Wide-field Infrared Survey Explorer

Subsystem Design Specification

A WISE Outlier Detector (AWOD)

Version 1.3, 28-July-2008

Prepared by: Frank Masci

Infrared Processing and Analysis Center
California Institute of Technology

WSDC D-D014

Concurred By:

Roc Cutri, WISE Science Data Center Manager

Tim Conrow, WISE Science Data Center Lead Architect

Frank Masci, WISE Science Data Center Cognizant Engineer/Scientist

Revision History

Date	Version	Author	Description
April 6, 2008	1.0	Frank Masci	Initial Draft
June 26, 2008	1.1	Frank Masci	Fixed bug on not closing mask FITS file using close_fits_file. This lead to overflow of buffer for number of FITS files open.
July 10, 2008	1.2	Frank Masci	Computed more robust SNR image to use in adaptive outlier avoidance near real sources.
July 12, 2008	1.3	Frank Masci	Replaced header reading function with more robust function: ffhdr2str. This properly parses standard distortion header keywords.

Table of Contents

1. INTRODUCTION

1.1 Purpose and Scope

This Subsystem Design Specification (SDS) document describes the basic requirements, assumptions, definitions, software-design details, algorithm and necessary interfaces for a module of the WISE CO-ADD subsystem. It will be used to trace incremental development, and contains sufficient detail to allow future modification or maintenance of the software by developers other than the original developer. 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 focuses on one component (module) of the CO-ADD subsystem - AWOD. Its purpose is to read in multiple image-frame exposures within a pre-defined region on the sky, re-project and interpolate them onto a common grid, and apply robust statistics to detect "out-of-bed" pixels, i.e., inconsistent measurements at the same sky location across frames in the stack. From hereon, these will be referred to as outliers. These outliers will be recorded in frame pixel masks for use downstream (e.g., source photometry). Potential outliers include cosmic rays, latents, other optical/instrumental artifacts, supernovae!, asteroids, and basically anything that has moved or varied appreciably with respect to the inertial sky (e.g., the CMB) over the observation span of all the input overlapping frames. This also includes inconsistencies due to poor frame registration, e.g., pointing errors greater than the typical size of a native pixel.

AWOD is executed prior to frame co-addition with AWAIC, which is used to generate the WISE Image Atlas. Specific design and implementation details of other modules in the CO-ADD subsystem are described in separate SDS documents (see references in §1.3 below).

1.2 Document Organization

This document is organized along the major themes of Requirements; Other Software Interfaces; Assumptions; Functional Descriptions and Dependencies; Input/Output; Algorithm Descriptions; Testing; and Major Liens.

The material contained in this document represents the current understanding of the capabilities of the major WISE systems and sub-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 subject to change.

1.3 Applicable Documents

- WISE Project Plan (Level 1 Requirements)
- WISE Science Requirements Document (Level 1.5 Requirements)

- WSDC Functional Requirements Document *WSDC D-R001* (FRD – Level 4 Requirements): *http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDC_Functional_Requirements_all.pdf*

- WSDS Functional Design Document *WSDC D-D001* (FDD)

- WSDC Software Management Plan *WSDC D-M002* (SMP): *http://web.ipac.caltech.edu/staff/roc/wise/docs/wsdc-smp-draft.pdf*

- WSDC Science Data Quality Assurance Plan *WSDC D-M004* (QAP): *http://web.ipac.caltech.edu/staff/roc/wise/docs/QA_Plan_WSDC_2007-03-01.pdf*

- Software Interface Specification (SIS) *WSDC D-I101* – Frame Processing Mask: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/InstruCal01.txt*

- Software Interface Specification (SIS) *WSDC D-I102* – Frame WCS FITS Header Keywords: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/SFPWrap01.txt*

- Frame Co-addition Critical Design Review (01/30/2008): *http://web.ipac.caltech.edu/staff/fmasci/home/wise/Co-addition_CDRJan08.pdf*

- Subsystem Design Specification (SDS) *WSDC D-D005* – AWAIC: A WISE Astronomical Image Co-adder: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/sds-wsdc-D005-awaic.pdf*

- Subsystem Design Specification (SDS) *WSDC D-D??? – Bmatch*: *Background Matcher* (in preparation).

- Subsystem Design Specification (SDS) *WSDC D-D??? – Frame Co-adder Wrapper Script* (in preparation).

1.4 Requirements

Below we summarize some requirements on the final release products that relate to the detection, flagging and recording of outlying pixels prior to co-addition. These are from the WSDC Functional Requirements Document (§1.3).

- *L4WSDC-001*: The WSDC shall produce a digital Image Atlas that combines multiple survey exposures at each position on the sky.
- *L4WSDC-021*: The images in the final WISE Image Atlas shall be re-sampled to a common pixel grid at all wavelengths.
- *L4WSDC-026*: The WSDC shall generate and archive coverage maps that show the number of independent observations that go into each pixel of the Image Atlas images in

each band. The coverage numbers shall take into account focal plan coverage and losses due to poor data quality, low responsivity and/or high noise masked pixels, and pixels lost because of cosmic rays and other transient events.

- *L4WSDC-080*: The final WISE Source Catalog shall have greater than 99.9% reliability for sources detected in at least one band with SNR>20, where the noise includes flux errors due to zodiacal foreground emission, instrumental effects, source photon statistics, and neighboring sources. This requirement shall not apply to sources that are superimposed on an identified artifact.
- *L4WSDC-084*: The WISE Image Atlas shall be constructed by combining all available science images covering the sky. This does not include image pixels rejected because of low responsivity, high dark current or read noise, transient behavior such as charged particle impacts, or scattered light due to moon proximity.

1.5 Acronyms

AWAIC	A WISE Astronomical Image Co-adder
AWOD	A WISE Outlier Detector
CPU	Central Processing Unit
CR	Cosmic Ray
CVZ	Continuous Viewing Zone
DN	Data Number
FDD	Functional Design Document
FRD	Functional Requirements Document
FITS	Flexible Image Transport System
FOV	Field of View
FPA	Focal Plane Array
FRESCO	Full RESolution CO-add
FWHM	Full Width at Half Maximum
HIRES	High Resolution
HST	Hubble Space Telescope
I/O	Input / Output
IPAC	Infrared Processing and Analysis Center
IRAC	Infra-Red Array Camera
IRAS	Infra-Red Astronomical Satellite
IRSA	NASA/IPAC Infra-Red Science Archive
MAD	Median Absolute Deviation
MCM	Maximum Correlation Method
MFPREX	Multi Frame Pointing REconstruction
MOPEX	[SSC's] MOsaicker and Point source Extractor
NEP	North Ecliptic Pole
PA	Position Angle
PRF	Point Response Function
PSF	Point Spread Function
SDS	Subsystem Design Specification

SEP	South Ecliptic Pole
SFPREX	Single Frame Pointing REconstruction
SIP	Simple Imaging Polynomial
SIS	Subsystem Interface Specification
SMP	Software Management Plan
SNR	Signal to Noise Ratio
SSC	*Spitzer* Science Center
TBD	To Be Determined
TBR	To Be Resolved
2MASS	Two Micron All Sky Survey
QA	Quality Assurance
QAP	Quality Assurance Plan
WCS	World Coordinate System
WISE	Wide-field Infrared Survey Explorer
WSDC	WISE Science Data Center
WSDS	WISE Science Data System
WST	WISE Science Team

2 OVERVIEW

WISE shall produce image data frames consisting of 1024 × 1024 pixels for bands 1, 2 and 3 with a projected size of 2.75 arcsec/pixel, and 512 × 512 pixels for band 4 with a size of 5.5 arcsec/pixel. This corresponds to image dimensions of ≈ 47 × 47 arcmin on the sky for all bands. Prior to co-adding and mosaicing, the image frames are first processed to remove instrumental signatures, their WCS refined using an astrometric catalog, and then a photometric zero-point is inserted into their headers to represent the photometric calibration. They are then ready for the multi-frame pipeline (described in the FDD). This consists of several modules, one of which is the outlier detection and flagging module: AWOD. This document describes the algorithm, philosophy and usage behind AWOD.

Inputs to AWOD are a list of FITS frame file names, a list of corresponding bad-pixel FITS image masks, and processing parameters. Interpolation onto the common grid is performed using a top-hat PRF kernel. This accentuates and localizes the outliers for optimal detection (for instance, single-spike cosmic rays). The primary outputs from AWOD are updated bit-masks containing specific values at the pixel locations of detected outliers, and optionally, an 8-bit mosaic (or map) in FITS format showing the locations of all outliers. Additional products can be generated in debug mode such as the median, robust-sigma, depth-of-coverage, and SNR mosaic. AWOD is written in ANSI/ISO C.

3 INPUT/OUTPUT SPECIFICATION

AWOD takes all of its input from the command-line, which is set up by a startup script and controlled by the WSDS pipeline executive, or, it can be set up manually and executed standalone. Prior to parsing the command-line inputs, default values for the optional input parameters are assigned. Table 1 summarizes all command-line inputs, their purpose and default assignments. An example command-line usage is given in §6.3.

Information on details of processing can be written to standard output by specifying the –v switch, and debug information that includes ancillary FITS file products can be generated by specifying the –g switch. A processing log with all I/O is written at the end. Quality assurance diagnostics and metrics will be generated by a separate module in the CO-ADD subsystem. These metrics are described in the *WSDC Science Data Quality Assurance Plan* (§1.3).

Variable name	**option**	**Description**	**Data-type / Format**	**Units**	**Default**
inp_image_list_fname	-f1	List of image frames in 32-bit floating point FITS format.	Char*256 (text file)	Null	Required input.
inp_mask_list_fname	-f2	List of bad-pixel masks to update in 32-bit integer FITS format. Only values $0 \rightarrow 2^{31}$ used	Char*256 (text file)	Null	Required input.
mosaic_size_x	-X	E-W mosaic dimension for crota2=0.	R*4 float	degrees	Required input.
mosaic_size_y	-Y	N-S mosaic dimension for crota2=0.	R*4 float	degrees	Required input.
RA_center	-R	RA of mosaic center.	R*4 float	degrees 0…360	Required input.
Dec_center	-D	Dec. of mosaic center.	R*4 float	degrees -90…+90	Required input.
mosaic_rotation	-C	Mosaic rotation in terms of crota2: +Y axis W of N.	R*4 float	degrees 0…360	Required input.
grid_pixelscale	-pa	Pixel scale of interpolation grid: same in X and Y. Must be ≤ input frame pixel scale.	R*4 float	arcsec	Required input.
num_tiles_along_x	-nx	Number of partitioning tiles along X dimension of interp. grid.	I*2 int	Null	1 (=> whole grid)
num_tiles_along_y	-ny	Number of partitioning tiles along Y dimension of interp. grid.	I*2 int	Null	1 (=> whole grid)
lower_tail_thres	-tl	Lower-tail threshold in number of sigma for outlier detection.	R*4 float	sigma	4.0
upper_tail_thres	-tu	Upper-tail threshold in number of sigma for outlier detection.	R*4 float	sigma	4.0

min_snr_to_rescale	-ts	Minimum (*median - bckgnd)/sigma* value for stack above which to rescale -tl and -tu thresholds by -r factor.	R*4 float	Null	1.0E+30
max_outin_area_thres	-ta	Maximum out/in pixel area ratio below which to use nearest-neighbor and max-overlap area weighted interpolation (improves speed).	R*4 float	Null	0.25
thres_rescale_factor	-r	Scaling factor for upper/lower-tail thresholds in stacks satisfying SNR > -ts <input>.	R*4 float	Null	1.0
sigmad_scale_factor	-s	Scaling factor for sigma_MAD estimates (for consistency with expected Gaussian sigma).	R*4 float	Null	1.0
smooth_sigmad_flag	-b	Smooth sigma_MAD images using median filter? 0=>no; 1=>yes	I*2 int	Null	0
med_filter_window	-w	Square window side length in pixels for median filter; must be odd integer.	I*2 int	grid pixels	3
out_mask_bits	-m	Mask template bit to set in input masks if outlier is detected.	I*4 unsigned int	Null	0 (=> no updating)
out_mask_mosaic	-om	Output FITS filename of 8-bit mosaic image showing outlier locations (with value 1 otherwise 0).	Char*256 (text file)	Null	No outlier map generated.
Debug Flag	-g	switch to print debug statements to stdout and generate debug FITS file products to execution directory.	Null	Null	0
Verbose Flag	-v	switch to print details of processing to stdout.	Null	Null	0

Table 1: Command-line inputs and options

When AWOD is executed with no command-line inputs, or, with a single "-help" (e.g., simply as "awod –help"), a command-line synopsis and tutorial is printed on the screen. This is reproduced below:

```
Program aWod: A WISE Outlier Detector, Version 1.3
```

```
Usage: awod
 -f1 <inp_image_list_fname>  (Required; list of images in FITS format)
 -f2 <inp_mask_list_fname>   (Required; list of bad-pixel masks to update in
                              32-bit INT FITS format; values 0 -> 2^31 used)
 -X  <mosaic_size_x>         (Required [deg]; E-W mosaic dimension
                              for crota2=0)
 -Y  <mosaic_size_y>         (Required [deg]; N-S mosaic dimension
                              for crota2=0)
 -R  <RA_center>             (Required [deg]; RA of mosaic center)
 -D  <Dec_center>            (Required [deg]; Dec. of mosaic center)
 -C  <mosaic_rotation>       (Required [deg]; in terms of crota2:
                              +Y axis W of N)
 -pa <grid_pixscale>         (Required [asec]; output interp-grid pixel scale
                              [<= input img scale])
 -nx <num_tiles_along_x>     (Optional [integer]; number of tiles along X
                              dimension of mosaic; Default=1 [whole mosaic])
 -ny <num_tiles_along_y>     (Optional [integer]; number of tiles along Y
                              dimension of mosaic; Default=1 [whole mosaic])
 -tl <lower_tail_thres>      (Optional; lower-tail threshold in number of
                              sigma for outlier detection; Default=4)
 -tu <upper_tail_thres>      (Optional; upper-tail threshold in number of
                              sigma for outlier detection; Default=4)
 -ts <min_snr_to_rescale>    (Optional; minimum "(median-bckgnd)/sigma" value
                              for stack above which to rescale <-tl> and <-tu>
                              thresholds by <-r> factor; Default=1E+30)
 -ta <max_outin_area_thres>  (Optional; maximum out/in pixel area ratio below
                              which to use nearest-neighbor and max-overlap
                              area weighted interpolation; Default=0.25)
 -r  <thres_rescale_factor>  (Optional; scaling factor for upper/lower-tail
                              thresholds in stacks satisfying <-ts>; Default=1)
 -s  <sigmad_scale_factor>   (Optional; scaling factor for sigma_MAD estimates;
                              Default=1.0)
 -b  <smooth_sigmad_flag>    (Optional; smooth sigma_MAD images using median
                              filter? 0=>no; 1=>yes; Default=0)
 -w  <median_filter_window>  (Optional; square window side length in pixels
                              for median filter; must be odd integer; Default=3)
 -m  <out_mask_bits>         (Optional [decimal]; mask template bit to set
                              for temporal outlier in input masks;
                              Default=0 => no updating)
 -om <out_mask_mosaic>       (Optional; output FITS filename of 8-bit mosaic
                              showing temporal outlier locations)
 -g                          (Optional; switch to print debug statements
                              to stdout and files [including FITS files])
 -v                          (Optional; switch to print more verbose output)
```

4 CO-ADD SUBSYSTEM OVERVIEW

Figure 1 shows the proposed major processing steps in the CO-ADD subsystem. AWOD processing is shown within the red dotted box and is expanded in Figure 3. All other modules and functions are described in separate SDS documents (see references in §1.3).

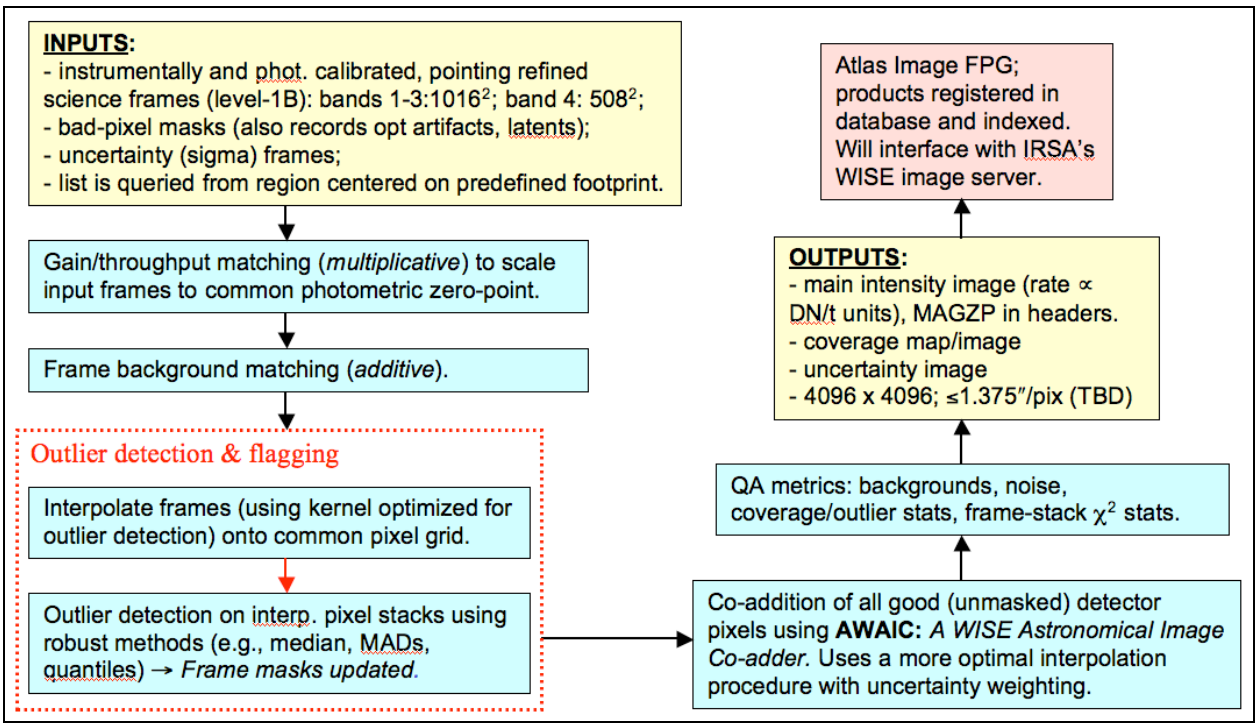

Figure 1: CO-ADD subsystem processing flow

5 BACKGROUND AND AWOD OVERVIEW

We will take advantage of the redundancy from multiple frame exposures and attempt to identify input pixels which when aligned in sky-coordinates (in a stack), have measurements that are inconsistent. These inconsistent measurements are also referred to as *temporal* outliers since they are detected amongst frames acquired at different times.

In general, the method involves first projecting and interpolating each input frame onto a *common* grid. Depending on the size of the input pixel relative to the characteristic size of the instrumental PSF, this grid may or may not have a finer pixel scale. A brute force approach is then taken whereby each pixel stack (in interpolated space) is taken in turn, and searched for outliers using robust statistics (see below).

We stress that it helps to have good sampling of the instrumental PSF for temporal outlier detection, i.e., at the Nyquist rate or better. Figure 2 shows a one-dimensional schematic for well-sampled and under-sampled cases. When well sampled (pixels shaded *green*), more detector pixels in a stack can be made to align within the span of the PSF. I.e., the profile of a star can be tracked through the stack and any pixel variations from frame-to-frame will be mostly due to variations in the PSF. Outlier detection performed within the interpolated pixel stack labeled *j* for example is likely to be reliable. However if the PSF were grossly under-sampled, a star will only be detected by the *red* pixels in Figure 2. The pixel marked "✕" contains the star, and this is in danger of being declared an outlier within the stack labeled *j*. WISE will be better than

critically sampled across all bands, and this improves the reliability of detecting outliers via stacking methods considerably.

When each frame has been projected and interpolated onto a common pixel grid, outliers can be searched for within a pixel stack *j* using robust non-parametric estimators for the first and second moments of the sample distribution. These estimators are relatively insensitive to the outliers themselves. We adopt the median, and the Median Absolute Deviation (MAD) from the median as a proxy for sigma (or dispersion):

$$\sigma_j \approx 1.4826\ median\left\{\left|p_i - median\{p_i\}\right|\right\}, \qquad \text{(Eq. 1)}$$

where p_i is the value of the i^{th} interpolated pixel within stack *j*. The factor of 1.4826 is the correction necessary to make this consistent with the standard deviation of a Normal distribution in the large sample limit. A pixel is then declared an outlier if for given "upper" (u_{thres}) and "lower" (l_{thres}) tail thresholds, its value p_i satisfies:

$$p_i > median\{p_i\} + u_{thres}\sigma_j$$

or

$$p_i < median\{p_i\} - l_{thres}\sigma_j \qquad \text{(Eq. 2)}$$

The upper and lower tail thresholds are respectively specified by the –tu and –tl command-line inputs.

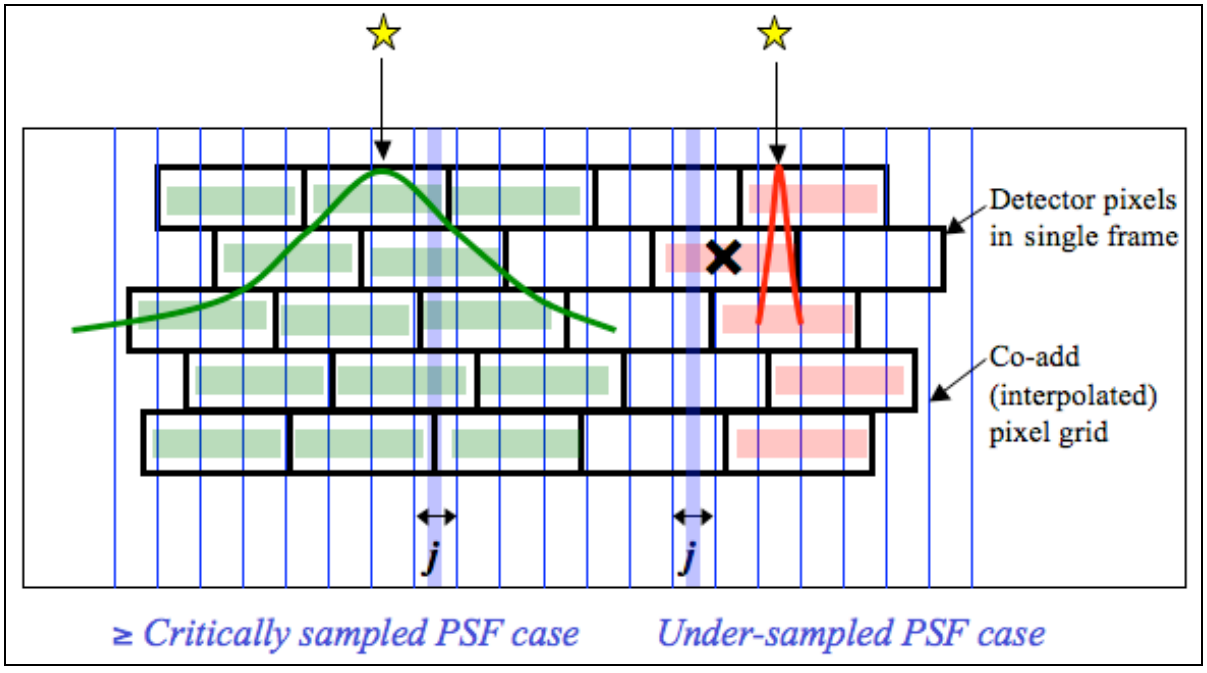

Figure 2: Schematic of the temporal method for sampled and under-sampled cases

6 AWOD PROCESSING

6.1 Assumptions and Advisories

Below we list the assumptions pertaining to the format, size and content of the image inputs. Many of these are checked internally by the program. If not satisfied, the program aborts with a message and a non-zero exit status written to *standard error*.

- The input lists of intensity and mask images must all have the same number of filenames listed in one-to-one correspondence.
- All image inputs are in FITS format.
- All input intensity images are expected to have the same native pixel scale (but $\Delta X \neq \Delta Y$ is allowed); the same projection type (CTYPE header keywords); the same NAXIS1, NAXIS2 values (with NAXIS1 ≠ NAXIS2 allowed); and the same EQUINOX.
- If optical-distortion information is available, this must be represented in the FITS headers of the intensity images using the Simple Imaging Polynomial (SIP) convention with the WCS keywords CDELT1, CDELT2, CROTA2 encoded in CD matrix format. For an example and more information, see SIS document *WSDC D-I102* referenced in §1.3.
- It is recommended that all image pixel scales (either from CDELT or CD keywords) be represented in the FITS headers to at least 8 significant figures.
- The only projection types recognized by the software are: TAN, SIN, ZEA, STG and ARC. These are specific to the fast plane-to-plane projection algorithm used.
- The maximum linear mosaic dimension supported by the image projection libraries in AWAIC is 16°. However, one is likely to run out of memory first (depending on the output pixel scales chosen) before the necessary arrays are allocated. The reason for this maximum is that for sizes greater than this, the SIN and TAN projections (the most common types) will give pixel scale distortions of >1% and >2% respectively at the extremities relative to the mosaic center. The baseline specification for WISE is to have Atlas Image grid sizes no larger than ≈ 1 square degree, so this won't be a problem. A maximum linear size of `MAXMOSAICDIM` = 16° is hard-coded in the *awaic.h* include file.
- The output grid pixel scale "–pa <*in arcsec*>" must satisfy: *minscale* ≤ pa ≤ *maxscale*, where *minscale* `= sqrt[MINAREAR*inp_image_CDELT1*inp_image_CDELT2]`; *maxscale* = `sqrt[inp_image_CDELT1*inp_image_CDELT2],` and `MINAREAR = 0.1` is currently hard-coded in the *awaic.h* include file. The `inp_image_CDELTs` are pixel scales in the input intensity images. Therefore, the grid pixel scale is constrained by the ratio of output/input pixel area.
- The input mask FITS images, if specified, are expected to have a BITPIX=32 (i.e., 32-bit signed integer format). However, only the first 31 bits (excluding the sign bit) are used in processing. Masks with BITPIX=16 or 8 or even -32 (floating point) can still be stored. Only the integer part of the float will be stored for BITPIX=-32.

- The output fatal bit-string template specification: `–m <out_mask_bits>` allows one to flag pixels affected by outliers. The bit definitions for WISE are outlined in the SIS document *WSDC D-I101* referenced in §1.3. Note that "–m 0" implies no mask updating. This setting could first be run in test mode (i.e., to get an outlier count and mosaic mask) since once the masks are updated, they cannot be easily restored. We always encourage making a backup copy of the input masks prior to updating them with AWOD.
- The minimum number of pixels in a stack (with valid data) to perform outlier flagging is hard-coded in the *awaic.h* include file as `NSAMPMIN = 5`. In other words, depth-of-coverages below this will not be searched for outliers. This limit is particular to the method used for computing robust measures of the spread (σ_{MAD}). One can set `NSAMPMIN = 4` as the absolute minimum, but we don't recommend it. For small samples, the outlier detection process becomes severely unreliable.
- The dominant memory usage in AWOD is a 3-D array for storing the stack of input image frames. Each image plane therein corresponds to the dimensions of a "common" interpolated tile grid. The *minimum* required system memory (in Gigabytes) can be computed from the following relation:

$$\mathrm{Mem} \approx 1.678 \frac{(L_x/1.564\,\mathrm{deg})(L_y/1.564\,\mathrm{deg})}{(s/2.75\ \mathrm{arcsec})^2} \frac{1}{N_x N_y} \left(\frac{N_{frames}}{100} \right) \ \mathrm{GB}, \qquad \text{(Eq. 3)}$$

where L_x and L_y are the dimensions of the master grid (command-line inputs –X and –Y), s is its desired pixel scale (input –pa), N_x and N_y are the number of tile partitions along the x and y axes of the master grid (inputs –nx and –ny), and N_{frames} is the number of input frames *assumed* to overlap with the master grid. If some of your input frames don't overlap with the master grid, memory is still allocated and hence wasted. Equation 3 has been scaled to 100 input frames, an output grid pixel size of 2.75 arcsec (optimal for WISE processing), and dimensions typical of a WISE Atlas Image (1.564×1.564 deg^2).

6.2 AWOD Processing Phases

A brief description of the methodology was outlined in §5. Here we outline the details of processing within AWOD. A processing flow is given in Figure 3, where each step is expanded upon below.

After reading all input parameters and filenames from the command-line, the first step is to check that they are within range and are valid according to the assumptions in §6.1. Defaults are then assigned to the optional unspecified parameters. The program then sets up the WCS of the master interpolation grid based on the input parameters -X, -Y, -R, -D, -C, -pa and other generic WCS keywords from the first listed input image frame.

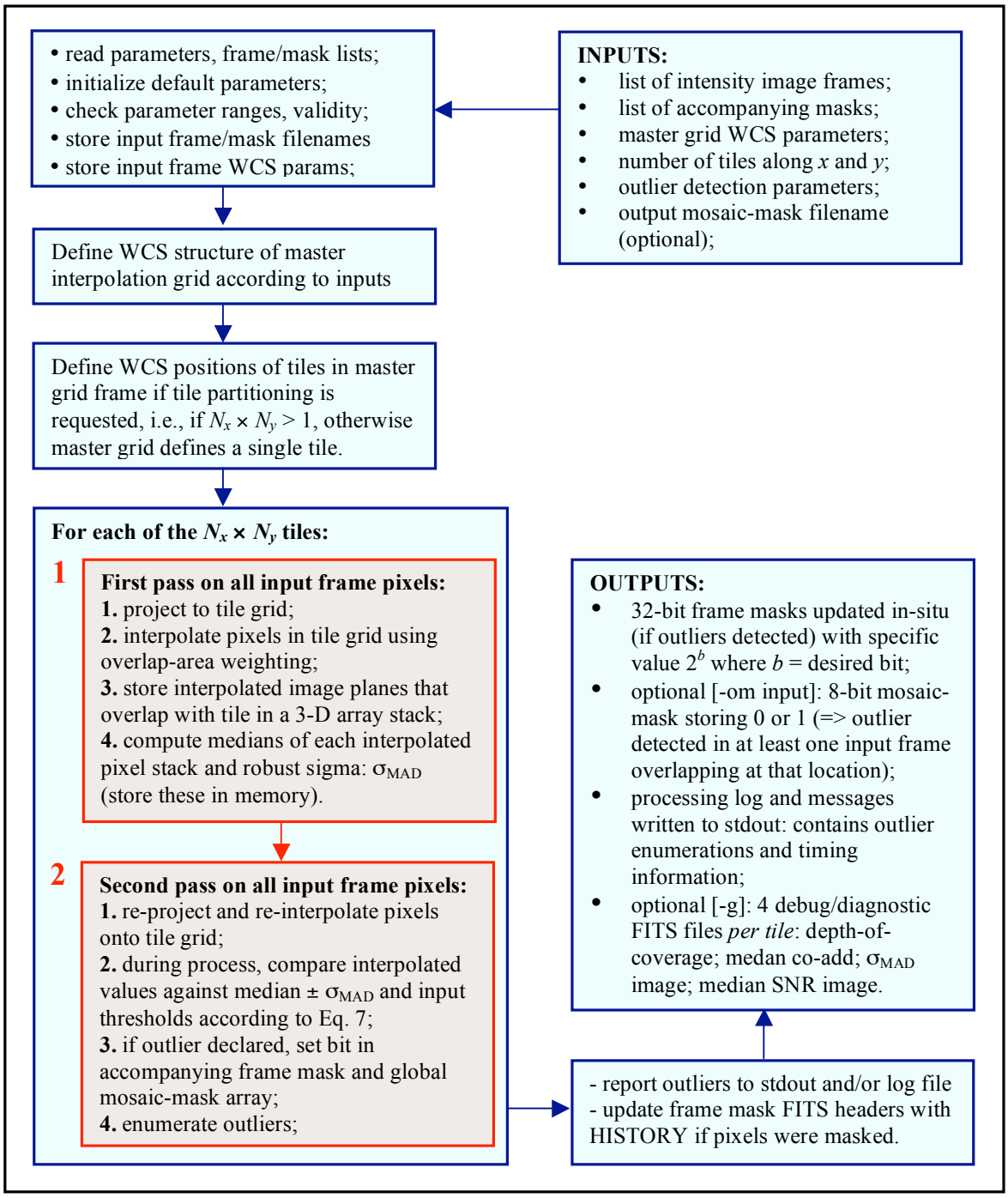

Figure 3: Processing flow in AWOD. Red boxes represent the main computational steps

The master grid can be partitioned into $N_x \times N_y$ tiles (via the –nx and –ny command-line inputs). This is to assist with memory management when storing 3-D cubes of interpolated image frames,

e.g., see Eq. 3. If tiling is desired, then each tile is assigned its own WCS with unique R.A., Dec and reference pixel coordinates and rotation (CROTA2) equal to that of the master grid. Each tile will define a "common" interpolation grid for all input frames that overlap with it. If there is ample system memory and partitioning is not desired (i.e., $N_x \times N_y = 1$), then the "common" tile grid is the master grid itself.

For each tile, or master grid for no tiling, there are two passes through all the frame pixels [see two red boxes embedded in Figure 3]. The goal of the first pass is to compute robust (≈ outlier resistant) measures of location and spread in all interpolated pixel stacks (e.g., the median and σ_{MAD}). These products will be used in the second processing pass to actually detect and flag outliers. We outline the specifics of each step.

First Pass Computations

The first pass projects frame pixels using the fast pixel-to-pixel WCS transformation library (*libtwoplane*) onto the interpolation grid. This transformation also corrects for FOV distortion using the SIP FITS representation. The interpolation uses the overlap-area weighting method to compute the flux in the output grid pixels. In essence, this method involves projecting the four corners of an input pixel directly onto the output grid (with rotation included). Input/output pixel overlap areas are then computed using a textbook algorithm for the area of a polygon, and then used as weights when summing the contribution from the pixels of an input frame with signals D_i. The signal in grid pixel j from frame k is given by:

$$f_{jk} = \frac{\sum_i a_{ikj} D_{ik}}{\sum_i a_{ikj}}, \qquad \text{(Eq. 4)}$$

where a_{ikj} is the overlap area between pixel i from input frame k with output grid pixel j (see right schematic in Figure 4).

The projection and interpolation is only performed for those frames that overlap with the tile grid of interest. This overlap pre-filtering is performed using a coarse method whereby an overlap is declared if the separation between the centers of a tile and input frame is less than the sum of the radii of circles that circumscribe the tile and frame. The interpolated planes for each overlapping frame are stored in a 3-D cube, whose memory allocation is described by Eq. 3.

The median is then computed for each interpolated pixel stack j by first sorting the pixel samples using the *heapsort* algorithm. This uses the *gsl_sort_float*() routine from the GNU Scientific Library. The median is computed as the 50th percentile of the sorted stack, and we account for even numbered samples. We also compute a robust measure of the dispersion. For speed and efficiency, we use an approximation to the actual σ_{MAD} as defined by Eq. 1 and call it the pseudo-MAD. This approximation works directly off a sorted array by selecting the appropriate indices.

We define the pseudo-MAD as follows, where $k = 0, 1, 2, 3, \ldots N$ is the sample number in a sorted pixel stack, $stack[k]$, at location j in the interpolation grid.

$$\sigma_{MADj} \approx 1.482602 \times \frac{1}{4} \times \left\{ \left(stack\left[\frac{3N}{4}\right] + stack\left[\frac{3N}{4} - 1\right] \right) - \left(stack\left[\frac{N}{4}\right] + stack\left[\frac{N}{4} + 1\right] \right) \right\}, \qquad \text{(Eq. 5)}$$

where the factor of 1.482… is the correction to ensure Normality in the large sample limit. The array indices are rounded down to the nearest integer. The depth-of-coverage N (number of samples in a stack) must satisfy $N \geq$ NSAMPMIN, where NSAMPMIN is currently hard-corded as 5 in the *awod.h* include file. Simulations have shown that Eq. 5 becomes "very noisy" when the sample size is less than 5. This adversely affects the ability to perform robust outlier detection. If $N <$ NSAMPMIN in processing, the σ_{MADj} value is set to NOMADVAL, where currently NOMADVAL = -10000.0 in *awod.h*. This is used as a flag to detect and avoid such values in downstream processing.

There is an option to smooth the σ_{MADj} image values using spatial median filtering. This is only performed if "-b 1" is specified on the command-line. The default is to perform no median filtering. If requested, the window side-length of the median filter (in odd number of grid pixels) can be specified via the –w command-line parameter. The reason for smoothing the σ_{MADj} image is to avoid using erroneous (usually underestimated) values of σ_{MADj} when the depth-of-coverage is low, i.e., N <~ 10. As mentioned above, σ_{MADj} itself is not an efficient estimator of scale for small samples. Smoothing replaces each value with a median of the neighborhood, and this is expected to be more-or-less representative of the "true dispersions" in stacks at each location. If for example, the σ_{MADj} values happen to be underestimated in some regions (due to their noisy nature), this will bias flux thresholds ($med_j \pm t\sigma_{MADj}$) towards low values and hence inadvertently flag "good samples" as outliers. There is also an option to scale the *post-filtered* σ_{MADj} values with a factor specified by the command-line parameter input: –s <*factor*> [default = 1]. This allows the user to scale the σ_{MADj} values for each stack for consistency with the expected RMS fluctuations in an image (Gaussian or otherwise). This enables one to set thresholds according to a prior measured image RMS, rather than the pseudo-MAD in pixel stacks that may be subject to under-sampling and other systematics.

AWOD includes an adaptive thresholding method in that if it encounters a pixel that's likely to be associated with a "real" signal (e.g., a source), then the upper threshold is automatically inflated by a specified amount to avoid (or reduce the incidence of) flagging such sources as outliers. To distinguish between what's real or not, we compute a "median SNR" image corresponding to the interpolated pixels j. This is computed using:

$$SNR_j \approx \frac{median_j - bckgnd}{RMS_j}, \qquad \text{(Eq. 6)}$$

where :

$$bckgnd = median\left(median_j;\ \text{for all } j \in \left\{median\ N_j\right\}\right);$$

$$RMS_j \approx RMS\left(median_j;\ \text{for all } j \in \left\{median\ N_j\right\}\right) \times \sqrt{\frac{median\ N_j}{N_j}},$$

and the base RMS :

$$RMS\left(median_j;\ \text{for all } j \in \left\{median\ N_j\right\}\right) \approx bckgnd - 16^{th}\ \text{ptile}.$$

The first term in the numerator is the median signal for pixel stack *j*, and the second term is a *global* background measure. This background is computed as the median signal over all stack medians such that their depth-of-coverage N_j is equal to the *median depth* over the entire tile grid. The denominator is an estimate of the spatial RMS fluctuation for pixel stacks at the median depth-of-coverage, relative to their median signal (the base RMS), and then appropriately rescaled to represent the pseudo-local RMS at any depth N_j (third line in Eq. 6). More precisely, the base RMS is computed using the lower tail values of the pixel distribution to avoid being biased by sources and other "spatial outliers". The 16th percentile (or more exactly, the quantile corresponding to a lower-tail probability of 0.1586) is used, analogous to the standard deviation of a Normal distribution, which can also be derived from quantiles: $\sigma = \mu - q_{0.1586} = 0.5*(q_{0.8413} - q_{0.1586})$.

Note that Eq. 6 represents a good approximation for the local SNR, assuming the background does not vary wildly. There is a lien to do this more robustly in future, i.e., compute a local median filtered image (the local smoothly varying background) within some tunable window, as well as a corresponding RMS, and then use these to compute the SNR for all grid pixels. The SNR image is used during the outlier flagging process (in the 2nd pass) to determine if the pixel contains a legitimate signal (e.g., a real source).

At the end of first pass computations, we have three image products stored in memory for a tile grid: median, pseudo-MAD (σ_{MAD}), and a SNR image. These are needed for the second pass. If the debug switch (–g) is set, these three images and the depth-of-coverage map are saved to disk as FITS images in the execution directory. These serve as valuable diagnostics.

Second Pass Computations

Given the robust metrics from the first pass, we now re-project (with distortion correction) and re-interpolate all the input frame pixels onto the tile grid again, just like in the first pass. The only difference here is that as each pixel from the k^{th} input frame is projected, it is compared to the *existing* median, σ_{MAD} and SNR values at the same pixel location *j* in the tile grid to

determine if it is an outlier. In general, an interpolated pixel from frame k, f_{jk} (Eq. 4) is declared an outlier with respect to other pixels in stack j if its value satisfies:

$$f_{jk} > median_j + u_{thres}\sigma_{MADj} \text{ and } SNR_j \le SNR_{\min} \qquad \text{(Eq. 7)}$$

or

$$f_{jk} > median_j + r * u_{thres}\sigma_{MADj} \text{ and } SNR_j > SNR_{\min}$$

or

$$f_{jk} < median_j - l_{thres}\sigma_{MADj} \text{ regardless of } SNR_j,$$

where the parameters u_{thres}, l_{thres}, r and SNR_{min} correspond to the command-line inputs: –tu, –tl, –r, and –ts respectively. This is a refinement to Eq. 2. The difference here is that the upper-tail threshold, u_{thres} is inflated by the factor r if the *median SNR* in pixel j is greater than some preset threshold (e.g., some pixel SNR of interest: 3, 5 or 10 etc… in σ units). The median is not expected to be biased by outliers (as long as they don't contaminate ≥50% of the sample), so if the median is relatively large, it's quite probable that the pixel contains signal from a real source. The factor r can be set to some arbitrarily large value to avoid any flagging whatsoever for pixels with *median SNR* above SNR_{min}.

If an input pixel from frame k leads to an outlier in the interpolated space of some tile, then a bit-value is set in it's accompanying mask image. The mask value to set is specified by the command-line input: –m <2^b *value*>, where b is the required bit. Specifying "–m 0" (also the default) means *no* mask updating. In addition to updating the input mask, a value "1" is set in an output mask array, at the appropriate location, with the same dimensions as the master grid. This is only performed if the –om <*outfname*> was provided on input. This mask array is then written to an image in FITS format with name *outfname*. This image has also been referred to as an *outlier map* in this document.

One detail specific to the second pass regards the estimate of the interpolated pixel flux f_{jk}. This can be either done "exactly" using the area-overlap method (e.g., Eq. 4), or approximately using nearest-neighbor interpolation. This approximate method was implemented for speed, and its accuracy depends on the actual ratio of output-to-input pixel area, a_j / a_i. The value at which the nearest-neighbor method kicks in can be controlled via the command-line parameter: –ta <*max out/in*>. This threshold represents the maximum ratio of output-to-input pixel area below which the nearest neighbor method is used. Above this threshold, the exact polygonal area-overlap method is used (Eq. 4). The default value for –ta is 0.25, i.e., if an output pixel occupies a quarter of an input pixel or less (by area, i.e., if $a_j / a_i \le 0.25$), then it's more probable that the input pixel overlaps an output pixel *entirely* (see left schematic in Figure 4). In this case, the output pixel that's nearest to the projected location of the input pixel is taken to "represent" the input pixel in interpolated space, and its signal is $f_j \approx (a_j / a_i) D_i$.

If however $a_j / a_i > 0.25$, it is *less likely* that an input pixel will entirely overlap an output pixel (see right schematic in Figure 4). In this case, all four corners of the input pixel need to be

projected to compute the exact overlap area enclosed by a polygon. This is a more time-consuming procedure, but it uses less system memory (i.e., there are fewer output pixels to store). If the ratio of output-to-input pixel area is >~ 0.25, then the exact overlap-area method is strongly recommended to ensure the best interpolation accuracy and proper representation of input pixels. Unless execution time is critical, we advise keeping "–ta 0.25" (the default). If the nearest neighbor method is used when the actual output/input pixel areas are > 0.25, the variance in pixel stacks will be artificially inflated. To compensate, this means we will need to inflate the thresholds for outlier detection in Eq. 7, otherwise many spurious outliers *may* be detected. It amounts to a tedious tuning problem.

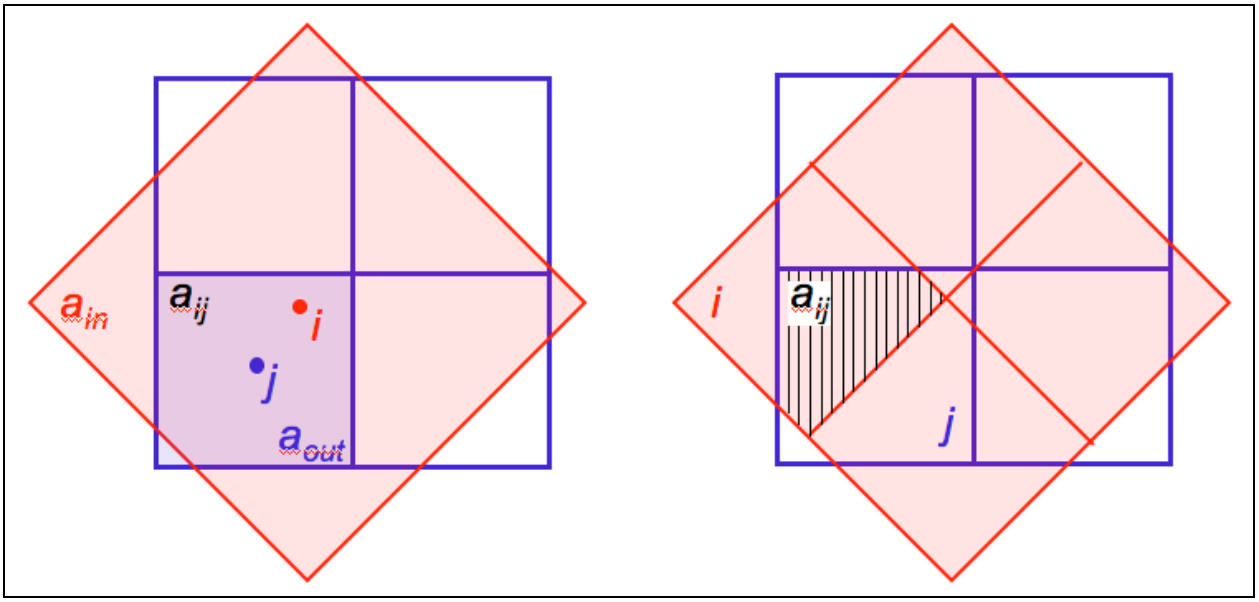

Figure 4: Left: schematic where a_{out}/a_{in} <~ 0.25 and overlap area $a_{ij} \approx a_{out}$. Here the nearest neighbor output pixel *j* to input pixel *i* is a good approximation. Right: schematic where $a_{out}/a_{in} = 1$ and the overlap area a_{ij} (shaded polygon) needs to be formally computed.

At the end of second pass processing, outlier pixels are enumerated *per frame* and reported to the standard output log. It's important to note that these enumerations may contain duplicates if partitioning of the master grid into tiles was specified (i.e., with –nx > 1 and/or –ny > 1). This is because each tile is padded with additional pixels to avoid incomplete representation of input pixels near tile boundaries after projection. Duplicates in the reported outlier enumerations will never occur if one executes in single-tile mode (i.e., –nx 1 and –ny 1), where the tile becomes the master grid itself. This assumes one has the system memory to do so, otherwise, the total number of outliers detected per frame can be inferred by counting the number of pixels in the mask images that have the relevant outlier-bit set.

6.3 Command-line Usage Example

Here is an example of a typical command-line that performs basic outlier detection and reports affected frame pixels in both the input frame masks and an output "global" mosaic-mask (outlier

map). For your interest, this run was used on WISE band-1 frames from Ned Wright's May 2008 simulation (see second bullet point in §7). An explanation of all inputs was given in Table 1 of §3.

```
% awod –f1 ImageList.txt –f2 MaskList.txt –X 1.56444 –Y 1.56444
–R 346.8 –D 27.6 –C 206.7 –pa 2.7499 –nx 1 –ny 1 –tl 10 –tu 10 –
ts 5.0 –ta 0.25 –r 300 –s 1.5 –b 1 –w 3 –m 134217728 –om
/wise/fmasci/mosaic-w1-msk.fits –v
```

To generate debug information and output diagnostic FITS files (e.g., median, σ_{MAD}, depth-of-coverage, and SNR images), append a "–g" to the above command-line.

7 EXAMPLES AND TESTING

Testing was performed primarily on simulated WISE data and *Spitzer* data (IRAC and MIPS). Analysis reports on AWOD (as well as subsequent co-addition with AWAIC) can be found on the following web pages:

- *AWOD: A WISE Outlier Detector*: summarizes the software with examples of testing on Ned Wright's Dec 2007 simulation. Also summarizes the results of a simple completeness and reliability analysis on data simulated by the author. *http://web.ipac.caltech.edu/staff/fmasci/home/wise/awod.html*

- *Mosaics from a WSDS-Processed Simulation* (*vsn 1.0*): illustrates capabilities of AWOD when integrated with AWAIC. Contains image co-adds and outlier maps generated from simulated frames pre-processed with version 1.0 of the WSDS. The raw frames are from Ned Wright's May 2008 simulation. *http://web.ipac.caltech.edu/staff/fmasci/home/wise/MidLatSimMosaics2.html*

- *South Ecliptic Pole* (*SEP*) *Mosaics*: reports testing on real data from the IRAC and MIPS detectors on *Spitzer*. AWOD was integrated with background matching and AWAIC to create mosaics of the SEP in five *Spitzer* bands: IRAC 1, 2, 3, 4 and MIPS 1. *http://web.ipac.caltech.edu/staff/fmasci/home/wise/sep_mosaics.html*

8 LIENS

Here is a summary of the current liens. All are classified as minor. These will be implemented if further testing reveals they significantly impact performance.

- More robust local background and RMS computation for use in SNR estimation (discussion in §6.2).

- Inflation of thresholds to ensure reliability when depth-of-coverage is below some minimum since outlier detection metrics become unreliable (discussion in §6.2).

- Exclude prior masked pixels (known bad pixels) from temporal outlier flagging. This will avoid duplicate (or multiple) flagging according to different conditions in input masks.

- Make memory allocation for 3D cube more efficient. E.g., only allocate memory for frames that actually overlap with a tile grid, instead of using blind frame count from input list. Currently, assumption is that input list only contains frames that overlap with the master grid (co-add footprint).

- More thorough completeness and reliability analysis using appropriate simulation where true outlier count is known, e.g., only cosmic rays, and no ambiguity from latents or moving objects.