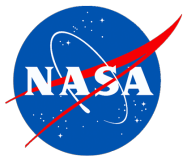


MOPEX User's Guide

MOsaicking and Point-source EXtraction

Spitzer Heritage Archive Documentation

Post-BCD Tools Team and Science User Support Team

Version 18.5.1, July 2018

Contents

Chapter 1. Introduction

MOPEX (MOsaicking and Point Source Extraction) is a package developed at the Spitzer Science Center for astronomical image processing. MOPEX requires input images in FITS format. As an example, if the user has basic calibrated FITS images from the Spitzer Space Telescope archive, MOPEX can be used to level the backgrounds of the input images, co-add them together to produce a mosaic with optional rejection of outliers (e.g. cosmic rays), and perform point source extraction and/or aperture photometry on the detected sources.

MOPEX can be run in two ways - either from the command line, or using the Graphical User Interface (GUI). The GUI is more user-friendly, especially for first-time users, but the command line contains some additional functionality that has not yet been implemented in the GUI. MOPEX is available for Mac, Linux and Solaris (see the MOPEX Download page for specific availability).

NEW (November 2010) Appendix C gives the current best correction factors for PSF-fitted fluxes for all Spitzer instruments.

NEW (March 2012) Appendix D gives some tips for Spitzer data.

1.1 Important Documentation

The following documents and webpages are recommended reading, and are referred to throughout this User's Guide:

- Spitzer Data Analysis Cookbook: http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/cookbook/
- IRAC Instrument Handbook: http://irsa.ipac.caltech.edu/data/SPITZER/docs/irac/iracinstrumenthandbook/
- MIPS Instrument Handbook: http://irsa.ipac.caltech.edu/data/SPITZER/docs/mips/mipsinstrumenthandbook/
- IRS Instrument Handbook: http://irsa.ipac.caltech.edu/data/SPITZER/docs/irs/irsinstrumenthandbook/
- MOPEX main page: http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/mopex/
- MOPEX download page:

http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/mopex/downloadmopex/

- MOPEX Bug List:

  http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/mopex/mopexbug/
- Spitzer Helpdesk:
  http://irsa.ipac.caltech.edu/data/SPITZER/docs/spitzerhelpdesk/

1.2 User Interface

The Graphical User Interface (GUI) and the command-line versions of MOPEX are packaged and released together, and all updates to both versions are done through the GUI. Both versions use the same set of C++ and C programs to carry out the data reduction. Generally, users may wish to start with the GUI as it provides a more intuitive interface, nice visualization of each processing step and its output, and includes useful image display tools. Users may use the command-line version to script their data reduction, take advantage of the slightly faster processing time, or to use some of the additional functionality not yet available through the GUI.

1.3 Getting Help

Online help for MOPEX is available from the Help function in the GUI. A full list of the known bugs, along with suggested solutions or work-arounds, can be found on the MOPEX Bug List. The Spitzer Helpdesk also contains a KnowledgeBase of frequently asked questions.

If you still have questions about MOPEX then you can get help by submitting a ticket to the Spitzer Helpdesk or emailing us at help@spitzer.caltech.edu. Please help us by including the following information:

- The operating system you are running (e.g. Mac OS10.6 or Linux RHEL);
- Which version of MOPEX you're using (GUI or command line) and the version number (e.g. 18.5.0 Final);
- The AOR ID of your data (if Spitzer data);
- Attach your namelist, unless it's a standard MOPEX template, in which case tell us which one;
- A description of the error message you saw;

- If you are using the GUI, please attach the most recent log file, found in the directory *~/.spot/*
- As much information as possible about the problem, including any processing steps you carried out since downloading the data from the archive.

Please note that our system does not autoreply, so if you receive a reply from us within a few minutes then please check it for further information.

1.4 Why MOPEX?

MOPEX provides a general-purpose tool to generate mosaics of FITS images and to perform point source extraction. While other tools are available for these tasks, MOPEX addresses several complications specific to Spitzer data:

- **Geometric Distortion:** Spitzer images have significant geometric distortion, described in the FITS header using the World Coordinate System (WCS). Mosaicking and photometry must accurately interpret this information, which is not yet standard in all packages.

- **Fast Projection:** Large Spitzer mosaics include many images, so a fast method is required when performing the projection. MOPEX's fast direct plane-to-plane projection is described in *Makovoz, D. 2004, PASP, 116, 971-974*.

- **Outlier Rejection:** Like many datasets, Spitzer data suffer from cosmic rays and other artifacts. The best algorithm for rejecting these outliers depends upon the observing mode, e.g. simple temporal outlier rejection works well with highly redundant sky coverage, but it can fail without it. MOPEX has single-frame spatial, temporal and two spatial-plus-temporal outlier rejection methods that can be adjusted for different observing modes.

- **Profile Fitting:** Spitzer pixels are relatively big relative to point sources, so MOPEX fits with Point Response Functions (PRFs), which take into account intra-pixel sensitivities. For multi-frame photometry, MOPEX does detection on a mosaic image, but point-source fitting on the original stack of images.

In addition to solving these specific problems, MOPEX provides some useful tools for mosaicking and source extraction. Mosaicking options include background matching, drizzle interpolation, exposure time weighting, and error propagation. Photometry options include detection using image segmentation, use of a previous list of detected positions for photometry, aperture photometry, and residual image generation.

1.5 How To Use This Manual

This manual is split into the following parts:

- Introductory material for MOPEX in general, including installation instructions and a quick start guide;
- An overview of the software and its capabilities;
- Input files and requirements for MOPEX;
- An in-depth description of all the pipelines and modules available in MOPEX;
- Basic Concepts in MOPEX, with in-depth discussions of the algorithms implemented;
- Running MOPEX on the command line;
- Appendices, with useful tables and supplementary material.

The bulk of this document describes how to run MOPEX from the GUI, but Chapter 9 provides details of how to run MOPEX from the command line.

For the purposes of this manual, we will call the directory containing your data <*data_dir*>, the directory in which you have installed MOPEX <*mopex_dir*>, and the directory from which you are planning to run MOPEX <*working_dir*> (which may be the same as <*data_dir*> or <*mopex_dir*> but doesn't have to be).

1.6 Installing MOPEX and Downloading Your Data

MOPEX can be used to reduce all Spitzer data except for spectra from the IRS. This includes imaging with IRAC, MIPS and the IRS Peak-Up arrays, and MIPS SED mode. MOPEX is distributed as a package containing both the GUI and command-line versions, and is available at the following URL, with full download and setup instructions:
http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/mopex/

If you are running MOPEX on a Mac, please ensure that you drag the MOPEX icon to either your Applications folder, or some other location with write privileges. MOPEX will not work if you try to run it from the read-only directory created by the .dmg file.

Once you have installed MOPEX, start the GUI by double-clicking on the MOPEX icon, or by running the following on the command line:

```
unix% /<mopex_dir>/mopex
```

for Linux and Solaris systems, and:

```
unix% /<mopex_dir>/mopex.app/Contents/MacOS/JavaApplicationStub
```

on Macs.

1.6.1 Getting Spitzer Data

All Spitzer data are downloaded from the Spitzer Heritage Archive . You will be given a choice of data products to download. MOPEX is designed to use the Level 1 (Basic Calibrated Data; BCD) products as input, but the Level 2 (Post-BCD; PBCD) products are useful for quick looks. In general, the user should begin science reductions with the BCD products. Once you have downloaded your data, unpack the zip files into a data directory (<*data_dir*>) of your choice.

1.7 Quickstart Guide

To create a quick mosaic of Spitzer data using MOPEX, follow the steps below. **This will only create a mosaic using a generic template, and may not be optimum for your data.**

1. Start the GUI as described above;

2. Load a template namelist from the File menu by clicking on *File > New Mosaic Pipeline;*

3. Select from the drop-down menu, e.g. if you have IRAC channel 1 data, select *Mosaic, IRAC ch1*;

4. Create lists of your input files on the command line, e.g.:

   ```
   unix% ls /<data_dir>/ch1/bcd/*bcd.fits > InputImageList.txt
   unix% ls /<data_dir>/ch1/bcd/*bunc.fits > InputUncList.txt
   unix% ls /<data_dir>/ch1/bcd/*bimsk.fits > InputMskList.txt
   ```

   (see Chapter 3 for more information on MOPEX input files);

5. Add the newly-created input list files into MOPEX. In the Initial Settings module:

   ```
   Image Stack File = InputImageList.txt
   Output Directory = /your/chosen/<working_dir>
   ```

   And under Optional Input and Mask files:

```
Sigma List File = InputUncList.txt
DCE Status Mask File = InputMskList.txt
```

6. Finally, run the mosaic creation by clicking on the green "play" arrow in the top left hand corner of MOPEX. Click OK to any warnings that appear and wait for the Mosaic pipeline to run through to the end.

7. You can view your output mosaic within MOPEX by clicking on the "View" button in the Mosaic Combiner module, next to "Mosaic image file" or you can open the mosaic in any viewer of your choice by going to the mosaic FITS file, saved as:

   <working_dir>/Combine-mosaic/mosaic.fits

Chapter 2. Overview of MOPEX

MOPEX was developed at the Spitzer Science Center to process Spitzer Level 1 Basic Calibrated Data (BCD) from the point at which the users download the files from the archive through to final point source extraction. While it is primarily designed to take Spitzer data as input, it is a general-purpose tool that can be applied to any images in standard FITS format (see §3.1 for MOPEX input image requirements).

2.1 How MOPEX Works

The MOPEX software package is a set of Perl wrapper scripts that call C++ and C modules. For the GUI, the Perl scripts were translated into Java. All of the Perl scripts can be found in the MOPEX installation directory, under */<mopex_dir>/bin/*.

These scripts are incorporated into the GUI as pipeline flows, and each pipeline calls a series of individual modules in sequence. Depending on the options you wish to use when processing your data, you will need to include different combinations of modules in a pipeline. For example, if you want to use one of the outlier rejection algorithms to remove the cosmic rays from your final mosaic, you need to set up the Mosaic flow to include the outlier rejection modules. Setups that you choose for a pipeline can be saved as a parameter file called a "namelist". This namelist can be read back into the GUI at a later date. Default namelists are also provided.

2.2 How to run MOPEX in the GUI

When you first start the MOPEX GUI, you are presented with a blank screen. To begin using MOPEX, you must initiate a pipeline from the File Menu. You can choose to initiate a pipeline in three different ways - you can read in a previously-saved parameter file (called a "namelist"; *File > Read Namelist*), you can start from a template of default modules and parameters (e.g. *File > New Mosaic Pipeline*), or you can choose to begin with a blank pipeline and add modules one at a time by hand (*File > New Empty Mopex Pipeline*). First time users are advised to begin with the template pipelines. Each module has a help page that explains what it does, what the inputs mean, and what outputs can be expected. The help page is accessed by the "?" button, or by reading the module descriptions contained in this manual.

Usually you will start by loading a template namelist as your starting point. We will assume for the moment that we have IRAC channel 1 data that we wish to background-match and mosaic

together. Background matching is performed by the Overlap pipeline, and mosaicking is carried out by Mosaic. One key feature of the GUI is that it can understand that we want to use the output of the first as input to the second.

Start by loading a template Overlap namelist by going to *File > New Overlap Pipeline* and selecting "Overlap, IRAC ch1" from the drop-down menu. Click OK, and the template Overlap pipeline is loaded. The current pipeline is shown in the main window, with clickable buttons to add input and output files, and input fields to change the settings for each included module. Off to the side is a separate window, listing all of the available modules for the Overlap pipeline. The ones that have already been included are greyed-out, while the ones that are currently excluded are in black. Modules can be added to the flow by clicking on them in this secondary window, and removed from the flow by clicking on the cross next to them in the main pipeline window. For assistance with any module, you can click on the question mark to pull up the integrated help system.

Once you have added the input files (see Chapter 3 for input file types and formats) and set up the Overlap pipeline, you need to add the Mosaic pipeline onto the end. To do this, click on the "Insert Mosaic…" button near the top of the pipeline window. Do not go through the *File* menu - this will start a whole new Mosaic flow instead of appending it to the bottom of the Overlap pipeline. By inserting the Mosaic flow, MOPEX will use the output of the Overlap pipeline (i.e. the background-corrected images) as the input into the Mosaic pipeline.

After setting up the Mosaic pipeline, you are ready to run MOPEX. There are three ways to do this:

1. Module-by-module: click on the icon with the black arrow and the blue rectangles on the left side of each module. This runs the reduction to the end of that module and then pauses.
2. By Pipeline: click on the icon with the black arrow and the blue rectangles near the top of each pipeline. This will run through the associated pipeline (e.g. Overlap) and pause at the bottom.
3. Run the whole flow: Click on the green "play" arrow at the top left-hand corner of the MOPEX window. This will run through all of the pipelines to the end.

Once you have started the flow running, you can pause at the end of the current module by pressing the black "pause" button in the top left-hand corner of the MOPEX window, or you can stop the reduction immediately by pressing the red "stop" button next to it. If you want to rewind back up to a previous module, click the blue curved arrow icon next to the module you wish to repeat. This will rewind the flow to that point so that you can, e.g., change parameters in that one module and re-run from that point onwards.

For instructions on how to run MOPEX on the command line, see Chapter 9.

NEW Note that with version 18.5.0 one can use multiprocessing to speed up MOPEX tasks. The tasks Overlap, Mosaic, Apex (Multi) and Apex User List (Multi) now allow multiprocessing of some steps. In the GUI, this is set in Initial Setup, Multi-Processing Mode. The options are "on", "off", and "manual". The default is "on" -- this grabs 3/4 of the available processors. Setting "manual" lets the user set the number of processors used. You should see speed-ups of at least 2x.

If using command-line, add these lines at the top of your namelist:
do_multiprocess = on | off | manual default = off
ncpu_multiprocess = 1 default = 1
If "manual" is set, set ncpu_multiprocess to the number of processors to use.

2.2.1 Encountering Error Messages in MOPEX

If MOPEX encounters any problems during the reduction, it will stop and open an error message window. If the explanation in the error message in unclear, you can check the log file found in *~/.spot/* for further information. If you are still stuck and have checked the manual, the MOPEX Bug List and the FAQ, please contact the Spitzer Helpdesk for assistance.

2.3 Basic Processing Steps in MOPEX

Here we outline the steps in a typical MOPEX task: combining a stack of FITS images into a mosaic and doing point-source photometry, then looking at the residuals. Spitzer Data Analysis Cookbook: Recipe 7 shows how to link all these pipelines together.

1. **Data Input**
   MOPEX reads in the data frames, the associated uncertainty frames, and the associated bad pixel masks from user-created lists in ASCII format.

2. **Fiducial Image Frame (FIF) Generation**
   A unified grid coordinate system is generated, which defines a common grid onto which the input images will be projected. The spatial boundaries of the FIF include all of the input images.

3. **Background Matching**

The Overlap pipeline performs background matching between overlapping input frames (see Chapter 4). An additive correction is calculated for each image in the input stack in order to bring them to a common background level (not a zero-background level - see Median Filtering below), so avoiding "patchy" mosaics. This process does not affect the photometry of the detected sources.

4. **Image Interpolation**
   The image interpolation module projects the input images onto the FIF and interpolates the input pixel values to the output array. Users can define the pixel size of the output array. This step corrects for the optical distortion in the input images.

5. **Outlier Detection and RMask Generation**
   A primary feature of MOPEX is outlier rejection (see §8.2). Four methods are available to use both spatial and temporal information to identify and mask outliers (e.g. cosmic rays). The combined outlier information can be stored in a single outlier mask -- the RMask -- which is then used by MOPEX to mask outliers when creating the mosaic.

6. **Mosaic Creation**
   The interpolated images are averaged to produce a co-added mosaic image (see Chapter 5). There are several weighting schemes available.

7. **Median Filtering**
   Mosaic images can be median-filtered to produce a background-subtracted image.

8. **Point Source Extraction**
   The APEX pipeline performs multi-frame (APEX Multiframe) and single-frame (APEX Single Frame) point source extraction (see Chapter 6). These scripts include non-linear matched filtering for point-source detection, image segmentation for separating blends, point-source fitting for position and flux estimation, and aperture photometry. Point sources are fit with a Point Response Function (PRF; see §8.7).

9. **Point Source Subtraction**
   The APEX_QA pipeline performs multi-frame and single-frame point source subtraction (it can also insert sources on to a frame).

A reduction such as this will result in the following output files: a single mosaicked image of all the input images; a coverage mosaic showing how many frames were combined to produce each pixel in the output mosaic; an uncertainty mosaic showing the uncertainties at each point in the mosaic, and a table of extracted flux densities for every point source, and a mosaic of the residuals after point-source subtraction. These are just the most commonly-used output products - you have the option to request the generation of others.

2.3.1 Additional Functionality

While the reduction process described above includes the most commonly used processing stages, there are many more functions available in MOPEX to carry out a more specialized reduction. See both §2.4 and Appendix A for more details.

2.3.2 Expected Input

MOPEX only requires a list of FITS images for basic mosaicking, but the additional files listed below will generally prove helpful:

- A set of FITS data images;
- Associated uncertainty images;
- Associated status masks, flagging bad pixels in each individual data frame (in Spitzer data, these are called DCE Status Masks, where DCE = Data Collection Event);
- Relevant masks of permanent detector artifacts (in Spitzer data, these are called PMasks).

For Spitzer data, all of the masks and calibration files can be downloaded from the Spitzer archive with your data. Permanently-damaged-pixel masks (PMasks) and PRF files are also stored in the *mopex/cal/* directory that comes with your MOPEX installation. Template namelists are accessible through the File menu in the GUI and from the subdirectory *mopex/cdf/*. See Chapter 3 for more information about the format of MOPEX input files.

MOPEX is set up for input data in units of surface brightness. It can recognize two units in the FITS header keyword BUNIT: MJy/sr or microJy/arcsec^2 (BUNIT is a text string and there are alternate ways of writing these units, but MOPEX will recognize several variants). If it recognizes the units, output mosaic will be in those units, and extracted source flux densities will be in **microJy**. If it doesn't recognize the units, all data will be treated as dimensionless "counts".

2.4 Introducing the Major Pipelines in MOPEX

The major pipelines available in MOPEX are as follows:

Overlap: Performs additive background matching of the input BCD images to bring them to a common background level and thereby avoid "patchy" mosaics.

Mosaic: Interpolates the input images onto a common grid, carries out outlier detection and masking, co-adds the images together and stitches them all into a single mosaic.

APEX Multiframe: Carries out point-source extraction on a stack of images. Optionally builds a mosaic along the way, and in any case, does detections on the mosaic, extractions on the stack. A PRF function (or an array-position dependent map) for the image stack is typically provided, and is required for point-source fitting photometry.

APEX Single Frame: Carries out point source extraction on a single image, e.g. a mosaic image. A PRF function (or an array-position dependent map) for the single image typically provided, and is required for point-source fitting photometry.

APEX User List Allows users to give MOPEX a list of sources to extract, rather than allowing MOPEX to do the extraction on all detected sources in the field. This is available in two flavors - Multiframe and Single Frame.

APEX QA: Creates residual images for both APEX Multiframe and APEX Single Frame by subtracting the detected point sources from the input images. This pipeline is invaluable for checking the results of PRF-fitting.

PRF Estimate: Allows user to generate their own PRF from their dataset. **Warning:** Only for use with well-sampled data that does not display any intra-pixel variability. For this reason, this script should not be used with IRAC channel 1 and 2 data.

2.4.1 Ancillary Scripts

While the seven major scripts above are the most commonly used, there are a few more that are available on the command-line. A full list of all of the MOPEX scripts can be found in Appendix A, but the most commonly used ones are:

flatfield.pl: Creates a "flat-field" image by taking the median of the image stack. Optionally divides the stack by the flat-field. Allows bright objects to be detected and masked beforehand. See the file *<mopex_dir>/readme/README_flatfield* for documentation.

mosaic_mask.pl: Allows making of a mosaic "OR"-mask, and optionally a mask cube. See http://irsa.ipac.caltech.edu/data/SPITZER/docs/files/spitzer/mosaic_mask.pdf for documentation. An example namelist is in cdf/mosaic_mask_I1_example.nl.

mosaic_sed.pl: MIPS SED data reduction. This script performs interpolation and coaddition of MIPS SED (spectral energy distribution) images, and extracts the spectra. See http://irsa.ipac.caltech.edu/data/SPITZER/docs/files/spitzer/mosaic_sed.pdf for documentation.

2.5 Which Modules Should I Choose?

When running MOPEX, each data reduction step should be tailored to the dataset. Users do this by selecting certain modules to run, and setting the input parameters for those modules. The number of modules and parameters in MOPEX can be a little daunting, so here we give a short introduction to the most-used modules. Detailed descriptions of the modules can be found later in the manual.

First-time users should start with the template namelists found in the File Menu, rather than trying to build up the namelist in an empty pipeline. These templates are intended only as guides, and will need fine-tuning for use with your data. With experience, you will be able to include or exclude modules as needed and tune the input parameters for specific purposes. MOPEX typically saves the results from each module in <*output_dir*>. These intermediate results can be examined to evaluate the performance at each step.

For data reduction walk-throughs, including more advice on options to use with different data sets, see the Spitzer Data Analysis Cookbook. We stress that these are not exhaustive, and are continually being developed.

2.5.1 Overlap (overlap.pl)

The Overlap pipeline (run on the command line with the script overlap.pl) performs background matching (§8.5) on the input images, and is fully described in Chapter 4. It does not set the background level to zero, rather it adds (or subtracts) a constant to bring it to an average level. This process is recommended for Spitzer data, as there can be variation in the sky level that will lead to a patchy-looking final mosaic if not corrected, but this step may not be necessary if the BCDs downloaded from the Spitzer archive have already been background-subtracted, e.g. by MOPEX's "Median" filtering.

The modules required for Overlap are:

- Fiducial Image Frame (unless the file *FIF.tbl* already exists; §4.3.4)
- Mosaic Interpolate (§4.3.7)
- Compute Overlap Correction (§4.3.8)

If there are bright objects in the field that could affect the estimate of the background level, MedFilter (§4.3.5) and Detect (§4.3.6) should be added and the *Mask Bright Objects* option in the Mosaic Interpolate module should be checked. Together, these will detect and ignore bright objects when calculating background levels.

Make sure to check the "Apply Overlap Correction" box in the Overlap Correction module to apply the correction to your input images. Overlap-corrected images can be used as the input to the Mosaic pipeline.

2.5.2 Mosaic (mosaic.pl)

The Mosaic pipeline (Chapter 5) has the largest number of modules to choose from, due to the many outlier detection options. While some outliers are flagged during the Spitzer pipeline processing (that produced the BCDs that you download from the archive), further rejection is recommended.

The basic mosaicking modules that you need, not including outlier rejection, are:

- Fiducial Image Frame (§5.6.4)
- Mosaic Interpolate (§5.6.8)
- Mosaic CoAdder (§5.6.19)
- Mosaic Combine (§5.6.20)

If any outlier rejection is employed, MOPEX will create a new status mask, the RMask, and use it to reinterpolate the rejected pixels. To use the RMask capability, you will need to add at least the following modules, plus the modules specific to the chosen outlier rejection scheme (see following discussion):

- Mosaic Coverage (§5.6.11)
- Mosaic RMask (§5.6.16)
- Mosaic Reinterpolate (§5.6.17)

Choosing the right kind of outlier rejection (§8.2) also requires some care. Rejection schemes utilize either temporal information (rejecting sources that do not appear at the same coordinates in every frame) or spatial information (rejecting outliers based on their shape or size), or both. In the case of good coverage per pixel (about 10 or more BCDs per pixel on the sky), multiframe temporal outlier rejection (§8.2.2) can be a good choice. To add multiframe temporal outlier rejection, use the Mosaic Outlier module (§5.6.14). When using multiframe temporal outlier rejection, be careful when setting the rejection thresholds. A three-sigma rejection is often too low if the coverage is very high (50 or more). If only shallow coverage is available, then either the box outlier rejection (§8.2.4) (include module Mosaic Box Outlier, §5.6.15) or dual spatial-temporal outlier rejection (§8.2.3) are preferred. Dual outlier rejection requires the following modules:

- MedFilter (§5.6.6)
- Detect (§5.6.9)
- Mosaic Projection (§5.6.10)
- Mosaic Dual Outlier (§5.6.12)
- Level (§5.6.13)

Multiple outlier rejection methods can be employed, and all can be set to contribute to the RMask by setting the *RMask Fatal Bit Pattern* (see §8.11: Fatal Mask Bit Patterns). The Mosaic RMask module (§5.6.16) can also use the dual outlier rejection results as a check on the temporal outlier identifications (the *Refine Outlier* option). This combination can be useful to prevent the false identification of temporal outliers inside bright sources.

The Mosaic pipeline can produce a number of final outputs. The most basic output files that you will need are the mosaic and its coverage map and uncertainty file (*mosaic.fits*, *mosaic_cov.fits*, and *mosaic_unc.fits*). For more options, see §5.5 "Mosaic Pipeline Stages."

2.5.3 APEX Single Frame (apex_1frame.pl)

Source extraction can be performed on the mosaic image using APEX Single Frame mode (Chapter 6). This takes a single image (together with an optional coverage map and optional uncertainty file) as input. It can be used independently or as a follow-on to the Mosaic pipeline.

A typical APEX Single Frame task might include background subtraction, noise estimation, non-linear filtering, point source detection, point source fitting and flux estimation, and aperture photometry. These are carried out by the following modules:

- Detect MedFilter (§6.5.8)
- Point Source Probability (§6.5.9)

- Gaussnoise (§6.5.10)
- Detect (§6.5.12)
- Extract MedFilter (§6.5.14)
- Source Estimate (§6.5.18)
- Aperture Photometry (§6.5.19)

A commonly used but optional task is Fit Radius. If an uncertainty image is available, the Fit Radius module can be run to define the fitting area based on source brightness above the noise. Setting the *Fitting Area X,Y* parameters in the Source Estimate block is the other method (and this will override the use of the Fit Radius results).

The output of APEX is the table of extracted sources (*extract.tbl*), which gives the position and flux for each object. Compared to the Mosaic pipeline, there are fewer module choices in APEX Single Frame, so it is a little easier to use.

2.5.4 APEX Multiframe (apex.pl)

Source extraction can also be performed in Multiframe mode. In this case, sources are detected in the mosaic but PRF-fitting (§8.6) is performed on the individual BCDs (after geometric distortion correction); aperture photometry is still performed on the mosaic. APEX Multiframe uses all of the same modules as the single frame mode, but can also optionally create the mosaic as well, by including the following modules:

- Fiducial Image Frame (§6.5.5)
- Mosaic Interpolate (§6.5.6)
- Mosaic CoAdder (§6.5.7)
- Mosaic Combine (§6.5.13)

The downside to having APEX create the mosaic is that it cannot do outlier rejection. Most users prefer to make their own mosaic beforehand with the Mosaic pipeline. To use the Mosaic output as the input into APEX, you need to insert the APEX Multiframe pipeline into the flow, and turn off the four modules listed above. APEX will automatically pick up the files it needs from the Mosaic output directories.

2.5.5 APEX User List (apex_user_list.pl; apex_user_list_1frame.pl)

APEX User List (§6.1.3; §6.1.4) gives users the option to extract a specified list of sources, rather than extracting all sources that were detected in the image(s). The sources are input as an ASCII list of positions in either RA and Dec, or pixel coordinates, and there is the option to either allow centroiding of the source positions during the source extraction (both PRF fitting and aperture

photometry), or to fix the positions (aperture extraction only). MOPEX also enables interactive selection of sources within the GUI window, using a point-and-click interface. Source extraction is performed in exactly the same way as for APEX Multiframe and APEX Single Frame, using the user-supplied list of sources as input instead of the source list generated by the Detect module. If you are running Single Frame mode, you only require the following:

- Extract MedFilter (§6.5.14)
- Source Estimate or Aperture Photometry (§6.5.18; §6.5.19).

If you are running Multiframe mode and creating your own mosaic, you will need the following (but see previous section for the advantage of using the results of Mosaic as input to APEX Multiframe):

- Fiducial Image Frame (§6.5.5)
- Mosaic Interpolate (§6.5.6)
- Mosaic CoAdder (§6.5.7)
- Mosaic Combiner (§6.5.13)
- Extract MedFilter (§6.5.14)
- Detection Map (§6.5.17)
- Source Estimate or Aperture Photometry (§6.5.18; §6.5.19).

2.5.6 APEX QA: Residual Image Creation (apex_qa.pl)

The APEX Quality Assurance (APEX QA) pipeline (§6.1.6) performs subtraction of the detected point sources from the input data and produces a residual image mosaic.

In Single Frame mode, the PRF is subtracted from the input images or mosaic, depending on whether it is being run in conjunction with APEX Multiframe or APEX Single Frame. Subtraction is performed by the Point Source Image module. In Multiframe mode, APEX QA can create a new mosaic from the individual residual images using the Mosaic Interpolate, Mosaic CoAdder, and Mosaic Combiner modules. There is only one module required for APEX QA, although its name on the command line changes depending on whether it is being run in Multiframe or Single Frame mode:

- (Mosaic) Point Source Image (§6.6.4)

2.5.7 PRF Estimate (prf_estimate.pl)

PRF Estimate (Chapter 7) allows users to create a Point Response Function (PRF) from their own data. However, it is not designed to work for severely undersampled data, such as IRAC channels 1 or 2. So PRFs are made available for use with IRAC data. The following modules are required:

- MedFilter (§7.5.3)
- Crop Stack (§7.5.4)
- PRF Estimate (§7.5.6)

You may also choose to generate a PRF Map, thereby creating a different PRF for different areas on the array. To do this you should also include the Split By Array Position module (§7.5.5).

Chapter 3. MOPEX Input

MOPEX requires three main input types in order to carry out any processsing. The following section provides more information about each of the input types:

1. Input FITS images. These include the data images, the associated mask and uncertainty files and any calibration files.

2. Text files listing the input images. These should be in ASCII format.

3. The namelist. This is a parameter file specifying all of the input parameters to be used by MOPEX. This is usually built up from within the GUI, either from an empty flow, or by modifying existing templates, but you can also read in a previously-saved namelist from an ASCII file.

3.1 MOPEX Input Images

Images input into MOPEX do not have to be Spitzer data, but they do have to be in FITS format, with the standard FITS and WCS keywords. This software cannot 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 CDELT1, CDELT2, CROTA2 should be also present.

All of the projections in the WCS library are implemented, including "SIN", "TAN", "ZEA" (Lambert's zenith equal-area), "ARC" (zenith equidistant), and "STR" (stereographic). "TAN" is the default.

There are no practical restrictions on the size of the area 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).

3.1.1 Spitzer Image Files

When you download your Spitzer data from the archive, you will find that there are many different files that are associated with each observation. Moreover, the filenames differ depending on the instruments. There are three types of image file that are commonly used as input into

MOPEX: Basic Calibrated Data (BCD) images, the associated uncertainty image, and the associated bad pixel mask. See Table 3.1 for the recommended filenames for each instrument.

3.1.2 Additional Mask Files

In addition to the Status Mask Files that come with your Spitzer data (which flag the temporary bad pixels in each frame) there are two other types of mask file that can be input into MOPEX. The first is the mask of permanently damaged pixels (PMask). The PMasks for the IRAC instrument are updated every few months, depending on solar activity. They can be found in the *<mopex_dir>/cal* directory, or on the Spitzer website. The applicable PMask depends on the observation date of your data - check your FITS headers for the keyword DATE_OBS (not DATE, which is the date the file was written). See the *pmasks.README* file in the PMasks directory for the date range for which each PMask should be applied. PMasks for the MIPS instrument are updated less often.

The final set of images that can be input into MOPEX are RMask files, which are masks of pixels rejected as outliers in each frame. Typically, the user will generate RMasks as a product of outlier rejection during mosaicking (see §8.2), so they are not provided as input. However, it is possible to use RMasks as input, either from an earlier reduction, or from the automated Spitzer pipelines (which use MOPEX). They can also be specified as input into APEX Multiframe to ensure that outlier-rejected pixels are not used when running the PRF fitting on the individual BCDs.

Table 3.1: BCD data, uncertainty and mask filenames for each instrument

Instrument	Default BCD	BCD Uncertainties	DCE Status Mask
IRAC	bcd.fits or cbcd.fits	bunc.fits or cbunc.fits	bimsk.fits
IRS PU-16	bcdb.fits	uncb.fits	mskb.fits
IRS PU-22	bcdr.fits	uncr.fits	mskr.fits
MIPS-24	bcd.fits	bunc.fits	bbmsk.fits
MIPS-Ge (70 & 160)	bcd.fits or fbcd.fits	bunc.fits	bmask.fits

3.2 MOPEX Input List Files

The data, uncertainty, status mask and, possibly, the outlier mask image files are input into MOPEX via ASCII text files, containing one image filename per line. You will need to make a separate list file for each of the input image types, and the filenames must be in the same order in each of the input lists (i.e. the first filename in the list of image files must correspond to the first filename in the list of uncertainty files and the list of status mask files). As a rule, we recommend always including the full path name of the image files in the input lists e.g., the list of data files, *InputImageList.txt*, should look something like this:

```
/home/joe/data/IRAC/ch1/bcd/SPITZER_I1_5504256_0001_0000_4_bcd.fits
/home/joe/data/IRAC/ch1/bcd/SPITZER_I1_5504256_0002_0000_4_bcd.fits
/home/joe/data/IRAC/ch1/bcd/SPITZER_I1_5504256_0003_0000_4_bcd.fits
```

etc.....

As an example, to create the input lists for IRAC channel 1 data, in Unix (Solaris/Linux/Mac), you could use the following syntax:

```
unix% ls /<data_dir>/*bcd.fits > InputImageList.txt
unix% ls /<data_dir>/*bunc.fits > InputUncList.txt
unix% ls /<data_dir>/*bimsk.fits > InputMskList.txt
```

The input lists are specified in the Initial Setup module. The input file format is identical for both the command-line and the GUI interfaces.

The default location for the input files is in <*working_dir*>. When specifying input files, you should use either the full path name or the path relative to the working directory.

For more information on the User List point source input file for use with APEX User List and APEX User List Single Frame, see §8.10

3.3 MOPEX Input Namelist

The namelist files for each of the MOPEX pipelines contain all of the information about the modules and parameters to be used in a particular reduction. The ASCII versions of the template namelists used by MOPEX are all stored in the directory <*mopex_dir*>*/cdf/*. When using the GUI, you will most likely start from one of the included MOPEX template namelists (go to e.g. *File > New Mosaic Pipeline* to access the templates), and edit the included modules and

parameters before running your reduction. Alternatively, you can choose to build up your namelist from an empty pipeline by going to *File > New Empty MOPEX Pipeline*.

Namelists can be written out to an ASCII file to save them for a later GUI reduction (*File > Save As*), or exported to command-line format to run them on the command line (*File > Export Inner Pipeline*), and namelists that have been set up on the command line (see Chapter 9) can be imported into the GUI and run there (*File > Read Name List;* this is valid for reading in both command-line and GUI namelists).

The names of some of the parameters on the command line have been changed in the GUI to improve clarity. You can find a full list of all the input parameters for the GUI, along with the command-line counterparts, in the MOPEX glossary pages, linked from the MOPEX webpage. For more information on the structure of the saved ASCII file namelist and running MOPEX on the command line, see Chapter 9.

Chapter 4. Overlap (*overlap.pl*)

The Overlap pipeline in MOPEX (on the command line, the script is *overlap.pl*) performs background matching between overlapping frames. An additive correction is calculated for each image in the input stack in order to bring them to a common background level. This is often the first step in processing Spitzer imaging data.

4.1 Overview

To load an Overlap pipeline, either start from a MOPEX template namelist (*File > New Overlap Pipeline*), load an existing Overlap namelist file (*File > Read Name List*) or load an empty flow and insert an Overlap pipeline from there (*File > New Empty MOPEX Pipeline*, then go to *Insert Overlap* at the top of the flow window and choose *Empty Pipeline*). MOPEX will load an Overlap flow to which you can add and subtract modules, and modify the input parameters. For information on running Overlap on the command line, see Chapter 9.

Overlap begins with a text list of input images (for Spitzer data, these are usually the Level 1 BCDs). This is loaded in *Image Stack File,* in the Initial Setup module. Uncertainty and mask files are loaded in *Optional Input & Mask Files.* The final output is a table of overlap correction factorsand a stack of offset-corrected images. If desired, a QuickLook mosaic can be created, incorporating the corrections.

Optionally, the user can run an additional preprocessing step in order to detect and mask all bright objects in the input images that might adversely affect the process of background matching.

If the user has no uncertainty files, Overlap can run a module called S/N Estimator, which estimates the uncertainties from the gain and read noise. The user should carefully read §4.3.3 to do this correctly. For Spitzer data, the pipeline uncertainties are generally preferred.

Overlap finds the additive offsets for each frame so that the pixel differences in the overlap regions (after interpolation to a common grid) are minimized. The metric that is minimized is the sum of the uncertainty-weighted squared differences between overlapping pixels in all pairs of interpolated input images, however, this does not determine the overall level that the frames are corrected to.

In the GUI, MOPEX determines the overall level after finding the offsets, by forcing the sum of the offsets of all the images to add up to 0 (see §8.5 for a full description of the algorithm). The

offsets are analyzed before applying the above condition. The outliers are found using the input parameters in the Overlap Correction module: *BOTTOM_THRESHOLD* and *TOP_THRESHOLD* in units of sigma, and *MIN_IMG_NUM*, the minimum number of images needed to detect outliers. This method should give good results for large numbers of frames without strong systematic deviations.

There is also a second method available on the command line, triggered by setting the parameter *WEIGHT* in the Overlap Correction module. In this method (nominally *WEIGHT = 1*), MOPEX minimizes the squared offsets as part of the original minimization. The user can experiment by sliding *WEIGHT* up or down, giving more or less weight to the squared offsets (*WEIGHT = 0* corresponds to the default method employed by the GUI). In principle, this method should allow correction for offsets even for a small number of frames with a varying background, but the range of effectiveness has not been thoroughly tested.

The methods are described in §8.5, with more information in the online document http://irsa.ipac.caltech.edu/data/SPITZER/docs/files/spitzer/background_matching.pdf, and the first is given in more detail in *Makovoz et al. 2005, PASP 117, 274-280*.

Overlap has a fault in that it finds additive corrections to all frames in one swoop. For a large number of frames, this leads to a big matrix inversion, requiring a big chunk of memory. If the memory limit is reached, MOPEX fails, usually with either a MALLOC error or no error at all. If you believe you are encountering this problem (usually seen with datasets of a few thousand input frames) then you can try using the Contributed Software IDL routine called *afrl_bcd_overlap (*http://irsa.ipac.caltech.edu/data/SPITZER/docs/dataanalysistools/tools/contributed/irac/afrlbcdoverlap/)

With version 10.5.0 there is a new option which inserts a sparse matrix solver to increase the limit on the number of files. In the GUI this is in the Overlap Correction Settings. In command-line namelists it is set in the Overlap Correction block: USE_SPARSE_FOR_MANY_FILES = 1,. You should be able to do at least 10,000 images in most cases.

Other common problems encountered with Overlap are:

1. Inclusion of wildly discrepant frames, e.g. MIPS-70 or MIPS-160 stim flashes, or IRAC frames affected by "droop" from very bright sources, will give the algorithm trouble, and may cause Overlap to fail with the error message "`UNSUCCESSFUL_SOLUTION`". Make sure you have removed all of the stim flash frames, and checked for any other large background-level discrepancies before running MOPEX.

2. Systematic offsets with heavy overlap, e.g. at the start of MIPS-24 scan legs, will not be corrected.

3. Background matching does not perform well in cases where data are divided evenly into two background levels. An example of this would be AORs taken in different epochs such that the zodiacal background has changed. In such cases, background can be brought into closer agreement by subtracting the estimated zodiacal background (check the *Apply Zodiacal Subtraction* box in the Overlap Settings module). The zodiacal background estimate is taken from the ZODI_EST keyword in the IRAC and MIPS FITS headers.

4.2 Overlap Processing Stages

The main processing stages of the Overlap pipeline are as follows:

Bright Object Masking (optional)

A preprocessing step can be run to mask bright objects that can bias the background levels of the images in which they are present. Two modules are run: MedFilter and Detect. The former produces background-subtracted images, which are then input into Detect. The latter detects bright objects in the background-subtracted images and creates a set of mask images, one per input image, in which pixels corresponding to the detected objects (or radhits) are set to positive values. The background-subtracted images created by MedFilter are only used for bright object detection, not for the actual overlap correction.

Interpolation

Before background matching, the input images are interpolated onto the output grid defined by the Fiducial Image Frame (FIF) table. If no table exists, it needs to be created by the Fiducial Image Frame module. If bright object masking is selected, masked pixels are not used in the interpolation.

Background Matching

Once the images have been interpolated to a common grid, the metric to be minimized is the combined, uncertainty weighted difference between the overlapping parts of each pair of input images. It is minimized simultaneously with respect to the constant offsets in question. More information on the algorithm used can be found in §8.5: Background Matching Algorithm.

Quicklook Mosaic

It is possible to coadd the interpolated, corrected frames into a QuickLook mosaic. This is a quick-and-dirty way to examine the results of overlap correction, and will most likely produce an ugly-looking mosaic with a lot of masked areas, where Bright Object Masking has been applied. Run the Mosaic pipeline to create a science-quality mosaic.

4.3 Overlap Modules

The following pages contain full descriptions of the modules that are available in the GUI Overlap pipeline. For information on how to turn these modules on and off on the command line, see Chapter 9. For suggestions on which modules to use, see §2.5: Which Modules Should I Choose?

Module listing:

- Initial Setup
- Overlap Settings
- S/N Estimator
- Fiducial Image Frame
- MedFilter
- Detect
- Mosaic Interpolate
- Compute Overlap Correction
- Quick Look Mosaic

4.3.1 Overlap Modules: Initial Setup

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: None

Important Notes: None

PURPOSE

Set up the input and output files for the loaded flow.

INPUT

Image Stack File: The text file containing the list of input data files for the pipeline. For Spitzer data, these are typically the **bcd.fits* or **cbcd.fits* files that you download from the Spitzer data archive. See Chapter 3 and Table 3.1 for more information on the input file requirements and format.

Output Directory: The directory that you want to set as your output directory for this run.

Multi-processing Mode: Switch to use multi-processing. See Discussion.

Processors for Multi-processing: Number of processors to use. See Discussion.

Optional Input and Mask Files:

Sigma List File: The text file containing the list of uncertainty images that correspond to the input data files (for Spitzer data, these are usually the **bunc.fits* or **cbunc.fits* files; see Table 3.1 for more information).

DCE Status Mask List: The text file containing the list of mask images that flag temporarily-bad pixels in the input data images. There should be a mask file for each of the input data files (for Spitzer data these depend on the instrument, but will be labeled something like **msk.fits*; see Table 3.1 for more information).

Fatal Mask Bit Pattern: This is the bit pattern that defines which type of flagged problems that you wish to set as fatal when combining the input images. See §8.11 for more information. There is a separate Fatal Bit Pattern for each type of mask that is being used as input (i.e. for the DCE Status masks, the RMasks and the PMask).

Rmask List File: The text file containing a list of outlier mask images. Usually this will be left blank, as the RMask images are created during the mosaicking process. This option allows you to specify a previously created list of RMask images as input on second and subsequent runs of the pipeline, to save having to re-generate them.

Pmask FITS File: The permanently-damaged pixel mask for the detector. This is a single FITS image, flagging the permanently damaged pixels in the instrument arrays. These files are stored in the *<mopex_dir>/cal/* directory. See §3.1.2 for more information.

FIF file: The Fiducial Image Frame file. This is an optional input file, as it is often generated from the input data using the Fiducial Image Frame module. There are, however, a number of cases in which you might wish to use a user-specified FIF (e.g. if you are trying to match the field of view to the other data channels).

COMMAND LINE INPUT

These parameters are set in the Global Parameters part of the namelist.

```
IMAGE_STACK_FILE_NAME = <working_dir>/imagelist.txt
OUTPUT_DIR = <working_dir>/output
SIGMALIST_FILE_NAME = <working_dir>/sigmalist.txt
DCE_STATUS_MASK_LIST = <working_dir>/masklist.txt
DCE_Status_Mask_Fatal_BitPattern = 32544
RMASK_LIST = <working_dir>/rmasklist.txt
RMask_Fatal_BitPattern = 7
PMASK_FILE_NAME = <mopex_dir>/cal/sep07/sep07_ch1_bcd_pmask.fits
PMask_Fatal_BitPattern = 32767
FIF_FILE_NAME = <working_dir>/FIF.tbl
verbose = 1
NICE = 1
save_namelist = 1
```

Verbose, *NICE* and *save_namelist* are only available on the command line. See §9.4.2.3 for more information.

OUTPUT

None

DISCUSSION

This module sets up the input files and top-level output directory for the first pipeline in the flow. Subsequent pipelines that have been inserted into the flow will use the output from the previous one (i.e. if you have inserted Overlap and Mosaic into your reduction flow, Overlap will use the Initial Settings files as input, but Mosaic will use the modified files output by Overlap). Only *Input Image Stack* and *Output Directory* are required, but *Sigma File List*, *DCE Status Mask List* and *Pmask FITS File* are recommended.

NEW Note that with version 18.5.0 one can use multiprocessing to speed up MOPEX tasks. The tasks Overlap, Mosaic, Apex (Multi) and Apex User List (Multi) now allow multiprocessing of some steps. In the GUI, this is set in Initial Setup, Multi-Processing Mode. The options are "on", "off", and "manual". The default is "on" -- this grabs 3/4 of the available processors. Setting "manual" lets the user set the number of processors used. You should see speed-ups of at least 2x.

If using command-line, add these lines at the top of your namelist:
do_multiprocess = on | off | manual default = off

ncpu_multiprocess = 1 default = 1
If "manual" is set, set ncpu_multiprocess to the number of processors to use.

4.3.2 Overlap Modules: Overlap Settings

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: Initial Settings

Important Notes: None

PURPOSE

To set the input parameters specific to the Overlap pipeline.

INPUT

Pixel by Size: Check this to set the output pixel size in degrees. Some default output pixel sizes are given in the example templates.

Pixel Size X (deg): Set the X-size of the output pixels in degrees. The pixel size in the X-direction is quoted as a negative value to comply with convention. A positive value will generate an error message.
Pixel Size Y (deg): Set the Y-size of the output pixels in degrees.

Pixel By Ratio: Check this to set the output pixel size by ratio of the input pixel to the output pixel, i.e. to make the output pixels half the size of the input pixels (over-sample the mosaic), set the values of *Pixel Ratio X(Y) = 2*. The ratio is taken after correction for geometric distortion.

Pixel Ratio X: Set the pixel ratio in the X-direction.
Pixel Ratio Y: Set the pixel ratio in the Y-direction.

Delete Intermediate Files: Check this to delete all of the intermediate files created by the Overlap pipeline. This leaves only the input files and the files created by the final module. This is useful for saving disk space once you are satisfied with your reduction setup. Note that MOPEX will prompt you with a pop-up box at the end of the flow, to be sure that you meant to select this irreversible option.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.** Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

Apply Zodiacal Subtraction: Check this to subtract the Zodiacal light from the input BCDs before running the background matching. The estimated Zodiacal light level is taken from the FITS header keyword ZODY_EST. In the case of IRAC images, Overlap also reads the keyword SKYDRKZB, which is the estimate of Zodiacal light in the IRAC skydark. For IRAC images, the value subtracted is ZODY_EST - SKYDRKZB.

Save Zodiacal Subtracted Images: Check this to save the Zodiacal light-subtracted BCDs.

COMMAND LINE INPUT

The following parameters are set in the Global Parameters part of the namelist. Note that if both *MOSAIC_PIXEL_SIZE_X(Y)* and *MOSAIC_PIXEL_RATIO_X(Y)* are both set, the former takes precedence and will override the latter.

```
MOSAIC_PIXEL_SIZE_X = -0.00033889
MOSAIC_PIXEL_SIZE_Y = 0.00033889
MOSAIC_PIXEL_RATIO_X = 1
MOSAIC_PIXEL_RATIO_Y = 1
delete_intermediate_files = 0
USE_REFINED_POINTING = 0
apply_zodiacal_correction = 0
save_zodiacal_subtracted_images = 0
```

OUTPUT

If the *Save Zodiacal Subtracted Images* option is checked, the corrected files are saved in the *<output_dir>/Zod_Sub* directory.

DISCUSSION

Overlap Settings sets up all of the input options that are specific to the Overlap pipeline. Some of them may be repeated in the Mosaic Settings, so you should be aware that you may need to match some settings (e.g. *Pixel Ratio*) between the pipelines.

4.3.3 Overlap Modules: S/N Estimator

Command Line Equivalent: compute_uncertainties_internally
Default Output Directory: <output_dir>/Sigma-overlap
Depends On: Initial Settings; Overlap Settings

Important Notes: Only needed if you do not have uncertainty images (these are typically provided with all Spitzer data).

PURPOSE

This module is used to estimate the uncertainty images from instrument gain and read noise when no uncertainty image is available. Warning: these are for an idealized case, and real uncertainties should be used if available. If the wrong *Gain* or *Read Noise* parameters are specified, this module may produce uncertainties that are too large, and this will result in no outlier detection.

INPUT

Gain in e-/image unit: (float) This is the parameter used to translate the measured data counts to physical units. In the case of Spitzer data, the BCDs are in surface brightness units of MJy/sr. **This parameter has the same name as stored in the Spitzer BCD FITS header, but they have different units.** MOPEX *Gain* has the unit of e-/MJy/sr (or e-/microJy/arcsec^(2)) and the FITS header GAIN has the unit of e-/DN. *Gain* used by MOPEX can be computed from the GAIN parameter in the FITS header by using formula:

$$Gain(MOPEX) = \frac{EXPTIME * GAIN(header)}{FLUXCONV}$$

Equation 4.1

FLUXCONV is the flux conversion factor between DN/s and surface brightness unit of MJy/sr (or microJy/arcsec^(2)) and EXPTIME is in seconds. FLUXCONV can be found in the Spitzer BCD FITS image header.

Read Noise in e-: (float) This parameter characterizes the noise in e- when the data are being read out from detector. For Spitzer data, the read noise is given by the FITS header keyword RONOISE.

Confusion Sigma in e-: (float) The 1-sigma confusion noise limit. The source of this noise is from the spatially unresolved background galaxies. This information can be found in the IRAC and MIPS Instrument Handbooks.

S/N estimator output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Sigma-overlap.

COMMAND LINE INPUT

```
&SNESTIMATORIN
 Gain = 66.8,
 Read_Noise = 8.8,
 Confusion_Sigma = 0,
&END
```

In Global Parameters:

```
SIGMA_DIR = Sigma-overlap
```

OUTPUT

S/N FITS Files (_sigma.fits*):** The estimated uncertainty image corresponding to each input BCD image.

DISCUSSION

This module allows users to estimate the noise from the BCD data, if no uncertainty images are provided. Here you need to input gain, readout noise and confusion limit appropriate for the dataset. We suggest that users look at the BCD image header to find the gain and read out noise. This module should not be used unless you cannot use the Spitzer archive-provided uncertainty images.

The module estimates the pixel uncertainty for each pixel in the image using the following model:

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

Equation 4.2

where the parameters *Read Noise* $\sigma_{readnoise}$, *Gain* g, and *Confusion Sigma* $\sigma_{confusion}$ are specified in the module settings. 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 units of surface brightness (MJy/sr; i.e. all Spitzer BCDs) then the *Gain* should include the conversion factor from surface brightness units to electrons (see the *INPUT* for this module). The product of this step is the uncertainty images.

4.3.4 Overlap Modules: Fiducial Image Frame

Command Line Equivalent: run_fiducial_image_frame
Default Output Directory: <output_dir>
Depends On: Initial Settings; Overlap Settings

Important Notes: Running this module will overwrite any pre-existing *FIF.tbl* file in the output directory. **If you are working with MIPS images and plan to use the Mosaic PRFs then please see the Discussion below for important information.**

PURPOSE

This module creates a unified grid coordinate system and defines the spatial boundaries that will include all BCD images. The output FIF table is required for later modules to run. It can be generated by this module, or set as an input using a pre-existing table.

INPUT

Edge Padding: (int) This number specifies how much extra space (arcseconds) will be added to the four edges of the final mosaicked image.

Projection: This parameter specifies the projection type of the output mosaic. The TAN and SIN projections are implemented using the fast direct plane-to-plane coordinate transformation. The rest of the projections in the WCS library - LIN, AZP, STG, ARC, ZPN, ZEA, AIR, CYP, CAR, MER, CEA, COP, COD, COE, COO, BON, PCO, GLS, PAR, AIT, MOL, CSC, QSC, TSC, NCP, DSS, PLT, TNX - are implemented as a two step plane-sky-plane projection. The default projection is TAN.

Coordinate System: The coordinate system of the output mosaic. The choices are J2000 (Equatorial), Galactic, or Ecliptic. The default is J2000.

Use average input orientation: Defines the orientation of the output mosaic. Checking this box indicates that MOPEX should use the average of the input BCD orientations. This is the default setting. To specify an alternative orientation, see *CROTA2*.

CROTA2: (float) Defines the orientation (degrees East of North) of the output mosaic. *CROTA2 = 0* indicates North up and East left. To use the average of the input BCD orientations (the default), check the *Use average input orientation* box. On the command line, set *CROTA2 = A* to use the average orientation.

COMMAND LINE INPUT

```
&FIDUCIALIMAGEFRAMEIN
 Edge_Padding = 10,
 Projection_Type = "TAN",
 Coordinate_System = "J2000",
 CROTA2 = A,
&END
```

In Global Parameters:

```
MARGIN = 0
```

MARGIN: (int; command line only) Specifies a margin in terms of the number of pixels around the Fiducial Image Frame. Normally this is set to zero or a (small) positive number, except when running the Mosaic flow, using the Mosaic Geometry module, when it should normally be set to a negative number to avoid including input images that have only marginal overlap with the FIF.

OUTPUT

Generated Headers List Table (*header_list.tbl*): This file lists the World Coordinate Systems (WCS) information from each of the BCD image headers. The information is used to make the Fiducial Image Frame table.

Generated FIF Table (*FIF.tbl*): This text file is used by later modules for putting together a mosaicked image with specified pixel size, orientation and the final image size. The module overwrites any input FIF table specified in the Initial Settings.

The outputs of this module are written to the top output directory specified in the Initial Settings.

DISCUSSION

This module creates a table specifying a unified coordinate system and the spatial boundaries of the output mosaic. This table is required for any interpolation or mosaicking using MOPEX. If this module is not included, an existing FIF table has to be specified in the Initial Settings to enable later modules to run. After this module is run once, the output FIF table can be modified manually if desired, and used as input to subsequent runs (this is useful for e.g. mosaicking small sections of a large dataset. See the Mosaic pipeline for more details).

Even if the input images use the CD matrix convention, the output mosaic images (and by extension the interpolated images) will use the set of keywords CDELT1, CDELT2 and CROTA2, since the mosaic image is undistorted.

If you plan to use the MIPS "mosaic" PRFs that are provided for PRF fitting on your mosaic, your mosaic must be built in the same way as the ones used to derive those PRFs, that is, from BCDs that have nearly the same orientation (close in time or taken at the same time of year) and with a *CROTA2* angle that is properly aligned with the BCDs (*Use average input orientation* is recommended). All other cases would require you to make your own PRF from your mosaic.

4.3.5 Overlap Modules: MedFilter

Command Line Equivalent: run_medfilter
Default Output Directory: <output_dir>/Medfilter-overlap
Depends On: Initial Settings; Overlap Settings

PURPOSE

This module performs a background subtraction of the individual input images in order to set the frames up for Bright Object Detection. The output background-subtracted BCDs are taken as input by the Detect module, which carries out the detection of bright pixels in the frame. There are two options: the Median case (default) and the Sbkg case, which uses a background estimate like that of SExtractor.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

SExtractor background filter size: (int) Median filter box size for the Panels when the *Use SExtractor background estimation* option is turned on. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Use SExtractor background estimation: Checking this box invokes the Sbkg background estimate based on SExtractor. Leaving it unchecked causes MOPEX to use the default Median estimate.

Med Filter output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Medfilter-overlap.

COMMAND LINE INPUT

```
&MEDFILTER
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 50,
 Sbkg_Filt_Size_for_Med = 3,
 Use_Sbkg_for_Med = 1,
&END
```

In Global Parameters:

```
MEDFILTER_DIR = Medfilter-overlap
```

OUTPUT

Generated Fits Files (*_minback.fits*): The background subtracted, individual images.

DISCUSSION

This module is required if you plan to use Bright Object Masking, and carries out a median background subtraction before running the Detect module to detect the bright objects in the frame. The background-subtracted images are not used for later mosaicking.

With the default Median option, the program computes a background value using the median of a running rectangular window of *Window X* by *Window Y* pixels, after omitting the *Outliers / Window* highest pixels. This is done for each pixel so can be very slow.

If *Use SExtractor background estimation* is set, the module switches to the Sbkg background estimation based on that of SExtractor (*Bertin and Arnouts, AASupp 117, 393, 1996*). The image is divided into Panels with size given by *Window X, Y*. In each Panel, iterative clipping is used to find a single estimate of the background in the Panel. You can optionally median filter the Panel background values, e.g. to avoid ones where bright objects skewed the background estimate. The median filter size is given by *SExtractor background filter size*. It then interpolates the Panel values to find the background at each pixel. This is much faster than calculating the median of a big window at each pixel.

Both background estimates generally give reasonable results, but if the data volume is large, the Sbkg option is strongly recommended as it is much faster. It also does not require an estimate of *Outliers / Window*.

4.3.6 Overlap Modules: Detect

Command Line Equivalent: run_detect
Default Output Directory: <output_dir>/Detect-overlap
Depends On: Initial Settings; Med Filter

Important Notes: Including this module does not automatically apply Bright Object Masking. The resulting bright object detection maps are only used if the *Mask Bright Object* box is checked in the Mosaic Interpolate module.

PURPOSE

Detect performs image segmentation (see §8.4). In the Overlap pipeline, it is used only for bright object detection - i.e. to identify bright pixels (both spurious and astronomical) that should not contribute to the background matching. Detect produces bright object detection maps.

INPUT

Detection Max Area: (int) The maximum area (pixels) of a detected pixel cluster before iterative thresholding.

Detection Min Area: (int) The minimum area (pixel) for a detected pixel cluster to be retained; smaller clusters are discarded.

Detection Threshold: (float) The number of sigma above the mean to be used as the initial threshold used for cluster detection.

Threshold Type: This parameter determines the way the image segmentation threshold is recalculated during iterative thresholding. The threshold type does not depend on the type of input image. See §8.3 for a description of the thresholding options.

Detect output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Detect-overlap.

COMMAND LINE INPUT

```
&DETECT
 Detection_Max_Area = 1000,
 Detection_Min_Area = 0,
 Detection_Threshold = 3,
 Threshold_Type = 'simple',
&END
```

In Global Parameters:

```
DETECT_DIR = Detect-overlap
```

OUTPUT

Generated Fits Files (detmap.fits*):** With the default parameters, these are the detection maps produced from the input images. A positive number indicates the threshold level reached for that cluster.

DISCUSSION

In the Overlap pipeline, Detect is part of Bright Object Masking (set *Mask Bright Object* in the Mosaic Interpolate module). The detection of bright pixels is performed on the background-subtracted images (the output of the Med Filter module), which are in original pixels. The output is a set of mask images, one per input image, in which pixels corresponding to detected objects are set to a positive value. The mask images are saved in the *Detect output subdirectory* unless the *Delete Intermediate Files* option was set in the Overlap Settings module. If *Mask Bright Object* is set in the Mosaic Interpolate module then these masks will be used in the further processing.

4.3.7 Overlap Modules: Mosaic Interpolate

Command Line Equivalent: run_mosaic_int
Default Output Directory: <output_dir>/Interp-overlap
Depends On: Initial Settings; Fiducial Image Frame; S/N Estimator (if included)

PURPOSE

This module performs a projection of input images onto a 2D plane defined by the FIF table, and an interpolation (see §8.4) of the input pixel values to the output array of pixels of the user-defined pixel size. It corrects for the optical distortion in the input images, using the WCS distortion parameters in the input FITS headers. The process is intended to accept images measuring surface brightness (MJy/sr or microJy/arcsec^2) and to yield images in the same units, but it is not restricted to this. If the input FITS header does not contain an allowed string specifying units of MJy/sr or microJy/arcsec^2, then it will assume the input units are counts.

INPUT

INTERP METHOD: There are four possible interpolation methods. These are discussed in §.5.6.8. Since high spatial frequency information is not critical for simple background-matching, the default templates for Overlap use a coarse Grid method for speed.

Mosaic Interpolate output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Interp-overlap.

Mask bright object: Check this box to tell MOPEX to implement Bright Object Masking. This masks bright objects in the BCDs, and so should provide a more accurate background-matching result where there are bright objects in the field of view.

COMMAND LINE INPUT

```
&MOSAICINTIN
 INTERP_METHOD = 3,
 FINERES = 0,
 DRIZ_FAC = 1,
 GRID_RATIO = 4,
 ALPHA = -0.5,
&END
```

In Global Parameters:

```
mask_bright = 1
INTERP_DIR = Interp-overlap
```

OUTPUT

Mosaic FIF Table (*mosaic_fif.tbl*): The FIF table describing the final mosaic, taking into account the user-selected pixel size. Note: This file is saved in the top level Output directory.

Geometry Output Table (*interp_ImageList.txt.tbl*): The interpolated images' offsets in x- and y-direction relative to the FIF and their sizes are specified in this file.

Interp Stack (*interp_*fits*): The output images, interpolated, distortion-corrected, and projected onto the output FIF. A list is also made.

Coverage Stack (*interp_*covg.fits*): The corresponding coverage maps for each output image, taking into account the bad pixels in the input. A list is also made.

DISCUSSION

When choosing your interpolation scheme, bear in mind that each frame is overlap-corrected by a single value for the entire frame. If speed or memory are a factor when running MOPEX on your machine, then you may want to consider a simple interpolation scheme here (e.g. *Grid*), and reserve the more complicated interpolation for the Mosaic script. The *Grid* option is the default in the Overlap Pipeline for this reason.

4.3.8 Overlap Modules: Compute Overlap Correction

Command Line Equivalent: compute_overlap_correction
Default Output Directory: <output_dir>/Overlap_corr
Depends On: Initial Settings; Mosaic Interpolate

Important Notes: Running this module does not automatically apply the overlap correction. In order to apply the correction and output the corrected BCDs, the *Apply Overlap Correction* box in this module must be checked.

PURPOSE

This module computes the correction needed to set the background of overlapping images to a constant value. Optionally, it applies the correction to the input BCD images and creates corrected images for input to Mosaic.

INPUT

Top Threshold: (float) Number of sigma above the mean to detect outliers among the computed offsets.

Bottom Threshold: (float) Number of sigma below the mean to detect outliers among the computed offsets.

Minimum Number of Images: (int) Minimum number of overlapping images to perform outlier rejection.

Use Sparse for Many Files: Turn on to use a sparse matrix solver to increase the number of images it can solve.

Skip Bad Image Check: If no bad images, can turn this on for some speed gain.

Apply Overlap Correction: Apply the calculated offsets to the BCD images to create the new background-corrected images.

COMMAND LINE INPUT

```
&COMPUTEOVRLAPCORRIN
 TOP_THRESHOLD = 3,
 BOTTOM_THRESHOLD = 3,
 MIN_IMG_NUM = 4,
 WEIGHT = 0.0,
&END
```

WEIGHT: (float; command line only) Determines the overall level that the frames should be corrected to. In the first method (*WEIGHT = 0.0*, the default), it determines the overall level after finding the offsets, by forcing the sum of the offsets of all the images to add up to 0. In the second method (nominally *WEIGHT = 1.0*), it minimizes the squared offsets as part of the original minimization. See §4.1 Background Matching Overview for more information.

OUTPUT

Overlap Correction Table (*offsets.tbl*): A table of calculated offsets for each input DCE is the primary output of the Overlap flow.

Corrected Images (*overlapCorrection_*.fits*): If *Apply Overlap Correction* is turned on, a new FITS stack is output, with the background-matching offsets applied. These outputs are written to the *Overlap_corr* subdirectory.

DISCUSSION

The background matching algorithm (see §8.5) calculates an offset to apply to each image in order to set their backgrounds to a constant value. After the initial calculation, the offsets are examined, and outliers are rejected, as long as the minimum number of images are present. The other inputs to this module control how much of an outlier an offset has to be in order to be rejected. A rejected outlier will not be used in determining the overall offset level for all frames, but will still be applied to the corresponding data frame to bring the background to the same offset as the other input frames.

The offsets and their uncertainties are written in the header of the corresponding output FITS files. The keywords are OVRLPDC and OVRLPDCD. They keyword OVRLPOUT is set to 1 if the offset was determined to be an outlier. A table, *offset.tbl*, is also created, containing the

offsets and their uncertainties for all the files. The "outliers" column is used to indicate whether the offset in that row was determined to be an outlier. Below is a sample output table:

```
|Image_id|   Offset    |  del_Offset  | outliers |
| int    |   real      |    real      |   int    |
       1   0.13305403   0.00988516      1
       2   0.09888604   0.00963296      0
       3   0.01376509   0.00964912      0
```

4.3.9 Overlap Modules: Quicklook Mosaic

Command Line Equivalent: mosaic_corrected_images
Default Output Directory: <output_dir>
Depends On: Initial Settings; Mosaic Interpolate

Important Notes: The mosaic produced by this module should only be used for checking the results of the Background Matching. **Do not use it for science.** To create a science-quality mosaic, use the Mosaic pipeline.

PURPOSE

This module produces a quick mosaic of the interpolated, overlap-corrected images produced during the Overlap pipeline.

INPUT

Mosaic CoAdder: TILEMAX_X, TILEMAX_Y: (int) The size of output Tiles (see §8.1) can be specified. For small mosaics, these numbers should be larger than the expected output size to avoid tiling.

COMMAND LINE INPUT

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

OUTPUT

Tile BCD file (*coadd_tile_bcd.txt*): This text file, *coadd_tile_bcd.txt*, lists the input images that contribute to each output tile. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Tile table (*coadd_tiles.tbl*): This output table, *coadd_tiles.tbl*, lists the output tile names, together with their size and their X and Y offsets from the origin of the FIF coordinate system.

Combined Mosaic Image File (*mosaic.fits*): The output mosaic, *mosaic.fits*. This images is a combination of the individual tiles, if multiple tiles are created.

The Tiles and associated tables are written to the *Coadd-overlap* subdirectory. The mosaic is written to the main output directory.

DISCUSSION

Since the interpolated images have already been created, the newly computed overlap correction for each frame is applied to the interpolated images, and the corrected frames are co-added with a simple average into a single mosaic. This mosaic should be considered a browse quality product only, and is a quick and dirty way to examine the results of overlap correction. Mosaic uncertainty images are not created, and the weighted coadd options used in the Mosaic Pipeline are not available. Large mosaics may be tiled. This module runs the equivalent of Mosaic CoAdder and Mosaic Combine.

The output tables, text files, individual tile images, and coverage images are written to the *Coadd* directory under the output directory specified in *Output Directory* in the Initial Setup module. The combined mosaic is written to the main output directory. This file may be overwritten by other coaddition modules (e.g. the Mosaic Combine module in the Mosaic Pipeline).

Chapter 5. Mosaicking (*mosaic.pl*)

The Mosaic pipeline performs interpolation and co-addition of FITS images. In Spitzer terms, the software takes the individual Basic Calibrated Data (BCD) frames and combines them to create a mosaic of the observed region. There is a choice of interpolation and co-addition schemes, and the package also carries out the user's choice of outlier rejection to remove cosmic rays and bad pixels. Three main files are output by the package: a mosaic of the input BCDs, a coverage map showing how many of the input frames were rejected, and an uncertainty mosaic showing the uncertainties in each output pixel.

5.1 Basic Input Requirements

In order to carry out a basic mosaicking, MOPEX will expect some or all of the following files as input. See Chapter 3 on MOPEX Input for more information on the format of these files.

- A text file containing a list of input data images (e.g. Spitzer BCD frames; required)
- A text file containing a list of input uncertainty images (optional)
- A text file containing a list of input status mask images (optional)
- A mask image flagging permanently damaged pixels in the detector (PMask; optional)
- A "namelist" file specifying the parameters to be used for mosaicking (Command-Line MOPEX only; required)

5.2 Running Mosaic

To load a standalone Mosaic pipeline (i.e. without appending to an existing Overlap pipeline), either start from a MOPEX template namelist (*File > New Mosaic Pipeline*), load an existing Mosaic namelist file (*File > Read Name List*) or load an empty flow and insert a Mosaic pipeline from there (*File > New Empty MOPEX Pipeline*, then go to *Insert Mosaic* at the top of the flow window and choose *Empty Pipeline*). If you choose the last option, MOPEX will load an empty Mosaic flow to which you can add and subtract modules, and modify the input parameters.

To append a Mosaic pipeline to the end of an existing Overlap pipeline so that you can use the background-corrected images as input, click on the *Insert Mosaic* rectangular button at the top of the Overlap window. This will pull up a dialogue box giving you three options. Choose *From File* to load a previously-created namelist from file, *From Template* to start from one of the inbuilt Mosaic templates, and *Empty Pipeline* to open an empty Mosaic flow that you can build up by

hand. If you have previously added and deleted a Mosaic pipeline from a particular flow, the last option will instead read *Re-add Pipeline*.

For information on how to run Mosaic on the command line, see Chapter 9.

5.3 Mosaicking a Portion of the Total FIF

If you have a list of images covering a large area of the sky, but only wish to make a mosaic covering a small portion, you can use the Mosaic Geometry module. This module can take a user-supplied FIF, along with the full lists of input images, and create lists of only those images that fall within the FIF area. To create a modified FIF and run Mosaic Geometry:

1. Create the Fiducial Image Frame based on the full input list of images. To do this, run Mosaic with only the Fiducial Image Frame activated in the namelist. This will create the *FIF.tbl* file that specifies the spatial boundaries of all of your input frames.

2. Open *FIF.tbl* in a text editor, and modify it to cover only the spatial area that you wish to mosaic. For example, the initial FIF.tbl format is as follows:

```
\char comment = Output from fiducial_image_frame, version 1.4
\char Date-Time = Mar 12, 2012, 09:21:44
\real CRVAL1 = 329.129598
\real CRVAL2 = 64.662007
\real CRPIX1 = 264.50
\real CRPIX2 = 382.50
\real CROTA2 = -71.01896
\real CDELT1 = -3.39817E-04
\real CDELT2 =  -3.39801E-04
\int  NAXIS1 = 528
\int  NAXIS2 = 764
\int  EQUINOX = 2000
\char PROJTYPE = TAN
\char COORDINATE_SYSTEM = J2000
\char CTYPE1 = RA---TAN
\char CTYPE2 = DEC—TAN
\real EXTENT_X = 0.179423
\real EXTENT_Y = 0.259608
```

These are standard FITS header keywords for the most part. The EXTENT_X and EXTENT_Y keywords are sizes (in degrees) of the nominal mosaic frame if the default pixel scale is used.

They do not necessarily reflect the size of final mosaic as the pixel size sent to Mosaic Interpolate can be different. They can be ignored.

This file can be edited to produce a sub-mosaic at the center with a size of 100 by 100 default pixels and orientation of -45.0 deg E of N by changing the following lines:

```
\real CROTA2 = -45.0
\real CRPIX1 = 50.5
\real CRPIX2 = 50.5
\int  NAXIS1 = 100
\int  NAXIS2 = 100
```

3. Re-run Mosaic but this time with the following settings:

   - In the Initial Settings, specify the name of the new FIF file as input;
   - Remove the Fiducial Image Frame module from the pipeline to prevent it overwriting the new FIF;
   - Include the Mosaic Geometry module in the pipeline;

   The Mosaic Geometry module will create new lists of the input data, uncertainty and mask files that cross the modified FIF, and will pass those lists on to the subsequent modules in the Mosaic flow. The new lists are saved as *geom_*.txt.* (Insertion of Mosaic Geometry is not required to make a mosaic, but creates the lists and shortens subsequent tasks to the shorter list.)

   Mosaicking to an alternate FIF from, for example, another dataset, just requires step 3 above.

5.4 Creating a Mosaic of a Moving Object

This functionality is limited to Spitzer data. Some Spitzer observations are designed for moving targets, such as comets or asteroids. Moving target coordinates are written in the input image header with the keywords RA_REF and DEC_REF. The keywords are written in the headers of all input images, regardless of whether the moving object is present in the frame.

MOPEX can create a mosaic in the rest frame of the moving object by setting the *Moving Object Mosaic* value in the Mosaic Settings module. On the command line, you should set the parameter *MOVING_OBJECT_MOSAIC* in the General Settings section. The value specifies the frame number that you wish to select as the stationary sky position. All other frames are then shifted

relative to this initial stationary frame. i.e., if *Moving Object Mosaic = 5* then MOPEX will use the pointing and target position in the 5th frame as the stationary reference frame, $CRVAL1_{stat}$, $CRVAL2_{stat}$, RA_REF_{stat} and DEC_REF_{stat}. The pointing and target information for the ith image, $CRVAL1_i$, $CRVAL2_i$, RA_REF_i and DEC_REF_i is therefore redefined as follows:

$$CRVAL1_i \rightarrow CRVAL1_i + (RA\_REF_{stat} - RA\_REF_i)$$
$$CRVAL2_i \rightarrow CRVAL2_i + (DEC\_REF_{stat} - DEC\_REF_i)$$

Equation 5.1

Setting *Moving Object Mosaic* to 0 is the same as omitting a value entirely, and will create a standard mosaic. Assigning any other value is the only action that the user needs to take to create the moving object mosaic. Be aware that outlier detection in this mode will reject not only cosmic ray hits and bad pixels, but real sources as well. This is because the real sources become transient in the rest frame of the moving object.

5.5 Mosaic Pipeline Stages

The main processing stages of the Mosaic pipeline are as follows:

Interpolation

The input images are interpolated onto the output grid defined by the FIF table. This interpolation is performed by the Mosaic Interpolate module, and there are four interpolation schemes available. This module also corrects for geometric distortion.

Outlier Detection

MOPEX employs four outlier detection methods to detect radiation hits, bad pixels and moving objects. Any number of these methods can be run on the data. The rejected pixels are flagged in the RMask, and the user can select which of the rejection schemes to set as "fatal".

Reinterpolation

The interpolated images are updated to include the outlier detection information. Only the pixels affected by the outlier rejection are recomputed.

Coaddition

The interpolated images are co-added to create one mosaic image. There are several co-addition schemes available, but the default is a straight average of the input pixel values. The interpolated uncertainty images are also co-added into one uncertainty mosaic, and the coverage maps are co-added to produce a coverage mosaic.

Combining

In cases with large mosaics, the coaddition step must be carried out in "tiles" to avoid running out of memory. This final step combines the tiles into one final mosaic. Note that even if the coaddition has been carried out in a single tile, this step must still be run in order to produce the expected output.

5.6 Mosaic Modules

The following pages contain full descriptions of the modules that are available in the Mosaic GUI pipeline. For information on how to turn these modules on and off on the command line, see Chapter 9. For suggestions on which modules to use, see §2.5 Which Modules Should I Choose?

Module listing:

- Initial Setup
- Mosaic Settings
- S/N Estimator
- MedFilter
- Detect Radhit
- Fiducial Image Frame
- Mosaic Geometry
- Mosaic Interpolate
- Detect (Outlier)
- Mosaic Projection
- Mosaic Coverage
- Mosaic Dual Outlier
- Level
- Mosaic Outlier
- Mosaic Box Outlier
- Mosaic RMask
- Mosaic Reinterpolate
- Fix Coverage
- Mosaic CoAdder
- Mosaic Combine
- Mosaic MedFilter
- Create RMask Mosaic
- Make Array Correction Files
- Make Array Correction Mosaic

5.6.1 Mosaic Modules: Initial Setup

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: None

PURPOSE

Set up the input and output files for the loaded flow.

INPUT

Image Stack File: The text file containing the list of input data files for the pipeline. Typically these are the **bcd.fits* files that you download from the Spitzer data archive. See Chapter 3 and Table 3.1 for more information on the input file requirements and format.

Output Directory: The directory that you want to set as your output directory for this run.

Multi-processing Mode: Switch to use multi-processing. See Discussion.

Processors for Multi-processing: Number of processors to use. See Discussion.

Optional Input and Mask Files:

Sigma List File: The text file containing the list of uncertainty images that correspond to the input data files (usually the **bunc.fits* files; see Table 3.1 for more information).

DCE Status Mask List: The text file containing the list of mask images that flag temporarily-bad pixels in the input data images. There should be a mask file for each of the input data files (these depend on the instrument, but will be labeled something like **msk.fits*; see Table 3.1 for more information).

Fatal Mask Bit Pattern: This is the bit pattern that defines which type of flagged problems that you wish to set as fatal when combining the input images. See §8.11 for more information. There is a separate Fatal Bit Pattern for each type of mask that is being used as input (i.e. for the DCE Status masks, the RMasks and the PMask).

Rmask List File: The text file containing a list of outlier mask images. Usually this will be left blank, as the RMask images are created during the mosaicking process. This option allows you to specify a previously created list of RMask images as input on second and subsequent runs of the pipeline, to save having to re-generate them.

Pmask FITS File: The permanently-damaged pixel mask for the detector. This is a single FITS image, flagging the permanently damaged pixels in the instrument arrays. These files are stored in the *<mopex_dir>/cal/* directory. There is a single PMask for each of the MIPS and IRS channels, but the IRAC ones have changed with time. See the file *<mopex_dir>/pmasks.README* for information on which PMask is applicable to your data. Note that the date that the data were taken is given in the DATE_OBS header keyword (**not** DATE - this is the date the file was written).

FIF file: The Fiducial Image Frame file (see §5.6.4). This is an optional input file, as it is often generated from the input data using the Fiducial Image Frame module. There are, however, a number of cases in which you might wish to use a user-specified FIF (e.g. if you are trying to match the field of view to the other data channels).

COMMAND LINE INPUT

The following are set in the Global Parameters part of the namelist:

```
IMAGE_STACK_FILE_NAME = <working_dir>/imagelist.txt
OUTPUT_DIR = <working_dir>/output
SIGMALIST_FILE_NAME = <working_dir>/sigmalist.txt
DCE_STATUS_MASK_LIST = <working_dir>/masklist.txt
DCE_Status_Mask_Fatal_BitPattern = 32544
RMASK_LIST = <working_dir>/rmasklist.txt
RMask_Fatal_BitPattern = 7
PMASK_FILE_NAME = <mopex_dir>/cal/sep07/sep07_ch1_bcd_pmask.fits
PMask_Fatal_BitPattern = 32767
FIF_FILE_NAME = <working_dir>/FIF.tbl
verbose = 1
NICE = 1
save_namelist = 1
```

Verbose, *NICE* and *save_namelist* are only available on the command line. See §9.4.2.3 for more information.

OUTPUT

None

DISCUSSION

This module sets up the input files and top-level output directory for the first pipeline in the flow. Subsequent pipelines that have been inserted into the flow will use the output from the previous one (i.e. if you have inserted Overlap and Mosaic into your reduction flow, Overlap will use the Initial Settings files as input, but Mosaic will use the modified files output by Overlap). Only *Input Image Stack* and *Output Directory* are required, but *Sigma File List*, *DCE Status Mask List* and *Pmask FITS File* are recommended.

NEW Note that with version 18.5.0 one can use multiprocessing to speed up MOPEX tasks. The tasks Overlap, Mosaic, Apex (Multi) and Apex User List (Multi) now allow multiprocessing of some steps. In the GUI, this is set in Initial Setup, Multi-Processing Mode. The options are "on", "off", and "manual". The default is "on" -- this grabs 3/4 of the available processors. Setting "manual" lets the user set the number of processors used. You should see speed-ups of at least 2x.

If using command-line, add these lines at the top of your namelist:
do_multiprocess = on | off | manual default = off
ncpu_multiprocess = 1 default = 1
If "manual" is set, set ncpu_multiprocess to the number of processors to use.

5.6.2 Mosaic Modules: Mosaic Settings

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: Initial Setup

Important Notes: None

PURPOSE

To set the input parameters specific to the Mosaic pipeline.

INPUT

Pixel by Size: Check this to set the output pixel size in degrees. The default pixel size depends on the instrument and channel: see the templates for the default sizes.

Pixel Size X (deg): Set the X-size of the output pixels in degrees. The pixel size in the X-direction is quoted as a negative value to comply with convention. A positive value will generate an error message.
Pixel Size Y (deg): Set the Y-size of the output pixels in degrees.

Pixel By Ratio: Check this to set the output pixel size by ratio of the input pixel to the output pixel, i.e. to make the output pixels half the size of the input pixels (over-sample the mosaic), set the values of *Pixel Ratio X(Y) = 2*. The ratio is taken after correction for geometric distortion.

Pixel Ratio X: Set the pixel ratio in the X-direction.
Pixel Ratio Y: Set the pixel ratio in the Y-direction.

Array-Location Dependent Photometric Correction Mosaic: This allows you to set the defaults for the two parameters below, based on when the IRAC data were taken.

Array Correction Frame: The name of the Array Location Dependence Correction image (for IRAC only). This file is only required if you have included the modules Make Array Correction Files or Make Array Correction Mosaic (see those modules for more information). The default location for this file is the *cal/* directory of your MOPEX installation, but you may need to download the correction files from the website.

PixArea Frame: The name of the Pixel Distortion Correction image (for IRAC only). This file is only required if you have included the modules Make Array Correction Files or Make Array Correction Mosaic (see those modules for more information). The default location for this file is the *cal/* directory of your MOPEX installation, but you may need to download the correction files from the website.

Delete Array Correction Intermediate Files: Check this box to delete the intermediate files created by the Array Location Dependence Correction calculation. This is useful for saving disk space once you are satisfied with your reduction setup.

Moving Object Mosaic: This setting gives you the option to create a mosaic in the rest frame of a moving object (e.g. a comet or asteroid). The "stationary" frame is chosen with the value of this keyword, and all other images are shifted relative to the stationary frame. i.e. if *Moving Object Mosaic = 5*, MOPEX uses the 5th frame in the input stack as the base image, and shifts all other images to put the moving object at the same sky position. This functionality is limited to Spitzer data. See §5.4 for more information on creating Moving Object Mosaics.

Delete Intermediate Files: Check this to delete all of the intermediate files created by the Mosaic pipeline. This leaves only the input files and the files created by the final module. This is useful for saving disk space once you are satisfied with your reduction setup. Note that MOPEX will prompt you with a pop-up dialogueue box at the end of the flow, to be sure that you meant to select this irreversible option.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.** Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

Create Absolute Minimum Mosaic: This option sets the Mosaic CoAdder module to create an absolute minimum mosaic (*mosaic.fits*) and the corresponding uncertainty mosaic (*mosaic_unc.fits*) **instead of the usual mosaic output**. It will cause MOPEX to overwrite any identically named files in the output directory. The mosaic pixel value is set to the input pixel value with the smallest absolute value in the stack of the interpolated images. The corresponding mosaic uncertainty pixel is set to the pixel value in the interpolated uncertainty image corresponding to the interpolated image with the smallest absolute pixel value. This setting is useful for outlier rejection.

COMMAND LINE INPUT

The following parameters are set in the Global Parameters part of the namelist. Note that if both *MOSAIC_PIXEL_SIZE_X(Y)* and *MOSAIC_PIXEL_RATIO_X(Y)* are both set, the former takes precedence and will override the latter.

```
MOSAIC_PIXEL_SIZE_X = -0.00033889
MOSAIC_PIXEL_SIZE_Y = 0.00033889
MOSAIC_PIXEL_RATIO_X = 1
MOSAIC_PIXEL_RATIO_Y = 1
ARRAY_CORR_IMAGE = <mopex_dir>/cal/ch1_photcorr_rj.fits
ARRAY_PIXAREA_IMAGE = <mopex_dir>/cal/ch1relpixarea.fits
MOVING_OBJECT_MOSAIC = 5
delete_intermediate_files = 0
USE_REFINED_POINTING = 0
run_absolute_minimum_mosaic = 0
```

OUTPUT

None

DISCUSSION

Mosaic Settings sets up all of the input options that are specific to the Mosaic pipeline. Some of them may be repeated from the Overlap Settings, so you should be aware that you may need to match some settings (e.g. *Pixel Ratio*) between the pipelines.

5.6.3 Mosaic Modules: S/N Estimator

Command Line Equivalent: compute_uncertainties_internally
Default Output Directory: <output_dir>/Sigma-mosaic
Depends On: Initial Setup; Mosaic Settings

Important Notes: Only required if you do not have the uncertainty images that come with all Spitzer data.

PURPOSE

This module is used to estimate the uncertainty images when no independent uncertainty measurement is available. This module is usually not turned on since almost all Spitzer data have independent uncertainty images (**unc*.fits*) that come downloaded with Spitzer data from the archive. If you decide not to use the provided uncertainty images, or if you are not using Spitzer data, this module needs to be switched on. Warning: if the wrong *Gain* or *Read Noise* parameters are specified, this module may produce uncertainties that are too large, and this will result in no outlier detection.

INPUT

Gain in e-/image unit: (float) This is the parameter used to translate the measured data counts to physical units of MJy/sr (or mJy/sq.arcsec). All Spitzer BCDs are in units of MJy/sr. **This parameter has the same name as stored in the Spitzer BCD FITS header, but they have different units.** MOPEX *Gain* has the unit of e-/MJy/sr (or e-/mJy/sq.arcsec) and the FITS header GAIN has the unit of e-/DN. *Gain* used by MOPEX can be computed from the GAIN parameter in the FITS header by using formula:

$$Gain(MOPEX) = \frac{EXPTIME * GAIN(header)}{FLUXCONV}$$

Equation 5.2

Here FLUXCONV is the flux conversion factor between e-/DN and surface brightness unit of MJy/sr (or mJy/sq.arcsec). FLUXCONV can be found in the Spitzer BCD FITS image header.

Read Noise in e-: (float) This parameter characterizes the noise in e- when the data are being read out from detector. It is also stored in the BCD FITS image header.

Confusion Sigma in e-: (float) The 1-sigma confusion noise limit. The source of this noise is from the spatially unresolved background galaxies. This information can be found in the IRAC and MIPS Instrument Handbooks.

S/N estimator output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Sigma-mosaic.

COMMAND LINE INPUT

```
&SNESTIMATORIN
 Gain = 66.8,
 Read_Noise = 8.8,
 Confusion_Sigma = 0,
&END
```

In Global Parameters:

```
SIGMA_DIR = Sigma-mosaic
```

OUTPUT

S/N FITS Files (_sigma.fits*):** The estimated uncertainty image corresponding to each input BCD image.

DISCUSSION

This module allows users to estimate the noise from the BCD data, if no uncertainty images are provided. Here you need to input gain, readout noise and confusion limit appropriate for the dataset. We suggest that users look at the BCD image header to find the gain and read out

noise. This module should not be used unless you cannot use the Spitzer archive-provided uncertainty images.

The module estimates the pixel uncertainty for each pixel in the image using the following model:

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

Equation 5.3

where the parameters *Read Noise* $\sigma_{readnoise}$, *Gain* g, and *Confusion Sigma* $\sigma_{confusion}$ are specified in the module settings. 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 units of surface brightness (MJy/sr; i.e. all Spitzer BCDs) then the *Gain* should include the conversion factor from surface brightness units to electrons (see the *INPUT* for this module). The product of this step is the uncertainty images.

5.6.4 Mosaic Modules: Fiducial Image Frame

Command Line Equivalent: run_fiducial_image_frame
Default Output Directory: <output_dir>
Depends On: Initial Settings; Mosaic Settings

Important Notes: Running this module will overwrite any pre-existing *FIF.tbl* file in the output directory. **If you are working with MIPS images and plan to use the Mosaic PRFs then please see the Discussion below for important information.**

PURPOSE

This module creates a unified grid coordinate system and defines the spatial boundaries that will include all BCD images. The output FIF table is required for later modules to run. It can be generated by this module, or set as an input using a pre-existing table.

INPUT

Edge Padding: (int) This number specifies how much extra space (arcseconds) will be added to the four edges of the final mosaicked image.

Projection: This parameter specifies the projection type of the output mosaic. The TAN and SIN projections are implemented using the fast direct plane-to-plane coordinate transformation. The rest of the projections in the WCS library - LIN, AZP, STG, ARC, ZPN, ZEA, AIR, CYP, CAR, MER, CEA, COP, COD, COE, COO, BON, PCO, GLS, PAR, AIT, MOL, CSC, QSC, TSC, NCP, DSS, PLT, TNX - are implemented as a two step plane-sky-plane projection. The default projection is TAN.

Coordinate System: The coordinate system of the output mosaic. The choices are J2000 (Equatorial), Galactic, or Ecliptic. The default is J2000.

Use average input orientation: Defines the orientation of the output mosaic. Checking this box indicates that MOPEX should use the average of the input BCD orientations. This is the default setting. To specify an alternative orientation, see *CROTA2*.

CROTA2: (float) Defines the orientation (degrees East of North) of the output mosaic. *CROTA2 = 0* indicates North up and East left. To use the average of the input BCD orientations (the default), check the *Use average input orientation* box. On the command line, set *CROTA2 = A* to use the average input orientation.

COMMAND LINE INPUT

```
&FIDUCIALIMAGEFRAMEIN
 Edge_Padding = 10,
 Projection_Type = "TAN",
 Coordinate_System = "J2000",
 CROTA2 = A,
&END
```

In Global Parameters:

```
MARGIN = 0
```

MARGIN: (int; command line only) Specifies a margin in terms of the number of pixels around the Fiducial Image Frame. Normally this is set to zero or a (small) positive number, except when using the Mosaic Geometry module, when it should normally be set to a negative number to avoid including input images that have only marginal overlap with the FIF.

OUTPUT

Generated Headers List Table (*header_list.tbl*): This file lists the World Coordinate Systems (WCS) information from each of the BCD image headers. The information is used to make the Fiducial Image Frame table.

Generated FIF Table (*FIF.tbl*): This text file is used by later modules for putting together a mosaicked image with specified pixel size, orientation and the final image size. The module overwrites any input FIF table specified in the Initial Settings. Note that the output mosaic will always use the set of keywords CDELT1, CDELT2 and CROTA2, even if the input images use the CD matrix convention.

The outputs of this module are written to the top output directory specified in the Initial Settings.

DISCUSSION

This module creates a table specifying a unified coordinate system and the spatial boundaries of the output mosaic. This table is required for any interpolation or mosaicking using MOPEX. If this module is not included, an existing FIF table has to be specified in the Initial Settings to enable later modules to run. After this module is run once, the output FIF table can be modified manually if desired, and used as input to subsequent runs (this is useful for e.g. mosaicking small sections of a large dataset - see §5.3 for more details).

Even if the input images use the CD matrix convention, the output mosaic images (and by extension the interpolated images) will use the set of keywords CDELT1, CDELT2 and CROTA2, since the mosaic image is undistorted.

If you plan to use the MIPS "mosaic" PRFs that are provided for PRF fitting on your mosaic, your mosaic must be built in the same way as the ones used to derive those PRFs, that is, from BCDs that have nearly the same orientation (close in time or taken at the same time of year) and with a *CROTA2* angle that is properly aligned with the BCDs (*Use average input orientation* is recommended). All other cases would require you to make your own PRF from your mosaic.

5.6.5 Mosaic Modules: Mosaic Geometry

Command Line Equivalent: run_mosaic_geom
Default Output Directory: <output_dir>
Depends On: Initial Settings; Mosaic Settings; Fiducial Image Frame

PURPOSE

This module allows users to mosaic only the BCDs that overlap a user-specified spatial area. It takes in the full lists of input images, uncertainty files and mask files, and creates new lists that include only those images that overlap a user-supplied Fiducial Image Frame (FIF) table. The new lists are then used by subsequent modules in the Mosaic pipeline.

INPUT

This module does not have any input parameters. However, it requires the FIF table specified in the input namelist to have been edited by hand to specify the area that the user wishes to mosaic.

COMMAND LINE INPUT

See *INPUT*.

OUTPUT

The module creates working lists of images, and associated masks and uncertainty files, that are then automatically used by later modules to create the mosaic image. The filenames created are as follows:

- *geom_ImageList.txt*
- *geom_DmaskList.txt*
- *geom_SigmaList.txt*
- *geom_rmask_ImageList.txt*
- *geom_fif.tbl*
- *geom_Tile.tbl*
- *geom_Int_Img.tbl*

DISCUSSION

For complete instructions on how to use this module to create a mosaic of a subset of a larger dataset, please see §5.3: Mosaicking a Portion of the Total FIF. This includes information on how to modify the *FIF.tbl* file to specify the new area.

5.6.6 Mosaic Modules: MedFilter

Command Line Equivalent: run_medfilter
Default Output Directory: <output_dir>/Medfilter-mosaic
Depends On: Initial Setup; Mosaic Settings

PURPOSE

This module performs a background subtraction of the individual input images. It needs to run if single-frame outlier rejection or dual outlier rejection are to be used in the Mosaic pipeline (see §8.2.1; §8.2.3). There are two options: the Median case (default) and the Sbkg case, which uses a background estimate like that of SExtractor.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

SExtractor background filter size: (int) Median filter box size for the Panels when the *Use SExtractor background estimation* option is turned on. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Use SExtractor background estimation: Checking this box invokes the Sbkg background estimate based on SExtractor. Leaving it unchecked causes MOPEX to use the default Median estimate.

Med Filter output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Medfilter-mosaic.

COMMAND LINE INPUT

```
&MEDFILTER
 Window_X = 45,
```

```
 Window_Y = 45,
 N_Outliers_Per_Window = 50,
 Sbkg_Filt_Size_for_Med = 1,
 Use_Sbkg_for_Med = 1,
&END
```

In Global Parameters:

```
MEDFILTER_DIR = Medfilter-mosaic
```

OUTPUT

Generated Fits Files (_minback.fits*):** The background subtracted, individual images. A list of the generated images is also written out.

DISCUSSION

This module is required in order to do single-frame (Detect Radhit) or dual (spatial & temporal) outlier rejection (see §8.2 Outlier Detection). The background subtraction is only for the purpose of outlier rejection, and the background-subtracted images are not used for later mosaicking.

With the default Median option, the program computes a background value using the median of a running rectangular window of *Window X* by *Window Y* pixels, after omitting the *Outliers / Window* highest pixels. This is done for each pixel so can be very slow.

If *Use SExtractor background estimation* is set, the module switches to the Sbkg background estimation based on that of SExtractor (*Bertin and Arnouts, AASupp 117, 393, 1996*). The image is divided into Panels with size given by *Window X, Y*. In each Panel, iterative clipping is used to find a single estimate of the background in the Panel. You can optionally median filter the Panel background values, e.g. to avoid ones where bright objects skewed the background estimate. The median filter size is given by *SExtractor background filter size.* It then interpolates the Panel values to find the background at each pixel. This is much faster than calculating the median of a big window at each pixel.

Both background estimates generally give reasonable results, but if the data volume is large, the Sbkg option is strongly recommended, as it is much faster. It also does not require an estimate *of Outliers / Window*.

5.6.7 Mosaic Modules: Detect Radhit

Command Line Equivalent: run_detect_outlier
Default Output Directory: <output_dir>/Dmask-mosaic
Depends On: Initial Settings; Mosaic Settings; Fiducial Image Frame; Med Filter

PURPOSE

This module performs single frame radhit rejection (§8.2.1). It is the simplest of the outlier detection schemes. Bright detections with small areas are classified as radhits. By default, single frame radhit rejection is already run during the Spitzer MIPS and IRAC BCD pipeline processing, so this module is rarely needed. The radhit pixels are flagged in the status masks provided with the BCD products. If DCE Status Mask List is specified in the Initial Setup module, this module does not need to run again if the masks are satisfactory.

INPUT

Segmentation Threshold: (float) The number of sigma above the mean to be used as the initial threshold used for cluster detection.

Detection Max Area: (int) The maximum number of pixels in a connected area (cluster) that could be classified as radhits.

Radhit Threshold: (float) In order to classify a cluster of pixels as a radhit, at least one of the pixels should be greater than this threshold in sigma. This parameter should be set much higher than the segmentation threshold value.

Output directory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Dmask-mosaic.

COMMAND LINE INPUT

```
&DETECT_RADHIT
 Segmentation_Threshold = 3,
 Detection_Max_Area = 3,
 Radhit_Threshold = 10,
&END
```

In Global Parameters:

```
DMASK_DIR = Dmask-mosaic
```

OUTPUT

New DCE Mask Files (**_dmask.fits*): This module will produces mask images flagging detected radhits by setting Bit 9 (see the Instrument Handbooks, available from the website, for bit definitions).

DISCUSSION

The first and most basic method of outlier rejection is the single frame outlier detection that represents spatial filtering of input images. It is performed on the input images. In the automated MIPS and IRAC pipelines, this step is included in the standard BCD processing. It is designed to detect relatively bright and spatially small radhits. The algorithm is a variant of image segmentation, and is based on the idea that when a relatively low detection threshold is applied, for each bright point source a large number of pixels will be detected above the threshold. Bright detections with small areas are, therefore, classified as radhits. For more information on all of the outlier rejection schemes available in MOPEX, see §8.2.

NOTE: For Spitzer data, radhit is run as part of pipeline processing, so users should generally not need to run this again.

5.6.8 Mosaic Modules: Mosaic Interpolate

Command Line Equivalent: run_mosaic_int
Default Output Directory: <output_dir>/Interp-mosaic
Depends On: Initial Settings; Fiducial Image Frame; S/N Estimator (if included)

PURPOSE

This module performs a projection of input images onto a 2D plane defined by the FIF table, and an interpolation (see §8.4) of the input pixel values to the output array of pixels of the user-defined pixel size. It corrects for the optical distortion in the input images, using the WCS distortion parameters in the input FITS headers. The process is intended to accept images measuring surface brightness (MJy/sr or microJy/arcsec^2) and to yield images in the same units, but it is not restricted to this. If the input FITS header does not contain an allowed string specifying units of MJy/sr or microJy/arcsec^2, then it will assume the input units are counts.

INPUT

INTERP METHOD: Four interpolation options are available:

1. Default: Each output pixel is a linear weighted sum of input pixels with weights equal to the area overlap with the output pixel. The optional parameter is *Fine Res* (int). The input pixel can be sub-divided with sub-pixel sizes of *Input Pixel Size / Fine Res* before projecting onto the output array; a typical value might be 2. The value of each sub-pixel is a linear interpolation of the input pixels with weights determined by the area overlap with a pixel the same size as the original but centered on the sub-pixel. So it's a convolution. The default value of 0 means no sub-dividing. **We strongly advise setting *Fine Res = 0* as the *Fine Res* option is not fully implemented in the presence of bad pixels.**

2. Drizzle: Each input pixel is shrunk *Drizzle Factor* (float) times its original size along each axis, e.g. 0.5. The value of the shrunken pixel is the same as the original pixel. The shrunken pixels are then projected onto the output pixels with their values distributed into output pixels according to area overlap. Note: When using Drizzle and creating RMasks from the data, several of the mosaicking steps are run twice. First, a normal (Default) interpolation is carried out so that MOPEX can run the outlier rejection scheme for the original pixels. Once the outlier rejection masks (RMasks) have been created, MOPEX returns to the Mosaic Interpolate module and re-runs the interpolation with the Drizzle algorithm, masking out pixels flagged in the RMasks. **When using this option, do not include the Mosaic Reinterpolate Module in the processing flow**.

3. Grid: This method is intended to create a crude first-look mosaic quickly. Each input pixel is filled with *Grid Ratio* (int) squared grid points. Each grid point is assigned the value of the pixel it belongs to. Each grid point is projected onto the output frame and the flux associated with the grid point is added to the output image pixel into which the grid point was projected. You may not run the Mosaic Reinterpolate module with this option. The gain in speed is up to 10 times that of the Default method. **The price you pay is the fidelity of the interpolated images, so in general these should not be used for science.**

4. Cubic This method uses a bicubic interpolation. It is like the Default method except that each output pixel is a weighted sum of the 16 nearest input pixels, with the weights determined by bicubic polynomials. A tunable parameter *Alpha* (float) pins the weights. The default value of -0.5 should be used. Note: this method could be useful in high S/N cases because it enforces smoothness in the interpolation function, but the major drawback is significantly more noise correlation than the Default linear interpolation.

Mosaic Interpolate output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Interp-mosaic.

COMMAND LINE INPUT

```
&MOSAICINTIN
 INTERP_METHOD = 1,
 FINERES = 0,
 DRIZ_FAC = 1,
 GRID_RATIO = 4,
 ALPHA = -0.5,
&END
```

In Global Parameters:

```
INTERP_DIR = Interp-mosaic
```

OUTPUT

Mosaic FIF Table (*mosaic_fif.tbl*): The FIF table describing the final mosaic, taking into account the user-selected pixel size. Note: This file is saved in the top level Output directory.

Geometry Output Table (*interp_ImageList.txt.tbl*): The interpolated images' offsets in x- and y-direction relative to the FIF and their sizes are specified in this file.

Interp Stack (*interp_*fits*): The output images, interpolated, distortion-corrected, and projected onto the output FIF. A list is also made.

Coverage Stack (*interp_*covg.fits*): The corresponding coverage maps for each output image, taking into account the bad pixels in the input. A list is also made.

DISCUSSION

Most users will choose between *Default* and *Drizzle*. The smaller you make *Drizzle Factor*, the more coverage you need to avoid holes in the mosaic. A rule of thumb is that if you have a coverage of 10 or less, you will probably want to use the *Default* interpolation.

In general, a projection onto the reference frame of an input image with optical distortions will not be a simple rectangle. Each interpolated image occupies a part of the FIF and is, in

general, of different size, with different offsets from the origin of the FIF. The interpolated images' offsets in x- and y- directions relative to the FIF, and their sizes (in integral numbers of pixels) are given in the file *interpolated_ImageList.txt.tbl*, and also written in the headers of the interpolated images in the keywords MINTOFFX and MINTOFFY.

5.6.9 Mosaic Modules: Detect

Command Line Equivalent: run_detect_outlier
Default Output Directory: <output_dir>/Detect-mosaic
Depends On: Initial Setup; Mosaic Settings; Med Filter

PURPOSE

Detect performs image segmentation (see §8.3). In the Mosaic pipeline, it is one of four modules that, together, carry out Dual Outlier detection (see §8.2.3). The other three modules are Mosaic Projection, Mosaic Dual Outlier and Level. Detect produces bright object detection maps.

INPUT

Detection Max Area: (int) The maximum area of a single outlier identified in the detection map. If the outlier is larger than this threshold, it will be considered an extended object.

Detection Min Area: (int) The minimum area of a pixel cluster to be detected.

Detection Threshold: (float) The number of sigma above the mean for pixels included in clusters. Default value is 3.

Threshold Type: There are three different ways to decide if a cluster of bright pixels will be segmented into several clusters. The choices are simple, combo and peak (see §8.3 Image Segmentation).

Detect output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Detect-mosaic.

COMMAND LINE INPUT

```
&DETECT
 Detection_Max_Area = 1000,
 Detection_Min_Area = 0,
```

```
 Detection_Threshold = 3,
 Threshold_Type = 'simple',
&END
```

In Global Parameters:

```
DETECT_DIR = Detect-mosaic
```

OUTPUT

Generated Fits Files (detmap.fits*):** The stack of detection maps produced from interpolated BCD images. Sources with positive signals, including both real science sources and outliers from cosmic rays and bad pixels, are set to the number of the cluster membership. The background is zero in the segmentation map.

DISCUSSION

In the Mosaic pipeline, Detect is part of Dual Outlier detection (see §8.2.3). The detection of bright pixels is performed on the background-subtracted images (the output of the Med Filter module), which are in original pixels. The output is a set of mask images, one per input image, in which pixels corresponding to detected objects are set to a positive value. The mask images are saved in the *Detect output subdirectory* unless the *Delete Intermediate Files* option was set in the Mosaic Settings module. Following this module, Mosaic Projection takes the output maps and projects them to a common reference frame (the FIF) before the later modules identify true outliers.

5.6.10 Mosaic Modules: Mosaic Projection

Command Line Equivalent: run_mosaic_proj
Default Output Directory: <output_dir>/DualOutlier-mosaic
Depends On: Fiducial Image Frame; Detect

PURPOSE

This module works together with three other modules, Detect, Mosaic Dual Outlier and Level, to perform Dual Outlier rejection using both temporal and spatial filtering methods (see §8.2.3). The detection maps produced by the Detect module are in the original pixel frame. This module will project the detection maps on to a common reference frame using the Fiducial Image Frame table, *FIF.tbl*.

INPUT

This module does not have any user input parameters.

COMMAND LINE INPUT

Despite not having any input parameters, the following parameter block must be included in the namelist:

```
&MOSAICPROJIN
&END
```

OUTPUT

Interp Detection Map Table (*interp_detmap_ImageList.txt*): a list of the projected detection maps, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Interp Detection Map Images (*proj_*detmap.fits*): The FITS stack of projected, distortion-corrected, and interpolated detection maps.

The output files from this module are written to the DualOutlier-mosaic/ directory.

DISCUSSION

The module projects the Dual Outlier detection maps onto the common reference frame of the FIF. It uses the same image interpolation concepts utilized by the Mosaic Interpolate module (see §8.4 and §5.6.8).

5.6.11 Mosaic Modules: Mosaic Coverage

Command Line Equivalent: run_mosaic_covg
Default Output Directory: <output_dir>/Rmask-mosaic
Depends On: Mosaic Interpolate

Important Notes: The preliminary coverage map, *covg_map.fits* is written to the top-level output directory, <output_dir>.

PURPOSE

This module produces a preliminary coverage map covering the whole FIF for use in the later module, Mosaic RMask. The mosaic may be broken up into tiles in order to avoid memory allocation problems if the mosaic is very large (see §8.1)

INPUT

Tile Max X (Y): (int) The X (Y) size in pixels of a tile when making a large mosaic image. The default values are 2000.

COMMAND LINE INPUT

```
&MOSAICCOVGIN
 TILEMAX_X = 500,
 TILEMAX_Y = 500,
&END
```

OUTPUT

Coverage Map Tile Table (*covg_TilesList.tbl*): A table listing the tiles of the coverage map, including their sizes and offsets in the x- and y-directions from the FIF.

Coverage Map Tile list (*covg_Tile_*.fits*): The individual tiles of the coverage map.

Coverage Map (*covg_map.fits*): The preliminary coverage map over the whole FIF.
The output tiles are written into the Rmask-mosaic/ subdirectory. The preliminary coverage map is written to the top-level output directory.

DISCUSSION

No further information.

5.6.12 Mosaic Modules: Mosaic Dual Outlier

Command Line Equivalent: run_mosaic_dual_outlier
Default Output Directory: <output_dir>/DualOutlier-mosaic
Depends On: Detect; Mosaic Projection

Important Notes: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must set *Use Dual Outlier For Rmask* in the Mosaic RMask module and set the RMask *Fatal Mask Bit Pattern* in the Initial Setup module to use bit 2. Note that this is not the same as setting it to a value of 2. See §8.11: Fatal Mask Bit Patterns for more information.

PURPOSE

This module takes the outputs from Detect and Mosaic Projection and applies the user-input criteria to classify outliers using both spatial and temporal information.

INPUT

Max Outlier Image: (int) The maximum number of images in which a potential outlier can be present and still be declared a dual outlier. Default value is 2. This value should be smaller than the total number of coverage at each pixel.

Max Outlier Fraction: (float) This parameter specifies the maximum fraction of the images in which a potential outlier can be present and still be declared a dual outlier, i.e. the number of images containing outlier pixels divided by the total stack of input BCD images.

Tile X(Y) Size: (int) The X (Y) size of an image tile used for this module. When the number of input images is very large, the recommended option is to use smaller Tile size than the total number of pixels of the final output mosaic image, in order to avoid memory shortage. The mosaic may be broken up into tiles in order to avoid memory allocation problems if the mosaic is very large (see §8.1).

Dual Outlier output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is DualOutlier-mosaic.

COMMAND LINE INPUT

```
&MOSAICDUALOUTLIERIN
 MAX_OUTL_IMAGE = 2,
 MAX_OUTL_FRAC = 0.95,
 TILE_XSIZ = 500,
 TILE_YSIZ = 500,
&END
```

In Global Parameters:

```
DUAL_OUTLIER_DIR = DualOutlier-mosaic
```

OUTPUT

Dual Outlier Images (*proj_*_detmap_dual_outlier.fits*): The output images with outlier pixels flagged. Outliers have negative values and real sources have positive values.

DISCUSSION

Outlier rejection employs a complicated set of algorithms. The basic concept of Dual Outlier detection is to use both spatial and temporal filtering method. If you do not have good coverage or you want to make sure the edges of your mosaic image are clean of outliers, you should use this option. See §8.2.3 for full details of Dual Outlier Detection.

The output outlier maps have likely outliers flagged on a pixel-by-pixel basis. However, clusters may contain both positive and negative identifications. The Level module next determines whether each cluster should be considered an outlier or not in its entirety.

5.6.13 Mosaic Modules: Level

Command Line Equivalent: run_level
Default Output Directory: <output_dir>/DualOutlier-mosaic
Depends On: Detect; Mosaic Projection; Mosaic Dual Outlier

PURPOSE

This module is the last step in Dual Outlier detection. This module makes the correction to the dual outlier maps produced in the Mosaic Dual Outlier module. The dual detection map is processed in order to eliminate detection of the outskirts of legitimate point sources as dual outliers. If a dual outlier belongs to a cluster where the majority of pixels are not dual outliers, the chances are the pixel has been wrongly marked because it is on the edge of the point source.

INPUT

Threshold Ratio: (float) The threshold at which the sign of pixels within a cluster should be flipped (see below).

COMMAND LINE INPUT

```
&LEVEL
 THRESHOLD_RATIO = 0.5,
&END
```

OUTPUT

Level Images (*proj*_detmap_dual_outlier_level.fits*): The stack of dual outlier rejection maps with all pixels within each detected cluster set to the same sign.

DISCUSSION

This step is the last of the Dual Outlier detection (see §8.2.3). The dual outlier detection represents a complicated dual spatial-temporal filtering. First, all spatial pixel outliers are detected and saved as detection maps. These detection maps include point sources and radhits. Detection maps are interpolated to a common grid. Then for each pixel position in the interpolated grid the values of the interpolated pixels in the detection maps are compared. If in the majority of the detection maps a given pixel location has not been detected by spatial filtering, then the detections are declared outliers. This method detects both moving objects and radhits. This method is expected to be reliable in the shallow coverage case.

The dual detection map is processed in order to eliminate detection of the outskirts of legitimate point sources as dual outliers. If a dual outlier belongs to a cluster where the majority of pixels are not dual outliers the chances are the pixel has been wrongly marked because it is on the edge of the point source. The sign of wrongly marked pixels is flipped based on the namelist parameter *Threshold Ratio*. The following is done:

if the number of negative pixels N- in a cluster with N pixels is smaller than the threshold

$$N- / N < \textit{Threshold Ratio},$$

then their signs are flipped. If the number of positive pixels N+ in a cluster with N pixels is smaller than the threshold

$$N+ / N < \textit{Threshold Ratio},$$

then their signs are flipped. If *Threshold Ratio = 0.5*, which is the default, the sign of pixels within each cluster will be determined by the majority of the pixels. The products of this step are the corrected dual detection maps.

5.6.14 Mosaic Modules: Mosaic Outlier

Command Line Equivalent: run_mosaic_outlier
Default Output Directory: <output_dir>/Outlier-mosaic
Depends On: Mosaic Interpolate

Important Notes: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must set *Use Outlier For Rmask* in the Mosaic RMask module and set the RMask *Fatal Mask Bit Pattern* in the Initial Setup module to use bit 1. Note that this is not the same as setting it to a value of 1. See §8.11: Fatal Mask Bit Patterns for more information.

PURPOSE

This module uses the Multiframe Temporal Outlier rejection method (see §8.2.2). For a stack of input images with good coverage, the mean and sigma at each pixel on the sky are computed. Pixels outside an asymmetric sigma envelope are flagged as outliers. If the data have good coverage (>10), this is the best option for finding outliers. For very shallow coverage, this method will not work well.

INPUT

Bottom Threshold: (float) Specifies the upper envelope in sigma to identify outliers from the pixel stack. Default value is 3.0.

Top Threshold: (float) Specifies the lower envelope in sigma for selecting outliers.

Min Pix Number: (int) Minimum number of pixels in the stack needed to estimate sigma.

Tile X(Y) Size: (int) Set these to a smaller size to avoid memory allocation problem if users have a very large mosaic. See the discussion of Tiling (§8.1) for details.

Outlier output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Outlier-mosaic.

COMMAND LINE INPUT

```
&MOSAICOUTLIERIN
 BOTTOM_THRESHOLD = 3,
```

```
 TOP_THRESHOLD = 3,
 THRESH_OPTION = 1,
 MIN_PIX_NUM = 3,
 TILE_XSIZ = 500,
 TILE_YSIZ = 500,
&END
```

In Global Parameters:

```
OUTLIER_DIR = Outlier-mosaic
```

THRESH_OPTION: (int; command line only) There are two options for this parameter: 1: "less aggressive" or 2: "more aggressive". The difference lies in the specific algorithm used to estimate sigma (see §8.2.2 for more details). We strongly recommend that all users use *THRESH_OPTION = 1.* For this reason, this parameter does not exist in the GUI, which uses option 1 by default.

OUTPUT

Outlier Output FITS files (*interp_*_outlier.fits*): The product of this step is an outlier map. The pixel value is the deviation of that pixel in the input image from the mean in the stack in terms of the number of standard deviations.

DISCUSSION

This module does Multiframe Temporal Outlier detection (§8.2.2) on the interpolated input images. The interpolated input frames are stacked, and for each pixel position in the interpolated grid, the trimmed mean and standard deviation of the pixel values in the stack is calculated. The pixel values outside of the user-specified asymmetrical multi-sigma thresholds are classified as outliers. This method detects both moving objects and radhits. It is not meant to be used in the cases of shallow coverage.

The thresholds - *Bottom Threshold* and *Top Threshold* - can be set to 0, which is the default. In this case, the decision of declaring a particular pixel an outlier is put off until running the Mosaic RMask module. The advantage of doing it this way is that the user can experiment with different thresholds for outliers in the Mosaic RMask module, without having to rerun the Mosaic Outlier module.

5.6.15 Mosaic Modules: Mosaic Box Outlier

Command Line Equivalent: run_mosaic_box_outlier
Default Output Directory: <output_dir>/BoxOutlier-mosaic
Depends On: Mosaic Interpolate

Important Notes: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must set *Use Box Outlier For Rmask* in the Mosaic RMask module and set the RMask *Fatal Mask Bit Pattern* in the Initial Setup module to use bit 3. Note that this is not the same as setting it to a value of 3. See §8.11: Fatal Mask Bit Patterns for more information.

PURPOSE

This module uses the Box Outlier rejection method to flag bad pixels (see §8.2.4). The method is designed to use both the temporal and spatial information like the Dual Outlier, but it uses the statistical analysis of the kind used by Multiframe Temporal Outlier. In the identification of temporal outliers, neighboring pixels are utilized to ensure the correct classification.

INPUT

Box Size X(Y) direction: (int) The X (Y) size of the box of neighboring pixels used for outlier detections.

Box Median Bias: (float) The computation of the mean and sigma for each pixel stack is done using a biased median. If there are N pixels in the stack, then the biased median is equal to the N/2 - Bias element of the stack:

$$biased\_median(I[k]) = I[N/2 - Bias]$$

Equation 5.4

Tile X(Y) Size: (int) Set these to smaller numbers to avoid memory problems with dealing with a large mosaic. See the discussion of Tiling for details (§8.1).

Box OutLier output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is BoxOutlier-mosaic.

COMMAND LINE INPUT

```
&MOSAICBOXOUTLIERIN
 BOX_X = 3,
 BOX_Y = 3,
 BOX_MEDIAN_BIAS = 1,
 TILE_XSIZ = 500,
 TILE_YSIZ = 500,
&END
```

In Global Parameters:

```
BOX_OUTLIER_DIR = BoxOutlier-mosaic
```

OUTPUT

Box Outlier Output FITS (*interp_*_box_outlier.fits*): The product of this step is an outlier map. The pixel value is the deviation of the pixel in the input image from the mean of that pixel in the stack, in terms of the number of sigma.

DISCUSSION

This module represents a more complicated dual spatial-temporal filtering (see §8.2.4). It extends the regular temporal outlier detection, which computes the statistics of the pixels in the stack for each mosaic pixel position, by including the pixels in the box region around that pixel in the statistics. This allows for detecting outliers even in the coverage = 2 case and, at the same time, provides a guard against detecting pixels inside bright point sources as outliers.

5.6.16 Mosaic Modules: Mosaic RMask

Command Line Equivalent: run_mosaic_rmask
Default Output Directory: <output_dir>/Rmask-mosaic
Depends On: Mosaic Outlier, Level, Mosaic Coverage

Important Notes: Setting an outlier scheme to be included in the RMasks does not guarantee that it will be set as "fatal" when running MOPEX. You must specify the RMask *Fatal Mask Bit Pattern* in the Initial Setup to correspond to the outlier rejection schemes that you want to set as "fatal". See the Discussion below, and §8.11: Fatal Bit Patterns for more information

PURPOSE

This module combines outlier information from a choice of different outlier detection methods into a single RMask by setting the following bits for each affected pixel: bit 0 (Single Frame Radhit detection), bit 1 (Multiframe Outlier detection), bit 2 (Dual Outlier detection), bit 3 (Box Outlier detection). Users can choose which of the available outlier detection schemes to include in the RMask.

INPUT

RM Thresh: (float) In most cases, the input BCD image is rotated, and its original pixel is re-sampled to a smaller size. This implies that any final re-sampled pixel could have fractional overlap with an original pixel. The threshold specifies the minimal fraction of an input pixel covered by the projection of outlier pixels to be marked in the RMask (if small, outliers will be fatter). The default value is 0.3.

Bottom Threshold, Top Threshold: (float) These two values are the lower and upper cutoffs used to select outlier pixels in the temporal rejection method. The outlier maps produced by the Mosaic Outlier module give the deviation of each pixel from the mean of the stack at that pixel. A pixel is flagged in the RMask when its value is less than *Bottom Threshold* * sigma or larger than *Top Threshold* * sigma. The default values are 10 for both.

Min Coverage, Max Coverage: (int) These two numbers are used to decide which method of outlier rejection is best for the dataset. The default values are 3 and 100.

If *coverage* >= *Min Coverage*, each temporal outlier map is projected onto the corresponding input image frame. The value of a projected pixel is equal to the sum of the overlap areas of this pixel with the outlier map pixels that were identified as outliers.

If *coverage* <= *Max Coverage* , each dual outlier map is projected onto the corresponding input image frame. The value of a projected pixel is equal to the sum of the overlap areas of this pixel with the dual outlier map pixels that were identified as outliers. Only dual outlier pixels with negative values are projected.

Refine Outlier: (int) A flag with value of either 1 or 0. If it is 1, it will trigger next parameter *Refine Outlier Threshold*, to be taken by the module. If it is 0, the module will not use the next parameter.

Refine Outlier Threshold: (float) A new threshold for pixels detected as temporal outliers based on their lack of detection as dual outliers. If *Refine Outlier* (above) is set to 1, this new threshold replaces *Top Threshold*.

Box Bottom Threshold, Box Top Threshold: (float) Similar to the above description for *Bottom Threshold, Top Threshold,* these two values are used for box outlier maps. Default values are 10.

Box Min Coverage: (float) If *coverage >= Box Min Coverage*, each box outlier map is projected onto the corresponding input image frame. The value of a projected pixel is equal to the sum of the overlap areas of this pixel with the outlier map pixels that were identified as outliers.

Mosaic Rmask output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is RmaskMosaic-mosaic.

Use Outlier for Rmask: Include the results of Multiframe Temporal Outlier detection in the final combined RMask.

Use Dual Outlier for Rmask: Include the results of Dual Outlier detection in the final combined RMask.

Use Box Outlier for Rmask: Include the results of Box Outlier detection in the final combined RMask.

COMMAND LINE INPUT

```
&MOSAICRMASKIN
 BOTTOM_THRESHOLD = 3,
 TOP_THRESHOLD = 3,
 RM_THRESH = 1,
 MIN_COVERAGE = 500,
 MAX_COVERAGE = 500,
 REFINE_OUTLIER = 1,
 REFINE_OUTLIER_THRESH = 10,
 BOX_BOTTOM_THRESHOLD = 10,
 BOX_TOP_THRESHOLD = 10,
 BOX_MIN_COVERAGE = 1,
&END
```

In Global Parameters:

```
RMASK_DIR = Rmask-mosaic
USE_OUTLIER_FOR_RMASK = 1
USE_DUAL_OUTLIER_FOR_RMASK = 1
USE_BOX_OUTLIER_FOR_RMASK = 1
```

OUTPUT

Outlier Output FITS (_rmask.fits*):** The output stack of mask images. The flagged pixels have positive bit values, with bit 0 for single frame, bit 1 for multi-frame rejection, bit 2 for dual outlier rejection, and bit 3 for box outlier rejection.

DISCUSSION

The module combines the results from the four different ways of rejecting outlier pixels. The outlier pixels in each interpolated image are traced back to the corresponding input image frame.

Bottom Threshold and *Top Threshold* can be set in both the Mosaic Outlier and Mosaic RMask modules. Their values can be increased going from Mosaic Outlier to Mosaic RMask, but cannot be decreased. The *Bottom Threshold* and *Top Threshold* parameters are often set to 3 in the pipeline namelists for IRAC and MIPS-24, and 3.5 for MIPS-70 and MIPS-160. However, these numbers are not optimal for all cases. For example, a value of 3 is too low for IRAC for coverage ~50 and leads to false detections. A value of 10 is preferable. Users need to experiment with these settings and find the optimal value for any specific data.

Refine Outlier Threshold: Multiframe Outlier detection detects outliers inside point sources, which can false in some (even many) cases. One can use the information saved in the dual outlier maps to prevent masking pixels inside what was determined by the Dual Outlier detection to be a real source. The positive pixels in the clusters in a dual outlier map (see §8.2) have been confirmed to have been detected in more than one frame. The chances are they belong to real sources. The Multiframe Outlier should be allowed to set a bit in the RMask for such a pixel if the deviation from the mean is significantly higher then would be required otherwise. To do so, set the *Refine Outlier Threshold* to a much larger value than the Multiframe Outlier *Top Threshold.*

Warning: setting an outlier scheme to be included in the RMask does not guarantee that the flagged pixels will be rejected by MOPEX. Outlier rejection is a three-step process. First the desired module for outlier rejection must be run, then Mosaic RMask must be told to use the results in the mask, and, finally, MOPEX must be told to set that particular code as

"fatal". For example, in order to use the results of Box Outlier Detection in the final mosaicking process, you must do the following:

1. Include the module Mosaic Box Outlier in the namelist:

2. Tell MOPEX to use the results of Mosaic Box Outlier to create the RMask by checking the Use Box Outlier for Rmask box

3. Set the RMask *Fatal Mask Bit Pattern* in the Initial Setup module to include the bit corresponding to the results from Mosaic Box Outlier, bit 3.

For more information on Fatal Mask Bit Patterns, see §8.11.

Note: When using the Drizzle option in the Mosaic Interpolate module, several of the mosaicking steps are run twice. Firstly, a linear interpolation is carried out so MOPEX can run the outlier rejection scheme. Once the outlier rejection masks (RMasks) have been created, MOPEX returns to the Mosaic Interpolate module and re-runs the interpolation with the Drizzle algorithm, masking out any pixels flagged in the RMasks. **When using the Drizzle interpolation scheme in Mosaic Interpolate, do not include the Mosaic Reinterpolate Module in the processing flow.**

5.6.17 Mosaic Modules: Mosaic Reinterpolate

Command Line Equivalent: run_mosaic_reint
Default Output Directory: <output_dir>/Reinterp-mosaic
Depends On: Mosaic Interpolate; Mosaic RMask

PURPOSE

This module runs the interpolation again using the RMasks. Only pixels flagged by the RMasks are re-interpolated.

INPUT

Mosaic Reinterpolate output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Reinterp-mosaic.

COMMAND LINE INPUT

```
&MOSAICREINTIN
&END
```

In Global Parameters:

```
REINTERP_DIR = Reinterp-mosaic
```

OUTPUT

ReInterp Stack (*interp_*.fits*): The new interpolated image stack, with Rmask-flagged pixels having been updated.

ReInterp coverage (*interp_*covg.fits*): The corresponding coverage map for each interpolated image in the stack.

ReInterp Sigma (*interp_unc_*sigma.fits*): The corresponding uncertainty files for each image in the stack.

DISCUSSION

Warning: Do not use this module when using the Drizzle or GRID interpolation schemes in the Mosaic Interpolate Module. See §5.6.8: Mosaic Interpolate for more information.

The process of reinterpolation is performed after the various outlier masks are combined into the RMasks by the Mosaic RMask module. Only the pixels masked in the RMask images are recomputed, which makes it much faster than doing interpolation from scratch again.

Reinterpolation uses the same interpolation scheme and other processing parameters as used by the original interpolation. All this information is recorded in the headers of the interpolated images. The newly interpolated images become the input to later modules.

5.6.18 Mosaic Modules: Fix Coverage

Command Line Equivalent: run_fix_coverage
Default Output Directory: <output_dir>/Interp-mosaic
Depends On: Mosaic Interpolate (or Mosaic Reinterpolate)

PURPOSE

After all outlier pixels have been identified in the previous modules, there is the option to suppress usage of pixels that have very small coverage (number of repeated observations at the same point of sky). Pixels whose usage is suppressed have their coverage set to 0 in new coverage maps.

INPUT

Min Single Coverage: (float) The minimum coverage for an individual pixel. The default value is 0.95.

Min Block Coverage: (float) The minimum average coverage in the block of 9 pixels centered on a given pixel for it to be retained. The default value is 0.83

COMMAND LINE INPUT

Within mosaic.pl, no parameters are set for this module, so default values are used for *Min Single Coverage* and *Min Block Coverage*

OUTPUT

Fix Coverage Images (*fixed_covg_*.fits*): The stack of corrected coverage maps.

DISCUSSION

Two thresholds are used for each coverage map pixel, one - *Min Single Coverage* - for the pixel value alone and the other one - *Min Block Coverage* - for the sum of the values of the block of 9 pixels centered on the pixel in question (see Figure 5.1 below). The following logic is used:

IF *C5* < *Min Single Coverage* THEN C5 = 0;

IF (C1 + C2 + C3 + C4 + C5 + C6 + C7 + C8 + C9) < 9 * *Min Block Coverage* THEN C5 = 0.

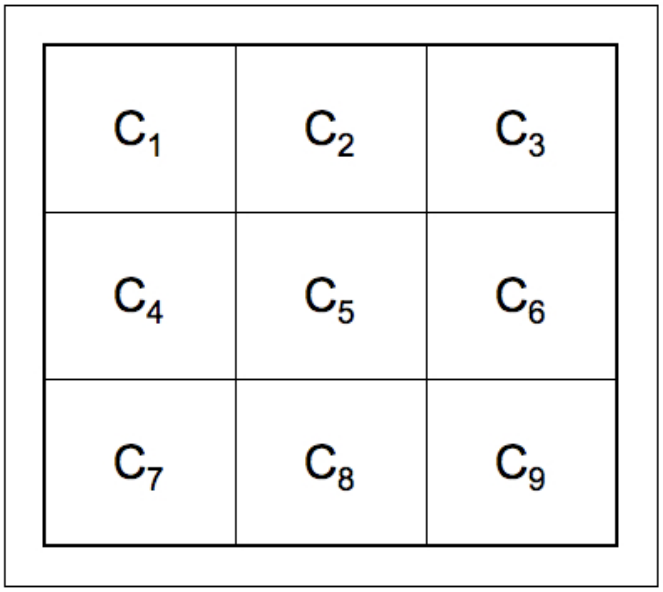


Figure 5.1: Coverage for the central pixel (C_5) is fixed based on the two thresholds*: Min Single Coverage* for C_5 and *Min Block Coverage* for the sum of all nine C_is.

5.6.19 Mosaic Modules: Mosaic CoAdder

Command Line Equivalent: run_mosaic_coadder
Default Output Directory: <output_dir>/Coadd-mosaic
Depends On: Mosaic Interpolate (or Mosaic Reinterpolate)

Important Notes: There are several coaddition schemes available, and all but one are set in this module. The final one, *Create Absolute Minimum Mosaic*, is set in the Mosaic Settings module.

PURPOSE

The interpolated images are co-added to create one mosaic image. The pixel values of the mosaic image are equal to the averaged interpolated pixel values, with the option to weight them with the integration time if needed. (Generally one should not weight by uncertainties when sources are present.) The interpolated uncertainty images are also co-added into one uncertainty mosaic image following the standard assumption of additive variances. The coverage maps are co-added into a single mosaic coverage map.

INPUT

Tile Max X, Y: (int) The X (Y)-size of the tile for performing the co-addition. This should be reduced if computer memory is an issue (see §8.1).

Integration Time Weighted Coadd: Check this box to weight the coaddition by exposure time. MOPEX searches for the header keyword indicated, the default is EXPTIME.

Mosaic Co-Adder output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Coadd-mosaic.

Keep coadded Tiles: If checked, the coadded tiles will be saved, even if Delete Intermediate Files was turned on in the Mosaic Settings module. Recommended if you are planning on subsequently running APEX.

Sigma Weighted Coadd: Check this box to use the uncertainty-weighted coaddition option (see below for details). **Not Recommended.**

Create UNC Mosaic: If checked, the mosaic of uncertainties is created, propagating from the individual input (or S/N Estimator) uncertainties.

Create Standard Deviation Mosaic: If checked, a different flavor of mosaic uncertainty image is created. The pixel values of this image will be equal to the standard deviation of the interpolated image pixels in the stack.

Create Outlier Mosaic: If checked, a mosaic is created of the temporal outliers detected by the Mosaic Outlier module.

Create Dual Outlier Mosaic: If checked, a mosaic is created of the dual outliers detected by the Mosaic Dual Outlier module.

Create Median Mosaic: If checked, a quicklook mosaic is created using the median of each interpolated pixel in the stack. **Flux is not conserved.**

COMMAND LINE INPUT

```
&MOSAICCOADDIN
 TILEMAX_X = 1000,
 TILEMAX_Y = 1000,
 INTEG_TIME_KWD = 'EXPTIME',
&END
```

In Global Parameters:

```
COADDER_DIR = Coadd-mosaic
keep_coadded_tiles = 1
sigma_weighted_coadd = 0
create_unc_mosaic = 1
create_std_mosaic = 0
create_outlier_mosaic = 1
create_dual_outlier_mosaic = 1
run_median_mosaic = 0
```

INTEG_TIME_KWD: This is the equivalent of checking the *Integration Time Weighted Coadd* box in the GUI. If *`INTEG_TIME_KWD = 'EXPTIME'`* is set in the parameter block, intergration time weighted coadd will be used.

OUTPUT

Tile BCD file (*coadd_tile_bcd.txt*): This text file lists the input images which contribute to each output tile. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Tile table (*coadd_tiles.tbl*): A table listing the tiles, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Tile uncertainty list (*coadd_Tile_*_Unc.fits*): The stack of tiled uncertainty images.

Mosaic uncertainty file (*coadd_Tile_*_Std_Unc.fits*): The stack of tiled standard deviation images.

The tile specification files corresponding to the other optional mosaics requested in the Global Parameters section of the namelist (e.g. the outlier mosaic) are created by the next module, Mosaic Combine, but are written to the output mosaic directories (e.g. *Outlier-mosaic/*).

DISCUSSION

The co-addition can be performed on Tiles (see §8.1), to address computer memory considerations. The Mosaic CoAdder module performs the co-addition on the individual tiles, and the Mosaic Combine module then combines the tiles into a single mosaic. Even if only a single tile is used, Mosaic Combine still needs to be run, in order to produce the expected products.

There are several weighting schemes available. The interpolated images are normally combined using straight averaging. Exposure time weighting is also available. The co-added uncertainty image is computed regardless of whether interpolated uncertainty images are used for weighting, as long as the uncertainty images are given as an input to the program.

The coverage mosaic is always the number of good pixels in the final stack. The total exposure time for a pixel (when time-weighting) is not available as a mosaic, but can be found per tile in the output_dir/Coadd-mosaic directory. (If there's only one tile, it is like the final mosaic.)

Straight Average: (default) If none of the weighting alternatives are selected, then the co-added pixel value O is given by the following expression:

$$O = \frac{\sum_j c_j I_j}{C}$$

Equation 5.5

Here I_j is the value of the interpolated pixel in image j, and c_j is the pixel value of the corresponding coverage map. Co-added uncertainty U = W-1 is given by the following expression:

$$W = \left(\sqrt{\sum_j \frac{c_j}{\sigma_j^2}} \right)$$

Equation 5.6

where σ_j is the pixel value of the corresponding uncertainty image, and co-added coverage C is simply the sum of the individual input coverages:

$$C = \sum_j c_j$$

Equation 5.7

Exposure Time Weighted Average: If *Integration Time Weighted Coadd* is selected then MOPEX weights the co-added pixels by the exposure time given in the FITS header keyword specified. For Spitzer data, the integration time is given by the keyword EXPTIME.

Sigma Weighting: This option is not recommended when sources are present. If *Sigma Weighted Coadd* is selected, then a co-added pixel value O is given by the following expression:

$$O = \frac{\sum_j \frac{c_j I_j}{\sigma_j^2}}{W^2}$$

Equation 5.8

Median Mosaic: If the *Create Median Mosaic* box is checked then the pixels are co-added by taking the median value of the stack. **Warning: Flux is not conserved.** This option is useful for a quick-look mosaic, but should never be used for science.

Absolute Minimum Mosaic: If the *Create Absolute Minimum Mosaic* box is checked in the Mosaic Settings module then the output mosaic pixel value is set to the minimum input pixel value in the stack. The corresponding uncertainty pixel is set to the value of the pixel in the interpolated uncertainty image corresponding to the interpolated image with the smallest absolute value pixel. This option is useful for outlier rejection

Standard Deviation Mosaic: If the Create Standard Deviation Mosaic box is checked then a different flavor of mosaic uncertainty image is created. The pixel values S of this image are equal to the standard deviation of the interpolated image pixels in the stack:

$$S = \sqrt{\frac{N\sum_j c_j T_j I_j^2 - \left(\sum_j c_j T_j I_j\right)^2}{(N-1)N^2}}; N = \sum_j c_j T_j$$

Equation 5.9

The variable T is the exposure time.

5.6.20 Mosaic Modules: Mosaic Combine

Command Line Equivalent: run_mosaic_combiner

Default Output Directory: <output_dir>/Combine-mosaic
Depends On: Mosaic CoAdder

PURPOSE

This module combines the co-added tiles into a mosaicked image. Even if the co-addition has been done in a single tile, this module must be run in order to produce the expected outputs.

INPUT

The choices of output products for this module are made in either the Mosaic CoAdder or Mosaic Settings module.

Mosaic Combiner output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Combine-mosaic.

COMMAND LINE INPUT

```
COMBINER_DIR = Combine-mosaic
```

OUTPUT

The output of this module will include a number of the following, depending on the output selected in Mosaic Settings and Mosaic CoAdder, and the outlier rejection scheme used.

Outlier Tile BCD File (*Outlier-mosaic/tile_bcd.txt*): This text file lists the outlier images that contribute to each output tile of the outlier mosaic. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Outlier Tile Table (*Outlier-mosaic/tiles.tbl*): A table listing the outlier tiles, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Mosaic Outlier File (*mosaic_outlier.fits*): The combined, co-added mosaic of temporal outlier detections.

Dual Outlier Tile BCD File (*DualOutlier-mosaic/tile_bcd.txt*): This text file lists the dual outlier images that contribute to each output tile of the outlier mosaic. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Dual Outlier Tile Table (*DualOutlier-mosaic/tiles.tbl*): A table listing the dual outlier tiles, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Mosaic Dual Outlier File (*mosaic_dual_outlier.fits*): The combined, co-added mosaic of dual outlier detections.

Median Tile BCD File (*Coadd-mosaic/coadd_median_tile_bcd.txt*): This text file lists the input images that contribute to each output tile of the median mosaic. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Median Tile Table (*Coadd-mosaic/coadd_median_tiles.tbl*): A table listing the tiles of the median mosaic, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Mosaic Median File (*median_mosaic.fits*): The output median mosaic.

Mosaic Image File (*mosaic.fits*): The output mosaic FITS image. This output image will change depending on the mosaic type requested in the Mosaic Settings and Mosaic CoAdder modules. The default is described in §5.6.19.

Mosaic Uncertainty File (*mosaic_unc.fits*): The output uncertainty mosaic.

Mosaic Standard Deviation File (*mosaic_std.fits*): The output standard deviation mosaic.

Mosaic Coverage File (*mosaic_cov.fits*): The output coverage mosaic.

DISCUSSION

After the Mosaic CoAdder module produces individual tiles, this module stitches together all tiles into a big mosaic. Even if the co-addition was done in a single tile, this module must still be run in order to produce the expected outputs.

Not all of the output files above will be produced by Mosaic Combine. Many will only be produced if requested in either the Mosaic Settings or Mosaic CoAdder modules.

5.6.21 Mosaic Modules: Mosaic MedFilter

Command Line Equivalent: run_mosaic_medfilter
Default Output Directory: <output_dir>/Combine-mosaic
Depends On: Mosaic Combine

PURPOSE

This module uses a median filter to create a background-subtracted version of the final mosaic.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

SExtractor background filter size: (int) Median filter box size for the Panels when the *Use SExtractor background estimation* option is turned on. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Use SExtractor background estimation: Checking this box invokes the Sbkg background estimate based on SExtractor. Leaving it unchecked causes MOPEX to use the default Median estimate.

COMMAND LINE INPUT

```
&MOSAIC_MEDFILTER
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 500,
 Sbkg_Filt_Size_for_Med = 1,
 Use_Sbkg_for_Med = 1,
&END
```

OUTPUT

Mosaic Minback file (*mosaic_minback.fits*): the background subtracted image saved in the output directory of Mosaic Combine.

DISCUSSION

This module is optional for mosaicking images. It is an optional input to APEX One Frame, if the global background subtraction option is chosen.

5.6.22 Mosaic Modules: Create RMask Mosaic

Command Line Equivalent: create_rmask_mosaic
Default Output Directory: <output_dir>/Rmask-mosaic
Depends On: Mosaic RMask

Important Notes: This module treats the RMasks as normal images, so co-adding them using this module causes all bit information to be lost in the output mosaic.

PURPOSE

This module creates an output mosaic of the RMask images.

INPUTS

There are no user-defined input parameters for this module.

COMMAND LINE INPUT

There are no user-defined input parameters for this module.

OUTPUT

RMask Mosaic (*mosaic.fits*): this single FITS file is a mosaic or the RMasks, showing all of the outlier pixels rejected by MOPEX for this dataset. It is written to the RMask output directory, so it does not overwrite the primary mosaic output, despite the similar name.

DISCUSSION

The RMask images are interpolated and co-added into a single mosaic. **However, they are treated as regular images, so that all the bit information is lost in the process.**

5.6.23 Mosaic Modules: Make Array Correction Files

Command-Line Equivalent: make_array_corr_files
Default Output Directory: <output_dir>/Array_Corr_Data
Depends On: Mosaic Settings

Important Notes: For use with IRAC data only. Only required if you are analyzing sources with a spectral energy distribution that is "blue", that is, falling with wavelength through the IRAC passbands.

PURPOSE

Creates the Array Location-Dependent Correction frames for each of the BCD files used in making the mosaic.

INPUT

Inputs are an Array Location-Dependent Correction file (*Array Correction Frame*) and a Pixel Area distortion file (*PixArea Frame*) in the Mosaic Settings module.

COMMAND LINE INPUT

See next section.

OUTPUT

Array Correction frames (corr_*.fits): The Array Correction frames corresponding to the BCDs. They are written to the Array_Corr_Data output directory.

DISCUSSION

For use with IRAC data only. This module creates the Array Location-Dependent Correction frames for each of the input BCDs to allow users to correct their photometry when observing blue sources on the final mosaic. In this case, "blue" means anything with a spectral energy distribution that is falling through the IRAC passbands and, in practice, includes most galactic and many extragalactic sources. For more information on the array location-dependent correction, and how to apply it to your photometry, see the IRAC Instrument Handbook. This is not an issue with MIPS data.

5.6.24 Mosaic Modules: Make Array Correction Mosaic

Command Line Equivalent: make_array_corr_mosaic
Default Output Directory: <output_dir>/Combine-mosaic
Depends On: Mosaic Settings

Important Notes: For use with IRAC data only. Only required if you are analyzing sources with a spectral energy distribution that is "blue", that is, falling with wavelength through the IRAC passbands

PURPOSE

Creates a mosaic of the Array Location-Dependent Correction frames for correcting IRAC photometry extracted from a mosaic.

INPUT

Requires the Array Correction frames - see previous section.

COMMAND LINE INPUT

Example settings for IRAC1:

```
make_array_corr_files = 1
make_array_corr_mosaic = 1
ARRAY_CORR_IMAGE = cal/ch1_photcorr_rj.fits
ARRAY_PIXAREA_IMAGE = cal/ch1relpixarea.fits
```

For further discussion, see the file starting with README_arraycorr in the Mopex download readme sub-directory.

OUTPUT

Array Correction mosaic (mosaic_arraycorr.fits): The mosaic of the Array Correction frames.

DISCUSSION

For use with IRAC data only. This module creates a mosaic of the Array Correction frames, with the same spatial extent and location as the mosaic of BCDs.

The Array Correction frames get header information from the input BCD's as described in the previous section. The modified Array Correction frames are then mosaicked together, to create an Array Correction mosaic that matches the spatial boundaries of the science mosaic.

The final product allows users to correct their photometry for array location dependence issues when observing blue sources. In this case, "blue" means anything with a spectral energy distribution that is falling through the IRAC passbands and, in practice, includes most galactic and many extragalactic sources. For more information on the array location-dependent correction, and how to apply it to your photometry, see the IRAC Instrument Handbook. This is not an issue with MIPS data.

Chapter 6. Point Source Extraction (APEX)

6.1 Overview

APEX is the inbuilt Astronomical Point source EXtractor for MOPEX. It takes either a list of input images and performs Multiframe point source extraction, or a single image (e.g. a mosaic) and carries out Single Frame point source extraction. The APEX User List pipelines allow users to input their own list of source positions rather than using the Detect module to automatically search for sources in the input frame(s). APEX QA can then be used to create residual mosaics by subtracting the fitted point sources from the input images to check the results of profile fitting. All APEX flows can do both profile fitting and aperture photometry.

6.1.1 APEX Multiframe (apex.pl)

For Multiframe point source extraction, the extraction is carried out simultaneously on a stack of input images to give a single position and flux density estimate for each detected source. Initial point source detection is carried out on the co-added images to ensure the best signal to noise ratio. The typical components of point source detection are non-linear matched filtering and image segmentation. They produce a detection list with the potential point source position estimates. Subsequent point source estimation is performed by Point Response Function fitting (PRF fitting; see §8.6) of the potential point sources from the detection list. The fitting is performed simultaneously in all the input images (hence the term "Multiframe"). Passive deblending, i.e. the separation of detected clusters into individual sources, is done by default. Active deblending based on the quality of the fit is available as an option. The advantage of APEX Multiframe over APEX Single Frame is seen in data with intra-pixel variability (e.g. IRAC data) or data where the PRF varies across the array (IRAC). Aperture photometry is always carried out on the mosaicked image, regardless of whether you run Multiframe or Single Frame.

6.1.2 APEX Single Frame (apex_1frame.pl)

For single-frame point source extraction, the detection and estimation as described above are performed on a single input image (e.g. a mosaic).

6.1.3 APEX User List Multiframe (apex_user_list.pl)

APEX User List allows the user to input lists of object positions instead of using the Detect module to automatically find sources in the field. The code is still experimental. With some

limitations discussed below, APEX User List will do PRF fitting and aperture photometry on the input list. With the default APEX User List parameters, the positions of objects will be allowed to move within a small radius as part of the fitting. PRF-fitted and aperture fluxes will be for that (possibly shifted) position. Aperture photometry on fixed positions is available on the command line. **Note that APEX User List will not search for additional objects, so if any are missing then proper deblending cannot be done,** i.e if an observed source is comprised of two blended sources, you must include two sources at that point in the User List input table, else APEX will attempt to fit the blend as a single source.

In User List mode, the user's input table becomes the detection list. This is set in the APEX User List multiframe/single frame Settings module with the parameter *User List Fixed Positions Table File Name*. It is also possible to interactively create the user list table by loading an image into MOPEX and clicking on the sources that you wish to extract. See §8.10 for full instructions on how to interactively create a User List table.

Nearby sources for each input source are identified and fitted in the Source Estimate module using the parameter *Angular Distance*. In User List mode, the parameters *Fitting Area X* and *Fitting Area Y,* which let the user define the fitting region, are **required**. Positions will move to "best-fit" values. Sources that are successfully fit but drift outside *Max Shift X*, *Max Shift Y* data pixels from the input position are flagged with the letter "O" (for "outside") in the deblend column of the output table. Typically, the Select module will include a condition to filter out sources flagged in this way. The format of the output extract table is otherwise the same as the usual APEX output.

For APEX User List Multiframe, the creation of a detection map is required: include the Detection Map module in the pipeline. This creates "detectionmap.tbl" in the output directory. Once it is created, this can be turned off for subsequent runs. Also note that in this case the *Input Type* in the Source Estimate module should be set to *Detection Map*. These steps are needed to track the input source positions through the image stack.

In the APEX namelists, both the Detect MedFilter and Gaussnoise should be turned on to provide the ability to use background RMS for the point-source SNR.

The input source positions are sorted first in Dec in ascending order. Bins are created in this sorted list using bin size = 2 * *Angular Distance*. The sources are again sorted in RA within each bin. Each input source position is then searched to find its distance from the other sources in the current bin and 2 adjacent bins on both sides. If this distance < *Angular Distance*, then the source is flagged as a nearby source and included in the fitting.

6.1.4 APEX User List Single Frame (apex_user_list_1_frame.pl)

For single frame mode, the point source extraction is carried out as above, only this time it is carried out on the mosaic rather than the stack of BCDs.

6.1.5 Aperture Photometry on Fixed Positions

This can be done on the command line in Single Frame mode on mosaics by using the namelists provided in the cdf directory with "aponly_fixed" in the filename.

6.1.6 Residual Image Creation (apex_qa.pl)

APEX QA creates residual images to test the quality of PRF fitting by subtracting the fitted point sources from the input image(s). Input requirements differ depending on whether you are running Multiframe or Single Frame mode; see §6.6.

6.2 Basic Input Requirements

In order to carry out point source extraction, MOPEX will expect some or all of the following files as input. See the sections on Input for more information on the format of these files.

- A text file containing a list of input data images or a single input image (e.g. Spitzer BCD frames; required)
- A text file containing a list of input uncertainty images or a single uncertainty image (optional)
- A text file containing a list of input status masks images or a single mask image (optional)
- A PRF image file or PRF map (e.g. *PRF.fits*; required)
- A permanently damaged pixel mask image (pmask; optional)
- A "namelist" file specifying the parameters to be used for mosaicking (Command-Line MOPEX only; required)
- A user-defined source list file (User List mode only), defining the positions of the sources to be extracted. See §8.10 for more information.

6.3 Running APEX

To load a standalone APEX pipeline (i.e. without appending to an existing Mosaic pipeline), either start from a MOPEX template namelist (e.g. *File > New APEX multiframe Pipeline*), load an existing APEX namelist file (*File > Read Name List*) or load an empty flow and insert an APEX pipeline from there (*File > New Empty MOPEX Pipeline*, then go to e.g. *Insert APEX single frame* at the top of the flow window and choose *Empty Pipeline*). If you choose the last option, MOPEX will load an empty APEX flow to which you can add and subtract modules, and modify the input parameters.

Many users will wish to append an APEX pipeline to the end of a Mosaic pipeline to use the previously-generated mosaic file as input. We generally encourage this, as the mosaicking within APEX does not carry out outlier rejection. To do this, click on the e.g. *Insert APEX User List multiframe* rectangular button at the top of the Overlap window. This will pull up a dialogue box giving you three options. Choose *From File* to load a previously-created namelist from file, *From Template* to start from one of the inbuilt APEX templates, and *Empty Pipeline* to open an empty APEX flow that you can build up by hand. If you have previously added and deleted an APEX pipeline from a particular flow, the last option will instead read *Re-add Pipeline*.

For information on running APEX on the command line, see Chapter 9.

6.3.1 Using a Previously-Generated Mosaic with APEX

As mentioned above, creating the mosaic for source detection within the APEX Multiframe pipeline has a major disadvantage in that it doesn't perform any outlier rejection, often leading to the detection of a number of spurious sources. We recommend creating the mosaic file using the Mosaic pipeline, and then using this as input into the APEX Multiframe flow. Do this by running the Mosaic pipeline with the *Keep coadded Tiles* option checked in the Mosaic CoAdder module, then appending the APEX Multiframe pipeline onto the end of the Mosaic flow. Remove the Fiducial Image Frame, Mosaic CoAdder and Mosaic Combine modules from the APEX pipeline and run it through to the end. APEX will automatically take the mosaic file from the Mosaic pipeline as input.

6.4 APEX Processing Stages

The basic steps of point source extraction with APEX are as follows:

Mosaicking

In APEX Single Frame, the processing is done on a single frame, so this step is not carried out. In APEX Multiframe, the first step is to mosaic the input images to create a single combined image that can be used for source detection (unless this has been done in a pre-pended Mosaic pipeline).

Background Subtraction

The MedFilter module performs background estimation in the input images, and outputs background-subtracted images. Background subtraction is performed on both the input images and the co-added images. The module is run twice. The first time (module Detect MedFilter), it is applied to the mosaic image, or the co-added tiles to subtract the background in preparation for point source detection. The second time (module Extract MedFilter), it is applied either to the input images (APEX Multiframe) or to the mosaic image (APEX Single Frame) for the purpose of subsequent point source fitting. The settings for *Window X*, *Window Y*, and *Outliers / Window* for the first run for detection should be more aggressive (smaller values) than for the fitting. There is also the option to use the faster Sbkg background fitting method, like that used by SExtractor.

Noise Estimation

There are two options available for noise estimation, set in the APEX Settings module. The default is to check the *Use Data Uncertainties for PRF-fitted SNR* option, which uses the data uncertainties to calculate the signal-to-noise ratio. The alternative is to leave the box unchecked, which will call the Gaussnoise module to estimate the background fluctuations in the images. It finds the 68-percentile range of the pixel values in a sliding window, which is defined to be the noise estimate. It can output the ratio of the input image to the noise, which is saved as a signal-to-noise ratio (SNR) image.

The results of noise estimation are output in the final extract table as the SNR.

Non-Linear Filtering

This step is performed to improve detectability of the point sources. The filtering is conceptually similar to a convolution of the input image by the PSF, which is commonly done during source extraction. It can be shown that using the ideas of maximizing SNR in the image that a filter can be derived to estimate the probability at each pixel of having a point source above the noise (see the document entitled "Bayesian Estimation of Point Source Probability" at the website).

The Point Source Probability module performs non-linear matched filtering and outputs point source probability (PSP) images. It is applied to the co-added images. This is an optional step.

Point Source Detection

A complex algorithm involving non-linear matched filtering and image segmentation has been developed. This algorithm is similar to that used by the SExtractor software. The purpose of filtering is to reduce fluctuations in the background noise and to enhance point source contributions. The design of the filtering is based on maximizing the SNR in the input images but APEX can also do detection on a number of types of images in addition to the filtered images.

Filtered images undergo a process of image segmentation. Contiguous clusters of pixels with values greater than a specified threshold are identified. The threshold is defined in terms of a robust estimate of the background fluctuations in the filtered, background-subtracted image. If the number of pixels in a cluster is smaller than a user specified threshold, the cluster is rejected. This is an additional guard against cosmic ray affected pixels and background noise fluctuations. If the number of pixels is greater than another user-specified threshold, the cluster is subject to further segmentation by progressively raising the threshold. This sub-segmentation process has several user-configurable parameters. The centroids of these final clusters are then calculated and stored in the detection list.

In APEX User List pipelines, this step is skipped in favor of using the user-input source list table.

Point Source Estimation

Final point source position and photometry estimation are performed at this stage for all point source candidates on the detection list. The position and flux of the point sources are fit simultaneously. Each detection comes with its own fitting area of a rectangular shape in the vicinity of the detection in each input image. The data in the fitting area are fit with the Point Response Function (PRF; see §8.7 and §8.6). In the multi-frame mode, the fitting is done simultaneously in all the input images containing the point source.

PRF fitting can be performed simultaneously for more than one point source. This process is called de-blending. The software is designed to perform two kinds of de-blending: passive and active. For passive de-blending the detected point sources, determined to be in a close proximity from one another so that their PRFs overlap, are fit simultaneously. The fitting areas are combined in this case. The classification of the detections as candidates for passive de-blending is done at the point source detection step. Passive deblending has been proven to be an essential component of point source extraction.

Active de-blending is a process that starts when a single point source is used to fit a single fitting area. The software can be configured to fit the same fitting area with more than one point source, if fitting with a single point source fails. Active de-blending is a very sensitive process and is currently under development.

In order to fit the data, a modified Simplex algorithm is employed. The Simplex algorithm does not use derivatives of the functions involved in minimization. This is a desirable feature since the transformation from local coordinates to global coordinates can be a very complicated function of its arguments.

In some cases, the PRF of a point source is not constant across all parts of the detector array. APEX allows users to use a variable PRF by creating and specifying a PRF map. The PRF map is a file that splits the array up into areas of constant PRF, so that a single PRF image can be used for sources that are detected within each area. A PRF map therefore identifies areas of constant PRF, and matches each area to an individual PRF file that should be used for point sources detected within that area.

The product of the Point Source Estimation stage is an extraction table that has over 20 columns, including point source positions in the sky and mosaic coordinates, fluxes, positional and flux uncertainties, SNR, etc.

Aperture Photometry

For each extracted point source, aperture photometry can be calculated. The user can specify an arbitrary number of aperture values. The results are appended to the output of the point source estimation. The aperture photometry is calculated using an exact algorithm to compute a fractional aperture-pixel overlap.

Figure 6.1 gives an overview of the APEX processing stages.

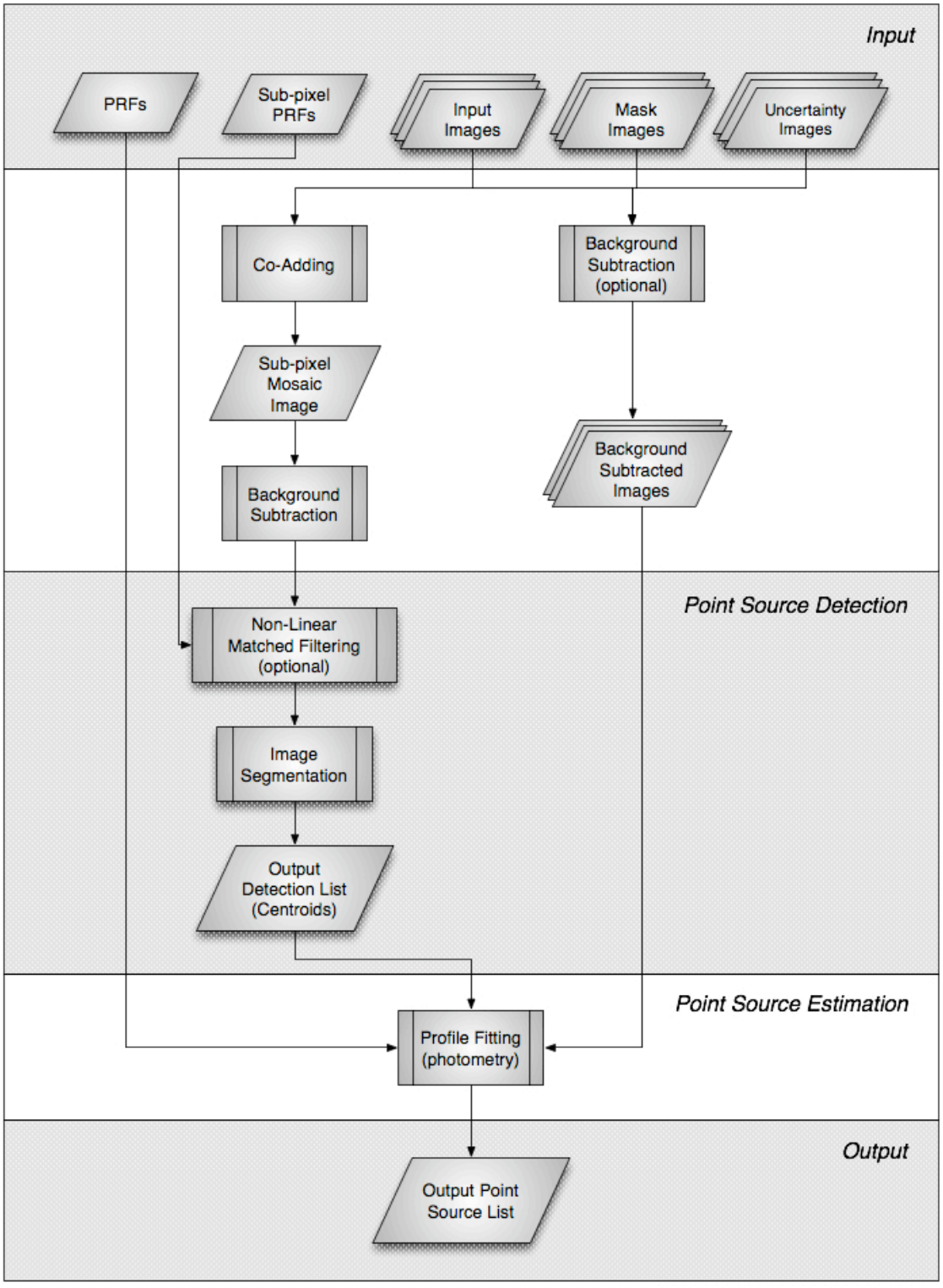


Figure 6.1: Overview of the main processing stages in APEX..

6.5 APEX Modules

The following pages contain full descriptions of the modules that are available in APEX Multiframe, APEX Single Frame and APEX User List (both Multiframe and Single Frame). In some cases, the order of execution of the modules, and the available parameters in each module will vary according to the flavor of the pipeline. The parameters that change between pipelines are clearly marked with e.g. "APEX Multiframe only". For information on how to turn the modules on and off in the image processing, please see §2.2. For suggestions on which modules to use, see §2.5: Which Modules Should I Choose?

For information on the modules available in APEX QA, see §6.6 APEX QA Modules.

Module listing:

- Initial Setup
- APEX Settings (actual name depends on pipeline)
- S/N Estimator
- Fiducial Image Frame
- Mosaic Interpolate
- Mosaic CoAdder
- Detect MedFilter
- Point Source Probability
- Gauss Noise
- Bright Detect
- Detect
- Mosaic Combiner
- Extract MedFilter
- Fit Radius
- Select Detect
- Detection Map
- Source Estimate
- Aperture Photometry
- Select

6.5.1 APEX Modules: Initial Setup

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: None

Relevant Pipelines: All

PURPOSE

Set up the input and output files for the loaded flow.

INPUT

Image Stack File: (Multiframe pipelines) The text file containing the list of input data files for the pipeline. Typically these are the **bcd.fits* files that you download from the Spitzer data archive. See Chapter 3 and Table 3.1 for more information on the input file requirements and format.

Input File Name: (Single Frame pipelines) The name of the single image that you wish to use as input. For Spitzer data, this is usually the *mosaic.fits* file output by the Mosaic pipeline.

Sigma File Name: (Single Frame pipelines) The name of the uncertainty file that corresponds to the *Input File Name*. This is usually the *mosaic_unc.fits* file output by the Mosaic pipeline.

Coverage Map: (Single Frame pipelines) The name of the coverage map file that corresponds to the *Input File Name*. This is usually the *mosaic_cov.fits* file output by the Mosaic pipeline.

Output Directory: The directory that you want to set as your output directory for this run.

Multi-processing Mode: Switch to use multi-processing. See Discussion.

Processors for Multi-processing: Number of processors to use. See Discussion.

Optional Input and Mask Files: (Multiframe Pipelines only)

Sigma List File: The text file containing the list of uncertainty images that correspond to the files listed in *Image Stack File* (usually the **bunc.fits* files; see Table 3.1 for more information).

DCE Status Mask List: The text file containing the list of mask images that flag temporarily-bad pixels in the files listed in *Image Stack File*. There should be a mask file

for each of the input data files (these depend on the instrument, but will be labeled something like **msk.fits*; see Table 3.1 for more information).

Fatal Mask Bit Pattern: This is the bit pattern that defines which type of flagged problems that you wish to set as fatal when combining the input images. See §8.11 for more information. There is a separate Fatal Bit Pattern for each type of mask that is being used as input (i.e. for the DCE Status masks, the RMasks and the PMask).

Rmask List File: The text file containing a list of outlier mask images. This option allows you to specify list of RMask images (e.g. those created by the Mosaic pipeline) to reject pixels in the input images that have been identified as outliers.

Pmask FITS File: The permanently-damaged pixel mask for the detector. This is a single FITS image, flagging the permanently damaged pixels in the instrument arrays. These files are stored in the *<mopex_dir>/cal/* directory. There is a single PMask for each of the MIPS and IRS channels, but the IRAC ones have changed with time. See the file *<mopex_dir>/pmasks.README* for information on which PMask is applicable to your data. Note that the date that the data were taken is given in the DATE_OBS header keyword (**not** DATE - this is the date the file was written).

FIF file: The Fiducial Image Frame file. This is an optional input file, as it is often generated from the input data using the Fiducial Image Frame module, or taken from a previous pipeline (e.g. Mosaic). There are, however, a number of cases in which you might wish to use a user-specified FIF (e.g. if you are trying to match the field of view to the other data channels).

COMMAND LINE INPUT

```
IMAGE_STACK_FILE_NAME = <working_dir>/imagelist.txt
INPUT_FILE_NAME = <working_dir>/mosaic.fits
SIGMA_FILE_NAME = <working_dir>/mosaic_unc.fits
COVERAGE_MAP = <working_dir>/mosaic_cov.fits
OUTPUT_DIR = <working_dir>/output
SIGMALIST_FILE_NAME = <working_dir>/sigmalist.txt
DCE_STATUS_MASK_LIST = <working_dir>/masklist.txt
DCE_Status_Mask_Fatal_BitPattern = 32544
RMASK_LIST = <working_dir>/rmasklist.txt
RMask_Fatal_BitPattern = 7
PMASK_FILE_NAME = <mopex_dir>/cal/sep07/sep07_ch1_bcd_pmask.fits
PMask_Fatal_BitPattern = 32767
FIF_FILE_NAME = <working_dir>/FIF.tbl
verbose = 1
```

```
NICE = 1
save_namelist = 1
```

Verbose, *NICE* and *save_namelist* are only available on the command line. See §9.4.2.3 for more information.

OUTPUT

None

DISCUSSION

This module sets up the input files and top-level output directory for the first pipeline in the flow. Subsequent pipelines that have been inserted into the flow will use the output from the previous one (i.e. if you have inserted Mosaic and APEX into your reduction flow, Mosaic will use the Initial Settings files as input, but APEX will use the modified files output by Mosaic).

NEW Note that with version 18.5.0 one can use multiprocessing to speed up MOPEX tasks. The tasks Overlap, Mosaic, Apex (Multi) and Apex User List (Multi) now allow multiprocessing of some steps. In the GUI, this is set in Initial Setup, Multi-Processing Mode. The options are "on", "off", and "manual". The default is "on" -- this grabs 3/4 of the available processors. Setting "manual" lets the user set the number of processors used. You should see speed-ups of at least 2x.

If using command-line, add these lines at the top of your namelist:
do_multiprocess = on | off | manual default = off
ncpu_multiprocess = 1 default = 1
If "manual" is set, set ncpu_multiprocess to the number of processors to use.

6.5.2 APEX Modules: APEX Multiframe and APEX User List Multiframe Settings

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: Initial Setup
Relevant Pipelines: APEX Multiframe; APEX User List Multiframe

PURPOSE

To set the input parameters specific to the APEX Multiframe and APEX User List Multiframe pipelines. Parameters that are specific to only one of these pipelines are marked.

INPUT

User Input Fixed Positions Table File Name: (User List only) The input file containing the list of user-specified positions for source extraction. See §8.10 for more information.

Use Input List Type: (User List only) The type of positional input listed in the *User Input Fixed Positions Table File.* Options are *0 - positions in [x,y]* (pixels); *1 - positions in [RA,Dec]* (degrees) or 3 - *positions in both [x,y] & [RA,Dec]*.

Pixel by Size: Check this to set the output pixel size in degrees. The default pixel size depends on the instrument and channel: see the templates for the default sizes.

Pixel Size X (deg): Set the X-size of the output pixels in degrees. The pixel size in the X-direction is quoted as a negative value to comply with convention. A positive value will generate an error message.
Pixel Size Y (deg): Set the Y-size of the output pixels in degrees.

Pixel By Ratio: Check this to set the output pixel size by ratio of the input pixel to the output pixel, i.e. to make the output pixels half the size of the input pixels (over-sample the mosaic), set the values of *Pixel Ratio X(Y) = 2*. The ratio is taken after correction for geometric distortion.

Pixel Ratio X: Set the pixel ratio in the X-direction.
Pixel Ratio Y: Set the pixel ratio in the Y-direction.

Use PRF File Name: Use a single fixed PRF file for fitting all sources in the field of view. PRF files are provided for both IRAC and MIPS, or you can use PRF Estimate to generate your own PRF from the data (not recommended for IRAC - see Chapter 7). The PRF files for MIPS and for the center of the IRAC arrays can be found in the <*mopex_dir*>*/cal/* directory.

Use PRF Map File Name: Use a variable PRF map file, which allows the use of different PRFs depending on the location of the source on the array. See §8.7 for the format of the PRF Map file. If you wish to use a PRF Map file for IRAC, you will need to download the full list of IRAC PRF files from the website.

PRF File Name: The name of the PRF file to use for source fitting.
PRF Map File Name: The name of the PRF Map file to use for source fitting.

Mosaic PRF File Name: The name of the PRF file to be used for source detection in the mosaic. Since APEX Multiframe carries out the source extraction on the stack of input images, not the mosaic, this does not have to be perfect. For IRAC data, the PRF for the center of the array will usually suffice, while for MIPS, there is a mosaic PRF supplied. These can be found in the <*mopex_dir*>*/cal/* directory.

Probability Threshold: (float from 0.0 to 1.0, default = not set). If the Point Source Probability image is used for detection, it is in principle possible that a large proportion of pixels will have values close to 1, which is the maximum. The calculation of the initial detection threshold could be skewed upward by these high values with a possible result of no detections. If this parameter is set, values larger than *Probability Threshold* will not be included in the initial threshold calculation. This is an unlikely scenario, and this parameter generally does not need to be set.

Use PSP to Detect: This parameter determines which image should be used as input to the Detect module.

Delete Intermediate Files: Check this to delete all of the intermediate files created by the APEX pipeline. This leaves only the input files and the files created by the final module. This is useful for saving disk space once you are satisfied with your reduction setup. Note that MOPEX will prompt you with a pop-up dialogue box when you run the flow, to be sure that you meant to select this irreversible option.

Use Background Subtracted Image for Fitting: If checked, the background subtracted version of the image created by the Extract MedFilter module is used for PRF fitting.

Use Background Subtracted Image for Aperture: If checked, the background subtracted version of the image created by the Extract MedFilter module is used for aperture photometry The Aperture Photometry module can perform an annular background subtraction, so it may be better not to use this option.

Use Data Uncertainties for PRF-fitted SNR: If checked, APEX will use the data uncertainties for calculating the PRF-fitted SNR. The default is off, in which case it will use the current Gaussnoise noise estimate (i.e. basically the background RMS).

Create Standard Deviation Mosaic: If checked, the Mosaic CoAdder module will create the standard deviation mosaic.

Create UNC Mosaic: If checked, the Mosaic CoAdder module will create a mosaic of the input (or S/N Estimator generated) uncertainties.

Use Standard Deviation to Detect: If checked, the standard deviation mosaic is used as the uncertainty estimate in image segmentation.

Use Uncertainty to Detect: If checked, the uncertainty mosaic is used as the uncertainty estimate in image segmentation.

Sigma Weighted Coadd: If checked, the input uncertainties will be used as weights by the Mosaic CoAdder module.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.** Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

Use Bright Object Mask: Select this option to use the object mask generated by the Bright Detect module in the full image segmentation.

COMMAND LINE INPUT

The following parameters are set in the Global Parameters part of the namelist. Note that if both *MOSAIC_PIXEL_SIZE_X(Y)* and *MOSAIC_PIXEL_RATIO_X(Y)* are both set, the former takes precedence and will override the latter.

```
INPUT_USER_LIST = <working_dir>/user_input.tbl
use_input_list_type = 1
MOSAIC_PIXEL_SIZE_X = -0.00033889
MOSAIC_PIXEL_SIZE_Y = 0.00033889
MOSAIC_PIXEL_RATIO_X = 1
MOSAIC_PIXEL_RATIO_Y = 1
PRF_file_name = cal/apex_sh_IRAC1_col129_row129_x100.fits
PRFMAP_FILE_NAME = cal/ch1_prfmap.tbl
MOSAIC_PRF_file_name = cal/apex_sh_IRAC1_col129_row129_x100.fits
PROBABILITY_THRESHOLD = 0.0
use_psp_to_detect = 1
delete_intermediate_files = 0
use_background_subtracted_image_for_fitting = 1
use_background_subtracted_image_for_aperture = 0
use_data_unc_for_fitted_SNR = 1
create_std_mosaic = 0
create_unc_mosaic = 1
```

```
use_std_to_detect = 0
use_unc_to_detect = 1
sigma_weighted_coadd = 0
USE_REFINED_POINTING = 0
use_bright_object_mask = 0
```

OUTPUT

None

DISCUSSION

None

6.5.3 APEX Modules: APEX Single Frame and APEX User List Single Settings

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: Initial Setup
Relevant Pipelines: APEX Single Frame; APEX User List Single Frame

PURPOSE

To set the input parameters specific to the APEX Single Frame and APEX User List Single Frame pipelines. Parameters that are specific to only one of these pipelines are marked.

INPUT

User Input Fixed Positions Table File Name: (User List only) The input file containing the list of user-specified positions for source extraction. See §8.10 for more information.

Use Input List Type: (User List only) The type of positional input listed in the *User Input Fixed Positions Table File.* Options are *0 - positions in [x,y]* (pixels); *1 - positions in [RA,Dec]* (degrees) or 3 - *positions in both [x,y] & [RA,Dec].*

PRF File Name: The name of the PRF file to use for source fitting. Standard MIPS PRFs are available in the <mopex_dir>/cal/ directory, or you can make your own PRF from your data

by using the PRF Estimate pipeline (Chapter 7). Note that you should not carry out PRF fitting on IRAC mosaics; use APEX Multiframe instead.

PRF Map File Name: The name of a variable PRF map file, which allows the use of different PRFs depending on the location of the source on the array. See §8.7 for more information on the format of the PRF Map file. Note that

DCE Status Mask: The (optional) status mask image flagging bad pixels in the input image or mosaic.

Fatal Mask Bit Pattern: The bit pattern identifying the flags in the DCE Status Mask that you wish to set as fatal. See §8.11 for more information about Fatal Mask Bit Patterns.

Use PSP to detect: This parameter determines which image should be used as input into the Detect module.

Probability Threshold: (float from 0.0 to 1.0, default = not set). If the Point Source Probability image is used for detection, it is in principle possible that a large proportion of pixels will have values close to 1, which is the maximum. The calculation of the initial detection threshold could be skewed upward by these high values with a possible result of no detections. If this parameter is set, values larger than *Probability Threshold* will not be included in the initial threshold calculation. This is an unlikely scenario, and this parameter generally does not need to be set.

Use Background Subtracted Image for Fitting: If checked, the background subtracted version of the image created by the Extract MedFilter module is used for PRF fitting.

Use Data Uncertainties for PRF-fitted SNR: If checked, APEX will use the data uncertainties for calculating the PRF-fitted SNR. The default is off, in which case it will use the current Gaussnoise noise estimate (i.e. basically the background RMS).

Use Background Subtracted Image for Aperture: If checked, the background subtracted version of the image created by the Extract MedFilter module is used for aperture photometry. The Aperture Photometry module can perform an annular background subtraction, so it may be better not to use this option.

Use Extract Table for Aperture: If checked, the output table of the extracted sources (from Source Estimate) is used as input to the Aperture photometry module. If not, the output table of detected sources (by the Detect module) is used.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.** Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

Use Bright Object Mask: Select this option to use the object mask generated by the Bright Detect module in the full image segmentation.

Delete Intermediate Files: Check this to delete all of the intermediate files created by the APEX pipeline. This leaves only the input files and the files created by the final module. This is useful for saving disk space once you are satisfied with your reduction setup. Note that MOPEX will prompt you with a pop-up dialogue box when you set the flow running, to be sure that you meant to select this irreversible option.

COMMAND LINE INPUT

The following parameters are set in the Global Parameters part of the namelist. Note that if both *MOSAIC_PIXEL_SIZE_X(Y)* and *MOSAIC_PIXEL_RATIO_X(Y)* are both set, the former takes precedence and will override the latter.

```
INPUT_USER_LIST = <working_dir>/user_input.tbl
use_input_list_type = 1
PRF_file_name = cal/apex_sh_IRAC1_col129_row129_x100.fits
PRFMAP_FILE_NAME = cal/ch1_prfmap.tbl
DCE_STATUS_MASK = ch1_imask.fits
DCE_Status_Mask_Fatal_BitPattern = 32544
use_psp_to_detect = 1
PROBABILITY_THRESHOLD = 0.0
use_background_subtracted_image_for_fitting = 1
use_data_unc_for_fitted_SNR = 1
use_background_subtracted_image_for_aperture = 0
use_extract_table_for_aperture = 1
USE_REFINED_POINTING = 0
use_bright_object_mask = 0
delete_intermediate_files = 0
```

OUTPUT

None

DISCUSSION

None

6.5.4 APEX Modules: S/N Estimator

Command Line Equivalent: compute_uncertainties_internally
Default Output Directory: <output_dir>/Sigma-apex
Depends On: Initial Settings; APEX Settings
Relevant Pipelines: All

Important Notes: Only required if you do not have the uncertainty images that come with all Spitzer data.

PURPOSE

This module is used to estimate the uncertainty images when no independent uncertainty measurement is available. This module is usually not turned on since almost all Spitzer data have independent uncertainty images (**unc*.fits*) that come downloaded with Spitzer data from the archive. If you decide not to use the provided uncertainty images, or if you are not using Spitzer data, this module needs to be switched on. Warning: if the wrong *Gain* or *Read Noise* parameters are specified, this module may produce uncertainties that are too large, and this will result in no outlier detection.

INPUT

Gain in e-/image unit: (float) This is the parameter used to translate the measured data counts to physical units of MJy/sr (or mJy/sq.arcsec). All Spitzer BCDs are in units of MJy/sr. **This parameter has the same name as stored in the Spitzer BCD FITS header, but they have different units.** MOPEX *Gain* has the unit of e-/MJy/sr (or e-/mJy/sq.arcsec) and the FITS header GAIN has the unit of e-/DN. *Gain* used by MOPEX can be computed from the GAIN parameter in the FITS header by using formula:

$$Gain(MOPEX) = \frac{EXPTIME * GAIN(header)}{FLUXCONV}$$

Equation 6.1

Here FLUXCONV is the flux conversion factor between e-/DN and surface brightness unit of MJy/sr (or mJy/sq.arcsec). FLUXCONV can be found in the Spitzer BCD FITS image header.

Read Noise in e-: (float) This parameter characterizes the noise in e- when the data are being read out from detector. It is also stored in the BCD FITS image header.

Confusion Sigma in e-: (float) The 1-sigma confusion noise limit. The source of this noise is from the spatially unresolved background galaxies. This information can be found in the IRAC and MIPS Instrument Handbooks.

S/N estimator output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Sigma-apex.

COMMAND LINE INPUT

```
&SNESTIMATORIN
 Gain = 66.8,
 Read_Noise = 8.8,
 Confusion_Sigma = 0,
&END
```

In Global Parameters:

```
SIGMA_DIR = Sigma-apex
```

OUTPUT

S/N FITS Files (_sigma.fits*):** The estimated uncertainty image corresponding to each input BCD image.

DISCUSSION

This module allows users to estimate the noise from the BCD data, if no uncertainty images are provided. Here you need to input gain, readout noise and confusion limit appropriate for the dataset. We suggest that users look at the BCD image header to find the gain and read out noise. This module should not be used unless you cannot use the Spitzer archive-provided uncertainty images.

The module estimates the pixel uncertainty for each pixel in the image using the following model:

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

Equation 6.2

where the parameters *Read Noise* $\sigma_{readnoise}$, *Gain* g, and *Confusion Sigma* $\sigma_{confusion}$ are specified in the module settings. 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 units of surface brightness (MJy/sr; i.e. all Spitzer BCDs) then the *Gain* should include the conversion factor from surface brightness units to electrons (see the *INPUT* for this module). The product of this step is the uncertainty images.

6.5.5 APEX Modules: Fiducial Image Frame

Command Line Equivalent: run_fiducial_image_frame
Default Output Directory: <output_dir>
Depends On: Initial Settings; APEX Settings
Relevant Pipelines: APEX Multiframe; APEX User List Multiframe

Important Notes: Running this module will overwrite any pre-existing *FIF.tbl* file in the output directory. **If you are working with MIPS images and plan to use the Mosaic PRFs then please see the Discussion below for important information.**

If you are using the *mosaic.fits* file from a pre-pended Mosaic pipeline as input into APEX then you should omit this module.

PURPOSE

This module creates a unified grid coordinate system and defines the spatial boundaries that will include all BCD images. The output FIF table is required for later modules to run. It can be generated by this module, or set as an input using a pre-existing table. If you have appended your APEX pipeline to a Mosaic pipeline then you can omit this module; APEX will automatically use the Mosaic FIF table.

INPUT

Edge Padding: (int) This number specifies how much extra space (arcseconds) will be added to the four edges of the final mosaicked image.

Projection: This parameter specifies the projection type of the output mosaic. The TAN and SIN projections are implemented using the fast direct plane-to-plane coordinate transformation. The rest of the projections in the WCS library - LIN, AZP, STG, ARC, ZPN, ZEA, AIR, CYP, CAR, MER, CEA, COP, COD, COE, COO, BON, PCO, GLS, PAR, AIT, MOL, CSC, QSC, TSC, NCP, DSS, PLT, TNX - are implemented as a two step plane-sky-plane projection. The default projection is TAN.

Coordinate System: The coordinate system of the output mosaic. The choices are J2000 (Equatorial), Galactic, or Ecliptic. The default is J2000.

Use average input orientation: Defines the orientation of the output mosaic. Checking this box indicates that MOPEX should use the average of the input BCD orientations. This is the default setting. To specify an alternative orientation, see *CROTA2*.

CROTA2: (float) Defines the orientation (degrees East of North) of the output mosaic. *CROTA2 = 0* indicates North up and East left. To use the average of the input BCD orientations (the default), check the *Use average input orientation* box.

COMMAND LINE INPUT

```
&FIDUCIALIMAGEFRAMEIN
 Edge_Padding = 10,
 Projection_Type = "TAN",
 Coordinate_System = "J2000",
 CROTA2 = A,
&END
```

In Global Parameters:

```
MARGIN = 0
```

MARGIN: (int; command line only) Specifies a margin in terms of the number of pixels around the Fiducial Image Frame. Normally this is set to zero or a (small) positive number, except when running the Mosaic flow, with the Mosaic Geometry module, when it should normally be set to a negative number to avoid including input images that have only marginal overlap with the FIF.

OUTPUT

Generated Headers List Table (*header_list.tbl*): This file lists the World Coordinate Systems (WCS) information from each of the BCD image headers. The information is used to make the Fiducial Image Frame table.

Generated FIF Table (*FIF.tbl*): This text file is used by later modules to specify the pixel size, orientation and final image size of the mosaic. The module overwrites any input FIF table specified in the Initial Setup. Note that the output mosaic will always use the set of keywords CDELT1, CDELT2 and CROTA2, even if the input images use the CD matrix convention.

The outputs of this module are written to the top output directory specified in the Initial Setup.

DISCUSSION

This module creates a table specifying a unified coordinate system and the spatial boundaries of the output mosaic. This table is required for any interpolation or mosaicking using MOPEX. If this module is not included, an existing FIF table has to be specified in the Initial Settings to enable later modules to run. Alternatively, this module can be omitted if the APEX pipeline is appended to a Mosaic pipeline; APEX will automatically pick up the Mosaic FIF table. After this module is run once, the output FIF table can be modified manually if desired, and used as input to subsequent runs (this is useful for e.g. mosaicking small sections of a large dataset - see §5.3 for more details).

Even if the input images use the CD matrix convention, the output mosaic images (and by extension the interpolated images) will use the set of keywords CDELT1, CDELT2 and CROTA2, since the mosaic image is undistorted.

If you plan to use the MIPS "mosaic" PRFs that are provided for PRF fitting on your mosaic, your mosaic must be built in the same way as the ones used to derive those PRFs, that is, from BCDs that have nearly the same orientation (close in time or taken at the same time of

year) and with a *CROTA2* angle that is properly aligned with the BCDs (*Use average input orientation* is recommended). All other cases would require you to make your own PRF from your mosaic.

6.5.6 APEX Modules: Mosaic Interpolate

Command Line Equivalent: run_mosaic_int
Default Output Directory: <output_dir>/Interp-apex
Depends On: Initial Setup; Fiducial Image Frame; S/N Estimator (if included)
Relevant Pipelines: APEX Multiframe; APEX User List Multiframe

Important Notes: If the APEX pipeline is appended to a Mosaic pipeline, and you intend to use the *mosaic.fits* file from Mosaic as input into APEX, this module should be omitted from the flow. Note that you must have checked the *Keep coadded Tiles* box in the Mosaic CoAdder module in the Mosaic pipeline.

PURPOSE

This module performs a projection of input images onto a 2D plane defined by the FIF table, and an interpolation (see §8.4) of the input pixel values to the output array of pixels of the user-defined pixel size. It corrects for the optical distortion in the input images, using the WCS distortion parameters in the input FITS headers. The process is intended to accept images measuring surface brightness (MJy/sr or microJy/arcsec^2) and to yield images in the same units, but it is not restricted to this. If the input FITS header does not contain an allowed string specifying units of MJy/sr or microJy/arcsec^2, then it will assume the input units are counts.

INPUT

INTERP METHOD: Four interpolation options are available:

1. Default: Each output pixel is a linear weighted sum of input pixels with weights equal to the area overlap with the output pixel. The optional parameter is *Fine Res* (int). The input pixel can be sub-divided with sub-pixel sizes of *Input Pixel Size / Fine Res* before projecting onto the output array; a typical value might be 2. The value of each sub-pixel is a linear interpolation of the input pixels with weights determined by the area overlap with a pixel the same size as the original but centered on the sub-pixel. So it's a convolution. The default value of 0 means no sub-dividing. **We strongly advise setting**

***Fine Res = 0* as the *Fine Res* option is not fully implemented in the presence of bad pixels.**

2. Drizzle: Each input pixel is shrunk *Drizzle Factor* (float) times its original size along each axis, e.g. 0.5. The value of the shrunken pixel is the same as the original pixel. The shrunken pixels are then projected onto the output pixels with their values distributed into output pixels according to area overlap.

3. Grid: This method is intended to create a crude first-look mosaic quickly. Each input pixel is filled with *Grid Ratio* (int) squared grid points. Each grid point is assigned the value of the pixel it belongs to. Each grid point is projected onto the output frame and the flux associated with the grid point is added to the output image pixel into which the grid point was projected. You may not run the Mosaic Reinterpolate module with this option. The gain in speed is up to 10 times that of the Default method. **The price you pay is the fidelity of the interpolated images, so in general these should not be used for science.**

4. Cubic This method uses a bicubic interpolation. It is like the Default method except that each output pixel is a weighted sum of the 16 nearest input pixels, with the weights determined by bicubic polynomials. A tunable parameter *Alpha* (float) pins the weights. The default value of -0.5 should be used. Note: this method could be useful in high S/N cases because it enforces smoothness in the interpolation function, but the major drawback is significantly more noise correlation than the Default linear interpolation.

Mosaic Interpolate output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Interp-apex.

COMMAND LINE INPUT

```
&MOSAICINTIN
 INTERP_METHOD = 1,
 FINERES = 0,
 DRIZ_FAC = 1,
 GRID_RATIO = 4,
 ALPHA = -0.5,
&END
```

In Global Parameters:

```
INTERP_DIR = Interp-mosaic
```

OUTPUT

Mosaic FIF Table (*mosaic_fif.tbl*): The FIF table describing the final mosaic, taking into account the user-selected pixel size. Note: This file is saved in the top level Output directory.

Geometry Output Table (*interp_ImageList.txt.tbl*): The interpolated images' offsets in x- and y-direction relative to the FIF and their sizes are specified in this file.

Interp Stack (*interp_*fits*): The output images, interpolated, distortion-corrected, and projected onto the output FIF. A list is also made.

Coverage Stack (*interp_*covg.fits*): The corresponding coverage maps for each output image, taking into account the bad pixels in the input. A list is also made.

DISCUSSION

Most users will choose between *Default* and *Drizzle*. The smaller you make *Drizzle Factor*, the more coverage you need to avoid holes in the mosaic. A rule of thumb is that if you have a coverage of 10 or less, you will probably want to use the *Default* interpolation.

In general, a projection onto the reference frame of an input image with optical distortions will not be a simple rectangle. Each interpolated image occupies a part of the FIF and is, in general, of different size, with different offsets from the origin of the FIF. The interpolated images' offsets in x- and y- directions relative to the FIF, and their sizes (in integral numbers of pixels) are given in the file *interpolated_ImageList.txt.tbl*, and also written in the headers of the interpolated images in the keywords MINTOFFX and MINTOFFY.

If the APEX pipeline is appended to a Mosaic pipeline, and you intend to use the *mosaic.fits* file from Mosaic as input into APEX, this module should be omitted from the flow. Note that you must have checked the *Keep coadded Tiles* box in the Mosaic CoAdder module in the Mosaic pipeline.

6.5.7 APEX Modules: Mosaic CoAdder

Command Line Equivalent: run_mosaic_coadder
Default Output Directory: <output_dir>/Coadd-mosaic
Depends On: Mosaic Interpolate (or Mosaic Reinterpolate)
Relevant Pipelines: APEX Multiframe; APEX User List Multiframe

Important Notes: There are only two coaddition schemes available in the APEX version of this module. If the APEX pipeline is appended to a Mosaic pipeline, and you intend to use the *mosaic.fits* file from Mosaic as input into APEX, this module should be omitted from the flow.

Note that you must have checked the *Keep coadded Tiles* box in the Mosaic CoAdder module in the Mosaic pipeline.

PURPOSE

The interpolated images are co-added to create one mosaic image. The pixel values of the mosaic image are equal to the averaged interpolated pixel values. The interpolated uncertainty images are also co-added into one uncertainty mosaic image following the standard assumption of additive variances. The coverage maps are co-added into a single mosaic coverage map.

INPUT

Tile Max X, Y: (int) The X (Y)-size of the tile for performing the co-addition. This should be reduced if computer memory is an issue (see §8.1).

Integration Time Weighted Coadd: Check this box to weight the coaddition by exposure time. MOPEX searches for the header keyword specified, the default is EXPTIME.

Mosaic Co-Adder output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Coadd-apex.

COMMAND LINE INPUT

```
&MOSAICCOADDIN
 TILEMAX_X = 1000,
 TILEMAX_Y = 1000,
 INTEG_TIME_KWD = 'EXPTIME',
&END
```

In Global Parameters:

```
COADDER_DIR = Coadd-apex
```

INTEG_TIME_KWD: This is the equivalent of checking the *Integration Time Weighted Coadd* box in the GUI. If *`INTEG_TIME_KWD = 'EXPTIME'`* is set in the parameter block, intergration time weighted coadd will be used.

OUTPUT

Tile BCD file (*coadd_tile_bcd.txt*): This text file lists the input images which contribute to each output tile. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Tile table (*coadd_tiles.tbl*): A table listing the tiles, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Tile uncertainty list (*coadd_Tile_*_Unc.fits*): The stack of tiled uncertainty images.

Mosaic uncertainty file (*coadd_Tile_*_Std_Unc.fits*): The stack of tiled standard deviation images.

DISCUSSION

If the APEX pipeline is appended to a Mosaic pipeline, and you intend to use the *mosaic.fits* file from Mosaic as input into APEX, this module should be omitted from the flow. Note that you must have checked the *Keep coadded Tiles* box in the Mosaic CoAdder module in the Mosaic pipeline.

The co-addition can be performed on Tiles (see §8.1), to address computer memory considerations. The Mosaic CoAdder module performs the co-addition on the individual tiles, and the Mosaic Combine module then combines the tiles into a single mosaic. Even if only a single tile is used, Mosaic Combine still needs to be run, in order to produce the expected products.

There are only two weighting schemes available in the APEX version of this module. The interpolated images can be combined using the straight averaging or they can be weighted by exposure time. The co-added uncertainty image is computed, as long as the uncertainty images are given as an input to the program.

Straight Average: (default) If none of the weighting alternatives are selected, then the co-added pixel value O is given by the following expression:

$$O = \frac{\sum_j c_j I_j}{C}$$

Equation 6.3

Here I_j is the value of the interpolated pixel in image j, and c_j is the pixel value of the corresponding coverage map. Co-added uncertainty U = W-1 is given by the following expression:

$$W = \left(\sqrt{\sum_j \frac{c_j}{\sigma_j^2}} \right)$$

Equation 6.4

where σ_j is the pixel value of the corresponding uncertainty image, and co-added coverage C is simply the sum of the individual input coverages:

$$C = \sum_j c_j$$

Equation 6.5

Exposure Time Weighted Average: If *Integration Time Weighted Coadd* is selected then MOPEX weights the co-added pixels by the exposure time given in the fits header keyword specified. For Spitzer data, the integration time is given by the keyword EXPTIME.

6.5.8 APEX Modules: Detect MedFilter

Command Line Equivalent: run_detect_medfilter
Default Output Directory: <output_dir>/Medfilter-apex
Depends On: Mosaic Interpolate
Relevant Pipelines: All

PURPOSE

This module is an instance of the same MedFilter that is run during outlier rejection in the *Mosaic* pipeline. In this case, it performs the background subtraction of the mosaicked image (or tiles), which will be used for source detection. There are two options - the Median case (default) and the Sbkg case, which uses a background estimate like that of SExtractor.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

Min Good Pixels in Window: (int) The minimum number of good (non-NaN, non-masked) pixels needed to compute the median. This parameter should generally not need to be adjusted. If insufficient good pixels are present, the background is recorded as "missing" and the "missing" output pixel is replaced by interpolation of its neighbors if possible.

Min Good Neighbors Number: (int) The minimum number of good neighbor pixels needed to perform the replacement of "missing" output pixels. This parameter should generally not need to be adjusted.

Use SExtractor background estimation: (int) A value of 1 invokes the Sbkg background estimate based on SExtractor. Any other value gives the default Median estimate.

SExtractor background filter size: (int) Median filter box size for the Panels when this option is set. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Med Filter output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Medfilter-apex.

COMMAND LINE INPUT

```
&DETECT_MEDFILTER
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 50,
 Min_Good_Pixels_In_Window = 9,
 Min_GoodNeighbors_Number = 4,
 Max_Bad_Pixels_OutputImage = 100,
 Sbkg_Filt_Size_for_Med = 1,
 Use_Sbkg_for_Med = 1,
&END
```

In Global Parameters:

```
MEDFILTER_DIR = Medfilter-apex
```

Max_Bad_Pixels_OutputImage: (int; command line only) The maximum number of allowed bad (NaN) pixels in the output image.

OUTPUT

Generated FITS *(*_Image_minback.fits* in Multiframe or *mosaic_minus_median_detect.fits* in Single Frame): The output is the background subtracted mosaic image (or tiles). In Single Frame mode, the mosaic is written to the main output directory, but in Multiframe mode the tiles are stored in the Coadd-apex subdirectory.

DISCUSSION

The module is run on the detection image. In Single Frame pipelines, this is the input image. In Multiframe pipelines, the mosaic may be a single image produced during the Mosaic or APEX pipeline, or it may be the several tiles produced by Mosaic CoAdder.

With the default Median option, the program computes a background value using the median of a running rectangular window of *Window X* by *Window Y* pixels, after omitting the *Outliers / Window* highest pixels. This is done for each pixel so can be very slow.

If *Use SExtractor background estimation* is set, the module switches to the Sbkg background estimation based on that of SExtractor (Bertin and Arnouts, AASupp 117, 393, 1996). The image is divided into Panels with size given by *Window X(Y)*. In each Panel, iterative clipping is used to find a single estimate of the background in the Panel. You can optionally median filter the Panel background values, e.g. to avoid ones where bright objects skewed the background estimate. The median filter size is given by *SExtractor background filter size*. It then interpolates the Panel values to find the background at each pixel. This is much faster than calculating the median of a big window at each pixel.

Both background estimates generally give reasonable results, but if the data volume is large, the Sbkg option is strongly recommended, as it is much faster. It also does not require an estimate of *Outliers / Window*.

There is a minimum required number of good (not-NaN and not marked by any mask) pixels per 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 Good Neighbors Number* the program finds the average and outputs this value for the pixel in question.

The same module is typically run twice in the APEX flow. In general, the input parameters for Detect Medfilter can be "tighter" than for Extract Medfilter, i.e. smaller window and number of outliers, since the goal is detection of a peak, not the best estimate of the background.

6.5.9 APEX Modules: Point Source Probability

Command Line Equivalent: run_pointsourceprob
Default Output Directory: <output_dir>/Coadd-apex
Depends On: Mosaic CoAdder (or the equivalent input from the Mosaic pipeline)
Relevant Pipelines: APEX Multiframe; APEX Single Frame

Important notes: The user has a choice of two possible input uncertainty images for this module - the std image or the unc image. **The choices of input are determined in the APEX Settings module**. See the Discussion below for more information on input image choices.

There is also a choice of several possible data input images into this module, but the difference only becomes significant in a later module, Detect. The reason we mention it here is that this module will carry out the processing on the image specified for input into the Detect module, and the name of the output file will reflect this (see Discussion below). The choice is specified in the APEX Settings module with the *Use PSP to Detect* parameter.

PURPOSE

This module filters the input co-added image to estimate the probability at each pixel of having a point source above the noise.

INPUT

PRF Resample X(Y) Factor: (int) The ratio of the PRF pixel size to the PRF sampling interval in the x- and y-direction.

PRF X(Y) size: (int) The size of the portion of the PRF image, in input pixels, used to convolve with the input image

Apriori_Probability (float): the lowest probability of a source. It is recommended that it be left to its default of 0.1.

COMMAND LINE INPUT

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

Noise_Type: (char, command line only) Options are 'external_noise' and 'internal_noise'. The first option requires an uncertainty image, supplied in the Initial Setup. For the second option the noise is estimated from the input image.

OUTPUT

Generated FITS (_PSP.fits* or **_Filtered.fits*):** The output is the PSP image(s), which shows the point source probability at each pixel. The filename depends on the input image (see Discussion below).

DISCUSSION

This step is performed to improve detectability of the point sources. The filtering is conceptually similar to a convolution of the input image by the PSF, which is commonly done during source extraction. It can be shown that by using the ideas of maximizing SNR in the image, a filter can be derived to estimate the probability at each pixel of having a point source above the noise (see the document Bayesian Estimation of Point Source Probability http://irsa.ipac.caltech.edu/data/SPITZER/docs/files/spitzer/bayesian_estimation_PSP.pdf).

$$P(j) = \left(1 + \frac{1 - Apriori\_\mathrm{Pr}obability}{Apriori\_\mathrm{Pr}obability} \exp\left(- \frac{\left(\sum_i \frac{s(i) \cdot PRF(j-i)}{\sigma^2(i)} \right)^2}{2 \sum_i PRF^2(j-i)} \right) \right)^{-1}$$

Equation 6.6

Here *s* is the input background subtracted image, σ is the input uncertainty image, which may be either the *mosaic_unc.fits* or *mosaic_std.fits* uncertainty image. This is determined in the APEX Settings module, with the switches *Use Uncertainty to Detect* and *Use Standard Deviation to Detect*. If both are set, the first overrides the second. If neither is set, it will calculate the noise internally.

The output of this filter *P(j)* at pixel *j* can be interpreted as a probablility of having a point source above the noise at this pixel. The summation is for all pixels *i* within the area defined by the *PRF X(Y) Size* parameters. The values of the probability 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 the whole range or having too many pixels saturated very close to 1. To prevent this from happening, the argument of the exp function is rescaled to utilize the full dynamic range.

There are several possible data input images to this module, but the difference only becomes significant in the later module, Detect. The only reason we mention the following here is that it may affect the name of the output file from this module. The choice of input is specified in the APEX Settings keyword *Use PSP to Detect* where the options available are *Filtered Image*, *PSP Image* (the output of this module) or *background subtracted input image*. *Filtered Image* is defined as the product of the PSF image times the background subtracted image: *F = P*s*. If the Filtered image is used, the name of the output file from this module will be **_Filtered.fits* instead of **_PSP.fits*. *PSP Image* uses the output of this module as input into the Detect module, and *background subtracted input image* uses the original background-subtracted input image for source detection. For more details, see the Detect module.

6.5.10 APEX Modules: Gaussnoise

Command Line Equivalent: run_gaussnoise
Default Output Directory: <output_dir>; <output_dir>/Coadd-apex
Depends On: Initial Setup
Relevant Pipelines: All

PURPOSE

This module estimates the background fluctuations in the input image(s). It is very similar to Detect MedFilter. It finds the 68-percentile range of the pixel values in a sliding window in order to measure the Gaussian noise.

INPUT

Window X: (int) The X size in pixels of the window used to compute the median of the background. The default value is 45

Window Y: (int) The Y size in pixels of the window used to compute the median of the background. The default value is 45

Number of Outliers Per Window: (int) The number of outlier pixels (N) being rejected from the X*Y window when computing the noise.

Minimum Good Pixels Per Window: (int) The minimum number of good (non-NaN, non-masked) pixels needed to compute the noise. If insufficient good pixels are present, the "missing" output pixel is replaced by the interpolation of its neighbors.

Minimum Good Neighbors Number: (it) The minimum number of good neighbor pixels need to perform the replacement of "missing" output pixels.

COMMAND LINE INPUT

```
&GAUSSNOISE
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 100,
 Min_Good_Pixels_In_Window = 9,
 Min_GoodNeighbours_Number = 4,
 Max_BadPixels_OutputImage = 100,
```

```
&END
```

if *use_psp_to_detect* is set in the APEX Settings and *Input_Type = "snr_input"* is set in the Detect module then you must also include the following parameter block. ***N_Outliers_Per_Window*** **must be set to 0:**

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

OUTPUT

Generated FITS (_noise.fits*):** The output is the tile noise image(s) measured from the input image(s).

DISCUSSION

The program computes the Gaussian noise (68 percentile) for each pixel in the input image using a rectangular window of *Window X* by *Window Y* pixels. It is achieve by omitting the *Number Of Outliers Per Window* highest pixels from each median window. There is a minimum required number of good (not-NaN and not marked by any mask) pixels per median window *Minimum Good Pixels Per 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 *Minimum Good Neighbors Number*, the program finds the average and stores this value as the median for the pixel in question.

6.5.11 APEX Modules: Bright Detect

Command Line Equivalent: run_bright_detect
Default Output Directory: <output_dir>; <output_dir>/Coadd-apex
Depends On: Initial Setup
Relevant Pipelines: APEX Multiframe; APEX Single Frame

PURPOSE

This module provides a different kind of detection table than the Detect. It includes shape information about the detections, but it does not record the blends, so in general it cannot replace Detect. It also produces a mask image (pixel value = 1 on the detections). This can be used to mask the brightest detections before running Detect.

INPUT

Same as for the Detect module (see next section, §6.5.12).

COMMAND LINE INPUT

See the Detect module

OUTPUT

Table (_detect_bright.tbl*):** a detection list with shape information.

FITS file (_detect_bright_mask.fits*):** a mask image showing the segmentation level of detected clusters.

For APEX Multiframe, the output files are written to the *Coadd-apex/* subdirectory.

DISCUSSION

The bright detect list created by this module provides information about the shapes of the detections. The shape parameters are mostly clear from the column names. The shape parameter Linearity is 0.25 * (Perimeter length) / sqrt (No. pixels). Below is an example of the detection table:

```
\char comment = Output from DETECT, version 4.00
\char Date-Time = Wed Feb 4 12:12:34 2004
\char comment = dlimage library version 2.00
\char FITS_In_Filename = Coadd-apex/coadd_Tile_001_Image_minback.fits
\char FITS_Out_Filename = Coadd-apex/coadd_Tile_001_Image_detect_bright_mask.fits
\char Sigma_In_Filename = Coadd-apex/coadd_Tile_001_Unc.fits
\char Bright_Table_Out_Filename = Coadd-apex/coadd_Tile_001_Image_detect_bright.tbl
\float Tile_OffsetX = 0
```

```
\float Tile_OffsetY = 0
\int Peaks_Radius = 1
\char CoverageMap_Filename = Coadd-apex/coadd_Tile_001_Cov.fits
\float Min_Coverage = 2
\float Effective_Threshold = 0.259142
\int Extended_Object_Area = 200
\int Max_Segmentation_Level = 50
\int Detection_Max_Area = 1000
\int Detection_Min_Area = 30
\int N_Edge = 0
\float Detection_Threshold = 6
\char Input_Type = image_input
\char Output_Type = centroids_and_pixels_output
\char Neighbor_Type = sides_only
\char Threshold_Type = peak
\int Number_Of_Detections = 118
srcid| x       | delta_x | y       | delta_y  | delta_xy | flux    | delta_flux|
maj_ax  | delta_maj| min_ax  | delta_min| elongation| ellipticity| theta    |
delta_theta| n_pixels| perimeter| linearity|
|  i |r        |      r  |r        |      r   |       r  |   r     |         r |     r
|        r |     r   |        r |         r |          r |    r     |          r |
i |        i |        r |
  1   6.292e+02 6.751e-02 2.552e+01 7.353e-02  3.333e-02  8.223e+02 1.407e+01
3.118e+00 7.856e -02 2.515e+00 6.159e-02  1.240e+00   1.934e-01    5.636e+01
5.636e+01    332        94        1.290e+00
  2   1.493e+02 1.350e-02 3.395e+01 1.430e-02  -4.271e-03 3.814e+03 2.266e+01
1.553e+00 1.465e -02 1.471e+00 1.312e-02  1.056e+00   5.289e-02    -6.515e+01 -
6.063e+01   278        132       1.979e+00
  3   1.232e+03 3.253e-02 3.493e+01 3.006e-02  1.101e-03  3.814e+02 7.638e+00
1.118e+00 3.354e -02 1.020e+00 2.893e-02  1.095e+00   8.712e-02    1.720e+01
4.057e+01     70         46       1.375e+00
```

The bright detect mask marks detections with integer values equal to the highest segmentation level (iteration) reached, and non-detection pixels with 0.

Bright Detect can provide useful information as an adjunct to the normal Detect module. In this case, set the available parameters to the same as those used for Detect, but be aware that because it does not include blend parameters, it cannot replace Detect for source extraction.

Advanced users could use this to try to mask out the brightest sources before running Detect. Set the *Detection Threshold* to a high value and turn on *Use Bright Object Mask* in the APEX Settings module. This tells Detect to mask out the bright detections. **Note: This feature has not been well tested and users should exercise caution.**

6.5.12 APEX Modules: Detect

Command Line Equivalent: run_detect
Default Output Directory: <output_dir>; <output_dir>/Coadd-apex
Depends On: Point Source Probability

Relevant Pipelines: APEX Multiframe; APEX Single Frame

Important notes: The user has a choice of several different input images to his module, but the choice is only partly set from within this module. The *Input Type* parameter gives the choice of whether to use a data or SNR image as input, but the choice of which version of these two types to use is set in the APEX Settings module using the parameters *Use PSP To Detect, Use Uncertainty to Detect* and *Use Standard Deviation to Detect*. See the *Input Type* and Discussion information below for a full description.

PURPOSE

Detect performs image segmentation and computes the centroids for the detected pixel clusters. It is another instance of the same module used for outlier detection in the Mosaic pipeline.

INPUT

Threshold Type: This parameter determines the way the image segmentation threshold is recalculated during iterative thresholding. The threshold type does not depend on the type of input image. See §8.3 for a description of the thresholding options.

Detection Threshold: (float) The number of sigma above the mean to be used as the initial threshold used for cluster detection. Vary this to determine how deep the detections can go.

Detection Min Peak Fraction: (float) A candidate sub-peak must have a minimum height above the current threshold. The height is given as the fraction of the "parent" peak height. Allowed values are between 0.0-1.0. If not set, it is 0.0. Vary this to limit false detections on the wings of bright sources.

Detection Min Area: (int) The minimum area (pixel) for a detected pixel cluster to be retained; smaller clusters are discarded.

Detection Max Area: (int) The maximum area (pixels) of a detected pixel cluster before iterative thresholding.

Extended Object Area: (int) The minimum area (pixels) to be considered an extended object. If the number of pixels in a cluster is larger than this area, it is not split by *Threshold Type = "Peak"*.

Input Type: Determines the type of image to be used as the input for detect. The first choice is between an "snr" image (*SNR Input*) or an image based on the original data (*Image Input*). Which image within these 2 types depends on the Use PSP to Detect setting in the initial Apex settings. See the Table below.

If Input Type is chosen to be "snr_input", then the Gaussnoise module is run on the input to Detect (e.g. the PSP image) to create the new SNR image. In that case, the Window and Outlier parameters of Gaussnoise can be set.

Window X: (int) The X size in pixels of the window used to compute the median of the background. The default value is 45.

Window Y: (int) The Y size in pixels of the window used to compute the median of the background. The default value is 45.

N Outliers Per Window: (int) The number of outlier pixels (N) being rejected from the X*Y window when computing the noise.

Output Type: *Centroids only output* is the normal choice. *Centroids and pixels output* allows a complete table (see below).

Minimum Coverage: (float) The minimum value in the coverage map needed for a pixel to be considered for segmentation.

Neighbor Type: Defines what an adjacent pixel means. "Sides only" is the normal choice.

Peaks Radius: (int) Radius in pixels that a pixel must be the maximum in order to be defined as a peak. 1 is the normal choice.

Max Segmentation Level: (int) Maximum number of segmentation passes. 50 is the normal choice.

N Edge: (int) 0 is the normal choice.

Probability Threshold: (float) 0.0 is the normal choice.

Additional output files: See "Output".

COMMAND LINE INPUT

```
&DETECT
 Input_Type = 'image_input',
 Output_Type = "centroids_only_output",
 Detection_Max_Area = 5,
 Detection_Min_Area = 2,
 Detection_Threshold = 10,
 Threshold_Type = 'combo',
 Extended_Object_Area = 500,
 Minimum_Coverage = 1.8,
 Complete_Table_Out_Filename = "completeout.tbl",
&END
```

if Input_Type = "snr_input" then you must also include the following parameter block. **N_Outliers_Per_Window must be set to 0**:

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

OUTPUT

Table (*mosaic_detect.tbl*): In Single Frame mode there is one detection list produced in this step. In Multiframe mode, the detection is performed for each co-added tile. The detection lists for each tile (i.e. each spatial area) are merged into one detection list for the full area.

For each source, the output table of detected sources lists:

- the source ID
- the position (X, Y in pixels)
- a rough "flux" estimate (the peak pixel value, in the same units as the input image),
- BlendId: keeps track of the sequential number of a detection blend in the table.
- BlendSize: gives the number of detections in the blend.

Additional Output Images: Other images can be output if specified: Complete table (has all detected pixels), FITS Output (image of highest detected pixels), Mask Output (image with all detected pixels marked with highest segmentation level), Peaks Image (image with peak

pixels set to 1), Number Cluster Image (pixels above last threshold marked with cluster number).

Detection Image: The image used for detection (input or SNR, for example) can be displayed.

DISCUSSION

The Detect module performs image segmentation and computes the centroids for the detected pixel clusters. Detected sources are written to two identical tables (*mosaic_detect.tbl* and *mosaic_detect_raw.tbl*). The output table may be filtered on certain criteria using the Select Detect module, in which case the "raw" table is retained and other is overwritten.

Input Image Type: The input to this module depends on the Detect parameter *Input Type* and the APEX Settings parameters *Use PSP to Detect.* In addition, the APEX Settings parameters *Use Uncertainty to Detect* and *Use Standard Deviation to Detect* affect the output of the Point Source Probability module, and therefore the type of PSP image that is used as input into this module. The table below gives the type of image that will be used for segmentation. The "Filtered" image mentioned in the table is defined as the product of the PSF image times the background subtracted image (see §6.5.9 APEX Modules: Point Source Probability for more information).

Table 6.1: Resultant input image to module Detect depending on the values of *Use PSP to Detect* and *Input Type.*

Use PSP to Detect	***Input Type***	**Resultant Input Image**
Background subtracted input image (= 2 on the command line)	*SNR input*	SNR image based on the background subtracted input image
Background subtracted input image (= 2 on the command line)	*Image input*	Background subtracted input image
PSP image (= *1* on the command line; default)	*SNR input*	SNR image based on the Point Source Probability image
PSP image (= *1* on the command line; default)	*Image input*	Point Source Probability image
Filtered image (= *0* on the command line)	*SNR input*	SNR image based on the Filtered image

Filtered image (= *0* on the command line)	*Image input*	Filtered image
Input image w/o background subtraction (= *-1* on the command line; Single Frame only)	*SNR input*	SNR image based on the input image without background subtraction
Input image w/o background subtraction (= *-1* on the command line; Single Frame only)	*Image input*	The input image without background subtraction

Output Format: Below is an example of the output table from APEX Single Frame:

```
\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
```

6.5.13 APEX Modules: Mosaic Combine

Command Line Equivalent: run_mosaic_combiner
Default Output Directory: <output_dir>; <output_dir>/Combine-apex
Depends On: Mosaic CoAdder
Relevant Pipelines: APEX Multiframe; APEX User List Multiframe

Important Notes: This module is only required in Multiframe mode. If the APEX pipeline is appended to a Mosaic pipeline, and you intend to use the *mosaic.fits* file from Mosaic as input into APEX, this module should be omitted from the flow. Note that you must have checked the *Keep coadded Tiles* box in the Mosaic CoAdder module in the Mosaic pipeline.

PURPOSE

This module combines the co-added tiles into a mosaicked image. Even if the co-addition has been done in a single tile, this module must be run in order to produce the expected outputs.

INPUT

Mosaic Combiner output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Combine-apex.

COMMAND LINE INPUT

In Global Parameters:

```
COMBINER_DIR = Combine-apex
```

OUTPUT

Median Tile BCD File (*Coadd-apex/coadd_median_tile_bcd.txt*): This text file lists the input images that contribute to each output tile of the median mosaic. The format of the file is the tile ID, the number of contributing images, and the image index within the stack.

Median Tile Table (*Coadd-apex/coadd_median_tiles.tbl*): A table listing the tiles of the median mosaic, their sizes, and their offsets in the x- and y-directions with respect to the FIF.

Mosaic Median File (*median_mosaic.fits*): The output median mosaic.

Mosaic image File (mosaic.fits): The output mosaic FITS image. When displayed in the GUI, the footprint of the interpolated input images is overlaid on the mosaic. It may be turned off using the "Mosaic" check box in the display window.

Mosaic uncertainty File (*mosaic_unc.fits*): The output uncertainty mosaic.

Mosaic coverage File (*mosaic_cov.fits*): The output coverage mosaic.

DISCUSSION

After module Mosaic CoAdder produces individual tiles, this module stitches together all tiles into a big mosaic. Even if the co-addition was done in a single tile, this module must still be run in order to produce the expected outputs.

Not all of the output files above will be produced by Mosaic Combine. Many will only be produced if requested in either APEX Settings or Mosaic CoAdder.

This module is only required in the APEX flow when running APEX Multiframe or APEX User List Multiframe. If a mosaic has previously been created using Mosaic then this module can be dropped from the APEX flow, and the mosaicking results used as input instead. Note that the APEX pipeline does not carry out any outlier rejection when combining the input images into a mosaic. In order to use an outlier rejection scheme, you must run Mosaic and use the output from Mosaic as input into APEX. See §6.3.1 for instructions on how to do this.

6.5.14 APEX Modules: Extract MedFilter

Command Line Equivalent: run_extract_medfilter
Default Output Directory: <output_dir>; <output_dir>/Medfilter-apex
Depends On: Detect or input user list
Relevant Pipelines: All

PURPOSE

This module is an instance of the same MedFilter that is run during outlier rejection in the *Mosaic* pipeline, and earlier in the APEX pipeline (before source detection). In this case, it performs the background subtraction of the image(s) that will be used for source extraction. In Multiframe pipelines, those are the individual input images, and in Single Frame pipelines it is the mosaic image. There are two options: the Median case (default) and the Sbkg case, which uses a background estimate like that of SExtractor.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

Min Good Pixels in Window: (int) The minimum number of good (non-NaN, non-masked) pixels needed to compute the median. If insufficient good pixels are present, the background

is recorded as "missing" and the "missing" output pixel is replaced by interpolation of its neighbors if possible.

Min Good Neighbors Number: (int) The minimum number of good neighbor pixels needed to perform the replacement of "missing" output pixels.

SExtractor background filter size: (int) Median filter box size for the Panels when this option is set. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Use SExtractor background estimation: (int) A value of 1 invokes the Sbkg background estimate based on SExtractor. Any other value gives the default Median estimate.

COMMAND LINE INPUT

```
&EXTRACT_MEDFILTER
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 50,
 Min_Good_Pixels_In_Window = 9,
 Min_GoodNeighbors_Number = 4,
 Max_Bad_Pixels_OutputImage = 100,
 Sbkg_Filt_Size_for_Med = 1,
 Use_Sbkg_for_Med = 1,
&END
```

Max_Bad_Pixels_OutputImage: (int; command line only) The maximum number of allowed bad (NaN) pixels in the output image.

OUTPUT

Generated FITS (_Image_minback.fits* in Multiframe modes or *mosaic_minus_median_detect.fits* in Single Frame modes):** The output is the background subtracted mosaic image (or tiles). In Single Frame modes, the files are written in the main output directory, but in Multiframe pipelines, the tiles are stored in the *Coadd-apex/* subdirectory.

DISCUSSION

The program computes an asymmetrically skewed median for each pixel in the input image using a rectangular window of *Window X* by *Window Y* pixels. It is achieved by omitting the *Outliers / Window* highest pixels from each median window.

There is a minimum required number of good (not-NaN and not flagged 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 Good Neighbors Number* the program finds the average and stores this value as the median for the pixel in question.

This module is run earlier in the APEX flow as Detect MedFilter. The input parameters for Extract MedFilter should always be more aggressive than for Detect MedFilter (i.e. smaller window size and number of outliers).

6.5.15 APEX Modules: Fit Radius

Command Line Equivalent: run_fit_radius
Default Output Directory: <output_dir>; <output_dir>/Coadd-apex
Depends On: Detect or input user list
Relevant Pipelines: APEX Single Frame; APEX Multiframe

PURPOSE

This module determines a fitting area for each source in the detect table. If not run, the fitting will be done for each detection using the fixed value of *Fitting Area X,Y* defined in the Source Estimate module. The term "Fit Radius" is a little misleading, since all fitting is done in a box, not a circle.

INPUT

None.

COMMAND LINE INPUT

```
&FIT_RADIUS
```

```
&END
```

OUTPUT

The fitting area size Fit X,Y in pixels is written into the existing detect table *(*_detect.tbl*).

DISCUSSION

Running of this module is optional, but it should generally be run for point sources with good signal-to-noise. Another option is to specify the fitting area size in the Source Estimate module using the 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 set of 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 are used.

The Fit Radius module 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 sqrt(coverage). For each detection, a search is performed in the 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.

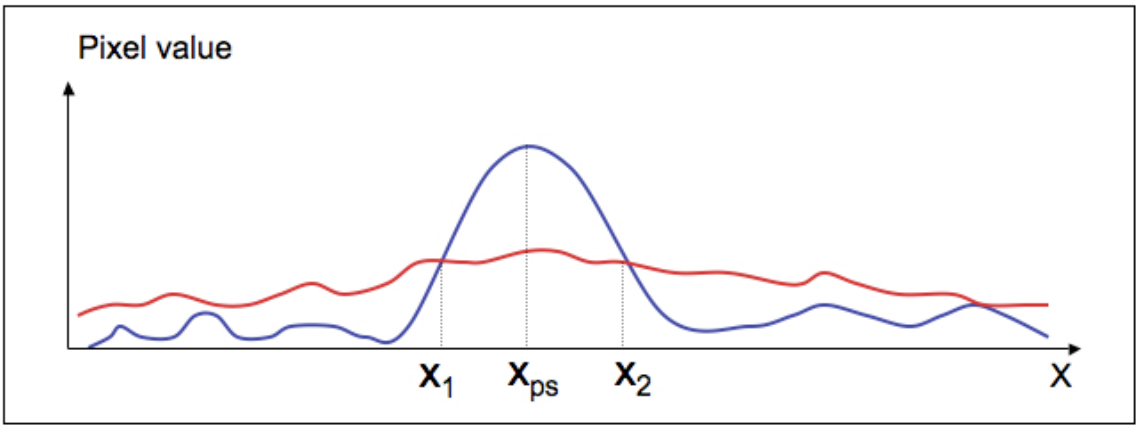


Figure 6.2: The mean-subtracted input image (blue) and the uncertainty image (red) have two intersection points x_1 and x_2. x_{ps} is the position of the detection. In this case, the fitting size FitX = 2(x_2 - x_{ps}), since (x_2-x_{ps}) < (x_{ps} - x_1).

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
```

6.5.16 APEX Modules: Select Detect

Command Line Equivalent: run_select_detect
Default Output Directory: <output_dir>
Depends On: Detect or input user list
Relevant Pipelines: APEX Single Frame; APEX Multiframe

PURPOSE

This module selects columns and rows from the Detect output table that satisfy user specified constraints.

INPUT

Columns selection:

srcid:	Extracted source number.
x:	x-coordinate of source in the mosaic (pixels).
y:	y-coordinate of source in the mosaic (pixels).
flux:	Peak pixel flux of the point source. Units are specified in the header. Currently set to microJy.
blendId:	Keeps track of the sequential number of a detection blend in the table.
Blend_size:	Gives the number of detections in the blend.

Condition Builder: Some simple selection ranges, such as > and <, can be added.

COMMAND LINE INPUT

In Global Parameters:

```
select_columns = "srcid, x, y, flux, BlendId, BlendSize"
select_conditions = "BlendSize < 5 and flux > 20"
```

OUTPUT

Output Table: The Detect table (*mosaic_detect.tbl*) is overwritten with the selected data. The full output of Detect is retained in the raw Detect table (*mosaic_detect_raw.tbl*).

DISCUSSION

Normally, the user might want to select out very big blends, since point source extraction may not be reliable for them. Another use would be to remove the BlendSize (column header "deblend_size") and BlendId (column header "deblend_id") columns, which prevents MOPEX from carrying out passive deblending.

6.5.17 APEX Modules: Detection Map

Command Line Equivalent: run_detection_map
Default Output Directory: <output_dir>
Depends On: Input user list

Relevant Pipelines: APEX User List Multiframe

PURPOSE

This module is required in APEX User List Multiframe to track the input source positions through the stack of input BCDs.

INPUT

N Edge: (int) A positive integer excludes point sources within this number of pixels from the edge of the map. **The selected value must match the value of *N Edge* set in the Source Estimate module.**

COMMAND LINE INPUT

```
&DETECTIONMAP
 N_Edge  = 4,
&END
```

OUTPUT

Detection Map table (*detectionmap.tbl*) created in the output directory.

DISCUSSION

This module is only required by the APEX User List Multiframe flow the first time it is run. For subsequent runs through with the same source table (e.g. when trying different extraction parameters), this module can be left out.

6.5.18 APEX Modules: Source Estimate

Command Line Equivalent: run_sourcestimate
Default Output Directory: <output_dir>
Depends On: Detect or Input user list; Fit Radius (optional)
Relevant Pipelines: All

PURPOSE

Fits the input image with the PRF to estimate fluxes and refine positions of the point-source candidates from the detection list.

INPUT

NOTE: The inputs marked with a * are the ones users will be most likely to adjust.

Input Type: *Image List* for APEX Single Frame and APEX User List Single Frame; *Tile Map* for APEX Multiframe*; Detection Map* for APEX User List Multiframe.

***Fitting Area X,Y:** (int) Size of the area in the input image to be fit with the PRF, in input image pixels. If not given, the detection table is expected to have FitX and FitY columns, whose content is used for the same purpose. These columns are produced by the Fit Radius module. If these columns are present, and *Fitting Area X,Y* are set then the latter takes precedence.

Max Number ps: (int) Maximum number of point sources to use in an attempt to deblend a detection (active de-blending). If *Max Number ps > 1*, the module will attempt to do an "active" deblend of the detections that were unsuccessfully fit. This should be used only with extreme caution.

***Chi Threshold:** (float) Maximum acceptable value of χ^2 / dof (degrees of freedom).

N Edge: (int) Positive number excludes point sources within *N Edge* pixels of the edge of the input image from fitting. A negative number appends an edge around the input images *N_Edge* pixels wide to include the point sources into the fitting. In APEX User List Multiframe, *N Edge* should match the parameter of the same name in the Detection Map module.

Max N Iteration: (int) Termination criterion for the maximum number of iterations to reach *Chi Threshold*.

Max N Success Iteration: (int) Termination criterion for the maximum number of iterations after *Chi Threshold* has been reached.

Minimize Ftol: (float) Termination criterion for the relative change in χ^2 / dof to achieve before *Chi Threshold* is reached.

Minimize Ftol Success: (float) Termination criterion for the relative change in χ^2 / dof to achieve after *Chi Threshold* is reached.

Dither Pixel Fraction: (float) See §8.6: PRF Fitting in MOPEX.

Dither Flux Fraction: (float) See §8.6: PRF Fitting in MOPEX.

Deblend Dither Pixel Fraction: (float) Same as *Dither Flux Fraction* but applied during active deblending.

***Background_Fit:** (switch) Switch on (= 1) to fit a constant background under the point source in the fitting area. Default is off (= 0).

Random_Fit: (switch) Switch on (= 1) to give 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) Acceptable change in χ^2 / dof for incremented number of point sources in active deblending.

***PRF ResampleX, Y Factor:** (int) Ratio of the input image pixel size to the PRF pixel size.

***Normalization Radius:** (int) PRF flux is normalized within this radius (in PRF pixels). By default, the PRF is normalized to the whole PRF array.

***Max Shift X, Y:** (float) User List Mode only. Maximum distance in data pixels from the input position for a successfully fitted source position. If greater than *Max Shift* away from the user-supplied position, the source is given the letter "O" in the deblend column (for "Outside"), along with any other letter flags.

***Angular Distance:** (float) User List mode only. The maximum angular distance (arcsec) used to identify the nearby sources for simultaneous fitting. If it is not given then the script will look for the Spitzer keywords "INSTRUME" and "CHNLNUM" in the input FITS header and the value gets set as follows: INSTRUME = "MIPS" and CHNLNUM = 1: *Angular_Distance = 6*. INSTRUME = "MIPS" and CHNLNUM = 2: *Angular_Distance = 18*. INSTRUME = "IRAC": *Angular_Distance = 2* for all four channels. If the script is unable to set a value, the default is 0.0.

***Use_Photerr_for_SNR:** (switch) Options for how the signal-to-noise ratio (SNR) of the fitted source can be estimated. These are approximations to the flux SNR, distinct from the formal flux uncertainty from the fit, delta_flux, which is also output. The SNR choices are

given below. On the command line, use the number indicated (e.g. *Use_Photerr_for_SNR = 2*).

0. no (default): This is APEX's old SNR. The noise is the noise in the peak pixel, times a scaling factor that takes into account that pixel's contribution to the PRF for a center-of-pixel hit. The noise should be from the Gaussnoise module (*Use Data Uncertainties for PRF-fitted SNR* turned off in the APEX Settings. The estimate is poor (an underestimate of noise) for undersampled pixels, e.g. IRAC channels 1 and 2.

1. yes, no gain: Use the sum (in quadrature) of the data uncertainties within a box roughly the size of the core of the PRF (*Use Data Uncertainties for PRF-fitted SNR* turned on in the APEX Settings. This is usually best.

2. yes, with gain: Use the background noise estimate from the Gaussnoise module within a box roughly the size of the core of the PRF, and add the photon noise calculated from the flux and the gain factor (*Use Data Uncertainties for PRF-fitted SNR* turned off)

Gain_for_SNR (float): The gain factor (electrons per data units) for use in the SNR estimate (above). For Spitzer data, the data units are given in MJy/sr. This is the gain for a single image. It will take into account the coverage to estimate an effective gain. In the case of integration-time weighting, it should be for unit time.

COMMAND LINE INPUT

```
&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,
 MaxShift_X = 3.0,
 MaxShift_Y = 3.0,
 Angular_Distance = 6.0,
 Use_Photerr_for_SNR = 1,
 Gain_for_SNR = 1000.0,
&END
```

OUTPUT

Table (_extract_raw.tbl*):** The preliminary extract table, which includes the fluxes and improved positions of the point sources, along with other parameters characterizing the fit. Details of the output parameters are given in the description of Select (§6.5.20).

DISCUSSION

Details of the fitting are given in §8.6 PRF Fitting in MOPEX. Two kinds of uncertainty estimates are output (see §8.9 Uncertainty Estimation). If the Photerr is chosen, note that the SNR value for any bad uncertainties in the box is -9.99, and these objects might be lost from the final extract table if a SNR cut "select" is done.

If set, the fitting area parameters (*Fitting Area X, Y*) will be used instead of the output from the Fit Radius module, so make sure to unset them (leave them blank) if Fit Radius is to be used. If Fit Radius is not run, *Fitting Area X, Y* must be set. They **must** be set in APEX User List pipelines, as these pipelines do not use the Fit Radius module.

APEX normalizes the PRF in the following way: if *Normalization Radius* (in units of PRF pixels) is set, it will normalize by the sum of the "central" PRF within that radius. By "central" we refer to the PRF corresponding to a center-of-pixel hit. If *Normalization Radius* is not set, APEX will normalize by the available range that the "central" PRF covers, i.e. the whole array box. Getting correct PRF-fitted absolute fluxes requires a correct PRF normalization, and possibly correction factors to take into account how your data have been calibrated.

In general, it is better to keep *Background_Fit* turned off, as it is difficult to get a robust measure of the background within the small fitting area that is best for PRF fitting. An exception might be a very crowded field. If not used, you are relying on the results of the Extract MedFilter module to subtract the backgrounds.

6.5.19 APEX Modules: Aperture Photometry

Command Line Equivalent: run_aperture
Default Output Directory: <output_dir>
Depends On: Detect or Input user list
Relevant Pipelines: All

Important notes: The input image for this module is not set from within the parameter block, but in the APEX Settings module, using the parameter *Use Background Subtracted Image For Aperture*. When using a sky annulus to subtract the background, we recommend using the original image as input rather than the background-subtracted one (i.e. do not turn on the *Use Background Subtracted Image For Aperture* option).

PURPOSE

This module calculates the flux(es) within each specified circular aperture for each source in the extract table. It will optionally subtract a background value from a user-defined annulus. The background value can be set to the median or mode of the good pixels in the annulus. The aperture photometry is calculated using an exact algorithm to compute a fractional aperture-pixel overlap.

INPUT

Number of Apertures: (int) Number of circular apertures to draw around each source, between 1 and 5.

Aperture Radius *N***:** (float) Radius of aperture *N* in pixels.

Min Number Pixels: (int) Minimum number of good pixels required in the annulus. If there are fewer, it will not subtract the background.

Annulus Compute Type: The method used to compute the sky background. Choices are median or mode of the good pixels in the background annulus.

Inner Radius: (float) Inner radius of annulus in pixels.

Outer Radius: (float) Outer radius of annulus in pixels.

Use Annulus: Check this box to subtract the background as calculated using a user-defined annulus.

COMMAND LINE INPUT

```
&APERTURE
 N_Apertures = 3,
 Aperture_Radius_1 = 5.0,
 Aperture_Radius_2 = 6.0,
 Aperture_Radius_3 = 7.0,
 Min_Number_Pixels = 10,
 Annulus_Compute_Type = 'mode',
 Inner_Radius = 15,
 Outer_Radius = 25,
 Use_Annulus = 1,
&END
```

OUTPUT

Output Table (*aperture.tbl*): The aperture fluxes are written into the extract table, along with the number of bad pixels in each aperture. A separate aperture table is also created.

DISCUSSION

The default for this module is to work on median-filtered background-subtracted images if they are there. Because subtracting the background can be difficult, it is generally better to work on the original images, unless the background is complex or you are sure it has been correctly subtracted. To do this, uncheck the *Use Background Subtracted Image for Aperture* box in the APEX Settings module.

For more details, see §8.8 Aperture Photometry.

6.5.20 APEX Modules: Select

Command Line Equivalent: run_select
Default Output Directory: <output_dir>
Depends On: Source Estimate
Relevant Pipelines: All

PURPOSE

This module allows you to select the columns to be written to the final extract table. The module also allows you to select which rows to write out, based on a range of conditions.

INPUT

Columns selection:

srcid: Extracted source number.

detid: Detected source number.

N_PS: Number of point sources needed to fit the data simultaneously. N_PS > 1 indicates passive or active de-blending.

RA: Right Ascension (deg) of source.

delta_RA: RA uncertainty.

Dec: Declination (deg) of source.

delta_Dec: Dec uncertainty.

delta_RAD: RA-Dec cross-term error determined from delta_xy.

x: x-coordinate of source in the mosaic (pixels).

delta_x: x-coordinate uncertainty.

y: y-coordinate of source in the mosaic (pixels).

delta_y: y-coordinate uncertainty.

delta_xy: x-y cross term of the error matrix.

flux: Flux of the point source. Units are specified in the header. Currently set to microJy.

delta_flux: Flux uncertainty from the fit. See also SNR below.

bckgrnd: Background estimate, constant in the fitting area. It is produced if *Background_Fit = 1* is set in the Source Estimate GUI panel or namelist file (or -back is given on the command line).

chi2/dof: χ^2 / dof (number of degrees of freedom) of the fit.

ps_chi2/dof: Partial χ^2 / dof (number of degrees of freedom) of one point source in the fit. Different from the previous column for passive deblending. See below for the explanation of the difference between the two.

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 χ2/dof were smaller than the limit set by MinumumFtol in the namelist.
-1	The maximum number of iterations Max_N_Iteration has been reached.
0	Fitting has not been performed.
1	Successful fitting; terminated because it converged, i.e. the changes in χ2/dof were smaller than the limit set by MinimumFtolSuccess in the namelist.
2	Successful fitting; terminated because the number of iterations exceeded the maximum Max_N_Success_Iteration.

SNR: The ratio of the estimated point-source flux to the noise in the fitting area. The default is the noise computed by the Gaussnoise module if present.

N_dof: The number of degrees of freedom, i.e. the number of pixels used in the fitting minus the number of the fitting parameters.

PRFPortion: The ratio of the sum of the PRF over the pixels used in the calculation to the total PRF.

depth: The number of input images used in fitting this point source.

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.

aperture#: N_apertures columns giving the aperture photometry.

bad_pix#: N_apertures columns giving unusable pixels of the aperture.

ap_unc#: N_apertures columns giving the uncertainty in the aperture flux. Bad value is "-9.99".

Condition Builder: Some simple selection ranges, such as > and <, can be added.

COMMAND LINE INPUT

In Global Parameters:

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

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

OUTPUT

Output Table (_extract.tbl*):** The requested columns are written into the final extract table.

DISCUSSION

Some simple selection ranges, such as > and <, can be added in the namelists in the *Condition Builder*. For example one might choose to select only those sources with *SNR > 5.0*.

6.6 APEX QA Modules

APEX QA performs a subtraction of the detected point sources from the input images or mosaic, depending on whether it is being run in conjunction with APEX Multiframe or APEX Single Frame. The output is the residual image(s). It is invaluable for testing the results of PRF-fitting.

Module listing:

- Initial Setup
- APEX QA Settings
- Point Source Image (APEX QA Multiframe only)
- Mosaic Point Source Image (APEX QA Single Frame Only)
- Mosaic Interpolate (APEX QA Multiframe only)
- Mosaic CoAdder (APEX QA Multiframe only)
- Mosaic Combiner (APEX QA Multiframe only)

6.6.1 APEX QA Modules: Initial Setup

Command Line Equivalent: N/A
Default Output Directory: N/A
Depends On: None
Relevant Pipelines: All

PURPOSE

Set up the input and output files for the loaded flow.

INPUT

Image Stack File: (Multiframe only) The text file containing the list of data files that you used as input into the APEX Multiframe pipeline. See Chapter 3 and Table 3.1 for more information on the input file requirements and format.

Mosaic File Name: (Single Frame only) The name of the single image that you used as input into the APEX Single Frame pipeline.

FIF File: (Single Frame only) The name of the *FIF.tbl* file that describes the input mosaic. This was either produced by the Mosaic pipeline and can be found in the output directory of the Mosaic run.

Output Directory: The directory that you want to set as your output directory for this run.

Multi-processing Mode: Switch to use multi-processing. See Discussion.

Processors for Multi-processing: Number of processors to use. See Discussion.

Optional Input and Mask Files: (Multiframe Pipelines only)

Sigma List File: The text file containing the list of uncertainty images that correspond to the files listed in *Image Stack File* (usually the **bunc.fits* files; see Table 3.1 for more information).

DCE Status Mask List: The text file containing the list of mask images that flag temporarily-bad pixels in the files listed in *Image Stack File*. There should be a mask file for each of the input data files (these depend on the instrument, but will be labeled something like **msk.fits*; see Table 3.1 for more information).

Fatal Mask Bit Pattern: This is the bit pattern that defines which type of flagged problems that you wish to set as fatal when combining the input images. See §8.11 for more information. There is a separate Fatal Bit Pattern for each type of mask that is being used as input (i.e. for the DCE Status masks, the RMasks and the PMask).

Rmask List File: The text file containing a list of outlier mask images. This option allows you to specify a list of RMask images (e.g. those created by the Mosaic pipeline) to reject pixels in the input images that have been identified as outliers.

Pmask FITS File: The permanently-damaged pixel mask for the detector. This is a single FITS image, flagging the permanently damaged pixels in the instrument arrays. These files are stored in the *<mopex_dir>/cal/* directory. There is a single PMask for each of the MIPS and IRS channels, but the IRAC ones have changed with time. See the file *<mopex_dir>/pmasks.README* for information on which PMask is applicable to your data. Note that the date that the data were taken is given in the DATE_OBS header keyword (**not** DATE - this is the date the file was written).

FIF file: The Fiducial Image Frame file (see §6.5.5). This is the file called (by default) *FIF.tbl*, that was produced by either the Mosaic or APEX Multiframe pipeline. It can be found in the output directory of those runs.

COMMAND LINE INPUT

In the Global Parameters section of the namelist:

```
IMAGE_STACK_FILE_NAME = <working_dir>/imagelist.txt
MOSAIC_FILE_NAME = <working_dir>/mosaic.fits
FIF_FILE_NAME = <working_dir>/FIF.tbl
OUTPUT_DIR = <working_dir>/output
SIGMALIST_FILE_NAME = <working_dir>/sigmalist.txt
DCE_STATUS_MASK_LIST = <working_dir>/masklist.txt
DCE_Status_Mask_Fatal_BitPattern = 32544
RMASK_LIST = <working_dir>/rmasklist.txt
RMask_Fatal_BitPattern = 7
PMASK_FILE_NAME = <mopex_dir>/cal/sep07/sep07_ch1_bcd_pmask.fits
PMask_Fatal_BitPattern = 32767
verbose = 1
NICE = 1
save_namelist = 1
```

Verbose, *NICE* and *save_namelist* are only available on the command line. See §9.4.2.3 for more information.

OUTPUT

None

DISCUSSION

This module sets up the input files and top-level output directory for the first pipeline in the flow. Subsequent pipelines that have been inserted into the flow will use the output from the previous one (i.e. if you have inserted Mosaic and APEX into your reduction flow, Mosaic will use the Initial Settings files as input, but APEX will use the modified files output by Mosaic).

NEW Note that with version 18.5.0 one can use multiprocessing to speed up MOPEX tasks. The tasks Overlap, Mosaic, Apex (Multi) and Apex User List (Multi) now allow multiprocessing of some steps. In the GUI, this is set in Initial Setup, Multi-Processing Mode. The options are "on", "off", and "manual". The default is "on" -- this grabs 3/4 of the available processors. Setting "manual" lets the user set the number of processors used. You should see speed-ups of at least 2x.

If using command-line, add these lines at the top of your namelist:
do_multiprocess = on | off | manual default = off
ncpu_multiprocess = 1 default = 1
If "manual" is set, set ncpu_multiprocess to the number of processors to use.

6.6.2 APEX QA Modules: APEX QA Settings (Multiframe)

Command Line Equivalent: None
Default Output Directory: <output_dir>
Depends On: None

PURPOSE

Set up the input files for the APEX QA Multiframe pipeline.

INPUT

Pixel by Size: Check this to set the output pixel size in degrees. The default pixel size depends on the instrument and channel: see the templates for the default sizes.

Pixel Size X (deg): Set the X-size of the output pixels in degrees. The pixel size in the X-direction is quoted as a negative value to comply with convention. A positive value will generate an error message.
Pixel Size Y (deg): Set the Y-size of the output pixels in degrees.

Pixel By Ratio: Check this to set the output pixel size by ratio of the input pixel to the output pixel, i.e. to make the output pixels half the size of the input pixels (over-sample the mosaic), set the values of *Pixel Ratio X(Y) = 2*. The ratio is taken after correction for geometric distortion.

Pixel Ratio X: Set the pixel ratio in the X-direction.
Pixel Ratio Y: Set the pixel ratio in the Y-direction.

Use PRF File Name: Select this if a single fixed PRF file was used to fit the sources in the APEX Multiframe pipeline

Use PRF Map File Name: Select this if a variable PRF map file was used to fit the sources in the APEX Multiframe pipeline.

PRF File Name: The name of the PRF file that was used in source fitting.
PRF Map File Name: The name of the PRF Map file that was used in source fitting.

Extract Table: Table of extracted sources, usually from an APEX Multiframe or APEX User List Multiframe pipeline. By default, this file is named extract.tbl, but some users may choose to use extract_raw.tbl instead (see Discussion).

Delete Intermediate Files: Check this to delete all of the intermediate files created by the APEX QA pipeline. This leaves only the input files and the files created by the final module. This is useful for saving disk space once you are satisfied with your reduction setup. Note that MOPEX will prompt you with a pop-up dialogue box when you set the flow running, to be sure that you meant to select this irreversible option.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.** Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

COMMAND LINE INPUT

In the Global Parameters section of the namelist:

```
MOSAIC_PIXEL_SIZE_X = -0.00033889
MOSAIC_PIXEL_SIZE_Y = 0.00033889
MOSAIC_PIXEL_RATIO_X = 1
MOSAIC_PIXEL_RATIO_Y = 1
PRF_file_name = cal/apex_sh_IRAC1_col129_row129_x100.fits
PRFMAP_FILE_NAME = cal/ch1_prfmap.tbl
EXTRACTION_TABLE = <output_dir>/extract.tbl
delete_intermediate_files = 0
USE_REFINED_POINTING = 0
```

OUTPUT

none

DISCUSSION

This module sets up the input files for the APEX QA Multiframe pipeline. If you run APEX QA and find that some sources were not subtracted in the final image, check whether the Select module was run in the APEX (User List) Multiframe pipeline. It is possible that some extracted sources were thrown out by the Select condition. In this case you may choose to use *extract_raw.tbl* as the *Extract Table*.

6.6.3 APEX QA Modules: APEX QA Settings (Single Frame)

Command Line Equivalent: None
Default Output Directory: <output_dir>
Depends On: None

PURPOSE

Set up the input files for the APEX QA Single Frame pipeline.

INPUT

Mosaic PRF File Name: The name of the PRF FITS file that was used for source fitting in APEX Single Frame.

Extract Table: The name of the table of extracted sources that was produced by APEX (User List) Single Frame. This is usually called *extract.tbl*, but in some cases users may choose to use *extract_raw.tbl* instead (see Discussion*)*.

COMMAND LINE INPUT

In the Global Parameters section of the namelist:

```
MOSAIC_PRF_FILE_NAME = cal/ apex_sh_IRAC1_col129_row129_x100.fits
EXTRACTION_TABLE = <output_dir>/extract.tbl
```

OUTPUT

None.

DISCUSSION

This module sets up the two input files for the APEX QA Single Frame pipeline. If you run APEX QA and find that some sources were not subtracted in the final image, check whether the Select module was run in the APEX (User List) Single Frame pipeline. It is possible that some extracted sources were thrown out by the Select condition. In this case you may choose to use *extract_raw.tbl* as the *Extract Table*.

6.6.4 APEX QA Modules: (Mosaic) Point Source Image

Command Line Equivalent: create_residual_images (for *apex.pl*) or create_residual_mosaic (for *apex_1frame.pl*).
Default Output Directory: <output_dir>
Depends on: Output of APEX (User List) Multiframe or APEX (User List) Single Frame

Important Notes: The name of this module is "Point Source Image" in the APEX QA Multiframe pipeline, and "Mosaic Point Source Image" in the APEX QA Single Frame Pipeline. The functionality is the same.

PURPOSE

This module is used to subtract the sources from the input images or the input mosaic to create a residual image(s). This is extremely useful for testing how well the PRF fitting in APEX is working.

INPUT

PRF Resample X(Y) Factor: (int) This is the same *PRF Resample X(Y) Factor* as used in the Source Estimate module. It is the ratio of the input image pixel size to the PRF pixel size in the x- and y-direction.

PS Image X(Y) size: (int, odd) If present and smaller than the PRF array size, the PRF array will be cut to this size. The units are data pixels.

Flux Column Name: (char) The name of the column in the *Extract Table* that corresponds to the flux of each point source.

Input RA Dec: The units of the positional input data. Choices are "Pixel Coordinates" and "RA & Dec".

X(Y) Column Name: (char) The name of the X and Y positional columns to be read from the *Extract Table*. Options are 'RA' and 'Dec' or 'x' and 'y'. If 'x' and 'y' are used, the Fiducial Image Frame table must also be specified in the Initial Setup.

Normalization Radius: (int) This is the same parameter as used in the Source Estimate module. The PRF flux is normalized within this radius (in PRF pixels). By default, the PRF is normalized to the whole PRF array.

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.

Residual output subdirectory: The name of the output subdirectory for the point source subtracted image(s). The default is *Residual-ApexQA*.

COMMAND LINE INPUT

When running in Multiframe mode (i.e. on the output of *apex.pl*):

```
&POINTSOURCEIMAGE
```

```
 PRF_ResampleX_Factor = 100,
 PRF_ResampleY_Factor = 100,
 Input_RA_Dec = 1,
 X_Column_Name = 'RA',
 Y_Column_Name = 'Dec',
 Normalization_Radius = 1000,
 Outer_Radius = 44,
 Hole_Radius = 21,
&END
```

When running in Single Frame mode (i.e. on the output of *apex_1frame.pl*) the parameters are identical but the module block starts with the keyword:

```
&MOSAIC_POINTSOURCEIMAGE
```

OUTPUT

The residual images are written to the specified output directory and are called *residual*.fits*

DISCUSSION

This module is used to subtract the sources from the input images or the input mosaic to create a residual mosaic. This is extremely useful for testing how well the PRF fitting is working. The two modes (Single Frame and Multiframe) use different keywords for triggering the module to run, and for specifying the parameter block. Parameters *PRF Resample X(Y) Factor* and *Normalization Radius* are the same parameters as used in the Source Estimate module, and should be set to the same values as previously used.

Chapter 7. PRF Estimate

PRF Estimate allows the user to make custom PRF files from their own data. In most cases, this pipeline is not necessary for Spitzer data, because the standard PRF files will be sufficient.

PRF Estimate was not designed for use with undersampled data, for example Spitzer IRAC Channels 1 and 2 data, and should not be used for such data.

7.1 Overview

The script *prf_estimate.pl* performs PRF estimation using either a single mosaic frame or a stack of input BCD images. The coordinates of the point sources in the input images to be used to derive a PRF are given in an input table. Optionally, background subtraction can be performed on the input images. A set of postage stamp images is cut out from each input image centered on each input point source position. The precise flux and point source positions can be optionally estimated by fitting a Gaussian to the postage stamp images. There is an option of rejecting and replacing outlier pixels and outright rejecting bad images. The postage stamp images are resampled and shifted using bicubic interpolation and combined into one final PRF image. The software has an option of deriving a set of PRFs mapped into the detector array.

7.2 Basic Input Requirements

In order to carry out PRF generation, MOPEX will expect some or all of the following files as input. See the sections on Input for more information on the format of these files.

- Either a text file containing a list of input data images from which the PRF is to be measured (e.g. Spitzer BCD frames), OR
- A single input image from which the PRF is to be measured;
- A text file containing a list of point sources to be used by PRF Estimate to create the PRF.

7.3 Running PRF Estimate

To load a PRF Estimate either start from a MOPEX template namelist (e.g. *File > New PRF Estimate Pipeline*), load an existing PRF Estimate namelist file (*File > Read Name List*) or load an empty flow and insert a PRF Estimate pipeline from there (*File > New Empty MOPEX Pipeline*, then go to e.g. *Insert PRF Estimate* at the top of the flow window and choose *Empty Pipeline*). If you choose the last option, MOPEX will load an empty PRF Estimate flow to which you can add and subtract modules, and modify the input parameters.

7.4 PRF Estimate Processing Stages

Background Subtraction

Background subtraction is performed by the same MedFilter module that is used in the Mosaic pipeline. 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 *Outliers / Window* highest pixels from each median window. If *Outliers / Window* is set to 0 the program calculates the regular median. There is also the option to use the faster Sbkg background fitting method, like that used by SExtractor.

Postage Stamp Image Creation

The Crop Stack module cuts out postage stamp images from the input images around each point source from the input *Point Source List*. If a single input image name is specified, then the postage stamps are cut out from that image. If a list of images is specified, then the postage stamp images are cut out from each image listed therein. The position of the point sources in the input list can be given in sky coordinates or in pixel coordinates. In the latter case, a Fiducial Image Frame table is required (see §5.6.4).

PRF Estimation

The PRF Estimate module computes a PRF image(s) (see §8.7) based on the input images of an isolated point source, not necessarily the same source, just a single source. An attempt is made to automate the process of weeding out bad images from the process of PRF estimation using several outlier rejection mechanisms and using data fitting to determine input point sources flux and positions. However, it is up to the users to verify that these efforts were ultimately successful, especially if they suspect that the PRF produced by the program has some problems.

The software has an option of creating either a single PRF image for the whole array or a set of PRF images for the different sectors of the detector array. The mapping of the detector

array into sectors is specified in a file called a PRF map. The assignment of postage stamps to different parts of the array is performed in the Split By Array Position module. After that, the PRF Estimate module runs consecutively on each partial list and a PRF image is created and saved for each detector array sector.

There are four steps in creating a PRF: NaN replacement, exact point source position and flux estimation, outlier rejection, and coaddition.

1. **NaN Replacement**: The program replaces each NaN in the input image with the median of a number of the neighboring pixels.

2. **Extract Point Source Flux and Position Estimation**: For both outlier rejection and coaddition, each input postage stamp image is shifted to the common grid, in which the peak of the point source position coincides with the center of the central pixel. By default the position of the point sources in the input images are computed as flux weighted centroids. Each postage stamp image is normalized by its flux. The flux can be given in the input point source list. If it is, then the column name of the appropriate column should be 'flux' or can be specified in the PRF Estimate module by the parameter *Flux Column Name*. If the flux is not given in the input table, then it is found by summing all the pixel values in the input image.

   The flux and the exact position of the point source in each Postage Stamp image are estimated by fitting the data with a Gaussian. The following quantity is minimized:

$$\chi^2 = \sum_{i \in W} \left(s(i) - G(i) \right)^2$$

$$G(i) = \frac{f}{2\pi} \cdot \exp\left(- \frac{(x_i - x_c)^2 + (y_i - y_c)^2}{2\pi\sigma^2} \right)$$

Equation 7.1

   with respect to the flux f, width σ, and position (x_c,y_c) of the point source in the Postage Stamp image. Here $s(i)$ is the pixel value of the input image for pixel i, W is the area in center of the image. The size of the area is specified with the PRF Estimate module parameters *Fit X, Fit_Y* both defaulting to 5 pixels. Simulated annealing method is used to minimize χ^2.

3. **Outlier Rejection**: Two kinds of outlier rejection are performed. The first kind is when the whole image is rejected based on some criteria. The second kind is when only some pixels in an image are detected as outliers and then replaced with some combination of the neighboring good pixels.

   Image rejection is done based on several different criteria. Two of the criteria are: having too many NaN's in the input image, or having a NaN pixel in the central box. Two criteria are based on the results of the Gaussian point source fitting (see above), specifically using the values of σ and χ^2 for each image. The third criterion is the number of outlier pixels detected in the image. The fourth criterion is having an outlier pixel in the central box of the image.

   After Gaussian fitting is completed for all input images the sets of σ and χ^2 are analyzed. Image *j* is declared an outlier, if any of the three conditions below are met:

$$\sigma_j > \sigma_{median} + Outlier\_Threshold \cdot \sigma_\sigma$$
$$\sigma_j < \sigma_{median} - Outlier\_Threshold \cdot \sigma_\sigma$$
$$\chi^2_j > Outlier\_Threshold \cdot \chi^2_{median}$$

Equation 7.2

   Here the subscript *median* indicates the median value of the quantity and the subscript σ indicates its standard deviation.

   Next step is detecting and rejecting pixels outliers. For the outlier rejection step, no resampling is performed. The input images are shifted to the common grid using the bicubic interpolation. The shifted images are stacked up and outliers are found. The parameter *Outlier Threshold* specifies the number of trimmed sigmas below and above the trimmed mean to be used to determine the outliers. If a particular image has too many outliers it is completely rejected from further processing. "Too many" is determined by two parameters: if the fraction of the outliers in an input image is greater than *Max Ratio Outliers* and the total number of the outliers is greater than *Max Number Outliers* then the image is rejected. An image is also rejected if the outlier pixel is found in the center of the image, the center being defined the same way as it is for the NaN pixels: it is specified with the input parameters (*Center Box X, Center Box Y*), both defaulting to 7. If an image has a number of outliers but is not rejected, then the values of the outlier pixels is replaced with the median of the neighboring pixel values in exactly the same way the

NaN's are replaced in the input images. There is an additional step of re-projection of the shifted images back on the input images, since the outlier detection is performed on the shifted images. A pixel in an outlier image is considered an outlier if the fractional area overlap of the pixel with the outlier pixels from the shifted image is greater than *Min Outlier Overlap*.

4. **Resampling and Combining**: For coadding, the input images are resampled and shifted to the common grid using the bicubic interpolation. The ratio of the pixel sizes to the sampling distance is given by the parameters *PRF Resample X Factor* and *PRF Resample Y Factor*. The resampled and shifted images are stacked up and a simple mean and standard deviation are found for each pixel position.

Output

If the Split By Array Position module is included, then the names of the output files are read from the PRF map, in which case the parameters *PRF filename* and *PRF Sigma Filename* are ignored if given in the Initial Setup. Otherwise, these parameters can be used to specify the names of the output images with the default names *PRF.fits* and *PRFSigma.fits*. A coverage map is output if *PRF Coverage Filename* is specified in the Initial Settings.

7.5 PRF Estimate Modules

The following pages contain full descriptions of the modules that are available in PRF Estimate.

Module listing:

- Initial Setup
- PRF Estimate Settings
- MedFilter
- Crop Stack
- Split By Array Position
- PRF Estimate

7.5.1 PRF Estimate Modules: Initial Setup

Command Line Equivalent: None
Default Output Directory: N/A
Depends on: None

PURPOSE

Set up the initial input files for the PRF Estimate flow. You can set either the Input File or the Image Stack file, depending on whether you wish to run PRF Estimate on a single image or a stack of images.

INPUT

Input File: The single input image to be used to measure the PRF, if you wish to use a single image as input. Either set this or *Image Stack File*.

Image Stack File: The list of images to be used to measure the PRF, if you wish to use a stack of images as input. Either set this or *Input File.*

Point Source List: The list of point sources that you wish to use to calculate the PRF. The table must be in IPAC format, and is usually a source list output by APEX, edited to only include the sources that you wish to use.

Output Directory: The output directory that you wish to use for this PRF Estimate run.

COMMAND LINE INPUT

```
INPUT_FILE_NAME = mosaic_M1.fits
IMAGE_STACK_FILE_NAME = M1/imagelist.txt
POINT_SOURCE_LIST = prf_stars.tbl
OUTPUT_DIR = output_prfestimate_M1
verbose = 1
NICE = 1
save_namelist = 1
```

Verbose, *NICE* and *save_namelist* are only available on the command line. See §9.4.2.3 for more information.

OUTPUT

None

DISCUSSION

You have the choice of using either a single input file or a stack of BCDs to create the PRF. Entering a filename for one of the parameters, *Input File* or *Image Stack File*, in the GUI will cause the other to disappear from the Initial Setup options. On the command line, be sure to enter only one of the two inputs.

7.5.2 PRF Estimate Modules: PRF Estimate Initial Setup

Command Line Equivalent: None
Default Output Directory: N/A
Depends on: Initial Setup

PURPOSE

Set up the input files specific to the PRF Estimate pipeline.

INPUT

AnnealStatus Filename: The output file name of a table containing diagnostic information about the fitting used to estimate point source flux and position. The default file name is *FitStat.tbl*.

PRF Coverage Filename: The name of the output coverage map corresponding to the output PRF image. Default is *PRFCov.fits*.

PRF Filename: The name of the output PRF image. Default is *PRF.fits*.

PRF Sigma Filename: The name of the output uncertainty image corresponding to the output PRF. Default is *PRFSigma.fits.*

Star Filename: The name of an optional output image file created using the computed PRF and placing the point source at the center of the central pixel of the image.

StatusList Filename: The name of the status file listing all the input images, along with whether or not they were used to create the PRF. If they were not used, the reason for rejection is also listed.

Use Refined Pointing: Check this to use the alternate FITS pointing keywords created by the offline pointing refinement script (*pointing_refine.pl*). **Do not switch this on unless you have manually run pointing refinement outside of the normal BCD pipeline products.**

Most users will never use this option, as pointing refinement is not recommended for Spitzer data. If you believe that there is a problem with the pointing of your data with respect to e.g. the 2MASS catalog, please email the Spitzer Helpdesk for assistance.

COMMAND LINE INPUT

In the Global Parameters section of the namelist:

```
AnnealStatus_file_name = FitStat.tbl
PRF_Coverage_file_name = PRFCov.fits
PRF_file_name = PRF.fits
PRF_Sigma_file_name = PRFSigma.fits
Star_file_name = star.fits
StatusList_file_name = Status.txt
USE_REFINED_POINTING = 0
```

OUTPUT

None

DISCUSSION

None

7.5.3 PRF Estimate Modules: MedFilter

Command Line Equivalent: run_medfilter
Default Output Directory: <output_dir>/Medfilter-PRF-Estimate
Depends On: Initial Setup; PRF Estimate Initial Setup

PURPOSE

This module performs a background subtraction of the individual input images. It is another occurrence of the MedFilter module used by Overlap, Mosaic and APEX. There are two options: the Median case (default) and the Sbkg case, which uses a background estimate like that of SExtractor.

INPUT

Window X, Y: (int) the X, Y size in input pixels of the window used to compute the background value.

Outliers / Window: (int) the number of high outlier pixels rejected from the X*Y window when computing the Median background. For a very crowded field, the fraction of rejected pixels should be higher than for a less crowded field. Values of a few percent should be acceptable for uncrowded fields. If *Outliers / Window* is set too high, the background will be under-estimated.

SExtractor background filter size: (int) Median filter box size for the Panels when the *Use SExtractor background estimation* option is turned on. A value less than 2 means no filtering. If greater than the minimum number of panels across an image, it will use the minimum.

Use SExtractor background estimation: Checking this box invokes the Sbkg background estimate based on SExtractor. Leaving it unchecked causes MOPEX to use the default Median estimate.

Med Filter output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Medfilter-PRF-Estimate.

COMMAND LINE INPUT

```
&MEDFILTER
 Window_X = 45,
 Window_Y = 45,
 N_Outliers_Per_Window = 500,
 Use_Sbkg_for_Med = 1,
 Sbkg_Filt_Size_for_Med = 3,
&END
```

In Global Parameters:

```
MEDFILTER_DIR = Medfilter-PRF-Estimate
```

OUTPUTS

Generated Fits Files (_minback.fits*):** The background subtracted, individual images. A list of the generated images is also written out.

DISCUSSION

With the default Median option, the program computes a background value using the median of a running rectangular window of *Window X* by *Window Y* pixels, after omitting the *Outliers / Window* highest pixels. This is done for each pixel so can be very slow.

If *Use SExtractor background estimation* is set, the module switches to the Sbkg background estimation based on that of SExtractor (Bertin and Arnouts, AASupp 117, 393, 1996). The image is divided into Panels with size given by *Window X, Y*. In each Panel, iterative clipping is used to find a single estimate of the background in the Panel. You can optionally median filter the Panel background values, e.g. to avoid ones where bright objects skewed the background estimate. The median filter size is given by *SExtractor background filter size.* It then interpolates the Panel values to find the background at each pixel. This is much faster than calculating the median of a big window at each pixel.

Both background estimates generally give reasonable results but if the data volume is large, the Sbkg option is strongly recommended, as it is much faster. It also does not require an estimate *of Outliers / Window*.

7.5.4 PRF Estimate Modules: Crop Stack

Command Line Equivalent: run_crop_stack
Default Output Directory: <output_dir>/Crop_Stack
Depends On: MedFilter

PURPOSE

This module cuts out postage stamp images from the input images around each point source from the input point source list given in *Point Source List*. If a single image name is specified (*Input File*), the postage stamps are cut from that image. If a list of images is specified (*Image Stack File*), then the postage stamp images are cut from each image in the list.

INPUT

PostStamp X(Y) size: (int) The X and Y size, in pixels, of the postage stamp to cut out around each point source.

X(Y) Column Name: The name of the columns within the input file that give the position of each point source. The position should be specified as either *RA, Dec* or *X, Y*. If using the output of APEX as the *Point Source File*, set this to *RA*, *Dec*.

Transformation Type: The type of transformation required to use the input positions. If the positions are given in sky coordinates then the transformation type should be set to *Sky To Plane*. If the positions are given in pixel coordinates in some image frame (e.g. mosaic image), then the transformation type should be set to *Plane To Plane*. In this case a file describing the image frame should also be given (e.g. an *FIF.tbl* file).

Crop Stack output subdirectory: The subdirectory of <*output_dir*> that you wish to use for the output files. Default is Crop_Stack.

COMMAND LINE INPUT

```
&CROP_STACK
 PostStamp_Xsize = 21,
 PostStamp_Ysize = 21,
 X_Column_Name = 'RA',
 Y_Column_Name = 'Dec',
 TransformationType = 'SkyToPlane',
&END
```

In Global Parameters:

```
CROP_STACK_DIR = Crop_Stack
```

OUTPUTS

Generated FITS files: The stack of postage stamps for the point sources in the input list.

DISCUSSION

See the discussion of the processing stages in §7.4 for more information on the centering of each point source.

7.5.5 PRF Estimate Modules: Split By Array Position

Command-Line Equivalent: split_by_array_position
Default Output Directory: N/A
Depends On: Input files

PURPOSE

This module allows the user to direct MOPEX to divide the detector array into a number of sectors and generate a separate PRF file for each. The array positions are given in the input PRF Map table specified with the input keyword *PRF Map Filename*.

INPUTS

PRF Map Filename: The name of the PRF Map file specifying the spatial areas of the array, and matching a PRF file to each section. See §8.7 for the format of the PRF Map table.

COMMAND LINE INPUT

In Global Parameters:

```
PRFMAP_FILE_NAME = ch1_prfmap.tbl
```

OUTPUTS

PRF#? Generated FITS Files (*img_i_j.fits*): The postage stamps corresponding to the point sources within each region. These files are given the name *img_i_j.fits*, corresponding to the i^{th} point source in the j^{th} image in the input stack.

DISCUSSION

See the discussion of the variable Pixel Response Function in §8.7 for more information about the syntax of the PRF map.

7.5.6 PRF Estimate Modules: PRF Estimate

Command Line Equivalent: run_prf_estimate

Default Output Directory: Output directory
Depends On: Crop Stack

PURPOSE

This module combines the postage stamp images generated by the Crop Stack module into the final PRF.

INPUT

Min Number Replacement Pixels: (int) The minimum number of pixels neighboring a NaN to be used for computing the median value that will replace the NaN.

Reject NaNs: (int) The maximum number of NaNs to be replaced before an image is rejected.

Outlier Threshold: (int) The number of trimmed sigmas below and above the trimmed mean to determine outliers from the stack of registered, interpolated postage stamps.

Min Outlier Overlap: (float) The fractional overlap between an original pixel and an outlier pixel in the common, interpolated grid that will trigger rejection of that original pixel.

Max Ratio Outliers: (float) The maximum allowable fraction of outlier pixels in an input postage stamp before that image is rejected. This condition AND the condition for *Max Number Outliers* must be exceeded before an image is excluded.

Max Number Outliers: (float) The maximum allowable number of outlier pixels in an input postage stamp before that image is rejected. This condition AND the condition for *Max Ratio Outliers* must be exceeded before an image is excluded.

Flux Column Name: (char) The name of the column in the input point source list corresponding to the flux of each source. If not given, the flux will be estimated as the sum of all the pixels within each postage stamp. E.g. *flux*

PRF Resample X(Y) Factor: (int) The ratio of the input image pixel size to the sampling distance in the X, Y directions in the final coaddition.

Zero Edge: (int) The size in pixels of the margin to the PRF to be set to 0.

Fit Seed: (int) The seed for random number generation to be used by simulated anneal fitting. If set to 0, the current time is used. If negative, the fitting is not performed.

Fit X(Y): (int) Fitting area size in pixels.

Center Box X(Y): (int) The size in pixels of the region of the point source images to be considered central. If pixels within the central region have NaN values, then that postage stamp is rejected from the stack.

COMMAND LINE INPUT

```
&PRF_ESTIMATE
 MinNumber_Replacement_Pixels = 8,
 Reject_NaNs = 0,
 Outlier_Threshold = 3,
 Min_Outlier_Overlap = 0.5,
 Max_Ratio_Outliers = 0.1,
 Max_Number_Outliers = 0,
 Flux_Column_Name = 'flux',
 PRF_ResampleX_Factor = 4,
 PRF_ResampleY_Factor = 4,
 Zero_Edge = 0,
 Fit_Seed = 0,
 Fit_X = 5,
 Fit_Y = 5,
 Center_Box_X = 7,
 Center_Box_Y = 7,
&END
```

OUTPUT

Generated Anneal Status File: The table containing diagnostic information about the fitting used to estimate the point source flux and position.

Generated PRF file: The output PRF file. Note that this file does not contain WCS information, so the MOPEX GUI will give a warning when the FITS file is viewed.

Generated PRF Coverage File: The corresponding coverage file.

Generated PRF Sigma file: The corresponding uncertainty file.

Generated Star file: The corresponding Star file.

Generated Status List file: Lists whether each input postage stamp image was included in the final stack and, if not, the reason it was rejected.

NOTE: If the Split By Array Position module is turned on, the output files will be replicated for each region of the PRF Map. The output file names will have have *_i* appended to them for the i^{th} region, and the GUI will label the output with *PRF#i.*

DISCUSSION

This module combines the postage stamps into the final PRF. The postage stamps are resampled and shifted to the common grid using bicubic interpolation. The stack is combined using a simple mean, and the standard deviation is found for each pixel position. During the processing, a variety of outlier rejection tests are applied to each point source image (see §7.4). Only those that pass each test are included in the final stack. The output status file describes the fate of each postage stamp.

In addition to outlier rejection, the module replaces each NaN in the input image with the median of a number of the neighboring pixels. If *Reject NaNs* is greater than 0, then bad pixels are only replaced if their number is less than or equal to *Reject NaNs*. Otherwise the image is excluded from the list of images to be used for PRF estimation. Another reason for rejecting an image is if a NaN pixel is found in the center of the input image. The size of the center is specified with the namelist parameters *Center Box X(Y)*, both defaulting to 7.

Chapter 8. Basic Concepts in MOPEX

There are a number of basic concepts in MOPEX that are repeatedly used by a number of modules. This section describes the following:

- Tiling
- Outlier Detection
- Single Frame Outlier Detection
- Multiframe Temporal Outlier Detection
- Dual Outlier Detection
- Box Outlier Detection
- Image Segmentation
- Image Interpolation
- Background Matching Algorithm
- PRF fitting in MOPEX
- The Spitzer Point Reponse Function (PRF)
- Aperture Photometry
- Uncertainty Estimation
- User List Creation for APEX

8.1 Tiling

There is no precise limit to the size of a mosaic that MOPEX will produce. However, large datasets may encounter computer memory limitations, so most MOPEX modules employ Tiles to reduce the number of pixels that need to be considered at one time.

Tiles are subsections of the final mosaic image. The size of tiles is set by the user, based on the computer memory considerations. It should be small enough so that all input images overlapping one tile can be held in memory without swapping. MOPEX first produces the tiles for each final mosaic image with the tile sizes specified by the user. Then all the tiles are glued together into a single mosaic image by the Mosaic Combine module.

The first step is estimation of the tile geometry. The interpolated and co-added images are defined on the same Fiducial Image Frame (FIF; see e.g. §4.3.4). The suggested tile sizes in the x- and y-directions *Tile Max X* and *Tile Max Y* are parameters in a number of modules. If both FIF dimensions `NAXIS1` and `NAXIS2` are smaller than *Tile Max X* and *Tile Max Y*, correspondingly, then there is only one output co-added image.

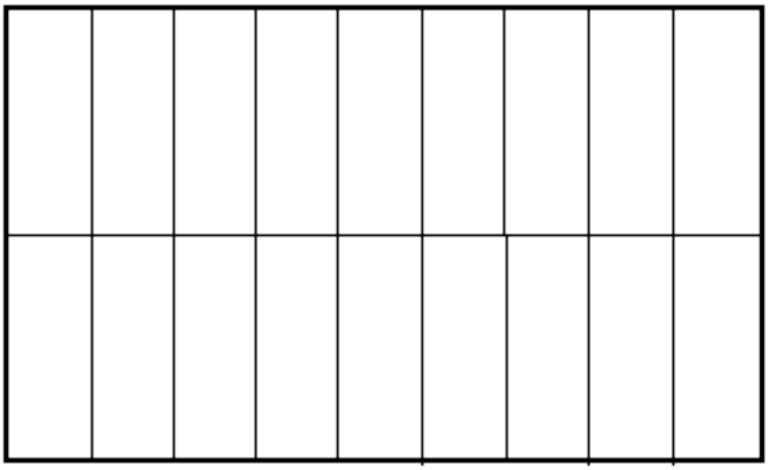


Figure 8.1: An example of tiling a FIF. Suppose NAXIS1 = 4450 and NAXIS2 = 1024. The suggested sizes *Tile Max X* = *Tile Max Y* = 500. In this case there are 2 tiles in the y-directions of the same size Tile Y = 512 and 9 tiles in the x-direction, 8 of the size Tile X = 494 and the last tile of the size Last Tile X = 498.

If the FIF dimensions are greater than *Tile Max X* or *Tile Max Y*, then the co-added image is tiled. Given the dimensions of the FIF (`NAXIS1` and `NAXIS2`) the program finds the number of the tiles, their sizes and relative positions (offsets) with respect to the FIF. The simple algorithm below assures that all the tiles are of almost the same size regardless of what the FIF size is (see Figure 8.1).

N_X = NAXIS1 / (*Tile Max X*);

If (NAXIS1 % *Tile Max X* > *Tile Max X/2*) N_X = N_X +1;

Tile X = NAXIS1 / N_X;

Last Tile X = Tile X + (NAXIS1%N_X).

The % (modulo) operator yields the remainder from the division of the first argument by the second. The calculations in the y-direction are identical. The tile sizes and offsets from the origin of the FIF along with their names are written in output IPAC table *coadd_tiles.tbl*.

8.2 Outlier Detection

Outlier rejection is central to the mosaicking process. MOPEX employs four algorithms to detect radiation hits and moving objects:

1. Single Frame Outlier Detection, implemented as the Detect Radhit module. It represents spatial filtering of input images. It requires the background-subtracted images created by MedFilter.

2. Multiframe Temporal Outlier Detection implemented by the Mosaic Outlier module. This algorithm is best suited for high-coverage data (i.e. many input frames per pixel).

3. Dual Outlier Detection: a more complicated algorithm that uses both spatial and temporal information. The implementation involves a series of modules: Detect Outlier, Mosaic Projection, Mosaic Dual Outlier, and Level. It requires the background-subtracted images created by MedFilter. It is best suited for low-to-medium coverage data.

4. Box Outlier Detection: a method designed to use both the temporal and spatial information like the Dual Outlier, but using the statistical analysis of the kind used by the Multiframe Temporal Outlier. It is best suited for low-to-medium coverage data.

The four detection results can be combined into one mask - the RMask - by module Mosaic RMask. **Warning: running the modules for these outlier schemes does not mean that they are automatically used to create the RMask file.** The default rejection scheme used to create the RMask is Single Frame Outlier Detection. In order to use the other schemes you must check the relevant boxes in the Mosaic RMask module (*Use Outlier for Rmask*; *Use Dual Outlier for Rmask*; *Use Box Outlier for Rmask*), and set the *RMask Fatal Mask Bit Pattern* in the Initial Setup to use the relevant bits (see §8.11: Fatal Mask Bit Patterns for more information). The equivalent on the command line is to set *RMask_Fatal_Bit_Pattern* and the following lines in the General Settings section of your namelist:

`USE_OUTLIER_FOR_RMASK = 1` (use Multiframe Temporal Outlier Detection)
`USE_DUAL_OUTLIER_FOR_RMASK = 1` (use Dual Outlier Detection)
`USE_BOX_OUTLIER_FOR_RMASK = 1` (use Box Outlier Detection)

The *RMask Fatal Mask Bit Pattern* is set as a separate step to allow users to compare the different outlier rejection schemes without having to run all of the outlier algorithms several times. Instead of re-running all of the modules, users can run the whole flow once, with all of the outlier rejection schemes enabled to create the RMask. Once the RMask has been created, the outlier schemes can be compared by turning off all of the Mosaic modules except for Mosaic

Reinterpolate, Mosaic Coadd and Mosaic Combine, and re-running Mosaic (*mosaic.pl)* with different *RMask Fatal Mask Bit Pattern* settings. You can also adjust the thresholds for some of the outlier schemes within RMask itself. Since the outlier rejection modules are among the most complex steps, this can save a lot of time, especially for large datasets.

The module Mosaic Coverage is run prior to Mosaic RMask, because the latter uses coverage thresholds to include the results of the temporal and dual outlier detection.

8.2.1 Single Frame Outlier Rejection

The single frame radhit detection represents spatial filtering of input images. It is implemented in the Detect Radhit module. It is a variant of image segmentation and is based on the idea that when a relatively low detection threshold is applied, for each bright point source a large number of pixels will be detected above the threshold. Bright detections with small areas are, therefore, classified as radhits. It should be applied to background subtracted images.

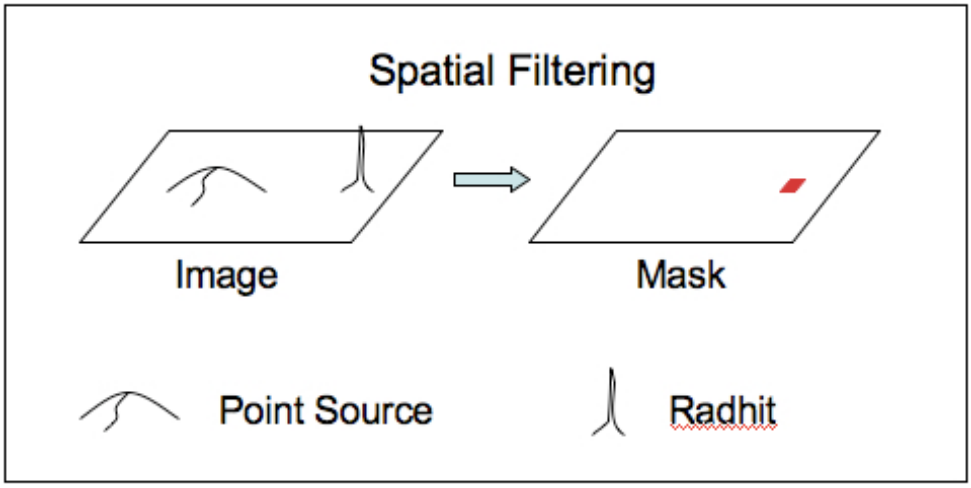


Figure 8.2: Cartoon of single-frame outlier detection.

It has three input parameters: *Segmentation Threshold*, *Detection Max Area, and Radhit Threshold*. It performs image segmentation based on a low threshold *Segmentation Threshold* and then applies two conditions to weed out point sources. The first condition is that the total area in pixels should be no greater than a user specified *Detection Max Area*. The second condition is that at least one pixel in the cluster should be greater than another user specified *Radhit Threshold*, which should be set much higher than *Segmentation Threshold*.

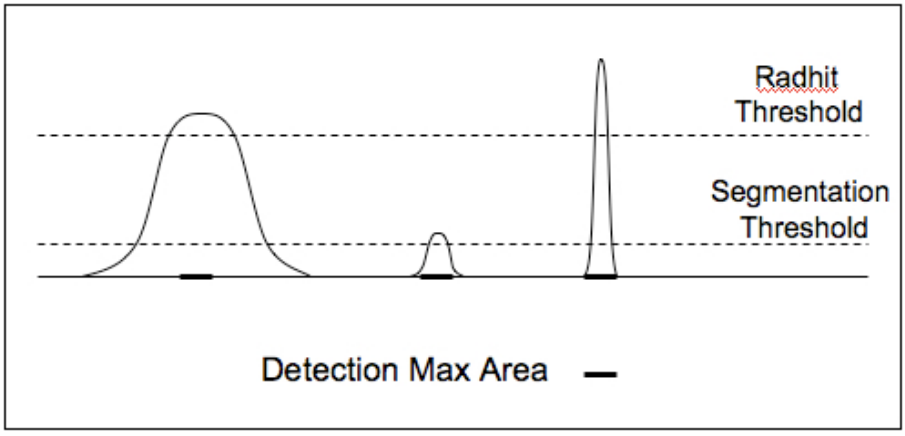


Figure 8.3: An illustration of how Detect Radhit selects a radhit over two point sources.

Figure 8.3 illustrates the above algorithm. The three objects depicted there are a bright point source, a faint point source, and a radhit. After segmentation with a low *Segmentation Threshold* is performed, the bright point source creates a cluster of pixels with the number of pixels greater than *Detection Max Area*. Then the second criterion of having at least one pixel greater than *Radhit Threshold* weeds out the faint point source.

8.2.2 MultiFrame Temporal Outlier Detection

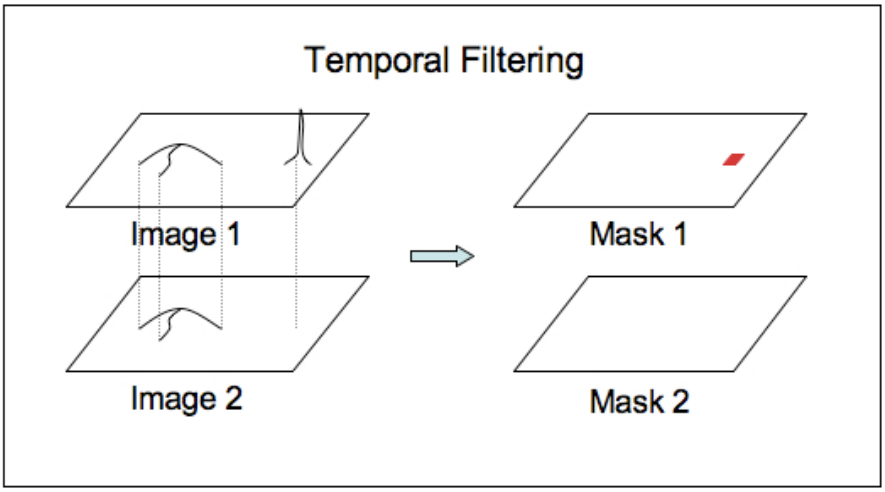


Figure 8.4: Cartoon of temporal outlier detection.

The second outlier detection method carries out temporal filtering of the interpolated images. It is performed by the Mosaic Outlier module. The input parameters used for this module are *Bottom Threshold, Top Threshold*, and *Min Pix Number*.

Important: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must check the *Use Outlier for RMask* box in the Mosaic RMask module and set the *RMask Fatal Mask Bit Pattern* in the Initial Setup module to use bit 1. If you are using the command line, include the *USE_OUTLIER_FOR_RMASK* trigger in the namelist and set *RMask_Fatal_BitPattern* to use bit 1. Note that this is not the same as setting it to a value of 1. See §8.11: Fatal Mask Bit Patterns for more information.

The input images are interpolated onto a common grid, such that the pixels in the interpolated images are perfectly lined up and create a stack of pixels. Each spatial location, i.e. a pixel in the common grid, is filtered in the time domain and outliers are found (see Figure 8.5).
If pixel value I_k for pixel k in the stack satisfies either of the following conditions:

$$I_k < M - \textit{Bottom Threshold} * \sigma$$
$$I_k > M + \textit{Top Threshold} * \sigma$$

pixel k is declared an outlier. M and σ are estimates of the mean and scatter of the pixel values in the stack. If pixel uncertainty images are provided σ above is set to the greater of two values: the scatter and the smallest pixel uncertainty in the stack. The scatter cannot be reliably estimated with only a handful of samples. If the stack size is less than *Min Pix Number*, then σ is set to the smallest pixel uncertainty in the stack.

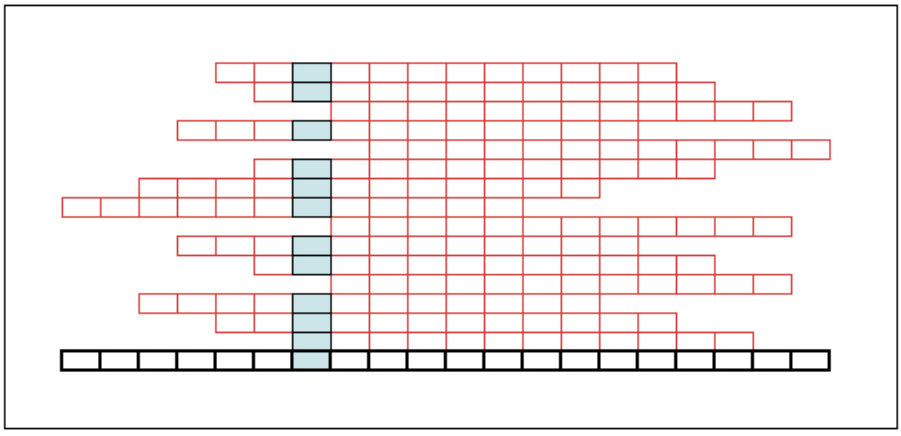


Figure 8.5: Interpolated images (red) are stacked up and shown here in the side view. The underlying grid is shown in black.

Two ways to estimate the mean M and scatter σ of the pixel values in the stack are implemented. It is controlled by the namelist variable *THRESH_OPTION*, but this variable is only available on the command line. The GUI uses the default value of *THRESH_OPTION = 1,* which is highly recommended for all users.

THRESH_OPTION = 1 is the more robust and highly recommended way of sigma estimation. Under this option,

$$M = \text{median}(I_k),\ \ \sigma = \text{MAD}(I_k) = \text{median}(|I_k - \text{median}(I_i)|)/0.6745$$

where MAD stands for the median absolute deviation. The factor of 0.6745 here represents the inverse of the third quartile of the normal distribution. The division is necessary because the numerator alone tends to underestimate the standard deviation, so dividing by this value makes the MAD more accurate.

THRESH_OPTION = 2 is less robust and based on the minimum difference between the values of two consecutive pixels in the stack. We strongly recommend that users do not use this option.
The product of this step is an outlier map, where only outlier pixels have a non-zero value. The pixel value O_k in an outlier map is the deviation of the pixel k in the input images from the mean in the stack in terms of the number of standard deviations σ:

$$O_k = \frac{(I_k - M)}{\sigma}$$

Equation 8.1

The thresholds - *Bottom Threshold* and *Top Threshold* - can be set to 0, which is the default. In this case the decision of declaring a particular pixel an outlier is put off until running the Mosaic RMask module. The advantage of doing it this way is that the user can experiment with different thresholds for outliers without having to rerun Mosaic Outlier.

The main limitation of temporal outlier rejection is the cases of shallow coverage.

8.2.3 Dual Outlier Detection

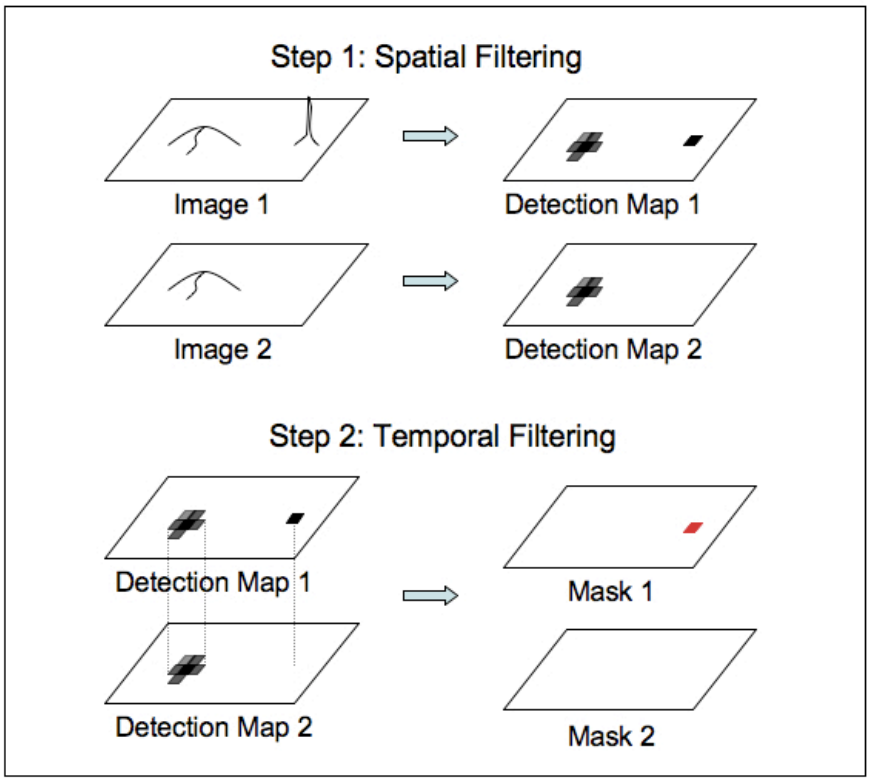


Figure 8.6: Cartoon of dual outlier detection.

The third method represents a more complicated dual spatial-temporal filtering.

Dual outlier detection consists of two-stage filtering. At the first stage, all spatial pixel outliers are detected and saved as detection maps. These detection maps include point sources and radhits. Detection maps are interpolated to a common grid. Then for each spatial location, the values of the interpolated pixels in the detection maps are compared. If in the majority of the detection maps this pixel location has not been detected by spatial filtering, then it is declared an outlier in the images where this pixel location has been detected. This method is expected to be reliable for a small number of images, i.e. in the shallow coverage case. Therefore it represents a suitable supplement of the multiframe temporal outlier detection. Below are the four steps of the dual outlier detection in more detail.

Important: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must check the *Use Dual Outlier for RMask* box in the Mosaic RMask module and set the *RMask Fatal Mask Bit Pattern* in the Initial Setup module to use bit 2. If you are using the command line, include the

USE_DUAL_OUTLIER_FOR_RMASK trigger in the namelist and set *RMask_Fatal_BitPattern* to use bit 2. Note that this is not the same as setting it to a value of 2. See §8.11: Fatal Mask Bit Patterns for more information.

Spatial detection

The Detect module, applied to background subtracted images, finds all the pixels with values greater than a specified number of standard deviations. The input parameters used here are *Detection Max Area*, *Detection Min Area*, and *Detection Threshold*. The product of this step is called a detection map. Each contiguous cluster is assigned a number and the values of all the pixels in the cluster are set to this number.

Detection map interpolation

The map interpolation is carried out by the Mosaic Projection module. The process is different from image interpolation. The non-zero input pixels set the values of the output pixels they overlap without any weighting. The process is ambiguous if two clusters are close, and there are output pixels that overlap input pixels from both clusters. The products of this step are the interpolated detection maps (see Figure 8.7).

		1	1							3	3		
	1	1	1	1									
	1	1	1	1					4	4			
		1	1						4				
											5		
		2	2	2						5	5	5	
	2	2	2	2						5	5	5	
		2	2								5	5	
		2	2	2									

Figure 8.7: A sample interpolated detection map. Clusters 1, 2 and 5 are point sources, clusters 3 and 4 are radhits. Other pixels are set either to 0 if they overlap non-outlier pixels in the input detection map, or to NaN (not a number) if they overlap no pixels in the input detection map.

Multiframe Comparison of Detection Maps

The comparison is performed by the Mosaic Dual Outlier module. The input parameters that are used are *Max Outlier Image* and *Max Outlier Fraction*. Interpolated detection maps are stacked up. For each stack, the number of spatial outlier pixels is found. If the fraction of outliers in the stack is smaller or equal to *Max Outlier Fraction* and the number of spatial outliers in the stack is smaller or equal to *Max Outlier Image*, then these pixels are now classified as dual (spatial and temporal) outliers, and the sign of the dual outlier pixels is changed. For example, in Figure 8.8, suppose *Max Outlier Fraction = 0.5* and *Max Outlier Image = 3*. Stack A is the perfect case when only one pixel in the stack is a spatial outlier (value of -12). Its sign is changed from positive to negative. Stack B has 4 pixels that are spatial outliers (1, 12, 5, 8). The fraction of spatial outliers is 0.4 < *Max Outlier Fraction*, but 4 > *Max Outlier Image*, so since the stack does not satisfy both conditions, their signs are unchanged and remain positive. In Stack C the fraction of spatial outlier is 0.6 > *Max Outlier Fraction*, so the signs remain unchanged. Stack D is the perfect case of a pixel in a point source detected in all overlapping images. The fraction of spatial outliers is 1. The products of this step are dual detection maps (see Figure 8.9).

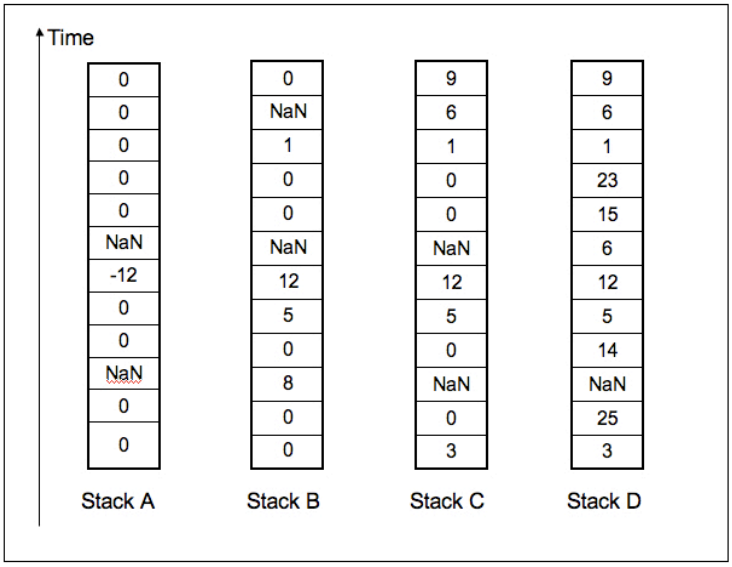


Figure 8.8: Suppose *Max Outlier Fraction = 0.5* and *Max Outlier Image = 3*. Stack A is the perfect case when only one pixel in the stack is a spatial outlier (value of -12). Its sign is changed from positive to negative. Stack B has 4 pixels that are spatial outliers (1, 12, 5, 8). The fraction of spatial outliers is 0.4 < *Max Outlier Fraction*, but 4 > *Max Outlier Image*, so since the stack does not satisfy both conditions, their signs are unchanged and remain positive. In Stack C the fraction of spatial outlier is 0.6 > *Max Outlier Fraction*, so the signs remain unchanged. Stack D is the perfect case of a pixel in a point source detected in all overlapping images. The fraction of spatial outliers is 1.

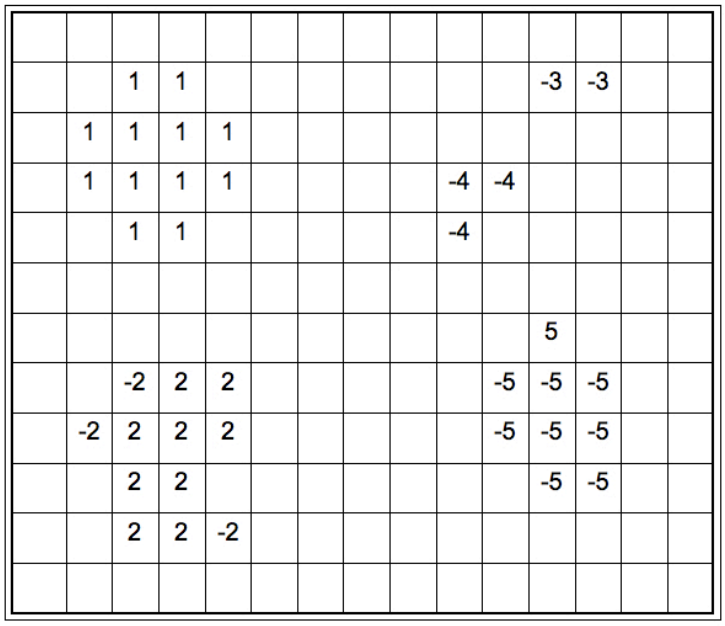


Figure 8.9: As a result of temporal detection, the sign of the pixels of clusters 3 and 4 has been changed. Also some of the pixels of cluster 2 have their sign changed. Cluster 5 is probably a moving source - the spatial detection suggests that it is real, but the temporal detection shows that it is no longer at the same coordinates in subsequent frames. One of the pixels in cluster 5 has its sign unchanged. This could happen if an overlapping frame has a radhit affecting the corresponding pixel.

Dual Detection Map Correction

The processing is done by the Level module. The dual detection map is processed in order to eliminate detection of the outskirts of legitimate point sources as dual outliers. If a dual outlier belongs to a cluster where the majority of pixels are not dual outliers the chances are the pixel has been wrongly marked because it is on the edge of the point source.

The sign of wrongly marked pixels is flipped based on the namelist parameter *Threshold Ratio*. If the number of negative pixels N- in a cluster with N pixels is smaller than the threshold

$$\frac{N-}{N} < Threshold\_Ratio$$

Equation 8.2

then their signs are flipped. If the number of positive pixels N+ in a cluster with N pixels is smaller than the threshold

$$\frac{N+}{N} < Threshold\_Ratio$$

Equation 8.3

then their signs are flipped. If *Threshold Ratio = 0.5*, which is the default, the pixels within each cluster will have the same sign.

The products of this step are the corrected dual detection maps (see Figure 8.10).

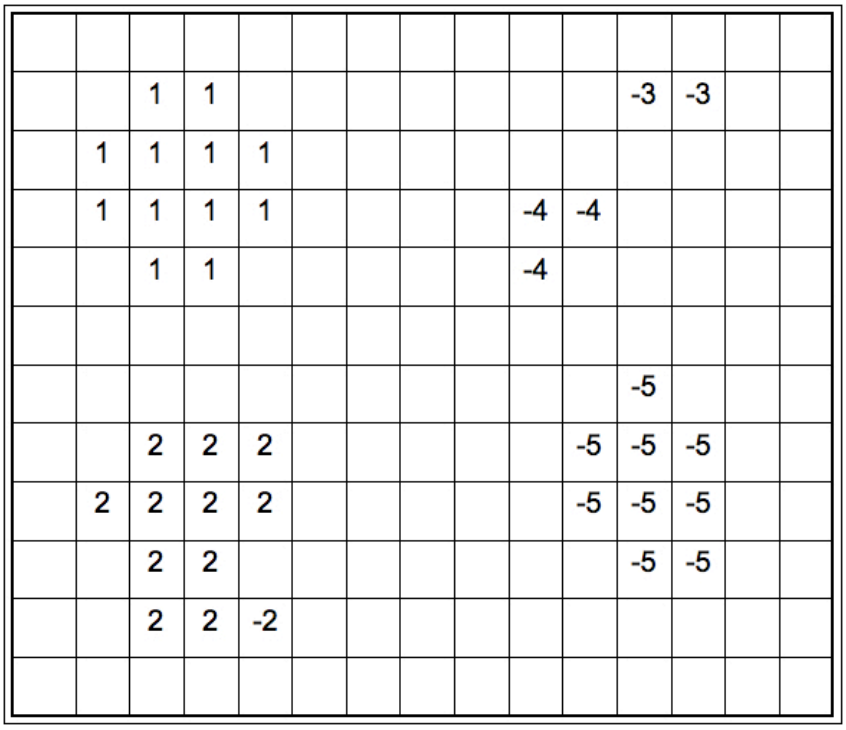


Figure 8.10: Suppose *Threshold Ratio = 0.5*. In Figure 4, cluster 2 has outlier fraction of 0.25, which is less than *Threshold Ratio*. The corrected dual map has all the pixels in cluster 2 set positive. Cluster 5 in Figure 4 has an outlier fraction of 0.9, which is greater than *Threshold Ratio*. The single positive pixel in cluster 5 has been set negative.

8.2.4 Box Outlier Detection

The fourth outlier detection method is designed to use both the temporal and spatial information like the dual outlier, but it uses the statistical analysis of the kind used by the temporal outlier. It is implemented by the Mosaic Box Outlier module.

Important: Using this module does not mean that MOPEX will automatically use the results for outlier detection. In order to use the results from this module, you must check the *Use Box Outlier for RMask* box in the Mosaic RMask module and set the *RMask Fatal Mask Bit Pattern* in the Initial Setup module to use bit 3. If you are using the command line, include the *USE_BOX_OUTLIER_FOR_RMASK* trigger in the namelist and set *RMask_Fatal_BitPattern* to use bit 3. Note that this is not the same as setting it to a value of 3. See §8.11: Fatal Mask Bit Patterns for more information.

For each mosaic pixel position, Mosaic Box Outlier creates two stacks of pixel values. The first stack is the same as the one created for the regular outlier detection method. It includes pixels from each interpolated image corresponding to the mosaic pixel position. This pixel in each interpolated image will be referred to as the main pixel. In each interpolated image a box of the user defined size is created centered on the main pixel. Input parameters *Box X* and *Box Y* define the size of the box. The second stack -- the box stack -- includes pixels from the above box in each interpolated image. The values of M and σ are computer in the box stack:

$$M = \text{biased\_median}(I_k)$$
$$\sigma = \text{MAD}(I_k) = \text{biased\_median}(|I_k - \text{biased\_median}(I_i)|) / 0.6745$$

The *biased_median* is defined as follows. If there are N pixels in the stack, then the biased median is equal to the $N/2$-*Bias* element of the stack:

$$biased\_median(I_k) = I[N/2 - Bias]$$

The input parameter *Box Median Bias* controls the value of *Bias*.

The product of this step is an outlier map. The pixel value O_k in an outlier map is the deviation of the pixel k in the input images from the mean M in the stack in terms of the number of standard deviations:

$$O_k = \frac{(I_k - M)}{\sigma}$$

Equation 8.4

If all the pixels in the main stack are outside of the M +/- σ range, the program falls back on the temporal outlier method 1 using the main stack and the uncertainty images, if the latter are available.

Inclusion of the neighboring pixels allows for outlier detection, even in the case of coverage of two, where the simple temporal detection is completely powerless. Clearly, a single pixel radhit on a relatively constant background can be easily identified as outlier, even with the smallest box size of 3. However, even radhit in the shape of wide and long streaks can be successfully detected by this method owing to the *Box Median Bias*. Suppose, the box size is 3, so the stack consists of 18 pixels, and the size of the radhit is at least 3x3 pixels. Out of the 18 pixels 9 background pixels have values relatively close to each other. The 9 radhit pixels will in general have greater scatter, then the 9 background pixels. Because of the bias, the median will be equal to the highest value of the background pixels. The difference between the highest and the lowest values of the background pixels are expected to be smaller, than the difference between the highest value of the background pixel and the lowest value of the radhit pixels. Consequently, the biased median deviation will be equal to the difference between the highest and lowest values of the background pixels.

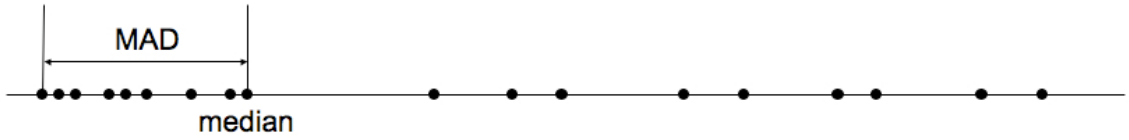


Figure 8.11: The 18 pixel values shown here are from two images and the box size of 3. The 9 lower pixel values are the background and the 9 higher pixel values are from a radhit.

The problem with temporal outlier rejection is that it sometimes rejects valid pixels in the center of bright sources. It is especially acute for the undersampled data. Interpolation of such data results in significant errors and some interpolated pixels have values higher than the uncertainties and the scatter in the stack. Including the neighboring pixels increases the estimate of the scatter and prevents identifying such pixels as outliers.

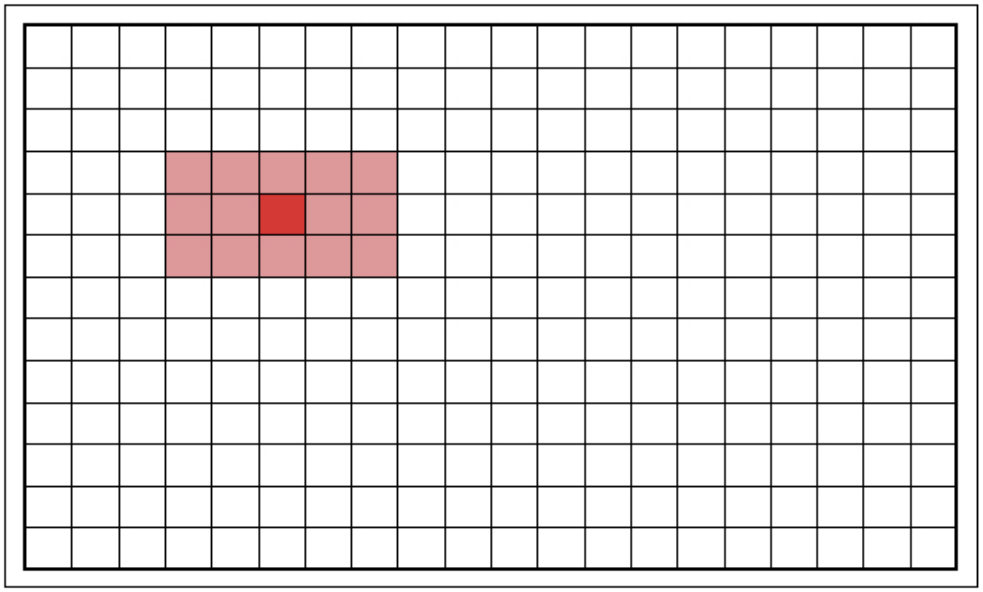


Figure 8.12: The pixels inside the red rectangle are included in the box stack. The red rectangle is centered around the solid red pixel, which is the mosaic pixel for which the estimation is performed. The solid red pixel is part of the main stack. In the figure *BOX X = 5* and *BOX Y = 3*.

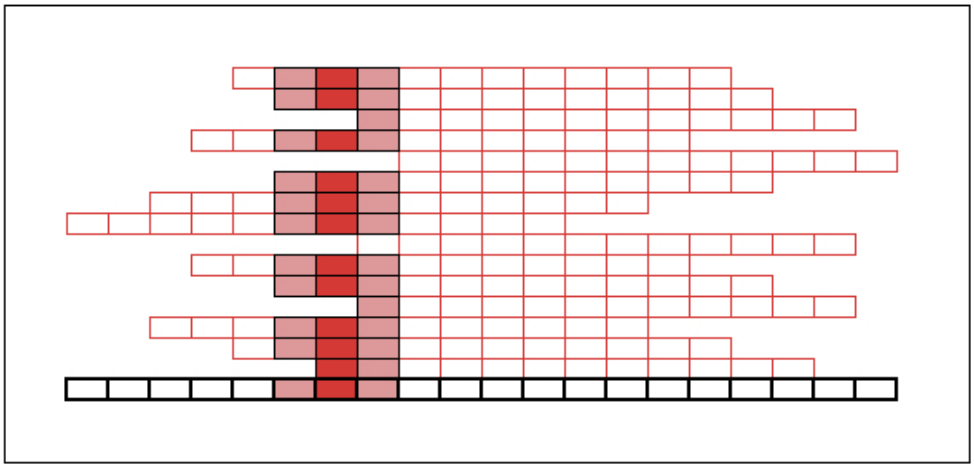


Figure 8.13: Interpolated images (red) are stacked up and shown here in the side view. The underlying FIF grid is shown in black. The solid red pixels comprise the main stack and the solid red and pink pixels comprise the box stack. In this figure *BOX X = 3* and *BOX Y* is not shown.

8.3 Image Segmentation

8.3.1 Overview

The Detect module performs image segmentation, and computes the centroids for the detected pixel clusters. These clusters may be real astronomical sources or they may be spurious. The module is called at the beginning of both the outlier rejection and point source extraction processes.

The program starts by computing the initial threshold based the user specified detection threshold (*Detection Threshold*). 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 (*Detection Max Area* and *Detection Min Area*). If the number of pixels in a particular cluster is less than the minimum number, the cluster is discarded. If the number of pixels in a cluster is greater than the maximum number, 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.

To this end, the program recalculates the threshold for this particular cluster and finds all the pixels above the new threshold. See Figure 8.14 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.

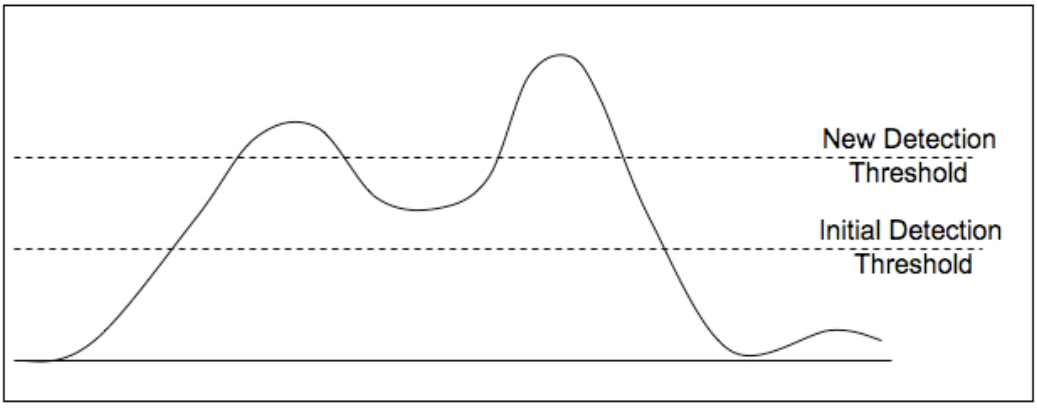


Figure 8.14: Raising the detection threshold splits a big cluster into two smaller clusters.

8.3.2 Thresholding

The calculation of the initial value of the threshold depends on the type of image being processed, which is specified by the user by setting input parameter *Input Type*. It has two settings: *Image Input* (default) and *SNR Input*. The first setting corresponds to a regular data image, the second one is used for the signal-to-noise ratio images. For *Image Input*, the initial threshold *T* is computed as follows:

$$T = M + Ds$$

Equation 8.5

where *D* is the detection threshold (number of sigma), *M* is the mean signal in the image, and *s* is the standard deviation of the image. The process is repeated iteratively, with *M* and *s* recomputed excluding pixels greater than *T*, until all pixels used are below *T*. For *SNR Input*, the intital threshold is simply set equal to the *D* parameter.

Initial image segmentation is performed. The clusters exceeding *Detection Max Area* size are subject to further segmentation. At this point the threshold is recalculated, so that the new higher threshold will either shrink the "oversized" clusters or break them into several smaller ones. This new threshold is calculated individually for each cluster.

***Threshold Type*:** The way the image segmentation threshold is recalculated is determined by the input parameter *Threshold Type*, which has three settings: *Simple*, *Combo*, and *Peak*. The threshold type does not depend on the type of input image.

- ***Threshold Type = Simple***. The mean value M_{cl} and standard deviation s_{cl} of all pixels in the cluster are calculated. The cluster specific threshold is

$$T_{cl} = M_{cl} + Ds_{cl}$$

Equation 8.6

  The new threshold is applied to the cluster. If the cluster is shrunk or split, the threshold is recalculated again for each new cluster. If the number of pixels after applying a new threshold doesn't change, the cluster is passed down for centroid computation, even though the number of pixels in it is greater than *Detection Max Area.*

  There are two problems with this simple-minded approach. First, once T_{cl} fails to reduce the cluster, this is the end of the segmentation process. Second, this approach will very often fail to resolve two or more point sources that end up in one cluster after the initial thresholding. If one of the point sources is significantly brighter than the others it will

drive T_{cl} to be higher than the pixels in the fainter point sources. The two schemes below were designed to alleviate these problems.

- *Threshold Type = Combo.* The following heuristic formula is used

$$T_{cl} = \sqrt{((SegLevel \cdot T_{\min} + Min_{cl}) \cdot M_{cl}}$$

Equation 8.7

Here T_{min} is the difference between the initial threshold T and the minimum of the whole image. *SegLevel* is the number of times the threshold for this cluster has been raised without any effect on the cluster. Here is how it works: Initially, for each cluster it is set to 1. Then a new threshold is calculated. If all the pixels in the cluster are higher than the threshold, then the threshold is raised and *SegLevel* is incremented. This is repeated until some of the pixels in the cluster end up below the threshold. At this point *SegLevel* is reset to 1. This approach works better then the first one, since the threshold is raised slower and segmentation doesn't stop after the first failure.

- *Threshold Type = Peak*. This is the most convoluted and is also the most effective one. It is the scheme that has been found to do the best for point source extraction. The condition, that a cluster is split when it is greater than *Detection Max Area*, is supplemented with one more condition:

  > A cluster is split as long as there is more then one "peak pixel" per cluster or it is greater than *Detection Max Area*.

  A peak pixel is any pixel greater than any other pixel within a certain radius. In command-line MOPEX, the namelist parameter *Peaks_Radius* is used to specify this radius. The GUI uses the default radius of 1, i.e. a pixel greater than the 8 adjacent pixels is declared a peak. If *Peaks_Image_Filename* is set in the namelist, an image of peak pixels will be written out. For each cluster the value P_{min} of the lowest peak is found. Segmentation threshold T_{cl} is first is set to

$$T_{cl} = P_{\min} - T_{\min}$$

Equation 8.8

If the cluster doesn't change (split or shrink), then the threshold is slowly raised:

$$T_{cl} = P_{\min} - \frac{T_{\min}}{SegLevel}$$

Equation 8.9

This is repeated up to *Max_Segmentation_Level* times. The value of *Max_Segmentation_Level* is selectable in command-line MOPEX, but is left to the default (50) in the GUI. After that the centroid is found for the resulting cluster, even though it might have more then one peak in it. There is one provision to prevent the program from splitting wings off of a bright star. If a cluster fails to be split, normally the threshold will be raised, but if the number of pixels in the cluster is greater than *Extended Object Area*, then this cluster is left the way it is, even though it might have more than one peak in it.

8.3.3 Co-added Images

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 Input* images. The alternative is to use a coverage map (*CoverageMap_Filename*). The coverage map is used to attenuate co-added images, i.e. an input co-added image is multiplied by the sqrt(coverage map), if a coverage map is provided.

8.3.4 Probability Images

The parameter *Probability Threshold* in the APEX settings is used for the so-called PSP images produced by the Point Source Probability module. The PSP images are products 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 then 1, which will lead to having no detections.

8.3.5 Passive Deblending

The output of this program is used for point source extraction. Point source extraction performs passive deblending. The detected point sources, determined to be in a close proximity from one another so that their PRFs (see §8.7) overlap, are fitted simultaneously. The Detect module provides the classification of detections as candidates for passive deblending. If a cluster created by the initial thresholding is consequently split into several clusters, the latter are classified as a blend of clusters. There are two columns in the output table 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, these columns are set to zero. Figure 8.15 shows an example of image segmentation after the recursive thresholding is performed.

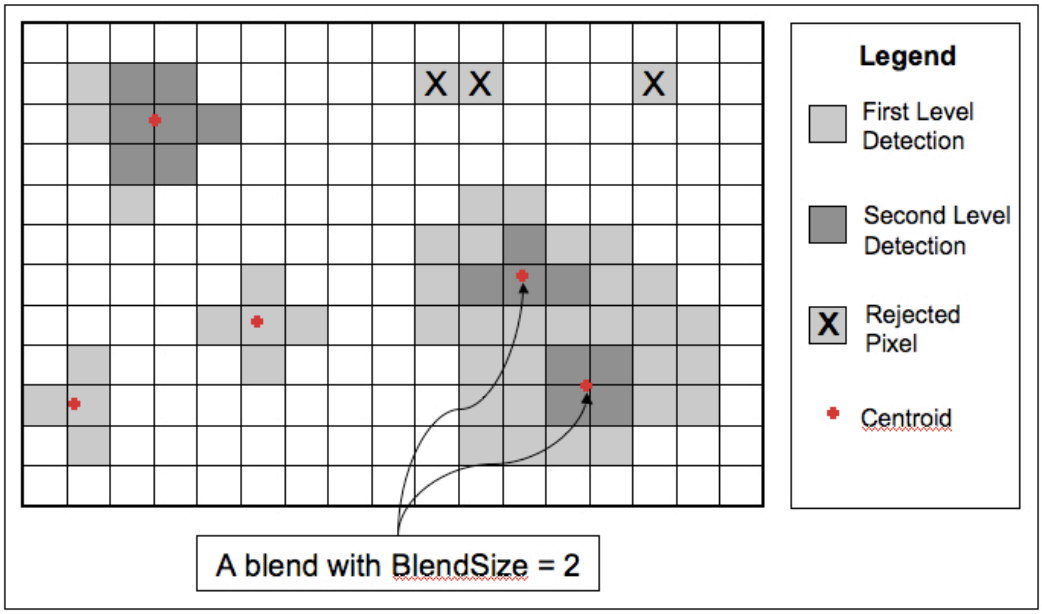


Figure 8.15: An example of image segmentation with *Detection Min Area* = *3*; *Detection Max Area* = *9*. A blend of detection with BlendSize = 2 is shown.

8.3.6 Centroid

For all detected clusters the centroid is found:

$$Centroid\_X = \frac{\sum_{i \in cluster} X(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)} \qquad Centroid\_Y = \frac{\sum_{i \in cluster} Y(i) \cdot flux(i)}{\sum_{i \in cluster} flux(i)}$$

Equation 8.10

The value of the greatest pixel in the cluster is saved as the "flux" in the output table. This quantity can be used only as a guide to the flux of a point source; the Detect module is not meant to compute photometry; the modules Source Estimate or Aperture Photometry should be used for that purpose.

8.4 Image Interpolation

8.4.1 Overview

The Mosaic Interpolate module (see e.g. §5.6.8) performs a projection of input images onto the Fiducial Image Frame (FIF; see e.g. §5.6.4) and an interpolation of the input pixel values to the output array of pixels of the user-defined pixel size. It corrects for the optical distortion in the input images. The process is intended to, but not restricted to, accept images measuring surface brightness and to yield images in the same units.

The FIF defines the sky position, orientation, and the size of the mosaic. The final mosaic pixel size is set in the Initial Setup module with the options *Mosaic Pixel Size X(Y)* or *Mosaic Pixel Ratio X(Y)*. On the command line, it is set with the namelist options *MOSAIC_PIXEL_SIZE_X(Y)* or *MOSAIC_PIXEL_RATIO_X(Y)*. See §5.6.1 for more information.

Four interpolation schemes are implemented: Default, Drizzle, Grid and Cubic. The first two schemes are based on area overlap, but they differ in the way the projection is done. The third scheme is a rough approximation of area overlap. The fourth scheme is the bicubic interpolation, which is a common technique in image processing.

Whatever interpolation scheme is used for the input images, the uncertainty images are interpolated in an identical fashion. A coverage map is created for each interpolated image. It reflects the presence of any input bad pixels or pixels on the edge of the input image.

In general, a projection onto the reference frame of an input image with optical distortions will not be a simple rectangle. Each interpolated image occupies a part of the FIF and is, in general, of

different size with different offsets from the origin of the FIF. The offsets of the interpolated images in x- and y- directions relative to the FIF, and their sizes (in integral numbers of pixels) are given in the file *interpolated_images.txt.tbl*, and also written in the headers of the interpolated images in the keywords `MINTOFFX` and `MINTOFFY`.

For maximum speed of operation, a special set of routines have been developed that allow direct plane-to-plane coordinate transformations that bypass computing the sky coordinates. For details, see the document "Fast Direct Plane-to-Plane Coordinate Transformations" (*Makovoz 2004, PASP, 116, 971*; http://www.journals.uchicago.edu/cgi-bin/resolve?PASP204097).

8.4.2 Default: Pixel Overlap Integration

The value O_j of an output interpolated pixel j is equal to the weighted average of the input pixels I_i overlapping the output pixel. It is weighted with the relative overlap areas as follows (Eq. 1):

$$O_j = \sum_i \frac{a_{ji}}{A_j} I_i, \qquad \textbf{where} \qquad A_j = \sum_i a_{ji}$$

Equation 8.11

Here a_{ji} is the area of the overlap of output pixel j with input pixel i.

For this scheme (see Figure 8.16), each output pixel is projected onto the input image frame and the flux from all the input pixels overlapping the output pixels is integrated and then normalized by the output pixel area to yield the result in the units of surface brightness.

If the *Fine Res* (*FINERES* `on the command line`) keyword is specified in the Mosaic Interpolate module, an additional bi-linear interpolation of the input pixels is performed before projecting onto the output grid. A grid of fineres pixels with the size defined as Input_Pixel_Size/*Fine Res* is introduced. In Figure 8.17 the input pixels are shown in thick black lines, the intermediate fineres pixels are shown in thin black lines. The value of a fineres pixel is a linear interpolation of the four closest input pixels, with the interpolation coefficients being the relative overlap area of the pixel, with the same size as the original input pixels, and centered around the fineres pixel. i.e. the value FP of shaded fineres pixel in Figure 8.17 is equal to:

$$FP = A_1 * IP_1 + A_2 * IP_2 + A_4 * IP_4 + A_5 * IP_5$$

Equation 8.12

where IP_i is the value of input pixel i, and A_i is the overlap area of pixel i with the pixel (blue fat lines) centered around the shaded pixel, with the same size as the input pixels. The output pixels are then computed using the pixel overlap interpolation scheme with the fineres pixels used as the input pixels.

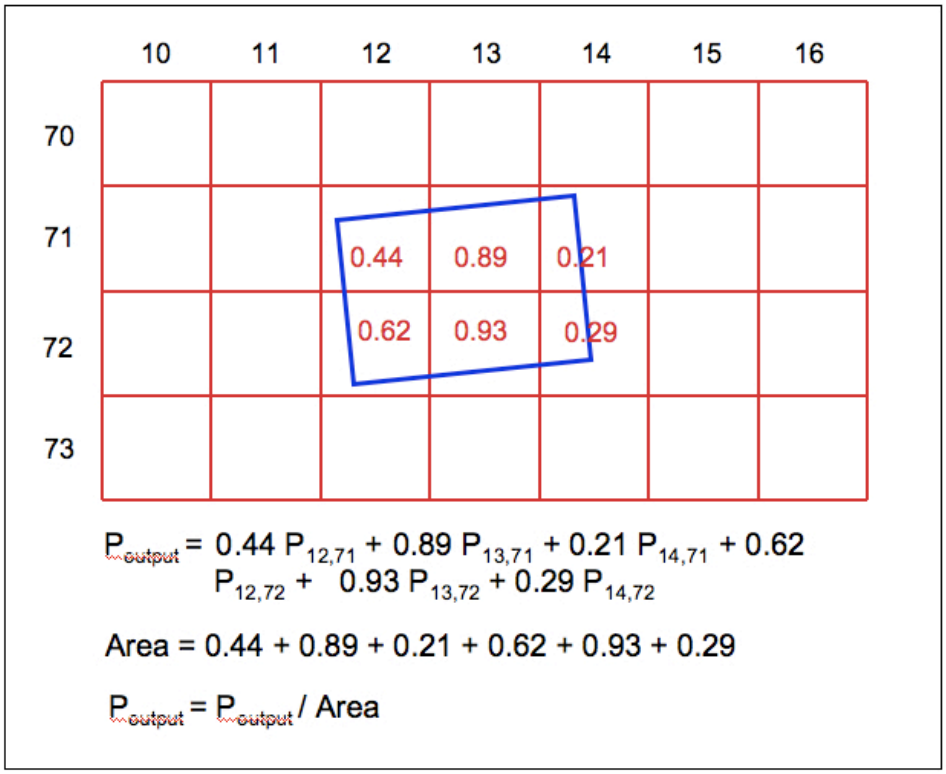


Figure 8.16: This figure shows the calculation of the weighted sum of the input image pixels (red, regular grid) overlapping the interpolated output image pixel (blue, thick line).

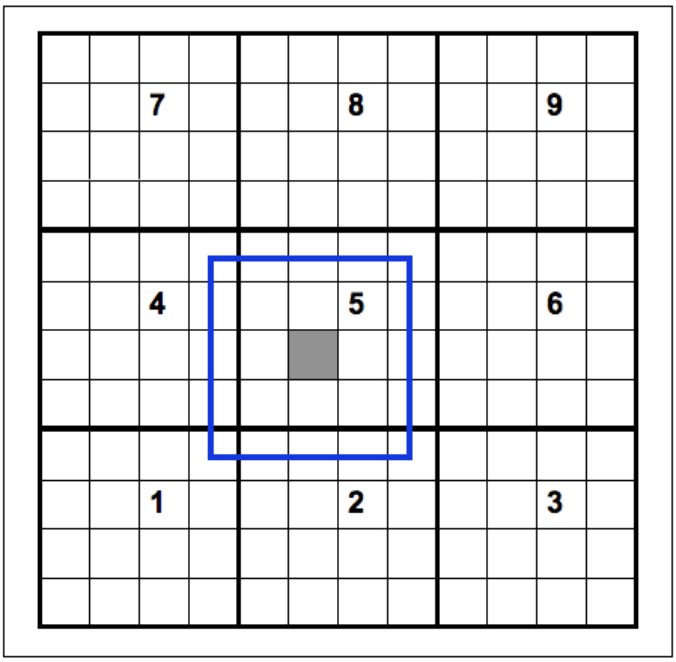


Figure 8.17: *Fine Res = 4*. The coordinates of the shaded pixel are j = 2, k = 2.

8.4.3 Drizzle: Pixel Overlap Distribution

The output pixels values are computed using Eq.1, just like in the previous scheme. The difference is that for this scheme (see Figure 8.18) the input pixels are projected onto the output frame. The flux of an input pixel gets distributed among the output pixels it overlaps.

Each input pixel is shrunk by the factor *Drizzle Factor* (*DRIZ_FAC* in command-line namelists). The values of the shrunken pixels are equal to the values of the original pixels. The shrunken pixels are projected onto the output image frame. This would be extremely hard to implement using scheme one, since then the overlap with the non-contiguous array of pixels would have to be computed (see Figure 8.19). On the other hand scheme one has been shown to be faster by ~25% than scheme two in the case of no drizzle, *Drizzle Factor = 1*.

Note: When using the Drizzle option, several of the mosaicking steps are run twice. Firstly, a linear interpolation is carried out so MOPEX can run the outlier rejection scheme. Once the outlier rejection masks (RMasks) have been created, MOPEX returns to the Mosaic Interpolate module and re-runs the interpolation with the Drizzle algorithm, masking out any pixels flagged in the RMasks. **When using this option, do not include the Mosaic Reinterpolate module in the processing flow.**

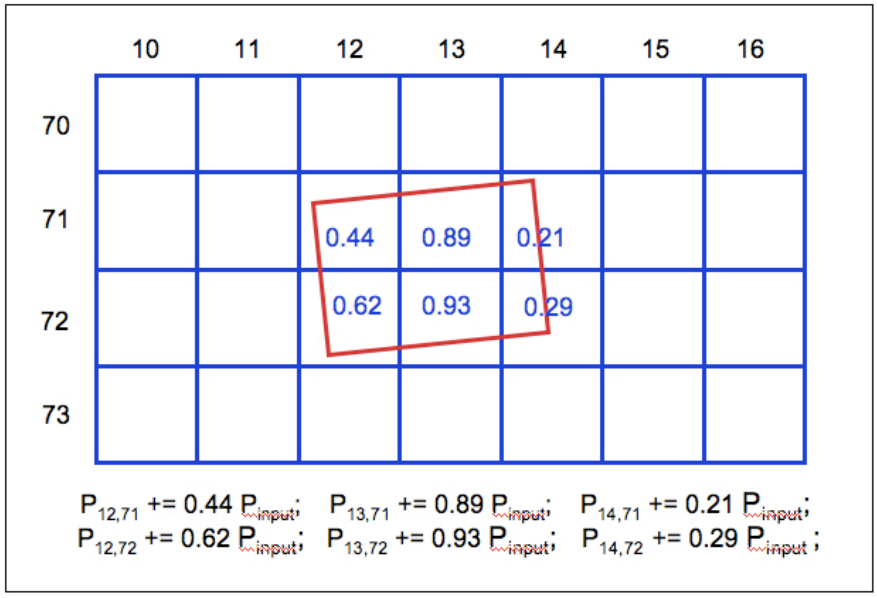


Figure 8.18: This figure shows the calculation of the weighted contributions of the input image pixel (red, thick line) to the overlapping interpolated output image pixels (blue, regular grid). The area of each output pixel in the output coordinate system is constant.

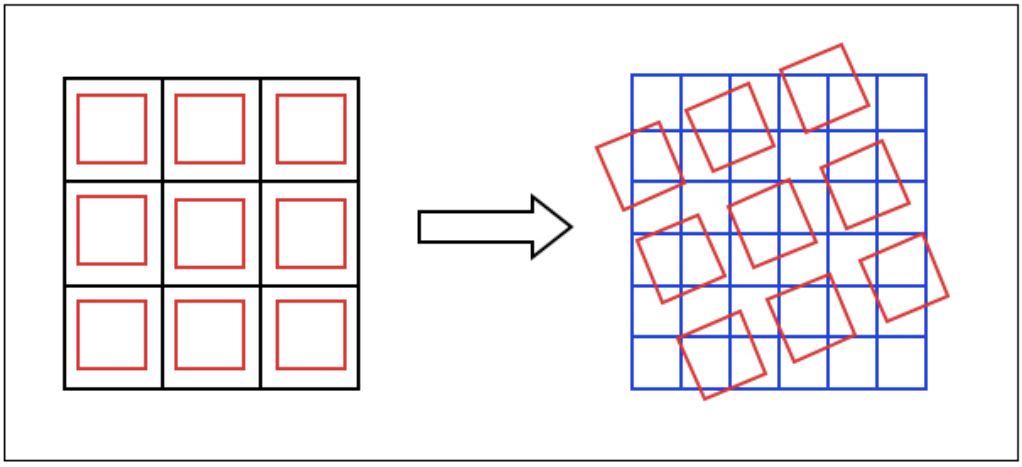


Figure 8.19: Input pixels are shown in black. The drizzle pixels, shown in red, are projected onto the output array of pixels shown in blue.

8.4.4 Grid Distribution

As a fast alternative of the area overlap method the grid interpolation is implemented. Each input pixel is filled with *Grid Ratio* (*GRID_RATIO*) squared grid points (see Figure 8.20). Each grid point is assigned the value of the pixel it belongs to. Each grid point is projected onto the output frame and the flux associated with the grid point is added to the output image pixel into which the grid point was projected. The value O_j of an output interpolated pixel j is equal to the weighted average of the input pixels I_i

$$O_j = \sum_i \frac{n_{ji}}{N_j} I_i, \qquad \textbf{where} \qquad N_j = \sum_j n_{ji}$$

Equation 8.13

where n_{ji} is the number of the grid points projected into the output pixel j from the input pixel i. The parameter *Grid Ratio* specifies the number of grid points in one direction in an input pixel. In other words there are *Grid Ratio* squared grid points per input pixel. If *Grid Ratio* = *1* and the input to output pixel size ratio *PR* = *1*, the method is equivalent to the nearest-neighbor interpolation. In the limit of infinite *Grid Ratio* the method approaches the area overlap interpolation. The main purpose of this scheme is to create first-look mosaic images fast, the gain in speed being up to 10 times as fast as scheme one. **The price you pay is the fidelity of the interpolated images. Do not use these images for science.**

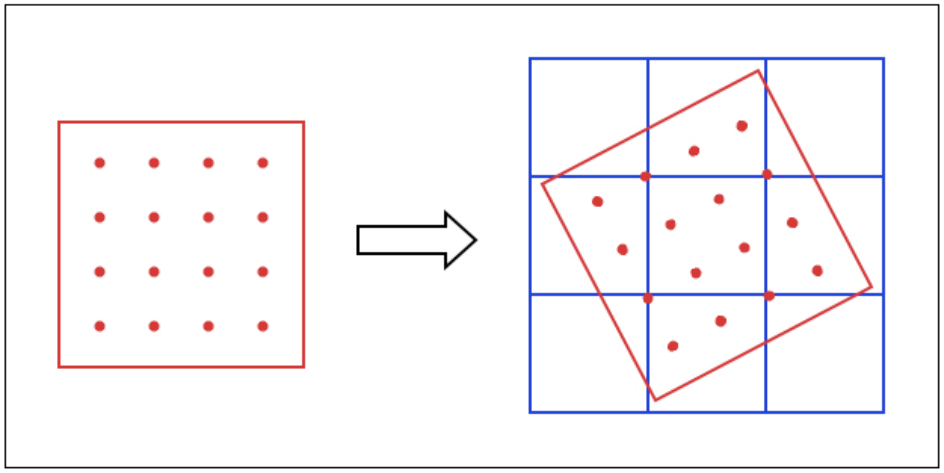


Figure 8.20: Input pixel (red) is filled with a grid of points. When the input pixel is projected onto the output array (blue) each point contributes an equal share of the input pixel value to the output pixel it is in.

8.4.5 Cubic: Bicubic Interpolation

For the bicubic interpolation the value *O(j)* of the output pixel *j* is computed as a linear combination of the input pixel values

$$O(j) = \sum_{i=1}^{16} C(j,i) I(i)$$

Equation 8.14

at the 16 positions *i* closest to *j*. The coefficients *C(j,i)* depend on the distance between the points:

$$C(j,i) = h(x_j - x_i) h(y_j - y_i);$$

$$h(x) = \begin{cases} a_{30}|x|^3 + a_{20}|x|^2 + a_{10}|x| + a_{00}, & 0 \le |x| < 1; \\ a_{31}|x|^3 + a_{21}|x|^2 + a_{11}|x| + a_{01}, & 1 \le |x| < 2; \\ 0, & |x| \ge 2; \end{cases}$$

Equation 8.15

The coefficients a_{kl} are derived by imposing the following constraints *h(0) = 1; h(1)=h(2) = 0; h'(1)* and *h'(2)* are continuous. These constraints yield seven equations for eight unknown coefficients. By allowing $a_{31} = \alpha$ the tunable parameter the system can be solved and the results are shown in the Table below. The parameter α is set in the interpolation module, with the default value equal to -0.5.

i	**a_{31}**	**a_{21}**	**a_{11}**	**a_{01}**
0	$\alpha+2$	$-(\alpha+3)$	0	1
1	α	-5α	-8α	-4α

8.5 Background Matching Algorithm

Background matching of the overlapping input images is performed by the Overlap pipeline. (*overlap.pl*). It calculates and applies an additive correction to each image in the input stack in

order to bring them to a common background level. This is often the first step in processing Spitzer imaging data, before mosaicking the frames together.

The images interpolated to a common grid can be subtracted pixel-by-pixel in order to match their backgrounds. We assume that the only correction required is a constant offset, ε_n, i.e. that the following corrections should be applied to the input image I_n

$$O_n(x,y) = I_n(x,y) - \varepsilon_n$$

Equation 8.16

Here O_n is the output corrected image. The metric to be minimized is the combined uncertainty weighted difference between the overlapping parts of each pair of input BCD's:

$$\mathbf{L} = \sum_{m,n=1(m\neq n)}^{N_{images}} \sum_{k \in overlap} \frac{(I_n(k_n) - I_m(k_m))^2}{\sigma_n^2(k_n) + \sigma_m^2(k_m)}$$

Equation 8.17

Here m and n are image indicies; k_n is the pixel number in image n. Minimization with respect to ε_n's

$$\frac{\partial \mathbf{L}}{\partial \varepsilon_n} = 0$$

Equation 8.18

leads to the following set of N_{images}-*1* linear equations:

$$\sum_{m=1}^{N_{images}-1} M_{nm}\varepsilon_m = z_n,$$

Equation 8.19

where

$$M_{nm} = -\frac{O_{nm}}{\sigma_n^2 + \sigma_m^2}, n \neq m; \qquad M_{nm} = \sum_{r(r\neq n)} \frac{O_{nr}}{\sigma_n^2 + \sigma_r^2};$$

$$z_n = \sum_{r(r \neq n)} \frac{O_{nr}(I_n - I_r)}{\sigma_n^2 + \sigma_r^2}.$$

Equation 8.20

The symbol O_{mn} respresents the fact that the summation is done over the overlap area of the *m*-th and *n*-th images. In Figure Figure 8.21 the case of 2 overlapping images is shown. The matrix element M_{12} is in this case equal to

$$M_{12} = -\frac{O_{12}}{\sigma_1^2 + \sigma_2^2} = -\left(\frac{1}{\sigma_1^2(36) + \sigma_2^2(0)} + \frac{1}{\sigma_1^2(37) + \sigma_2^2(1)} + \frac{1}{\sigma_1^2(38) + \sigma_2^2(2)} + \frac{1}{\sigma_1^2(39) + \sigma_2^2(3)} \right).$$

Equation 8.21

The actual ranges of indices symbolized by O_{mn} are calculated using the input table with the offsets and sizes of the interpolated images.

Since the problem is invariant under any arbitrary global shift δ of all images, one of the shifts can be picked to be the last one $\varepsilon_{Nimages-1}$. It is set to 0 at first. The additional constraint that will fix the global shift is to have the total shift of all images add up to 0:

$$\sum_{m=1}^{N_{images}} (\varepsilon_m - \delta) = 0,$$

$$\delta = \frac{1}{N_{images}} \sum_{m=1}^{N_{images}-1} \varepsilon_m;$$

$$\varepsilon_m = \varepsilon_m - \delta \text{ ; for } m=1, N_{images}.$$

Equation 8.22

The ε's are analyzed before applying the above condition. The outliers among the ε's are found. The following parameters are used:

- *Bottom Threshold*, *Top Threshold*: the thresholds for outlier detection in terms of sigma
- *Minimum Number of Images*: the minimum number of images required to detect outliers.

If N_{images} is greater than or equal to *Minimum Number of Images*, then the outlier ε's are excluded from calculating δ, but δ is applied to all the ε's.

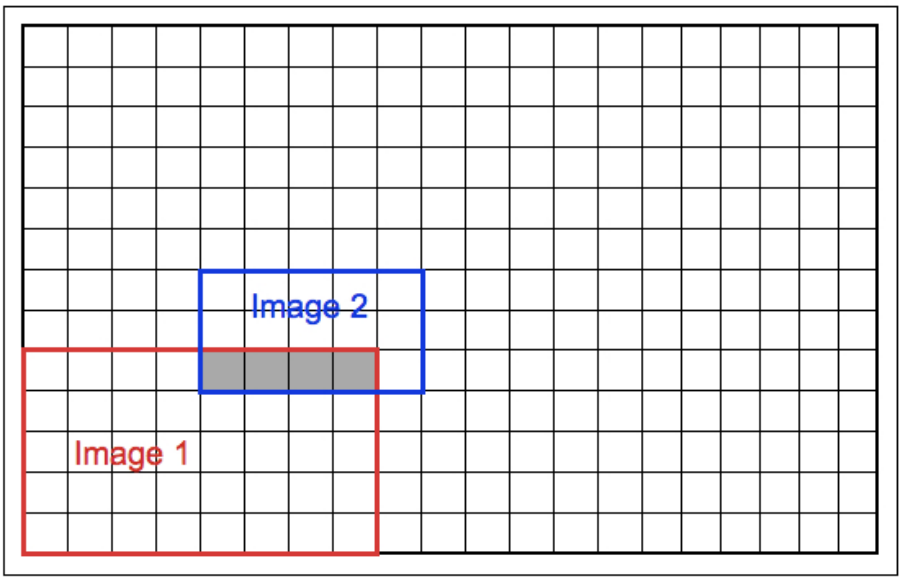


Figure 8.21: The overlap of Image1 and Image2 is an area of four pixels with the following pixel indices: k_1= {36,37,38,39}; k_2 = {0,1,2,3}, given that the first pixel has index = 0 and the x-direction is scanned first.

8.6 PRF fitting in MOPEX

Final point source position and photometry estimation are performed for all point source candidates on the detection list by module Source Estimate. For each point source candidate the data in the input images are fit with the Point Response Function (PRF; see §8.7). For the multi-frame point source extraction the fitting is performed simultaneously in all input images.

The size of the fitting area is determined by the parameters *Fitting Area X* and *Fitting Area Y*, if they are given. If they are not given, the Fit Radius module 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 Fit Radius module is not foolproof 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 *R(n)* (see Figure 8.22). If

the 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 $R_i(n)$ in the *i*-th input image are in general a complicated function of the global coordinates *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 8.23

Here *s* is the input image, possibly background subtracted, and σ is the input uncertainty image. Fitting is performed simultaneously in N_{images} images using N_{PS} point source candidates. First, χ^2 is minimized for $N_{PS} = 1$. The success of the fitting is determined by the parameter *Chi Threshold*. If

$$\chi^2 / dof < Chi\ Threshold$$

then 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 the namelist parameter *Max Number ps* is greater than 1, then active de-blending can be performed. The same data are fit with more than one point source. The number of sources is incremented until it reaches the limit set by the *Max Number ps* parameter or the success condition (above) is satisfied, whichever comes first. Since active de-blending is not a well-defined process, a more stringent condition needs to be met in order to accept extra point sources:

$$(\chi^2 / dof)_{new} < \frac{Chi\_Threshold + \chi^2 / dof}{2 \cdot Chi2\_Improvement}.$$

Equation 8.24

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 for a given cluster is greater than zero in the output of the Detect module, then passive de-blending is performed, and all the detections from one blend are fit simultaneously. That is, N_{PS} is set to BlendSize, even if BlendSize is greater than *Max Number ps*. The fitting area

in this case is more complicated than a simple rectangle, and is a combination of all the rectangles for each detection in the blend (see Figure 8.23). Passive de-blending has been proven to be an essential component of point source extraction.

In order to fit the data, a modified Simplex algorithm is employed (see §8.6.1 below). The Simplex algorithm does not use derivatives of the functions involved in minimization. This is a desirable feature since the transformation from local coordinates to global coordinates can be a very complicated function of its arguments.

Two parameters, *Dither Flux Fraction* and *Dither Pixel Fraction* determine the initialization of the Simplex algorithm. Each detection is split into three vertices. Their positions r_i and fluxes f_i are initialized by randomly dithering the position R and flux F of the detection:

$$r_i = R + rand * DitherPixelFraction$$
$$F_i = F * (1 + rand * DitherFluxFraction)$$

Equation 8.25

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

The unsuccessful termination of the fitting for each point source is determined by the two parameters *Minimize Ftol* and *Max N Iteration*; if successful, then the two parameters *Max N Success Iteration* and *Minimize Ftol Success* govern the termination. If the number of iterations exceeds *Max N Iteration* or the relative change in χ^2 becomes smaller than *Minimize Ftol* before χ^2 / dof reaches *Chi Threshold*, then 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 Iteration* or the relative change in $\chi 2$ becomes smaller than *Minimize Ftol Success*. After that, the point source is assigned the corresponding success status.

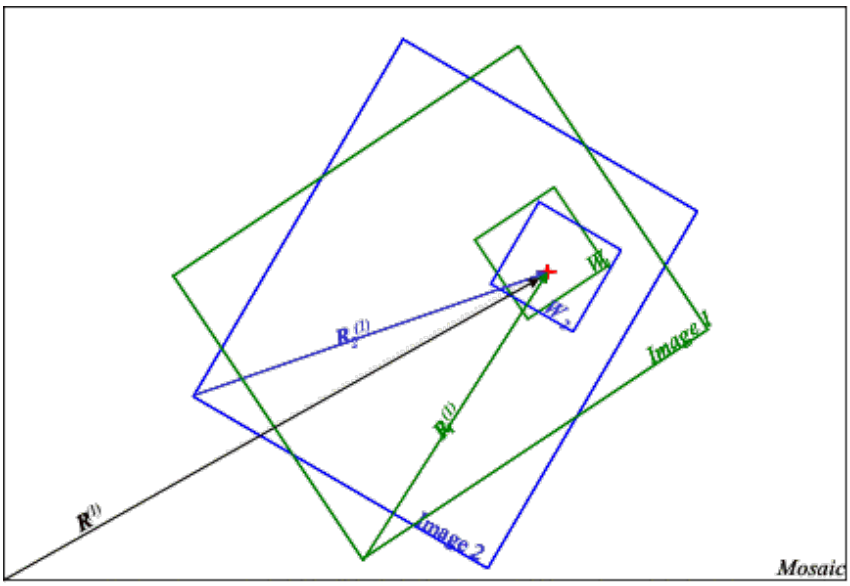


Figure 8.22: Point source 1(red) has global coordinates R(1) and local coordinates $R_1(1)$ and $R_2(1)$ in images 1 and 2, correspondingly.

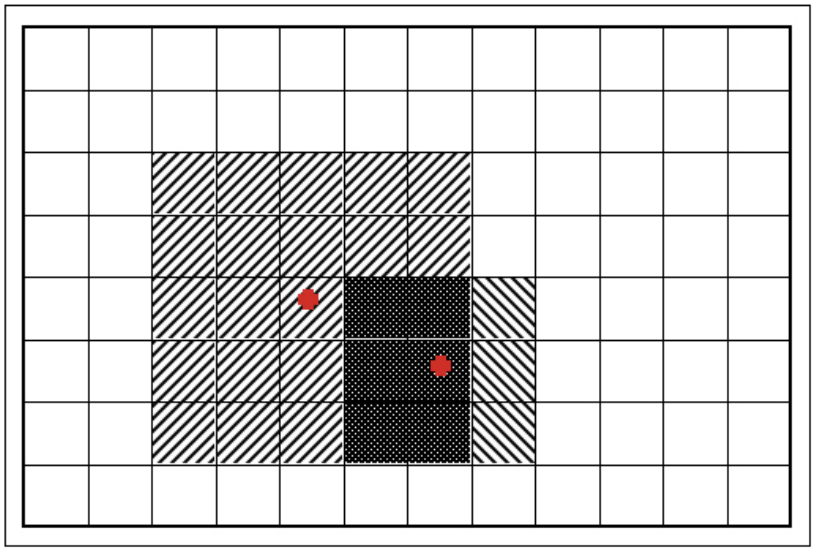


Figure 8.23: Total fitting area W_j for two detections, represented by the two red stars, consists of two fitting areas W_{j1} and W_{j2}, each one striped differently.

8.6.1 Simplex Algorithm Modification

In order to do the minimization, 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 8.24.

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.

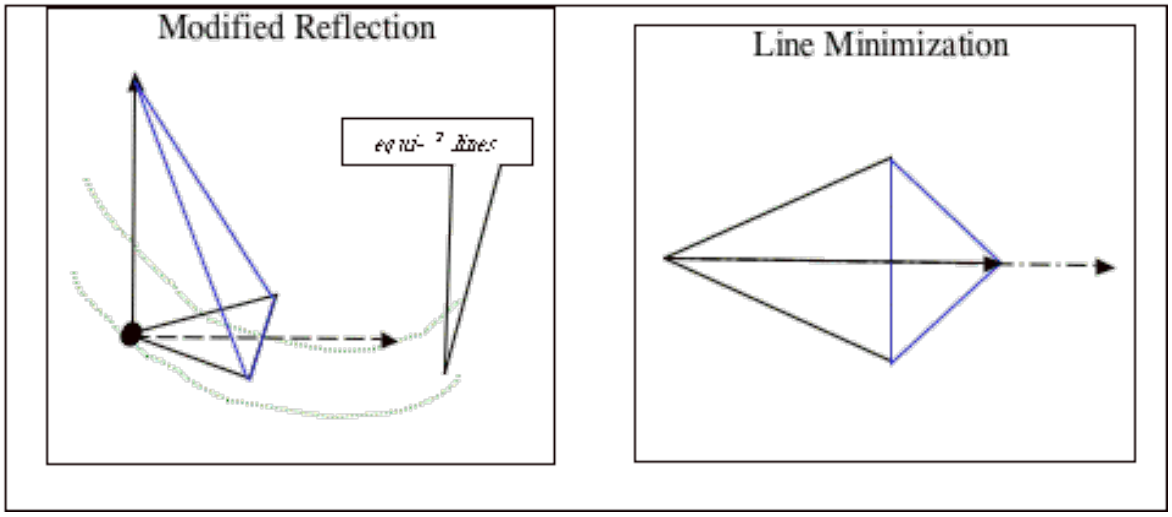


Figure 8.24: Illustration of the two modifications made to the original Simplex method.

8.7 Point Response Function

Point source detection and extraction in APEX rely upon a good measurement of the point response function (PRF). The term PRF is not just an alternative way of saying point-spread function (PSF). PSF-fitting is a commonly used term. However, PRF and PSF are two different objects. A PSF is an image of a point-source, and it is often oversampled; i.e., the pixel size of the PSF image is a fraction of the pixel size of the detector array or the mosaic image for which the the PSF is applicable. A PRF, however, is not an image of a point source. One way to think of a PRF image is as a kind of "table" of values of the responses of detector pixels to point source "hits" at different sub-pixel positions of a central pixel, stored in a convenient, interleaved form

that resembles an image. Normally, a PRF is oversampled, so that there are n x n samples stored, with n being the resampling factor. This means that the PRF holds pixel response images for hits in each n x n grid point in a pixel. You can find more information in the document entitled "How to make the PRF from the PSF", available at:
http://irsa.ipac.caltech.edu/data/SPITZER/docs/files/spitzer/PRF_howto.pdf

NOTE: For Spitzer data, standard PRFs may be used in APEX, or they may be measured from the mosaic with which they are to be used. Standard PRFs are available from the website. Users may produce their own PRFs using the PRF Estimate pipeline (*prf_estimate.pl*), which is part of the MOPEX distribution. **DO NOT** use PRF Estimate with and undersampled data, for example IRAC Channels 1 and 2. For IRAC PRF fitting you should use the provided PRFs.

IMPORTANT: APEX normalizes the PRF before fitting. If *Normalization Radius* is specified in the Source Estimate module, the normalization is performed over the pixels within that radius (IN PRF PIXEL UNITS) from the center of the PRF. If *Normalization Radius* is not specified, APEX will normalize over the available range that the central PRF covers in the PRF FITS file. Getting correct PRF-fitted absolute fluxes requires having the correct PRF normalization for your data!

Variable PRF

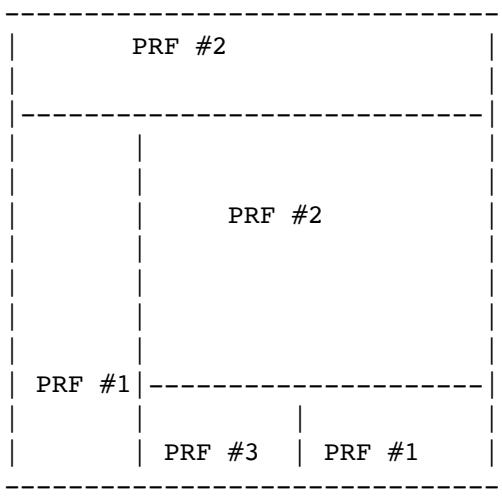
A provision is made to use a variable in the field of view PRF using the so-called PRF maps. It is assumed that several areas in the detector field of view will be identified, such that PRF variation within each area will be negligible and a constant PRF can be used for each area. A PRF map identifies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

A PRF map maps an image with respect to the PRF image used for any pixel of the image. In the PRF map file header, 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 set in 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:

```
\char comment = PRF Map
\int Number_PRF = 5
\int HaveSigma = 1
\int ImageX = 256
\int ImageY = 256
```

```
\int PRF_Filename_1 = /home/joe/Mopex/PRFs/IRAC.3.8um.PRF.12.fits
\int PRF_Filename_2 = /home/joe/Mopex/PRFs/IRAC.3.8um.PRF.12.fits
\int PRF_Filename_3 = /home/joe/Mopex/PRFs/IRAC.3.8um.PRF.12.fits
\int PRFSigma_Filename_1 = /home/joe/Mopex/PRFs/IRAC.3.8um.PRF.12.Sigma.fits
\int PRFSigma_Filename_2 = /home/joe/Mopex/PRFs/IRAC.3.8um.PRF.12.Sigma.fits
\int PRFSigma_Filename_3 = /home/joe/Mopex/PRFs/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:



8.8 Aperture Photometry

Circular aperture photometry can be done for each source. The photometry can be calculated on the original or background-subtracted image. If done on the original image, the background can be estimated in a circular annulus. The program sums the flux encircled by the aperture centered on the source using fractional pixel areas (see Figure 8.25). APEX Multiframe does aperture photometry on the mosaic, not the individual images. Aperture photometry *AP* is equal to

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

Equation 8.26

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

Optionally, if *Use Annulus* is selected, 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, as

B = *Operation* (Pixels with *R* >= *Inner Radius* and *R* <= *Outer_Radius*).

R is the distance from the point source position to the center of the pixel. Fractional pixel computation is not done; each pixel is either in or out (see Figure 8.26). *Operation* is defined by the *Annulus Compute Type* selection, and can be either "*mode*" or "*median*". In order to proceed with the background computation, there must be at least *Min Number Pixels* of good (non-NaN) pixels in the annulus. The background subtracted aperture photometry is computed as:

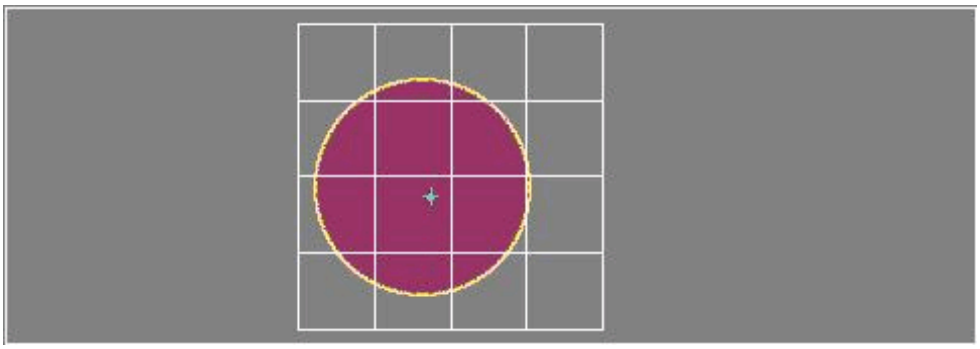


Figure 8.25: A circle of a specified radius centered on the point source (red) is overlaid on the image and aperture photometry is computed by summing the pixel values weighted by the overlap area.

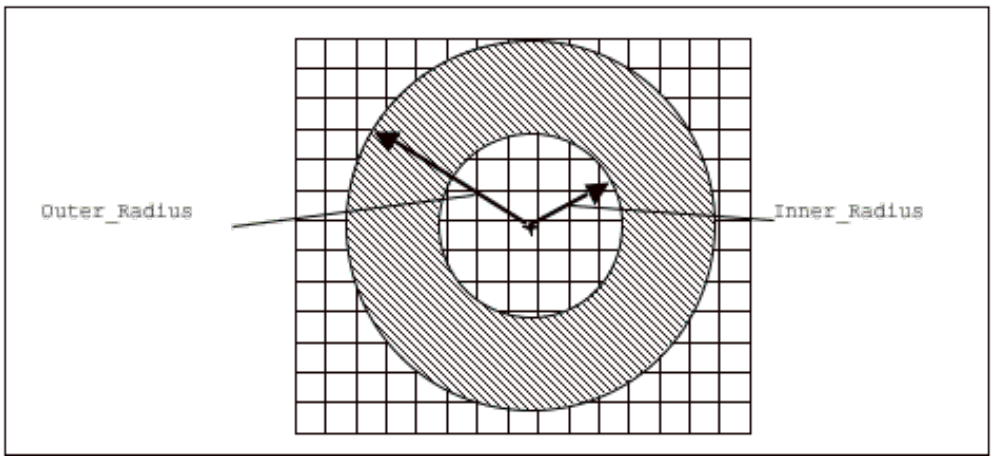


Figure 8.26: The geometry of the annulus used for background estimation.

8.9 Uncertainty Estimation

APEX computes two kinds of point-source flux uncertainties, as well as aperture uncertainties. The extract table output contains the following columns: delta_flux, SNR (signal-to-noise ratio), and ap_unc (aperture uncertainties).

8.9.1 delta_flux

delta_flux is the uncertainty in the PRF_fitting parameters, determined by the Hessian matrix:

$$H_{ab} = \frac{\partial^2 \chi^2}{\partial \lambda_a \partial \lambda_b}\left(\overline{\lambda_a}, \overline{\lambda_b}\right)$$

Equation 8.27

where $\overline{\lambda_a}$'s are the optimal fitting parameters: $\left(\mathbf{R}_n, f_n\right)$

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

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

Equation 8.28

In particular the positional (delta_x, delta_y) and flux (delta_flux) uncertainties computed by MOPEX are equal to delta_x = C_{xx}, delta_y = C_{yy}, delta_flux = C_{ff}.

Often, delta_flux will underestimate the true flux uncertainty because of correlated errors. The flux uncertainty is correlated with the positional uncertainty, and the formal uncertainties do not reflect this.

8.9.2 SNR Computation

In addition to the formal uncertainty in the PRF fit, APEX provides a rough estimate of the signal-to-noise ratio (SNR) of the source flux. This estimate is given in the output column "SNR". Three options are available, controlled by the *Use Photerr for SNR* parameter in the Source Estimate module. In addition, the user can swap in either APEX's background noise estimates (Gaussnoise), or the data uncertainties, with the APEX Settings parameter, *Use Data Uncertainties for PRF-fitted SNR* (this is *use_data_unc_for_fitted_snr* in namelists), where on (= 1) means use the data uncertainties. The SNR choices are given below. In namelists, use the number, e.g. `Use_Photerr_for_SNR = 2` is equivalent to "*yes, with gain*" in the GUI.

0. no: This is APEX's old SNR. Scale the peak pixel uncertainty by that pixel's contribution to the total central PRF.

1. yes, no gain: Use the sum (in quadrature) of the data uncertainties within a box roughly the size of the core of the PRF.

2. yes, with gain: Use the background noise estimate from **Gaussnoise** within a box roughly the size of the core of the PRF, and add the source photon noise calculated from the flux and the gain factor.

Each option is described below. The basic problem is that in the general case of a multi-source, multi-parameter fit, the true flux uncertainty limits lie in a multi-dimensional plane that is difficult to describe with a simple analytic expression. In addition, APEX was designed to handle undersampled data, and some simple expressions for flux uncertainty break down when the true center of the point source cannot be pinned down accurately. All of the following should be taken as rough estimates, and have no formal statistical significance. Uncertainties in background level or due to confusion are not included. Best estimates of true uncertainties come from simulations and/or multiple measurements.

0. No Photerr: This option is held over from previous versions of APEX. It has been shown to perform poorly for undersampled (IRAC channels 1 and 2) data. The noise is estimated by scaling the noise in the pixel nearest to the source by that pixel's contribution to the total "central" PRF. The central PRF is that corresponding to a center-of-pixel hit. The pixel noise should be from

Gaussnoise, which produces images called, e.g. *coadd_Tile_*_Image_noise.fits*. Defining the factor, *NP*, as the ratio of the total central PRF volume to that under the peak pixel, the SNR is the extracted flux F_{ps} for a point source divided by the scaled noise $\sigma(x_{ps},y_{ps})$, at the peak pixel (x_{ps},y_{ps}):

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

Equation 8.29

This calculation assumes only one PRF, the central one. It becomes poor when the pixels are large relative to the PRF width (e.g. IRAC channels 1 and 2 data), and generally gives an uncertainty estimate that is too low. It does not take into account the positional uncertainty. However, it is reasonable in background-limited, well-sampled cases.

1. Photerr, no gain: Simulations of Spitzer data have shown that a better estimate of the SNR is obtained by summing the data uncertainties within a box roughly the size of the core of the PRF. The box extends out to about 10% of the peak and was estimated from the simulations; no sub-pixel summing is done. For this option, the data uncertainties are used. The data uncertainties include the photon noise from the source itself, so this method is more general than the previous estimate, based solely upon the background. This is usually the best option.

2. Photerr, with gain: If the data uncertaintes are unavailable, or unreliable, then the source photon noise can be put in explicitly. This option brings APEX in line with noise estimates from other standard software (e.g. SExtractor; Bertin & Arnouts 1996).

In the case with gain, the noise for the SNR is:

$$\sigma^2 = \sum_i \sigma_i^2 + F/g$$

Equation 8.30

where σ_i are the pixel uncertainties in the same box described above, F is the source flux in data units, and g is the gain (electrons per data unit). The final noise is converted to flux units. The input σ_i in this case should be the estimates obtained from the Gaussnoise module. The gain to use is the gain for a single image. It will take into account the coverage to estimate an effective gain. In the case of integration-time weighting, it should be for unit time.

8.9.3 Aperture Uncertainties

Aperture uncertainties are calculated in the **Aperture** module. It uses the data uncertainties (or "gain-estimated" ones, if enabled). The aperture flux uncertainty is a sum over the aperture area:

$$\sigma^2 = \sum(\sigma(i)^2)$$

Equation 8.31

The sum takes into account fractional pixels. If any have bad uncertainties, ap_unc is set to -9.99. Uncertainty in the background subtraction is not included.

The results are written out in the *aperture.tbl* and *extract_raw.tbl* as "ap_unc1", "ap_unc2", etc. The APEX module, Select, can select on them when creating the final *extract.tbl*.

Uncertainties will only be calculated if the uncertainty file is found, and is the right size. If not found, or not the right size, Aperture Photometry will proceed but set the aperture uncertainties to -9.99.
It is possible to run Aperture Photometry by itself on an extract list, but if the extract list already contains aperture columns, then Aperture Photometry will append new columns.

8.10 The User List Input Table

When running APEX User List and APEX User List Single Frame, the user is required to enter a table listing the point sources that APEX should extract. This replaces the running of the Detect module. For more information on APEX User List, see Chapter 6. This table is given in the APEX User List Settings module as *User Input Fixed Positions Table File Name*, or in the command-line namelist as *INPUT_USER_LIST*.

There are several ways to create this table:

1. Provide a text file with a list of RA and Dec coordinates, with **exactly** the following format:

```
|RA                |Dec               |
|double            |double            |
 274.072341         -2.544706
 274.072895         -2.531989
```

2. Edit a previously-generated source table that was output by the Detect module, to include only those sources that you wish to extract. Note that in this case, you may find it easier to simply run APEX with Select Detect, rather than running APEX User List;

3. If you are running APEX User List Single Frame then you can use the GUI to interactively select the sources that you wish to extract. Start by selecting an input file in the Initial Setup module, open the APEX User List single frame Settings. You have the choice to either input an existing User List file, or Create User Input List. Click on the Create User Input List button, and MOPEX will bring up your input mosaic in a separate window, alongside a User Input List control window. Left-clicking on a source in the image window will add its position to the control window. Ctrl-Left click on a previously-added point source to select it without adding a second source at the same position. Shift-Left click on a source to center the image at that point. If you wish to delete a source from the input table, select it in the control window and click the "Delete Selected" button. Note that selecting a source is not the same as checking the box to the left. The checkbox simply controls whether the source is shown in the image window or not. You can also delete all sources except the ones you have selected by using the "Keep Selected" button.

   Once you have finished selecting your sources, press the "Save Table" button, give the file a sensible name, and then click "Save". Click "Done" to return to the APEX User List single frame Settings window. Finally, load your newly-saved table into the *User Input Fixed Positions Table File Name.*

8.11 Fatal Mask Bit Patterns

It is highly likely that when you reduce your Spitzer data, you will include at least one set of mask files as input into the pipeline. The pixels in the mask files all contain coded information, or "bit definitions", depending on the status of a particular pixel. For example, a pixel with bit 10 flagged in the IRAC channel 1 PMask file indicates that that particular pixel in the IRAC channel 1 array has an excessive dark current that will affect all observations, while bit 14 indicates that the pixel is dead. For a full list of bit definitions, see the IRAC and MIPS Instrument Handbooks. When running MOPEX, you need to decide which of these codes you wish to set as "fatal", i.e. which of the flagged problems you consider so bad that the corresponding pixel in your data frame should be discarded. This is done by specifying a "Fatal Mask Bit Pattern" for each mask type, in the Initial Setup module. On the command line, the Fatal Bit Patterns are set in the general settings part of the namelist. The Fatal Mask Bit Patterns are calculated as follows:

$$\textit{Fatal Bit Pattern} = 2^{(\text{value of 1st required fatal bit})} + 2^{(\text{value of 2nd fatal bit})} + ... + 2^{(\text{value of nth fatal bit})}$$

e.g. if you wished to set bits 7, 8, 9, 10 and 14 as fatal for your IRAC PMask file then you would set your *PMask Fatal Mask Bit Pattern* (*PMask_Fatal_BitPattern* in namelists) to: $(2^7 + 2^8 + 2^9 + 2^{10} + 2^{14}) = 18304$. On the command line, you would include the following line in your namelist:

```
PMask_Fatal_BitPattern = 18304
```

The Fatal Bit Patterns for the other mask files can be set in the command-line namelists with the keywords:

```
DCE_Status_Mask_Fatal_BitPattern
```

```
RMask_Fatal_BitPattern
```

The currently recommended Fatal Mask Bit Patterns for each instrument are listed in the GUI templates.

Chapter 9. Running MOPEX on the Command Line

9.1 Setting up MOPEX

MOPEX can be downloaded from the website. We assume that you have downloaded the installation file and followed the installation instructions. The following steps must then be followed to set up MOPEX for the command line:

- Edit the file *mopex-script-env.csh* (found in the MOPEX installation directory) to set the installation path MOPEX_INSTALLATION to the current directory (pwd) e.g. if MOPEX is in */home/joe/mopex*, then you should set the following in *mopex-script-env.csh*:

  ```
  setenv MOPEX_INSTALLATION /home/joe/mopex.
  ```

- Running cshell (not bash), source *mopex-script-env.csh*:

  ```
  unix% source mopex-script-env.csh
  ```

- MOPEX can now be run from the command line using the syntax described at the bottom of this section.

9.2 Setting up the MOPEX directory structure

Once MOPEX has been set up for the command line, you need to create the expected directory structure. We will call the directory containing your data <*data_dir*>, the directory containing the MOPEX software <*mopex_dir*>, and the directory from which you are planning to run MOPEX <*working_dir*>. Your <*working_dir*> can be <*mopex_dir*>, but doesn't have to be.

MOPEX expects 4 basic input types:

1. your input FITS images (e.g. Spitzer BCDs);
2. ASCII text files listing the input images;
3. a file listing the parameters you want to use to reduce your data (the "namelist");
4. any required calibration files (e.g. pmasks, sample PRFs etc).

In order to run MOPEX, your FITS images should be in <*data_dir*>, your lists containing the input images can be stored anywhere (default is in <*working_dir*> but can be changed by specifying the full path) and your namelist must be in a subdirectory called <*working_dir*>*/cdf/*. The calibration files must be in the subdirectory <*working_dir*>*/cal/*. Once this directory structure has been set up, you are ready to run MOPEX. Please see Chapter 3 on MOPEX Input for details on how to set up the input list files.

9.3 Running MOPEX

Once your directory structure is set up and you have created your input files and stored them in the correct place, you are ready to run MOPEX from <*working_dir*>. MOPEX is run on the command line by calling a set of Perl wrapper scripts (stored in <*mopex_dir*>*/bin/*) to perform the basic reduction tasks (e.g. background matching, mosaicking of the input images, point source extraction). Each of these wrapper scripts runs a series of individual "modules" in sequence to read in the data, carry out the reduction and write the output. Depending on how you wish to process your data, some of these individual modules need to be turned on or off within the processing flow. The choice of modules, along with various other options, is controlled from within an ASCII parameter file called the "namelist".

MOPEX comes with example template namelists in the directory <*mopex_dir*>*/cdf/* .For a full list of reduction scripts available from the command line, please see Appendix A.

As an example of how to run MOPEX, a typical reduction involves background-matching your input images, mosaicking them together, and then carrying out point source extraction on the resultant mosaic. These three steps require three scripts: *overlap.pl*, *mosaic.pl* and *apex_1frame.pl*. The syntax for running MOPEX is as follows:

```
unix% script.pl <option flag> <specification>
```

where the option flags can be used to specify input and output files, and must be matched with a specification. More than one option flag can be specified, and they can be given in any order. The option flags that can be used vary from script to script, and are all listed in the sections for the different pipelines below (§9.5 - §9.8).

Each script expects a list of FITS images and a namelist file as input. The format of the input list is given in §3.2 MOPEX Input List Files, and the format of the namelist is described in §9.4 MOPEX Input Namelist.

The default locations for all of the files except for the namelist file can be overridden by providing the full path name, for example:

```
unix% overlap.pl -n overlap_I1.nl -I /data/IRAC/bcd/ImageList.txt
```

Not all of the input and output options have to be specified on the command line - most can be specified from within the namelist. The only option that cannot is the name of the namelist itself. The simplest possible (and recommended) way to run MOPEX would therefore be to specify all of the options in the namelist. This option would look something like:

```
unix%  mosaic.pl -n mosaic_namelist.nl
```

If a parameter is defined in both the namelist and on the command line, the command line takes precedence.

The default location for the input files is in <*working_dir*>. When specifying input files, you should use either the full path name or the path relative to the working directory. Calibration files (e.g. PRF files and PMasks) are assumed by MOPEX to be in the *cal/* subdirectory of the current working directory.

While running, the Perl scripts will send information to the standard output, usually the screen. Under normal circumstances, this information is not vital, but you can save it to a file with a redirect "> save.txt". Some modules also save a log file in *cdf/log_(module name)*.

9.4 MOPEX Input Namelist

See Chapter 3 for information about all of the other input files.

The namelist file for each of the Perl processing scripts contains all of the input parameters to be used by MOPEX. Namelists can either be set up in a text file for use on the command line, or set up within the GUI. Namelists can be imported into/exported from the GUI and are interchangeable between the two interfaces. While the underlying code for MOPEX is identical for both the GUI and the command line, some of the parameter names were changed in the GUI to make their function clearer. Links to lists of all of the parameter names in the GUI and on the command line can be found near the bottom of the main MOPEX webpage.

The namelist must be saved in a subdirectory of the working directory called <*working_dir*>*/cdf/*. Specifying a full path name to the namelist when calling MOPEX will not over-ride this. This is only true when running MOPEX on the command line - if running MOPEX

from the GUI, the namelist can reside in any directory. An example of a namelist can be found in the *cdf/* directory of your MOPEX distribution.

The namelist has the following structure:

1. **Processing flow control:** depending on the options you wish to use when running MOPEX, you will want to run some of the available modules but not others. This first part of the namelist controls the switching on and off of modules in the processing flow by setting "triggers" (keywords) for the modules to 1 or 0. For example:

   ```
   run_medfilter = 1              indicates "run the MedFilter module"
   run_medfilter = 0              indicates "do not run the MedFilter module"
   run_mosaic_dual_outlier = 1    indicates "run the Mosaic Dual Outlier module" etc.
   ```

   If a module trigger word is not included in the namelist, it is set to the default value of 0.

2. **Global parameters:** a list of input parameters that are common to all modules, e.g. input files, pixel sizes, etc. For example:

   ```
   IMAGE_STACK_FILE_NAME = /home/joe/data/IRAC/ch1/bcd/ImageList.txt
   MOSAIC_PIXEL_SIZE_X = -0.00033889
   MOSAIC_PIXEL_SIZE_Y = 0.00033889
   OUTPUT_DIR = /home/joe/data/IRAC/ch1/Mosaic
   ```

   The default values differ for different parameters - check the module descriptions for the individual processing flows.

3. **Module parameter blocks:** for each module, a set of input parameters specific to that module must be given. For example:

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

Module triggers, settings, names and parameters, both inside and outside the module blocks, can be given in any order. Their order in the namelist will not affect the order in which they are run by MOPEX. Comments in the namelist are preceded by a "#".

IMPORTANT NOTE: all input variables in the namelists require a space preceding and following the equals sign, i.e. "variable = value". An entry like "variable=value" will not be

read and the hard-coded default will be used. You will not be warned that your input value is not being read.

9.4.1 Processing Flow Control

Depending on the pipeline you are intending to run (e.g. Overlap, Mosaic, APEX), you will need to activate different modules in your namelist. For more information on which modules you are likely to need, you should see §2.5: "Which Modules Should I Choose?" and the example instrument-specific namelists in the <*mopex_dir*>*/cdf/* directory.

For example, if you are planning to run the Background Matching (Overlap) pipeline then you may choose to run the Fiducial Image Frame, Mosaic Interpolate and Compute Overlap Correction modules, and then apply the correction to your images. In this case you would include the following lines in your namelist:

```
run_fiducial_image_frame = 1
run_mosaic_interp = 1
compute_overlap_correction = 1
apply_overlap_correction = 1
```

The keywords required to run each module are given on the individual module description pages (along with input/output parameters and an in-depth discussion of the purpose of the module) and in the table in 0: MOPEX Module List.

If a module is turned on in the pipeline then a corresponding parameter block must be included in the namelist. For more detailed information on each pipeline and each module, please see the relevant pipeline- or module-specific pages.

9.4.2 Global Parameters Outside the Module Blocks

There are 3 main types of information in the Global Parameters section: Input/Output file and directory names; Fatal Bit Patterns for use with the Spitzer mask files; and more general options that control preferences in how the processing is carried out. The available parameters are sorted by the modules that they apply to, and everything in the module descriptions that is labeled with "In Global Parameters" should go into this section.

9.4.2.1 Input Files and Output Directories

The namelist can be used to specify the input files and output directories for each module, instead of specifying them on the command line when running a pipeline. The namelist also allows you to customize the output directory name for each processing script, and each module within that

script. The default output directory for all the scripts is the current working directory. To change this you should add the following line into your namelist:

```
OUTPUT_DIR = <your choice of output dir>
```

All modules have a default output directory that is a subdirectory of OUTPUT_DIR. In order to change these you should add lines similar to the one above into your namelist. The specific keyword you need to change the output directory of each module is given on the individual module help pages, along with the default directory name. For example, in order to change the output directory for the MedFilter module, look up the MedFilter Module section (e.g. §5.6.6), scroll down to the COMMAND LINE INPUT, and set the new directory name by adding the keyword into your namelist:

```
MEDFILTER_DIR = /your/new/medfilter/output/dir
```

Generally there is no need to change the default subdirectory names.

9.4.2.2 Setting Fatal Bit Patterns for Mask Files

See §8.11 for information on the Fatal Mask Bit Patterns. They are set here in the Global Parameters section of the namelist.

9.4.2.3 Other Global Parameters

The global parameters that can be specified outside the module blocks vary from pipeline to pipeline. Options include e.g. creating different types of mosaics, choosing whether MOPEX should print error messages to the command line, or whether the intermediate steps in the processing should be saved or not. Most of the available options are included in the module descriptions, but there are a few extra options only available on the command line:

verbose: Instructs MOPEX to print the output from each individual module. This is extremely useful for troubleshooting. If MOPEX encounters a problem while verbose is turned off, it may stop without printing an error message.

NICE: Runs MOPEX with the UNIX command "nice 19". If you are not familiar with nice, you can check out the Wikipedia description at http://en.wikipedia.org/wiki/Nice_(Unix).

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 the output directory is given a unique name, which is created by appending the namelist filename to the time of execution. For example, if you ran:

```
unix% 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* and will be overwritten in subsequent runs that use the same namelist.

9.4.3 Module Parameter Blocks

Each of the modules that is switched on must be accompanied by a parameter block with the input parameters for that particular module. A typical parameter block looks like this:

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

where the "`&MEDFILTER`" line tells MOPEX that this is the start of the parameter block for the MedFilter module and "`&END`" denotes the end of the block. Examples of parameter blocks for each module can be found on the individual module help pages, along with a full list of input parameters.

The important thing to note when editing the module blocks is that each line in a block, even a comment line headed by a "#", must end in a comma.

9.5 Running Overlap on the command line

The syntax for running *Overlap* is as follows:

```
unix% overlap.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.1. For example, to run Overlap using a namelist *cdf/overlap_I1.nl* on a list of images in */data/IRAC/bcd/ImageList.txt*, you would type the following:

```
unix% overlap.pl -n overlap_I1.nl -I /data/IRAC/bcd/ImageList.txt
```

Table 9.1: List of options that can be called from the command line when running *overlap.pl*

Option Flag	Specification File (example)	Default Location	Required/Optional

-n	namelist of input parameters (*mosaic.nl*)	*./cdf/*	required
-I (capital i)	list of input images (*ImageList.txt*)	*./*	required
-S	list of input uncertainty images (*SigmaList.txt*)	*./*	optional
-D	list of DCE status mask images (*IMaskList.txt*)	*./*	optional
-M	permanently damaged pixel mask image (**pmask.fits*)	*./cal/*	optional
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-O	Output directory	*./*	optional

9.6 Running Mosaic on the command line

Mosaicking is done on the command line using the Perl wrapper script *mosaic.pl*. The syntax is as follows:

```
unix% mosaic.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.2. The default locations for all of the files except for the namelist file can be overridden by providing the full path name, for example:

```
unix% mosaic.pl –n mosaic_I1.nl –I /data/IRAC/ch1/bcd/ImageList.txt
```

would run the Mosaicking script mosaic.pl using the namelist *cdf/mosaic_I1.nl* and the input list in */data/IRAC/ch1/bcd/ImageList.txt*.

Table 9.2: List of options that can be called from the command-line when running *mosaic.pl*

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist of input parameters (*mosaic.nl*)	*./cdf/*	required
-I (capital i)	list of input images (*ImageList.txt*)	*./*	required
-S	list of input uncertainty images (*SigmaList.txt*)	*./*	optional
-d	list of DCE status mask images (*IMaskList.txt*)	*./*	optional
-M	permanently damaged pixel mask image (**pmask.fits*)	*./cal/*	optional
-R	list of outlier mask images (*RMaskList.txt*)	*./*	optional
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-O	Output directory	*./*	optional

9.7 Running APEX on the command line

9.7.1 APEX Multiframe (apex.pl)

APEX Multiframe is run on the command line using the perl wrapper script *apex.pl*. The syntax is as follows:

```
unix% apex.pl <option flag> <specification>
```

where <option flag> and the associated <specification> are given in Table 9.3. The default locations for all the files except for the namelist file can be overridden by providing the full path name. For example:

```
unix% apex.pl –n apex_I1.nl –I /data/IRAC/ch1/bcd/ImageList.txt
```

will run apex.pl using the namelist *cdf/apex_I1.nl* and the input image list given in */data/IRAC/ch1/bcd/ImageList.txt*.

9.7.2 APEX User List Multiframe (apex_user_list.pl)

APEX User List Multiframe is run as follows:

```
unix% apex_user_list.pl <option> <specification>
```

The available options are the same as for APEX Multiframe, with the addition of one more flag, `-u`.

Table 9.3: List of options that can be called from the command line when running *apex.pl and apex_user_list.pl.*

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist of input parameters (*apex.nl*)	*./cdf/*	required
-I (capital i)	list of input images (*ImageList.txt*)	*./*	required
-u	name of user input source list (user_list.tbl)	*./*	required for User List mode only.
-m	PRF map (*PRFmap.tbl*)	*./cal/*	either a PRF map or a PRF image is required
-p	PRF image (*PRF.fits*)	*./cal/*	either a PRF map of a PRF image is required
-P	mosaic PRF image (*Mosaic_PRF.fits*)	*./cal/*	required
-S	list of input uncertainty images (*SigmaList.txt*)	*./*	optional
-d	list of DCE status mask images (*IMaskList.txt*)	*./*	optional
-M	permanently damaged pixel mask image (*pmask.fits*)	*./cal/*	optional
-R	list of outlier mask images (*RMaskList.txt*)	*./*	optional
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-O	Output directory	*./*	optional

9.7.3 APEX QA Multiframe (apex_qa.pl)

Residual image creation for APEX Multiframe is run on the command line as follows:

```
unix% apex_qa.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.4. As with the *apex.pl* script, default locations for all of the files except for the namelist file can be overridden by providing the full path name.

Table 9.4: List of options that can be called from the command line when running *apex_qa.pl* in multiframe mode (i.e. with *apex.pl* or *apex_user_list.pl*)

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist for creating residual images (*apex_qa.nl*)	*./cdf/*	required
-u	namelist for mosaicking of residual images (*mosaic_qa.nl*)	*./cdf/*	required
-E	point source list from apex.pl (*extract.tbl*)	*./*	required
-I (capital i)	list of input images (*ImageList.txt*)	*./*	required
-m	PRF map (*PRFmap.tbl*)	*./cal/*	either a PRF map or a PRF image is required
-p	PRF image (*PRF.fits*)	*./cal/*	either a PRF map of a PRF image is required
-S	list of input uncertainty images (*SigmaList.txt*)	*./*	optional
-d	list of DCE status mask images (*IMaskList.txt*)	*./*	optional
-M	permanently damaged pixel mask image (*pmask.fits*)	*./cal/*	optional
-R	list of outlier mask images (*RMaskList.txt*)	*./*	optional
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-O	Output directory	*./*	optional

9.7.4 Single-frame mode (apex_1frame.pl)

APEX Single Frame is carried out on the command line using the Perl wrapper script *apex_1frame.pl*. The syntax is as follows:

```
unix% apex_1frame.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.5. The default locations for all of the files except for the namelist file can be overridden by providing the full path name, for example:

```
unix% apex_1frame.pl –n apex_1frame_I1.nl –i /data/IRAC/mosaic_ch1.fits
```

See Chapter 3 on MOPEX Input for more information on the input image and list files, and §9.4 for a description of the namelist file.

9.7.5 APEX User List Single Frame (apex_user_list_1frame.pl)

APEX User List Single Frame is run as follows:

```
unix% apex_user_list_1frame.pl <option> <specification>
```

The available options are the same as for APEX Single Frame, with the addition of one more flag, -u. See Table 9.5 for the available input options.

9.7.6 Single Frame Residual Image Creation (apex_qa.pl)

Residual image creation for APEX Single Frame is run on the command line as follows:

```
unix% apex_qa.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.6. As with the *apex_1frame.pl* script, default locations for all of the files except for the namelist file can be overridden by providing the full path name.

Table 9.5: List of options that can be called from the command line when running *apex_1frame.pl or apex_user_list_1frame.pl*

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist of input parameters (*apex_1frame.nl*)	*./cdf/*	required
-i	input image (*mosaic.fits*)	*./*	required
-u	name of user input source list (user_list.tbl)	*./*	required for User List mode only.
-m	PRF map (*PRFmap.tbl*)	*./cal/*	either a PRF map or a PRF image is required
-p	PRF image (*PRF.fits*)	*./cal/*	either a PRF map of a 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_imask.fits*)	*./*	optional
-M	permanently damaged pixel mask image (*pmask.fits*)	*./cal/*	optional
-R	outlier mask image (*mosaic_rmask.fits*)	*./*	optional
-O	Output directory	*./*	optional

Table 9.6: List of options that can be called from the command line when running *apex_qa.pl* in single-frame mode (i.e. with *apex_1frame.pl*)

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist for creating residual images (*apex_qa.nl*)	*./cdf/*	required
-T	input image (*mosaic.fits*)	*./*	required
-E	point source list from apex_1frame.pl (*extract.tbl*)	*./*	required
-P	mosaic PRF image (*PRF.fits*)	*./cal/*	required
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-O	Output directory	*./*	optional

9.8 Running PRF Estimate on the command line

PRF Estimate is run on the command line using the script *prf_estimate.pl*. The script expects a list of FITS images and a namelist file as input. The format of the input list is given in §3.2 MOPEX Input List Files, and the format of the namelist is described in §9.4. The syntax for running PRF Estimate is as follows:

```
unix% prf_estimate.pl <option flag> <specification>
```

where `<option flag>` and the associated `<specification>` are given in Table 9.7. The default locations for all of the files except for the namelist file can be overridden by providing the full path name, for example:

```
unix% prf_estimate.pl -n prf_estimate_M1.nl -i /data/MIPS24/mosaic.fits
```

Table 9.7: List of options that can be called from the command line when running *prf_estimate.pl*

Option Flag	Specification File (example)	Default Location	Required/Optional
-n	namelist of input parameters (*prf_estimate.nl*)	*./cdf/*	required
-I (capital i)	list of input images (if running on a stack of images) (*ImageList.txt*)	*./*	required
-i	input image name (if running on a single image) (*mosaic.fits*)	*./*	optional
-E	Point Source List (*mips24_extract.tbl*)	*./*	optional
-M	permanently damaged pixel mask image (*pmask.fits*)	*./cal/*	optional
-F	Fiducial Image Frame file (*FIF.tbl*)	*./<output directory>/*	optional
-d	list of DCE status mask images (*BMaskList.txt*)	*./*	optional
-R	list of outlier rejection mask (RMask) images (*RMaskList.txt*)	*./*	optional
-O	Output directory	*./*	optional

Appendix A. Full List of MOPEX Scripts

Here is a full list of all of the scripts available in MOPEX. Some of them are not available in the GUI, and so can only be run from the command-line. For the command-line only scripts, the documentation can be found in the given README files, which can be found in the *readme/* directory of your MOPEX distribution.

The scripts included with MOPEX are split into "Supported" and "Under Construction". Supported scripts are verified and fully supported, and are split into "Commonly Used" and "Auxiliary" scripts. Those under construction are either unfinished or have not been verified, and are used entirely at the user's risk. We do not support scripts under construction.

A.1 Supported Scripts

A.1.1 Commonly Used

Tool	Description	GUI?	Documentation
aperture.pl	Computes aperture fluxes	N	aperture.pdf
apex.pl / apex_1frame.pl	Performs multiframe / single frame point source extraction	Y	This manual, Chapter 6
apex_user_list.pl / apex_user_list_1frame.pl	Carries out multiframe / single frame point source extraction on a user-defined source list.	Y	This manual, Chapter 6
apex_qa.pl	Creates point source subtracted mosaic	Y	This manual, Chapter 6
cosmetic.pl	For IRAC data only; removes muxbleed and column pull-down artifacts in IRAC images. Superceded by Sean Carey's IDL code, available as Contributed Software.	N	README_cosmetic (txt)
fiducial_frame.pl	Creates mosaic projection table.	N	--
flatfield.pl	Performs final flatfielding. Not recommended for use with IRAC data	N	README_flatfield (txt)
mosaic.pl	For mosaicking the BCDs with/without outlier detection	Y	This manual Chapter 5
mosaic_sed.pl	Performs interpolation and coadition of MIPS SED (spectral energy distribution) images, and extraction	N	mosaic_sed.pdf

Tool	Description	GUI?	Documentation
	of the spectra		
overlap.pl	Performs background matching on BCDs	Y	This manual Chapter 4
prf_estimate.pl	Runs PRF (Point Response Function) estimation	Y	This manual Chapter 7

A.1.2 Auxiliary

Tool	Description	GUI?	Documentation
copy_HDR_keywords.pl	Copies HDR-mode header keywords	N	--
easymosaic.pl	Produces a quick-look mosaic without having to specify parameters	N	README_easymosaic (txt)
hdr_mask.pl	Replaces saturated pixels in the long integration time frames with their counterparts from the short integration time frames	N	README_hdr_mask (txt)
HDR_Split.pl	Splits file lists into short, medium and long exposures in HDR mode	N	--
mosaic_mask.pl	Creates a mosaic of the mask images(e.g. creates a mosaic of RMask images)	N	mosaic_mask.pdf
numbertable.pl	Numbers the rows in a table	N	--
pointing_refine.pl	Performs absolute/relative pointing refinement (generally not needed as pointing refinement is now done automatically as part of the IRAC pipeline)	N	pointingrefine.html
PSEutils.pl	Library functions. DO NOT EDIT	N	--
swap_refined_keywords.pl	Swaps the refine pointing and the default keywords (e.g. to view in ds9 since it won't automatically see the refined keywords)	N	--
table_1col_stat.pl	Computes histogram for a table column	N	--
table_stat.pl	Computes statistics for a table column	N	--

A.1.3 Under Construction

Tool	Description	GUI?	Documentation
fastsortrealmatch.pl	Source matching	N	--
myscore.pl	Performs completeness test	N	--
photo.pl	Flux error comparison	N	--
simulate.pl	Makes BCD-like data from the mosaic	N	--

Appendix B. Full List of Modules in MOPEX

The following tables contain a full list of all of the modules available in MOPEX, with their namelist triggers, a brief description, and the prerequisites for each module. The modules are organised in the order that you might encounter them in the MOPEX scripts.

B.1 Overlap Modules

Module Name	Namelist Trigger	Brief Description	Module Prerequisites
Initial Setup	N/A	Set up the input files. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
Overlap Settings	N/A	Sets up the pipeline-specific global input parameters. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
S/N Estimator	compute_uncertainties_internally	Internally estimates the uncertainty for each pixel in the image when no independent uncertainty estimate is available. Only use if you do not trust the uncertainty images provided with your Spitzer data download.	Initial Setup, Overlap Settings
MedFilter	run_medfilter	Performs background subtraction of the individual images. Only required if you intend to use Bright Object Masking	Initial Setup, Overlap Settings
Fiducial Image Frame	run_fiducial_image_frame	Creates the unified grid coordinate system and defines the spatial boundaries that will include all of the input data frames (the FIF table).	Initial Setup, Overlap Settings
Detect (Outlier)	run_detect_outlier	Performs image segmentation and produces detection maps of bright objects. This identifies bright pixels that should not contribute to the background matching calculation.	Initial Setup, Overlap Settings
Mosaic Interpolate	run_mosaic_int	Performs a projection of input images onto a 2D plane defined by the FIF table and interpolates the input pixels to the output array. Also corrects for optical distortion in the input images.	Fiducial Image Frame; S/N Estimator (if S/N Estimator is turned on).

Compute Overlap Correction	compute-overlap-correction	Computes the correction needed to set the background of all overlapping images to a constant value. Optionally applies this correction to the input images.	Mosaic Interpolate
Quicklook Mosaic	mosaic-corrected-images	Creates a quick and dirty mosaic of the background-corrected images to check the results of the overlap correction.	Mosaic Interpolate

B.2 Mosaic Modules

Module Name	**Namelist Trigger**	**Brief Description**	**Module Prerequisites**
Initial Setup	N/A	Set up the input files. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
Mosaic Settings	N/A	Sets up the pipeline-specific global input parameters. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
S/N Estimator	compute_uncertainties_internally	Internally estimates the uncertainty for each pixel in the image when no independent uncertainty estimate is available. Only use if you do not trust the uncertainty images provided with your data download.	Initial Setup, Mosaic Settings
MedFilter	run_medfilter	Performs background subtraction of the individual images. Only required if you intend to use Single Frame or Dual Outlier detection.	Initial Setup, Mosaic Settings
Detect RadHit	run_detect_radhit	Performs single frame radhit rejection. It is the simplest of the outlier rejection schemes.	Initial Setup, Mosaic Settings
Fiducial Image Frame	run_fiducial_image_frame	Creates the unified grid coordinate system and defines the spatial boundaries that will include all of the input data frames (the FIF table).	Initial Setup, Mosaic Settings
Mosaic Geometry	run_mosaic_geom	Finds all of the input images that overlap with the FIF table, allowing the user to make a mosaic of a subset of the input data frames.	None, but requires a hand-edited version of the Fiducial

			Image Frame table.
Mosaic Interpolate	run_mosaic_int	Performs a projection of input images onto a 2D plane defined by the FIF table and interpolates the input pixels to the output array. Also corrects for optical distortion in the input images.	Uses the output from S/N Estimator (but only if S/N Estimator is turned on).
Detect (Outlier)	run_detect_outlier	First of four modules run as part of dual outlier detection. Performs image segmentation and produces detection maps of point sources and radhits.	None
Mosaic Projection	run_mosaic_proj	Second of four modules run as part of dual outlier detection. Projects the detection maps from Detect (Outlier) onto a common reference frame, and interpolates.	Detect (Outlier)
Mosaic Coverage	run_mosaic_covg	Produces a preliminary coverage map covering the whole FIF for use in the later module Mosaic RMask.	Mosaic Interpolate
Mosaic Dual Outlier	run_mosaic_dual_outlier	Third of four modules run as part of dual outlier detection. Applies the user-defined criteria to classify outliers using both spatial and temporal information.	Mosaic Projection
Level	run_level	Last of four modules run as part of dual outlier detection. Corrects the dual outlier maps produced in Mosaic Dual Outlier to ensure that the edges of point sources are not accidentally rejected	Mosaic Dual Outlier
Mosaic Outlier	run_mosaic_outlier	Runs the multiframe outlier rejection method to flag bad pixels.	Mosaic Interpolate
Mosaic Box Outlier	run_mosaic_box_outlier	Runs the Box Outlier rejection method to flag bad pixels.	Mosaic Interpolate
Mosaic RMask	run_mosaic_rmask	Combines outlier rejection information into a single RMask: bit 0 (single frame radhit detection), bit 1 (multiframe temporal outlier detection), bit 2 (dual outlier detection), bit 3 (box outlier detection)	Mosaic Outlier, Level, Mosaic Coverage
Mosaic Reinterpolate	run_mosaic_reinterp	Re-runs the interpolation, this time including information from the RMasks. Only pixels flagged by the	Mosaic Interpolate,

		RMasks are reinterpolated.	Mosaic RMask
Fix Coverage	run_fix_coverage	After all outlier pixels have been identified, this module gives the option to suppress the use of pixels that have very small coverage (repeated observations at the same point on the sky). Pixels whose usage is suppressed have their coverage set to 0 in new coverage maps.	Mosaic Interpolate (Mosaic Reinterpolate)
Mosaic Coadder	run_mosaic_coadder	Co-adds the interpolated images to create one mosaic image. Co-addition can be performed on Tiles to address computer memory considerations. Several co-addition schemes are available.	Mosaic Interpolate (Mosaic Reinterpolate)
Mosaic Combiner	run_mosaic_combiner	Combines the co-added tiles into a single mosaicked image. This module must be run, even when there is only one tile, in order to create the expected output files.	Mosaic Coadder
Mosaic Medfilter	run_mosaic_medfilter	Performs median background subtraction of the mosaic image.	Mosaic Combine

B.3 APEX (User List) and APEX (User List) 1Frame Modules

Module Name	**Namelist Trigger**	**Brief Description**	**Module Prerequisites**
Initial Setup	N/A	Set up the input files. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
APEX Settings	N/A	Sets up the pipeline-specific global input parameters. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
Fiducial Image Frame	run_fiducial_image_frame	Creates the unified grid coordinate system and defines the spatial boundaries that will include all of the input data frames (the FIF table).	Initial Setup, APEX Settings
Mosaic Interpolate	run_mosaic_int	Performs a projection of input images onto a 2D plane defined by the FIF table and interpolates the input pixels to the output	Fiducial Image Frame

		array. Also corrects for optical distortion in the input images. Only required if you are not using the results from the Mosaic processing.	
Mosaic Coadder	run_mosaic_coadder	Co-adds the interpolated images to create one mosaic image. Co-addition can be performed on Tiles to address computer memory considerations. Several co-addition schemes are available. Only required if you are not using the results from Mosaic processing	Mosaic Interpolate
Mosaic Combiner	run_mosaic_combiner	Combines the co-added tiles into a single mosaicked image. This module must be run, even when there is only one tile, in order to create the expected output files. Only required if you are not using the results from Mosaic processing.	Mosaic Coadder
Detect MedFilter	run_detect_medfilter	Performs background subtraction of the individual or mosaicked images that will be used for source detection.	Mosaic Combine (or the output of mosaic.pl)
Gaussnoise	run_gaussnoise	Estimates the background flucuations in the input image for use in the signal-to-noise ratio estimation	none
Point Source Probability	run_pointsourceprob	Filters the input coadded image to estimate the probability at each pixel of having a point source above the noise.	Mosaic Coadd (or the output of mosaic.pl)
Bright Detect	run_bright_detect	Detects bright sources in the input images, and includes source shape information. Does not record the blends, and so generally cannot replace Detect.	Point Source Probability
Detect	run_bright_detect	Performs image segmentation and computes the centroids for detected point sources.	Point Source Probability
Extract MedFilter	run_extract_medfilter	Performs background subtraction of either the mosaicked image (apex_1frame.pl) or tiles (apex.pl) that will be used for point source extraction.	Detect
Detection Map	run_detectionmap	Required by APEX User List multiframe to track the input positions through the stack of input BCDs.	Initial Setup, APEX Settings
Fit Radius	run_fit_radius	Determines a fitting area for each source in the detect table. This module is optional if the parameters Fitting_Area_X,Y are	Detect

		defined in the Source Estimate module.	
Select Detect	run_select_detect	Allows the user to select the columns and rows from the detect table to be written out to the full extract table	Detect
Source Estimate	run_sourcestimate	Performs point source estimation using PRF-fitting. Estimates fluxes refines the positions of the point source candidates from the detect table.	Detect; Fit Radius
Aperture (Photometry)	run_aperture	Calculates the flux(es) within the specified circular aperture(s) for each source in the extract table. The module will optionally subtract a background value calculated from a user-defined annulus.	Source Estimate
Select	run_select	Allows the user to select the columns and rows to be written to the final extract table, based on a range of user-specified conditions.	Source Estimate; Aperture (Photometry)

B.4 APEX QA Modules

Module Name	**Namelist Trigger**	**Brief Description**	**Module Prerequisites**
Initial Setup	N/A	Set up the input files. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
APEX QA Settings	N/A	Sets up the pipeline-specific global input parameters. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
Point Source Subtract	create_residual_images (with apex.pl) create_residual_mosaic (with apex_1frame.pl)	Subtracts the extracted point sources from the input images or the input mosaic to create residual images. Invaluable for testing the results of PRF-fitting.	Initial Setup, APEXQA Settings, Output from either apex.pl or apex_1frame.pl

B.5 PRF Estimate Modules

Module Name	Namelist Trigger	Brief Description	Module Prerequisites
Initial Setup	N/A	Set up the input files. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
PRF Estimate Settings	N/A	Sets up the pipeline-specific global input parameters. On the command line, this is done in the main body of the namelist. See §9.4 for details.	None
MedFilter	run_medfilter	Performs the background subtraction of the individual input images.	Initial Setup, PRF Estimate Settings
Crop Stack	run_crop_stack	Cuts out postage stamp images from the input images around each point source from the input point source list	MedFilter
Split By Array Position	split_by_array_position	Allows the user to direct MOPEX to divide the detector array into a number of sectors and generate a separate PRF file for each.	Initial Setup, PRF Estimate Settings
PRF Estimate	run_prf_estimate	Combines the postage stamp images generated by the module Crop Stack into the final PRF/PRF Map.	Crop Stack, Split By Array Position

Appendix C. PRF Fitting Correction Factors for APEX: Version 1 (10/20/2010)

When PRF fitting, APEX normalizes the PRF, either out to the limits of the file provided, or with a Normalization Radius given in the Source Estimate block. Given the finite size of the PRF files, and differences in the way the Spitzer instruments are calibrated, correction factors are needed to the PRF-fitted fluxes (the column labeled "flux" in the extract tables).

The following is a wide table, broken up to fit vertically. If using the PRF and Normalization Radius values indicated, which are the defaults, the user should divide "flux" by the final column. The factors for IRAC are discussed in detail in the IRAC Instrument Handbook. These are for data from the cryogenic mission.

Instrument	**Script/Flow**	**Default Namelist**	**Fit on Images or Mosaic?**	**Pixel Size (arcsec)**
IRAC 1	Apex	apex_I1_have_mosaic.nl[a]	Images	1.22[b]
2		I2		
3		I3		
4		I4		
MIPS 24	Apex_1frame	apex_1frame_M1.nl	Mosaic	2.45
70		M2		4.0
160		M3		8.0
IRS PUI Blue	Apex_1frame	apex_1frame_PUI_blue.nl	Mosaic	1.8
Red		red		

Instrument	PRF	Samp[c]	Norm Radius (PRF pixels)[d]	Divide PRF Fit Fluxes
IRAC 1	apex_sh_IRAC1_col129_row129_x100.fits[e]	100x	1000	1.021
2				1.012
3				1.022
4				1.014
MIPS 24	mips24_prf_mosaic_2.45_4x.fits[f]	4x	57	0.94
70	mips70_prf_mosaic_4.0_4x.fits		None	0.83
160	mips160_prf_mosaic_8.0_4x.fits		None	0.83
IRS PUI Blue	b0_prf_blue_new.fits	4x	48	0.91
Red	red		52	0.89

[a]Presumes mosaic has already been made in the output directory.
[b]Approximate pixel size on images; mosaic pixel size is 0.6 arcsec.
[c]Factor by which the PRF samples the data pixel size.
[d]PRF normalization radius (input in Source Estimate block).
[e]PRFs derived from cryogenic mission data.
[f]Valid for most 24 micron data. Some fast scan maps may differ – a PRF derived from the data may be preferable.

Appendix D. Some Tips for MOPEX with Spitzer Data: Version 1 (3/16/2012)

D.1 Mosaics (IRAC and MIPS)

The end result of a normal Mosaic run is mosaic files in output_dir/Combine-mosaic (Combine in command-line). Generally no weighting of the pixel stack is used, and especially no sigma weighting if sources are present. You can combine data with different exposure times and weight them by time if the FITS keyword that has the time in it is provided. For Spitzer data, this is 'EXPTIME'. It should be set in the GUI Mosaic Coadd Settings. (In command-line, put INTEG_TIME_KWD = 'EXPTIME', in the Coadd block of the namelist.nl file.)

The coverage mosaic is always the number of good pixels in the final stack. The total exposure time for a pixel (when time-weighting) is not available as a mosaic, but can be found per tile in the output_dir/Coadd-mosaic directory. (If there's only one tile, it is like the final mosaic.)

D.2 Fluxes and Uncertainties (IRAC and MIPS)

The end result of a normal point-source fitting run (Apex or Apex 1frame) is a table in the Output directory called *extract.tbl. It's in IPAC table format, readable by many common tools. A tool for reading an IPAC table into an IDL structure is available at http://irsa.ipac.caltech.edu/tools/irsa_idl.html.

The point-source fitted fluxes (in microJy for Spitzer data) are in the "flux" column. For the uncertainty, use the signal-to-noise column called "SNR", i.e. uncertainty = flux/SNR (for a quick discussion, see the file in the Mopex directory readme/README_new_snr_option_081108.pdf). The aperture fluxes and uncertainties (in microJy) have obvious names in *extract.tbl. (If you ran only aperture photometry, they are in aperture.tbl.)

There are some small correction factors needed for the highest accuracy. Aperture corrections for apertures other than the standard calibration aperture and annnulus can be found in the IRAC and MIPS Data Handbboks. Some corrections for PSF-fitted fluxes can be found in Appendix C.

IRAC array location-dependent corrections with MOPEX are discussed in Spitzer Data Analysis Cookbook: Recipes 6 and 7. Color corrections are in the IRAC and MIPS Data Handbooks.

D.3 Point-Source Fitting: Apex for IRAC stacks, Apex_1frame for MIPS mosaic

The point source response functions and sample namelists that come with MOPEX are for fitting on the (C)BCD frames for IRAC (Apex) and on the default mosaics for MIPS (Apex 1frame).

D.4 Use the Mosaic Task to make Mosaics (IRAC data)

Sounds obvious, but the confusion comes when running APEX (multiframe) on IRAC data. APEX is set up to make a mosaic from the image stack for point-source detection (and does so in the default templates and namelists, e.g. apex_I1_gui.nl). This is because APEX needs some files that are part of the mosaic process. But it can't do all the fancy outlier rejection that Mosaic can. For most data, it's better to make mosaics with the Mosaic task.

One approach might be called "all at once". Set up a GUI Mosaic flow, then click on "Insert APEX multiframe..." and use one of the standard templates. Shut off (hit the X) APEX mosaic commands: Fiducial Image Frame, Mosaic Interpolate, Mosaic Coadd, and Mosaic Combine. This is described in the Spitzer Data Analysis Cookbook: Recipe 7. The GUI version of APEX will now be able to find the Mosaic files it needs. Start the combined flow with the green arrow at the top.

What if you made a nice mosaic last week and don't want to re-do it? Provided you saved the original output directory, you can just run APEX. Set the APEX output directory to the original output directory. You can use a standard template and turn off the APEX mosaic commands as above, or, for your convenience, there are example namelist files that do this for you, cdf/apex*have_mosaic*gui.nl.

(In command-line, run mosaic.pl as usual, then drop the mosaicing tasks from your Apex namelist and run apex.pl to the same output directory used for mosaic.pl, or use one of the cdf/apex*have_mosaic*.nl namelists.)