Background Matching

By David Makovoz
10/15/04

Overview

Background matching is one of the tasks performed by the package MOPEX. Background matching is a two step process. First, the images are interpolated to a common grid. After that, the actual matching is performed. The cumulative pixel-by-pixel difference between the overlapping areas of all pairs of images is minimized with respect to unknown constant offsets of the input images. Since interpolation is a linear process the constant offset is conserved, i.e. the offset found for the interpolated images is applicable directly to the input images. 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.
Perl script overlap.pl performs the above tasks by setting the necessary infrastructure and calling the data processing modules.

Input Data

The script *overlap.pl* requires a set of input images and a namelist (configuration) file. It can optionally use uncertainty images and mask images. The purpose of the latter is to mark pixels in the input images unsuited for processing. Table 1 lists the names of the input files for *overlap.pl*. These names can be set in the namelist file, or on the command line; some of them have a default value. Except for the namelist, the names of the input files can be specified using a relative or the absolute path. The command line settings override the namelist settings.

Input File	Default Name	Namelist name	Command line option	Required
Namelist *	overlap.nl	N/A	-n	y

List of input images	image_stack.txt	`IMAGE_STACK_FILE_NAME`	-I	y
List of input uncertainty images	-	`SIGMALIST_FILE_NAME`	-S	n
Permanently damaged pixels mask image	-	`PMASK_FILE_NAME`	-M	n
List of DCE status mask images	-	`DCE_STATUS_MASK_LIST`	-d	n
Fiducial frame table	-	`FIF_FILE_NAME`	-F	n

Table 1 Input data for *overlap.pl*. *- namelist needs to be in `cdf/` sudirectory.

Namelist – Configuration file

The namelist contains several blocks of various parameter settings, input image names, and running options. Most of the parameter settings for the modules are in the corresponding blocks delineated by “&” following by the capitalized module name in the beginning and “&END” at the end of the block. Several parameters affecting more than one module are set outside of the individual modules’ blocks. Also, the locations of the output final and intermediate products are set in the namelist file.
Table 2 lists the names of the modules, along with their purpose and namelist triggers. To run a module, its trigger should be set to 1.

Module	Namelist trigger	Purpose
snestimator	`compute_uncertainties_internally*`	Compute uncertainties using the model
medfitler	`run_medfilter`	Background subtraction of the input images
detect	`run_detect`	Produce detection maps of bright objects
fiducial_image_frame	`run_fiducial_image_frame`	Compute Fiducial Image Frame (FIF)
mosaic_int	`run_mosaic_int`	Interpolate input images to the FIF
compute_overlap_corr	`compute_overlap_correction`	Computes the constant offsets to match the background
compute_overlap_corr	`apply_overlap_correction`	Adds the computed offsets to the input images

Table 2 Modules, their namelist triggers, and purpose. * If `have_uncertainties` is set to 1, then the user is expected to provide the uncertainty images name. This overrides the `compute_uncertainties_internally` option.

The intermediate and final products are written in several subdirectories. The names of the subdirectories can be configured in the namelist file. Table 3 lists the default names of the output subdirectories, the keywords used in the namelist, and all the products written in the subdirectory. `OUTPUT_DIR` can be specified as a relative or absolute path.

Subdirectory set in Namelist	Default name	Output files
`OUTPUT_DIR`	`./`	FIF.tbl
`SIGMA_DIR`	`Sigma`	Uncertainty images, if computed by snestimator
`MEDFILTER_DIR`	`Medfilter`	Background subtracted input images
`DETECT_DIR`	`Detect`	Detection map images
`INTERP_DIR`	`Interp`	Interpolated images, interpolated uncertainty images, and corresponding coverage maps
`OVERLAP_CORR_DIR`	`Overlap_Corr`	Corrected input images, the table with the corrections offset.tbl

Table 3 Output directories and the products written there.

The subdirectories are created by *overlap.pl.*
Table 4 lists all the parameters, along with their default values, if any, a short description, and the name of the module(s) using this parameter.
The input variables require a space preceding and following the equal sign, i.e. "variable = value". An entry like "variable=value" will not be read and the hard-coded default will be used so you will not notice that your input value is not being used.

Parameter Name	Description	Default	Module
`USE_REFINED_POINTING`	Switch to use refined pointing keywords (int)	0	*mosaic_int*
`MOSAIC_PIXEL_RATIO_X` `MOSAIC_PIXEL_RATIO_Y`	The ratio of the input pixel *x*-(*y*-) size to the mosaic pixel *x*-(*y*-) size. (float)	1	*mosaic_int*
`Gain`	Gain in e^{-1}/[image units] (float)	-	*snestimator*
`Confusion_Sigma`	Confusion noise in e^{-1} (float)	-	*snestimator*
`Read_Noise`	Read Noise in e^{-1} (float)	-	*snestimator*
`Edge_Padding`	The number of pixels the FIF is padded with on all four	0	*fiducial_image_frame*

	sides (int)		
CROTA2	If set, this is the orientation of the FIF. If not set the program will compute the optimal orientation (float or char 'A')	-	*fiducial_image_frame*
Window_X Window_Y	*x*-size and *y*-size in pixels of the sliding window placed around each pixel to estimate the median (int)	-	*medfilter*
N_Outliers_Per_Window	Number of pixels excluded from the sliding window described above (int)	0	*medfilter*
Detection_Max_Area	The maximum number of pixels in a cluster (int)	9	*detect*
Detection_Min_Area	The minimum number of pixels in a cluster (int)	0	*detect*
Detection_Threshold	Number of of sigmas above the mean (float)	-	*detect*
INTERP_METHOD	Options are 1, 2, 3; the default is 1 (int)	1	*mosaic_int*
DRIZ_FAC	Drizzle factor. Use for `INTERP_METHOD` = 2 (float)	1	*mosaic_int*
GRID_RATIO	The number of grid points per pixel in one direction. Used for `INTERP_METHOD` = 3 (int)	-	*mosaic_int*
TOP_THRESHOLD	Number of sigma's above the mean to detect outliers among the computed offsets(float)	-	*compute_overlap_corr*
BOTTOM_THRESHOLD	Number of sigma's below the mean to detect outliers among the computed offsets(float)	-	*compute_overlap_corr*
MIN_IMG_NUM	Minimum number of overlapping images to perform outlier rejection (int)	4	*compute_overlap_corr*

Table 4 The processing parameters for overlap.*pl*. The shaded fields are for the parameters set outside of the individual module blocks.

Use of Uncertainty Images

If the `have_uncertainties` switch is set, then, regardless of the setting of the `compute_uncertainties_internally` switch, the script expects to find a list of

uncertainty images. The uncertainty images are interpolated and co-added. The co-added uncertainty images are used in the co-addition of interpolated input images.
If `compute_uncertainties_internally` is set and `have_uncertainties` is not set, then the *snestimator* module is executed. It produces uncertainty images, which are subsequently interpolated, and used for co-addition.
Uncertainty images are required for overlap.pl, therefore either `compute_uncertainties_internally` or `have_uncertainties` should be set.
Module *snestimator* estimates pixel uncertainty for each pixel in the image using the following model:

$$s = \sqrt{\frac{s^2_{readnoise}}{g^2} + \frac{s^2_{confusion}}{g^2} + \frac{I}{g}}.$$

Here the parameters of the model `Read_Noise` $s_{readnoise}$, `Gain` g, and `Confusion_Sigma` $s_{confusion}$ are specified in the namelist. The last term is the Poisson noise determined by the pixel value I.
The units of `Read_Noise` and `Confusion_Sigma` here are electrons. The product of this step is the uncertainty images.

Quality Control Mask Images

Quality control mask images can be used in the processing. Two kinds of mask images can be used: permamently damaged pixels masks (Pmask) and DCE status masks (Dmask). Normally there will be a single Pmask for a set of input images.
Each bit of the pixel value in a mask image corresponds to a particular condition. A fatal bit pattern is a short integer that has the bits of interest set. Each pixel in a mask image is matched against the fatal bit patter. If any of the bits specified by the fatal bit pattern is set in the value of a pixel in a mask image, then in the corresponding image the corresponding pixel is considered unusable.
If the name of the mask image or list of mask images is not set the scripts will proceed without using them. If they are set then the corresponding fatal bit pattern should be specified in the namelist.

Mask Name	Fatal Bit Pattern Name
Pmask	`Pmask_Fatal_BitPattern`
Dmask	`DCE_Status_Mask_Fatal_BitPattern`

Table 5: Setting of fatal bit patterns for various mask images in the namelist file.

Other Options

1. Switch `mask_bright` should be set to used bright object mask images for interpolation.

2. Switch `NICE.` If `NICE` =1 all the modules called by the script with "nice 19." The default is 0.
3. Switch `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 in the namelist the namelist copied to output directory will be given a unique name, which is created by appending the namelist name to the time of execution. For example, if you ran "`mosaic.pl -n myname.nl`" at 12:32:53, then the namelist will be copied to the output directory as `12h32m53s_myname.nl`. The default is 0, in which case the file is copied as `myname.nl`.
4. Switch `delete_intermediate_files`. If `delete_intermediate_files` = 1 is set in the namelist the products of all the modules run this time will be deleted except for the last module. The default is 0.
5. Switch `mosaic_corrected_images`. If it is set to 1, then a mosaic of corrected images will be created as OUTPUT_DIR/mosaic.fits. Since the interpolated images have already been made, it is only a matter of applying the computed offsets to the interpolated images and coadding them into a mosaic image. This is a quick-and-dirty way to examine the results of overlap correction. You have to have the namelist block for *mosaic_coadd* in the namelist in order to run this option. See Spitzer_mosaicker for more details on mosacking images.

Bright Object Masking (optional)

Optionally, 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. Module medfilter produces background subtracted images. Module detect 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 are set to positive values. See Spitzer_apex.doc and Image_Segmentation.doc for more information on these modules. The mask images are saved (if switch `delete_intermediate_files` is not set) in `DETECT_DIR` subdirectory (default name "Detect"). If switch `mask_bright` is set in the namelists, these mask images are used in the further processing.

Fiducial Image Frame (FIF) Computation

This step is optional. If a table with the fiducial image frame had been created previously it can be used by supplying its name in the namelist file using `FIF_FILE_NAME` keyword.
This step creates a unified grid coordinate system that is used for creating a mosaic image. Given a list of input images, the perl script *fiducial_image.pl* creates a list of pointing parameters - `CRVAL1, CRVAL2, CRPIX1, CRPIX2, CROTA2, CDELT1, CDELT2, NAXIS1, NAXIS`. Run subsequently, module *fiducial_image_frame* generates the pointing parameters for bounding region of a minimal size that encloses the input images. Even if the input images use the CD matrix convention, the mosaic image (and by extension the interpolated images) will use the set of keywords `CDELT1`, `CDELT2`, and `CROTA2`, since the mosaic image is undistorted.

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

Interpolation

See document SIRTF_Mosaicer for the description of the interpolation step. Here we only mention that since the high spatial frequency information in the input images is not important for background matching it is suggested to use undersampled interpolated images (`MOSAIC_PIXEL_RATIO_X(Y) < 0`). It will also have the added benefits of faster processing and reduced memory usage. The module has been tested with simulated data with `MOSAIC_PIXEL_RATIO_X(Y)` down to 0.25 without any significant loss in the precision of the computed offsets. Also interpolated scheme 3 (Grid) with `GRID_RATIO ~ 2 * MOSAIC_PIXEL_RATIO_X(Y)` can be used to speed up the interpolation step.

Background Matching Algorithm

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 e^n, i.e. that the following corrections should be applied to the input image I^n

$$O^n(x,y) = I^n(x,y) - e^n$$

Equation 1

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:

$$\mathbf{L} = \sum_{m,n=1(m\neq n)}^{N_{images}} \sum_{k\in overlap} \frac{(I^n(k^n) - I^m(k^m))^2}{s_n^2(k^n) + s_m^2(k^m)}$$

Here m and n are image indices, k^n is the pixel number in image n.
Minimization with respect to e^n's

$$\frac{\partial \mathbf{L}}{\partial e^n} = 0$$

leads to the following set of N_{images} *-1* linear equations:

$$\sum_{m=1}^{N_{images}-1} M^{nm} e^m = z^n,$$

where

$$M^{nm} = -\frac{O^{nm}}{s_n^2 + s_m^2}, n \neq m; \quad M^{nn} = \sum_{r(r\neq n)} \frac{O^{nr}}{s_n^2 + s_r^2};$$

$$z^n = \sum_{r(r\neq n)} \frac{O^{nr}\left(I^n - I^r\right)}{s_n^2 + s_r^2}.$$

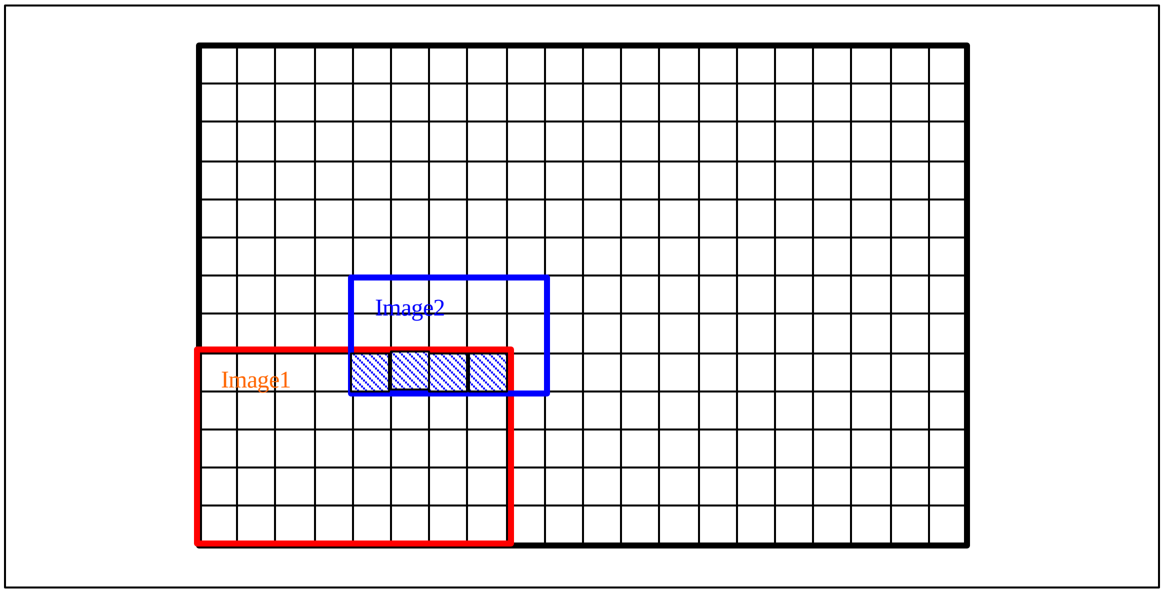

Figure 1 The overlap of Image1 and Image2 is an area of four pixels with the following pixel indices: k_1= {36,37,38,39}; k_2 = {0,1,2,3}, given that the first pixel has index = 0 and the x-direction is scanned first.

The symbol O^{mn} represents the fact that the summation is done over the overlap area of the *m*-th and *n*-th images. In Figure 1 the case of 2 overlapping images is shown. The matrix element M^{12} is in this case equal to

$$M^{12} = -\frac{O^{12}}{s_1^2 + s_2^2} = -\left(\frac{1}{s_1^2(36) + s_2^2(0)} + \frac{1}{s_1^2(37) + s_2^2(1)} + \frac{1}{s_1^2(38) + s_2^2(2)} + \frac{1}{s_1^2(39) + s_2^2(3)} \right).$$

Equation 2

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 *d* of all images, one of the shifts will be left undetermined. Only N_{images} *-1* equations are solved. The undetermined shift can be picked to be the last one $e^{Nimages}$. It is set to 0 first in order to solve the system **Error! Reference source not found.**. The additional constraint that will fix the global shift is to have the total shift all images to add up to 0:

$$\sum_{m=1}^{N_{images}} (e^m - d) = 0,$$

$$d = \frac{1}{N_{images}} \sum_{m=1}^{N_{images}-1} e^m;$$

$$e^m = e^m - d; \text{for } m = 1, N_{images}.$$

Equation 3

The *e*'s are analyzed before applying the above condition. The outliers among the *e*'s are found. The following namelist parameters are used: `BOTTOM_THRESHOLD`, `TOP_THRESHOLD`, the thresholds for outlier detection in terms of sigma, `MIN_IMG_NUM` - minimum number of images required to detect outliers.

If N_{images} >= `MIN_IMG_NUM` equations modify N_{images} to exclude the outlier e's from calculating d but apply d to all the e's.

Output

The offsets e^n's and their uncertainties are written in the header of the corresponding fits files. The keywords are `OVRLPDC` and `OVRLPDCD`. Also a table offset.tbl is created which contains the offsets and their uncertainties for all the files. Column `outliers` is used to indicate whether the offset e was determined to be an outlier. The corresponding fits header keyword is `OVRLPOUT`. 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
```

References

All of the documents referenced in this document can be obtained from the Spitzer Science Center website, http://spitzer.caltech.edu/SSC/. They include:

1. Spitzer _mosaicer.
2. Spitzer_apex.
3. Image_Segmentation.