VERA C. RUBIN OBSERVATORY

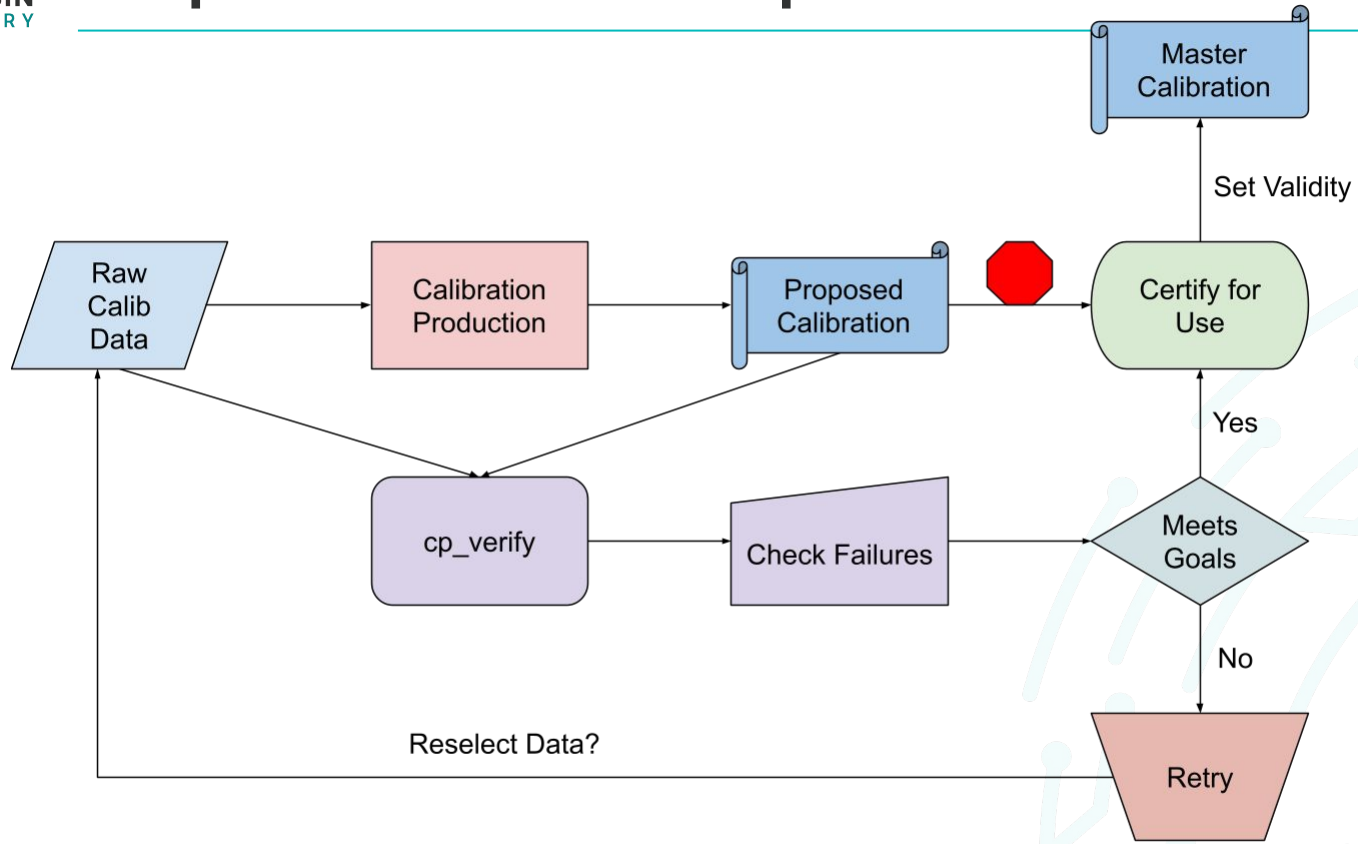
Introduction to cp_verify

March 15, 2022

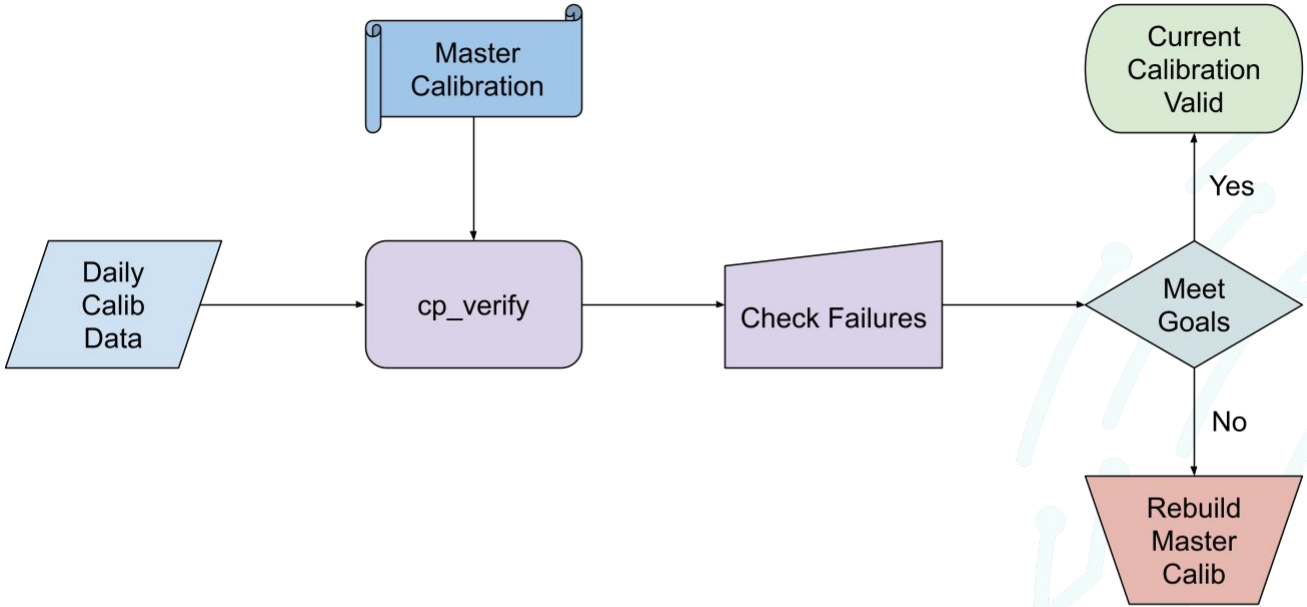
What is cp_verify?

- Measure some set of metrics on a calibration residual image:
  - Example: Apply overscan and master bias to bias frames; measure image mean and scatter.
  - Expectation: mean = 0, scatter = read noise
- Some calibrations (crosstalk, linearity, etc) will attempt to remeasure the values from the residual images and confirm there is no residual signal.
- If the metrics are all within DMTN-101 limits, the calibration is valid and can be certified for use.
  - Certification assigns the date range within which the new master calibration will be used.
  - End date usually not known.
- Used with daily calibrations to confirm that existing master calibrations are still good. Monitor the camera/telescope stability.
- DMTN-101 will be updated and partially rewritten once all calibrations have verification code.

As part of calibration production:



As part of afternoon checks:

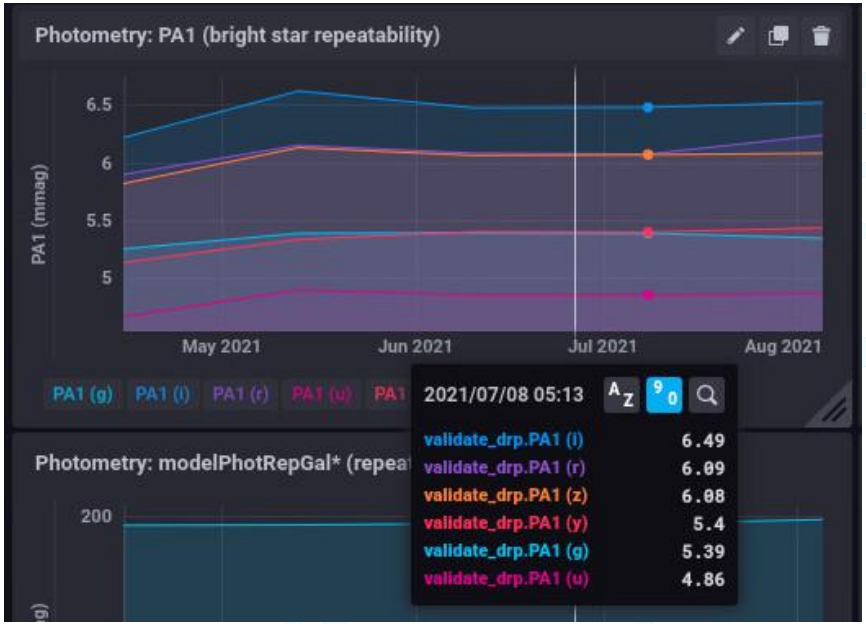


What do the metrics look like now?

- Bias (bias corrected bias exposures):
  - Mean consistent with zero
  - Clipped stdev consistent with read noise.
  - CR rejected stdev consistent with read noise.
- Dark (bias, dark corrected dark exposures):
  - Same as bias metrics.
- Flat (bias, dark, flat corrected flat exposures):
  - Noise consistent with Poissonian.
  - Amp-to-amp mean scatter small.
  - Detector-to-detector mean scatter small.
- Brighter-fatter correction (full ISR processed science exposures):
  - Slope of source second-moment size as a function of source magnitude small.
- Zero-residual tests are in development for crosstalk, linearity, and fringes.

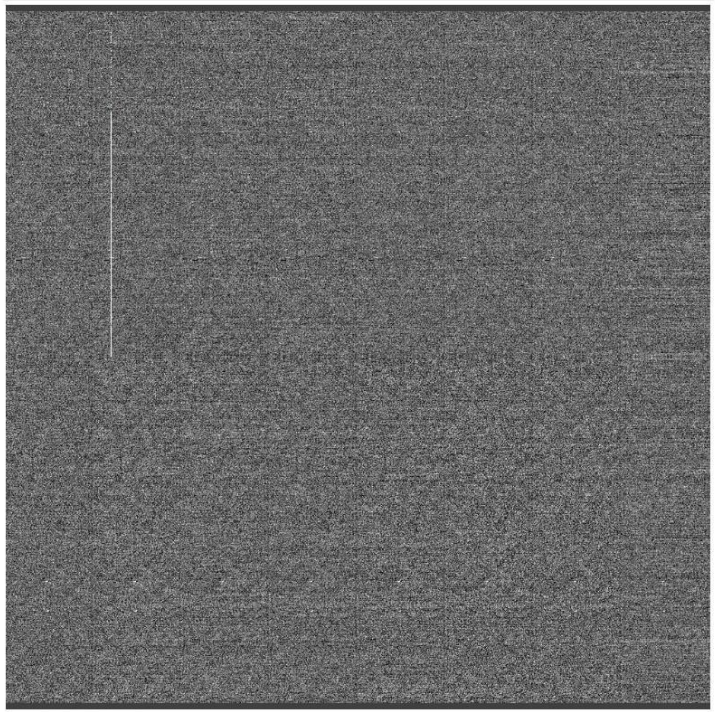
Metrics handling

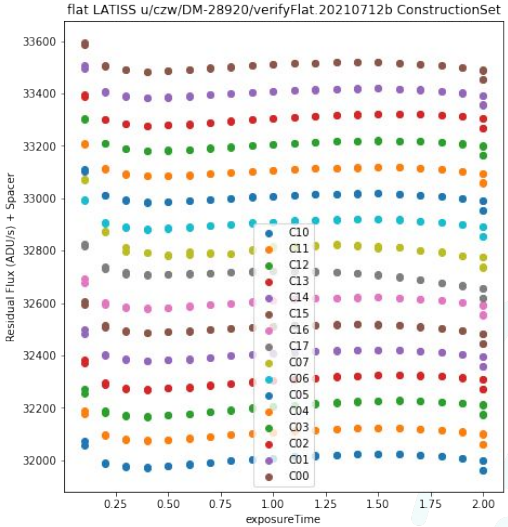
- We would like to have the cp_verify metrics available in a database.
- DM's faro package passes other types of metrics for display in the chronograf system.
  - System already built.
- Currently metrics are written as yaml.
  - Easy to work with and read.
  - Flexible as we develop cp_verify.
- These will need to be translated for faro.
- Open to other solutions, but want to avoid redesigning something that exists.

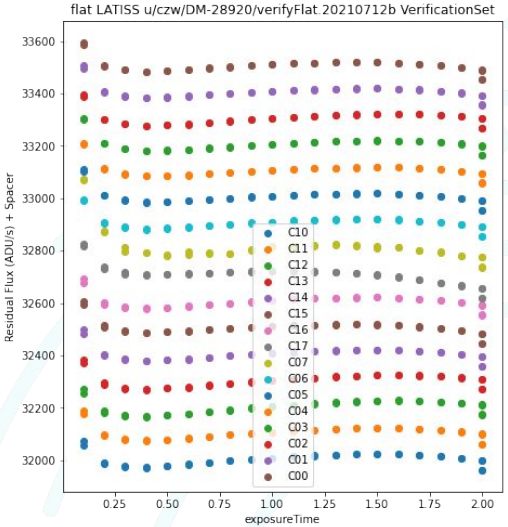


Visualizing cp_verify results.

```
In [9]: # IW = astrowidgets.ImageWidget(image_width=1000, image_height=1000)
display = afwDisplay.Display(dims=(1000, 1000))
display.embed()
```

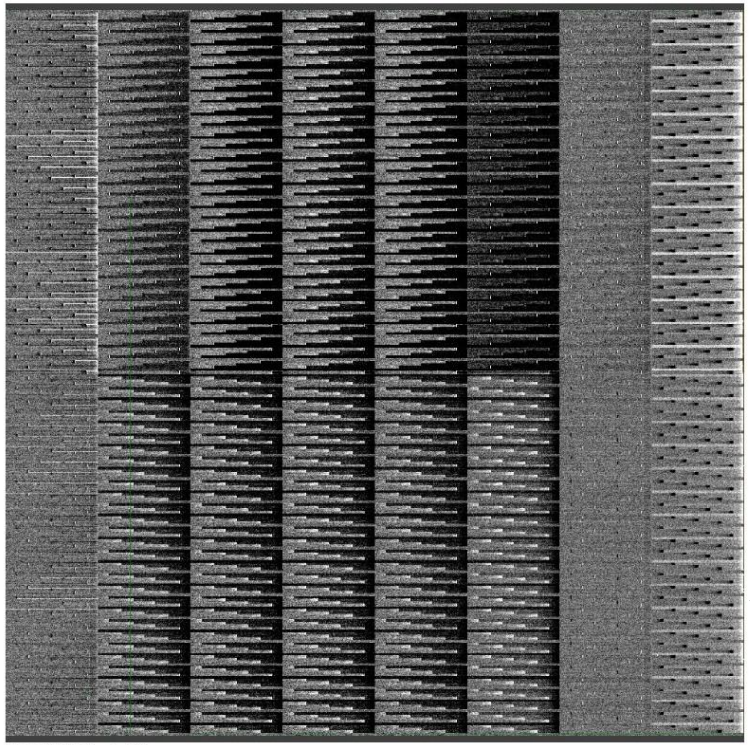






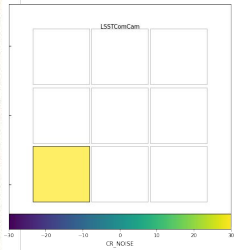
Does correctly catch bad data:

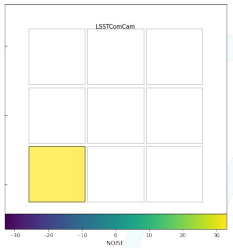
```
In [5]: # IW = astrowidgets.ImageWidget(image_width=1000, image_height=1000)
display = afwDisplay.Display(dims=(1000, 1000))
display.embed()
```

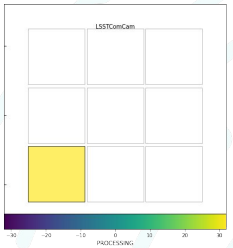


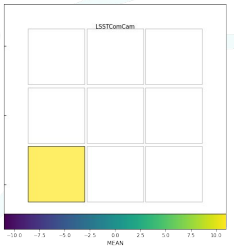

```
In [28]: failureTable(runStats)
```

Exposure	Detector	NOISE	MEAN	CR_NOISE	PROCESSING
2021081900004	R22_S00	14	1	14	16
2021081900003	R22_S00	16	10	16	16
2021080500003	R22_S00	3	0	0	0









OCPS Overview, and How it Ties Together:

- OCPS is the OCS Controlled Pipeline System.
  - OCS is the Observatory Control System.
- Runs the same pipeline tasks used for cp_pipe and cp_verify as part of a pre-defined script.
- The script configuration can define a set of exposures for the camera to take:
  - Number of exposures.
  - Exposure times for each.
  - Filter selection.
- This will be the interface the observers will use.
- Currently running bias, dark, flat production, verification, and automatic certification.

Conclusion

- Main development will likely end by summer 2022.
- Documentation of processes, tests, test criteria may extend somewhat.
- Visualization is currently a major issue:
  - Easy to do with LATISS
  - Unwieldy with ComCam
  - Will require better full focal plane visualization for final camera.
- Integration with observing means this can be run daily to monitor calibration quality; detect camera changes.
- Storing the results for time series analysis needs to be solved.