Test Report for Jiggle Mapping using the Naive Mapmaker

Rene Gastaud
Nicola Schneider-Bontemps

SPIRE-CEA-REP-003196

9th February 2009

1. Overview

The goal of this test is to demonstrate that the *Naïve APPP (Averaged Pointed Photometer Product) Mapper* module functions as specified by the documentation on the module. This includes the specifications put forward in the following documents:

- SPIRE Data Processing Pipeline Module Requirements (SPIRE-ICS-DOC-002998)
- SPIRE Pipeline Description (SPIRE-RAL-DOC-002437), paragraph 4.2.16 and 4.1.13

This test outlines the results of testing the *Naive APPP Mapper* Module with a set of dummy Pointed Photometer Products (PPP). This product contains timelines for each detector and the associated coordinates in the sky (pointing ra, dec).

The *Naive APPP Mapper* Module is used with the AOT jiggle map 64 nods.

The *Naive APPP Mapper* Module computes three maps (see figures):

the image

the error

the exposure.

The requirements to check are (from doc 1 **6.12.5)**:

- **SCNMAP-FUN-022** The SPIRE interface shall provide an interface between jiggle-mode observations and the mapping module's naïve mapper.
- **SCNMAP-FUN-031** The jiggle-mode SPIRE interface shall operate on data from a Pointed Photometer product.
- **SCNMAP-FUN-072** The jiggle-mode SPIRE interface of the MADmap mapper shall produce one SimpleImage Product as output for each array. This product shall contain at least the following information for each bolometer array:

  Estimated surface brightness

  Estimated error

  Unit

  Coverage

  Astrometry headers to convert map and sky coordinates

- Requirement xxx1: the signal is the mean of the signals falling on the same pixel of the final map
- Requirement xxx2: the error on the signal must be computed as the standard deviation of the input signal.
- Requirement xxx3: the exposure map (or coverage map) is computed by adding one to a given pixel of the final map each time a signal falls on this given map
- Requirement xxx4: the astrometry (pointing) of the final map must be consistent with the coordinates of the input
- Requirement xxx7: give an unit for the image

- **Input Data Product: Pointed Photometer Product (PPP)** This product contains timeline for each detector and the associated coordinates in the sky (pointing ra, dec). The signal is in a table dataset named signal and the coordinates ra in a tabledataset named ra. We need signal, error (for the error on the signal), ra, dec.

- **Output Data Product: Image** The output is one or three images, one image per array. You have one image for PSW (if present in the PPP), one for PMW (if present in the PPP, one for PLW (if present in the PPP). Each image contains 3 images (see above) one for the signal, another for the error, and the last one for the exposure.
- **Calibration Products:** None

- **Calling Procedure:** Assuming that *pp* is a PPP product

```
task = NaiveApppMapperTask()

mapPsw = task(input=ppp, array="PSW")

mapPlw = task(input=ppp, array="PLW")

mapPmw = task(input=ppp, array="PMW")
```

2. User Interface and API

The name of the short cut of the task in jide/hipe should be Task *naiveApppMapper* because the task class name is *NaiveApppMapperTask* . This has been corrected, but this compels not to use this short cut if we want the scripts to be used both with the old version and new version.

```
from herschel.spire.ia.pipeline.phot.scanmap import NaiveApppMapperTask
```

```
task = NaiveApppMapperTask()

mapPsw = task(input=ppp, array="PSW")
```

3. Test Design and Implementation

In order to exercise the *Naive APPP Mapper* module and to check that the average over a wide range of input values dummies PPP products are created with signal columns filled with different strategies.

- Bruce Sibthorpe Simulations: Bruce has created several ppp for the jiggle map 64 nods with different input skies
- Very Crude Simulator: Rene created one ppp for the jiggle map 64 nods with Lena (lady's face) as input

The different steps are outlined in the flowdiagram of Fig. 1.

It was easier to use for the first tests the ppp from the very crude simulator. The image has all the artifacts due to chopper, nodding and all the defaults of the instrument.

The steps of the tests are:

- Inside jide/hipe:
- Create dummy or read it from a fits file
- Check that the PPP is correct with some plots
-
- Run the task *Naive APPP Mapper*
- Check that we have something with display, then save it as a fits file.
-
- Inside IDL
- Read the PPP
- Read the Map
- Produce an IDL map from the PPP with the astrometry of the previous map
- Compare the map and IDL map
- Compare the exposure and IDL exposure

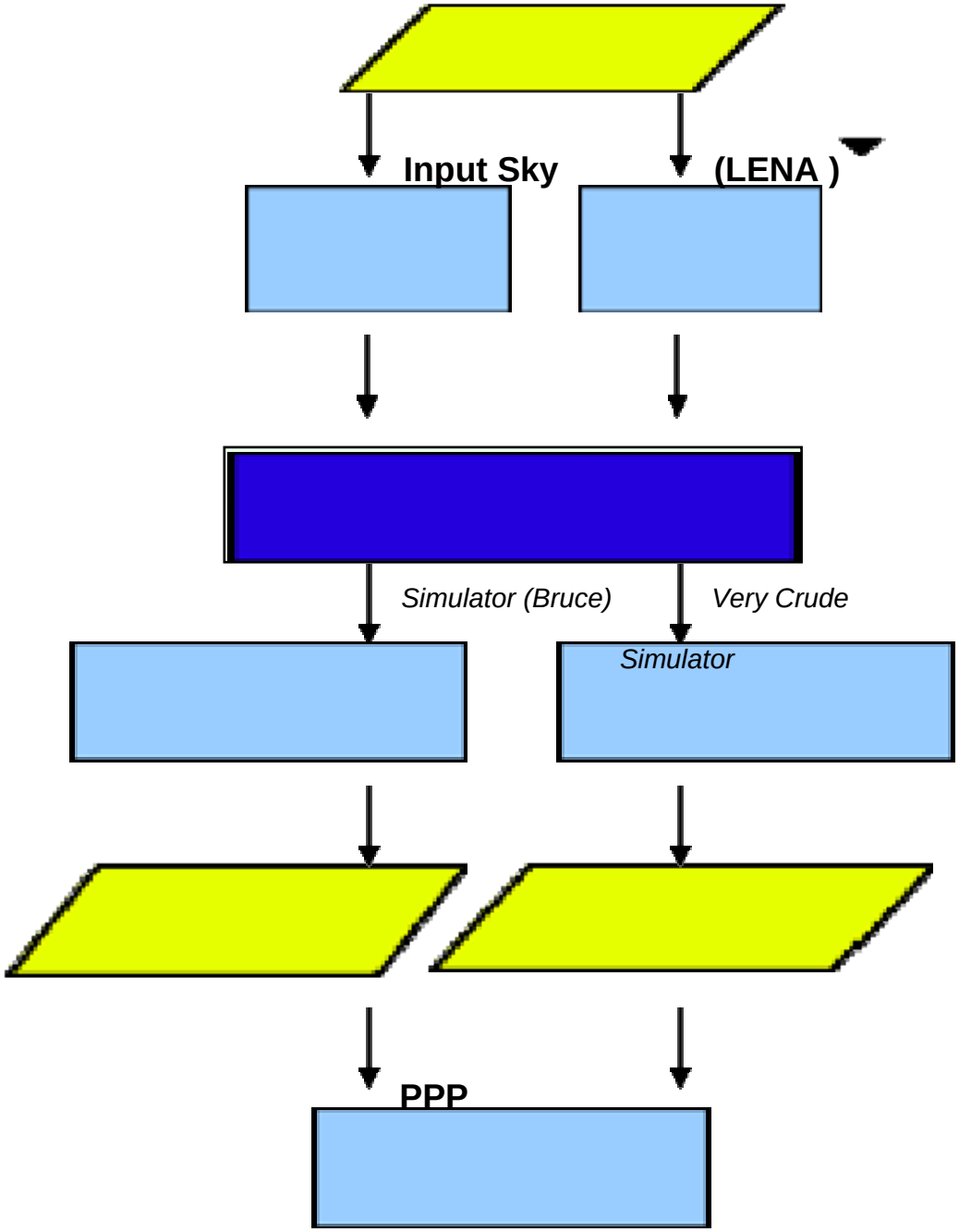


Fig.1 Flow diagram of the testing procedure

The IDL scripts, the IDL routines, the java scripts, the java modules, the input sky are under cvs. It is easier than to put them into the Twiki pages, where you can not put big files or keep the versions of a file.

The IDL scripts and routines are under cvs in the usr tree http://www.rssd.esa.int/herschel_scripts/cvsweb.cgi/develop/user/fr/saclay/rene/map/.

The java scripts and modules are in: http://www.rssd.esa.int/herschel_scripts/cvsweb.cgi/develop/proto/spire_pipeline/herschel/spire/ia/toymodel/scanmap/TestLenaJiggle64NaiveApppMapper.py?cvsroot=Herschel_CVS

The input sky is in: http://www.rssd.esa.int/herschel_scripts/cvsweb.cgi/develop/data/pacs/data/inputskies/lena512.fits?cvsroot=Herschel_CVS

4. Test with the Very Crude Simulator

The image of Lena has been used as input.

Fig. 2 IDL image, naïve image, and difference

The difference is less than 1e-6, and the mean of the image is 100, so relative error 1e-8 !

Fig. 3 Exposure

The results of both IDL crudeMosaiker and naiveAPPPMapperTask for the exposure are the same. Exposure is a matrix of integers, so here they are no errors by truncation.

5. Test with IDL Simulator

The output of the simulator for an exponential disk has been used. The disk is big compared to the detectors, so it is fairly constant. The image is weird, but we get the same image from the idl crudeMosaiker.

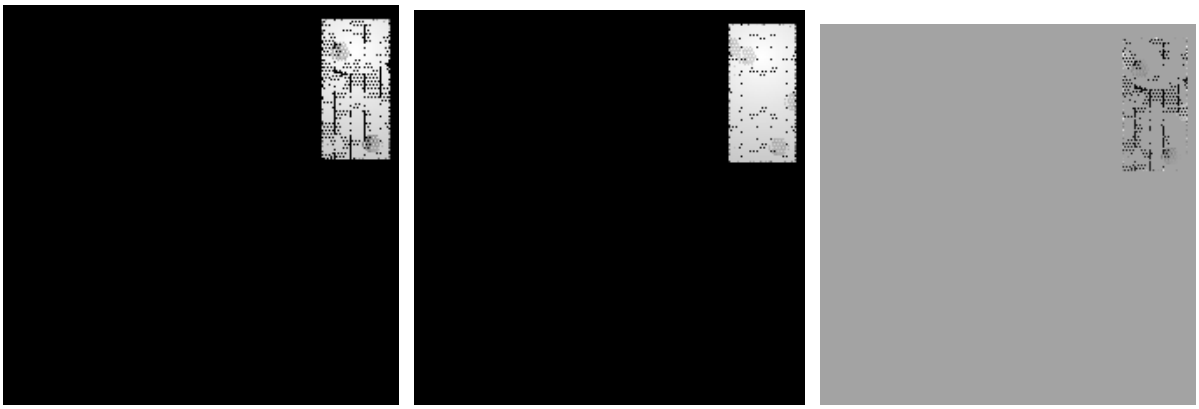


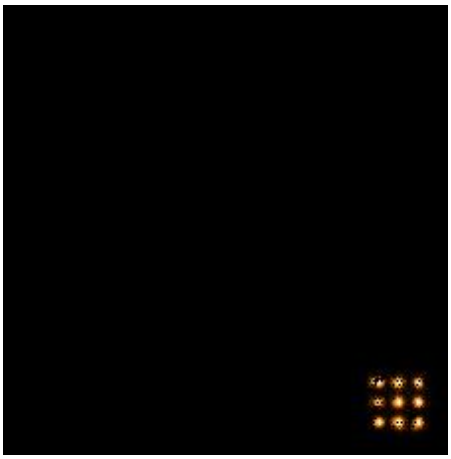
Fig. 4 Naïve image crudeMosaiker image difference of the two images

```
mean(abs(image1-naiveImage)) = 0.0068651006

mean(abs(naiveImage[ii]))= 0.87851875  (non zero pixels)
```

With the blank sky, the output image is blank, minimum is zero and maximum is zero.

With the psf 1 Jansky we got as final map this both with naïve app mapper and idl crudeMosaiker.



6. The obvious test

The obvious test is to compare the input sky with the image of the naïve appp mapper. It is difficult to do this test on Bruce simulated data, because they are chopped and are too realistic (too much defaults). It is easier with the very crude simulator.

With Lena, the first result is that there is an inversion right/left.

I correct it by changing the sign of cdelt1 in the astrometry.

In IDL I use hastrom to align image1 with respect to image2. The size of the total image, the size of the pixel, the coordinates of the pixels are changed so that the two images can be easily compared: it is the same part of the sky, with the same resolution and same orientation.

The input image (sky) has a resolution of 1 arcseconds, the output image has a resolution of 6 arcseconds (to be compared with the beam fwhm of 17).

1) First try regrid skyImage to naiveImage, so that both have the dimensions [47,90] .

   The difference shows a pattern, perhaps dued to a shift. The idl routine CORREL_OPTIMIZE gives a shift of -1, -1. With a shift of -1, -1 the difference of image show no more pattern:

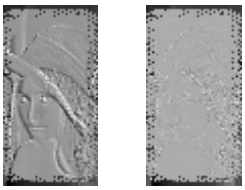


Fig. 5 No shift shifted

2) To improve the precision on the shift, I try to regrid the output image on the skyImage.

   The idl routine correl_optimize gives absurd result: the offset is as big as the image.

   My own correlation routine gives an offset of 4, 21 pixels or arcseconds (for skyimage 1 pixel=1arcseconds) to be compared to 1,1 pixels of the naiveImage or 6, 6 arcseconds.

   The difference of images shows a pattern, so this is not conclusive.

3) As the difference of the input image and the output shifted image is non zero and big in the previous test, another way to check the shift is to use a point as image. A perfect point is not easy to use, it can fall on non detected area (in real life no point exists, only psf/beam image). So after infructuous test with a point I use a gaussian, fwhm = 17.0. I then use the idl procedure gauss2dfit on the output image to check the

size and the spatial offsets.

Fig. 6: the reconstructed Gaussian, pixel size =6”

Spatial offset x=9.3 arcseconds and y=6 arcseconds, which is compatible with the previous result with Lena. The amplitude of the Gaussian is 9.7 to be compared with the input 10.: precision is 3%, which is good. The size of the Gaussian is 7.7 and 7.4 arseconds, to be compared to 7.23 arcseconds (fwhm=17.), precision is 5%.

The flux is given by 2*PI*a*b*amplitude, the precision on the flux is 4%

The reconstructed Gaussian is lower, and larger: it is blurred, which is normal: the naïve mapper is naïve and blurs the input image. The flux of the reconstructed image is greater of the input image.

7. Test Pass / Fail Criteria

When the difference of the reference and the output of the task is less than a given threshold (say by example 1E-5), the test is passed.

8. Input Data

Input data is a dummy PPP Product with, signal, ra, dec created by the script TestLenaJiggle64NaiveApppMapper.py

9. Test Results

The tests test1, test2 passes but not test3.

The test test6 works for “usual” data but fails for ra=[1,359], the computed mean is 180 instead of zero. Test5 is not yet implemented

10. Conclusions and Recommendations

The *Naive APPP Mapper* module module has been tested on dummy PPP. The following conclusions are drawn:

1. The module computes a correct image.
2. The module computes a correct exposure.
3. The astrometry is correct, with the problem of cdelt1/ flip left/right

The following recommendations are made:

More tests :

1 test the error

2 test the astrometry with other images