



Spitzer Astronomical Point Source EXtraction - APEX

Version 1.9

Release Date: March 1, 2006

Issued by the Spitzer Science Center

California Institute of Technology

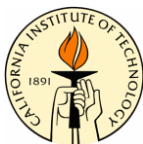
Mail Code 314-6

1200 E. California Blvd

Pasadena, California 91125 USA

<http://ssc.spitzer.caltech.edu/>

help@spitzer.caltech.edu



Issue Dates

Version 1.9 Release: March 1, 2006
Version 1.8 Release: September 30, 2005
Version 1.7 Release: March 29, 2005
Version 1.6 Release: October 15, 2004
Version 1.5 Release: June 4, 2004
Version 1.4 Release: March 7, 2004
Version 1.3 Release: January 7, 2004
Version 1.2 Release: October 20, 2003
Version 1.1 Release: August 28, 2003

Acknowledgements

This document was written by David Makovoz.
The contributions of James Colbert, Francine Marleau, Luisa Rebull, and Gillian Wilson in editing and proofreading of this document are acknowledged.

Table of Contents

1	Overview.....	4
1.1	Input Image Requirements.....	4
1.2	Installation.....	4
2	Using apex.pl, apex_1frame.pl and apex_qa.pl.....	5
2.1	Syntax.....	5
2.2	Namelist.....	7
2.3	Example of a Namelist for apex.pl.....	8
2.4	Example of a Namelist for apex_1frame.pl.....	9
2.5	Example of a Namelist for apex_qa.pl (Multi-frame mode on the left) and (Single-frame Mode on the right).....	10
2.6	Input.....	11
2.7	Bit Set in Quality Control Mask Images.....	12
2.8	Other Parameters Outside Module Blocks.....	12
2.9	Modules.....	14
2.10	Parameters for Modules run by apex.pl and apex_1frame.pl.....	15
2.11	Parameters for run_detect_medfilter and run_extract_medfilter.....	15
2.12	Parameters for run_gaussnoise.....	16
2.13	Parameters for run_pointsourceprob.....	17
2.14	Parameters for run_detect.....	17
2.15	Parameters for run_fit_radius.....	18
2.16	Parameters for run_sourceestimate.....	18
2.17	Parameters for run_aperture.....	20
2.18	Parameters for run_select.....	20
2.19	Parameters for Point Source Subtraction (apex_qa.pl).....	20
2.20	Output.....	21
2.21	Output Table Format.....	21
3	Astronomical Point Source Extraction.....	26
3.1	Processing Stages.....	26
3.2	Use of Uncertainty Images.....	27
3.3	Fiducial Image Frame (FIF) Computation.....	27
3.4	Mosaic Image Creation.....	28
3.5	Background Subtraction.....	28
3.6	Noise Estimation.....	29
3.7	Non-Linear Filtering.....	29
3.8	Point Source Detection.....	30
3.9	Fitting Area Estimation.....	34
3.10	Point Response Function (PRF) Fitting.....	35
3.11	SNR (Signal-to-Noise Ratio) Computation.....	38
3.12	Uncertainties Estimation.....	39
3.13	Variable PRF Input.....	39
3.14	Aperture Photometry.....	40
3.15	Final Selection.....	42

4	References.....	42
---	-----------------	----

1 Overview

The software in this package performs multi-frame (perl script *apex.pl*) and single-frame (*apex_1frame.pl*) point source extraction. For multi-frame point source extraction, the individual images are interpolated and co-added. Point source detection is performed on the co-added images. The main components of point source detection are non-linear matched filtering and image segmentation. They produce a detection list with the potential point sources position estimates. Subsequent point source estimation is performed by PRF (Point Response Function) fitting of the potential point sources from the detection list. The fitting is performed simultaneously in all the input images. It features passive and active de-blending. For single frame point source extraction the detection and estimation are performed on a single input image. Figure 3 illustrates the main processing stages and products of multi-frame point source extraction.

One can create residual images by subtracting point sources from input images using results of point source extraction. The perl script *apex_qa.pl* performs this task for both single-frame (*apex_1frame.pl*) and multi-frame (*apex.pl*) point source subtraction.

The software is available for three platforms: Solaris 2.8, Red Hat Linux 8.0, Mac OS X 10.3 and 10.4.

Additional information on Spitzer and the Post-BCD software is available on the Spitzer Science Center website: <http://ssc.spitzer.caltech.edu/postbcd/documentation.html>.

1.1 Input Image Requirements

The input images have to be in FITS format. This software doesn't process data cubes. The following keywords are required in the headers of the input files for the software to work: BITPIX, NAXIS, NAXIS1, NAXIS2, CRVAL1, CRVAL2, CRPIX1, CRPIX2, CTYPE1, CTYPE2. CD-Matrix elements or/and CDELTA1, CDELTA2, CROTA2 should also be present.

The following projections are implemented so far: "SIN", "TAN", "ZEA" (Lambert's zenithal equal-area), "ARC" (zenithal equidistant), and "STR" (stereographic).

There are no practical restrictions on the size of sky covered by the set of images in order to be processed by the mosaicker. The only nominal restriction is that the set of images should only cover one hemisphere (a solid angle $< 2\pi$ Sr).

1.2 Installation

The software comes in two tar files: *mopex_data_version#.tar* and *mopex_software_version#.tar*. Untarring these files creates the directory *mopex_version#*. Follow the directions in the README file.

The process of point source extraction is split into several tasks, each performed by a separate module. Perl script *apex.pl* runs all the modules involved in multi-frame point source extraction which includes image interpolation and co-addition. Perl script

apex_1frame.pl runs the modules involved in single frame point source extraction. Via a namelist file (Section **Error! Reference source not found.**), one can configure the script to perform a subset of all possible processing steps, which is very useful for reprocessing data. Various processing options are also set in the namelist file.

The package comes with several sets of sample data and sample namelist files, so that the user can verify the functionality of the software in the package.

2 Using apex.pl, apex_1frame.pl and apex_qa.pl

2.1 Syntax

The *apex.pl*, *apex_1frame.pl* and *apex_qa.pl* command-line format is as follows.

1. Multi-frame mode

```
apex.pl <option flags> <specifications>
```

where <option flags> may be: -n, -I, -m, -p, -P, -S, -d, -M, -R, -F any one of which, if specified, must be followed by a blank and a corresponding specification, as follows. The option flags (with corresponding specifications) may be given in arbitrary order.

```
-n namelist: default name apex.nl; file needs to be in cdf/
subdirectory; required,
-I list of input images (list of *.fits): default name image_stack.txt;
required,
-m PRF map (PRFmap.tbl): the PRF map or PRF image is required,
-p PRF image (PRF.fits): default name PRF.fits; file needs to be in
cal/ subdirectory; the PRF map or PRF image is required,
-P mosaic PRF image (Mosaic_PRF.fits): default name Mosaic_PRF.fits;
file needs to be in cal/ subdirectory; required,
-S list of input uncertainty images (list of *_bunc.fits): default name
SigmaList.txt; optional,
-d list of DCE status mask images (list of *_bdmsk.fits): optional,
-M permanently damaged pixels mask image (pmask.fits): file needs to be
in cal/ subdirectory; optional,
-R list of outlier rmask images (list of *_brmsk.fits): optional,
-F FIF filename (FIF.tbl): optional.
```

```
apex_qa.pl <option flags> <specifications>
```

where <option flags> may be: -n, -u, -E, -I, -m, -p, -S, -d, -M, -R, -F any one of which, if specified, must be followed by a blank and a corresponding specification, as follows. The option flags (with corresponding specifications) may be given in arbitrary order.

```
-n namelist for creating residual images: default name apex_qa.nl; file
needs to be in cdf/ subdirectory; required,
-u namelist for mosaicking of residual images: default name
mosaic_qa.nl; file needs to be in cdf/ subdirectory; required,
-E point source list (extract.tbl): required,
-I list of input images (list of *.fits): default name image_stack.txt;
```

required

-m PRF map (PRFmap.tbl): the PRF map or PRF image is required,
-p PRF image (PRF.fits): default name PRF.fits; file needs to be in cal/ subdirectory; the PRF map or PRF image is required,
-S list of input uncertainty images (list of *_bunc.fits): default name SigmaList.txt; optional,
-d list of DCE status mask images (list of *_bdmsk.fits): optional,
-M permanently damaged pixels mask image (pmask.fits): file needs to be in cal/ subdirectory; optional,
-R list of outlier rmask images (list of *_brmsk.fits): optional,
-F FIF filename (FIF.tbl): optional.

Note that in the case where the namelists for creating residual images (e.g. *apex_qa.nl*) and for mosaicking the residual images (e.g. *mosaic_qa.nl*) are merged into one single file (see Namelist Example Section 2.5), the `-n` and `-u` option flags can simply be followed by the same filename.

2. Single-frame mode

`apex_lframe.pl <option flags> <specifications>`

where `<option flags>` may be: `-n`, `-i`, `-m`, `-p`, `-s`, `-C`, `-d`, `-M`, `-R`, `-C` any one of which, if specified, must be followed by a blank and a corresponding specification, as follows. The option flags (with corresponding specifications) may be given in arbitrary order.

-n namelist: default name *apex.nl*; file needs to be in cdf/ subdirectory; required,
-i input image (mosaic.fits): required,
-m PRF map (PRFmap.tbl): the PRF map or PRF image is required,
-p PRF image (PRF.fits): default name PRF.fits; file needs to be in cal/ subdirectory; the PRF map or PRF image is required,
-s input uncertainty image (mosaic_unc.fits): optional,
-C input coverage map image (mosaic_cov.fits): optional,
-d DCE status mask image (mosaic_dmsk.fits): optional,
-M permanently damaged pixels mask image (mosaic_pmask.fits): file needs to be in cal/ subdirectory; optional,
-R outlier rmask image (mosaic_rmsk.fits): optional.

`apex_qa.pl <option flags> <specifications>`

where `<option flags>` may be: `-n`, `-T`, `-E`, `-R`, `-F` any one of which, if specified, must be followed by a blank and a corresponding specification, as follows. The option flags (with corresponding specifications) may be given in arbitrary order.

-n namelist: default name *apex_qa.nl*; file needs to be in cdf/ subdirectory; required,
-T input image (mosaic.fits): required,
-E point source list (extract.tbl): required,
-R mosaic PRF image (PRF.fits): file needs to be in cal/ subdirectory; required,
-F FIF filename (FIF.tbl): optional.

2.2 Namelist

The perl scripts *apex.pl*, *apex_lframe.pl* and *apex_qa.pl* read a namelist file which contains a list of input parameters. The name of this file must be given on the command line. To use a namelist, the *apex.pl*, *apex_lframe.pl* and *apex_qa.pl* command-line format is as follows.

1. Multi-frame mode

```
apex.pl -n <apex_namelist>
apex_qa.pl -n <apex_qa_namelist> -u <mosaic_qa_namelist>
```

2. Single-frame mode

```
apex_lframe.pl -n <apex_lframe_namelist>
apex_qa.pl -n <apex_qa_namelist>
```

The namelist or configuration file contains several blocks of various parameter settings, input image names, and running options. Most of the parameter settings for the modules are in the corresponding blocks delineated by “&” followed by the capitalized module name in the beginning and “&END” at the end of the block. Several parameters affecting more than one module are set outside of the individual modules’ blocks. Also, the locations of the output final and intermediate products are set in the namelist file.

The namelist has the following structure:

1. Lines specifying which modules to be run. For example:

```
run_detect = 0 indicates “do not run”
run_detect = 1 indicates “run”
```

2. For each module, a set of input parameters must be specified. For example:

```
&DETECT
...input parameters...
&END
```

3. The input data, the output directories and several parameters affecting more than one module are set outside of the individual module blocks. For example:

```
IMAGE_STACK_FILE_NAME = ImageList.txt
MOSAIC_PIXEL_SIZE_X = -0.00033
MOSAIC_PIXEL_SIZE_Y = 0.00033
OUTPUT_DIR = apex
```

The input variables **require a space preceding and following the equal sign**, i.e. “variable = value”. An entry like “variable=value” will not be read and the hard-coded

default will be used so you will not notice that your input value is not being used. The lines specifying which modules to be run, the parameters inside and outside blocks, as well as the blocks themselves **can be given in any order in the namelist**. The order will not affect how the modules are run and their output.

2.3 Example of a Namelist for apex.pl

```

have_uncertainties = 1
compute_uncertainties_internally = 0
run_fiducial_image_frame = 1
run_mosaic_interp = 1
run_mosaic_coadder = 1
run_mosaic_combiner = 1
run_mosaic_medfilter = 1
run_medfilter = 1
run_gaussnoise = 1
run_pointsourceprob = 1
run_detect = 1
run_fit_radius = 0
run_sourcestimate = 1
run_aperture = 1
run_select = 1

NICE = 0
verbose = 1
delete_intermediate_files = 0
save_namelist = 1

USE_REFINED_POINTING = 1

use_psp_to_detect = 1
use_background_subtracted_image_for_fitting = 1

select_conditions = "SNR > 5 and deblend ! NO and deblend ! PO
                    and deblend ! AO and deblend ! PAO"
select_columns = "srcid, RA, Dec, x, y, flux, SNR, chi2/dof, deblend"

INPUT_FILE_NAME = mosaic.fits
PRF_FILE_NAME = PRF.fits
SIGMA_FILE_NAME = mosaic_unc.fits
COVERAGE_MAP = mosaic_cov.fits

MOSAIC_PIXEL_SIZE_X = -0.000333333
MOSAIC_PIXEL_SIZE_Y = 0.000333333

OUTPUT_DIR = apex

&SNESTIMATORIN
&END

&FIDUCIALMAGEFRAMEIN
  Edge_Padding = 10,
  CROTA2 = A,
&END

&MOSAICINTIN
&END

&MOSAICCOADDIN
  TILEMAX_X = 2000,
  TILEMAX_Y = 2000,
&END

&MEDFILTER
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

```

```

&MOSAIC_MEDFILTER
  Window_X = 25,
  Window_Y = 25,
  N_Outliers_Per_Window = 100,
&END

&GAUSSNOISE
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

&POINTSOURCEPROB
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  PRF_Xsize = 3,
  PRF_Ysize = 3,
&END

&PSP_GAUSSNOISE
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

&DETECT
  Detection_Max_Area = 5,
  Detection_Min_Area = 2,
  Detection_Threshold = 10,
  Input_Type = 'image_input',
  Threshold_Type = 'peak',
  Extended_Object_Area = 5000,
&END

&FIT_RADIUS
&END

&SOURCESTIMATE
  InputType = 'tile_map',
  Fitting_Area_X = 5,
  Fitting_Area_Y = 5,
  Max_Number_PS = 5,
  Chi_Threshold = 2,
  N_Edge = 1,
  Max_N_Success_Iteration = 10,
  MinimizeFtolSuccess = 0.00001,
  DitherPixelFraction = 0.1,
  DitherFluxFraction = 0.8,
  Background_Fit = 0,
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  Normalization_Radius = 44,
&END

&APERTURE
  N_Apertures = 3,
  Aperture_Radius_1 = 3,
  Aperture_Radius_2 = 4,
  Aperture_Radius_3 = 5,
&END

```

2.4 Example of a Namelist for apex_1frame.pl

```

have_uncertainties = 1
compute_uncertainties_internally = 0
run_detect_medfilter = 1
run_gaussnoise = 1
run_pointsourceprob = 1
run_detect = 1
run_extract_medfilter = 1
run_fit_radius = 0
run_sourceestimate = 1
run_aperture = 1
run_select = 1

NICE = 0
verbose = 1
delete_intermediate_files = 0
save_namelist = 1

use_psp_to_detect = 1
use_background_subtracted_image_for_fitting = 1

select_conditions = SNR > 5 and "deblend ! NO and deblend ! PO
and deblend ! AO and deblend ! PAO"
select_columns = "srcid, RA,Dec,x,y,flux,SNR,chi2/dof,deblend"

INPUT_FILE_NAME = mosaic.fits
PRF_FILE_NAME = PRF.fits
SIGMA_FILE_NAME = mosaic_unc.fits
COVERAGE_MAP = mosaic_cov.fits

OUTPUT_DIR = apex_1frame

&SNESTIMATORIN
&END

&DETECT_MEDFILTER
  Window_X = 25,
  Window_Y = 25,
  N_Outliers_Per_Window = 100,
&END

&EXTRACT_MEDFILTER
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

&GAUSSNOISE
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

```

```

&POINTSOURCEPROB
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  PRF_Xsize = 3,
  PRF_Ysize = 3,
&END

&PSP_GAUSSNOISE
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 500,
&END

&DETECT
  Detection_Max_Area = 5,
  Detection_Min_Area = 2,
  Detection_Threshold = 10,
  Input_Type = 'image_input',
  Threshold_Type = 'combo',
&END

&FIT_RADIUS
&END

&SOURCEESTIMATE
  InputType = 'image_list',
  Fitting_Area_X = 5,
  Fitting_Area_Y = 5,
  Max_Number_PS = 1,
  Chi_Threshold = 3,
  N_Edge = 4,
  Max_N_Success_Iteration = 1000,
  Background_Fit = 0,
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  Normalization_Radius = 44,
&END

&APERTURE
  N_Apertures = 1,
  Aperture_Radius_1 = 9,
&END

```

2.5 Example of a Namelist for apex_qa.pl (Multi-frame mode on the left) and (Single-frame Mode on the right)

```

create_residual_images = 1

IMAGE_STACK_FILE_NAME = image_stack.txt
EXTRACTION_TABLE = extract.tbl
PRF_file_name: PRF.fits
SIGMALIST_FILE_NAME = SigmaList.txt
DCE_STATUS_MASK_LIST = MaskList.txt
PMASK_FILE_NAME = pmask.fits
RMASK_LIST = rmask.txt
FIF_FILE_NAME = FIF.tbl

OUTPUT_DIR = apex_qa
RESIDUAL_DIR = residual

&POINTSOURCEIMAGE
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  X_Column_Name = 'RA',
  Y_Column_Name = 'Dec',
  Input_RA_Dec = 1,
&END

mosaic_residual_images = 1

run_mosaic_interp = 1
run_mosaic_coadder = 1
run_mosaic_combiner = 1

USE_REFINED_POINTING = 1

MOSAIC_PIXEL_SIZE_X = -0.000333333
MOSAIC_PIXEL_SIZE_Y = 0.000333333

MOSAIC_DIR = mosaic

&MOSAICINTIN
&END

&MOSAICCOADDIN
&END

```

```

create_residual_mosaic = 1

MOSAIC_FILE_NAME = mosaic.fits
EXTRACTION_TABLE = extract.tbl
MOSAIC_PRF_file_name: PRF.fits
FIF_FILE_NAME = FIF.tbl

OUTPUT_DIR = apex_qa
RESIDUAL_DIR = residual

&MOSAIC_POINTSOURCEIMAGE
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  X_Column_Name = 'RA',
  Y_Column_Name = 'Dec',
  Input_RA_Dec = 1,
  Normalization_Radius = 3,
&END

```

2.6 Input

Both scripts require input image(s), a namelist (configuration) file and PRF image(s) for processing. They can optionally use uncertainty images and mask images. The purpose of the latter is to mark pixels in the input images unsuitable for processing. The names of the input files for *apex.pl*, *apex_lframe.pl* and *apex_qa.pl* in multi-frame and single-frame mode can be set in the namelist file or on the command line (see Section 2.1). The command line settings override the namelist settings. Except for the namelist and PRF images, the names of the input files can be specified using a relative or an absolute path using the following parameter names:

1. Multi-frame mode

In *apex.nl*:

IMAGE_STACK_FILE_NAME: list of input images (list of *.fits)
 PRF_MAP_FILE_NAME: table identifying the areas of constant PRF (PRFmap.tbl)
 PRF_file_name: name of the PRF image (PRF.fits)
 MOSAIC_PRF_file_name: name of the mosaic PRF image (Mosaic_PRF.fits)
 SIGMALIST_FILE_NAME: list of uncertainty images (list of *_bunc.fits)
 DCE_STATUS_MASK_LIST: list of DCE status mask images (list of *_bdmsk.fits)
 PMASK_FILE_NAME: pmask image (pmask.fits)
 RMASK_LIST: list of outlier rmask images (list of *_brmsk.fits)
 FIF_FILE_NAME: name of the FIF.tbl file (FIF.tbl)

In *apex_qa.nl*:

IMAGE_STACK_FILE_NAME: list of input images (list of *.fits)
 EXTRACTION_TABLE: point source list (extract.tbl)
 PRF_MAP_FILE_NAME: table identifying the areas of constant PRF (PRFmap.tbl)
 PRF_file_name: name of the PRF image (PRF.fits)
 SIGMALIST_FILE_NAME: list of uncertainty images (list of *_bunc.fits)
 DCE_STATUS_MASK_LIST: list of DCE status mask images (list of *_bdmsk.fits)
 PMASK_FILE_NAME: pmask image (pmask.fits)
 RMASK_LIST: list of outlier rmask images (list of *_brmsk.fits)
 FIF_FILE_NAME: name of the FIF.tbl file (FIF.tbl)

2. Single-frame mode

In *apex_lframe.nl*:

INPUT_FILE_NAME: input images (mosaic.fits)
 PRF_MAP_FILE_NAME: table identifying the areas of constant PRF (PRFmap.tbl)
 PRF_file_name: name of the PRF image (PRF.fits)
 SIGMA_FILE_NAME: uncertainty image (mosaic_unc.fits)
 COVERAGE_MAP: coverage map image (mosaic_cov.fits)
 DCE_STATUS_MASK: DCE status mask images (mosaic_dmsk.fits)
 PMASK_FILE_NAME: pmask image (mosaic_pmask.fits)
 RMask: outlier rmask image (mosaic_rmsk.fits)

In *apex_qa.nl*:

MOSAIC_FILE_NAME: input images (mosaic.fits)
 EXTRACTION_TABLE: point source list (extract.tbl)
 PRF_MAP_FILE_NAME: table identifying the areas of constant PRF (PRFmap.tbl)
 MOSAIC_PRF_file_name: name of the PRF image (PRF.fits)
 FIF_FILE_NAME: name of the FIF.tbl file (FIF.tbl)

2.7 Bit Set in Quality Control Mask Images

Quality control mask images can be used in the processing. Three kinds of mask images can be used: permanently damaged pixels masks (Pmask, see Data Handbooks for bit definitions), DCE status masks (Dmask, see Data Handbooks for bit definitions), and outlier masks (Rmask, see Section 4.5 of Mosaicker User's Guide for bit definitions). Normally there will be a single Pmask for a set of input images.

Each bit of the pixel value in a mask image corresponds to a particular condition. A fatal bit pattern is a short integer that has the bits of interest set. Each pixel in a mask image is matched against the fatal bit pattern. If any of the bits specified by the fatal bit pattern is set in the value of a pixel in a mask image, then in the corresponding image the corresponding pixel is considered unusable.

If the name of the mask image or list of mask images is not set, the scripts will proceed without using them. If they are set, then the corresponding fatal bit pattern should be specified in the namelist.

If one of the bits set in a fatal bit pattern is also set in a pixel of the corresponding mask, then the value of the corresponding pixel in the input image is ignored for further processing.

If single-frame outlier detection is performed (see Mosaicker User's Guide), then the Dmasks are updated. If the namelist switch `overwrite_dmask` is set, then the input Dmasks are modified in place. If it is not set, then the modified Dmasks are written in the `DMASK_DIR` directory. If no Dmasks are given, then single frame outlier detection writes them from scratch and sets the corresponding bit. Namelist parameter `DCE_Status_Mask_Radhit_Bit` is used to indicate which bit in the Dmask should be set by single frame outlier detection, and this is the same bit that `mosaic_rmask` copies into its first bit. Unlike the fatal bit patterns in the table above, this parameter is given in terms of the bit number with the default of 9, which is the single frame radhit bit used by the Spitzer masks

The setting of fatal bit patterns for various mask images can be done using the following parameter names:

`DCE_Status_Mask_Fatal_BitPattern` = 32544 (bits 5,8-14)
`PMask_Fatal_BitPattern` = 18304 (bits 7-10,14)
`RMask_Fatal_BitPattern` = 7 (bits 2,1,0)

2.8 Other Parameters Outside Module Blocks

`NICE`: if `NICE` = 1 all the modules called by the script using "nice 19"; default is 0.

`save_namelist`: the namelist used in the current run is always copied to the output directory. By default the name of the namelist is not changed. By setting `save_namelist = 1` the namelist copied to output directory will be given a unique name, which is created by appending the namelist name to the time of execution. For example, if you ran “`mosaic.pl -n myname.nl`” at 12:32:53, then the namelist will be copied to the output directory as `12h32m53s_myname.nl`. The default is 0, in which case the file is copied as `myname.nl`.

`delete_intermediate_files`: if `delete_intermediate_files = 1` is set in the namelist the products of all the modules run this time will be deleted except for the last module; default is 0.

`verbose`: if `verbose = 1`, prints out output of individual modules.

`sigma_weighted_coadd`: if `sigma_weighted_coadd = 1`, the interpolated images are coadded weighted with the uncertainty images.

`create_unc_mosaic`: if `create_unc_mosaic = 1`, the uncertainty images are coadded into `mosaic_unc.fits`.

`create_std_mosaic`: if `create_std_mosaic = 1`, `mosaic_std.fits` is created.

`use_background_subtracted_image_for_fitting`: if `use_background_subtracted_image_for_fitting = 0` in the namelist, input images without background subtraction will be used for point source extraction. The default is 1, which is to perform point source extraction on the background subtracted images.

`use_background_subtracted_image_for_aperture`: if `use_background_subtracted_image_for_aperture = 0` in the namelist, input images without background subtraction will be used for aperture photometry computation. `Use_Annulus = 1` is recommended in this case (see 2.17). The default is 1, which is to compute aperture photometry on the background subtracted images.

`use_psp_to_detect`: by setting `use_psp_to_detect = 2` in the namelist detection will be performed on the background subtracted images. The default is 1, which means detection is performed on the so called PSP (point source probability) images, which are the product of non-linear matched filtering of the input background subtracted images. If it set to 0, the detection is performed on the “Filtered” image (see Section 3.7).

`use_std_to_detect`: if `use_std_to_detect = 1`, then `mosaic_std.fits` is used (see Section 3.7).

`use_unc_to_detect`: if `use_unc_to_detect = 1`, then `mosaic_unc.fits` is used (see Section 3.7).

`create_residual_mosaic`: if `create_residual_mosaic = 1`, creates the residual mosaic.

`create_residual_images`: if `create_residual_images = 1`, creates the residual images.

`mosaic_residual_images`: if `mosaic_residual_images = 1`, creates the mosaic of the residual images.

`MOSAIC_PIXEL_SIZE_X`, `MOSAIC_PIXEL_SIZE_Y`: The size in degrees of mosaic pixel in the x - (y -) directions. `MOSAIC_PIXEL_SIZE_X` should be negative according to the convention that `CDELTA1 < 0`. Takes precedence over `MOSAIC_PIXEL_RATIO_X(Y)` (float).

`MOSAIC_PIXEL_RATIO_X`, `MOSAIC_PIXEL_RATIO_Y`: The ratio of the input pixel x - (y -) size to the mosaic pixel x - (y -) size. (float); default 1. . The default is used if neither mosaic pixel size or pixel ratio are given.

2.9 Modules

The following table shows the modules, the namelist triggers and the descriptions of the modules.

Module	Namelist trigger	Purpose
<i>snestimator</i>	<code>compute_uncertainties_internally*</code>	Compute uncertainties using the model
<i>fiducial_image_frame</i>	<code>run_fiducial_image_frame</code>	Compute Fiducial Image Frame (FIF)
<i>mosaic_int</i>	<code>run_mosaic_int</code>	Interpolate input images to the FIF
<i>mosaic_coadd</i>	<code>run_mosaic_coadder</code>	Co-adds interpolated images into tiles
<i>mosaic_combine</i>	<code>run_mosaic_combiner</code>	Combines co-added tiles into a mosaic image
<i>medfilter</i>	<code>run_detect_medfilter</code>	Background subtraction of the co-added tiles for detection
<i>detect</i>	<code>run_detect</code>	Image segmentation and point source detection. Centroids are computed.
<i>select</i>	<code>run_select_detect</code>	Cleans up the detection table before using it for point source fitting applying user specified constraints and copying user specified columns
<i>medfilter</i>	<code>run_extract_medfilter</code>	Background subtraction of the

		input images (apex.pl) or the coadded tiles (apex_1frame.pl) for extraction
<i>gaussnoise</i>	run_gaussnoise	Estimates background noise in the mosaic image
<i>fit_radius</i>	run_fit_radius	Estimates fitting area size for each detection
<i>sourcetimate</i>	run_sourcestimate	Performs point source estimation by means of profile (PRF) fitting
<i>aperture</i>	run_aperture	Compute aperture photometry for each extraction source
<i>select</i>	run_select	Creates the final table by applying user specified constraints and copying user specified columns

Table 1. Modules, their namelist triggers, and purpose. * If have_uncertainties is set to 1, then the user is expected to provide the uncertainty images name. This overrides the compute_uncertainties_internally option.

2.10 Parameters for Modules run by apex.pl and apex_1frame.pl

Below is a description of the input parameters for each module that can be run by *apex.pl*, *apex_1frame.pl* and *apex_qa.pl*. Refer to the *Mosaicker User's Guide* for a description of the modules that are used for the mosaicking of images.

2.11 Parameters for run_detect_medfilter and run_extract_medfilter.

Outputs background subtracted images.

Window_X,Y (int): x,y-size (in pixels) of window placed around each pixel to estimate the median.

N_Outliers_Per_Window (int): number of pixels excluded from window.

Min_Good_Pixels_In_Window (int): minimum number of good pixels in a sliding window to compute the median.

Min_GoodNeighbors_Number (int): minimum number of adjacent pixel to interpolate the value of missing_pixel.

Max_BadPixels_OutputImage (int): maximum admissible number of NaN pixels in the output image.

&DETECT_MEDFILTER

Window_X = 25,

Window_Y = 25,

N_Outliers_Per_Window = 100,

```

    Min_Good_Pixels_In_Window = 9,
    Min_GoodNeighbors_Number = 4,
    Max_BadPixels_OutputImage = 100,
&END

```

The parameters for *run_extract_medfilter* are identical to the one for *run_detect_medfilter*. The only difference is that the name of the namelist block is preceded with “EXTRACT_”:

```
&EXTRACT_MEDFILTER
```

...

These parameters are used when the background subtraction is done on the mosaic image for the purpose of subsequent detection. The settings should be more aggressive, i.e. smaller window size and number of outliers. For *apex_lframe.pl* background subtraction is done on the mosaic image for point source fitting, as well. In this case the name of the product is different.

2.12 Parameters for *run_gaussnoise*

Estimates the background fluctuation in the image.
Output noise images used for SNR estimation.

Window_X,Y (int): x,y-size (in pixels) of window placed around each pixel to estimate the noise.

N_Outliers_Per_Window (int): number of pixels excluded from window.

Min_Good_Pixels_In_Window (int): minimum number of good pixels in a sliding window to compute the median.

Min_GoodNeighbors_Number (int): minimum number of adjacent pixel to interpolate the value of missing_pixel.

Max_BadPixels_OutputImage (int): maximum admissible number of NaN pixels in the output image.

```
&GAUSSNOISE
```

```
    Window_X = 45,
```

```
    Window_Y = 45,
```

```
    N_Outliers_Per_Window = 500,
```

```
    Min_Good_Pixels_In_Window = 9,
```

```
    Min_GoodNeighbors_Number = 4,
```

```
    Max_BadPixels_OutputImage = 100,
```

```
&END
```

If *use_psp_to_detect* is set and *Input_Type* = “snr_input” for detect, then *gaussnoise* is run on the PSP images as well. The user has to make sure that a block *&PSP_GAUSSNOISE* exists in the namelist. *N_Outliers_Per_Window* in that block should be set to 0, or not set at all. See below in Section Point Source Detection on the pros and cons of doing detection on the snr images.

```
&PSP_GAUSSNOISE
  Window_X = 45,
  Window_Y = 45,
  N_Outliers_Per_Window = 0,
&END
```

2.13 Parameters for *run_pointsourceprob*

Performs linear match filtering.
Outputs point source probability image.

PRF_ResampleX,Y_Factor (int): ratio of the PRF pixel size to the PRF sampling interval in the x- and y-direction.

PRF_X,Ysize (int): size of the portion of the PRF image, in input image pixels, used to convolve with the input image.

Noise_Type: options are 'external_noise' and 'internal_noise'. The first option requires an uncertainty image. For the second option the noise is estimated from the input image.

Apriori_Probability: better left to its default of 0.1.

```
&POINTSOURCEPROB
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  PRF_Xsize = 3,
  PRF_Ysize = 3,
  Noise_Type = 'external_noise',
  Apriori_Probability = 0.1,
&END
```

2.14 Parameters for *run_detect*

Performs image segmentation of point source probability image.
Outputs a detection table (*_detect.tbl).

Detection_Min,Max_Area (int): min/max number of pixels in a cluster to be declared a detection (min: smaller clusters are ignored, max: for bigger clusters, the program raises the detection threshold and either splits the cluster into two or shrinks it down to the required size).

Detection_Threshold (float): program finds pixels above this SNR threshold.

Input_Type (with use_psp_to_detect = 0,1,2): options are 'image_input' and 'snr_input'.
'image_input': use (filtered,PSP,background-subtracted) image for detection.

'snr_input': use SNR image based on (filtered,PSP,background_subtracted) image.

Threshold_Type: options are 'simple', 'combo', 'peak' (see *Image Segmentation* documentation for a description of these options).

```
&DETECT
  Detection_Max_Area = 5,
  Detection_Min_Area = 2,
```

```
Detection_Threshold = 10.,
Input_Type = 'image_input',
Threshold_Type = 'combo',
&END
```

2.15 Parameters for run_fit_radius

Calculates a FitX, FitY value for each detection.
 Adds new columns to the detection table (*_detect.tbl).
 If not run, the fitting will be done for each detection using the value of the Fitting_Area_X,Y defined in SOURCESTIMATE.

```
&FIT_RADIUS
&END
```

2.16 Parameters for run_sourceestimate

Performs point source estimation.
 Fits the input image with the PRF to estimate fluxes and refine positions of the point source candidates from the detection list.
 If Max_Number_PS > 1, the module will attempt to deblend the detections that cannot be successfully fit.
 Outputs a table (*_extract_raw.tbl) which includes the fluxes and improved positions of the point sources, along with other parameters characterizing the fit.

Input_Type: options are 'image_list' and 'tile_map'

'image_list': single frame point source extraction.

'tile_map': multi-frame point source extraction.

Fitting_Area_X,Y (int): size of the area in the input image to be fit with PRF, in input image pixels. If not given the detection list is expected to have FitX and FitY columns, whose content is used for the same purpose.

Max_Number_PS (int): maximum number of point sources to use in an attempt to deblend a detection (active de-blending).

Chi_Threshold (float): maximum acceptable value of χ^2/dof (degrees_of_freedom).

N_Edge (int): positive number excludes point sources in the edge around the input image N_Edge pixels wide from fitting. A negative number appends an edge around the input images N_Edge pixels wide to include the point sources into fitting.

Max_N_Iteration (int): termination criterion in terms of the maximum number of iterations before Chi2_Threshold is reached.

Max_N_Success_Iteration (int): termination criterion in terms of the maximum number of fitting iterations after Chi2_Threshold has been reached.

MinimizeFtol (float): termination criterion in terms of the relative change in χ^2/dof to achieve before Chi_Threshold is reached.

MinimizeFtolSuccess (float): termination criterion in terms of the relative change in χ^2/dof to achieve after Chi_Threshold has been reached.

DitherPixelFraction (float): see Section 3.10.

DitherFluxFraction (float): see Section 3.10.

DeblendDitherPixelFraction (float): same as DitherFluxFraction but applied during active deblending.

Background_Fit (int): switch to fit the background level.

Random_Fit (int): gives the minimization routine a random start, so that the results of fitting on the same set of data will not be identical for every run.

Chi2_Improvement (float): controls the acceptable change in χ^2/dof for incremented number of point sources in active deblending.

PRF_ResampleX,Y_Factor: ratio of the PRF pixel size to the PRF sampling interval in the x- and y-direction.

Normalization_Radius (int): PRF flux is normalized inside this radius; by default the PRF flux is normalized to the whole image.

&SOURCESTIMATE

```

  InputType = 'image_list',
  Fitting_Area_X = 5,
  Fitting_Area_Y = 5,
  Max_Number_PS = 1,
  Chi_Threshold = 3,
  N_Edge = 4,
  Max_N_Iteration = 1000,
  Max_N_Success_Iteration = 0,
  MinimizeFtol = 0.001,
  MinimizeFtolSuccess = 0.00001,
  DitherPixelFraction = 0.1,
  DitherFluxFraction = 0.8,
  DeblendDitherPixelFraction = 1.,
  Background_Fit = 0,
  Random_Fit = 0,
  Chi2_Improvement = 1.,
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  Normalization_Radius = 44,

```

&END

2.17 Parameters for *run_aperture*

Calculates aperture photometry for each point source.

N_Apertures (int): number of different apertures to compute the flux around each point source.

Aperture_Radius_1,2,3 (float): radius of each aperture in pixels.

Use_Annulus is a switch to compute and subtract the background from the aperture flux. The background is computed as a mean or median or mode (Annulus_Compute_Type) of the pixels in the annulus defined by the Inner_Radius and Outer_Radius. It is recommended to combine computing annulus background with using the input mosaic image instead of the background subtracted mosaic image for aperture computation: `use_background_subtracted_image_for_aperture = 0`.

&APERTURE

```
N_Apertures = 1,
Aperture_Radius_1 = 5.0,
Aperture_Radius_2 = 6.0,
Aperture_Radius_3 = 7.0,
Use_Annulus = 1,
Min_Number_Pixels = 10,
Annulus_Compute_Type = 'mode',
Inner_Radius = 15,
Outer_Radius = 25,
```

&END

2.18 Parameters for *run_select*

The perl script *select.pl* is run to create the final table (*_extract.tbl) by applying user specified constraints and copying user specified columns.

Deblend value N,O,P,A indicates whether passive or active deblending has been performed (N: no-deblending has been performed, O: the point source is outside the fitting area, P: passive deblending has been performed, A: active deblending has been performed).

The conditions and the columns do not require any separate namelist block and appear in the namelist as follows:

```
select_conditions = SNR > 5 and "deblend ! NO and deblend ! PO and deblend ! AO and
deblend ! PAO"
```

```
select_columns = "srcid, RA,Dec,x,y,flux,SNR,chi2/dof,deblend"
```

2.19 Parameters for Point Source Subtraction (*apex_qa.pl*)

The module *pointsourceimage* is used to subtract the sources from the input images or the input mosaic. The two modes use different keywords for the namelist block of the module. The keywords are &POINTSOURCENAME and &MOSAIC_POINTSOURCENAME.

X,Y_Column_Name: 'RA' and 'Dec' or 'x' and 'y' (in which case you need to specify the FIF_FILE_NAME).

Input_RA_Dec (int): 0 (pixel coordinates), 1 ('RA' and 'Dec').

Outer_Radius (int): cuts a circular PRF image of this size in pixels; by default the whole image is used.

Hole_Radius (int): cuts a hole in the PRF image of this size in pixels.

Here PRF_ResampleX,Y_Factor and Normalization_Radius are the same as used by *apex.pl*.

```
&POINSOURCEIMAGE
  PRF_ResampleX_Factor = 4,
  PRF_ResampleY_Factor = 4,
  X_Column_Name = 'RA',
  Y_Column_Name = 'Dec',
  Input_RA_Dec = 1,
  Normalization_Radius = 44,
  Outer_Radius = 44,
  Hole_Radius = 21,
&END
```

2.20 Output

The intermediate and final products are written in several subdirectories. The names of the subdirectories can be configured in the namelist file. One can think of *run_<module>* as switches and *<MODULE>_DIR* as routers directing the output of the module to the directory and getting the results of the *<module>* for the module down the stream from that directory. An individual *<MODULE>_DIR* is treated as a subdirectory of *OUTPUT_DIR*. The only exception is if a subdirectory name is specified as a full path, in which case it is not treated as a subdirectory *OUTPUT_DIR* and the name is used as specified.

The default names of the output subdirectories, the keywords used in the namelist, and all the products written in the subdirectory are:

```
OUTPUT_DIR = ./
MEDFILTER_DIR = Medfilter
SIGMA_DIR = Sigma
INTERP_DIR = Interp
COADDER_DIR = Coadd
COMBINER_DIR = Combine
```

OUTPUT_DIR can be specified as a relative or absolute path. If a subdirectory name is specified as a full path it is not treated as a subdirectory *OUTPUT_DIR* and the name is used as specified.

2.21 Output Table Format

The header contains information about the input data, processing parameters copied from the namelist list. In addition the following information is written in the header.

The total number of point sources in the list is given by the keyword `Total_PS_Number`. Three values for the average χ^2/dof for all point sources, for successfully fit point sources, and for failed point sources. Also some statistics on the number of successfully fit point sources, number of point sources for which fitting failed, etc. is written in the header. The values of the pixel size and mosaic image size are recorded. Here is a sample of such a header.

```
\int Number of successful passive deblend detections = 131
\int Number of other successful detections = 940
\int Number of other successful detections resolved into 1 point
sources = 809
\int Number of successful fit point sources from passive deblend
detections = 131
\int Total successful point sources = 940
\int Number of detections with no images left to fit = 158
\int Number of failed passive deblend detections = 7137
\int Number of other failed detections = 17767
\float Percentage of failed detections = 95.8768
\int Number of other detections failed to resolve into 1 point sources
= 10630
\int Number of failed to fit point sources from passive deblend
detections = 7137
\int Total failed point sources = 17767
\float Average chi2 = 4.42162
\float Average chi2 for successfully fitted point source = 1.77383
\float Average chi2 for failed point sources = 4.5617
\int Total_PS_Number = 18865
\float CDELTA1 = -0.000167
\float CDELTA2 = 0.000167
\int NAXIS1 = 1890
\int NAXIS2 = 2176
\int Fitting_Area_X = 5
\int Fitting_Area_Y = 5
```

`Fitting_Area_X` and `Fitting_Area_Y` are written in the header if they are set in the namelist. Otherwise the data from columns `FitX` and `FitY` from the detection table are used for the fitting area size and written in the columns `FitX` and `FitY` of this table.

The table itself has over 20 columns whose meaning is described in the Table 2.

#	Column Name	Description	
1	srcid	The index of the extracted point source.	
2	detid	The index of the detection from the detection table.	
3	N_PS	Number of point source to fit the data simultaneously. N_PS > 1 indicates passive or active de-blending.	
4	RA	Right Ascension of the point source	
5	Delta_RA	RA uncertainty	
6	Dec	Declination of the point source	
7	Delta_Dec	Dec uncertainty	
8	Delta_RAD	RA-Dec cross-correlation term ($\text{sign}(\langle \text{RA Dec} \rangle) \times \sqrt{ \langle \text{RA Dec} \rangle }$)	
9	x	x-coordinate of the point source in the mosaic image.	
10	delta_x	x-coordinate uncertainty.	
11	y	y-coordinate of the point source in the mosaic image.	
12	delta_y	y-coordinate uncertainty.	
13	delta_xy	x-y cross-correlation term ($\text{sign}(\langle xy \rangle) \times \sqrt{ \langle xy \rangle }$)	
14	flux	Flux of the point source, units are specified in the header	
15	delta_flux	Flux uncertainty	
16	bckgrnd	Background estimate; constant in the fitting area. It is produced if Background_Fit = 1 is set in the namelist (or -back is given on the commandline)	
17	chi2/dof	$\chi^2 / (\text{number of degrees of freedom})$ of the fit.	
18	ps_chi2/dof	Partial $\chi^2 / (\text{number of degrees of freedom})$ of one point source in the fit. Different from the previous column for passive de-blending. See below the explanation of the difference between the two.	
19	status	The fit is considered successful if χ^2 / dof is lower than Chi2_Threshold.	
		Status value	Meaning
		-4	Initialization in Simplex failed

		-3	Number of degrees of freedom ≤ 0 . The reasons for "losing" pixels are bad pixels and point sources on the edge or even outside of the image.
		-2	Fitting converged, i.e. the changes in χ^2/dof were smaller than the limit set by <code>MinimumFtol</code> in the namelist
		-1	The maximum number of iterations <code>Max_N_Iteration</code> has been reached
		0	Fitting has not been performed
		1	Successful fitting; terminated because converged, i.e. the changes in χ^2/dof were smaller than the limit set by <code>MinimumFtolSuccess</code> in the namelist
		2	Successful fitting; terminated because number of iteration exceeded the maximum of <code>Max_N_Success_Iteration</code>
20	SNR	It is the ratio of the point source estimated flux to the value of the corresponding pixel in the noise image.	
21	nfunk	Produced only in the debug mode (-d) and contains the number of times χ^2 calculations that has been performed in the process of fitting.	
22	PRFPortion	The ratio of the sum of PRF over the pixels used in the calculation to the total PRF norm.	
23	N_dof	The number of the degrees of freedom, i.e. the number of pixels used in the fitting minus the number of the fitting parameters.	
24	FitX	The x-size of the fitting area for this point source (written if <code>Fitting_Area_X</code> is not set in the namelist and <code>FitX</code> from the detect table is used).	
25	FitY	The y-size of the fitting area for this point source (written if <code>Fitting_Area_Y</code> is not set in the namelist and <code>FitY</code> from the detect table is used).	
26	depth	The number of input images used in fitting this point source.	
27	deblend	Indicates whether passive or active de-blending has been performed.	
		deblend value	meaning
		N	No de-blending has been performed

		O	The point source is outside of the fitting area
		P	Passive de-blending has been performed
		A	Active de-blending has been performed
		There can be the following combination of the values above: N, NO, P, PO, A, AO, PA, PAO.	
28+2n	aperture#	N_Apertures columns giving the aperture photometry	
29+2n	bad_pix#	N_Apertures columns giving unusable portion of the aperture	

Table 2 The format of the output point source table.

Figure 1 below illustrates the difference between χ^2/dof and ps_chi^2/dof , which only exists for passive de-blending. In this case the fitting area W_j is a combination of the fitting areas W_j^n for each individual detection in the blend. χ^2/dof is calculated by applying Equation 2 over the whole fitting area W_j and counting dof for the whole fitting area, whereas ps_chi^2/dof for each point source is calculated by applying Equation 2 to its individual W_j^n and counting dof only for this area.

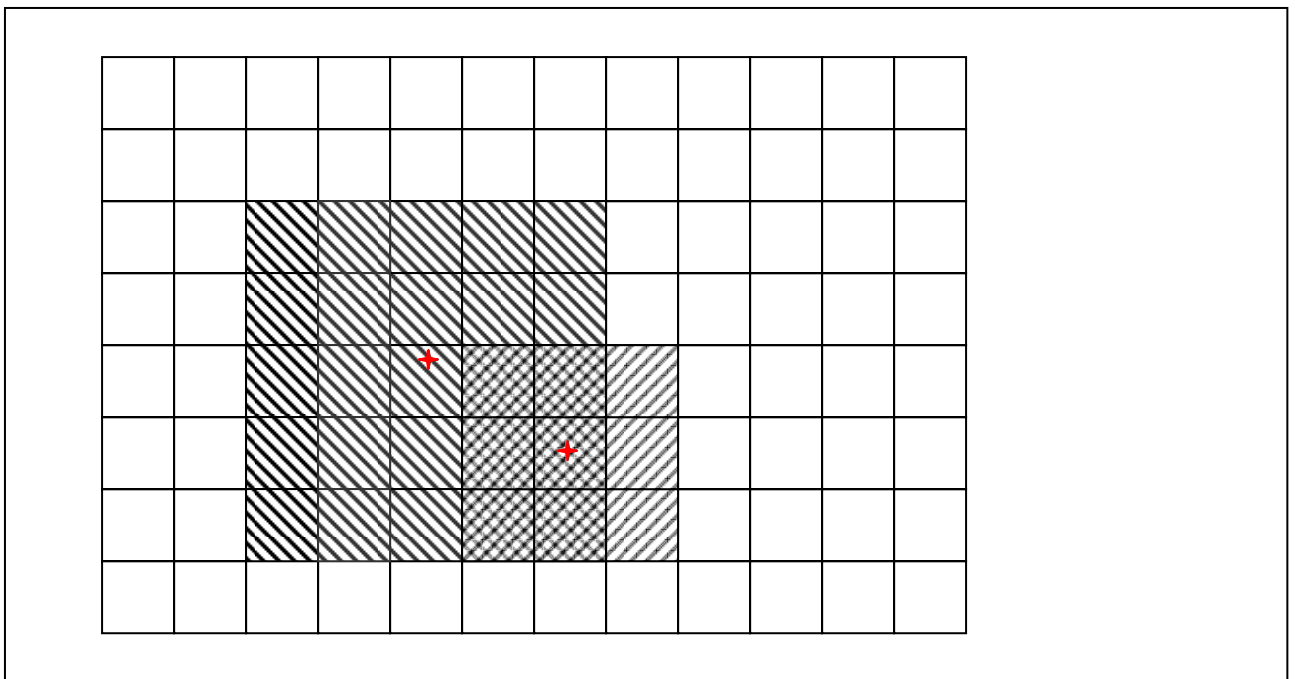


Figure 1. Total fitting area W_j for two detections, represented by the two red stars, consists of two fitting areas W_j^1 and W_j^2 , each one striped differently.

3 Astronomical Point Source Extraction

3.1 Processing Stages

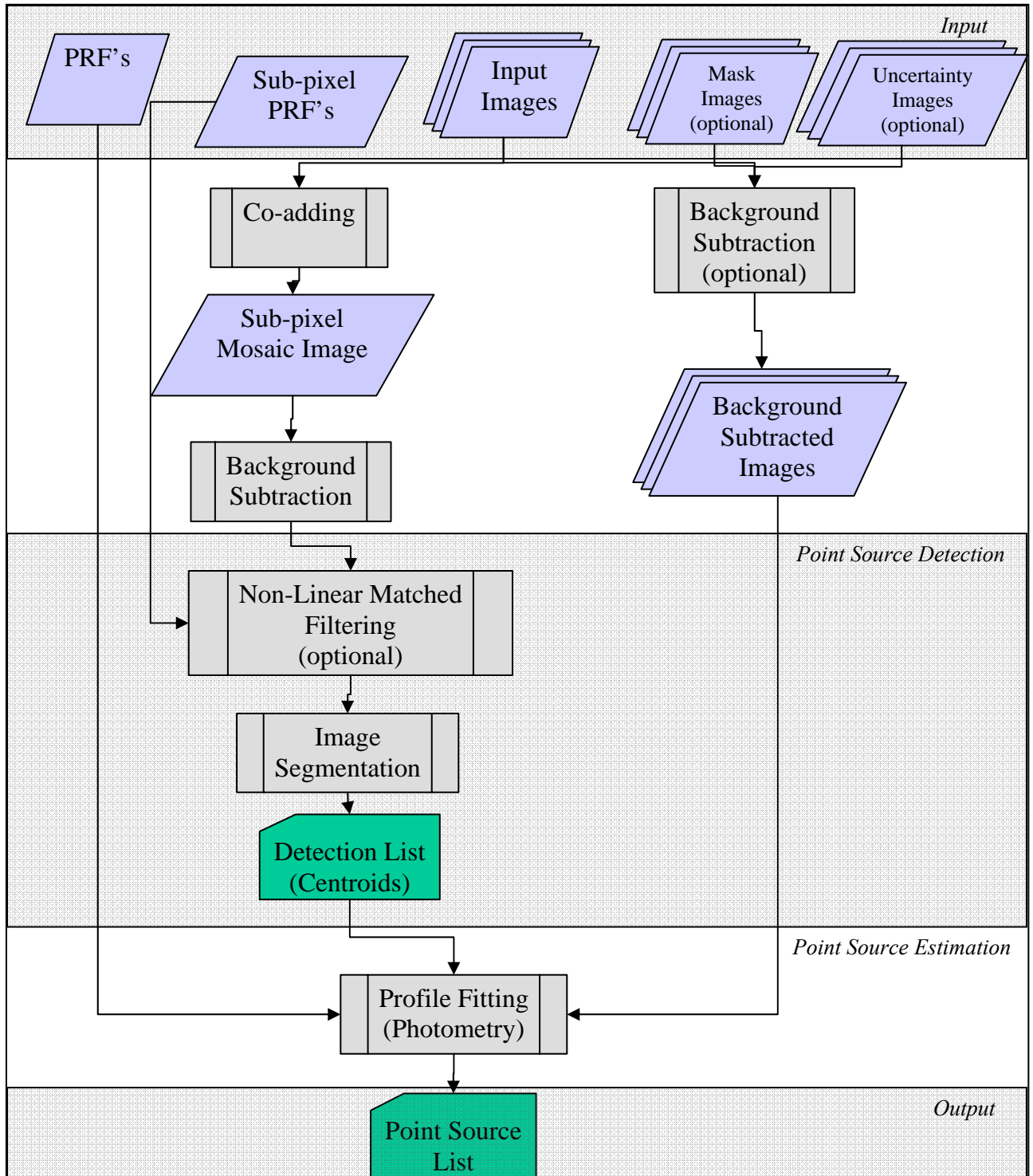


Figure 2. The main processing stages and products of multi-frame point source extraction.

3.2 Use of Uncertainty Images

If the `have_uncertainties` switch is set, then, regardless of the setting of the `compute_uncertainties_internally` switch, the script expects to find a list of uncertainty images. The uncertainty images are interpolated and co-added. The co-added uncertainty images are used in the co-addition of interpolated input images.

If `compute_uncertainties_internally` is set and `have_uncertainties` is not set, then the *sne* module is executed. It produces uncertainty images, which are subsequently interpolated, and used for co-addition.

If neither `compute_uncertainties_internally` nor `have_uncertainties` is set, then interpolation, co-addition and point source extraction is performed without using the uncertainty images.

Module *sne* estimates pixel uncertainty for each pixel in the image using the following model:

$$\sigma = \sqrt{\frac{\sigma_{readnoise}^2}{g^2} + \frac{\sigma_{confusion}^2}{g^2} + \frac{I}{g}}.$$

Here the parameters of the model `Read_Noise` $\sigma_{readnoise}$, `Gain` g , and `Confusion_Sigma` $\sigma_{confusion}$ are specified in the namelist. The last term is the Poisson noise determined by the pixel value I .

The units of `Read_Noise` and `Confusion_Sigma` here are electrons. The module is designed to work with the images in DN units. If you are working with the images in the units of surface brightness the gain should include the conversion factor from the surface brightness units to *electrons*. The product of this step is the uncertainty images.

If `sigma_weighted_coadd` is set then the mosaic image is created by averaging the interpolated images with the interpolated uncertainty images as well as the coverage maps. By default the mosaic image is created by averaging the interpolated images weighted only with the coverage maps.

3.3 Fiducial Image Frame (FIF) Computation

This step is optional. If a table with the fiducial image frame had been created previously it can be used by supplying its name in the namelist file using `FIF_FILE_NAME` keyword.

This step creates a unified grid coordinate system that is used for creating a mosaic image. Given a list of input images, the perl script *fiducial_image.pl* creates a list of pointing parameters - `CRVAL1`, `CRVAL2`, `CRPIX1`, `CRPIX2`, `CROTA2`, `CDELTA1`, `CDELTA2`, `NAXIS1`, `NAXIS`. Run subsequently, module *fiducial_image_frame* generates the pointing parameters for the bounding region of a minimal size that encloses the input images. Even if the input images use the CD matrix

convention, the mosaic image (and by extension the interpolated images) will use the set of keywords `CDELTA1`, `CDELTA2`, and `CROTA2`, since the mosaic image is undistorted.

Two namelist parameters are used by this module. `Edge_Padding` specifies the size of the margin in arcsec padded around the FIF on all four sides. `CROTA2`, if set, specifies the orientation of the FIF. If `CROTA2 = A`, then the orientation of the FIF is found by averaging the twist angles of the input images. If `CROTA2` is not set the program will compute the optimal orientation. The product of this step is the `FIF.tbl` table.

3.4 Mosaic Image Creation

Module `mosaic_int` performs projection of input images onto the FIF. Each interpolated image occupies a portion of the FIF. If given, uncertainty images are also interpolated. The interpolation is based on the area overlap between the input and output pixels. The pixel size of the output interpolated images is determined in the namelist by the following two parameters: `MOSAIC_PIXEL_RATIO_X` and `MOSAIC_PIXEL_RATIO_Y`. These parameters specify the ratio of the input image to the interpolated image pixels sizes. The FIF is recomputed based on the new pixel size and is saved in the file `fif.tbl`. Module `mosaic_coadd` performs co-addition of the interpolated images. If uncertainty images are given and interpolated they are used to compute the weighted average of the interpolated pixel values. If not, straight averaging is performed. By default all the interpolated images will be co-added into one mosaic image. If the co-addition for the whole mosaic image cannot be performed because of the computer memory constraints the user can request the mosaic image to be tiled. The suggested tile size is set in the namelist file. For more details on how image interpolation and co-addition are performed see the document entitled “Spitzer Mosaicker,” available from the SSC website. The products of this stage are the updated FIF file `fif.tbl`, interpolated images, uncertainty images, coverage maps, co-added tiles, co-added uncertainty tiles, coverage maps for the co-added tiles, the complete mosaic image, the mosaic of the background subtracted image, the uncertainty mosaic image, and the coverage mosaic image.

Point source detection is performed on the tiled co-added images. Tiling is done if the all the images cannot be kept in RAM in order to perform co-addition of the whole mosaic. If at all possible the tile size should not be set, so that there will be one tile equal to the whole mosaic. The downside of using several tiles is that the point sources near the internal edges of the tiles will cause multiple detections. A special clean-up step should be added to remove the redundant entries from either the detection list or the extraction list. The current version of the software doesn’t include this step and it is up to the user to make sure there are no duplicate entries in the detection table.

3.5 Background Subtraction

Module `medfilter` performs background estimation in the input images and outputs background subtracted images. Background subtraction is performed on the input images and the co-added images. The values in the namelist are used for background subtraction of the input images.

The module is run twice. The first time is it applied to the mosaic image, or the coadded tiles. It is done for the purpose of subsequent point source detection. The second time it

is applied either to the input images (*apex.pl*) or to the mosaic image (*apex_lframe.pl*) for the purpose of subsequent point source fitting. The settings for `Window_X`, `Window_Y`, and `N_Outliers_Per_Window` for the first run for the detection should be more aggressive (smaller values) than for the the fitting.

The program computes an asymmetrically skewed median for each pixel in the input image using a rectangular window of `Window_X` by `Window_Y` size. It is achieved by omitting `N_Outliers_Per_Window` highest pixels from each median window. If `N_Outliers_Per_Window` is set to 0 the program calculates the regular median.

There is a minimum required number of good (not-NAN and not marked by any mask) pixels per median window `Min_Good_Pixels_In_Window`. If the number of good pixels is below this threshold then the corresponding pixel in the output image is marked as a “missing” pixel. When the median calculation is finished, the values of the marked pixels are interpolated from the neighboring pixels for which the median has been found. In order to do so the program scans around the pixel in question and accumulates values of good pixels. When the number of accumulated values reaches or exceeds the minimum number given by the input parameter `Min_GoodNeighbors_Number` the program finds the average and stores this value as the median for the pixel in question.

If the number of “missing” pixels exceeds the maximum number given by the parameter `Max_Bad_Pixels_OutputImage`, the programs aborts, printing the appropriate error message.

3.6 Noise Estimation

Module *gaussnoise* estimates the background fluctuations in the images. It is very similar to *medfilter*. It finds the 68 percentile range of the pixel values in the sliding window, which is the estimate of the Gaussian noise. It can output the ratio of the input image to the Gaussian noise, which is saved as a signal-to-noise ratio (snr) image.

It is run once to compute noise in the co-added tiles. These noise tiles are used to estimate the SNR of the extracted point source. If non-linear filtering is not performed and module *detect* uses “`snr_input`”, *gaussnoise* will produce in addition to the noise images the snr images which will be subsequently used for image segmentation. The user needn’t do anything extra to produce the snr images. If `use_psp_to_detect` is set and `Input_Type = “snr_input”` for *detect*, then *gaussnoise* is run on the PSP images as well. The user has to make sure that a block `&PSP_GAUSSNOISE` exists in the namelist. `N_Outliers_Per_Window` in that block should be set to 0, or not set at all. See below in Section 3.8 on the pros and cons of doing detection on the snr images.

3.7 Non-Linear Filtering

Module *pointsourceprob* performs non-linear matched filtering and outputs point source probability (PSP) images. It is applied to the co-added images. It uses the following namelist parameters: `PRF_ResampleX_Factor`, `PRF_ResampleY_Factor`, `PRF_Xsize`, `PRF_Ysize`, `Noise_Type`, `Apriori_Probability`.

This step is performed to improve detectability of the point sources. It is an optional step and it is run if `use_psp_to_detect` switch is set to 1 or 0 in the namelist. It can be

shown that using the ideas of maximizing SNR in the image the following filter can be derived (see the document entitled “Bayesian Estimation of Point Source Probability”):

$$P(j) = \left(1 + \frac{1 - \text{Apriori_Probability}}{\text{Apriori_Probability}} \exp \left(- \frac{\left(\sum_i \frac{s(i) \cdot \text{PRF}(j-i)}{\sigma^2(i)} \right)^2}{2 \sum_i \text{PRF}^2(j-i)} \right) \right)^{-1}$$

Equation 1

Here s is the input background subtracted image, σ is the input uncertainty image. If `use_unc_to_detect` is set, then `mosaic_unc.fits` or the corresponding tiles are used. If `use_unc_to_detect` is not set and `use_std_to_detect` is set, then `mosaic_std.fits` or the corresponding tiles are used. See `Spitzer_mosaicker` for the details on what those images are and how they are created.

The output of this filter $P(j)$ at pixel j can be interpreted as a probability of having a point source above the noise at this pixel. The summation is for all pixels i within the area defined by the namelist parameters `PRF_Xsize` and `PRF_Ysize` around pixel j . The values of $P(j)$ should use the full dynamic range from `Apriori_Probability` to 1. If the uncertainty is not well estimated it may lead to the output of the filter either not using up the whole range or having too many pixels saturated very close to 1. To prevent this from happening the argument of `exp` function is rescaled in order to utilize the full dynamic range of P from `Apriori_Probability` to 1. If `Noise_Type = "internal_noise"` instead of using the uncertainty image a single estimate of pixel uncertainty is derived from the image itself and used for all pixels in the image.

If `use_psp_for_detection = 0`, then Filtered image is used for detection. Filtered image F is defined as the product of the point source probability image times the background subtracted image: $F = P * s$.

3.8 Point Source Detection

Module `detect` performs image segmentation and computes the centroids for the detected pixel clusters. The input to this module depends on namelist parameter `Input_Type` and switch `use_psp_for_detection`; see Table 2.

<code>use_psp_to_detect</code>	<code>Input_Type</code>	Input Image
2	“snr_input”	SNR image based on the background subtracted input image
2	“image_input”(default)	Background subtracted input image
1(default)	“snr_input”	SNR image based on the PSP image

1(default)	“image_input”(default)	PSP image
0	“snr_input”	SNR image based on the Filtered image
0	“image_input”(default)	Filtered image

Table 3 Input image for *detect*.

The program starts by computing the initial threshold based on the parameter `Detection_Threshold` specified by the user. Upon the first pass the program finds all the pixels above the initial threshold. It creates a list of all contiguous clusters of pixels above the initial threshold. Then it compares the number of pixels with the minimum and maximum allowed sizes of a cluster specified by the user through parameters `Detection_Min_Area` and `Detection_Max_Area`. If the number of pixels in a particular cluster is less than the minimum number `Detection_Min_Area` the cluster is discarded. If the number of pixels in a cluster is greater than the maximum number `Detection_Max_Area`, the program goes through an iterative process of raising the threshold with the intention of either shrinking the cluster and/or splitting it into smaller clusters.

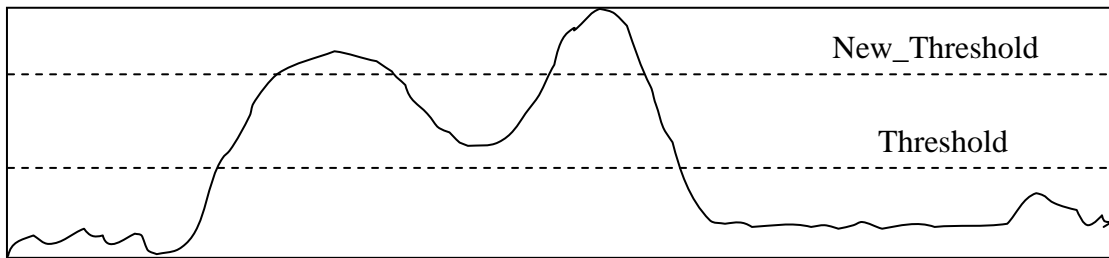


Figure 3: Raising the threshold splits a big cluster into two smaller clusters.

To this end, the program recalculates the threshold for this particular cluster and finds all the pixels above the new threshold. See Figure 3 for an illustration of this process. When the iterative procedure is finished, a list of estimated detection locations is created. The centroid is found for each cluster, which is the estimated location of the point source corresponding to this cluster.

$$Centroid_X = \frac{\sum_{i \in cluster} X(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)} \quad Centroid_Y = \frac{\sum_{i \in cluster} Y(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)}$$

The value of the greatest pixel in the cluster is saved as the flux.

The output of this program is used for point source extraction. Point source extraction performs passive de-blending. The detected point sources, determined to be in close proximity of one another, such that their PRF's overlap, are fit simultaneously. This module provides the classification of detections as candidates for passive de-blending. If a cluster created by the initial thresholding is consequently split into several clusters, the latter are classified as a blend of clusters. Figure 4 is an illustration of several possible cases of rejected and detected clusters, including a blend of clusters. There are two

columns in the output table, `Blendid` and `Blendsize`, that are used for detection blend classification. `Blendid` keeps track of the sequential number of a detection blend in the table. `Blendsize` gives the number of detections in the blend. The columns have the same values for each detection in a particular blend. For non-blend detections the columns are set to 0. See an example of the output table below.

There is one issue with processing co-added images. Due to the variable coverage the noise level, being inversely proportional to the square root of the coverage, varies throughout such an image. One way to deal with this problem is to use the *gaussnoise* module to produce an snr image. *gaussnoise* produces a local estimate of the noise and therefore the effects of the variable coverage will be reflected in the snr image. There are two problems with this approach. First, it is time consuming. Second, the process of raising threshold to split/shrink clusters has been designed with the `Input_Type = "image_input"` in mind. It is not clear how it will work for snr images. The alternative is to use a coverage map (specified in the namelist by `CoverageMap_Filename`). The coverage map is used to attenuate the co-added images, i.e. an input coadded image is multiplied by the $\sqrt{\text{coverage map}}$, if a coverage map is provided. For multi-frame point source extraction for `Input_Type = "image_input"` the coverage map is provided automatically by the script *apex.pl*. For the single frame point source extraction when working with mosaic images it is user's responsibility to produce and specify a coverage map in the namelist of script *apex_1frame.pl*.

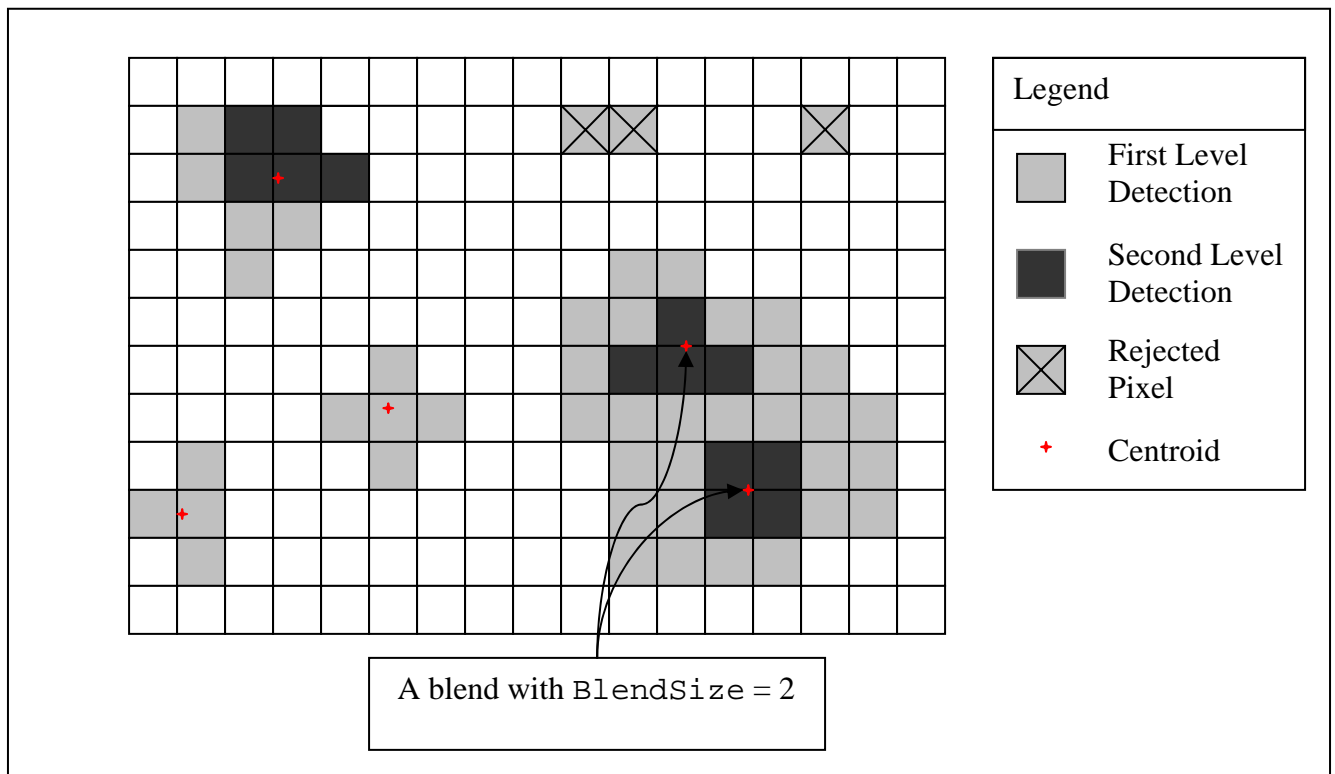


Figure 4. An example of image segmentation with `Detection_Min_Area = 3`; `Detection_Max_Area = 9`. A blend of detections with `BlendSize = 2` is shown.

The parameter `Probability_Threshold` is meant to be used for the PSP images produced by the `pointsourceprob` module (Non-Linear Filtering). The PSP images are results of non-linear filtering of regular images. They have the maximum value of 1. They very often have a cluster of pixels with the values saturated very close to 1. If the probability threshold is set, then pixels greater than the probability threshold are excluded from calculation of the initial threshold. Without using it there is a possibility of having the initial threshold greater than 1, which will lead to having no detections.

See the document entitled, “Image Segmentation” for a detailed description of the algorithm and I/O in module `detect`.

In the single frame mode there is one detection list produced in this step. In the multi frame mode the detection is performed for each co-added tile. The detection lists for each tile are combined into one detection list `mosaic_detect.tbl`. Below is an example of such a table.

```
\char comment = Output from DETECT, version 1.00
| srcid|      x|      y|      flux| BlendId| BlendSize|
|  i   |      r|      r|      r   |      i |      i |
|-----|-----|-----|-----|-----|-----|
|  0   |  2.50 | 229.50 | 1.328e+03 | 0      | 0      |
|  1   |  3.30 |  77.50 | 1.369e+03 | 0      | 0      |
|  2   |  3.40 | 190.55 | 1.338e+03 | 1      | 2      |
|  3   |  4.82 | 190.24 | 1.435e+03 | 1      | 2      |
|  4   |  4.50 | 111.50 | 1.332e+03 | 0      | 0      |
|  5   |  8.40 | 205.00 | 1.275e+04 | 2      | 3      |
|  6   |  5.50 | 205.00 | 1.330e+03 | 2      | 3      |
|  7   |  6.38 | 206.70 | 1.853e+03 | 2      | 3      |
|  8   |  5.50 | 127.50 | 1.328e+03 | 0      | 0      |
```

Optionally, perl script `select` can be run to select user specified columns and rows that satisfy the constraints specified by the user in the namelist and copies them into the detection table. The namelist trigger is `run_select_detect`. The format of the condition is as following. The columns are listed in a coma separated fashion:

```
select_detect_columns = "srcid,x,y"
```

By default all the columns are copied. Either the name or the number of the desired column can be used. The constraints are combined with the only available logical operation of “and”:

```
select_detect_conditions = "deblend_size < 10"
```

“Not equal to” is represented by the “!” sign. By default all rows are copied.

Normally, the user might want to select out very big blends, since for them point source extraction may not be done reliably. Another usage would be to remove *deblend_size* and *deblend_id* columns (the way it is shown above), which is the way to avoid doing passive deblending.

3.9 Fitting Area Estimation

The module *fit_radius* reads in a detection list with two required fields “x” and “y”. It reads the image corresponding to the detection list and the uncertainty image. The input image is mean subtracted. The module also optionally reads a coverage map. If a coverage map is given, then the uncertainty values are multiplied by the sqrt(coverage). For each detection a search is performed in x and y directions. The program starts at the detection pixel and goes up and down in the y-direction and left and right in the x-direction. It compares the values of the mean-subtracted input image and the corresponding uncertainty values. It finds the closer distance one has to go in each direction until the mean-subtracted image pixel values drop below the uncertainty level.

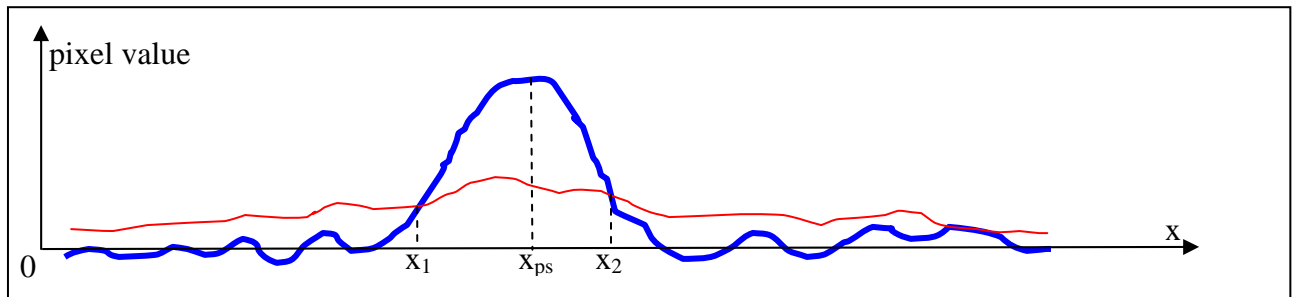


Figure 5. The mean-subtracted input image (blue, thick) and the uncertainty image (red) have two intersection points x_1 and x_2 . x_{ps} is the position of the detection. In this case the fitting size $FitX = 2(x_2 - x_{ps})$, since $(x_2 - x_{ps}) < (x_{ps} - x_1)$.

All the content of the input table is copied into the output table. Two more columns *FitX* and *FitY* are added.

```
\char comment = Output from DETECT, version 1.00
| srcid | x | y | flux | BlendId | BlendSize | FitX | FitY |
| i | r | r | r | i | i | i | i |
0 | 2.50 | 229.50 | 1.328e+03 | 0 | 0 | 5 | 5 |
1 | 3.30 | 77.50 | 1.369e+03 | 0 | 0 | 7 | 5 |
2 | 3.40 | 190.55 | 1.338e+03 | 1 | 2 | 5 | 9 |
3 | 4.82 | 190.24 | 1.435e+03 | 1 | 2 | 7 | 10 |
4 | 4.50 | 111.50 | 1.332e+03 | 0 | 0 | 3 | 4 |
5 | 8.40 | 205.00 | 1.275e+04 | 2 | 3 | 15 | 14 |
6 | 5.50 | 205.00 | 1.330e+03 | 2 | 3 | 5 | 6 |
7 | 6.38 | 206.70 | 1.853e+03 | 2 | 3 | 4 | 5 |
8 | 5.50 | 127.50 | 1.328e+03 | 0 | 0 | 3 | 5 |
```

Running of this module is optional. Another option is to specify the fitting area size for *sourceestimate* using namelist parameters *Fitting_Area_X* and *Fitting_Area_Y*. However in this case the same fitting area size is used for all point sources. If both namelist parameters *Fitting_Area_X*, *Fitting_Area_Y* and columns *FitX*, *FitY* in the detect table are given as input for point source fitting, the former is used.

3.10 Point Response Function (PRF) Fitting

Final point source position and photometry estimation are performed at this stage for all point source candidates on the detection list by module *sourceestimate*. For each point source candidate the data in the input images are fit with the Point Response Function (PRF). A number of parameters are set in the namelist for this module. The input to the module is determined by the switch

`use_background_subtracted_image_for_fitting`. If it is set to 1, which is the default, the fitting is performed in the background subtracted images. Otherwise, the input images are used without subtracting the background. For the multi-frame point source extraction the fitting is performed simultaneously in all input images.

`InputType` should be set to “`tile_map`” in this case. For single frame point source extraction, `InputType` is set to “`input_image`”. The size of the fitting area is determined by the namelist parameters `Fitting_Area_X` and `Fitting_Area_Y`, if they are given. If they are not given, the module *fit_radius* should have been run to estimate the size of the fitting area for each detection. This is the preferable method of setting the size of the fitting areas, since the size will vary with the brightness of the point sources – brighter sources in general will have larger fitting areas. The user should be aware, however, that the module *fit_radius* is not fool proof and may produce erroneous results.

The goal of the fitting is to estimate the fluxes and to refine the positions of the point sources. The n -th point source is characterized by the flux $f(n)$ and mosaic position $\mathbf{R}(n)$ (see Figure 7). If namelist parameter `Background_Fit` is set to 1, then the fitting will also estimate the *background*, which is assumed to be constant within the fitting area. For the i -th input image the data in fitting area W_i around the potential point source position is used in the fitting. The local coordinates of the point sources $\mathbf{R}_i(n)$ in the i -th input image are in general a complicated function of the global coordinates $\mathbf{R}(n)$. The following quantity is minimized:

$$\chi^2 = \sum_{j=1}^{N_Images} \sum_{i \in W_j} \frac{\left(s_j(i) - \sum_{n=1}^{N_{PS}} f(n) PRF(i, \mathbf{R}_j(n)) - background \right)^2}{\sigma_j^2(i)}$$

Equation 2

Here s_j is the input image, possibly background subtracted, σ_j is the input uncertainty image, fitting is performed simultaneously in N_Images images using N_{PS} point sources candidates. First, χ^2 is minimized for $N_{PS}=1$. The success of the fitting is determined by the namelist parameter `Chi_Threshold`. If $\chi^2/dof < Chi_Threshold$, the fitting is successful. The number of degrees of freedom *dof* is equal to the sum of all the good pixels in the fitting areas W_j in all the images minus the number of fitting parameters.

If the fitting is not successful and namelist parameter `Max_Number_PS` > 1, active deblending can be performed. The same data is fit with more than one point source. N_{PS} is incremented until it reaches the limit set by the namelist parameter `Max_Number_PS` or $\chi^2/dof < Chi_Threshold$ is satisfied, whichever comes first. Since active de-

blending is not a well defined process in order to accept an extra point source the new $(\chi^2 / dof)_{new}$ should satisfy the following condition

$$(\chi^2 / dof)_{new} < \frac{Chi_Threshold + \chi^2 / dof}{2 \cdot Chi2_Improvement}.$$

This is meant to prevent the algorithm from splitting detections into several point sources, unless it has a really good reason to do so.

If the `BlendSize` > 0 then passive de-blending is performed and all the detections from one blend are fit simultaneously, i.e. N_{PS} is set to `BlendSize`, even if `BlendSize` > `Max_Number_PS`. The fitting area in this case is a more complicated than a simple rectangle and is a combination of all the rectangles for each detection in the blend (see Figure 1). Passive de-blending has been proven to be an essential component of point source extraction.

In order to minimize χ^2 , the Simplex algorithm has been modified. The original downhill Simplex algorithm minimizes a function by using the values of the function at several vertices and trying to move away from the highest vertex. There are four basic ways in the original Simplex to move a vertex: reflection, expansion, contraction and shrinkage.

Two modifications have been made. They are illustrated in Figure 6.

First, reflection is modified in the following way. If there is an indication that reflection is done almost parallel to the lines of constant χ^2 , it is replaced with moving the highest vertex in the direction *perpendicular* to the reflection direction. Second, contraction and shrinkage are replaced with line minimization.

Two namelist parameters `DitherFluxFraction` and `DitherPixelFraction` determine the initialization of Simplex algorithm. Each detection is split into three vertices. Their positions \mathbf{r}_i and fluxes f_i are initialized by randomly dithering the position \mathbf{R} and flux F of the detection:

$$\begin{aligned} \mathbf{r}_i &= \mathbf{R} + rand \times DitherPixelFraction \\ f_i &= F \times (1 + rand \times DitherFluxFraction), \end{aligned}$$

where *rand* is a random number uniformly distributed from -1 to 1.

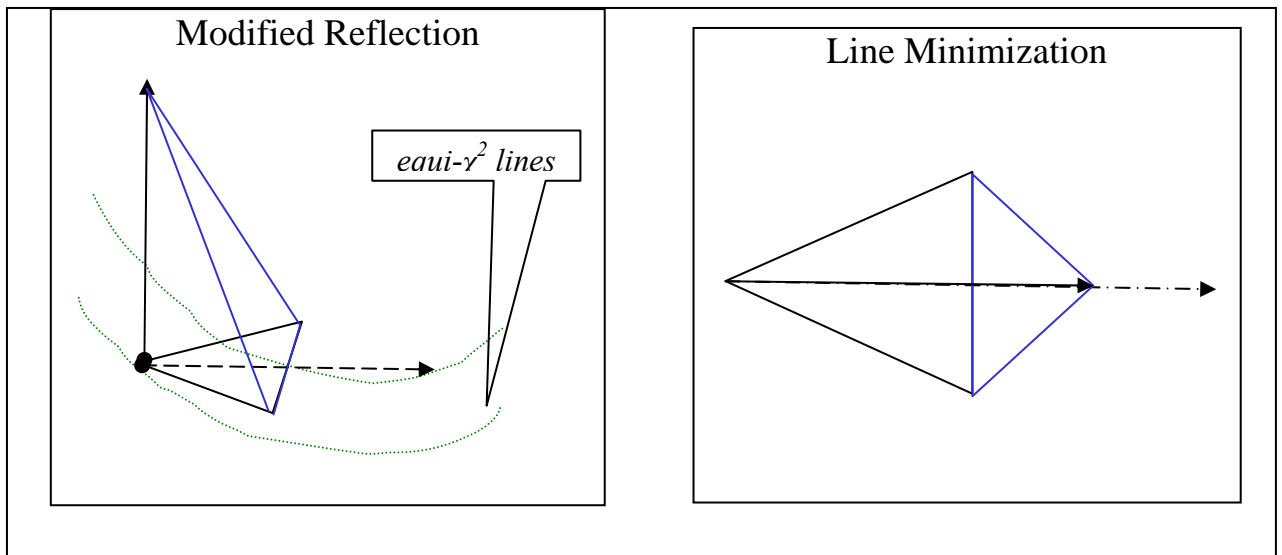


Figure 6 Illustration of the two modifications made to the original Simplex method.

The unsuccessful termination of the fitting for each point source is determined by the two parameters `MinimizeFtol` and `Max_N_Iteration`; if successful, then the two parameters `Max_N_Success_Iteration` and `MinimizeFtolSuccess` govern the termination. If the number of iterations exceeds `Max_N_Iterations` or the relative change in χ^2 becomes smaller than `MinimizeFtol` before χ^2/dof reaches `Chi_Threshold`, the fitting for this point source terminates and the point source is assigned the corresponding failure status. If, on the other hand, the fitting is successful and χ^2/dof drops below `Chi_Threshold`, the program will continue fitting in order to improve the results until either the number of iterations after reaching `Chi_Threshold` exceeds `Max_N_Success_Iterations` or the relative change in χ^2 becomes smaller than `MinimizeFtolSuccess`. After that, the point source is assigned the corresponding success status.

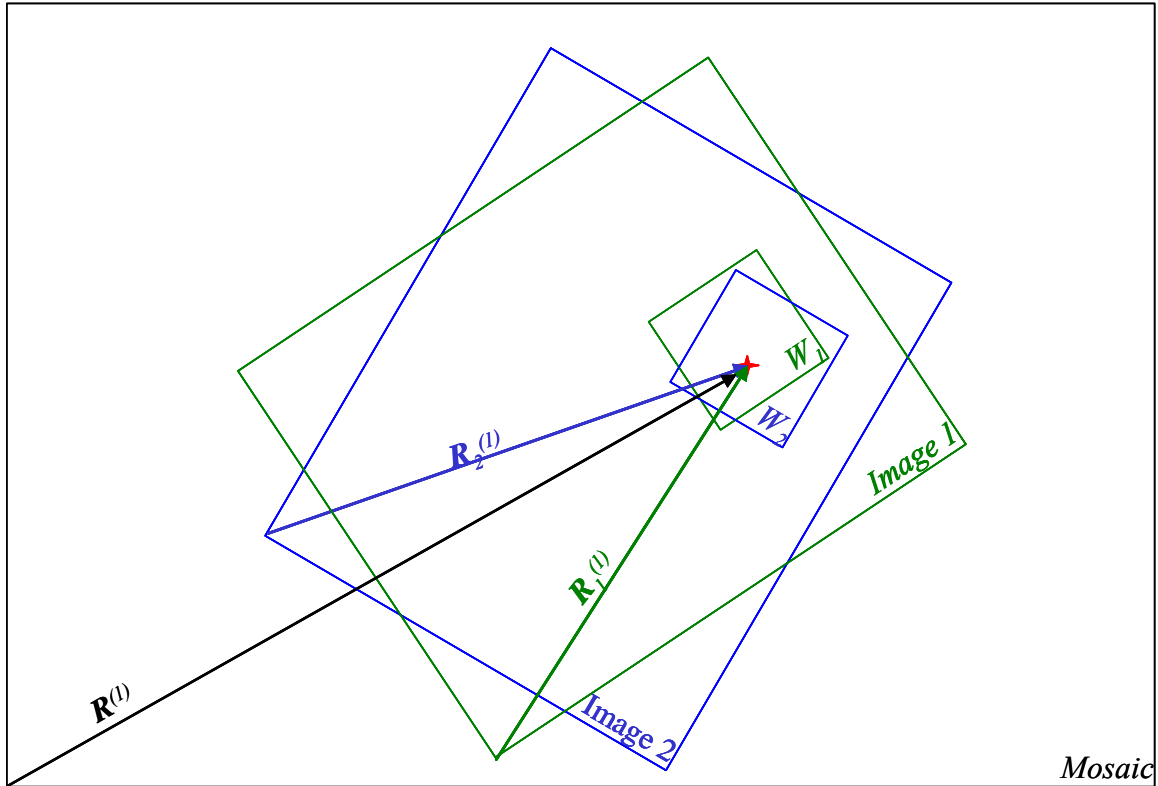


Figure 7 Point source 1 (red) has global coordinates $R(I)$ and local coordinates $R_1(I)$ and $R_2(I)$ in images 1 and 2, correspondingly.

3.11 SNR (Signal-to-Noise Ratio) Computation

In order to compute the SNR of the extracted point sources the image of the background fluctuations in the mosaic image is produced. The images `coadd_Tile_#_Image_noise.fits` are produced by *gaussnoise*. The noise in those images is attenuated by the value of noise pixels NP , defined as the equivalent number of pixels whose noise contributes to the measurement of the flux of a point source from the image. The value of NP is computed as

$$NP = \frac{1}{PRF_normalized(center_pixel)}.$$

Then the extracted flux F_{ps} for a point source is divided by the attenuated noise $\sigma(x_{ps}, y_{ps})$ at the location (x_{ps}, y_{ps}) of the point source to yield the SNR_{ps} for the point source:

$$SNR_{ps} = \frac{F_{ps}}{\sigma(x_{ps}, y_{ps}) \cdot NP}$$

Equation 3

The product of this stage is the extraction table `extract.tbl` that has over 20 columns, including point source positions in the sky and mosaic coordinates, fluxes, positional and flux uncertainties, SNR, etc. Output Table Format has the description of the output parameters. The output `extract.tbl` is subsequently modified by module `aperture`.

3.12 Uncertainties Estimation

The uncertainties of the fitting parameters are determined by the Hessian matrix:

$$H_{ab} = \frac{\partial^2 \chi^2}{\partial \lambda_a \partial \lambda_b}(\bar{\lambda}_a, \bar{\lambda}_b)$$

where $\bar{\lambda}_a$'s are the optimal fitting parameters : (\mathbf{R}_n, f_n)

Provided the fit is successful the covariance matrix C_{ab} of the fitting parameters is defined by the inverse Hessian matrix:

$$C_{ab} = 2H_{ab}^{-1}$$

In particular the positional (`delta_x`, `delta_y`) and flux (`delta_flux`) uncertainties computed by the program are equal to `delta_x = Cxx`, `delta_y = Cyy`, `delta_flux = Cff`.

3.13 Variable PRF Input

A provision is made to use a variable in the field of view PRF using the so-called PRF maps. It is assumed that several areas in the detector field of view will be identified, such that PRF variation within each area will be negligible and a constant PRF can be used for each area. A PRF map identifies the areas of constant PRF.

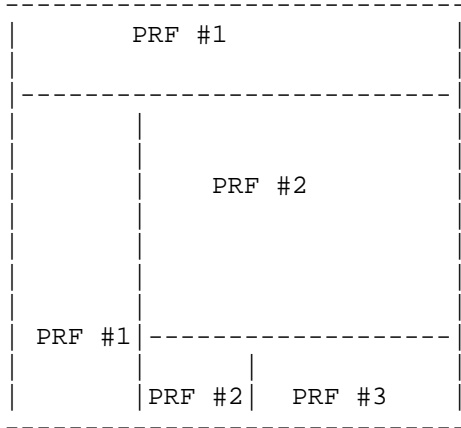
A PRF map maps an image with respect to the PRF image used for any pixel of the image. The keyword `Number_PRF` gives the number of different PRF's that should be used for an image. The names of the files containing the PRF's are given by the keywords `PRF_Filename_#`, where `#` runs from 1 to `Number_PRF`. The PRF's can be in the form of a fits file (`.fits`) or an IPAC table (`.tbl`). The keyword `HaveSigma` is used to indicate whether PRF Sigma images are available. If they are, `HaveSigma = 1`, `HaveSigma = 0` otherwise. The names of the images with the Sigmas are given by the keywords `PRFSigma_Filename_#`. The keywords `ImageX` and `ImageY` give the sizes of the images for which the PRF's are to be used.

Here is a sample PRF Map file.

```
\char comment = PRF Map
\int Number_PRF = 5
\int HaveSigma = 1
\int ImageX = 256
\int ImageY = 256
\int PRF_Filename_1 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\int PRF_Filename_2 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\int PRF_Filename_3 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\int PRFSigma_Filename_1 =
/ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.Sigma.fits
\int PRFSigma_Filename_2 =
/ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.Sigma.fits
\int PRFSigma_Filename_3 =
/ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.Sigma.fits
```

PRFNum	NAXIS1	NAXIS2	PRFPos1	PRFPos2
i	i	i	i	i
1	100	200	1	1
2	256	56	1	201
3	50	50	101	1
2	156	150	101	51
1	106	50	151	1

The geometry for the above table file:



The PRF is normalized before fitting:

$$norm = \sum PRF(i);$$

$$PRF_normalized(i) = PRF(i) / norm.$$

If Normalization_Radius is specified the summation is performed over the pixels i with the specified radius from the PRF center.

3.14 Aperture Photometry

Module *aperture* calculates aperture photometry for each point source. The photometry is calculated on the background subtracted image. The number of apertures (N_Apertures) and the apertures themselves (Aperture_Radius_#) are specified in the namelist. The program integrates the flux encircled by the aperture centered on the point source (see Figure 8). The script *apex.pl* uses the background subtracted mosaic image (*mosaic_minback.fits*) for this purpose. Aperture photometry AP is equal to

$$AP = \sum_i a_i \cdot I_i,$$

which is the sum of the products of the pixel values I_i weighted with the exact area overlap a_i .

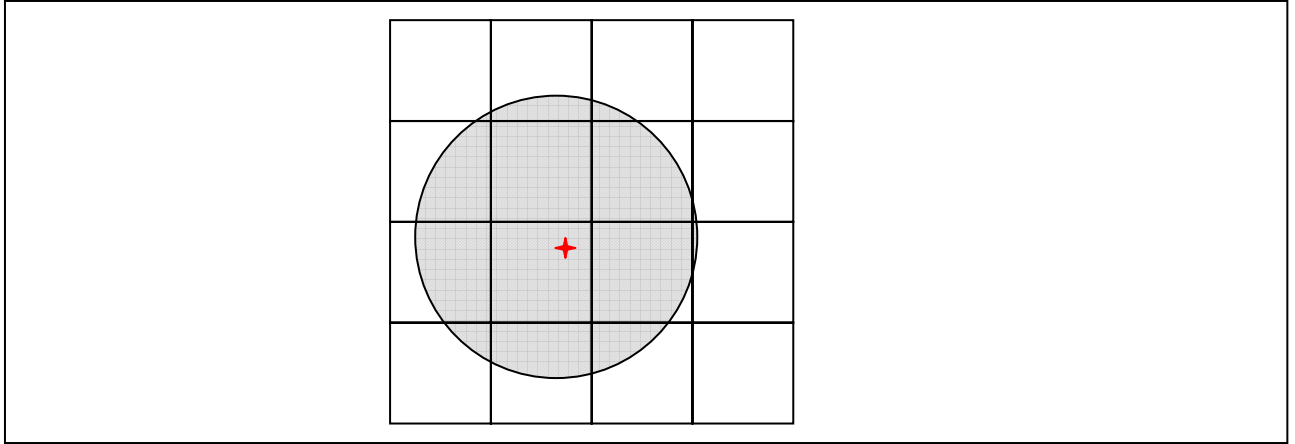


Figure 8: A circle of a specified radius centered on the point source (red) is overlaid on the image and aperture photometry is computed by summing the pixel values weighted by the overlap area.

If the image is in surface brightness units (FITS keyword `BUNIT = MJy/Sr` or `microJy/square_arcsec`), it computes the results in the units of μJy . The module `aperture` produces an output table `aperture.tbl` and modifies the extraction table by appending new columns with aperture photometry and the number of bad pixels for each aperture. See Output Table Format for the description of the format of the modified extraction table.

Optionally, if `Use_Annulus` is set, the background B can be estimated and subtracted from the aperture photometry for each point source. It is estimated within an annulus of a user specified size.

$B = \text{Operation}(\text{Pixels with } R \geq \text{Inner_Radius} \text{ and } R \leq \text{Outer_Radius})$. R is the distance from the point source position to the center of the pixel, fractional pixel computation is not done, pixel is either in or out.

Operation is defined by the `Annulus_Compute_Type` keyword and can be of three types: 'mean', 'median' (default), and 'mode'.

`Min_Number_Pixels` is the minimum number of good (non-NaN) pixels in the annulus to compute the background. If the number of good pixels $< \text{Min_Number_Pixels}$, then the background is not estimated and the aperture photometry is written in the output table without background subtraction. The background subtracted aperture photometry is computed as

$$AP = \sum_i a_i \cdot I_i - B \sum_i a_i$$

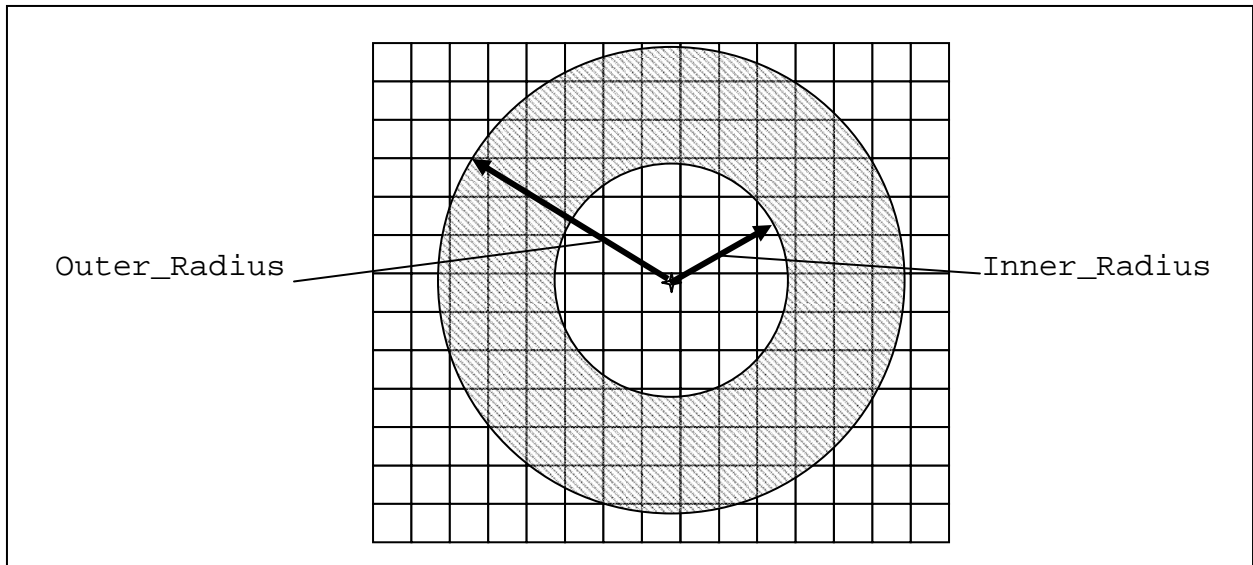


Figure 9. The geometry of the annulus used for background estimation.

3.15 Final Selection

The perl script *select* selects user specified columns and rows that satisfy the constraints specified by the user in the namelist and copies them into the final extraction table. The format of the condition is as following. The columns are listed in a coma separated fashion:

```
select_columns = "1,3,RA,Dec"
```

Either the name or the number of the desired column can be used. The constraints are combined with the only available logical operation of "and":

```
select_conditions = "SNR > 20 and flux > 0 and  $\chi^2/\text{dof} < 3$  deblend ! NO"
```

"Not equal to" is represented by the "!" sign.

4 References

The following documents can be found on the Spitzer Science Center website at <http://ssc.spitzer.caltech.edu>.

1. APEX Performance
2. Bayesian Estimation of Point Source Probability
3. Image Segmentation
4. Mosaicker User's Guide