Process DECam Images

This document is in early draft form! Not all intended content has been created, nor has it been vetted by experts!

This page will serve as a central reference for efforts being made to reduce DECam data using the LSST Stack. These may be broadly grouped into 2 sets of analysis: those using the obs_camera package, and those using some version of the obs_decam package. This page will contain practical techniques and use cases that demonstrate the quality of these reductions, as well as serve as a reference for co-development of an obs_decam package.

In This Document

Using obs_camera

See work by Scott Dodelson

Using obs_decam

The URL for the obs_decam package is here.

There is currently only one "mapper" defined for the analysis of DECam data, the DecamInstcalMapper. As the name suggests, this operates using the instcal (or Instrument Calibrated) data products. These have the instrumental signature removed, the background still in the image, and some measure of astrometric and photometric zeropointing. It also requires the associated dqmask (mask file) and wtmap (weight map, stored as inverse variance) files for the same EXPNUM (visit), and requires that they are in the same subdirectory, which serves as the main input repository for processing. In particular, it requires:

```
dqmask/ instcal/ wtmap/
```

The first step in this process is to build a registry that stores all the possible CCD images that can be analyzed. Since each DECam filename is unique, and they do not share a common string that associates the 3 files (instcal, dqmask, wtmap) together, we associate the 3 files by using their common EXPNUM in the header. These file names are stored in the registry. If there are not the requisite 3 files (instcal, dqmask, wtmap) having a common EXPNUM, this EXPNUM will not be entered in the registry.

The registry is built through the command

```
ingestImagesDecam.py decam/ --mode=link instcal/*fits
```

Things to note

. `ingestImagesDecam.py` is located in your path at `$OBS_DECAM_DIR/bin`

. The name of the output directory is given above as *decam/*. It is required that there is a file *_mapper* in this directory with the content: */lsst.obs.decam.decamMapper.DecamInstcalMapper*

. The "link" mode creates symbolic links to the original instcals. Other options are to "move" and "copy"

. Finally, point the script at the instcals you intend to analyze. Note the script expects the *dqmask/* and *wtmap/* directories at the same level as *instcal/*. Also note that any `.fits.[fz/gz]` files must be uncompressed beforehand

This will create the file `decam/registry.sqlite3` which contains the table raw with the schema:

sqlite3 registry.sqlite3

```
sqlite> .tables
raw         raw_visit
sqlite> .schema raw
CREATE TABLE raw (id integer primary key autoincrement, taiObs text,
instcal text,wtmap text,visit int,side text,filter text,ccdnum int,dqmask
text,date text,ccd int,expTime double, unique(visit,ccdnum));
```

These fields are defined here.

The next step is to process the images. This looks like:

```
processCcdDecam.py decam --id visit=283453 ccdnum=10 --config calibrate.
doPhotoCal=False calibrate.doAstrometry=False calibrate.measurePsf.
starSelector.name="secondMoment" doWriteCalibrate=False --clobber-config
```

Finally, to cluster the source measurements into lightcurves, run:

```
python $OBS_DECAM_DIR/examples/clusterSrc.py decam/ 0283453 0283454
0283455 ...
```