spurious objects are detected when it is set to 5, but large numbers of spurious objects pass the test when it is set to 3. The calculation of the noise passing through the filter depends upon the noise in the individual pixels, which in turn depend upon the read noise and the ratio of photons per DN.

Upstream of the filter, a decision must be made about which groups of pixels are to be tested. This is done looking for *single high pixels*. When a single pixel is significantly higher than the sky, a group of pixels surrounding this pixel is tested through the filter. In retrospect, the best single pixel test might have been a simple signal-to-noise test. Unfortunately, in the present version of DoPHOT the test is more complicated. The pixel must have a value higher which is greater than the sky by a pixel threshold and the noise level must be lower than a specified fraction (by default unity) of this threshold.

This pixel threshold is decremented in steps, starting with a high value, so as to trigger on only the brightest stars, and proceeding to the fainter stars. The user should specify both the highest and lowest value desired and the logarithmic decrement between steps. The highest threshold should be chosen to trigger on the brightest few stars. At lower thresholds, great numbers of pixels pass the pixel test only to fail on the more time consuming stellar filter. The user should carefully choose a lower threshold value that allows DoPHOT to trigger on mostly real stars and not waste too much CPU time on false alarms.

It is often the case that data values outside certain limits are suspect and should not be used in estimating identifying of measuring objects. The user has the option of specifying these limits. Astronomical detectors often exhibit complicated non-linear behavior when exposed to high light levels. There is a user specifiable upper limit to the central intensity of the stars the program will attempt to fit. Brighter stars will be ex-

cised from the image, if the user hasn't already done so. Also, if the absolute data value of a given pixel is less than a constant (user changeable through parameter SNLIM – set to 0.5 by default) times the noise, the pixel will be ignored during PSF fitting. This can happen if the image has been sky subtracted. We recommend that you use the original data so that the noise model is calculated correctly, but if you must subtract the sky, make sure SNLIM is set to zero to avoid problems.

Having read the above paragraphs, the user should now look at the default input parameter file distributed with the source code and displayed in Section A.5 of Appendix A. The names of the parameters and the comments next to them should indicate which ones will need to be changed to accommodate the user's image. The user should create a modification parameters file with values for these parameters (starting with the FWHM) appropriate to the image. Only parameters that differ from those in the default file need to be specified. Before running DoPHOT, the user should make sure, at the very least, that the FWHM of the stellar images is specified and that the sky level, and the highest and lowest pixel thresholds, as specified either in the default parameter file or the parameter modification file, are suitable.

Having run DoPHOT, the user would do well to mull over the output object list and to look at the original image, the object-subtracted image, and the difference between the two.

7. DETAILS FOR CONNOISSEURS

This section is intended for primarily for users who have run DoPHOT several times and who are comfortable with the material covered in the previous two sections.

7.1 Pixels versus Models

A theme that keeps recurring in these pages is that of modeling an array of pixels. A small subraster is filtered through a model profile to determine whether or not an object should be added to the list. A somewhat larger subraster is fit with a model to determine the shape of an image. The image is modeled by the entire list of objects. Wherever possible the sky, as sampled by the pixels, is modeled. A goal in the design of DoPHOT was to use model fitting in preference to pixel based algorithms wherever possible.

An immediate advantage of such a model based approach is that one can fit the model even when the image is not uniformly sampled. If pixels are missing for some reason (e.g. a cosmic ray, or a bad column, or a saturated pixel at the center of an image), one can still fit a model to the data. Odd-shaped pixels are also easily accommodated. If one had data which was oversampled by a factor of 2, such that the FWHM was of order 4 pixels, then it should be possible (and sometimes desirable) to compress the data by a factor of 2 in both directions and obtain very nearly the same output object list.

This ideal is sacrificed in the use of a single high pixel to trigger the subsequent test for an object, in the use of a single pixel model for a particle event, and (inevitably) in the specification of raster sizes to which models are fit.

7.2 Point Spread Functions and Accuracy

While an elliptical Gaussian was suggested as a possible model for a model for a stellar point spread function, the surface brightness profiles of stellar images tend to look more like power laws. The actual model used for stellar images, consists of similar ellipses of the form

$$I(x,y) = I_0 \left(1 + z^2 + \frac{1}{2}\beta_4 (z^2)^2 + \frac{1}{6}\beta_6 (z^2)^3\right)^{-1} + I_s. \tag{1}$$

where

$$z^2 = \left[\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + 2\sigma_{xy} xy + \frac{y^2}{\sigma_y^2}\right)\right], \tag{2}$$

and

$$x = (x' - x_0) \quad ; \quad y = (y' - y_0), \tag{3}$$

with the nominal center of the image at (x_0, y_0). The dimensionless quantities β_4 and β_6 are user specifiable, but are ordinarily taken to be unity. There are seven free parameters in this function: the shape parameters σ_x, σ_y, and σ_{xy}; the image center, (x_0, y_0); the central intensity, I_0; and the background intensity, I_s.

There is an additional point spread function supplied with DoPHOT in v. 4.0 to complement the above pseudogaussian or 'PGAUSS' model. This newer extended pseudogaussian model 'EXTPGAUSS' has 9 parameters, the extra two being β_4 and β_6. While this extra freedom does result in an improved χ^2 fit on the images tested, the β_4 and β_6 parameters are highly degenerate with the existing σ parameters. Therefore the fit does not always converge. In the case of non-convergence on a given object with this new-model, DoPHOT reverts to the ordinary PGAUSS model and flags the object as such in the output file.

The choice of an analytic, as opposed to a tabulated empirical point spread function was driven by several considerations. First was expedience. The authors had experience in fitting analytic functions but none in fitting tabulated functions. Next was speed of calculation. Another was the hunch that one could do good *relative* photometry within an image even with an imperfect approximation to the point spread function, since the systematic error made by adopting arising from the approximate nature of the model should be the same for all stars. This is not strictly correct, since the profiles of bright stars are weighted by the photon statistics of the star itself, while the profiles of faint stars are weighted by the photon statistics of the sky. Results from a limited set of experiments would indicate that the associated scale errors are less than 0.01 magnitude per magnitude.

The reader will note that the free parameters in the above equations are not the same as those in the output parameter list. The transformation to magnitudes, FWHM, axis ratio and position, the external representation of the parameters, is made only during when writing this list to disk. For all other purposes, DoPHOT uses as its internal representation of the parameters the quantities indicated in equations (1) - (3). If de-

sired, output files can easily be generated that preserve this internal DoPHOT parameter representation (see Appendix A). Moreover, error files which contain the square roots of the diagonal covariance matrix elements for these variables can be generated as well (see Appendix A).

7.3 Phantom Stars and the Noise Array

The most obvious shortcomings of adopting an analytic, as opposed to empirical, point spread function is the large residuals one gets from the best fitting model. Since DoPHOT makes multiple passes through the object-subtracted image, there is a danger that the program will trigger on a positive residual and identify a spurious "phantom" star. Phantom stars may be checked for in the "synthetic" image, generated by subtracting the object-subtracted image from the original image. Another useful diagnostic is to plot the (x, y) positions of objects using a plotting program, keeping an eye out for unusual grouping of faint objects around bright objects or at the edges of obliteration boxes.

DoPHOT uses its noise array to avoid such phantom stars. Whenever an object is subtracted from an image, a user specified fraction of the subtracted signal is added, in quadrature, to the noise array. The larger one makes this fraction, the less likely one is to detect a spurious object. In increasing this fraction, one pays two penalties: one is less likely to detect faint stars in the vicinity of bright stars and the fits to objects which are detected in the vicinity of another object will be more uncertain than they would otherwise be.

While this scheme seems to work reasonably well at the centers of images, phantom stars still appear on the peripheries of bright stars when the actual stellar profile has broader wings than the analytic model. Ideally, one ought to try an different analytic approximation. Short of that, one can adjust a factor which expands the shape parameters used in augmenting the noise array by a user specifiable factor, or, if the stellar objects are well-sampled, rebin the data to achieve object profiles with FWHM of about 3 pixels.

7.4 Star/Galaxy/Double-Star Classification

When a subraster of pixels gives a significant signal when tested against the stellar

model filter, one doesn't know whether one is seeing the light from a single star, from a galaxy, or from two or more stars which lie relatively close to each other.

DoPHOT attempts to deal with this classification question economically, taking advantage of the fact that it must determine the shapes of objects to determine the parameters associated with a "typical" star. When those shape parameters are significantly different from the typical image (by a user specifiable amount) and when the sense of that difference is that the area of the associated footprint is larger, DoPHOT declares the object to be "very big."

The significance of the difference between the shape parameters for an individual object and those for a "typical" star depend first, upon the errors in the fit for the individual star and second, on the star-to-star scatter in those parameters. Given a perfect instrument, and an ensemble of stars of the same brightness, one would expect these to be the same. For many instruments (*e.g.* wiggly CCDs in fast beams) they are not. DoPHOT computes a star-to-star scatter in the stellar parameters that weights the bright stars, those for which the accidental errors from the fit are likely to be small, more heavily. The scatter for each shape parameter is added in quadrature to the uncertainties for that parameter obtained from the fit to an individual star. The significance of the difference between the measured and expected shape parameters is computed taking this as the expected difference.

An *ad hoc* (and not at all satisfying) modification of the above scheme is that the user is allowed to specify a minimum scatter expected in the shape parameters. This keeps stars on the second pass through the data from being classified as galaxies when the first few stars have an unusually small scatter.

When an object is found to be "big," DoPHOT attempts to fit two typical stellar profiles to the subraster. The goodness of fit parameters return from the fits to the single object and the two typical objects are compared, and based on a user specifiable parameter a decision is made on its classification. If the object is double, it is "split" and two entries are made in the object list. These entries are then subject to further testing and splitting on subsequent passes through the image. Galaxies are object type 2, while siblings of split stars are object type 3. Objects of type 3 can be reclassified as galaxies

but never as single stars of type 1.

At high galactic latitudes, in regions of low star densities, the user will want to adjust the parameter which control the galaxy/double-star decision to favor galaxies. At low galactic latitudes and in star clusters the decision should favor double-stars. The synthetic image is useful for diagnosing how this parameter should be set.

One difficulty with modeling objects is that at progressively lower signal-to-noise ratios one can support fewer and fewer free parameters. Since the fitting in DoPHOT is carried out iteratively, the symptom of too low a signal-to-ratio is a failure to converge. There is a user specifiable signal-to-noise limit such that only objects with higher signal to noise are fitted for shape parameters. Objects which are fainter than this are classified as object type 7 in the output list and are only fit for sky, (x, y) position, and central intensity while adopting the current best estimates of the shape parameters as fixed.

At high galactic latitudes some large fraction of such faint objects may nonetheless be galaxies. To provide at least some handle on whether or not such objects are galaxies/double-stars or single stars, an "extendedness" parameter is computed for every object. Using the typical stellar parameters for an initial guess, the fitting program is allowed to determine the first step toward improving the shape parameters. The change in the goodness-of-fit parameter predicted by its second derivative matrix is then estimated. If the sense of the change is to decrease the area of the footprint, this change is reported as a negative number; otherwise it is reported as positive.

Limited tests indicate that while this quantity does help to separate stars from galaxies (and from globular clusters in nearby galaxies) its value is not independent of the brightness of the object. The user is encouraged to investigate this further, perhaps by comparing images of a given field obtained in very different seeing conditions and for which the user can readily classify objects in the better-seeing data that are harder to classify in the poorer data.

7.5 Completeness and Faint-End-Bias

The model that is used as the filter for measuring the signal-to-noise ratio for a potential object is identical to that of the typical stellar profile used for fitting the object. It should not come as a surprise, therefore, that when the numbers of objects identified

is plotted as a function of apparent magnitude obtained from the fit, one finds that it drops to zero quite abruptly. The detection and the measurement are carried out consistently.

The naive user might be deceived into thinking that the sample of detected is remarkably complete. It isn't. Consider several stars for which the expected signal-to-noise ratio is exactly the limiting value. Some of those will be superposed on positive noise fluctuations, and will be detected, with measured fluxes brighter than their true fluxes. Others will be superposed on negative noise fluctuations, and missed. Not only will the sample be incomplete, the estimated magnitudes will be biassed by an amount which depends upon the logarithmic slope of their number-magnitude distribution and upon the limiting signal-to-noise ratio.

7.6 Model Sky and Average Sky

For every star in the object list, DoPHOT produces an estimate of the sky brightness. It is therefore possible to construct a "model" of the variation of the sky brightness over the face of the chip. This model is then used to *estimate* the sky value when a potential object is filtered through the typical stellar profile.

If the object passes through the filter using the model sky, it is tested once again, this time using a weighted average of the sky computed from the fit subraster. This second test avoids spurious detections where the background level is changing in a way which is not modeled.

The model for the sky need not be a constant. Typically it is a plane, which allows for a small gradient across the field. An option which has proved quite useful is modeling the sky as a "Hubble" profile plus a constant. In fitting this model (and the much simpler plane) the data points are the values of the sky derived from fits to individual stars. The "Hubble" profile requires a relatively large number of points (of order 50) to converge.

7.7 Warmstarts

There are a variety of circumstances under which it is helpful to give DoPHOT a starting list of objects. For example, having done a first pass on the data, one might want to try lowering the threshold without having to start over from the beginning. Or

might want to insert an object by hand into the list, or to change the parameter which controls whether objects are stars or galaxies. In the latter case, if one thought that DoPHOT had erred on the side of classifying galaxies as double-stars, one would need to remove the offending type 3 objects before re-starting the program.

In galaxy fields, we have found it useful to 'train' DoPHOT on preselected stars so it doesn't misclassify bright, narrow galaxies as stars before it reaches the dimmer stellar objects.

7.8 Obliteration

Bright stars present a several problems for automated photometry. Since detectors often go non-linear and then saturate at high surface brightness levels, it is difficult to subtract badly overexposed stars from the image. Rather than try to fit and subtract such an object, DoPHOT allows for the automatic the excision of a rectangular subraster around bright objects.

The user has considerable (perhaps too much) control over this process. There is a user-adjustable parameter which places a limit on the allowable number of saturated pixels. Objects with more than this number are excised from the image, "obliterated." There is also a user specified upper limit to the central intensity of an object. If the fit to a subraster gives a larger value than this, the star is excised.

The size of the region excised will depends upon the best available estimate of the central intensity of the object. If this has not been fit for, it is taken to be proportional to the number of saturated pixels, with the constant of proportionality specified by the user. The surface brightness level down to which pixels are excised is also user specifiable.

In the event of exceedingly bright objects, or diffraction spikes from such objects, the user can specify a region to be excised by entering an object of type 8 in the input object list and "warmstarting" the program. The test image provides an example of how the user may explicitly obliterate regions of an image. The user may choose to excise rectangular regions of the image or elliptical ones.

7.9 Trouble

A number of circumstances arise where, try as it might, DoPHOT cannot come up

with magnitudes or shapes for objects. The most common of these is when there are too few samples to do an adequate job of fitting the model. There are two user adjustable parameters which specify the fraction of the fit subraster that must be present for an object to be fit. If too few pixels are present for the shape of an object to be determined, it is classified as object type 5. If an attempt to determine the shape of an object fails to converge, it is classified as object type 9.

If an attempt to determine the magnitude of an object using the typical shape fails to converge, the object is classified as type 4. If too few pixels are present for the object to be fit using the typical shape parameter, the object is declared object type 6. Objects of this type are *not* subtracted from the image – they are said to have been "deactivated." They are left in the object list so that a record is left that something out of the ordinary has happened. If the signal-to-noise ratio of an object which was previously identified is found to be less than the user specified value for the identification of objects, the object is likewise classified as type 6, and "deactivated."

7.10 Internal Parameter Representation

DoPHOT fits for the parameters ${\sigma_x}^2$ and ${\sigma_y}^2$ rather than σ_x and σ_y so that the program can recover more easily from iterations which might otherwise make σ_x and σ_y negative. The quantity σ_{xy} is measured in *inverse* square pixels so that round images produces parameters which behave well. When fitting models, DoPHOT divides the central intensity and sky values by 100 to avoid overflows.

When fitting analytic objects, DoPHOT treats the logarithm of the central intensity as the free parameter; this allows for more robust convergence of single stars and galaxies and recovery from overzealous attempts to decrease the brightness of one of the two elements of a double star for double objects.

8. GUIDELINES FOR PROGRAMMERS

8.1 DoPHOT 4.0 Improvements from Fortran

DoPHOT is now in C, but it was converted modularly from the Fortran 77 subroutines. Thus, perhaps to a C expert's chagrin, it reads and executes just like the Fortran versions. And just like in the Fortran version, comments explaining the logic within the code are sparse. However, there are improvements over the Fortran version which have made the transition worthwhile, at least for the person who converted it.

8.1.1 Obsolete diskio.c Routine Replaced with CFITSIO

No more diskio.c that uses obsolete (though admittedly still functional) read write functions. The tradeoff is dependence on the CFITSIO library. We are sorry to those who hate when software comes with "and add this library/package strings, but we were disinclined to reinvent the wheel when such a nice round one was available. If you are desperate for a version without CFITSIO, email rsobel@mit.edu for an old, but clean copy of Dophot with limited features but no CFITSIO dependence. However, CFITSIO is very nice, and we encourage you to just get it working and use the current and maintained DoPHOT version with all the features.

8.1.2 #defines confined to structs/tuneable.h

All #defines (in Fortran, parameter) are confined to the header file **structs/tuneable.h**. No others hide throughout the program. To developers, please keep it that way.

8.1.3 Commonblocks Replaced by Structs

Common block variables are now anonymous structs. While these might seem to be no better, there are subtle improvements which we will lay out in this section.

All struct definitions exist in a subdirectory called **structs**. If a function uses struct "thisone", a dedicated file with the struct called "thisone_struct.h" exists in the **structs** subdirectory, and all routines that use that struct must "#include thisone_struct.h" at the top of the routine. This structure makes finding all routines that use a given struct very easy because the word thisone_struct will appear in every routine where the struct is used, and nowhere else.

Additionally every time a struct variable is used in a routine, it is explicit which struct it belongs to, the variable type, and whether or not the variable is changed in the

routine. This is done differently for lower level and higher level subroutines.

For lesser routines, the struct variables are renamed to a convenient local variable name at the beginning of the routine and then reassigned at the end if changed. This makes both the type explicit and if the struct variable is changed in the subroutine. And when time (and gold and pixie dust) allows we will go through and make a comment in the beginning of each routine and corresponding header file stating in words which variables are changed.

For higher level routines (**shape, isearch, improve, dophot**) where the struct variables are changed frequently in the routine and their updated values are needed by the subroutines called therein, the variable types and the fact that they are used in the routine are mentioned in an appropriately commented area at the top of the routine before the routine 'substance' begins. Assume the struct variables are changed throughout all higher level routines unless otherwise stated.

Tuneable variables, tune##_.variable, are ALMOST never changed except in **tuneup.c** where they are defined from the parameter files. The exception is in the routine **bestab.c** where there are angry comments letting developers know that this is happening and is supposed to happen, although its uncharacteristic of the tunable variables and will probably be changed at some point for consistency's sake. Changes and lack of changes to the tuneable variables aren't a new feature of the C version, but worth mentioning since it is angrily commented in the code.

All struct *arrays* are actually pointers. 2d arrays are pointers to pointers. Their length is not fixed in the struct definition itself, but rather malloced in each run in **tuneup.c**, or at the very beginning of **dophot.c**, even if the theoretical length of the array is fixed. These ARE THE ONLY TWO PLACES WHERE MEMORY FOR STRUCT ARRAYS IS ALLOCATED. Developers, please keep it that way. The strict locations for memory allocation should make it easier when swapping in PSF models, or debugging segfaults generally. Also, here is a good place to mention for those who care, the code is Valgrind clean.

In a future version we should convert the anonymous structs into proper C structs. Their current shy nature is a hold over from the conversion from Fortran 77, which hap-

pened subroutine by subroutine, and thus required that the C structs were able to communicate with the Fortran common blocks.

8.1.4 No EQUIVALENCES

The Fortran version of DoPHOT had EQUIVALENCE statements which took an array of two short integers (integer*2) and treated them as one float (real*4) to be passed into structs (common blocks) that accepted floats. This was useful for passing around two numbers within existing structs, but is not really supportable (read: trefe). Remnants of the equivalences in C (bit math) are now gone. Short ints never become floats unless recast as such.

8.2 DoPHOT 4.0: Things Unimproved from Fortran

For the sake of fairness, we should lay out those aspects of the code that are as yet unimproved from Fortran 77. For the first, we made a time call. For the second, we learned better, but too late.

8.2.1 Commenting

The Fortran code logic had very little commenting. The C version has the same logic and thus the same lack of commenting. The few bits which are new in C, rather than just translated, *are* commented.

8.2.2 Naming Conventions

Fortran naming conventions and capitalizations are carried over from the old code, especially in the smaller subroutines. C conventions are not used. This flaw will be slowly rectified as new versions are released.

8.2.3 Pointer Passing

Fortran passes everything by pointer, so so does the C version of DoPHOT, even if the variable in question is not changed in the routine. While this flaw has been fixed in a select few subroutines, it should be fixed more globally in a subsequent version to make it more explicit which variables are and aren't changed in routines.

8.3 Keeping the Point Spread Function Hidden

When DoPHOT was written, a deliberate effort was made to keep the details of the point spread function hidden from most of the higher level subroutines. It was hoped that this would make it possible to use DoPHOT with very different point spread func-

tions. The authors were only partly successful in approaching this ideal. Some subroutines work even when the number and order of the parameters are changed. Others work if one changes the dimensionality of the parameters (switching, say from central intensity to log central intensity), but not if the order of the parameters is changed. Still others require intimate knowledge of the details of the PSF (e.g. the subroutine which makes a first guess of the double-star separation when fitting two "typical" stars to a single "big" image).

Below is what you need to know and do when you make a new model, referred to here as NEWMODEL. This guide is somewhat step by step, but will still require some user knowhow and should not be attempted by the novice. Moreover, this guide is good only for models which add additional parameters to the existing core 7 (SKY, X position, Y position, Peak Intensity, sigma_x, sigma_xy, sigma_y) of the pseudogaussian model PGAUSS supplied with the code and/or change the space in which they are fit (to log space etc.). If one wishes to fit a model with *fewer* parameters or change the order or nature of these core 7 parameters, this guide will be helpful but incomplete. However, we feel that a sky value, position, intensity, and some elliptical shape parameters are necessary to most models and thus this guide will be helpful for many users.

The instructions will start from the most seeming peripheral and end with the most core. This is by design to maintain usability of the code. If you do get a new model working, consider it to be worthwhile,and are willing to share, please submit your proposed updated to github!

1.) Change **params_default_c** to reflect the newly available model type. Keep the default as 'PGAUSS'. Note: DoPHOT only checks for model type by reading the first five characters of a model name, so make these characters unique to your model, (i.e. EXTPGAUSS, not PGAUSSEXTENDED). The relevant line in the **params_default_c** file should now read something like:

   PSFTYPE = 'PGAUSS' PSF type: (PGAUSS, GAUSS, NEWMODEL)

2.) Update **tuneup.c** to recognize NEWMODEL as a new model type. Additionally, make certain that it updates tune4_.npar and tune4_.nfit2 to the appropriate number of shape parameters of your model, as well as tune15_.ava[] to the appropri-

ate average shape values. Last, ensure that the appropriate number of tune15_.acc values are drawn from the parameter files. Remnants of a SERSIC model with 8 shape parameters exist in the code, though the model itself was deemed far less worthwhile than the PGAUSS model and therefore abandoned. The SERSIC model switches can be used as a guide within the code for necessary updates for new models.

3.) Add the header file for the NEWMODEL to **dophot.c**. The header file must match the PGAUSS header file **structs/pgauss.h** exactly except for the defines and the name of the relevant '2d' and 4d' functions. The '2d' function is for isolated objects. The '4d' function is for double stars. These functions needn't have the same form.

4.) In **dophot.c** add the model to the list of options which can be switched by the model flags. This statement will tell DoPHOT to use your model everywhere an analytic model is called for. If newmodel2d_ and newmodel4d_ are the names of your analytic models for single objects and double stars that you had declared in the header file, your statement should look like:

```
if (strncmp(flags[0], "NEWMODEL", 5) == 0){
    model2d = &newmodel2d_;
    model4d = &newmodel4d_;
}
```

5.) Modify the files which generate INCOMPLETE, COMPLETE, and SHADOW outputs to display the fits of your new variables (if there are new variables). These files are **sumout.c, stdotpt.c** and **shdout.c** respectively. The type of output should be toggled by a switch on the flag, and we recommend that the new variables are displayed immediately after the sigma or semi-major and tilt fit variables where appropriate. If you follow the next steps, your new variables will likely be stored in STPARR[7-10] (or more) in this routine, so feel free to output these variables if you're intending to follow the remainder of this instruction guide. (Actually struct 2darray starlist_.starpar contains the fit parameters for all stars. The 1d array corresponding to the parameters for each output star are passed to the output routines as STPARR.) Again, look for the SERSIC example for a guide within the code.

6.) Modify the files which read INCOMPLETE and COMPLETE warmstart files, namely **suminpt.c** and **stdinpt.c**. They should read files of the type you just designated as output and put the new variables in STARPAR[7-10] (or more) as appropriate.

7.) Write the model function **newmodel.c**. DoPHOT uses models in two ways, for fitting shape parameters to found objects and for subtracting off the models of objects once appropriate shape parameters are found. Therefore each model must take four parameters: 1.) a pointer to an x,y position of a pixel in the field at which the model value is to be computed, 2.) a pointer to the array of fit parameters, 3.) a pointer to the array of derivatives of the fit parameters which are only needed for the fitting routine, and 4.) a pointer to an int, fitcall_ptr, which can be used to tell the function if it is being called by the adding or subtracting routine **addstar.c** or the fitting routine **chisq.c**. The function must return the value of the analytic function at the position of the pixel passed in the first parameter. There is a fifth passed parameter, m_ptr, which remains for historical reasons but which has no utility in any of the supplied model routines. With these parameters passed to your fitting function, you should be able to write the analytic model of your choice. Follow **pgauss.c** as a guide for the set up of both the one star and double object functions and necessary parameter updates. The fit parameters 'a' passed to the routine are taken from the struct starlist_.starpar or fitarrays_.a depending on if the analytic model is being subtracted/added to the image or fit by **chisq.c**. If you have followed the guide so far, your additional parameters should appear there after the core 7 parameters of the pseudogaussian model (beginning at a[7]). The same goes for the derivative matrix 'fa' which is taken from the struct fitarrays_.fa.

8.) Update the **onefit.c** and **twofit.c** chisq wrappers for the fit of the analytic model. The onefit.c wrapper was created to support the fitting of models in a different parameter space than that in which they are subtracted from the images and displayed to the user. Twofit.c was created to minimize the number of free parameters used in the fitting of double objects, but it also allows the same parameter space switching for fits. The one star fitting functions and double star fitting functions needn't be the same. Regardless your model type, you should check these wrappers

and make sure they are still valid your model. If you have chosen to fit your model in a different space than that in which you add and subtract it from the image, you will have to change your variables to the fitting space and update your covariance matrix back from that fitting space after the fit in these two functions. WARNING: The working covariance matrix in the code, while referred to as a covariance matrix, is not a covariance matrix at all. Along the diagonal of the matrix are the square roots of the variances. On the off diagonals are correlations. This convention was adopted to avoid extremely small numbers when in the matrix. However, it does mean that the matrix will have to be converted back to a traditional covariance matrix to switch parameter spaces. The EXTPGAUSS model, which is a pseudogaussian function where $\beta 4$ and $\beta 6$ are free to vary, but are fit in log space, can be used as a rough guide for altering **onefit.c** and **twofit.c** if the new parameters of your function vary in an alternate space from that in which they are subtracted.

9.) Check **shape.c** to make sure that the starting parameters given to the fit of your function are reasonable. On each first pass for a newly found object, **shape.c** will attempt to fit a PGAUSS model to each object. This PGAUSS fit will inform the sky, intensity, x, y, and three sigma values of the starting parameters for the NEWMODEL fit, with no alterations. Only if the PGAUSS fit succeeds will **shape.c** try a NEWMODEL fit. If the NEWMODEL fit succeeds, the NEWMODEL is kept for that object. However, if the NEWMODEL fit fails, the object is flagged to use a PGAUSS model from then forward so as to improve the object subtraction and lessen the chances that DoPHOT detects spurious objects in the residual image on subsequent passes. Therefore, if your point spread function is sufficiently different from gaussian, or if your parameters are significantly different from a gaussian's, you may want to enter an exemption to this preliminary PGAUSS fitting for your new model, or adjust the parameter estimate between the PGAUSS output and the NEWMODEL input.

10.) Add your new model, **newmodel.c** to the **Makefile**. Compile the code.

11.) Test that the PGAUSS model still works on the **verif_data** images. You should still get exact results.

12.) Test your model.

Troubleshooting. If you run into a segfault, check NPMAX in **structs/tuneable.h** to ensure that it is greater than or equal to the number of parameters in your new model. Check that MMAX is 3+ NPMAX. If either of these conditions does not hold, fix them, remove all .o files and remake dophot.

8.4 "Typical" Parameters and the Shadow List

The output object list uses the "typical" shape parameters for the listed stars. But those typical values are computed from actual values determined from the individual stars. A "shadow" list with these individual values is stored in memory, and is used to compute the "typical" values.

The errors in the shadow parameters are also stored in memory. The simplest "typical" value is a weighted mean of the parameters. Some versions of DoPHOT (but not this one) allow the shape parameters to vary with position on the chip, in which case the shadow list and the associated errors are used to constrain models for that variation.

8.5 Magic Values and the Noise Array

DoPHOT works on an image which is stored in a (as of 4.0) 32 bit signed integer array. The noise is stored as a variance (the square of the noise) in a 32 bit integer array. In 3.0 and earlier the data was stored as a 16 bit signed integer array. In a still earlier version of DoPHOT, the noise itself was stored as a 16 bit integer, but the repeated additions, subtractions and square roots had round off errors which led to negative variances.

In the routine which computes the noise array, image pixel values are compared to user-specified upper and lower limits. When the data value lies outside this range, the variance for that pixel is set to a "magic" value, $2^{31} - 1$. The routine which extracts a subraster from the image for fitting with an object model does not include pixels for which the noise array has this magic value. If the subtraction (or addition) of the model for an object ever causes the data value in the object-subtracted image to be smaller than a #defined OBLITVAL set to -60000 or greater than the user-specified max obliteration value CENTINTMAX, the variance in the noise array is likewise set to this magic value, and the pixel is henceforth ignored. When a subraster around a sat-

urated star is "obliterated," the associated elements of the noise array are likewise set equal to this magic value.

8.6 First Guesses

DoPHOT uses an iterative nonlinear weighted least squares fitting for estimating the parameter values associated with the various models it uses. The particular nonlinear least squares scheme converges rapidly if the current guess of the parameter values is close to the best value. But one needs a good first guess – a bad first guess will leave the routine through wandering the valleys of the goodness of fit parameter, searching for a minimum. For this reason the shape parameters for individual objects are saved in the above mentioned "shadow" lists, for use as starting values on subsequent fits.

The initial guesses which have proved the most difficult to obtain and the least satisfying, both esthetically and as measured by ultimate convergence, have been the choices of the separation and central intensity ratio for the two components of a double-star. One knows the footprint of the typical star, the "oversized" footprint of the object to be split, and the positions and brightnesses obtained, respectively, using the typical footprint and the best fitting footprint. The initial brightness ratio is arbitrarily taken to be 2, with the brighter object taken to be closer to the position obtained using the typical profile.

The separation vector is taken to be along the position angle of the oversized footprint. While this is reasonable for a typical spherical footprint which is circular, one could imagine a situation in which the "typical" footprint is elliptical and the "oversized" footprint is more nearly circular, but with its elongation perpendicular to the separation vector. Modifications which would help in this situation have been suggested but have not been incorporated into the present version.

8.7 Nonlinear Weighted Least Squares

At the core of DoPHOT lies an iterative nonlinear weighted least squares routine in which Bevington's (1969) implementation of Marquardt's algorithm has been imitated. There are user-specifiable parameters which control the convergence criteria, expressed either in terms of relative accuracy, suitable for intensities and some shape parameters, or absolute accuracy, which is more appropriate for positions. The user can also specify

extreme upper and lower bounds beyond which the routine declares itself to be hopelessly lost in parameter space and gives up. This last feature permits the DoPHOT to recover relatively gracefully from bad initial guesses and badly confused object configurations.

8.8 Data for PSF Fitting

Since pixels can be excised from the image for a variety of reasons, not every pixel contained within the requested subraster is passed to the fitting program. The x and y coordinates for each usable pixel, computed with respect to the center of the subraster, are passed to the fitting program, along with the data value for that pixel. The fitting program was written to fit a scalar function of scalar variable, both of which are represented as 32 bit floating point numbers. Since the subraster is centered on a pixel, the pixel coordinates can be represented as two "short" 16 bit integers. These numbers are recast to floats for fitting, and NO LONGER EQUIVALENCED in 4.0

8.9 Switches versus Custom Versions

A limited number of switches are provided which allow the user to choose among image formats, among parameter output formats, and among models for the background sky. Switches both complicate and slow down the program, but they avoid the need for multiple versions of subroutines. As a rule of thumb, switches should be incorporated only when one expects them to get frequent use. The authors suggest that the user modify the program rather than incorporate new switches.

8.10 Variable Point Spread Functions

There are versions of DoPHOT which allow the PSF to vary as a function of one or both of the positions within an image. The subroutines in DoPHOT are small and modular. At most four of these must be changed in order to accommodate a variable PSF. Since the model for the variation of the PSF across an image will depend upon the details of the instrument used, the present version of DoPHOT incorporates only the simplest of models, a constant PSF.

8.11 Subtracting a Smooth Image

There are cases in which the user will want to subtract a smooth model for the background light, but to keep the photon statistics as they were prior to subtracting the

smoothed model. There are several ways in which one might do this, but one method that has been somewhat successful involves subtracting the model of the smooth background as part of the preprocessing, but then using that smooth model to insure that the photon statistics of the noise array are correct. One could imagine running DoPHOT on an image, finding the brighter objects, smoothing the object subtracted images, subtracting them and iterating.

8.12 Analytic/Empirical PSF

It is the authors' suspicion that the greatest losses involved in the use an analytic, as opposed to an empirical, point spread function are associated with the object-subtracted image. If the residuals were smaller, one could obtain more complete identifications, and better photometry and classification.

In the Version 3 release, DoPHOT was extended to include the option of using an empirical PSF. Tests show that the residuals after subtraction are dramatically smaller, and the number of 'outliers' in photometry of severely crowded fields is significantly smaller. A fuller account of this implementation is given in section 11 (Enhancements).

9. MONITORING OUTPUT AND DIAGNOSING DIFFICULTIES

9.1 Monitoring Output

The single most useful thing one can do to get a qualitative feel for how DoPHOT is doing is to create a synthetic image by subtracting the object-subtracted image from the original image. Displaying this will show whether DoPHOT is successfully classifying galaxies and double-stars, and whether spurious objects are being detected.

Another useful check is to compare outputs obtained for two separate images of the same field. One might ask how well the fit magnitudes agree, after adjustment for possible variations in transparency and correction to an aperture magnitude system, how well the classifications and positions agree, and which objects were detected on one frame but not on the other. Comparisons such as these can profitably be done by hand, but can also be done using matching programs which may be available from the person in the office next door.

Much can be learned just from examination of the output object list. The scatter in the difference between aperture and fit magnitudes should be consistent with the quoted errors. The object classifications should make sense (*e.g.*, in an open cluster most of the objects should be single stars and not galaxies and split stars, while in a more crowded globular cluster field, one might find a number of double stars, and in a high-latitude field galaxies should be common). The uncertainties in the fit magnitudes should be small for bright stars and larger for faint stars. They will be larger for stars which lie near other stars.

One can produce a large "log" file records DoPHOT's progress in identifying objects, estimating positions and magnitudes and shapes, classifying objects, and determining typical shape parameters and sky values. The output of the "log" file can be easily directed to the user's terminal or window. Since the entries in this file are more or less self explanatory, no attempt will be made here to describe the (many) possible entries in the log file. One useful quantity reported in the log file is the number of pixels used in fitting a model. If this is very much smaller than the number computed from the dimensions of the fit subraster, it may be that the object is at the edge of the image, or that a nearby object has been "obliterated."

The user would be well advised to look at the output parameters file to make sure that the values used were those intended. This file also serves as a record of the reduction of that image with one caveat described in Appendix A. With the output parameters file, the user can *precisely* reproduce that reduction.

The user is also encouraged to reduce some image that has been reduced with some other program, perhaps borrowing one from a colleague who wouldn't mind having his reduction double checked.

10. KNOWN BUGS AND SHORTCOMINGS

10.1 Maximum Number of Objects

The maximum number of objects is compiled into DoPHOT through a #define in **structs/tuneable.h** *The user must recompile and relink all* DoPHOT *modules after changing any #define s* as described in Appendix A. At present DoPHOT does *not* check this maximum before adding objects to the object list.

10.2 Initial FWHM Limitations

When DoPHOT encounters objects on its first pass through the data, it must decide whether they are single stars, galaxies or double-stars. This decision is based on the user supplied full width at half maximum. If this number is too small, DoPHOT will decide that *all* the single stars it encounters are galaxies and double-stars. Since it encounters no single stars, it cannot update its notion of what a typical star looks like and single stars are classified as galaxies and double-stars on all subsequent passes through the data as well. Thus, the user's best estimate of the FWHM is automatically multiplied by 1.2 to ensure that DoPHOT's initial guess of the shape parameters are somewhat overestimated. Note that the FWHM provided by the user, *not* the inflated value, is saved in the output parameters file.

10.3 Lost Siblings

DoPHOT does not maintain pointers linking those objects which are double-star siblings. As a result, the values obtained at the moment when the stars are split are probably *better* than those ultimately reported. When one sibling of a pair is encountered on a subsequent pass through the data, it is fit as a single star with the "typical" shape parameters. Since it's sibling has been subtracted, the noise in the nearby pixels is higher than it would otherwise be, increasing the formal uncertainties from the fit. But since the sibling is *not* being fit simultaneously, the error estimates for the parameters do not take into account the covariance with the estimates for the sibling. This second effect would underestimate errors.

10.4 No Longer and issue: Maximum Image Size

We assume that all modern computers can dynamically allocate memory. If your computer cannot, stick with DoPHOT 3.0 in Fortran. The image size is read form the

image header and the memory for the image is dynamically allocated to the correct size.

ACKNOWLEGEMENTS

The authors gratefully acknowledge the help of colleagues who generously agreed to try using early versions of DoPHOT, tested it on a variety of different kinds data, reported the results of their experiments, and suggested a great number of the improvements which have been incorporated into the present version: John Caldwell, Kem Cook, Eileen Friel, Wendy Freedman, Doug Geisler and John Tonry. They also thank John Tonry for his routines to read "disk fits" files, even though they are no longer implemented. DoPHOT was originally written with the support of the U.S. National Science Foundation through grant AST83-18504. DoPHOT 4.0 modifications were supported by a NSF Graduate Research Fellowship to Rebecca Levinson.

11. VERSION 2.0 and 3.0 ENHANCEMENTS

Since its first release (version 1.0), several enhancements have been incorporated in succeeding versions 2.0 and 3.0. These are in addition to bug fixes and other minor changes that have improved the running of DoPHOT. We have thus far been able to keep DoPHOT reverse compatible: specifically DoPHOT should run exactly the same way (aside from bug fixes) with your modified parameters file, with the only change being to point to the new DEFAULT parameters file and, as of 4.0, a slight change in how empty parameters should be designated in the modified parameter files. Details in the Appendix.

11.1 Fixed Position Warmstarts

An option introduced with version 2.0, called a "fixed position warmstart" permits a user to specify the x and y coordinates of stars. The photometry reported for appropriately flagged stars is then the result of a two parameter fit (sky and central intensity), with the two position parameters fixed at the specified values. This option might be used to advantage if one had a deep frame taken in good seeing, and wanted to do photometry (or set upper limits) on less deep images obtained at a different epoch, or an image obtained with a different filter.

While we had long considered this option among those we had hoped to implement, we were spurred to do so by the success of D. Bennett of the Lawrence Livermore Laboratory's MACHO project. At the 178th meeting of the American Astronomical Society in Seattle, Bennett reported that his implementation of the fixed position idea in a program called SoDoPHOT produced substantial improvement in accuracy over that obtained using 4 parmeter fits on the less deep frames. Our implementation is within `DoPHOT` itself, allowing for new stars (and galaxies) to be found in addition to those specified on the input list. Since the positions for the newly found objects are not specified, the photometry is based on four parameter fits in IMPROVE.

`DoPHOT` uses the image type associated with each star to distinguish between stars for which 4 parameter fits are to be carried out and those for which the positions are fixed. In the latter case, the image types in the input file are augmented by 10. Hence an object of type 11 is a star for which 7 parameter fits are to be carried out (to deter-

mine the shape parameters), but for which the magnitude is to be determined from a 2 parameter fit. An object of type 17 is too faint for 7 parameter fits. The meanings of all image types are summarized in Appendix C.

The input list is much like the input to an ordinary "warmstart", but with the image types augmented by 10. Since the deregistration between images is likely to be more than 1% width of a typical stellar image, the user will need to transform the positions from his starting list (henceforth the "template") to those of the "program" image. For this purpose one needs to obtain positions of the brighter stars on the program image. A working example of a program to do this (`OFFSET`) is given in the subdirectory "transform", described in more detail below. It uses output from a standard `DoPHOT` run on the program image in conjunction with the template positions to produce a warmstart file. In addition to transforming the coordinates from the template system, a crude photometric transformation is also made. Preliminary tests indicate that `DoPHOT` does run faster and provides somewhat better photometry reproducibility when the coordinates are fixed. Reports of additional tests by `DoPHOT` users would be most welcome.

When the user requests a `DoPHOT` run with fixed positions, the program goes through the first threshold differently than if the positions were not fixed (see the crypto program in Section 3 for reference). Specifically, rather than run `ISEARCH`, `DoPHOT` goes directly to `IMPROVE`, calculating only two-parameter fits for the appropriate objects read in from a template object list. It is a good idea at this point to have at least rough estimates of the offsets between the template and program photometry results. Nevertheless, if these estimates are poor, it is the job of this first pass in `IMPROVE` to improve the sky values and central intensities of each star using some initial guess of the shape parameters. Then, `DoPHOT` re-enters the "standard" reduction stream in `SHAPE` where full seven-parameter fits are performed on stars will sufficient S/N. Additional passes through `DoPHOT` allow the program to find stars that – for whatever reason – may not be in the template list. The only difference between these subsequent passes and "normal" passes in `DoPHOT` is that the positions of most objects from the template list remain fixed and only two-parameter fits (for sky and central intensity) are performed in `IMPROVE`.

11.2 Running Fixed-Position Warmstarts

Fixed-position warmstarts are as yet untested with version 4.0. We are happy to receive reports.

Within the **verif_data** directory are a number of files designed to allow the user to test the fixed-position option in DoPHOT. This section provides step-by-step instructions of how to run these tests and make sure that DoPHOT is running correctly on your machine. For simplicity, the directories and file names will be referred to in their !TEX encoding = UTF-8 UnicodeUnix versions; it should be obvious how to translate these to the corresponding VMS names. The basic aims of the test described below are (1) reduce two frames of the same field obtained on different nights and exhibiting large positional offsets, (2) make a "template" file that will serve as the input to the fixed-position version of DoPHOT, (3) reduce a frame using this template file. *Be sure to copy all files from the* **verif_data** *directory to the* **working_data** *or else you will overwrite the files supplied with* DoPHOT*.*

Two disk-fits files are supplied in the **verif_data** directory called **image_in_template** and **image_in_program**. Assuming you have now compiled DoPHOT 4.0 (see Section 4), simply run the program twice to reduce these frames. The input parameter file for the first image is (naturally) **param_in_template** and for the second image **param_in_program**. The results of these runs will be two files (**obj_out_template1** and **obj_out_program1**, respectively) that correspond to full-blown, standard DoPHOT reductions of these frames. You should check to ensure that these files are identical to their counterparts in the **verif_data** directory as described at the end of Section 4.1. You might also want to look at the various output (subtracted) images generated by these (and subsequent) DoPHOT runs. These reductions will serve as benchmarks for the steps that follow. Note that if all went well, 2382 stars were found in the template image, and only 1838 in the program image. Both images are of the same field (NGC 2031 in the LMC), but the seeing is somewhat worse on the program image.

The whole point of the fixed-position option is that one can, and should use positions obtained from a deeper or better-seeing frame to reduce any given image. Although we fully reduced the program image above, we really didn't have to. All we need are enough positions to be able to precisely transform the template coordinates to the

program positions. In general, only a few dozen stars are needed to do this, not 1838. Thus, a more efficient method of running the fixed-position option is to reduce the program frame "once over lightly". To illustrate this, run DoPHOT 4.0 using **param_program1** as the input parameter file. If you look at this parameter file (and study the meanings of the relevant parameters in Appendix A) you will see that this reduction of the program image stops at a very high minimum threshold (200, in fact); no attempt is made to find and measure all the stars. In fact, if all went well, only 315 stars will have been found, and you should note that the run was much faster then when you reduced the frame to completion. The resulting file, **obj_out_program1** will only be used to transform the template results to the program frame coordinate system.

To do this, we have supplied a routine called OFFSET in the **transform** subdirectory. *This routine is not thoroughly tested and is intended only to give the user an idea of how one might change the program coordinates to the template system.* That said, one should now run OFFSET, answering the questions as follows. The primary file name is **obj_out_program1**; this file represents the "target" coordinate system. The secondary file name is **obj_out_template1**. The ratio of the scales is 1.0 and a tolerance of 0.02 (in pixels) is fine. The resulting offsets in x and y are -1.45 and 9.17 pixels, respectively; the scales are nearly 1.0, and the rotations tiny. After this information is spewed out, answer the next query with a 'yes' and name the output file **obj_in_template2**. You should answer the final question with 'no'. For the curious, OFFSET matches coordinate lists by identifying similar triangles. It is supposed to be able to handle linear offsets (assuming the x and y scales are the same), rotations, and reflections. You might want to supply a different routine to do this task; however, OFFSET is still useful as a guide for what the format of the output file should be.

Now that you have a transformed version of the template file, run DoPHOT 4.0 again using **param_program2** as the input parameter file. The key feature of this parameter file is that the parameter FIXPOS is set equal to YES, and the OBJECT_IN parameter is set equal to the file **obj_in_template2**. The latter tells DoPHOT this is a "warmstart", the former informs the program that this is the special sort of warmstart which assumes fixed positions from a template file. You will note two very special points about

this final run of DoPHOT. First, the initial threshold is 35.44!! Since you are now using positions from a (deep) template file, there is no need to search for new stars at a high threshold. In fact, this final run has been set up so that DoPHOT will only search once for any possible stars not in the template file, but present in the program frame (*e. g.* variables, objects beyond the edge of the template frame, *etc.*). Second, the final list contains 2458 objects, compared to the 1838 objects in the list generated by the 'straight' run of DoPHOT without a template file. To be sure, not all the objects in the output list are reliable stars, but a number of objects that would not have been split as two or more objects on the program frame, were treated as such because of the independent positional information from the template list. Note too that some stars have *negative* fluxes. This is OK, and simply represents cases where a star in the template frame was not recovered in the program frame (cosmic ray? very faint star? variable?). The errors for such stars are negative, reflecting the fact that the error is computed as 1.086*((Δ Flux)/Flux). Upper limits may be computed from the reported flux and the reported magnitude of the error. For certain applications, this might be quite useful. One final point: you should have noticed that this final run of DoPHOT was somewhat faster than the original reduction of the program frame.

The best way to understand what files are needed is to study the parameter files mentioned above. The nomenclature of the file names has been designed to help you understand how the various files interact. Although the procedure described above is a bit cumbersome for a program designed originally to run in an automatic manner, it provides a method of using a pre-determined position list in DoPHOT. We hope to try to implement a more automated version of the fixed-position option, but making it bullet-proof will probably not be simple.

11.3 Median-Filtered Background Model

The object finding algorithm relies on a MODEL sky for initial estimation of the background when finding new objects. A candidate object must then have sufficient signal to noise relative to a local average sky computed from the actual image data to be included in the object list for subsequent processing. If the model overestimates the background in the vicinity of an object, the initial object recognition may fail, and the

object will not be found. For images where the background is not well described by a PLANE or HUBBLE model, it is possible to describe the background by median filtering the data image with a filtering box. The size of the filtering box corresponds to the minimum length scale of features that can be tracked by this model, while features smaller than the box are suppressed. By keeping this box a little larger than the size of a star (say 2*FWHM), an adequate model of the background can be realized that tracks irregular backgrounds on scales larger than those for stars.

Note that just as for the PLANE and HUBBLE models, this option uses the background image as a MODEL for initial background estimation only: ALL PHOTOMETRY IS DONE ON THE DATA IMAGE.

This procedure is invoked by setting the SKYTYPE parameter to MEDIAN. A background model picture is then created before the first pass at the top threshold level. It may take some time to compute this image. It should not be necessary to compute this model at every threshold level: it is computed once again at the last but one pass after all the bright objects have been subtracted. If you are really, really unhappy about this and want to change the frequency of updating the model, you will need to change the relevant controlling code, all of which is in the main DOPHOT module.

The median filtering routine itself has been tailored to the needs of DoPHOT. We have used an algorithm that is fairly fast. It will use only legal values of the data between ITOP and IBOTTOM (parameters), and will make appropriate compensations in saturated regions and 'holes'. Images with smoothly varying backgrounds are filtered much faster than those where the backgrounds fluctuate wildly. There are several associated parameters you will need to set when using a MEDIAN skytype. These are `JHXWID`, `JHYWID`, `MPREC`, and `NTHPIX`. Their function are described in Appendix A.

11.4 EMPIRICAL Point Spread Function

The principal change in version 3 is the addition of empirical PSF fitting to the analytic PSF fitting of versions 1 and 2. The empirical fitting proceeds in parallel with the analytic fitting, starting at a specified threshold (which ought to be somewhat lower than the initial threshold, say a factor of 10, to give the program some time to clean up the template star). The empirical PSF, rather than the analytic PSF, is used for the cre-

ating the star subtracted image (with the significant exception that the analytic PSF is subtracted from the template star itself). The empirical template may be written out as a subraster for examination. The analytic PSF is used for those objects classified as galaxies.

Version 3 includes a new function which interpolates values from the empirical PSF using a 12 point scheme which is third order in displacement. It represents a compromise between using only the nearest pixels (which clips peaks and fills valleys) and using distant pixels which may be dominated by noise. We are anxious to receive feedback and comments from users regarding this particular approach to interpolation, which is different from the commonly used one of first subtracting an analytic function, such as a Gaussian, and then interpolating quadratically.

In principal empirical PSF fitting ought to produce better photometry and cleaner star subtracted images. Version 3 therefore permits a different (and presumably smaller) residual noise factor to be used.

We have been slightly inconsistent in not using the empirical PSF in splitting suspected double stars, though it might be especially useful for this purpose.

For the moment a single star (cleaned of its neighbors) is used as the template. It is chosen automatically, starting from the brightest star and working down. One can ask that the n brightest stars be skipped. One can choose the size of the subraster required. The larger the subraster the more stars will be disqualified, either because they lie close to the edge of the image or because there are bad pixels close to the center. Single bad pixels far from the center are smoothed over by linear interpolation. Adjacent (same row or same column) bad pixels cause a star to be rejected. Alternatively, one can **specify** the empirical PSF star manually by giving its x, y and approximate central intensity. The code was written with an eye to future construction of the template from more than one star. The use is controlled by new parameters THRESHEMP, N_EMP_SKIP, EMP_STAR_X(YZ), EMP_PSF_BOX, EMP_REJ_RAD_SIG, and EMP_SUBRAS_OUT. A fuller description of what these parameters do can be found in Appendix A.

The empirical central intensities and positions are scaled and shifted by the best analytic fit to the template star, so as to make the two agree. Thus reported positions

and magnitudes from empirical fitting have **no systematic difference**. The version 3 output files in COMPLETE and INTERNAL modes have 5 new additional columns with the results from empirical PSF fitting. The results from analytic fitting are also reported (in their original columns). Note that even the analytic fitting benefits from the improved object subtraction when empirical fitting is invoked. The output columns are described in detail in Appendix B.

An example of using the empirical PSF mode is provided: see the modified parameter file 'pm' provided with the test image (in the verif_data subdirectory).

TURN BACK!

Read preceding sections carefully before proceeding

APPENDIX A: DoPHOT PARAMETERS

DoPHOT contains a large number of user-specified parameters. This should not immediately panic most users because in practice, only a small number of these parameters typically need to be adjusted. Moreover, many of the parameters can be allowed to scale with such quantities as the FWHM of the stellar objects on an image or the approximate mean sky level. Many of the functions controlled by DoPHOT's parameters have been discussed in earlier sections of this manual, especially Sections 6-9. The purpose of this Appendix is to describe the actual parameters used by DoPHOT and to illustrate how a user can adjust them. Some hints as to how a user *should* adjust a given parameter are also provided where appropriate. *It is very strongly recommended that users become acquainted with the material described in this Appendix before attempting to run DoPHOT on any of their own images.* If you adjust the parameters blindly, you may get burned. Used with care, however, they allow DoPHOT to successfully analyze a wide variety of images.

A.1. A Brief Comment on Units

"Digital Numbers" (DN) refer to the counts in an image that is ready for processing by DoPHOT or any other photometry program. DN can be converted to electrons via the detector's 'gain' in electrons/data number. DN are sometimes referred to as Analog-Digital-Units (ADU), or Data Numbers (also DN). DoPHOT does not care what the scale of a picture is; consequently all (x, y) coordinates are expressed in pixels.

A.2 The Parameter File

When DoPHOT is started, it reads the default values of the parameters from a "default parameters" file that is specified by the user or in a "modified parameters" file. The default parameters file suppled with version 4.0 of DoPHOT is called **param_default_c**. This file includes *every* legal DoPHOT parameter and assigns a default value to each. The values given for the parameters listed in the default parameters file can be adjusted to suit the user's needs, but the names of the parameters cannot. *A default parameters file that includes all legal DoPHOT parameters must exist for* DoPHOT *to run.*

The modified parameters file is designed to include only the parameters whose values the user wishes to alter compared to what is provided in the default parameters file.

Thus, let's say that you have a frame where `FWHM` = 3.2, not 2.0 as is listed in the default parameters file (we haven't defined the parameter `FWHM` yet, but you can guess what it means). The modification parameters file might then only include a line with the new value of `FWHM`; all other values for DoPHOT parameters will then be taken from the default parameters file. Of course, you can explicitly list any or all other parameters in the modified parameters file, even if their assigned values do not differ from what is in the default parameters file. The basic rule is that whatever values are assigned in the modified parameters file will be adopted, and the values of any parameters not assigned in that file will be taken from the default parameters file. Ideally, when you run DoPHOT the program will ask you only for the modified parameters file name. If all is well with this file and with the data, there will be no more questions or input of any sort as the program proceeds to reduce your data.

The remainder of this appendix describes the rules that govern the parameter files so that you can properly and reliably transmit your the parameter values you wish DoPHOT to use. The rules that govern the usage of default, modification, and output (see below) parameters files are identical so we will usually just refer to "parameter files" in a generic manner.

A typical line in a parameter file may look like this:

```
FWHM = 3.2      Approx FWHM of objects in pixels.
```

The important features of this line are: (a) a *keyword* – in this case "`FWHM`" – that begins in the first column of the line, (b) an "equals sign" character with at least one blank space immediately before and after it, (c) a value – in this case `3.2` – that will be assigned to the variable associated with the keyword, and (d) a comment statement that briefly describes the meaning of the keyword. The comment is optional. Additionally, the equals sign is optional which will tell dophot that the value assigned is NULL, however it is better practice to comment out NULL assigned keywords with an equals sign as the first character of the line. DoPHOT parameter files look a little like FITS headers, but they are not identical, and the differences should be appreciated.

This example of a parameter file line illustrates the use of a keyword to assign a numeric value to a variable. The variable *type* (*e.g.* REAL or INTEGER) is still specified

using the standard FORTRAN convention, even though the code is now in C. Thus, any keyword starting with a letter between "I" and "N" (inclusive) is assumed to represent an integer variable; otherwise the variable is assumed to be real.

That is, if it is not a character string. The following line illustrates an example of how one specifies a character string in a parameter file:

```
IMAGE_IN = 'A_nice_picture'        Input image name.
```

This line also includes a keyword that begins in the first column, an equals sign with blanks before and after, a character string (in quotes), and a comment. Again, the comment is optional, and lack of an equals sign will indicate that the keyword is set to null, though it is better practice to indicate that the line is a comment by putting an equals sign at the beginning of the line. Unlike versions 3.0 and earlier, setting a character string equal to a string of one or more spaces will NOT set it equal to NULL and may cause problems. It is worth emphasizing that the character string *must* be surrounded by single quotes as shown although trailing blanks are fine. The valid FORTRAN convention of using two consecutive single quotes to represent an apostrophe is *not* supported in DoPHOT parameter files. There are no restrictions/conventions regarding the naming of keywords associated with character strings – the quotes tell DoPHOT what it needs to know in such cases.

As shown in the examples above, comments can be appended to the end of a keyword line. A full-line comment can be placed in any parameter file simply by placing the equals sign in the first column. The following is a valid full-line comment:

```
= This is a full-line comment. The '=' is in the first column.
```

Notice that the quotes and the righthand equals sign are valid since the position of the lefthand equals sign identifies the entire line as a comment. Any line in a parameter file will be truncated at 200 characters when it is read. These rules are summarized using full-line comments that are included in the file **params_default_c** that is included with the code. The contents of the default parameters file is listed below at the end of this appendix.

The following are examples illegal parameter file lines:

```
This is supposed to be a comment, but look where the '=' is!
```

```
ThreshMin = 20.0    Keywords CANNOT mix upper and lower case, unless of course
you have defined new ones
OBJECTS_IN='file'    Illegal; gotta have spaces on BOTH sides of the '='.
OBJTYPE_IN = 'INterNAL'    Character strings CANNOT mix upper and lower case,
unless of course you have defined new ones
```

If the illegal lines shown above were included in a parameter file, we'd hate to think what might happen. You should be advised that the authors have not tried to program DoPHOT's parameter input routines to cope with all possible contingencies, so be careful and follow the rules laid out in this section.

A.3 The structs/tuneable.h File

This file is used to pass variables between DoPHOT subroutines using the C `#include` statement and it is not really designed to be user-adjustable. However, if (a) the number of objects measured on a single image exceed 20000 DoPHOT's default array sizes will be too small to cope and the corresponding parameter `NSMAX` should be changed here. Of course, if memory is in short supply, the user can likewise decrease `NSMAX` here. Unless you really mean business, (i.e. are inserting a new model with a greater number of parameters than the default) *no other parameters in this file should ever be adjusted!* **VERY IMPORTANT:** If **structs/tuneable.h** is ever changed, *all* of the DoPHOT modules will have to be re-compiled or they will be unaware of the changes you have made in the include file. Makefiles to recompile all of DoPHOT are supplied with the code. Because Makefiles look for modifications, Unix users must first delete all the *.o (that is a lower-case letter o, not zero, not upper-case) files in their working DoPHOT directory first.

A.4 A Description of DoPHOT's Parameters.

This section describes the parameters found in the parameter files used by DoPHOT. A full understanding of some of these can be best attained by looking carefully at the code. The parameters described here are listed in the same order that they are found in the parameter file. This order is not random. Specifically, the most frequently-adjusted parameters are listed first, with subsequent sections describing parameters that are adjusted less frequently.

A.4.1 Frequently updated parameters.

Some or all of these parameters can be expected to vary from frame to frame.

`FWHM` – The initial guess of the full width at half maximum along the major axis of a typical, isolated stellar object in pixels. A modest accuracy (say, 10-20%) in the estimate of this is fine, but avoid providing FWHM estimates that are significantly too small.

`AXIS_RATIO` – The initial guess of the ratio of the minor axis divided by the major axis for star objects. This parameter is 1.0 for perfectly round objects. This parameter rarely needs to be changed from 1.0 unless the images are *very* elongated, say for axis ratios less than about 0.7 or so.

`TILT` – The initial guess of the position angle of the major axis with respect to the positive x-axis for a typical unblended stellar object on an image. This parameter has no effect if the axis ratio is 1.0. The tilt is given in degrees between -90 and $+90$, with the positive y-axis at $+90$ degrees.

`SKY` – The initial guess of the mean sky value of the frame in DN. An accuracy of 10-20% is fine, but a better estimate is advantageous.

`NFITBOX_X, NFITBOX_Y` – The sizes of the sides of the fitting box in the x and y directions, respectively. These parameters can be automatically scaled by the FWHM using `AUTOSCALE` (see below).

`MASKBOX_X, MASKBOX_Y` – The sizes of the sides of the mask box used for finding objects in the x and y directions, respectively. These parameters can be automatically scaled by the FWHM using `AUTOSCALE` (see below).

`APBOX_X, APBOX_Y` – The sizes of the sides of the aperture photometry box in the x and y directions, respectively. In the current version of DoPHOT, the *sky* measurement for the aperture photometry is performed in a square annulus with outer side lengths equal to twice `APBOX_X` and `APBOX_Y` in x and y, respectively. The inner boundary of this sky annulus is defined by the aperture photometry box. These parameters are scalable by the FWHM using `AUTOSCALE` (see below).

`THRESHMIN` – The value above sky of the lowest threshold that DoPHOT uses to search for objects, in DN. Thus, if this parameter is 20, and the actual sky value is 450, the fi-

nal threshold will occur at 470. This parameter can be scaled using the estimate of the sky value and the readout noise of the detector (see AUTOTHRESH below).

THRESHMAX – The maximum possible first threshold that DoPHOT uses to search for objects, in DN. Because of the function of the following parameter, THRESHDEC, it is important to understand that THRESHMAX may *not* be the actual value used for the first threshold pass.

THRESHDEC – The separation between successive thresholds in units of 2.0 raised to the power THRESHDEC. Thus if this parameter is 1.0, and THRESHMIN = 20, and THRESHMAX = 1000, the thresholds (relative to the sky value) would be 640, 320, 160, 80, 40, and 20. In this example, each level is separated by a factor of 2^1 from its neighboring levels.

THRESHEMP – If THRESHEMP is less than THRESHMIN (0.0 is default), it does nothing. For positive value V, once the threshold drops below V, empirical PSFs are fit. Ensure that stars will have been found by the time the threshold drops below V.

MAX_PERF – *new feature* The maximum number of type 1 stars that will be used to compute the weighted averages of shape parameters for star objects. It is useful to set this number low only if the error estimates on shape parameters in new models are underestimated for dimmer objects or dimmer objects are likely to be misclassified as stars. Otherwise the weighted averages will account for the error in the parameter estimations of the dimmer stars.

EPERDN – Electrons per DN for the detector.

RDNOISE – The readout noise of the detector in electrons.

AUTOSCALE – This flag can only be set to YES or NO and if DoPHOT has any trouble reading the value of AUTOSCALE it will default its value to NO. If AUTOSCALE is YES, then the box sizes described above are scaled by the FWHM using the parameters SCALEFITBOX, FITBOXMIN, SCALEAPBOX, APBOXMIN, SCALEMASKBOX, and AMASKBOXMIN. If AUTOSCALE is NO, then the values of the box sizes are taken directly from the parameter file without any scaling. The scaling parameters used by AUTOSCALE are described as follows:

SCALEFITBOX, FITBOXMIN – *These parameters are used only if* AUTOSCALE *is* YES

– NFITBOX_X and NFITBOX_Y are set equal to the smallest odd integer that is larger

than SCALEFITBOX times the object FWHM in the x and y directions (these do not necessarily equal the value of FWHM described above), respectively. If either or both of these box sizes end up smaller than FITBOXMIN, then the relevant fit-box size is set equal the nearest odd integer larger than FITBOXMIN.

SCALEAPBOX, APBOXMIN – *These parameters are used only if* AUTOSCALE *is* YES – APBOX_X and APBOX_Y are set equal to the smallest odd integer that is larger than SCALEFITBOX times the object FWHM in the x and y directions, respectively. If either or both of these box sizes are computed to be smaller than APBOXMIN, then the relevant fit-box size is set equal the nearest odd integer larger than APBOXMIN.

SCALEMASKBOX, AMASKBOXMIN – *These parameters are used only if* AUTOSCALE *is* YES – MASKBOX_X and MASKBOX_Y are set equal to the smallest odd integer that is larger than SCALEMASKBOX times the object FWHM in the x and y directions, respectively. If either or both of these box sizes turn out to be smaller than AMASKBOXMIN, then the relevant fit-box size is set equal the nearest odd integer larger than AMASKBOXMIN.

As a rule of thumb, it is best not to use AUTOSCALE (*i.e.*, set it to NO) until you are quite familiar with DoPHOT. Moreover, the default values for the scaling parameters might be expected to vary among different telescope/detector combinations, so be wary.

AUTOTHRESH – This flag can only have the values YES or NO. Any illegal or unitelligible value for this keyword in the parameter file will cause this parameter to default to NO. If AUTOTHRESH is YES, then IBOTTOM and THRESHMIN are scaled according to the sky+read noise using the scaling parameters SIGMAIBOTTOM and SIGMATHRESHMIN. If AUTOTHRESH is NO, then DoPHOT takes the values of IBOTTOM and THRESHMIN supplied by the user. The threshold scaling parameters controlled by AUTOTHRESH are described as follows:

FIXPOS – This flag can only have the values YES or NO. Any illegal or unitelligible value for this keyword in the parameter file will cause this parameter to default to NO. If FIXPOS is YES, then the program assumes you have an input template file with the positions of objects as described in detail in Section 11.1 and 11.2. This template file should be specified with the OBJECTS_IN parameter (see below). Be careful: the fixed-position version of DoPHOT is tricky to use and one should use this parameter with caution. If

FIXPOS is NO, then DoPHOT runs without fixing any of the object positions regardless of what file (if any) is specified by OBJECTS_IN.

SIGMAIBOTTOM – *This parameters is used only if* AUTOTHRESH *is* YES – IBOTTOM is set equal to SKY *minus* SIGMAIBOTTOM times the sky+read noise. This noise equals $[gs + r^2]^{1/2}/g$, where g = EPERDN in electrons per DN, s = SKY in DN, and r = RDNOISE in electrons.

SIGMATHRESHMIN – *This parameters is used only if* AUTOTHRESH *is* YES – The lowest threshold is set equal to the sky+read noise times SIGMATHRESHMIN where the noise is defined as for SIGMAIBOTTOM.

As in the case of AUTOSCALE, be wary of using the AUTOTHRESH parameters until you become familiar with how DoPHOT works without using this switch.

PARAMS_DEFAULT – This keyword specifies the name of the default parameter file. If DoPHOT cannot find the file, it will be requested. When you run DoPHOT, you must create a 'modification' parameter file that only needs to contain the parameters you wish to change. The values of any parameters that the program cannot find in the modification parameter file will be taken from the default parameter file. Any bogus (or misspelled) parameters in the modification file are ignored. However, a default parameter file with *all* legitimate parameters *must* exist or DoPHOT will not run. Such a file is provided with this release of the program, and it should be guarded carefully.

PARAMS_OUT – DoPHOT records the values of all the parameters it used for a given run in the file listed with this keyword. If this line is blank (i. e., just two single quotes or any number of blanks surrounded by single quotes), then DoPHOT assumes you don't want such a file and none is saved.

Because DoPHOT does not check the legitimacy of all the required files read from the default and/or modified parameters file until after the output parameters file is written, some invalid file names can end up being saved in the output parameters file. This bug was identified too late to correct in version 2.0 of DoPHOT.

IMAGE_IN – This keyword identifies the input image name. It *must* exist, naturally, and if the name in the parameter file does not correspond to any existing file, then DoPHOT will query the user for the correct file name.

IMAGE_OUT – While DoPHOT is reducing an image, the program continuously updates a version of the original picture from which all known objects have been subtracted. This subtracted picture can be saved when DoPHOT is finished in the file specified by IMAGE_OUT. If the string assigned to this keyword is blank, the program assumes you don't wish to save the subtracted picture and none is written to disk.

EMP_SUBRAS_OUT – File name for writing out the adopted empirical PSF subraster and saving it when DoPHOT is finished. If left blank, the empirical PSF image is not saved.

OBJECTS_IN – This keyword specifies the name of the file with the input object list. This file is mandatory for a WARMSTART (see below) and will be requested if needed. If no WARMSTART is desired, this keyword is ignored.

OBJECTS_OUT – This keyword identifies the name of the output photometry file. Since you have gone to the trouble of using DoPHOT, you presumably want to keep the output photometry; thus, if no legal file is specified here, the program will request its name until you give it something it likes.

SHADOWFILE_IN – One option in DoPHOT is to do a WARMSTART (see below) and continue a reduction that failed due to an error in DoPHOT or a computer crash or any similar problem. The shadow files contain the intermediate steps of the reductions, and if you have one you save a bit of time on a WARMSTART. If you are doing a WARMSTART, DoPHOT will use this keyword to determine the name of the input shadow file. If this line is blank, the program assumes you don't have a shadow file (which is fine) even if the READSHD flag is set to YES (see below). If the program finds a file name but can't open the file, it will ask for a name. Shadow files are only stored in INTERNAL format (see OBJTYPE_IN and OBJTYPE_OUT, below).

SHADOWFILE_OUT – If you plan to do a warm start in the future or believe it would be prudent to have one in case of future problems, you can save a shadow file. The name of such a file is specified using this keyword. If no name is given, then the program assumes you don't want an output shadow file.

ERRORS_OUT – *new feature* If you save a shadow file, an identical file containing the squared errors on each of the shadow parameters is automatically generated. The name of such a file is specified using this keyword. If no name is given, the default file 'error_out' is

created.

`COVARS_OUT` – *new feature* Optionally generate a file containing a 'covariance' matrix for the fit of each object. The name of such a file is specified using this keyword. This matrix is not a true covariance matrix but actually contains the square root of the autovariances along the diagonal and the normalized correlations along the off-diagonals. The parameter order is Sky, I_0, x, y, σ_x^2, σ_{xy}, σ_y^2, + optional fit parameters. Covariance matrices are generated in *shape.c* and therefore reflect the covariances of the 7+ parameter fits for objects, including stars. Use with caution.

`LOGFILE, LOGVERBOSITY` – The user can specify varying levels of detail in the messages that DoPHOT provides. These messages range from information about which threshold the program is currently working on, to iteration counters for the nonlinear least-squares fitting routine. `LOGFILE` specifies the name of the file into which these messages are printed. Any name that your operating system is happy with is valid except one. If this keyword is set equal to TERM (all upper-case letters!), then DoPHOT's output is directed to the terminal. `LOGVERBOSITY` specifies the detail in the messages: 0 means no messages are printed out at all and no file is opened even if `LOGFILE` is specified. Increasingly larger values of `LOGVERBOSITY` (up to 4 is currently supported) provide progressively more detailed messages. Try different values to see what you like, but be warned that at full verbosity (4) you will get *many* messages, so make sure you have loads of disk space. Also, the program (naturally) runs significantly slower if it is printing out many messages. The messages *are* useful, though, if things are going wrong. At low verbosity, DoPHOT's messages help to reassure the user that the program is at work and provide information how far along the reductions have proceeded.

A.4.2 Infrequently updated parameters.

These parameters are not typically changed very frequently but may vary for images obtained with different detectors, for example.

`RESIDNOISE` – This parameter specifies how much *extra* noise to add(subtract) to the noise file when subtracting(adding) an object to an image and is designed to discourage DoPHOT from finding 'phantom' objects in the wings of previously-subtracted bright objects.

EMP_RESIDNOISE – If empirical PSFs are used, the images of stars should subtract extremely well, so the *extra* noise needed (see RESIDNOISE above) to prevent detection of 'phantom' objects is much smaller. This is implemented via this parameter, which is used in lieu of RESIDNOISE when empirical PSFs are being fitted.

FOOTPRINT_NOISE – The linear dimensions of a star's footprint in the noise file are enlarged by this amount. The purpose of this parameter is – as in the case of RESIDNOISE described above – to discourage DoPHOT from trying to find faint stars in the residuals of the wings of bright or extended objects. This linear expansion is **not** performed when and empirical, rather than analytic PSF is subtracted.

NPHSUB – When subtracting an object from an image, DoPHOT goes out to where the PSF fitted to an object reaches this surface brightness.

NPHOB – When obliterating objects, DoPHOT determines where the PSF that has been fitted to an 'obliteratable' object (see the discussion of ICRIT, CENTINTMAX and CTPERSAT below) reaches a surface brightness equal to NPHOB, in DN. The size of the side of an obliteration box for a given object is then set equal to twice this linear distance. The sizes of the x and y sides of obliteration boxes can differ if the PSF that has been fitted to the object is significantly elongated. The calculation of the obliteration box using NPHOB does *not* use the BETA4 and BETA6 parameters described below.

ICRIT – If there are this many contiguous saturated pixels (as determined by testing the pixel values with ITOP), then DoPHOT obliterates the object. The size of the obliteration box is controlled by NPHOB (see above).

CENTINTMAX – If the fitted central intensity of an object exceeds this value, it is obliterated using NPHOB to determine the obliteration box size.

CTPERSAT – If an object is to be obliterated – either because it contains too many bright pixels (the ICRIT test) or is simply too bright (the CENTINTMAX test) – then its central intensity is assumed to be CTPERSAT times the number of saturated pixels in the object. This helps DoPHOT calculate reasonably-sized obliteration boxes.

STARCOSKNOB – Every object found by DoPHOT is initially fit as a single star and a cosmic ray. If STARCOSKNOB times χ^2(cosmic ray) $\geq \chi^2$(single star), the object is classified as a cosmic ray.

STARGALKNOB – Every non-cosmic ray object is fitted as a galaxy after being fitted to a single star. If the galaxy fit is significantly better than the fit for a single star, then the object is also fit as a double star If STARGALKNOB times χ^2(galaxy) $\leq \chi^2$(double star), the object is classified as a galaxy; otherwise as a double star.

SNLIM7 – An object's S/N must exceed this limit to fit all seven parameters.

SNLIM – A given pixel's S/N must exceed this limit in order for it to be included in a fit subraster.

SNLIMMASK – This parameter is the minimum S/N that must be detected through the mask in order for DoPHOT to trigger on an object. Excessively low values of SNLIMMASK will cause DoPHOT to trigger on noise as it approaches the lower thresholds. This increases execution time with very little gain in detection efficiency.

SNLIMCOS – This is the minimum S/N required for an object to be classified as a cosmic ray. Without this parameter, noise spikes (lots of them!) would be called cosmic rays at low thresholds.

NBADLEFT, NBADRIGHT, NBADTOP, NBADBOT – Pixels that are located closer to the edges of the image than these values are ignored by DoPHOT – sort of like 'software trimming'. Left, right, top, and bottom correspond to small x, large x, large y, and small y, respectively.

PSFTYPE – This flag specifies the form of the analytic PSF used in DoPHOT's fits for objects. The only allowed response currently is PGAUSS, so no matter what you specify in the parameter file, you'll get PGAUSS. Additional PSF types may be available in future versions of DoPHOT.

SKYTYPE – This flag identifies the form of the analytic sky function used by DoPHOT for finding objects. It is worth emphasizing that the individual sky values for fitted objects *are not* derived from this sky function – the sky function is only used to help DoPHOT finding things on an image. The currently allowed options for this flag are PLANE, HUBBLE, and MEDIAN. For a description of the latter, see Section 11.3.

JHXWID, JHYWID – These specify the HALF size of the median filtering box along X and Y respectively. They must be integer values. If you specify a non-positive value, the size will be scaled automatically from the FWHM parameter. The default for both is

0, which forces auto-scaling.

`MPREC` – This controls the precision to which the median model is calculated. If the noise in the image background is say 100 DN, you do not need a model computed to 1 DN precision. You can ask instead for precision to n DN by setting the value of the MPREC parameter to n. This must be an integer. Non-positive values of MPREC will result in setting MPREC to 0.25*THRESHMIN. The default is 1, the most conservative setting.

`NTHPIX` – This is really not a filtering parameter, but one used in the search for objects. It controls how often the search routine updates the value of the background from the model (be it PLANE, HUBBLE or MEDIAN) when initially raster searching along X for peaks in the data. NTHPIX is the interval in pixels for this update. Setting NTHPIX 0 or negative (the default) sets NTHPIX = SQRT(X-size of image in pixels) to the nearest integer. This is adequate for all but the grottiest backgrounds. If you must have a smaller value you can change this to about 2*JHXWID. Smaller values cause the search routine to run considerably slower.

`OBJTYPE_IN, OBJTYPE_OUT` – Three input/output formats are supported by DoPHOT. These are identified by the character strings COMPLETE, INTERNAL, and INCOMPLETE. The 'COMPLETE' input/output format type provides all the information DoPHOT has to offer about the objects it has identified in an image using 'friendly' units (*i.e.*, magnitudes, FWHM, *etc.*; Appendix B describes DoPHOT's data formats in detail). The 'INTERNAL' input/output type provides the same information that is provided in the 'COMPLETE' format type but using the units employed internally by DoPHOT. The 'INCOMPLETE' output type is compatible with DAOPHOT standard output files, but some information is lost if this format is chosen.

The `OBJTYPE_IN` flag identifies the input object file format type, and is needed if one will be doing a "warmstart". Only the COMPLETE, and INTERNAL format types are allowed because there is not enough information in the 'INCOMPLETE' format files to initiate a warmstart. The `OBJTYPE_OUT` flag identifies the output file format type, and all three options are allowed. Note that shadow files are only written and read using 'INTERNAL' format types since these files are rarely of interest to most users.

`PSDIR, PSNAME_ROOT, CPSNAME_ROOT, MPSNAME_ROOT, RPSNAME_ROOT` – *new feature* If

you want to output a postage stamp (PS) image for each obj detected (i.e. an image which is cut around the obj center and is the size of the model subtracted) specify a keyword for PSNAME_ROOT such that the desired PS file names are 'PSDIR/d#_PSNAME_ROOT.fits' where the number corresponds to the number of the object in the object file. Similar PS files can be created for the neighbor cleaned images of the objects, images of the models for each object, and images of just the residuals for each object once the neighbors and models have been subtracted off, by specifying names for CPSNAME_ROOT, MPSNAME_ROOT, or RPSNAME_ROOT respectively. Specifying a postage stamp directory PSDIR will not cause any files to be generated, but will properly direct files if any of the PSNAME_ROOT keywords are specified. Postage stamps are a new feature DoPHOT 4.0 which are a useful diagnostic for seeing the quality of neighbor subtraction and the fit of any individual object, but generating PS images does slow down the run as individual images are generated for each object detected.

A.4.3 Very rarely updated parameters.

Not only are these parameters rarely changed, but they can cause serious problems if adjusted indiscriminately. Nevertheless, they are included in the parameter file to increase DoPHOT's flexibility in dealing with unforeseen reduction problems. However, even more so than for the parameters described above, *be sure you understand the functions of these parameters before changing their values.*

`NFITITER` – The value of this parameter controls the maximum number of iterations DoPHOT performs during its non-linear least-squares fits to individual objects on an image.

`NPARAM` – NOW OBSOLETE- For developers, set to update in tuneup.c automatically for an updated model type. For users, if you do not change the model, or if you use a model supplies with the package, this number will set correctly (to 7 for a pseudo-gaussian) automatically. Previously, NPARAM specified the maximum number of fit-parameters in the analytic PSF function. For example, the fit-parameters used by the pseudo-gaussian function (which is specified by setting `PSFTYPE` = PGAUSS; see above) are the sky, the (x, y) centroid of the object, the central intensity, width parameters in x and y and a cross term to account for tilted images.

NFITMAG – The maximum number of fit-parameters used in fitting the analytic PSF to objects when the shape fit-parameters are assumed. Thus for the pseudo-gaussian function (see PSFTYPE above) these fit-parameters are the sky, the (x, y) centroid of the object, and the central intensity. The shape fit-parameters are assumed to be equal to the current best guess that DoPHOT has for these.

NFITSHAPE – NOW OBSOLETE- For developers, set to update in tuneup.c automatically for an updated model type. For users, if you do not change the model, or if you use a model supplies with the package, this number will set correctly (to 7 for a pseudogaussian) automatically. Previously, NFITSHAPE specified the maximum number of fit-parameters used in fitting the analytic PSF to objects when all the fit-parameters are free to vary. For the pseudo-gaussian function (see the description for PSFTYPE above) these fit-parameters are the same as described for NPARAM above.

NFITBOXFIRST_X, NFITBOXFIRST_Y – Because the initial estimates of the shape parameters provided by the user may not be especially good (nor must they be), DoPHOT should use a particularly large fitting-box on the first threshold in which stars are detected. This parameter controls the size of sides of the 'first-pass' fit subraster boxes (see NFITBOX_X and NFITBOX_Y for related information).

N_EMP_SKIP – Setting this to a positive value N (0 is default) will tell DoPHOT to ignore the N brightest bonafide stars when selecting a star to use as the empirical PSF.

EMP_STAR_X, EMP_STAR_Y, EMP_STAR_Z – You can specify a particular star in the data frame as the empirical PSF template by specifying its location in X and Y, and its approximate peak height (Z) through these parameters. The default values are zero, which makes DoPHOT select a star itself, constrained by N_EMP_SKIP.

EMP_PSF_BOX – is the size of the empirical PSF subraster

EMP_REJ_RAD_SIG – Used when selecting a PSF star. If a bad pixel is encountered within V*sigma of the center of a star, it disqualifies it from being used as a template for the empirical PSF. (For this purpose, sigma is FWHM/2.35).

CHI2MINBIG – This parameter specifies the critical value of χ^2 for three degrees of freedom above which an object is classified as "VERY BIG" (*i.e.*, either a double star or a galaxy). Such objects are then subjected to fits as double stars, and DoPHOT de-

cides whether they are best classified as galaxies or double stars. A related parameter is STARGALKNOB described above.

XTRA – This parameter has the same function as CHI2MINBIG except that it is used if the object has previously already been classified as a member of a double star.

SIGMA1, SIGMA2, SIGMA3 – These parameters control the minimum fractional amount by which the PSF shape parameters of a galaxy must vary from the shape parameters of a single stellar object, and are used by DoPHOT to calculate the χ^2 statistic for galaxy objects.

ENUFF4, ENUFF7 – At least ENUFF4 of the pixels in a fit subraster must be present in order to perform a four-parameter fit (see NFITMAG above). ENUFF7 specifies this limit for seven-parameter fits (see NFITSHAPE above).

COSOBLSIZE – This specifies the size of the obliteration box (in both x and y directions) that is employed to remove objects that have been classified as cosmic rays.

APMAG_MAXERR – Only aperture photometry results with errors smaller than the value specified by this parameter are reported in the output photometry file.

PIXTHRESH – This parameter allows DoPHOT's finding algorithm to trigger on pixels that are above PIXTHRESH times the local noise. This parameter represents one of the masking tape/baling wire type fixes that has been incorporated into the code in order to allow the finding algorithm to find faint stars in well-sampled data. By setting PIXTHRESH too low, however, you force the program waste time trying to find objects where only a single noisy pixel exists. Binning well-sampled data and leaving PIXTHRESH at 1.0 is usually more effective at finding faint stars.

BETA4, BETA6 – These parameters modify the z^4 and z^6 terms in the pseudo-gaussian PSF function as illustrated in Equation (1) of Section 7.2. If you wish to experiment with these parameters, you can plot Equation (1) with different values of BETA4 and BETA6 to appreciate how they affect the shape of the pseudo-gaussian PSF function used in the present version of DoPHOT and to compare the analytic PSF with the radial profiles of actual PSFs. For most real data, the pseudo-gaussian will never perfectly fit the observed PSF, but you will probably be able to find values of BETA4 and BETA6 that do a good job far out into the wings of the observed PSF. We suggest letting BETA4 =

BETA6 while varying them for the first time on new data sets.

D. Fit-parameter limits.

The fit-parameter limits below provide convergence limits for DoPHOT's non-linear least squares fitting routine and help avoid the program from producing absurd results or attempting to drive the fit-parameters to values that result in numeric overflows. Two sets of seven fit-parameter limits are used by DoPHOT and are identified with indices running from 1 to 7. For reference, these indices represent (1) sky, (2) x-position of center, (3) y-position of the center, (4) central intensity above sky, (5) pseudo-gaussian sigma in the x-direction, (6) the pseudo-gaussian cross term, and (7) the pseudo-gaussian sigma in the y-direction, respectively. These fit-parameters are defined by Equation (1) in Section 7.2.

RELACC1 through RELACC11 – These parameters control the convergence criteria for the fit-parameters. Positive values for any RELACC parameter refer to fractional changes in the fit-parameters between successive iterations, while negative values refer to absolute changes. For example, if RELACC1 = 0.01, then if the sky value changes by less than 1% between successive iterations, that parameter will be flagged as having converged. Similarly, if RELACC3 = −0.03, then if the object's y position changes by less than 0.03 pixels between iterations, that parameter is flagged as having converged. It should be noted, however, that DoPHOT continues to iterate until *all* of of the fit-parameters have satisfied their individual convergence criteria. Because of this, the actual convergence achieved for most parameters is much better than one might guess from the values of the RELACC parameters. Note that for the standard pseudo gaussian model, only RELACC1-7 are used. RELACC8-11 are left available for models with additional parameters.

ABSLIM1 through ABSLIM7 – These parameters control the permitted range of the fit-parameters and are used between iterations of the non-linear least-squares PSF fit. Positive values for any ABSLIM parameter controls fractional changes in the fit-parameters between sucessive iterations. Negative values for the ABSLIM parameters restrict the absolute ranges of the fit-parameters. Thus, setting ABSLIM1 = 100.0 causes DoPHOT to halt its attempt to fit the current object if the sky value changes by more than a factor of 100 between iterations. Setting ABSLIM4 = -1.0×10^8 causes a fit to halt if the central

intensity of an object ever exceeds 10×10^8 DN.

A.5 A Sample DoPHOT 4.0 Parameter File

```
= Default Parameter File for DoPHOT Version 4.0 . June 2012; MIT.
=
= THIS IS THE MASTER DoPHOT DEFAULT PARAMETER FILE AND IT SHOULD NOT BE
= DELETED OR MODIFIED!
=
= DoPHOT Parameter files are ALMOST pseudo-FITS headers.
= Each line for which a keyword is assigned a value must have an
= equals sign and at least one blank space immediately before and after it.
= Comments may be placed after the parameter value on any line and the equal
= sign may be located anywhere on a line. A maximum of 80 characters is
= allowed per line. Character variables MUST be enclosed in single quotes.
= All keyword names must start in the first column.
= As a hold over from when this was in Fortran, normal Fortran naming
= conventions are used to identify integer and real variables: keywords
= starting with letters between I and N (inclusive) represent integer
= variables, all others are reals except for variables in quotes which are
= assumed to be character variables. No restrictions apply to character
= variable keyword names except that all keyword names must be kept to under
= 20 characters.
= Null character variables are represented as having
= no equals sign. An equals sign means a value is assigned. This is
= DIFFERENT THAN THE FORTRAN VERSION!!!
=
= In this master file, the parameters are grouped according to the typical
= frequency with which they are modified. Short descriptions for each
= parameter are provided following the default value. These defaults are
= designed to work for a 'typical' TI CCD image.
=
=
= FREQUENTLY UPDATED PARAMETERS. These parameters mostly depend on the
= typical size of stellar objects on a given frame and on the mean sky value
= for the image. These parameters tend to be different for every frame.
= Also included are the bookkeeping parameters that specify file names.
=
FWHM = 2.0 Approx FWHM of objects (pixels) along major axis.
AXIS_RATIO = 1.0 For star objects. AR=b/a; b=minor axis.
TILT = 0.0 Angle of major axis in degrees; +x=0; +y=90.
SKY = 10.0 Approximate mean sky value in data numbers.
NFITBOX_X = 7 Size of fit box in the x-direction.
NFITBOX_Y = 7 Size of fit box in the y-direction.
MASKBOX_X = 5 Size of mask box size in x.
```

```
MASKBOX_Y = 5 Size of mask box size in y.
APBOX_X = 13.0 Size of aperture photometry box in x.
APBOX_Y = 13.0 Size of aperture photometry box in y.
IBOTTOM = -50 Lowest allowed data value in data numbers.
ITOP = 16384 Maximum allowed data value in data numbers.
THRESHMIN = 20.0 Value of lowest threshold.
THRESHMAX = 10000.0 Value of maximum threshold.
THRESHEMP = 0.0 Fit empirical PSF at and below this value.
THRESHDEC = 1.0 Threshold decrement in powers-of-2.
MAX_SOUGHT = 32768 Quit after this number of improved stars.
MAX_PERF = 2000 Only average up to this number of stars to get shape parame-
ters.
RANGE_MAG = 30. Sets threshhold some magnitudes fainter than brightest fit.
EPERDN = 2.0 Electrons per data number.
RDNOISE = 15.0 Readout noise in=
AUTOSCALE = 'NO' Auto-scaling of sizes by FWHM.
AUTOTHRESH = 'NO' Auto-scaling of thresholds.
FIXPOS = 'NO' Fix star positions?
=
PARAMS_DEFAULT = 'param_default_c' Default parameters file name.
PARAMS_OUT = 'param_out' Output parameters file name.
IMAGE_IN = 'image_in' Input image name.
IMAGE_OUT = 'image_out' Output image name.
EMP_SUBRAS_OUT Empirical PSF subraster (most recent).
OBJECTS_IN Input object list file name.
OBJECTS_OUT = 'objects_out' Output object list file name.
SHADOWFILE_IN Input shadow file name.
SHADOWFILE_OUT = 'shadowfile_out' Output shadow file name.
ERRORS_OUT = 'error_out' Errors on fit to be output if shadow file is requested
LOGFILE = 'TERM' Log file name. TERM for screen.
LOGVERBOSITY = 1 Verbosity of log file; (0-4).
=
=
= OCCASIONALLY UPDATED PARAMETERS. These parameters tend to not change for a
= set of images obtained during a single observing run or for frames of a
= single field. The defaults for the flags PSFTYPE, SKYTYPE, and OUTTYPE are
= PGAUSS, PLANE, and FULL, respectively.
=
RESIDNOISE = 0.3 Fraction of noise to ADD to noise file (analytic PSF).
EMP_RESIDNOISE = 0.03 Fraction of noise to ADD to noise file (empirical PSF).
FOOTPRINT_NOISE = 1.3 Expand stars in noise file by this amount (analytic).
NPHSUB = 1 Limiting surface brightness for subtractions.
NPHOB = 1 Limiting surface brightness for obliterations.
ICRIT = 10 Obliterate if # of pixels > ITOP exceeds this.
```

```
CENTINTMAX = 20000.0 Obliterate if central intensity exceeds this.
CTPERSAT = 1.0e4 Assumed intensity for saturated pixels.
STARGALKNOB = 1.0 Star/galaxy discriminator.
STARCOSKNOB = 1.0 Object/cosmic-ray discriminator.
SNLIM7 = 7.0 Minimum S/N for 7-parameter fit.
SNLIM = 0.5 Minimum S/N for a pixel to be in fit subraster.
SNLIMMASK = 4.0 Minimum S/N through mask to identify an object.
SNLIMCOS = 3.0 Minimum S/N to be called a cosmic ray.
NBADLEFT = 0 Ignore pixels closer to the left edge than this.
NBADRIGHT = 0 Ignore pixels closer to the right edge than this.
NBADTOP = 0 Ignore pixels closer to the top edge than this.
NBADBOT = 0 Ignore pixels closer to the bottom edge than this.
=
PSFTYPE = 'PGAUSS' PSF type: (PGAUSS, GAUSS, EXTPGAUSS)
SKYTYPE = 'PLANE' SKY type: (PLANE, HUBBLE, MEDIAN)
=
JHXWID = 0 X Half-size of median box (.le. 0 -> autoscale)
JHYWID = 0 Y (same as above)
MPREC = 1 Median precision in DN (use .le. 0 for autocalc)
NTHPIX = 0 Frequency of sky updates in pixels for 1st pass
=
= Default value of NTHPIX is zero. DoPHOT translates it to SQRT(NFAST)
= once the data frame is read. For MEDIAN sky, positive NTHPIX gives
= control over NTHPIX for finer initial sky sample.
=
=
OBJTYPE_IN = 'COMPLETE' Input format: (COMPLETE, INTERNAL)
OBJTYPE_OUT = 'COMPLETE' Output format: (COMPLETE, INCOMPLETE, INTERNAL)
=
PSDIR = './' directory in which to put the PS files
PSNAME_ROOT ps files names: 'd####_PSNAME_ROOT.fits'
CPSNAME_ROOT cleaned ps file names: 'd####_CPSNAME_ROOT.fits'
MPSNAME_ROOT model ps file names: 'd####_MPSNAME_ROOT.fits'
RPSNAME_ROOT residual ps file names: 'd####_RPSNAME_ROOT.fits'
=
= RARELY UPDATED PARAMETERS. These are specialized parameters that rarely
= need changing even when measuring images of very different sorts of fields
= and/or from different telescope/detector combinations.
=
NFITITER = 10 Maximum number of iterations.
=NPARAM = 7 OBSOLETE Maximum number of PSF fit parameters. obsolete, now set
with model
NFITMAG = 4 No. of PSF parameters to get magnitudes.
=NFITSHAPE = 7 OBSOLETE No. of PSF parameters to get shape and mags. obsolete
```

```
NFITBOXFIRST_X = 31 Size of fit box in x for first pass.
NFITBOXFIRST_Y = 31 Size of fit box in y for first pass.
N_EMP_SKIP = 0 Skip n stars brighter than empirical template.
EMP_STAR_X = 0 X position of empirical template.
EMP_STAR_Y = 0 Y position of empirical template.
EMP_STAR_Z = 0 Central intensity of empirical template.
NEMP_PSF_BOX = 33 Size of empirical PSF box
EMP_REJ_RAD_SIG = 2.0 Minimum distance of bad pixel from peak in sigma.
CHI2MINBIG = 16 Critical CHI-squared for a large object.
XTRA = 25 We need more S/N if some pixels are missing.
SIGMA1 = 0.10 Max. frac. scatter in sigma_x for stars.
SIGMA2 = 0.10 Max. scatter in xy cross term for stars.
SIGMA3 = 0.10 Max. frac. scatter in sigma_y for stars.
ENUFF4 = 0.50 Fraction of pixels needed for 4-param fit.
ENUFF7 = 0.65 Fraction of pixels needed for 7-param fit.
COSOBLSIZE = 0.9 Size of obliteration box for a cosmic ray.
APMAG_MAXERR = 0.1 Max anticipated error for aperture phot report.
PIXTHRESH = 1.0 Trigger on pixels higher than noise*PIXTHRESH.
BETA4 = 1.0 R**4 coefficient modifier.
BETA6 = 1.0 R**6 coefficient modifier.
=
=
= AUTO SCALING PARAMETERS. These parameters are used only if the auto-scaling
= flags are turned on. Box sizes, and threshold levels can be scaled
= according to the FWHM of objects and the sky and readout noise values.
=
SCALEFITBOX = 3.0 Size of fit box in units of FWHM.
FITBOXMIN = 5.0 Smallest allowed fit box size.
SCALEAPBOX = 6.0 Size of aperture phot box in units of FWHM.
APBOXMIN = 7.0 Smallest allowed aperture phot box size.
SCALEMASKBOX = 1.5 Size of mask box in units of FWHM.
AMASKBOXMIN = 5.0 Smallest allowed mask box size.
SIGMAIBOTTOM = 10.0 Level of IBOTTOM below sky in units of noise.
SIGMATHRESHMIN = 2.0 Level of THRESHMIN above sky in units of noise.
=
=
= PARAMETER LIMITS. These variables limit the legal ranges of the PSF and
= sky parameters. Be sure to understand their function well before changing
= any of these values. Positive values refer to fractional changes; negative
= values to absolute changes; zero ABSLIM's turn the corresponding tests off.
=
RELACC1 = 0.01 Convergence criterion for sky.
RELACC2 = -0.03 Convergence criterion for x-position.
RELACC3 = -0.03 Convergence criterion for y-position.
```

```
RELACC4 = 0.01 Convergence criterion for for central intensity.
RELACC5 = 0.03 Convergence criterion for sigma-x.
RELACC6 = 0.1 Convergence criterion for sigma-xy.
RELACC7 = 0.03 Convergence criterion for sigma-y.
RELACC8 = 0.03 Convergence criterion for additional parameter 1
RELACC9 = 0.03 Convergence criterion for additional parameter 2
RELACC10 = 0.03 Convergence criterion for additional parameter 3
RELACC11 = 0.03 Convergence criterion for additional parameter 4
=
ABSLIM1 = -1.0e8 Allowed range for sky value.
ABSLIM2 = -1.0e3 Allowed range for x-position.
ABSLIM3 = -1.0e3 Allowed range for y-position.
ABSLIM4 = -1.0e8 Allowed range for central intensity.
ABSLIM5 = -1.0e3 Allowed range for sigma-x.
ABSLIM6 = -1.0e3 Allowed range for sigma-xy.
ABSLIM7 = -1.0e3 Allowed range for sigma-y.
END
```

APPENDIX B: OUTPUT FILE FORMATS

Formats for object photometry output can be done in 3 different ways, INTERNAL, COMPLETE and INCOMPLETE. Warmstarts are possible only from the INTERNAL or COMPLETE types, and additional models will only support outputs in the INTERNAL and COMPLETE types as well. It is important to note that the INTERNAL format measures x and y positions assuming that the center of the first pixel in a row or column of the image is 1.0, whereas for the COMPLETE and INCOMPLETE formats, the centers of the first pixels are measured as 0.5, in agreement with most conventions.

INTERNAL format.

This is the form in which DoPHOT internally tracks the objects and their associated properties. For each object the following quantities are reported (the last 5 are from empirical PSF fitting, the remainder from analytic fitting or aperture measures):

1) Object Number
2) Object type
3) Fit sky value
4) x position
5) y position
6) Height (maximum) of fitted model PSF in Data numbers
7) σ_x^2 (see section 7.2 for the form of fitted function)
8) σ_{xy} (as above)
9) σ_y^2 (as above)

9.5) *optional extra parameters from alternate models

10) Aperture flux (in DN)
11) Aperture sky value
12) Aperture magnitude minus fit magnitude
13) Estimated error in fit magnitude
14) Extendedness parameter
15 Sky value from empirical fit
16 Height (maximum) of fitted empirical PSF in Data numbers

17 x position from empirical PSF fitting

18 y position from empirical PSF fitting

19 Estimated error in fit magnitude from empirical PSF

The format of the output is:

```
%4d %2d %10.4E %10.2f %10.2f %10.3E %10.3E %10.3E %10.3E %10.3E %10.3E %7.3f
%7.3f %10.2E %10.3E %10.3E %10.2f %10.2f %7.3f
```

For Object type 8 (obliterations), item 7 and 9 above are the x and y dimensions of the obliterated subraster. For a rectangular obliteration, a value of 0.0 is stored in item 8 above and items 7 and 9 are the full x and y widths in pixels. For elliptical obliterations, the tilt of the ellipse in degrees in stored in item 8 above and items 7 and 9 are the semi major and minor axis widths. If an object is judged to be a cosmic-ray, in which case the value in item 8 is -1.0. Thus, for obliterating an ellipse, do not set the value of the tilt to identically -1.0 or 0.0, though -1.01 or 0.01 will do just fine. For similar reasons discussed below, avoid exactly 90.0 degrees as well. All other values are fine.

For the extended pseudogaussian model, where β_4 and β_6 are free parameters of the fit, the β_4 and β_6 values as well as an integer flag for each object are placed between items 9 and 10. The flag is 0 if the extended pseudogaussian model is indeed fit to the object in question and subtracted from the image. The flag is 1 if the extended pseudogaussian model did not converge on the object and so a standard pseudogaussian model (β_4 and β_6 fixed to the values specified in the parameter files) was fit to the object. For objects of type 2 which are flagged as fitted with a pseudogaussian model rather than an extended pseudogaussian model, this same pseudogaussian model is subtracted from the image, regardless the reported parameters in the β_4 and β_6 columns. For objects of types 1 and 3 which converge with the pseudogaussian model but not the extended pseudogaussian model, a typical star model (extended pseudogaussian) whose shape parameters are composed of the weighted averages of the converged type 1 stars are subtracted from that position– the same as for all other stars. However the shadow and error file parameters will reflect the fit from the converged pseudogaussian model, and the β_4 and β_6 columns in those files should be ignored.

COMPLETE format (also sometimes referred to as "external" format in the manual).

This is the form that is most convenient for the user. For each object, the following quantities are put out (the last 5 are from empirical PSF fitting, the remainder from analytic fitting or aperture measures):

1) Object Number
2) Object type
3) x position
4) y position
5) Fit magnitude
6) Estimated error in fit magnitude
7) Fit sky value
8) FWHM of major axis
9) FWHM of minor axis
10) Tilt (degrees: 0 implies major axis along positive x-axis; 90 is along positive y-axis)

10.5) *optional extra parameters from alternate models

11) Extendedness parameter
12) Aperture magnitude
13) estimated error in aperture magnitude
14) aperture sky value
15) aperture mag minus fit mag
16 Sky value from empirical fit
17 Fit magnitude from empirical PSF
18 x position from empirical PSF fitting
19 y position from empirical PSF fitting
20 Estimated error in fit magnitude from empirical PSF

The format of the output line is:

`%4d %2d %8.2f %8.2f %8.3f %6.3f %9.2f %9.3f %9.3f %7.2f %10.3E %10.3E %10.3E %10.3E %10.2E %8.3f %6.3f %9.2f %7.3f %9.2f %8.3f %8.2f %8.2f %6.3f`

If the object type is 8, the shape descriptors for rectangular obliterations are set as follows:

FWHM of major axis shows the longer side of the obliterated rectangle.
FWHM of minor axis shows the shorter side of the obliterated rectangle.
Tilt of 0.0 shows that the long side of the obliterated rectangle is along x.
Tilt of 90. shows that the long side of the obliterated rectangle is along y.

All other tilt values indicate an ellipse was obliterated rather than a rectangle and the 'FWHM' values are in fact the semi-major and minor widths of the obliterated ellipse.
Empirical fit magnitudes are flagged at 99.999 is the fitted intensity was zero or negative. It is flagged at -99.999 if no empirical PSF fit was attempted for that object.

INCOMPLETE format.

For the die-hard DAOPHOT compatibility freak.

Output quantities in order are:

1) Object Number
2) x position
3) y position
4) Fit magnitude
5) Estimated error in fit magnitude
6) Fit sky value
7) Object type (converted to floating number)
8) Extendedness parameter
9) Aperture mag minus fit mag

The format of the output lines is:

```
"%6d%9.2f%9.2f%9.3f%9.3f%9.3f%8d.%9.2f%9.3f
```

APPENDIX C: SUMMARY OF OBJECT TYPES

Type 1: A 'perfect' star. Was used in computing weighted mean shape parameters for stars.

Type 2: Object is not as peaked as a star, and has been interpreted to be a 'galaxy'. Shape parameters were calculated specifically for this object.

Type 3: An object found to be not as peaked as a single star was interpreted to be two very close stars. This is one component of such a close pair. Final fit to this component used mean shape parameters for single star.

Type 4: Failed to converge on a 4-parameter fit using mean single star shape. See section 7.9. Photometry is extremely unreliable.

Type 5: Not enough points with adequate S/N could be found around this object while attempting a 7-parameter fit to determine shape. Indicates a problem with the local environment, although a 4-parameter fit with mean star shape succeeded. Photometry is suspect, and this may not be a bona-fide star. See section 7.9.

Type 6: Too few points with adequate S/N to do even a 4-parameter fit with mean star shapes. Object NOT subtracted from image. See section 7.9.

Type 7: Object was too faint to attempt a 7-parameter fit. No object classification could be done, so it might be a faint galaxy or two closely spaced stars. Photometry is OK (from successful 4-parameter fit with mean shapes) provided the object is really a star. See section 7.4 for discussion of star/galaxy discrimination for such objects.

Type 8: These are obliterated regions due to data saturation or identification of cosmic rays. See Section 7.8. The shape descriptors are different for Type 8s than for other types: see Appendix B.

Type 9: If an attempt to determine the shape of an object fails to converge, it is classified as object type 9. See section 7.9. Photometry is unreliable.

Adding 10 to any of these image types means that the position was fixed if `FIXPOS` was set equal to 'YES'.