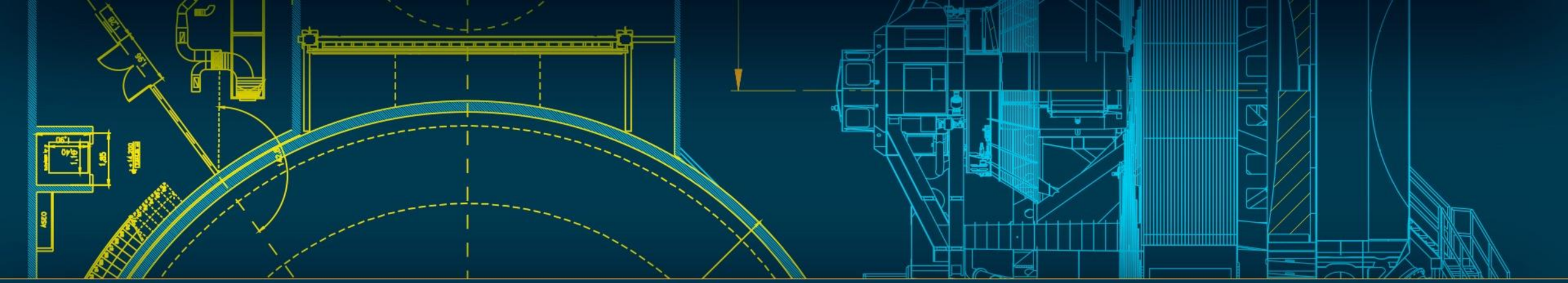
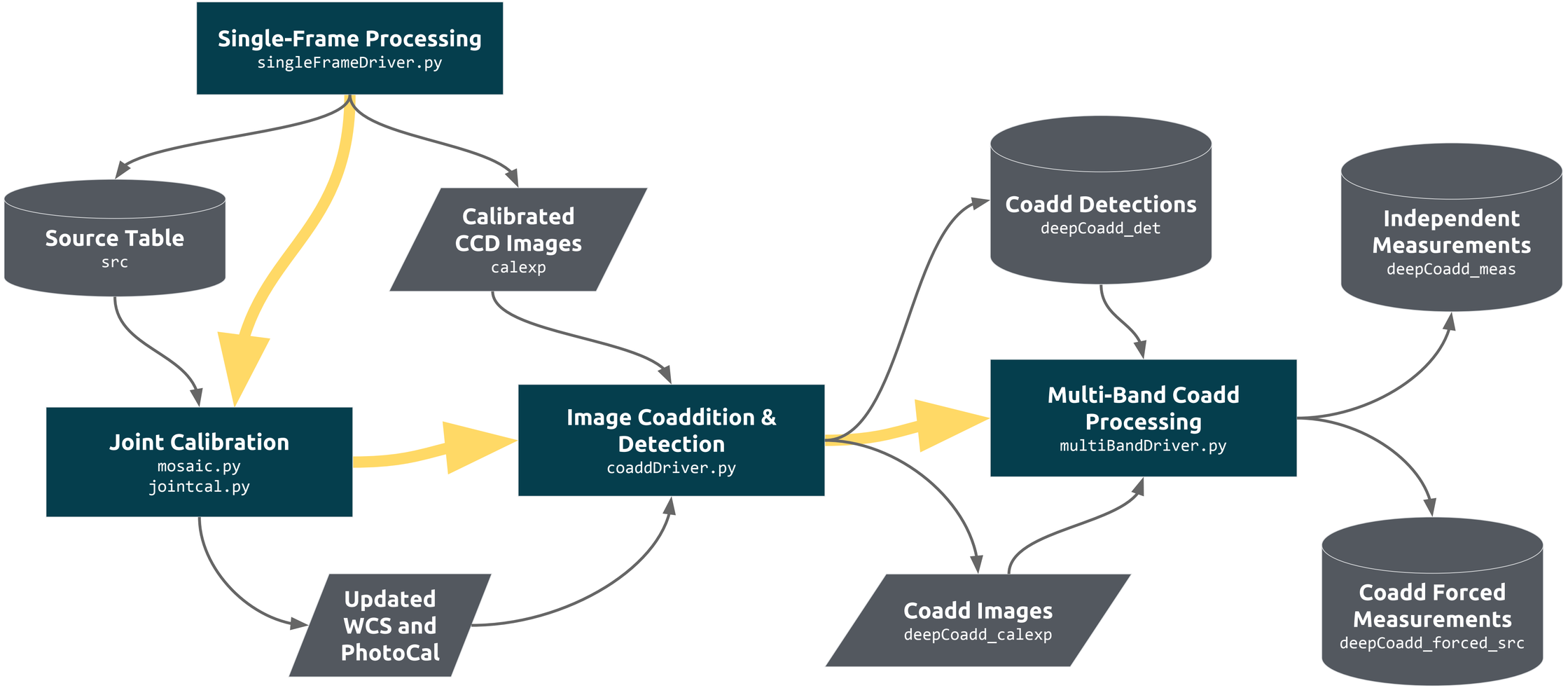


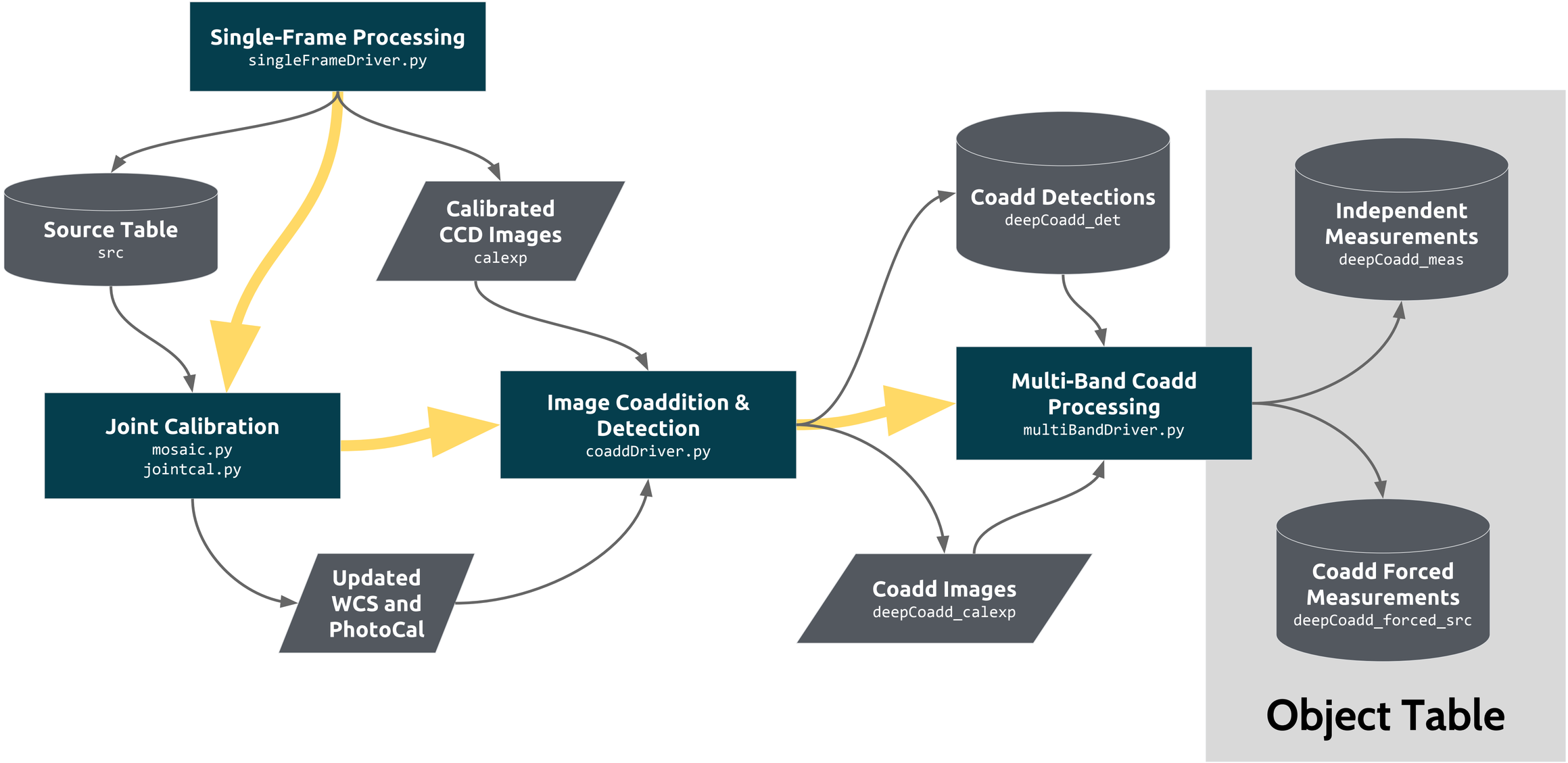
State of the Level 2 Pipelines

Jim Bosch, Princeton University

LSST2016 PROJECT AND COMMUNITY WORKSHOP

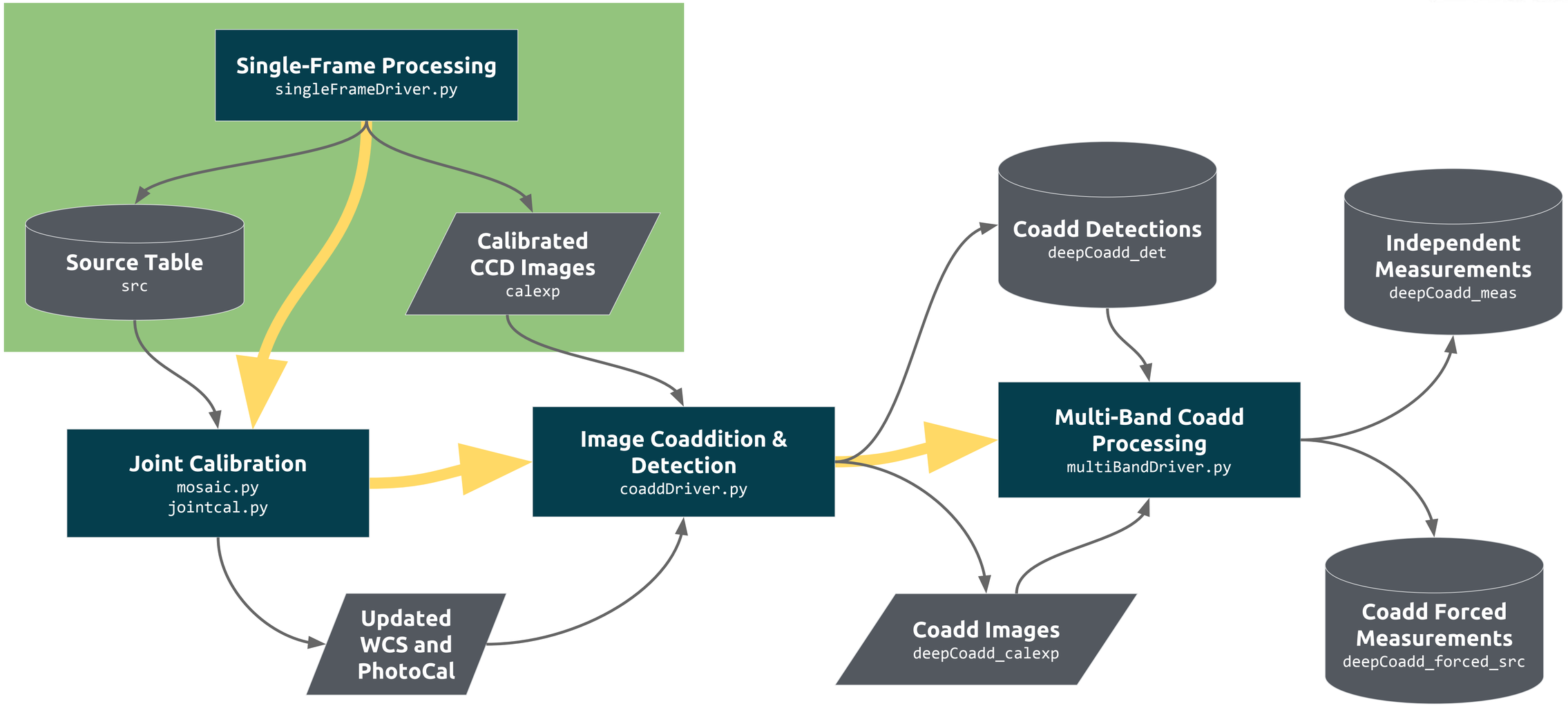
AUGUST 15-19, 2016 | TUCSON, ARIZONA


Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated CCD Images
calexp
Joint Calibration
mosaic.py
jointcal.py
Updated WCS and PhotoCal
Image Coaddition & Detection
coaddDriver.py
Coadd Detections
deepCoadd_det
Multi-Band Coadd Processing
multiBandDriver.py
Coadd Images
deepCoadd_calexp
Independent Measurements
deepCoadd_meas
Coadd Forced Measurements
deepCoadd_forced_src


Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated CCD Images
calexp
Joint Calibration
mosaic.py
jointcal.py
Updated WCS and PhotoCal
Image Coaddition & Detection
coaddDriver.py
Coadd Detections
deepCoadd_det
Coadd Images
deepCoadd_calexp
Multi-Band Coadd Processing
multiBandDriver.py
Independent Measurements
deepCoadd_meas
Coadd Forced Measurements
deepCoadd_forced_src
Object Table

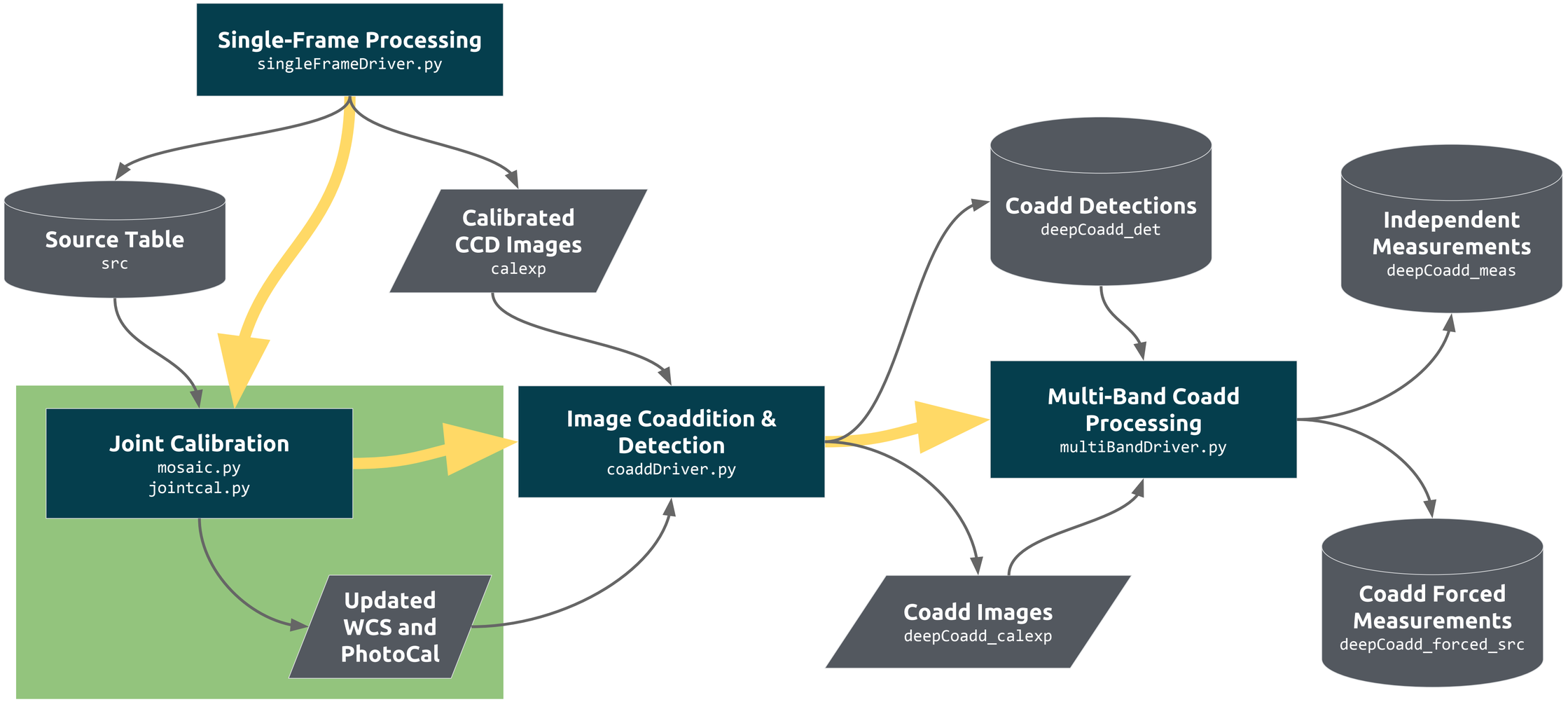
Pipeline Overview

- Each step is a Python class with an executable script that runs it.
- Can be parallelized over big clusters with MPI+Slurm/PBS (just a placeholder).
- Nearly all data products are FITS files, but if you use our Python packages to read them you won't care.


Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated CCD Images
calexp
Joint Calibration
mosaic.py
jointcal.py
Updated WCS and PhotoCal
Image Coaddition & Detection
coaddDriver.py
Coadd Detections
deepCoadd_det
Multi-Band Coadd Processing
multiBandDriver.py
Coadd Images
deepCoadd_calexp
Independent Measurements
deepCoadd_meas
Coadd Forced Measurements
deepCoadd_forced_src

Single-Frame Processing

- Instrument Signature Removal (`IsrTask`)
  - Frequently customized for each camera.
  - Can do partial (~90%) brighter-fatter correction.
- Image Characterization (`CharacterizeImageTask`)
  - Detect, deblend, and measure bright sources (`icSrc`).
  - Select stars and estimate a PSF model.
  - Estimate aperture corrections.
- Single-Frame Calibration (`CalibrateTask`)
  - Detect, deblend, and measure faint sources (`src`).
  - Match to a reference catalog to fit WCS and magnitude zeropoint.


Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated CCD Images
calexp
Joint Calibration
mosaic.py
jointcal.py
Updated WCS and PhotoCal
Image Coaddition & Detection
coaddDriver.py
Coadd Detections
deepCoadd_det
Multi-Band Coadd Processing
multiBandDriver.py
Coadd Images
deepCoadd_calexp
Independent Measurements
deepCoadd_meas
Coadd Forced Measurements
deepCoadd_forced_src

Joint Calibration

In each tract:

- Match sources from all visits.
- Fit new WCSs for all visits (including distortion and optionally CCD positions)
- Fit magnitude zeropoint and smooth transparency term for all visits.

Joint calibration is not yet as easy to run and use as the rest of the pipeline (and only runs on CFHT and HSC at present), but it can be skipped.

Joint Calibration

- `meas_mosaic`
  - only runs on Subaru (Hyper Suprime-Cam and maybe Suprime Cam) data
  - uses dense matrices: slow (30 min on 4 HSC visits), doesn't scale well with # of visits

- `jointcal` (formerly `meas_simastrom`)
  - photometric fit not yet fully integrated into pipeline
  - only runs on CFHT and HSC (somewhat on lsstSim) at present
  - uses sparse matrices: fast (10 sec on 4 HSC visits), scales very well with # of visits.

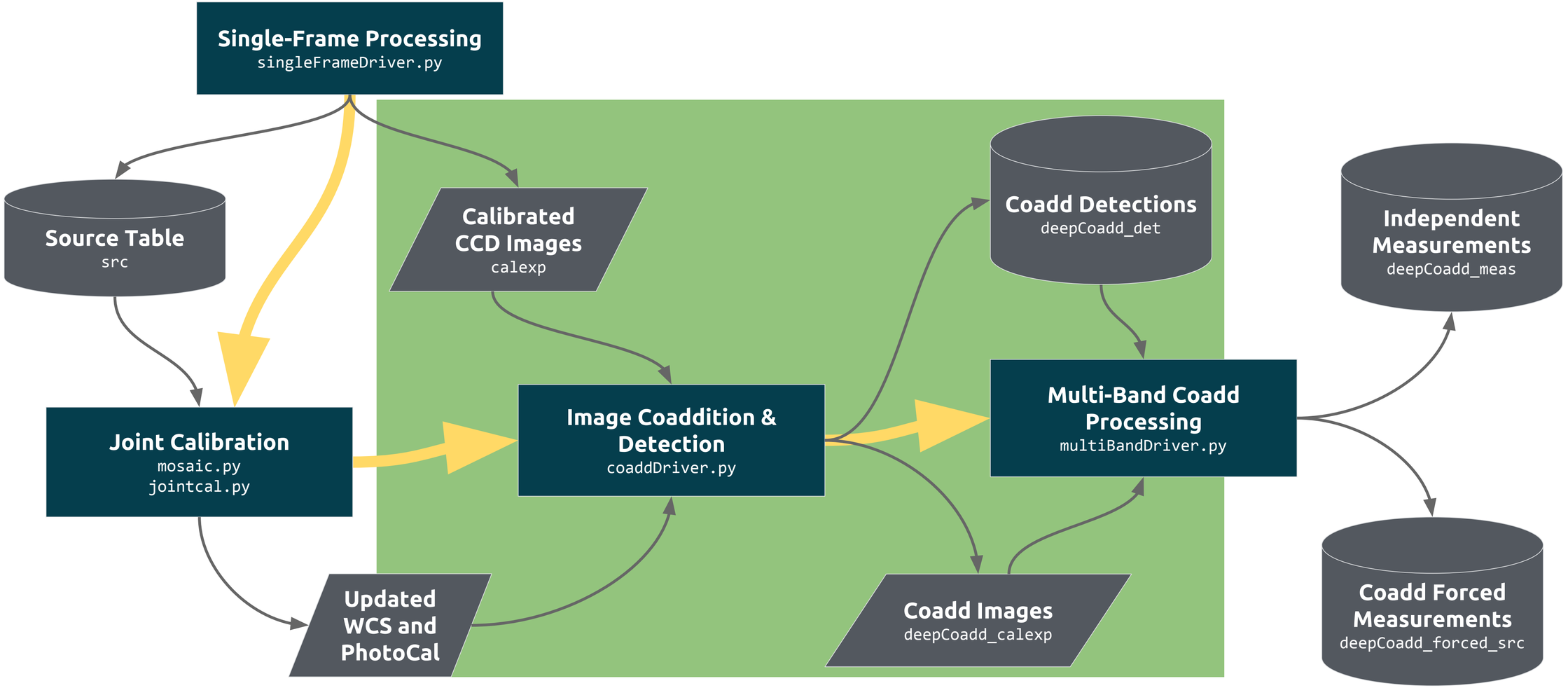
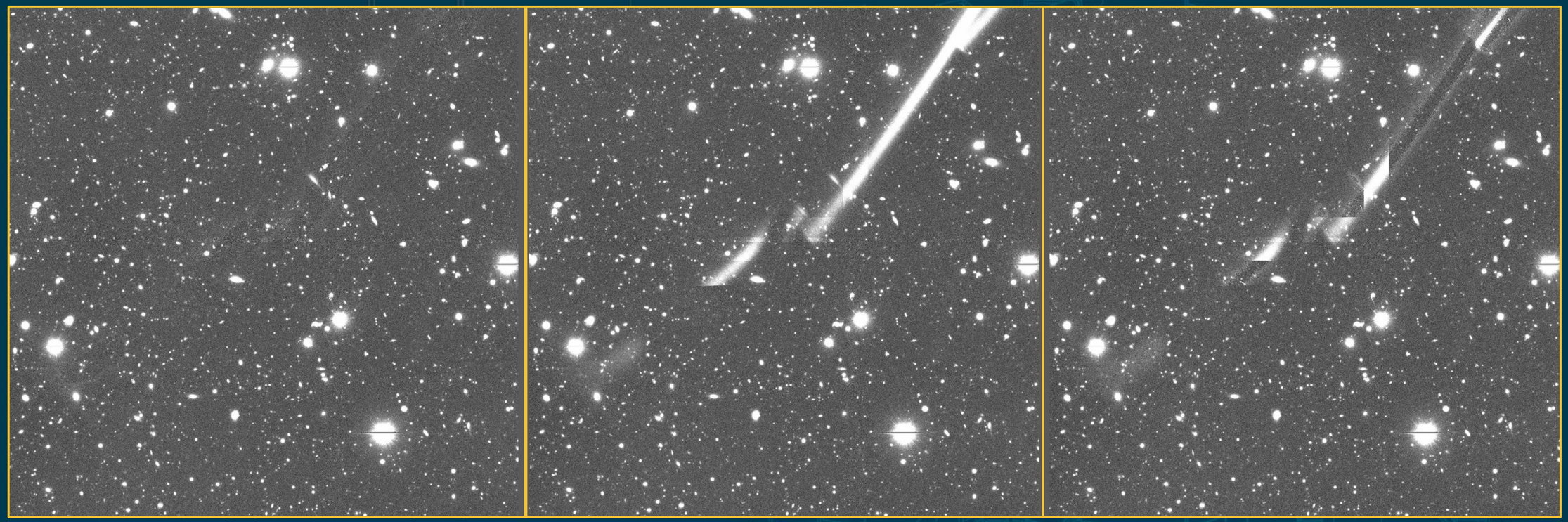

Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated
CCD Images
calexp
Coadd Detections
deepCoadd_det
Independent
Measurements
deepCoadd_meas
Joint Calibration
mosaic.py
jointcal.py
Image Coaddition &
Detection
coaddDriver.py
Multi-Band Coadd
Processing
multiBandDriver.py
Updated
WCS and
PhotoCal
Coadd Images
deepCoadd_calexp
Coadd Forced
Measurements
deepCoadd_forced_src

Image Coaddition

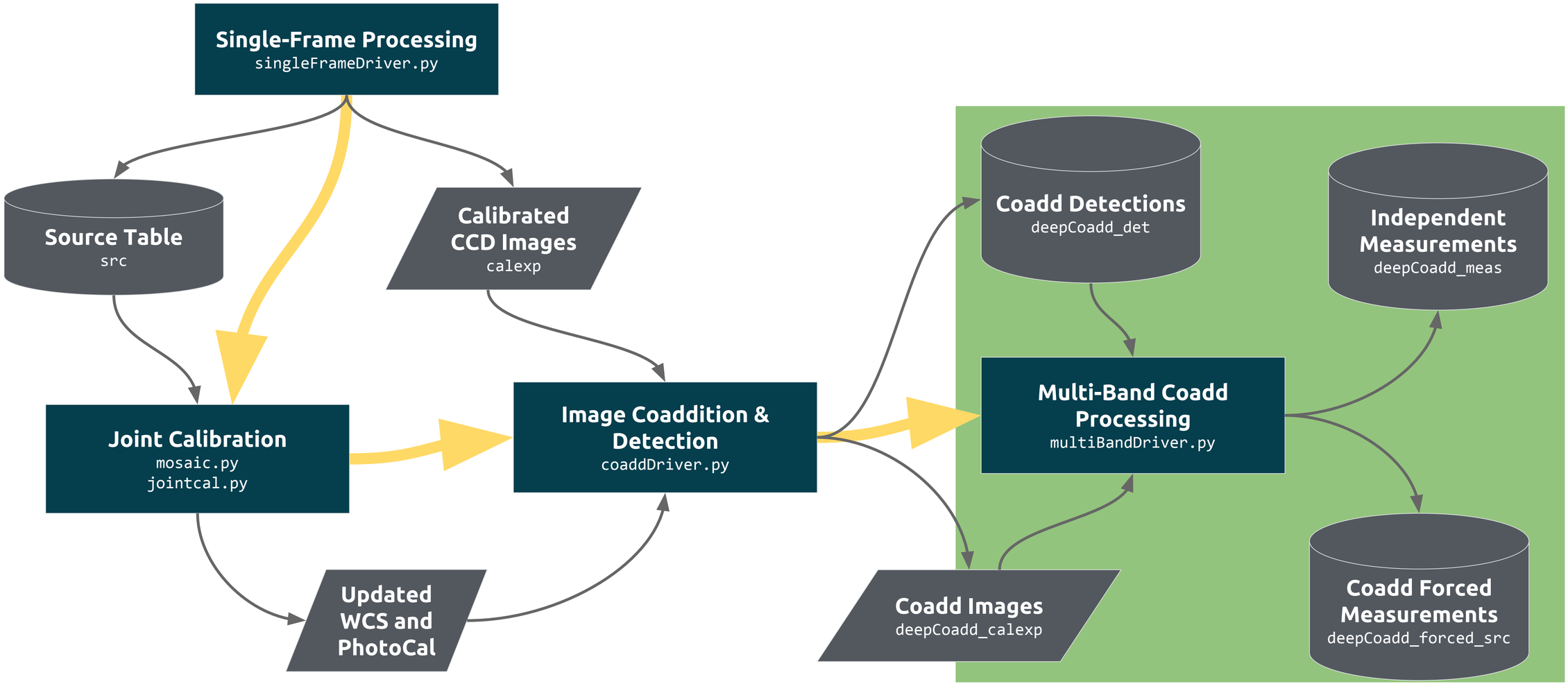
- Resample images to common coordinate system (`MakeCoaddTempExpTask`)
  - Assumes Nyquist-sampled input images (i.e. not appropriate for HST).
  - Creates a temporary image (`deepCoadd_tempExp`) for every visit+patch combination.

- Combine resampled images (`AssembleCoaddTask`)
  - Important not to do naive outlier rejection - makes coadd PSF ill-defined.
  - The PSF model on the coadd is computed by averaging visit PSF models with the same weights (the same for aperture corrections).

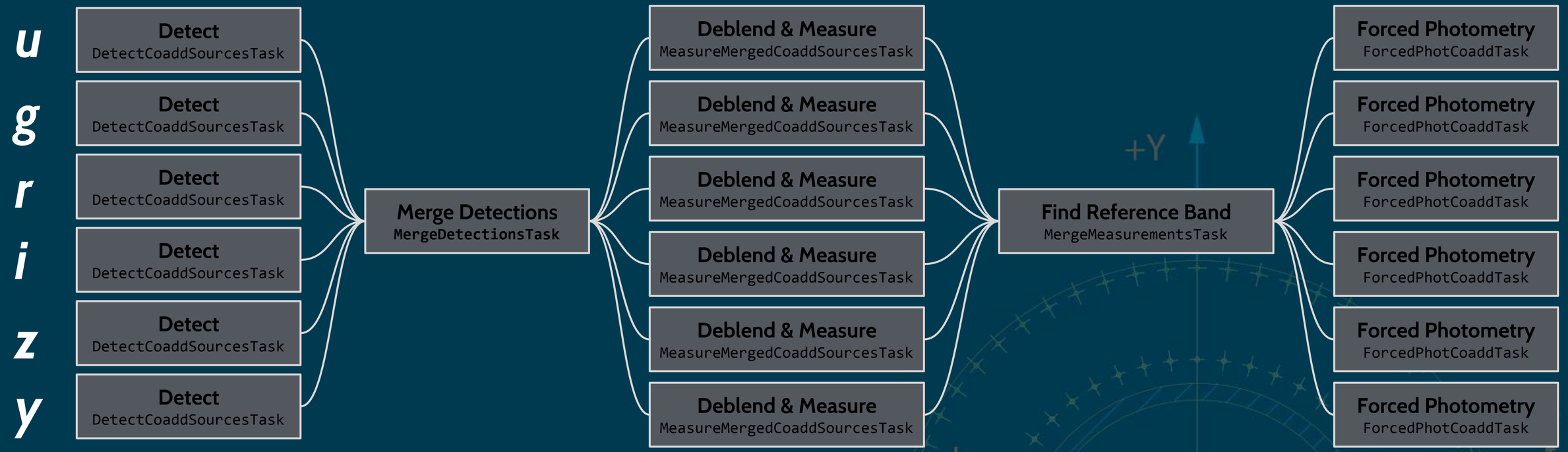
Safe Clipping in Coadds



safe clipping | direct mean | 3σ clipped mean

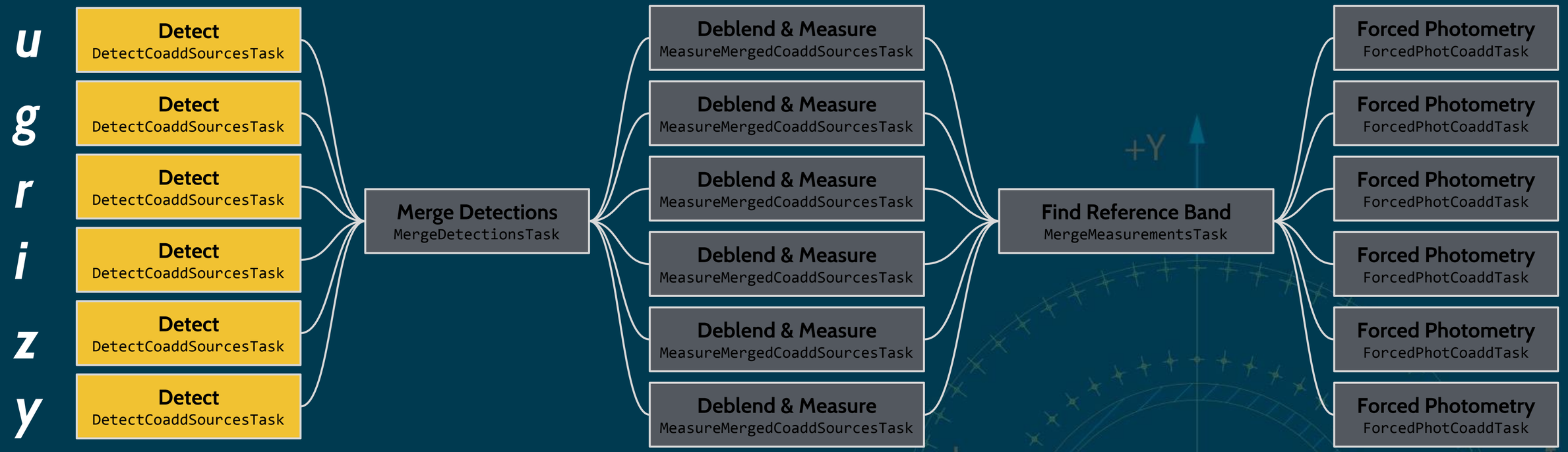

Single-Frame Processing
singleFrameDriver.py
Source Table
src
Calibrated CCD Images
calexp
Joint Calibration
mosaic.py
jointcal.py
Updated WCS and PhotoCal
Image Coaddition & Detection
coaddDriver.py
Coadd Detections
deepCoadd_det
Independent Measurements
deepCoadd_meas
Multi-Band Coadd Processing
multiBandDriver.py
Coadd Images
deepCoadd_calexp
Coadd Forced Measurements
deepCoadd_forced_src

Multi-Band Coadd Processing



This is our attempt to generate consistent Objects across all bands despite a *temporary*, *artificial* constraint: we can't have image data from all bands in memory at once.

Multi-Band Coadd Processing

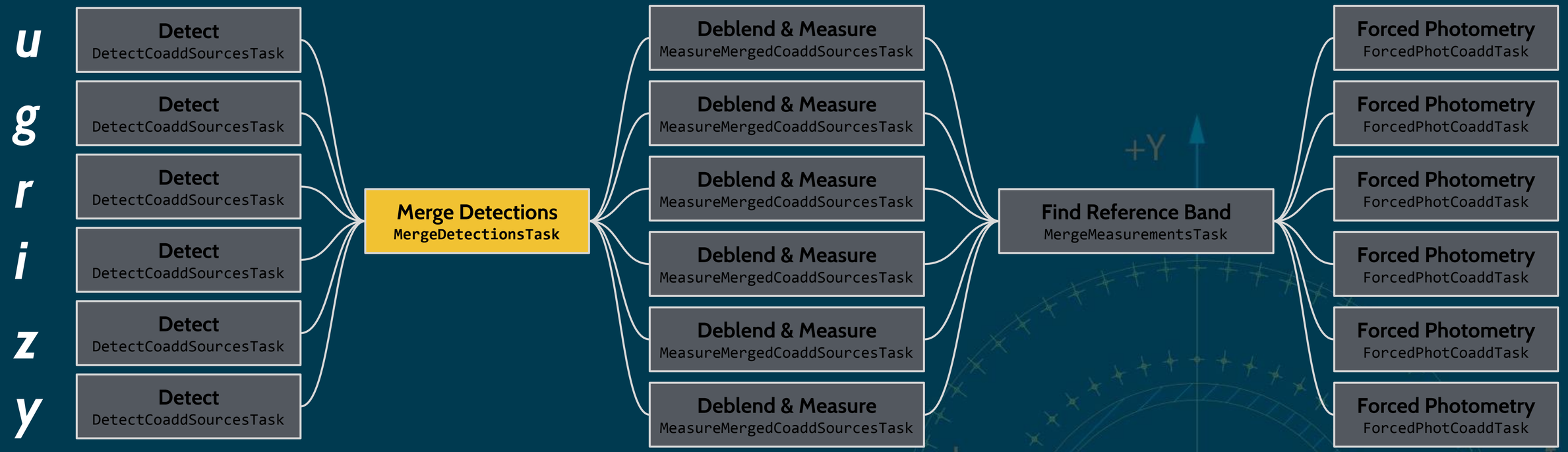


Detect in each band separately. Generates:

- *Footprints* (regions of above-threshold pixels)
- *Peaks* (peaks in a smoothed image.

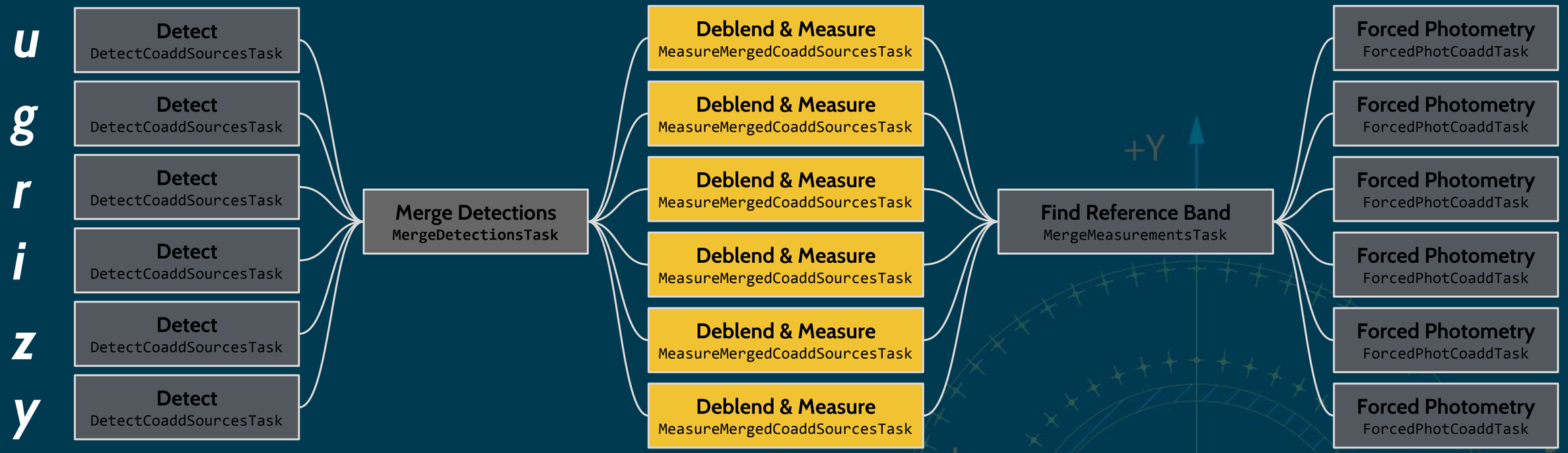
Actually done by coaddDriver.py to avoid writing two coadd images.

Multi-Band Coadd Processing



- Merge Footprints that overlap (union).
- Propagate all Peaks, and merge those that are close enough.
- Requires a priority order for bands (HSC uses irzyg) - Peak position comes from higher-priority bands.

Multi-Band Coadd Processing

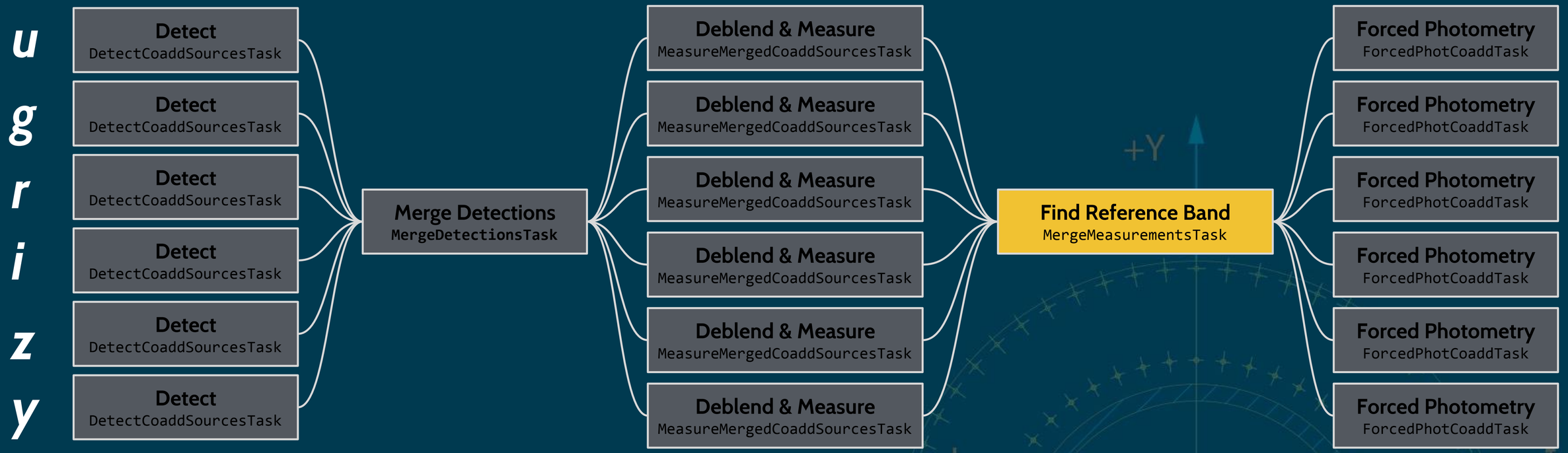


In each band, independently:

- Deblend each Peak within a Footprint into a child Object.
- Measure all parent and child Objects.

Same Objects in all bands, but they may not use the same pixels.

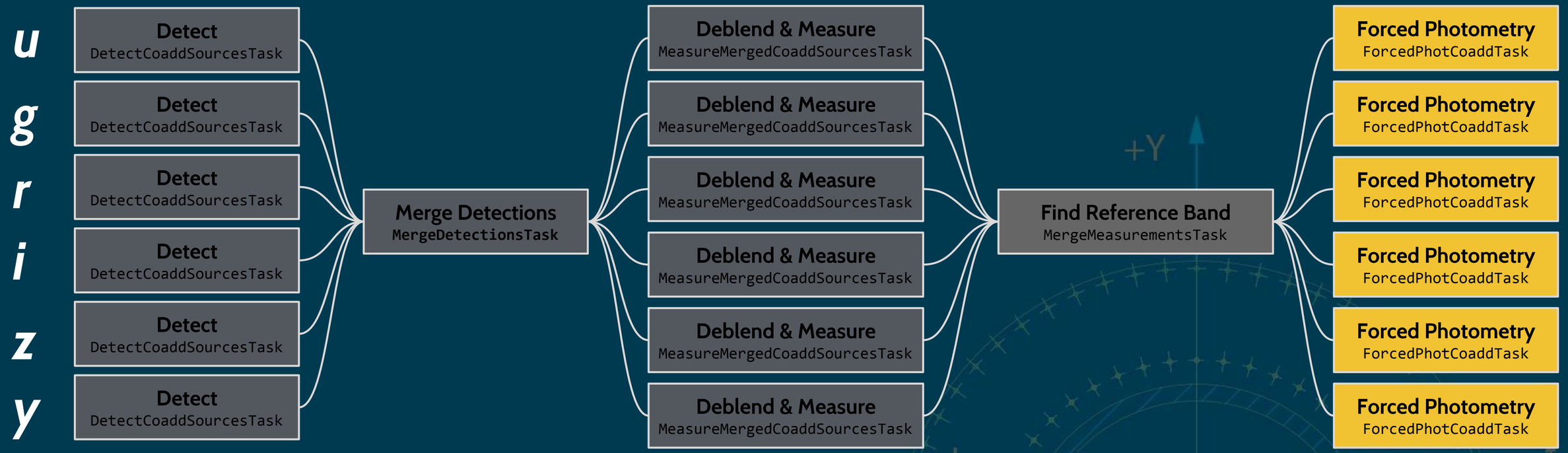
Multi-Band Coadd Processing



In band priority order (e.g. irzyg), choose the first band with both:

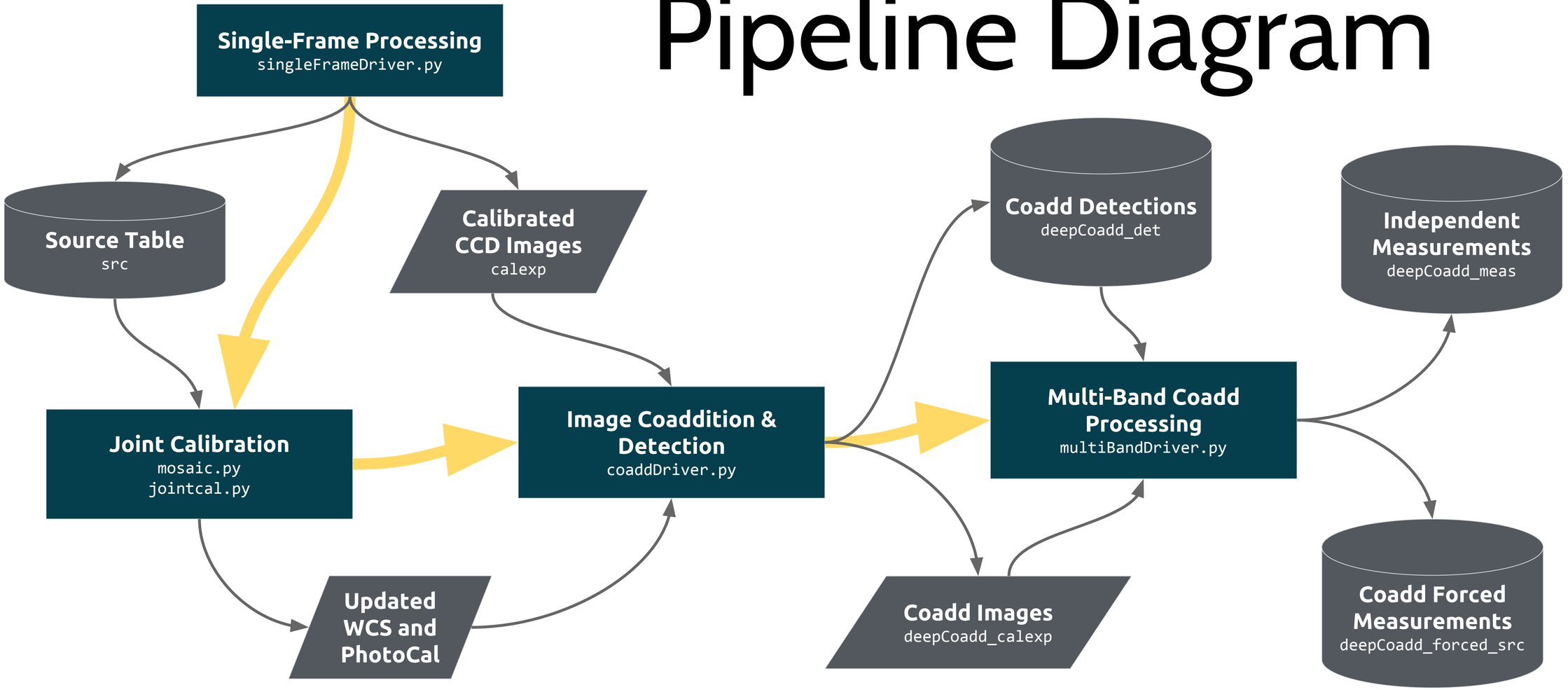
- SNR > 10
- SNR - SNR(previous) > 3

Multi-Band Coadd Processing

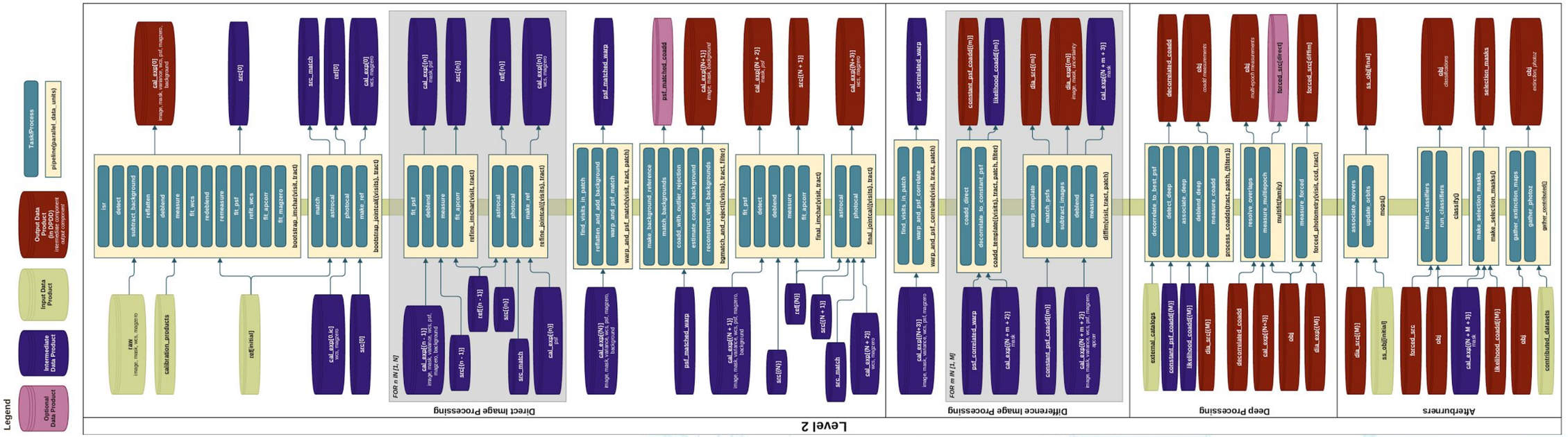


Measure in each band, using positions and shapes previously measured in the reference band.

Pipeline Diagram



Pipeline Diagram *From The Future*



State of the Pipeline: Overview

- It's State of the Art: this is the highest-quality *general-purpose, general-dataset* pipeline in astronomy.
  - It's not the best *single-purpose* pipeline: customized SExtractor+X runs can do better for weak lensing or detecting high-z objects, etc; DOPHOT can do *much* better in crowded fields.
  - It's not the best *single-dataset* pipeline (compare to e.g. SDSS Photo and PanSTARRS IPP).
- LSST needs to build a general purpose pipeline: one catalog to rule them all.
  - This is hard: different science cases prefer to make different mistakes.

State of the Pipeline: Deblending

- We're using a partial reimplementation of the SDSS deblender.
  - Doesn't have guards against galaxy shredding (and they do get shredded).
  - Only works on a single band at a time (makes it hard to throw away junk).

- LSST's blending problem is *much* worse than SDSS's; the SDSS algorithm probably just won't be sufficient.

For more on our plans for the future, come to the next session.

State of the Pipeline: PSF Models

- We can plug in PSF modeling and star selection algorithms separately.
- Our current best plugin is a custom version of Emmanuel Bertin's PSFEx (refactored to be callable as a library).
- Eventually we want to model the PSF over the full focal plane in wavefront space.

State of the Pipeline: Galaxy Photometry

- CModel algorithm fits:
  - PSF-convolved exponential model (ellipticity, orientation, radius, flux)
  - PSF-convolved de Vaucouleur model (ellipticity, orientation, radius, flux)
  - a linear combination of each of the above, with only the two fluxes varied.

- This works well for most objects.
- It works poorly on very low (SNR < 5) objects.
- It runs very slowly, especially on very large objects.
  - What it's doing is simply quite computationally expensive - it should speed up in the future, but perhaps not by much.

State of the Pipeline: Crowded Fields

No attempt to do this well yet, but if it's only moderately crowded the pipeline may do okay.

Come to the next session to hear more on this too.

State of the Pipeline: Galaxy Shapes

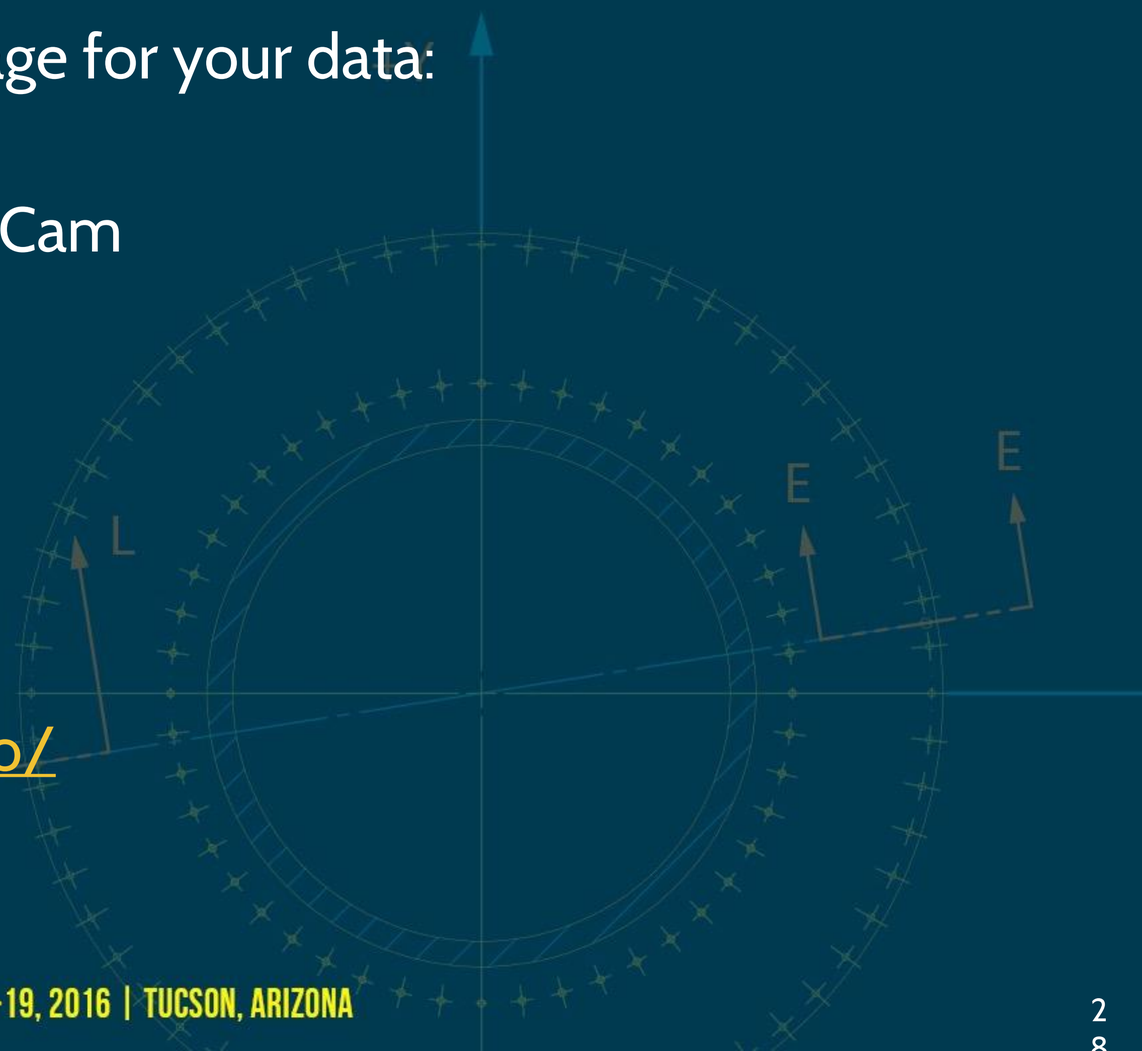
- We have a wrapper for GalSim's HSM algorithms (e.g. Regaussianization).
  - Battle-tested.
  - Fast.
  - Robust.
  - And biased.
- We don't yet have a "harness" for more modern methods that use all exposures simultaneously or do Monte Carlo sampling.

How to Use This Code

- You'll need an existing support (`obs_*`) package for your data:
  - CFHT Megacam
  - Subaru Hyper Suprime-Cam and Suprime-Cam
  - CTIO DECam
  - LSST Sims (PhoSim or lookalike)
  - SDSS

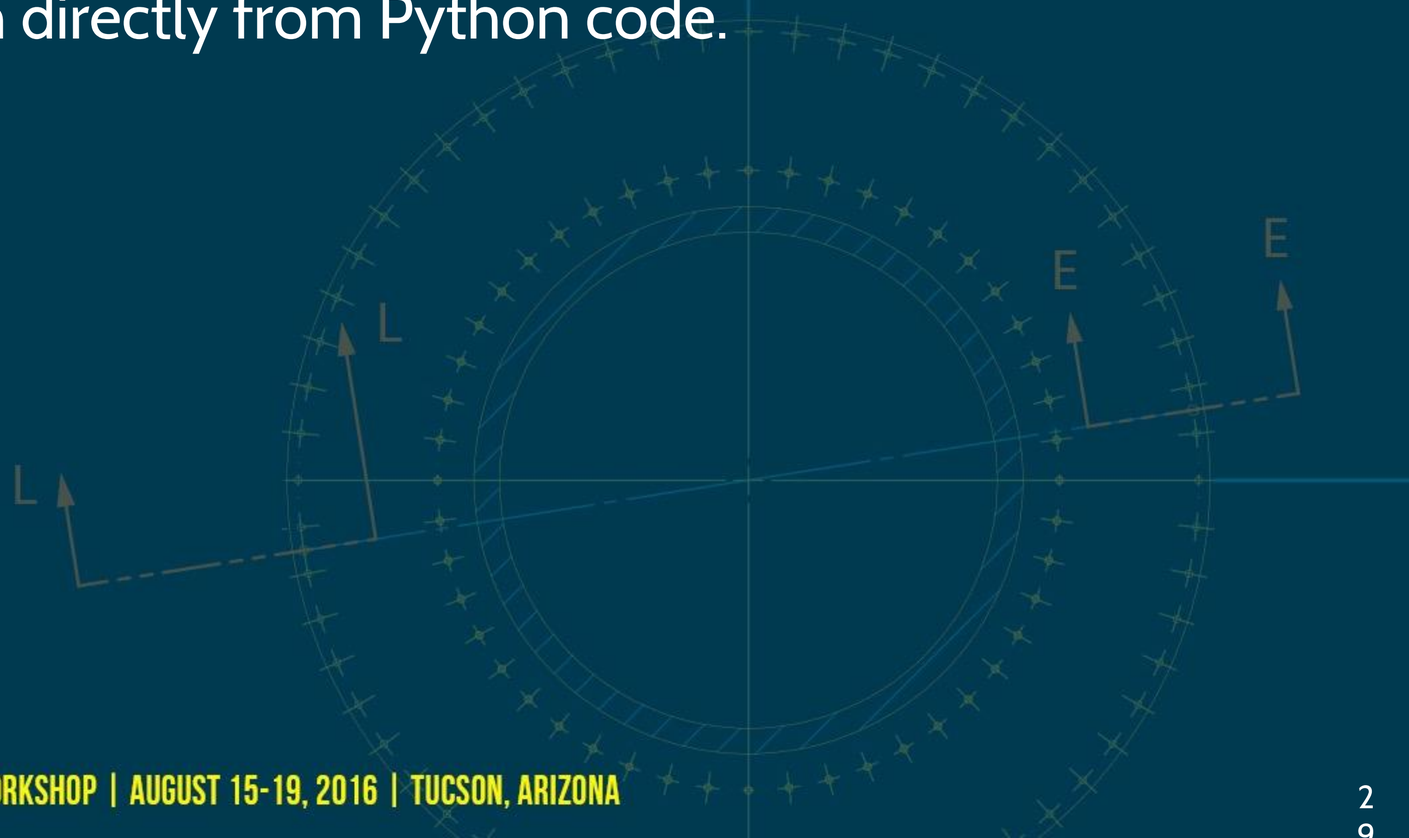
...or you'll need to write your own (hard).

Check out the tutorial: https://dmtn-023.lsst.io/

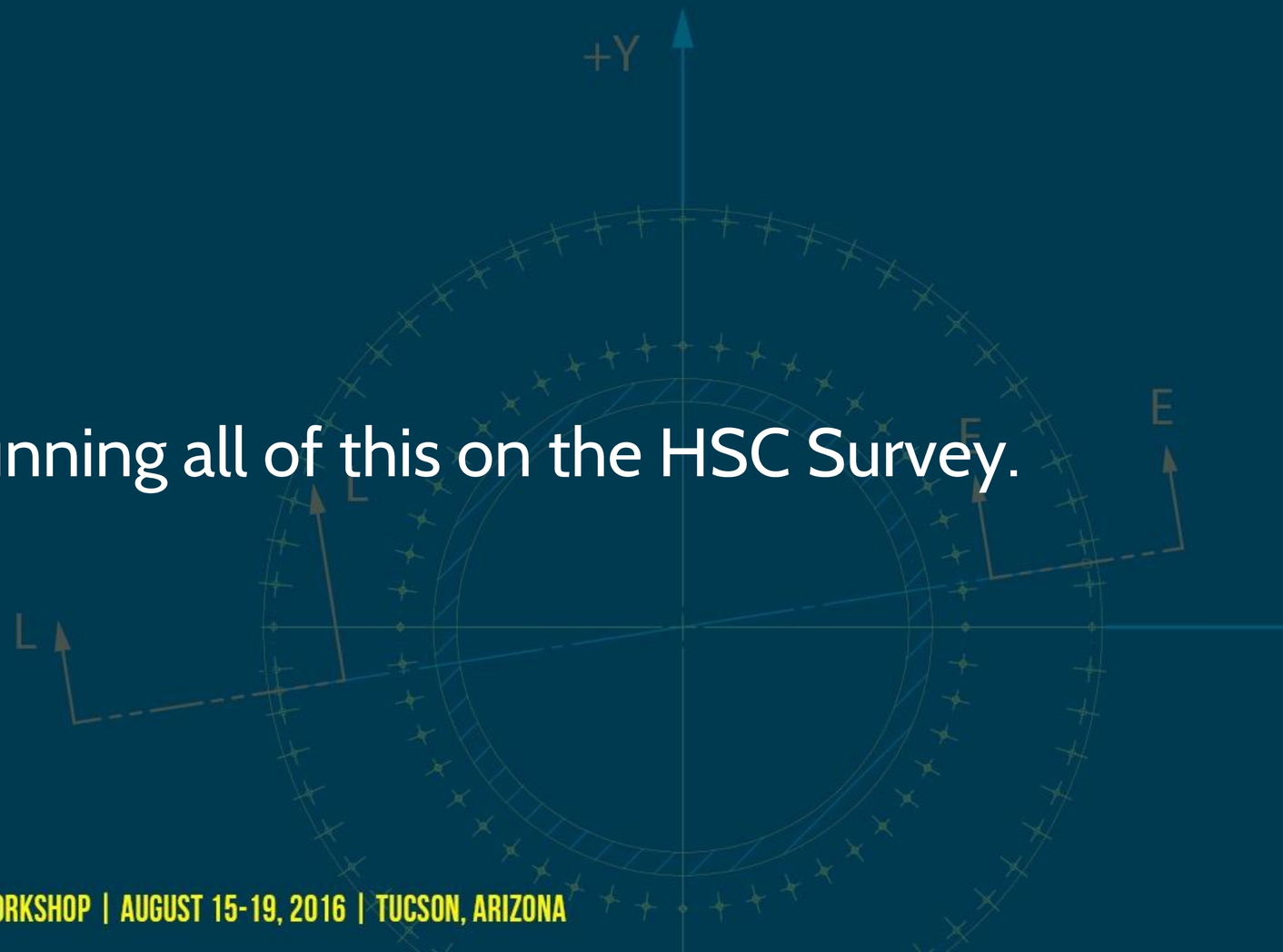


How to Use This Code

If you want to use the stack on small-scale simulated data (e.g. postage stamps), you're *probably* better off just using it as a library. Every step I've discussed here can be imported, configured, and run directly from Python code.



Want to see it in action?



That's the next talk: we're running all of this on the HSC Survey.