Using the OM Imaging SAS pipeline

Simon Rosen (MSSL, srr@mssl.ucl.ac.uk)
Chris Brindle (MSSL, cb1@mssl.ucl.ac.uk)

October 29 2001

Abstract

This document describes, and shows by examples, the step-by-step reduction of OM image data using the OM imaging SAS tasks.

1 Introduction

The OM SAS currently comprises 2 processing chains (perl scripts), one to reduce imaging mode data, the other, the fast (timing) mode observations, together with their constituent tasks.

The tasks that make up these chains (or pipelines) can be run individually. However, a given task will usually need to operate on the output results of a preceding task, so in general it is necessary for files from a given OM exposure to pass through all the chain tasks in sequence. For example, it is not possible to run the flat-fielding algorithm on a raw ODF image file because at the flat-fielding stage, the file should contain extra or modified header keywords and a quality array which are inserted by a routine at an earlier stage in the imaging chain.

The following sections describe the files required and the tasks used to process imaging mode data from raw ODF files to finished products.

2 Calibration set-up

Before processing OM data, it is necessary to create a calibration index file (e.g. ccf.cif file) using the SAS task, *cifbuild* - it is probably best to create it in the ODF directory (see contribution by *gvacanti_2*).

3 Flatfield

One file required by both the Imaging and Fast chains is the flatfield map. It was originally envisaged that flatfield images, generally obtained during spacecraft slews or periods when the OM is inactive, would be stored in an archive. This was intended, partly to monitor the evolution of the detector response and also to allow users to accumulate a statistically good flatfield from flats taken close in time to their observation. However, for a number of reasons this has so far proved unnecessary and a single OM flatfield is to be stored as a component of an OM Calibration file (OM_PIXTOPIXSENS). Due to the complexity of analysing the OM flatfield, at the current time the stored flat is a unit image (which is a reasonable approximation to ∼5%). This will be superceded by a 'real' flat once a better understanding of the nature of the flat-field variations and the gross response are attained.

The following SAS command will create the flatfield file from the calibration data.

omflatgen `outset`=flat.fit

- At present a warning will be issued noting that a template (ie a unit flatfield) is being used (see above).
- Cal usage: Reads the flat-field image from OM_PIXTOPIXSENS_vvvv.CCF (vvvv being the version number of the file).

This flat-field file is created automatically within the chain task, *omichain*

4 Imaging data types

Four types of science imaging mode data are already generated, or will be from AO2. These are:

- Rudi-5: A pattern of 5 separate OM exposures, each comprising two OM Science Windows (OSW) (images); a small (∼2'x2') high resolution (0.5"/pixel) window, centred in the FOV and a low resolution (1"/pixel) window. The location of the low resolution window is moved between exposures to consecutively cover a large (∼8'x8') region at the centre and 4 surrounding strips which together cover the whole FOV.
- Full-frame low resolution mode: (also known as ENG-2). A low resolution (1"/pixel) image capturing the whole FOV in one go (though data currently arrive in the ODF divided into 4 sequential files in detector space).
- Full-frame high resolution: (also known as ENG-4). A high resolution (0.5"/pixel) image covering the whole FOV. The data appears as a single file in the ODF. This mode will be available to GO's in AO2.
- User defined: Up to 5 windows per exposure whose sizes, locations and binning were pre-specified by the PI, subject to certain memory and overlap constraints.

5 Imaging chain versions and capabilities

While earlier versions can process Rudi-5 format data, versions V5.2.1 and beyond will contain an OM imaging chain which additionally can process low resolution full-frame (ENG-2) and high resolution full-frame (ENG-4) imaging mode data (there is a known problem with the dead-time fraction calculation for

ENG-2 data which can introduce ~3% errors in source count rates for this mode - this should be rectified in SAS V5.3).

6 Required ODF files

Before attempting to use the *omichain* or individual OM tasks, first run the SAS task, *odfingest*. This will create a *SUM.SAS file from the *SUM.ASC file that comes with the ODF - it is generally a good idea to keep this in the same directory as the ODF. With this and the other files listed below, which should already be in the ODF, it should then be possible to process the OM images.

rrrr_iiiiiiiiii_SCX00000ATS.FIT - spacecraft attitude history file

rrrr_iiiiiiiiii_SCX00000TCS.FIT - spacecraft time correlation file

rrrr_iiiiiiiiii_SCX00000SUM.SAS - modified ascii summary file

rrrr_iiiiiiiiii_OMX00000NPH.FIT - non-period housekeeping file

rrrr_iiiiiiiiii_OMX00000PEH.FIT - periodic housekeeping file

rrrr_iiiiiiiiii_OMX00000FFX.FIT - flatfield from omflatgen (rename from, or create as, the output of *omflatgen* (see section 3))

For each exposure to be processed we also need

rrrr_iiiiiiiiii_OMSeeewwIMI.FIT - raw image (window) file(s)

rrrr_iiiiiiiiii_OMSeeewwWDX.FIT - auxiliary window file

rrrr_iiiiiiiiii_OMSeeewwTHX.FIT ** - OM tracking history file

where **rrrr** is the 4 digit XMM rev number, **iiiiiiiiii** is the 10 digit observation id, **eee** is the exposure number (e.g. 006, 401 etc) and **ww** is the window identifier (00 or 01 for Rudi-5 patterns, 00, 01, 02 and 03 for ENG-2) data. Note that single ENG-4 mode images are immediately identifiable via the 'E4I' tag (i.e. files of the form rrrr_iiiiiiiiii_OMSeeewwE4I.FIT).

** - if these are not present, no tracking corrections can be applied but this is often not critical since XMM's tracking generally appears to be good to around 1 arc-second. Tracking files are not available with full-frame, high/low resolution mode data and are sometimes absent for user-defined and Rudi-5 modes, particularly when the UVW2 filter is used, due to a dearth of suitable tracking stars. If THX files are absent, a dummy file needs to be created (see *omprep* usage below) - this is the default action of *omichain* (section 7.2).

7 Approaches to OM imaging data reduction

OM imaging mode data can be reduced either by use of the 'pipeline' task, *omichain*, which encompasses all the processing tasks in one 'chain' and can act on a directory of data in one go, or by application of the separate tasks that constitute *omichain* to individual files.

Two points should be borne in mind when using the current version (1.28 or earlier) of *omichain*.

1. it does not provide user control of parameters affecting tasks within it - they are fixed at values used in the pipeline processing system (PPS) at Leicster.
2. a number of intermediate files are overwritten making it difficult to isolate such files from specific windows for subsequent processing by separate tasks.

Both of these draw-backs will be remedied in a later version of *omichain* (possibly in SAS V5.3).

7.1 Using the omichain task

With the ODF data stored in a working directory and with the SAS_ODF environment variable pointing to this directory, one can process data automatically, creating output files in the same directory, simply by typing

omichain

from that directory. Alternatively, one should be able to pick up data from an inpdirectory (which overrides any setting of SAS_ODF) and output to an outdirectory by typing

omichain `inpdirectory` =........ `outdirectory`=........

Where the .... are the directory paths - but note this does not work properly in versions prior to, and including, SAS V5.2. It should also be noted that *omichain* does not strictly adhere to the SAS standard for the use of SAS_ODF - this will be corrected in a future version.

The *omichain* can be considered as being divided into 5 stages (see figure 1). In the first, a flat-field image is created and any ENG-2 files are combined into full-FOV images. Three subsequent stages run in a loop, processing all the windows (images) from all exposures.

1. Tracking history information is turned into graphic products and a file used by the image analysis part of the chain.
2. The image is prepared for analysis (including flat-fielding and mod-8 noise correction).
3. The prepared image is processed and products derived from it.

When processing of exposures is completed, two further tasks form a final stage where source lists are combined and filter-specific full-frame images are constructed (see figure 1).

If the chain runs to completion, a number of important output files will exist (for naming conventions, refer to [1] - note some new OM product names are being added, principally for the new modes and mosaiced images) together with some (generally overwritten) intermediate files. You can access varying levels of information at run-time by changing the SAS_VERBOSITY environment variable. At SAS_VERBOSITY=5, for example, *omdetect* will output messages explaining where it has got to. Higher levels of verbosity yield greater (sometimes rather opaque) details.

7.2 Using separate tasks

The following describes the separate use of each task in the *omichain*. Note that most tasks can make use of more parameters than are shown but default values are generally satisfactory, or the resulting output files may be of interest but are not essential (see task descriptions for details).

In the following, we illustrate the command line use of each task with files from the data provided for the workshop. The observation used to demonstrate the use of *omcomb* is ENG-2 data and is different from that used for all subsequent tasks - the latter is a V filter band image which we will take through all the reduction stages. Where a given file name is used in the command line invocation of different tasks, this indicates that the same output file from an earlier routine is used by the later tasks (e.g., image3.fit is created by *ommodmap* and is then supplied to both *omdetect* and *omatt*).

The first task in the imaging chain is *omcomb*. This task, and the version of omichain that uses it, are only available in SAS V5.2.1 onwards. It is only needed for low resolution full-frame mode (ENG-2) data - if the data is Rudi-5 format, user-defined or high res full-frame data, you can skip *omcomb*. For ENG-2 data, you should first run

omcomb `imagesets`="0277_0136550101_OMS00600IMI.FIT 0277_0136550101_OMS00601IMI.FIT 0277_0136550101_OMS00602IMI.FIT 0277_0136550101_OMS00603IMI.FIT" `outset`=comb.fit

where comb.fit is a user named output file for the combined single image which would then become the argument of the 'set' parameter value for the second use of *omprep* below. In the above example we are combining the windows 00, 01, 02 and 03 from exposure 006.

Thereafter the task sequence is as follows (NOTE - the first 3 tasks form the tracking analysis part of the imaging pipeline, tasks 4-7, the image preparation stage and tasks 8-10, the source detection and parameterisation parts.

1) *omprep* `set`=0081_0123920101_OMS00600THX.FIT `nphset`=0081_0123920101_OMX00000NPH.FIT `pehset`=0081_0123920101_OMX00000PEH.FIT `wdxset`=0081_0123920101_OMS00600WDX.FIT `outset`=tracking.fit

- In its first use, *omprep* derives a tracking file for later application. It is important to note that if there is no THX file for the exposure, *eee* (006 in the example), you must make set=DUMMYTHX.FIT. Omprep will then generate a dummy tracking file needed by the rest of the chain with zero drift in it.

2) *omdrifthist* `set`=tracking.fit `plotfile`=tracking.ps

- Extracts information about the tracking history during the exposure and constructs plots of the tracking vector and tracking statistics. When used in the *omichain* a PS to PDF conversion utility generates a tracking.PDF file.

3) *omthconv* `nphset`=0081_0123920101_OMX00000NPH.FIT `thxset`=tracking.fit `outset`=tracking_star_timeseries.FIT

- Generates time series for the 'tracking' stars in the field.

4) *omprep* `set`=0081_0123920101_OMS00601IMI.FIT `nphset`=0081_0123920101_OMX00000NPH.FIT `pehset`=0081_0123920101_OMX00000PEH.FIT `wdxset`=0081_0123920101_OMS00600WDX.FIT `outset`=image1.fit

- Used again, here in the image preparation part of the pipeline, where it augments the header contents of the image file with information derived from summary and housekeeping files (e.g. incorporating filter, exposure, attitude and some window information).

5) *omcosflag* `set`=image1.fit `thxset`=tracking.fit

- Tracking shifts and then adds a bad pixel (quality) array. You will probably get a warning that support for sub-windows is not implemented yet. This is not important.
- Cal usage: Picks up the bad pixel data from the OM_BADPIX_vvvv.CCF file

6) *omflatfield* `set`=image1.fit `thxset`=tracking.fit `inorbitflatset`=0081_0123920101_OMS00000FFX.FIT `ppsflatset`=flat.fit `outset`=image2.fit

- Tracking shifts the flatfield (obtained from *omflatgen*) and applies it - note that the parameter, `ppsflatset`, is renamed `tsflatset` in V5.2.1 and beyond.

7) *ommodmap* `set`=image2.fit `outset`=image3.fit

- Gauges the mod-8 noise and applies a correction for it. Optional parameters allow output of the mod-8 pattern. It may be necessary in some versions to set an additional parameter (`ppsproduct` = y or n) - if yes, an output file for the mod8 noise pattern tile should be given as the argument of another parameter, `ppsmod8set`. From V5.2.1 onwards, these parameter names will be changed to `mod8product` and `mod8set`.

8) *omdetect* `set`=image3.fit `outset`=oswsourcelist00601.fit `outputregionfile`=yes `regionfile`=reg.dat

- Source detection and parameterisation algorithm. The main output file is the source list for the image (OSW). This contains filled values for the image coordinates of each source, source count rate, background rate, significance, source ellipse parameters and flags for sources that are potentially confused, extended or affected by bad pixels. Optional parameters allow output of an SAOimage-type region file. Other useful options include output of the background image and significance image.
- Cal usage: Makes use of the PSF function stored in OM_PSF1DRB_vvvv.CCF and the platescale available from OM_ASTROMET_vvvv.CCF.

9) *ommag* `set`=oswsourcelist00601.fit `wdxset`=0081_0123920101_OMS00600WDX.FIT

- Generates coincidence-loss and dead-time corrected count rates and subsequently, instrumental magnitudes (inserted into the sourcelist file output by *omdetect*)
- Cal usage: Exploits the PSF function stored in OM_PSF1DRB_vvvv.CCF, coincidence loss information in OM_PHOTTONAT_vvvv.CCF and obtains instrumental magnitude zero points from OM_COLORTRANS_vvvv.CCF

10) *omatt* `set`=image3.fit `sourcelistset`=oswsourcelist00601.fit `ppsoswset`=skyim00601.fit

- Performs astrometry conversions (pixel positions → α, δ, b_{II}, l_{II}) and produces a rotated skymap image. You will see a warning that the usecat option is not implemented. This indicates that the task currently does not perform a catalogue cross correlation and so source positions are not astrometrically corrected. Thus, overlays of, for example, the USNO catalogue on the sky image will generally reveal a distinct offset (up to $\sim$ 10 arcsecs) if attempted. The catalogue X-correlation option is being tested and will be implemented in the future.
- Cal usage: Uses distortion map parameters from OM_LINCOORDS_vvvv.CCF to linearise the pixel coordinates. Boresight details for the OM are derived from XMM_BORESIGHT_vvvv.CCF

The above tasks are run for each OM image. In fact, the tracking part of the pipeline need be done only once per exposure though each image must pass through the image preparation and reduction stages. Once completed, source lists can be combined using *omsrclistcomb* and a single full-FOV image can be generated for each filter using *ommosaic*.

In the example below, we combine the source lists of the low resolution central window exposures taken in the V band (exposure 006, window 01) and B band (exposure 008, window 01). The respective source-list files emerging from *omatt* are assumed to be oswsourcelist00601.fit (as in the above processing) and oswsourcelist00801.fit. We then mosaic together the 5 V band low resolution images (all with window value 01) from exposures 006, 485, 486, 487 and 488 which make up a full-FOV Rudi-5 pattern - the respective sky maps provided by *omatt* are assumed to be named skyim00601.fit, skyim48501 etc).

omsrclistcomb `sourcelistsets`="oswsourcelist00601.fit oswsourcelist00801.fit"
`outset`=combinedsrclist.fit

- Merges source lists created for separate OSW images into a single list (sorted by RA), considering sources which are within some tolerance to be the same.

  An optional parameter, nsigma, controls the limiting separation between two objects if they are to be considered the same object - separation = nsigma*position error.

  Where multi-filter data exist, the task also computes standard magnitudes (for optical band filters) and instrumental and standard colours.
- Cal usage: OM_COLORTRANS_vvvv.CCF is used to obtain colour coefficients for transforming instrumental to standard magnitudes where possible.

ommosaic `imagesets`="skyim00601.fit skyim48501.fit skyim48601.fit skyim48701.fit skyim48801.fit" `mosaicedset`=mosaicimage.fit

- This task takes a white-space-separated list of sky images from *omatt* and combines them into a single full-FOV image. In the *omichain* this is performed on a per filter basis. A warning may be issued if overlapping data from the same exposure are processed or if data from different filters are combined. The resulting file is currently only suitable for cosmetic usage, since exposure division does not include corrections.

8 Sanity checks

A number of things are worth checking with the *omichain* products.

The first thing to do is to check for missing data. If there is Rudi-5 data present, check that each field has 5 separate exposures, each with an 00 and 01 window.

Where ENG-2 data is expected, there should be a set of four files per ENG-2 exposure (windows 00, 01, 02 and 03). For ENG-4 data there should be 1 image per exposure. With user defined mode, there should be no more than 5 windows with a given exposure.

A useful first step relating to the products is to overlay the source list of each window onto the corresponding skymap using *implot*, e.g.

implot `set`=skyim00601.fit `withsrclistset`=y `srclistset`=oswsourcelist00601.fit `device`=yourdevice `itf`=1 `radius`=5 `maxsrc`=200 `colour`=7

One should look for sources that have been missed or spurious sources which are sometimes seen along readout streaks from bright stars, in the wings of bright or extended objects and near the scattered-light artefacts that are often present (especially in the central windows). It is possible to improve the detection ability via the `boxscale`, `smoothsize`, `nsigma` and `contrast` parameters of *omdetect* though this can sometimes be at the expense of detection quality elsewhere in the image. The reliable detection of sources in the vicinity of the scattered light features is a particularly difficult problem (and sometimes around large areas of raised background) and effort will need to be directed to tackling this issue, either through forward simulation or by some form of filtering/restoration algorithm.

It is also worth inspecting the combined source list (with the ftool, fv) to check that sources of interest have been picked up in all filters where they were seen and that the combined source list contains colour and (possibly) standard magnitude details for them. Check also that objects labelled as extended appear as such in the images.

Another product to examine is the tracking plot. The tracking vector should usually be confined to a region <1-2 arcsecs. Any exposure in which a large excursion is noted may be unreliable - check it for apparent underexposure or smeared images.

References

[1] Julian Osborne. Data products specifications. Technical Report SSC-LUX-SP-0004, SSC, 2000.

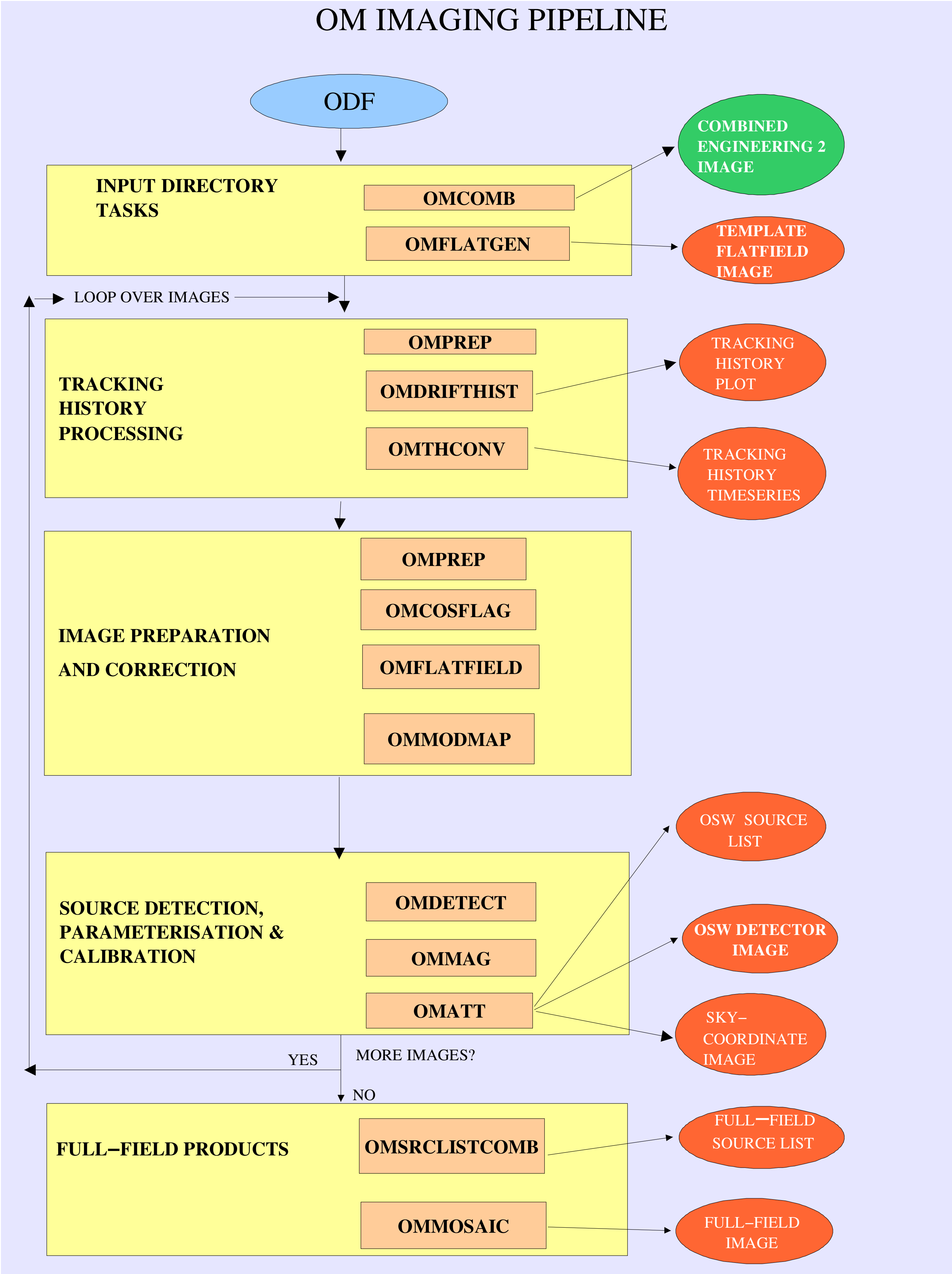

Figure 1: The data processing flow for OM imaging data within the OMICHAIN