

ssphot

A Photometry Package

Document Version 1.17

Prepared Using L^AT_EX

by

Stefan Spännare

E-mail: stefan@spaennare.se

July 12, 2007

Contents

1. Introduction

- 1.1 About this document
- 1.2 Some properties of the *ssphot* photometry package
- 1.3 About the contents of this document
- 1.4 Directories and files in the package
- 1.5 General information about the software
- 1.6 Disclaimer
- 1.7 Other photometry packages

2. Some useful theory for photometry

- 2.1 Analytical PSFs
 - 2.1.1 The Moffat function
 - 2.1.2 The Gaussian
 - 2.1.3 A comparison between the Moffat function and the Gaussian
- 2.2 Theoretical errors caused by noise

3. FINDSTARS star finding and photometry program

- 3.1 Algorithm
- 3.2 FSORT
- 3.3 Timing
- 3.4 How to use FINDSTARS and FSORT
 - 3.4.1 FINDSTARS
 - 3.4.2 FSORT

4. POLYFIT photometry program

- 4.1 Algorithm
- 4.2 How to use POLYFIT

5. Programs to make analytical PSFs

- 5.1 Programs to make analytical image PSFs
 - 5.1.1 `setmofff` and `setgausf`
- 5.2 programs to fit a Moffat function to the image of a star
 - 5.2.1 `moffit` and `moffit2`

6. Programs to make simulated stellar fields

- 6.1 `imstarsf` and `imstarsmf`

7. Programs to add noise to images

- 7.1 Poisson noise
 - 7.1.1 `poissonf`
- 7.2 Gaussian noise
 - 7.2.1 `normalf`

8. Astronomical image formats and image conversion programs

- 8.1 The (`*.fff`) floating point format
- 8.2 The GOP format (`*.imf`)
- 8.3 Image conversion programs

9. Some image processing programs

- 9.1 Programs to add and multiply images
 - 9.1.1 Addition of a constant to an image
 - 9.1.2 `imadd` and `imaddf`
 - 9.1.3 Multiplication of an image with a constant
 - 9.1.4 `immul` and `immulf`
 - 9.1.5 Addition and subtraction of two images
 - 9.1.6 `imimadd` and `imimaddf`
- 9.2 Programs to resample images
 - 9.2.1 Programs to sample up images
 - 9.2.2 `rebinu` and `rebinuf`
 - 9.2.3 Programs to sample up images
 - 9.2.4 `irebinu` and `irebinuf`
 - 9.2.5 Programs to sample down images
 - 9.2.6 `rebind` and `rebindf`
- 9.3 Programs to cut and merge images
 - 9.3.1 Programs to cut images
 - 9.3.2 `imcut` and `imcutf`
 - 9.3.3 Programs to merge images
 - 9.3.4 `imcomb` and `imcombff`
- 9.4 Some additional image processing programs
 - 9.4.1 `mmfw` and `mmfwf`
 - 9.4.2 `imsig` and `imsigf`
 - 9.4.3 `medianh` and `medianhf`
 - 9.4.4 `imrot` and `imrotf`
 - 9.4.5 `imshift` and `imshiftf`
 - 9.4.6 `slice` and `slicef`
 - 9.4.7 `plots`
 - 9.4.8 `imdot`, `imdot2` and `imdot3`
- 9.5 A simple aperture photometry program
 - 9.5.1 `aperf`

10. Some auxiliary programs

- 10.1 Some auxiliary programs for photometry
 - 10.1.1 `sscomp` and `sscomp3`
 - 10.1.2 `setpos`
 - 10.1.3 `setposlf`
 - 10.1.4 `cadd`
 - 10.1.5 `ctransf` and `ctransf2`
 - 10.1.6 `cparam`
 - 10.1.7 `chist`
 - 10.1.8 `int2mag` and `mag2int`
 - 10.1.9 `movavi` and `movavm`
 - 10.1.10 `magnoise`
 - 10.1.11 Programs to make groups for POLYFIT
 - 10.1.12 `c2g` and `group`
- 10.2 Other auxiliary programs
 - 10.2.1 `colcol`
 - 10.2.2 `sigcol`
- 10.3 A program to calculate horizontal coordinates
 - 10.3.1 `eq2hor`

11. Some Fast Fourier Transform using programs

- 11.1 Information about the FFT code
- 11.2 Programs to resample images using FFT interpolation
 - 11.2.1 Programs to sample up images
 - 11.2.2 `frebinu` and `fredbinuf`
 - 11.2.3 Programs to sample down images
 - 11.2.4 `frebind` and `fredbindf`
- 11.3 FFT convolution of images
 - 11.3.1 `convf`
- 11.4 Standard Lucy-Richardson deconvolution of images
 - 11.4.1 Algorithm
 - 11.4.2 How to use the program
 - 11.4.3 `lucyf`

12. Some examples how to use the programs

- 12.1 FINDSTARS measurement of a simulated stellar image
 - 12.1.1 To make the simulated stellar image
 - 12.1.2 To display the stellar image
 - 12.1.3 FINDSTARS measurement
 - 12.1.4 To compare the result with original data
- 12.2 POLYFIT measurement
- 12.3 How to fit a Moffat function to an image PSF
- 12.4 Lucy-Richardson deconvolution
- 12.5 How to display images
 - 12.5.1 How to make and display grey scale images
 - 12.5.2 How to make and display coloured images

13. Compiling and installing the package

- 13.1 Compiling under Unix and Linux
- 13.2 Compiling under Windows 98/2000/XP

14. References

- 14.1 Book references etc
 - 14.2 Internet references
- References

1 Introduction

1.1 About this document

This is a user manual for some photometry and image processing computer programs that the author made as a graduate student at Lund Observatory during the years 1992-1997. The package contains about 100 separate C-programs in total. Some of them are larger and more complicated, but most of them are quite small. Some additions and changes have been made since then. The main part of this document was written already summer 1998 but it has been much updated since then.

1.2 Some properties of the *ssphot* photometry package

- Fast and accurate routine for star finding and photometry in crowded stellar fields.
- A special routine for very accurate photometry of stars on uneven backgrounds.
- Routines for fitting analytical PSFs to stellar images.
- Routines for making simulated stellar field images.
- Many useful auxiliary image processing and photometry routines.
- Many useful image conversion routines.
- It is fast.
- It is simple to use.
- It is command line based.
- It is free ware.
- It is stand alone.
- It is written in standard C.
- It works on Unix, Linux and DOS under Windows 98/2000/XP.

1.3 About the contents of this document

The main purpose of this manual is to describe the star finding and photometry program FINDSTARS with the help program FSORT (section 3) and the program POLYFIT for photometry of well sampled stars on uneven backgrounds (section 4). Detailed descriptions are given for these programs. But many other useful routines are also described in this manual. For example the program **aperf** for simple aperture photometry (section 9.5) and several programs and tools to make simulated stellar field images (section 5, 6, 7, and 9). Section 8 is dealing with astronomical image formats and image conversion programs. It is recommended to read this section before using the programs. Some auxiliary programs are described in section 10. The programs **sscomp** and **sscomp3** are used to compare measured photometry data with original data for simulated stellar fields (section 10.1.1). A short theory section describing analytical PSFs and stellar errors caused by noise are included in section 2. A program **magnoise** to calculate theoretical such errors is described in section 10.1.10. The program **eq2hor** is used make accurate conversion between equatorial and horizontal coordinates for astronomical objects (section 10.3.1). Some programs that use FFT are described in section 11. Some useful examples how to use the programs (i.e. to make simulated stellar images and then measure the intensities of the stars) are found in section 12. Finally how to compile the C-programs is described in section 13.

1.4 Directories and files in the package

The *ssphot* package consists of the files "ssphot.tar.gz" (source and examples for Unix and Linux), "ssphot.zip" (executables, source and examples for Windows 98/2000/XP) and this documentation "ssphot.pdf" and "ssphot.ps.gz" as separate files.

The home page of the author, the *ssphot* photometry package and this document are found on the web-pages:

<http://www.spaennare.se/index.html>

<http://www.spaennare.se/ssphot.html>

Unpack the "ssphot.tar.gz" or "ssphot.zip" file whichever is appropriate. The SSPHOT directory then has the following contents (under directories):

EXE: executables and source for all the C-programs. Scripts to compile the programs are also present here. Note, for Unix and Linux these files must be compiled.

SRC: source for all the C-programs (the same as in the **EXE** directory).

DOC: contains a very short "readme.txt" file with package version and a link to the web-page of the photometry package and this documentation. A short package history is also given here.

EXAMPLES: contains files for the easy to use examples. The under directory **ALL** contains all the example files in one directory.

1.5 General information about the software

The programs work on Unix and Linux computers, but are also available for Windows 98/2000/XP platforms using the GNU DJGPP compiler for DOS.

The program names have at maximum 8 characters to be easy to use with DOS.

The source code is written in C, but is not recommended for non specialists since the comments are sparse. The code of many of the programs has unfortunately no comments at all of historical reasons.

Input parameters ending with "i" are of integer type and parameters ending with "r" are of real type (i.e. double precision floating point type).

Note, the programs make no check at all that the input parameters have appropriate values.

The software use images in GOP format (*.imf) (i.e. 16 bit integer) for display purposes and a "home-made" 32-bit (*.fff) floating point format for data. For further information on these formats see section 8.

1.6 Disclaimer

For all the programs in this package the following statement is valid:

I make no warranties that this program is (1) free of error, (2) consistent with any standard merchantability, or (3) meeting the requirements of a particular application. This software shall not, partly or as a whole, participate in a process, whose outcome can result in injury to a person or loss of property. It is solely designed for analytical work. Permission to use, copy, and distribute is hereby granted without fee, providing that the header above including this notice appears in all copies.

If you have problems or suggestions to improvements or changes, please send comments to:

E-mail: stefan@spaennare.se

1.7 Other photometry packages

Examples of other photometry and reduction packages are RICHFLD (Tody [1980]), the Aurière & Cordoni ([1981]) package, ROMAFOT (Buonanno et al. [1980], [1983]), Blecha's ([1984]) software, STARMAN (Penny & Dickens [1986]), WOLF (Lupton & Gunn [1986]), DAOPHOT (Stetson [1987]), HAOPHOT (Gilliland & Brown [1988]), PAWSPHOT (Mighell [1989]), The Lund package (Linde [1989]) and CAPELLA (Debray et. al. [1994]). Some of them, for instance ROMAFOT and PAWSPHOT, use the analytical Moffat function for the shape of the PSF (Moffat [1969]). Blecha uses a modified Gaussian profile and STARMAN a combination of a Gaussian and a Lorentzian function. RICHFLD, WOLF, the Lund package and CAPELLA employ an empirical PSF. DAOPHOT and HAOPHOT use a semi empirical PSF constructed from an analytical profile corrected by an array of differences between the instrumental and modeled profiles. Two other packages, INVENTORY (West & Kruszewski [1981], Kruszewski [1989]) and DoPHOT (Mateo & Scheechter [1989]), perform stellar photometry, and in addition, provide information on the nature (stellar or extended) of the object. INVENTORY uses a radial one-dimensional empirical PSF and DoPHOT a modified Gaussian for the PSF.

2 Some useful theory for photometry

2.1 Analytical PSFs

2.1.1 The Moffat function

The stars in astronomical images (taken with ground based telescopes) can often be approximated by a circular symmetric Moffat function:

$$h_r = h_0(1 + cr^2)^{-\beta}$$

where

$$c = \frac{4(2^{\frac{1}{\beta}} - 1)}{\text{FWHM}^2}$$

Normally $1.5 \leq \beta \leq 4.0$. A typical value is 2.5.

The total integrated intensity for a circular symmetric Moffat function with height (h) and ($\beta > 1$) is given by:

$$I = \frac{\pi \text{FWHM}^2 h}{4(2^{\frac{1}{\beta}} - 1)(\beta - 1)}$$

The fraction intensity found inside a radius (r) is given by:

$$\frac{I_r}{I} = 1 - (1 + \alpha^2(2^{\frac{1}{\beta}} - 1))^{1-\beta}$$

where

$$\alpha = \frac{2r}{\text{FWHM}}$$

2.1.2 The Gaussian

The circular symmetric Gaussian is usually not used as a PSF in astronomical photometry, but a comparison with the Moffat function is interesting. For the Gaussian the corresponding equations are given by:

$$h_r = h_0 e^{-cr^2}$$

where

$$c = \frac{4 \ln 2}{\text{FWHM}^2}$$

The total integrated intensity of a Gaussian with height (h) is given by:

$$I = \frac{\pi \text{FWHM}^2 h}{4 \ln 2}$$

The fraction intensity found inside a radius (r) is given by:

$$\frac{I_r}{I} = 1 - 2^{-\alpha^2}(\alpha^2 \ln 2 + 1)$$

where

$$\alpha = \frac{2r}{\text{FWHM}}$$

2.1.3 A comparison between the Moffat function and the Gaussian

The fraction intensity found inside different radii for the Moffat function (different values of β) and the Gaussian are found in the table below. It is obvious that the Gaussian is sharper than the Moffat function also for high values of β .

α	Gaussian	$\beta = 1.5$	$\beta = 2.0$	$\beta = 2.5$	$\beta = 3.0$	$\beta = 3.5$	$\beta = 4.0$
1	0.153426	0.206299	0.292893	0.340246	0.370039	0.390493	0.405396
2	0.764213	0.453609	0.623615	0.709156	0.759633	0.792563	0.815578
4	0.999816	0.689890	0.868894	0.933822	0.962424	0.976775	0.984691
6	1.000000	0.787505	0.937153	0.977379	0.990678	0.995750	0.997902
8	1.000000	0.839031	0.963649	0.989933	0.996784	0.998856	0.999556
10	1.000000	0.870620	0.976427	0.994713	0.998627	0.999602	0.999874

2.2 Theoretical errors caused by noise

It is often interesting to compare measured magnitude errors and theoretical magnitude errors caused by pure noise for stars in astronomical images. If the gain factor is (g), the integrated intensity of the star is (I) ADU, the read out noise (r) ADU and the background level (b) ADU, the error (standard deviation) caused by noise (σ_{gI}) can be calculated from (Lindgren [1996]):

$$\sigma_{gI} = \left(\sum_{ij} \frac{\text{PSF}_{ij}^2}{\text{PSF}_{ij}gI + gb + (gr)^2} \right)^{-\frac{1}{2}}$$

where σ_{gI} is measured in electrons and the error in ADU is given by:

$$\sigma_I = \frac{\sigma_{gI}}{g}$$

The magnitude error is then given by:

$$\Delta\text{mag} = -2.5 \log_{10} \left(1 + \frac{\sigma_I}{I} \right)$$

The PSF is normalized to an integrated intensity of unity and the sum is taken over all pixels (ij) where the PSF is significantly larger than zero. In practice for a large radius around the center of the PSF. If the background and read out noise are zero (i.e. pure Poisson noise) the formula above leads to the well known result $\sigma_{gI} = \sqrt{gI}$.

An ideal magnitude measurement gives these theoretical errors, but it is not possible to come lower because of the noise. For practical measurements the goal is to come as close as possible to this limit. It should be noted that this calculation does not take into account noise caused by stellar crowding in the image.

3 FINDSTARS star finding and photometry program

FINDSTARS is a computer program, written in C, that finds stars and performs photometry in crowded (and uncrowded) stellar fields. The program is especially good when the background is uneven. The program works in a simple way, but despite that, the performance is good both regarding photometry and number of stars found. The output must be sorted with a special program (FSORT) to obtain the final data for the stars. A C-like pseudo code is found in Figure 1.

```
/* FINDSTARS */

{
  initmasks;
  initPSFarrays;

  step=1.0/nsteps;

  for (x=0; x<xsize; x=x+step) {
    for (y=0; y<yssize; y=y+step) {
      getimage(x,y);
      leastsquares(x,y);
      if (int > minint) {
        getsigma(x,y);
        if (sigma < (sigmafacs*int)) {
          print(position(x,y));
          print(intensity);
          print(background);
          print(sigma);
        } /* if */
      } /* for y */
    } /* for x */
  } /* End */
```

Figure 1: C-like pseudo code for FINDSTARS. For description see the text.

3.1 Algorithm

FINDSTARS perform a linear least squares fit, which is called `leastsquares(x,y)` in the pseudo code,

$$c_1 \text{PSF}(x, y) + c_0 = \text{image}(x, y)$$

for all pixels inside the fitting radius r_{fit} around each sub pixel position (x, y) in the image. The equations are equally weighted (no dependencies on r). To make the program fast and to make it possible to use a very small fitting radius there is no possibility to reject bad pixels from the fit. The parameter r_{fit} is fed to the program. The coefficient c_1 is the intensity of the local maximum and c_0 is the background level in the point (x, y) . The PSF is normalized to an integrated intensity of unity.

The sub pixel step size is $\text{step} = 1/\text{nsteps}$ where `nsteps` is an integer number that is fed to the program. One can assume that the position errors are rectangular distributed ($\pm 1/(2 \cdot \text{nsteps})$). Then the standard deviation in the position error can not be smaller than $\sigma = 1/(\sqrt{12} \cdot \text{nsteps})$. In the pseudo code `xsize` and `yssize` are the width and the height of the image. Thus $\text{nsteps}^2 \cdot \text{xsize} \cdot \text{yssize}$ least squares fits are performed when running FINDSTARS. Before the main loop of the program starts, circular masks

for the fitting radius and PSF-arrays for the `nsteps`² sub pixel locations must be set up (i.e. `initmasks` and `initPSFarrays` in the pseudo code). The PSF can either be an empirical image that is fed to the program, or an analytical, circular symmetric, Moffat function. (a special version of FINDSTARS). If the PSF is empirical, it must be carefully centered, and it is then bicubic spline interpolated to the `nsteps`² sub pixel positions necessary. The PSFs are stored in arrays to speed up the program.

The pixel values of the image (inside r_{fit} around the position (x, y)) are obtained with `getimage(x,y)`. After `leastsquares(x,y)` is calculated and if the threshold condition (`int > minint`) is fulfilled the standard deviation (`sigma`) of the residuals is calculated (`getsigma(x,y)`) in the position (x, y) . If then the threshold condition (`sigma < (sigmafac * int)`) is fulfilled, data is written to the output file. Here `int` is the intensity of the local maxima and `minint` and `sigmafac` are parameters that are fed to the program. The output file from FINDSTARS contains the position (x, y) , intensity, background and square sum of the local maximum. This file is then sorted using FSORT (see below). FINDSTARS is a fast program, but it is important to note that large values of `nsteps` and the fitting radius r_{fit} increases the computational time very much. It is also important to note that the temporary binary output files from FINDSTARS can be large, typically 5 Mbyte to 50 Mbyte. These files can be removed after the run with FSORT.

3.2 FSORT

After the data file with data sets from the local maximum has been obtained using FINDSTARS, the values are sorted using FSORT to obtain the final data for the stars. Each data set includes the x - and y -position, intensity, background and square sum of the local maxima. The sorting finds the highest intensity in the data file. Then every data set that is inside a radius r_{rej} around the highest intensity is rejected. This is to avoid multiple detection of the stars. The procedure is repeated for the remaining data sets in decreasing intensity order. To improve the positions the position with the lowest square sum inside another radius r_{step} around the intensity maximum is chosen. Usually r_{rej} is chosen slightly less than the FWHM of the stars in crowded fields. This means that no faint stars are present closer than FWHM to a brighter star in the output file. Experiments have shown that this works well in practice. If `step = 1/nsteps` usually r_{step} is somewhat larger than $\sqrt{\text{step}^2 + \text{step}^2}$. Thus the improved position is found close to the intensity maximum. The output file from FSORT contains the x - and y -position, intensity, magnitude, background and square sum of the stars in the image. If the file contains too many faint artifacts, it can be truncated easily, because the stars are sorted in decreasing intensity order. Because the files from FINDSTARS can be large FSORT requires much memory in the computer.

3.3 Timing

As mentioned above FINDSTARS is a fast computer program, and the time complexity is given by

$$\text{time} = c_1 \cdot \text{xsize} \cdot \text{ysize} \cdot r_{\text{fit}}^2 \cdot \text{nsteps}^2 + c_2 \cdot \text{nmaxima}$$

where c_1 and c_2 are constants, `xsize` and `ysize` is the size of the image, r_{fit} the fitting radius, `nsteps` the number of steps per pixel (as described above) and `nmaxima` is the number of local maxima found. It is obvious that the computation time increases very much if the fitting radius and `nsteps` are large. However, usually `nsteps` can be chosen smaller if the FWHM of the stars (and the fitting radius) is large, which partly eliminates this effect. The first term in the expression above is by far the most time consuming, why the time for a run with FINDSTARS only depends very little on the number of stars in the image. As mentioned above the program produces large temporary output files. This means that FSORT requires much computer memory (1.0 Mbyte per Mbyte of the file).

The time complexity for FSORT is about n^2 where n is the number of local maxima found in the output file from FINDSTARS.

3.4 How to use FINDSTARS and FSORT

3.4.1 FINDSTARS

The FINDSTARS and FSORT photometry programs are very easy to use. FINDSTARS is available in two versions, one with an analytical Moffat PSF and one with an empirical (image) PSF. The programs print the size of the PSF (in pixels), a matrix containing the number of pixels fitted in each sub pixel point and the CPU-time for the run. If you type the names of the programs the following appears:

```
>Usage: findsmbf image.fff fwhmr betar rfitr minintr sigmafacr  
        ampr nstepsi outfile.bin
```

```
>Usage: findsbf image.fff psf.fff rfitr minintr sigmafacr  
        ampr nstepsi outfile.bin
```

Parameter description:

image.fff: astronomical image in (*.fff) floating point format.

psf.fff: empirical PSF in (*.fff) floating point format. The PSF must be well centered and the background subtracted.

fwhm: FWHM of the Moffat PSF (i.e. the stars in the image).

beta: Moffat β value of the PSF. Normally $1.5 \leq \beta \leq 4.0$. A typical value is 2.5.

rfit: fitting radius in pixels. Normally $0.5 \cdot \text{FWHM} \leq \text{rfit} \leq 1.5 \cdot \text{FWHM}$

minint: minimal integrated intensity (in ADU) of the stars to be accepted.

sigmafac: fraction of the intensity that should be larger than the standard deviation of the residuals for a local maxima to be accepted. Normally $0.005 \leq \text{ssfac} \leq 0.040$. The sharper PSF (i.e. smaller FWHM) the higher value.

amp: output amplification factor (usually 1.0).

nsteps: number of sub pixel steps per pixel. Some hints are given in the table below:

FWHM	nsteps
≤ 2.0	4 or 8
≈ 4.0	4
≈ 8.0	2
≥ 8.0	1 or 2

outfile.bin: the binary output file to be processed with FSORT.

3.4.2 FSORT

FSORT sorts the local maxima found by FINDSTARS into final stars. The program prints the CPU-time of the run. If you type the name of the program the following appears:

```
>Usage: fsortb findsfile.bin outfile.asc rrejrr rstepr
```

Parameter description:

`findsfile.bin`: binary output file from FINDSTARS.

`outfile.asc`: resulting output file containing the final x- and y-position, intensity, magnitude, background level and square sum of the stars.

`rrej`: star rejection radius. No other stars are accepted inside the radius around an already accepted star. Normally `rrej` is about the same as the FWHM of the stars.

`rstep`: position improvement radius. Inside this radius around a found star the best position of the star is searched (i.e. the position with the smallest square sum). If `step = 1/nsteps` then `rstep` is somewhat larger than $\sqrt{\text{step}^2 + \text{step}^2}$. Some useful values are found in the table below:

nsteps	step	rstep
1	1.000	1.50
2	0.500	0.75
4	0.250	0.40
8	0.125	0.20

4 POLYFIT photometry program

The computer program POLYFIT, written in C, was developed to give accurate photometry of well sampled stars also on uneven luminous backgrounds consisting of galaxies or nebulae. The program can also be used to remove stars from galaxy images, if the galaxy itself is the relevant object.

4.1 Algorithm

In POLYFIT the background of the star is modeled with a two-dimensional polynomial of a degree that can be set to 0, 1, 2, 3, 4 or 5. A polynomial of degree 0 corresponds to a flat plane while degree 1 corresponds to a tilted plane. Higher order polynomials correspond to more complicated shapes allowing higher spatial frequencies. There is no special theoretical argument for polynomials as background estimators (other functions can be used as well) but they are easy to implement and experiments show that they give quite accurate results if the parameters are properly tuned.

The PSF and the background polynomial are least squares fitted simultaneously to the star and the background in the image. A circular mask is created for all n_{fit} pixels with positions (x, y) inside the fitting radius r_{fit} counted from the center of the star (x_0, y_0) . The linear equation system then has n_{fit} rows of the form:

$$\sum_{\substack{i,j \\ 0 \leq i+j \leq d}} c_{ij} x^i y^j + c \cdot \text{PSF}(x, y) = I(x + x_0, y + y_0)$$

$$\text{for } x^2 + y^2 \leq r_{\text{fit}}^2$$

and for different degrees (d) of the polynomial. This equation system is solved using a standard least squares fit to obtain the coefficients c_{ij} . c is the intensity of the star when the PSF is normalized to an integrated intensity of unity. $I(x + x_0, y + y_0)$ is the image in the position $(x + x_0, y + y_0)$. Obviously at least as many equations (i.e. points in the mask) as there are coefficients c_{ij} in the equation (i.e. $\pi r_{\text{fit}}^2 \approx n_{\text{fit}} \geq n_{\text{coeff}}$) are needed. It is possible to reject pixels from the fit, pixels that deviates more than a pre-defined number of standard deviations (σ) from the mean. In the case of a two-dimensional polynomial of degree 5, 22 coefficients have to be determined, 21 for the polynomial and 1 for the PSF. To have enough equations this requires a fitting radius $r_{\text{fit}} > \sqrt{22/\pi} \approx 2.6$ pixels. In practice the radius

must be much larger for a proper fit. For a polynomial of degree 0 (i.e. a constant) the radius can be as small as 1 pixel. In the least squares fit the n_{fit} equations are equally weighted, but other weighting functions such as $1/(r + 1)$ or $1/(r^2 + 1)$ have been tested. Here r is the radius $\leq r_{\text{fit}}$ from the center of the stellar image. However, brief experiments show that equal weighting works best in most cases.

To be able to obtain a sub-pixel position for the star the PSF is bicubically spline-interpolated to fractional pixel positions. This interpolation works well for well sampled stars. To obtain the best value of the stellar magnitude and the position of the star, a simple "grid-search" is performed to find the local residual minimum. This grid-search means that the least squares fit is performed in every point in a grid ($n \cdot n$ points, normally $n = 5$) of a pre-defined size around the initial position of the star. This initial position must be fed to the program and can be obtained for example by using FINDSTARS (described in section 5.3). The improved position of the star is then the grid-point where the standard deviation of the residuals has the lowest value. The position can be further improved by decreasing the grid size in more than one step. This grid-search works well in practice but requires substantial computer time. POLYFIT has some capabilities to make a simultaneous fit of many stars (in a group) at the same time, which is important when the target star is affected by bright neighbors. POLYFIT is not optimized for speed, as it is usually used only for a small number of stars with large fitting radius. The output from the program contains the (improved) x and y position, intensity, magnitude, background level, square sum and computation time for each star in the input file.

4.2 How to use POLYFIT

POLYFIT was developed to give accurate photometry on well sampled stars also on uneven backgrounds consisting of galaxies and nebulae. FWHM of the PSF (stars) should typically be ≥ 4.0 pixels. The empirical PSF is bicubic spline interpolated to fractional pixel positions. Initial coordinates for the stars must be given to the program. POLYFIT has several parameters. If you type the name of the program the following appears:

```
>Usage: polyfitf image.fff psf.fff maxnstarsi polydegi xystep xyfacr xyepsr
        gridsi rfitr sigmafacr coordepsr ampr groupfile outfile
```

Parameter description:

image.fff: astronomical image in (*.fff) floating point format.

psf.fff: empirical PSF in (*.fff) floating point format. The PSF must be well centered and the background subtracted.

maxnstars: maximum number of stars in a group. Usually equal to 1. This value is equal to the maximum number of stars in a group if you want to fit several close stars simultaneously.

polydeg: degree of the background polynomial. $0 \leq \text{polydeg} \leq 5$. Normally a polynomial degree of 2 or 3 gives the best result.

xystep: center of the stars (i.e. the best fit) is searched in a square area with a side length of $2 \cdot \text{xystep}$ pixels (i.e. $\pm \text{xystep}$ pixels around the initial position in the x- and y-direction). If the final position must be better than can be achieved with only one iteration, **xystep** can be further decreased by setting the values of **xyfac** and **xyeps** properly. Another way is to make the value of **grids** larger (see below).

xyfac: factor **xystep** is decreased for each search iteration. A typical value is 0.5.

xyeps: smallest size of **xystep** to be used. Normally only one iteration is necessary why $\text{xyfac} \cdot \text{xystep} < \text{xyeps} < \text{xystep}$ can be chosen.

grids: number of integer grid points (in the x- and y-direction) in which the square area with a side length of $2 \cdot \text{xystep}$ pixels is divided. If for example **grids** = 5, $5 \cdot 5 = 25$ points will be searched for the best fit. The initial position is in the central grid point.

rfit: fitting radius in pixels. Typically **rfit** = $1.5 \cdot \text{FWHM}$ of the PSF (stars).

sigmafacs: number of times σ (i.e. the standard deviation) of the fit bad pixels are rejected. A typical value is 3.0.

coordeps: smallest distance in pixels between two stars in a group that is accepted. If the groups only contain 1 star each (i.e. **maxnstars** = 1) this value is a dummy value. A typical value is 0.5 (pixels).

amp: output amplification factor (usually 1.0).

groupfile: file containing the initial coordinates for the stars to be measured formatted in a special way. A small extra program **c2g** converts a list of x-, y-coordinates and dummy intensity values to a group list. If several stars should be grouped and fitted simultaneously a more advanced program **group** can be used. The initial positions of the stars can be obtained with other star finding and photometry programs (for example FINDSTARS).

outfile: output text file. The file contains the x-, y-position, integrated intensity, magnitude, background level, square sum and computation time for each star.

5 Programs to make analytical PSFs

The photometry programs described above require either an empirical (image) PSF or parameters for an analytical (Moffat) PSF. If you use an empirical PSF the image of a bright star often works well. Then the PSF (star) must be well centered and the background subtracted. It is also possible (and sometimes better) to fit a Moffat function to an empirical image of a star. Some programs to perform this are described here. Some theory for the Moffat function and the Gaussian is described in section 2.

5.1 Programs to make analytical image PSFs

5.1.1 setmofff and setgausf

To make analytical circular symmetric (image) PSFs programs **setmofff** and **setgausf** can be used:

```
>Usage: setmofff image.fff sizei fwhmr betar
```

```
>Usage: setgausf image.fff sizei fwhmr
```

Parameter description:

image.fff: PSF image in (*.fff) floating point format. The top intensity is 32767.0.

size: size of the square PSF image. Normally $> 10 \cdot \text{FWHM}$ of the PSF. Should be an odd number for a PSF centered in the central pixel.

fwhm: FWHM of the PSF.

beta: Moffat β value.

5.2 Programs to fit a Moffat function to the image of a star

5.2.1 moffit and moffit2

The programs `moffitf` and `moffitf2` fits an analytical elliptical symmetric Moffat function to the image of a (bright) star. For `moffitf` the β value is given to the program. For `moffitf2` the best β value is fitted by the program. The programs can also be used to check if a PSF is well centered. Several parameters for the fit is printed when the programs are run. A image PSF is also created from the fitted parameters.

```
>Usage: moffitf inimage.fff outimage.fff bgr limr betar
```

```
>Usage: moffitf2 inimage.fff outimage.fff bgr limr betaminr betamaxr betaepsr betastepi
```

Parameter description:

`inimage.fff`: image (star) to be fitted.

`outimage.fff`: resulting PSF image calculated from the fitted parameters.

`bg`: estimated background level of the star.

`lim`: height (pixel value) of the star over which the fit is calculated. Normally 10 % to 20 % of the highest pixel value of the star to avoid problems with the wings of the star.

`beta`: Moffat β value.

`betamin`: minimum start value for $\beta > 1.0$.

`betamax`: maximum start value for β (normally < 10.0).

`betaeps`: required accuracy for β (for example 0.0001).

`betastep`: integer number of points to be fitted between `betamin` and `betamax` (for example 10).

6 Programs to make simulated stellar fields

6.1 imstarsf and imstarsmf

Here are described two programs to make simulated stellar fields. The first (`imstarsf`) use an empirical image PSF, which is bicubic spline interpolated to fractional pixel positions. Thus any shape of the PSF can be used, although large interpolation errors can occur if the PSF is very sharp (i.e. $\text{FWHM} \leq 2.0$ pixels). The other program `imstarsmf` use an analytical Moffat PSF (thus avoiding interpolation) and can be used also for sharp PSFs. If you have many stars in the image the computation time can be quite long for these programs.

```
>Usage: imstarsf image.fff xsizei ysizei psf.fff ampr coord(x y int)
```

```
>Usage: imstarsmf image.fff xsizei ysizei fwhmr betar  
ampr coord(x y int)
```

Parameter description:

`image.fff`: the created image in (*.fff) floating point format.

`xsize`: x-size (width) of the image in pixels.

`ysize`: y-size (height) of the image in pixels.

`psf.fff`: image PSF in (*.fff) floating point format. The PSF must be well centered and the background subtracted.

`fwhm`: FWHM of the analytical Moffat PSF.

`beta`: β value of the analytical Moffat PSF.

`amp`: arbitrary amplification factor (normally 1.0).

`coord`: input file containing a list of the x- and y-position and the integrated intensity of the stars.

7 Programs to add noise to images

7.1 Poisson noise

7.1.1 `poissonf`

This program adds Poisson noise to an image in (*.fff) floating point format. The algorithm for random numbers is from Numerical Recipes in C, Second Edition.

```
>Usage: poissonf inimage.fff outimage.fff seedi gainr
```

Parameter description:

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`seed`: integer random seed.

`gain`: CCD gain factor in electrons/ADU.

7.2 Gaussian noise

7.2.1 `normalf`

This program adds Gaussian noise to an image in (*.fff) floating point format. The algorithm for random numbers is from Numerical Recipes in C, Second Edition.

```
>Usage: normalf inimage.fff outimage.fff seedi sigmaer gainr
```

Parameter description:

`inimage.fff`: input image in (*.fff) floating point format.

outimage.fff: output image in (*.fff) floating point format.

seed: integer random seed.

sigmae: standard deviation of the noise in electrons.

gain: CCD gain factor in electrons/ADU.

8 Astronomical image formats and image conversion programs

Unfortunately there are many different image formats available in the astronomical community. One of the most popular the FITS (Flexible Image Transportation System) format (*.fit or *.mt). Another common format is the MIDAS (*.bdf) format. These formats are available both as short integer, integer and floating point format and includes a header describing the image. At Lund Observatory also a signed short integer GOP-format (*.imf) is used. The GOP-format can easily be converted to an encapsulated Postscript file (*.ps or *.eps). For display purposes also the 24-bit RGB Targa (*.tga) format is used.

The programs described in these manual use the GOP format and a "homemade" floating point format (*.fff). Programs are also available to convert these formats to FITS, MIDAS, Targa and Postscript.

Unfortunately binary stored numbers are bit-swapped between Linux and other Unix versions. Therefore some of these programs are available in two versions (ending with l for Linux and u for Unix).

The program `imf2eps` to convert GOP-images to encapsulated postscript files was kindly included in this package with permission from:

Bo Nilsson, Lund Observatory

E-mail: bo@astro.lu.se

8.1 The (*.fff) floating point format

The *.fff format is a simple binary format and consists of two 32-bit integers defining the xsize and ysize of the image. After that follows the 32-bit floating point image data (with the x-variable changing most rapidly). The origin is in the upper left corner (0,0). Read and write routines for this format are found in the source code.

8.2 The GOP format (*.imf)

The GOP format (*.imf) is mostly used to prepare images for display purposes. This is a 16-bit signed short integer binary format and has a 512 byte header. The origin is in the upper left corner (0,0).

For Unix and Linux must an environmental variable (IMGPATH) be set to define where the GOP images are stored. Three examples are given below:

```
>setenv IMGPATH . /* Dot means current directory. */
>setenv IMGPATH `pwd` /* Print working directory. */
>setenv IMGPATH /home/stefans/IMAGES
```

For Windows 98/2000/XP the GOP image path must be set in the text file "imgpath.txt". This file must be present in the same directory as the program executables.

8.3 Image conversion programs

The following image conversion programs are available (U means Unix and L means Linux or Windows 98/2000/XP):

Program	Conversion	Type	OS
<code>bdf2fff</code>	MIDAS (*.bdf) to (*.fff)	32-bit fp	U and L
<code>bdf2imf</code>	MIDAS (*.bdf) to GOP (*.imf)	16-bit int	U and L
<code>fff2fit1</code>	(*.fff) to FITS (*.fit)	32-bit fp	L
<code>fff2fitu</code>	(*.fff) to FITS (*.fit)	32-bit fp	U
<code>fff2imf</code>	(*.fff) to GOP (*.imf)	32-bit fp to 16-bit int	U and L
<code>fff2tgal</code>	(*.fff) to Targa (*.tga)	32-bit fp to 3*8-bit RGB	L
<code>fff2tgau</code>	(*.fff) to Targa (*.tga)	32-bit fp to 3*8-bit RGB	U
<code>ff12ffu</code>	(*.fff) to (*.fff)	32-bit fp	L to U or U to L
<code>fits2fff1</code>	FITS (*.fit) to (*.fff)	32-bit fp	L
<code>fits2fffu</code>	FITS (*.fit) to (*.fff)	32-bit fp	U
<code>imf2eps</code>	GOP (*.imf) to PS (*.ps)	8-bit int to PS	U and L
<code>imf2fff</code>	GOP (*.imf) to (*.fff)	16-bit int to 32-bit fp	U and L
<code>imf2fit</code>	GOP (*.imf) to FITS (*.fit)	16-bit int	U and L
<code>imf2tgal</code>	GOP (*.imf) to Targa (*.tga)	16-bit int to 3*8-bit RGB	L
<code>imf2tgau</code>	GOP (*.imf) to Targa (*.tga)	16-bit int to 3*8-bit RGB	U
<code>imf2tgl2</code>	GOP (*.imf) to Targa (*.tga)	3*8-bit int to 3*8-bit RGB	L
<code>imf2tgu2</code>	GOP (*.imf) to Targa (*.tga)	3*8-bit int to 3*8-bit RGB	U
<code>copy1</code>	GOP (*.imf) to GOP (*.imf)	16-bit int to 16-bit int	U and L
<code>copy2</code>	GOP (*.imf) to GOP (*.imf)	16-bit int to 8-bit int	U and L
<code>prn2fff</code>	(*.prn) to (*.fff)	fp text to 32-bit fp	U and L

The image conversion programs work in the same way:

```
>program inimage.ext outimage.ext
```

For the GOP-images no extension should be given and the image path must also be set. The programs `fff2tgal` and `fff2tgau` are used for display purposes and also have an amplification factor. The size of the image must be given to the program `ff12ffu`. The program `imf2eps` works in a special way. See the examples in section 12. Unfortunately this program does not work under Windows 98/2000/XP.

9 Some image processing programs

Some of the programs are available both for the GOP format (*.imf) and the (*.fff) floating point format. Other programs are only available for the GOP format (*.imf) for display purposes.

9.1 Programs to add and multiply images

Here are described some programs to add and multiply images in the GOP 16 bit integer format (*.imf) and the (*.fff) floating point format. The programs for the (*.fff) format ends with an "f". The calculated pixel values (p) for the GOP format are truncated in the range ($0 \leq p \leq 32767$). The GOP format should only be used for display purposes. The usages of these programs are quite obvious:

9.1.1 Addition of a constant to an image

9.1.2 `imadd` and `imaddf`

>Usage: `imadd inimage outimage termi`

>Usage: `imaddf inimage.fff outimage.fff termr`

9.1.3 Multiplication of an image with a constant

9.1.4 `immul` and `immulf`

>Usage: `immul inimage outimage ampr`

>Usage: `immulf inimage.fff outimage.fff ampr`

9.1.5 Addition and subtraction of two images

9.1.6 `imimadd` and `imimaddf`

>Usage: `imimadd inimage1 inimage2 outimage amp1r amp2r`

>Usage: `imimaddf inimage1.fff inimage2.fff outimage.fff amp1r amp2r`

The parameters `amp1` and `amp2` are the relative weights for the images (i.e. 1.0, 1.0 for addition and 1.0, -1.0 for subtraction).

9.2 Programs to resample images

Here are described some programs to resample images. Programs are available for images in GOP format (*.imf) and (*.fff) floating point format.

9.2.1 Programs to sample up images

These programs use simple pixel replication to sample up the image.

9.2.2 `rebinu` and `rebinuf`

>Usage: `rebinu inimage outimage binfactori`

>Usage: `rebinuf inimage.fff outimage.fff binfactori`

Parameter description:

`inimage`: input image in GOP format.

`outimage`: output image in GOP format.

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`binfactor`: resampling factor. The new image will be `binfactor` times larger than the original image.

9.2.3 Programs to sample up images

These programs use bicubic spline interpolation to sample up the image.

9.2.4 `irebinu` and `irebinuf`

```
>Usage: irebinu inimage outimage binfactori
```

```
>Usage: irebinuf inimage.fff outimage.fff binfactori
```

Parameter description:

`inimage`: input image in GOP format.

`outimage`: output image in GOP format.

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`binfactor`: resampling factor. The new image will be `binfactor` times larger than the original image.

9.2.5 Programs to sample down images

A resulting pixel value is the simple average of `binfactor`² input pixels. It is important to note the the resolution of the image is decreased with these programs. It is also important to always check that the image size is possible to divide evenly with `binfactor`. The programs make no check for this.

9.2.6 `rebind` and `rebindf`

```
>Usage: rebind inimage outimage binfactori
```

```
>Usage: rebindf inimage.fff outimage.fff binfactori
```

Parameter description:

`inimage`: input image in GOP format.

`outimage`: output image in GOP format.

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`binfactor`: resampling factor. The new image will be `binfactor` times smaller than the original image. It is important to always check that the image size is possible to divide evenly with `binfactor`. The programs make no check for this.

9.3 Programs to cut and merge images

This section describes some programs to cut and merge images in GOP format (*.imf) and (*.fff) floating point format. Images in GOP format are usually used for display purposes.

9.3.1 Programs to cut images

9.3.2 imcut and imcutf

```
>Usage: imcut inimage outimage x1i y1i x2i y2i
```

```
>Usage: imcutf inimage.fff outimage.fff x1i y1i x2i y2i
```

Parameter description:

`inimage`: input image in GOP format.

`outimage`: output image in GOP format.

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`x1 y1 x2 y2`: the corners of the area to be cut. These values must fit into the size of the input image. There is no check for this.

9.3.3 Programs to merge images

These programs are used to merge (combine) several images of equal size to a larger image. The program for GOP images also have the possibility to have white edges between the images for display purposes.

9.3.4 imcomb and imcombf

```
>Usage: imcomb nrowi ncoli
        im_11 im_12 ... im_1nc
        im_21 im_22 ... im_2nc
        ... ..
        im_nr1 im_nr2 ... im_nrnc
        outimage edgei
```

```
>Usage: imcombf nrowi ncoli
        im_11.fff im_12.fff ... im_1nc.fff
        im_21.fff im_22.fff ... im_2nc.fff
        ... ..
        im_nr1.fff im_nr2.fff ... im_nrnc.fff
        outimage.fff
```

Parameter description:

`nrow`: number of image rows ($nrow \leq 10$).

`ncol`: number of image columns ($ncol \leq 10$).

`im_11 ... im_nrnc`: the `nrow · ncol` input images in GOP format.

`im_11.fff ... im_nrnc.fff`: the `nrow · ncol` input images in (*.fff) floating point format.

`outimage`: merged output image in GOP format.

`outimage.fff`: merged output image in (*.fff) floating point format.

`edge`: the width of the white edges between the input images (for GOP images only).

9.4 Some additional image processing programs

Some of the programs are available both for the GOP format (*.imf) and the (*.fff) floating point format. Other programs are only available for the GOP format (*.imf) for display purposes.

9.4.1 mmmfw and mmmfwf

These useful programs print the maximum value, minimum value, average integrated intensity per pixel, FWHM of the whole image and the x- and y-size of the image. The programs can be used to make a simple check of the FWHM of a PSF image.

```
>Usage: mmmfw image
```

```
>Usage: mmmfwf image.fff
```

9.4.2 imsig and imsigf

These programs print the x-size, y-size, number of pixels, the mean and standard deviation and a robust estimation of the mean and the standard deviation. The program can be used to estimate the noise level in the background of an image (without stars). The robust estimator of the mean and standard deviation is based on the median value and the outer quartiles of the data set. This method sometimes gives a better estimate of the average and the standard deviation if the data set has many outliers and a non normal distribution.

```
>imsig image
```

```
>imsigf image.fff
```

9.4.3 medianh and medianhf

These are very simple program to make median filtering of images. The programs are intended to remove Cosmic Rays, but better programs are available. Here follows however a description.

```
>Usage: medianh inimage outimage orderi limiti
```

```
>Usage: medianhf inimage.fff outimage.fff orderi limitr
```

Parameter description:

`inimage`: input image in GOP format.

outimage: output image in GOP format.

inimage.fff: input image in (*.fff) floating point format.

outimage.fff: output image in (*.fff) floating point format.

order: the (odd) size of the median filter in pixels.

limit: the median filter operates on pixels which differs more than **limit** from the median value. Set this value to 0 for a pure median filtering and a quite high value for removing Cosmic Ray events.

9.4.4 imrot and imrotf

These programs rotate an image 90 degrees anti clockwise.

```
>imrot inimage outimage
```

```
>imrotf inimage.fff outimage.fff
```

9.4.5 imshift and imshiftf

These programs shift the image a fractional pixel position in the x- and y-direction using bicubic spline interpolation. The shift is to the left in the x-direction and upwards in the y-direction ($0.0 \leq dx \leq 1.0$ and $0.0 \leq dy \leq 1.0$).

```
>Usage: imshift inimage outimage dxr dyr
```

```
>Usage: imshiftf inimage.fff outimage.fff dxr dyr
```

9.4.6 slice and slicef

These programs make a slice (cut) through the image and print the result to standard output. The programs are useful to plot slices from images. Always check that x1, y1, x2 and y2 fits into the size of the image. The programs make no check for this.

```
>Usage: slice image x1r y1r x2r y2r
```

```
>Usage: slicef image.fff x1r y1r x2r y2r
```

9.4.7 plots

Plots make a 3-dimensional image of an image in GOP format (*.imf). The program is especially useful to display image of PSFs.

```
>Usage: plots inimage outimage rotangelr vertangelr ampr powerr
```

Parameter description:

inimage: input image in GOP format.

outimage: output image in GOP format.

rotangel: rotation angle of the image in degrees (usually 0.0).

vertangel: vertical angle of the image in degrees (usually about 30.0).

amp: amplification factor ($0.0 \leq \text{amp} \leq 1.0$).

power: power the image is raised to (usually 1.0, 0.5 for square root and 2.0 for square for example).

9.4.8 imlog

This program makes a logarithmic image of an image in GOP format (*.imf). The program is only for display purposes. Usually ($0.5 \leq \text{amp} \leq 10.0$).

>Usage: `imlog inimage outimage ampr`

9.4.9 imdot, imdot2 and imdot3

These programs are used to mark stars with either dots (`imdot`) or circles (`imdot2` and `imdot3`) in an image in GOP format (*.imf). The programs are only for display purposes. Data for the stars must be available in a file containing the x- and y-position and intensity (or magnitude) of the stars. The program `imdot3` automatically gives colors to the circles that are easily seen against the background.

>Usage: `imdot inimage outimage file(x y int) orderi coli`

>Usage: `imdot2 inimage outimage file(x y int) rr rwr coli`

>Usage: `imdot3 inimage outimage file(x y int) rr rwr`

Parameter description:

inimage: input image in GOP format.

outimage: output image in GOP format.

file: data file for the stars containing the x- and y-positions and intensities (or magnitudes).

order: size of the dot in pixels.

col: color of the dots/circles ($0 \leq \text{col} \leq 32767$).

r: the radius of the circles in pixels.

rw: the with of the circles in pixels (usually 0.25 or 0.50).

9.5 A simple aperture photometry program

This program performs simple aperture photometry on stellar images in (*.fff) floating point format. The program should only be used in sparse stellar fields, when the stars do not overlap at all. The program prints the star number, x- and y-position, integrated intensity, magnitude and background level of the stars to standard output. The positions of the stars must be found by another program (for example FINDSTARS).

9.5.1 aperf

```
>Usage: aperf image.fff rbgmaxr rbgminr raperr file(x y) outfile
```

Parameter description:

image.fff: image in (*.fff) floating point format.

rbgmax: maximum radius for estimation of the background level.

rbgmin: minimum radius for estimation of the background level.

raperr: radius for the aperture to be measured. Normally ($\text{raperr} < \text{rbgmin} < \text{rbgmax}$).

file: input file containing the x- and y-positions of the stars to be measured.

outfile: output text file. The file contains the x-,y-position, integrated intensity, magnitude and background level of the stars.

10 Some auxiliary programs

Here are described some auxiliary programs for photometry and some other non image processing programs.

10.1 Some auxiliary programs for photometry

10.1.1 sscomp and sscomp3

These programs compare two or three files containing the x- and y position and the intensities of measured stars from an image. The programs are used to compare measured data with original data from simulated stellar images. The program **sscomp3** is usually used to find the common set of measured stars in two different measurements (for example with two different photometry programs) that compares to the original data. The star in the original data that best compares to a star in the measured data (both regarding position and intensity) is found for each star in the measured data. The positions and intensities for the original and measured stars are then printed to the outfile (together with the position errors and magnitude errors for **sscomp** only). The files are first sorted in decreasing intensity order to give bright stars highest priority. The input files should be scaled to the same intensity range if necessary.

```
>Usage: sscomp file1(x y int) file2(x y int) rmaxr intminr outfile
```

```
>Usage: sscomp3 file1(x y int) file2(x y int) file3(x y int) rmaxr intminr outfile
```

Parameter description:

file1: file containing the original data (x- and y-position and intensity).

file2: file containing the measured data (x- and y-position and intensity).

file3: file containing the measured data (x- and y-position and intensity).

rmax: maximum difference (radius) in position that is accepted to a correct identification of a star.

intmin: minimum intensity for stars to be accepted (> 1.0).

outfile: output text file containing x- and y-positions and intensities for stars in the original data and measured stars that are accepted during the comparison.

10.1.2 setpos

This program makes a random position file for stars from a luminosity function. The program is used to make simulated stellar images. The program prints the x- and y-position and intensity for each star to standard output. The algorithm for random numbers is from Numerical Recipes in C, Second Edition.

```
>Usage: setpos seedi xsizei ysizei xyinti(0/1) edgei rminr
        lumfunc(n int) outfile
```

Parameter description:

seed: integer random seed.

xsize: maximum x-position of a star in pixels.

ysize: maximum y-position of a star in pixels.

xyint: if this flag is set to 1 the positions will have integer values otherwise floating values.

edge: width of an edge without stars.

rmin: closest distance in pixels between two nearby stars.

lumfunc: luminosity function, i.e. a file containing the number of stars for each intensity.

outfile: output text file.

10.1.3 setposlf

This program makes a random position file for stars from a built in luminosity function. The luminosity function is based Hipparcos data (Holmberg [1998]) from stars in the Solar neighborhood. The program is used to make simulated stellar images. The program prints the x- and y-position and intensity for each star to standard output. The algorithm for random numbers is from Numerical Recipes in C, Second Edition.

```
>Usage: setposlf seedi xsizei ysizei xyinti(0/1) edgei rminr
        nstarsi intmaxr intminr outfile
```

Parameter description:

seed: integer random seed.

xsize: maximum x-position of a star in pixels.

ysize: maximum y-position of a star in pixels.

xyint: if this flag is set to 1 the positions will have integer values otherwise floating values.

edge: width of an edge without stars.

rmin: closest distance in pixels between two nearby stars.

nstars: the number of stars with (intensity > **intmin**). There will be many more faint stars in the output file contributing to the background noise.

intmax: maximum possible intensity for a star in the field.

intmin: minimum intensity for the **nstars** stars.

outfile: output text file.

10.1.4 cadd

This is a program to rescale the positions and intensities of a file containing the x- and y-positions and intensity of the stars. The result is printed to standard output.

$$x_{\text{new}} = wx \cdot x_{\text{old}} + dx$$

$$y_{\text{new}} = wy \cdot y_{\text{old}} + dy$$

$$int_{\text{new}} = wint \cdot int_{\text{old}}$$

>Usage: `cadd file(x y int) wxr wyr dxr dyr wintr`

10.1.5 ctransf and ctransf2

These programs make a transformation between coordinates in one image to another. The transformation allow rotation and translation. The first program **ctransf** makes a least squares fit of a subset (> 3) equal identified stars from the two images. The second program **ctransf2** perform the actual transformation from coordinates in image 1 to coordinates in image 2 using the coefficients fitted with the first program.

$$x_2 = a_2x_1 + a_1y_1 + a_0$$

$$y_2 = b_2x_1 + b_1y_1 + b_0$$

where a_i and b_i are constants to be fitted.

>Usage: ctransf file1(x y int) file2(x y int) coefffile

>Usage: ctransf2 file1(x y int) coefffile outfile

Parameter description:

file1: input file containing the x-, y-position and intensity of the stars in the first image.

file2: input file containing the x-, y-position and intensity of the stars in the second image.

coefffile: file containing the fitted coefficients.

outfile: output file containing the transformed coordinates.

10.1.6 cparam

This program calculates the number of items, the minimum, maximum, mean and standard deviation of the distance between stars above a minimum intensity in a file containing the x-, y-position and intensity. The program is used to estimate the star density of an image from a measured or simulated data file. A robust estimator of the mean and standard deviation is also calculated. The robust estimator of the mean and standard deviation is based on the median value and the outer quartiles of the data set. This method sometimes gives a better estimate of the average and the standard deviation if the data set has many outliers and a non normal distribution.

>cparam file(x y int) intminr

10.1.7 chist

This program makes a simple histogram of star intensities from a file containing the x-, y-position and intensity of the stars. The number of stars to each intensity decade is presented together with the cumulative number of stars.

>chist file(x y int)

10.1.8 int2mag and mag2int

These programs convert a file containing positions and intensities for stars to positions and magnitudes and vice versa.

$$\text{mag} = -2.5 \log_{10}(\text{int}) + \text{mag0}$$

$$\text{int} = 10^{-0.4(\text{mag}-\text{mag0})}$$

>Usage: int2mag infile(x y int) mag0r

>Usage: mag2int infile(x y mag) mag0r

10.1.9 movavi and movavm

These programs calculate the moving average errors of stellar intensity or magnitude data. The magnitude, magnitude median and errors $\pm 34\%$ around the median are given in the output file.

```
>Usage: movavi infile(int1 int2) outfile mi mstepi
```

```
>Usage: movavm infile(mag1 mag2) outfile mi mstepi
```

Parameter description:

infile: infile with two columns with stellar intensities or magnitudes.

outfile: outfile containing magnitude, median and magnitude errors.

m: number of data points in the moving average window.

mstep: number of data point shift between the moving average windows.

10.1.10 magnoise

This program calculates theoretical errors caused by noise for stars with assumed Moffat shape in an image (see section 2.2). The "pseudo" magnitude, magnitude error, intensity and intensity error is printed to the output file for each magnitude step.

An ideal magnitude measurement gives these theoretical errors, but it is not possible to come lower because of the noise. For practical measurements the goal is to come as close as possible to this limit. It should be noted that this calculation does not take into account noise caused by stellar crowding in the image.

```
>Usage: magnoise m1r m2r mstepr fwhmr betar  
gainfr bgr renr outfile
```

Parameter description:

m1: lower "pseudo" magnitude (i.e. $-2.5 \log_{10}(\text{intensity})$).

m2: upper "pseudo" magnitude (i.e. $-2.5 \log_{10}(\text{intensity})$).

mstep: magnitude step between **m1** and **m2**.

fwhm: FWHM of the Moffat function.

beta: β value of the Moffat function.

gain: CCD gain factor (electrons/ADU).

bg: the (constant) background level of the stars in electrons.

ren: CCD readout noise in electrons.

outfile: output text file.

10.1.11 Programs to make groups for POLYFIT

These programs are used to group stars for the photometry program POLYFIT (see section 5). The program `c2g` can be used if you have only one star in each group. The program `group` can be used if you want to fit and measure several nearby stars simultaneously. This is an early experimental attempt to perform this and it can probably be improved. Initial positions (and intensities) for POLYFIT can for example be obtained with the FINDSTARS program (see section 3).

10.1.12 `c2g` and `group`

>Usage: `c2g file(x y int)`

>Usage: `group file(x y int) maxgroupi rmaxr`

Parameter description:

`file`: input file containing the x- and y-position and intensities of the stars.

`maxgroup`: maximum number of stars in a group.

`rmax`: maximum connection radius for nearby stars in a group.

10.2 Other auxiliary programs

10.2.1 `colcol`

This program extracts chosen columns from a text table with numbers.

>Usage: `colcol datafile maxcoli ncoli colli ... colni`

Parameter description:

`datafile`: input data file.

`maxcol`: maximum (total) number of columns in the input table.

`ncol`: number of columns in the output file.

`col1 .. coln`: position numbers for the columns in the input file to be printed.

10.2.2 `sigcol`

This program prints the number of items, the mean, the standard deviation, a robust estimation of the mean and the standard deviation and the maximum and minimum values from a column in text table. The robust estimator of the mean and standard deviation is based on the median value and the outer quartiles of the data set. This program sometimes gives a better estimate of the average and the standard deviation if the data set has many outliers and a non normal distribution.

>Usage: `sigcol datafile maxcoli sigmacoli`

Parameter description:

`datafile`: input data text file.

`maxcol`: maximum (total) number of columns in the input table.

`sigmacol`: position number of the calculated column in the input file.

10.2.3 `dos2lin` and `lin2dos`

These programs convert Microsoft DOS text files to Unix (Linux) text files and vice versa. The result is printed to standard output.

```
>Usage: dos2lin dosfile.txt
```

```
>Usage: lin2dos linfile.txt
```

10.3 A program to calculate horizontal coordinates

This program calculates the horizontal coordinates and the zenith distance for an astronomical object from the equatorial coordinates of the object, the position on earth, the epoch and the local time. The program use a very accurate algorithm to deal with the precession of Earth. No compensation is made for the refraction in the atmosphere for the Earth.

The algorithms are taken from the book "Practical astronomy with your calculator", Third Edition by Peter Duffett-Smith, Cambridge University Press 1988.

10.3.1 `eq2hor`

```
>Usage: eq2hor halphai malphai salphai
          ddeltai mdeltai sdeltai
          dloi mloi sloi
          dlai mlai slai
          epochi
          yeari monthi dayi houri mini seci
```

Parameter description (note that all are integers):

`halpha malpha salpha`: right ascension for the object in hours, minutes and seconds.

`ddelta mdelta sdelta`: declination for the object in degrees, minutes and seconds.

`dlo mlo slo`: longitude for the observation place on Earth in degrees, minutes and seconds.

`dla mla sla`: latitude for the observation place on Earth in degrees, minutes and seconds.

`epoch`: the astronomical epoch for the right ascension and declination.

`year month day hour min sec`: the local time for the observation.

11 Some Fast Fourier Transform using programs

11.1 Information about the FFT code

The FFT code used in these programs is an old very fast Fortran code that can take data of any size (not only powers of 2) and of any dimension. The code is about 2 times faster than Numerical Recipes code on normal workstations. The C-code used together with these programs is converted with f2c.

The author does not know the copy rights for this code. R. Hook at ESO (Garching bei München) left the code on the computers at Lund Observatory 1993. The code was then part of MIDAS.

The main authors of the code are: Norman Brenner (the Fortran code) and Charles Radar (original BASIC code), MIT Lincoln Laboratory 1967.

11.2 Programs to resample images using FFT interpolation

Here are described some programs to resample images using FFT interpolation. Programs are available for images in GOP format (*.imf) and (*.fff) floating point format.

11.2.1 Programs to sample up images

11.2.2 frebinu and frebinuf

```
>Usage: frebinu inimage outimage binfactori ampr
```

```
>Usage: frebinuf inimage.fff outimage.fff binfactori ampr
```

Parameter description:

inimage: input image in GOP format.

outimage: output image in GOP format.

inimage.fff: input image in (*.fff) floating point format.

outimage.fff: output image in (*.fff) floating point format.

binfactor: resampling factor. The new image will be **binfactor** times larger than the original image.

ampr: arbitrary amplification factor (usually 1.0);

11.2.3 Programs to sample down images

It is important to note the the resolution of the image is decreased with these programs. A resulting pixel value is the simple average of **binfactor**² input pixels. It is also important to always check that the image size is possible to divide evenly with **binfactor**. The programs make no check for this.

11.2.4 frebind and frebindf

```
>Usage: frebind inimage outimage binfactori ampr
```

```
>Usage: frebindf inimage.fff outimage.fff binfactori ampr
```

Parameter description:

`inimage`: input image in GOP format.

`outimage`: output image in GOP format.

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`binfactor`: resampling factor. The new image will be `binfactor` times smaller than the original image. It is important to always check that the image size is possible to divide evenly with `binfactor`. The programs make no check for this.

`ampr`: is an arbitrary amplification factor (usually 1.0);

11.3 FFT convolution of images

This program makes a FFT convolution of an image with an image PSF. Both the image and the PSF are in (*.fff) floating point format.

11.3.1 convf

```
>Usage: convf inimage.fff outimage.fff psf.fff ampr
```

Parameter description:

`inimage.fff`: input image in (*.fff) floating point format.

`outimage.fff`: output image in (*.fff) floating point format.

`psf.fff`: the PSF in (*.fff) floating point format. The PSF should be centered and have an odd size smaller than the image size.

`ampr`: arbitrary amplification factor (usually 1.0);

11.4 Standard Lucy-Richardson deconvolution of images

Here follows a brief description of the standard Lucy-Richardson algorithm. After that follows a description of a program implementation. For theory see Lucy [1974]. Many improvements have to the standard algorithm have been developed during the last years. Among them acceleration and smother background treatment can be mentioned.

11.4.1 Algorithm

Here is a simple pseudo code for the standard Lucy-Richardson deconvolution algorithm:

```
temp = originalimage
for i = 1 to itmax
    tempc = temp
```

```

    tempc = tempc * PSF          /* convolution */

    tempc = originalimage / tempc

    tempc = tempc * PSFconjugate /* convolution */

    temp = temp x tempc

next i

save temp

```

Here `x` and `/` means normal multiplication and division, and `*` means convolution.

`temp` can be a real array and `tempc`, `PSF` and `PSFconjugate` are complex arrays. The `x` and `/` operators only affects the real (not imaginary) parts of the arrays. In the resulting array the imaginary part should be set to zero if it is complex for these operators. The convolution (`*`) normal consists of a complex FFT followed by a complex multiplication and after that an inverse complex FFT of the arrays. It is smart to Fourier transform the PSF only one time before the iterative loop starts.

If `tempc` contains 0 (zeroes) in the expression, `tempc = originalimage / tempc`, these values must be set to 1 before the division.

The `PSF` and `PSFconjugate` arrays should be shifted around zero (modulo size) to avoid that the resulting (saved) image is shifted.

11.4.2 How to use the program

11.4.3 lucyf

```

>Usage: lucyf infile
Parameters: imagename
            savename
            psfname
            saveit_1 saveit_2 ... saveit_n flag(<=0)

```

The parameters are stored in the file `infile` before running the program.

Parameter description:

imagename: image to be convolved in (*.fff) floating point format. The extension (.fff) should not be given.

savename: save name for the deconvolved images (usually the same as `imagename`. The images are saved in the form `savename_it.fff`, where `it` is the number of iterations.

psfname: the name of the PSF in (*.fff) floating point format. The extension (.fff) should not be given. The PSF must be well centered and the background subtracted. The PSF must also have an odd size smaller than the size of the image.

saveit_1 saveit_2 ... saveit_n flag(<=0): are the iterations for which images are saved (up to 1000 numbers). `flag` defines the end of the numbers and is usually a negative number (i.e. `-1`).

12 Some examples how to use the programs

12.1 FINDSTARS measurement of a simulated stellar image

12.1.1 To make the simulated stellar image

It is not an easy task to make realistic simulated stellar images. It we however make some simplifying assumptions it is easier. These assumptions are: Stars with Moffat shape, flat background and no cosmic rays. One can also use an empirical image PSF.

Suppose we want to make a stellar image (`stars4.fff`) with the following data:

Size:	1024 × 1024 pixels
Moffat PSF FWHM:	4.0 pixels
Moffat PSF β :	2.5
Luminosity function:	By Hipparcos
Background:	40 ADU
CCD gain factor:	2.0
CCD read out noise:	5.0 e^- (2.5 ADU)

The maximum possible integrated intensity of the stars is $2.0 \cdot 10^6$ ADU and we want 16000 stars with an integrated intensity > 100.0 ADU. These stars can not be closer than 4.0 pixels to each other.

Then we need only 5 lines to make the image:

```
>Usage: setposlf seedi xsizei ysizei xyinti(0/1) edgei rminr
        nstarsi intmaxr intminr outfile

>Usage: imstarsmf image.fff xsizei ysizei fwhmr betar
        ampr coord(x y int)

>Usage: imaddf inimage.fff outimage.fff termr

>Usage: poissonf inimage.fff outimage.fff seedi gainr

>Usage: normalf inimage.fff outimage.fff seedi sigmaer gainr

>setposlf 3 1024 1024 0 8 4.0 16000 2000000.0 100.0 coord4

>imstarsmf temp1.fff 1024 1024 4.0 2.5 1.0 coord4

>imaddf temp1.fff temp2.fff 40.0

>poissonf temp2.fff temp3.fff 3 2.0

>normalf temp3.fff stars4.fff 3 5.0 2.0
```

12.1.2 To display the stellar image

The image editor XV (for Unix and Linux) can be downloaded from Internet (see the references). Use your favorite image editor or viewer under Windows 98/2000/XP.

It is then easy to display the stellar image:

```
>Usage: fff2tgal image.fff image.tga ampr
```

```
>Usage: fff2tgau image.fff image.tga ampr
```

```
>fff2tgal stars4.fff stars4.tga 250.0 /* Linux and Windows 98/2000/XP. */
```

```
>fff2tgau stars4.fff stars4.tga 250.0 /* Unix. */
```

```
>xv -24 stars4.tga
```

12.1.3 FINDSTARS measurement

The star finding and photometry program FINDSTARS is described in detail in section 3.

To measure the simulated image only 2 lines are necessary:

```
>Usage: findsmbf image.fff fwhmr betar rfitr minintr sigmafacr  
       ampr nstepsi outfile.bin
```

```
>Usage: fsortb findsfile.bin outfile.asc rrejr rstepr
```

```
>findsmbf stars4.fff 4.0 2.5 2.75 10.0 0.0125 1.0 4 stars4.bin
```

```
>fsortb stars4.bin stars4.asc 3.5 0.4
```

12.1.4 To compare the result with original data

The program `sscomp` can be used to compare measured stellar intensities and positions with original data. First we extract the x-, y-position and intensity from the `stars4.asc` file:

```
>Usage: colcol datafile maxcoli ncoli colli ... colni
```

```
>Usage: sscomp file1(x y int) file2(x y int) rmaxr intminr outfile
```

```
>colcol stars4.asc 7 3 2 3 4 >stars4.meas
```

```
>sscomp coord4 stars4.meas 1.0 10.0 star4.sscomp
```

12.2 POLYFIT measurement

Suppose we have a galaxy image (`galaxy.fff`) of 512×512 pixels size containing 3 well sampled stars to be measured. FWHM of the stars is 8.0 pixels and a good PSF (`psf.fff`) has been obtained. The value of `rfit` is set to $12.0 = 1.5 \cdot \text{FWHM}$ pixels. The degree of the background polynomial is chosen to 3. The center of a star is searched in a grid of $5 \cdot 5 = 25$ points inside a square with a side of $2 \cdot 3.0$ pixels around the initial position. Only one iteration is used. Only one star is present in each group. A list of coordinates for the 3 stars is given in a file called (`coord.asc`):

```
113.875 65.987 0.0
387.355 128.079 0.0
256.112 300.827 0.0
```

Convert the coordinate file to a group file:

```
>Usage: c2g coordfile(x y int)
>c2g coord.asc >group.asc
```

And then run polyfit:

```
>Usage: polyfitf image.fff psf.fff maxnstarsi polydegi xystepr xyfacr xyepsr
        gridsi rfitr sigmafacr coordpsr ampr groupfile outfile
>polyfitf galaxy.fff psf.fff 1 3 3.0 0.5 2.0 5 12.0 3.0 0.5 1.0 group.asc galaxy.asc
```

The output is to the file `galaxy.asc`. This file contains the (improved) x- and y-position, intensity, magnitude, background level, square sum and computation time for each star in the input file.

12.3 How to fit a Moffat function to an image PSF

Make a Moffat PSF with `setmofff` and then use `moffitf` and `moffitf2` to fit the parameters. In a real case the PSF is from a star in a real observed image.

```
>Usage: setmofff image.fff sizei fwhmr betar
>setmofff psf.fff 101 8.0 2.5
>Usage: moffitf inimage.fff outimage.fff bgr limr betar
>moffitf psf.fff temp.fff 0.0 500.0 2.5
>Usage: moffitf2 inimage.fff outimage.fff bgr limr betaminr betamaxr betaepsr betastepi
>moffitf2 psf.fff temp.fff 0.0 500.0 1.5 4.0 0.0001 10
```

12.4 Lucy-Richardson deconvolution

Suppose we have an image (`stars4.fff`) and a PSF (`psf4.fff`). And we want to deconvolve the image 1, 2, 5, 10, 20, 50 and 100 iterations. The image has a size of say 512×512 pixels and the PSF has a size of 101×101 pixels. The PSF must be well centered and the background subtracted. It is also good to subtract as much background as possible from the image without affecting the stars.

First we create the input `inlucy` file to the `lucyf` program:

```
stars4
```

```
stars4
psf4
1 2 5 10 20 50 100 -1
```

And then it is very simple to run the program:

```
>Usage: lucyf infile
Parameters: imagename
           savename
           psfname
           saveit_1 saveit_2 ... saveit_n flag(<=0)

>lucyf inlucy
```

12.5 How to display images

12.5.1 How to make and display grey scale images

It is easy to make grey scale images in Targa (*.tga) and Postscript (*.ps) format with these programs. To make a grey-scale RGB-image from a 16-bit GOP image (called star) the commands below can be used. The parameters for `imf2eps` after "-box" are the x- and y-offset and x- and y-size on the printer paper (in mm).

Unfortunately the program `imf2eps` does not work under Windows 98/2000/XP.

The image editor XV (for Unix and Linux) can be downloaded from Internet (see the references). Use your favorite image editor or viewer under Windows 98/2000/XP.

```
>setenv IMGPATH .          /* Unix and Linux. Dot means current directory. */
>imf2tgal star temp.tga   /* Linux and Windows 98/2000/XP. */
>imf2tgau star temp.tga   /* Unix. */
>copy2 star temp
>imf2eps -bw -box 15 50 180 180 temp >temp.ps
>xv -24 temp.tga
>ghostview temp.ps
```

12.5.2 How to make and display coloured images

It is also easy to make coloured RGB images in Targa (*.tga) and Postscript (*.ps) format with these programs. To make a RGB-image from 3 grey-scale 16-bit GOP images (called starr, starg and starb) the commands below can be used. The parameters for `imf2eps` after "-box" are the x- and y-offset and x- and y-size on the printer paper (in mm).

Unfortunately the program `imf2eps` does not work under Windows 98/2000/XP.

The image editor XV (for Unix and Linux) can be downloaded from Internet (see the references). Use your favorite image editor or viewer under Windows 98/2000/XP.

```
>setenv IMGPATH .          /* Unix and Linux. Dot means current directory. */
>copy2 starr tempR
```

```
>copy2 starg tempG
>copy2 starb tempB
>imf2tgal2 temp temp.tga /* Linux and Windows 98/2000/XP. */
>imf2tgau2 temp temp.tga /* Unix. */
>imf2eps -rgb -box 15 50 180 180 temp >temp.ps
>xv -24 temp.tga
>ghostview temp.ps
```

13 Compiling and installing the package

Unpack the "ssphot.tar.gz" file (Unix and Linux) or the "ssphot.zip" (Windows 98/2000/XP) file whichever is appropriate. Use for example WinZip for Windows 98/2000/XP to unpack and for Unix and Linux run the following commands:

```
>gunzip ssphot.tar.gz
>tar -xvf ssphot.tar
```

It is then recommended to add the path for the executables (the under directory /SSPOT/EXE) to your program path both under Unix, Linux and Windows 98/2000/XP. Check so there are no conflicts between the names of other programs installed on the computer.

13.1 Compiling under Unix and Linux

To compile all the C-programs under Unix and Linux the `compall` script can be used. It is assumed that the GNU gcc compiler is used. If another compiler is used the script must perhaps be edited somewhat. The script and source files are located in the EXE directory. The expression `chmod +x` makes the target file executable.

```
>chmod +x compall
>compall
```

The executables only can be removed by the command:

```
>chmod +x rmfiles
>rmfiles
```

13.2 Compiling under Windows 98/2000/XP

Note, executables are already included in the Windows 98/2000/XP package.

To recompile all the C-programs under Windows 98/2000/XP the `compall.bat` script can be used. It is assumed that the GNU DJGPP gcc compiler for DOS under Windows is used (see the Internet references). If another compiler than DJGPP is used the script must perhaps be edited somewhat. The script and source files are located in the EXE directory.

Note, under Windows just double click on the `compall.bat` icon to run the script or run the script at the DOS prompt (command line interpreter under Windows 2000/XP).

```
>compall.bat
```

The executables only can be removed by the command:

```
>delfiles.bat
```

14 References

14.1 Book reference etc

[1] "Practical Astronomy With Your Calculator", Third Edition, by Peter Duffett-Smith, 1988, Cambridge University Press.

[2] The main authors of the Fast Fourier Transform code used in this package are: Norman Brenner (the Fortran code) and Charles Radar (original BASIC code), MIT Lincoln Laboratory 1967.

14.2 Internet references

The home page of the author, the *ssphot* photometry package and this document are found on the web-pages:

<http://www.spaennare.se/index.html>

<http://www.spaennare.se/ssphot.html>

[3] Lund Observatory.

<http://www.astro.lu.se>

[4] ESO (European Southern Observatory. Here can for example the data reduction package MIDAS (with DAOPHOT and ALLSTAR) be downloaded.

<http://www.hq.eso.org>

[5] Numerical Recipes Home Page.

<http://www.nr.com>

[6] DJGPP, a C/C++ compiler for DOS under Windows 98/2000/XP.

<http://www.delorie.com/djgpp/>

[7] XV, an image editor for Unix and Linux.

<http://www.trilon.com/xv/downloads.html>

References

- [1981] Aurière M., Cordoni J.-P., 1981 A&AS 46, 347
- [1984] Blecha R., 1984, A&A 135,401
- [1983] Buonanno R., Buscema G., Corsi C.E., Ferarro F.R., Iannicola G., 1983, A&A 126, 278
- [1980] Buonanno R., Buscema G., Corsi C.E., Iannicola G., 1980, Mem.Soc.Astron.Ital. 51, 483
- [1994] Debray B., Llebaria A., Dubout-Crillon R., Petit M., 1994, A&A 281, 613
- [1988] Gilliland R.L., Brown T.M., 1988, PASP 100, 754
- [1992] Gilliland R.L., 1992, In: Astronomical CCD observing and reduction techniques, A.S.P conf. ser. Vol 23, ed. S.B. Howell, p. 68
- [1998] Holmberg J., Lund Observatory, Sweden, (Luminosity function based on Hipparcos data), Private communication.
- [1989] Kruszewski A., 1989, In: 1st ESO/ST-ECF Data Analysis Workshop, eds. Grosbol P.J., Murtagh F., Warmels R.H., European Southern Observatory, p. 29
- [1989] Linde P., 1989, In: Highlights in Astronomy, Vol 8, ed. Mc Nally D., 651
- [1996] Lindegren L., Lund Observatory, Sweden, Private communication.
- [1993] Lucy L.B., 1993, Workshop: The Restoration of HST Images and Spectra-II, eds. R.J. Hanisch, R.L. White, p. 79
- [1974] Lucy, L., 1974, Astron. J. 79, 745
- [1986] Lupton R., Gunn J.E., 1986, AJ 91(2), 317
- [1989] Mateo M., Schechter P.L., 1989, In: 1st ESO/ST-ECF Data Analysis Workshop, eds. Grosbol P.J., Murtagh F., Warmels R.H., European Southern Observatory, p. 69
- [1989] Mighell K.J., 1989 MNRAS 238, 807
- [1969] Moffat A.J., 1969, A&A 3, 455
- [1986] Penny A.J., Dickens R.J., 1986, MNRAS 220, 845
- [1987] Stetson P.B., 1987, PASP 99, 191
- [1980] Tody D., 1980, Image Processing in Astronomy, SPIE 264, 171
- [1981] West R.M., Kruszewski A., 1981, Ir.Astron.J. 15, 25