Draft version March 31, 2025
Typeset using LaTeX **twocolumn** style in AASTeX7

The LSST Science Pipelines Software: Optical Survey Pipelined Reduction and Analysis Environment

James F. Bosch [1] Yusra AlSayyad [1] Tim Jenness [2] Eric C. Bellm [3] Robert H. Lupton [1]
Nate B. Lust [1] Ian S. Sullivan [3] Christopher Z. Waters [1] Krzysztof Findeisen [3]
Erfan Nourbakhsh [1] Agnès F. Ferté [4] Arun Kannawadi [5,1] Eli S. Rykoff [6]
And Andrés A. Plazas Malagón [4,6]

The Rubin Observatory Science Pipelines Team

[1] Department of Astrophysical Sciences, Princeton University, Princeton, NJ 08544, USA
[2] Vera C. Rubin Observatory Project Office, 950 N. Cherry Ave., Tucson, AZ 85719, USA
[3] University of Washington, Dept. of Astronomy, Box 351580, Seattle, WA 98195, USA
[4] SLAC National Accelerator Laboratory, 2575 Sand Hill Rd., Menlo Park, CA 94025, USA
[5] Department of Physics, Duke University, Durham, NC 27708, USA
[6] Kavli Institute for Particle Astrophysics and Cosmology, SLAC National Accelerator Laboratory, 2575 Sand Hill Rd., Menlo Park, CA 94025, USA

ABSTRACT

The NSF-DOE Vera C. Rubin Observatory will produce the Legacy Survey of Space and Time (LSST) and produce 11 data releases over the ten-year survey. The LSST Science Pipelines Software will be used to create these data releases and to perform the nightly alert production. This paper provides an overview of the LSST Science Pipelines Software and describes the components and how they are combined to form pipelines.

Keywords: Astrophysics - Instrumentation and Methods for Astrophysics — methods: data analysis — methods: miscellaneous

1. INTRODUCTION

The NSF-DOE Vera C. Rubin Observatory will be performing the 10-year Legacy Survey of Space and Time (LSST; Ž. Ivezić et al. 2019) starting in 2025. Rubin Observatory is located on Cerro Pachon in Chile and consists of the 8.4 m Simonyi Survey Telescope with the 3.4-gigapixel LSSTCam survey camera performing the main survey and the Rubin Auxiliary Telescope providing supplementary atmospheric calibration data. The Data Management System (DMS; W. O'Mullane et al. 2022) is designed to handle the flow of data from the telescope, approaching 20 TB per night, in order to issue alerts and to prepare annual data releases. A central component of the DMS is the LSST Science Pipelines software that provides the algorithms and frameworks required to process the data from the LSST and generate the coadds, difference images, and catalogs to the user community for scientific analysis.

The LSST Science Pipelines software consists of the building blocks and pipeline infrastructure required to construct high performance pipelines to process the data from LSST. It has been under development since at least 2004 (T. Axelrod et al. 2004) and has evolved significantly over the years as the project transitioned from prototyping (T. Axelrod et al. 2010) and entered into formal construction (M. Jurić et al. 2017). The software is designed to be usable by other optical telescopes and this has been demonstrated with Hyper Suprime Cam on the Subaru Telescope in Hawaii (J. Bosch et al. 2018) and also with data from the Dark Energy Camera (DECam), the VISTA infrared camera (VIRCAM), the Wide Field Survey Telescope (WFST; M. Cai et al. 2025), and the Gravitational-wave Optical Transient Observer (GOTO; J. R. Mullaney et al. 2021).

In this paper we provide an overview of the components of the software system. This includes a description of the support libraries and data access abstraction, the pipeline task system, and an overview of the algorithmic components. We do not include details of the science validation of the individual algorithms. The other components of the LSST DMS, such as the workflow system (M. Gower et al. 2022), the Qserv database (D. L. Wang et al. 2011) and the Rubin Science Platform (M. Jurić et al. 2019), are not covered in this paper.

2. FUNDAMENTALS

The LSST Science Pipelines software is written in Python with C++ used for high-performance algorithms and for core classes that are usable in both languages. We use Python 3 (having ported from python 2, T. Jenness 2020, currently with a minimum version of Python 3.11), and the C++ layer can use C++17 features with pybind11 being used to provide the interface from Python to C++. Additionally, the C++ layer uses `ndarray` to allow seamless passing of C++ arrays to and from Python `numpy` arrays. This compatibility with `numpy` is important in that it makes LSST data structures available to standard Python libraries such as Scipy and Astropy (T. Jenness et al. 2016; Astropy Collaboration et al. 2018).

Although all the software uses the `lsst` namespace, the code base is split into individual Python products in the LSST GitHub organization[7] that can be installed independently and which declare their own dependencies. These dependencies are managed using the "Extended Unix Product System" (EUPS) (N. Padmanabhan et al. 2015; T. Jenness et al. 2018) where most of the products are built using the SCons system (S. Knight 2005) with LSST-specific extensions provided in the `sconsUtils` package enforcing standard build rules.

For logging we always use standard Python logging with an additional `VERBOSE` log level between `INFO` and `DEBUG` to provide additional non-debugging detail that can be enabled during batch processing. This verbose logging is used for periodic logging where long-lived analysis tasks are required to issue a log message every 10 minutes to indicate to the batch system that they are still alive and actively performing work. For logging from C++ we use Log4CXX wrapped in the `lsst.log` package to make it look more like standard Python logging, whilst also supporting deferred string formatting such that log messages are only formed if the log message level is sufficient for the message to be logged. These C++ log messages are forwarded to Python rather than being issued from an independent logging stream. Finally, we also provide some LSST-specific exceptions that can be thrown from C++ code and caught in Python.

As of April 2024, the Science Pipelines software is approximately 640,000 lines of Python and 225,000 lines of C++. The number of lines in the pipelines code as a function of time is given in Fig. 1.

[7] https://github.com/lsst

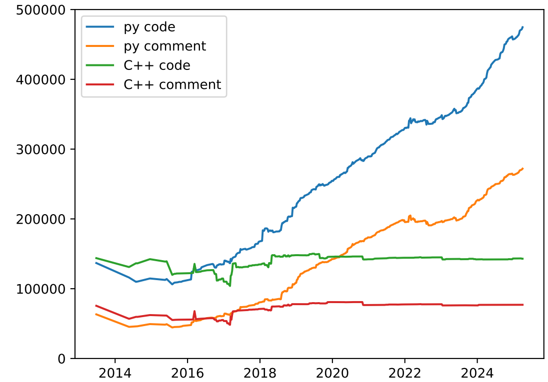

Figure 1. The number of lines of code comprising the LSST Science Pipelines software as a function of time. Line counts include comments but not blank lines. Python interfaces are implemented using `pybind11` and that is counted as C++ code. For the purposes of this count Science pipelines software is defined as the `lsst_distrib` metapackage and does not include code from third party packages.

2.1. *Python environment*

An important aspect of running a large data processing campaign is to ensure that the software environment is well defined. We define a base python environment using conda-forge via a meta package named `rubinenv`[8]. This specifies all the software needed to build and run the science pipelines software. A Docker container is built for each software release and the fully-specified versions of all software are recorded to ensure repeatability.

2.2. *Unit Testing and Code Coverage*

Unit testing and code coverage are critical components of code quality (T. Jenness et al. 2018). Every package comes with unit tests written using the standard `unittest` module. We run the tests using `pytest` (H. Krekel 2017) and this comes with many advantages in that all the tests run in the same process and requiring global parameters to be well understood, test can be run in parallel in multiple processes, plugins can be enabled to extend testing and record test coverage, and a test report can be created giving details of run times and test failures. Coding standards compliance with PEP 8 (G. van Rossum 2013) is enforced using GitHub actions and `pre-commit` checks. A Jenkins system provides the team with continuous integration facilities.

3. DATA ACCESS ABSTRACTION

3.1. *Butler*

Early in the development of the LSST Science Pipelines software it was decided that the algorithmic

[8] https://github.com/conda-forge/rubinenv-feedstock

Table 1. Common dimensions present in the default dimension universe.

Name	Description
`instrument`	Instrument.
`band`	Waveband of interest.
`physical_filter`	Filter used for the exposure.
`day_obs`	The observing day.
`group`	Group identifier.
`exposure`	Individual exposure.
`visit`	Collection of 1 or 2 exposures.
`tract`	Tesselation of the sky.
`patch`	Patch within a tract.

code should be written without knowing where files came from, what format they were written in, where the outputs are going to be written or how they are going to be stored. All that the algorithmic code needs to know is the relevant data model and the Python type. To meet these requirements we developed a library called the Data Butler (see e.g., T. Jenness et al. 2022; N. B. Lust et al. 2023).

The Butler internally is implemented as a registry, a database keeping track of datasets, and a datastore, a storage system that can map a Butler dataset to a specific collection of bytes. A datastore is usuall a file store (including POSIX file system, S3 object stores, or WebDAV) but could also be implemented as a NoSQL database or a metrics database such as Sasquatch (A. Fausti 2023).

A core concept of the Butler is that every dataset must be given what we call a "data coordinate." The data coordinate locates the dataset in the dimensional space where dimensions are defined in terms that scientists understand. Some commonly used dimensions are listed in Table 1. Each dataset is uniquely located by specifying its dataset type, its run collection, and its coordinates, with Butler refusing to accept another dataset that matches all three of those values. The dataset type defines the relevant dimensions and the associated Python storage class. The run collection can be thought of as a folder but does not have to be a folder within datastore.

As a concrete example, the file from one detector of an LSSTCam observation taken sometime in 2025 could have a data coordinate of `instrument="LSSTCam", detector=42, exposure=2025080300100` and be associated with a `raw` dataset type. The `exposure` record itself implies other information such as the physical filter and the time of observation. A deep coadd on a patch of sky would not have `exposure` dimensions at all and would instead be something like `instrument="LSSTCam", tract=105, patch=2, skymap="something"`, which would tell you exactly where it is located in the sky since you can calculate it from the tract and patch and skymap.

3.2. *Instrument Abstractions: Obs Packages*

The Butler and pipeline construction code know nothing about the specifics of a particular instrument. In the default dimension universe there is an `instrument` dimension that includes a field containing the full name of a Python `Instrument` class. This class, which uses a standard interface, is used by the system to isolate the instrument-specific from the pipeline-generic. Some of the responsibilities are:

- Register instrument-specific dimensions such as `detector`, `physical_filter` and the default `visit_system`.
- Define the default `raw` dataset type and the associated dimensions.
- Provide configuration defaults for pipeline task code that is processing data from this instrument.
- Provide a "formatter" class that knows how to read raw data.
- Define the default curated calibrations known to this instrument.

By convention we define the instrument class and associated configuration in `obs` packages. As an extension to the base definition of an "instrument", the LSST Science Pipelines define a modified `Instrument` class that includes focal plane distortions using the `afw` package (see §4.3). There are currently project-supported `obs` packages for:

- LSSTCam (A. Roodman et al. 2024; T. Lange et al. 2024; Y. Utsumi et al. 2024; S. M. Kahn et al. 2010), LATISS (P. Ingraham et al. 2020), and associated Rubin Observatory test stands and simulators.
- Hyper-SuprimeCam (S. Miyazaki et al. 2018).
- The Dark Energy Camera (B. Flaugher et al. 2015; D. L. DePoy et al. 2008).
- CFHT's MegaPrime (O. Boulade et al. 2003).

Additionally, teams outside the project have developed `obs` packages to support Subaru's Prime Focus Spectrograph (S.-Y. Wang et al. 2020), VISTA's VIRCAM (W. Sutherland et al. 2015), the Wide Field Survey Telescope (WFST; M. Cai et al. 2025), and the Gravitational-wave Optical Transient Observer (GOTO; J. R. Mullaney et al. 2021).

3.3. *Metadata Translation*

Every instrument uses different metadata standards but the Butler data model and pipelines require some form of standardization to determine values such as the coordinates of an observation, the observation type, or the time of observation. To perform that standard extraction of metadata each supported instrument must provide a metadata translator class using the `astro_metadata_translator` infrastructure.[9] The translator classes can understand evolving data models and allow the standardized metadata to be extracted for the lifetime of an instrument even if headers changed. Furthermore, in addition to providing standardized metadata the package can also provide programmatic or per-exposure corrections to data headers prior to calculating the translated metadata. This allows files that were written with incorrect headers to be recovered.

4. CORE INFRASTRUCTURE LIBRARIES

4.1. *Region Handling*

`geom` and `sphgeom`?

Use ICRS coordinates everywhere. All coordinate transformations are done within Astropy.

4.2. *Time and Hierarchical Data Structures*

`daf_base`.

Use Datetime only to store times in C++ objects. Use `astropy.time` for all other time handling, following the recommendations from T. Jenness et al. (2016).

`PropertySet` and `PropertyList` to allow `dict`-like data structures to be passed from Python to C++ and back again.

4.3. *Application Framework*

`afw` – this is called the "Application Framework" in T. Axelrod et al. (2010)[10]

- Image/MaskedImage/Exposure
- Table and Catalogs.
- Detection
- Math
- Camera geometry
- FITS I/O
- WCS: AST library (D. S. Berry et al. 2016) backs the world coordinate system handling.

`coadd_utils` ?

5. INSTRUMENT SIGNATURE REMOVAL

Raw images from charge-coupled devices (CCDs) contain instrumental effects, such as dark currents, clocking artifacts, or crosstalk between neighboring amplifiers, that can be removed in the data processing. In the Rubin pipeline, this step is called Instrument Signature Removal (ISR) and is the first processing applied to a raw CCD exposure. The package performing the ISR on an exposure, called `ip_isr`, is detailed below in Sec. 5.1: it is a critical package for Data Release Pipeline (DRP) used to process LSST images and requires calibration products produced and verified by `cp_pipe` and `cp_verify` respectively as described in Sec. 5.3. For further information about the life cycle of a calibration product and the procedures it entails, see C. Waters (2025). In Sec. 5.2, we specifically describe the correction of amplifier offset in more detail. A general overview of the ISR steps (based on the model in Fig. 2) and calibration products production (including generation, verification, certification, approval, and distribution) is given in A. A. Plazas Malagón et al. (2024).

We note that we focus here on our approach to performing ISR on data from LSST cameras only (LSSTCam, ComCam, and LATISS), although we also provide calibration pipelines for other cameras such as DECam and HSC (using a different ISR approach).

5.1. *ISR package*

Exposures from LSST cameras are affected by instrumental effects, ranging from well-known CCD effects like dark currents or bias levels to effects more recently characterized like tree-rings (see H. Y. Park et al. (2017); H. Park et al. (2020); J. H. Esteves et al. (2023); Y. Okura et al. (2015, 2016) for more details on tree rings in LSSTCam and their impact on science) or the Brighter-Fatter effect as discussed in A. Broughton et al. (2024). Correcting for these effects requires specific calibrations, which we refer to as calibration products. In LSST cameras, calibration products typically are a combined bias,

[9] https://astro-metadata-translator.lsst.io

[10] This document can be downloaded from https://ls.st/Document-9349

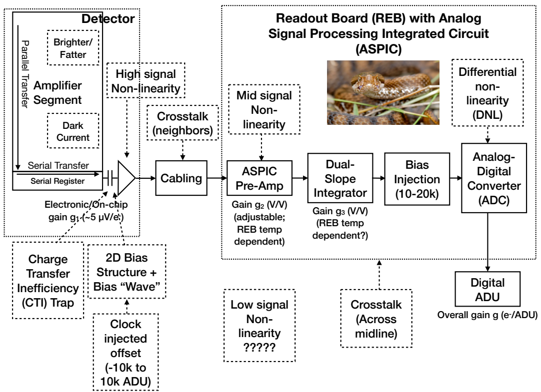

Figure 2. Schematic of the instrument model for detector effects in LSST cameras which `isrTaskLSST` is based on at the time of publication. More details about the model can be found in P. Fagrelius & E. Rykoff (2025) and A. A. Plazas Malagón et al. (2024).

a combined dark, a Photon Transfer Curve (PTC), a crosstalk matrix, a list of defects, and a look-up table of non-linearity parameters. The meaning of these calibration products and the details on the Rubin Observatory's ISR and calibration approach can be found in A. A. Plazas Malagón et al. (2024) and (P. Fagrelius & E. Rykoff 2025).

The `ip_isr` package[11] contains the codes needed to remove instrument signatures in exposures from LSST cameras and to produce calibration products. To inform our ISR approach, we first designed a model of the instrument, displayed in Fig. 2, based on our knowledge of the hardware and electronics. This model states the order in which the different known instrumental effects happen, from a photon hitting the CCD to the output ADC unit (ADU) signal. In turn, `isrTaskLSST` in `ip_isr` sequentially applies corrections of these effects in the opposite order as their effects occur in the model, as we are attempting to remove the impact of those effects on the image. Such corrections are typically done by calling other `Tasks` (*e.g.* overscan, crosstalk, etc.) also implemented in `ip_isr`.

Overall, `isrTaskLSST` takes a `raw` CCD exposure, and calibration products if available, and outputs a `Struct` containing the output exposure, the `postISRCCD` output exposure as well as its binned version for easier display, the exposure without interpolation and statistics on the output exposure. `IsrTaskLSSTConfig` defines the configurations used in this `Task`, they are set by default to their expected value to perform ISR on a typical LSSTCam exposure. Configuration parameters starting with `do` will typically correspond to an ISR step, they are turned on or off in the pipelines when producing the different calibration products. We have also developed `isrMockLSST` which simulates a raw exposure and corresponding calibration products and is used to test `isrTaskLSST`.

5.2. *Amplifier Offset Correction*

The amplifier offset correction (commonly referred to as amp-offset correction, or pattern continuity correction) runs as part of the instrument signature removal (ISR) process. This correction is designed to address systematic discontinuities in background sky levels across amplifier boundaries. We believe that these discontinuities arise from electronic biases between adjacent amplifiers, persisting even after the application of dark and flat corrections.

Drawing on the `PANSTARRS`' Pattern Continuity algorithm (C. Z. Waters et al. 2020), our method aims to eliminate these offsets, thereby preventing problems such as background over-/under-subtraction at amplifier boundaries caused by discontinuities across the detector.

The amp-offset algorithm initially computes a robust flux difference measure between two narrow strips on opposite sides of each amplifier-amplifier interface. Regions containing detected sources, or pixel data which have been masked for other reasons, are not considered. These amp-interface differences are stored in an amp-offset matrix; diagonal entries represent the number of neighboring amplifiers, and off-diagonal entries encode information about the associations between amplifiers. A complementary interface matrix encodes directional information for these associations. Using this information, a least-squares minimization is performed to determine the optimal pedestal value to be added or subtracted to each amp which would reduce the amp-offset between that amplifier and all of its neighboring amplifiers. This method is generalized to support 2D amplifier geometries within a detector, as with LSSTCam, incorporating length-based weighting into the matrices to account for amplifiers that are not square.

5.3. *Calibration pipelines*

The pipelines to build calibration products (`cp`) for the LSST cameras are defined in `cp_pipe`[12]. They set `isrTaskLSST` configuration parameters needed for each calibration product, by enabling all the sequential steps of the ISR task up to the step before the correction be-

[11] https://github.com/lsst/ip_isr

[12] https://github.com/lsst/cp_pipe and see documentation at https://pipelines.lsst.io/modules/lsst.cp.pipe/constructing-calibrations.html

ing generated. In some cases, configurations also specify whether to combine exposures (for bias or dark exposures for instance) and to bin exposures to support display.

Once calibration products are produced, they are "verified" (see C. Waters (2025) for more details) using `cp_verify`[13] pipelines by checking they pass metrics defined in R. Lupton et al. (2025). In this case, verify configuration parameters enable all corrections in the ISR task up to and including the application of the correction being verified. As a result, the calibration products can then be certified to be available in the `butler` and used to ISR an exposure.

6. MEASUREMENT SYSTEM

Measurement plugin system.
`meas_base` and `meas_algorithms`

6.1. *meas_deblender*

6.2. *meas_extensions_convolved*

6.3. *meas_extensions_gaap*

6.4. *meas_extensions_photometryKron*

6.5. *meas_extensions_piff*

6.6. *meas_extensions_psfex*

6.7. *meas_extensions_scarlet*

6.8. *meas_extensions_shapeHSM*

`meas_extensions_shapeHSM` package contains the plugins to measure the shapes of objects. The plugins measure the moments of the sources and PSFs with adaptive Gaussian weights. The algorithm was initially described in C. Hirata & U. Seljak (2003) and was modified later in R. Mandelbaum et al. (2005). The implementation of these algorithms lives within the `hsm` module of the GalSim package (B. T. P. Rowe et al. 2015). `meas_extensions_shapeHSM` now interacts directly with the Python layer of GalSim to make the measurements.

The base plugin for measuring moments is the `HsmMomentsPlugin` and is the parent class of the `HsmSourceMomentsPlugin` and `HsmPsfMomentsPlugin` which are specialized to measure on the sources (and objects) and PSFs respectively. `HsmSourceMomentsRoundPlugin` is a further specialized plugin that measures the moments with circular Gaussian weights instead of the elliptical ones in `HsmSourceMomentsPlugin`. The `HsmPsfMomentsDebiasedPlugin` adds noise to the PSF image to degrade it to have the same signal-to-noise ratio (SNR) as the source image. This makes the ellipticity calculated from this plugin have the same bias as the source ellipticity The PSF moments from this plugin should be used when calculating ellipticity residuals so the bias is largely cancelled. Having the various specializations as distinct plugins allows an object to be measured under different configurations simultaneously and included in the output catalogs.

In addition to the plugins that measure (adaptive) weighted moments, there are also a series of `HsmShape` plugins to estimate the PSF-corrected ellipticities of objects. In particular, the outputs from `HsmShapeRegaussPlugin` have been used to measure weak gravitation lensing signals in the Hyper Suprime-Cam SSP data (R. Mandelbaum et al. 2018; X. Li et al. 2022).

6.9. *meas_extensions_simpleShape*

6.10. *meas_extensions_trailedSources*

6.11. *meas_modelfit*

6.12. *meas_transiNet*

7. DIFFERENCE IMAGING

`ip_diffim`

8. ASTROMETRIC AND PHOTOMETRIC CALIBRATION

8.1. *Astrometric Calibration*

`meas_astrom` `gbdes` (G. M. Bernstein 2022; G. M. Bernstein et al. 2017)
Jointcal no longer discussed.

8.2. *Photometric Calibration*

8.3. *fgcmcal*

FGCM (D. L. Burke et al. 2018)

9. SOURCE ASSOCIATION

`ap_association`, for both DiaSource and Solar System processing

10. ALERT GENERATION AND DISTRIBUTION

`ap_association`, `alert_packet`

11. PIPELINES

11.1. *Pipeline Support*

Tasks and PipelineTask overview.

The `Task` Python class provides a standard interface for how to execute an algorithm. The `PipelineTask` variant provides stronger guarantees on configuration and provides a means by which the pipeline execution

[13] https://github.com/lsst/cp_verify

framework can determine how to link a task into a pipeline and how to determine what type of data should be read from a Butler and what should be written out to a Butler.

Maybe describe `pex_config` because it's not described anywhere.

11.2. *Task library*

`pipe_tasks` `drp_tasks`

11.3. *Pipeline Collections*

`drp_pipe`

The `ap_pipe` package defines the pipeline(s) to be used for real-time Alert Production processing (**??**). These pipelines include instrument signature removal (§5), calibration (§**??**), measurement plugins (§6), image differencing (§7), source association (§9), and alert generation (§10). Some of these tasks are shared with the pipelines in `drp_pipe`, but configured to prioritize speed over strict quality; for example, they use a minimal set of measurement plugins.

`ap_pipe` currently has pipeline variants for LATISS, the Rubin Observatory simulators, Hyper-SuprimeCam, and the Dark Energy Camera. Because these variants serve as testbeds for AP-specific algorithms and configuration settings, they are, as much as possible, the "same" pipeline, differing almost entirely in loading instrument defaults from `obs` packages (§3.2). The only other customization is an extra task for handling DECam's interchip crosstalk, which does not have an equivalent for Rubin instruments.

12. CATALOG SCHEMAS

Must transform pipeline products from the internal data model to the public data model defined in M. Jurić et al. (2023).

`sdm_schemas` `felis`

13. DISPLAY ABSTRACTIONS

Display plugins for:

matplotlib (J. D. Hunter 2007), firefly (W. Roby et al. 2020), ds9 (W. A. Joye & E. Mandel 2003)

14. DATA ANALYSIS

`analysis_tools` `verify` `faro`

15. VALIDATING THE SCIENCE PIPELINES

We use small, of order of a few gigabyte, datasets that can be processed as part of continuous integration. These take of order an hour to process. There are regular re-processings of standard datasets that can take a few days to process. For formal data releases there are additional metrics calculated and a formal test report is issued.

16. CONCLUSIONS

The LSST Science Pipelines Software has been developed over 20 years to support the processing of the Legacy Survey of Space and Time.

ACKNOWLEDGMENTS

This material is based upon work supported in part by the National Science Foundation through Cooperative Agreement AST-1258333 and Cooperative Support Agreement AST-1202910 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory managed by Stanford University. Additional Rubin Observatory funding comes from private donations, grants to universities, and in-kind support from LSSTC Institutional Members.

Facilities: Rubin:Simonyi (LSSTCam), Rubin:1.2m (LATISS)

Software: ndarray (https://github.com/ndarray/ndarray), astropy (Astropy Collaboration et al. 2022), pytest (H. Krekel 2017), matplotlib (J. D. Hunter 2007), galsim (B. T. P. Rowe et al. 2015), numpy (C. R. Harris et al. 2020), gbdes (G. M. Bernstein 2022), Starlink's (D. Berry et al. 2022) AST (D. S. Berry et al. 2016), fgcm (https://github.com/erykoff/fgcm),

REFERENCES

Astropy Collaboration, Price-Whelan, A. M., Sipőcz, B. M., et al. 2018, The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f

Astropy Collaboration, Price-Whelan, A. M., Lim, P. L., et al. 2022, The Astropy Project: Sustaining and Growing a Community-oriented Open-source Project and the Latest Major Release (v5.0) of the Core Package, ApJ, 935, 167, doi: 10.3847/1538-4357/ac7c74

Axelrod, T., Connolly, A., Ivezic, Z., et al. 2004, in American Astronomical Society Meeting Abstracts, Vol. 205, American Astronomical Society Meeting Abstracts, 108.11

Axelrod, T., Kantor, J., Lupton, R. H., & Pierfederici, F. 2010, in Proc. SPIE, Vol. 7740, Software and Cyberinfrastructure for Astronomy, ed. N. M. Radziwill & A. Bridger, 15, doi: 10.1117/12.857297

Bernstein, G. M. 2022, gbdes: DECam instrumental signature fitting and processing programs,, Astrophysics Source Code Library, record ascl:2210.011

Bernstein, G. M., Armstrong, R., Plazas, A. A., et al. 2017, Astrometric Calibration and Performance of the Dark Energy Camera, PASP, 129, 074503, doi: 10.1088/1538-3873/aa6c55

Berry, D., Graves, S., Bell, G. S., et al. 2022, in Astronomical Society of the Pacific Conference Series, Vol. 532, Astronomical Data Analysis Software and Systems XXX, ed. J. E. Ruiz, F. Pierfedereci, & P. Teuben, 559

Berry, D. S., Warren-Smith, R. F., & Jenness, T. 2016, AST: A library for modelling and manipulating coordinate systems, Astronomy and Computing, 15, 33, doi: 10.1016/j.ascom.2016.02.003

Bosch, J., Armstrong, R., Bickerton, S., et al. 2018, The Hyper Suprime-Cam software pipeline, PASJ, 70, S5, doi: 10.1093/pasj/psx080

Boulade, O., Charlot, X., Abbon, P., et al. 2003, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 4841, Instrument Design and Performance for Optical/Infrared Ground-based Telescopes, ed. M. Iye & A. F. M. Moorwood, 72–81, doi: 10.1117/12.459890

Broughton, A., Utsumi, Y., Plazas Malagón, A. A., et al. 2024, Mitigation of the Brighter-fatter Effect in the LSST Camera, PASP, 136, 045003, doi: 10.1088/1538-3873/ad3aa2

Burke, D. L., Rykoff, E. S., Allam, S., et al. 2018, Forward Global Photometric Calibration of the Dark Energy Survey, AJ, 155, 41, doi: 10.3847/1538-3881/aa9f22

Cai, M., Xu, Z., Fan, L., et al. 2025, The 2.5-meter Wide Field Survey Telescope Real-time Data Processing Pipeline I: From raw data to alert distribution, arXiv e-prints, arXiv:2501.15018, doi: 10.48550/arXiv.2501.15018

DePoy, D. L., Abbott, T., Annis, J., et al. 2008, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 7014, Ground-based and Airborne Instrumentation for Astronomy II, ed. I. S. McLean & M. M. Casali, 70140E, doi: 10.1117/12.789466

Esteves, J. H., Utsumi, Y., Snyder, A., et al. 2023, Photometry, Centroid and Point-spread Function Measurements in the LSST Camera Focal Plane Using Artificial Stars, PASP, 135, 115003, doi: 10.1088/1538-3873/ad0a73

Fagrelius, P., & Rykoff, E. 2025, Rubin Baseline Calibration Plan, Vera C. Rubin Observatory. https://sitcomtn-086.lsst.io/

Fausti, A. 2023, Sasquatch: beyond the EFD, Vera C. Rubin Observatory. https://sqr-068.lsst.io/

Flaugher, B., Diehl, H. T., Honscheid, K., et al. 2015, The Dark Energy Camera, The Astronomical Journal, 150, 150, doi: 10.1088/0004-6256/150/5/150

Gower, M., Kowalik, M., Lust, N. B., Bosch, J. F., & Jenness, T. 2022, Adding Workflow Management Flexibility to LSST Pipelines Execution, arXiv e-prints, arXiv:2211.15795, doi: 10.48550/arXiv.2211.15795

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. 2020, Array programming with NumPy, Nature, 585, 357, doi: 10.1038/s41586-020-2649-2

Hirata, C., & Seljak, U. 2003, Shear calibration biases in weak-lensing surveys, MNRAS, 343, 459, doi: 10.1046/j.1365-8711.2003.06683.x

Hunter, J. D. 2007, Matplotlib: A 2D Graphics Environment, Computing in Science and Engineering, 9, 90, doi: 10.1109/MCSE.2007.55

Ingraham, P., Clements, A. W., Ribeiro, T., et al. 2020, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11452, Software and Cyberinfrastructure for Astronomy VI, ed. J. C. Guzman & J. Ibsen, 114520U, doi: 10.1117/12.2561112

Ivezić, Ž., Kahn, S. M., Tyson, J. A., et al. 2019, LSST: From Science Drivers to Reference Design and Anticipated Data Products, ApJ, 873, 111, doi: 10.3847/1538-4357/ab042c

Jenness, T. 2020, in Astronomical Society of the Pacific Conference Series, Vol. 522, Astronomical Data Analysis Software and Systems XXVII, ed. P. Ballester, J. Ibsen, M. Solar, & K. Shortridge, 541, doi: 10.48550/arXiv.1712.00461

Jenness, T., Economou, F., Findeisen, K., et al. 2018, in Proc. SPIE, Vol. 10707, Software and Cyberinfrastructure for Astronomy V, 1070709, doi: 10.1117/12.2312157

Jenness, T., Bosch, J., Owen, R., et al. 2016, in Proc. SPIE, Vol. 9913, Software and Cyberinfrastructure for Astronomy IV, 99130G, doi: 10.1117/12.2231313

Jenness, T., Bosch, J. F., Salnikov, A., et al. 2022, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 12189, Software and Cyberinfrastructure for Astronomy VII, 1218911, doi: 10.1117/12.2629569

Joye, W. A., & Mandel, E. 2003, in Astronomical Society of the Pacific Conference Series, Vol. 295, Astronomical Data Analysis Software and Systems XII, ed. H. E. Payne, R. I. Jedrzejewski, & R. N. Hook, 489

Jurić, M., Ciardi, D., Dubois-Felsmann, G., & Guy, L. 2019, LSST Science Platform Vision Document, Vera C. Rubin Observatory. https://lse-319.lsst.io/

Jurić, M., Kantor, J., Lim, K. T., et al. 2017, in Astronomical Society of the Pacific Conference Series, Vol. 512, Astronomical Data Analysis Software and Systems XXV, ed. N. P. F. Lorente, K. Shortridge, & R. Wayth, 279, doi: 10.48550/arXiv.1512.07914

Jurić, M., Axelrod, T., Becker, A., et al. 2023, Data Products Definition Document, Vera C. Rubin Observatory. https://lse-163.lsst.io/

Kahn, S. M., Kurita, N., Gilmore, K., et al. 2010, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 7735, Ground-based and Airborne Instrumentation for Astronomy III, ed. I. S. McLean, S. K. Ramsay, & H. Takami, 0, doi: 10.1117/12.857920

Knight, S. 2005, Building software with SCons, Computing in Science Engineering, 7, 79, doi: 10.1109/MCSE.2005.11

Krekel, H. 2017, pytest: helps you write better programs, https://docs.pytest.org

Lange, T., Nordby, M., Pollek, H., et al. 2024, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 13096, Ground-based and Airborne Instrumentation for Astronomy X, ed. J. J. Bryant, K. Motohara, & J. R. D. Vernet, 130961O, doi: 10.1117/12.3019302

Li, X., Miyatake, H., Luo, W., et al. 2022, The three-year shear catalog of the Subaru Hyper Suprime-Cam SSP Survey, PASJ, 74, 421, doi: 10.1093/pasj/psac006

Lupton, R., Malagón, A. A. P., & Waters, C. 2025, Verifying LSST Calibration Data Products, Vera C. Rubin Observatory. https://dmtn-101.lsst.io/

Lust, N. B., Jenness, T., Bosch, J. F., et al. 2023, Data management and execution systems for the Rubin Observatory Science Pipelines, arXiv e-prints, arXiv:2303.03313, doi: 10.48550/arXiv.2303.03313

Mandelbaum, R., Hirata, C. M., Seljak, U., et al. 2005, Systematic errors in weak lensing: application to SDSS galaxy-galaxy weak lensing, MNRAS, 361, 1287, doi: 10.1111/j.1365-2966.2005.09282.x

Mandelbaum, R., Miyatake, H., Hamana, T., et al. 2018, The first-year shear catalog of the Subaru Hyper Suprime-Cam Subaru Strategic Program Survey, PASJ, 70, S25, doi: 10.1093/pasj/psx130

Miyazaki, S., Komiyama, Y., Kawanomoto, S., et al. 2018, Hyper Suprime-Cam: System design and verification of image quality, PASJ, 70, S1, doi: 10.1093/pasj/psx063

Mullaney, J. R., Makrygianni, L., Dhillon, V., et al. 2021, Processing GOTO data with the Rubin Observatory LSST Science Pipelines I: Production of coadded frames, PASA, 38, e004, doi: 10.1017/pasa.2020.45

Okura, Y., Petri, A., May, M., Plazas, A. A., & Tamagawa, T. 2016, Consequences of CCD Imperfections for Cosmology Determined by Weak Lensing Surveys: From Laboratory Measurements to Cosmological Parameter Bias, The Astrophysical Journal, 825, 61, doi: 10.3847/0004-637X/825/1/61

Okura, Y., Plazas, A. A., May, M., & Tamagawa, T. 2015, Spurious shear induced by the tree rings of the LSST CCDs, Journal of Instrumentation, 10, C08010, doi: 10.1088/1748-0221/10/08/C08010

O'Mullane, W., Economou, F., Lim, K.-T., et al. 2022, Software Architecture and System Design of Rubin Observatory, arXiv e-prints, arXiv:2211.13611, doi: 10.48550/arXiv.2211.13611

Padmanabhan, N., Lupton, R., & Loomis, C. 2015, EUPS — a Tool to Manage Software Dependencies, https://github.com/RobertLuptonTheGood/eups

Park, H., Karpov, S., Nomerotski, A., & Tsybychev, D. 2020, Tree rings in Large Synoptic Survey Telescope production sensors: its dependence on radius, wavelength, and back bias voltage, Journal of Astronomical Telescopes, Instruments, and Systems, 6, 011005, doi: 10.1117/1.JATIS.6.1.011005

Park, H. Y., Nomerotski, A., & Tsybychev, D. 2017, Properties of tree rings in LSST sensors, Journal of Instrumentation, 12, C05015, doi: 10.1088/1748-0221/12/05/C05015

Plazas Malagón, A. A., Waters, C., Broughton, A., et al. 2024, Instrument Signature Removal and Calibration Products for the Rubin Legacy Survey of Space and Time, arXiv e-prints, arXiv:2404.14516, doi: 10.48550/arXiv.2404.14516

Roby, W., Wu, X., Dubois–Felmann, G., et al. 2020, in Astronomical Society of the Pacific Conference Series, Vol. 527, Astronomical Data Analysis Software and Systems XXIX, ed. R. Pizzo, E. R. Deul, J. D. Mol, J. de Plaa, & H. Verkouter, 243

Roodman, A., Rasmussen, A., Bradshaw, A., et al. 2024, in Ground-based and Airborne Instrumentation for Astronomy X, ed. J. J. Bryant, K. Motohara, & J. R. D. Vernet, Vol. 13096, International Society for Optics and Photonics (SPIE), 130961S, doi: 10.1117/12.3019698

Rowe, B. T. P., Jarvis, M., Mandelbaum, R., et al. 2015, GALSIM: The modular galaxy image simulation toolkit, Astronomy and Computing, 10, 121, doi: 10.1016/j.ascom.2015.02.002

Sutherland, W., Emerson, J., Dalton, G., et al. 2015, The Visible and Infrared Survey Telescope for Astronomy (VISTA): Design, technical overview, and performance, A&A, 575, A25, doi: 10.1051/0004-6361/201424973

Utsumi, Y., Antilogus, P., Astier, P., et al. 2024, in X-Ray, Optical, and Infrared Detectors for Astronomy XI, ed. A. D. Holland & K. Minoglou, Vol. 13103, International Society for Optics and Photonics (SPIE), 131030W, doi: 10.1117/12.3019117

van Rossum, G. 2013, PEP 8 – Style Guide for Python Code, Python Software Foundation. https://www.python.org/dev/peps/pep-0008/

Wang, D. L., Monkewitz, S. M., Lim, K.-T., & Becla, J. 2011, in State of the Practice Reports, SC '11 (New York, NY, USA: ACM), 12:1–12:11, doi: 10.1145/2063348.2063364

Wang, S.-Y., Huang, P.-J., Chen, H.-Y., et al. 2020, in Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 11447, Ground-based and Airborne Instrumentation for Astronomy VIII, ed. C. J. Evans, J. J. Bryant, & K. Motohara, 114477V, doi: 10.1117/12.2561194

Waters, C. 2025, Calibration Generation, Verification, Acceptance, and Certification., Vera C. Rubin Observatory. https://dmtn-222.lsst.io/

Waters, C. Z., Magnier, E. A., Price, P. A., et al. 2020, Pan-STARRS Pixel Processing: Detrending, Warping, Stacking, ApJS, 251, 4, doi: 10.3847/1538-4365/abb82b