omqualitymap

January 12, 2017

Abstract

This program will ***either*** add **OM** source **quality-flag** information to a quality-map image stored in a product **OM** unrotated image file, ***or*** add source **quality-flag** information to an **OM** source-list file using information stored in a mosaiced quality-map image.

1 Instruments/Modes

Instrument	Mode
OM	IMAGING

2 Use

pipeline processing	yes
interactive analysis	yes

3 Description

Omqualitymap was conceived from a requirement for **OM** source-detection to be performed on mosaiced sky-images. Its main purpose is to add source quality-flag information to **OM** source-lists that have been produced from these sky-images, and to achieve this it operates in two different operating modes, which are described separately. The operating mode is determined by the input parameter **mode**, either-

- **setqualityimage\SETQUALITYIMAGE** for mode 1
- **usequalityimage\USEQUALITYIMAGE** for mode 2.

3.1 Mode 1- Quality-image pixel setting

The purpose of this mode is to populate the **QUALITY** image extension within individual **OM** images using the source-list data produced by **omdetect**.

Source-detection over each individual image is performed in detector coordinates using the task **omdetect**. This yields individual source-lists with catalogued detector pixel-coordinates, count-rates and source morphology data. Quality issues are related to either source-brightness, proximity to other sources, or image location. This information is collated from the source-list and recorded in a two-dimensional bit map that we will refer to as the "quality map". Quality issues are generally recorded over a cluster of pixels in the quality image.

The following is a list of the **QUALITY** pixel bits set, together with a description of the size and shape of the region containing the pixels whose bits are set. All coordinates refer to detector coordinates (0-2048, 0-2048) and heights, radii, etc, to unbinned pixels.

1. **Bit 0** (Bad Pixel) **Not set- please see section below**
2. **Bit 1** (Read-out streak) A series of 16 unbinned vertical columns, horizontally-centred on a star of raw count-rate $\geq$ 60 counts/sec.
3. **Bit 2** (Smoke-ring) A circular region centred on the positionally-dependent, empirically-derived centre of a smoke-ring. The circle has a radius of 38 arcsecs and the flag is set for all smoke rings with parent sources $\geq$ 60 counts/sec.
4. **Bit 3** (Diffraction spikes) Four rectangular regions rotated from the vertical by $\pm$45 and $\pm$ 135 degrees. The half-width of a rectangle is 10 unbinned pixels and they extend radially, 50 to 250 unbinned pixels from the centre of the star. This flag is only set if the central source $\geq$ 70 counts/sec.
5. **Bit 4** (Modulo-8 fixed pattern) Rectangular region centred on a source $\geq$ 60 counts/sec. The rectangle has a minimum half-width of 16 unbinned pixels. Pattern extent is related to source brightness, source morphology and local field density. The algorithm will attempt to determine the extent of the pattern and will adjust the dimensions of the rectangle accordingly.
6. **Bit 5** (Central-enhancement region) A circular region centred upon detector coordinate (992,1063) with a radius of 160 pixels.
7. **Bit 6** (Bright-source close proximity) is set by **omdetect**.
8. **Bit 7** (Image edge) Any pixel within a border distance of 12 unbinned pixels.
9. **Bit 7** (OM detector corner) Any pixel outside the circle centred on x=979.2, y=1016.0 with a radius of 1300 unbinned pixels.
10. **Bit 8** (Point-source lies within extended source) is set by **omdetect**.
11. **Bit 9** (Weird source- bright (hot) isolated pixels) Box centred on any abnormally isolated bright pixel of half-width 12 unbinned pixels.
12. **Bit 10** (One or more different exposure pixels within source photometry-aperture). This bit is set by **omdetect** and only when source-photometry is performed using an exposure image .

3.1.1 Notes

1. **Bit 0** (Bad pixel) is initially set by **omcosflag**. When **omdetect** does photometry on the detected point-like sources, any quality-pixel within the aperture that has this bit set will cause the quality-flag bit 0 of the source to be set. Note that pixels within the background-annulus with quality-bit 0 set are not used. Similarly, for extended-sources, any quality-pixel assigned to the source that has bit 0 set will cause the quality-flag bit 0 of the source to be set.

2. **Bit 6** (Bright-source close proximity) is set by **omdetect**
3. **Bit 8** (Point-source lies within extended source) is set by **omdetect**- no sense in setting the quality-image pixels.

3.2 Mode 2- Source-list quality flag setting

In this mode, after **OM** source-detection has been performed on a mosaiced sky-image, **omqualitymap** will set the **quality** flags of each source in the detection source-list using information from the **QUALITY** image stored in the sky-image file.

The program loops through each source and sets a particular bit of the **quality** flag if any of the 9 **QUALITY**-image pixels within a box centred on the source have that bit set.

It will then either create a new source-list file, **or** modify the input source-list file, depending on whether the parameter **outset** is the same as the input parameter **srclistset** or not.

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
srclistset	yes	string	none	

Name of the **OM** input source-list fits file.

set	yes	string	none	

Name of the **OM** input image fits file.

outset	yes	string	none	

Name of the output source-list **or** image fits file.

mode	yes	string	none	

Operating mode. either setqualityimage or usequalityimag. Their uppercase values are also accepted.

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be docu-

mented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

File does not exist! *(fatal)*
Name of an input file

Could not open file *(fatal)*
Name of the file

Could not load column from SRCLIST table- *(fatal)*
Name of the column

No SRCLIST table in file *(fatal)*
Name of the file

Unable to load SRCLIST table parameters *(fatal)*
Name of the file

The file name will be overwritten *(warning)*

corrective action: Ignore (if you don't want the file overwritten set the environment variable **SAS_CLOBBER** to 0)

The SRCLIST table in file name has zero rows- exiting *(warning)*

corrective action: Ignore

6 Input Files

1. **OM** Source-list file produced by **omdetect**.
2. **OM** image file produced by **ommodmap**, **or** a mosaiced sky-image produced by **ommosaic**

7 Output Files

1. **Either** a source-list file will be produced based on the input source-list file but containing extra **quality-flag** information, **or** an image file based on the input input image-file but containing a modified **QUALITY** image.

8 Algorithm

The program is written in **c++** and performs two different tasks:

1. Populates the **QUALITY** image extension within individual **OM** images using the source-list data produced by **omdetect**.
2. To set the quality flags of sources in an **OM** source-list produced by **omdetect**, using the information stored in the **quality** image of the image file.

8.1 Setting quality-image pixels (Program mode 1)

When the parameter **mode** is set to **setqualityimage)** or **SETQUALITYIMAGE**, the program works as follows:

1. Load the image from the **Primary fits file extension** in the input image file (**parameter imageset**).
2. Load the 16-bit 2-d quality-array from the **QUALITY fits file extension** in the input image file.
3. Load the x and y image coordinates (**XPOS and YPOS columns**), the raw-count rates (**RATE column**) and the source quality-flags (**QFLAG column**) from the input source-list file (parameter **srclistset**).
4. Set bit number 5 of all the pixels in the quality-image that lie within the central-enhancement region (see section 8.6).
5. Set bit number 7 of all the pixels in the quality-image that lie close to an image edge or corner of the OM detector (see section 8.7).
6. Loop through the list of sources and for any one that has a raw count-rate of at least 60 counts/sec
   - Loop through the list of sources and for each one
   - Check raw count-rate and if it is at least the minimum value, or if its quality-bit 4 is set -
     - Set bit number 1 of all the pixels on the quality-image in a column of specified width centred on the source (see section 8.2)..
     - Set bit number 2 of all the pixels on the quality-image that lie within a circle of given radius centred at the predicted position of its associated smoke-ring (see section 8.3)
     - If the raw count-rate is at least 70 counts/sec, check for a diffraction-spike and, if found, set bit 3 of the pixels (see section 8.4)
     - Check for a mod-8 pattern and, if one found, set bit 4 of the pixels (see section 8.5)
7. Create the output file (**parameter outset** that will store the modified image (the output file can have the same name as the input file).

8.2 Setting of bit 1 (Read-out streak)

Any source with a raw-count rate ≥ 60 counts/sec, or one that has **bit 4** of its quality-flag set (modular-8 pattern) will be assumed to have produced a read-out streak. **bit 1** of the quality-image pixels within a series of 16 unbinned vertical columns, horizontally-centred on the star, will be set.

8.3 Setting of bit 2 (Smoke-rings)

The predicted position of a smoke-ring is obtained from the following:

1. Compute the radial distance of the star (xstar, ystar) from the centre of the OM field of view (xCentre=1024.5, yCentre=1024.5)

2. Compute the position of the smoke-ring using:

   $x(smokering) = a_1 + b_1 \times x + c_1 \times y + d_1 \times x + e_1 \times y^2 + f_1 \times x \times y + xCentre$

   where x = xStar - xCentre, y = yStar - yCentre

   and $a_1 = 9.6950, b_1 = 1.2052$, $c_1 = 1.1027d-03, d_1 = 2.9117d-06$ $e_1 = 1.8790d-06, f_1 = 4.2347d-07$

   $y(smokering) = a_2 + b_2 \times x + c_2 \times y + d_2 \times x^2 + e_2 \times y^2 + f_2 \times x \times y + yCentre$

   and

   $a_2 = -4.8165, b_2 = 5.0884$ $c_2 = 1.2071, d_2 = -2.8810d-07$, $e_2 = 1.549d-06, f_2 = -1.0783d-06$

   These coefficients were obtained by Simon Rosen from a least-squares fit to about 100 measured positions of bright stars on OM images and their associated smoke-ring.

Bit 2 of all the quality-image pixels within a circular region of radius 38 unbinned pixels, centred on the predicted coordinates of the smoke-ring, will be set.

8.4 Setting of bit 3 (Diffraction-spikes)

Any object with either a raw count-rate > 70 counts/sec or flagged as having a mod-8 pattern is checked to see if it might have diffraction spikes.

The algorithm works as follows:

1. Count the number of objects along position angles of 45 and 135 degrees from the centre of the star that lie within a radius of 10 and 120 pixels from the centre and a distance of no more than 5 pixels from the radius line.

2. Compute the search area from:

   SearchArea = (maxRadius - minradius) × 4 × maxDistance

   where minradius=10, maxradius=200, maxDistance=5 minradius is the minimum radius from the centre of the star maxradius is the maximum radius from the centre of the star maxDistance is the maximum distance from the centre of the star.

   These define a search area along the radius vector from the star (4 angles), 45, 135, 225, 315 degrees) for faint stars associated with a star spike.

3. Compute the number of objects within a box of width and height 200 pixels centred on the star centre, ignoring objects with a raw count-rate greater than 5 counts/sec (n1).

4. Compute the number of stars expected within the search area, again ignoring objects with a raw count-rate greater than 5 counts/sec (n2).

5. If the number of star along the spikes (n1) is > 3 × n2 then conclude that there are diffraction spikes and flag sources along the spikes. All quality-image pixels within rectangular regions rotated from the vertical by ±45 and ± 135 degrees have bit 2 set. The half-width of a rectangle is 10 unbinned pixels and they extend radially, 50 to 250 unbinned pixels from the centre of the star.

8.5 Setting of bit 4 (Modula-8 pattern)

The algorithm uses a simple algorithm to search for a line of pixels in the horizontal\vertical directions to the left\right and top\bottom of the centre of given source. If 7 or more unbinned pixels are brighter than the corresponding ones along an adjacent column or row a modular-8 pattern may exist around the source. If at least 2 such patterns are found it is concluded that a definite modular-8 pattern exists. All pixels within a box of minimum width and height 21 pixels, centred on the source, have bit 4 set. The algorithm will attempt to determine the maximum distance of the pattern from the centre of the star and adjust the width and height accordingly.

8.6 Setting of bit 5 (Within central-enhancement region)

Any quality-image pixel whose detector-coordinates lie on or within a circle centred on x=992.0, y=1063.0, with radius of 160 unbinned pixels, will have **bit 5** set.

8.7 Setting of bit 7 (Near image edge/ corner of OM detector)

Any quality-image pixels $\leq$ 12 unbinned pixels from an edge will have **bit 7** set.

Additionally, any quality-image pixels outside a circle centred on x=979.2, y=1016.0, with radius 1300 unbinned pixels, will also have **bit 7** set.

8.8 Setting of bit 9 (Isolated exceptionally bright pixels)

The algorithm works as follows

1. Loop through each column of the image and for each column number i
2. Loop through each row of the image and for each row number j
   (a) Examine the 8 (less if near an edge) neighbouring image pixels around the pixel i, j and count how many have a value at least 1/20th of the central pixel.
   (b) If the number is less than 2 then assume the pixel i, j is abnormal- set bit 9 of the quality pixels within a box of width and height 41 unbinned pixels.

8.9 Setting source-quality flags

When the parameter **mode** is set to **usequalityimage**, the program works as follows:

1. Load the quality map image from the **QUALITY fits file extension** in the input image file (**parameter set**).
2. If the input image is a sky-image load the RA and DEC coordinates from the **SRCLIST** fits table in the input source-list file (**parameter srclistset**, and convert these coordinates to image x and y coordinates using the **CDELT1**, **CDELT2**, **CRPIX1**, **CRPIX2**, **CRVAL1** and **CRVAL2 FITs** keywords in the header. **otherwise** load the x and y coordinates from the **SRCLIST** fits table in the input source-list file (**parameter srclistset.**

3. Load the **QUALITY** flags from the **SRCLIST** fits table in the input source-list file (**parameter srclistset.**
4. Set the **QUALITY** flags of each source (see section 9
5. Create the output file (**parameter outset** that will store the modified source-list (the output file can have the same name as the input file).

9 Setting source quality-flag pixels using the QUALITY image

The algorithm works as follows

Loop through all the sources in the source-list and for each one:

1. Convert the source **RA** and **Dec** to **x** and **y** image coordinates using the WCS keywords in the FITS header of the image file.
2. Loop from bitnumber=0 to bitnumber=10
   - Using the **x** and **y** coordinates, check to see if bit **bitnumber** of the nine quality-image pixels in the box centred on the source are set- if so set bit **bitnumber** of the source quality-flag.

9.1 Notes

1. All **FITS** file operations are performed using the **SAS Data Access Layer** interface.
2. All arrays are implemented as c++ **vectors**.

10 Example QUALITY map produced by omqualitymap

Figure 1 shows a 16-bit **QUALITY** image produced by **omqualitymap** using the image on the left, the original boolean **QUALITY** image and information from the sources detected on the image.

The main features of the **QUALITY** image are

1. The circular light-blue region near the centre of the image, which corresponds to the central-enhancement feature.
2. A vertical brown strip associated with a read-out streak caused by a bright star centred at x=158 and y=192.
3. A small circular dark-blue region centred on this bright star.
4. A small circular yellow region, centred at x=143, y=178, associated with a smoke-ring caused by the the previous bright star.
5. A white strip around the whole image of width 6 pixels.

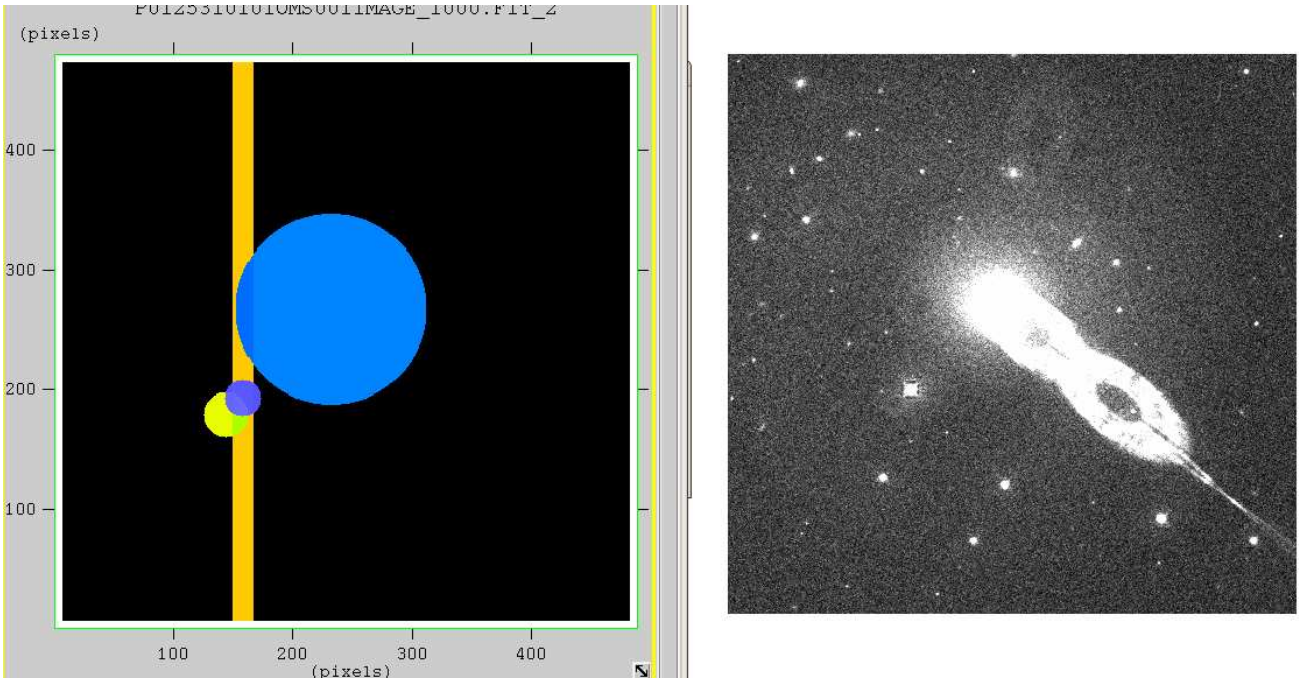

Figure 1: Example of a QUALITY image (left) produced by omqualitymap using source information derived from the image on the right

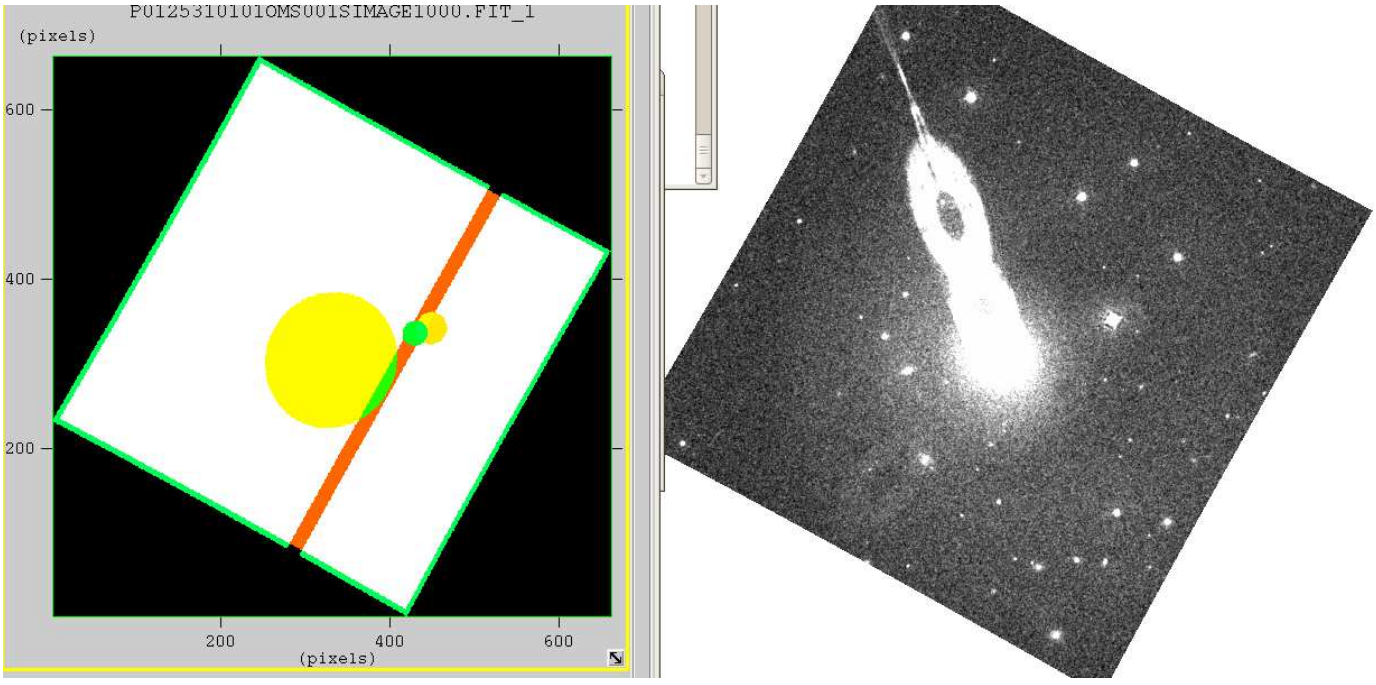

Figure 2: The same images after being processed by **omatt**

Figure 2 shows the same **QUALITY** image after it has been processed by **omatt**, together with the sky-image. Unfortunately, due to the presence of null pixels in these images, it was not possible to display them in similar colours.

Figure 3 shows a **QUALITY** image that has been produced by **ommosaic** mosaicing the individual

QUALITY images in each input sky image.

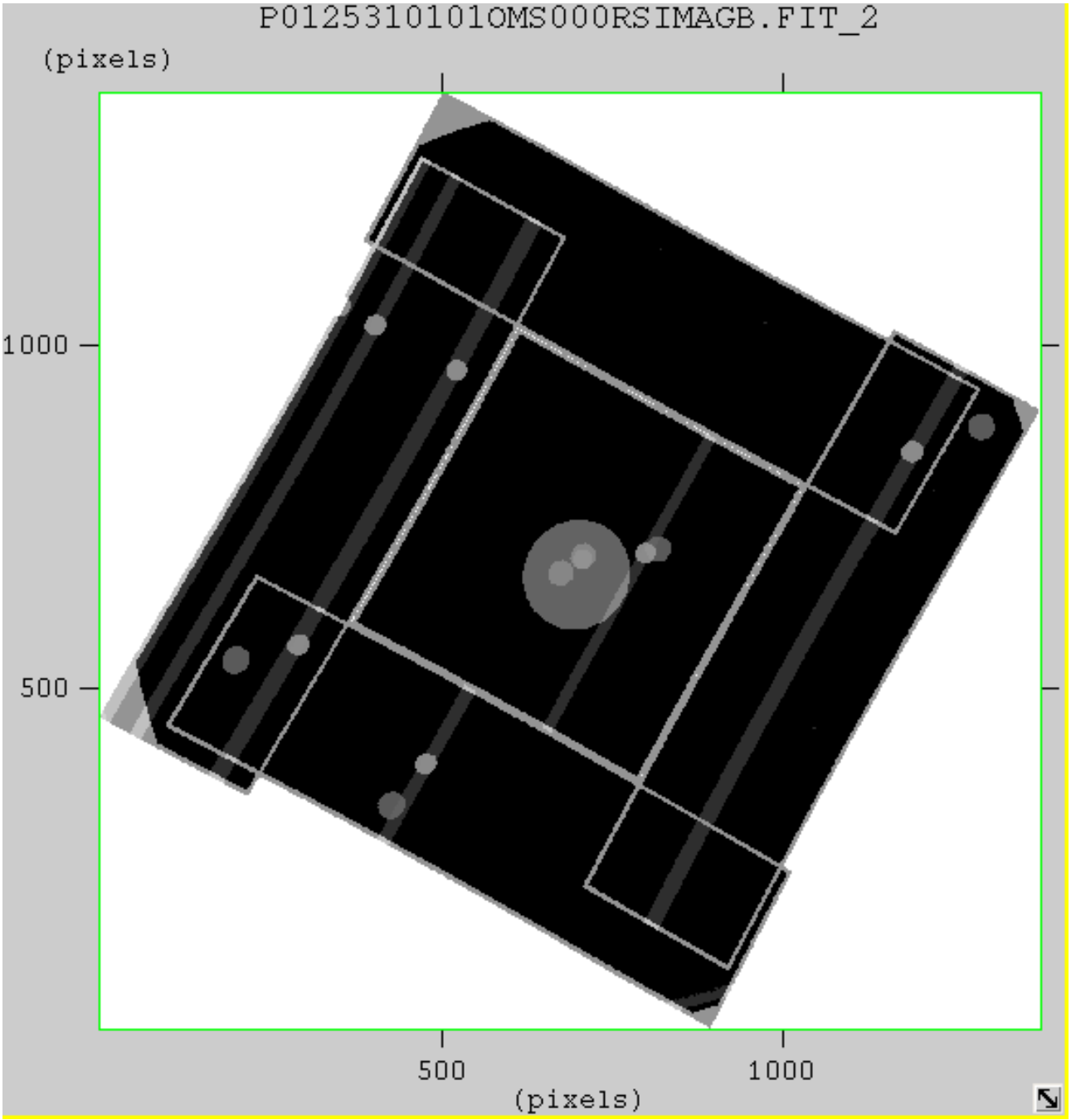

Figure 3: A mosaiced QUALITY image produced by **ommosaic**

The main features of this image are

1. The borders around each of the input sky images.
2. The read-out streaks (NB that some do not traverse the whole of the image- to be fixed).
3. The central enhancement circular region.
4. Bright sources that produce the read-out streaks.

5. A circular region round these bright sources to flag fant sources in their vicinity.

11 Testing

The actual testing is controlled by a shell program in the **ommergelists** test directory and is initiated by typing the command **make test**. To do the testing in **low memory** mode, you should set the environment variable **SAS_MEMORY_MODEL** equal to **low**. **Note** that at the daily builds at Vilspa the testing of all SAS software packages is done in both **high** and **low** memory modes (referring to the memory model used by the **SAS Data Access Layer**).

The tests work as follows:

1. A program **createtest** will execute and-
   - Produce a 2-d image with random sizes in the x and y directions.
   - Produce a 2-d quality-map image with the same dimensions as the image.
   - Produce a source-list containing a random number of sources. For each source-
     - Compute a random x and y image coordinate, within the image boundaries, and a random raw count-rate between 1 and 100 counts/second.
     - If the count-rate is at least 60 counts/sec-
       * Set bit number 1 of all the pixels in the quality map within a column of width 24 unbinned pixels centred on the source.
       * Set bit number 2 of all the pixels in the quality map that lie within a circle of radius 36 unbinned pixels centred on the computed position of a smoke ring produced by the source.
       * Set bit number 6 of all the pixels that lie within a radius of 36 unbinned pixels from the centre of the source.
       * Set quality flags bits 1, 2 and 6 of the source.
   - Produce a source-list file (**test.fits** with the same structure as an **OM** source-list file.
   - Produce an image file (**testimage.fits** containing the 2 images. with the same structure as an **OM** sky-image.
2. Execute **omqualitymap** with the parameter **updateimage** set to **true**, **set** set to **testimage.fits**, **srclistset** set to **test.fits** and **outset** set to test1.fits.
3. A program **checkoutput** will execute and will-
4. Read in the data for each source from **test,fits**
5. Load the **QUALITY** image from **testimage.fits**
6. Check that the **edge quality flags** have been set properly.
7. Loop through all the sources and for any with a raw count-rate of at least 60 counts/sec
   - Check that the pixels along a read-out streak centred on the source have bit 1 set.
   - Check that the pixels with a circle of radius 36 pixels centred at the predicted position of the smoke-ring have bit 2 set.
   - Check that the pixels around the source within a radius of 20 pixels have bit 6 set.
8. If any check has failed record a test failure and give some information where it failed.
9. SECOND PART OF TEST (MODE 2)

10. Execute **omqualitymap** with the parameter **mode** set to **true**, **set** set to **testimage.fits**, **srclistset** set to **test.fits** and **outset** set to test1.fits.
11. A program **checkoutput1** will execute and will-
12. Read in the data for each test source from **test.fits**
13. Read in the data for each source from **test1.fits**
14. Load the **QUALITY** image from **testimage.fits**
15. Check that the **edge quality flags** have been set properly.
16. Loop through all the sources from test.fits and for any with a raw count-rate of at least 60 counts/sec
    - Check to see that bits 2 and 6 of the quality-flag have been set in the sources of test1.fits
17. If any check has failed record a test failure and give some information where it failed.

omqualitymap correctly adds quality flags to an **OM** image, and also that it correctly transfers **quality-flag information** from a **quality** image to an **OM** source-list.

12 Future developments

The next version of omqualitymap will treat the propagation for faint and extended sources and crowded fields more exactly.

References