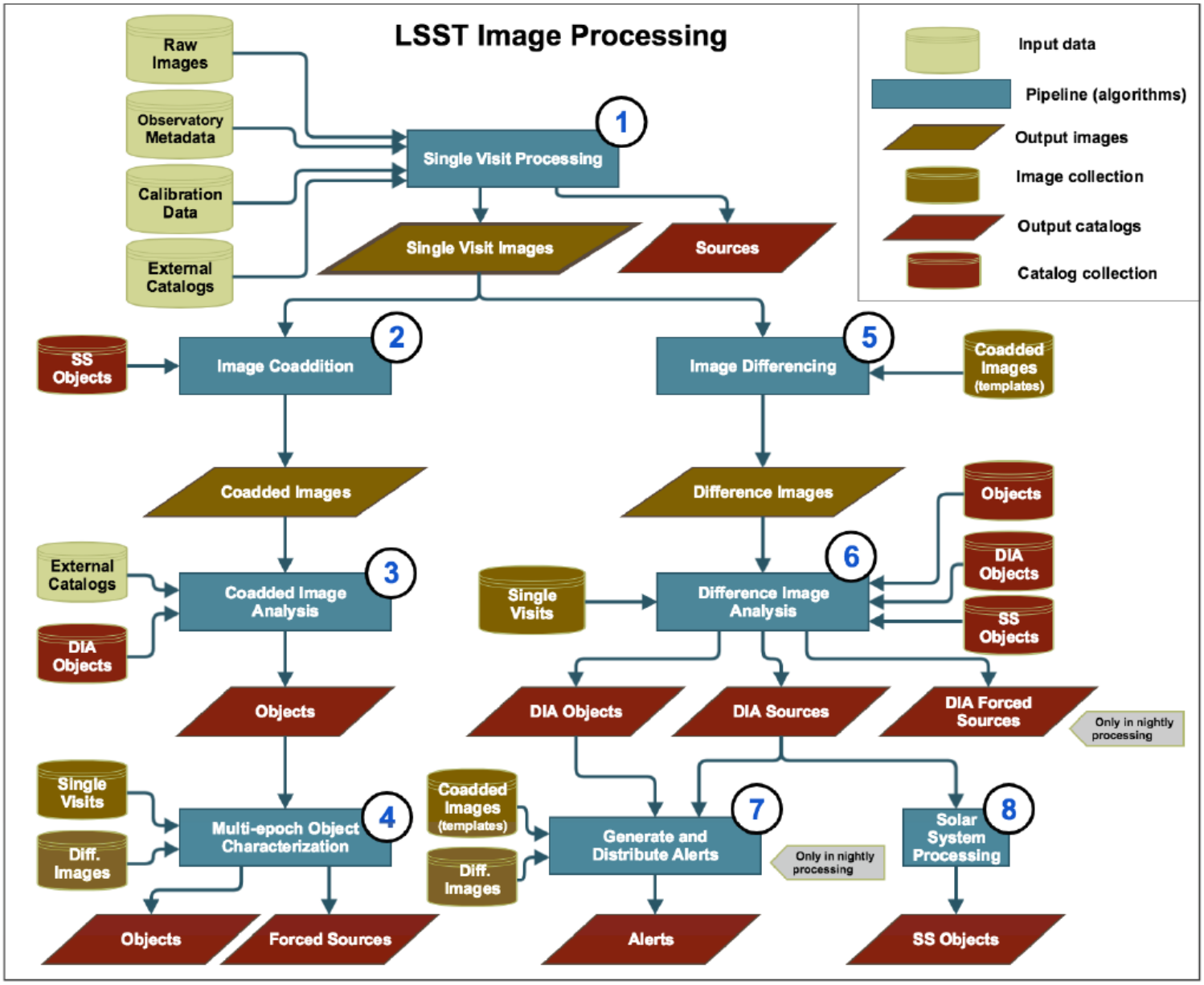
DM pipelines
ISR
Running an ISR job

Thibault Guillemin

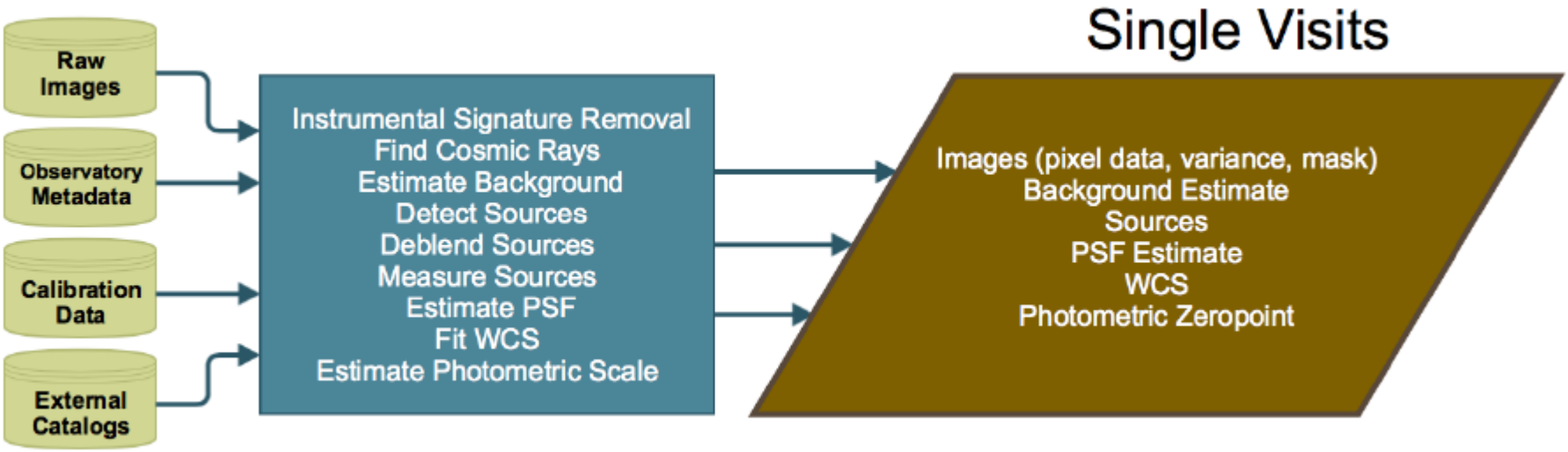
Focal plane commissioning workshop

March 27-29, 2029

Overall view

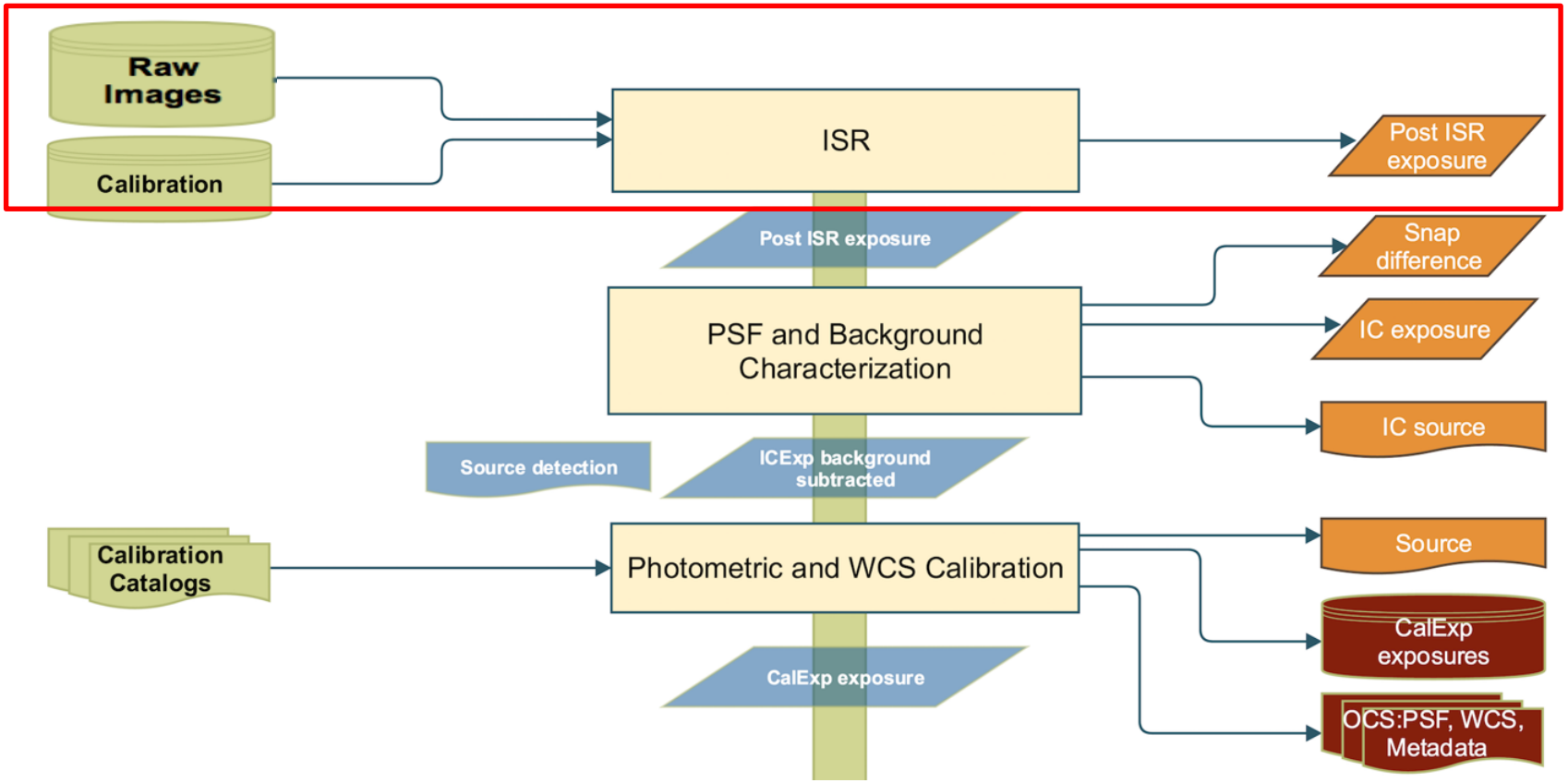


Single visit processing (1/2)



Single visit processing (2/2)

Today's focus



Example: AuxTel image

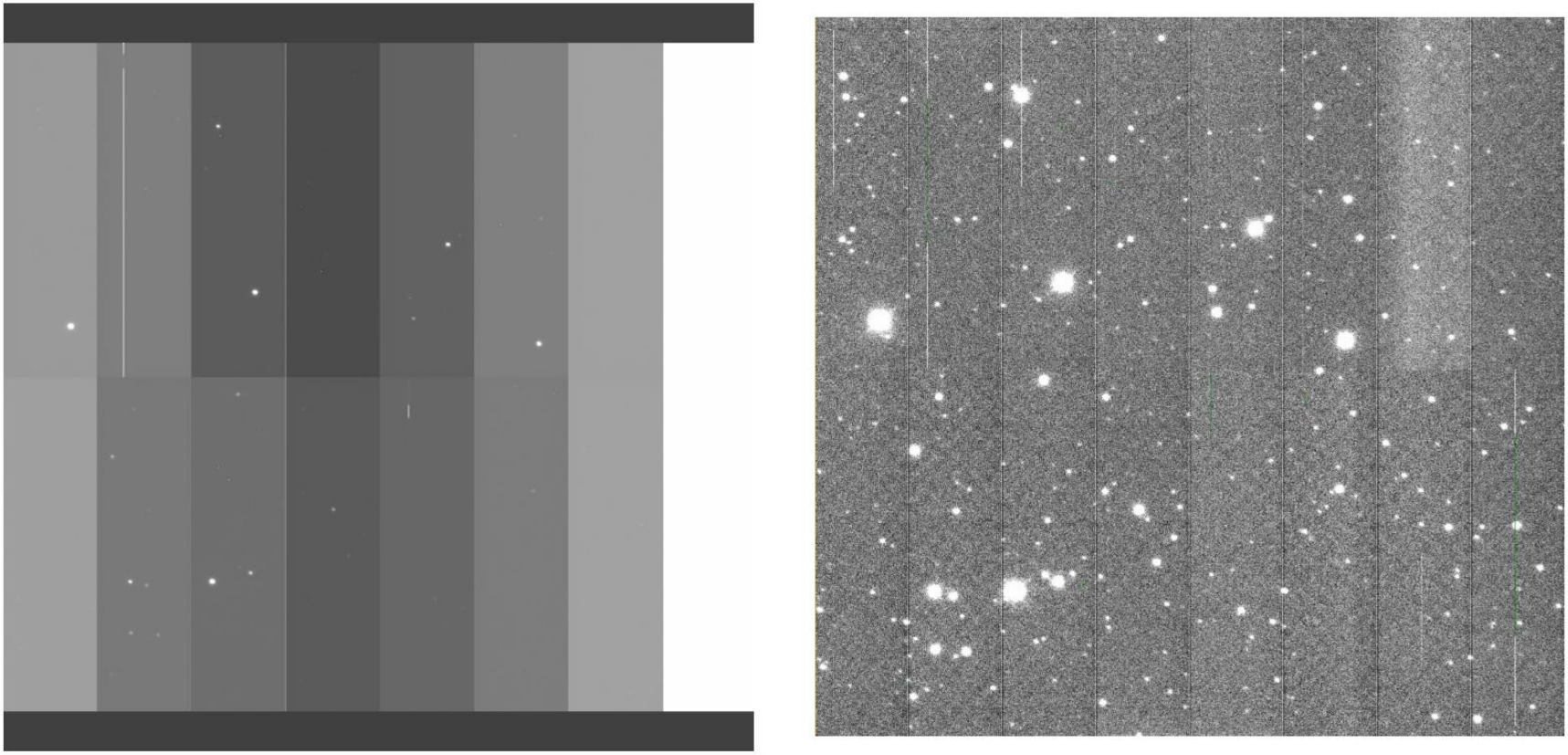


Figure 1. Example of an image before and after partial Instrument Signature Removal (bias, dark, and flat applied).

From A. Malagon

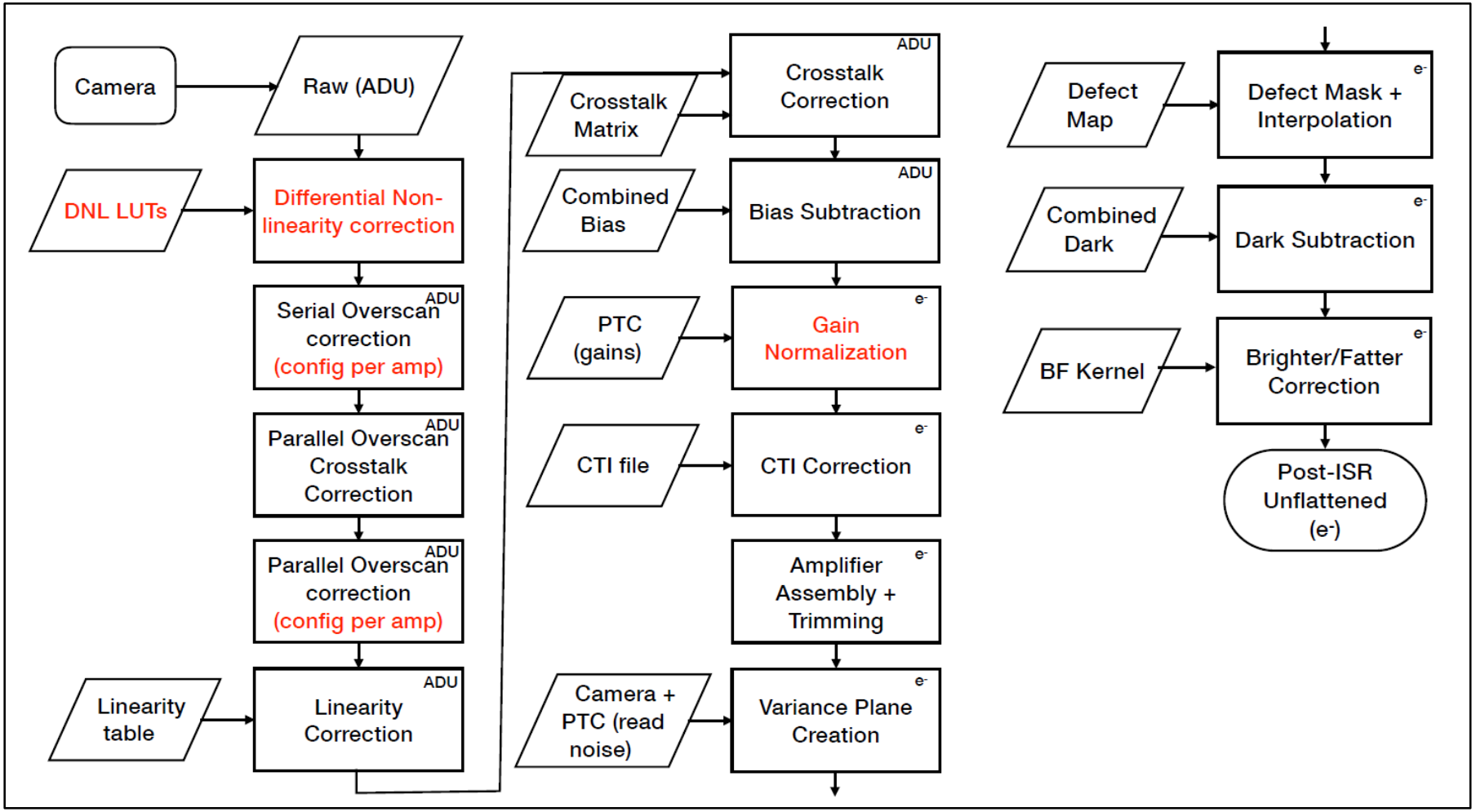
ISR steps in words

1. Integer to float conversion.
2. Correction of the non-linearities of the electron to ADU relation.
3. Correction of the serial overscan.
4. Bad amplifier and `SATURATED`/`SUSPECT` pixel masking.
5. Correction of the crosstalk from the image to the parallel overscan region.
6. Correction of the parallel overscan.
7. Linearity correction.
8. Correction of the crosstalk between amplifiers.
9. Bias subtraction, using a combined bias frame.
10. Charge-Transfer Inefficiency (CTI) correction.
11. CDD assembly and trimming.
12. Variance calculation (variance image construction).
13. Defect mask interpolation.
14. Saturation trail widening.
15. Dark subtraction, using a combined dark frame.
16. Brighter-fatter effect correction.
17. Fringe correction for exposures taken with redder filters.
18. Flat-field corrections to normalize image to units of fluence.

From A. Malagon

Optimization and validation of each step

ISR steps in boxes



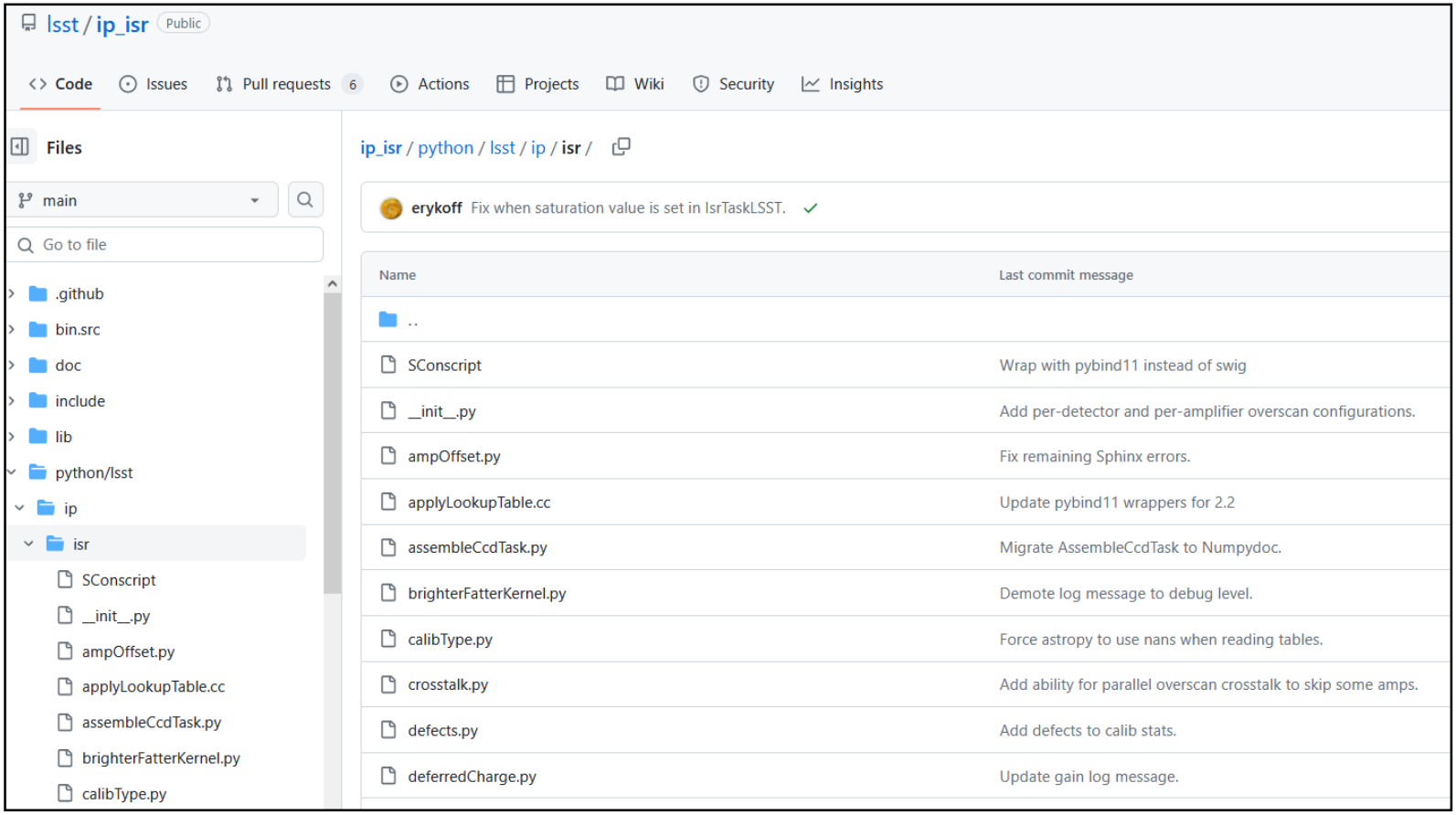
From E. Rykoff

ISR for real

README License

Description

The ip_isr package provides Instrument Signature Removal related tasks. ISR includes steps such as combining multiple amplifiers into one full CCD image, corrections for overscans, crosstalk, bias and dark frames, and the creation of variance and mask planes.



VERA C. RUBIN OBSERVATORY

Pipelines

- Sets of PipelineTasks can be grouped together to create workflows which are referred to as Pipelines
- Pipelines are declaratively written in YAML

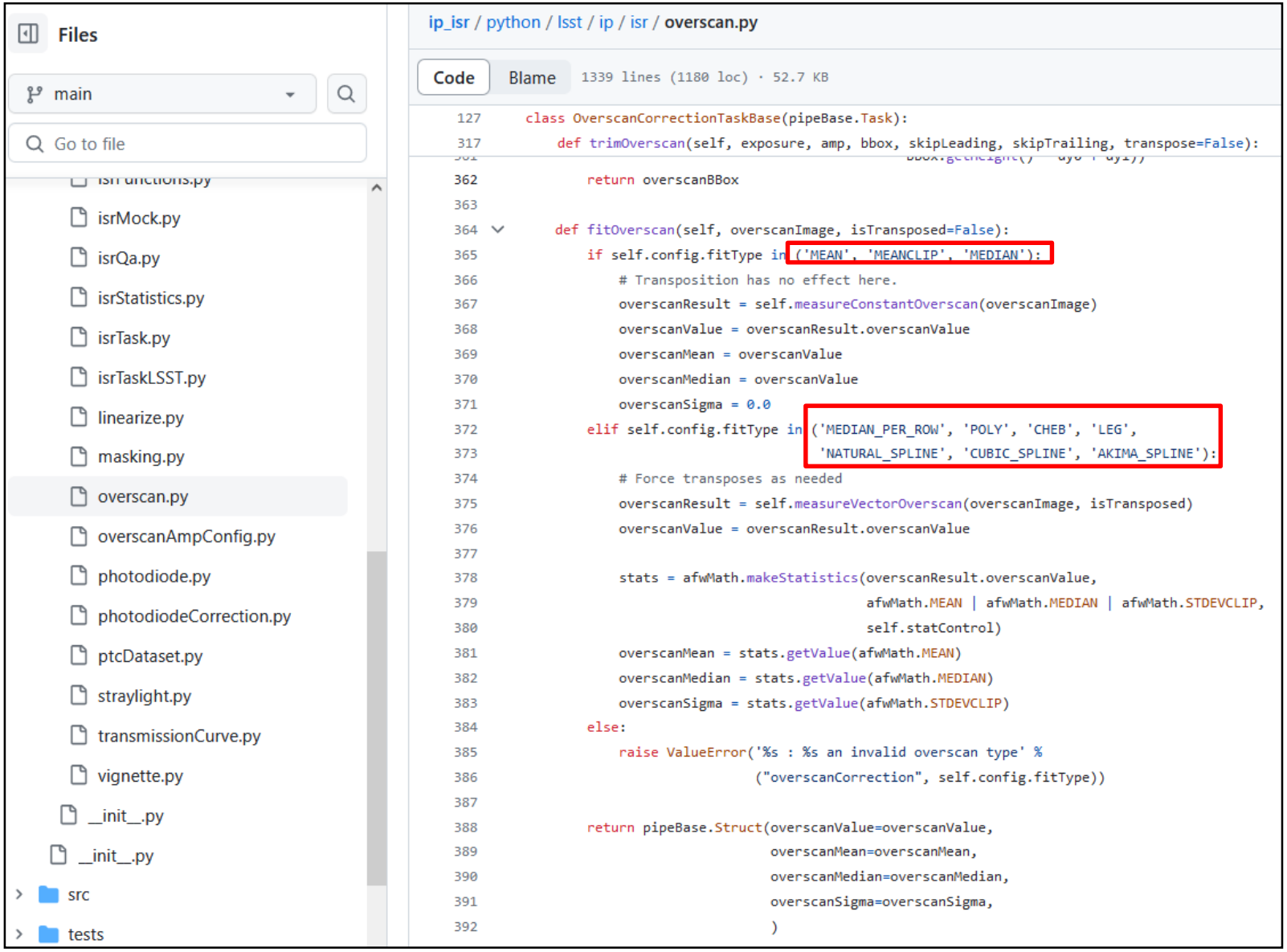
```yaml
description: A demo pipeline in the how-to guide
tasks:
  isr:
    class: lsst.ip.isr.IsrTask
    config:
      doVignette: true
      vignetteValue: 0.0
  characterizeImage: lsst.pipe.tasks.characterizeImage.CharacterizeImageTask
  calibrate:
    class: lsst.pipe.tasks.calibrate.CalibrateTask
    config:
      astrometry.matcher.maxOffsetPix: 300
```

- Pipelines contain at minimum a description of the Pipelines purpose, and a set of tasks to run.
- Tasks are specified with a label pointing to a (python namespace resolved) task

Documentation / tutorial can be found here

From N. Lust

One task of the ISR pipeline



Contracts and subsets

Configuration can be set in many different ways at different times. Some Pipelines may require a very specific set of configuration. To validate the supplied config Pipelines have `contracts` which validate the final configuration:

```
contracts:
  - characterizeImage.applyApCorr == calibrate.applyApCorr"
```

Pipelines define complete workflows, but sometimes it is helpful to run only part of a Pipeline. For this a Pipeline author can create a special subset label that can refer to a specific set of task labels. This label can be used to refer to this set of tasks in other contexts.

```
subsets:
  processCcd:
    - isr
    - characterizeImage
    - calibrate
```

From N. Lust

Real example

butler

pipeline

//

inputs

outputs

contracts

data query

```
export REPO=/sps/lsst/groups/FocalPlane/SLAC/run6/butler/main/butler.yaml

EXPOSURES='3023102600140'

pipetask --log-level DEBUG --long-log run -b $REPO \
    -p cp_pipe/pipelines/cpBias_corr.yaml \
    -j 8 \
    -i LSSTCam/raw/all,LSSTCam/calib,u/lsstccs/defects_13392_w_2023_24 \
    -o u/tguillem/run_6_validation/debug_run_6b_0D_20231027a \
    -c isr:doDefect=False \
    -d "instrument='LSSTCam' AND exposure IN ($EXPOSURES) AND
detector.full_name='R13_S11'" \
    --register-dataset-types
```

Using a local version of a stack package

- Recipe:

```
git clone https://github.com/lsst/cp_pipe.git
cd cp_pipe
git checkout w.2022.34
setup -k -r .
scons -j 4
eups declare -r . -t tguillem
setup cp_pipe -t tguillem
```

To do do every time

- If you want to store your modifications in a branch, you will have to fork the repository (not allowed to push in the official git respository by default).

Example:
https://github.com/tguillemLSST/ip_isr/blob/main/python/lsst/ip/isr/overscan.py#L49