674-SO-43, Version 1.1
SSC-PD-4087

SIRTF Science Center

Downlink Segment

Subsystem Design Specification

AOT Products Subsystem: FFTREGIST

28 June 2005

California Institute of Technology
SIRTF Science Center

National Aeronautics and
Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

THIS PAGE IS INTENTIONALLY LEFT BLANK

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

SIRTF Science Center

Subsystem Design Specification

Prepared by:

Frank Masci

Concurred by:

Dave Shupe

Approved by:

Russ Laher

Concurred by:

Suzy Dodd

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

1 Revision History

Version	Description	Date
1.0	Initial version written which updates RA, Dec in target image after registering with reference image.	July 25, 2003
1.1	Modified to simply write orthogonal X, Y offsets between target and reference images to specified output IPAC table file.	June 28, 2005

2 Table of Contents

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

List of Figures

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

List of Tables

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

1. Introduction

1.1. Purpose and Scope

The Subsystem Design Specification is a document that describes the basic requirements, assumptions, definitions, software-design details and necessary interfaces for each subsystem. The document will be used to trace the incremental development of each subsystem and also to allow trace-back of levied requirements; this document should have sufficient detail to allow future modification or maintenance of the software by developers other than the original developers. This document is an evolving document as changes may occur in the course of science instrument hardware design and maturity of operational procedures. This document is not intended to repeat sections or chapters from other Project documents; when appropriate, references to proper sections of primary reference documents will be made.

1.2. Document Organization

This document is organized along the major themes of Requirements; Assumptions; Operational Concept; Functional Descriptions; Functional Dependencies; Input; Output; Other S/S Interfaces; Algorithm Descriptions (when applicable); and Major Liens.

The material contained in this document represent the current understanding of the capabilities of the major SIRTF systems. Areas that require further analysis are noted by TBD (To Be Determined) or TBR (To Be Resolved). TBD indicates missing data that are not yet available. TBR indicates preliminary data that are not firmly established and are subject to change.

1.3. Relationship to Other Documents

The requirements on the operation of SIRTF flow down from the Science Requirements Document (674-SN-100) and the Facility Requirements Document (674-FE-100). The Science Operations System is governed by the SOS Requirements Document (674-SO-100). The current document is also cognizant of the requirements that appear in the Observatory Performance and Interface Control Document (674-SEIT-100) as well as the Flight Ground Interface Control Document (674-FE-101). This document is also affected by the FOS/SOS Interface Control Document (674-FE-102) that governs interfaces between the Flight Operations System and the Science Operations System. Related Software Interface Specifications (SIS) will be as indicated in Section 2.2 of this document.

1.4. Change Procedure

This document is a level 4 document according to the SIRTF Project Documentation Plan (674-FE-103). Changes to this document after approval require the approval of the SOS Change Board (TBD). The process for change control is described in the SOS Configuration Management Plan.

2. Overview

FFTREGIST reads in data from two FITS images (a reference and target image), a set of processing parameters and registers them using a phase-correlation, cross-power-spectrum method. Power spectra are computed via use of Fast Fourier Transforms (FFTs). The main output is a table of orthogonal X and Y translational offsets in pixel units of the input reference image. The program allows optional windowing, thresholding and filtering of the input images to attain optimal registration. FFTREGIST is written in standard ANSI/ISO C.

2.1. FFTREGIST Requirements

FFTREGIST is initiated by a startup script under the control of the pipeline executive and does its required functions for a given DCE image or pre-processed DCE image; this involves performing the following tasks.

A.) Retrieve the command line parameters passed by the start up script and use them to run the program.

B.) Read in as input two standard FITS images (a reference and target image) and processing parameters.

C.) Produce as primary output, a table in IPAC format containing orthogonal translational offsets.

D.) Provide exit codes to the pipeline executive and also provides logon and logoff messages identifying the version number and write any error messages to the standard output devices.

E.) Produce a processing summary.

2.2. Applicable Documents

The following documents are relevant to the FFTREGIST program of the AOT PRODUCTS Subsystems.

A.) The SOS Requirements Document

B.) The SOS Downlink Requirements Document

C.) The SOS Downlink Software Development Guidelines

2.3. Version History

2.3.1. Version 1.0

Initial version created on July 25, 2003. This initial version updated the RA, Dec in the target image after registering with reference image.

2.3.2. Version 1.1

Created on June 28, 2005. The software was modified to simply compute and write orthogonal X, Y offsets between target and reference images to a specified output table file.

2.4. Liens

No liens are identified on the current version, however as a further enhancement, the software could be modified to also compute rotational offsets between images. This will be a non-trivial undertaking.

3. Input

3.1. FFTREGIST Input

FFTREGIST takes all of its input from either the command line or namelist file, which is set up by the startup script that is controlled by the pipeline executive or standalone. If the namelist is not specified, then all required inputs are expected from the command line. If both namelist and command-line inputs are specified, then the command-line inputs override the namelist values. Prior to reading namelist and/or command-line parameters, default values for the relevant parameters are assigned.

3.1.1. FFTREGIST NAMELIST Input

FFTREGIST reads the NAMELIST file whose name is passed to it by start-up script. The name of the NAMELIST "blockname" is POINTINGEFIN. The parameters that can be defined in the NAMELIST are listed in Table 1.

Namelist variable	**Description**	**Dim.**	**Type**	**Units**	**Default**
Input_Reference_FITS_Image	Required input reference FITS-image filename	256	C	-	Null
Input_Test_FITS_Image	Required input test (or target) FITS image filename	256	C	-	Null
Output_Shifts_Filename	Required output filename of table containing results	256	C	-	Null
MIN_OVERLAP_FRAC	Optional minimum overlap fraction between reference and input test image below which they cannot be registered	1	R*4	-	0.5
FILTERFLAG	Filtering options for input images: 0 = no filtering, 1 = low-pass filtering, 2 = high pass filtering	1	I*2	-	0
FILTERITERNUM1	Optional number of recursive filterings to apply to reference image	1	I*2	-	1
FILTERITERNUM2	Optional number of recursive filterings to apply to test image	1	I*2	-	1

SIGMAFILTER1	Sigma for high-pass Gaussian filter applied to reference image in pixel units	1	R*4	pixels	1.0
SIGMAFILTER2	Sigma for high-pass Gaussian filter applied to test image in pixel units	1	R*4	pixels	1.0
BASEMASKWIDTH1	Width of filter mask applied to reference image as a multiple of SIGMAFILTER1 above	1	R*4	sigma1	1.0
BASEMASKWIDTH2	Width of filter mask applied to test image as a multiple of SIGMAFILTER2 above	1	R*4	sigma2	1.0
CROPNUMBORDERPIX	Optional number of border pixels to crop due to possible filtering artifacts	1	I*2	-	0
WINDOWFLAG	Windowing options for input image data to reduce high frequency leakage from sharp edges: 0 = no windowing; 1 = Hamming window; 2 ="Masci" window. See parameters below	1	I*2	-	0
MASCIINDEX	Index for the "Masci" window option (WINDOWFLAG=2) above. Must be even number	1	I*2	-	6
LOWFLUXCLIPFLAG	Flag to retain all pixels in both images which are greater than some SNR*sigma with remainder set to zero. This is performed on windowed filtered images and sigma is computed internally: 0 = no low-flux clipping; 1 = perform low-flux clipping	1	I*2	-	0
SNRTHRESHOLD1	Signal-to-noise ratio below which all pixel values are set to zero in reference image. (Only if LOWFLUXCLIPFLAG = 1)	1	R*4	-	5.0

SNRTHRESHOLD2	Signal-to-noise ratio below which all pixel values are set to zero in test image. (Only if LOWFLUXCLIPFLAG = 1)	1	R*4	-	5.0
NPEAKSEARCH	Optional number of recursive searches for peak on correlation surface. Search down to Nth highest peak until the corresponding X,Y offsets there-from lie within interval SEARCHRADIALTOL (see below) of the expected offsets inferred from the raw-pointing	1	I*2	-	3
SEARCHRADIALTOL	Optional radial tolerance between expected offsets (from image pointings) and those corresponding to peak on correlation surface	1	R*4	arcsec	4.0
Log_Filename	Optional output log filename	256	C	-	stdout
Ancillary_File_Path	Optional pathname where supporting source files are installed.	256	C	-	./

Table 1. Namelist Parameters

The following is an example of the contents of a “POINTINGEFIN” NAMELIST file that might be used, where the values specified are not necessarily realistic.

```
&POINTINGREFIN
 # Generic namelist file for fftregist,
 Ancillary_File_Path =           '../fftregist_v1',
 Log_Filename =                  'stdout',

 # Input/Output files (all required),
 Input_Reference_FITS_Image =    './testing/bcd_0138.fits',
 Input_Test_FITS_Image =         './testing/bcd_0141.fits',
 Output_Shifts_Filename =        './testing/Cartesianshifts.txt',

 # Min overlap fraction between Reference and input Test image,
 # below which they cannot be correlated (Optional, default = 0.5),
 MIN_OVERLAP_FRAC =              0.5,

 # Filtering options for input images: 0 => no filtering,
```

```
# 1 => low-pass filtering, 2 => high-pass filtering (see parameters,
# below; Optional, default = 0),
FILTERFLAG =                    2,

# Number of recursive filterings to apply to Reference image,
# (Optional, default = 1),
FILTERITERNUM1 =                1,

# Number of recursive filterings to apply to Test image,
# (Optional, default = 1),
FILTERITERNUM2 =                1,

# Sigma for high-pass Gaussian filter applied to Reference image in,
# pixel units (Optional, default = 1.0),
SIGMAFILTER1 =                  1.0,

# Sigma for high-pass Gaussian filter applied to Test image in,
# pixel units (Optional, default = 1.0),
SIGMAFILTER2 =                  1.0,

# Width of filter mask applied to Reference image as a multiple of,
# SIGMAFILTER1 above (Optional, default = 1.0),
BASEMASKWIDTH1 =                5.0,

# Width of filter mask applied to Test image as a multiple of,
# SIGMAFILTER1 above (Optional, default = 1.0),
BASEMASKWIDTH2 =                5.0,

# Number of border pixels to crop due to possible filtering artifacts,
# (Optional, default = 0),
CROPNUMBORDERPIX =              2,

# Windowing options for input image data to reduce high frequency,
# leakage from sharp edges: 0 => no windowing, 1 => Hamming window,
# 2 => "Masci" window (see parameters below; Optional, default = 0),
WINDOWFLAG =                    2,

# Index for the "Masci" window option (WINDOWFLAG=2) above. Must be,
# an even number (Optional, default = 6),
MASCIINDEX =                    6,

# Flag to retain all pixels in both images which are greater,
# than some SNR*sigma with remainder set to zero. This is,
# performed on windowed, filtered images and sigma is computed,
# internally. 0 => no low-flux clipping, 1 => perform low-flux clipping,
# (Optional, default = 0),
LOWFLUXCLIPFLAG =               0,

# Signal-to-noise ratio below which all pixel values are set to zero,
# in Reference image. (Only if LOWFLUXCLIPFLAG = 1; Optional, default = 5.0),
SNRTHRESHOLD1 =                 30.0,

# Signal-to-noise ratio below which all pixel values are set to zero,
# in Test image. (Only if LOWFLUXCLIPFLAG = 1; Optional, default = 5.0),
SNRTHRESHOLD2 =                 30.0,

# Number of recursive searches for peak on correlation surface. Search,
# down to Nth highest peak until the corresponding X,Y offsets therefrom,
# lie within an interval SEARCHRADIALTOL (see below) of the expected,
# offsets inferred from the raw-pointing WCS. (Optional, default = 3),
NPEAKSEARCH =                   10,

# Radial tolerance in arcsec +/- between expected offsets,
```

```
  # (from image pointings) and those corresponding to peak on,
  # correlation surface. (Optional, default = 4.0 arcsec),
  SEARCHRADIALTOL =               6.0,
 &END
```

3.1.2. FFTREGIST Command-Line Input

Alternatively, all inputs can be specified via command line, in which case, a namelist file is not needed. Or, inputs can be provided with a hybrid of both namelist and command-line mechanisms, with the latter over-riding the former. Table 2 lists the available command-line options associated with their namelist-variable counterparts, as well as other options for specifying the namelist-file name and making the standard output more verbose.

3.1.3. FFTREGIST FITS Input

FFTREGIST uses the FITSIO library routines to read in the FITS-formatted input data file. The routines used are: fits_open_file, fits_read_keys_lng, fits_read_keys_dbl, fits_read_img, and fits_close_file.

Command-line option	**Variable**
-n	Namelist_Filename
-f1	Input_Ref_FITS_Image
-f2	Input_Test_FITS_Image
-o	Output_Shifts_Filename
-m	MIN_OVERLAP_FRAC
-g	FILTERFLAG
-g1	FILTERITERNUM1
-g2	FILTERITERNUM2
-s1	SIGMAFILTER1

-s2	SIGMAFILTER2
-b1	BASEMASKWIDTH1
-b2	BASEMASKWIDTH2
-c	CROPNUMBORDERPIX
-w	WINDOWFLAG
-i	MASCIINDEX
-h	LOWFLUXCLIPFLAG
-q1	SNRTHRESHOLD1
-q2	SNRTHRESHOLD2
-p	NPEAKSEARCH
-r	SEARCHRADIALTOL
-d	Debug_Switch
-l	Log_Filename
-a	Ancillary_File_Path
-v (verbose switch)	-
-vv (super-verbose switch)	-

Table 2. Command-line options

4. Processing

4.1. FFTREGIST Processing

FFTREGIST begins processing by writing its name and version number to standard output (verbose mode only), and then it initializes relevant variables with defaults values, and checks that the required namelist parameters and/or command-line parameters were passed to it. If this condition is not true, it writes a message stating which parameters are missing, recommends a look at this document, and terminates by issuing an appropriate exit code to the pipeline executive; otherwise it proceeds as follows.

If an error occurs during processing, then an error message is written to standard output, a termination-status code is written to the log file, and an exit code to the pipeline executive issued.

After processing, the program name and version number, namelist filename (if used), input, and output filenames, values of other input parameters, date and time, processing time, and a termination-status code are written to a log file.

4.2 FFTREGIST Processing Phases

FFTREGIST operates in thirteen phases: initialization, FITS image and keyword data input, input image overlap consistency check, computation of expected offsets from raw image pointings, image windowing, creation of 2D filters and application to input images, image pixel thresholding, Fourier transforming of processed image inputs, cross-power spectrum computation in frequency space, real-space 2D correlation function computation, X-Y offset computation from peak on correlation surface, refinement of X-Y offsets using quadratic interpolation, results output, and termination. This processing level is depicted in Figure 1.

4.2.1. FFTREGIST Initialization

FFTREGIST initializes itself by performing the following tasks.

A.) A message is printed to STDOUT (verbose mode only), which includes the program name and version number.

B.) If specified on the command line, the NAMELIST file is opened and read. If any errors are encountered, a message is printed, and execution aborts.

C.) The remaining command-line inputs are read, checked for correct data range and consistency and missing (optional) inputs replaced with hard-coded defaults. If any errors are encountered, a message is printed, and execution aborts.

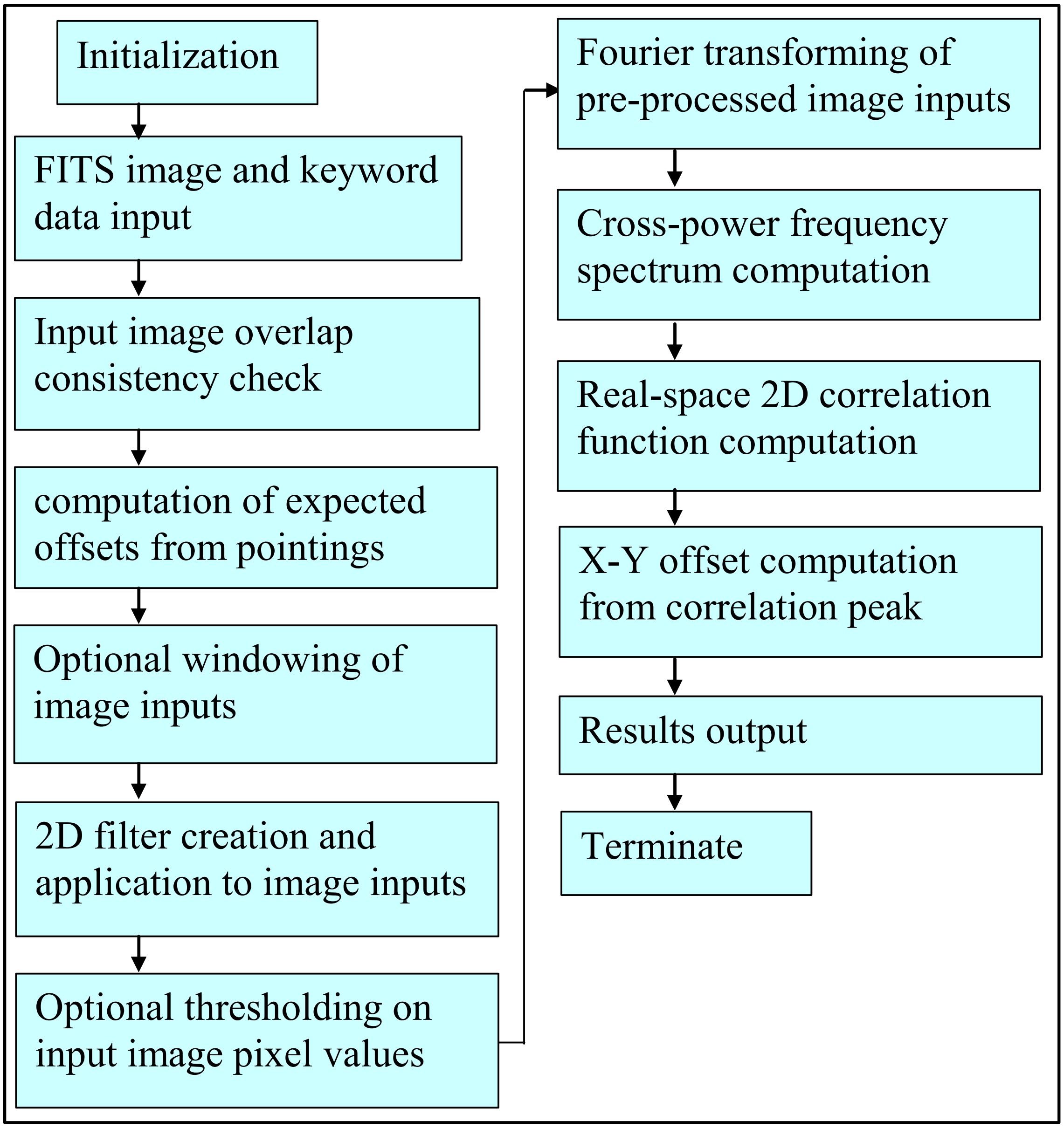


Figure 1. FFTREGIST data and processing flow

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.

4.2.2. FITS Image and Keyword Data Input

Image pixel data from input reference and test (target) images are read and stored in memory. The fits_read_img() CFITSIO subroutine is used. The memory used to store the images is computed from the image dimensions, NAXIS1 and NAXIS2 keywords, using the fits_read_keys_lng(). WCS-related keywords for estimating initial image offsets from the raw pointing are read using the standard WCS library routine fitsrhead().

4.2.3. Image-Overlap Consistency Check

Due to the periodic nature of the images in frequency space after Fourier transforming, the FFTREGIST software is only capable of computing offsets along X and Y for images whose a-priori offsets are known to be smaller than (NAXIS1)/2 and (NAXIS2)/2 respectively, i.e., no larger than half the image size in each dimension. The user has can optionally specify the input minimum overlap fraction (namelist parameter: MIN_OVERLAP_FRAC) to ensure this requirement is met (default is 0.5). This parameter specifies the minimum overlap fraction below which the program cannot register the two input images and will abort.

4.2.4. Estimation of Expected Image Offsets

To assist in searching for the peak on the correlation surface after inverse Fourier-transforming the cross-power spectrum (i.e., the location of which gives the orthogonal image offsets, see below), approximate image offsets are used to isolate the appropriate region on the correlation surface to refine the true peak location. These approximate offsets are estimated from the relative WCS pointings of the input images. The RA, Dec of the rotational center of the test image is transformed to the x,y frame of the reference image and Cartesian offsets are then computed therefrom in pixel units of the reference image. Note that the input test and reference images are assumed to be co-linear (i.e., no rotational offset exists).

4.2.5. Image Windowing

The input reference and test images can be optionally “windowed” to reduce high frequency leakage from the sharp (discontinuous) edges. Three options are available as specified by the WINDOWFLAG input variable: 0 = no windowing; 1 = Hamming window; 2 ="Masci" window.

The Hamming window (also known as a “raised cosine”) has the functional form:

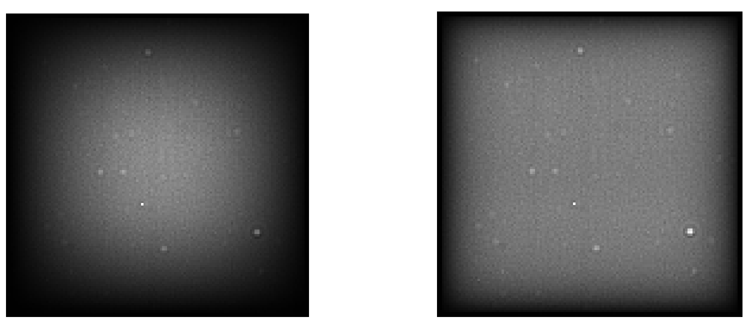
$$W(x,y) = \cos\left[\pi\left(\frac{x}{N_x} - 0.5\right)\right]\cos\left[\pi\left(\frac{y}{N_y} - 0.5\right)\right],$$

where x, y is a pixel coordinate and N_x , N_y are the number of pixels in each dimension of the image. Figure 2 shows an example of an image after it has been multiplied by a Hamming window.

The “Masci” window was designed by Frank Masci (*Spitzer* Science Center) as a means to preserve the geometry of an image. Its purpose is to retain as much image information and structure as possible within the square or rectangular geometry. It has the functional form:

$$W(x,y) = \left[1 - \left(\frac{2x}{N_x} - 1\right)^m\right]\left[1 - \left(\frac{2y}{N_y} - 1\right)^m\right],$$

where m is a user-specifiable parameter (variable MASCIINDEX) and its value must be even. The software checks that it is even and aborts otherwise. The larger the m value, the more abrupt is the falloff in intensity towards the boundaries of an image. An example of a “Masci”-windowed image is shown in Figure 2. It appears that the Hamming window will be optimal for “circular” images although it performs just as well as the Masci window for small image offsets, i.e., where most of the structure to be correlated resides near the image centers.



Hamming-windowed **Masci-windowed**

Figure 2. Hamming and Masci-wondowed image examples

4.2.6. Image Filtering

The input reference and test images can also be optionally *low or high-pass* filtered to accentuate structure at a particular scale (or frequency). With prior knowledge of the characteristic structure in the input images, one has the option of designing a filter to specifically select that structure for use in optimal cross-correlation. Only symmetric Gaussian filters can be made by the software. These have the following functional form in the spatial domain:

$$F(x,y) = N \exp\left[-\frac{1}{2\sigma^2} \left\{ \left(x - \frac{1}{2}[W_x + 1] \right)^2 + \left(y - \frac{1}{2}[W_y + 1] \right)^2 \right\} \right],$$

where *x, y* is a pixel position in the image filter, W_x , W_y are the filter widths in pixels (variables BASEMASKWIDTH1 or BASEMASKWIDTH2 for the reference or test image respectively) and σ is the standard deviation (variables SIGMAFILTER1 or SIGMAFILTER2). Note that W_x , W_y are forced to be *odd-numbered* by the software by rounding up to the nearest odd-number. This is to ensure symmetry about a central (peak) pixel. *N* is a normalization factor which is computed from the constraint that all pixel values in the filter sum to unity. Note that the above filter is also a type of "low-pass" (low-frequency) filter. See below for more details.

Once constructed and stored in memory, the filters are applied to the input images (I_{inp}) in the spatial domain using a discrete convolution:

$$I_{out}(x,y) = \sum_{k=1}^{W_x} \sum_{l=1}^{W_y} I_{inp}\left(x + k - \frac{[W_x + 1]}{2},\ y + l - \frac{[W_y + 1]}{2} \right) F(k,l),$$

where $I_{out}(x,y)$ is the output (filtered) image and $F(k,l)$ the filter image described above. Two types of filtering operations are available as specified by the FILTERFLAG variable: 1 = low-pass filtering; 2 = high pass filtering. Low pass filtering is accomplished by directly applying the Gaussian filter above. High pass filtering is defined by the operation:

$$I_{out}(x,y)_{HP} = I_{inp}(x,y) - I_{out}(x,y),$$

i.e., the input (original) pixel value minus the low-pass filtered value as defined by the above convolution.

The user also has the option of applying recursive filterings to the input images. In other words, by applying the filter *N* times, where *N* is user-specifiable by the FILTERITERNUM1 or

FILTERITERNUM2 parameters (for the reference and test images respectively). The reason for this is to accentuate structure even further on a desired scale.

4.2.7. Pixel Thresholding

Optional pixel thresholding can be applied next where only those pixels which are greater than some SNR*sigma are retained and remainder set to zero. This is performed on windowed filtered images and the sigma is computed internally from the full image standard deviation with *no outlier rejection*. Thresholding is controlled by the user-specifiable parameter: "LOWFLUXCLIPFLAG" (0 = no low-flux clipping; 1 = perform low-flux clipping) and the input SNRs can be specified using the SNRTHRESHOLD1 and SNRTHRESHOLD2 parameters for the reference and test images respectively. The purpose of thresholding is to prevent unwanted low level structure or random noise spikes from being correlated. In other words, it provides another means at accentuating the desired structure to correlate between images.

4.2.8. Forward Fourier Transforms

The forward Fourier transforms of the (optionally) windowed, filtered and thresholded reference and test images are computed using the "Fast Fourier Transforms" (FFT) algorithm. This is performed using the version 3.0.1 FFTW library routines: fftw_plan_dft_r2c_2d() and fftw_execute(). We shall refer to the Fourier-transformed reference and test images as $I_{F1}(\xi, \eta)$ and $I_{F2}(\xi, \eta)$ respectively, where ξ, η are frequency variables corresponding to each axis.

4.2.9. Cross-Power Spectrum Computation

The reason for computing the "cross-power spectrum" between the images in frequency space is because it is related to the translation property of the Fourier transform, sometimes referred to as the Fourier shift theorem. For example, if $I_1(x, y)$ and $I_2(x, y)$ are two images that differ only by orthogonal translations x_0 and y_0, i.e.,

$$I_2(x, y) = I_1(x - x_0, y - y_0),$$

then their corresponding Fourier transforms $I_{F1}(\xi, \eta)$ and $I_{F2}(\xi, \eta)$ will be related by

$$I_{F2}(\xi, \eta) = e^{-j2\pi(\xi x_0 + \eta y_0)} I_{F1}(\xi, \eta).$$

The *normalized* cross-power spectrum between the two images is defined as:

$$\frac{I^*_{F1}(\xi,\eta)I_{F2}(\xi,\eta)}{\left|I_{F1}(\xi,\eta)I_{F2}(\xi,\eta)\right|} \equiv e^{-j2\pi(\xi x_0+\eta y_0)},$$

where $I^*(\xi, \eta)$ is the complex conjugate of $I_{F1}(\xi, \eta)$. Therefore, the Fourier shift theorem guarantees that the phase of the cross-power spectrum is equivalent to the phase difference between the images. By taking the Inverse Fourier Transform (IFT) of the cross-power spectrum in the frequency domain, we will have a function that is theoretically an impulse (a delta function), i.e.,

$$e^{-j2\pi(\xi x_0+\eta y_0)} \xleftrightarrow{IFT} \delta(x-x_0, y-y_0)\,.$$

4.2.10. Real Space 2-D Correlation Function Computation

The inverse of the cross-power spectrum is simply known as the real-space 2D cross-correlation, sometimes called the correlation surface function. The last expression of the previous section says that this inverse should be a nice clean impulse, that is, it is approximately zero everywhere except at the displacement that is needed to optimally register the two images. In reality, it will not be a clean impulse due to the presence of noise in the input images. In fact, its height can in practice be used as a measure for the quality of the match.

4.2.11. X-Y Offsets Computation and Refinement

We now attempt to search for the *correct* peak on the correlation surface function (see previous section) corresponding *to the desired translational shifts* using an iterative procedure. Due to the presence of noisy (correlated) structure, the highest peak in this surface may not be the true peak sought for. Therefore, we make use of the expected translational shifts computed using the raw image pointings (see section 4.2.4) as a sanity check, i.e., to ensure that our selected peak is not too far from where we expect it to be. To do this, we allow the user to specify a parameter "NPEAKSEARCH", which is the number of recursive searches for the *desired* peak on the correlation surface. We search down to *Nth* highest peak until the corresponding x_0, y_0 offsets lie within an interval "SEARCHRADIALTOL" of the expected offsets inferred from the raw-pointings. If after "NPEAKSEARCH" iterations no peak is found within the "SEARCHRADIALTOL" limit, the shifts are explicity set to those computed using the raw pointings and reported as the final result.

The translational image shifts found using the above algorithm are at a resolution of one reference-image pixel at the most. We will desire sub-pixel accuracy for best results. This is accomplished by fitting a quadratic in each of the X and Y axes on the correlation surface using three points which straddle the peak, and then interpolating to the maximum value of the quadratic on each axis independently. More specifically, given quadratics in the X and Y directions about the correlation surface peak:

$$p_x = C_{1x} + C_{2x}x + C_{3x}x^2 \qquad (A)$$
$$p_y = C_{1y} + C_{2y}y + C_{3y}y^2 \qquad (B),$$

where the C_{nx} or C_{ny} are coefficients, the maxima of these quadratics along X and Y respectively will be located at

$$x_m = -\frac{C_{2x}}{2C_{3x}} \qquad and \qquad y_m = -\frac{C_{2y}}{2C_{3y}}. \qquad (C)$$

The location of these maxima will be the "best" interpolated (refined) position of the peak on the correlation surface and all that remains to be done is to compute the coefficient values. These can be computed using at least three data points straddling the peak. If we take equation (A) for example, then given three unique data points [(x_i , p_{xi}); i = 1, 2, 3], we can form three simultaneous equations. In matrix form:

$$\begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{pmatrix} \begin{pmatrix} C_{1x} \\ C_{2x} \\ C_{3x} \end{pmatrix} = \begin{pmatrix} p_{x1} \\ p_{x2} \\ p_{x3} \end{pmatrix}.$$

The 3 x 3 matrix on the left belongs to a particular class of matrices called "Vandermonde" matrices. We can solve for the coefficients C_{nx} using standard matrix techniques. In fact, being a 3 x 3 system, this is relatively easy. Given the determinant of the 3 x 3 matrix is:

$$Det = x_1^2(x_3 - x_2) + x_2^2(x_1 - x_3) + x_3^2(x_2 - x_1),$$

the coefficients are explicitly given by:

$$C_{1x} = \frac{1}{Det}\left[p_{x1}(x_3^2 x_2 - x_2^2 x_3) + p_{x2}(x_1^2 x_3 - x_3^2 x_1) + p_{x3}(x_2^2 x_1 - x_1^2 x_2)\right]$$

$$C_{2x} = \frac{1}{Det}\left[p_{x1}(x_2^2 - x_3^2) + p_{x2}(x_3^2 - x_1^2) + p_{x3}(x_1^2 - x_2^2)\right]$$

$$C_{3x} = \frac{1}{Det}\left[p_{x1}(x_3 - x_2) + p_{x2}(x_1 - x_3) + p_{x3}(x_2 - x_1)\right]$$

Given these, the X-position of the maximum (interpolated) peak location on the correlation surface can be computed from equation (C) above. The same procedure is repeated for computing the Y-position. Note that the interpolated peak height (maxima of the quadtratic in either X or Y) must be greater than the initial peak height from the initial search above. This can be used to sanity check that the quadratic interpolation procedure gave “better” refined peak locations and hence registration offsets. Note that if by any chance the determinant from solving the above simultaneous system turns out to be zero, then no sub-pixel refinement of the correlation peak location can be performed.

4.2.12. Results Output

See section 6.1 below for an example of the results table and standard output produced.

4.2.13. Termination

Summary output is appended to the log file (the log file is created if previously non-existent), which includes diagnostic reports for the Q/A Subsystem and the appropriate exit code issued to be picked up by the pipeline executive. A detailed list of log file contents is given in Section 6.1.2.

5. Algorithm Requirements

A. FFTREGIST requires that all standard WCS keywords be present in the headers of the input FITS images, e.g., CRVAL1, CRVAL2, CRPIX1, CRPIX2, CD matrix (or CDELT1, CDELT2, CROTA2 if CD matrix is not present).

B. Note that the input test and reference images are required to be co-linear (i.e., no rotational offset exists). A slight rotational offset will manifest itself as an error in the computed orthogonal offets.

C. Due to the periodic nature of the images in frequency space after Fourier transforming, this algorithm is only capable of computing offsets along X and Y for images whose a-priori offsets are known to be smaller than (NAXIS1)/2 and (NAXIS2)/2 respectively, i.e., no larger than half the image size in each dimension.

D. A tutorial which lists all the command-line options can be generated by typing “fftregist” on the command line with no arguments. This tutorial will indicate which parameters are required and those which are optional. Optional parameters are assigned the default values defined in Table 1. If any of the required parameters are not specified, or are unacceptably out of range, the program will abort with a message sent to standard output to indicate this. Below is the tutorial generated when “fftregist” is executed without any command line arguments.

```
Program fftregist, Version 1.1

*** See namelist "fftregist.nl" for definitions of parameter variables.

Usage: fftregist
 -n  <inp_namelist_fname>     (Optional)
 -f1 <Input_Ref_FITS_Image>   (Required)
 -f2 <Input_Test_FITS_Image>  (Required)
 -o  <Output_Shifts_Filename> (Required)
 -l  <Log_Filename>           (Optional, Default='stdout')
 -a  <Ancillary_File_Path>    (Optional, Default='./')
 -m  <MIN_OVERLAP_FRAC>       (Optional, Default=0.5)
 -g  <FILTERFLAG>             (Optional, 0=none; 1=HiPass; 2=LoPass; Default=0)
 -g1 <FILTERITERNUM1>         (Optional, Default=1)
 -g2 <FILTERITERNUM2>         (Optional, Default=1)
 -s1 <SIGMAFILTER1>           (Optional, Default=1.0)
 -s2 <SIGMAFILTER2>           (Optional, Default=1.0)
 -b1 <BASEMASKWIDTH1>         (Optional, Default=1.0)
 -b2 <BASEMASKWIDTH2>         (Optional, Default=1.0)
 -c  <CROPNUMBORDERPIX>       (Optional, Default=0)
 -w  <WINDOWFLAG>             (Optional, 0=none; 1=Hamming; 2=Masci; Default=0)
 -i  <MASCIINDEX>             (Optional, Default=6)
 -h  <LOWFLUXCLIPFLAG>        (Optional, 0=no; 1=yes; Default=0)
 -q1 <SNRTHRESHOLD1>          (Optional, Default=5.0)
 -q2 <SNRTHRESHOLD2>          (Optional, Default=5.0)
 -p  <NPEAKSEARCH>            (Optional, Default=3)
 -r  <SEARCHRADIALTOL>        (Optional, Default=4.0)
 -d  (prints debug statements to stdout and files)
 -v  (verbose output)
 -vv (superverbose output)
```

6. Output

6.1. FFTREGIST Output Summary

FFTREGIST is capable of generating the following output:

A.) Standard-output processing and status messages.

B.) An output table in IPAC format containing orthogonal Cartesian offset information for the input images. An example is as follows:

```
\character Cartesian_Tangent_Shifts_File
\character From_fftregist_Program = "fftregist", Version  1.10
\character Creation_Date_Time = Tue Jun 28 17:42:34 2005
\character Reference_Image (shifts relative to) = ./testing/bcd_0138.fits
\character Test input image = ./testing/bcd_0141.fits
\-------------------------------------------------
|XT                              |YT                              |
|real                            |real                            |
|pixel                           |pixel                           |
       -1.0643266148569188e+00           4.3512760806682969e+01
```

C.) A log file containing processing statistics, status messages and ancillary information.

All FFTREGIST disk output is written to the pathnames that are specified with the output filenames in the command-line or namelist inputs.

6.2. FFTREGIST Log-File and Standard Verbose Output messages

The information stored in the log file at the output of this program includes: program name and version number, values of all namelist and/or command-line inputs, a message indicating the type of calculation performed, status code, processing time, date and time, and a message indicating program termination. An example of the information written to standard output and the log file is shown below by executing the program with the –v and –vv on the command line.

```
fftregist_parse_args: Program fftregist, Version 1.1
fftregist_parse_args: Namelist File = fftregist.nl
fftregist_parse_args: Ancilllary File Pathname = ../fftregist_v1
fftregist_parse_args: Output text file of Cartesian tangent shifts =
!./testing/Cartesianshifts.txt
fftregist_parse_args: Input Reference FITS Image = ./testing/bcd_0138.fits
fftregist_parse_args: Input Test FITS image = ./testing/bcd_0141.fits
fftregist_parse_args: MIN_OVERLAP_FRAC = 0.500000
fftregist_parse_args: FILTERFLAG = 2
```

```
fftregist_parse_args: FILTERITERNUM1 = 1
fftregist_parse_args: FILTERITERNUM2 = 1
fftregist_parse_args: SIGMAFILTER1 = 1.000000
fftregist_parse_args: SIGMAFILTER2 = 1.000000
fftregist_parse_args: BASEMASKWIDTH1 = 5.000000
fftregist_parse_args: BASEMASKWIDTH2 = 5.000000
fftregist_parse_args: CROPNUMBORDERPIX = 2
fftregist_parse_args: WINDOWFLAG = 2
fftregist_parse_args: MASCIINDEX = 6
fftregist_parse_args: LOWFLUXCLIPFLAG = 0
fftregist_parse_args: SNRTHRESHOLD1 = 30.000000
fftregist_parse_args: SNRTHRESHOLD2 = 30.000000
fftregist_parse_args: NPEAKSEARCH = 10
fftregist_parse_args: SEARCHRADIALTOL = 6.000000
fftregist_parse_args: Log File = stdout
fftregist_parse_args: I_Debug = 0
fftregist_parse_args: I_Verbose = 1
fftregist_parse_args: I_SuperVerbose = 0

fftregist_read_image1: Reading input data from FITS file 1...
fftregist_read_image1: Input FITS file 1: ./testing/bcd_0138.fits
fftregist_read_image1: I_Length_X1 = 128
fftregist_read_image1: I_Length_Y1 = 128
fftregist_read_image1: I_Num_Frames = 1

fftregist_read_image2: Reading input data from FITS file 2...
fftregist_read_image2: Input FITS file 2: ./testing/bcd_0141.fits
fftregist_read_image2: I_Length_X2 = 128
fftregist_read_image2: I_Length_Y2 = 128
fftregist_read_image2: I_Num_Frames = 1

fftregist_compute_results: Using WCS library VSN 2.8.6, 2 January 2001, Doug Mink, SAO
fftregist_compute_results: Reading WCS-related keywords from images...
fftregist_compute_results: Input image pair coverage fraction =  66.41%

fftregist_shiftsfrompntg: Estimating Cartesian shifts using blind pointing alone...
fftregist_shiftsfrompntg: Expected (predicted) offsets from relative pointing:
fftregist_shiftsfrompntg: (X/refpix,Y/refpix): -0.419427  43.177399

fftregist_window: Windowing input image data with window function...

fftregist_compute_filters: Making Gaussian filter masks...
fftregist_compute_filters: Width of mask 1 = 5 pixels, Width of mask 2 = 5 pixels

fftregist_apply_filter: Convolving filters with input images...
fftregist_apply_filter: Cropping 2 pixel border in images to remove filtering artifacts...

fftregist_compute_fwdFFTs: Computing Fourier transforms (FFTs) of input images...

fftregist_compute_correlation: Computing cross-power spectrum of input images in frequency
space...
fftregist_compute_correlation: Computing 2-d correlation function in real space...

fftregist_compute_tangentshifts: Computing Cartesian offsets in tangent reference plane by
searching for peak(s) on correlation surface...
fftregist_compute_tangentshifts: Radial search interval (about predicts) in reference frame
pixels = 2.356961
fftregist_compute_tangentshifts: Number of iterations used to search for concordant peak on
correlation surface = 2
fftregist_compute_tangentshifts: Peak value on X-Y correlation surface = 2231.756421
fftregist_compute_tangentshifts: "Coarse" (integer pixel) offsets in reference frame of
astrometric image: shifts in pixel units of reference image:
fftregist_compute_tangentshifts: Xshift/refpix=-1.000000, Yshift/refpix=44.000000
```

```
fftregist_interp_corrsurface: Refining integer pixel offsets to sub-pixel accuracy using
quadratic fits on correlation surface...
fftregist_interp_corrsurface: (x,y) Data for X-direction fit: (-2.000000,467.497020) (-
1.000000,2231.756421) (0.000000,-53.484242)
fftregist_interp_corrsurface: (x,y) Data for Y-direction fit: (43.000000,2201.908989)
(44.000000,2231.756421) (45.000000,-77.388725)
fftregist_interp_corrsurface: X-directional fit peak value on correlation surface=2240.134661
fftregist_interp_corrsurface: Y-directional fit peak value on correlation surface=2509.397216
fftregist_interp_corrsurface: Difference: X-direction fit peak - initial peak value = 8.378240
fftregist_interp_corrsurface: Difference: Y-direction fit peak - initial peak value = 277.640795
fftregist_interp_corrsurface: Final offsets in reference image frame: shifts in pixel units of
reference image:
fftregist_interp_corrsurface: Xshift/refpix=-1.064327,Yshift/refpix=43.512761
fftregist_output: Writing tangent shifts wrt reference image frame to
!./testing/Cartesianshifts.txt


fftregist_log_writer: Printing Log output to stdout...
Program fftregist, Version 1.1
Namelist File  = fftregist.nl
Input Reference FITS Image filename = ./testing/bcd_0138.fits
Input FITS image file to refine = ./testing/bcd_0141.fits
Output Cartesian tangent shifts file = ./testing/Cartesianshifts.txt
Decoding messages...
Message-code file = ../fftregist_v1/fftregist_errcodes.h
Performed pointing refinement computation.
Program fftregist: Status Message: 0x0000
Normal exit from Function 0x0000: LOG_WRITER
Processing time: 0.520000 seconds
Current date/time: Tue Jun 28 17:44:04 2005
Program fftregist, version 1.1, terminated successfully.
```

7. Testing

FFTREGIST has been successfully unit-tested as a stand-alone program for a variety of different input cases. The tests were designed to check FFTREGIST robustness and capability of generating corrected results.

Here is a summary of the unit tests that were conducted:

1. Tested FFTREGIST on simulated MIPS and IRAC images with known input registration offsets. The offsets were reproduced to an accuracy of 1 part in 10^5.

2. Executed FFTREGIST with inputs read from and output written to directories different from where the program was run. All namelist and command-line input mechanisms were exercised.

3. Executed FFTREGIST for all combinations of input parameters, in order to test that they function properly.

8. Usage Examples

Using a namelist file with verbose (-v) output re-directed to a file “out.log”:

```
fftregist –n fftregist.nl -v | & tee out.log
```

Without using a namelist file (i.e., using only command-line arguments).

```
fftregist -f1 bcd_0138.fits -f2 bcd_0141.fits –o
Cartesianshifts.txt -a . -m 0.5 -g 2 -g1 1 -g2 1 -s1 1.0 -s2 1.0
-b1 5.0 -b2 5.0 -c 2 -w 2 -i 6 -h 0 -q1 30.0 -q2 30.0 -p 10 –r
6.0 -v
```

9. Glossary

AOR	Astronomical Observer Request
DAC	Digital-to-Analog Conversion
DCE	Data Collection Event
FFT	Fast Fourier Transform
FPS	Focal Plane Survey
DN	Data Number
IOC	In-Orbit Checkout
LSB	Least Significant Bit
MIPL	Multi-Mission Processing Laboratory
MIPS	Multi-band Imaging Photometer for SIRTF
SDS	Subsystem Design Specification

SIRTF	Space Infrared Telescope Facility
SIS	Software Interface Specification
SSC	SIRTF Science Center
TBD	To Be Determined
TBR	To Be Resolved

10. References

Srinivasa Reddy, B., and Chatterji, B. N. 1996, "*An FFT-Based Technique for Translation, Rotation, and Scale-Invariant Image Registration*", IEEE Transactions on Image Processing, Vol. 5, No. 8.

THIS IS A PRELIMINARY DOCUMENT, the module described here may or may not be utilized in the final pipelines as described.