Wide-field Infrared Survey Explorer

Subsystem Design Specification

Dynamic Flat-field Estimation (*compflat*)

Version 1.9, 29-July-2009

Prepared by: Frank Masci

Infrared Processing and Analysis Center
California Institute of Technology

WSDC D-D021

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

Stefanie Wachter, WISE Science Data Center Instrument Calibration Scientist

Deborah Padgett, WISE Science Data Center Instrument Calibration Scientist

Carl Grillmair, WISE Science Data Center Instrument Calibration Scientist

Revision History

Date	Version	Author	Description
January 12, 2009	1.0	Frank Masci	Initial Draft
January 14, 2009	1.2	Frank Masci	Implemented robust trimmed average and standard-deviation of input stack
January 15, 2009	1.3	Frank Masci	QA metrics and plots on final products
January 16, 2009	1.4	Frank Masci	Implemented more flexible 2D surface fitter for *–method* 2 and *–fltnorm* option 3.
February 10, 2009	1.5	Frank Masci	Fixed flipping bug in y axis numbering for 2d surface fit subroutine for *–fltnorm* 3.
June 6, 2009	1.6	Frank Masci	Implemented filtering of input frame list according to FDYNAFLG FITS keyword - inserted upstream by ICal. New switch: *–filt* can turn filtering on/off.
July 22, 2009	1.8	Frank Masci	Added new command-line option for *flatcal*: –rm <relative min. sigma to set>
July 28, 2009	1.9	Frank Masci	Tweaks to output QA log-histogram plots – flag zero values.

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, algorithms, QA, and necessary interfaces for the *compflat* subsystem of the WISE Science Data System (WSDS). It will be used to trace the incremental development of this subsystem, 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, characterization and maturity of operational procedures.

The purpose of *compflat* is to create responsivity (flat-field) calibration products from on-orbit frames, i.e., in a dynamic manner. It reads in a list of pre-calibrated and pre-filtered frames optimized for flat-field estimation, corrects each for possible large scale gradients, and uses either of two methods to compute a responsivity map: the gradient method (as implemented in the *flatcal* module and described in a separate SDS document; see §1.3), or the classic frame-stacking method. The flat product can be normalized using either a single median, a low-pass median filter, or a 2-D surface fit. The latter enables one to remove non-uniform (and non-responsivity related) illumination patterns. Ancillary products are an uncertainty map and a mask indicating pixels for which a responsivity estimate was unreliable, not possible, or abnormally low/high. QA metrics and plots are also generated.

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 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

- 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):
  http://web.ipac.caltech.edu/staff/roc/wise/docs/WSDS_FDD_v1.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-I137* – "Dynamic flats" QA metadata: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/QAoutput_icl06.txt*

- Instrumental Calibration Peer Review (09/28/2007): *http://web.ipac.caltech.edu/staff/fmasci/home/wise/InstruCal_PeerReview.pdf*

- Instrumental Calibration Critical Design Review (01/29/2008): *http://web.ipac.caltech.edu/staff/fmasci/home/wise/InstruCal_CDRJan08.pdf*

- Software Interface Specification (SIS) – *flatcal*: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/flatcal_specs.pdf*

- Subsystem Design Specification (SDS), *WSDC D-D012 – flatcal*: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/sds-flatcal.pdf*

- Subsystem Design Specification (SDS), *WSDC D-D018 – instruframecal*: *http://web.ipac.caltech.edu/staff/fmasci/home/wise/sds-wsdc-D018-ical.pdf*

1.4 Requirements

Below we summarize the requirements impacted by flat-field estimation in general. These are from the WSDC Functional Requirements Document (§1.3).

• *L4WSDC-042*: The WSDS Pipeline processing shall remove the instrumental signature from Level 0 image frames.
• *L4WSDC-037*: The WSDC Pipelines subsystem shall convert raw WISE science and engineering data into calibrated images and extracted source lists from which the preliminary and final WISE data products will be derived.
• *L4WSDC-039*: Within 3 days from receipt of a given data set at the WSDC all data shall be processed through the WSDS Scan/Frame pipeline which performs basic image calibration and source extraction from on images from individual orbits. The results of this processing step shall be Level 1 source extractions and image data, which are loaded into the WISE Level 1 extracted Source Working Database (L1WDB) and Image Archive allowing access by the WISE Science Team for external quality assessment.
• *L4WSDC-024*: The WSDC shall generate and maintain an archive of the calibrated, single epoch WISE images for the duration of the project for use by the Project Team. The purposes of this archive are quality assurance, transient analysis and moving object identification. Self-derived Demonstration Define duration of project.

• *L4WSDC-012*: Flux measurements in the WISE Source Catalog shall have a SNR of five or more for point sources with fluxes of 0.12, 0.16, 0.65 and 2.6 mJy at 3.3, 4.7, 12 and 23 micrometers, respectively, assuming 8 independent exposures and where the noise flux errors due to zodiacal foreground emission, instrumental effects, source photon statistics, and neighboring sources (traceable to Level-1).
• *L4WSDC-013*: The root mean square error in relative photometric accuracy in the WISE Source Catalog shall be better than 7% in each band for unsaturated point sources with SNR>100, where the noise flux errors due to zodiacal foreground emission, instrumental effects, source photon statistics, and neighboring sources. This requirement shall not apply to sources that superimposed on an identified artifact (traceable to Level-1).

1.5 Acronyms

2-D	Two Dimensional
ANSI	American National Standards Institute
FDD	Functional Design Document
FRD	Functional Requirements Document
FITS	Flexible Image Transport System
I/O	Input / Output
ISO	International Organization for Standardization
ICAL	Instrumental CALibration
IPAC	Infrared Processing and Analysis Center
JB	Jarque-Bera test statistic
LSB	Least Significant Bit
MED	Median
NaN	Not-a-Number
RMS	Root Mean Square deviation from the mean
SD	Standard Deviation
SDS	Subsystem Design Specification
SIS	Subsystem Interface Specification
SNR	Signal-to-Noise Ratio
SVG	Scalable Vector Graphics format
TBD	To Be Determined
TBR	To Be Resolved
QA	Quality Assurance
QAP	Quality Assurance Plan
WDB	Working Data Base
WISE	Wide-field Infrared Survey Explorer
WSDC	WISE Science Data Center
WSDS	WISE Science Data System

2 OVERVIEW

WISE shall downlink 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. Responsivity correction is one of the important steps in the instrumental calibration (ICAL) pipeline (see document WSDC D-D018 referenced in §1.3). In order to capture instrumental "gain-like" variations, it is important to derive calibrations that are matched as close as possible to the science frames, i.e., in a dynamic manner. The purpose of *compflat* is to compute a responsivity map given any number of input frames. It is not confined to solely use on-orbit data. *compflat* is a script written in Perl with calls to modules written in Fortran and ANSI/ISO C++.

3 INPUT/OUTPUT SPECIFICATION

3.1 Inputs

compflat takes all of its input from the command-line, which is set up by a startup wrapper 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.

Option	Description	Data-type / Format	Units	Default
-imglist	Input text file name containing list of pre-calibrated and pre-filtered 32-bit / pixel FITS intensity images	Char*256	Null	Required input
-unclist	Input text file name containing list of 32-bit / pixel FITS uncertainty (1-sigma) images	Char*256	Null	None used
-msklist	Input text file name containing list of 32-bit / pixel (long int) FITS mask images for flatcal <-method 1>	Char*256	Null	None used
-filt	Switch to filter frames from input lists using FDYNAFLG FITS keyword flag if present; independent of method	Null	Null	No filtering performed
-m_fcal	Mask template bitstring [decimal equiv] specifying bits to omit for flatcal processing <-method 1>	I*4 int	Null	0 => no flagging
-method	Responsivity estimation method: 1 = slope-fitting method (flatcal module); 2 = classic frame	I*2 int	Null	1

	stacking method			
-n_fcal	Namelist file for flatcal <-method 1>	Char*256	Null	None used
-lt_fcal	Lower-tail SNR threshold for in-frame outlier trimming for flatcal <-method 1>	R*4 float	Null	5.0
-ut_fcal	Upper-tail SNR threshold for in-frame outlier trimming for flatcal <-method 1>	R*4 float	Null	5.0
-lf_fcal	Minimum median frame signal to retain for flatcal <-method 1>	R*4 float	Image units	-9.9e+25
-hf_fcal	Maximum median frame signal to retain for flatcal <-method 1>	R*4 float	Image units	9.9e+25
-r_fcal	Switch to rescale input uncertainties to obtain reasonable chi-squares and parameter uncertainties; for flatcal <-method 1>	Null	Null	Not performed
-rm_fcal	Minimum value of relative sigma = rm_fcal*median such that if robust sigma from percentiles in residuals < this, reset sigma to this; only used if no input prior uncs provided for flatcal <-method 1>	R*4 float	Null	0.001
-nmed	Read first -nmed frames from input list to use for computing median/robust-sigma for outlier detection to support method 2; depends on available memory	I*2 int	Null	300
-lthres	Lower-tail threshold (# sigma) for outlier rejection to support method 2	R*4 float	Null	4.0
-uthres	Upper-tail threshold (# sigma) for outlier rejection to support method 2	R*4 float	Null	4.0
-fthres	Symmetric threshold (# sigma) for flagging lo/hi responsivity values in final flat and recording in output mask <-fltmsk> for method 2. Also used for QA meta-table metrics	R*4 float	Null	5.0
-normeth	Method for normalizing input frames to support method 2; 1 => single frame median; 2 => robust plane fit; 3 => none;	I*2 int	Null	3
-svbgrid	Number of partitions along an axis of final flat frame for median block-filter normalization; to support method 2	I*2 int	Null	5
-ksize	Size of Gaussian smoothing kernel as a fraction of median-	R*4 float	Null	1.5

	filter window length set by <-svbgrid>; to support method 2			
-ksig	sigma_x, sigma_y of Gaussian smoothing kernel as a fraction of <-ksize> in x, y respectively; to support method 2	R*4 float	Null	0.5
-fltnorm	Method for normalizing final flat to support method 2; 1 => single global median; 2 => median block filter over <-svbgrid> grid; 3 => 2D polynomial fit; 4 => none	I*2 int	Null	4
-order	Order of polynomial for option <-fltnorm 3>; to support meth 2	I*2 int	Null	Default=3 => quadratic fit
-outdir	Pathname for ancillary FITS file products and working directory	Char*256	Null	Required input
-archdir	Pathname for archivable products	Char*256	Null	–outdir <path>
-qadir	Pathname for output QA diagnostic files	Char*256	Null	–outdir <path>
-qameta	Filename for QA meta-data to store metrics on final flat-field products. Will be written under <-archdir>	I*2 int	Null	meta-flat.tbl
-fltprd	Output flat-field image product FITS filename	Char*256	Null	Required input
-fltunc	Output flat-field uncertainty image FITS filename	Char*256	Null	Required input
-fltmsk	Output 8-bit mask FITS filename for bad responsivity estimates	Char*256	Null	Required input
-o3_fcal	Optional QA: intercept-fit image FITS filename from flatcal <-method 1>	Char*256	Null	None generated
-o4_fcal	Optional QA: intercept-fit uncertainty image FITS filename from flatcal <-method 1>	Char*256	Null	None generated
-o5_fcal	Optional QA: co-standard deviation (between slope and intercept) image FITS filename from flatcal <-method 1>: sign[cov] sqrt[|cov|])	Char*256	Null	None generated
-o7_fcal	Optional QA: output chi-square-fit image FITS filename from flatcal <-method 1>	Char*256	Null	None generated
-o8_fcal	Optional QA: output #points-fit image FITS from flatcal <-method 1>	Char*256	Null	None generated
-o9_fcal	Optional QA: output frame medians table-file name from flatcal <-method 1>	Char*256	Null	None generated
-qa	Switch to generate QA metrics	Null	Null	0

	and plots on flat-field products			
-dbg	Switch to print debug info. to stdout	Null	Null	0
-v	Switch to increase verbosity to stdout	Null	Null	0

Table 1: Command-line inputs and options

Command-line inputs suffixed by a "_fcal" in Table 1 are exclusively used for the *flatcal* module (see document WSDC D-D012 referenced in §1.3). If *compflat* is executed with a "-help2" (e.g., as "compflat –help2"), a command-line synopsis and tutorial is printed on the screen. This is a summary of the information presented in Table 1. Execution using "–help" yields a shorter summary of the I/O.

3.2 Input List of Science Frames

We require that the input frames for computing the responsivity have been pre-calibrated from a first pass run of the *instruframecal* pipeline. These are intermediate products that have been corrected for electronic signatures, dark current, and non-linearity (see Figure 1 in document WSDC D-D018).

3.3 Frame Filtering

We need to ensure the input list is optimal for responsivity estimation. In other words, frames affected by on-orbit anneals, excessive cosmic-rays, high source density, and bright saturating sources have been removed. The first pass run of the *instruframecal* pipeline (if run in its special "dynacal" mode to save linearized products) adds a keyword FDYNAFLG with value 0 or 1. 0 implies the frame did not satisfy the filtering thresholds in and 1 implies it did, in which case new frame lists are constructed for use by *compflat*. This filtering is only performed in the *–filt* command-line switch is specified on input. Furthermore, if this switch is specified and the FDYNAFLG keyword does not exist in an input frame header, a warning is issued and it assumed that the frame is *un*usable for flat-field estimation purposes.

Below we list the metrics and their (unrealistic) thresholds used for the filtering in *instruframecal* at the time of writing. These are from a calibration file, e.g., *simcal-meta.tbl* stored in the WSDS operations archive.

```
ical:fdtanneal  0.0            Closest time since last anneal [sec]
ical:ffpatemp   34.0           Maximum FPA Temperature [K]
ical:fminelat   -90.0          Minimum ecliptic latitude [deg]
ical:fmaxelat   90.0           Maximum ecliptic latitude [deg]
ical:fminglat   -90.0          Minimum galactic latitude [deg]
ical:fmaxglat   90.0           Maximum galactic latitude [deg]
ical:fnspikelo  258064         Max. number of low-tail outliers (glitches)
ical:fnspikehi  258064         Max. number of high-tail outliers (glitches)
ical:fnumsat    258064         Max. number of saturated pixels (at any SUR sample)
ical:fminillum  0.0            Minimum median illumination/pixel [DEB DN]
ical:fmaxillum  1.0E+5         Maximum median illumination/pixel [DEB DN]
ical:fmaxrmsmad 1.0E+5         Maximum tolerable pixel RMS (from intSigMADMED)
```

```
ical:fmaxrmspti 1.0E+5          Maximum tolerable pixel RMS (from intMed16ptile)
ical:fmaxrat    5               Maximum tolerable ratio: (84%tile-Med)/(Med-16%tile)
```

The actual frame values are then compared to the above thresholds using the following conditional in *instruframecal*. If all are satisfied, FDYNAFLG is set to 1.

```
if( ($satpixcount <= $fnumsat) &&
    ($med >= $fminillum) &&
    ($med <= $fmaxillum) &&
    ($rmsmad <= $fmaxrmsmad) &&
    ($hsixteenptile <= $fmaxrmspti) &&
    ($ratio_ptilesmed <= $fmaxrat) &&
    ($numlo <= $fnspikelo) &&
    ($numhi <= $fnspikehi) &&
    ($dtanneal >= $fdtanneal) &&
    ($elat0 >= $fminelat) &&
    ($elat0 <= $fmaxelat) &&
    ($glat0 >= $fminglat) &&
    ($glat0 <= $fmaxglat) &&
    ($fptemp <= $ffpatemp) ) {

    $fdynaflg = 1;
}
```

3.4 Output Files

There are three FITS image products generated by *compflat*: the main normalized flat-field calibration containing relative responsivity estimates, accompanying uncertainty image, and a mask image flagging abnormal values. The uncertainty image contains a 1-sigma error estimate in the responsivity for every pixel. The method 1 (*flatcal*) uncertainty computation requires an input list of uncertainty frames to be specified (e.g., priors propagated from pre-processing in the *instruframecal* pipeline), while for method 2 (the stacking method), the output uncertainties are computed *a posteriori* from the data. Algorithms for method 1 are described in a separate SDS document: WSDC D-D012, while algorithms for method 2 are described in §4. Quality Assurance (QA) diagnostic files are generated if the –qa switch is specified. These consist of (i) metrics written to a table specified by –metatbl and archived under –archdir, and (ii) plots in SVG format written to the path specified by –qadir. These are discussed in §5.

4 COMPFLAT PROCESSING AND ALGORITHMS

4.1 Overview of Processing Steps

Figure 1 captures the processing flow in *compflat*. The blue boxes are for method 1 (execution of the *flatcal* module), the red boxes show the steps for method 2 (the frame-stack method), and the yellow boxes are generic to the two methods. The algorithm for method 2 is described below. The method 1 algorithm is described in a separate SDS document: WSDC D-D012 (see §1.3).

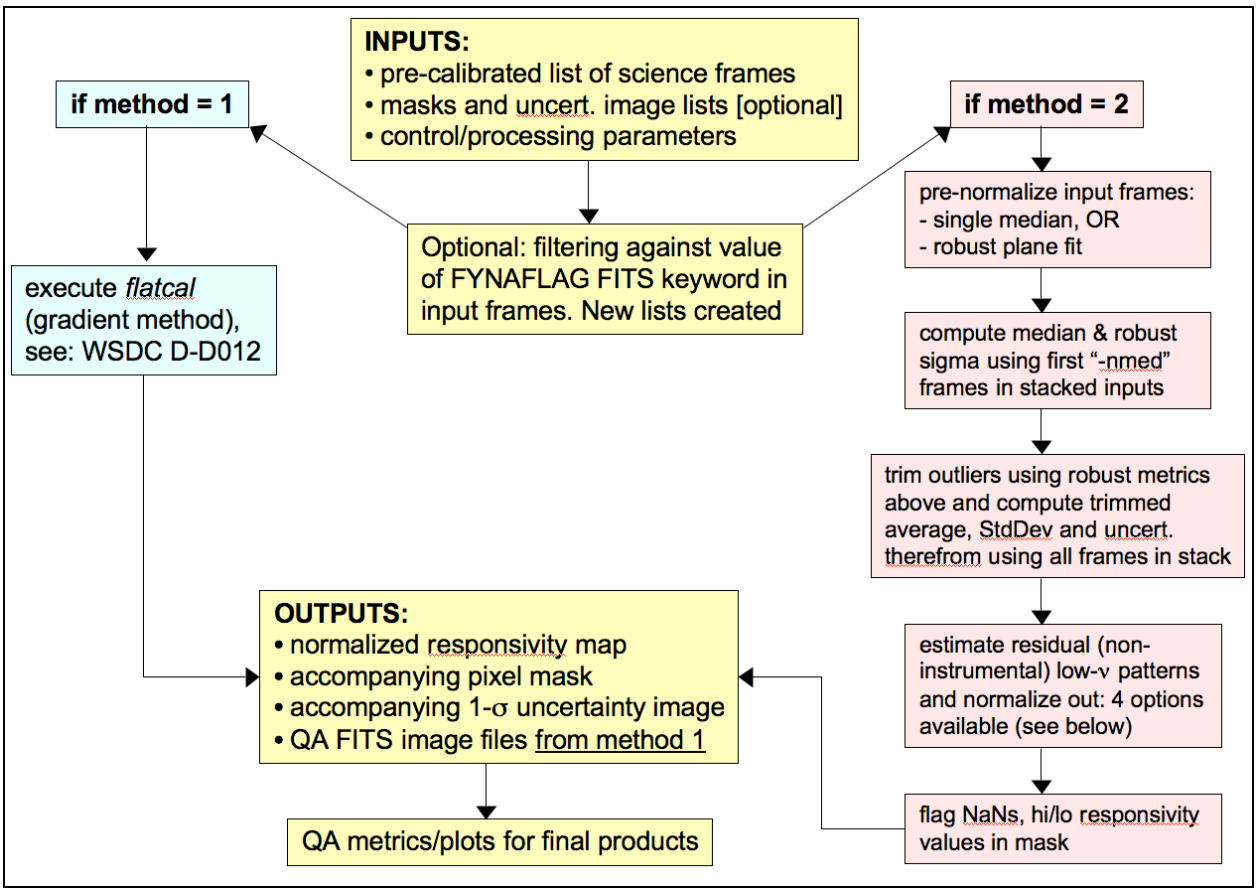

Figure 1: processing flow in *compflat* script

4.2 Frame Pre-normalization

The first step in method 2 is to "regularize" the science frame inputs, or place them on equal footing (around unity) in terms of background levels and gradients. This step is optional. Its purpose is to provide a better handle on variations between the frames in a stack. In other words, the final uncertainty as estimated from the stacked standard deviation will be less subject to background level and gradient differences between the frames. The bulk of this variation is expected to be non-responsivity related in nature.

Two pre-normalization methods are available: "–normeth 1" divides each frame by its respective median pixel value (a constant) and "–normeth 2" computes a robust planar fit (with slope) to each frame. By "robust" we mean relatively immune to the presence of bright sources and extended structure. The fit is computed using a piecewise non-parametric method whose goal is to capture the underlying background level (with possible slope). The planar fit is then divided into each respective frame. The –normeth 2 option writes out FITS images of the planar background fits into the directory specified by –outdir with name "<*infile*>*_bckgnd.fits*". The

normalized (divided) frames are written out with name "<*infile*>_*norm.fits*". Note that –normeth 3 (the default) implies no pre-normalization.

4.3 Robust Metrics for Outlier Trimming

The first "–nmed" frames from the input list (pre-normalized or not) are then used to compute *robust* (≈outlier-resistant) 1st and 2nd moments of the pixel stacks. These are the median m and a robust measure of σ respectively, where $\sigma \approx 0.5\,[q_{0.84} - q_{0.16}]$ and the q_x are quantiles of each pixel-stack distribution. The default for –nmed is 300. The reason why we impose a limit is because this step can consume a lot of memory, i.e., all frames need to be stored to efficiently compute distribution quantiles. The first –nmed frames should provide an approximate (representative) measure of m and σ for the whole input ensemble, assuming of course the input frames are randomly listed in terms signal levels. If the number of input frames is less than that specified by –nmed, then all frames are used in the stack to compute m and σ. The median of the stack is written to a file in FITS format under –outdir with name "<*fltprd*>_*median.fits*".

4.4 Trimmed Average and Standard-Deviation Computation

The robust metrics above are then used to compute an outlier-trimmed average and standard-deviation for each pixel-stack using all the input frames. So, why not use the median computed above instead of a trimmed average? There are two reasons: first, not all the input frames may have been used to compute the median due to memory limitations. An average can be built up incrementally using minimal resources, regardless of the number of inputs. Second, an average is less noisier than a median, in fact it's ~25% less noisier in terms of its RMS with respect to "truth" for the same number of inputs. The procedure is as follows:

1. Compute the minimum and maximum pixel threshold values in each stack beyond which a pixel should be declared an outlier. This uses the above robust metrics (m and σ) and the user-specified thresholds –lthres <T_L> and –uthres <T_U>:

$$p_{\min} = m - T_L\sigma$$
$$p_{\max} = m + T_U\sigma \qquad \text{(Eq. 1)}$$

2. For each FITS frame read from disk, loop through the pixels, and if a pixel with value p_i satisfies: $p_i < p_{min}$ OR $p_i > p_{max}$ OR $p_i \neq p_i$ (i.e., a NaN), then that pixel is flagged an outlier and its value is reset to 0. In the process, two internal arrays are incremented to store (i) the cumulative sum of the "good" pixel values, and (ii) the cumulative number (or stack-depth) of the non-outlier pixels at each location.

3. When all frames have been read and outliers flagged, the cumulative-value and stack-depth arrays (with outliers effectively squeezed out) are combined to compute an average for each pixel.

4. Steps 2 and 3 are repeated to compute the trimmed standard-deviation image, using the trimmed average as input. The trimmed standard-deviation image is then converted to an uncertainty image by dividing by the square-root of the stack-depth image.

At the end of this step, the following ancillary products are written under –outdir: an image of the trimmed average: *<fltprd>_pre.fits*; an image of the number of samples used in each stack, i.e., the stack-depth: *<fltprd>_Nsamp.fits*; and the uncertainty image: *<fltunc>_pre.fits*. The suffix "*_pre*" indicates that these are "pre-normalized" intermediate responsivity products. They are written out for debug purposes.

4.5 Post Normalization

The trimmed average and uncertainty image from the previous step are then normalized in order to estimate pixel-to-pixel responsivities *relative to unity*. It's important that the 1st moment of the final pixel responsivity distribution does not deviate systematically from unity and that the 2nd moment (variance) is minimal, i.e., there should be no contamination by extraneous *non-responsivity* related structure. Otherwise, biases (or flat-fielding residuals) will be introduced into the science frames when calibrations are applied.

Low-frequency, *non-responsivity* related patterns may not have been fully eliminated after the trimmed averaging described above. These may be due to other instrumental effects or structure in the astrophysical background. They need to be estimated and normalized out if possible. The key is determining whether a low-frequency pattern is responsivity related or not. If so, it needs to be retained in the final responsivity map. Possible tests include (i) examining if a suspect pattern is persistent in time (although transient behavior is still possible), or (ii) if inclusion of the low-frequency pattern (transient or otherwise) improves photometric accuracy (e.g., through repeatability metrics). Four methods are available in *compflat*, specifiable via –fltnorm <1,2,3 or 4> with default = 4. To ensure the uncertainties remain statistically compatible with the final responsivities, these methods also normalize (or recale) the accompanying uncertainty image. The methods are as follows:

1. The first option is the simplest and divides through by the median of the trimmed average image. In other words, a constant value. This is useful if no extraneous low-frequency structure exists.

2. The second option computes a median-block filtered image using an $n \times n$ grid with post-Gaussian smoothing. This yields an image with high-frequency responsivity variations smoothed out and any extraneous low-frequency structure retained. The number of partitions in the square grid can be set by the –svbgrid *<n>* specification (default = 5). The Gaussian smoothing kernel parameters can be set by –ksize and –ksig, the linear size and σ of the kernel respectively, expressed as a multiple of the median filter block length *n*. The smoothed image is written to a FITS file under –outdir named *<fltprd>_pre_bckgnd.fits*. The trimmed average image is then normalized (divided) by this image to estimate the relative responsivities at the desired frequency. It's important

that the filter block size *n* is not made too small. Otherwise, such that *real* (low-frequency) responsivity structure will be 'explained away' after normalization.

3. The third option involves performing a parametric 2-D polynomial surface fit to the trimmed average image. This is performed by calling the *PowFit2D* module. It uses a standard linear-least squares routine to estimate the coefficients A_{ij} of a generic 2-D function:

$$f(x,y) = \sum_{i=0}^{N} \sum_{j=0}^{i} A_{ij} x^{i-j} y^{j},$$

where $f(x,y)$ are the responsivity values, and N is the maximum order with $i + j \leq N$. This is specified by the –order <*N*> parameter with default = 3, implying a cubic surface fit. The total number of coefficient terms is $(N + 1)(N + 2)/2$. For details on the underlying fitting algorithm, see: *http://web.ipac.caltech.edu/staff/fmasci/home/statistics_refs/surface-fit.pdf*. The surface fit is written to a FITS file under –outdir named <*fltprd*>*_pre_bckgnd.fits*. The surface fit is then divided into the trimmed average image to estimate the relative responsivities at the desired frequency. It's important that the order N is not set too high. Otherwise, *real* (low-frequency) responsivity structure will be 'explained away' after normalization. Simplest is best, and we expect that $N \leq 4$ will serve most cases.

4. The fourth option (the default) performs no post-normalization. This is generally not recommended. It only makes sense to *not* post-normalize if (i) all the input frames were pre-normalized (or regularized – see §4.2), and (ii) one is sure that the trimmed average image contains no extraneous structure or bias in the pixel distribution that significantly deviates from unity. If so, then it could benefit from either of the above three normalization methods.

4.6 Responsivity Mask

The end of processing for method 2 consists of creating a bit-mask to record those pixels whose responsivity value is abnormally high or low with respect to the full distribution. NaN's are also flagged. The procedure involves flagging pixel values p_i in the responsivity map which satisfy:

$$p_i < m - T_F\sigma \quad \text{OR} \quad p_i > m + T_F\sigma,$$

where m is the median over all responsivity values and σ is a robust (≈outlier resistant) measure of the spread: $\sigma \approx 0.5\,[q_{0.84} - q_{0.16}]$, where the q_x are distribution quantiles. T_F is the SNR flagging threshold (parameter –fthres <default=5>) and assumed to be symmetric for the low and high tails. Pixels satisfying the above conditions are encoded in an 8-bit FITS mask (–fltmsk <*outfile*>) with values defined as follows:

Bit #	Decimal Equivalent	Condition
0	1	Value is NaN
1	2	Low responsivity (including dead)
2	4	High responsivity (hot)

Table 2: Bit-mask assignments written to –fltmsk <*outfile*>

All output products are then assembled, renamed according to their desired output names specified by –fltprd; –fltunc; and –fltmsk, and propagated to the QA step (see below).

5 QUALITY ASSURANCE OUTPUTS

5.1 Metrics

Quality Assurance diagnostics are only performed if the –qa switch is specified. This is generic to the two flat-field estimation methods and is only performed on the primary responsivity and uncertainty output products. Below we list the responsivity (*flt*) and uncertainty (*unc*) metrics written to an output meta-data table in IPAC format (–qameta <*outfile*> under directory: –archdir). An example with all columns can be found in WSDC D-I137 (referenced in §1.3). Below we only show the metric names and definitions.

```
\ WISE QA metadata for flat-field calibration products
\ Generated by compflat, v.1.3 on 2009-01-16 at 16:42:58
\ Definitions of metric identifiers:
\ flt: flat-field (responsivity) values
\ unc: flat-field uncertainty values
\ Metric units pertain to dimensionless responsivity values
\ FITS file products represented:
\ 1: /wise/fmasci/testdata/flatcal1/outputs/flat-w1-est.fits
\ 2: /wise/fmasci/testdata/flatcal1/outputs/flat-w1-unc.fits
\
|name                      |comment
\
\ Flat-field responsivity metrics
\
 flatf:flt:numframes       Number of input frames used
 flatf:flt:NumNaN          Number of NaN pixels in responsivity map
 flatf:flt:Min             flt pixels: Minimum pixel value in flat
 flatf:flt:Max             flt pixels: Maximum pixel value in flat
 flatf:flt:Mean            flt pixels: Mean pixel value in flat
 flatf:flt:Median          flt pixels: Median pixel value in flat
 flatf:flt:StdDev          flt pixels: Standard (RMS) Deviation from mean
                                       (unbiased estimate) in flat
 flatf:flt:Mode            flt pixels: Mode pixel value (fuzzy) in flat
 flatf:flt:Med16ptile      flt pixels: Robust sigma from Median - 16%-tile
 flatf:flt:84-16ptile      flt pixels: Robust sigma from [84%-tile - 16%-tile]/2
 flatf:flt:Skewness        flt pixels: Sample skewness
 flatf:flt:Kurtosis        flt pixels: Sample kurtosis
 flatf:flt:JBCoeff         flt pixels: Jarque-Bera normality test coefficient;
                                       larger JBCoeff => more non-normal
 flatf:flt:Locount         flt pixels: Number of low responsivity pixels
                                       [< med - 5*sigma = 0.999768435955048]
 flatf:flt:Hicount         flt pixels: Number of high responsivity pixels
                                       [> med + 5*sigma = 1.00024104118347]
\
```

```
\ Flat-field uncertainty metrics
\
 flatf:unc:Min             unc pixels: Minimum pixel value in flat uncert
 flatf:unc:Max             unc pixels: Maximum pixel value in flat uncert
 flatf:unc:Mean            unc pixels: Mean pixel value in flat uncert
 flatf:unc:Median          unc pixels: Median pixel value in flat uncert
 flatf:unc:MeanAccu        unc pixels: Mean accuracy in responsivity;
                                       = 100*<uncert/responsivty> %
 flatf:unc:MedianAccu      unc pixels: Median accuracy in responsivity;
                                       = 100*MED[uncert/responsivity] %
```

These metrics will enable one to:

- Perform routine health/sanity checks on instrumental gain variations in general;
- Select appropriate calibration products for the instrumental calibration pipeline;
- Tune parameters for optimum generation of calibrations from on-orbit data.

Below are descriptions of the not-so-obvious metrics and their purpose.

Mode

This metric represents an estimate of the most frequently occurring responsivity value. It represents a "fuzzy" measure since it is based on an approximate binning method and requires a sample size of $>\sim 500$. This is usually always satisfied when all pixels of a WISE frame are used, but the estimate will be highly inaccurate for small samples. The method first partitions the histogram of all image pixel values into 10 equal area bins (or "10%-tiles"); it then takes the bin with the smallest width as the one most probable to contain the mode since this bin is located near the peak of the histogram; the median of the data in this bin is then used as an estimate of the mode.

Med16ptile & 84-16ptile

These represent robust ($\approx$ outlier/bad-pixel resistant) estimates of sigma using quantile differences in the responsivity distribution: $q_{0.5} - q_{0.16}$ and $0.5[q_{0.84} - q_{0.16}]$ respectively. For normally distributed data, these should be approximately equal to the standard deviation. A significant discrepancy between the two indicates the distribution is skewed, i.e., there is an asymmetry about the central moment. This may be caused by for example, outliers (including sources) in the high tail that were not properly filtered during the estimation procedure.

Skewness, Kurtosis & the Jarque-Bera Normality Test

The sample *skewness* is defined:

$$s = \frac{\sum_{i=1}^{N} (p_i - \mu)^3}{(N-1)\sigma_{SD}^3},$$

where μ and σ_{SD} are the sample mean and standard-deviation respectively. The skewness provides a relative indicator of the presence of spurious sources/outliers in the responsivity map, i.e., that were not properly filtered during the estimation procedure. A perfectly symmetric distribution will have $s \approx 0$.

The sample *kurtosis*, in-excess of that of a normal distribution, 3, is defined:

$$k = \frac{\sum_{i=1}^{N}(p_i - \mu)^4}{(N-1)\sigma_{SD}^4} - 3.$$

The kurtosis is included here since it can be used together with the skewness to define a statistic to test for normality of the responsivity distribution. This is called the Jarque-Bera test statistic and defined:

$$JB = \frac{N}{6}\left[s^2 + \frac{k^2}{4}\right],$$

where s and k were defined above. For details, see: *http://en.wikipedia.org/wiki/Jarque-Bera_test* The *JB* statistic can be used to test the null-hypothessis that the data are normally distributed with skewness and kurtosis excess = 0. This test is not formally performed here, we only use the *JB* statistic as a *relative* measure. Large values (relative to measures that are redeemed plausible from analysis) indicate a very "abnormal" responsivity distribution and will need to be investigated. Also, there is no reason to believe that the underlying population of pixel responsivities (from which the data were drawn) should follow a perfect normal distribution. It should be close, and the *JBCoeff* statistic in the QA meta-table will still provide a useful metric to weed out the bad cases.

Locount & Hicount

These represent the number of *low* and *high* responsivity pixels flagged in exactly the same manner when creating the responsivity mask. See description in §4.6.

MeanAccu & MedianAccu

One may be interested in the overall accuracy of a flat-field calibration product. We quantify this using the computed uncertainties. Recall that these uncertainties are *a posteriori* estimates computed using the input data, and not from a prior error model. This removes any possible incompatibility with the statistical distribution of the input data (i.e., we expect reduced $\chi^2 \approx 1$ across all pixels). We quantify the overall accuracy using the mean and median relative percentage error (or noise-to-signal) in the responsivity f across all pixels:

$$\text{MeanAccu} = 100\,\frac{1}{N}\sum_{i=1}^{N}\frac{\sigma(f_i)}{f_i}\ \ \%$$

$$\text{MedianAccu} = 100\ median\left\{\frac{\sigma(f_i)}{f_i}\right\}\ \ \%$$

5.2 Plots

Two diagnostic plots are generated as part of the QA step. These are in SVG format and written to the directory specified by the –qadir command-line input. The generic names are as follows:

- **<fltprd>*hist*.svg :** histogram of the final responsivity values: f_i.
- **<fltunc>*hist*.svg :** histogram of the relative percentage error: $100[\sigma(f_i)/f_i]$

Examples of these are shown below. **Note: these are based on totally unrealistic data and portray no information on current performance of the WISE detectors.**

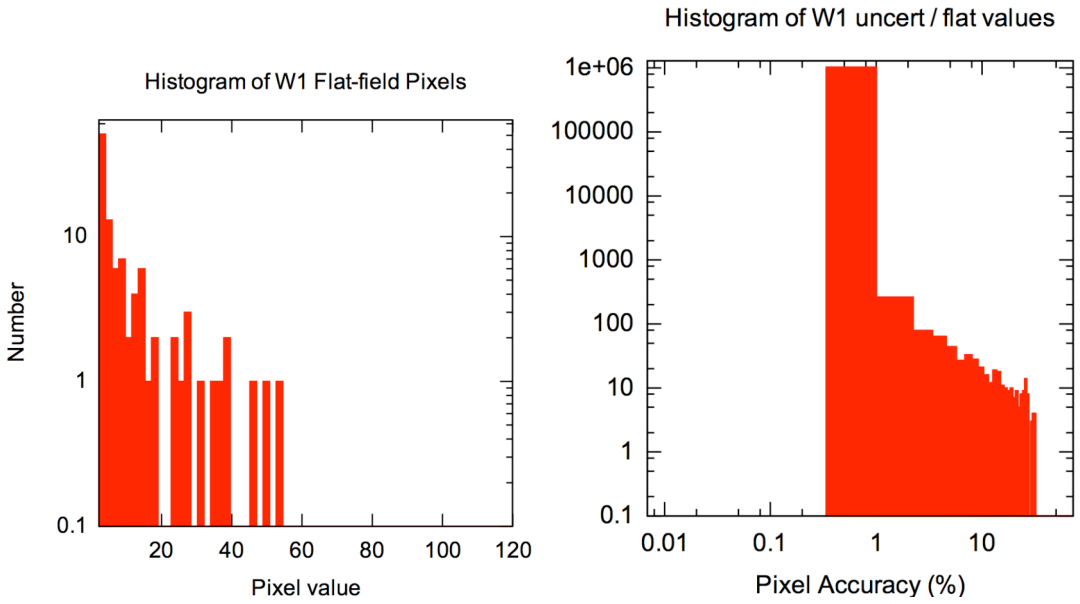

Figure 2: Example histograms based on simulated data (see §5.2). The strong 'high' tails are indicative of sources that were not properly filtered in the flat estimation process

6 USAGE EXAMPLE

The command-line example below executes *compflat* in a C-shell script using method 2 (the stacking method). All parameters have been optimized to a WISE band-1 simulation. Method 1 (the gradient method) can be invoked on the same input data by setting "–method 1" and keeping

the options suffixed with "*_fcal*" as they appear here. See the *flatcal* SDS (WSDC-D-D012 in §1.3) for details on method 1.

```
#! /bin/tcsh –f

set inpdir = /wise-ops/01/wise/fmasci/flatcal1

compflat -imglist $inpdir/ImageList.txt \
 #            -unclist $inpdir/UncertList.txt \
            -msklist $inpdir/MaskList.txt \
            -filt \
            -m_fcal 4 \
# method 1 => flatcal; method 2 => plain stacking: \
            -method 2 \
            -n_fcal  flatcal.nl \
            -lt_fcal 3.5 \
            -ut_fcal 3.5 \
            -lf_fcal -9.9e25 \
            -hf_fcal  9.9e25 \
#             -r_fcal \
            -rm_fcal 0.0001 \
# next ten params are for method 2: \
            -nmed 255 \
            -lthres 3.0 \
            -uthres 3.0 \
            -fthres 5.0 \
            -normeth 2 \
            -fltnorm 3 \
# following param is for fltnorm = 3: \
            -order 2 \
# following three params are for fltnorm = 2: \
            -svbgrid 5 \
            -ksize 1.5 \
            -ksig 0.5 \
            -outdir $inpdir/outputs \
            -qadir $inpdir/qa \
            -archdir $inpdir/meta \
            -qa \
            -qameta meta-flat.tbl \
            -fltprd $inpdir/outputs/flat-w1-est.fits \
            -fltunc $inpdir/outputs/flat-w1-unc.fits \
            -fltmsk $inpdir/outputs/flat-w1-msk.fits \
            -o3_fcal flatintcpt-w1-est.fits \
            -o4_fcal flatintcpt-w1-unc.fits \
            -o5_fcal flat-w1-cov.fits \
            -o7_fcal flat-w1-chi.fits \
            -o8_fcal flat-w1-num.fits \
            -o9_fcal flat-w1-frmeds.tbl \
            -dbg \
            -v
```

7 TESTING

The *compflat* script has been tested on simulated data-frames provided by Ned Wright. These were further modified to simulate varying background levels and gradients. This data is not of sufficient quality to compare the accuracy achieved between the gradient and stacking methods. Better simulations are available at the time of writing and testing of both flat-fielding methods is currently underway.

8 LIENS

- Low priority: use a more memory-efficient stack-medianing algorithm, i.e., that uses all the input frames instead of a subset when determining robust metrics for outlier detection.

Acknowledgments

The author is indebted to the WISE Calibration and QA Scientists, and John Fowler for illuminating and philosophical discussions. The author also thanks Roc Cutri for guidance.