Wide-field Infrared Survey Explorer (WISE)

Straw-man Subsystem Design Specification: WISE Artifact Identification (ARTID)

Draft Version 0.2

28-May-2008

Prepared by: Rosemary Alles

Infrared Processing and Analysis Center
California Institute of Technology

WSDC D-D009

Approved By:

Ned Wright, WISE Principal Investigator

Donald Royer, WISE Mission Operations Center Manager

Roc Cutri, WISE Science Data Center Manager

[Other Appropriate Names], WISE Science Data Center [Title]

[Other Appropriate Names], WISE Science Data Center [Title]

Revision History

Date	**Version**	**Author**	**Description**
05/28/08	Draft 0.1	Rosemary Alles	Initial Draft
06/01/08	Draft 0.2	Rosemary Alles	Modified per comments and suggestions from Conrow and Cutri.

Table of Contents

1 Introduction

1.1 Document Scope

This document contains a straw-man proposal for the design and –eventual- implementation of the pipeline subsystem for **Artifact Identification** within the **WISE** frame set pipeline. It draws from the document ***2MASS Data Processing*** (***Cutri and Evans – is this the same as MAPCOR SDS?***).

At present the document's scope does not extend to address the scan pipeline or co-add factors.

The **W**ise **A**rtifact **I**dentification subsystem will henceforth be referred to as **ARTID**.

To ensure the implementation of a reliable, efficient and complete subsystem, the following design criteria are addressed:

1. Interface control (specifically, what design criteria must be met in order to satisfy the requirements of interface compatibility with WISE subsystems interacting with **ARTID**)
2. Requirements/specifications within **ARTID** (specifically, what do we want to see coming out of the subsystem? Besides the obvious requirement of a table of sources flagged as effected by signatory artifacts per source per band.)
3. Artifact identification, characterization and flagging methods.
4. Optimization
5. Straw-man proposal for an initial prototype.
6. Other

1.2 Applicable Documents

This straw-man proposal conforms to the specifications in the following project documents:

1. TBD
2. TBD

1.3 Acronyms

Acronym	Description
ARTID	Artifact Identification (and flagging) for the WISE pipeline
WPHOT	Wise photometry for the pipeline
cc_map	Confusion and contamination map relating to artifacts
cc_flag	Confusion and contamination flag relating to artifacts

[Incomplete]

2 Summary of ARTID functional requirements (per frame set/per band)

Given a table of sources identified and categorized by the **Wise Photometry (WPHOT)** pipeline subsystem, the **ARTID** subsystem is required to fulfill the following:

1. **ARTID** will identify sources (*Parent Sources*) contributing to optical and detector related artifacts.
2. **ARTID** will flag each of the sources in the table contaminated by such artifacts -both real and spurious- through the use of predetermined models (TBD) and characterization (TBD). This flag will be termed the **confusion and contamination flag/map (cc_flag/cc_map)**.
3. The flag in #2 will specify the type(s) of artifact and its category (i.e. whether real or spurious).
4. **ARTID** will update the table with the ID of the *Parents Source* causing contamination of each of the affected sources, exempting spurious sources.
5. **ARTID** will provide utilities and tools to “unwrap” the flag in #2 and “other” bundled data in order to clearly and simply identify:
    a. The one/more artifact(s) affecting a particular source.
    b. Its category: real or spurious.
    c. The parent source causing the contamination.
6. **ARTID will not mitigate** (specifically it will not purge artifacts from) WISE images.
7. Other

3 Interface Control

1. The **ARTID** system will be triggered by a (TBD) preceding WISE subsystem, or alternately by an administrative/executive system during frame-set pipeline processing.
2. The **ARTID** system may (also) be launched independent of the WISE frame-set pipeline, specifically during testing.
3. The **ARTID** system will access a predetermined (TBD) table from the source detection phase of the WISE pipeline in order to assemble frame-set related source lists for artifact detection and flagging.
4. The table (resident in a specified location) will meet all of requirements set by the sub system chronologically preceding the **ARTID** subsystem within the frame-set pipeline. [See sec#/doc].
5. Missing frame-sets or missing frames within a frame-set will be noted by the preceding/some-other subsystem (TBD) such that completeness is guaranteed at the point **ARTID** starts processing the table. The preceding/some-other (TBD) subsystem guarantees: (via a database/dynamic status server)
    a. Completeness of a sequence. (Define sequence here?)
    b. Provides flags (or data) somehow/somewhere (TBD) to ensure deterministic results re: missing frames/data such that **ARTID** may continue its processing without confusion.

5. **ARTID** will update the input table with artifact data related to frame-sets. It may also create new data sets in a predetermined format (TBD) to supplement artifact data. These data will be located at a predefined (TBD) location.
6. All output data from **ARTID** will meet output requirements as specified in this/other document. [See sec#/doc]
7. **ARTID** will populate a predefined (TBD) database/location/status server with information relating to "problematic" units of frame-sets/frames in the course of **ARTID** processing. These in turn (may) ensure satisfying requirements (for completeness) of subsystems downstream from **ARTID**.
8. Other

4 Requirements/specification within ARTID

1. The atomic unit of processing in the **ARTID** sub system will be one of:
    a. The source list related to the "next" frame set: Each WISE band per exposure has an associated frame; hence the (natural) atomic unit for AD processing appears to be the extracted source list for the four related frames, one per band.
    b. An alternate approach is to designate the atomic unit of processing as the source list related to the "next" in a sequence of frames for a given band. Per this approach, **ARTID** will ingest the "next" source list

corresponding to a particular band. Since four such frames (one per band) exist within the context of a single frame set, **ARTID** will possibly (TBD) launch four processes, one per band when triggered. There appears to be no reason to guarantee synchronicity between these threads (?).

2. **ARTID** will process source lists corresponding to a sequence/series of frames per one of the following request criteria:
    a. Default: The "next" frame as defined in #1.
    b. A supplied/given set of frames as defined by a begin:end frame/frame-set identifier(s).
    c. A continuous set of frames as "seen" in a predefined (TBD) segment/sub-set of the input table.
    d. A set of frames matching a criterion of interest.
3. The core function of the **ARTID** subsystem is to update the input source list table with artifact related information. Processing for each artifact [See sec# doc] will be undertaken sequentially, given an assigned degree of significance per artifact [See sec# doc]. (Question: is there a reason why the artifact id and flagging process –per artifact- can't be handled in parallel for optimization?)
4. The **ARTID** system will process data in the following order: (At this stage we assume that the modeling and characterization of artifacts is well established).
    a. Extract "a/the" source list of interest from the table created by the source detection phase, **WPHOT**.
    b. Search the list in a) for detections that meet the required brightness threshold to be viable *Parent Source* candidates for artifacts.
    c. Search the list in a) for detections that have the correct/predetermined positional and brightness relationship (This come from simulations, modeling and characterization) to the bright stars in b). These are sources contaminated by artifacts.
    d. Differentiate as best as possible between real and spurious artifacts per predetermined modeling and characterization techniques.
    e. Update the input table for each band as follows:
        i. Flag each source contaminated by an artifact by specify the type(s) of artifact(s) affecting the source. This flag shall be called ***the confusion and contamination flag/map: cc_flag/cc_map.***
        (*Note: flag value also includes information on confusion with other sources -e.g. in a crowded field- and the category of artifact, i.e. real or spurious*)
        ii. Note the *Parent source* causing the artifact (as an attribute of the contaminated source) in the input table.
        iii. *Cull (or mark to be culled or register) those detections that are (solely) highly probable artifacts.* [Leave this for FPB stage]

5. **ARTID** will expect to find environment configuration files/setup details (if any) required by its internal sub systems/modules in a predefined location (TBD).
6. **ARTID** will handle warnings, errors and exceptions (non fatal problems) by reporting the problem (log/e-mail/cell-phone, blood on the screen etc - TBD.) per specifications [See sec# doc]. Subsequent to the generation of an error report, **ARTID** will continue processing data (the next frame/frame-set source list) unless explicitly halted by an operator *or* the context of the exception warrants a halt.
7. A crash (fatal failure) of the **ARTID** system will require the attention of an operator. A crashed **ARTID** system will not be re-started automatically. (TBD) (Questions: The pipeline is run continuously? Triggered by downlink events? Access to a DSN type system?)
8. Other

5 Artifact identification, characterization and flagging methods

5.1 Identification and characterization

Artifacts caused by the interaction of flux from bright stars (for any given band) and characteristics of the instrument will most likely contaminate the WISE images. Such artifacts will presumably fit these categories:

1. Latent images
   Persistent images (>1 frame) from bright sources – detector artifact
2. Dichroic or filter glints
   Point-like images that result from internal reflections of bright stars within the filter and dichroics of the WISE camera – optical artifact
3. Diffraction spikes
   A lens flare artifact due to light diffracting around the support vanes of the WISE secondary mirror from bright stars – optical artifact
4. Bright star halos
   Off-axis rays from bright stars, i.e. not all the light from the bright star is in focus – optical artifact
5. Optical ghosts
   Internal reflections of a bright source within the filter mechanism – optical artifact
6. Electronic ghosts
   Detector amplifier crosstalk – detector artifact
7. Transients
   High-energy cosmic ray hits - ?

8. Confusion

    The proximity of a bright star or a crowded field resulting in false detections or confusion – source/field artifact?
9. Other artifacts (TBD)

As of this date, Latency and Diffraction spikes have been simulated via WISE instrumentation.

On-frame bright-star artifact templates and geometric algorithms (f(mag, band)) will be used to identify and characterize candidate artifacts and detections effected by artifacts. Refer to the document **2MASS Data Processing** (**Cutri and Evans**) for details on potential identification/characterization methods.

[Incomplete]

5.2 Flagging

During the artifact detection phase of the frame/frame-set pipeline, each source-list relating to a frame/frame-set will be searched for sources contaminated by artifacts by matching the sources' positional relationship (f(mag, band)) to a *Parent source* causing artifacts. All true sources contaminated by artifacts will be **flagged (cc_flag/cc_map)** in the input table. Detections deemed to be (only) true artifacts, termed spurious artifacts, are also flagged, however, these will most likely be eliminated from the final release catalogue (TBD).

The contamination and confusion flag/map values will be inserted in the table per band per source. Hence four new columns will be inserted and populated to accommodate the **cc_flag/cc_map.**

The format of the WISE **cc_flag/cc_map** is defined as follows: (two possibilities)

5.2.1 cc_flag:

(***cc_flag is a priority-based summary of cc_map)***

The flag has 1 character value (highest priority/significance artifact affecting the source) per band per source. (*Probably will not implement this way*)

Flag value (1 character per band). Upper case = real artifact Lower case =spurious	Nature of contamination or confusion
0	Source is unaffected by artifacts or source is not detected in that band
P or p	**P**ersistence (Latent) Image from Bright Star
G or g	Dichroic or filter **G**lints
S or s	Nearby diffraction **S**pikes
H or h	Bright star **H**alos
O or o	**O**ptical ghosts
E or e	**E**lectronic ghosts
T or t	**T**ransients
C or c	**C**onfusion

Given the above, *a real source* that is identified as both contaminated by proximity to a latent image *and a* diffraction spike will have its flag attribute set as follows in the affected bands:

cc_flag = 'P'

An analogous spurious artifact will be set as follows:

cc_flag='p'

5.2.1.1 Packing/Unpacking the cc_flag

[Incomplete]

5.2.2 cc_map

The flag is a bit array/bit map per band per source representing *all* artifacts affecting the source in that band. *A real source* identified as both contaminated by the proximity to a latent image *and a* diffraction spike will have its **cc_map** (confusion and contamination map) set as follows for the affected band:

p	g	s	h	o	e	t	c	r
1	0	1	0	0	0	0	0	0

The 0 position **(r)** column (above) specifies the artifact as non-spurious.

An analogous spurious artifact will be set as follows:

p	g	s	h	o	e	t	c	r
1	0	1	0	0	0	0	0	1

5.2.2.1 Packing/Unpacking the cc_map

An *n* [TBD] bit array will be assigned for each source (object) for each band as defined above. The value of a named bit (on/off) will indicate contamination/not by a defined artifact. The numerical values given in the following table are the offsets of the named bits. For example, the named constant (FLAG) SOURCE_DIFF_SPIKE (2) corresponds to an integer value of (1 << 2) = 0x4.

Offset	FLAG Name	Description
0	SOURCE_LATENT	Latent contamination
1	SOURCE_GLINT	Glint contamination
2	SOURCE_DIFF_SPIKE	Diffraction Spike contamination
3	SOURCE_HALO	Halo contamination
4	SOURCE_OPT_GHOST	Optical ghost contamination
5	SOURCE_ELEC_GHOST	Electronic ghost contamination
6	SOURCE_TRANSIENT	Transient contamination
7	SOURCE_CONFUSION	Confusion contamination
8	SOURCE_SPURIOUS	Spurious
9		
10		

The following pseudo-code (resembling C) roughly corresponds to the packing and unpacking of the **cc_map**:

```
var |= FLAG;
        //Set the FLAG bit in the variable var
if(var & FLAG)
        //Is var's FLAG bit set?
```

```
var &= ~FLAG;  //Clear the FLAG bit in the variable var
```

To pack: (example)

```
If (source_contaminated_by_latent()) {
    source->flags |= SOURCE_LATENT;
}
```

and

```
if (!source_contaminated_by_latent()) {
    source->flags &= ~SOURCE_LATENT;
}
```

To Unpack: (example)

```
if (source->flags & SOURCE_LATENT) {
    // Do something
}
```

6 Optimization

[Nothing here yet]

7 Proposal for initial prototype

A simple and contrived proposal: (Use executive level currently existent, Perl wrappers and C/CFISTIO).

1. Locate a (simulated?) nL file/frame. Assuming the image was a point source simulation, we should be able to extract positional (x,y) data and four numbers representing flux (four bands per source).
2. Create a table of the sources from the frame/frame-set with positional and flux values (x,y, fluxBand1, fluxBand2, fluxBand3, fluxBand4) for each. (Or use the IPAC tables generated from **WPHOT**). We'll assume this to be our *working table* for a 'frame/frame-set'.
3. Extract a source list (or sub-set of it) into memory. Organize data structures/IO/memory optimally.
4. Identify *Parent sources* meeting threshold criteria causing artifacts (Simulated).
5. Identify sources contaminated by #4 above. (Simulated).
6. Identify (probable) spurious artifacts (Simulated).
7. Append 8 columns to the table - these columns are for the **cc_map**(contamination and confusion map) per band per source and the *Parent source* for each band.

   Parent Source IDs:

We'll create an integer array, of an equal number of indices/slots as ***cc_map*** *to ID the parent source – an array per band – assuming source IDs are assembled as bit arrays from the run number of the scan, the camera column etc. etc… We'll copy over the ID of the parent from the relevant column of the table –should be there? – Or figure out a way to create an array of links to the parent – dangerous?*

8. For each contaminated source, create an appropriate cc_map (and parent ID map) and populate the table columns accordingly.

That's all for now.