HSTphot User's Guide

version 1.1.7b

Andrew Dolphin

adolphin@as.arizona.edu

June 2005

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

Contents

1 Introduction

HSTphot is a stellar photometry package specifically designed for use with HST WFPC2 images. It uses a library of Tiny Tim PSFs for different locations of the star on the camera as well as locations of the star's center within the pixel, to center the star and to find its magnitude. HSTphot is optimized for running in undersampled conditions (like WFPC2), and could certainly be modified, given some effort, to other instruments if undersampled conditions are present. It would not be worth the effort, however, to modify HSTphot for use in non-undersampled conditions, as it would run very inefficiently.

A number of utility programs are also included with the HSTphot distribution, which are covered in sections 4 (dealing with photometry utilities) and 5 (dealing with analysis utilities).

The literature reference for HSTphot is Dolphin, A. E. 2000, PASP, 112, 1383. The PASP paper emphasizes the algorithms used by HSTphot and tests of its reliability, and is thus useful reading for those who wish to know more about the details of HSTphot. A CTE and calibration solution using HSTphot also exists on the author's web site, updated from the one given by Dolphin, A. E. 2000, PASP, 112, 1397.

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

Andrew Dolphin
adolphin@as.arizona.edu

2 Installation

To install HSTphot, you must decide on a location where you want the files to reside, and uncompress and unpack the distribution into that directory. HSTphot requires about 3 Mb for the sources and binaries, and 13 Mb per filter. The PSFs are distributed separately, as you only need PSFs for the filters that you will actually be using.

2.1 Makefile

The Makefile contains five lines at the beginning which may be edited for your particular system. The first two select the compiler and compiler options. The default Makefile will run gcc with options -O3 for optimization (unnecessary but useful) and -lm for math libraries (essential).

The third line (-DLINUX) needs to be commented out if you are not compiling HSTphot on a machine with flipped 32-bit float byte order. Computers with Intel CPUs (most linux machines, as well as Macs with Intel chips) will need this line in default form. Most Sun workstations and Macs with Power PC chips need the line commented out with a hash (#). I am told that VMS machines should leave the flag in.

BASEDIR should be set to the location that HSTphot has been installed into (the directory that Makefile resides in). TTDIR is used only if you have Tiny Tim installed and will be computing your own PSFs. In such a case, set TTDIR to the directory in which tiny1 and tiny2 are located. For both directories, the quotes must remain in their default format (' " directory name " '), and trailing slashes should not be given.

If you will be computing PSFs, and do not have Tiny Tim 5.0b installed, you will need to install it. Tiny Tim is written by John Krist and Richard Hook, and information is located at http://www.stsci.edu/software/tinytim/tinytim.html.

Also, if you will be computing your own PSFs, and csh is not located in /bin/csh (type "which csh" to find out), then you will need to edit the file "psfs/makettpar.c", changing "/bin/csh" to the correct location of csh.

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

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

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

  ```
  #define LINUX
  #define BASEDIR ‘‘/data0/dolphin/hstphot’’
  #define TTDIR ‘‘/data0/dolphin/tinytim50a’’
  ```

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

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

You should probably add the directory in which HSTphot is installed to your path. (This is done by editing your .cshrc or .bashrc file in your home directory.)

2.2 PSFs

You can either make your own PSFs or use precomputed ones. Making your own will take a very long time, but is necessary if you are not using a filter for which precomputed PSFs are available. If you will be building your own PSFs, you must have Tiny Tim installed, and must set the TTDIR flag in Makefile before compiling.

Precomputed PSF files are available via the web, and are separate for each filter. To get these, you should place the FXXXX.tar.gz file in the psfs/ directory, uncompress, and untar. The resulting file will be named X.6.5.psf, where X is the suffix for the filter. Precomputed PSFs are currently available for the following filters: F170W, F300W, F336W, F380W, F410M, F439W, F450W, F467M, F547M, F555W, F569W, F606W, F622W, F675W, F702W, F785LP, F791W, F814W, F850LP, and F1042M. PSFs for the other ultraviolet filters have not been generated (and will probably not be, unless I have need for them), nor have PSFs for the narrow-band filters. There are also default residuals and aperture corrections for F555W and F814W.

To build your own PSFs, enter the "psfs" directory, and type "makettpar FILTNAME," where FILTNAME is the name of the WFPC2 filer (ie, F555W). This will create a directory with the same name as the filter, which includes parameter files for use with Tiny Tim and a script "makepsf". You can move to the new directory, and type "makepsf" to start it running. Depending on your system and the filter (wide filters take the longest), it may take anywhere between 8 hours to a few days to finish.

If you wish to change the default PSF computation, you will need to edit the parameter files and scripts before running makepsf. Two things of note can be changed. The jitter values are located in the parameter files (*.par), and are set by default to smaller than the values suggested in the Tiny Tim manual. The values set here by default were determined by experiment to match observed PSFs better than the recommended values.

Also, located in the makepsf script, is the center-to-edge QE difference. The default value is 0.07, meaning that a photon hitting the center edge (not the corner) of a pixel is 7% less likely to be detected than one hitting the dead center of the pixel. What experimentation has been done indicates that this is a reasonable setting. The values following this, 13 and 15, give the amount of additional smoothing of the Tiny Tim PSF that is needed to reproduce the observed PSF sharpness. This is also a model of charge diffusion. These values were determined empirically as well.

Tiny Tim does not modify PSFs for geometric distortion, so the PSFs from the library are modified within HSTphot. They are expanded in both directions to account for the geometric distortion. An additional PSF modification is made to account for the 34th row error, with the PSF spread in the Y direction whenever it crosses an affected row. The images are multiplied by the correction factors for both effects, so that the input data should be in the processed form downloaded from STScI. Note that the PSF corrections do not affect the central positions of stars, so the *distort* utility described later in this manual is required to correct the stellar positions.

2.3 Filters

If you compute PSFs for filters other than the ones included, you will also probably need to add information to the "filters.dat" and "filters.h" files.

First you will need to insert three lines into the first set of information. Filters should be listed in order of central wavelength. The format is as follows:

- The first line is the name of the filter (ie, F555W). Make sure it is capitalized.
- The second line is the standard bandpass that the filter transforms to (again capitalized; V for F555W), followed by a suffix for photometry for that filter (lower case, v for F555W). Two filters may share the same bandpass of course (both F547M and F555W transform to V), but the suffixes must be unique (for example v2 for F547M and v for F555W).
- The third line lists the zero point for the untransformed photometric system.
- The fourth line lists the number of transformations, which can be zero if a filter does not transform (ie, the UV or narrow-band filters). After this line come the transformations, each line with the following information:
  - standard bandpass used for the transformation (cannot be the same as the filter's standard bandpass)
  - linear color term
  - quadratic color term
  - zero point

It should be noted that, although HSTphot uses my zero point calibration, the values in filters.dat are from Holtzman et al. (1995). (UV and Narrowband calibrations are taken from the VEGAMAG system in the WFPC2 handbook.) The calibration modifications are made in the CTE() function within *hstphot*.

Editing filters.h is also necessary. First, change the definition on the first line to reflect the total number of filters listed in filters.dat. After this, you will need to adjust the correction arrays and insert appropriate values. bgcorr and bgcorr_warm contain the empirically-determined dependence of measured background on star brightness. (At the distance used by *getsky*, there is still a measurable contribution of light from the star.)

One of the calibration corrections (a filter and temperature dependence) is also filter-dependent. Finally, there is a filter-dependent plate scale correction. Without appropriate calibration data, it is probably impossible to determine the zero point correction, but the other values can be estimated reasonably well.

2.4 Other Operating Systems

HSTphot has only been compiled and run on Unix systems. You are welcome to attempt to port it to a VMS or DOS system, and I am interested to know if it is successful.

3 Preprocessing Steps

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

The raw input data HSTphot must be plain FITS files, in 32 bit IEEE floating point format. This is the format in which data from the STScI archive is provided. Before running *hstphot* to obtain photometry, some initial preprocessing steps are required. These are listed in the order in which they should be done.

3.1 *fitsclean*: Strip Vectors from FITS

```
fitsclean <base> <output>
```

If receiving data from a source other than the STScI archive, the FITS file may contain extra information other than the header and 10240000 bytes of data. *Fitsclean* will attempt to remove that data so that it can be read by crclean. Input data to *fitsclean* is in the format <base>d and <base>h; the output will be a standard fits file.

3.2 *mask*: Mask Image Using Data Quality File

```
mask <fits image> <DQ image>
```

Mask uses the data quality image (*.c1f.fits in the STScI archive filename convention) to mask bad pixels and other image defects in the data image (*.c0f.fits). Bad pixels (types 1, 2, 4, 16, 32, 128, 256, and 512) are set to the bad data value (usually -100), while saturated or likely-saturated pixels (type 8 or with a value of at least 3500) are set to the saturated data value (usually 4095). The original fits image is moved to <filename>.old, and the masked image replaces the original.

A masked or saturated pixel is skipped by all other HSTphot routines - it is not used in sky determination, photometry, aperture corrections, etc.

Mask has the most flexible FITS reader of any in the HSTphot package, and will read either 32-bit int or 32-bit float data. It can also read scaled data. *Mask* will also compute read noise if it is not given and will determine whether the image was taken with warm or cold temperatures. Because of these features, it is mandatory that *mask* be run on all data before any other routines (except *fitsclean*, if needed).

3.3 *getsky*: Calculate Sky for Image

```
getsky <fits base>

getsky <fits base> 1

getsky <fits base> 2
```

Getsky is a mandatory part of the preprocessing procedure. It will read in <fits base>.fits, and create a file called <fits base>.sky. The sky computation is made by taking all pixels in an annulus around the pixel in question, and using a robust mean estimator to determine the sky value.

The second call of *getsky* (with the 1 as a second parameter) will run in abbreviated mode; instead of measuring the sky value at every pixel the program will measure the sky value at selected pixel values and interpolate to compute the sky image. This is recommended if the background is variable and you plan to use either sky option within *hstphot*.

The third call of *getsky* (with the 2 as a second parameter) will run it in fast mode; the routine will compute a quick sky measurement for each chip and set all sky values in that chip to that value. This is useful if the background is completely flat and you plan to use either sky option within *hstphot*; however option 1 runs sufficiently quickly that there really isn't a need for using this option.

The sky image cannot be directly viewed (it is not a fits file), but can be viewed with a little work. First, copy the sky file (X.sky) to a fits filename (X.sky.fits). Second, cut and paste the fits header from the original image (X.fits; the header is the first 2880 bytes) to the start of the sky image (X.sky.fits). You should then be able to view the image.

3.4 *crmask*: Mask Cosmic Rays in Unregistered Images or Different Filters

```
crmask <reg factor> <σ threshold> <use minimum value> ( <fits base>
    <scale> <minscale> <maxscale> <dx> <dy> per image )
```

Crmask is HSTphot's cosmic ray removal routine. It has the capability of cleaning images that are not perfectly aligned (such as dithered data), as well as handling images from different filters (if needed). The identification of cosmic rays is made on a pixel-by-pixel basis, as follows.

In order to avoid clipping stars, the procedure is conservative. "High" and "low" values for each image are determined, based on the highest and lowest values of the pixel in question and the surrounding eight pixels (after subtracting the sky). (The "high" value for any one image is computed using the maximum of the pixel in question, 0,5 times the four adjacent pixels, and 0.25 times the four diagonal pixels; the "low" value is the lowest of the nine pixel values.)

From these values, "high" and "low" base count rates for the pixel are determined, using either the minimum or median values of the individual-image values. Setting <use minimum value> to 0 uses the medium; setting to 1 uses the minimum. If different filters are being used, the count rates are adjusted according to the scale settings (see below).

After determination of the base count rates for a pixel, each image is examined. The base count rates are adjusted for the exposure time (and filter, if different), and minimum and maximum acceptable values determined using

$$min = base_{lo} + sky - \sqrt{(\sigma\text{threshold})^2(\frac{base_{lo} + sky}{GAIN} + RNOISE^2) + (\text{reg factor})^2 base_{lo}^2}$$

and

$$max = base_{hi} + sky + \sqrt{(\sigma\text{threshold})^2(\frac{base_{hi} + sky}{GAIN} + RNOISE^2) + (\text{reg factor})^2 base_{hi}^2}.$$

In both equations, *sky* is the sky value of the image being examined. If the image has a value between the minimum and maximum values, the pixel is kept; otherwise it is masked.

For normal cleaning of same-filter images, σ threshold should be about 3.0 and the registration factor no more than 1.0. It should never be necessary to use the minimum value. In this case, the scale, minscale, and maxscale parameters should all be zero, and dx and dy PC shifts relative to a "reference" image in pixels. If a very large number of frames are being run through *crmask*, you may wish to run it two or three times as there may be more than one image with a cosmic ray at one pixel.

If you will be searching for variable stars in your data, you will likely want to run *crmask* with a higher registration factor. A value of 5 will safely pass a variable star with an amplitude of 1.5 magnitudes. If you have many epochs of registered data and two or three images per epoch, you may wish to run *crmask* in two passes. For the first pass, run all frames with the high registration factor; in the second, clean images for each epoch separately with a registration factor of 1.

Finally, if you have two-filter data with only one image per filter, *crmask* can be used to clean the worst of the cosmic rays. For this example, we will assume the filters are V (F555W) and I (F814W). If the typical object has a F555W-F814W color of 1.0, the scale for F555W should be one more than that for F814W, such as 1.0 and 0.0 respectively. The overall settings of the scales do not matter, so long as the difference between the two is correct. The minimum and maximum scales give the permissible range in color that would be allowed. For example, if the smallest F555W-F814W color is 0.0, this would correspond to a minimum F555W scale of 0.0 (ie, 0 magnitudes brighter than the average F814W setting of 0.0) and a maximum F814W scale of 1.0 (ie, 0 magnitudes fainter than the average F555W setting of 1.0). Likewise, a maximum color of 2.5 would result in a maximum F555W scale of 2.5 and a minimum F814W scale of -1.5. Note that all scale values are expressed in terms of magnitude differences, rather than flux fractions.

The masked FITS file replaces the original (<base>.fits); the original is moved to <base>.old.fits. *Crmask* can be run multiple times; two passes is recommended for 3-4 images and three passes for 5 or more images.

3.5 *coadd*: Combine Registered Images

```
coadd <input fits files> <fits output>
```

Coadd produces a deep frame from images that are properly registered (same pointing and rotation). The input images need not have the same gain settings or exposure times. (In fact short GAIN=14 images are routinely combined with long GAIN=7 images.)

3.6 *hotpixels*: Remove Hot Pixels

```
hotpixels <fits base>
```

Because the cosmic ray cleaning program uses somewhat of a median filter routine, hot pixels, which are equally high in all images, will not be located and removed. The *hotpixels* routine can only be run after *getsky*, and will attempt to locate and remove all hot pixels. *Hotpixels* is recommended, since hot pixels will create false detections in hstphot. Such detections can easily be removed later with a sharpness cut, but can throw off the PSF solutions in the meanwhile.

An average and standard deviation of all eight adjacent pixels is calculated (after sky subtraction), and a pixel is deemed to be hot if its value is higher than 10 times the average of the adjacent pixels and if it is more than 7 standard deviations away. Such a routine should adequately locate hot pixels that are on blank sky, although it will not locate those that are on stellar images or adjacent to other hot pixels. The strict limits are used to avoid any chance of cutting off WF stellar profiles, which can have very sharp peaks as well. If a hot pixel is found, it will be masked with the bad data value.

Hotpixels can be run multiple times; generally two is sufficient to remove all pixels. The masked FITS file replaces the original (¡base>.fits); the original is moved to <base>.old.fits.

4 Running HSTphot

The raw input data into *hstphot* must be plain FITS files, in 32 bit IEEE floating point format. This is the format in which data from the STScI archive is provided. Before running HSTphot, you must solve for the sky image with *getsky*. Both programs are described in the previous section.

A word about the chip and position standards for HSTphot. HSTphot gives three values, a chip, X, and Y. The chip can be 0, 1, 2, or 3, for PC, WF2, WF3, and WF4, respectively. The X and Y positions are made such that the integer value means the star is centered in the lower left corner of the pixel, and pixel values range from 0 to 799 for each chip. This will give positions identical to DoPHOT, but 0.5 lower in both X and Y than DAOPHOT.

4.1 HSTphot

The workhorse of the HSTphot package is the routine *hstphot* (named "multiphot" in version 1.0). *HSTphot* takes the input images and produces calibrated flight system and standard (UBVRI) magnitudes. Because of the complexities of such a task and of combining the magnitudes, hstphot includes alignment and aperture corrections, and thus the output data are generally ready for scientific use (after trimming bad detections).

```
hstphot <output> <ind threshold> <total threshold> <chip> <Xmin>
    <Xmax> <Ymin> <Ymax> <options> ( <fits base> <dX> <dY> ) per
    exposure <reference image>
```

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

Two detection thresholds are used: independent (minimum S/N for a given image or filter for star detection and output) and total (minimum total S/N for a star to be kept). Typical values for two-band photometry are 3.5σ detections in one or more images and 5.0σ overall detections. It is advisable to run *hstphot* with a lower signal-to-noise threshold than the final science product requires and apply signal-to-noise cuts to the photometry at a later stage; the recommended thresholds given above work well when requiring 5σ per filter in the final data.

To photometer all four chips, set the chip to -1 and the X and Y limits to 0. You may also photometer a part of one chip; the limits should be set accordingly. Note that throughout the HSTphot package, the PC is number 0 and the WFC chips are numbers 1, 2, and 3.

The options flag should be set zero for default (I recommend option 512 by default unless option 2 is being used). The following values can be added to turn on/off various options:

1: Select aperture photometry instead of PSF-fit. Selecting this option will turn HSTphot into a small aperture photometry package (2 WFC pixels, 3 PC pixels), rather than a PSF-fitting package. Given that the PSF fit uses all information used by the aperture photometry package, as well as the expected shape of the star, the default

PSF-fitting routine should be used in nearly all cases. If aperture photometry is used, option 4 (turn off PSF residuals) is also recommended.

2: Turn on local sky determination. This will determine a "local sky" value (actually, a correction to the original *getsky* sky value), using the values immediately outside the photometry aperture (typical average distance of 4 WFC pixels or 5.5 PC pixels). This is quite useful in fields with rapidly-varying background (on a scale smaller than an arcsec), such as stars in nebulae, galaxies well beyond the Local Group, or globular clusers.

4: Turn off PSF residual determination. This setting should be used only if setting 1 (aperture photometry) is used, if there are no well-measured stars in the field, or if the galaxy is distant enough that clusters are likely to be mistaken for stars (and there are many clusters). No PSF residual determination will create essentially no systematic error for small aperture photometry or for bright stars in PSF-fitting photometry, but can create a systematic error of up to ± 0.1 magnitudes for faint stars, with an RMS of about ± 0.05 magnitudes. *hstphot* will use the default residual image for the filter (if available), which is an improvement but still subject to large systematic errors for faint stars.

8: Turn off aperture corrections. This setting will be beneficial in the case in which there are no good aperture stars. If this option is selected, *hstphot* will use the default aperture corrections for the filter (if available), which are probably accurate in general to 0.02 magnitudes. If no default value is available, the corrections are assumed to be zero and have an uncertainty of about 0.05 magnitudes.

16: Turn off alignment, using the user-entered dX and dY values. This should probably never be used, unless the alignment routine is messing up badly on an image.

32: Turn off CTE and zero point corrections.

64: Run artificial star tests (see below for details).

128: Force all objects to be classs 1, 2, or 3 (stars).

256: Use semi-empirical PSF adjustments (should be used with option 4).

512: Re-fit sky (as in *getsky*) during photometry (recommended unless using option 2).

1024: Use predetermined star list instead of searching for stars.

2048: Select a weighted PSF-fit instead of a standard PSF-fit. The difference is that each pixel is given additional weight (PSF/σ^2), forcing the program to focus more on the central pixels and less on outer pixels. This can be highly advantageous in very crowded fields. Since statistical principles are violated in this fit, I would recommend that you also make a standard photometry run for comparison.

4096: Distribute artificial stars evenly across the chip. (The default is to distrubute artificial stars according to the distribution of light.)

The fits base for the input images is identical to *getsky* and *hotpixels*. Because dithered images can be processed, an initial estimate of the PC position offsets (relative to the reference image) must be given. The most common dithering patterns involve values of 0.00, 5.49, 10.98, and 16.47 pixels. The reference image is listed after the data images. If none is given (“”), the first image is used instead.

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

4.2 HSTphot Options

Three of the options require additional parameters. If you use more than one of these, the additional parameters should be put in the order given here (option 1024, then 64, then 256).

Use of option 1024 allows you to define a predetermined star list for the photometry. This can be useful if you are re-running the photometry with slightly different parameters, as it greatly speeds up *hstphot*. One additional parameter is required, the filename of the star list.

```
hstphot . . . <filename>
```

The file should contain four columns: chip, X, Y, and object type (1-5).

HSTphot contains an artificial star routine, which can be accessed by selecting option 64 as noted above. A grid of artificial stars is generated on a 2-diminsional CMD and distributed according the the flux of the image. This requires ten additional parameters to be appended to the command line, as follows.

```
hstphot . . . <Color1> <Color2> <Mmin> <Mmag> <Cmin> <Cmax>
    <PC wt> <WFC2 wt> <WFC3 wt> <WFC4 wt>
```

The first two new parameters are the colors to define the grid, which should be U, B, V, R, or I. In general, you will want the extreme colors in your photometry to be used here. For example, if you have BVI photometry, B and I should be used as the colors. Second are the limits of the CMD grid, with minimum and maximum magnitude (as measured in color1) and in color (as measured in color1−color2). Magnitude steps are multiples of 0.5, while color steps are by 0.25.

Finally are the weights of the four cameras. A weight of 1 will produce about 900 artificial stars in each CMD bin per chip and will roughly double the amount of time required for *hstphot* to run per 100 CMD bins. Alternately, a negative weight will attempt to distribute artificial stars on the CMD in accordance with the number observed. A weight of −1 places a comparable number of artificial stars as the number of observed stars.

The output from the artificial stars is placed in <output>.fake. At the start of each line are the input position (chip, X, and Y), standard magnitudes (UBVRI, with the three extra filters estimated), and the input WFPC2 flight system magnitudes in any filters used. Following this information is a set of data identical to the regular *hstphot* photometry output.

Use of option 256 (semi-empirical PSF residuals) requires additional command-line parameters, placed after the reference frame (and after the artificial star parameters, if applicable). There are twelve new parameters per image, as follows:

```
hstphot . . . <PC sh=0> <PC fit> <PC function> <WFC2 sh=0>
    <WFC2 fit> <WFC2 function> <WFC3 sh=0> <WFC3 fit>
    <WFC3 function> <WFC4 sh=0> <WFC4 fit> <WFC4 function>
```

The two values for each chip can be determined by *calcsh*, and correspond roughly to the FWHM for a fit to a "sharpness=0" PSF and for the observed PSF in the image. Instructions for use of *calcsh* are given in Section 5.3.

The advantage of using option 256 + option 4 instead of the default PSF solution comes when there are no bright stars (ie, Cepheid work) for which a pixel-by-pixel PSF can be built. In such a case, the PSF is modeled as the default PSF plus the difference between two Gaussians, with the widths calculated by *calcsh*.

4.3 HSTphot Output

HSTphot's output is quite complex, and breaks into simple sections. All information for one star is contained on one line.

The first set of values are the chip, position (X and Y), and global solution information (χ, signal-to-noise, sharpness, roundness, major axis), and object type. Object types are as follows:

- 1: good star
- 2: possible unresolved binary: two stars combined during photometry iterations
- 3: bad star: centered on saturated pixel or bad column
- 4: single-pixel cosmic ray or hot pixel
- 5: extended object

A good, clean star in an uncrowded field should have a χ of 1.5 or less, with values of up to 2.5 reasonable. A blended but unresolved pair can have a χ up to 4. In very crowded fields, χ values for good stars of up to 5 are possible. The sharpness is zero for a perfectly-fit star, positive for a star that is too sharp (perhaps a cosmic ray), and negative for a star that is too broad (perhaps a blend, cluster, or galaxy). In the uncrowded field, good stars should have sharpness values between -0.3 or 0.3, with up to ± 0.5 reasonable. Roundness is usually not used for star discrimination, but can be used to discriminate between extended objects and diffraction spikes.

After this come the blocks of photometry. The first blocks are for any filters for which more than one image was entered into *hstphot*, with filters ordered by increasing central wavelength. After this comes the photometry from each image. Each photometry block contains the same information: counts, background, flight system magnitude, standard magnitude, magnitude uncertainty, χ, signal-to-noise, sharpness, roundness, and crowding. Note that transformations are determined by the combined values for each filter, with the difference betwen combined instrumental and standard magnitude then applied to each exposure. Objects are ordered first by chip (0 to 3), and second by signal. Magnitudes are calibrated first using the Holtzman et al. (1995) calibration recipe, with CTE and zero point corrections made as per Dolphin (2000). Tests on multiple images of a single field show a limiting accuracy of 0.011 to 0.014 magnitudes for *hstphot* photometry. The crowding parameter is the change in brightness in the star if neighbors are not subtracted, as measured in magnitudes. Generally this is zero if the star has no close neighbors or positive if the star has neighbors, but can occasionally be negative from roundoff errors in the calculations.

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

An image residual and PSF residual (*.res and *.psf) show the subtracted image and PSF alterations. As with the sky image, they do not contain FITS headers. It is worth examining the image residual to ensure that most of the image was indeed subtracted.

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

4.4 Filters

As noted in the installation instructions, HSTphot includes a database of filters, zero points, and transformations, as “filters.dat”. If you wish to run HSTphot on an image using a filter not included in filters.dat, you will need to add the appropriate information to the file and to filters.h, as noted above. You will also likely need to compute PSFs for the filter using Tiny Tim.

Each filter has a name (ie, F555W), a standard bandpass (V), and a unique suffix (v). When naming HSTphot output files, you should use a base name that will be shared for all photometry for a given pointing and the suffix for the filter. For example, if you have F555W and F814W images for a single LMC pointing, you may wish to make the output files lmc.v and lmc.i, respectively. This will later aid programs such as filtcombine, which locate, combine, and transform the individual photometry lists.

4.5 Routines

The heart of HSTphot is a routine which solves for a star’s center and brightness. The program uses a library of Tiny Tim PSFs which has PSFs for an 8 by 8 grid on each chip, or 256 total positions. At each position on the PC chip, a 5 by 5 grid of possible sub-pixel centerings of the star is stored, and a 10 by 10 grid is used in the WF chips. (This large amount of information is why PSF computations can take a very long time.)

When photometry is attempted on a given pixel, a brightness solution using PSF fitting is made on each of the sub-pixel positions. (Since the PSFs are calculated using radii of 0.5 arcsec, aperture corrections should be unnecessary in theory, although this is not the case in practice.) The sub-pixel position returning the smallest χ / S/N ratio is then selected as the star's center. Solutions are also made in adjacent pixels to determine if the star's true center is in a different chip.

The overall structure of the photometry program is that for each chip, a two-pass search for stars is made, followed by an iterative solution. Each search pass locates peaks in the image brightness, and attempts photometry on each that is found. Should two peaks correspond to one star, only one is put into the star list. The first pass is made on the image minus the sky; the second is made with stars located in the first pass subtracted. To avoid false detections in the wings of bright stars, the second pass will only locate stars that have brightness peaks in the original frame as well as in the subtracted frame. Additionally, any pair of stars centered within 1.5 pixels of each other is combined into a single star.

The iterative solution is made for up to 25 iterations, depending on how long is needed for the solutions to converge. For each iteration, stars whose neighbors' photometry changed in the previous iteration will be remeasured. Each stars is measured with the sky and all other stars subtracted, meaning that in theory, the frame is blank except for the star being measured and stars below the detection threshold.

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

After all stars in a frame converge (or after 25 iterations have been made), a "fine" solution is made for all stars, attempting to determine the best positions to 0.01 pixels. HSTphot's measured astrometric accuracy with this method is approximately 0.03 pixels in both X and Y directions. After each star is solved, its data is output into the photometry file. The external accuracy of the photometry of bright stars appears to be roughly 0.01 magnitudes for multiple images or pointings, and 0.02 magnitudes for a single pointing only.

5 Other Utility Reference

5.1 *arith*: Add/Subtract Images

`arith <output> ( <input> <scale> ) per image`

Arith will add images, multiplying each by a scale factor. For example, to subtract an image by the average of two adjacent (in time) images, a scale factor of 1.0 for the image and -0.5 for the adjacent images would produce the desired result. This will be most useful for variable star work, when trying to determine whether or not a star is variable. If you wish to use the output image for other purposes (such as running photometry on an Hα subtracted R frame), the base image (the R frame in this case) should be the first input image in your list.

5.2 *calcclus*: Calculate Cluster Information

Calcclus is a complex and undocumented routine that attempts to derive physical properties of slightly extended objects (sharpness< 0) in *hstphot* output.

5.3 *calcsh*: Calculate Semi-empiricial PSF Residual

`calcsh <filter> <chip> <sharp> <e- counts> <e- counts + noise`^{2} `>`

See the *hstphot* description of option 256 for an explanation of when and why this might be useful. To use *calcsh*, you will want to plot sharpness versus magnitude (individually for each chip), and determine "typical" sharpness, counts, and background for a star at a magnitude level. The counts and background should be converted to electrons, and the background should include the number of exposures times the square of the readout noise. For example, if the average is 500 DN of counts and 10 DN of background for an image combined from 4 gain=7 images (noise = 5 electrons), you would use counts=3500 and background=170 when invoking *calcsh*.

5.4 *distort*: Correct Geometric Distortion

`distort <input> <output> <global>`

Distort will correct the positions of stars in an image, using the Holtzman et al. (1995) distortion corrections. If the global flag is set to a non-zero value, the output corrections will be on the global coordinate system. Otherwise, the values will be rotated and scaled back to the individual chip coordinates, so that (400,400) will not be moved.

5.5 *extract*: Create One Chip FITS File

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

Extract will extract one chip from the stacked FITS frame. The primary use for this is to allow SAOimage or ximtool to display one of the chips. Chip numbers, as usual, are 0-3.

5.6 *fake_image*: Generate Simulated Image

```
fake_image <filter> <stars> <PC background> <gain> <rd-noise>
    <output image>
```

```
fake_image <filter> <stars> <PC background> <gain> <rd-noise>
    <output image> <noise>
```

Fake_image creates a simulated image. The filter should be the usual filter name (i.e. F555W), the background and readout noise in DN, and the gain in electrons/DN. Setting noise to 1 will add Poisson noise to the image; leaving blank produces a noiseless image).

The list of stars should contain four columns: chip, X, Y, and star brightness (again in DN). Setting X or Y to negative values will randomly place stars on the image; a revised star list (with X and Y values set) will be output to the screen.

5.7 *fitsinfo*: Information From FITS Header

```
fitsinfo <file>
```

Fitsinfo will display useful information from the header of an input fits file, such as target name, exposure time, gain, filter, etc.

5.8 *hstmosaic*: Create Mosaic

```
hstmosaic <input> <output>
```

HSTmosaic will generate a standard "bat wing" mosaic, with the four chips aligned and trimmed. Bad columns and pixels will be filled in with values from adjacent pixels to produce a clean image.

5.9 *lf*: Luminosity Function

```
lf <photometry> <Mmin> <Mmax> <bins/mag>
```

LF makes a simple luminosity function from a file containing the star magnitudes (one per line). The output is the magnitude limits for each bin, the number of stars observed, and a Sobel filter value.

5.10 *makergb*: Make RGB .BMP image

```
makergb <blue mosaic> <blue min> <blue max> <green mosaic>
    <green min> <green max> <red mosaic> <red min> <red max>
    <output> <scale> <color contrast>
```

```
makergb <blue mosaic> <blue min> <blue max> ‘‘’’ <blue fraction>
    <red fraction> <red mosaic> <red min> <red max> <output>
    <scale> <color contrast>
```

Makergb makes a three-color .BMP image from either two- or three-band data. If three bands are available, the input should be the three mosaics (from *hstmosaic*) and minimum and maximum values for each. If only two bands are available, the blue and red values will be taken from two sets of data, while the green is a weighted average of the two (weights based on the "blue fraction" and "red fraction" parameters).

The output parameter is the output filename, while the scale function determines the display scaling:

- 0: linear
- 1: $x^{1/4}$
- 2: $x^{1/3}$
- 3: $x^{1/2}$
- 4: $\ln(x)$

In practice, the $x^{1/3}$ and $\ln(x)$ options appear to produce the most realistic-looking pictures.

When the image is scaled, only the intensity level (red+green+blue)/3 is scaled; the ratios between the three color levels are preserved. This is an attempt to prevent the colors from washing out to white in the brighter parts of the image. The color contrast parameter allows for a fine tuning of the color. When the three colors used correspond to three adjacent passbands (ie, UBV, BVR, or VRI), this scale parameter should be set to 1.0. When there is more wavelength separation, the color scale should be decreased (UVI photometry should have a scale of 0.5), or when there is less wavelength separation it should be decreased (BV photometry should probably have a scale of 2.0).

5.11 *photcompare*: Star-by-star Comparison

```
photcompare <input 1> <input 2> <output>
```

Photcompare will take two output photometry lists and match stars. This should rarely be necessary (since *hstphot* can photometer multiple images simultaneously), but can occasionally be useful comparing different photometry lists. The output file will have the chip, X, and Y positions (averaged), the remainder of the input file 1 data, and the remainder of the input file 2 data.

5.12 *photmerge*: Merge *HSTphot* Outputs

```
photmerge <input 1> <input 2> <output> <dX> <dY> <dθ > <mode>
```

Photmerge merges photometry lists from two *hstphot* runs. This is usually unnecessary (since *hstphot* can photometer multiple frames), but is needed when rotation or large shifts (many objects moved to different chips) are present.

The input files are the same as the *hstphot* output files; the output file can be named anything. The dX, dY, and dθ values are the initial corrections that will be applied to the file 2 positions to correct produce file 1 positions. When correcting positions, the rotation is applied first, followed by the X and Y offsets. A good initial guess for dθ can be obtained from the PA_V3 settings at the two positions. The X and Y offsets can be estimated by using the *distort* routine.

Photmerge will attempt to improve the X and Y offsets, but will not the rotation. You may wish to run *photmerge* with slightly different dθ values to see which produces the smallest scatter in the alignment correction. In practice, the PA_V3 values are generally accurate to better than $0.1°$.

The output from *photmerge* contains the merged photometry list plus a merged .info file.

6 PSF Utility Reference

These utilities are all located in the psfs/ subdirectory.

6.1 *avdpsf*: Calculate PSF Residual and Aperture Corrections

```
avdpsf <output base> <hstphot output> <img 1> <img 2> . . .
    <img N>
```

Avdpsf is used to generate the default PSF residual and aperture corrections for a filter, taking values generated by *hstphot*. The image numbers are the image numbers in the *hstphot* output (0 through N-1).

6.2 *makepsf*: Compute PSF for Filter

```
makepsf <filter suffix> <PC PSF size> <PC sub-pixel centerings>
    <QE loss in edge>
```

Makepsf is generally not called from the command line. Information regarding it is located above in section 2.2.

6.3 *makettpar*: Make Tiny Tim Parameter Files and Script

```
makettpar <filter>
```

Makettpar is located in the psfs/ directory, and is used to generate the parameter files and script to compute a new PSF. To run *makettpar*, see the instructions in section 2.2.

6.4 *showpsf*: Compute PSF for Filter

```
showpsf <filter> <chip> <X/100> <Y/100> <dX> <dY>
```

Showpsf will display the values of one library PSF on the screen. The dX and dY values are the subpixel centering, and have ranges of ±3 in the PC and ±6 in the WFC (±0.6 pixels in both cases). PSF values are multiplied by a factor of 100,000 to make display easier.

7 HSTphot Recipe

This section outlines a recipe for photometry with HSTphot. Again, it is assumed that the data arrived in the standard format from STScI; if that is not the case some preliminary work with *fitsclean* may be necessary.

1. Run *mask* to mask out the bad pixels on each image.
2. Run *getsky* on each image to generate the sky image.
3. Run *crmask* for cosmic ray removal if there is more than one image. The rule of thumb is to run once if there are two images, run twice if three or four, and run three times if five or more.
4. If you have multiple images per filter that are well-registered, run *coadd* combine them and produce deep frames. If you are searching for variable stars, you should create separate images from different epochs.
5. Run *hotpixels* on each image to clean hot pixels.
6. Run *hstphot*. The output is as described previously in this manual.
7. Clean the photometry list from bad objects (types 3, 4, and 5 or bad values of χ, sharpness, or roundness).

After completing these steps, the *hstphot* output will contain the WFPC2 flight system and standard magnitudes for all stars that were determined.

8 Changes

Changes for version 1.1.7b (June 7, 2005)

1. The *arith* routine has been updated to provide FITS output suitable for photometry.

Changes for version 1.1.7a (December 1, 2004)

1. The default aperture correction (a function of the PSF adjustment) has been added to the .info file.

Changes for version 1.1.7 (November 18, 2004)

1. An additional output column has been added to indicate the effects of crowding.
2. The aperture correction routine has been rewritten. Instead of making a single correction, it now computes an aperture correction between the photometry and a small aperture, and a second correction between the small aperture and the standard 0.5” aperture.

Changes for version 1.1.6b (June 17, 2004)

1. The FITS reader in “mask” has been made more robust, to allow for unusual characters in the header.

Changes for version 1.1.5 and 1.1.5b (May 28, 2003)

1. Option 4096 for *hstphot* has been added. This ignores the “light distribution” map in fake star tests and distributes stars evenly.
2. S/N calculations are more accurate for cases in which both deep and shallow bands are used.

Changes for version 1.1.4 (March 9, 2003)

1. *crmask* corrects for the sky when making the registration correction. This makes for more pixels rejected, and means that you must calculate a sky image before masking the cosmic rays.
2. *hstphot*’s PSF star selection routine has been relaxed to allow more PSF stars. This allows HSTphot to run correctly on images with bad PSFs.
3. The PSF-fitting radius in *hstphot* was restored to its original values: 3.0 pixels in the PC, 2.0 pixels in the WFC for normal photometry; 2.1 and 1.4 respectively for small-aperture photometry.
4. Option 2048 for *hstphot* has been added. This weights the χ^2 minimization so that outlying pixels are less significant, producing generally deeper photometry and better fits.