Large Synoptic Survey Telescope
Data Management Applications Design

Mario Jurić[*], R.H. Lupton, T. Axelrod, J.F. Bosch, P.A. Price,
G.P. Dubois-Felsmann, Ž. Ivezić, A.C. Becker, J. Becla,
A.J. Connolly, J. Kantor, K-T Lim, D. Shaw,
for the LSST Data Management

Thursday 16th June, 2016, 18:49hrs

---

[*] Please direct comments to <mjuric@lsst.org>.

Abstract

The LSST Science Requirements Document (the LSST SRD) specifies a set of data product guidelines, designed to support science goals envisioned to be enabled by the LSST observing program. Following these guidlines, the details of these data products have been described in the LSST Data Products Definition Document (DPDD), and captured in a formal flow-down from the SRD via the LSST System Requirements (LSR), Observatory System Specifications (OSS), to the Data Management System Requirements (DMSR). The LSST Data Management subsystem's responsibilities include the design, implementation, deployment and execution of software pipelines necessary to generate these data products. This document, in conjunction with the UML Use Case model (LDM-134), describes the design of the scientific aspects of those pipelines.

Contents

1 Preface

The purpose of this document is to describe the design of pipelines belonging to the Applications Layer of the Large Synoptic Survey Telescope (LSST) Data Management system. These include most of the core astronomical data processing software that LSST employs.

The intended audience of this document are LSST software architects and developers. It presents the baseline architecture and algorithmic selections for core DM pipelines. The document assumes the reader/developer has the required knowledge of astronomical image processing algorithms and solid understanding of the state of the art of the field, understanding of the LSST Project goals and concepts, and has read the LSST Science Requirements (SRD) as well as the LSST Data Products Definition Document (DPDD).

This document should be read in conjunction with the LSST DM Applications Use Case Model (LDM-134). They are intended to be complementary, with the Use Case model capturing the detailed (inter)connections between individual pipeline components, and this document capturing the overall goals, pipeline architecture, and algorithmic choices.

Though under strict change control[1], this is a ***living document***. Firstly, as a consequence of the "rolling wave" LSST software development model, the designs presented in this document will be refined and made more detailed as particular pipeline functionality is about to be implemented. Secondly, the LSST will undergo a period of construction and commissioning lasting no less than seven years, followed by a decade of survey operations. To ensure their continued scientific adequacy, the overall designs and plans for LSST data processing pipelines will be periodically reviewed and updated.

---

[1] LSST Docushare handle for this document is `LDM-151`.

2 Introduction

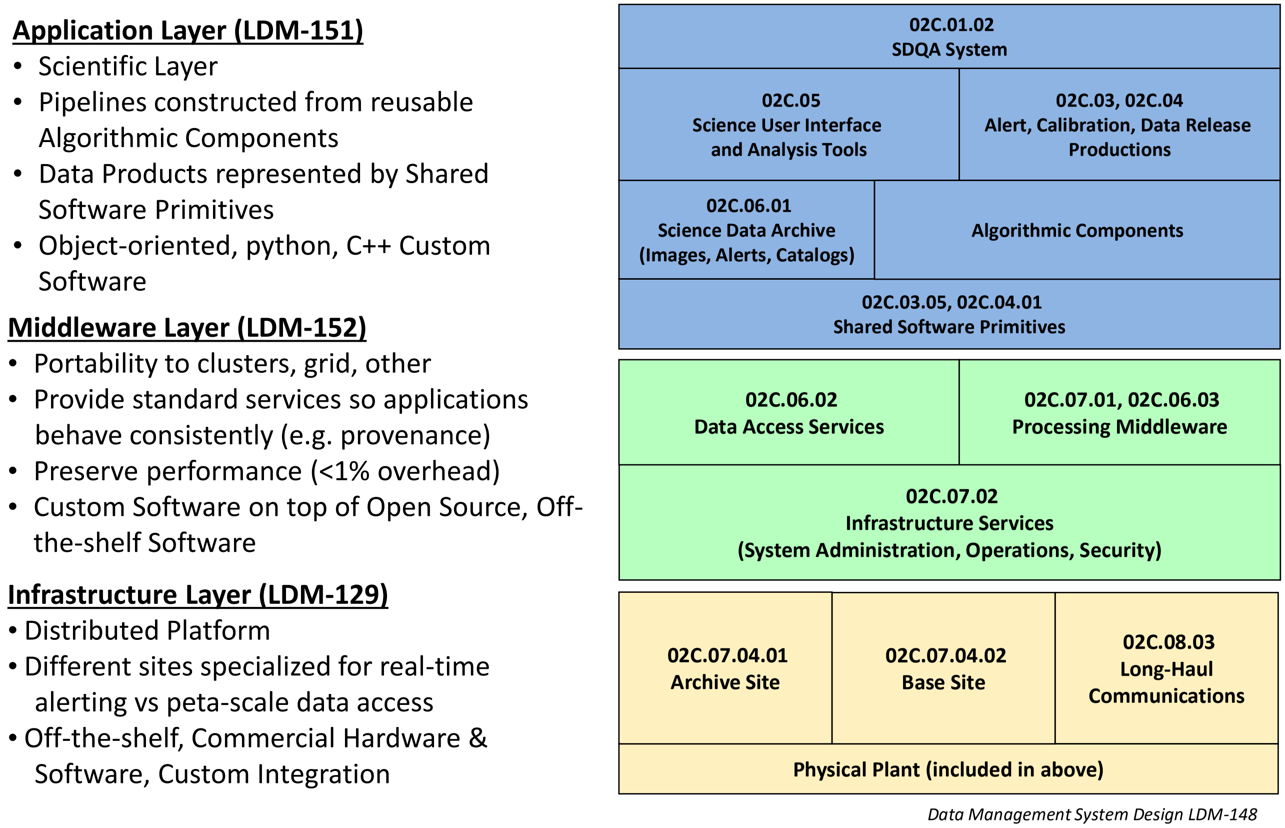
2.1 LSST Data Management System

To carry out this mission the Data Management System (DMS) performs the following major functions:

- Processes the incoming stream of images generated by the camera system during observing to produce transient alerts and to archive the raw images.
- Roughly once per year, creates and archives a Data Release ("DR"), which is a static self-consistent collection of data products generated from all survey data taken from the date of survey initiation to the cutoff date for the Data Release. The data products (described in detail in the DPDD), include measurements of the properties (shapes, positions, fluxes, motions, etc.) of all detected objects, including those below the single visit sensitivity limit, astrometric and photometric calibration of the full survey object catalog, and limited classification of objects based on both their static properties and time-dependent behavior. Deep coadded images of the full survey area are produced as well.
- Periodically creates new calibration data products, such as bias frames and flat fields, that will be used by the other processing functions, as necessary to enable the creation of the data products above.
- Makes all LSST data available through interfaces that utilize, to the maximum possible extent, community-based standards such as those being developed by the Virtual Observatory ("VO"), and facilitates user data analysis and the production of user-defined data products at Data Access Centers ("DAC") and at external sites.

The overall architecture of the DMS is discussed in more detail in the Data Management System Design (DMSD) document. The overall architecture of the DMS is shown in Figure 1.

This document discusses the role of the Applications layer in the first three functions listed above (the functions involving *science pipelines*). The fourth is discussed separately in the SUI Conceptual Design Document (SUID).



Application Layer (LDM-151)

- Scientific Layer
- Pipelines constructed from reusable Algorithmic Components
- Data Products represented by Shared Software Primitives
- Object-oriented, python, C++ Custom Software

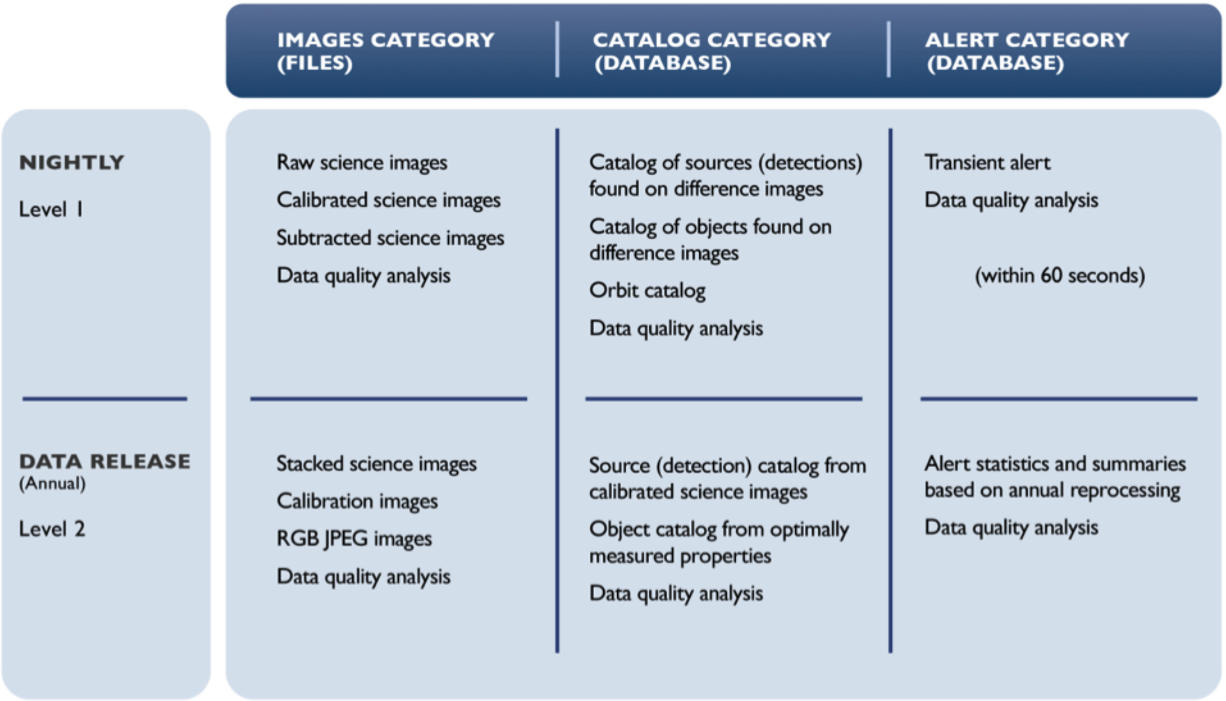
Middleware Layer (LDM-152)

- Portability to clusters, grid, other
- Provide standard services so applications behave consistently (e.g. provenance)
- Preserve performance (<1% overhead)
- Custom Software on top of Open Source, Off-the-shelf Software

Infrastructure Layer (LDM-129)

- Distributed Platform
- Different sites specialized for real-time alerting vs peta-scale data access
- Off-the-shelf, Commercial Hardware & Software, Custom Integration

Figure 1: Architecture of the Data Management System



	IMAGES CATEGORY (FILES)	CATALOG CATEGORY (DATABASE)	ALERT CATEGORY (DATABASE)
NIGHTLY Level 1	Raw science images Calibrated science images Subtracted science images Data quality analysis	Catalog of sources (detections) found on difference images Catalog of objects found on difference images Orbit catalog Data quality analysis	Transient alert Data quality analysis (within 60 seconds)
DATA RELEASE (Annual) Level 2	Stacked science images Calibration images RGB JPEG images Data quality analysis	Source (detection) catalog from calibrated science images Object catalog from optimally measured properties Data quality analysis	Alert statistics and summaries based on annual reprocessing Data quality analysis

Figure 2: Organization of LSST Data Products

2.2 Data Products

The LSST data products are organized into three groups, based on their intended use and/or origin. The full description is provided in the Data Products Definition Document (DPDD); we summarize the key properties here to provide the necessary context for the discussion to follow.

- **Level 1** products are intended to support timely detection and follow-up of time-domain events (variable and transient sources). They are generated by near-real-time processing the stream of data from the camera system during normal observing. Level 1 products are therefore continuously generated and / or updated every observing night. This process is of necessity highly automated, and must proceed with absolutely minimal human interaction. In addition to science data products, a number of related Level 1 "SDQA"[2] data products are generated to assess quality and to provide feedback to the Observatory Control System (OCS).

- **Level 2** products are generated as part of a Data Release, generally performed yearly, with an additional data release for the first 6 months of survey data. Level 2 includes data products for which extensive computation is required, often because they combine information from many exposures. Although the steps that generate Level 2 products will be automated, significant human interaction may be required at key points to ensure the quality of the data.

- **Level 3** products are generated on any computing resources anywhere and then stored in an LSST Data Access Center. Often, but not necessarily, they will be generated by users of LSST using LSST software and/or hardware. LSST DM is required to facilitate the creation of Level 3 data products by providing suitable APIs, software components, and computing infrastructure, but will not by itself create any Level 3 data products. Once created, Level 3 data products may be associated with Level 1 and Level 2 data products through database federation. Where appropriate, the LSST Project, with the agreement of the Level 3 creators, may incorporate user-contributed Level 3 data product pipelines into the DMS production flow, thereby promoting them to Level 1 or 2.

---

[2] Science Data Quality Analysis

The organization of LSST Data Products is shown in Figure 2.

Level 1 and Level 2 data products that have passed quality control tests will be accessible to the public without restriction. Additionally, the source code used to generate them will be made available, and LSST will provide support for builds on selected platforms.

The pipelines used to produce these public data products will also produce many intermediate data products that may not be made publically available (generally because they are fully superseded in quality by a public data product). Intermediate products may be important for QA, however, and their specification is an important part of describing the pipelines themselves.

2.3 Data Units

In order to describe the components of our processing pipelines, we first need standard nomenclature for the units of data the pipeline will process.

The smallest data units are those corresponding to individual astrophysical entities. In keeping with LSST conventions, we use “object” to refer to the astrophysical entity itself (which typically implies aggregation of some sort over all exposures), and “source” to refer to the realization of an object on a particular exposure. In the case of blending, of course, these are just our best attempts to define distinct astrophysical objects, and hence it is also useful to define terms that represent this process. We use “family” to refer to group of blended objects (or, more rarely, sources), and “child” to refer to a particular deblended object within a family. A “parent” is also created for each family, representing the alternate hypothesis that the blend is actually a single object. Blends may be hierarchical; a child at one level may be a parent at the level below.

LSST observations are taken as a pair of 15-second “snaps”; together these constitute a “visit”. Because snaps are typically combined early in the processing (and some special programs and survey modes may take only a single snap), visit is much more frequently used as a unit for processing and data products. The image data for to a visit is a set of 189 “CCD” or “sensor” images. CCD-level data from the camera is further data divided across the 16 amplifiers within a CCD, but these are also combined at an early stage, and the 3×3 CCD “rafts” that play an important role in the hardware design are relativley unimportant for the pipeline. This leaves visit and CCD the main identifiers of most exposure-level data products and pipelines.

Our convention for defining regions on the sky is deliberately vague; we hope to build a codebase capable of working with virtually any pixelization or projection scheme (though different schemes may have different performance or storage implications). Our approach involves two region concepts: “tracts” and “patches”. A tract is a large region with a single Cartesian coordinate system; we assume it is larger than the LSST field of view, but its maximum size is essentially set by the point at which distortion in the projection becomes significant enough to affect the processing (by e.g. breaking the assumption that the PSF is well-sampled on the pixel grid). Tracts are divided into patches, all of which share the tract coordinate system. Most image processing is perfomed at the patch level, and hence patch sizes are chosen largely to ensure that patch-level data products and processing fit in memory. Both tracts and patches are defined such that each region overlaps with its neighbors, and these overlap regions must be large enough that any individual astronomical object is wholly contained in at least one tract and patch. In a patch overlap region, we expect pixel values to be numerically equivalent (i.e. equal up to floating point round-off errors) on both sides; in tract overlaps, this is impossible, but we expect the results to be scientifically consistent. Selecting larger tracts and patches thus reduces the overall fraction of the area that falls in overlap regions and must be processed multiple times, while increasing the computational load for processing individual tracts and patches.

2.4 Science Pipelines Organization

As shown in Figure 1, the Applications Layer is itself split into three levels. In sections 3, 4, and 5, we describe the Alert Production, Calibration Products Production, and Da ta Release Production (respectively), breaking them down into *pipelines*. In this document, a pipeline is a high-level combination of algorithms that is intrinsically tied to its role in the production in which it is run. For instance, while both Alert Production and Data Release Production will include a pipeline for single-visit processing, these two pipelines are *distinct*, because the details of their design depend very much on the context in which they are run. Section 6 describes the Science Data Quality Analysis System, a collection of pipelines and mini-productions designed to assess and continuously validate the quality of both the data and the processing system. The SDQA System is not a single production; its components are either directly integrated into other productions or part of a set of multiple mini-productions run on different cadences.

Pipelines are largely composed of Algorithmic Components: mid-level algorithmic code that we expect to reuse (possibly with different configuration) across different productions. These components constitute the bulk of the new code and algorithms to be developed for Alert Production and Data Release Production, and are discussed in section 8. Most algorithmic components are applicable to any sort of astronomical imaging data, but some will be customized for LSST.

The lowest level in the Applications Layer is made up of our shared software primitives: libraries that provide important data structures and low-level algorithms, such as images, tables, coordinate transformations, and nonlinear optimizers. Much (but not all) of this content is astronomy-related, but essentially none of it is specific to LSST, and hence we can and will make use of third-party libraries whenever possible. These primitives also play an important role in connecting the Science User Interface Toolkit and Level 3 processing environment with Level 1 and Level 2 data products, as they constitute the programmatic representation of those data products. Shared software primitives are discussed in section 9.

3 Level 1 Pipelines

3.1 Single Frame Processing Pipeline (WBS 02C.03.01)

3.1.1 Key Requirements

Single Frame Processing (SFM) Pipeline is responsible for reducing raw image data to *calibrated exposures*, and detection and measurement of `Sources` (using the components functionally a part of the Object Characterization Pipeline).

SFM pipeline functions include:

- Assembly of per-amplifier images to an image of the entire CCD;
- Instrumental Signature Removal;
- Cosmic ray rejection and snap combining;
- Per-CCD determination of zeropoint and aperture corrections;
- Per-CCD PSF determination;
- Per-CCD WCS determination and astrometric registration of images;
- Per-CCD sky background determination;
- Source detection and measurement

Calibrated exposure produced by the SFM pipeline must possess all information necessary for measurement of source properties by single-epoch Object Characterization algorithms.

It shall be possible to run this pipeline in two modes: a "fast" mode needed in nightly operations for Level 1 data reductions where no source characterization is done beyond what's required for zero-point, PSF, sky, and WCS determination (image reduction); and a "full" mode that will be run for Level 2 data reductions.

3.1.2 Baseline Design

Single Frame Processing pipeline will be implemented as a flexible framework where different data can be easily treated differently, and new processing steps can be added without modifying the stack code.

It will consist of three primary components:

- A library of useful methods that wrap a small number of atomic operations (e.g., `interpolateFromMask`, `overscanCorrection`, `biasCorrection`, etc.)
- A set of classes (`Tasks`) that perform higher level jobs (e.g., `AssembleCcdTask`, or `FringeTask`), and a top level class to apply corrections to the input data in the proper order. This top level class can be overridden in the instrument specific `obs_*` packages, making the core SFM pipeline camera agnostic.
- A top-level Task to run the SFM pipeline.

In the paragraphs to follow, we describe the adopted baseline for key SFM algorithms. If not discussed explicitly, the algorithmic baseline for all other functionallity is assumed to be the same as that used by SDSS *Photo* pipeline [18].

Output information for OCS telemetry: ACTION clarify OCS interactions

3.1.2.1 Instrumental Signature Removal: Clarify interaction with butler

Input Data

- Camera corrected (crosstalk, overscan, linearity) images
- Sensor defect lists
- Metadata including electronic parameters (saturation limits, readnoise, electronic footprint)

Output Data

- Calexp images

Ancillary Products?

- Source detection and measurements
- ICExp background subtracted images
- Post ISR exposure

Actions in case of failure?
Actions in case camera data are not available due to network outage longer than buffer of data at summit **Alternative procedures?**

Subtasks:

- Mask defects and saturation
- Assembly
- Full frame corrections: Dark, Flats (includes fringing)
- Pixel level corrections: Brighter fatter, static pixel size effects
- **QUESTION is this run prior to pixel level corrections** Interpolation of defects and saturation
- CR rejection
- Generate snap difference
- Snap combination

3.1.2.2 PSF determination and background determination:

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:
Iterate till convergence (convergence criteria TBD)

- Background estimation
- Source detection
- Selection of PSF candidate stars
- PSF determination

3.1.2.3 Source measurement:
Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Source measurement - Single Frame Measurement
- Aperture correction

3.1.2.4 Photometric and Astrometric calibration:
Input Data?
DRP's internal reference catalog **Output Data?**
OCS PSF, WCS, metadata (TBD) **Anscillary Products?**
Actions in case of failure?
Alternative procedures?

Subtasks:

- Source association
- CCD level photometric solution
- Visit level photometric solution
- Remove known astrometric distortions
- Fit remaining residual
- Single visit composed astrometric solution

- Output information for OCS telemetry: WCS ACTION clarify OCS interactions

OUTPUT: Calibrated Exposure and Calibrated Catalog

3.1.3 Prototype Implementation

The prototype codes are available in the following repositories: `https://github.com/lsst/ip_isr`, `https://github.com/lsst/meas_algorithms`, `https://github.com/lsst/meas_astrom`, `https://github.com/lsst-dm/legacy-meas_mosaic`, `https://github.com/lsst/pipe_tasks`.

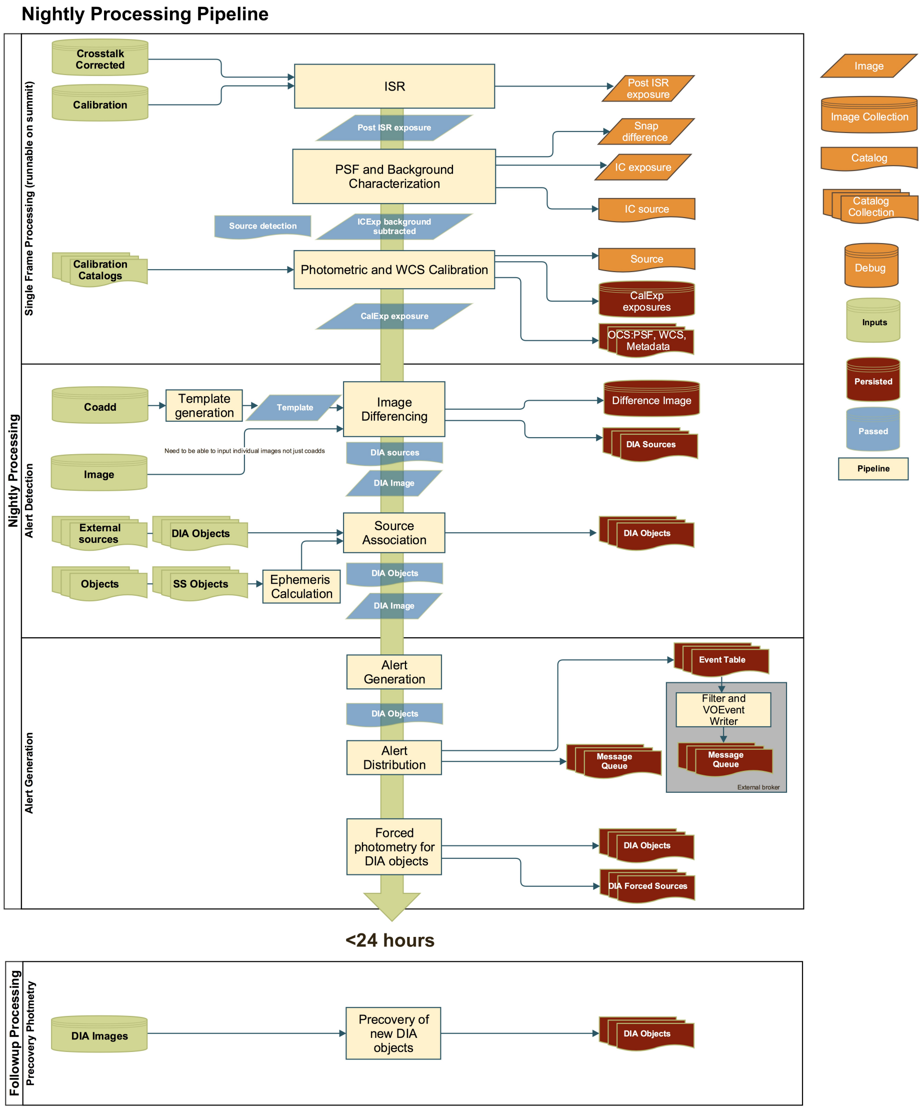


Figure 3: The nightly processing flowchart describing the flow of images and data through single frame processing, image differencing, alert generation and production

3.2 Alert Detection (WBS 02C.03.04)

3.2.1 Key Requirements

The alert detection pipeline shall difference a visit image against a deeper template, and detect and characterize sources in the difference image in the time required to achieve the 60 second design goal for Level 1 alert processing (current timing allocation: 24 seconds). The algorithms employed by the pipeline shall result in purity and completeness of the sample as required by the DMSR . Image differencing shall perform as well in crowded as in uncrowded fields.

3.2.2 Baseline Design

3.2.2.1 Template Generation

Input Data?
Coadded CalExps or Series of CalExp from which to interpolate a template.
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Determine appropriate template to use
- Generate template for observation

3.2.2.2 Image differencing

Input Data?
Internal reference catalog for CalExp from DRP PSF for science image **Output Data?**
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- match DRP sources and sources from SFP

- Determine relative astrometric solution
- Warp template and measurements to science image frame
- Correlate science image with science PSF (pre-convolution)
- Determine appropriate PSF matching sources
- Compute PSF matching kernel and spatial model using ZOGY approach
- Difference science and template images
- Apply correction for correlated noise
- Difference image source detection
- Difference image source measurement: dipole fit, trailed source measurement
- Measure flux on snap difference for all DIASources

3.2.2.3 Real-Bogus classification

Input Data?
Internal reference catalog for CalExp from DRP PSF for science image **Output Data?**
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Application of random forest or other classification algorithm
- Update DIASources with probabilistic classification
- Filter DIASource list based on classifier

3.2.2.4 Ephemeris Calculation

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Calculate positions for all solar system objects that may overlap the current exposure.

3.2.2.5 Source Association

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Match all DIASources to predicted Solar System object positions and DIAObject catalog positions
- Perform forced photometry of un-associated DIAObjects.(Maybe not if we force photometer all DIAObjects?). SSObjects will not be force photometered because the precision of the prediction will not be good enough. Force photometry for external DIAObjects?
- Update associated DIAObjects with aggregate quantities: e.g. parallax, proper motion, and variability metrics
- New spuriousness calculation?

3.2.3 Prototype Implementation

The prototype code is available at `https://github.com/lsst/ip_diffim`. The current prototype, while functional, will require a partial redesign to be transfered to construction to address performance and extensibility concerns.

3.3 Alert Generation Pipeline (WBS 02C.03.03)

3.3.1 Key Requirements

Alert Generation Pipeline shall take the newly discovered `DIASources` and all associated metadata as described in the DPDD, and generate alert packets in `VOEvent` format. It will transmit these packets to VO Event Brokers, using standard IVOA protocols (eg., VOEvent Transport Protocol; VTP). End-users will primarily use these brokers to classify and filter events for subsets fitting their science goals.

To directly serve the end-users, the Alert Generation Pipeline shall provide a basic, limited capacity, alert filtering service. This service will run at the LSST U.S. Archive Center (at NCSA). It will let astronomers create simple filters that limit what alerts are ultimately forwarded to them. These *user defined filters* will be possible to specify using an SQL-like declarative language, or short snippets of (likely Python) code.

3.3.2 Baseline Design

3.3.2.1 Alert generation

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Generate postage stamps for all DIASources: direct image and difference image
- Push alert records to alert database

3.3.2.2 Alert Distribution

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Filter event records (for content as well as for events)
- Author VOEvent
- Push to messaging queue

3.3.2.3 Forced Photometry on all DIAObjects

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Compute forced photometry on all DIAObjects in the field. This does not end up in the alerts.

3.3.3 Prototype Implementation

3.4 Precovery Photometry Pipeline

3.4.1 Key Requirements

Within 24 hrs.

3.4.1.1 Precovery of new DIAObjects

Input Data?
Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Force photometer in difference images for all new DIAObjects for the past 30 days.

3.5 Moving Object Pipeline (WBS 02C.03.06)

3.5.1 Key Requirements

The Moving Object Pipeline System (MOPS) has two responsibilities within LSST Data Management:

- First, it is responsible for generating and managing the Solar System[3] data products. These are Solar System objects with associated Keplerian orbits, errors, and detected `DIASources`. Quantitatively, it shall be capable of detecting 95% of all Solar System objects that meet the findability criteria as defined in the OSS . The software components implementing this function are known as ***DayMOPS***.
- The second responsibility of the MOPS is to predict future locations of moving objects in incoming images so that their sources may be associated with known objects; this will reduce the number of spurious transient detections and appropriately flag alerts to detections of known Solar System objects. The software components implementing this function are known as ***NightMOPS***.

3.5.2 Baseline Design

3.5.2.1 Generate Tracklets

Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Make all tracklet pairs
- Merge multiple chained observation into single longer tracklets
- Purge any tracklets inconsistent with the merged tracklets

---

[3] Also sometimes referred to as 'Moving Object'

3.5.2.2 Attribution and precovery

Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Predict locations of known Solar System objects
- Match tracklet observation to predicted ephimerides taking into account velocity
- Update SSObjects
- Possibly iterate

3.5.2.3 Fit Orbits

Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Merge unassociated tracklets into tracks.
- Fit orbits to all tracks.
- Purge unphysical tracks.
- Update SSObjects
- Possibly iterate

3.5.2.4 Association and Precovery: New SSObjects

Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Do association and precovery just for SSObjects just found
- Update SSObjects

3.5.2.5 Merge Orbits

Output Data?
Anscillary Products?
Actions in case of failure?
Alternative procedures?

Subtasks:

- Merge orbits with high probability of being the same orbit into a single SSObject

3.5.3 Prototype Implementation

Prototype MOPS codes are available at `https://github.com/lsst/mops_daymops` and `https://github.com/lsst/mops_nightmops`. We expect it will be possible to transfer a significant fraction of the existing code into Construction. Current DayMOPS prototype already performs within the computational envelope envisioned for LSST Operations, though it does not yet reach the required completeness requirement.

4 Calibration Products Production

4.1 Calibration Products Pipeline (WBS 02C.04.02)

4.1.1 Key Requirements

The work performed in this WBS serves two complementary roles:

- It will enable the production of calibration data products as required by the Level 2 Photometric Calibration Plan (LSE-180) and other planning documents [20][4]. This includes both characterization of the sensitivity of the LSST system (optics, filters and detector) and the transmissivity of the atmosphere.
- It will characterize of detector anomalies in such a way that they can be corrected either by the instrument signature removal routines in the Single Frame Processing Pipeline (WBS 02C.03.01) or, if appropriate, elsewhere in the system;
- It will manage and provide a catalog of optical ghosts and glints to other parts of the system upon demand.

4.1.2 Baseline Design

4.1.2.1 Instrumental sensitivity We expect laboratory measurements of the filter profiles. We further baseline the development of a procedure for measuring the filter response at 1 nm resolution using the approach described in [20].

We baseline the following procedure for creating flat fields:

1. Record bias/dark frames;
2. Use "monochromatic" (1 nm) flat field screen flats with no filter in the beam to measure the per-pixel sensitivity;
3. Use a collimated beam projector (CBP) to measure the quantum efficiency (QE) at a set of points in the focal plane, dithering those points to tie them together;

---

[4] Resolving contradictions between these documents is out of scope here.

4. Combine the screen and CBP data to determine the broad band (10–100 nm) QE of all pixels;
5. Fold in the filter response to determine the 1 nm resolution effective QE of all pixels.

This WBS is responsible for the development of the data analysis algorithms and software required and the ultimate delivery of the flat fields. Development and commissioning of the CBP itself, together with any other infrastructure required to perform the above procedure, lies outwith Data Management (see 04C.08 *Calibration System*).

4.1.2.2 Atmospheric transmissivity Measurements from the auxiliary instrumentation—to include the 1.2 m "Calypso" telescope, a bore-sight mounted radiometer and satellite-based measurement of atmospheric parameters such as pressure and ozone—will be used to determine the atmospheric absorption along the line of sight to standard stars. The atmospheric transmission will be decomposed into a set of basis functions and interpolated in space in time to any position in the LSST focal plane.

This WBS will develop a pipeline for accurate spectrophotometric measurement of stars with the auxiliary telescope. We expect to repurpose and build upon publicly available code e.g. from the PFS[5] project for this purpose.

This WBS will construct the atmospheric model, which may be based either on MODTRAN (as per LSE-180) or a PCA-like decomposition of the data (suggested by [20]).

This WBS will define and develop the routine for fitting the atmospheric model to each exposure from the calibration telescope and providing estimates of the atmospheric transmission at any point in the focal plane upon request.

4.1.2.3 Detector effects An initial cross-talk correction matrix will be determined by laboratory measurements on the Camera Calibration Optical Bench (CCOB). However, to account for possibile instabilities, this WBS will develop an on-telescope method. We baseline this as being based on measurement with the CBP, but we note the alternative approach based on cosmic rays adopted by HSC [13].

Multiple reflections between the layers of the CCD give rise to spatial variability with fine scale structure in images which may vary with time [20,

---

[5] Subaru's Prime Focus Spectrograph; `http://sumire.ipmu.jp/pfs/`.

§2.5.1]. These can be characterized by white light flat-fields. Preliminary analysis indicates that these effects may be insignificant in LSST [23]; however, the baseline calls for a a routine developed in this WBS to analyse the flat field data and generate fringe frames on demand. This requirement may be relaxed if further analysis (outside the scope of thie WBS) demonstrates it to be unnecessary.

This WBS will develop algorithms to characterize and mitigate anomalies due to the nature of the camera's CCDs.

Note:
There's a complex inter-WBS situation here: the actual mitigation of CCD anomalies will generally be performed in SFM (WBS 02C.03.01), based on products provided by this WBS which, in turn, may rely on laboratory based research which is broadly outside the scope of DM. We baseline the work required to develop the corrective algorithms here. We consider moving it to WBS 02C.03.01 in future.

The effects we anticipate include:

- QE variation between pixels;
- Static non-uniform pixel sizes (e.g. "tree rings" [27]);
- Dynamic electric fields (e.g. "brighter-fatter" [2]);
- Time dependent effects in the camera (e.g. hot pixels, changing cross-talk coefficients);
- Charge transfer (in)efficiency (CTE).

Laboratory work required to understand these effects is outwith the scope of this WBS. In some cases, this work may establish that the impact of the effect may be neglected in LSST. The baseline plan addresses these issues through the following steps:

- Separate QE from pixel size variations[6] and model both as a function of position (and possibly time);

---

[6] Refer to work by Rudman.

- Learn how to account for pixel size variation over the scale of objects (e.g. by redistributing charge);
- Develop a correction for the brighter-fatter effect and develop models for any features which cannot be removed;
- Handle edge/bloom using masking or charge redistribution;
- Track defects (hot pixels);
- Handle CTE, including when interpolating over bleed trails.

4.1.2.4 Ghost catalog The Calibration Products Pipeline must provide a catalog of optical ghosts and glints which is available for use in other parts of the system. Detailed characterization of ghosts in the LSST system will only be possible when the system is operational. Our baseline design therefore calls for this system to be prototyped using data from precursor instrumentation; we note that ghosts in e.g. HSC are well known and more significant than are expected in LSST.

Note:
It is not currently clear where the responsibility for characterizing ghosts and glints in the system lies. We assume it is outwith this WBS.

4.1.3 Constituent Use Cases and Diagrams

Produce Master Fringe Exposures; Produce Master Bias Exposure; Produce Master Dark Exposure; Calculate System Bandpasses; Calculate Telescope Bandpasses; Construct Defect Map; Produce Crosstalk Correction Matrix; Produce Optical Ghost Catalog; Produce Master Pupil Ghost Exposure; Determine CCOB-derived Illumination Correction; Determine Optical Model-derived Illumination Correction; Create Master Flat-Spectrum Flat; Determine Star Raster Photometry-derived Illumination Correction; Create Master Illumination Correction; Determine Self-calibration Correction-Derived Illumination Correction; Correct Monochromatic Flats; Reduce Spectrum Exposure; Prepare Nightly Flat Exposures;

4.1.4 Prototype Implementation

While parts of the Calibration Products Pipeline have been prototyped by the LSST Calibration Group (see the LSE-180 for discussion), these have not been written using LSST Data Management software framework or coding standards. We therefore expect to transfer the know-how, and rewrite the implementation.

4.2 Photometric Calibration Pipeline (WBS 02C.03.07)

4.2.1 Key Requirements

The Photometric Calibration Pipeline is required to internally calibrate the relative photometric zero-points of every observation, enabling the Level 2 catalogs to reach the required SRD precision.

4.2.2 Baseline Design

The adopted baseline algorithm is a variant of “ubercal” [22, 25]. This baseline is described in detail in the Photometric Self Calibration Design and Prototype Document (UCAL).

4.2.3 Constituent Use Cases and Diagrams

Perform Global Photometric Calibration;

4.2.4 Prototype Implementation

Photometric Calibration Pipeline has been fully prototyped by the LSST Calibration Group to the required level of accuracy and performance (see the UCAL document for discussion).

As the prototype has not been written using LSST Data Management software framework or coding standards, we assume a non-negligible refactoring and coding effort will be needed to convert it to production code in LSST Construction.

4.3 Astrometric Calibration Pipeline (WBS 02C.03.08)

4.3.1 Key Requirements

The Astrometric Calibration Pipeline is required to calibrate the relative and absolute astrometry of the LSST survey, enabling the Level 2 catalogs to reach the required SRD precision.

4.3.2 Baseline Design

Algorithms developed for the Photometric Calibration Pipeline (WBS 02C.03.07) will be repurposed for astrometric calibration by changing the relevant functions to minimize. This pipeline will further be aided by WCS and local astrometric registration modules developed as a component of the Single Frame Processing pipeline (WBS 02C.03.01).

Gaia standard stars will be used to fix the global astrometric system. It is likely that the existence of Gaia catalogs may make a separate Astrometric Calibration Pipeline unnecessary.

4.3.3 Constituent Use Cases and Diagrams

Perform Global Astrometric Calibration;

4.3.4 Prototype Implementation

The Astrometric Calibration Pipeline has been partially prototyped by the LSST Calibration Group, but outside of LSST Data Management software framework. We expect to transfer the know-how, and rewrite the implementation.

5 Data Release Production

TODO:
Update figure to reflect changes to sections in text.

A Data Release Production is run every year (twice in the first year of operations) to produce a set of catalog and image data products derived from all observations from the beginning of the survey to the point the production began. This includes running a variant of the difference image analysis run in Alert Production, in addition to direct analysis of individual exposures and coadded images. The data products produced by a Data Release Production are summarized in table 1.

From a conceptual standpoint, data release production can be split into five groups of pipelines, executed in approximately the following order:

1. We characterize and calibrate each exposure, estimating point-spread functions, background models, and astrometric and photometric calibration solutions. This iterates between processing individual exposures independently and jointly fitting catalogs derived from multiple overlapping exposures. These steps are described more fully in section 5.1.

2. We alternately combine images and subtract them, using differences to find artifacts and time-variable sources while building coadds that produce a deeper view of the static sky. Coaddition and difference imaging is described in section 5.2.

3. After all image processing is complete, we run additional catalog-only pipelines to fill in additional object properties. Unlike previous stages, this postprocessing is not localized on the sky, as it may use statistics computed from the full data release to improve our characterization of individual objects. Postprocessing pipelines are described in section 5.6.

This conceptual ordering is an oversimplification of the actual processing flow, however; as shown in Figure 4, pipeline groups are actually interleaved.

Each pipeline in this the diagram represents a particular piece of code excuted in parallel on a specific unit of data, but pipelines may contain additional (and more complex) parallelization to further subdivide that data unit. The processing flow also includes the possibility of iteration between pipelines, indicated by cycles in the diagram. The number of iterations in each cycle will be determined (via tests on smaller productions) before the start of

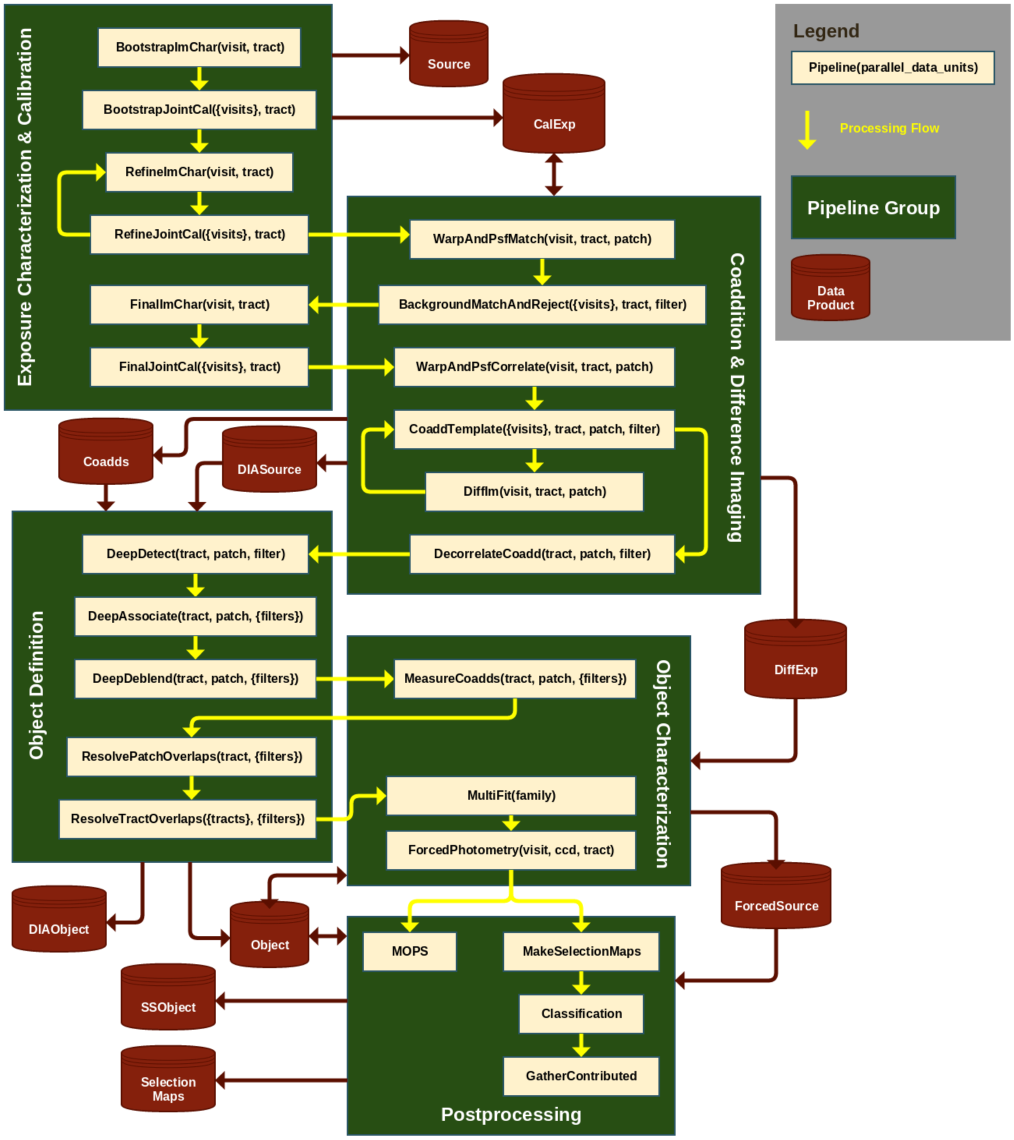


Figure 4: Summary of the Data Release Production processing flow. Processing is split into multiple pipelines, which are conceptually organized into the groups discussed in sections 5.1-5.6.

Name	**Availability**	**Description**
Source	Stored	Measurements from direct analysis of individual exposures.
DIASource	Stored	Measurements from difference imagine analysis of individual exposures.
Object	Stored	Measurements for a single astrophysical object, derived from all available information, including coadd measurements, simultaneous multi-epoch fitting, and forced photometry. Does not include solar system objects.
DIAObject	Stored	Aggregate quantities computing by associating spatially colocated DIASources.
ForcedSource	Stored	Flux measurements on each direct and difference image at the position of every Object.
SSObject	Stored	Solar system objects derived by associating DIASources and inferring their orbits.
CalExp	Regenerated	Calibrated exposure images for each CCD/visit (sum of two snaps).
DiffExp	Regenerated	Difference between CalExp and PSF-matched template coadd.
DeepCoadd	Stored	Coadd image with a reasonable combination of depth and resolution.
EpochRangeCoadd	Renegerated	Coadd image that cover only a limited range of epochs.
BestSeeingCoadd	Regenerated	Coadd image built from only the best-seeing images.
PSFMatchedCoadd	Regenerated	Coadd image with a constant, predetermined PSF.

Table 1: Table of public data products produced during a Data Release Production. A full description of these data products can be found in the Data Products Definition Document (LSE-163).

the production, allowing us to remove these cycles simply by duplicating some pipelines a fixed number of times. The final data release production processing can thus be described as a directed acyclic graph (DAG) to be executed by the orchestration middleware, with pipelines as edges and (intermediate) data products as vertices. Most of the graph will be generated by applications code before the production begins, using a format and/or API defined by the orchestration middleware. Howver, some parts of the graph must be generated on-the-fly; this will be discussed further in section 5.5.1.

5.1 Image Characterization and Calibration

ImChar/JointCal Diagram:
Extract ImChar/JointCal pipelines from "DRP Top-Level Overview" on confluence and expand detail to show data flow and ordering of "Task/Process" boxes.

The first steps in a Data Release Production characterize the properties of individual exposures, by iterating between pixel-level processing of individual visits ("ImChar", or "Image Characterization" steps) and joint fitting of all catalogs overlapping a tract ("JointCal", or "Joint Calibration" steps). All ImChar steps involve fitting the PSF model and measuring Sources (gradually improving these as we iterate), while JointCal steps fit for new astrometric (WCS) and photometric solutions while building new reference catalogs for the ImChar steps. Iteration is necessary for a few reasons:

- The PSF and WCS must have a consistent definition of object centroids. Celestial positions from a reference catalog are transformed via the WCS to set the positions of stars used to build the PSF model, but the PSF model is then used to measure debiased centroids that feed the WCS fitting.
- The later stages of photometric calibration and PSF modeling require secure star selection and colors to infer their SEDs. Magnitude and morphological measurements from ImChar stages are aggregated the reference catalog in the subsequent JointCal stage, allowing these colors and classifications to be used for PSF modeling in the following ImChar stage.

The ImChar and JointCal iteration is itself interleaved with background matching and difference imaging, as described in section 5.2. This allows the better backgrounds and masks to be defined by comparisons between images before the final Source measurements, image characterizations, and calibrations.

Each ImChar pipeline runs on a single visit, and each JointCal pipeline runs simultaneously on all visits within a single tract, allowing tracts to be run entirely independently.

The final output data products of the ImChar/JointCal iteration are the Source table and most of the CalExp (calibrated exposure) images. CalExp is an Exposure, and hence has Image, Mask, Variance, Background, PSF, WCS, and PhotoCalib components that we will track separately.

5.1.1 BootstrapImChar

The BootstrapImChar pipeline is the first thing run on each science exposure in a data release. It has the difficult task of bootstrapping multiple quantities (PSF, WCS, photometric calibration, background model, etc.) that each normally require all of the others to be specified when one is fit. As a result, while the algorithmic components to be run in this pipeline are generally clear, their ordering and specific requirements are not; algorithms that are run early will have a harder task than algorithms that are run later, and some iteration will almost certainly be necessary.

A plausible (but by no means certain) high-level algorithm for this pipeline is given below in pseudocode. Highlighted terms are described in more detail below the pseudocode block.

```
def BootstrapImChar(raw, reference):
    # Some data products components are visit-wide and some are per-CCD;
    # these imaginary data types lets us deal with both.
    # VisitExposure also has components; most are self-explanatory, and
    # {mi} == {image,mask,variance} (for "MaskedImage").
    calexp = VisitExposure()
    sources = VisitCatalog()
    snaps = VisitMaskedImageList()  # holds both snaps, but only {image,mask,variance}
    parallel for ccd in ALL_SENSORS:
        snaps[ccd] = [RunISR(raw[ccd]) for snap in SNAP_NUMBERS]
        snaps[ccd].mask = SubtractSnaps(snaps[ccd])
        calexp[ccd].mi = CombineSnaps(snaps[ccd])
    calexp.psf = FitWavefront(calexp[WAVEFRONT_SENSORS].mi)
    calexp.{image,mask,variance,background}
        = SubtractBackground(calexp.mi)
    parallel for ccd in ALL_SENSORS:
        sources[ccd] = DetectSources(calexp.{mi,psf})
    sources[ccd] = DeblendSources(sources[ccd], calexp.{mi,psf})
    sources[ccd] = MeasureSources(sources[ccd], calexp.{mi,psf})
    matches = MatchSemiBlind(sources, reference)
    while not converged:
```

```
        SelectStars(matches, exposures)
        calexp.wcs = FitWCS(matches, sources, reference)
        calexp.psf = FitPSF(matches, sources, calexp.{mi,wcs})
        WriteDiagnostics(snaps, calexp, sources)
        parallel for ccd in ALL_SENSORS:
            snaps[ccd] = SubtractSnaps(snaps[ccd], calexp[ccd].psf)
            calexp[ccd].mi = CombineSnaps(snaps[ccd])
            calexp[ccd].mi = SubtractStars(calexp[ccd].{mi,psf}, sources[ccd])
        calexp.{mi,background} = SubtractBackground(calexp.mi)
        parallel for ccd in ALL_SENSORS:
            sources[ccd] = DetectSources(calexp.{mi,psf})
            calexp[ccd].mi, sources[ccd] =
                ReinsertStars(calexp[ccd].{mi,psf}, sources[ccd])
            sources[ccd] = DeblendSources(sources[ccd], calexp.{mi,psf})
            sources[ccd] = MeasureSources(sources[ccd], calexp.{mi,psf})
        matches = MatchNonBlind(sources, reference)
    calexp.psf.apcorr = FitApCorr(matches, sources)
    parallel for ccd in SCIENCE_SENSORS:
        sources[ccd] = ApplyApCorr(sources[ccd], calexp.psf)
    return calexp, sources
```

5.1.1.1 Input Data Product: Raw Raw amplifier images from science and wavefront CCDs, spread across one or more snaps. Needed telescope telemetry (seeing estimate, approximate pointing) is assumed to be included in the raw image metadata.

5.1.1.2 Input Data Product: Reference A full-sky catalog of reference stars derived from both external (e.g. Gaia) and LSST data.

The StandardJointCal pipeline will later define a deeper reference catalog derived from this one and the new data being processed, but the origin and depth of the initial reference catalog is largely TBD. It will almost certainly include Gaia stars, but it may also include data from other telescopes, LSST special programs, LSST commissioning observations, and/or the last LSST data release. Decisions will require some combination of negotation with the LSST commissioning team, specification of the special programs, quality analysis and experimentation with the Gaia catalog, and policy decisions from DM leadership on the degree to which data releases are required to be independent. Depending on the choices selected, it could also require a major separate processing effort using modified versions of the data release production pipelines.

5.1.1.3 Output Data Product: Source A preliminary version of the Source table. This could contain all of the columns in the DPDD Source schema if the MeasureSources is appropriately configured, but some of these columns are likely unnecessary in its role as an intermediate data product that feeds StandardJointCal, and it is likely that other non-DPDD columns will be present for that role.

BootstrapImChar also has the capability to produce even earlier versions of the Source table for diagnostic purposes (see WriteDiagnostics). These tables are not associated with any photometric calibration or aperture correction, and some may not have any measurements besides centroids, and hence are never substitutable for the final Source table.

5.1.1.4 Output Data Product: CalExp A preliminary version of the CalExp (calibrated direct exposure). CalExp is an Exposure object, and hence it has several components. BootstrapImChar is the only pipeline that actually updates all of them. Some CalExp components are determined at the scale of a full FoV and hence should probably be persisted at the visit level (PSF, WCS, PhotoCalib, Background), while others are straightforward CCD-level data products (Image, Mask, Variance).

5.1.1.5 RunISR Delegate to the ISR algorithmic component to perform standard detrending as well as brighter-fatter correction and interpolation for pixel-area variations (Warping Irregularly-Sampled Images). It is possible that these corrections will require a PSF model, and hence must be backed-out and recorrected at a later stage when an improved PSF model is available.

We assume that the applied flat field is appropriate for background estimation.

5.1.1.6 SubtractSnaps Delegate to the Snap Subtraction algorithmic component to mask artifacts in the difference between snaps. If passed a PSF (as in the second call), also interpolate them by delegating to the Artifact Interpolation algorithmic component.

We assume here that the PSF modeled on the combination of the two Snaps is sufficient for interpolation on the Snaps individually; if this is not true, we can just mask and interpolate both Snaps when an artifact appears on either of them (or we could do per-Snap PSF estimation, but that's a lot more work for very little gain).

5.1.1.7 CombineSnaps Delegate to the Image Coaddition algorithmic component to combine the two Snaps while handling masks appropriately.

We assume there is no warping involved in combining snaps. If this is needed, we should instead advocate for dropping snaps in favor of a a single longer exposure.

5.1.1.8 FitWavefront Delegate to the Wavefront Sensor PSF algorithmic component to generate an approximate PSF using only data from the wavefront sensors and observational metadata (e.g. reported seeing).

Processing the wavefront sensors will likely require some form of detection and measurement; we currently consider this to be part of the Wavefront Sensor PSF code, though it may delegate to e.g. Source Detection and/or Single Frame Measurement.

The required quality of this PSF estimate is TBD; setting preliminary requirements will involve running a version of BootstrapImChar with at least mature detection and PSF-modeling algorithms on precursor data taken in crowded fields, and final requirements will require proceessing full LSST camera data in crowded fields. However, robustness to poor data quality and crowding is much more important than accuracy; this stage need only provide a good enough result for subsequent stages to prcoeed.

5.1.1.9 SubtractBackground Delegate to the Background Estimation algorithmic component to model and subtract the background consistently over the full field of view.

The multiple backgrounds subtracted in BootstrapImChar may or may not be cumulative (i.e. we may or may not add the previous background back in before estimating the latest one).

5.1.1.10 DetectSources Delegate to the Source Detection algorithmic component to find above-threshold regions (Footprints) and peaks within them in a PSF-correlated version of the image.

In crowded fields, each iteration of detection will decrease the threshold, increasing the number of objects detection. Because this will treat fluctuations in the background due to undetected objects as noise, we may need to extend PSF-correlation to the appropriate filter for an image with correlated noise and characterize the noise field from the image itself.

Because we will use wavefront data to constrain the PSF, we also run detection on the wavefront sensors. It is possible that this will require a different algorithmic component if we cannot just treat the wavefront sensors as science sensors with an out-of-focus PSF.

5.1.1.11 DeblendSources

Delegate to the Single Frame Deblending algorithmic component to split Footprints with multiple peaks into deblend families.

Because we will use wavefront data to constrain the PSF, we also run deblending on the wavefront sensors. It is possible that this will require a different algorithmic component if we cannot just treat the wavefront sensors as science sensors with an out-of-focus PSF, and we need deblending to extract wavefront information.

5.1.1.12 MeasureSources

Delegate to the Single Frame Measurement algorithmic component to measure source properties.

In BootstrapImChar, we anticipate using the Neighbor Noise Replacement approach to deblending, with the following plugin algorithms:

- Centroids
- Second-Moment Shapes
- Pixel Flag Aggregation
- Aperture Photometry (but only for one or two radii)
- Static Point Source Models

Because we will use wavefront data to constrain the PSF, we also run measurement on the wavefront sensors (but probably without any flux measurement algorithms, and perhaps with modified versions of other algorithms). It is possible that this will require a different algorithmic component if we cannot just treat the wavefront sensors as science sensors with an out-of-focus PSF.

5.1.1.13 MatchSemiBlind

Delegate to the Single Visit Reference Matching algorithmic component to match source catalogs to a global reference catalog. This occurs over the full field of view, ensuring robust matching even when some CCDs have no matchable stars due to crowding, flux limits, or artifacts.

"Semi-Blind" refers to the fact that the WCS is not yet well known (all we have is what is provided by the observatory), so the matching algorithm must account for an unknown (but small) offset between the WCS-predicted sources positions and the reference catalog positions.

5.1.1.14 SelectStars Use reference catalog classifications and source flags to select a clean sample stars to use for later stages.

If we decide not to rely on a pre-existing reference catalog to separate stars from galaxies and other objects, we will need a new algorithmic component to select stars based on source measurements.

5.1.1.15 FitWCS Delegate to the Single Visit Astrometric Fit algorithmic component to determine the WCS of the image.

We assume this works by fitting a simple mapping from the visit's focal plane coordinate system to the sky and composing it with the (presumed fixed) mapping between CCD coordinates and focal plane coordinates. This fit will be improved in later pipelines, so it does not need to be exact; <0.05 arcsecond accuracy should be sufficient.

As we iterate in crowded fields, the number of degrees of freedom in the WCS should be allowed to slowly increase.

5.1.1.16 FitPSF Delegate to the Full Visit PSF Modeling algorithmic component to construct an improved PSF model for the image.

Because we are relying on a reference catalog to select stars, we should be able to use colors from the reference catalog to estimate SEDs an include wavelength dependence in the fit. If we do not use a the reference catalog early in BootstrapImChar, PSF estimation here will not be wavelength-dependent. In either case the PSF model will be further improved in later pipelines.

PSF estimation at this stage must include some effort to model the wings of bright stars, even if this is tracked and constrained separately from the model for the core of the PSF.

As we iterate in crowded fields, the number of degrees of freedom in the PSF model should be allowed to slowly increase.

5.1.1.17 WriteDiagnostics If desired, the current state of the `source`, `calexp`, and `snaps` variables may be persisted here for diagnostic purposes.

5.1.1.18 SubtractStars Subtract all detected stars above a flux limit from the image, using the PSF model. In crowded fields, this should allow subsequent SubtractBackground and DetectSources steps to push fainter by removing the brightest stars in the image.

Sources classified as extended are never subtracted.

5.1.1.19 ReinsertStars Add stars removed in SubtractStars back into the image, and merge corresponding Footprints and peaks into the source catalog.

5.1.1.20 MatchNonBlind Match a single-CCD source catalog to a global reference frame, probably by delegating to the same matching algorithm used in JointCal pipelines. A separate algorithm component may be needed for efficiency or code maintenance reasons; this is a simple limiting case of the multi-way JointCal matching problem that may or may not merit a separate simpler implementation.

"Non-Blind" refers to the fact that the WCS is now known well enough that there is no significant offset between WCS-projected source positions and reference catalog positions.

5.1.1.21 FitApCorr Delegate to the Aperture Correction algorithmic component to construct a curve of growth from aperture photometry measurements and build an interpolated mapping from other fluxes to the predicted integrated flux at infinity.

5.1.1.22 ApplyApCorr Delegate to the Aperture Correction algorithmic component to apply aperture corrections to flux measurements.

5.1.2 StandardJointCal

In StandardJointCal, we jointly process all of the Source tables produced by running BootstrapImChar on each visit in a tract. There are four steps:

1. We match all sources and the reference catalog by delegating to JointCalMatching. This is a non-blind search; we assume the WCSs output

by BootstrapImChar are good enough that we don't need to fit for any additional offsets between images at this stage. Some matches will not include a reference object, as the sources will almost certainly extend deeper than the reference catalog.

2. We classify matches to select a clean sample of low-variability stars for later steps, delegating to JointCalClassification. This uses morphological and possibly color information from source measurements as well as reference catalog information (where available). This step also assigns an inferred SED to each match from its colors; for matches associated with a reference object, whether this supersedes SEDs or colors in the reference catalog is depends on our approach to absolute calibration.

3. We fit simultaneously for improved astrometric solution by requiring each star in a match to have the same position. This may need to correct (perhaps approximately) for centroid shifts due to DCR and/or proper motion; if it does not, it must be robust against these shifts (perhaps via outlier rejection). The models and parameters to fit must be determined by experimentation, but they will represent further perturbation of the WCS fit in BootstrapImChar. This fit generates a new WCS component for each CalExp.

4. We fit simultaneously for photometric zeropoints by requiring each star in a match to have the same flux after applying smoothed monochromatic flat fields produced by the calibration products pipeline. There is a small chance this fit will also be used to further constrain those monochromatic flat fields. This fit generates a new PhotoCalib component for each CalExp.

In addition to updating the CalExp WCS and PhotoCalib, StandardJointCal generates a new Reference dataset containing the joint-fit centroids and fluxes for each of its match groups as well as their classifications and inferred SEDs.

StandardJointCal may be iterated with RefineImChar to ensure the PSF and WCS converge on the same centroid definitions. StandardJointCal is always run immediately after BootstrapImChar, but RefineImChar or StandardJointCal may be the last step in the iteration run before proceding with WarpAndPsfMatch.

5.1.3 RefineImChar

RefineImChar performs an incremental improvement on the measurements and PSF model produced by BootstrapImChar, using the improved reference catalog, WCS, and PhotoCalib produced by StandardJointCal. Its steps are thus a strict subset of those in BootstrapImChar. A pseudocode description of RefineImChar is given below, but all steps refer to back to the descriptions in 5.1.1:

```
def RefineImChar(calexp, sources, reference):
    matches = MatchNonBlind(sources, reference)
    SelectStars(matches, exposures)
    calexp.psf = FitPSF(matches, sources, calexp.{mi,wcs})
    parallel for ccd in ALL_SENSORS:
        calexp[ccd].mi = SubtractStars(calexp[ccd].{mi,psf}, sources[ccd])
    calexp.{mi,background} = SubtractBackground(calexp.mi)
    parallel for ccd in ALL_SENSORS:
        sources[ccd] = DetectSources(calexp.{mi,psf})
        calexp[ccd].mi, sources[ccd] =
            ReinsertStars(calexp[ccd].{mi,psf}, sources[ccd])
        sources[ccd] = DeblendSources(sources[ccd], calexp.{mi,psf})
        sources[ccd] = MeasureSources(sources[ccd], calexp.{mi,psf})
    calexp.psf.apcorr = FitApCorr(matches, sources)
    parallel for ccd in ALL_SENSORS:
        sources[ccd] = ApplyApCorr(sources[ccd], calexp.psf)
    return calexp, sources
```

This is essentially just another iteration of the loop in in BootstrapImChar, without the WCS-fitting or artifact-handling stages. We assume that we continue to process the wavefront sensors here (because we will use them in the FitPSF step), but it may be that previous processing may be sufficient.

Note that RefineImChar does not update the CalExp's WCS, PhotoCalib, Image, or Variance (and its Mask is only updated to indicate new detections).

5.1.4 FinalImChar

FinalImChar is responsible for producting the final PSF models and source measurements. While similar to RefineImChar, it is run after at least one iteration of the BackgroundMatchAndReject and possibly UpdateMasks pipelines, which provide it with the final background model and mask.

The steps in FinalImChar are identical to those in RefineImChar, with just a few exceptions:

- The background is not re-estimated and subtracted.

- The suite of plugin run by Single Frame Measurement is expanded to included all algorithms indicated in the first column of Figure 6. This should provide all measurements in the DPDD Source table description.
- We also classify sources by delegating to Single Frame Classification, to fill the final Source table's *extendedness* field. It is possible this will also be run during RefineImChar and BootstrapImChar for diagnostic purposes.

5.1.5 FinalJointCal

FinalJointCal is *almost* identical to StandardJointCal, and the details of the differences when surrounding pipelines are more mature and the approach to absolute calibration is more clear. Because it is responsible for the final photometric calibration, it may be need to perform some steps that could be omitted from StandardJointCal because they have no impact on the ImChar pipelines. This could include a role in determining the absolute photometric calibration of the survey, especially if an external catalog (e.g. Gaia) is relied upon exclusively to tie different tracts together.

There is no need for FinalJointCal to produce a new or updated Reference dataset (except for its own internal use), as subsequent steps do not need one, and the DRP-generated reference catalog used by Alert Production will be derived from the Object table.

5.2 Coaddition and Difference Imaging

The next group of pipelines in a Data Release Production consist of image coaddition and image differencing, which we use to separate the static sky from the dynamic sky in terms of both astrophysical quantities and observational quantities. This group also includes an iteration between pipelines that combine images and pipelines that subtract the combined images from each exposure. At each differencing step, we better characterize the features that are unique to a single epoch (whether artifacts, background features, or astrophysical sources); we use these characterizations to ensure the next round of coadds include only features that are common to all epochs.

The processing flow in this pipeline group again centers around incremental updates to the CalExp dataset, which are limited here to its Background and Mask component (the Image component is also updated, but only to subract

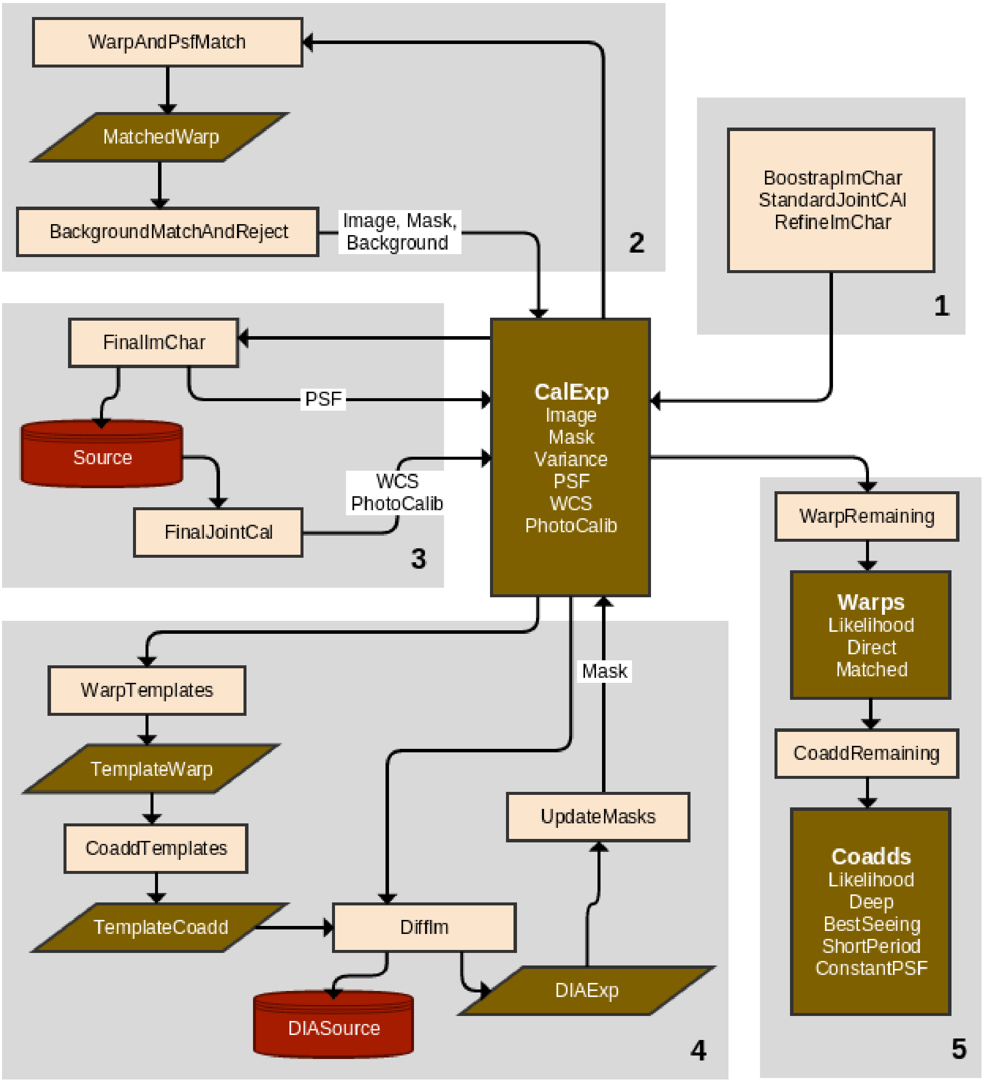


Figure 5: Data flow diagram for the Data Release Production coaddition and difference imaging pipelines. Processing proceeds roughly counterclockwise, starting from the upper right with pipelines described in Section 5.1. Each update to a component of the central CalExp dataset can in theory trigger another iteration of a previous loop, but in practice we will "unroll" these loops before production begins, yielding an acyclic graph with a series of incrementally updated CalExp datasets. The nature of this unrolling and the number of iterations will be determined by future algorithmic research. Numbered steps above are described more fully in the text.

the updated background). It will also return to the previous pipeline group described in Section 5.1 to update other CalExp components. The data flow between pipelines is shown in Figure 5, with the numbered steps are described further below:

1. The first version of the CalExp dataset is produced by running the BootstrapImChar, StandardJointCal, and RefineImChar pipelines, as described in Section 5.1.

2. We generate an updated Background and Mask via the WarpAndPsfMatch and BackgroundMatchAndReject pipelines. This produces the final CalExp Background and Image, and possibly the final Mask.

3. If the CalExp Mask has been finalized, we run the FinalImChar and FinalJointCal pipelines. These produce the final PSF, WCS, and PhotoCal. If the Mask has not been finalized, we execute at least one iteration of the next step before this one.

4. We run the WarpTemplates, CoaddTemplates, and DiffIm pipelines to generate the DIASource and DIAExp datasets. We may then be able to generate better CalExp Masks than we can obtain from BackgroundMatchAndReject by comparing the DIAExp masks across visits in the UpdateMasks pipeline.

5. After all CalExp components have been finalized, we run the WarpRemaining and CoaddRemaining to build additional coadd data products.

The baseline ordering of these steps is thus {1,2,3,4,5}, but {1,2,4,3,4,5} is perhaps just as likely, and we may ultimately require an ordering that repeats steps 2 or 3. Final decisions on the ordering and number of iteration will require testing with mature pipelines and a deep dataset taken with a realistic cadence; it is possible the configuration could even change between data releases as the survey increases in depth. Fortunately, this reconfiguring should not require significant new algorithm development.

This pipeline group is responsible for producing the following final data products:

CalExp.{Image,Background,Mask} See above.

DIAExp A CCD-level Exposure that is the difference between the CalExp and a template coadd, in the coordinate system and with the PSF of the CalExp.

DIASource A SourceCatalog containing sources detected and measured on the DIAExp images.

ConstantPSFCoadd A coadd data product (Exposure or subclass thereof) with a constant, predefined PSF.

DeepCoadd A coadd data product built to optimize depth at the possible expense of seeing.

BestSeeingCoadd A coadd data product built to optimize image quality at the possible expense of depth. Depending on the algorithm used, this may be the same as DeepCoadd.

ShortPeriodCoadd A coadd data product built from exposures in a short range of epochs, such as a year, rather than the full survey.

LikelihoodCoadd A coadd formed by correlating each image with its own PSF before combining them, used for detection and possibly building other coadds.

ShortPeriodLikelihoodCoadd Short-period likelihood coadds will also be built.

TemplateCoadd A coadd data product used for difference imaging in both DRP and AP, *or* some other tract/patch/filter data product from which the template for a given airmass can be derived.

The nature of these coadd data products depends critically on whether we are able to develop efficient algorithms for optimal coaddition, and whether these coadds are suitable for difference imaging. These algorithms are mathematically well-defined but computationally difficult; see DMTN-15 for more information. We will refer to the coadds produced by these algorithms as "decorrelated coadds"; a variant with constant PSF ("constant-PSF partially decorrelated coadd") is also possible. This choice is also mixed with the question of how we will correct for differential chromatic refraction in difference imaging; some algorithms for DCR correction involve templates that are the result of inference on input exposures rather than coaddition. There are five main scenarios for our use of decorrelated coadds:

A We use decorrelated coadds for all final coadd products. DeepCoadd and ShortPeriodCoadd will be standard decorrelated coadds with a spatially-varying PSF, and ConstantPSFCoadd and TemplateCoadd will be constant-PSF partially-decorrelated coadds. The BestSeeingCoadd data product will be dropped, as it will be redundant with DeepCoadd. This will make coadds more expensive and complex to build, and require more algorithm development for coaddition, but will improve coadd-based measurements and make it easier to warm-start multi-epoch measurements. Difference imaging may be easier, and more visits may be usable as inputs to templates due to softened or eliminated seeing cut.

B We use decorrelated coadds for all coadds but TemplateCoadd. Measurement is still improved, and the additional computational cost of coaddition is limited to a single pipeline that is not run iteratively. Difference imaging may be harder, and the number of visits eligible for inclusion in templates may be reduced. In this scenario, we still have two options for building templates:

- **B1** Templates will be built as PSF-matched coadds, or a product of PSF-matched coadds.
- **B2** Templates are the result of inference on resampled exposures with no PSF-matching.

C We do not use decorrelated coadds at all. DeepCoadd, BestSeeingCoadd, and ShortPeriodCoadd will be direct coadds, and ConstantPSFCoadd will be a PSF-matched coadd. Coaddition will be simpler and faster, but downstream algorithms may require more sophistication, coadd measurements may be lower quality, and multi-epoch measurements may be more difficult to optimize. Here we again have the same two options for templates as option **B**:

- **C1** Templates will be built as PSF-matched coadds, or a product of PSF-matched coadds.
- **C2** Templates are the result of inference on resampled exposures with no PSF-matching.

It is also possible to combine multiple scenarios across different bands. In particular, we may not need special templates to handle DCR in redder bands, so we may select a simpler approach to dealing with them in those bands. The final selection between these options will require experiments on LSST data or precursor data with similar DCR and seeing, though decorrelated coaddition algorithms and some approaches to DCR correction may be ruled out earlier if preliminary algorithm development does not go well.

Further differences in the pipelines themselves due to the presence or absence of decorrelated coadds will be described in the sections below.

5.2.1 WarpAndPsfMatch

This pipeline resamples and then PSF-matches CalExp images from a visit into a single patch-level image with a constant PSF. The resampling and PSF-matching can probably be accomplished separately by delegating to the Image Warping and PSF Homogenization algorithmic components, respectively. These operations can also be performed in the opposite order if the matched-to PSF is first transformed to the CalExp coordinate systems (so subsequent resampling yields a constant PSF in the coadd coordinate system). Doing PSF-matching first may be necessary (or at least easier to implement) for undersampled images.

It is possible these operations will be performed simultaneously by a new algorithmic component; this could potentially yield improved computational performance and make it easier to properly track uncertainty. These improvements are unlikely to be necessary for this pipeline, because these images and the coadds we build from them will only be used to estimate backgrounds and find artifacts, and these operations only require approximate handling of uncertainty. However, other coaddition pipelines may require building an algorithmic component capable of warping and PSF-matching simultaneously, and if that happens, we would likely use it here as well.

The only output of the WarpAndPsfMatch pipeline is the MatchedWarp Exposure intermediate data product. It contains all of the usual Exposure components, which must be propagated through the image operations as well. There is a separate MatchedWarp for each {patch, visit} combination, and these can be produced by running WarpAndPsfMatch independently on each such combination. However, individual CCD-level CalExps will be required by multiple patches, so I/O use or data transfer may be improved by running all WarpAndPsfMatch instances for a given visit together.

5.2.2 BackgroundMatchAndReject

This pipeline is responsible for generating our final estimates of the sky background and updating our artifact masks. It is one of the most algorithmically uncertain algorithms in Data Release Production, particularly from the standpoint of large-scale data flow and parallelization, because while the algorithm is well-defined and understood for a small patch of sky, we do not yet have a concrete approach for extending it to large fields containing multiple dithered images (though it has been demonstrated on SDSS data, for which the drift-scan observation strategy makes the problem simpler). It still processes tracts independently, but below that scale the parallelization is very much uncertain, and may in fact involve splitting these operations further into smaller pipelines.

The steps involved in background matching are described below. All of these operations are performed on the MatchedWarp images; these are all in the same coordinate system and have the same PSF, so they can be meaningfully added and subtracted with no additional processing.

1. We define one of the visits that overlap an area of the sky as the *reference image*. At least in the naive local specification of the algorithm, this image must be smooth and continous over the region of interest. This is done by the Build Background Reference pipeline, which must artificially (but reversibly) enforce continuity in a reference image that stitches together multiple visits to form a single-epoch-deep full tract image, unless we develop an approach for dealing with discontinuity downstream.

2. We subtract the reference image from every other visit image. This must account for any artifical features due to the construction of the reference image.

3. We run Source Detection on the per-visit difference images to find artifacts and transient sources. We do not generate a traditional catalog of these detections, as they will only be used to generate improved CalExp masks; they will likely be stored as a sequence of Footprints.

4. We estimate the background on the per-visit difference images by delegating to the Background Estimation algorithmic component. This difference background should be easier to be model than a direct image

background, as the image will be mostly free of sources and astrophysical backgrounds.

5. We build a PSF-matched coadd by adding all of the visit images (including the reference) and subtracting all of the difference image backgrounds; this yields a coadd that contains only the reference image background, which we then model and subtract by again using the Background Estimation algorithmic component. Combining the images will be performed by the Coaddition algorithmic component, which will also generate new CalExp masks by analyzing the per-pixel, multi-visit histograms of image and mask values (e.g. generalized statistical outlier rejection) to distinguish transients and artifacts from variable sources.

We are assuming in the baseline plan that we can use a matched-to PSF in WarpAndPsfMatch large enough to match all visit images to it without deconvolution. If a large marged-to PSF adversely affects subsequent processing in BackgroundMatchAndReject, we may need to develop an iterative approach in which we apply WarpAndPsfMatch only to better-seeing visits first, using a smaller target PSF, run BackgroundMatchAndReject on these, and then re-match everything to a larger target PSF and repeat with a larger set of input visits. However, this problem would suggest that the DiffIm and UpdateMasks pipelines would be even better at finding artifacts, so a more likely mitigation strategy would be to simply defer final Mask generation to after at least one iteration of those pipelines, as described in the discussion of Figure 5 at the beginning of Section 5.2.

The outputs of BackgroundMarchAndReject are updated Background and Mask components for the CalExp product. Because it is not built with the final photometric and astrometric calibration, the PSF-matched coadd built here is discarded.

5.2.3 WarpTemplates

This pipeline is responsible for generating the resampled visit-level images (TemplateWarp) used to build template coadds for difference imaging. The algorithmic content of this pipeline and the nature of its outputs depends on whether we are using decorrelated coadds (option **A**) at the beginning of 5.2), PSF-matched coadds (**B1** or **C1**), or inferring templates (**B2** or **C2**).

If we are using decorrelated coadds (option **A**), the output is equivalent to the LikelihoodWarp data product produced by the WarpRemaining pipeline (aside from differences due to the state of the input CalExps), and the algorithm to produce it the same:

- We correlate the image with its own PSF by delegating to the Convolution Kernels software primitive.
- We resample the image by delegating to the Image Warping algorithmic component.

As with other convolution-and-warping pipelines, it is possible we will instead develop a single algorithmic component to perform both operations. These operations must include full propogation of uncertainty.

If we are not using decorrelated coadds (**B1** or **C1**), the output is equivalent to the MatchedWarp data product, and the algorithm is the same as the WarpAndPsfMatch pipeline that produces. We cannot reuse existing MatchedWarps simply because we need to utilize updated CalExps.

If we are inferring templates (**B2** or **C2**), this pipeline is only responsible for resampling, producing an output equivalent to the DirectWarp data product produced by the WarpRemaining pipeline. This work is delegated to the Image Warping algorithmic component.

5.2.4 CoaddTemplates

This pipeline generates the TemplateCoadd dataset used as the reference image for difference imaging. This may not be a simple coadd, at least in bluer bands; in order to correct for differential chromatic refraction during difference imaging, we may need to add a wavelength or airmass dimension to the usual 2-d image, making a 3-d dimensional quantity. The size of the third dimension will likely be small, however, so it should be safe to generally consider TemplateCoadd to be a small suite of coadds, in which a 2-d image is the result a different sum of or fit to the usual visit-level images (the TemplateWarp dataset, in this case).

Most of the work is done by the DCR-Corrected Template Generation algorithmic component, but its behavior depends on which of the coaddition scenarios is selected from the list at the beginning of Section 5.2):

A,B1,C1 One or more coadd-like images (corresponding to different wavelengths, airmasses, etc.) are created by delegating to the Coaddition algorithmic component to sum the TemplateWarp images with different weights. **A only:** coadded images are then partially decorrelated to constant PSF by delegating to the Coadd Decorrelation algorithmic component.

B2,C2 The template is inferred from the resample visit images using an inverse algorithm that is yet to be developed.

5.2.5 DiffIm

In the DiffIm pipeline, we subtract a warped TemplateCoadd from each CalExp, yielding the DIAExp image, where we detect and characterize DIASources. This is quite similar to Alert Production's Alert Detection pipeline but may not be identical for several reasons. The AP variant must be optimized for low latency, and hence may avoid full-visit processing that is perfectly acceptable in DRP. In addition, the input CalExps will have been better characterized in DRP, which may make some steps taken in AP unimportant or even counterproductive. However, we expect that the algorithmic components utilized in DRP are the same as those used by AP.

The steps taken by DRP DiffIm are:

1. Retrieve the DiffIm template appropriate for the CalExps to be processed (probably handling a full visit at a time), delegating to the Template Retrieval algorithmic component. This selects the appropriate region of sky, and if necessary, collapses a higher-dimensional template dataset to a 2-d image appropriate for the CalExp's airmass.

2. (optional) Correlate the CalExp with its own PSF, delegating to the Convolution Kernel software primitive. This is the "preconvolution" approach to difference imaging, which makes PSF matching easier by performing PSF-correlation for detection first, reducing or eliminating the need for deconvolution. This approach is theoretically quite promising but still needs development.

3. Resample the template to the coordinate system of the CalExp, by delegating to the Image Warping algorithmic component.

4. Match the template's PSF to the CalExp's PSF and subtract them, by delegating to the Image Subtraction algorithmic component.

5. Run Source Detection on the difference image. We correlate the image with its PSF first using the Convolution Kernels software primitive unless this was done prior to subtraction.

6. (optional) Decorrelate the CalExp by delegating to the Difference Image Decorrelation algorithmic component.

7. Run DiffIm Measurement on the difference image to characterize difference sources. If preconvolution is used but decorrelation is not, the difference image cannot be measured using algorithms applied to standard images; alternate algorithms may be developed for some measurements, but perhaps not all.

DiffIm can probably be run entirely independently on each CCD image; this will almost certainly be taken in Alert Production. However, joint processing across a full visit may be more computationally efficient for at least some parts of template retrieval, and PSF-matching may produce better results if a more sophisticated full-visit matching algorithm is developed.

5.2.6 UpdateMasks

UpdateMasks is an optional pipeline that is only run if DIAExp masks are being used to update CalExp masks. As such, it is not run after the last iteration of DiffIm, and is never run if BackgroundMatchAndReject constructs the final CalExp masks.

Like BackgroundMatchAndReject, UpdateMasks compares the histogram of mask values at a particular spatial point to determine which masks correspond to transients (both astrophysical sources and artifacts; we want to reject both from coadds) and which correspond to variable objects. This work is delegated to Coaddition.

5.2.7 WarpRemaining

This pipeline is responsible for the full suite of resampled images used to build coadds in CoaddRemaining, after all CalExp components have been finalized. It produces some combination of the following data products, depending on the scenario(s) described at the beginning of Section 5.2:

LikelihoodWarp CalExp images are correlated with their own PSF, then resampled, via the Convolution Kernels software primitive and the

Image Warping algorithmic component. LikelihoodWarp is computed in all scenarios, but in option **C** it may not need to propogate uncertainty beyond the variance, as the resulting coadd will be used only for detection.

MatchedWarp As in WarpAndPsfMatch, CalExp images are resampled then matched to a common PSF, using Image Warping and PSF Homogenization. MatchWarp is only produced in option **C**.

DirectWarp CalExp images are simply resampled, with no further processing of the PSF, using Image Warping. MatchWarp is only produced in option **C**.

Given that all of these steps involve resampling the image, it would be desirable to do the resampling once up front, and then proceed with the PSF processing. While this is mathematically possible for all of these cases, it would signficantly complicate the PSF correlation step required for building LikelihoodWarps.

5.2.8 CoaddRemaining

In CoaddRemaining, we build the suite of coadds used for deep detection, deblending, and object characterization. This includes the Likelihood, ShortPeriodLikelihood, Deep, BestSeeing, ShortPeriod, and ConstantPSF Coadds.

The algorithms again depend on the scenarios outlined at the beginning of Section 5.2:

A,B All non-template coadds are built from LikelihoodWarps. We start by building ShortPeriodLikelihoodCoadds by simple coaddition of the LikelihoodWarps, using the Image Coaddition algorithmic component. We decorrelate these using the Coadd Decorrelation algorithmic component to produce ShortPeriodCoadds, then sum the ShortPeriodLikelihoodCoadds to produce the full LikelihoodCoadd. The full LikelihoodCoadd is then decorrelated to produce DeepCoadd and ConstantPSFCoadd.

C We generate LikelihoodCoadd and ShortPeriodLikelihoodCoadds using the same approach as above (though the accuracy requirements for uncertainty propagation are eased). ShortPeriodCoadd, DeepCoadd, and BestSeeingCoadd are then built as different combinations of DirectWarp images, again using the Image Coaddition algorithmic component. ConstantPSFCoadds are built by combining MatchedWarps.

These coadds must propagate uncertainty, PSF models (including aperture corrections), and photometric calibration (including wavelength-dependent photometric calibration), in addition to pixel values.

5.3 Coadd Processing

In comparison to the previous two pipeline groups, coadd processing is relatively simple. All pipelines operate on individual patches, and there is no large-scale iteration between pipelines. These pipelines may individually require complex parallelization at a lower level, as they will frequently have memory usage above what can be expected to fit on a single core.

5.3.1 DeepDetect

This pipeline simply runs the Source Detection algorithmic component on combinations of LikelihoodCoadds and ShortPeriodLikelihoodCoadds, then optionally performs additional preliminary characterization on related coadds. These combinations are optimized for detecting objects with different SEDs, and there are a few different scenarios for what combinations we'll produce (which are not mutually exclusive):

- We could simply detect on each per-band LikelihoodCoadds separately.
- We could build a small suite of cross-band LikelihoodCoadds corresponding to simple and artificial but approximately spanning SEDs (flat spectrums, step functions, etc.).
- We could build a large suite of cross-band LikelihoodCoadds corresponding to a library of real star, galaxy, and QSO SEDs.
- We could build a single χ^2 coadd, which is only optimal for objects the color of the sky, but may be close enough to optimal to detect a broad range of SEDs.

Any of these combinations may also be used to combine ShortPeriodLikelihoodCoadds.

We may also convolve the images further or bin them to improve our detection efficiency for extended objects.

Actual detection on these images may be done with a lower threshold than our final target threshold of 5σ, to account for loss of efficiency due using the incorrect SED or morphological filter.

The details of the suite of detection images and morphological filters is a subject requiring further algorithmic research on precursor data (or LSST/ComCam data) at full LSST depths with at least approximately the right filter set.

After detection, CoaddSources may be deblended and characterized by running the Single Frame Deblending, Single Frame Measurement, and Single Frame Classification algorithmic components on DeepCoadd and ShortPeriodCoadd combinations that correspond to the LikelihoodCoadd combinations used for detection. These characterizations (like the rest of the CoaddSource tables) will be discarded after the DeepAssociate pipeline is run, but may be necessary to inform higher-level association algorithms run there. The requirements on characterization processing in this pipeline will be set by the needs of the DeepAssociate pipeline, but we do not expect it to involve significant new code beyond what will be used by the various ImChar pipelines.

The only output of DeepDetect is the suite of CoaddSource tables (one for each detection image) containing Footprints, peaks, and characterizations necessary for association.

5.3.2 DeepAssociate

In DeepAssociate, we perform a sophisticated spatial match of all CoaddSources and DIASources, generating tables of DIAObjects, Object candidates, and SSObject detections. We do *not* include the Source table in this merge, as virtually all Sources correspond to astrophysical objects better detected in DeepDetect or DiffIm.

The baseline plan for association is to first associate DIASources into DIAObjects using the same approach used in Alert Production (i.e. the DIAObject Generation algorithmic component), then associate DIAObjects with the multiple CoaddSource tables (using the Object Generation algorithmic component). DIASources not associated into DIAObjects will be considered candidates for merging SSObjects, which will happen in the MOPS pipeline.

These association steps must be considerably more sophisticated than simple spatial matching; they must utilize the limited flux and classification information available from detection to decide whether to merge sources detected in different contexts. This will require astrophysical models to be included in the matching algorithms at some level; for instance:

- We must be able to associate the multiple detections that correspond to high proper-motion stars into a single Object.
- We must not associate supernovae with their host galaxies, despite the fact that their positions may be essentially the same.

To meet these goals (as well as similar ones which still need to be specifed), DeepAssociate will have to generate *multiple* hypotheses for some blend families. Some of these conflicting hypotheses will be rejected by the DeepDeblend, while others may be present in the final Object catalog (flags will be used to indicate different interpretations and our most likely interpretation). This is a generalization of the simple parent/child hierarchy used to describe different blend hypotheses in the SDSS database (see Section 2.3).

It is possible that associations could be improved by doing both merge steps simultaneously (under the hypothesis that CoaddSource presence or absence could be used to improve DIASource association). This is considered a fallback option if the two-stage association procedure described above cannot be made to work adequately.

The output of the DeepAssociate pipeline is the first version of the Object table, containing a superset of all Objects that will be characterized in later pipelines.

5.3.3 DeepDeblend

This pipeline simply delegates to the Multi-Coadd Deblending algorithmic component to deblend all Objects in a particular patch, utilizing all non-likelihood coadds of that patch. This yields HeavyFootprints containing consistent deblended pixels for every object in every (non-likelihood) coadd, while rejecting as many deblend hypotheses as possible to reduce the number of hypotheses that must be subsequently measured.

While the pipeline-level code and data flow is simple, the algorithmic component is not. Not only must deblending deal with arbirarily complex superpositions of objects with unknown morphologies, it must do so consistently across bands and the epoch ranges (with different PSFs) and ensure proper handling of Objects spawned by DIASources that may not even appear in coadds. It must also parallelize this work efficiently over multiple cores; in order to fit patch-level images for all coadds in memory, the processing of at least the largest individual blend families must themselves be parallelized. This may be done by splitting the largest blend families into smaller groups that can be processed in parallel with only a small amount of serial iteration; it may also be done by using low-level multithreading over pixels.

The output of the DeepDeblend pipeline an update to the Object table, which adds columns (mostly flags) and removes some rows from the table produced by DeepAssociate.

5.3.4 MeasureCoadds

The MeasureCoadds pipeline delegates to the Multi-Coadd Measurement algorithmic component to jointly measure all Objects on all coadds in a patch.

Like DeepDeblend, this pipeline is itself quite simple, but it delegates to a complex algorithmic component (but a simpler one than Multi-Coadd Deblending). There are three classes of open questions in how multi-coadd measurement will proceed:

- What parameters will be fit jointly across bands, and which will be fit independently? The measurement framework for multi-coadd measurement is designed to support joint fitting, but it is likely that some algorithms will simply be Single Frame Measurement or Forced Measurement plugins that are simply run independently on the DeepCoadd and/or ConstantPSFCoadd in each band. Making these decisions will require experimentation on deep precursor and simulated data.

- How will we measure blended objects? Coadd measurement will at least begin by using the HeavyFootprints produced by DeepDeblend to use the Neighbor Noise Replacement approach, but we may then use Simultaneous Fitting to generate improved warm-start parameters for MultiFit or to build models we can use as PSF-deconvolved templates to enable the Deblend Template Projection approach in MultiFit and/or ForcedPhotometry.

- How will we parallelize? As with DeepDeblend, keeping the full suite of coadds in memory will require processing at least some blend families using many cores. For algorithms that don't require joint fitting across different coadds, this could be done by measuring each coadd independently in parallel, but the most expensive algorithms (e.g. galaxy model fitting) are likely to be the ones where we'll want to fit jointly across bands.

The output of the MeasureCoadds pipeline is an update to the Object table, which adds columns containing measured quantities.

5.4 Overlap Resolution

The two overlap resolution pipelines are together responsible for finalizing the definitions of Objects by merging redundant processing done in tract and patch overlap regions. In most cases, object definitions in the overlap region will be the same, making the problem trivial, and even when the definitions are different we can frequently resolve the problem using purely geometrical arguments. However, some difficult cases will remain, mostly relating to blend families that are defined differently on either side.

We currently assume that overlap resolution actually drops Object rows when it merges them; this will avoid redundant processing in the performance critical MultiFit pipeline. A slower but perhaps safer alternative would be to simply flag redundant Objects. This would also allow tract overlap resolution to be moved after the MultiFit and ForcedPhotometry pipelines, which would simplify large-scale parallelization and data flow by moving the first operation requiring more than one tract (ResolveTractOverlaps until after all image processing is complete.

5.4.1 ResolvePatchOverlaps

In patch overlap resolution, all contributing patches to an area (there can be between one and four) share the same pixel grid, and we furthermore expect that they will have the same coadd pixel values. This should ensure that any above-threshold pixel in one patch is also above threshold in all others, which in turn should guarantee that patches agree on the extent of each blend family (as defined by the parent Footprint).

A common pixel grid also allows us to define the overlap areas as exact rectangular regions; we consider each patch to have an inner region (which directly abuts the inner regions of neighboring patches) and an outer region (which extends into the inner regions of neighboring patches). If we consider the case of two overlapping patches, blend families in those patches can fall into five different categories:

- If the family falls strictly within one patch's inner region, it is assigned to that patch (and the other patch's version of the family is dropped).
- If the family crosses the boundary between patch inner regions...
  - ...but is strictly within both patches' outer regions, it is assigned to the patch whose inner region includes more of the family's footprint area.
  - ...but is strictly within only one patch's outer region, it is assigned to that patch.
  - ...and is not strictly within either patch's outer region, the two families must be merged at an Object-by-Object level. The algorithm used for this procedure is yet to be developed, but will be implemented by the Blended Overlap Resolution algorithmic component.

Overlap regions with more than two patches contributing have more possibilities, but are qualitatively no different.

[**TODO:**
Add figure explaining inner and outer patch regions.]

If pixel values in patch overlap regions cannot be guaranteed to be identical, patch overlap resolution becomes significantly harder (but no harder than tract overlap resolution), because adjacent patches may disagree on the above categories to which a family belongs.

Patch overlap resolution can be run independently on every distinct overlap region that has a different set of patches contributing to it; in the limit of many patches per tract, there are three times as many overlap regions as patches (each patch has four overlap regions shared by two patches, and four overlap regions each shared by four patches).

5.4.2 ResolveTractOverlaps

Tract overlap resolution operates under the same principles as patch overlap resolution, but the fact that different tracts have different coordinate systems and subtly different pixel values makes the problem significantly more complex.

While we do not attempt to define inner and outer regions for tracts, we can still define discrete overlap regions in which the set of contributing tracts is constant (though these regions must now be defined using spherical geometry). Because tracts may differ on the extent and membership of blend families, it will be useful here to define the concept of a "blend chain": within an overlap region a families blend chain is the recursive union of all families it overlaps with in any tract that contributes to that overlap region see Figure TODO. A blend chain is thus the maximal cross-tract definition of the extent of a blend family, and hence we can use it to categorize blends in tract overlaps:

- If a blend chain is strictly contained by only one tract, all families within that chain are assigned to that tract.
- If a blend chain is strictly contained by more than one tract, all families within that chain are assigned to the tract whose center is closest to the centroid of the blend chain.
- If a blend chain is not strictly contained by any tract, all families in the chain must be merged at an Object-by-Object level. This is done by the Blended Overlap Resolution algorithmic component, after first transforming all measurements to a new coordinate system defined to minimize distortion due to projection (such as a tangent projection at the blend chain's centroid).

ResolveTractOverlaps is the first pipeline in Data Release Production to require access to processed results from more than one tract.

[**TODO:**
Add figure explaining blend chains.]

5.5 Multi-Epoch Object Characterization

The highest quality measurements for the vast majority of LSST objects will be performed by the MultiFit and ForcedPhotometry pipelines. These measurements include stellar proper motions and parallax, galaxy shapes and fluxes, and light curves for all objects. These supersede many (but not all) measurements previously made on coadds and difference images by using deep, multi-epoch information to constrain models while fitting directly to the original CalExp (or DIAExp) images.

The difference between the two pipelines is their parallelization axis: an instance of the MultiFit pipeline processes a single Object family at a time, utilizing all of the CalExps that overlap that family as input, while ForcedPhotometry processes one CalExp or DIAExp at a time, iterating over all Objects within its bounding box. Together these three pipelines must perform three roles:

- Fit moving point source and galaxy models to all Objects, adding new columns or updating existing columns in the Object table. This requires access to all images simultaneously, so it must be done in MultiFit.

- Fit fixed-position point source models for each object (using the MultiFit-derived positions) to each DIAExp image separately, populating the ForcedSource table. This *differential forced photometry* could concievably be done in MultiFit, but will probably be more efficient to do in ForcedPhotometry.

- Fit fixed-position point source models for each object to each CalExp image separately, also populating the ForcedSource table. This *direct forced photometry* can easily be done in either pipeline, but doing it MultiFit should give us more options for dealing with blending, and it may decrease I/O costs as well.

5.5.1 MultiFit

MultiFit is the single most computationally demanding pipeline in Data Release Production, and its data flow is essentially orthogonal to that of all previous pipelines. Instead of processing flow based on data products, each MultiFit job is an Object family covering many distinct images, and hence efficient I/O will require the orchestration layer to process these jobs in an order that minimizes the number of times each image is loaded.

From the Science Pipelines side, MultiFit is implemented as two routines, mediated by the orchestration later:

- The MultiFit "launcher" processes the Object table and defines family-level MultiFit jobs, including the region of sky required and the corresponding data IDs and pixel-area regions (unless the latter two are more efficiently derived from the sky area by the orchestration layer).

- The MultiFit "fitter" processes a single Object family, accepting all required image data from the orchestration layer and returning an Object record (and possibly a table of related ForcedSources). This is the Multi-Epoch Measurement algorithmic component.

This simple picture is complicated by the presence of extremely large blend families, however. Some blend families may be large enough that a single MultiFit job could require more memory than is available on a full node (or require more cores on a node than can be utilized by lower-level parallelization). We see two possibilities for addressing this problem:

- The fitter could utilize cross-node communication to extend jobs over more nodes. The most obvious approach would give each node full responsibility for any processing a on group of full CalExps it holds in memory, as well as responsibility for "directing" a number of MultiFit jobs. These jobs would delegate pixel processing on CalExps to the nodes responsible for them (this constitutes the bulk of the processing). This would require low-latency but low-bandwidth communication; the summary information passed between the directing jobs and the CalExp-level processing jobs is much smaller than the actual CalExps or even the portion of a CalExp used by a particular fitting job, but this communication happens within a relatively tight loop (though not the innermost loop). This approach will also require structuring the algorithmic code to abstract out communication, and may require an alternate mode to run small jobs for testing.

- The launcher could define a graph of sub-family jobs that correspond to an iterative divide-and-conquer approach to large families. This approach will require more flexibilty in the algorithmic code to handle more combinations of fixed and free parameters (to deal with neighboring objects on the edges of the images being considered), more tuning and experimentation, and more sophisticated launcher code. Fitting individual large objects in this scenario could also require binning images in the orchestration or data access layer.

It is unclear which of these approaches will be more computationally expensive. The first option may reduce I/O or total network usage at the expense sensitivity to network latency. The second option may require redundant processing by forcing iterative fitting, but that sort of iterative fitting may lead to faster convergence and hence be used even in the first option.

If direct forced photometry is performed in MultiFit, moving-point source models will simply be re-fit with per-epoch amplitudes allowed to vary independently and all other parameters held fixed. The same approach could be used to perform differential forced photometry, but this would require also passing DIAExp pixel data to MultiFit.

Significant uncertainty also remains in how MultiFit will handle blending even in small families, but this decision will not have larger-scale processing impacts, and will be discussed further in Section 8.6.3.

5.5.2 ForcedPhotometry

In ForcedPhotometry, we simply measure point-source and possibly aperture photometry (the baseline is point source photometry, but aperture photometry should be implemented for diagnostic use and as a fallback) on individual CalExp or DIAExp images, using positions from the Object table.

Aside from querying the Object table for the list of Objects overlapping the image, all work is delegated to the Forced Measurement algorithmic component. The only algorithmic challenge is how to deal with blending. If only differential forced photometry is performed in this pipeline, it may be appropriate to simply fit all Objects within each family simultaneously with point source models. The other alterative is to project templates from MultiFit or possibly MeasureCoadds and replace neighbors with noise (as described in Sections 8.6.3.1 and 8.6.3.2).

5.6 Postprocessing

The pipelines in the postprocessing group may be run after nearly all image processing is complete, and with the possible exception of MakeSelectionMaps, include no image processing themselves. While we do not expect that these pipelines will require significant new algorithm development, they include some of the least well-defined aspects of Data Release Production; many of these pipelines are essentially placeholders for work that may ultimately be split out into multiple new pipelines or included in existing ones. Unlike the rest of DRP, a more detailed design here is blocked more by the lack of clear requirements and policies than a need for algorithmic research.

5.6.1 MOPS

MOPS plays essentially the same role in DRP that it plays in AP: it builds the SSObject (Solar System Object) table from DIASources that have not already been associated with DIAObjects. We will attempt to make its implementation as similar as possible to the AP DayMOPS pipeline, but the fact that DRP will run MOPS on all DIASources in the survey at once (instead of incrementally) make this impossible in details. The steps in MOPS are (with some iteration):

- Delegate to the Make Tracklets algorithmic component to combine unassociated DIASources into *tracklets*.
- Delegate to the Attribution and Precovery algorithmic component to predict the positions of known solar system objects and associate them with tracklets. The definition of a "known" solar system object cleraly depends on the input catalog; this may be an external catalog or a snapshot of the Level 1 SSObject table.
- Delegate to the Orbit Fitting algorithmic component to merge unassociated tracklets into tracks and fit orbits for SSObjects where possible.

The choice of initial catalog largely depends on the false-object rate in the Level 1 SSObject; if the only improvements in data release production are slightly improved orbit and/or new SSObjects, using the Level 1 SSObject table could dramatically speed up processing – but it may also remove the possibility of removing nonexistent objects.

Note:
TODO Reference appropriate subsection of AP section.

MOPS represents a full-survey sequence point in the production, but we expect that it will be a relatively easy one to implement, because it operates on relatively small inputs (unassociated DIASources) and produces a single new table (SSObject) as its only major output (though IDs linking DIASources and SSObjects must also be stored in either DIASource or a join table). This should mean that it can be run after most other data products have already been ingested, while requiring little temporary storage as the rest of the processing proceeds tract-by-tract.

5.6.2 ApplyCalibrations

The processing described in the previous sections produces four tables that ultimately must be ingested into the public database: Source, DIASource, Object, DIAObject, SSObject, and ForcedSource. The quantities produced by previous pipelines are in raw units, however (e.g. fluxes are in counts, positions in pixels). These must be transformed into calibrated units via our astrometric and photometric solutions, a process we delegate to the Raw Measurement Calibration algorithmic component.

This is the primary place where the wavelength-dependent photometric calibrations generated by the Calibration Product Pipelines are applied. This will require inferring an SED for every object (or source) from its measured colors. The families of SEDs and the choice of color measurements used are subjects for future algorithmic research, but it should be possible to resolve these questions with relatively little effort. The inferred SED must be recorded, allowing science users to recalibrate as desired with their own preferred SED. One possible complication here is that PSF models are also wavelength dependent, and the SED for this purpose must be inferred much earlier in the processing. Because it is highly desirable that the SEDs used for PSF-dependent measurement be the same as those used for photometric calibration, we may need to either infer SEDs early in the processing from preliminary color measurements or estimate the response of measurements to changes in PSF-evaluation SED so it can be approximately updated later.

Note:
TODO Reference appropriate subsection of CPP section.

It is currently unclear when and where calibrations will be applied; there are several options:

- We could apply calibrations to tables before ingesting them into the public database; this would logically create new calibrated versions of each table data product.
- We could apply calibrations to tables *as* we ingest them into the final database.
- We could ingest tables into the temporary tables in the database and apply the calibrations within the database.

Regardless of which option is chosen for each public table, the Raw Measurement Calibration algorithmic component will probably need to support operation both outside the database on in-memory table data and within the database (via, e.g. user-defined functions). The former will be needed to apply calibrations to intermediate data products for diagnostic purposes, while the latter will be needed to allow Level 3 users to recalibrate objects according to their own assumed SEDs.

5.6.3 MakeSelectionMaps

The MakeSelectionMaps is responsible for producing multi-scale maps that describe LSST's depth and efficiency at detecting different classes of object. The details of what metrics will be mapped, the format and scale of the maps (e.g. hierarchical pixelizations vs polygons), and the way the metrics will be computed are all unknown.

The approach must be extensible at Level 3: science users will need to build additional maps that can be utilized as efficiently by large collaborations as DM-produced maps. This will ease the pressure on DM to provide a large suite of maps, but the details of what DM will provide still needs to be clarified to the community.

One potential major risk here is that the most common way to determine accurate depth and selection metrics is to add fake sources to the data and reprocess, and this can require reprocessing each unit of order 100 times. Because the reprocessing does not need to include all processing steps (assuming the skipped steps can be adequately simulated), this should not automatically be ruled out – if the pipelines that must be repeated (e.g. DeepDetect) are significantly faster than skipped steps (such as MultiFit), the overall impact on processing could still be negligible. Regardless, the role of DM in this sort of characterization also needs to be clarified to the community.

[**Note:**
TODO Cite Balrog paper (Suchyta and Huff 2016)]

5.6.4 Classification

In its simplest realization, this pipeline computes variability summary statistics and probabilistic and/or discrete classification of each Object as a star or galaxy; this may be extended to include other categories (e.g. QSO, supernova).

Variabiliy summary statistics are delegated to the Variability Classification algorithmic component.

Type classification is delegated to the Object Classification algorithmic component. This may utilize any combination of morphological, color, and variability/motion information, and may use spatial information such as galactic latitude as a Bayesian prior. Classifications based on only morphology will also be available.

Both variability and type classification may require "training" on the full Object and ForcedSource tables and/or similar tables derived from special program data. This represents a potential full-survey sequence point for the production.

The possible need for full-dataset processing suggests that it may be more efficient to perform classification in the final public database, in order to utilize it for large-scale aggregation calculations. This may not be feasible if final public database tables are write-once, as classification may require both read and write operations on Object. Putting the classification sequence point before ingest would then require keeping the Object data products for all tracts on disk before ingesting any of them.

Further specification of special programs (and DM plans for processing them) and algorithmic research are needed to determine whether classification actually will require a full-survey sequence point in the production.

5.6.5 GatherContributed

This pipeline is just a placeholder for any DM work associated with gathering, building, and/or validating major community-contributed data products.

In addition to data products produced by DM, a data release production also includes official products (essentially additional Object table columns) produced by the community. These include photometric redshifts and dust reddening maps. While DM's mandate does not extend to developing algorithms or code for these quantities, its responsibilities may include validation and running user code at scale. The parties responsible for producing these datasets and their relationship to DM needs to be better defined in terms of policy before a system for including community-contributed data products in a data release can be designed.

5.7 UNSTRUCTURED OPEN QUESTIONS / WORK NOT CAPTURED

- Where does the initial reference catalog at the start of the DRP come from? This could require special observations in commissioning or before the start of the survey, as well as addition algorithms and software. If DRP always uses a reference catalog for star selection in ImChar, we need to actually do the classification for that at some point.

- How do we test all of the wavelength-dependent photometric calibration and PSF stuff on precursor data? Are we going to characterize DECam well enough to just use it directly, or do we need to mock things up or rely more on JointCal?

- Input selection for DRP. How do we decide what to throw away? Is this a SQuaRE job? Do we ever throw images away because the pipeline can't process them? Do we report these back to the scheduler, or just hope it all averages out?

- Do we need to resolve tract overlaps in Source and DIASource as well?

6 Services for Data Quality Analysis (SDQA)

6.1 Key Requirements

SQDA is a set of loosely coupled services intended to service LSST's quality assessement needs through all phases of Construction, Commissioning and Operations. Consumers of these services may include developers, facility staff, DAC (eg. L3) users, and the general LSST science user community. Use of these services is intended for routine characterisation, fault detection and fault diagnosis.

- SDQA shall provide sevices for science data quality analysis of Level 1, 2, and Calibration Processing pipelines.
- SDQA shall provide services to support software development in Construction, Commissioning and Operations.
- SDQA shall provide for the visualization, analysis and monitoring capabilities needed for common science quality data analysis usecases. Its inputs may be gathered from SDQA itself, the production pipelines, engineering data sources and non-LSST data sources.
- SDQA shall have the flexibility to support execution of ad-hoc (user-driven) tests and analyses of ad-hoc data within a standard framework.
- SDQA shall support usecases involving interactive "drill-down" of QA data exposed through its visualisation interfaces.
- SDQA shall allow for notifications to be issued when monitoring quantities that exceed their permissible thresholds and/or have degraded over historical values.
- SDQA shall be able to collect and harvest the outputs and logs of execution of the production pipelines, and extract and expose metrics from these logs.
- SQDA shall make provision to store outputs that are not stored through other LSST data access services.
- SDQA should be deployable as high-reliability scalable services for production as well as allow for core data assessment functionality to be executed on a developer's local machine.

- SDQA shall be architected in a manner that would enable it to be deployable on standard cloud architectures outside of the LSST facilities.

6.2 Key Tasks for Each Level of QA

SDQA system will provide a framework that is capable of monitoring QA information at four different stages of capabiliti and maturity:

- QA Level 0 - Testing and Validation of the DM sub-system in pre-commissioning
- QA Level 1 - Real-time data quality and system assesment during commissioning + operations (also, forensics)
- QA Level 2 - Quality assessment of Data Releases (also, forensics)
- QA Level 3 - Ability for the community to evaluate the data quality of their own analyses. These should available as well-documented and reployable versions of core QA Level 0–2 services.

6.2.1 QA0

Test the DM software during pre-commissioning as well as test software improvements during commissioning and operations, quantifying the software performance against known and/or expected outputs.

Validating the software and its performance on stamdardized data.

("Make me a three-color diagram, compute the width of point sources in the blue part of the locus")

("I have 20 visits all over the sky, I want to match up the results")

The main components:

1. CI system that compiles code
2. Test execution harness – that runs test on a regular cadence eg nightly/weekly/monthly
3. A suite of validation metrics codes – some has to come from Science Pipelines, but KPMs are delivered by SQuaRE
4. Capability to instrument the production pipelines for computational performance metrics

5. Library of "instrumentations"
6. Interface to data products and QA metrics (including visualization)
    (a) Tabular query result interface
    (b) Visualizer for images
    (c) Visualiser for scalar data (eg. Plotter)
    (d) SuperTask execution on selected data
7. Curated datasets to use in tests
8. Capability for interactive analysis of QA outputs (drilldown into existing tests, ad hoc tests, ad hoc afterburners) – some has to come from Science Pipelines or SUIT but SQuaRE provides examples [move to Shared Software Components section]
    (a) Tools that perform computations
    (b) Tools that perform visualization (using Butler if astronomical, maybe direct database if not)
9. Connection from analysis toolkit to validation metrics (attach common interactive plots to validation metrics)
10. QA database including ingestion
11. Notification system for threshold crossings

Prototypes for all of these exist except "Toolkit for analysis of QA outputs" and "Connection from analysis toolkit to validation metrics"

"Toolkit for analysis of QA outputs" will take more resources than the others listed above, but some may be already scheduled in other teams

6.2.2 QA1

Data quality in real time during commissioning and operations. Analyzes the data stream in near-real time, information about observing conditions (sky brightness, transparency, seeing, magnitude limit) as well as characterize subtle deterioration in system performance.

Validating the operational system.

Main components reused from Level QA0:

1. Library of validation metrics codes
2. Instrumentation capability for computational performance metrics
3. Library of "instrumentations"
4. Interface to results (including visualization)
5. Curated datasets to use in tests
6. Capability for interactive analysis of failures (drilldown into existing tests, ad hoc tests, ad hoc afterburners) – some has to come from Science Pipelines or SUIT but SQuaRE provides examples
7. Connection from analysis toolkit to validation metrics
8. QA data access service (including ingesting it as well as querying it)

Main components reused from Level QA0: All.
Additional components required for Level QA1 services:

1. Harness for analyzing alert contents (and perhaps format)
2. Faster metrics codes to meet overall 60 second performance requirement for alert publication (but not necessarily for all QA processing, which must meet only throughput requirements)
3. Additional metrics/instrumentation codes (that must not disturb production system, including its performance, when dynamically inserted)
4. Output interface to "comfort" display (aggregation, trending, etc.)
5. Output interface to automated systems (drop alerts, reschedule field, etc.)
6. Correlator between telemetry streams and metrics
7. Input interface from sources of data not already present in Prompt Processing system
8. Fake source injection and analysis
9. Metrics codes specific for calibration/engineering/special- purpose images

6.2.3 QA2

Assess the quality of data releases (including the co-added image data products) performing quality assessment for astrometric and photometric calibration and derived products, looking for problems with the image processing pipelines and systematic problems with the instrument. Validating the Data Release products. All components from QA0

New main components:

1. DRP-specific dataset
2. Release data product editing tools (including provenance tracking)
3. Output interface to workflow system based on QA results and provenance
4. Provenance analysis tools
5. Output interface to Science Pipelines, including from QA database
6. Comparison tools for overlap areas due to satellite processing
7. Metrics/products for science users to understand quality of science data products (depth mask/selection function, etc.)
8. Characterization report for Data Release

6.2.4 QA3

Data quality based on science analysis performed by the LSST Science Collaborations and the community. Level 0-2 visualization and data exploration tools will be made available to the community. Make all results from the above available. Make all of the above components available to some part of the community (could be just affiliated data centers or could be individual scientists) as a supported product. Ingest external science analysis data as Level 3 data products; ingest useful external science analysis tools.

6.2.5 Prototype Implementation of PipeQA

The pipeQA prototype is a useful reference for exploring ideas and we mention it here to capture this prototype work.

A prototype implementation of the SDQA was implemented in LSST Final Design Phase. The existing prototype was tested with image simulation inputs, as well as real data (SDSS Stripe 82).

The prototype used a set of statically and dynamically generated pages (written in php) to display the results of data production runs. While proving invaluable for data analysis, the prototype design was found it to be difficult to extend with new analyst-developed tests.

The prototype code is available in the `https://github.com/lsst/testing_displayQA` git repository.

7 Science User Interface and Toolkit

7.1 Science Pipeline Toolkit (WBS 02C.01.02.03)

7.1.1 Key Requirements

The Science Pipeline Toolkit shall provide the software components, services, and documentation required to construct Level 3 science pipelines out of components built for Level 1 and 2 pipelines. These pipelines shall be executable on LSST computing resources or elsewhere.

7.1.2 Baseline Design

The baseline design assumes that Level 3 pipelines will use the same `Tasks` infrastructure (see the Data Management Middleware Design document; DMMD) as Level 1 and 2 pipelines[7]. Therefore, Level 3 pipelines will largely be automatically constructible as a byproduct of the overall design.

The additional features unique to Level 3 involve the services to upload/-download data to/from the LSST Data Access Center. The baseline for these is to build them on community standards (VOSpace).

7.1.3 Constituent Use Cases and Diagrams

Configure Pipeline Execution; Execute Pipeline; Incorporate User Code into Pipeline; Monitor Pipeline Execution; Science Pipeline Toolkit; Select Data to be Processed; Select Data to be Stored;

7.1.4 Prototype Implementation

While no explicit prototype implementation exists at this time, the majority of LSST pipeline prototypes have successfully been designed in modular and portable fashion. This has allowed a diverse set of users to customize and run the pipelines on platforms ranging from OS X laptops, to 10,000+ core clusters (e.g., BlueWaters), and to implement plugin algorithms (e.g., Kron photometry).

---

[7] Another way of looking at this is that, functionally, there will be no fundamental difference between Level 2 and 3 pipelines, except for the level of privileges and access to software or hardware resources.

8 Algorithmic Components

8.1 Instrument Signature Removal

AUTHOR: Merlin

- Mask defects and saturation
- Assembly
- Overscan
- Linearity
- Crosstalk
- Full frame corrections: Dark, Flats (includes fringing)
- Pixel level corrections: Brighter fatter, static pixel size effects
- Interpolation of defects and saturation
- CR rejection
- Generate snap difference
- Snap combination

8.1.1 AP: just skip some steps?

AUTHOR: Simon

- Indicate steps to be done by camera
- call out other steps that are omitted/modified relative to the DRP version

8.1.2 DRP: do all the steps

AUTHOR: Merlin

8.2 Artifact Detection

8.2.1 Single-Exposure Morphology

AUTHOR: Simon

- Find CRs via morphology.
- Find some optical ghosts (etc?) from bright star catalog and optics predictions.
- Needs to work without PSF (amybe using placeholder PSF), but also make use of PSF if available.

8.2.2 Single-Exposure Aggregation

AUTHOR: Simon

- Find satellites via Hough transform.

8.2.3 Snap Subtraction

AUTHOR: Simon

- All of the above, but improve by looking at both snaps.

8.2.4 Warped Image Comparison

AUTHOR: Jim

- Find more optical artifacts by looking at differences between warped images (this is run during background matching).
- Find transient astronomical sources we don't want to include in coadds.

8.3 Artifact Interpolation

AUTHOR: Jim

- Set mask planes for all artifacts.
- Eliminate small artifacts by interpolating them.
- Uses PSF model as interpolant.

8.4 Source Detection

AUTHOR: Jim

- Detect above-threshold regions and peaks in direct or difference images.
- Needs to work on preconvolved and unconvolved images.
- May need multi-pass variants: detect bright objects first, then faint; detect with approximate PSF, then improved.
- Need to work on wavefront sensors (with out-of-focus PSFs)

8.5 Deblending

AUTHOR: Jim

For templates, try:

- symmetry ansatz with additional regularization
- simultaenous fit of galaxy models
- spline-based models with regularization?
- (multi-coadd only) optimize color uniformity

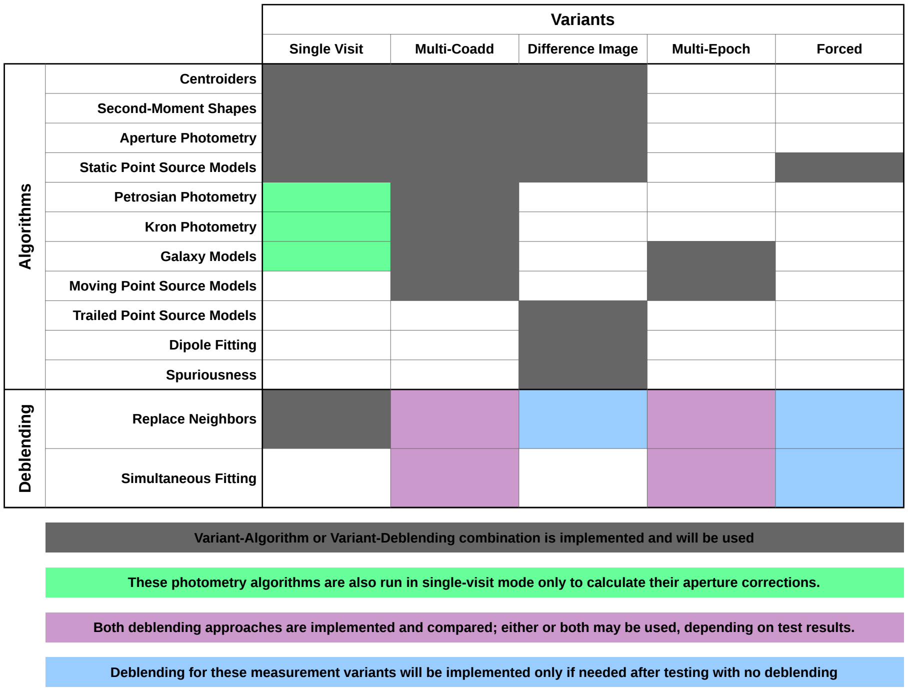
Will be especially challenging in crowded fields, but it needs to work in that regime as well.

8.5.1 Single Frame Deblending

- Generate HeavyFootprint deblends using only a single image.
- May need to be able to work with approximate/guess PSF, even in crowded fields, if we need to deblend before PSF estimation in DRP.
- May need to work on wavefront sensors (with out-of-focus PSFs)

8.5.2 Multi-Coadd Deblending

- Generate consistent HeavyFootprint deblends from coadds over multiple bands and possibly epoch ranges.



		Variants				
		Single Visit	Multi-Coadd	Difference Image	Multi-Epoch	Forced
Algorithms	Centroiders	⬛	⬛	⬛		
	Second-Moment Shapes	⬛	⬛	⬛		
	Aperture Photometry	⬛	⬛	⬛		
	Static Point Source Models	⬛	⬛	⬛		⬛
	Petrosian Photometry	🟩	⬛			
	Kron Photometry	🟩	⬛			
	Galaxy Models	🟩	⬛		⬛	
	Moving Point Source Models		⬛		⬛	
	Trailed Point Source Models			⬛		
	Dipole Fitting			⬛		
	Spuriousness			⬛		
Deblending	Replace Neighbors	⬛	🟪	🟦	🟪	🟦
	Simultaneous Fitting		🟪		🟪	🟦

⬛ Variant-Algorithm or Variant-Deblending combination is implemented and will be used

🟩 These photometry algorithms are also run in single-visit mode only to calculate their aperture corrections.

🟪 Both deblending approaches are implemented and compared; either or both may be used, depending on test results.

🟦 Deblending for these measurement variants will be implemented only if needed after testing with no deblending

Figure 6: Matrix showing combinations of measurement variants, algorithms, and deblending approaches that will be implemented.

8.6 Measurement

AUTHOR: Jim

8.6.1 Variants

Measurement is run in several contexts, but always consists of running an ordered list of algorithm plugins on either individual objects or families thereof. Each context corresponds to different variant of the measurement driver code, and has a different set of plugin algorithms and approaches to measuring blended objects.

8.6.1.1 Single Frame Measurement: Measure a direct single-visit CCD image, assuming deblend information already exists and can be used to replace neighbors with noise (see 8.6.3.2).

Single Frame Measurement is run in both AP's Single Frame Processing pipeline) and DRP's BootstrapImChar, RefineImChar, and FinalImChar. It must be capable of running on wavefront sensor images, though this may require different plugin algorithms.

The driver for Single Frame Measurement is passed an input/output SourceCatalog and an Exposure to measure. Plugins take an input/output SourceRecord and an Exposure containing only the object to be measured.

8.6.1.2 Multi-Coadd Measurement: Simultaneously measure a suite of coadds representing different bandpasses, epoch ranges, and flavors. This is run only in DRP's MeasureCoadds pipeline.

The driver for Multi-Coadd Measurement is passed an input/output ObjectCatalog and a dict of Exposures to be measured. Plugins take an input/output ObjectRecord and a dict of Exposures, each containing only the object to be measured. Some plugins will also support simultanous measurement of multiple objects, which requires they be provided the subset of the ObjectCatalog to be measured and a dict of Exposures containing just those objects.

8.6.1.3 Difference Image Measurement: Measure a difference image, potentially using the associated direct image as well. Difference image measurement is run in AP's Alert Detection pipeline and DRP's DiffIm pipeline.

The signatures of difference image measurement's drivers and algorithms are at least somewhat TBD; they will take at least a difference image Exposures and a SourceCatalog/SourceRecord, but some plugins such as dipole measurement may require access to a direct image as well. Because difference imaging dramatically reduces blending, difference image measurement may require any approach to blended measurement (though any use of the associated direct image would require deblending).

If preconvolution is used to construct difference images, but they are not subsequently decorrelated, the algorithms run in difference image measurement cannot be implemented in the same way as those run in other measurement variants, and algorithms that cannot be expressed as a PSF-convolved model fit (such as second-moment shapes and all aperture fluxes) either cannot be implemented or require local decorrelation.

8.6.1.4 Multi-Epoch Measurement: Measure multiple direct images simultaneously by fitting the same WCS-transformed, PSF-convolved model to them. Blended objects in Multi-Epoch Measurement will be handled by *at least* fitting them simutaneously (8.6.3.3), which may in turn require hybrid galaxy/star models (8.6.3.4). These models may then be used as templates for deblending and replace-with-noise (8.6.3.2) measurement if this improves the results.

Because the memory and I/O requirements for multi-epoch measurement of a single object or blend family are substantial, we will not provide a driver that accepts an ObjectCatalog and measures all objects within it; instead, the peline will submit individual family-level jobs directly to the orchestration layer. The multi-epoch measurement driver will thus just operate on one blend family at a time, and manage blending while executing its plugin algorithms.

Multi-epoch measurement for DRP only includes two plugin algorithms, so it is tempting to simply hard-code these into the driver itself, but this driver will also need to support new plugins in Level 3.

Multi-epoch measurement will also be responsible for actually performing forced photometry on direct images, which it can do by holding non-amplitude parameters for moving point-source models fixed and adding a new amplitude parameter for each observation.

8.6.1.5 Forced Measurement: Measure photometry on an image using positions and shapes from an existing catalog.

In the baseline plan, we assume that forced measurement will only be run on difference images; while forced photometry on direct images will also be performed in DRP, this will be done by multi-epoch measurement.

Because difference imaging reducing blending substantially, forced measurement may not require any special handling of blends. If it does, simultaneous fitting (with point-source models) should be sufficient.

The driver for Forced Measurement is passed an input/output SourceCatalog, an additional input ReferenceCatalog, and an Exposure to measure. Plugins take an input/output SourceRecord, an input ReferenceRecord and an Exposure. If simultaneous fitting is needed to measure blends, plugins will instead receive subsets of the catalogs passed to the driver instead of individual records.

Forced measurement is used by the DRP ForcedPhotometry pipeline and numerous pipelines in AP.

[**TODO:**
Add references to specific AP pipelines that will use forced measurement.]

8.6.2 Algorithms

8.6.2.1 Centroids

- should be equivalent to PSF model fit for stars
- use larger weight function (TBD) for extended objects
- need variant that doesn't require a PSF model (or can work with a poor guess) to run before PSF estimation.
- need to have a version (possibly the main version) that works on wavefront sensors

8.6.2.2 Pixel Flag Aggregation

- Compute summary statistics of masked pixels in the neighborhood of the source/object.

8.6.2.3 Second-Moment Shapes

- probably adaptive elliptical Gaussian weights, with fall back to unweightd, PSF-weighted, or some fixed Gaussian
- add regularization for unresolved objects - avoid crazy ellipticities for objects much smaller than PSF
- Should also compute moments of PSF model.
- Need to have a version (possibly the main version) that works on wavefront sensors to characterize the donut-like out-of-focus sources.

8.6.2.4 Aperture Photometry

- Aperture fluxes are computed by summing the total flux within an elliptical region defined on the image.
- Aperture fluxes are computed at a series of logarithmically spaced aperture sizes. Per the DPDD, the total number of apertures will vary depending on the size of the source.
- When computing fluxes for small apertures—for configurable values of "small"–we use sinc interpolation [5]. For large apertures, we use a naive summation of pixel values.
- May need to change ellipticity as a function of aperture radius.
- If run before PSF estimation, will need a variant that does not rely on the PSF model to choose aperture size/ellipticity.

8.6.2.5 Static Point Source Models

- Fit PSF model for flux only (hold center fixed at centroid or reference value)
- Doesn't use per-pixel variances for flux measurement, but might also provide measurement with per-pixel variances (for diagnostics?)

8.6.2.6 Kron Photometry

- Compute Kron radius (hard to make this robust)
- Compute flux in elliptical aperture at Kron radius.

8.6.2.7 Petrosian Photometry

- Compute Petrosian radius. Harder than it seems due to need for improvements to splines? (ask RHL)
- Compute flux in elliptical aperture at Petrosian radius.

8.6.2.8 Galaxy Models

- Some sort of bulge+disk model. Lots of need for experimentation.
- Will Monte Carlo sample in MultiFit (and maybe on coadds, too, if that helps).
- May also fit to PSF-matched coadds for consistent colors.
- Will need to support simultaneous fitting (and sampling).
- Hybrid model candidate

8.6.2.9 Moving Point Source Models

- Fit point source with flux, centroid, parallax, and proper motion parameters.
- May need to support simultaneous fitting.
- Might want to sample this too, at least if we fit it simultaneously with sampled galaxy models.
- Hybrid model candidate

8.6.2.10 Trailed Point Source Models

- Fit PSF convolved with line segment to individual images

8.6.2.11 Dipole Models

- Fit PSF dipole for separation and flux to a combination of difference image and direct image.
- Deblending on direct image very problematic.

8.6.2.12 Spuriousness

- Some per-source measure of likelhood the detection is junk (in a difference image).
- May use machine learning on other measurements or pixels.
- May be augmented by spuriouness measures that aren't purely per-source.

8.6.3 Blended Measurement

- Integrate text from blended-measurement doc here.

8.6.3.1 Deblend Template Projection

8.6.3.2 Neighbor Noise Replacement

8.6.3.3 Simultaneous Fitting

8.6.3.4 Hybrid Models

8.7 Background Estimation

AUTHOR: Simon

- Fit or interpolate large-scale variations while masking out detections.
- Needs to work in crowded fields.
- Needs to work on both difference images and direct images.
- Need to be able to compose backgrounds measured in different coordinate systems on different scales.
- Needs to work on single CCDs for AP even if we use full FoV in DRP.

8.8 Build Background Reference

AUTHOR: Simon

- Given multiple overlapping visit images (already warped to a common coordinate system), synthesize a continuous single-epoch image that can be used as a reference for background matching.

8.9 PSF Estimation

8.9.1 Single CCD PSF Estimation

AUTHOR: Simon

- Fit simple empirical PSF model to stars from a single exposure.
- No chromaticity.
- May use external star catalog, but doesn't rely on one.
- Used only in Alert Production.

8.10 Wavefront Sensor PSF Estimation

AUTHOR: Jim

- Build an approximate PSF model using only the very brightest stars in the wavefront sensors. Because WF sensors are out-of-focus, these stars may be saturated on science CCDs.
- Model can have very few degrees of freedom (very simple optical model + elliptical Moffat/Double-Gaussian?)
- Only needs to be good enough to bootstrap PSF model well enough to bootstrap processing of science images (but it needs to work in crowded fields, too).
- Being able to go to brighter magnitudes may be important in crowded fields because the shape of the luminosity function may make it easier to find stars with (relatively) significant neighbors.

8.10.1 Full Visit PSF Estimation

AUTHOR: Jim

- Decompose PSF into optical + atmosphere.
- May also use wavefront sensors.
- Constrain model with stars, telemetry, and wavefront data.

- Wavelength-dependent.
- Used in RefineImChar in DRP.
- Must include some approach to dealing with wings of bright stars.

8.11 Model Spatial Variation of PSF

8.11.1 Within a CCD

- Estimate PSF at discrete locations using a set of basis functions
- Fit interpolation functions to fit coefficients to enable interpolation

8.11.2 Over a focal plane – Do we need this?

8.12 Aperture Correction

AUTHOR: Jim

- Measure curves of growth from bright stars (visit-level, at least in DRP)
- Correct various flux measurements to infinite (CCD-level)
- Propagate uncertainty in aperture correction to corrected fluxes; covariance is tricky.

8.13 Astrometric Fitting

AUTHOR: Simon

8.13.1 Single CCD

Used by AP, probably (RHL worries we might need full-visit)

- If this uses DRP's internal reference catalog, this does all we need. THIS IS A NEW DEPENDENCY BETWEEN DRP AND AP.

8.13.2 Single Visit

- Fit multi-component WCS to all CCDs in a single visit simultaneously after matching to reference catalog.

8.13.3 Joint Multi-Visit

- Fit multi-component WCS to all CCDs from multiple visits simultaneously after matching to reference catalog.

8.14 Photometric Fitting

AUTHOR: Simon (and Merlin?)

8.14.1 Single CCD (for AP)

- Match to photometric calibration reference catalog
- Calculate single zeropoint using available color terms

8.14.2 Single Visit

- Fit zeropoint (and some small spatial variation?) to all CCDs simultaneously after matching to reference catalog.
- Need for chromatic dependence unclear; probably driven by AP.

8.14.3 Joint Multi-Visit

- Derive SEDs for calibration stars from colors and reference catalog classifications.
- Utilize additional information from wavelenth dependent photometric calibration built by calibration products production.
- Fit zeropoint and possibly perturbations to all CCDs on multiple visits simultaneously after matching to reference catalog.

8.15 Retrieve Diffim Template for a Visit

AUTHOR: Simon

- Determine appropriate template to use
- Generate template for observation (may include DCR correction)

8.16 PSF Matching

AUTHOR: Simon

8.16.1 Image Subtraction

- Match template image to science image, as in Alert Production and DRP Difference Image processing.
- Includes identifying sources to use to determine matching kernel, fitting the kernel, and convolving by it.

8.16.2 PSF Homogenization for Coaddition

- Match science image to predetermined analytic PSF, as in PSF-matched coaddition.

8.17 Image Warping

AUTHOR: Jim

8.17.1 Oversampled Images

Oversampled images are warped to a new WCS and resampled using a two dimensional Lancsoz kernel of configurable order. The baselined default order is 3.

The one dimensional Lancsoz kernel of order a is defined as

$$L(x) = \begin{cases} \text{sinc}(x)\ \text{sinc}(x/a) & \text{if} \quad -a < x < a \\ 0 & \text{otherwise.} \end{cases}$$

The two dimensional Lancsoz kernel is $L(x, y) = L(x) \cdot L(y)$.

For each integer pixel position in the remapped image, the associated pixel position in the source image is determined using the source and destination WCS. The warping kernel is then applied to the source image to compute the remapped pixel value. A flux conservation factor is applied based on the relative sizes of the pixel in the source and destination WCS.

For performance reasons, it is desirable to reduce the total number of WCS calculations. It is therefore acceptable to perform the mapping between source and destination images over a regular grid and linearly interpolate between grid points, rather than mapping every pixel independently.

Since chromaticity is accounted for in the PSF rather than the WCS, no special account is taken of color when warping.

> **Note:**
> The above describes the current warping implementation in afw. We should identify deficiencies with the current implementation to establish resource requirements.

8.17.2 Undersampled Images

- Can use PSF model as interpolant if we also want to convolve with PSF (as in likelihood coadds). Otherwise impossible?

8.17.3 Irregularly-Sampled Images

- Approximate procedure for fixing small-scale distortions in pixel grid.

8.18 Image Coaddition

AUTHOR: Jim

- Must be able to do generalized outlier rejection, using histograms of detection masks produced on difference images.
- Needs to propagate full uncertainty somehow.
- Needs to propagate PSFs.
- Needs to propagate wavelength-dependent photometric calibration.
- May need to propagate larger-scale per-exposure masks to get right PSF model or other coadded quantities.
- Should be capable of combining coadds from different bands and/or epoch ranges ranges as well as combining individual exposures.
- Also needs to support combining snaps

8.19 DCR-Corrected Template Generation

AUTHOR: Simon

- Somwewhat like coaddition, but may need to add dimensions for wavelength or airmass, and may involve solving an inverse problem instead of just compute means.

8.20 Image Decorrelation

8.20.1 Difference Image Decorrelation

AUTHOR: Simon

- Fourier-space (?) deconvolution of preconvolved difference images before measurement - ZOGY as reinterpreted by Lupton (could apply correction in real space, too)
- Need to test with small-scale research before committing to this approach.

8.20.2 Coadd Decorrelation

AUTHOR: Jim

- Fourier-space/iterative deconvolution of likelihood coadds, as in DMTN-15.
- Need to test with small-scale research before committing to this approach.

8.21 Star/Galaxy Classification

AUTHOR: Jim

8.21.1 Single Frame S/G

- Extendedness or trace radius difference that classifies sources based on single frame measurements that can utilize the PSF model. Used to select single-frame calibration stars, and probably aperture correction stars.

8.21.2 Multi-Source S/G

- Aggregate of single-visit S/G post-PSF numbers in jointcal.

8.21.3 Object Classification

- Best classification derived from multifit and possibly variability.

8.22 Variability Characterization

Following the DPDD, lightcurve variability is characterized by providing a series of numeric summary 'features' derived from the lightcurve. The DPDD baselines an approach based on Richards et al. [24], with the caveat that ongoing work in time domain astronomy may change the definition, but not the number or type, of features being provided.

Richards et al. define two classes of features: those designed to characterize variability which is periodic, and those for which the period, if any, is not important. We address both below.

All of these metrics are calculated for both Objects (DPDD table 4, `lcPeriodic` and `lcNonPeriodic`) and DIAObjects (DPDD table 2, `lcPeriodic` and `lcNonPeriodic`). They are calculated and recorded separately in each band. Calculations for Objects are performed based on forced point source model fits (DPDD table 5, `psFlux`). Calculations for DIAObjects are performed based on point source model fits to DIASources (DPDD table 1, `psFlux`). In each case, calculation requires the fluxes and errors for all of the sources in the lightcurve to be available in memory simultaneously.

8.22.1 Characterization of periodic variability

- Characterize lightcurve as the sum of a linear term plus sinusoids at three fundamental frequencies plus four harmonics:

$$y(t) = ct + \sum_{i=1}^{3} \sum_{j=1}^{4} y_i(t|jf_i) \tag{1}$$

$$y_i(t|jf_i) = a_{i,j} \sin(2\pi j f_i t) + b_{i,j} \cos(2\pi j f_i t) + b_{i,j,0} \tag{2}$$

where i sums over fundamentals and j over harmonics.

- Use iterative application of the generalized Lomb-Scargle periodogram, as described in [24], to establish the fundamental frequencies, f_1, f_2, f_3:
    - Search a configurable (minimum, maximum, step) linear frequency grid with the periodogram, applying a $\log f/f_N$ penalty for frequencies above $f_N = 0.5\langle 1/\Delta T\rangle$, identifying the frequency f_1 with highest power;
    - Fit and subtract that frequency and its harmonics from the lightcurve;
    - Repeat the periodogram search to identify f_2 and f_3.
- We report a total of 32 floats:
    - The linear coefficient, c (1 float)
    - The values of f_1, f_2, f_3. (3 floats)
    - The amplitude, $\mathrm{A}_{i,j} = \sqrt{a_{i,j}^2 + b_{i,j}^2}$, for each i, j pair. (12 floats)
    - The phase, $\mathrm{PH}_{i,j} = \arctan(b_{i,j}, a_{i,j}) - \frac{jf_i}{f_1}\arctan(b_{1,1}, a_{1,1})$, for each i, j pair, setting $\mathrm{PH}_{1,1} = 0$. (12 floats)
    - The significance of f_1 vs. the null hypothesis of white noise with no periodic signal. (1 float)
    - The ratio of the significance of each of f_2 and f_3 to the significance of f_1. (2 floats)
    - The ratio of the variance of the lightcurve before subtraction of the f_1 component to its variance after subtraction. (1 float)

  NB the DPDD baselines providing 32 floats, but, since $\mathrm{PH}_{1,1}$ is 0 by construction, in practice only 31 need to be stored.

8.22.2 Characterization of aperiodic variability

In addition to the periodic variability described above, we follow [24] in providing a series of statistics computed from the lightcurve which do not assume peridoicity. They define 20 floating point quantities in four groups which we describe here, again with the caveat that future revisions to the DPDD may require changes to this list.

Basic quantities:

- The maximum value of delta-magnitude over delta-time between successive points in the lightcurve.
- The difference between the maximum and minimum magnitudes.
- The median absolute deviation.
- The fraction of measurements falling within 1/10 amplitudes of the median.
- The "slope trend": the fraction of increasing minus the fraction of decreasing delta-magnitude values between successive pairs of the last 30 points in the lightcurve.

Moment calculations:

- Skewness.
- Small sample kurtosis, i.e.

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)} \tag{3}$$

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2} \tag{4}$$

- Standard deviation.
- The fraction of magnitudes which lie more than one standard deviation from the weighted mean.
- Welch-Stetson variability index J [26], defined as

$$J = \frac{\sum_k \operatorname{sgn}(P_k) \sqrt{|P_k|}}{K},$$

  where the sum runs over all K pairs of observations of the object, where sgn returns the sign of its argument, and where

$$P_k = \delta_i \delta_j \tag{5}$$

$$\delta_i = \sqrt{\frac{n}{n-1}} \frac{\nu_i - \overline{\nu}}{\sigma_\nu}, \tag{6}$$

where n is the number of observations of the object, and ν_i its flux in observation i. Following the procedure described in Stetson [26], the mean is not the simple weighted algebraic mean, but is rather reweighted to account for outliers.

- Welch-Stetson variability index K [26], defined as

$$K = \frac{1/n \sum_{i=1} N|\delta_i|}{\sqrt{1/n \sum_{i=1} N|\delta_i^2|}},$$

  where N is the total number of observations of the object and δ_i is defined as above.

Percentiles. Taking, for example, $F_{5,95}$ to be the difference between the 95% and 5% flux values, we report:

- All of $F_{40,60}/F_{5,95}$, $F_{32.5,67.5}/F_{5,95}$, $F_{25,75}/F_{5,95}$, $F_{17.5,82.5}/F_{5,95}$, $F_{10,90}/F_{5,95}$
- The largest absolute departure from the median flux, divided by the median.
- The radio of $F_{5,95}$ to the median.

QSO similarity metrics, as defined by Butler & Bloom [10]:

- $\chi^2_{\mathrm{QSO}}/\nu$.
- $\chi^2_{\mathrm{False}}/\nu$.

8.23 Proper Motion and Parallax from DIASources

AUTHOR: Simon

- Fit proper motion and parallax models to all potisions of DIASources belonging to a DIAObject taking into account errors.

8.24 Association and Matching

8.24.1 Single CCD to Reference Catalog, Semi-Blind

AUTHOR: Simon

- Want to match in image coordinates, so also needs to transform reference catalog.
- Run prior to single-visit WCS fitting, with only telescope's best guess as a starting WCS.
- Single CCD form needed by AP.

8.24.2 Single Visit to Reference Catalog, Semi-Blind

AUTHOR: Simon

- Want to match in focal plane coordinates, so also needs to transform reference catalog.
- Run prior to single-visit WCS fitting, with only telescope's best guess as a starting WCS.

8.24.3 Multiple Visits to Reference Catalog

AUTHOR: Jim

- Match sources from multiple visits to a single reference catalog, assuming good WCS solutions.

8.24.4 N-Way Matching Across Multiple Visits

AUTHOR: MWV

- Match sources and associate objects from M catalogs each with ∼N sources. The API should match in either (x, y) or (RA, Dec). Positions for source detections solutions will be assumed to already be correct. Order of individual catalogs should not matter. Algorithm will need to be able to run on M∼1,000 visits. Such a tool will allow flexible analyses without the requirement for a larger database structure or full coadd-based object identifiction and forced photometry. Even within the framework of a complete Level-2 DRP release, such a N-way matching capability will also be important for comparing the results of single-visit photometry with the deep coadd-based object detection and forced photometry. A specific example use case for lightweight quality assessment is taking the processed catalogs for M=1,000 images each with N=2,000 sources and creating object associations add derived repeatability and time-variable summary statistics. This algorithm and associated API should provide a general purpose tool useful for algorithm developers, data quality assessment, and science users. A trivial in-memory version (using full catalogs), a streamlined in-memory version (load only the coordinates), and a larger-than-memory version will each be useful and important and will entail increasingly more significant design and performance efforts.

8.24.5 DIAObject Generation

AUTHOR: Simon

- Match all DIASources to predicted Solar System object positions and existing DIAObjects and generate new DIAObjects. Definitely run in AP, maybe run in DRP.

8.24.6 Object Generation

AUTHOR: Jim

- Match coadd detections from different bands/SEDs/epoch-ranges, merging Footprints and associating peaks.
- Also merge in DIASources or (if already self-associated) DIAObjects.

8.24.7 Blended Overlap Resolution

AUTHOR: Jim

- Given two or more overlapping blend families (with associated measurements), merge them by selecting the “best” measurement for each child object.

8.25 Raw Measurement Calibration

AUTHOR: Jim

- Apply astrometric and photometric calibrations to measurements in raw units, transforming them to calibrated quantities.
- May be applied within the database after ingest in some contexts, but needs to be available outside the database as well.

8.26 Ephemeris Calculation

AUTHOR: Simon

- Calculate positions for all solar system objects in a region at a given time.

8.27 Make Tracklets

AUTHOR: Simon

- Make all tracklet pairs
- Merge multiple chained observation into single longer tracklets
- Purge any tracklets inconsistent with the merged tracklets

8.28 Attribution and precovery

AUTHOR: Simon

- Predict locations of known Solar System objects
- Match tracklet observation to predicted ephimerides taking into account velocity
- Update SSObjects
- Possibly iterate

8.29 Orbit Fitting

AUTHOR: Simon

- Merge unassociated tracklets into tracks.
- Fit orbits to all tracks.
- Purge unphysical tracks.
- Update SSObjects
- Possibly iterate

9 Software Primitives

9.1 Images

9.1.1 Exposure

Image A 2-d array of calibrated, background-subtracted pixel values in counts.

Mask A boolean representation of artifacts, detections, saturation, and other image. This may include (but is not limited to) a 2-d integer arrays with bits interpreted as different "mask planes"; it may also include using Footprints to describe labeled regions.

Variance A representation of the uncertainty in the image. This includes at least a 2-d array capturing the variance in each pixel, and it may involve some other scheme to capture the variance.

Background An object describing the background model that was subtracted from the image; the original unsubtracted image can be obtained by adding an image of this model to the Exposure's image plane. Backgrounds are more complex than merely an image or even an interpolated binned image; background estimation will proceed in several stages, and these stages (which may happen in different coordinate systems) must be combined to form the full background model.

PSF A model of the PSF; see PSF. This includes a model for aperture corrections.

WCS The astrometric solution that related the image's pixel coordinate system to coordinates on the sky; see WCS.

PhotoCalib The photometric solution that relates the image's pixel values to magnitudes as a function of source wavelength or SED. Some PhotoCalibs may represent global calibration and some may represent relative calibration.

9.2 Tables

9.2.1 Source

9.2.2 Object

9.2.3 Reference

9.3 Footprints

9.4 Convolution Kernels

Must support correlation as well.

9.5 Basic Statistics

9.6 Point-Spread Functions

9.7 Coordinate Transformations

10 Glossary

API Applications Programming Interface

CBP Collimated Beam Projector

CCOB Camera Calibration Optical Bench

CTE Charge Transfer Efficiency

DAC Data Access Center

DAQ Data Acquisition

DMS Data Management System

DR Data Release.

EPO Education and Public Outreach

Footprint The set of pixels that contains flux from an object. Footprints of multiple objects may have pixels in common.

FRS Functional Requirements Specification

MOPS Moving Object Pipeline System

OCS Observatory Control System

Production A coordinated set of pipelines

PFS Prime Focus Spectrograph. An instrument under development for the Subaru Telescope.

PSF Point Spread Function

QE Quantum Efficiency

RGB Red-Green-Blue image, suitable for color display.

SDS Science Array DAQ Subsystem. The system on the mountain which reads out the data from the camera, buffers it as necessary, and supplies it to data clients, including the DMS.

SDQA Science Data Quality Assessment.

SNR Signal-to-Noise Ratio

SQL Structured Query Language, the common language for querying relational databases.

TBD To Be Determined

Visit A pair of exposures of the same area of the sky taken in immediate succession. A Visit for LSST consists of a 15 second exposure, a 2 second readout time, and a second 15 second exposure.

VO Virtual Observatory

VOEvent A VO standard for disseminating information about transient events.

WCS World Coordinate System. A bidirectional mapping between pixel- and sky-coordinates.

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel and J. Sander, **OPTICS: Ordering Points To Identify the Clustering Structure**, Proc ACM SIGMOD (1999).

[2] P. Antilogus, P. Astier, P .Doherty, A. Guyonnet and N. Regnault **The brighter-fatter effect and pixel correlations in CCD sensors** Journal of Instrumentation, Volume 9, Issue 3, article id. C03048 (2014).

[3] A. Becker et al, **Report on Late Winter 2013 Production: Image Differencing** `http://ls.st/x9f`.

[4] A. Becker, **Report on Summer 2014 Production: Analysis of DCR**, `https://github.com/lsst-dm/S14DCR/blob/master/report/S14report_V0-00.pdf`.

[5] **An algorithm for precise aperture photometry of critically sampled images**, MNRAS 431, 1275–1285, 2013

[6] J. S. Bloom et al, **Automating discovery and classification of transients and variable stars in the synoptic survey era**, PASP 124, 1175–1196 (2012).

[7] J. F. Bosch, **Modeling Techniques for Measuring Galaxy Properties in Multi-Epoch Surveys**, PhD Thesis, University of California, Davis (2011). `http://adsabs.harvard.edu/abs/2011PhDT.......226B`

[8] J. Bosch, P. Gee, R. Owen, M. Juric and the LSST DM team, **LSST DM S13 Report: Shape measurement plans and prototypes**, `https://docushare.lsstcorp.org/docushare/dsweb/ImageStoreViewer/Document-15298`

[9] J. Bosch, **Measurement of Blended Objects in LSST**.

[10] **Optimal Time-Series Selection of Quasars**, ApJ 141, 93 (2011).

[11] **Automated Supervised Classification of Variable Stars I. Methodology**, A&A 475, 1159–1183 (2007).

[12] E. M. Huff et al, **Seeing in the dark – I. Multi-epoch alchemy**, `http://arxiv.org/abs/1111.6958`.

[13] H. Furusawa et al, **Hyper Suprime-Cam Survey Pipeline Description**, `http://hsca.ipmu.jp/pipeline_outputs.pdf`.

[14] M. J. Jee and J. A. Tyson, **Toward Precision LSST Weak-Lensing Measurement. I. Impacts of Atmospheric Turbulence and Optical Aberration**, PASP 123, 596(2011).

[15] M. J. Jee, J. A. Tyson, M. D. Schneider, D. Wittman, S. Schmidt and S. Hilbert, **Cosmic shear results from the Deep Lens Survey. I. Joint constraints on Ω_M and σ_8 with a two-dimensional analysis**, ApJ 765 74 (2013).

[16] J. Kubica et al, **Efficiently Tracking Moving Sources in the LSST**, Bulletin of the American Astronomical Society, 37, 1207 (2005).

[17] D. Lang, D. Hogg, S. Jester and H.-W. Rix, **Measuring the undetectable: Proper motions and parallaxes of very faint sources**, AJ 137 4400–4111 (2009).

[18] R. H. Lupton et al, **SDSS Image Processing II: The *Photo* Pipelines**. `http://www.astro.princeton.edu/~rhl/photo-lite.pdf`

[19] R. Lupton and Ž. Ivezić, **Experience with SDSS: the Promise and Perils of Large Surveys**, Astrometry in the Age of the Next Generation of Large Telescopes, ASP Conferences Series, Vol 338 (2005). `http://adsabs.harvard.edu/abs/2005ASPC..338..151L`

[20] R. Lupton, M. Jurić and C. Stubbs, **LSST's Plans for Calibration Photometry**, July 2015.

[21] L. Denneau, J. Kubica and R. Jedicke, **The Pan-STARRS Moving Object Pipeline**, Astronomical Data Analysis Software and Systems XVI ASP Conference Series, Vol. 376, proceedings of the conference held 15-18 October 2006 in Tucson, Arizona, USA. Edited by Richard A. Shaw, Frank Hill and David J. Bell., p.257.

[22] N. Padmanabhan et al, **An Improved Photometric Calibration of the Sloan Digital Sky Survey Imaging Data**, ApJ 674 1217–1233 (2008).

[23] A. Rasmussen, **Sensor Modeling for the LSST Camera Focal Plane: Current Status of SLAC Originated Code** July 2015. `https://docushare.lsstcorp.org/docushare/dsweb/Get/Document-8590`.

[24] J. Richards et al, **On Machine-learned Classification of Variable Stars with Sparse and Noisy Time-series Data**, ApJ 733 10 (2011).

[25] E. F. Schlafly et al, **Photometric Calibration of the First 1.5 Years of the Pan-STARRS1 Survey**, ApJ 756 158 (2012).

[26] Stetson, P.B., **On the Automatic Determination of Light-Curve Parameters for Cepheid Variables**, PASP 108 851–876 (1996).

[27] C. W. Stubbs, **Precision Astronomy with Imperfect Fully Depleted CCDs – An Introduction and a Suggested Lexicon**, Journal of Instrumentation, Volume 9, Issue 3, article id. C03032 (2014).

[28] A. S. Szalay, A. J. Connolly and G. P. Szokoly, **Simultaneous Multicolor Detection of Faint Galaxies in the Hubble Deep Field**, AJ 117 68–74 (1999).