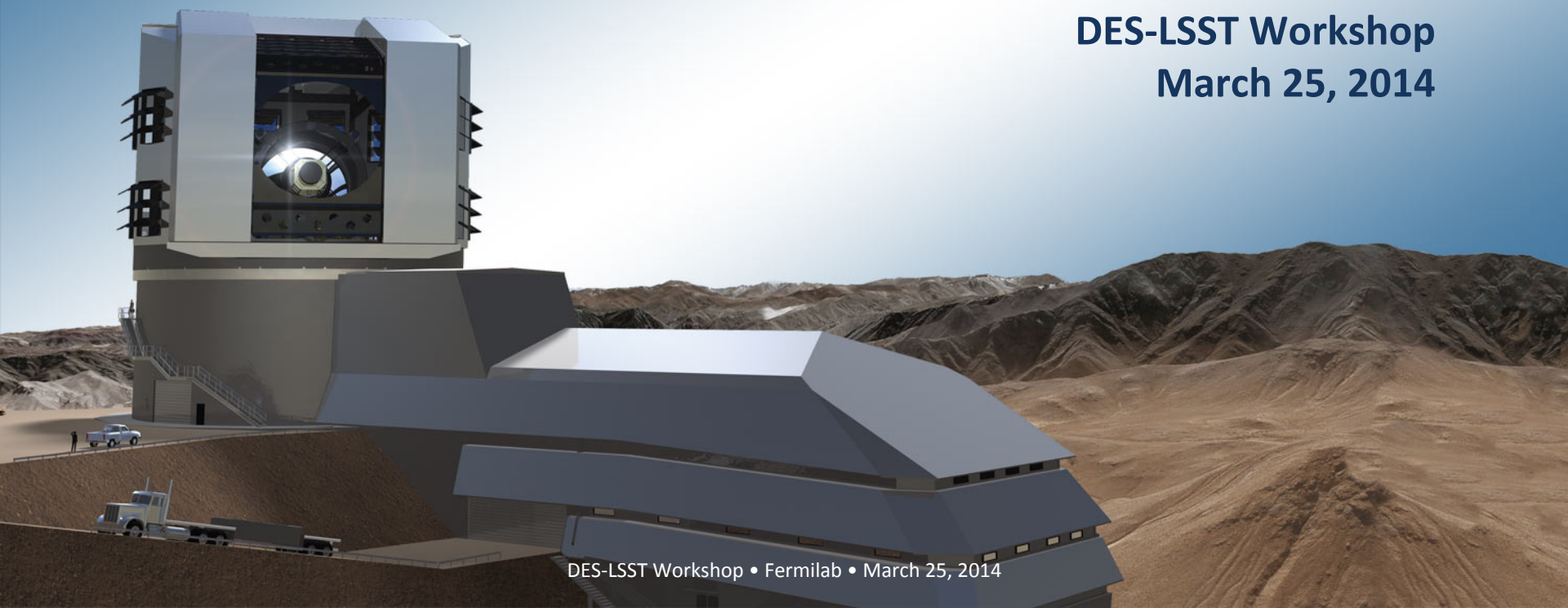
A Quick Tour of the LSST Software Stack

Kian-Tat Lim
LSST Data Management System Architect

DES-LSST Workshop
March 25, 2014



Welcome Aboard

Choo Yut Shing @flickr, CC BY-NC-SA 2.0

Acknowledgements

- Slides from:
  - Mario Juric
  - Robert Lupton
- Work by:
  - Entire LSST DM team

Languages, Platforms, and Tools

- The LSST stack is written in Python 2.7, unless computational demands require the use of C++

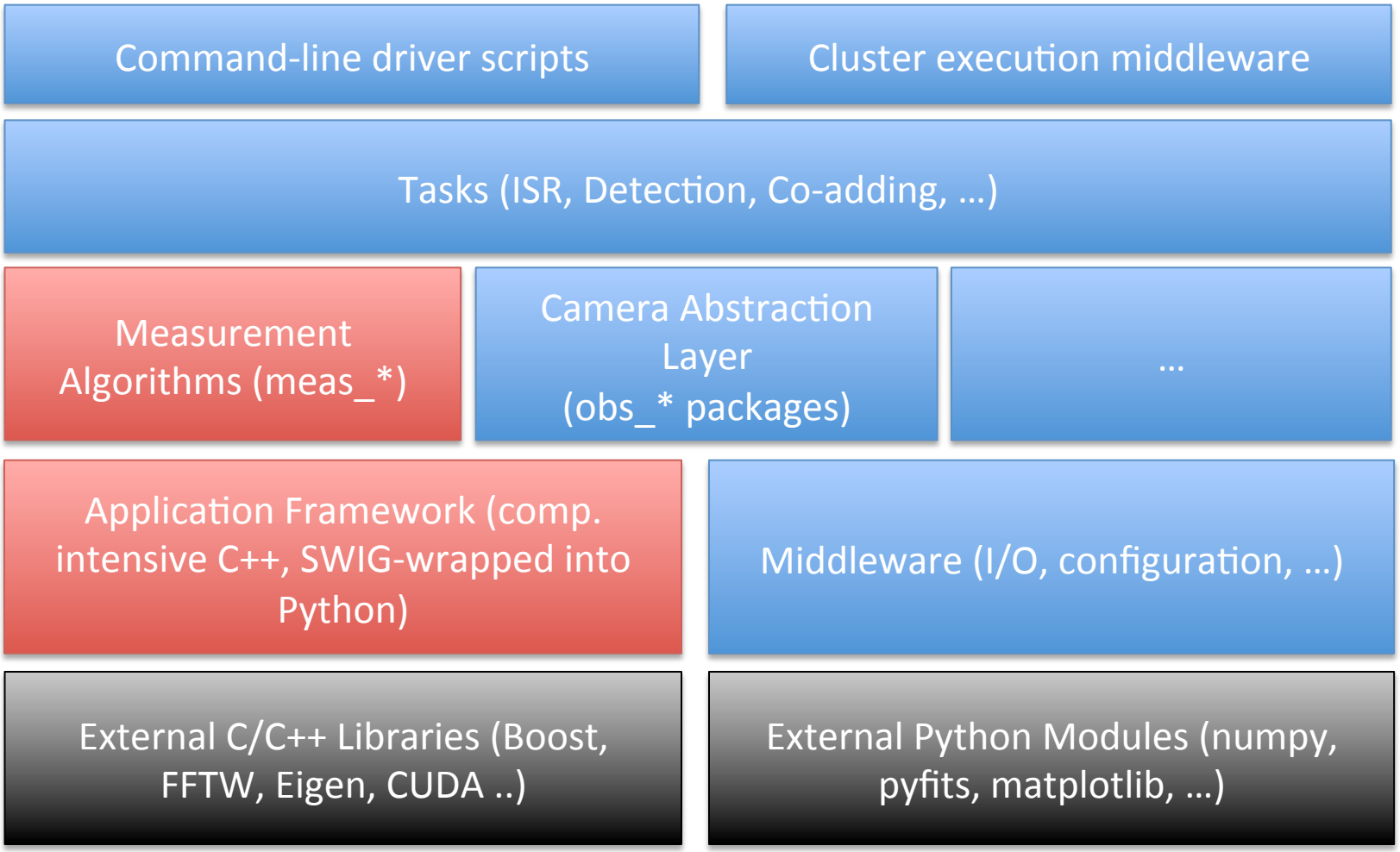
- Languages
  - **C++**:
    - Computationally intensive code
    - Made available to Python via SWIG
  - **Python**:
    - All high-level code
    - Prefer Python to C++ unless performance demands otherwise

- Red Hat Enterprise Linux 6 (gcc), MacOS X 10.9 (clang), others work

- Build system: **scons**
- Version control: **git**

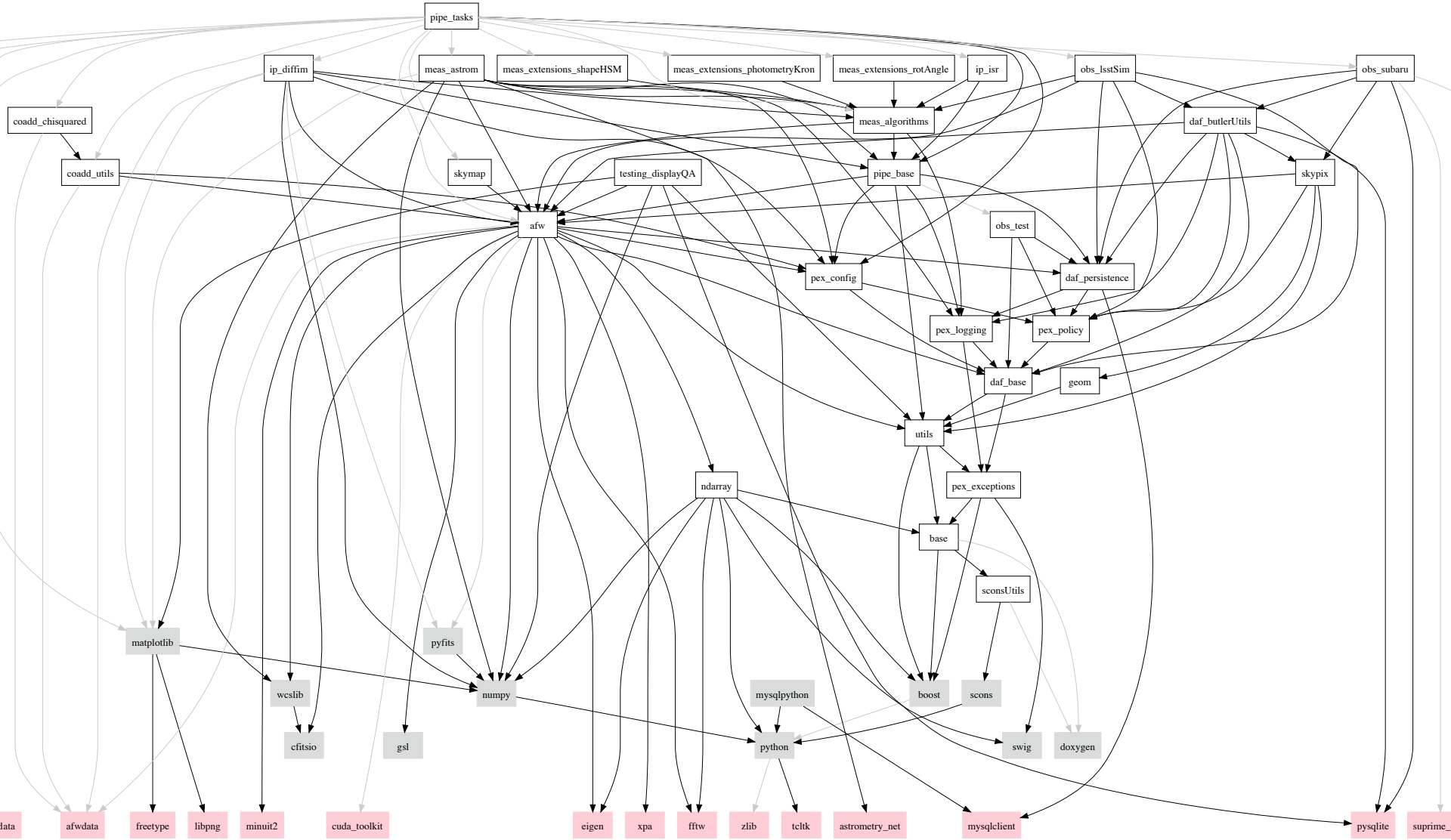
- Everything wrapped into **EUPS packages**
  - Allows one to install multiple versions of packages, and mix & match

Architecture

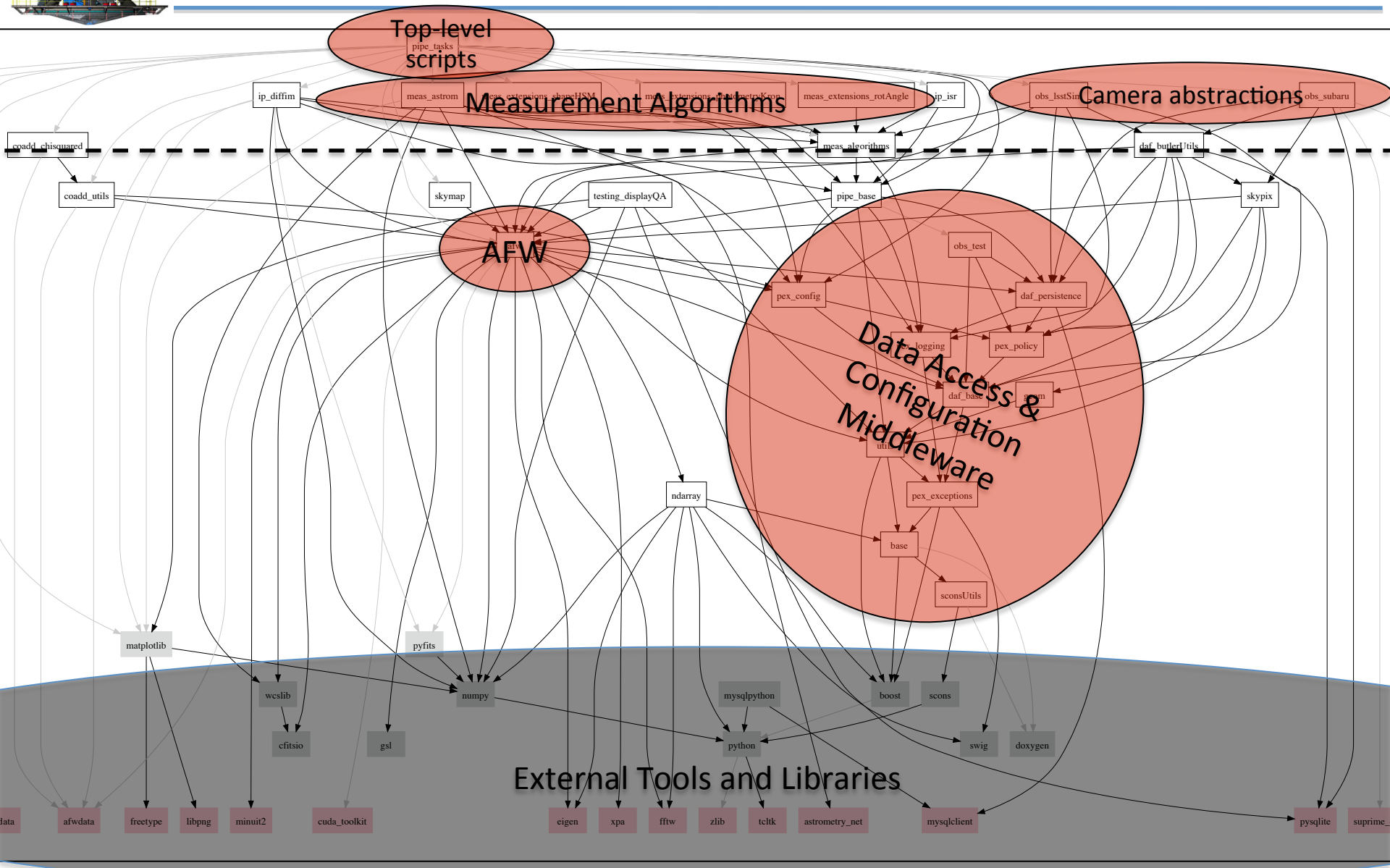


Red: Mostly C++ (but Python wrapped); Blue: Mostly Python; Black: External Libraries

Module Dependency Tree



Module Dependency Tree



Basics

- ndarray: efficient n-dimensional numpy-compatible arrays
- geom: Cartesian and spherical geometry in Python
- daf_base
  - `DateTime`: TAI/UTC, MJD/JD/J2000, string conversions
  - `PropertySet/List`: simple key/value pairs (e.g. FITS headers)
- pex_exceptions: basic exception handling
- pex_logging: basic logging

Application Framework

- A library of primitives for image processing
- image: pixel array, mask, metadata including WCS, filter, calibration, defects
- math: kernels, convolution, Gaussian processes, interpolation, least squares
- coord: on sky and observatory positions
- geom: Cartesian and spherical geometry in C++
- cameraGeom: camera physical and electronic description
- detection: footprint, Peak, Source, PSF
- table: persistence of detections and exposure metadata

afw Examples

- When you want array operations, afw supports numpy:

```
import lsst.afw.image as afwImage
import numpy as np
im = afwImage.ImageF(100, 200)
imarr = im.getArray()
imarr[np.where(np.logical_not(np.isfinite(imarr)))] = 0.0
```

- "Schrodinger's catalog":

```
import lsst.afw.table as afwTable
cat = afwTable.SourceCatalog.readFits("goo.fits")
for src in cat:
    print src.getCentroid()
print cat.get("psfFlux")
```

i.e. the catalog is both a list of Sources and a set of measurement arrays

Measurement

- meas_algorithms (=> meas_base): framework with plugins
  - meas_extensions_multiShapelet
  - meas_extensions_photometryKron
  - meas_extensions_shapeHSM
- meas_deblender
- meas_multifit: multi-component galaxy properties

Pipeline (Task) Framework

- pex_config: parameter and plugin configuration
- pipe_base: argument handling

Astrometric and Photometric Calibration

- meas_astrom: single-visit astro/photometric calibration
- meas_mosaic: cross-visit astro/photometric calibration

Coaddition and Differencing

- skymap: overlapping rectangular patches covering the sky
- coadd_utils: coaddition utilities
- ip_diffim: difference imaging

ISR and Camera-Specific Customization

- ip_isr: instrument signature removal
- obs_*: observatory-specific packages
  - test
  - cfht
  - decam: https://github.com/LSST-nonproject/obs_decam
  - lsstSim
  - sdss
  - sst
  - subaru
  - wiyn

Top Level pipe_tasks

- processCcd/processCoadd/processImage
  - calibrate (repair, astrometry, measurePsf)
  - detection
  - deblend
  - measurement
- makeSkyMap
- makeCoaddTempExp (warpAndPsfMatch)
- assembleCoadd (selectImages, interpImage, matchBackgrounds, scaleZeroPoint)
- processCoadd
- forcedPhot
- imageDifference (registerImage)

Command line examples

Process a single SDSS frame

```
processCcdSdss.py sdss /sdss/dr7/runs\
    --id run=1033 camcol=2 field=111 filter=g \
    --output /sdss/dr7/coadds
```

Co-add a number of SDSS frames

```
makeCoaddTempExp.py calexp_dir --output coadd_u_dir \
     --id filter=u tract=0 patch=375,0^375,1^375,2
assembleCoadd.py coadd_u_dir \
     --config maxMatchResidualRatio=2.5 \
              maxMatchResidualRMS=0.5 \
     --id filter=u tract=0 patch=375,0^375,1^375, run=5823
```

Data Butler and Database

- daf_persistence: general persistence framework for repositories of datasets (Data Butler)
- daf_butlerUtils: astronomy-specific tailoring of Butler framework
- db: generic SQL database utilities
- qserv: scalable distributed database with spherical partitioning and neighbor query support

Execution Control Middleware

- ctrl_orca: orchestration of jobs and task graphs
- ctrl_execute: parallel task execution tool
- ctrl_platform_*: platform configurations for ctrl_execute
- ctrl_stats: tools for computing execution statistics
- ctrl_events: event messaging system
- ctrl_evmon: event monitoring (complex event processing)

QA Framework

- testing_pipeQA: compute QA measures and plots
- testing_displayQA: generate Web pages with QA results

PipeQA Example (http://goo.gl/zsQR0)

lsst1.ncsa.uiuc.edu/pipeQA/public/latest/summary.php?test=test_885335881-r_pipeQa.PsfShapeQaAnalysis&active=all

Q.A. Test Summary

Go to main rerun list.

Home | TestList | Group | Summary | Help

Astr | Comp | Empt | Phot ap-cat | Phot inst-cat | Phot mod-cat | Phot mod-inst | Phot psf-ap | Phot psf-cat | Phot psf-inst | Phot psf-mod | PsfS | Vign | Zero

<<- prev-group (none) | next-group ->> (none) | <<- prev-r (none) | next-r ->> (885335891-r)

test_885335881-r_pipeQa.PsfShapeQaAnalysis

[show description]

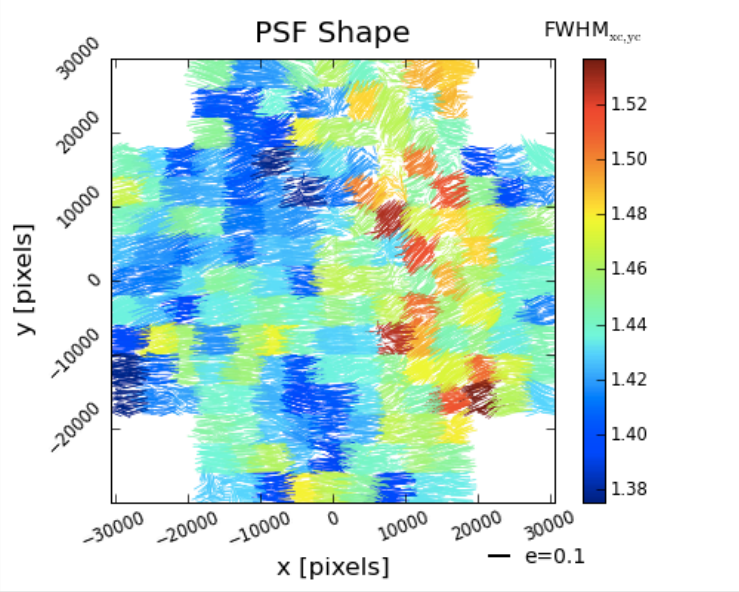


Figure 2.0: PSF ellipticity all
psfEllip-all.png: timestamp=2011-07_31 16:09:11

dataset: rplante_PT1_2_u_pt12prod_im3000 rerun=None
visit-filter: 885335881-r
Active: all

Show all

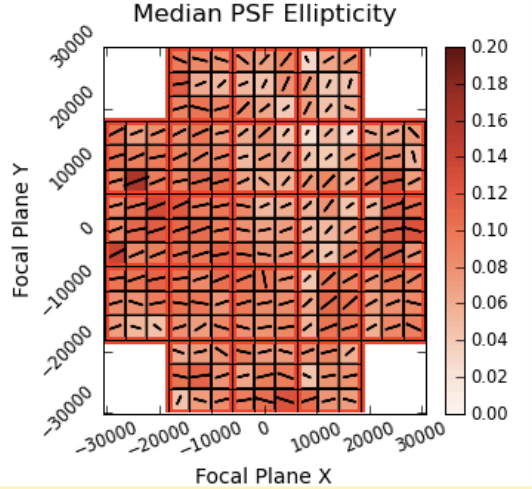
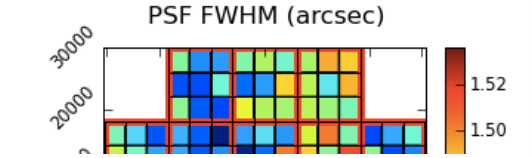


Figure 1.0: Median PSF Ellipticity
medPsfEllip.png: timestamp=2011-07_31 16:08:56

Show all



Using LSST Stack: Component Maturity

- Application and Measurement Libraries
  - afw and meas_* modules:
    - Image manipulation classes and algorithms
    - Flexible table classes
    - Image/Table FITS IO routines
    - Source detection routines
    - Source measurement routines (e.g., fluxes, shapes, etc…)
    - …

Using LSST Stack: Component Maturity

- Configuration and Data Access Middleware
  - pex_config, daf_* modules:
    - Configuration classes (with provenance)
    - Dataset repository management
    - Cons:
      - Less mature than afw and meas_*
      - You may already have something better in place

?

- Tasks framework
  - pipe_base, pipe_tasks, obs_* module:
    - Pros:
      - Can directly re-use complete DM tasks (e.g., instrumental signature removal)
      - Standardized argument parsing
      - Will work with any future DM orchestration mw / workflow mgmt tools
    - Cons:
      - More up-front effort than “just write main()”

Using LSST Stack: Component Maturity

- Pipeline Execution Middleware
- Workflow Management
- Database

Using LSST Stack: Component Maturity

- Quality Assessment and Visualization
  - PipeQA framework:
    - Framework for writing tests operating on ***catalogs***
    - Cons:
      - Immature, will be rewritten/redesigned once we find the resources
    - Pros:
      - Exists. May be better than starting from scratch.
      - Any improvements/development you make may help DM
    - PipeQA example: http://goo.gl/zsQR0

?

Installing the Stack

- Install prerequisites: https://confluence.lsstcorp.org/display/LSWUG/Prerequisites
- Install packages: https://confluence.lsstcorp.org/display/LSWUG/Building+the+v8.0+LSST+Stack+from+Source
- …
- Profit!

Getting the Source Code

- git repository:
  - `git://dev.lsstcorp.org/LSST/DMS` (anonymous access)

- code browser:
  - http://dev.lsstcorp.org/cgit/
  - Note: Browse subsets of repositories by appending a prefix, e.g.: http://dev.lsstcorp.org/cgit/LSST/DMS/meas will show all repositories matching DMS/meas*.

- git repository documentation
  - http://dev.lsstcorp.org/trac/wiki/GitDemoAndTutorial

Documentation

- Most (though not all) DM code is ***not*** well documented at the highest level. Documentation exists at the low level (doxygen generated). There's some documentation on the wiki, but it's poorly organized.
- Doxygen:
  - https://lsst-web.ncsa.illinois.edu/doxygen/x_masterDoxyDoc/
- Some packages have README files
- WARNING: Some documentation may be out of date!
- We will pay much more attention to documentation in the future.