674-SO-43, Version 1.0,
SSC-PD-4052

SIRTF Science Center

Downlink Segment

Subsystem Design Specification

AOT Products Subsystem: DELTAFPTCORR

1 August 2001

National Aeronautics and
Space Administration

California Institute of Technology
SIRTF Science Center

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:

Mehrdad Moshir

Concurred by:

Bill Green

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	August 1, 2001

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

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.

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.

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

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

DELTAFPTCORR uses point source positions extracted from a set of standard FITS images and correlates them to compute the translational and rotational offsets between frames in a rectilinear coordinate system relative to either: a reference frame defined by a single image, or, an "average reference" defined by the mean pointing of all frames in the input list. The software uses a "chain-ladder" algorithm where all possible correlated frame *pairs* are tied together in a continuous path (and their offsets summed) until the frame in question becomes tied to the reference image. An inverse variance weighted mean offset of *all possible* correlated-image paths is computed to reduce the final uncertainty. The frames need NOT overlap with the reference image, they must only be tied to the reference image via a series of pairwise shifts between adjacent frames. An image which fails to have at least two point sources in common with any other image in the input list, or, is physically disjoint from the reference image will not be registered. These images will require registration using pointing header keywords. See the DELTAFPOINT SDS (SSC-PD-4045) for more details. It is important to note that DELTAFPTCORR can be executed as a stand-alone program with no dependence on DELTAFPOINT.

As a further detail, the user can choose the search radius for frame-to-frame source matching, a flux difference threshold for flux matching, final error threshold for controlling and terminating the *number* of frame-to-reference weighted paths, the rotational (x, y) center for the input frames and the pixel-scale in the reference frame. "DELTAFPTCORR" gets its name to stand for: "*Delta-File* from *Point-Source Correlation*".

The software uses routines from the standard World Coordinate Library (WCS) (Doug Mink, 2001, SAO) to convert point-source pixel positions to (absolute) celestial coordinates. All standard types of map-projections (as specified by relevant FITS keywords) are supported. The primary output from DELTAFPTCORR is a table in IPAC format listing the input images and their relative offsets and uncertainties from a user-specified reference frame. This table is commonly referred to as a "Delta-File". DELTAFPTCORR is primarily written in standard C with the exception of a subroutine written in FORTRAN 77.

2.1. DELTAFPTCORR Requirements

DELTAFPTCORR 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 a list of standard FITS images, a corresponding list of source extraction tables in IPAC format, an optional single reference frame FITS image and various processing parameters.

C.) Produce as primary output an IPAC table.

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

E.) Produce a processing summary either to standard output or a log file.

2.2. Applicable Documents

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

A.) The SOS Requirements Document

B.) The SOS Downlink Requirements Document

C.) The SOS Downlink Software Development Guidelines

D.) The following Software Interface Specifications (SIS):

SOSDL-SIS-PD-3009 (primarily for the IPAC table output)

2.3. Version History

2.3.1. Version 1.0

Initial version created on August 1, 2001.

2.4. Liens

The current software assumes that point sources are present in all images to be registered. If the final output delta-file table does not contain an entry for a particular image, then no point sources were found that were in common with others in the input list. Although this software gives a more accurate delta-file table than that computed via pointing keywords alone, it cannot guarantee that all frames will be registered. A mechanism (such as a script) that ties the output from this software to that which computes delta-files from pointing (e.g. DELTAFPOINT) is required.

3. Input

3.1. DELTAFPTCORR Input

DELTAFPTCORR 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. DELTAFPTCORR NAMELIST Input

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

Namelist variable	Description	Dim.	Type	Units	Default
FITS_Image_List_Filename	Required filename containing list of FITS-images.	256	C	(redundant)	Null
Source_Table_List_Filename	Required filename containing list of source extraction tables in IPAC format	256	C	(redundant)	Null
FITS_Ref_Image_Filename	Optional FITS image from list representing a "single" image reference frame. **Must specify if RefFrameFlag = 2 below.**	256	C	(redundant)	Null
Data_Out_Filename	Required name of Delta-File.	256	C	X,Y/pixels theta/deg.	Null
RefFrameFlag	1 = use average reference frame, 2 = use single reference frame specified above.	1	I*1	-	1
Max_Search_Radius	Search radius around each point source to perform source matching	1	R*4	arcseconds	1.5
Max_Flux_Diff	Largest flux difference tolerable to perform source matching	1	R*4	Percent	5%
ErrorThres	Maximum desired error in X and Y pixel shifts.	1	R*4	Input pixels	0
CenterFlag	1 = use CRPIX values as rotational center, 2 = use Xcenter, Ycenter below.	1	I*1	-	1

Xcenter	*x*-coord. of rotational center.	1	R*4	Input pixels	0
Ycenter	*y*-coord. of rotational center.	1	R*4	Input pixels	0
CdeltFlag	1 = use pixel scales from input reference image or 1st image in list. 2 = use values below.	1	I*1	-	1
Xpixscale	Pixel scale in *x* coordinate.	1	R*4	deg./pixel	0.001
Ypixscale	Pixel scale in *y* coordinate.	1	R*4	deg./pixel	0.001
Log_Filename	Optional output log filename	161	C	-	stdout
Ancillary_File_Path	Pathname where supporting source files are installed.	161	C	-	./ (current directory)

Table 1. Namelist file

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

```
&DELTAFPTCORRIN
  Comment = 'Generic namelist file for deltafptcorr, default values.',
  Ancillary_File_Path = '../deltafptcorr_v1',
  Comment = 'Following required if RefFrameFlag = 2 below:',
  FITS_Ref_Image_Filename = './testing/sws4322a00205.fits',
  FITS_Image_List_Filename = './testing/deltafptcorr_images.list',
  Source_Table_List_Filename = './testing/deltafptcorr_src_tbls.list',
  Data_Out_Filename = './testing/pointcorr.tbl',
  Comment = '1 = compute average, 2 = use ref_file above, Default = 1:',
  RefFrameFlag = 2,
  Comment = '[arcsec], Default = 1.5',
  Max_Search_Radius = 1.5,
  Comment = '[percent], Default = 5',
  Max_Flux_Diff = 5,
  Comment = '[pixels], Default = 0, set to 0 for best results!',
  ErrorThres = 0,
```

```
  Comment = '1 = use CRPIX values, 2 = use values below, Default = 1:',
  CenterFlag = 1,
  Comment = 'Xcenter, Ycenter in units of pixels:',
  Xcenter = 0,
  Ycenter = 0,
  Comment = '1 = use keyword CDELT values from ref. frame, 2 = use values
below, Default = 1:',
  CdeltFlag = 1,
  Comment = 'X-pixelscale, Y-pixelscale in units of [deg/pixel]:',
  Xpixscale = 0.001,
  Ypixscale = 0.001,
  Log_Filename = 'stdout',
 &END
```

3.1.2. DELTAFPTCORR 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 overriding 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. DELTAFPTCORR FITS Input

DELTAFPTCORR 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, and fits_close_file.

Command-line option	**Variable**
-n	Namelist_Filename
-f1	FITS_Image_List_Filename
-f2	Source_Table_List_Filename
-i	FITS_Ref_Image_Filename
-o	Data_Out_Filename

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

-g	RefFrameFlag
-sr	Max_Search_Radius
-sf	Max_Flux_Diff
-e	ErrorThres
-cf	CenterFlag
-cx	Xcenter
-cy	Ycenter
-pf	CdeltFlag
-px	Xpixscale
-py	Ypixscale
-l	Log_Filename
-a	Ancillary_File_Path
-v (verbose switch)	-
-vv (super-verbose switch)	-
-d (debug switch)	-

Table 2. Command-line options

4. Processing

4.1. DELTAFPTCORR Processing

DELTAFPTCORR 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, then 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 all input parameters, date and time, processing time, and a termination-status code are written a log file.

4.2. DELTAFPTCORR Processing Phases

DELTAFPTCORR operates in eleven phases: initialization, conversion of point-source pixel positions to celestial coordinates using FITS pointing keywords, point-source position and flux matching for all possible frame pairs, computation of pairwise image shifts and rotations, path search from image to reference using a "chain ladder" algorithm, computation of final translational and rotational offsets using inverse variance weighting, computation of "average" reference frame if specified, rotational and reference-pixel re-scaling, error-propagation and estimation, delta-file output table and termination. This processing level is depicted in Figure 1.

4.2.1. DELTAFPTCORR Initialization

DELTAFPTCORR 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 and checked for correct data range, consistency, etc. If any errors are encountered, a message is printed, and execution aborts.

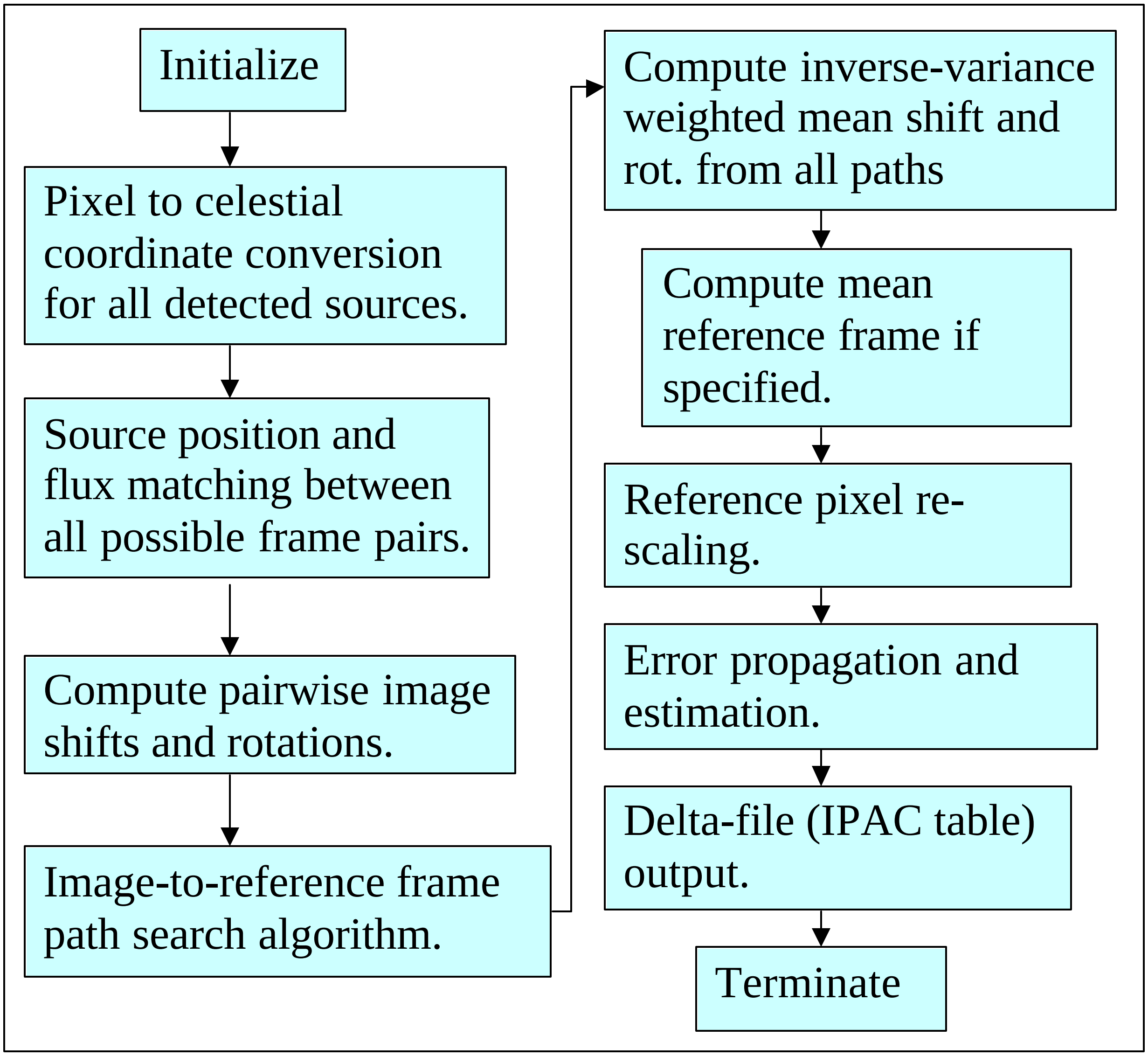


Figure 1. DELTAFPTCORR data and processing flow

4.2.2. Pixel to Celestial (RA, DEC) Coordinate Conversion

From point-source extracting software, each image will have an associated list (in IPAC-table format) of point sources detected therein with pixel coordinates (x, y). To minimize centroiding errors, the brightest sources should be extracted. The list of source extraction tables is specified by the namelist

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

parameter: Source_Table_List_Filename. An example of an input point-source extraction table is shown below. Only columns relevant to this software are shown.

The pointing keywords in the list of input FITS images (namelist parameter: FITS_Image_List_Filename) are read using the *fitsrhead* routine from the WCS library. Each set of pointing keywords from each image header are stored in memory as structure variables using the WCS routine *wcsinit*. In general, all images must be composed of a single data plane.

The RA, DEC pointing keywords (CRVAL1, CRVAL2), corresponding pixel coordinates of the tangent point (CRPIX1, CRPIX2), and all distortion keywords (e.g. associated with the TNX convention) are used to transform the pixel coordinates of all detected point sources in every image to an “absolute” (RA, DEC) coordinate system. This is done to perform the source matching (see below).

Next, the RA, DEC position of all sources are transformed back to their respective x, y image planes using a pure TAN projection. The reason for this is to facilitate “robust” source matching in a rectilinear x, y coordinate system <u>corrected for distortion</u>. Since the distortion will inevitably be a function of position in an image, sources situated at random x, y locations will be subject to differing amounts of scale change and skewness relative to each other. These effects must be normalized-out before correlating point sources.

```
\char comment = Output from SOURCESTIMATE, version  3.00
\char Date-Time = Mon Jun 11 15:42:10 2001
\char InDetectList = /ssc/pipe/davidm/pipe/Detect/PSP1_A.detect.tbl
\char OutExtractList = sws4322a00201.tbl
\char DeltaFileName = DeltaFile.tbl
\char SigmaFilenameList = SigmaList
\char TileMapFilename = TileMap
\char PRF_Map = PRFMap.tbl
\char PRF_Filename_1 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\char PRF_Filename_2 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\char PRF_Filename_3 = /ssc/pipe/davidm/pipe/Test/IRAC.3.8um.PRF.12.fits
\float Pixel_RatioX = 1
\float Pixel_RatioY = 1
\float Mosaic_CenterX = 0
\float Mosaic_CenterY = 0
\int Fitting_Area_X = 7
\int Fitting_Area_Y = 7
\int Max_Number_PS = 3
```

```
\float Chi_Threshold = 1.5
\int N_Edge = -4
\int PRF_Array_Xsize = 45
\int PRF_Array_Ysize = 45
\int PRF_RescaleX_Factor = 12
\int PRF_RescaleY_Factor = 12
\char Mode = prf_mode
\char InputType = delta_file
| srcid| detid|  N_PS|        x|  delta_x|        y|  delta_y|     flux|
|i     |i     |i     |r        |r        |r        |r        |r        |
      1      1      1     45.34  1.88e+00      8.98  1.89e+00  1.00e+02
      2      2      1     36.22  2.63e+00     13.22  2.61e+00  7.10e+01
      3      3      1     18.14  1.12e+00     19.15  1.11e+00  1.75e+02
      4      4      1     55.30  1.86e+00     19.97  1.85e+00  1.02e+02
      5      5      1     38.57  2.57e+00     32.18  2.66e+00  7.21e+01
      6      6      1    125.63  2.09e+00     37.20  2.10e+00  8.99e+01
            .
            .
            .
```

4.2.3. Point Source Correlation Between Frame Pairs

Every possible frame pair combination from the input list is searched for a "common" set of point sources in RA, DEC space. For a list of N images, there is a maximum of $\frac{1}{2}N(N-1)$ possible pairs that can be correlated. Both position and flux matching is performed. The position matching step attempts to find sources that fall within a nominal radius (namelist parameter: Max_Search_Radius). If more than one match is found within this search radius, the "closest" match is reported.

In addition to position matching, sources are simultaneously matched in flux. To avoid use of an absolute flux scale, a flux-match is satisfied if any two fluxes fall within a maximum tolerable flux difference threshold (namelist parameter: Max_Flux_Diff). In units of percent, this threshold is defined:

$$\mathrm{Max\_Flux\_Diff} = 100\frac{|f_2 - f_1|}{f_2 + f_1},$$

where f_1 , f_2 are the respective fluxes of two sources from two different images (or source extraction tables) to be correlated.

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

The software also includes logic to ensure that there exists at least two point sources common between a pair of images separated by a minimum distance of 5 pixels. Formally, this is defined as the parameter CLOSEST_SOURCE_SEPN in the include file "*deltafptcorr.h*". This minimizes the uncertainty in the rotational offset between a pair of images, in particular when only two correlated sources are found. A minimum of two point sources is required to estimate a relative rotational offset.

4.2.4. Pair-wise Image Shifts and Rotations

Using the re-computed "TAN" projected x, y positions of correlated sources in each image pair (see Section 4.2.2), rotational (q_i) and translational (X^i_T , Y^i_T) offsets are computed for each image pair. The general image registration equations which account for rotations are used:

$$X_T^i = x'_i - (x_i - x_c)\cos J_i - (y_i - y_c)\sin J_i - x_c \qquad (1)$$

$$Y_T^i = y'_i + (x_i - x_c)\sin J_i - (y_i - y_c)\cos J_i - y_c\, , \qquad (2)$$

where [x_i , y_i] is an input-image (point-source) pixel position, [$x¢_i$, $y¢_i$] is its corresponding position in the reference image of the pair and [x_c , y_c] are input-image coordinates of the rotational center. [x_c , y_c] default to the FITS keywords CRPIX1, CRPIX2 unless specified by the namelist parameters Xcenter, Ycenter. In essence, X^i_T and Y^i_T are relative orthogonal translations between the reference frame and the co-rotated input-image, that is, image origin offsets after rotating the input-image so that it is unrotated with respect to the reference frame.

To compute a unique set of offsets (q_i , X^i_T , Y^i_T) for each image pair, a robust minimization algorithm is used. This algorithm is written in FORTRAN 77 and was originally designed for the WIRE project (for more details see http://spider.ipac.caltech.edu/staff/laher/rotlsqdr.html). We call it as a subroutine from the main C program. The method uses a fitting technique to solve for the three offset parameters with the constraint:

$$\cos^2 J_i + \sin^2 J_i = 1 \qquad (3)$$

The coupled set of equations (1), (2) and (3) are solved using a non-linear least squares method. A Newton-Raphson iterative method is used to recursively solve for q_i separately. For added security in the reliability of point-source matches, the algorithm also includes logic to identify and reject outlier (discrepant) matches. *An assumption of the algorithm is that positional uncertainties are taken to be equal in the x and y direction of the image plane.*

The original stand-alone program has a number of input parameters. We have isolated the most important for our current application and set their values in the include file "*deltafptcorr.h*". There is no need to change their values as they have been set to yield the "best" results when integrated with the main C program. Their definition and default values are as follows:

RADTOL: Radial position error threshold for determining a candidate outlier in units of pixels. Default = 1.0.

CHI2TOL: Threshold for relative change in χ^2 fit after rejection of candidate outliers. Outlier rejection will be performed if this value is greater than CHI2TOL. Default = 1.0.

NPOSTOL: Number of sigmas of the x or y position error for determining a candidate outlier. Default = 5.

NFIRST: Maximum number of sources to reject in the first stage of outlier-rejection processing. Default = 2.

Furthermore, the algorithm requires a relative "weight" to be specified for each correlated point-source measurement. This is computed by taking the mean inverse variance of the point source positional error between the two images (1 and 2) of a correlated pair:

$$\text{Weight } = \frac{1}{2}\left(\frac{1}{s^2(x_1)} + \frac{1}{s^2(x_2)}\right).$$

The above subroutine sends diagnostic output messages to standard output when DELTAFPTCORR is executed in the verbose (-v switch) mode. Both the average (model fit - measurement) residuals in x and y direction and the final χ^2 is sent to standard output. The final output captured from the above algorithm consists of five quantities for each correlated image pair – the three offsets (q_i , X^i_T , Y^i_T) and uncertainties in the translational offsets: $\sigma(X^i_T)$ and $\sigma(Y^i_T)$.

This algorithm does not estimate the uncertainty in q_i. Instead, we compute $\sigma(q_i)$ by solving for q_i from the registration equations (1) and (2) and propagating errors using uncertainties in the translational offsets. From Equations (1) and (2), we have:

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

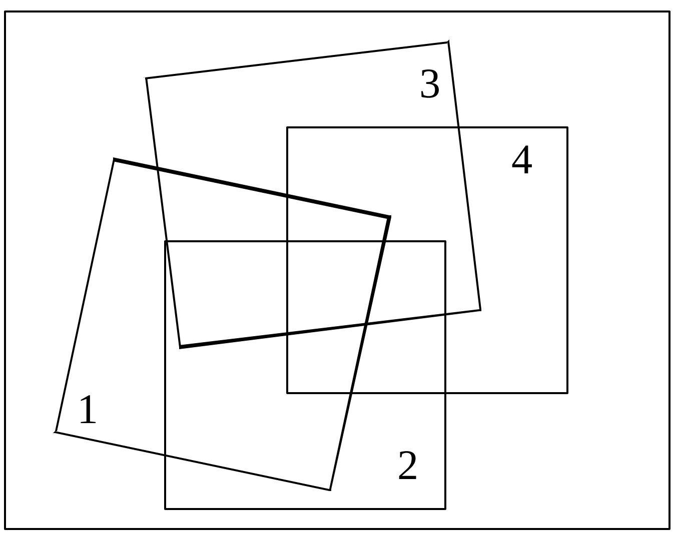
$$\tan(J_{i,k}) = \frac{(X_T^i - x'_k + x_c)(y_k - y_c) - (Y_T^i - y'_k + y_c)(x_k - x_c)}{(Y_T^i - y'_k + y_c)(y_k - y_c) + (X_T^i - x'_k + x_c)(x_k - x_c)}, \qquad (4)$$

where $[x_k , y_k]$ and $[x¢_k , y¢_k]$ are the respective coordinates of a single point source in each image of a correlated pair i with translations (X^i_T , Y^i_T). After performing error propagation in equation (4) to compute $\sigma(q_{i,k})$ in terms of uncertainties in the translational offsets, a mean uncertainty is computed for all N correlated point sources in an image pair:

$$s(J_i) = \left[\sum_k^N \frac{1}{s^2(J_{i,k})} \right]^{-1/2}. \qquad (5)$$

4.2.5. Image-to-Reference Path (Chain) Searching

For all correlated image pairs in the input list, we now have a set of six quantities for each (rotational and translational offsets and their respective uncertainties). To compute the overall offset of a single image from a user-specified reference image, we use a “chain ladder” algorithm where successive pairwise offsets are summed between the two images. We describe the method using an example of a random collection of four overlapping images shown in Figure 2.



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

Figure 2. Example mosaic to illustrate path search.

For $N = 4$ images containing a common set of point sources, the total number of image-pair correlations is $\frac{1}{2}N(N-1) = 6$. The image pairs are as follows: **1-2; 1-3; 1-4; 2-3; 2-4; 3-4**. Let us assume that image number **1** in the above figure is the reference image from which we wish to compute the rotational and translational offsets of image number **2.** The most robust way to compute these offsets is to make use of all neighboring images which overlap with image number **2**, and compute the offsets from all possible distinct "paths" that traverse these intermediate images when going from image-to-reference. An inverse-variance weighted mean of all possible paths will then yield a robust measure of the offset. The greater the number of paths, the lower the final error in the weighted mean. In the above example, 11 distinct (non-repetitive) paths can be formed from image **2—to—1:**

(1) **2—1.**

(2) **2—3; 3—1.**

(3) **2—4; 4—1.**

(4) **2—3; 3—2; 2—1.**

(5) **2—4; 4—3; 3—1.**

(6) **2—3; 3—4; 4—1.**

(7) **2—4; 4—2; 2—1.**

(8) **2—4; 4—3; 3—2; 2—1.**

(9) **2—3; 3—4; 4—2; 2—1.**

(10) **2—3; 3—2; 2—4; 4—1.**

(11) **2—4; 4—2; 2—3; 3—1.**

In the general case, the above algorithm is automated as follows: first, each image of a correlated pair *i* is stored in a separate array: ***A(i), B(i)*** . Second, given a target reference image **R** and an image **I** whose offset we wish to compute, the list of correlated pairs is searched recursively until a unique path:

> **I** = ***A(i)—A'(i)…B'(i)—*R** = ***B(i)*** is found for all possible intermediate frames
>
> ***A'(i)*** ^{1} **I** and ***B'(i)*** ^{1} **R.**

The final rotational and orthogonal translational offset for each unique path k can be computed by summing the individual offsets of each image pair i (see section 4.2.4) used to define the path. The total error (or variance) for each path is computed by summing the variances of all pairs N_i within the path:

$$q^k = \sum_i^{N_i} q_i \quad , \quad s^2(q^k) = \sum_i^{N_i} s^2(q_i) \tag{6}$$

$$X_T^k = \sum_i^{N_i} X_T^i \quad , \quad s^2(X_T^k) = \sum_i^{N_i} s^2(X_T^i) \tag{7}$$

$$Y_T^k = \sum_i^{N_i} Y_T^i \quad , \quad s^2(Y_T^k) = \sum_i^{N_i} s^2(Y_T^i) \tag{8}$$

4.2.6. Inverse Variance Weighted Mean Shifts and Rotations

Having defined all the paths connecting an image in the input list to some pre-specified reference image, an inverse-variance weighted mean of all offsets from these paths (Equations 6, 7, 8) is computed together their uncertainties. The equations for computing the mean rotational offset <θ> amongst N(k) paths for the image in question is as follows. Equations for X, Y are analogous. First, a weight is defined:

$$w_k = \frac{1}{s^2(q^k)} \tag{9}$$

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

$$\langle q \rangle = \frac{\sum_{k}^{N(k)} w_k q^k}{\sum_{k}^{N(k)} w_k} \tag{10}$$

$$s\langle q \rangle = \frac{1}{\sqrt{\sum_{k}^{N(k)} w_k}} \tag{11}$$

When the number of overlapping images containing common sources becomes large, the number of paths from image-to-reference also becomes large and is costly computationally. To ameliorate this, we have defined a namelist parameter "ErrorThres" which represents the lowest possible error desired in the final X, Y offsets when computing a weighted average over all possible paths. In other words, the path search is terminated until the final error in X and Y (as computed by equations analogous to Eqn. 11) becomes no smaller than the user-specified value "ErrorThres":

$$\text{ErrorThres} = \sqrt{s^2\langle X \rangle + s^2\langle Y \rangle} \tag{12}$$

If ErrorThres is larger than any computable error from the algorithm, then only the offsets for the path (or paths) that lead to the largest computable error is returned. If the specified value for ErrorThres is too small, then the smallest computable error (involving all paths) is returned. For accurate results, it is advised that ErrorThres be set to zero.

4.2.7. Defining the Average Reference Frame (Optional)

The final (inverse-variance weighted) mean offsets discussed in the algorithm above assumed they were computed relative to a single reference image from the input list (namelist parameter FITS_Ref_Image_Filename) with the flag RefFrameFlag = 2. If however RefFrameFlag = 1, then a frame of reference defined by an average of all the input pointings is computed and all offsets are re-computed relative to this new fiducial frame. An average frame of reference and offsets computed there from is accomplished as follows:

(1). The position angles (header keyword CROTA2; measured East from North on the sky) from all input images are averaged. To avoid ambiguity in the average over the range 0 - 360° if

CROTA2 > 180°, the following algorithm is used. A set of CROTA2 values: $C_1, C_2, C_3 \ldots C_N$ is first averaged in the normal way which we designate as C_{Av1}:

$$C_{Av1} = \frac{1}{N}\sum_i C_i$$

If any values C_i exceed 180°, they are replaced by $C_j = C_i - 360°$ and a new average C_{Av2} is computed:

$$C_{Av2} = \frac{1}{N}\sum_j C_j$$

Next, the "absolute differences" D_1, D_2 of these averages from the initial C_i are computed:

$$D_1 = \sum_i \left| C_{Av1} - C_i \right| \quad \text{and} \quad D_2 = \sum_i \left| C_{Av2} - C_i \right|$$

The correct average value we will want to use is that which yields the smallest difference value. Let us define this average by <crota2>.

(2). We start by selecting a frame with known RA, DEC (keywords CRVAL1, CRVAL2) and corresponding pixel coordinates (CRPIX1, CRPIX2) by choosing an image from the input list which has the *maximum* number of overlaps (or correlations) with all other images. The CROTA2 and map projection (CTYPE keywords) of this "basis" frame are re-defined to be <crota2> and TAN. In other words, we will project onto a "TANGENT" (undistorted) reference plane. The *wcsdeltset* and *wcstype* routines are used to define this basis frame. Let us call this frame "**R**"**.** Effectively, this frame represents an "intermediate" image from which relative offsets are computed using the above algorithm. A simple transformation will then be used to transform these offsets relative to our "average" fiducial frame.

(3). All input images *i* are transformed to pixel coordinates in the basis reference frame defined in 2 above. We will define a pixel coordinate in the basis frame as (x'_i, y'_i). The *wcs2pix* routine is used:

$$RA_i, DEC_i \xrightarrow[<crota2>,TAN]{} (x'_i, y'_i)$$

(4). All coordinates corresponding to all input images in the basis frame are averaged, yielding ($<x'_i>$, $<y'_i>$). These are then transformed to a "mean" sky pointing using a TANGENT projection. The *pix2wcs* routine is used:

$$< x'_i >, < y'_i > \xrightarrow[<crota2>,TAN]{} < RA >, < DEC >.$$

The following set of keywords therefore fully define the ***average frame*** of reference:

CRVAL1 = $< RA >$
CRVAL1 = $< DEC >$
CRPIX1 = $CRPIX1_R$
CRPIX2 = $CRPIX2_R$
CROTA2 = $< crota2 >$
CDELT1 = $CDELT1_R$
CDELT2 = $CDELT2_R$
CTYPE1 = $RA{-}{-}{-}TAN$
CTYPE2 = $DEC{-}{-}TAN$

The quantities with a "**R**" subscript refer to values in the header of the FITS image defined in step 2 above. The above keywords are copied to the header of the output delta-file if the averaging option is chosen (see Section 6.1.2).

Our method for computing rotational and translational offsets of input images relative to the "average" reference image defined above is as follows. Coordinates shown with a prime (e.g: x'_i , y'_i ; x''_i , y''_i etc..) refer to quantities transformed into the average *reference* frame as shown in Figure 3. Un-primed coordinates are measured in the intermediate frame **R**.

(1). First, the rotational and translational offset of an image **I** from our intermediate image **R** is computed using the formalism in sections 4.2.4 – 4.2.6. Let us define these quantities as $q_{\mathbf{I-R}}$, $\mathbf{X_{I-R}}$, $\mathbf{Y_{I-R}}$.

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

(2). The CRPIX1, CRPIX2 values of image **R** are transformed to pixel coordinates in the average reference frame using a TANGENT projection. The *wcs2pix* routine is used:

$$RA_R, DEC_R \xrightarrow[\text{Reference Frame Parameters}]{} (x'_R, y'_R)$$

(3). To compute the rotational offset of image **R** with respect to the average frame, we define a second point in image **R** with coordinates [x_2 , y_2] = [0.5*NAXIS$_i$ + CRPIX1$_i$, CRPIX2$_i$] to transform into our average reference frame. Our first point is simply [x_1 , y_1] = [CRPIX1$_R$, CRPIX2$_R$]. This choice for the second point is arbitrary. It does not need to exist in the actual image, but together with first point, it must form a perpendicular bisector with the y-axis in the input image. Figure 3 shows the geometry involved. The primed axes refer to the reference frame.

In order to transform this second point from each image into the reference frame, we must first transform onto the sky:

$$x_2, y_2 \xrightarrow[\text{Input image } R \text{ parameters with TAN projection}]{} RA_2,\ DEC_2 .$$

To avoid uncertainties due to distortion in the telescope focal plane when projecting an input image onto the sky, a TANGENT projection is ensured. In other words, if the FITS header indicates a map projection other than “TAN” (e.g. “TNX”) then the relevant keywords in the image header (CTYPE1, CTYPE2) are reset to “TAN”. This is accomplished using the *wcstype* routine. The celestial coordinates of this second point are then transformed to the average reference frame:

$$RA_2, DEC_2 \xrightarrow[\text{Av.Reference Frame Parameters}]{} (x''_R, y''_R)$$

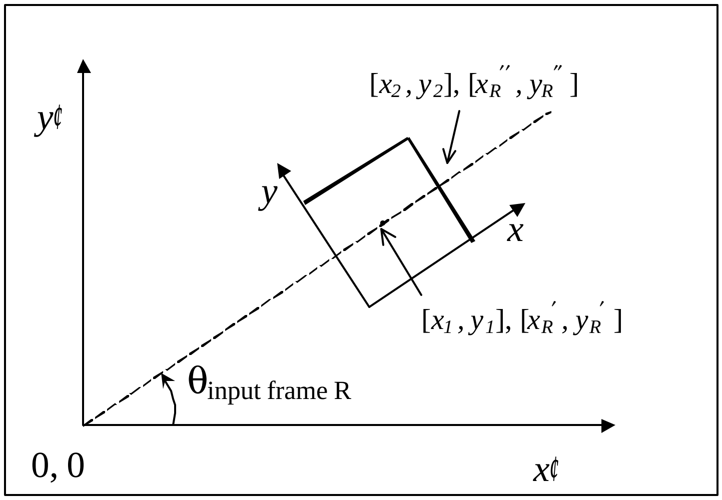


Figure 3. Schematic showing intermediate and reference image planes

(4). Having transformed these two points from image **R** into the average reference frame [$x¢_R$, $y¢_R$] and [$x¢¢_R$, $y¢¢_R$]) the rotational offset of image **R** from the average image frame can be computed:

$$q_{R-Av} = -\arctan\left[\frac{y''_R - y'_R}{x''_R - x'_R}\right] . \tag{13}$$

(5). The orthogonal translations ($X_{R\text{-}Av}$, $Y_{R\text{-}Av}$) between image **R** and the average reference image are computed using the value of theta from equation (13) in the registration equations (1) and (2).

(6). The final offsets of image **I** relative to our fiducial "average" reference image can now be computed by the following transformations:

$$q_{I-Av} = q_{I-R} + q_{R-Av} . \tag{14}$$

$$X_{I-Av} = X_{I-R} + X_{R-Av} . \tag{15}$$

$$Y_{I-Av} = Y_{I-R} + Y_{R-Av} . \tag{16}$$

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

4.2.8. Rotational and Reference-Pixel Re-scaling

One can specify the pixel scale (in degrees/pixel) for the reference frame or units in which translational offsets are measured if desired. This is specified by setting the namelist/command-line parameter CdeltFlag = 2 and setting values for the parameters Xpixscale and Ypixscale. On the other hand if CdeltFlag = 1, then the pixel scale from the input images is used which must be the same for all images.

4.2.9. Error Propagation and Estimation

Uncertainties in rotational and translational offsets for each input image are computed from propagating two components: first, the point-source centroiding errors from the source extractor (required by the pairwise image registration algorithm described in section 4.2.4), and second, uncertainties associated with the pointing keywords CRVAL1 (RA) and CRVAL2 (DEC) when transforming pixel positions of point sources to RA and DEC (section 4.2.2). This pointing error remains imprinted on each source when it is re-transformed back to the *x*, *y* image plane using a TAN projection. These two components can therefore be combined (in quadrature) to define an effective uncertainty in the centroids of the initial extracted point sources. This will eventually propagate to uncertainties in the final offsets as represented by equations (6) – (11) above.

The uncertainties purely due to pointing (in units of input image pixels) can be written:

$$s^2_{px} = \frac{1}{s^2_x}[s^2(RA) + s^2(DEC)]. \qquad (17)$$

$$s^2_{py} = \frac{1}{s^2_y}[s^2(RA) + s^2(DEC)], \qquad (18)$$

where s_x and s_y are image scales in the x and y image plane respectively given in units of degrees/pixel and $s(RA)$, $s(DEC)$ are pointing uncertainties given by the FITS header keywords CRDER1 and CRDER2 in degrees. The image scale parameters are also taken from the FITS headers (namely the CDELT1 and CDELT2 keywords). The final effective uncertainty in the (x, y) centroid of an extracted point source (that is propagated along) is given by:

$$\Delta x_{eff} = \sqrt{(\Delta x)^2 + s^2_{px}}, \qquad (19)$$

$$\Delta y_{eff} = \sqrt{(\Delta y)^2 + s_{py}^2}, \qquad (20)$$

where Δx and Δy represent the initial positional uncertainties determined from the source-extraction algorithm (taken from the input source extraction table – see section 4.2.2).

4.2.10. Delta-File (IPAC Table) Output

All relevant keywords that define the reference frame, user-specified parameters, rotational and translational offsets (from equations 9 –11 and their counterparts), uncertainties and coordinates of the rotational center for all input images are output to what's called a delta-file in IPAC table format. A detailed description of the output is given in section 6.1.2.

4.2.11. 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 6.1.1.

5. Algorithm

5.1. Algorithm Specifics

The algorithm used by DELTAFPTCORR has been adequately described in the previous section. As a detail, the WCS software library supports 26 different map projections with which to perform coordinate transformations. Pointing keywords need not conform to the standard FITS conventions (e.g. CRVAL, CRPIX, CROTA2 etc..), as long as they are recognised by the WCS library. In general, all celestial coordinates are measured in degrees with $0 \leq RA \leq 360°$, $-90° \leq DEC \leq 90°$ and $0 \leq CROTA2 \leq 360°$ (the position angle measured East from North).

5.2. Assumptions and Requirements

A. DELTAFPTCORR assumes that each FITS image in the input list has a *single FITS header* that defines a unique pointing for a single plane contained therein, in other words, with the standard FITS keyword values: NAXIS = 2 or NAXIS3 = 1. If this is not true, the program will abort with a message sent to standard output.

B. Every image in the input list must have the same pixel scale, i.e. same values for the standard CDELT FITS keywords. Also, they must have the same dimensions as specified by NAXIS1 and NAXIS2. They can be non-square with NAXIS1 ≠ NAXIS2.

C. The user-specified pixel scales for the reference frame (namelist parameters: Xpixscale, Ypixscale) must be greater than zero.

D. FITS images in the input list (namelist parameter: FITS_Image_List_Filename) are listed one per line and do not have to be in any specific order. Source extraction tables in the input list (namelist parameter: Source_Table_List_Filename) must have a one-to-one correspondence with respective images in the FITS image list.

E. It is recommended that the source extraction tables contain the brightest sources from each image in order to reduce source-matching errors due to centroiding errors.

F. The maximum number of images allowed in the input list is currently 1000. This is defined by the MAX_NUMBER_IMAGES parameter in the include file *deltafptcorr.h*.

G. The maximum number of entries (starting from the first entry) read from each source extraction table is currently 50. This is defined by the MAX_NUM_ENTRIES parameter in the include file *deltafptcorr.h*.

H. If a single reference image is specified in the namelist or command-line (FITS_Ref_Image_Filename), this image *must* be listed in the input list. If it is not in the input list, the program will abort with an error message sent to standard output.

I. DELTAFPTCORR searches FITS headers for the keywords CRDER1 and CRDER2 which specify the uncertainties corresponding to CRVAL1 (RA) and CRVAL2 (DEC) respectively. If these are not found, they default to the value zero and a message is sent to standard output to indicate this.

J. If an entry for an input image in the final delta-file table is missing, this implies that either the image was "disjoint" from the reference image when searching for a path amongst all correlated input images, or, not enough point sources were found that were in common with other images.

K. If the delta-file output from DELTAFPTCORR is required to resemble (in the same order) the output generated from the pointing software DELTAFPOINT, the input images in the input lists must be in the same order prior to executing each program.

6. Output

6.1. DELTAFPTCORR Output

DELTAFPTCORR is capable of generating the following output:

A.) Standard-output processing and status messages.

B.) An IPAC table file (the "delta-file"). See Section 6.1.2 for details.

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

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

6.1.1 DELTAFPTCORR Log-File Output

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.

6.1.2 Delta-File (IPAC-Table) Output

The primary output from DELTAFPTCORR is a table in IPAC format which consists of two parts: first, the table header and second, the relative offsets for each image. The table header lists: the program name and version number, time and date the table was generated, input image list filename, filename of single reference image if specified, otherwise a message indicating that an average reference of all frames was computed, pointing information of the single reference image if specified, otherwise these are replaced by values that define the average computed reference frame.

The table portion lists in column order left to right: image index label, image filename (including path), offset in *x*- and *y*-directions in units of reference frame pixels with scales (in degrees/pixel) indicated in the table header (i.e. by CDELT1 and CDELT2), rotational offset in degrees, uncertainties in these three quantities, and last, *x*,*y* coordinates of the rotational center for each individual input image from the input list. The "Index" label (column 1) refers to the image number in the input FITS image list. If an image is uncorrelated with any of the other input images, it will not have an entry in the delta-file.

Below are two examples of delta-file outputs corresponding to two different frames of reference for the relative offsets where the numerical values are not necessarily realistic.

Offsets relative to single input image:

```
\character Delta_File_Program = Output from deltafptcorr, version  1.00
\character Creation_Date_Time = Tue Aug 14 15:44:48 2001
\character Input_Image_List = ./testing/test_fits.list
\character Input_Source_Table_List = ./testing/test_tbl.list
\character Reference_Image = ./testing/sws4322a00205.fits
\integer Number_of_Frames = 9
\real RefFrCRVAL1 = 82.218150
\real RefFrCRVAL2 = 35.836323
\real RefFrCRPIX1 =   64.5
\real RefFrCRPIX2 =   64.5
\real RefFrCROTA2 = -0.092466
\real RefFrCDELT1 = -0.004323
\real RefFrCDELT2 = 0.004297
|Index   |Filename           |Xshift   |Yshift     |Rotation  |err_Xshift |err_Yshift |err_Rot
|Xcenter |Ycenter |
|i       |c                  |r        |r          |r         |r          |r          |r          |r
|r       |
        1 sws4322a00206.fits   -24.299     4.921    -0.15818     0.000       0.003     0.93896     64.5
64.5
        2 sws4322a00207.fits     3.310    -5.575    -0.03681     0.012       0.009     1.14023     64.5
64.5
        3 sws4322a00208.fits    12.798    -6.536    -0.06428     0.004       0.008     0.80954     64.5
64.5
        5 sws4322a00201.fits    -2.490    -5.805    -0.06592     0.011       0.009     0.73348     64.5
64.5
        6 sws4322a00202.fits    -5.880   -10.995    -0.05715     0.001       0.010     1.27114     64.5
64.5
        8 sws4322a00205.fits     0.000     0.000     0.00000     0.000       0.000     0.00000     64.5
64.5
        9 sws4322a00203.fits    -2.490    -1.453    -0.11596     0.001       0.009     0.77874     64.5
64.5
```


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

Offsets relative to an average of all input images:

```
\character Delta_File_Program = Output from deltafptcorr, version  1.00
\character Creation_Date_Time = Fri Aug 10 15:45:12 2001
\character Input_Image_List = ./testing/test_fits.list
\character Input_Source_Table_List = ./testing/test_tbl.list
\character Reference_Image = Average of all input frames
\integer Number_of_Frames = 9
\real RefFrCRVAL1 = 82.219688
\real RefFrCRVAL2 = 35.829962
\real RefFrCRPIX1 =   64.5
\real RefFrCRPIX2 =   64.5
\real RefFrCROTA2 = -0.084781
\real RefFrCDELT1 = -0.004323
\real RefFrCDELT2 = 0.004297
|Index   |Filename           |Xshift    |Yshift    |Rotation   |err_Xshift |err_Yshift |err_Rot     |Xcenter
|Ycenter |
|i       |c                  |r         |r         |r          |r          |r          |r           |r
|r       |
        1 sws4322a00206.fits  -23.916     6.290   0.00674     0.001       0.011      0.73944     64.5
64.5
        2 sws4322a00207.fits    3.682    -4.075   0.08733     0.016       0.007      1.54626     64.5
64.5
        3 sws4322a00208.fits   13.177    -5.050   0.00464     0.004       0.007      1.01716     64.5
64.5
        5 sws4322a00201.fits   -2.096    -4.427   0.00370     0.054       0.015      1.01122     64.5
64.5
        6 sws4322a00202.fits   -5.498    -9.482   0.00124     0.000       0.000      1.13697     64.5
64.5
        8 sws4322a00205.fits    0.382     1.505   0.11661     0.001       0.007      0.63790     64.5
64.5
        9 sws4322a00203.fits   -2.108    -0.002   0.00502     0.000       0.000      0.00000     64.5
64.5
```


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

7. Testing

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

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

1. Tested DELTAFPTCORR on a list of WIRE images and compared the delta-file output with image offsets computed by independent programs (e.g. the DELTAFPOINT program). Image offsets were also compared with those computed from routines in the IRAF package.

2. Tested DELTAFPTCORR on lists of simulated FITS images by updating all relevant pointing keywords to test different combinations of relative rotations, translations and map projections.

3. Tested DELTAFPTCORR on lists of simulated FITS images containing varying degrees of optical distortion to ensure that this is corrected for when transforming into sky coordinates.

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

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

6. Executed DELTAFPTCORR on non-square images.

7. Executed DELTAFPTCORR on lists of $\approx$200 FITS images to test for memory limitations.

8. Usage Examples

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

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

Without using a namelist file (with offsets relative to a **single input frame**):

```
DELTAFPTCORR –f1 image_list.txt –f2 table_list.txt –i
ref_image.fits –o delta_file.tbl –sr 1.5 –sf 3 –e 0 –g 2 –cf 2
–cx 0 –cy 0 –pf 2 –px 0.005 –py 0.005 –v
```

Without using a namelist file (with offsets relative to the **average pointing of all frames** – this is the default):

```
DELTAFPTCORR –f1 image_list.txt –f2 table_list.txt –o
delta_file.tbl –sr 1.5 –sf 3 –e 0 –cf 2 –cx 0 –cy 0 –pf 2 –px
0.005 –py 0.005 –v
```

9. Glossary

DCE	Data Collection Event
DN	Data Number
IOC	In-Orbit Checkout
IRAF	Image Reduction and Analysis Facility.
SDS	Subsystem Design Specification
SIS	Software Interface Specification
TBD	To Be Determined
TBR	To Be Resolved
WCS	World Coordinate System
WIRE	Wide-Field Infrared Explorer