Process PhoSim Images

PhoSim is a high-fidelity package for simulating full focal-plane LSST Images, given a specification of an observation (pointing, time, environmental factors), the physical properties of the camera and detectors, and a catalog of sources that cover a region of sky to be observed. This package will be used to generate a small portion of the LSST **focal plane array** in multiple passbands, for subsequent processing and analysis using the **LSST Stack**.

Instructions for installing the **phosim** package and for building **Astrometry.net** index files are included in this tutorial, although the latter is not strictly needed.

In This Tutorial

Also of interest...

Preliminaries

Load the LSST Environment

You must have the **LSST Stack** installed on your system (see LSST Stack Installation) to proceed. The commands listed in the code blocks below primarily assume you are using the **bash** shell; analogous commands for (t)csh should work as well. If you have not already done so, load the LSST environment:

```
source $INSTALL_DIR/loadLSST.bash          # bash users
```

where `$INSTALL_DIR` is the directory where the **LSST Stack** was installed.

Tutorial Scripts

If you have not already done so, download the **tutorials** package, which includes scripts and data files that will be useful for this tutorial. Then create an environment variable for the python script directory:

```
cd /path/to/working/directory
git clone https://github.com/lsst-dm/tutorials.git
export PHODEMO_BIN=$PWD/tutorials/phoSimTutorial/python
```

Install PhoSim

You will need to install the **PhoSim** package (available from the LSST software repository) if it is not already available on your system. See the instructions for Installing PhoSim on your target system. Be sure the **SED** library is installed in the `phosim/data` sub-directory (or make a symlink to wherever the **SED** library is installed). The PhoSim reference manual (Peterson et al., 2013) is the definitive source of technical information about the package.

Simulating images with **PhoSim** is an extremely CPU intensive activity. This tutorial will minimize the extent of simulations needed, including the input physics, the number of sensors to be simulated, and the number of **passbands** and epochs to be simulated, while still illustrating the key steps for interesting processing and analysis. While these image simulations may be a substantial challenge for a modest desktop machine, users with significant compute resources will find it easy to extend this tutorial to more passbands, larger area, larger time coverage, or higher fidelity.

Create Simulated Images

For this tutorial the image simulations will be drawn from the portion of **SDSS Stripe 82** as observed by a single CCD **sensor** at the center of the **focal plane array**. (View the LSST focal plane geometry for reference.) The simulations will include the *g, r, i* **passbands**, and a small range of dates and observing conditions.

Obtain the Reference Catalog

We will use the *SDSS Stripe 82 Standard Star Catalog* of Ivezic et al. (2007 AJ, 134, 973) which contains *u,g,r,i,z* photometry of over 1 million stars with *r*-band standard errors < 0.05 mag. Download the catalog (67 MB) and unpack it in a working directory.

```
cd /path/to/phosim/working/directory
mkdir phoSimData
cd phoSimData
curl -O
http://www.astro.washington.edu/users/ivezic/sdss/catalogs/stripe82calibSt
ars_v2.6.dat.gz
gunzip stripe82calibStars_v2.6.dat.gz
```

Create Instance Catalogs

An **instance catalog** must be created to specify the observational parameters (pointing, seeing, etc.); and the positions, **SEDs**, and brightnesses of sources in the simulation. The **trim file** for this tutorial will be created from the reference catalog, using a python script (`refCalCat.py` in the **tutorials** package). To run the script, specify the filter of interest, a visit ID, spatial extent of the image(s) to be simulated, and the path to the **SED** library (relative to the `phosim/data` sub-directory).

For this example we will use the range `-0.5 < RA < 0.5` and `-0.5 < Dec < 0.5`, which is larger than the extent of the central **Sensor**. For simplicity a flat **SED** is used for all catalog stars, although the brightnesses will be taken from the reference catalog per filter. The visit ID is arbitrary, but it is handy to encode a date and some sort of sequence to aid bookkeeping. Create separate trim files for each filter of interest:

```
$PHODEMO_BIN/refCalCat.py --filter 'u' --ID 2014040 --path flatSED \
    --ralim -0.5 0.5 --declim -0.5 0.5 \
    stripe82calibStars_v2.6.dat > S82_u.trim
$PHODEMO_BIN/refCalCat.py --filter 'g' --ID 2014041 --path flatSED \
    --ralim -0.5 0.5 --declim -0.5 0.5 \
    stripe82calibStars_v2.6.dat > S82_g.trim
$PHODEMO_BIN/refCalCat.py --filter 'r' --ID 2014042 --path flatSED \
    --ralim -0.5 0.5  --declim -0.5 0.5 \
    stripe82calibStars_v2.6.dat > S82_r.trim
$PHODEMO_BIN/refCalCat.py --filter 'i' --ID 2014043 --path flatSED \
    --ralim -0.5 0.5  --declim -0.5 0.5 \
    stripe82calibStars_v2.6.dat > S82_i.trim
$PHODEMO_BIN/refCalCat.py --filter 'z' --ID 2014044 --path flatSED \
    --ralim -0.5 0.5  --declim -0.5 0.5 \
    stripe82calibStars_v2.6.dat > S82_z.trim
```

Astrometry Index Files

In order to perform astrometric calibration, index files must be available for the **astrometry_net_data** package to enable **WCS** solutions for science images. Since we are using the *SDSS Stripe 82 Standard Star Catalog* (i.e., the same reference catalog that was used for the Summer 2013 **Data Release Production** of **Stripe 82** data), the index files have already been built; we need only to download and install the index files and setup the appropriate package to make use of them:

```
curl -O http://lsst-web.ncsa.illinois.edu/sdss-s12/sdss-2012-05-01-0.tgz
tar xzf sdss-2012-05-01-0.tgz
setup -r sdss-2012-05-01-0
```

In the above, `$INSTALL_DIR` refers the directory where the LSST Stack was installed. To build index files from a different reference catalog, it will likely need re-formatting for input to the index-building task. See the the tutorial Building Astrometry.net Index Files or the Astrometry.net web site for details.

Simulate Images

Setup the following packages and set some handy environment variables, including the path to the **PhoSim** installation directory, and another to contain the number of cores available on your machine to enable parallel processing:

```
setup cfitsio
setup fftw
export PHOSIM_DIR=/path/to/phoSim
export NCORES=$((sysctl -n hw.ncpu || (test -r /proc/cpuinfo && grep
processor /proc/cpuinfo | wc -l) || echo 2) 2>/dev/null)
```

For this example we will simulate images for a single **Sensor** (1,1) in the central **Raft** (2,2) of the LSST focal plane. Create the output directory and start the **PhoSim** processing:

```
# Use the following sensor specification for the entire central raft:
# export
SENSORS="R22_S00|R22_S01|R22_S02|R22_S10|R22_S11|R22_S12|R22_S20|R22_S21|R
22_S22"
# A single sensor takes considerably less processing time:
export SENSORS="R22_S11"

# Invoke the following commands for each filter, substituting the output
and filter name as needed:
mkdir work
mkdir 2014042.out
python $PHOSIM_DIR/phosim.py $PWD/S82_r.trim \
    -w $PWD/work \
    -o $PWD/2014042.out \
    -c $PWD/clean.params \
    -p $NCORES \
    -s $SENSORS >& phosim_r_log.txt
```

Now create images in the other colors, taking care to match the trim file name to the passband, and creating subdirectories for the output.

```
mkdir 2014040.out
python $PHOSIM_DIR/phosim.py $PWD/S82_u.trim -w $PWD/work -o
$PWD/2014040.out -c $PWD/clean.params-p $NCORES -s $SENSORS >&
phosim_u_log.txt
mkdir 2014041.out
python $PHOSIM_DIR/phosim.py $PWD/S82_g.trim -w $PWD/work -o
$PWD/2014041.out -c $PWD/clean.params-p $NCORES -s $SENSORS >&
phosim_g_log.txt
mkdir 2014043.out
python $PHOSIM_DIR/phosim.py $PWD/S82_i.trim -w $PWD/work -o
$PWD/2014043.out -c $PWD/clean.params-p $NCORES -s $SENSORS >&
phosim_i_log.txt
mkdir 2014044.out
python $PHOSIM_DIR/phosim.py $PWD/S82_z.trim -w $PWD/work -o
$PWD/2014044.out -c $PWD/clean.params-p $NCORES -s $SENSORS >&
phosim_z_log.txt
```

Creating an image from a single CCD sensor with the stars in this catalog takes ~1.5 hours on a 2006-era Mac Pro with 8 cores.

Calibrate the Images

Create the Data Repository

The images in the `*.out` subdirectories do not follow the **LSST Stack** convention for data organization or file nomenclature. Thus, they need to be re-named and placed into a repository for subsequent processing by the LSST pipeline tasks. This can be accomplished with the **tutorials** scri pt `rename_images.py`. Now run the script for each `*.out` directory, looping with your favorite shell:

(t)csh loop

```
cd /your/phoSim/working/directory
foreach i (*.out)
  cd $i
  python $PHODEMO_BIN/rename_images.py ../imSim *E000.fits.gz
  cd ../
  end
```

bash loop

```
cd /your/phoSim/working/directory
for i in $( ls -d *.out); do
    cd $i
    python $PHODEMO_BIN/rename_images.py ../imSim *E000.fits.gz
    cd ../
done
```

Setup the Processing Packages

Source the LSST environment (`$INSTALL_DIR` refers the directory where the **LSST Stack** was installed), and setup the following packages to run the processing pipeline and related tasks:

```
setup pipe_tasks
setup obs_lsstSim --keep  # to process LSST Sims
```

Now create the input registry for pipeline processing:

```
export SIM_BIN=$OBS_LSSTSIM_DIR/bin
echo "lsst.obs.lsstSim.lsstSimMapper.LsstSimMapper" >> imSim/_mapper
$SIM_BIN/genInputRegistry.py ./imSim -o ./imSim/registry.sqlite3
```

Process Images

Run the pipeline to process the simulated images. This will generate catalogs of source measurements.

```
export SIM_DIR=$PWD
$SIM_BIN/processEimage.py $SIM_DIR/imSim --output $SIM_DIR/calexp_dir
--clobber-output --id visit=2014040..2014044
```

> This tutorial will be expanded at a later time to process raw input (amp) images. This will require generating flat-field reference images.

Create Merged Catalog of Sources

The results of the processing in the prior section generates an output source catalog (in the form of FITS binary tables) for each image. In order to evaluate the resulting flux measurements it is handy to collect the photometric results in one table. The following script (download: makeCat.py) will create a merged catalog in the form of a CSV file, containing a subset of the columns in the input tables. This is very similar to the script used to demo the **LSST Stack** installation, and illustrates the use of the **butler** to fetch results in a repository.

When running the script to create the merged catalog, the visit selector (last argument) may be a single visit number, a sequence of visits joined by carrots (2004040^2004044), or a range of visits that are contiguous in sequence, separated by two dots, as in the following:

```
$PHODEMO_BIN/makeCat.py ./calexp_dir  2014040..2014044 > mergeCat.txt
```

This merged catalog can be input to a large number of analysis tools. For analysis, it is possible to use the **butler** to fetch data directly from the repository for analysis with other packages in the **LSST Stack**, without the bother of writing an intermediate, merged catalog.