Large Synoptic Survey Telescope
Data Management Applications Design

M. Jurić*, R.H. Lupton, T. Axelrod, A.C. Becker, J. Becla,
J.F. Bosch, D.R. Ciardi, A.J. Connolly, G.P. Dubois-Felsmann,
F. Economou, M. Fisher-Levine, Ž. Ivezić, M. Graham,
T. Jenness, R.L. Jones, J. Kantor, S. Krughoff, K-T. Lim,
F. Mueller, D. Petravick, P.A. Price, D. Shaw, C. Slater,
J. Swinbank, M. Wood-Vasey, X. Wu,
for the LSST Data Management

Monday 6th February, 2017, 21:36hrs

*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 **13**

2 Introduction **14**
2.1 LSST Data Management System 14
2.2 Data Products 17
2.3 Data Units 18
2.4 Science Pipelines Organization 19

3 Alert Production **21**
3.1 Single Frame Processing Pipeline (WBS 02C.03.01) 22
3.1.1 Input Data 22
3.1.2 Output Data 23
3.1.3 Instrumental Signature Removal 24
3.1.3.1 Pipeline Tasks 24
3.1.4 PSF and background determination 25
3.1.4.1 Pipeline Tasks 25
3.1.5 Source measurement 26
3.1.6 Photometric and Astrometric calibration 26
3.1.6.1 Pipeline Tasks 27
3.2 Alert Generation Pipeline (WBS 02C.03.04) 28
3.2.1 Input Data 28
3.2.2 Output Data 29
3.2.3 Template Generation 30
3.2.3.1 Pipeline Tasks 30
3.2.4 Image differencing 30
3.2.4.1 Pipeline Tasks 30
3.2.5 Source Association 32
3.2.5.1 Pipeline Tasks 32
3.3 Alert Distribution Pipeline (WBS 02C.03.03) 36
3.3.1 Input Data 37
3.3.2 Output Data 37
3.3.3 Alert postage stamp generation 37
3.3.3.1 Pipeline Tasks 38
3.3.4 Alert queuing and persistance 38
3.3.4.1 Pipeline Tasks 38

3.4 Precovery and Forced Photometry Pipeline . . . 40
3.4.1 Input Data . . . 41
3.4.2 Output Data . . . 41
3.4.3 Forced Photometry on all DIAObjects . . . 41
3.4.3.1 Pipeline Tasks . . . 41
3.4.4 DIAObject Forced Photometry: . . . 42
3.4.4.1 Pipeline Tasks . . . 42
3.5 Moving Object Pipeline (WBS 02C.03.06) . . . 43
3.5.1 Input Data . . . 43
3.5.2 Output Data . . . 43
3.5.3 Tracklet identification . . . 44
3.5.3.1 Pipeline Tasks . . . 44
3.5.4 Precovery and merging of tracklets . . . 44
3.5.4.1 Pipeline Tasks . . . 45
3.5.5 Linking tracklets and orbit fitting . . . 45
3.5.5.1 Pipeline Tasks . . . 45
3.5.6 Global precovery . . . 46
3.5.6.1 Pipeline Tasks . . . 46
3.5.7 Prototype Implementation . . . 47

4 Calibration Products Pipeline 49
4.1 Key Requirements . . . 49
4.2 Inputs . . . 49
4.2.1 Bias Frames . . . 50
4.2.2 Gain Values . . . 50
4.2.3 Linearity . . . 50
4.2.4 Darks . . . 50
4.2.5 Crosstalk . . . 50
4.2.6 Defect Map . . . 51
4.2.7 Saturation levels . . . 51
4.2.8 Broadband Flats . . . 51
4.2.9 Monochromatic Flats . . . 51
4.2.10 CBP Data . . . 51
4.2.10.1 CBP dataset 1 . . . 51
4.2.10.2 CBP dataset 2 . . . 52
4.2.10.3 CBP dataset 3 . . . 52
4.2.10.4 CBP dataset 4 . . . 52
4.2.10.5 CBP dataset 5 . . . 52

4.2.10.6 CBP Crosstalk Measurement 52
4.2.11 Filter Transmission 53
4.2.12 Stellar spectra 53
4.2.13 Other stellar spectra (*n.b.* != 4.2.12) 53
4.2.14 Atmospheric Characterization 53
4.2.15 Photometric Standards 53
4.3 Outputs from the Calibration Product Pipelines == Inputs to the AP/DRP Pipelines 54
4.3.1 Master Bias 54
4.3.2 Master Darks 54
4.3.3 Master Linearity 55
4.3.4 Master Fringe Frames 55
4.3.5 Master Gain Values 55
4.3.6 Master Defects 55
4.3.7 Saturation Levels 56
4.3.8 Crosstalk 56
4.3.9 Impure Monochromatic Flats 57
4.3.10 Pure Monochromatic Flats 57
4.3.11 PhotoFlats 57
4.3.12 Low-res narrow-band flats 57
4.3.13 Pixel Sizes 57
4.3.14 Brighter-Fatter Coefficients 58
4.3.15 CTE Measurement 58
4.3.16 Filter Transmission 58
4.3.17 Ghost catalog 59
4.4 CBP Control 59
4.5 Calibration Telescope Input Calibration Data 59
4.6 Calibration Telescope Output Data 60
4.6.1 Atmospheric Absorption 61
4.6.2 Night Sky Spectrum 61
4.7 Miscellaneous Algorithms 63
4.8 Prototype Implementation 64

5 Data Release Production **65**
5.1 Image Characterization and Calibration 68
5.1.1 BootstrapImChar 70
5.1.1.1 Input Data Product: Raw 71
5.1.1.2 Input Data Product: Reference 71

5.1.1.3 Input Data Product: Calibrations 72
5.1.1.4 Output Data Product: Source 72
5.1.1.5 Output Data Product: CalExp 72
5.1.1.6 RunISR 73
5.1.1.7 SubtractSnaps 73
5.1.1.8 CombineSnaps 73
5.1.1.9 FitWavefront 73
5.1.1.10 SubtractBackground 74
5.1.1.11 DetectSources 74
5.1.1.12 DeblendSources 74
5.1.1.13 MeasureSources 74
5.1.1.14 MatchSemiBlind 75
5.1.1.15 SelectStars 75
5.1.1.16 FitWCS 75
5.1.1.17 FitPSF 76
5.1.1.18 WriteDiagnostics 76
5.1.1.19 SubtractStars 76
5.1.1.20 ReinsertStars 76
5.1.1.21 MatchNonBlind 76
5.1.1.22 FitApCorr 77
5.1.1.23 ApplyApCorr 77
5.1.2 StandardJointCal 77
5.1.3 RefineImChar 79
5.1.4 FinalImChar 80
5.1.5 FinalJointCal 80
5.2 Image Coaddition and Image Differencing 81
5.2.1 WarpAndPsfMatch 85
5.2.2 BackgroundMatchAndReject 85
5.2.3 WarpTemplates 87
5.2.4 CoaddTemplates 88
5.2.5 DiffIm 89
5.2.6 UpdateMasks 90
5.2.7 WarpRemaining 90
5.2.8 CoaddRemaining 91
5.3 Coadd Processing 91
5.3.1 DeepDetect 92
5.3.2 DeepAssociate 93
5.3.3 DeepDeblend 94

5.3.4 MeasureCoadds . . . 95
5.4 Overlap Resolution . . . 96
5.4.1 ResolvePatchOverlaps . . . 97
5.4.2 ResolveTractOverlaps . . . 98
5.5 Multi-Epoch Object Characterization . . . 99
5.5.1 MultiFit . . . 101
5.5.2 ForcedPhotometry . . . 102
5.6 Postprocessing . . . 103
5.6.1 MovingObjectPipeline . . . 103
5.6.2 ApplyCalibrations . . . 104
5.6.3 MakeSelectionMaps . . . 105
5.6.4 Classification . . . 106
5.6.5 GatherContributed . . . 106

6 Science Data Quality Assurance 108
6.1 Overview of SDQA . . . 108
6.1.1 Quality Analysis (QA) . . . 108
6.1.2 Quality Control (QC) . . . 108
6.2 Key Features of SDQA (QA and QC) systems . . . 109
6.3 Key Tasks for Each Tier of QC . . . 110
6.3.1 QC Tier 0 . . . 110
6.3.1.1 Continuous Integration Services . . . 111
6.3.1.2 Test Execution Harness . . . 111
6.3.1.3 Verification Metrics Code . . . 111
6.3.1.4 Computational Metrics . . . 112
6.3.1.5 Curated Datasets . . . 113
6.3.1.6 SQUASH - Science Quality Analysis Harness 113
6.3.2 QC Tier 1 . . . 114
6.3.2.1 Alert QC . . . 115
6.3.2.2 Validation Metrics Performance . . . 116
6.3.2.3 Dome / Operator Displays . . . 116
6.3.2.4 Telescope Systems . . . 116
6.3.2.5 Camera Calibration . . . 116
6.3.2.6 Engineering and Commissioning . . . 117
6.3.2.7 Data Release Production . . . 117
6.3.3 QC Tier 2 . . . 117
6.3.3.1 DRP-specific dataset . . . 118

6.3.3.2 Interfaces to Workflow and Provenance System(s) . . . 118
6.3.3.3 Output Interface to Science Pipelines . . . 118
6.3.3.4 Comparison tools for overlap areas due to satellite processing . . . 119
6.3.3.5 Metrics/products for science users to understand quality of science data products (depth mask/selection function, etc.) . . . 119
6.3.3.6 Characterization report for Data Release . . . 119
6.3.4 QC Tier 3 . . . 119
6.3.5 Quality Analysis . . . 119
6.3.6 Who validates the validator? . . . 120
6.3.6.1 Intrinsic Design Features . . . 121
6.3.6.2 Known Truth . . . 121
6.3.6.3 Reference Truth . . . 121

7 Data Access, the Science User Interface and Tools 122
7.1 Data Access Layer . . . 122
7.2 Science User Interface and Tools . . . 123

8 Algorithmic Components 124
8.1 Reference Catalog Construction . . . 124
8.1.1 Alert Production Reference Catalogs . . . 124
8.1.2 Data Release Production Reference Catalogs . . . 124
8.2 Instrument Signature Removal . . . 125
8.2.1 ISR for Alert Production . . . 127
8.3 Artifact Detection . . . 127
8.3.1 Cosmic Ray Identification . . . 127
8.3.2 Optical ghosts . . . 129
8.3.3 Linear feature detection and removal . . . 130
8.3.4 Snap Subtraction . . . 131
8.3.4.1 Cosmic Rays . . . 131
8.3.4.2 Ghosts . . . 131
8.3.4.3 Linear features . . . 131
8.3.5 Warped Image Comparison . . . 132
8.4 Artifact Interpolation . . . 132
8.5 Source Detection . . . 133
8.6 Deblending . . . 134

8.6.1 Single Frame Deblending . . . 136
8.6.2 Multi-Coadd Deblending . . . 136
8.7 Measurement . . . 137
8.7.1 Drivers . . . 138
8.7.1.1 Single Frame Measurement: . . . 138
8.7.1.2 Multi-Coadd Measurement: . . . 138
8.7.1.3 Difference Image Measurement: . . . 138
8.7.1.4 Multi-Epoch Measurement: . . . 139
8.7.1.5 Forced Measurement: . . . 139
8.7.2 Algorithms . . . 140
8.7.2.1 Centroids . . . 140
8.7.2.2 Pixel Flag Aggregation . . . 141
8.7.2.3 Second-Moment Shapes . . . 141
8.7.2.4 Aperture Photometry . . . 141
8.7.2.5 Static Point Source Photometry . . . 142
8.7.2.6 Kron Photometry . . . 142
8.7.2.7 Petrosian Photometry . . . 143
8.7.2.8 Galaxy Models . . . 143
8.7.2.9 Moving Point Source Models . . . 146
8.7.2.10 Trailed Point Source Models . . . 147
8.7.2.11 Dipole Models . . . 147
8.7.2.12 Spuriousness . . . 148
8.7.3 Blended Measurement . . . 149
8.7.3.1 Neighbor Noise Replacement . . . 150
8.7.3.2 Deblend Template Projection . . . 151
8.7.3.3 Simultaneous Fitting . . . 152
8.7.3.4 Hybrid Models . . . 153
8.8 Spatial Models . . . 154
8.9 Background Estimation . . . 154
8.9.1 Single-Visit Background Estimation . . . 154
8.9.2 Coadd Background Estimation . . . 155
8.9.3 Matched Background Estimation . . . 155
8.10 Build Background Reference . . . 156
8.10.1 Patch Level . . . 156
8.10.2 Tract Level . . . 156
8.11 PSF Estimation . . . 157
8.11.1 Single CCD PSF Estimation . . . 157
8.11.2 Wavefront Sensor PSF Estimation . . . 158

8.11.3 Full Visit PSF Estimation . . . 158
8.12 Aperture Correction . . . 159
8.13 Astrometric Fitting . . . 159
8.13.1 Single CCD . . . 159
8.13.2 Single Visit . . . 160
8.13.3 Joint Multi-Visit . . . 160
8.14 Photometric Fitting . . . 160
8.14.1 Single CCD (for AP) . . . 160
8.14.2 Single Visit . . . 160
8.14.3 Joint Multi-Visit . . . 161
8.14.4 Large-Scale Fitting . . . 161
8.14.4.1 Global Fitting . . . 161
8.14.4.2 Interim Wavelength-Dependent Fitting . . . 162
8.15 Retrieve Diffim Template for a Visit . . . 163
8.16 PSF Matching . . . 163
8.16.1 Image Subtraction . . . 163
8.16.2 PSF Homogenization for Coaddition . . . 165
8.17 Image Coaddition . . . 166
8.18 DCR-Corrected Template Generation . . . 166
8.18.1 Refraction from the Atmosphere . . . 167
8.18.2 Generating a DCR Corrected Template . . . 167
8.19 Image Decorrelation . . . 168
8.19.1 Difference Image Decorrelation . . . 168
8.19.2 Coadd Decorrelation . . . 169
8.20 Star/Galaxy Classification . . . 170
8.20.1 Single Frame S/G . . . 170
8.20.2 Multi-Source S/G . . . 170
8.20.3 Object Classification . . . 170
8.21 Variability Characterization . . . 170
8.21.1 Characterization of Periodic Variability . . . 171
8.21.2 Characterization of Aperiodic Variability . . . 172
8.22 Proper Motion and Parallax from DIASources . . . 174
8.23 Association and Matching . . . 174
8.23.1 Single CCD to Reference Catalog, Semi-Blind . . . 175
8.23.2 Single Visit to Reference Catalog, Semi-Blind . . . 176
8.23.3 Multiple Visits to Reference Catalog . . . 176
8.23.4 DIAObject Generation . . . 176
8.23.5 Object Generation . . . 177

8.23.6 Blended Overlap Resolution . . . 178
8.24 Raw Measurement Calibration . . . 178
8.25 Ephemeris Calculation . . . 178
8.26 Make Tracklets . . . 179
8.27 Attribution and Precovery . . . 179
8.28 Orbit Fitting . . . 180
8.29 Orbit Merging . . . 181

9 Software Primitives 182
9.1 Cartesian Geometry . . . 182
9.1.1 Points . . . 182
9.1.2 Arrays of Points . . . 183
9.1.3 Boxes . . . 183
9.1.4 Polygons . . . 183
9.1.5 Ellipses . . . 184
9.2 Spherical Geometry . . . 184
9.2.1 Points . . . 185
9.2.2 Arrays of Points . . . 185
9.2.3 Boxes . . . 185
9.2.4 Polygons . . . 185
9.2.5 Ellipses . . . 185
9.3 Images . . . 186
9.3.1 Simple Images . . . 186
9.3.2 Masks . . . 186
9.3.3 MaskedImages . . . 186
9.3.4 Exposure . . . 187
9.4 Multi-Type Associative Containers . . . 188
9.5 Tables . . . 188
9.5.1 Source . . . 188
9.5.2 Object . . . 189
9.5.3 Exposure . . . 189
9.5.4 AmpInfo . . . 189
9.5.5 Reference . . . 189
9.5.6 Joins . . . 189
9.5.7 Queries . . . 190
9.5.8 N-Way Matching . . . 190
9.6 Footprints . . . 191
9.6.1 PixelRegions . . . 191

9.6.2 Functors . . . 191
9.6.3 Peaks . . . 191
9.6.4 FootprintSets . . . 191
9.6.5 HeavyFootprints . . . 192
9.6.6 Thresholding . . . 192
9.7 Basic Statistics . . . 192
9.8 Chromaticity Utilities . . . 192
9.8.1 Filters . . . 192
9.8.2 SEDs . . . 193
9.8.3 Color Terms . . . 193
9.9 PhotoCalib . . . 193
9.10 Convolution Kernels . . . 194
9.11 Coordinate Transformations . . . 194
9.12 Numerical Integration . . . 195
9.13 Random Number Generation . . . 195
9.14 Interpolation and Approximation of 2-D Fields . . . 195
9.15 Common Functions and Source Profiles . . . 195
9.16 Camera Descriptions . . . 196
9.17 Numerical Optimization . . . 196
9.18 Monte Carlo Sampling . . . 196
9.19 Point-Spread Functions . . . 196
9.20 Warping . . . 197
9.21 Fourier Transforms . . . 197
9.22 Tree Structures . . . 197
9.23 Tools . . . 197

10 Glossary **199**

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

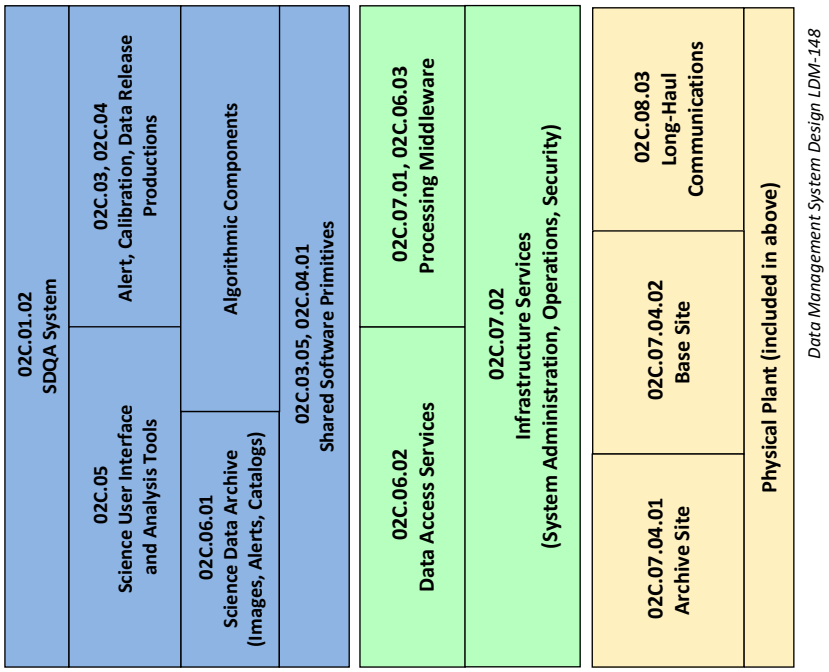
2.1 LSST Data Management System

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 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

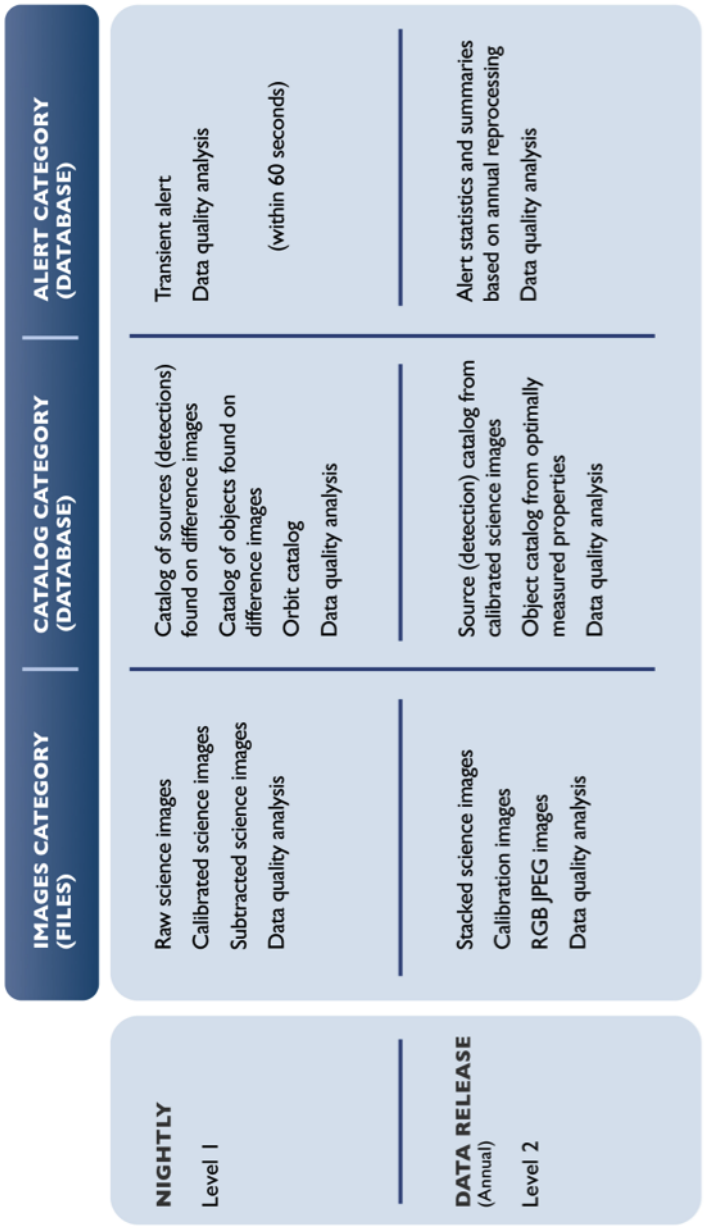


Figure 2: Organization of LSST Data Products

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 relatively 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 Data 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.

Name	Availability	Description
`DIASource`	Stored	Measurements from difference image analysis of individual exposures.
`DIAObject`	Stored	Aggregate quantities computed by associating spatially colocated `DIASources`.
DIAForcedSource	Stored	Flux measurements on each difference image at the position of a `DIAObject`.
`SSObject`	Stored	Solar system objects derived by associating `DIASources` and inferring their orbits.
CalExp	Stored	Calibrated exposure images for each CCD/visit (sum of two snaps) and associated metadata (e.g. WCS and estimated background).
TemplateCoadd	Temporary	DCR corrected template coadd.
DiffExp	Stored	Difference between CalExp and PSF-matched template coadd.
VOEvent	Stored	Database of VOEvents as streamed from the Alert Production
Tracklets	Persisted	Intermediate data product for the generation of `SSObjects` generated by linking moving sources within a given night

Table 1: Table of derived and persisted data products produced during Alert Production. A detailed description of these data products can be found in the Data Products Definition Document (LSE-163).

3 Alert Production

Alert Production is run each night to produce catalogs and images for sources that have varied or moved relative to a previous observation. The data products produced by Alert production are given in Table 1.

Alert Production is designed as five separate components: single frame processing, alert generation, alert distribution, precovery photometry, and a moving objects pipeline. The first four of these components run as a linear pass through of the data. The moving objects pipeline is run independently of the rest of the alert production. The flow of information through this system is shown in Figure 3.

In this document we do not address estimation of the selection function

for alert generation through the injection of simulated sources. Such a process could be undertaken in batch mode as part of the DRP. Source detection thresholds can be estimated through the use of sky sources (PSF photometry measurements positioned in areas of blank sky).

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

The Single Frame Processing (SFM) Pipeline (see Figure 4) is responsible for reducing raw or camera-corrected image data to *calibrated exposures*, the detection and measurement of `Sources` (using the components functionally part of the Object Characterization Pipeline), the characterization of the point-spread-function (PSF), and the generation of an astrometric solution for an image. Calibrated exposures produced by the SFM pipeline must possess all information necessary for measurement of source properties by single-epoch Object Characterization algorithms.

Astrometric and photometric calibration requires the detection and measurement of the properties of `Sources` on a CCD. Accurate centroids and fluxes for these `Sources` require an estimation of the PSF and background, which in turn requires knowledge of the positions of the `Sources` on an image. The SFM pipeline will, therefore, iterate over background estimation (see 3.1.4) and source measurement (see 3.1.5)

The SFM pipeline will be implemented as a flexible framework where new processing steps can be added without modifying the stack code (this would include the ability to process non-crosstalk corrected images should a network outage between the base and processing center result in only the raw data being available). The pipeline, or a subset of the pipeline, should be capable of being run at the telescope facility during commissioning and operations.

3.1.1 Input Data

Raw Camera Images: Amplifier images that have been corrected for crosstalk and bias by the camera software. All images from a visit should be available to the task (including snaps). An approximate WCS is assumed to be available as metadata derived from the Telescope Control System with an absolute pointing uncertainty (for a full focal plane) of 2 arcseconds (OSS-REQ-0298) and the field rotation known to an accuracy of 32 arcseconds (LTS-206).

Reference Database: A full-sky astrometric and photometric reference catalog of stars derived either from an external dataset (e.g. Gaia) or from the Data Release Processing. Given the current Gaia data release timeline the initial reference catalog is expected to have an astrometric uncertainty of < 0.5 milliarcseconds and a photometric uncertainty of <20 millimag (for a $V = 19$ G2V star). The expected release of these calibration catalogs is 2018 and will be derived from the Gaia spectrophotometric observations of non-variable sources.

Calibration Images: Flat-field calibration images for all passbands and all CCDs appropriate for the time at which the observations were undertaken. No corrections will be made in the flat-fields for non-uniform pixel sizes - the flat-fields will correct to a common surface brightness. A flat SED will be assumed for all flat field corrections. Fringe frame calibration images scaled to an amplitude derived from the sky background (i.e. no sky spectrum will be available).

Image Metadata: List of the positions and extents of CCD defects for all CCDs within the focal plane; electronic parameters for all CCDs (saturation limits, readnoise parameters), electronic and physical footprint for the CCDs, linearity functions, models for the variation in the PSF width with source brightness (brighter-fatter), and parameterized models for a component-based WCS (e.g. a series of optical distortion models) as needed.

3.1.2 Output Data

CalExp Images: A calibrated exposure (CalExp) is an Exposure object. The CalExp contains the image pixel values, a variance image, a bitwise mask, a representation of the PSF, the WCS (possibly decomposed into separable components), a photometric calibration object, and a model for the background. For the alert production, it is not anticipated that a model of the per-pixel covariance will be persisted but this will be revisited dependent on the performance of image subtraction and anomaly characterization as described in 3.2.

Source Databases: A catalog of `Sources` with measured features described in 3.1.5.

OCS Database A parameterization of the PSF, WCS, photometric zero-point, and depth for each CCD in a visit. The PSF may be a simplified version (e.g. a single Gaussian) of that derived for the Alert production. These data will be made available to the Telescope Control System (TCS) to assess the success of each observation. A limited version of nightly SFM could be run on the summit to generate this information or the data will be persisted within a database at the data center that will be accessible to the TCS.

3.1.3 Instrumental Signature Removal

Instrumental Signature Removal characterizes, corrects, interpolates and flags the camera (or raw) amplifier images to generate a flat-fielded and corrected full CCD exposure.

3.1.3.1 Pipeline Tasks

- Mask the image defects at the amplifier level based on the CCD defect lists, and the per CCD saturation limits
- Assemble the amplifiers into a single frame (masking missing amplifiers)
- Apply full frame corrections: dark current correction, flat field to preserve surface brightness, fringe corrections. Flat fields will assume a flat spectral energy distribution (SED) for the source. Fringe frames will be normalized by fitting to the observed sky background.
- Apply pixel level corrections: apply a correction model for brighter-fatter to homogenize the PSF, correct for static pixel size effects based on a model
- Interpolate across defects and saturated pixels assuming a model for the PSF (with a nominal FWHM). An estimate of the PSF will be needed for this operation (from the TCS/OCS) or interpolation may be needed to be performed at the end of 3.1.4.
- Apply a cosmic ray detection algorithm as described in 8.3.1
- Generate a summed and difference image from the individual snaps propagating the union of the mask pixels in each snap

Dependent on the properties of the delivered LSST image quality for 15 second snaps it may be required to model any bulk motion between snaps prior to combination (e.g. if dome seeing or the ground layer dominate the lower order components of the seeing).

3.1.4 PSF and background determination

Given exposures that have been processed through Instrument Signature Removal, `Sources` must be detected to determine the astrometric and photometric calibration of the images. As noted previously an iterative procedure will be adopted to generate an estimate of the background and PSF, and to characterize the properties of the detected sources. Convergence criteria for this procedure are not currently defined. The default implementation assumes three iterations.

3.1.4.1 Pipeline Tasks

The iterative process for PSF and background estimation comprises,

- Background estimation on the scale of a single CCD is as described in 8.9, which divides the CCD into subregions and estimates the background using a robust mean from non-source pixels.
- Subtraction of the background and the detection of sources as described in 8.5. The initial detection threshold for source detection will be 5σ, with σ estimated from variance image plane.
- Measurement of the properties of the detected sources (see 3.1.5). Dependent on the density of sources it may be necessary to deblend the images as described in 8.6
- Selection of isolated PSF candidate stars based on a signal-to-noise threshold (default 50 σ). This threshold is significantly deeper than the magnitude limit for Gaia astrometric catalogs but is the threshold at which the astrometric error on the centroid due to photon noise is less than 10 mas and the photometric noise is less than 2% for the case of the use of a deeper DRP derived reference catalog.
- Single CCD PSF determination using the techniques described in 8.11.1 and the selected bright sources

- Masking of source pixels within the CCD (growing the footprint of the `Sources` to mask the outer regions of the `Source` profiles will likely be required to exclude contributes to the background from low surface brightness features).

The default expectation is that all tasks within this procedure would iterate until convergence. There maybe significant speed optimizations to be gained by excluding the `Source` detection step after an initial detection if the number of sources does not change significantly with updates to the background model.

3.1.5 Source measurement

For the `Source` catalog generated in 3.1.4, source properties are measured using a subset of features described in 8.7. Source measurement is for all sources within the `Source` catalog and not just the bright subset used to calibrate the PSF. We anticipate using the following plugin algorithms within the `Source` measurement step,

- Centroids based on a static PSF model fit (see 8.7.2.1 and 8.7.2.5)
- Aggregation of pixel flags as described in 8.7.2.2
- Aperture Photometry as geven in 8.7.2.4 (but only for one or two radii)
- PSF photometry given in 8.7.2.5 assuming a static PSF model fit
- An aperture correction estimated assuming a static PSF model and measurement of the curve of growth for detected sources as given in 8.12

3.1.6 Photometric and Astrometric calibration

Photometric and astrometric calibration entails a "semi-blind" cross match (because the pointing of the telescope is known to an accuracy of 2 arcseconds) of a reference catalog derived either from the DRP `Objects` or from an external catalog (see 3.1.1), the generation of a WCS (on the scale of a CCD or full focal plane), and the generation of a photometric zeropoint (on the scale of a CCD). These algorithms must degrade gracefully for the case of larger pointing errors

(e.g. during the initial calibration of the system during commissioning) and may need to operate in a "blind" mode where the pointing and orientation of the telescope is not known.

3.1.6.1 Pipeline Tasks

The photometric and astrometric calibration is expected to be performed at the scale of a single CCD. It is possible that the calibration process will need to be extended to larger scales (up to a full focal plane) if there is significant structure in the photometric zero point, or if astrometric distortions cannot be calibrated at the scale of the CCD with sufficient accuracy (i.e. the astrometric distortions do not dominate the false positives in the image subtraction). A full focal plane level calibration strategy will introduce synchronization points within the processing of the CCDs as the detections on all CCDs will need to be aggregated prior to the astrometric fit.

The procedures used to match and calibrate the data are,

- CCD level source association between the DRP reference catalog (or external catalog) and `Sources` detected during the PSF and background estimation stage will use a simplified Optimistic B approach described in 8.23.1. Given an astrometric accuracy of < 0.5 milliarcseconds from external catalogs such as Gaia (for a $V = 19$ G2V star) or an accuracy of < 50 milliarcseconds for the DRP catalogs the search radii for sources will be dominated by the uncertainties in the pointing of the telescope and the rotation angle of the camera.

- Generation of a photometric solution on the scale of a single CCD as described in 8.14.1

- Fitting of a WCS astrometric model for a single CCD using the algorithms given in 8.13.1. The WCS model is expected to be composed of a sum of transforms or astrometric components (e.g. a optical model for the telescope, a lookup table or model for sensor effects such as tree rings).

- Persistance of the astrometric, PSF, and photometric solutions for possible use by the Telescope Control system (TCS) (see 3.1.2)

Given the number of stars available on a CCD or the complexity of the astrometric solutions for the LSST (e.g. the decomposition of the WCS

into components) it may be necessary that the astrometric and photometric solutions be performed for a full focal plane and not just a CCD. For these cases the algorithms used will be single visit matching (see 8.23.2), single visit photometric solutions (see 8.14.1), and single visit astrometric fits (see 8.13.2). Fitting to a full focal plane introduces a synchronization point in the alert processing where all CCDs must have completed their previous processing steps prior to the astrometric calibration.

Astrometric and photometric solutions within crowded fields will utilize the bright and easily isolated sources within a CCD image. The order of the WCS used in the astrometric fits will, therefore, depend on the number of calibration `Sources` that are available.

3.2 Alert Generation Pipeline (WBS 02C.03.04)

The Alert Generation pipeline identifies variable, moving, and transient sources within a calibrated exposure by subtracting a deeper template image (see Figure 5). The `DIASources` detected on a DiffExp are associated with known `DIAObjects` and `SSObjects` (that have been propagated to the date of the CalExp exposure) and their properties measured. The process for image differencing requires the creation or retrieval of a TemplateCoadd, the matching of the astrometry and PSF of the TemplateCoadd to a CalExp, and subtracting the template image from the CalExp. Spurious `DIASources` will be removed using morphological and environment based classification algorithms.

The Alert Generation pipeline is required to difference, and detect and characterize `DIASource` sources within 24s (allowing for multiple cores and multithreading of the processing).

3.2.1 Input Data

CalExp Images: Calibrated exposure processed through 3.1 with associated WCS, PSF, mask, variance, and background estimation.

Coadd Images: TemplateCoadd images that spatially overlap with the CalExp images processed through 3.1. This coadded image is optimized for image subtraction and is expected to be characterized in terms of a tract/patch/filter. Generation of this template may account for differential

chromatic refraction or be generated for a limited range of airmass, seeing, and parallactic angles.

Object Databases: `Objects` that spatially overlap with the CalExp images processed through 3.1. This `Object` catalog will provide the source list for determining nearest neighbors to the detected `DIASources`.

DIAObject Databases: `DIAObjects` that spatially overlap with the CalExp images processed through 3.1. This `DIAObject` catalog will provide the association list against which the `DIASources` will be matched.

SSObject Databases: The `SSObject` list at the time of the observation. The `SSObject` positions will be propagated to the date of the CalExp observations and will provide an association list for cross-matching against the detected `DIASources` to identify known Solar System objects.

Reference classification catalogs: Classification of `DIASources` based on their morphological features (and possibly estimates of the local density or environment associated with the `DIASource`) will be undertaken prior to association in order to reduce the number of false positives. The data structures that define these classifications will be required as an input to this spuriousness analysis.

3.2.2 Output Data

DiffExp Images: Image differences derived by subtracting a Template-Coadd from a CalExp image.

DIASource Databases: `DIASources` detected and measured from the DiffExps using the set of parameters described in Table **??** will be persisted.

DIAObject Databases: `DIASource` will be associated with existing `DIAObjects` and persisted. New `DIASource` (i.e. those not associated) will generate a new instance of a `DIAObject`.

3.2.3 Template Generation

Template generation requires the creation or retrieval (see 8.15) of a TemplateCoadd that is matched to the position and spatial extent of the input CalExp. Generation of the TemplateCoadd could be from a persisted Coadd that was generated from CalExp exposures with comparable (within a predefined tolerance) airmass and parallactic angles, or from a model that corrects for the effect of differential chromatic refraction (see 8.18). It is expected that these operations would be undertaken on a CCD level but for efficiency the TemplateCoadd might be returned for a full focal plane or a series of *patches* or a *tract*.

3.2.3.1 Pipeline Tasks

- Query for a TemplateCoadd images that are within a given time interval of the CalExp (default 2 years) of the current CCD image, and are within a specified airmass and parallactic angle.
- (optional) Derive a seeing and DCR corrected TemplateCoadd from a model (see DCR template generation in 8.18). The current prototype approach assumes that the TemplateCoadd will be derived for the zenith and will comprise a data cube with spatial and wavelength dimensions (a low resolution spectrum per pixel). Propagating to the observation will require aligning the DCR correction in the direction of the parallactic angle of the CalExp.

3.2.4 Image differencing

Image differencing incorporates the matching of a TemplateCoadd to a CalExp (astrometricly and in terms of image quality), subtraction of the template image, detection and measurement of `DIASources`, removal of spurious `DIASources`, and association of the `DIASources` with previously identified `DIAObjects`, and `SSObjects`.

3.2.4.1 Pipeline Tasks

- Determine a relative astrometric solution from the WCS of the TemplateCoadd image and CalExp image

- Match the DRP `Sources` for the TemplateCoadd (see 5.1.4) against `Sources` from the SFM pipeline (see 3.1) of the raw images.
- Warp or resample the TemplateCoadd using a Lanczos filter (as described in 9.20) to match the astrometry of the CalExp. It is possible that astrometricly matching the TemplateCoadd and CalExp using faint source will need to be undertaken dependent on the accuracy of the WCS.
- For CalExp images with an image quality that is better than the TemplateCoadd preconvolve the CalExp image with the PSF. Use a convolution kernel (see 9.10) that is matched to the source detection kernel. This reduces the need for deconvolution in the PSF matching (see 8.16.1)
- Match the PSF of the CalExp and TemplateCoadd images as described in 8.19.1 and construct a spatial model for the matching kernel. This approach may include matching to a common PSF through homogenization of the PSF (see 8.16.2.
- Apply the matching kernel to the TempCoadd and subtract the images to generate a DiffExp (as described in 8.16.1). Dependent on the relative signal-to-noise in the science and template image decorrelation of the template image due to the convolution of the template with a matching kernel may be necessary (see 8.19.1)
- Detect `DIASources` on the DiffExp using the algorithms described in 8.5. Convolution with a detection kernel will depend on whether the CalExp was preconvolved in item 4.
- Measurements of the `DIASources` on the DiffExp will include dipole models and trailed PSF models (see 8.7.2.11 and 8.7.2.10 and parameters described in Table 2 of the DPDD. The specific algorithms used for measurement of `DIASources` will depend on whether the CalExp image was preconvolved.
- Measurement of the PSF flux on snap difference images for all `DIASources`.
- The application of spuriousness algorithms, also known as “real-bogus”, may be applied at this time dependent on whether the number of

false positives is less than 50% of the detected sources (see 8.7.2.12)[3]. `DIASources` classified as spurious at this stage may not be persisted (dependent on the density of the false positives). The default technique will be based on a trained random forest classifier. It is likely that the training of this classifier will need to be conditioned on the image quality and airmass of the observations.

3.2.5 Source Association

In Source Association `DIASources` detected within a given CCD will be cross-matched or associated (see 8.23.4) with the `DIAObject` table and the `SSObjects` (whose ephemerides have been generated for the time of the current observation). The association will be probabilistic and account for the uncertainties within the positions. The association may include flux and priors on expected proper motions for the sources. External targets (e.g. well localized transient events from other telescopes or instruments) can be incorporated within this component of the nightly pipeline (essentially treating external sources as additional `DIAObjects` and associating them with the `DIASources`) enabling either matching to `DIASources` or generation of forced photometry at the position of the external source.

3.2.5.1 Pipeline Tasks

- Generate the positions of `SSObjects` that overlap a DiffExp given its observation time by propagating the `SSObject` orbit (see 8.25)
- As described in 8.23.4 source association will be undertaken for all `DIASources`. Matching will be to `DIAObjects`, and the ephemerides of `SSObjects`. Positions for `DIAObjects` will be based on a a time windowed (default 30 day) average of the `DIASources` that make up the `DIAObject`. A linear motion model for parallax and proper motion will be applied to propagate the `DIAObject` to the time of the observation. A probabilistic association may need to account for one-to-many and many-to-one associations. In dense regions it may be necessary to generate joint associations across all `DIAObjects` (and associated `DIASources`) in the local vicinity of a `DIASource` to correct for mis-assignment from

[3]The requirement for a 50% false positive rate is given in the XXX and impacts the sizing model for the alert stream

previous observations. This could include the pruning and reassignment of `DIASources` between `DIAObjects`. A baseline approach for nightly processing will be to select based on a maximum a posteriori estimate for the association.

- `DIASources` will be positionally matched to the nearest 3 stars and 3 galaxies in the DRP `Object` database. In its simplest case the search algorithm will be a tree-based nearest neighbor search (the default radius for association is not defined) . The matched `Objects` will be persisted as a measure of local environment.
- `DIASources` unassociated with a `DIAObject` will instantiate a new `DIAObject`.
- The aggregate positions for the `DIAObjects` will be updated based on a rolling time window (default 30 days).
- Proper motion and parallax of the `DIAObject` will be updated using a linear model as described in 8.22.

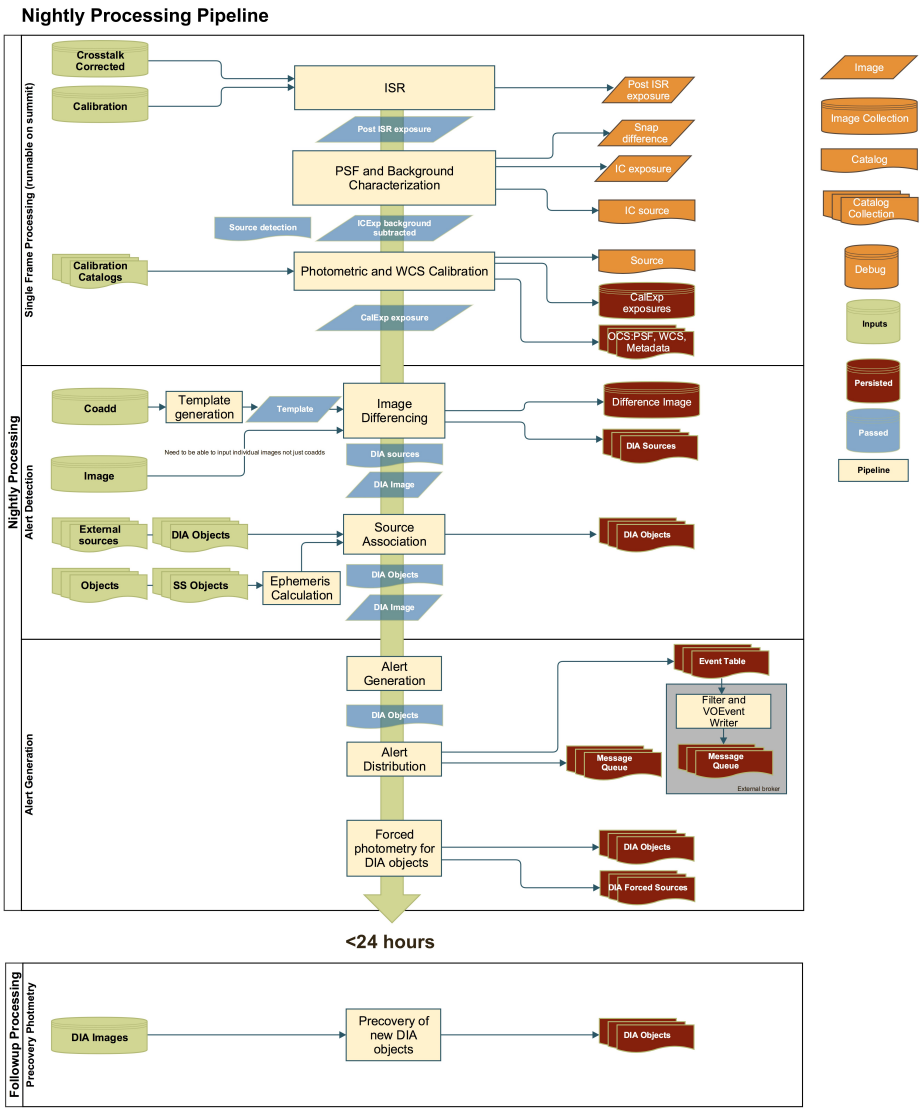


Figure 3: The alert production flow of data through the processing pipelines (single frame processing, alert generation, alert distribution, precovery photometry)

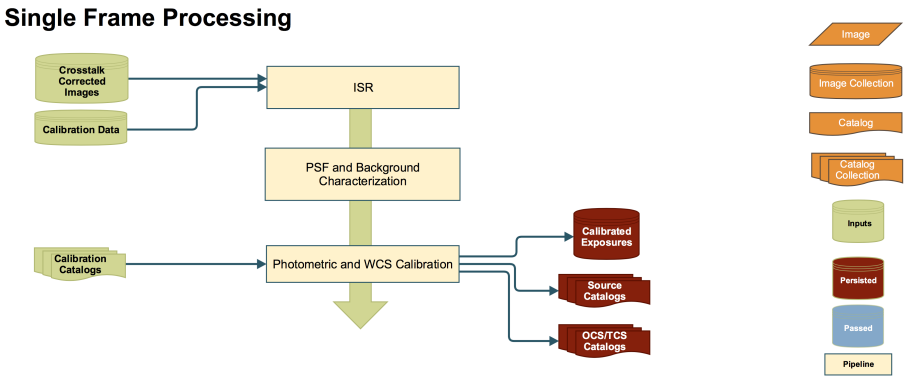


Figure 4: Single frame processing of the nightly data: instrument signature removal, astrometric and photometric calibration, background and PSF estimation from the cross-talk corrected camera images.

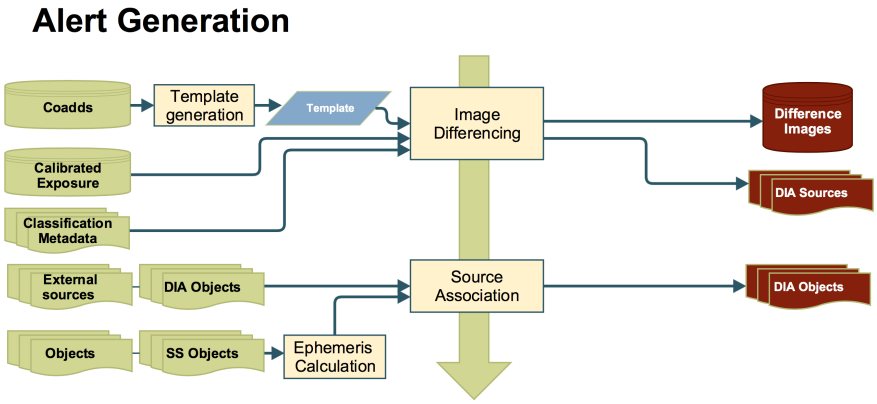


Figure 5: Generation of alerts from the nightly data: image differencing and measurement of the properties of the `DIASources`, identification and filtering of spurious events, association of previously detected `DIAObjects` and `SSObjects` with the newly detected `DIASources`.

3.3 Alert Distribution Pipeline (WBS 02C.03.03)

The Alert Distribution Pipeline takes the newly discovered `DIAObjects` (including their associated historical observations) and all related metadata as described in the DPDD, and delivers alert packets in `VOEvent` format to a variety of endpoints via standard IVOA protocols (eg., `VOEvent` Transport Protocol; VTP). Packaging of the event will include the generation of postage stamp cutouts (30x30 pixels on average) for the difference image and the template image together with the variance and mask pixels for these cutouts.

On average, each LSST visit will result in $\sim 2M$ alerts per night. This number includes both real and spurious detections and will be highly dependent on where we look on the sky. Because of this, the SRD requires that the design of the LSST alert system should be able to handle 10^7 events per night, which corresponds to 10^4 alerts per visit or 50 alerts per CCD (with each visit lasting 39 seconds). All alerts should be transmitted within 60s of the closure of the shutter of the final snap within a visit.

For a nightly event rate of 10^7, and assuming the schema described in Tables 1 and 2 in the DPDD together with the generation of the postage stamp cutouts, the compressed `VOEvents` data stream amounts to approximately 600GB of data per night (assuming no filtering of the data). The Alert Distribution pipeline is designed to distribute these alerts with a workflow, including the access point of external event brokers, shown in Figure 6.

In addition to the full data stream the Alert Generation Pipeline will provide a basic alert filtering service. This service will run at the LSST U.S. Archive Center (at NCSA). It will enable astronomers to create filters (see 3.3.4) that limit what alerts, and what fields from those alerts, are ultimately forwarded to them. These *user defined filters* will be configurable with a simplified SQL-like declarative language. Access to this filtering service will require authentication by a user.

`VOEvent` alerts will be persisted in an alert database as well as distributed through a message queue. The alert database (AlertDB) will be synchronized at least once every 24 hours and will be queriable by external users. The message queue that distributes the alerts is expected to have the capability to replay events for the case of a break in the network connection between the queue and client but not to support general queries.

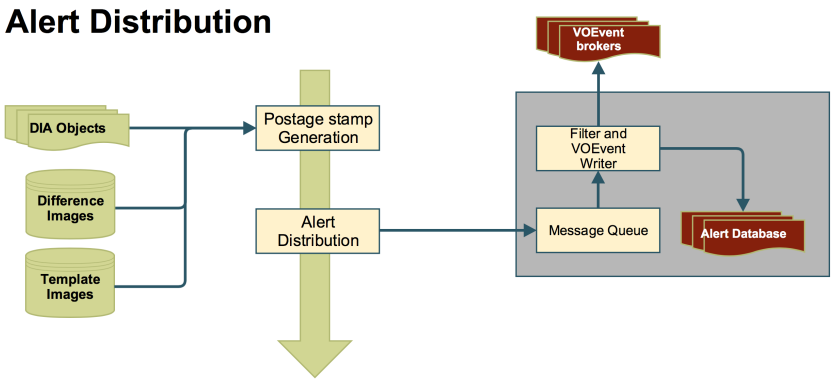


Figure 6: Distribution of alerts from the nightly processing: generation of postage stamps around each detected `DIASource`, distribution of the `DIAObjects` as VOEvents, simple filtering of the event stream, and persistence of the events in a database.

3.3.1 Input Data

DIAObject Database: `DIAObjects`, with new `DIASources`, generated through image differencing will be used to create alert packets.

Difference Images: The DiffExp will be used to generate postage stamp (cut-out) images of `DIASources` within the CCD.

Coadd Images: The TemplateCoadd used in image subtraction will be used to generate postage stamp images of the template image for `DIAObjects`.

3.3.2 Output Data

`VOEvent` Database: `VOEvents` generated from the `DIAObjects` and cutouts will be persisted within a database (e.g. a noSQL database) or object store.

3.3.3 Alert postage stamp generation

Creates the associated image cutouts (30x30 pixels on average) for all detect `DIAObjects` (cutouts are generated from the current observation and not from historical observations).

3.3.3.1 Pipeline Tasks

- Extract from the DiffExp the cutout of each `DIAObject` with a `DIASource` detected within the current observation. Cutout images will be scaled to the size of the `DIASource` but on average will be 30x30 pixels. Variance and mask planes, WCS, background model, and associated metadata will be persisted. The prototype implementation assumes that these cutouts will be persisted as FITS images with a projection that is the native projection of the DiffExps.
- Extract from the TemplateCoadd a cutout of each `DIAObject` with a `DIASource` detected within the current observation. Cutout images will be identical in size and footprint as those derived from the DiffExp. Variance and mask planes, WCS, and associated metadata will be extracted with the pixel data. The prototype implementation assumes that these cutouts will be persisted as FITS imagesand that the projection will be that of the DiffExps.

3.3.4 Alert queuing and persistance

The alert queue distributes and persists `DIAObject` with new `DIASources` as `VOEvents` through a message queue. It includes a *limited* filtering interface but persists the full `VOEvents` in an AlertDB. The event message stream and the AlertDB will be synchronized at least once every 24 hours.

3.3.4.1 Pipeline Tasks

- Publish `DIAObjects` to a caching message queue (e.g. Apache Kafka) through the butler. The prototype implementation assumes a distributed and partitioned messaging system that uses a publication-subscription model for communication between clients and the queue. This model maintains feeds of messages in categories called topics. An example topic would be a `DIAObject`. Whether a topic would comprise a full `DIAObject` or a subset of the data remains open (passing subsets of parameters as individual topics would require that the client be able

to synchronize and join topics into a full `DIAObject`). For each of the 189 CCDs, approximately 50 events will be passed as messages to the messaging queuing system. The distribution of the events from a given CCD will not be synchronized with other CCDs within the focal plane (alerts from each CCD will be independently processed).

- A consumer layer will subscribe to the message queue and package them as `VOEvents` and distribute these events to external users. To allow for network outages between the message queue and the consumer the message queue must be able to replay previous events.

- The consumer layer will provide a command line API to define simple queries or filters of the events (limited to querying on existing `DIAObject` fields, or filtering the attributes of the `DIAObject`). Web-based interfaces to the consumer layer will be developed by SUIT.

- Filtered or the full stream of `DIAObjects` will be packaged into `VOEvents` and broadcast to VOEvent clients through the consumer layer

- A full, unfiltered, VOEvent alert stream will be broadcast to the AlertDB using the consumer layer.

- Prior to the start of the subsequent night's observations, the message queue will be flushed and synchronized with the AlertDB. It is possible to persist the message queue on longer timescale but it is a requirement hat synchronization be performed within 24 hours of the observations.

To cope with the variation in density of events as a function of position on the sky and the need for fault tolerance the message queue will need to be able to partition and replicate data. Given the 600GB of data generated per night from the alert distribution, each full `DIAObject` stream will require about 0.1 Gb/s network capacity. Whether the consumer layer will instantiate a new consumer for each filter (or client) or will orchestrate the `VOEvents` from a single subscription to the message queue is an open question that will depend on the expected network topology (internal and external to the data center at NCSA).

The AlertDB will have an interface that can be queried (to enable historical searches of events) including searches on other than timestamps. It is expected that the AlertDB will be a noSQL datastore (e.g. Cassandra).

3.4 Precovery and Forced Photometry Pipeline

The precovery and forced photometry pipeline performs two tasks (see Figure 7). Forced PSF photometry is undertaken for all `DIAObjects` that have a detected `DIASource` within a, default 1 year, window of time from the observation. Second, within 24 hours, precovery forced photometry is performed on all unassociated `DIASources` within an image (i.e. new `DIAObjects`). For each new `DIAObject`, forced (PSF) photometry will be measured at the position of the source in each of the preceding 30-days of DiffExps.

Forced photometry is not required prior to alert generation. Completion of the precovery photometry is required within 24 hours of the completion of the observations. Forced and precovery can be undertaken as part of the nightly workflow if they do not impact the time required to distribute the alerts.

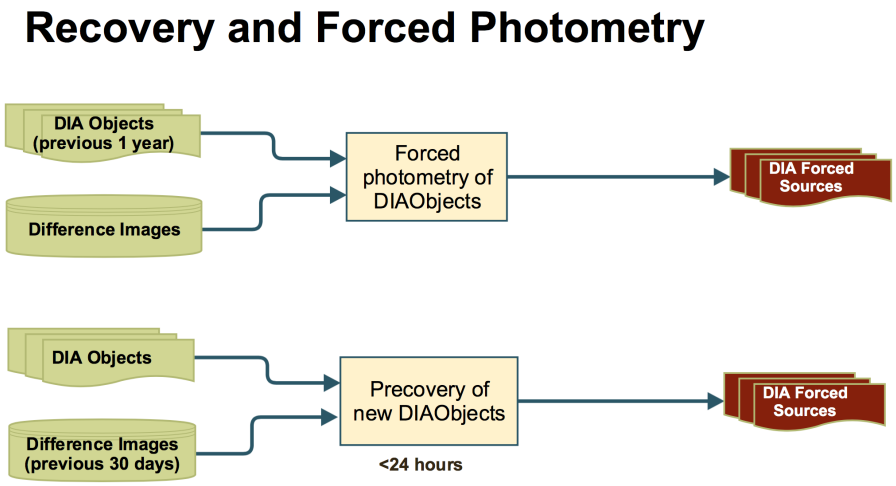


Figure 7: Forced photometry for `DIAObjects`: forced photometry on a night's DiffExp for all `DIAObjects` that have detected `DIASources` within the last year, precovery photometry for the previous 30 days of DiffExps for new `DIAObjects`

[**Note:**
For ZI: I moved the precovery to a single pipeline]

3.4.1 Input Data

Difference images: A cache of DiffExps within a finite time interval (default 30 days) of the previous nights observations (inclusive of the previous nights data)

DIAObject Database: All `DIAObjects` with a `DIASource` detection within the last 12 months and all unassociated (new) `DIAObjects` observed within the previous night

3.4.2 Output Data

DIAForcedSource Databases: Forced PSF photometry at the centroid (from the aggregated individual `DIASource` centroids) of a `DIAObject`. The forced photometry is undertaken on the current night's DiffExp for all `DIAObjects` with `DIASources` detected within the last year, and on the previous 30 days of DiffExp for all newly detected `DIASources`.

3.4.3 Forced Photometry on all `DIAObjects`

Generate forced (PSF) photometry on the DiffExp for all `DIAObjects` that overlap with the footprint of the CCD. Forced photometry is only generated for `DIAObjects` for which there has been a `DIASource` detection within the last 12 months. The forced photometry is persisted in the forced photometry table in the Level 1 database. Alerts are released prior to the generation of forced photometry and forced photometry is not released as apart of an alert which means that this component of the processing is not subject to the 60 second processing requirements for nightly processing.

3.4.3.1 Pipeline Tasks

- Extract all `DIAObjects` within the Level 1 database with a detected `DIASource` within the last year (including the current nights observations). This information is available from the `DIASource` and `DIAObject` association.

- For the aggregate positions within the `DIAObject` undertake a PSF forced measurement as described in section 8.7.1.5
- Update the forced photometry tables in the Level 1 database.

3.4.4 DIAObject Forced Photometry:

Updated forced photometry table for all new `DIAObjects`

3.4.4.1 Pipeline Tasks

- Extract from the Level 1 database all `DIAObjects` that were unassociated (i.e. new `DIASource` detections) from the previous nights reduction. Filtering of the `DIAObjects` will need to account for cases where new `DIASources` are observed more than once within a night (where the second or subsequent observations do not result in a new `DIAObject`).
- Extract DiffExps within a default 30 day window prior to the observation
- Force photometer the extracted images as described in 8.7.1.5 using a PSF model and the centroid defined in the `DIAObject`
- Update the forced photometry table within the Level 1 database

3.5 Moving Object Pipeline (WBS 02C.03.06)

The Moving Object Pipeline (MOPS) is responsible for generating and managing the Solar System 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 criteria specified in the OSS (i.e. the observations required to define an orbit). Each visit within 10 degrees of the Ecliptic will detect approximately 4,000 asteroids.

Components of MOPS are run during and separately from nightly processing (see Figure 8). MOPS for nightly processing is described in 3.2.5 as part of source association. "Day MOPS" processes newly detected `DIAObjects` to search for candidate asteroid tracks. The procedure for Day-MOPS is to link `DIASource` detections within a night (called tracklets), to link these tracklets across multiple nights (into tracks), to fit the tracks with an orbital model to identify those tracks that are consistent with an asteroid orbit, to match these new orbits with existing `SSObjects`, and to update the `SSObject` table. By its nature this process is iterative with `DIASources` being associated and disassociated with `SSObjects`. It is expected that a frequency of one day for these iterations (i.e. the `SSObjects` will be update each day) will be sufficient.

3.5.1 Input Data

DIAObject Database: Unassociated `DIASources` from the previous night of observing. This means `DIAObjects` that were newly created during the previous night because they could not be associated with known `DIAObjects`. `DIASources` associated with an `SSObject` in the night are still passed through the MOPS machinery

SSObject Database: The catalog of known solar system sources

Exposure Metadata: A description of the footprint of the observations including the positions of bright stars or a model for the detection threshold as a function of position on the sky (including gaps between chips)

3.5.2 Output Data

SSObject Database: An updated `SSObject` database with `SSObjects` both added and pruned as the orbital fits are refined

DIASource Database: A updated `DIASource` database with `DIASources` assigned and unassigned to `SSObjects`

Tracklet Database: A temporary database of tracklets measured during a night. This database will be persisted for at least a lunation.

3.5.3 Tracklet identification

From multiple visits within a night, link unassociated `DIASources` to form tuples (or n-tuples) of `DIASources`

3.5.3.1 Pipeline Tasks

- Extract unassociated `DIASources` from the Level 1 database
- Link `DIASources` into tracklets assuming a maximum velocity for the moving sources. The maximum velocity will be based on a prior as described in [39]. For each tracklet a velocity vector will be calculated to enable pruning or merging of degenerate tracklets within a data set.
- Merge tracklets by clustering in velocity and position (propagated to a common visit time). Tracklets can contain multiple points and all permutations of the asteroid tuples will be stored. In the process of merging tracklets `DIASources` that are not a good fit for the merged tracklet will be remove and their associated tracklets returned to the tracklet database. Moving or trailed sources will incorporate the position angle of the source when linking. Details of the implementation of the `DIASource` linkage is described in 8.26
- Temporarily persist a database of tracklets. This database will be required for at least 30 days of data but, depending on resources available, may persist for longer.

3.5.4 Precovery and merging of tracklets

Tracklets are matched and merged with existing `SSObjects` and removed from the Tracklet database. This culls any tracklets or `DIASources` that obviously belong to an existing `SSObject` from the rest of the processing.

3.5.4.1 Pipeline Tasks

- Return all tracklets identified within a given night of observations
- Return the footprints of each visit and the time of the observation
- Extract `SSObjects` from the `SSObject` database and propagated those orbits to the position and time of a visit. Details of this orbit propagation for precovery are described in 8.25.
- Merge (precovery) the tracklets with the projected `SSObject` trajectories and refit the `SSObject` orbit model. `DIASources` previously associated with an `SSObject` may no longer fit the updated `SSObject` orbits. These `DIASources` will be removed from the `SSObject` and returned as unassociated `DIAObjects` to the level 1 database. All tracklets associated with these `DIAObjects` will be returned to the tracklet database. Details of this attribution and precovery are described in 8.27

3.5.5 Linking tracklets and orbit fitting

Given a database of tracklets constructed from a window (default 30 days) of time, link the tracklets into tracks assuming a quadratic approximation to the trajectory. Fit these tracks with orbital models and update the `SSObject` database.

3.5.5.1 Pipeline Tasks

- Extract all tracklets from the tracklet database for a specified window in time (default 30 days)
- Merge tracklets into tracks based on their velocities and accelerations. Candidate tracks are pruned by fitting a quadratic relation to the positions (after applying a topocentric correction to the positions of the sources). Efficiency in this matching procedure is provided by a spatial index such as a kd-tree (see 8.28).
- Fit an orbit to each candidate track using a tool such as OOrb (https://github.com/oorb/oorb) and, for poorly fitting points, return the `DIASources` and associated tracklets to their respective databases for subsequent reprocessing.

- Merge SSObjects that have similar orbital parameters based on range searches within the six dimensional orbital parameter space. Merged SSObjects will need to be refit and any poorly fitting DIASources (and associated tracklets) returned to their respective databases for subsequent reprocessing. Details of this procedure are given in 8.29

3.5.6 Global precovery

For all new or updated SSObjects propagate the orbits to the positions and times of the observations of all tracklets and orphan DIAObjects to "precover" further support for the orbits. This will prune the number of tracklets and DIAObjects that will require merging in subsequent observations.

3.5.6.1 Pipeline Tasks

- Return all tracklets identified within a given night of observations
- Return the footprints of each visit and the time of the observation
- Extract orbits for all new or updated SSObjects and propagate the positions to the times of the observations for all visits covering the extent of the tracklet database, default 30 days, (see 8.25)
- Merge the tracklets with the projected SSObject positions and refit the SSObject orbit model. Poorly fitting DIASources (and associated tracklets) will be removed from the SSObject and returned as unassociated DIAObjects to the Level 1 database (as described in 8.27).

The process for precovery and updating of the SSObject models is naturally iterative (given the pruning of poorly fitting DIAObjects and tracklets). Updates of the SSObjects as part of each night of operations should enable sufficient iterations without requiring Day-MOPS to be rerun multiple times per day. The computationally expensive operations in this pipeline are the orbit propagation and the orbit fitting. Resources required for orbit propagation could be reduced be removing the initial precovery stage but at the cost of increasing the number of tracklets that would be available for matching into tracks. Orbital trajectories could be pre-calculated and modelled as polynomials to enable fast interpolation during Day-MOPS.

Extending the Global Precovery to include singleton DIASources (i.e. one that are not merged into tracklets) would enable the identification of asteroids

at the edge of the nightly footprint (where an object moves outside of the nightly survey footprint prior to the second visit or a second visit is not obtained for a given field).

3.5.7 Prototype Implementation

Prototype MOPS codes are available at https://github.com/lsst/mops_daymops and https://github.com/lsst/mops_nightmops. Current DayMOPS prototype already performs within the computational envelope envisioned for LSST Operations, though it does not yet reach the required completeness requirement.

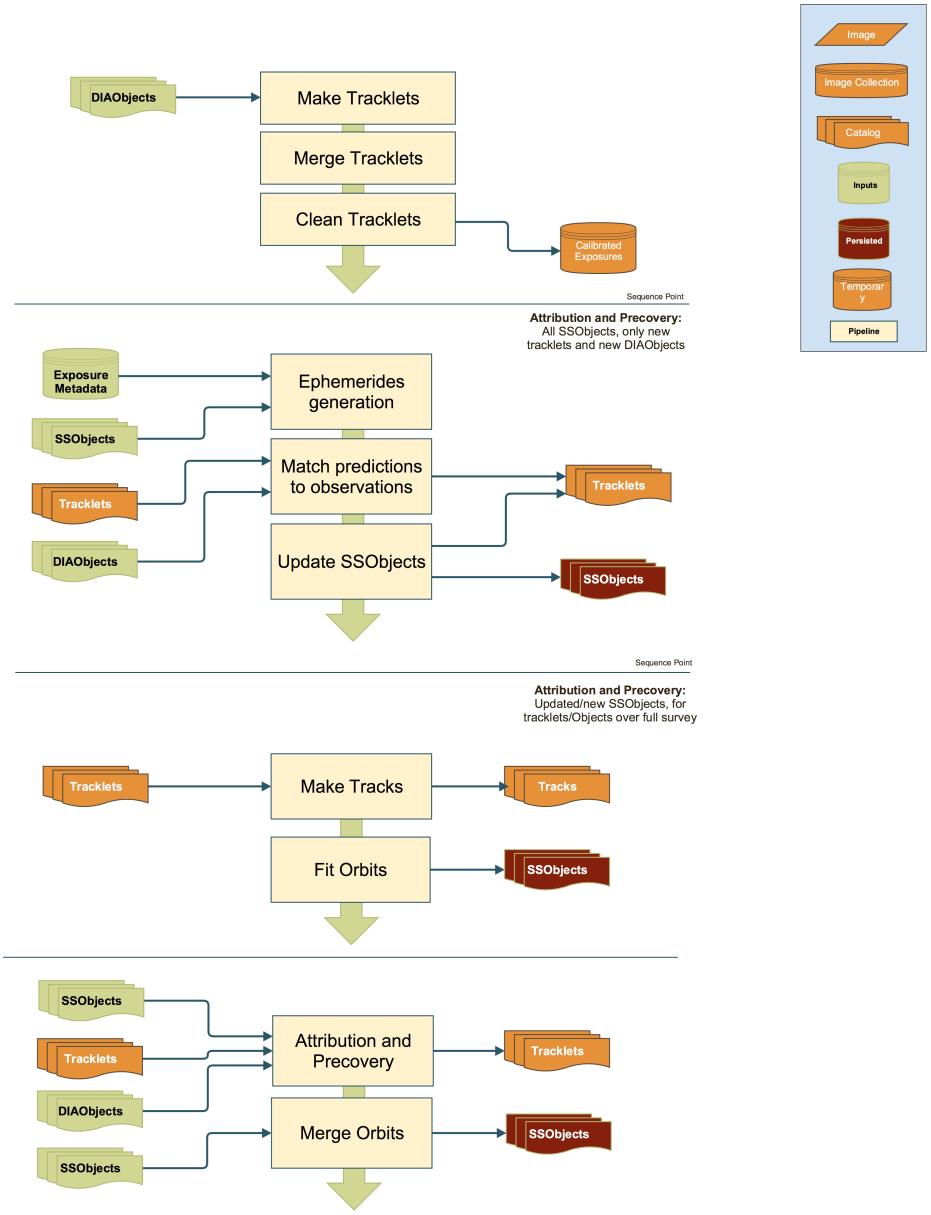


Figure 8: Detection and orbital modelling of moving sources within the nightly data: Tracklet generation from revisits, filtering of tracklets based on known `SSObjects`, fitting of tracks and orbits to tracklets, pruning of tracklets and `DIAObjects` based on new and updated `SSObjects`.

4 Calibration Products Pipeline

This section details the input data and algorithms required to generate all data products necessary for the photometric calibration of the LSST survey. Details of the application of these products is covered in other sections of this document.

The details of the input datasets are given in §4.2 and §4.5, which define the source of these data, *i.e.* which will be provided by the camera team and which will be measured on the mountain. §4.4 covers miscellaneous algorithms which will be necessary to develop in order to acquire the input datasets with the collimated beam projector (CBP). Finally, sections §4.3 and §4.6 list the various output data products from the Calibration Products Pipeline.

4.1 Key Requirements

The work performed in this WBS serves several 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, and emission from, the atmosphere;
- It will characterize 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 provide updated values of the crosstalk matrix to the camera DAQ (for AP) and DM (for DRP) for correction of the raw data;
- It will allow for characterization of the optical ghosts and scattered light in the system.

4.2 Inputs

The following section lists the input datasets which will be available to the Calibration Products Pipeline. It should be noted that these are the raw

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

inputs, and as such, the algorithmic sections for items that are listed as camera team deliverables are shown as "None", as these will have been previously derived. However, many of these items are re-listed in the outputs section, where the algorithms to recalculate/monitor these on the mountain are defined.

4.2.1 Bias Frames

A set of bias frames used for the production of the master bias frame, obtained by acquiring many zero second exposures with the shutter remaining closed, taken at the normal LSST cadence.

• Algorithmic component: None - these just need to be taken.

4.2.2 Gain Values

Camera Team deliverable

The gain values for all amplifiers in the camera, in e^-/ADU; note that these are required to high accuracy (0.1%), as they are used in determination of the photometric flats.

• Algorithmic component: Both ^{55}Fe and PTC gain measurement techniques need to be applied with care to get good results. Given the "brighter-fatter effect" and non-linearity, it is not clear to what accuracy PTC can measure the gain, but ^{55}Fe can provide gain to $\ll$0.1% for small signal levels. ^{55}Fe flux measurement will use the standard stack source finding and flux measurement algorithms.

4.2.3 Linearity

Camera Team deliverable

The linearity curve for each amplifier in the camera, as well as the level above which these non-linearity curves should be considered unreliable.

• Algorithmic component: None.

4.2.4 Darks

Sets of long dark frames (~300s) with the actual exposure length optimized for the dark current in the delivered sensors, the delivered read-noise, and considering the trade-off against the integrated cosmic ray muon flux.

• Algorithmic component: None - these just need to be taken.

4.2.5 Crosstalk

Camera Team deliverable

The crosstalk matrix for every pair of amplifiers in the camera. It is worth noting that this is expected to be a very sparse.

• Algorithmic component: None.

4.2.6 Defect Map

Camera Team deliverable

A list of all bad (unusable) pixels in each CCD, as well as list of possibly suspect pixels, *i.e.* ones which should be flagged as such during processing.

• Algorithmic component: None.

4.2.7 Saturation levels

Camera Team deliverable

The lowest level (in electrons), for each amplifier, at which charge bleeds into the neighboring pixels. If necessary, they will also provide the level at which the serial register saturates (*i.e.* if the serial saturates at a lower level than the parallels).

• Algorithmic component: None.

4.2.8 Broadband Flats

Sets of flats taken through the standard LSST filters. Flats will be taken at a number of flux levels to measure the "brighter-fatter effect"coefficients and to check linearity, including sets of "superflats" - sets of high-flux flats with many repeats (> 50, possibly > 100). The superflats taken for "brighter-fatter effect"characterization will not need to be taken regularly as this effect is not expected to evolve with time.

• Algorithmic component: None - these just need to be taken.

4.2.9 Monochromatic Flats

Sets of 'monochromatic' (*c.* 1nm bandwidth and spacing) flat-field screen images taken with no filter/glass in the beam.

• Algorithmic component: None - these just need to be taken.

4.2.10 CBP Data

Sets of images taken with the Collimated Beam Projector (CBP). The proposed resolutions and steps in these datasets are preliminary. All CBP data will be processed using the standard LSST ISR, except without the application of flat-fielding. Standard LSST aperture photometry will then be used to measure the number of counts associated with each CBP spot.

• Algorithmic component: Scripting the CBP/8.4m to take each of these datasets in concert. The scripting/control requirements for the CBP are dealt with separately in §4.4.

4.2.10.1 CBP dataset 1 Sets of CBP images scanned in wavelength at 1nm resolution[5] every 1nm for a fixed set of spot positions on the camera, and for fixed footprint on M1. No filter should be in the beam.

4.2.10.2 CBP dataset 2 Sets of CBP images scanned in wavelength at 20nm bandwidth every 100nm, while rotating the CBP about a pupil to move the spot pattern around the camera for a fixed footprint on M1. No filter should be in the beam.

4.2.10.3 CBP dataset 3 Sets of CBP images scanned in wavelength at 20nm resolution every 100nm for a fixed set of spot positions on the camera, and for a number of footprints on M1; the minimum number of footprints is *c.* 6 for a 30cm CBP, but in reality the use of more pointings will be explored to test the assumption of azimuthal symmetry. No filter should be in the beam.

4.2.10.4 CBP dataset 4 Sets of CBP images scanned in wavelength at 1nm resolution every 1nm for a fixed set of spot positions on the camera, and for a fixed footprint on M1. Repeated for every filter. *N.b.* the wavelength range for each scan need only cover the range for which the filter transmits appreciable light.

4.2.10.5 CBP dataset 5 Sets of CBP images scanned in wavelength at 20nm resolution every 20nm for a fixed set of spot positions on the camera, and for fixed footprint on M1. Repeated for every filter.

4.2.10.6 CBP Crosstalk Measurement Sets of CBP images taken with a suitably-designed sparse mask to allow identification and measurement of all ghost images arising from electronic crosstalk. The simplest sparse mask would have only a single spot, used to illuminate each amplifier in the camera in turn (but less sparse solutions are likely also possible). The wavelengths used are unimportant, and there are no constraints on beam footprints on M1 or filter choice. This will be particularly necessary should LSST be operated

[5] 1nm 'resolution' here denotes the bandwith of the light source, and can be this width, or any amount lower. It should, however, be noted that the accuracy on the wavelength calibration of the light source needs to be at the 0.1nm level **XXX Add a reference to the DESC people showing that this is necessary for SN cosmology in LSST.**

in a slow-readout mode, for example for use with 30s integrations, as crosstalk coefficients would change considerably.

4.2.11 Filter Transmission

The transmission curves (transmission as a function of wavelength) for each filter, as a function of filter position. This is to be delivered by the filter vendors rather than the camera team, but is input data which will not be measured by DM. The required resolution is 1nm or better, in keeping with the resolution of the monochromatic flats.

• Algorithmic component: None.

> **Note:**
> We need to check what the proposed wavelength resolution and accuracy the vendors are proposing to use for this is. I spoke to Steve Ritz at the AHM and he seemed very positive about the vendors proposal for this, but we should check what the plan is.

4.2.12 Stellar spectra

One or more spectrophotometrically-calibrated star(s) in the field of view for every visit.

• Algorithmic component: How these will be chosen or produced is TBD.

4.2.13 Other stellar spectra (*n.b.* != 4.2.12)

Known spectra for bright stars in the field of view for all visits.

• Algorithmic component: How these will be chosen or produced is TBD.

4.2.14 Atmospheric Characterization

These are the external measurements of atmospheric parameters, *e.g.* the barometric pressure, ozone and temperature.

• Algorithmic component: Interfacing with the site team or parties responsible for the equipment, to automate obtaining the measurements in a machine-readable form, including the ozone data from satellites.

4.2.15 Photometric Standards

A set of photometric standard stars covering a range of colors lying within an appropriate magnitude range. The likely source of this data is GAIA. However, should this prove not to be a suitable source, a catalog would be carefully generated using the survey's most photometric data, utilizing an übercal/jointcal type approach.

• Algorithmic component: Constructing color transformations from the GAIA measurements, based on assumptions about the objects' intrinsic SEDs, *i.e.*

not using only color terms. In the case that GAIA does not provide the catalog, a process similar to the Forward Global Calibration Model implemented by Eli Rykoff and David Burke for DES would be used. The latter process would likely be able to share atmospheric modeling code with the reductions performed for the auxiliary/Calpyso/calibration telescope.

4.3 Outputs from the Calibration Product Pipelines == Inputs to the AP/DRP Pipelines

This section details the outputs from the Calibration Products Pipeline. Algorithms for the production of each item are defined, and includes provision for the re-calculation of items previously listed as "camera team deliverables".

4.3.1 Master Bias

A trimmed, overscan subtracted, master bias frame from the entire camera, produced by taking the median of several-to-many bias frames for each CCD on the focal plane.

• Algorithmic component: Given LSST's 2s readout, we do not expect to need to remove cosmic rays explicitly; a robust stacking algorithm should be sufficient. A prototype construction algorithm currently exists in `pipe_drivers`. The final version must be configurable to use scalar-, vector- or array-type overscan subtraction. [6]

4.3.2 Master Darks

A trimmed, overscan and bias-frame subtracted, master dark frame for each CCD on the focal plane. These are produced by taking the median of several-to-many long (*c.* 300s) dark exposures, which are subsequently scaled to 1s exposure length.

• Algorithmic component: The individual frames will be run through the standard ISR processing (including cosmic ray removal) before being combined; this combination may be done using the standard LSST image stacking code, and a prototype construction algorithm currently exists in `pipe_drivers`. The final version must be configurable to use scalar-, vector- or array-type overscan subtraction, and be robust to contamination from cosmic rays when coadding.

[6] If the readout noise in any channel is too low (relative to the gain) to properly sample the noise distribution, a simple fix is to add sets of n (*e.g.* 3) bias exposures before creating the stacked image.

4.3.3 Master Linearity

Linearity curves for each amplifier in the camera; identical to §4.2.3, unless updated during operations.

• Algorithmic component: An algorithm will need to be written to generate the linearity curves from raw data, either from binned flats, CBP data or "ramp frames". This requires careful treatment, as the "brighter-fatter effect"can masquerade as non-linearity. Code to apply the non-linearity correction during ISR is currently being implemented by Russell Owen. Care must be taken to calculate these after bias subtraction, or be consistent with the way in which they are applied during ISR.

4.3.4 Master Fringe Frames

Compound (polychromatic) fringe frames, dynamically created to match the emission spectrum of the atmosphere at the time of observation.

• Algorithmic component: Construction of these fringe frames proceeds from monochromatic flats, likely using the existing PCA algorithm in `pipe_drivers`.

4.3.5 Master Gain Values

The gain values for all amplifiers in the camera, in e^-/ADU ; identical to §4.2.2, unless updated during operations.

• Algorithmic component: The algorithm to determine this on the mountain is potentially difficult, and needs to be developed. Almost arbitrarily accurate *initial* gain measurements will exist as an input, but monitoring the evolution of these gains to the required accuracy is currently an unsolved problem.

It will be possible to monitor the relative gain within a given CCD by demanding that the flat fields be continuous across amplifier boundaries; this is, however, more difficult across device boundaries. Ticket DM-6030 exists to explore the possibility of using cosmic ray muons and the unavoidable radioisotope contamination inside the camera for this purpose. If this fails [7] then another method will need to be devised. The necessary accuracy of this measurement should be firmly established.

4.3.6 Master Defects

A list of all the bad pixels in each CCD; identical to §4.3.6, unless updated during operations.

• Algorithmic component: Perform statistical analysis of dark frames, flats and "pocket-pumping" exposures to derive an updated defect list.

[7] Merlin's estimate is that the likelihood of failure is moderate-to-high, Robert disagrees.

4.3.7 Saturation Levels

The level (in electrons), for each amplifier, at which charge bleeds into a neighboring pixel; identical to §4.3.7, unless updated during operations.

• Algorithmic component: This will be measured using the CBP if the spots are well-formed and stable. If not, these levels can be measured by saturating many stars in long sky exposures and dithering, with custom processing used to calculate the PSF from the unsaturated stars. Subsequent code will be developed to detect where saturation is occurring and calculate the saturation levels.

4.3.8 Crosstalk

The crosstalk matrix element for every pair of amplifiers in the camera; identical to §4.3.8, unless updated during operations. The probability that this will need to be updated is high, as the validity of these values depends on them being measured with the camera in its final configuration. This is because the crosstalk levels are determined by the inter-CCD and inter-raft capacitive couplings, which, though supposedly small (especially in the case of the inter-raft coupling), depend on the exact physical locations of all the circuit boards and flex cables with respect to one another. It is therefore necessary to be able to remeasure this on the mountain using the CBP.

• Algorithmic component: In the un-multiplexed limit, this involves dithering a single CBP spot around the focal plane and measuring the crosstalk ghosts (whilst carefully disambiguating these from optical ghosts using CBP trickery). Some multiplexing will clearly be possible using a multi-pinhole CBP mask, though the level of this remains to be determined, and depends on the final properties of the camera and the optical system. We baseline for a single spot mask, a one-spot-per-CCD mask, a one-spot-per-raft-mask, and ideally a one-spot-per-amplifier mask.

CBP dithering scripts will be written which will involve mask-specific raster scanning routines, followed by either performing a camera rotation or by re-raster scanning at a different M1 position for the previous focal plane positions to differentiate between the crosstalk and optical ghosts. Code to perform this differentiation will be written, which will then measure the coupling coefficients. Further reading on crosstalk in LSST CCDs can be found in [22],

Confirmation of the measured crosstalk matrix will be performed using either CBP data, saturated stars' bleed trails, or cosmic rays in dark frames.

4.3.9 Impure Monochromatic Flats

Sets of 'monochromatic' (*c.* 1nm) trimmed, overscan-subtracted, dark-corrected flat-field images for each filter. These flats will *include* any ghosted or scattered light.

• Algorithmic component: Construction algorithm exists in `pipe_drivers`.

4.3.10 Pure Monochromatic Flats

Sets of 'monochromatic' (*c.* 1nm) trimmed, overscan-subtracted, dark-corrected flat-field images for each filter. These flats will *exclude* any ghosted or scattered light.

• Algorithmic component: Constructed from §4.3.9 and corrected using 2D spline fits to the photometric measurements of the CBP spots. See **XXX cite RHL here** for further reading.

4.3.11 PhotoFlats

A linear combination of pure monochromatic flats, weighted by a flat-spectrum source (or other defined standard SED), absorbed by a standard atmosphere, and observed through each filter.

• Algorithmic component: The combination of pure monochromatic flats is simple; the standard atmosphere" and "standard SED" remain to be defined. This will only be necessary if per-object corrections are not being applied.

4.3.12 Low-res narrow-band flats

A low-resolution (both in space and wavelength) version of §4.3.10.

• Algorithmic component: Scripting to perform the necessary sweeps of the laser light source, and characterization of its output, as the pulse energy will need to be normalized to.

4.3.13 Pixel Sizes

A map of the (effective) pixel-size distortions. At worst, this will be a $n_{\rm width} \times n_{\rm height} \times 2$ datacube of floats. Pixel size distortions include small-scale quasi-random size variations, mask-stitching/tiling artifacts, tree-rings, and any other effects not dynamical in nature.

• Algorithmic component: The algorithm to measure this is currently a (somewhat) unsolved problem. It has been claimed by Aaron Roodman & co. that these can be measured from flat-fields, but the problem is under-constrained, and thus the stability (nay, validity?) of their measurements is questionable, despite seeming to work. Further thought is required to establish whether their method can be used, and if not, devise another one (though it is not obvious how the problem can be made to be well constrained).

Here be dragons...?

4.3.14 Brighter-Fatter Coefficients

The coefficients needed to model the "brighter-fatter effect". It is hoped that these are a small number of floats per CCD, but this is not yet entirely clear. The input data necessary to calculate these will likely be restricted to superflats at various flux levels, with the possible addition of some star fields for verification of the coefficients.

• Algorithmic component: A number of techniques exist to measure these (mostly developed by members of the DESC SAWG). Code already exists to estimate the kernel/coefficients, and apply the corrections using an enhanced version of the Astier/Antilogus technique.

Here be dragons...?

4.3.15 CTE Measurement

Measurement of the charge transfer efficiency for each amplifier/column in the camera. In the most simple case, where the dominant trap is close to the amplifier in the serial register and thus affects all columns equally, this would be a single number per amplifier. The next level of complexity would be a number per column, with the still more complex version involving characterizing the specific defects and their locations on the chips, in which case this becomes a per-pixel product, though this could be simplified with the use of bounding-boxes as with defect maps. The nominal case should likely be considered as per-column or per-amplifier, because if the number of columns with significant effects is small, these columns would most likely just be masked out rather than corrected.

• Algorithmic component: Measurement of CTE is subtle, though several established methods exist for doing so. Using the ^{55}Fe method will not be possible due to the lack of a radioisotope source in the camera, but the flat-field overscan excess measurement method and flat-field correlation method would both be possible using existing input data products. See **XXX insert citation** for further reading on these methods, one of which would need to be implemented within the DM framework.

4.3.16 Filter Transmission

Monitoring of filter transmission *in-situ*. As well as the filter transmission measurement provided by the camera team/vendor in §4.2.11, we further baseline the development of a procedure for monitoring the filter response at 1 nm resolution using the approach described in [20], *i.e.* by making suitable

CBP measurements with and without the filter in the beam, and averaging over the angles.
• Algorithmic component: Created from measurements in §4.2.10.

4.3.17 Ghost catalog

A catalog of the 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 once the system is operational. The baseline design therefore calls for this system to be prototyped using data from precursor instrumentation; we note that ghosts and ghoulies in *e.g.* HSC are well known and more significant than are expected in LSST.

> **Note from John Swinbank:**
> It is not currently clear where the responsibility for characterizing ghosts and glints in the system lies. We assume it is outside this WBS. RHL instructed to note that this is a possible new entry to the risk registry.

4.4 CBP Control

The procurement of the CBP hardware includes that of the necessary low-level control drivers/software. T&S TCS own the task of taking the vendor-provided low-level routines and turning these into real-world usable routines by constructing higher level functions for *e.g.* homing, slew-to-position, mount a given filter *etc.*.

However, it will still remain to write scripts for the CBP, and interfaces with the OCS, to allow making all of the desired measurements, especially as several, if not all of these, require doing so in concert with the 8.4m.
• Algorithmic component: As well as writing the necessary scripts to acquire the raw data products outlined in §4.2, it will also be necessary to deliver the coordinate transformation and pointing model to allow the CBP to maintain a fixed position on the focal plane whilst illuminating different portions of the pupil, and vice versa.

4.5 Calibration Telescope Input Calibration Data

This section details the input data required to calibrate the auxiliary/Calpyso/-calibration telescope itself. Broadly, this will include most of the ingredients listed in §4.2, but namely:

- Gain values [8]
- Crosstalk matrix
- Linearity curves for each amplifier
- Defect map
- Saturation levels
- Bias frames
- Dark frames
- Broadband flat-fields
- Narrowband flat-fields[9].
- Filter/grating/grism transmissions

Further to these standard camera calibration data products, an illumination/ghost correction will also be required, which will either be derived from star field observations or using the final CBP prototype for direct measurement.

4.6 Calibration Telescope Output Data

This section details the calibrated outputs from the auxiliary/Calpyso/calibration telescope, which, like items in section §4.3, are outputs from the Calibration Products Pipelines to be used during photometric calibration at various levels.

[8]Less accuracy is needed here than for the main camera; a PTC-based measurement will likely be sufficient if correcting for the "brighter-fatter effect".

[9]It is confirmed to be part of the baseline design that there will be both broadband and narrowband light sources at the auxiliary/Calpyso/calibration telescope.

4.6.1 Atmospheric Absorption

As shown in Figure 9, the determination of the atmospheric transmission starts with a two images, one dispersed and one direct, acquired back-to-back with the auxiliary/Calpyso/calibration telescope, where the camera rotator has been set to align the spectra with the parallactic angle. A scaled version direct image is subtracted from the dispersed image, with the scaling factor governed by the geometric efficiency of the disperser in the beam, resulting in the removal of the zeroth order light, and leaving only the spectra. Regions in which the spectra fall are identified, and the strongest lines in these crude uncorrected-spectra are identified and used to determine the incident wavelength as function of position on the detector. This information is then used in conjunction with the system throughput data-cube to correctly flat-field the dispersed images, and the normal ISR processing is then performed.

The 1D spectra are then extracted from the image, flux calibrated, and corrected for second order light contamination. A more precise wavelength calibration is then performed using the spectral lines in this corrected spectrum, taking into account the effect of differential chromatic refraction, resulting in a spectrophotometrically calibrated measurement.

The source's true SED and the calibrated spectrophotometric observation are then used in conjunction with the observational meta-data, *e.g.* the zenith angle, temperature and barometric pressure, to derive an empirical measurement of the atmospheric transmission. This measurement is then matched to atmospheric models to provide an enhancement in the measured absorption spectrum as well as parametric description of the state of the atmosphere at the time of observation.

4.6.2 Night Sky Spectrum

Although the acquisition of a night sky spectrograph is uncertain, in the eventuality that such an instrument is obtained, we provision for the determination of the emission spectrum of the night sky near the auxiliary/Calpyso/calibration telescope boresight, with $R \sim 200$[10], which will be used to synthesize flat-field images matching the sky's SED using the monochromatic dome flats.
• Algorithmic component: Assuming we have a sky spectrograph this is simple. In the absence of a sky spectrograph, an $R \sim 10$ spectrum will be acquired using standard/narrowband filters.

[10]It is not entirely clear yet whether these will be taken on the Calypso or 8.4m boresight.

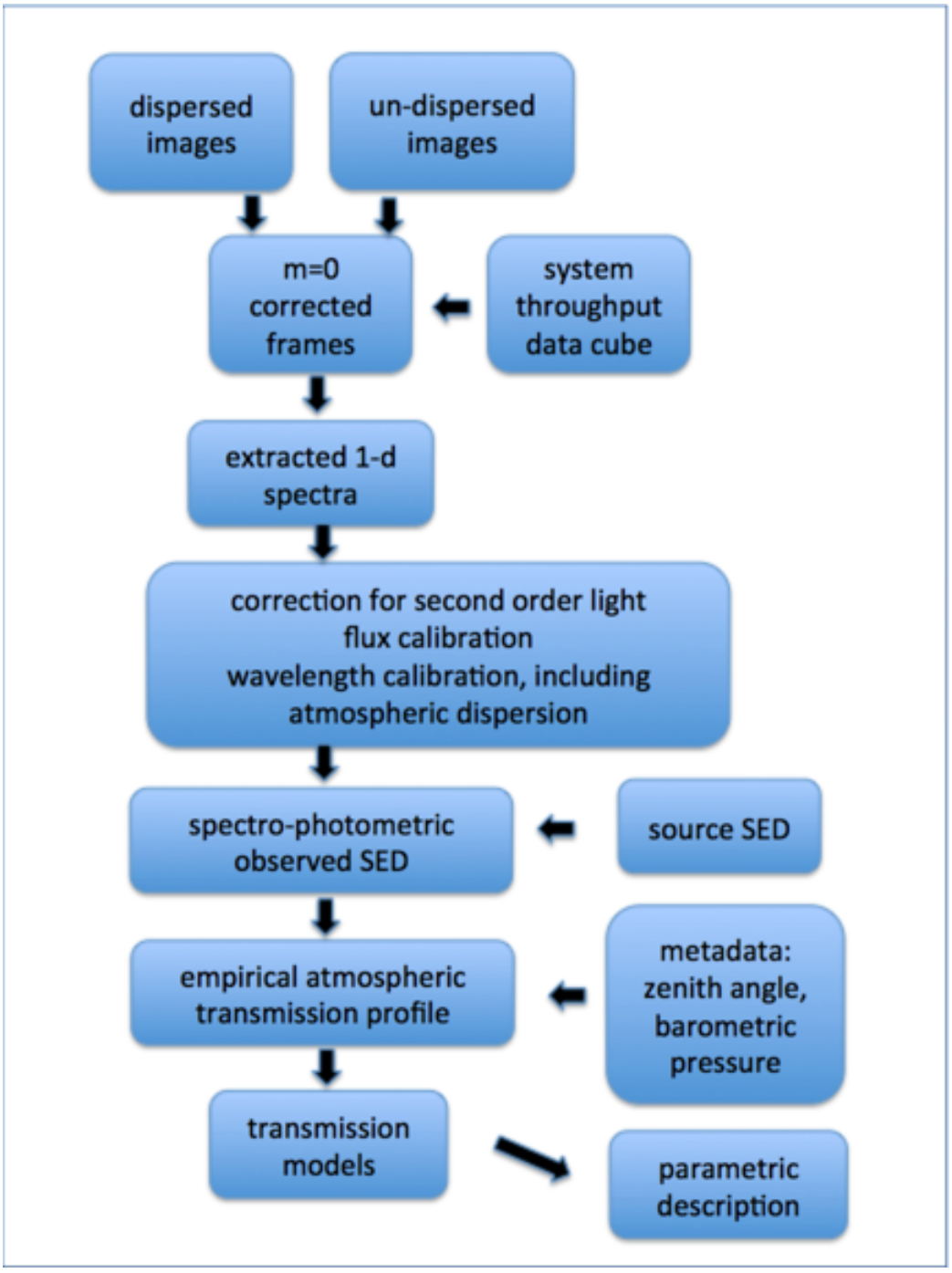


Figure 9: Flowchart depicting the atmospheric absorption measurement pipeline.

.

4.7 Miscellaneous Algorithms

- CBP data processing algorithms:

  – A starflat-like processing of CBP spots to solve for spot brightnesses

  – Optical ghost / crosstalk disambiguation for §4.2.10.6.

  – Derivation of ghosting model, depending on whether or not it lies within this WBS.

- Construction of the photometric flats from the CBP data, which is performed roughly as follows:

  – Having performed a starflat-like processing, and normalized to the results, fit a surface through the CBP values (either per-CCD or for the whole camera); a spline would be a reasonable choice (either the product of two 1-D splines, or a thin plate spline. RHL would start with the former as they are easier to understand).

  – Divide the dome flat by this surface, giving an estimate of the illumination and chip-to-chip correction

  – Fit a curve to this correction, and correct the dome screen. This should be close to the values derived from the CBP data (and can preserve discontinuities in the QE across chips which the fitted curves have a hard time following).

  – Iterate a couple of times; each iteration should result in a smaller and smoother correction, which we are therefore better able to model.

  – Then repeat the above at a suitable set of wavelengths, chosen so that the variation of these corrections as a function of wavelength is well captured; we now know the the relative QE for all pixels in the camera, as a function of wavelength, in the absence of a filter.

  – Using the filter transmission curves, the relative QE for all the pixels in the camera for each filter can be determined at 1nm resolution; this is our monochromatic photometric flatfield.

- Ghosting model: Following the above analysis we will have information which, whilst not needed to perform the calibration of LSST, will provide a valuable cross-check, and will inform the camera and telescope teams

about the state of the optical elements, and is therefore an output of the Calibration Products Pipeline. This can be done in two ways:

– By analysis of the CBP data, concentrating not on the spots but on the scattered/ghost light.

– By looking at the corrections applied to the CBP spot data to arrive at the dome screen data.

4.8 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.

5 Data Release Production

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 2.

From a conceptual standpoint, data release production can be split into six 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 image differencing is described in section 5.2.

3. We process coadds to generate preliminary object catalogs, including detection, deblending, and the first phase of measurement. This is discussed in section 5.3.

4. We resolve overlap regions in our tiling of the sky, in which the same objects have been detected and processed multiple times. This is described in section 5.4.

5. We perform more precise measurements of objects by fitting models to visit-level images, either simultaneously or individually, as discussed in section 5.5.

6. 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

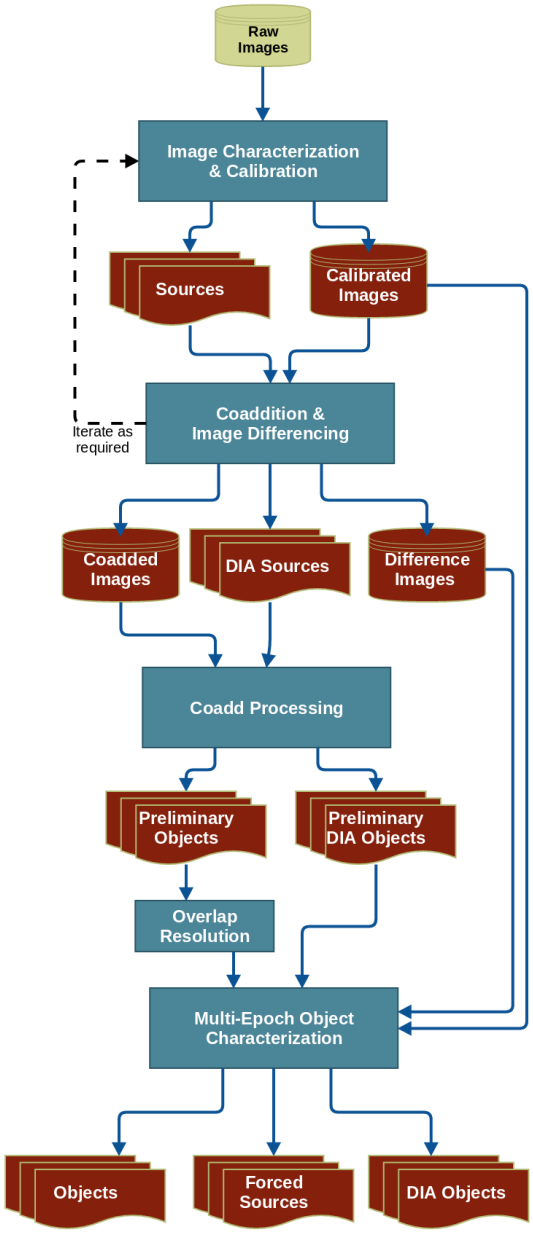


Figure 10: Summary of the Data Release Production image processing flow. Processing is split into multiple pipelines, which are conceptually organized into the groups discussed in sections 5.1-5.5. A final pipeline group discussed in section 5.6 simply operates on the catalogs and is not shown here.

Name	Availability	Description
Source	Stored	Measurements from direct analysis of individual exposures.
DIASource	Stored	Measurements from difference image 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 2: 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).

individual objects. This stage is not shown in Figure 10, but postprocessing pipelines are described in section 5.6.

This conceptual ordering is an oversimplification of the actual processing flow, however; as shown in Figures 10 and 11, the first two groups are 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 the production, allowing us to remove these cycles simply by duplicating some pipelines a fixed number of times. Decisions on the number of iterations must be backed by QA metrics. 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. However, 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

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[11]) 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.

[11] This is not limited to FITS standard transformations; see Section 9.11.

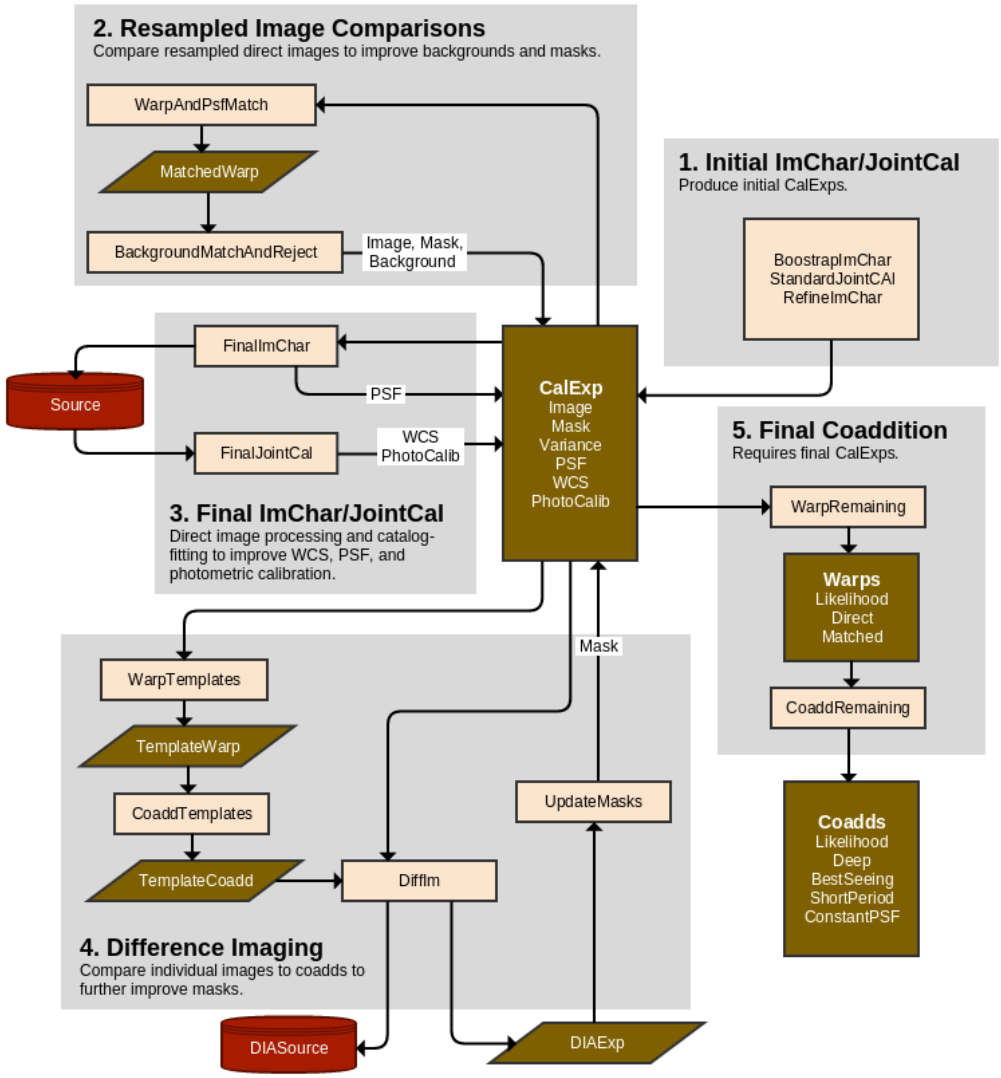


Figure 11: Data flow diagram for the Data Release Production image coaddition and image differencing pipelines. Processing proceeds roughly counterclockwise, starting from the upper right with pipelines described in Section 5.1. Each update to a component of the central CalExp dataset can in theory trigger 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 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 that supersede those in the reference catalogs are aggregated and used to update it 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. Some visits may overlap multiples tracts, however, and will hence be processed multiple times.

The final output data products of the ImChar/JointCal iteration are the Source table and the CalExp (calibrated exposure) images. CalExp is an Exposure, and hence has multiple 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, 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, calibrations):
    # 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
```

Much of this pipeline is an iteration that incrementally improves detection depth while improving the PSF model. This loop is probably only necessary in crowded fields, where it will be necessary to subtract brighter stars in order to detect fainter ones; we expect most high-latitude visits to require only a single iteration. The details of the convergence criteria and changes in behavior between iterations will be determined by future algorithm research. It is also likely that some of the steps within the loop may be moved out of the loop entirely, if they depend only weakly on quantities that change between iterations.

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, experiments on our ability to accurately type faint stars using 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 Input Data Product: Calibrations Calibration frames and metadata from the Calibration Products Pipeline. This may include any of the data products listed in Section 4.3, though some will probably not be used until later stages of the production.

5.1.1.4 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.

5.1.1.5 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 creates the first versions of all of these components (though some, such as the VisitInfo, are merely copied from the raw images). 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, Uncertainty).

5.1.1.6 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.

5.1.1.7 SubtractSnaps Delegate to the Snap Subtraction algorithmic component to mask artifacts in the difference between snaps. If passed a PSF (as in the iterative stage of BootstrapImChar), 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).

5.1.1.8 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 consider treating each snap as a completely separate visit.

5.1.1.9 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). Note that we expect this algorithmic component to be contributed by LSST Systems Engineering, not Data Management. We start with a PSF estimated from the wavefront sensors only because these should be able to use bright stars that are saturated in the science exposures, mitigating the effect of crowding; in high-latitude fields this step may be unnecessary.

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.

5.1.1.10 SubtractBackground Delegate to the Single Visit 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).

5.1.1.11 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. We may first detect on the original image (i.e. without PSF correlation) at a higher threshold to improve peak identification for bright blended objects.

In crowded fields, each iteration of detection will decrease the threshold, increasing the number of objects detected. 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.

5.1.1.12 DeblendSources Delegate to the Single Frame Deblending algorithmic component to split Footprints with multiple peaks into deblend families, and generate HeavyFootprints that split each pixel's values amongst the objects that contribute to it.

5.1.1.13 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
- Static Point Source Model Photometry

These measurements will not be included in the final Source catalog, so they need only include algorithms necessary to feed later steps (and we may not measure the full suite of apertures).

5.1.1.14 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.

5.1.1.15 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.

5.1.1.16 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.

5.1.1.17 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 and include wavelength dependence in the fit. If we do not use a 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. This aspect of PSF modeling is considerably less developed, and may require significant algorithmic research.

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

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

5.1.1.19 SubtractStars Subtract all detected stars above a flux limit from the image, using the PSF model (including the wings). 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.

5.1.1.20 ReinsertStars Add stars removed in SubtractStars back into the image, and merge corresponding Footprints and peaks into the source catalog. Information about the nature of these detections will be propagated through the peaks.

5.1.1.21 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.

5.1.1.22 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 (essentially all flux measurements aside from the suite of fixed apertures) to the predicted integrated flux at infinity.

Additional research may be required to determine the best aperture corrections to apply to galaxy fluxes. Our baseline approach is to apply the same correction to galaxies that we apply to stars, which is correct for small galaxies and defines a consistent photometric system. This is formally incorrect for large galaxies, but there is (to our knowledge) no formally correct approach.

5.1.1.23 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 samples of stars for later steps, delegating to JointCalClassification. The samples for photometric and astrometric calibration may be different (for instance, we may require low variability only in the photometric fit and no proper motion only in the astrometric fit). 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; whether this supersedes SEDs or colors in the reference catalog depends on our approach to absolute calibration.

3. We fit simultaneously for an improved astrometric solution by requiring each star in a match to have the same position, delegating to the Joint Astrometric Fit algorithmic component. This will need to correct

(perhaps approximately) for centroid shifts due to DCR, proper motion, and parallax; if it does not, it must be robust against these shifts (perhaps via outlier rejection). This requires that StandardJointCal have access to the VisitInfo component of each CalExp, in order to calcluate DCR. The models and parameters to fit must be determined by experimentation on real data (as they depend on the number of degrees of freedom in the as-built system on different timescales), and hence the algorithm must be flexible enough to fit a wide variety of models. This fit updates the WCS component for each CalExp.

4. We fit simultaneously for a per-visit zeropoint and a smooth atmospheric transmission correction by requiring each star in a match to have the same flux after applying the per-poch smoothed monochromatic flat fields produced by the calibration products pipeline, delegating to the Joint Photometric Fit algorithmic component. This fit should also have the ability to fit per-CCD photometric zeropoints for diagnostic purposes. There is a small chance this fit will also be used to further constrain those monochromatic flat fields. This fit updates the 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. The sources included in the reference catalog will be a securely-classified bright subset of the full source catalog.

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.

If the Gaia catalog cannot be used to tie together the photometric calibration between different tracts, a larger-scale multi-tract photometric fit must also be run (see Global Photometric Calibration), which would upgrade this step from a tract-level procedure to a larger sequence point. It is unlikely this sequence point would extend to the full survey. It would only be run once, but may happen in either StandardJointCal or FinalJointCal. If the Gaia catalog is sufficient for large-scale photometric calibration, Global Photometric Fitting may instead be run after the data release production as complete as a form of QA.

Before LSST's atmospheric monitoring telescope, the Gaia catalog, and the suite of monochromatic flats are available, photometric calibration will be considerably more difficult, and hence pipeline commissioning (on both precursor data and some LSST commissioning data) will require a more sophisticated global fit (see Interim Wavelength-Dependent Photometric Fitting) that uses multiple observations of stars to infer their SEDs and the wavelength-dependent transmission of the system as well as their magnitudes and the spatial dependence of the transmission.

5.1.3 RefineImChar

RefineImChar performs an incremental improvement on the PSF model produced by BootstrapImChar, then uses this to produce improved source measurements, assuming 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 SCIENCE_SENSORS:
        calexp[ccd].mi = SubtractStars(calexp[ccd].{mi,psf}, sources[ccd])
    calexp.{mi,background} = SubtractBackground(calexp.mi)
    parallel for ccd in SCIENCE_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 SCIENCE_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. Previously-extracted wavefront information may again be used in PSF modeling, but we do not expect to do any additional processing of the wavefront sensors in this pipeline.

Note that RefineImChar does not update the CalExp's WCS, PhotoCalib, or Uncertainty; the WCS and PhotoCalib will have already been better constrained in StandardJointCal, and no changes have been made to the pixels. The Image is only updated to reflect the new background, and the Mask is only updated to indicate new detections.

5.1.4 FinalImChar

FinalImChar is responsible for producing 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 15. 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 will depend on the approach to absolute calibration and the as-built performance of the surrounding pipelines. Because it is responsible for the final photometric calibration, it may 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 a 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. It will produce an updated WCS and PhotoCalib for each CalExp, with the PhotoCalib possibly now reflecting absolute as well as relative calibration.

As discussed in section 5.1.2, this pipeline may require a multi-tract sequence point.

5.2 Image Coaddition and Image Differencing

The next group of pipelines in a Data Release Production consists 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. Variable objects will be particularly challenging in this context, as our models of their effective coadded PSFs will be incorrect unless variability is included in those models.

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 subtract the updated background). It will also return to the previous pipeline group described in Section 5.1 to update other CalExp components. As in the previous pipeline group, tracts are processed independently, and since some visits overlap multiple tracts, multiple CalExps (one for each tract) will be produced for the CCDs in these visits. The data flow between pipelines is shown in Figure 11, with the numbered steps 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 Background-MatchAndReject pipeline. 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 Photo-Cal. 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 See above.

DIAExp A CCD-level Exposure that is the difference between the CalExp and a template coadd, in the coordinate system of the CalExp. It may have the same PSF as the CalExp (if traditional PSF matching is used) or its own PSF model (if the difference image is decorrelated[12] after matching).

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 emphasize depth at the possible expense of seeing.

[12] *Decorrelated images* refer here to a technique for convolving images by the transpose of the PSF, summing or differencing them, and then deconvolving the transpose of the effective PSF of the resulting image. See DMTN-015 for more information.

[**Note:**
Add link to technote.]

BestSeeingCoadd A coadd data product built to emphasize 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. Aside from the cut on epoch range, this would use the same filter as DeepCoadd.

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. In order to produce templates appropriate for the level of DCR in a given science image, these coadds may require a third dimension in addition to the usual two image dimensions (likely either wavelength or a quantity that is a function of airmass).

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. The alternative strategies for using decorrelated coadds yield five main scenarios:

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 most bands, so we may select a simpler approach 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 probably use it here as well. Simultaneously warping and PSF matching could also yield important computational performance improvements.

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 from the standpoint of large-scale data flow and parallelization, and a working prototype has not yet been demonstrated except for SDSS data, for which the drift-scan observing

strategy makes the problem easier. The algorithm is simple over any patch of sky where the set of input images is constant, and we do not anticipate significant difficulty in extending this to an algorithm that works across image boundaries. The main challenge is likely to be the parallelization and data flow necessary to efficiently ensure consistent backgrounds over a full tract. Separate tracts are stil processed independently, however.

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 continuous 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 Matched 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. This stage must involve at least some communication between patches to ensure that the background is continuous and consistent in patch overlap regions.

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 via the Coadd Background Estimation algorithmic component. This background estimation must also involve communication between patches to ensure consistency. Combining the images will be performed by the Coaddition algorithmic component, while the Warped Image Comparison component is used to 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.

6. We combine the relevant difference backgrounds with the coadd background and transform them back to the CalExp coordinate systems to compute new background models for each CalExp.

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 matched-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 11 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.

Here we should strongly consider developing 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. 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 g (and possibly u and r); 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 level of DCR.

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 propagate 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. MatchWarped is only produced in option C.

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

Given that all of these steps involve resampling the image, it would be desirable for computational reasons 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 significantly 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 spatial- and wavelength-dependent photometric calibration), in addition to pixel values.

5.3 Coadd Processing

In comparison to the previous two pipeline groups, the large-scale processing flow in 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.

Coadd processing begins with the DeepDetect pipeline, which simply finds above-threshold regions and peaks in multiple detection coadds. These are merged in catalog-space in DeepAssociate, then deblended at the pixel level in DeepDeblend. The deblended pixels are measured in MeasureCoadds, which may also fit multiple objects simultaneously using the original undeblended pixels.

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 spectra, step functions, etc.).
- We could build a single χ^2 coadd from the per-band coadds, which is only optimal for objects the color of the sky noise, 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 (including their Peaks and any 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 a table of unassociated DIASources that will be used to construct SSObjects in MOPS.

We do *not* include the Source table in this merge, as virtually all Sources correspond to astrophysical objects better detected elsewhere. Non-moving or slowly-moving astrophysical objects (even variable non-transient objects) will be detected at much higher significance in DeepDetect (as CoaddSources). Transients and fast-moving objects will be detected at similar significance with significantly less blending (and much easier classification) in DiffIm (as DIASources). While a small number of transient/moving Sources near the detection limit may not be detected in difference images due to extra noise from the template, these will be nearly impossible to recover without a large false positive rate from a spatial match of the Source table.

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 MovingObjectPipeline 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 specified), 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 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 is an update to the Object table, which adds columns to indicate the origins of Objects and the decisions taken by the deblender as well as modifying the set of rows to reflect the current object definitions. It also includes attaching pixel-level deblend information to each Object. If stored directly in the form of HeavyFootprints, this would be a large dataset (comparable to the coadd pixel data). This form must be available at least to the MeasureCoadds pipeline, but it almost certainly needs to be available to science users as well. Depending on the deblender implementation, it may be possible to instead store analytic models or some other compressed form that would allow the full HeavyFootprints to be reconstructed quickly on the fly, while requiring a relatively small amount of additional per-object information. If this compression is lossy, it should probably be applied before the deblend results are first used in MeasureCoadds so the deblends used there can be exactly reconstructed later.

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. If the deblender utilizes simultaneous fitting internally, we may also be able to use the results of those fits directly as measurement outputs or to reduce the amount of subsequent fitting that must be done.

- 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, 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; see Figure 12) 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.

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

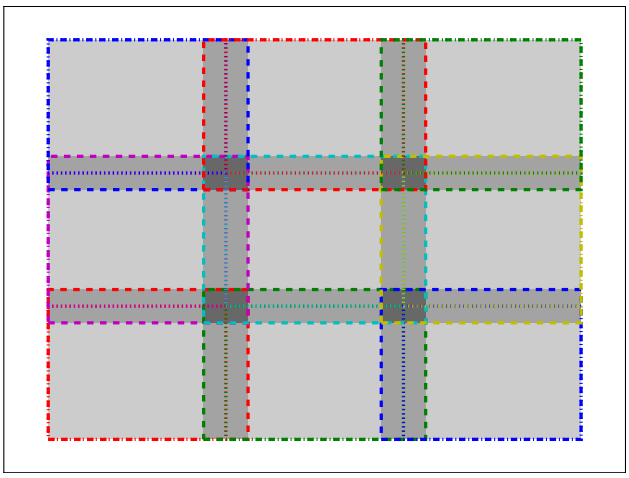


Figure 12: Patch boundaries and overlaps regions for a single tract with 3×3 patches. Different colors represent different patches; dashed lines show outer patch regions and dotted lines show inner patch regions. Light gray regions are processed as part of only one patch, medium regions as part of two, and dark regions as part of four.

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 family's blend chain is the recursive union of all families it overlaps with in any tract that contributes to that overlap region see Figure 13. 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:

1. If a blend chain is strictly contained by only one tract, all families within that chain are assigned to that tract. Note that this can occur

even if the blend chain overlaps multiple tracts, as in Figure 13; region 1 there is wholly contained only by the blue tract even though it overlaps the green tract.

2. 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. This is illustrated by region 2 in Figure 13, which would be assigned to the red tract.

3. 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.

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 Object families 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.

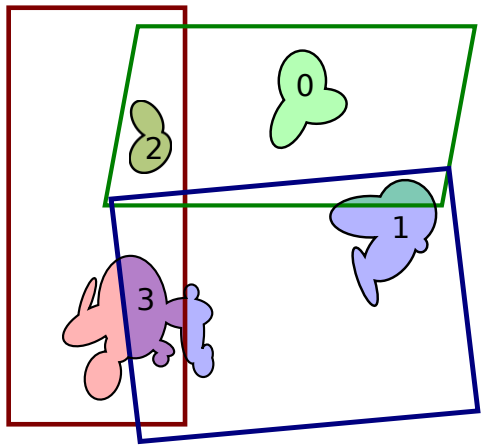


Figure 13: Tract overlap scenarios, corresponding to the enumerated list in the text. Each region outlined in black is a blend chain; transparent filled regions within these indicate the contributes from individual tracts. The region labeled "0" is strictly contained by the green tract and does not touch any others, so it does not participate in tract overlap resolution at all.

- 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 layer:

- 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 on a 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 flexibility 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 of 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 8.7.3.

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 8.7.3.2 and 8.7.3.1).

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 MovingObjectPipeline

The Moving Object Pipeline 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 Moving Object Pipeline, but the fact that DRP will run 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 clearly

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.

The DRP Moving Object Pipeline 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 six tables that ultimately must be ingested into the public database: Source, DIASource, Object, DIAObject, SSObject, and ForcedSource. The quantities inSource are either in raw units (e.g. fluxes are in counts, positions in pixels) or pseudo-raw relative units (e.g. coadd-pixel counts or tract pixel coordinates). These must be transformed into calibrated units via our astrometric and photometric solutions, a process we delegate to the Raw Measurement Calibration algorithmic component. For the pseudo-raw relative units used for coadd measurements and multifit results, these transformations are exact and hence do not introduce any new uncertainty, but must still be applied.

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 or deterministic, 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 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).

Variability summary statistics are delegated to the Variability Characterization 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" a representative subset of the Object and ForcedSource tables and/or similar tables derived from special program data. Rather than imposing a full-survey sequence point here, we'll probably use previous data releases or results from a small-area validation release.

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.

6 Science Data Quality Assurance

6.1 Overview of SDQA

Science Data Quality Assurance (SDQA) describes a collection of capabilities for Quality Analysis (QA) and Quality Control (QC) that collectively ensure a high quality performing software system as well as the ability to meet the scientific goals of the system. These are designed to service LSST's quality assessment needs through all phases of Construction, Commissioning and Operations. Consumers of these services may include developers, facility staff, DAC (e.g., Level 3) users, and the general LSST science user community.

SDQA capabilities can be divided into Quality Analysis (QA) and Quality Control (QC) Capabilities.

6.1.1 Quality Analysis (QA)

Quality Analysis describes a range of *activities* aimed at understanding the algorithmic performance of the code and eventually the properties of the data. These are achieved by a set of *tools* that leverage components of the Data Management subsystem, as well as additional capabilities as necessary, to allow for the scientific inspection of data both by DM developers and scientists during construction as well as commissioning engineers and scientists during commissioning and operations.

QA activities include processing, analysing and inspecting data for forensic purposes. They also include the development of tests for verifying and characterising algorithmic performance, and are closely tied to the development of the Science Pipelines and the Science User Interface.

6.1.2 Quality Control (QC)

Quality Control describes a range of *services and processes* aimed at measuring and monitoring and characterising the system (both software and data) to verify and characterise its performance. Whereas Quality Assurance describes the activities of people, Quality Control describes systems running autonomously and only notify people when an anomaly that cannot be automatically rectified is detected.

QC services include regression detection, measurement of Key Performance Metrics, a notification infrastructure and quality monitor dashboards. They are primarily the responsibility of the Science Quality and Reliability Engineering (SQuaRE) team.

6.2 Key Features of SDQA (QA and QC) systems

- SDQA provides capabilities for science data quality analysis of Level 1, 2, and Calibration Processing pipelines.
- SDQA provides services to support software development in Construction, Commissioning and Operations.
- SDQA provides for the visualization, analysis and monitoring capabilities needed for common science quality data analysis use cases. Its inputs may be gathered from SDQA services, the production pipelines, engineering data sources and non-LSST data sources.
- SDQA has the flexibility to support execution of ad-hoc (user-driven) tests and analyses of ad-hoc datasets (provided they are supported by the LSST stack) within a standard framework.
- SDQA supports use cases involving interactive "drill-down" of QC data exposed through its visualization interfaces.
- SDQA allows for notifications to be issued when monitoring quantities that fall outside permissible bounds and/or have degraded over historical values.
- SDQA is able to collect and harvest the outputs and logs of execution of a pipeline, and extract and expose metrics from these logs.
- SQDA makes provision to store outputs that are not stored through other LSST data access services.
- SDQA is 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 is architected in a manner that would enable it to be deployable on standard cloud architectures outside of the LSST facilities so that community-based L3 development activities can be supported.

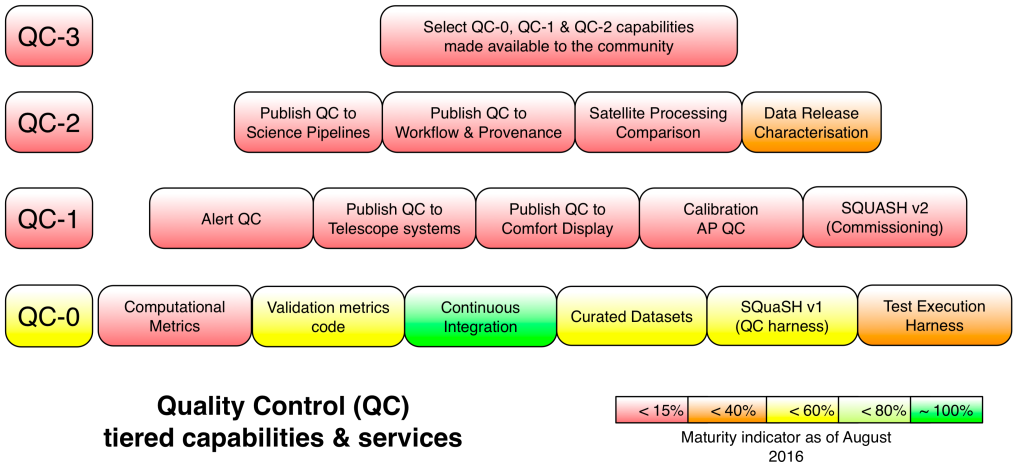


Figure 14: Layer of progressive QC capabilities across all 4 tiers of QC.

6.3 Key Tasks for Each Tier of QC

SDQA system provides a framework that is capable of monitoring QC information at four different stages of capability and maturity:

QC Tier 0 Testing and Validation of the DM sub-system during software development

QC Tier 1 Real-time data quality and system assesment during commissioning and operations (also, forensics)

QC Tier 2 Quality assessment of Data Releases (also, forensics)

QC Tier 3 Ability for the community to evaluate the data quality of their own analyses. These should made available as well-documented and deployable versions of core QC Tier 0–2 services.

A view of the tiered QC services is shown in 14. Capabilities are increasingly layered (eg. tier QC-1 includes QC-0 services). A definition of the QC tiers is given in the rest of this section.

6.3.1 QC Tier 0

The first step to good quality data is good quality software. The purpose of QC-0 services is to enable testing of the DM software during development as

well as validate software improvements during commissioning and operations, quantifying the software performance against known and/or expected inputs and outputs.

The core capabilities of QC-0 services are:

6.3.1.1 Continuous Integration Services

- Continuous integration services compile code to uncover build errors and to trap failures in unit tests.
- Builds of references (tags, branches) that can happen on a schedule, on developer request or on development events (e.g., merge to master)
- SDQA provides CI services on multiple reference platforms and uses OS and compiler portability testing as a way to ensure the codebase is well engineered for future use.

6.3.1.2 Test Execution Harness

- A test execution harness runs tests (such as data analysis unit tests) on a regular cadence (e.g. nightly/weekly/monthly) to allow basic functional checkout of the code. Tests written as part of Tier 1-3 QC can be added directly by developers and be caused to execute without manual intervention, for example by checking in code or a specification in a purpose built test repository.
- The test execution harness also allows the selection of a number of appropriate reference datasets (see 6.3.1.5
- Results from such tests are exposed in such a way as to allow summary reports and meaningful failure notifications.

6.3.1.3 Verification Metrics Code

- During Construction, progress towards meeting DM subsystem requirements revolve around the Key Performance Metrics (KPMs) outlined in LDM-240. SDQA implements code to calculate these KPMs. Consult *reference to KPM Verification document* for a list of those metrics and how (and by whom and on what) they will be calculated.

- Additional metrics must be calculated to be met in order for the DM subsystem to demonstrate its operational readiness. The list of those metrics and how (and by whom) they will be calculated will be in *reference to DM Verification Plan CoDR document*. In terms of QC infrastructure, these metrics will not require different capabilities than the KPMs.

- Verification code will be implemented in such a way that it can run with normal pipeline processing on developer's laptops, either inline or as after-burners as appropriate.

- Additional metrics may be proposed during construction that are helpful to development or algorithm characterisation, either by developers investigating algorithms or as part of characterising the complete scientific and computational performance of the DM subsystem. SDQA will provide ways of executing that code in a similar way to KPMs, but apps developers may need to contribute the code (or at least document the algorithmic approach) to calculate those metrics.

6.3.1.4 Computational Metrics

- While the scope of this document is the scientific aspects of the pipelines, SDQA must also accommodate non-scientific KPMs and other metrics, such as computational performance characterisation.

- SDQA will provide a capability to instrument the production pipelines to calculate computational performance metrics, and harvest data from those instrumentations (via logs or more direct ways of populating the QC database tables).

- The computational performance metrics that SDQA calculates will be in practice surrogates for the actual computational performance in production since those will depend on the production system architecture. The purpose of calculating those as part of SDQA is to continuously monitor relative performance to alert the developers that a regression has occurred.

- SDQA can calculate modeled system performance from the surrogate computational metrics if a model is provided to it (e.g., from Architecture).

- A library of these instrumentations will be provided so that they can be mixed and matched to pipelines depending on the performance metric of interest.

6.3.1.5 Curated Datasets

- Part of the process for validating the software and its performance is selecting rich but targeted standardized data sets to generate directly comparable metrics between different versions of the software.
- SDQA will select and curate a combination of simulated and precursor datasets that are modest enough for "canary" test runs but rich enough to characterise the envelope of algorithmic performance.
- SDQA will "publish" (make available) these datasets so developers can run the validation tests directly against them in their own environments.

6.3.1.6 SQUASH - Science Quality Analysis Harness SQUASH is a QC-as-a-service architecture that comprises the following elements:

1. The execution of simple pipeline workflows for the purposes of QC
2. The construction of those QC workflows with an emphasis on usability (not necessarily performance)
3. The collection and exposure of the results of those runs for further retrieval and analysis
4. A monitoring system to detect threshold trespass and excursions from past trends

Notes:

- As construction progresses, first-party DM systems to underwrite the functions of SQUASH will become production ready. In the meantime, basic implementations of minimum viable functionality may be done with boostrap or off-the-shelf solutions either as an interim measure or, in some cases, a more lightweight solution (for example, working around features the butler does not have yet).

- A simple example of a "factory" analysis based on SQUASH is "Calculate the astrometric repeatability on this dataset; display the trend; drill down to to show the histogram of the points that went into calculating this trend".

- An advanced example of a bespoke analysis based on SQUASH is "Display a three-color diagram of the sources in this run; compute the width of point sources in the selected – e.g., blue – part of the locus".

- SQUASH will likely expose results to the LSST Science User Interface and Tools (SUIT) for advanced interaction scenarios (both because of the SUIT team's front-end expertise but also because they are likely to be similar to science-driven interactions in intent and in execution). See 6.3.5

6.3.2 QC Tier 1

QC-1 designates the capability to assess data quality in real-time observing scenarios such as integration, commissioning and operations (as well as data release production); if the role of QC-0 is to validate the software, the role of QC-1 is to validate the performance of the facility.

There are two distinct aspects to this capability:

1. Some metric products and services serve stand-alone user-driven use cases as in QC-0 but with additional data sources, such as the Engineering and Facilities Database (EFD), and with real LSST data as opposed to simulated data or precursor data sets. An example use case is "Show the width of point sources on data taken this week in windy conditions with all vents closed versus only the vents in the wind direction as a function of wind speed".

2. Some metric products are produced as part of the routine operational processing for Level 1 and Calibration pipelines. These will predominantly use the production DM architecture at the Archive Center and its satellites and produce metric products either through QC-specific steps in the processing or via the outputs of task instrumentation, though there might be additional metrics optimised for Commissioning that are required to run in different architectures. An example use case is "show execution times of the deblending task as a function of galactic latitude".

In the first case the architecture is based on components re-used from QC-0 (with modifications made if made necessary by more stringent performance concerns). Additional out-of-scope (for DM) work may be funded by the Commissioning WBS to support "quick-look" or "comfort display" scenarios where some facility health data is gathered directly from Telescope & Site systems as telemetry, in which case a component will be added to the QC-0 architecture to support this.

In the second case, the Level 1 DM system software and processing infrastructure at the Archive Center is used. The Data Access framework (DAX) is used to access all data including values from the EFD and Calibration products.

Note that the EFD is specified to hold all telemetry generated by any observatory system.

All QC-0 components will be involved in QC-1 workflows. The following additional components originate from QC-1 requirements:

6.3.2.1 Alert QC There are two QC components developed for Alert Production:

- A static analysis component that can check, for example, whether the notifications in the alert stream conform to a valid format. This kind of component can be incorporated in the normal Alert Production pipeline.
- A component to receive alerts (akin to a mini-broker) and collect statistics on received events. This would run as a canary node outside LSST facilities to test the alert system is functioning correctly.
- Source injection will be useful for non-producting testing of the Alert Pipeline (see 6.3.6).
- SDQA can provide upper limits to verify AP requirements such as "no more than 0.1% of images shall fail to produce alerts"
- Given the aggressive time budget for AP, SDQA can use an API or other interface to the workflow system (if available) to abort further processing in the event that image quality metrics are too low for successful Alert Production to proceed.

6.3.2.2 Validation Metrics Performance As noted, the components of QC-0 to devise key metrics are qualitatively suitable for QC-1. However:

- We expect to make some optimizations to prevent them from consuming a significant portion of the 60-second alert time budget.
- In the area of computational performance metrics, additional metrics or instrumentations could be needed due to specific elements of the data center architecture, which at this point is still under design. These will be provided under the Processing Control and Site Infrastructure WBS (02C.07).

6.3.2.3 Dome / Operator Displays Some QC displays may be useful as "comfort displays" (or "facility heartbeats") to staff on site at the telescope, or remote operators. If the design of the control room requires displays that could not be generated from the DM-required SQuaSH capabilities, this work will be provided from a non-DM (Commissioning) WBS.

6.3.2.4 Telescope Systems Outputs of the SQQA system may be required by the Observatory Control System in order to take some automated action (e.g., reschedule a field). Whatever information is required will be published as telemetry by means of the OCS Middleware.

6.3.2.5 Camera Calibration The SDQA system will also provide QC of Calibration images and products.

- Images taken from the Camera will require "prompt QC" that will run in the quasi-real-time image processing system. Camera is interested in the monitoring infrastructure of SDQA for tracking parameters such as read noise, cross-talk, linearity etc.
- QC of Calibration Products Production data products (i.e., master calibration images and calibration database entries). These are similar in architecture and implementation to other DRP-related tests. The one exception to the above is the daily daytime/twilight calibration operations prior to night-time observing. QC done for this calibration sequence needs to run under Obsetvatory Control System. There is therefore an explicit or implicit (via the DMCS) interface to the OCS that is yet to be finalised.

- SDQA is responsible for prompt QC of the spectrometer and potentially the sky background spectrometer.

6.3.2.6 Engineering and Commissioning Some data that is taken specifically for engineering or commissioning purposes will require custom treatment (e.g., an image that is taken with deliberately defocussed optics should not trigger QC alarm and instead should have the noted characteristics of the defocussed sources analysed). While architecturally these are the same as other QC tests, the scope and work for this will be defined as part of the Systems Engineering WBS.

6.3.2.7 Data Release Production The daily progress of DRP is characteristically similar that of AP and will be instrumented and monitored by SDQA in the same way.

6.3.3 QC Tier 2

QC-2 designates the capability to assess the periodic Data Release Products that will be published by LSST. The key aspects that will add on to QC-1 capabilities are:

1. the ability to characterise and inspect large data sets;
2. detecting failure modes (excursions from expectation or specification) that are rare in QC-0 analysis or real-time QC-1 processing, but represent an identifiable and systematic population or effect on the scale of a full Data Release;
3. additional characterisation derived from calibration efforts in support of the stringent relative color calibration requirements

The principal focus of QC-2 is to assess the quality of catalogue and image data products of the data releases, perform quality assessment for astrometric and photometric calibration and derived products, and look for problems with the image processing pipelines and systematic problems with the instrument.

In addition to the components provided in QC-0, and QC-1, the new components for QC-2 are:

6.3.3.1 DRP-specific dataset

- The scale of a DRP will impose additional performance requirements on the calculation of key performance metrics and associated quality metrics.
- The need to drill down with random access to the entire DRP data set will fully exercise the SUIT capabilities.

6.3.3.2 Interfaces to Workflow and Provenance System(s) If the SDQA system determines that data (whether science or calibration) is defective, it provides all the information required for the workflow system to take action on this information.

A simple example of this is that a calibration is bad, and it needs to be marked as such so that it is not used in further DRP processing (similar to how if a data frame is bad the compute time should not be wasted processing it further for AP)

A more complex implementation is that a data product previously thought to be good is on further processing or new tests determined to be bad. In this case will be combined with provenance information to mark *all* data polluted with the bad frame as bad, and provide sufficient information to the workflow system to allow it to trigger the necessary reprocessing with that data excluded.

These are implemented in a manner that is agnostic as to the implementation of the Workflow (e.g., they are values in a database table or API methods that different workflow systems can utilize).

In order to support the interface to the provenance system it would be useful to have some provenance analysis tools, that will allow an operator to query specifically what data went into a particular data product or used a specific data product. These would be very useful to QC but will be provided by the Data Access Services WBS (02C.06).

[@KT - who deals wih the bad data system?]

6.3.3.3 Output Interface to Science Pipelines QA results may provide key feedback to model and parameter choices in the Science Pipelines. The result of the QC system should be made available to the Science Pipelines processing in clearly-tracked analysis and provenance via the normal pipeline Data Access Services.

6.3.3.4 Comparison tools for overlap areas due to satellite processing Data Release Processing may be distributed across multiple geographic data centers. It is important to verify consistency (identity even) of the results across these data centers by analyzing both subsets of the overall data processing that are processed redundantly by each data center. A framework to define the splits and overlap region and a coherent dashboard and QC configuration to analyze these overlap regions will be key in building confidence in the merged Data Release.

6.3.3.5 Metrics/products for science users to understand quality of science data products (depth mask/selection function, etc.) The Data Release Processing should generate statistics of depth, typical seeing, etc. for regions of the sky; as well as selection functions for the sensitivity to various types of objects. The code to produce those statistics will need to be validated by processing of well-understood data.

6.3.3.6 Characterization report for Data Release

- Each Data Release will be accompanied by a detailed description of its key data statistics, coverage, and quality metrics.

6.3.4 QC Tier 3

Data quality services will be made available for use with science analysis performed by the LSST Science Collaborations and the community. Tier 0–2 visualization and data exploration tools will be made available (either as a service or as documented deployable systems) to the community.

As community-driven data processing is not fully specified at this point, further requirements of SDQA for L3 will be included in the Level 3 requirements and design documentation.

6.3.5 Quality Analysis

Interactive visualisation and free-form data exploration are critical parts of scientific and engineering insight, and for a system the size of LSST it cannot be effectively done on a developer's laptop and/or using traditional tooling. It follows that for the QA process to happen effectively, more powerful tooling will be necessary to support discovery workflows.

The design of these workflows is out of scope for the this document, which is focused on pipelines generating the products defined in the Data

Products Definition Document and the design is described in a document under preparation. But briefly, they fall into three categories:

1. Capabilities that involve structured pre-defined high-semantics displays (e.g., dashboards) with fixed drill-down workflows. These will be defined by the QC system, specifically the Science Quality Analysis Harness interactive dashboards.

2. Capabilities that are similar to science-user workflows in that they involve generic free-form exploration of the dataset. These will be serviced through the Science User Interface through the Science User Interface Data Analysis and Visualization Tools WBS (02C.05.02), with the Data Access services acting as interface between the SUI and SDQA. This is partly to leverage the superior features of the SUI system, and partly to encourage early in-house testing of the SUI features.

3. Custom User Interfaces aimed at algorithm development and facility commissioning, as well as displays for dome or remote operators, may be built that integrate QC dashboards with QA (SUI) elements as required.

4. A more complex case is the situation where curated pre-defined display is desired, but free-form generic exploration of the results is required. In this situation, QC will have an API or facility for exporting the former into a tool suitable for the latter. One example of this would be a QC report on, say, a standardised KPM measurement that is produced as a Jupyter Notebook; the user can inspect it, or take it and further interact with the results. Further design is underway in this area.

5. In some cases specific algorithms need to be implemented to drive required visualization scenarios; these are provided as part of the Alert Production (02C.03) or Data Release Production (02C.04) as appropriate. An example of this is N-way matching across multiple visits (9.5.8).

6.3.6 Who validates the validator?

SDQA comprises of a system of high semantic value to multiple audiences - dome operators, software developers, science operations staff, data release production engineers and science consumers. Therefore care must be taken to design into the system sanity self-checks to ensure the reliability of its own resuls as well as its upstream pipelines. This section outlines some of the planned features in this area:

6.3.6.1 Intrinsic Design Features Many of the features described so far provide an alert path for misbehaviours of the QC system. For example a trending excursion for a specific key performance metric could either be due to an algorithmic error or a validation code error. Either way, detection will be a necessary first step to investigation.

6.3.6.2 Known Truth While it may be a matter of debate as to how accurate construction-era simulations are compared to the eventual on-sky data, they are extremely valuable as a fixed source of "known truth" which allow for algorithmically simple QC tests that result in quantifiable performance.

6.3.6.3 Reference Truth Comm Cam may allow us to early on develop a small library of representative "reference fields" (eg at different galactic latitudes or ecliptic planes) to provide a minimal standard dataset against which competing algorithmic approaches can be compared (this is similar to the approach taken in Construction with percursor datasets). There would be made available outside the project too alloweing groups working on alternative algoritms and/or implementations to compare their results with the "factory" reductions. Finally, the possibility exists that these reference fields could be unencumbered by proprietary periods so that scientific groups without data rights (and perhaps not even interested in LSST per se) could also utilise them for algorithmic and/or software development.

7 Data Access, the Science User Interface and Tools

An integral part of the LSST Data Management (DM) system is the provision of user access to the LSST data products. Users will be able to access and interact with the data products hosted at Data Access Centers (DACs) through a layered set of graphical and programmatic interfaces that may be invoked both on the DAC systems and on remote computers.

The present document focuses on the scientific pipelines and algorithms necessary to produce the data described in the Data Products Definition Document. Details of the requirements, design, and implementation plan for the data access interfaces and the DACs themselves thus are set forth elsewhere. Nevertheless, we provide a short description of the expected user interfaces here in order to place their design in context with the rest of DM.

Access to the LSST data products, including the raw image data, the Level 1, Level 2, and Calibration catalog and image data products, the Engineering and Facilities Database, and Level 3 data products contributed by users, is enabled by two major software components: the Data Access layer (DAX) and the Science User Interface and Tools (SUIT). This software will run on a combination of the project-provided Data Access Centers' hardware and on users' remote devices, and will be supported at the DACs by a number of IT-infrastructure services such as authentication and authorization, resource provisioning, and the like.

7.1 Data Access Layer

The Data Access Layer (DAX) will consist of both network APIs and Python APIs providing access to the above data products. They will also enable the storage and retrieval of Level 3 data products created by the community, depending on the availability of the resources needed for a product or set of products, supporting the sharing of such data products between individuals, within collaborations, or with the LSST community as a whole. DAX APIs will be provided both for image and tabular (e.g., catalog) data. The DAX layer will include support for a variety of widely-used VO services and protocols, as well as LSST-specific interfaces where appropriate. The Python APIs will enable access to LSST data products at the Data Access Centers from remote systems as well as to user processes running at the DACs. The DAX interfaces

will provide access to a capability to run additional user-specific computations next to the data, operating on the results of large-scale database queries.

7.2 Science User Interface and Tools

The Science User Interface and Tools (SUIT) will provide an interactive, exploratory analysis and visualization environment for the users of LSST data. It is designed to enable creativity and flexible use by the astronomical community, and will be highly configurable and customizable by individuals and collaborations. The SUIT will provide a web-based entry portal suitable for the discovery, searching, and exploration of the LSST data products and the interactive visualization of images, tabular data, and plots, while also providing a workspace environment accessible through the portal or programmatically for additional data manipulation, analysis, and the storage and retrieval of results. The workspace environment will enable users to utilize the resources made available in the DAC for additional processing and storage of data and results. The workspace environment will feature a strong integration of the SUIT's UI components for data searching and visualization with a Python Jupyter notebook environment, in which users will be able to take advantage of the full power of the LSST software stack as well as common community scientific and astronomical Python libraries and tools. The SUIT's components will be available through both JavaScript and Python APIs to facilitate the development of custom portals and the integration of these components into individuals' scientific workflows.

The DAX and SUIT interfaces and frameworks are being designed so that they are suitable for use in the quality assessment of the results from the DM algorithmic pipelines, as well as for operational data quality assessment during observing. This will take full advantage of their configurability, and the resulting system will provide pre-defined views of standard metrics and informative displays as well as an interactive exploration and visualization ("drill down") capability.

All of the DAX and SUIT software will be part of the open-source code base of the LSST project, and much of this software will be developed to be widely useful beyond the project-provided DAC environment.

8 Algorithmic Components

This section describes mid-level Algorithmic Components that are used (possibly in multiple contexts) by the pipelines described in Sections 3, 4, and 5. These in turn depend on the even lower-level Software Primitives described in Section 9. Algorithmic Components will typically be implemented as Python classes (such as the `Task` classes in the codebase as of the time this was written) that frequently delegate to C++. Unlike the pipelines discussed in previous sections, which occupy a specific place in a production, Algorithmic Components should be designed to be reusable in slightly different contexts, even if the baseline design only has them being used in one place. Many components may require different variants for use in different contexts, however, and these different variants may or may not require different classes. These context-specific variants are identified below.

We expect that these components will form the bulk of the LSST Science Pipelines codebase.

8.1 Reference Catalog Construction

8.1.1 Alert Production Reference Catalogs

Alert Production will use a subset of the DRP Object table as a reference catalog. As the DRP Object table is regenerated on the same schedule as the template images used in Alert Production, we should always be able to guarantee that the reference catalog and the template images are consistent and cover the same area.

Obtaining this catalog from the DRP database should simply be a matter of executing a SQL query, though some experimentation and iteration may be necessary to define this query.

8.1.2 Data Release Production Reference Catalogs

The reference catalog used in Data Release Production is expected to be built primarily from the Gaia catalog, but it may be augmented by data taken by LSST during commissioning (e.g. a short-exposure, full-survey layer). DRP processing will also iteratively update this catalog (utilizing LSST survey data) in the course of a single production, but it is not yet decided whether these changes will be propagated to later data releases.

Constructing the DRP reference catalog is thus more of a one-off activity rather than a reusable software component, and much of the work will be

testing and research to determine how much we need to augment the Gaia catalog.

8.2 Instrument Signature Removal

Two variants of the instrument signature removal (ISR) pipeline will exist for the main camera, with the difference arising from the real-time processing constraints placed by AP. This section outlines the baseline design for DRP, with the differences for AP given in §8.2.1.

- Overscan subtraction: per-amplifier subtraction of the overscan levels, as either a scalar, vector or array offset. After cropping out the first one or two overscan rows to avoid any potential contamination from CTI or transients, a clipped mean or median will be subtracted if using a scalar subtraction (to avoid contamination by cosmic rays or bleed trails), and a row-wise median subtracted if using a vector subtraction. If array subtraction turns out to be necessary (unlikely, especially given the subtraction of a master bias frame later in the process), some thought should be given as to how to avoid introducing extra noise to the image.

- Assembly: per-amplifier treatment of each CCD flavor (e2v and ITL sensors assembly differently) followed by per-CCD / per-raft assembly of the CCDs onto the focal plane. Application of `EDGE` mask-bit to appropriate regions of the CCDs, *i.e.* around the edges of both sensor flavors, and around the midline region of e2v sensors due to distortions from the anti-blooming implant.

- Linearity: apply linearity correction using the master linearity table, marking regions where the linearity is considered unreliable as `SUSPECT`.

- Gain correction: applied for CBP measurements where flat-fielding is not performed; multiply by the absolute gains to convert from ADUs to electrons, and estimate the per-pixel variance.

- Crosstalk: Apply crosstalk correction to the raw data-stream from the DAQ using the appropriate version of the master crosstalk matrix.

- Mask defects and saturation: application of master defect list and master saturation levels to set the `BAD/SAT` bit(s) in the maskplane.

- Full frame corrections:

  – Bias: Subtract the master bias frame.

  – Dark: Subtract an exposure length multiple of the master dark frame, perhaps including the slew time, depending on when the array is cleared.

  – Flats: Divide by the appropriate linear combination of master flats.

  – Fringe frames: this will involve the subtraction of a fringe frame composed of some combination of monochromatic flats to match the night sky's spectrum and the filter in use at the time of observation, though the plan for how this combination will be derived remains to be determined.

- Pixel level corrections:

  – The "brighter-fatter effect": Apply brighter fatter correction using the coefficients from §4.3.14. **XXX Need to add proper section about this and reference it as this is a non-trivial ISR algorithm. Just not sure where to put it.**

  – Static pixel size effects: Correction of static effects such as tree rings, spider legs *etc.* using data from §4.3.13. **XXX As above - needs details and referencing.**

- CTE correction: The method used to correct for CTE will depend on what was needed to fully characterize the charge transfer (see §4.3.15).

- Interpolation over defects and saturation: interpolate over defects previously identified using the PSF, and set the `INTERP` bit in the mask plane.

- Cosmic rays: identification of cosmic rays (see §8.3.1), interpolation over cosmic rays using PSF, and setting of `CR/INTERP` bit(s) in the mask plane.

- Generate snap difference: simple pixel-wise differencing of snaps to identify cosmic rays and fast moving transients for removal is baselined, though a more complex process could be involved (see §xxx[13] for discussion).

[13] Zeljko, on your full read-through could you find where this is and insert the link please?

- Snap combination: the baseline design is for a simple pixel-wise addition of snaps to create the full-depth exposure for the visit. However, provision should be made for a less simplistic treatment in the event that there is a non-negligible mis-registration of the snaps arising from either the telescope pointing or atmospheric effects *e.g.* in the dome/ground layer.

8.2.1 ISR for Alert Production

The ISR for the AP pipeline differs slightly to the one used in DRP due to the realtime processing constraint.

Crosstalk correction will be performed inside the DAQ, where the most recent crosstalk matrix will have been loaded. However, whilst there is therefore no default action for AP, in the event of network outages where the local data buffer overflows and the crosstalk-corrected data is lost, crosstalk correction would need to be applied.[14]

Flat fielding will be performed as for DRP, but because the night sky's spectrum will not be available to AP, the fringe frame subtracted will likely be constructed based on a look-up table which crudely describes the expected night sky spectrum based on the position and phase of the moon at the time of observation.

Note:
Does AP plan on performing "brighter-fatter effect"and tree-ring corrections? No reason why it shouldn't I don't think (and it would likely need to if they were of DECam's magnitude), I am just not sure and should include here if it won't.

8.3 Artifact Detection

8.3.1 Cosmic Ray Identification

Cosmic rays will be identified using a Laplacian edge detection algorithm [**?**]. Laplacian edge detection involves convolving the image (I) with a smoothing function (f) tuned to the size of the expected edge width.

$$L = \bigtriangledown^2 f * I$$

[14] This assumes that alerts are still being generated in this eventuality.

In the case of cosmic rays, this implies a very sharp edge. A natural choice is a Gaussian with small σ. The discrete kernel chosen in [**?**] is

$$\bigtriangledown^2 = \frac{1}{4}\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{1}$$

If just used on the native image, this filter would attenuate the signal from CRs that effect multiple adjacent pixels. So, the image is subsampled by some factor f_s. Before downsampling to the native resolution, the negative pixel values are clamped to zero. The resultant image in native pixelization will be referred to as L^+.

We construct a SNR image by dividing the Laplacian image by the noise in the image and adjusting by the subsampling factor.

$$S = \frac{L^+}{f_s N}$$

This CR detection image can be cleaned further by removing extended sources using a median filter. M_n is the median over a $n \times n$ box.

$$S' = (S * M_5)$$

If the PSF is well sampled, this SNR image can be used to detect the CRs by simply drawing a threshold in SNR. If the PSF is undersampled, the author uses the assumption that undersampled point sources are more symmetric than CRs. By constructing a "fine-structure" image, one can place a minimum contrast between the Laplacian image and the fine structure image that further improves differentiation between CRs and point sources.

$$F = (M_3 * I) - [(M_3 * I) * M_7]$$

So the final CR selection criteria are: $S' > \sigma_{lim}$ and $L^+/F > F_{lim}$. The tuning parameters are: f_s the subsampling rate, σ_{lim} the SNR of the CRs in the detection image, f_{lim} the minimum contrast relative to the "fine-structure" image. Assuming reasonable values for optical images, we will define a set of default values.

For the case of Alert Production cosmic ray detection can be applied to the difference of the individual snaps which should improve the performance of the algorithm within the cores of sources.

Add to SFM section:
We can use something like L.A.COSMIC (or CRBLASTER) but if it is too slow, we can fall back to the SDSS algorithm which does a similar thing, but does no convolutions. We should also consider why we do not use a Canny algorithm instead.

8.3.2 Optical ghosts

We will have a set of optical ghost models. Some of these will be models of stationary ghosts (e.g. pupil ghost). Others will be a set of ghosts produced by point sources as a function of source position and brightness. The structure of the stationary ghosts can be measured using stacked, dithered star fields. The latter will likely be modeled using raytracing tools or measured using projectors.

The stationary ghosts will need to be fit for since they will depend on the total light through the pupil rather than on the brightness of a given source and we do not expect to have the data necessary to compute the total flux over the focalplane in a single thread in the alert production processing. Using the fit to stationary models S and the predictions of the single source ghosts, P, we will construct a ghost image

$$I_g = \Sigma_i S_i + \sigma_j P_j$$

where i runs over the stationary ghost models and j runs over the sources contributing to single source ghosts. We can then correct the image by:

$$I' = I - I_g$$

Point source ghosting:
It may not be possible to do point source ghost correction in alert production. We will know the model of the point source ghosts, but we will not know the location of the bright sources in other chips. Since point source ghosts can appear at significant separations, this may be a source of spurious detections.

dependence on PSF:
The CR rejection algorithm does not depend on the PSF of the image. The single source ghosts may be a function of the PSF, but not very strongly I don't think.

8.3.3 Linear feature detection and removal

Satellite trails, out of focus airplanes, and meteors all cause long linear features in astronomical images. The Hough Transform [**?**] is a common tool used in computer vison applications to detect linear features. Linear features are parameterized by r, the perpendicular distance to the line from the origin and θ, the the angle of r with the x-axis. The (r, θ) space is binned and each pixel in the image adds its flux to all the bins consistent with that pixel location. For bright linear features, the bin at the true location of the feature will fill up because more than one bright pixel is contributing to that location in parameter space. After all pixels have been polled, the highest bins correspond to the linear features in the image.

This works very well in high signal-to-noise images, but is very computationally expensive. It is also susceptible to bright point sources overwhelming faint linear features.

An algorithm that takes care of both of these issues is presented in [37]. We will use this as our baseline. First the image is rescaled to maximize the contrast of faint linear features. Next an edge detection algorithm is run on the image. The reference implementation uses a Canny algorithm [36]. This algorithm produces a set of edges that can then be mined for linear features. They use a probababalistic Hough Transforms [38] to cut down on computational costs. The probabalistic version limits the number of pixels that vote. This results in a list of line segments. The segments are binned in angle and any sement that is outside some tolorance of the mode is culled. This cleaned set of segments is fed to the masking algorithm.

The masking algorithm traverses each line segment found in the previous step by selecting a subregion around the segment and flattening the subregion. A weighted mean of the subregion is computed and any pixels above some threshold are considered part of the trail and masked. The subregion is moved along the segment until the end is reached. This is repeated for every segment.

Outstanding questions:

- Is the rescaling to improve linear feature contrast necessary?
- Can we relax the requirement that the trail spans the image?
- Is the Canny algorithm step actually necessary, i.e. can we run a Hough directly on the detected pixels?
- Can we use the fact that we have access to aircraft transponders to remove some plane trails?

Bickerton writeup:
Note that there is a writeup by Steve Bickerton on a different way to modify the Hough Transform to find satelite trails and it has been tried on HSC, but the paper is not complete. Thus, I didn't use it as the baseline here. The writeup is linked from DM-5872.

8.3.4 Snap Subtraction

8.3.4.1 Cosmic Rays We will need to still run some sort of morphological identifier like the one outlined above. This is because there will be real transients and we still only want to pick out the sharp features as CRs. It will help to have less crowding, so we should do CR rejection on the snap difference if we have it.

8.3.4.2 Ghosts Snap differences will not help with ghosting as the ghosts should difference almost perfectly.

8.3.4.3 Linear features Snap differences will provide significant leverage for masking linear features. Since each segment will appear in at most one snap we can mask based on the pixels marked as detected in the difference images that are part of the trail. This will help in crowded regions. This technique will require running some sort of trail detection algorithm, but the requirements will be less stringent since the image will be so much less crowded.

8.3.5 Warped Image Comparison

Additional artifacts will be detected in DRP by comparing multiple vists that have already been resampled to the same coordinate system. This is similar conceptually to Span Subtraction, but will operate quite differently in practice, in that we do not expect to combine this stage with the morphological detection stages; instead we assume that everything we can detect morphologically will have already been detected.

Instead, this stage will examine the full 3-d data cube (two spatial dimensions as well as the epoch dimension) for outliers in the epoch dimension that are contiguous in the spatial dimensions. This is an extension of traditional coadd outlier-rejection, which can cause spurious rejections of single pixels (or small groups of pixels) due to noise and differing PSFs. This can obviously detect astrophysical transients as well as image artifacts, and this is usually desirable; this stage is responsible for determing which pixels should contribute to our definition of the static sky, and we want to reject astrophysical transients from that as well.

The largest challenge for this algorithm is probably handling highly variable astrophysical sources that are the nevertheless present in most epochs. For these, defining the static sky is more subjective, and we may need to modify our criteria for rejecting a region on a visit as an outlier.

8.4 Artifact Interpolation

This component is responsible for interpolating over small (PSF scale or smaller) artifacts such as cosmic rays. By utilizing the PSF model, this interpolation should be good enough that many downstream algorithms do not need to worry about masked pixels (especially those that do not have a built-in formalism for missing data, such as aperture fluxes or second-moment shapes). Interpolated pixels will also be masked (both as interpolated and with a bit indicating the reason why).

This will likely use Gaussian processes, but an existing implementation in the stack should be considered to be a placeholder, as it only interpolates in one direction (to deal with satellite trails).

Artifact interpolation will not handle regions significantly larger than the PSF size; these must be either be subtracted or masked.

8.5 Source Detection

Detection is responsible for identifying new sources in a single image, yielding approximation positions significance estimates. We expect the same algorithm (or only slightly different algorithms) to be run on single-visit direct images, difference images, and coadds. For difference images, this must include detection of negative and dipole sources.

The output of detection is a set of Footprints (containing peaks).

In the limit of faint, isolated objects and white noise, detection should be equivalent to a maximum likelihood threshold for (at least) point sources, which can be achieved by correlating an image with its PSF and thresholding. Other approaches may be necessary for different classes of objects, such as:

- In crowded stellar fields, we expect to need to detect iteratively while subtracting the brightest objects at each iteration (see e.g. Section 5.1.1).
- Optimal detection of diffuse galaxies may require correlating with kernels broader than the PSF.
- When blending or any signficant sub-threshold objects are present, the noise properties may be sufficiently different from the usual assumptions in maximum-likelihood detection to allow those methods, and an alternate approach may be necessary.
- When processing preconvolved difference images or likelihood coadds, detection will need to operate on images that have already been correlated with the desired filter.
- When operating on non-likelihood coadds and standard difference images, detection may need to operate on images with significant correlated noise.

In deep DRP processing (Section 5.3), detection is closely tied to the deep association and deep deblending algorithms, and may change signficantly from the baseline plan based on developments in those algorithms. For example, we may need to adopt a multi-scale approach to these operations (and background estimation) that essentially merges these into a single algorithmic component with no well-defined boundaries.

8.6 Deblending

Deblending is both one of the most important and one of the most difficult algorithmic challenges for LSST, and our plans for deblending algorithms are best described as a research project at this stage.

The baseline interface takes as input a set of Footprints (including peaks), possibly merged from detections on several images (see Object Generation, and a set of images (related to, but not necessarily identical to the set of deytection images). It returns a tree of HeavyFootprints that contain the deblended images of objects. The tree may have multiple levels, indicating a sequence of blend hypotheses that subdivide an image into more and more objects. There may be different HeavyFootprints for each deblended image (at least one for every band), making the size of all HeavyFootprints comparable to this size of the image data, at least for coadds. Depending on the deblending algorithm chosen, a more compact representation of the deblend results may be possible (in that that it would allow the full HeavyFootprints to be regenerated quickly from the image data).

As deblending may involve simultaneous fitting of galaxy and point source models, it may also output the parameters of these models directly as measurements, in addition to generating a pixel-level separation of neighboring objects that can be used by other measurement algorithms via NeighborReplacement.

Deblending large objects is also closely related to background estimation. Some science cases (focusing on small, rare objects) may prefer aggressive background subtraction that removes astrophysical backgrounds such as intracluster light or galactic cirrus, while other science cases obviously care about preserving these structures (as well as the wings of bright galaxies, which are frequently difficult to model parametrically). Rather than produce independent catalogs with different realizations of the background, it makes more sense to include these smaller-scale astrophysical background features in the deblend tree, which already provides a way to express multiple interpretations of the sky.

The baseline approach to deblending involves the following steps:

1. Define a "template" for each potential object in the blend (a model that at least approximately reproduces the image of the object).
2. Simultaneously fit the amplitudes of all templates in the blend.
3. Remove redundant templates/objects according to some criteria (and loop back to the first step).

4. Apportion each pixel's flux to objects according to the value of the objects amplitude-scaled template at the position of that pixel divided by the sum of all amplitude-scaled templates.

Regardless of the templates used, this approach strictly preserves flux, and it can preserve the morphology of even complex objects in the limit that they are widely separated. The complexity in this approach is of course in the definition of templates and the procedure for dealing with redundancy.

The deblender in the SDSS Photo pipeline uses a rotational symmetry ansatz to derive templates directly from the images. This approach is probably too underconstrained to work in the deeper, more blended regime of LSST, and hence we plan to try at least using various parametric models (both PSF-convolved and not). An ansatz that requires each object to have a approximately uniform color over its image may also be worth exploring, and we may also investigate other less-parametric models such as Gaussian mixtures, wavelet decompositions, or splines. Hybrid approaches, such as using a symmetry ansatz for the brightest object(s) in blends and more constrained models for the rest, will also be explored.

This approach yields exactly only one level of parent/child relationships in the output blend tree; each peak in a blend generates at most one child, and all peaks have the same parent. To extend this to the multi-level tree we expect to need to support all science cases, we expect to repeat this approach at multiple scales – though it is currently unclear exactly how we will treat each scale differently; some possibilities include multiple detections on with different spatial filters and building a tree of peaks based on their detection significance and location.

A key facet of any approach to deblending is to utilize the PSF model as a template for any children that can be safely identified as unresolved. This provides a way to build a deblender that can operate in crowded stellar fields as effectively as a traditional crowded field codes: as the density of the field increases (either as detected by the algorithm or as a function of position on the sky), we can increase the probability with which we identify objects as unresolved. The simultaneous template fit then becomes a simultaneous fit of PSF models, and if we iterate this procedure with detection (after subtracting previously-fit stars), we recover the traditional crowded-field algorithm.

A final major challenge in developing an adequate deblender is characterizing its performance. Not only do we lack quantitative requirements on the deblender's performance, we also lack metrics that would quantify im-

provement in the deblender across science cases. Poor deblender performance will clearly impact existing science requirements, but this sort of indirect testing makes iterative improvement more difficult, and it is certain that some deblender failure modes will adversely affect important science cases without affecting any existing requirements. Deblender development will thus have to also include significant work on characterizing deblender performance.

8.6.1 Single Frame Deblending

In single-frame processing (e.g. DRP's BootstrapImChar and possibly AP's Single Frame Processing pipelines), deblending will be run on individual CCD images, which requires that it work without any access to color information and in some cases only a preliminary model of the PSF (since it may be run before a quality PSF model has been fit).

Because single-epoch images are shallower than coadds, we expect blending to be less severe than in coadds. Combining this with the fact that only a single image is being operated on, it is unlikely the single-epoch deblender will be constrained by memory even if run in a single thread.

8.6.2 Multi-Coadd Deblending

Deep deblending on coadds will require a deblender that can simultaneously process a suite of coadds. This will include at least the deep coadds for each band, but it may also include short-period coadds (again, for each band) and possibly cross-band coadds. Merely keeping all of these in memory together would probably necessitate multithreading to avoid requiring more memory/core than most other pipeline algorithms, but we also expect the number of objects in blends to be large on average and extreme in the worst case, and memory use by the deblender scales with this as well. This will almost certainly require some sort of divide-and-conquer approach in addition to some combination of the already-complex algorithmic concepts described above.

The outputs of the deep deblender will need to be "projected" to images other than the coadds actually used by the deblender. This includes at least PSF-matched coadds (which will have the same pixel grid but different PSFs) and possibly individual epoch images (which will have different pixel grids and different PSFs) for forced photometry and multi-epoch fitting; see Section 8.7.3.2 for more information.

Figure 15: Matrix showing combinations of measurement variants, algorithms, and deblending approaches that will be implemented.

8.7 Measurement

Source and object measurement involves a suite of algorithmic components at different levels; it is best thought of as a matrix (see Figure 15) of drivers and algorithms. Drivers correspond to a certain context in which measurement is performed, and are described in Section 8.7.1. Drivers iterate (possibly in parallel) over all sources or objects in their target image(s), and execute measurement algorithms on each; each measurement algorithm (see Section 8.7.2) processes either a single object or a group of blended objects. One of the main tasks of the drivers is to help the algorithms measure blended objects; while some algorithms may handle blending internally by simultaneous fitting (Section 8.7.3.3), most will be given deblended pixels by the driver, which will utilize deblender outputs and the neighbor-replacement procedure described in Section 8.7.3.1 to provide the algorithms with deblended images.

8.7.1 Drivers

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.

8.7.1.1 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 8.7.3.1).

Single Frame Measurement is run in both AP's Single Frame Processing pipeline) and DRP's BootstrapImChar, RefineImChar, and FinalImChar.

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.

8.7.1.2 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.

8.7.1.3 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 Exposure 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

not 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.

8.7.1.4 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 (8.7.3.3), which may in turn require hybrid galaxy/star models (8.7.3.4). These models may then be used as templates for deblending and replace-with-noise (8.7.3.1) 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 pipeline 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.

8.7.1.5 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 in the course of multi-epoch measurement.

Because difference imaging reduces blending substantially, forced measure-

ment 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.

8.7.2 Algorithms

8.7.2.1 Centroids Centroid measurements are run on single images to measure the position of objects. Despite the name, these don't measure just the raw centroid of the photons that correspond to an object; we generally also expect our centroid algorithms to correct for offsets introduced by convolution with the PSF. While they may not be implemented this way, centroid algorithms should thus return results that are equivalent to the best-fit position parameters of a PSF-convolved symmetric model. This model should be a delta function for unresolved objects and something approximately matched to the inferred size of extended objects.

When run in the very first stages of processing, a full PSF model will not be available, making PSF correction impossible, and here centroid measurements will be expected to yield the raw centroid of the light. Note that this must still be corrected for any weighting function used by the algorithm.

Centroids will probably be run independently on each coadd during Multi-Coadd Measurement, to allow for centroid shifts due to proper motion in short-period coadds. Centroid measurements are superceded by Moving Point Source Models and Galaxy Models in MultiFit, which impose different models for centroid differences between epochs that are consistent with morphology. Forced Measurement in production will never include centroid measurement, as the goal is explicitly to measure photometry at predetermined positions, but it may be useful to have the capability to centroid in forced measurement for diagnostic purposes.

8.7.2.2 Pixel Flag Aggregation The pixel flag "measurement algorithm" simply computes summary statistics of masked pixels in the neighborhood of the source/object. This provides a generic way to identify objects impacted by e.g. saturation or cosmic rays while allowing other measurement algorithms to ignore these problems (especially when they have been interpolated over.

8.7.2.3 Second-Moment Shapes Shape measurements here are defined as an estimate of a characteristic ellipse for a source or object that does *not* attempt to correct for the effect of the PSF, corresponding to the second moments of its image. To make this measurement practical in the presence of noise, a weight function must be used, and our baseline plan is to use an elliptical Gaussian matched (adaptively) to the shape of the image. This may be unstable for sources with extremely low SNR, and for these the PSF, a fixed circular Gaussian, or a top-hat may be used as the weight function. We may also include regularization that ensures the size of the object is no smaller than the size of the PSF.

To enable downstream code to correct shapes for the PSF, the shape algorithm must also measure the moments of the PSF model at the position of every object or source (though we expect the best PSF-corrected shape measures for galaxies to come from Galaxy Model Fitting).

8.7.2.4 Aperture Photometry Aperture photometry here refers to fluxes measured within a sequence of fixed-size (i.e. same for all objects) circular or elliptical annuli. The radii of the annuli will be logarithmically spaced radii, though fluxes at the largest largest radii will not be measured for objects significantly smaller than those radii. Together these aperture fluxes will provide a measurement of the radial profile of the object.

The baseline plan for LSST is to use circular apertures, but we also plan to investigate using ellipses, which would provide more meaningful and higher SNR measurements if problems in robustly defining per-object (and perhaps per-radius) ellipses for faint objects can be solved.

While aperture fluxes with radii much larger than the pixel size can be measured naively by simply summing pixel values, smaller apertures will be measured using the sinc interpolation algorithm of [5], which integrates exactly over sub-pixel regions. To avoid contamination from bleed trails when measuring heavily saturated objects, we plan to measure fluxes within

azimuthal segments of annuli instead of circular regions; any the flux within any contaminated segments can be replaced by the mean of the remaining segments (thus assuming approximate circular symmetry).

For aperture fluxes with radii close to the PSF size to be scientifically useful, they must be performed on PSF-matched images. We thus expect to plan run aperture photometry only on PSF-matched coadds and visit-level images, with the latter accompanied by a caution that smaller apertures may not be meaningful without user-level correction for the PSF.

8.7.2.5 Static Point Source Photometry In single-visit, difference image, and forced measurement, PSF fluxes will be measured with position held fixed at the valued determined by the Centroid algorithm, with only the amplitude allowed to vary. We will not use per-pixel weights (as these can lead to bias as a function of magnitude when the PSF model is slightly incorrect) to fit for the amplitude, but we will use per-pixel variances to compute the uncertainty on the flux. PSF fluxes will be aperture corrected (see Section 8.12).

In multi-coadd measurement we may use either static point source models or moving point source models to estimate PSF fluxes; this depends on the number and depth of short-period coadds, and hence it is likely we will use static point source models for early data releases and moving point source models near the end of the survey. In either case we expect these measurements to be entirely superceded for science purposes by multi-epoch fitting results using a moving point-source model; these measurements on coadds are largely for QA and to warm-start multi-epoch fitting.

8.7.2.6 Kron Photometry Kron fluxes are aperture fluxes measured with an aperture radius set to some multiple (usually 2 or 2.5) of the Kron radius, which is defined as:

$$R_{\rm kron} = \frac{\sum r\, I(r)}{\sum I(r)}$$

In our implementation, we use an elliptical aperture (and compute the above radius using elliptical moments), using the Second-Moment Shape to set the ellipticity.

Measuring the Kron radius itself is difficult in the presence of noise; as with any moment measurement, pixels far from the center with low SNR are

given higher weight than central pixels with high SNR. In practice, the sums over pixels in the Kron radius definition must be truncated at some point, and the resulting Kron radius can be sensitive to this choice. Our current approach uses a fixed multiple of the Second-Moment Shape ellipse. This may be less robust than an adaptive approach, but it more closely matches the procedure used by SExtractor's `MAG_AUTO`, which is by far the most popular implementation of Kron photometry in astronomy.

8.7.2.7 Petrosian Photometry

> **Note:**
> Need to get RHL to write this section.

- Compute Petrosian radius.
- Requires taut splines and more robust measurement of standard elliptical aperture suite.
- Compute flux in elliptical aperture at multiple Petrosian radius.

8.7.2.8 Galaxy Models Galaxy models will be fit to all objects in both Multi-Epoch Measurement and Multi-Coadd Measurement. Coadd fitting may be performed only on deep coadds and used to warm-start multi-epoch fitting that would supersede it, but it may also be run on PSF-matched coadds to generate consistent colors (the consistent colors referred to by the DPDD may be derived from galaxy models fit to PSF-matched coadds or aperture fluxes on PSF-matched coadds, but they may also be derived from multi-epoch fitting results).

The baseline plane for the galaxy models themselves is a restricted bulge + disk model, in which the two components are restricted to the same ellipticity and the ratio of their radii is fixed; practically this is more analogous to a single Sersic model with a linearized Sersic index. This may be extended to models with more flexible profiles and/or different ellipticity and radii for the two components if these additional degrees of freedom can still be fit robustly. Bayesian priors and possibly other regularization methods will likely be necessary even with the baseline degrees of freedom.

Designing and constraining priors that provide the right amount of information in the right way is a major challenge. One possibility is an empirical

prior derived from external datasets such as deep HST fields and precursor ground-based surveys, which would almost certainly require custom processing of those datasets using the models intended for production use. Hierarchical modeling – in which the prior is derived from the LSST wide survey itself as individual objects are fit – is unlikely to be feasible (a naive implementation would either introduce *several* full-survey, all-object sequence points in the processing or treat galaxies processed late differently from those processed early). An empirical prior derived from LSST special program data (e.g. deep drilling fields) or previous data releases would be feasible, however, and should be considered. Even an ideal prior that reflects the true distribution of galaxy parameters may not be appropriate for galaxy photometry, however; fluxes must be rigorously defined to be unbiased to changes in observing conditions, and are most useful when they can be defined in a way that is redshift-independent as well. The "correct" Bayesian prior *explicitly* treats galaxies with different radii differently, making both of these properties harder to guarantee. As a result, the prior we use for fitting may be some compromise between the statistically appropriate distribution and a regularization that attempts to reconcise Bayesian modeling with the requirements of traditional maximum-likelihood photometry.

In addition to maximum-posterior fitting, we will draw Monte Carlo samples (nominally 200 samples per object, at least on average) from the posterior in multi-epoch fitting mode. Fitting this within LSST's computational budget will be a serious challenge, requiring new algorithm development in several areas:

- Evaluating PSF-convolved galaxy models on every epoch at every sample point or optimizer iteration is extremely expensive. Because galaxy models are generally massively undersampled before convolution with the pixel grid, naive pixel convolution is impossible without considerable subsampling, which genreally makes it computationally impractal. Fourier-space methods require galaxy models with anayltic Fourier transforms as well as a great deal of care in accounting for the differences between discrete and continuous Fourier transforms. Multi-Gaussian and Multi-Shapelet approximation methods are only computationally feasible if the PSF can consistently be approximated well by those functions, which may not be known until relatively late in commissioning. It may be possible to combine the multi-Gaussian and Fourier-space convolution approaches by using multi-Gaussian approximations to galaxy

models to evaluate them efficiently in Fourier space. We may also be able to address large residuals from multi-Gaussian/multi-Shapelet PSF approximations by convolving the residuals themselves with a simple proxy for the galaxy model (which could be a delta function for small galaxies) and adding this as a correction to the multi-Gaussian/multi-Shapelet convolution.

- Most Monte Carlo methods require many more than 200 draws to converge to a fair sample (for galaxy-fitting problems, $\sim 10^4$ is common). We plan to use importance sampling in multi-epoch fitting, starting with samples drawn from the posterior distribution in coadd-fitting (where we can evaluate likelihoods faster by a factor of the number of exposures in each band on average). These samples must be self-normalized, which introduces a bias that may be significant if the number of samples is small, and it is currently unclear whether this will be a problem in our case. It is also unlikely we will be able to draw as many as $\sim 10^4$ samples even in coadd measurement in order to achieve convergence there. Results from fitting with a greedy optimizer first should provide enough information allow for fair and efficient sampling with a smaller number of draws, but devising a sampling algorithm to make use of that information may be challenging.

Challenges in galaxy modeling are not limited to sampling; the effective number of galaxy model evaluations involved in a typical greedy optimizer fit is also at least 200, if evaluations needed to estimate derivatives via finite differences are included (and analytic derivates are usually not significantly faster than numerical derivatives). Galaxy model parameterizations are intrinsically difficult for most optimizers near the zero radius limit, as this forces the derivative of the model with respect to other parameters (such as ellipticity) to approach zero as well. Issues with Bayesian priors causing flux biases and the general lack of sufficient information to constrain more complex models are also present for optimizer-based fitting. Priors are perhaps a larger concern for fitting than sampling, in fact, because users can reweight samples to replace a DM-selected prior with a prior of their own choosing, but this is only approximately possible for optimizer results.

Galaxy models may be fit simultaneously to multiple objects (see Simultaneous Fitting) as well as fit to individual objects after replacing neighbors with noise. In simultaneous fitting, it will sometimes be inappropriate to fit all objects in a blend with galaxy models. Fitting Hybrid Models that

can transition smoothly between a galaxy model and a Moving Point Source Model is one approach to avoid fitting all permutations of model types to a blend.

[**TODO:**
Cite Lensfit paper for restricted bulge-disk model. Cite Hogg and Lang, Bosch 2010 for Gaussian/Shapelet approximation.]

8.7.2.9 Moving Point Source Models In the Multi-Epoch Measurement, all objects will be fit with a moving point source model that includes proper motion and parallax as free parameters in addition to positional zeropoint and per-band amplitudes. This model may be extended to include parameterized variability or per-epoch amplitudes if this can be done without degrading the astrometric information that can be extracted from the fit. Moving point source models may be fit in Multi-Coadd Measurement as well if the suite of coadds contains enough short-period coadds to constrain the fit, but these results will be fully superseded by the Multi-Epoch Measurement results.

Bayesian priors may be used in the fit (making this "maximum posterior" instead of "maximum likelihood"), if necessary to ensure robustness when fitting for faint objects or if they significantly improve the quality of the results. These will generally be global and relatively uninformative (reflecting e.g. the expected distribution of proper motion of stars as a function of apparent magnitude), but may be highly informative for stars that can be unambiguously associated with the Gaia catalog, if including Gaia and LSST astrometric solutions at the catalog level proves inconsistent with this (more rigorous) Bayesian approach to including Gaia data at the pixel level. All priors will be reported, but unlike Monte Carlo samples, results from a fit with a greedy optimizer cannot be reweighted to change to a user-provided prior except in a perturbative, first-order sense. Monte Carlo sampling with moving point source models is not included in the baseline plan, but will be considered if it proves important for joint fitting of blended stars and galaxies (see Hybrid Models, below) or Star/Galaxy Classification, and it can be done without significantly affecting the compute budget.

[**TODO:**
Cite Lang+Hogg paper that did this in Stripe 82.]

8.7.2.10 Trailed Point Source Models

[**Note:**
Need to find someone (probably on AP team) to write this section.]

- Fit PSF convolved with line segment to individual images

8.7.2.11 Dipole Models

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

Arising primarily due to slight astrometric alignment or PSF matching errors between the two images, or effects such as differential chromatic aberration, flux "dipoles" are a common artifact often observed in image differences. These dipoles will lead to false detections of transients unless correctly identified and eliminated. Importantly, dipoles will also be observed in image differences in which a source has moved less than the width of the PSF. Such objects must be correctly identified and measured as dipoles in order to obtain accurate fluxes and positions of these objects.

Putative dipoles in image differences are identified as a positive and negative source whose footprints overlap by at least one pixel. These overlapping footprints are merged, and only the sources containing one and only one positive and negative merged footprint are passed to the dipole modeling task. There is a documented degeneracy (http://dmtn-007.lsst.io) between dipole separation and flux, such that dipoles with closely-separated lobes of high flux are statistically indistinguishable from ones with low flux and wider separations. We remove this degeneracy by using the *pre-subtraction images* (i.e., the warped, PSF-matched template image and the pre-convolved science image) to constrain the lobe positions (specifically, to constrain the centroid of the positive lobe in the science image and of the negative lobe in the template image). This is done by first fitting and subtracting a second-order 2-D polynomial to the background within a subimage surrounding each lobe footprint in the pre-subtraction images to remove any flux from

background galaxies (we assume that this gradient, if it exists, is identical in both pre-subtraction images). Then, a dipole model is fit simultaneously to the background-subtracted pre-subtraction images and the image difference.

The dipole model consists of positive and negative instances of the PSF in the difference image at the dipole's location. The six dipole model parameters (positive and negative lobe centroids and fluxes) are estimated using non-linear weighted least-squares minimization (we currently use the Levenberg-Marquardt minimization algorithm). The resulting reduced χ^2 and signal-to-noise estimates provide a measure by which the source(s) may be classified as a dipole.

We have tested the described dipole measurement algorithm on simulated dipoles with a variety of fluxes, separations, background gradients, and signal-to-noise. Including the pre-subtraction image data clearly improves the accuracy of the measured fluxes and centroids. We have yet to thoroughly assess the dipole measurement algorithm performance on crowded stellar fields. Such crowded fields may confuse the parameter estimates (both centroids and/or fluxes) when using the pre-subtraction images to constrain the fitting procedure, and in such situations, we may have to adjust the prior constraint which they impose.

Note that deblending dipole sources is a complicated process and we do not intend on implementing such an algorithm. As with all fitting algorithms, speed may be a concern. We will optimize the dipole measurement for speed.

8.7.2.12 Spuriousness

[**Note:**
Need to find someone (probably on AP team) to write this section.]

- Some per-source measure of likelihood 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.

8.7.3 Blended Measurement

Most LSST objects will overlap one or more of its neighbors enough to affect naive measurements of their properties. One of the major challenges in the deep processing pipelines will be measuring these objects in a way that corrects for and/or characterizes the effect of these blends.

The measurement algorithms of Section 8.7.2 can be split up broadly into two categories:

- weighted moments (includes Second-Moment Shapes, Aperture Photometry, Kron Photometry, and PetrosianPhotometry;
- forward modeling (includes Galaxy Models, Moving Point Source Models, and Trailed Point Source Models).

Most measurements that involve the PSF or a PSF-convolved function as a weight function can be interpreted in either way (this includes most Centroid algorithms and all PSF flux algorithms), though only the weighted-moment interpretation provides a motivation for ignoring per-pixel variances, as is necessary to ensure unbiased fluxes in the presence of incorrect models.

The statistical framework in which weighted moments make sense assumes that each object is isolated from its neighbors. As a result, our only option for these measurements is removing neighbors from the pixel values prior to measurement, which we will discuss further in 8.7.3.1.

In forward modeling, we convolve a model for the object with our model for the PSF, compare this model to the data, and either optimize to find best-fit parameters or explore the full likelihood surface in another way (e.g. Monte Carlo sampling). We can use the removing-neighbors approach for forward fitting, simply by fittting each object separately to the deblended pixels. However, we can also use simultaneous fitting (Section 8.7.3.3), in which we optimize or sample the models for multiple objects jointly.

Both neighbor-replacement and simultaneous fitting have some advantages and disadvantages:

- Neighbor-replacement provides no direct way to characterize the uncertainties in an object's measurements due to neighbors, while these are naturally captured in the full likelihood distribution of a simultaneous fit. This likelihood distribution may be very high-dimensional in a fit that involves many objects, however, and may be difficult to characterize or store.

- Neighbor-replacement generally allows for more flexible morphologies than the analytic models typically used in forward fitting, which is particularly important for nearby galaxies and objects blended with them; simultaneous fitting is only statistically well-motivated to the extent the models used can reproduce the data.

- Once neighbor-free pixels are available, fitting objects simultaneously will almost always be more computationally expensive than fitting them separately to the deblended pixels. At best, simultaneous fitting will have similar performance but still require more complex code. And because we will need to deblend pixels to support some measurement algorithms, we'll always have to deblend whether we want to subsequently do simultaneous fitting or not.

8.7.3.1 Neighbor Noise Replacement We do not perform measurements directly on the deblended-pixel HeavyFootprints output by the Deblender for two reasons:

- The deblended pixels typically have many zero entries, especially for large blend families (i.e. many pixels for which a particular object has no contribution). These zero pixels make the noise properties of a deblended object qualitatively different from those of an isolated object, which may be problematic for some measurement algorithms.

- Many measurements utilize pixels beyond the blend family's Footprint, and in fact may extend to pixels that are in another family.

To address these issues, we measure deblended objects using the following procedure:

1. Replace every above threshold pixel in the image (all Footprints) with randomly generated noise that matches the background noise in the image.

2. For each blend family:

   (a) For each child object in the current blend family:

      i. Insert the child's HeavyFootprint into the image, replacing (not adding to) any pixels it covers.

ii. Run all measurement algorithms to produce *child* measurements.

iii. Replace the pixels in the child's Footprint region with (the same) random noise again.

(b) Revert the pixels in the parent Footprint to their original values.

(c) Run all measurement algorithms to produce *parent* measurements.

(d) Replace the parent Footprint pixels with (the same) random noise again.

This procedure double-counts flux that is not part of a Footprint, but this is considered better than ignoring this flux, because most measurement algorithms utilize some other procedure for downweighting the contribution of more distant pixels.

8.7.3.2 Deblend Template Projection When deblending is performed on one image and measurement occurs in another, the deblender outputs must be "projected" to the measurement image. In general, this requires accounting for three potential categories of differences between the images:

- differing coordinate systems
- differing PSFs
- different epoch.

We do not currently have a use case for projecting deblender results between images with different filters; we expect that we will have deblender results from at least each per-band coadd, and projection is required for different per-epoch images in Forced Measurement and Multi-Epoch Measurement. It may be entirely unnecessary if Simultaneous Fitting can be used to address all blended measurement issues in these contexts.

When variability can be ignored, deblended pixel values can be resampled using the same algorithms that operate on images, and PSF Matching kernels can be used to account for PSF differences (though some regularization will be required if this involves a deconvolution). When variability cannot be ignored, these operations should instead be applied to the deblend templates, which can then be re-fit to produce new per-epoch deblend results.

These operations are significantly easier if the deblend templates themselves are defined via analytic models that must be convolved with the PSF to generate the template; the models can simply be transformed, convolved with the per-epoch PSF and re-fit.

8.7.3.3 Simultaneous Fitting For measurement algorithms that can be fully expressed as a likelihood-based fit using a model that closely approximates the data, an alternative to Neighbor Noise Replacement is to fit all objects in a blend simultaneously (either with a greedy optimizer or with Monte Carlo sampling). This is statistically straightforward: the parameter vectors for individual per-object models can simply be concatenated, and the pixels to fit to are the union of all of the pixels that would be used in fitting individual objects.

Simultaneously fitting a group of objects is almost certainly slower than fitting those objects individually, in essentially every case, but the decrease in performance may be mild. The per-iteration calculations in optimizer-based methods have a worst-case complexity that is $O(N^2)$ in the number of parameters, but the $O(N)$ evaluation (and convolution) of models typically completely dominates the $O(N^2)$ linear algebra, and this should hold for all but the very largest blend families. The more important scaling factor is the number of iterations required to converge to the solution; as this is expected to scale roughly with the volume of the space that much be searched, it could scale as poorly as $O(k^N)$. This can be ameliorated by starting the optimizer close to the correct solution; fitting objects independently to deblended pixel values before simultaneous fitting should provide a reasonably good guess. Simultaneous fitting in Multi-Epoch Measurement can also be initialized from the results of Multi-Coadd Measurement, where model evaluations are significantly faster.

With Monte Carlo methods, a high-dimensional parameter space is less of a problem; Monte Carlo methods are valued precisely because they scale (on average) better with dimensionality. We still expect to require warm-start methods for simultaneous Monte Carlo sampling of multiple objects, such as using importance sampling in Multi-Epoch Measurement to re-weight samples drawn during independent per-object sampling or Multi-Coadd Measurement.

For both optimizer-based fitting and Monte Carlo sampling, it should be possible to explore the parameter space in a way that does not require evaluating the model for every object at every iteration or draw; because

objects on opposite sides of a blend should affect each other only weakly, optimizer steps and samples that explicitly ignore these weak correlations in choosing the next parameter point to evaluate may be more efficient. In Monte Carlo sampling, this would probably utilize techniques from Gibbs Sampling; with optimizers, one can probably ignore the step for any objects whose optimal step size falls below some threshold (with some extra logic to guarantee these objects are are still sometimes updated). These improvements would likely reuqire significant development effort, and they would almost certainly make using off-the-sheld optimizers and samplers impossible.

The fact that simultaneous fitting itself may be used to produce templates for deblending – and that simultaneous fitting may require non-simultaneous fitting, using deblended results, for a warm start – suggests that the boundary between the Deblending and measurement algorithmic components may be somewhat vague. This could be represented in software as an iteration between these algorithmic components, or perhaps a hierarchy of components in which the same low-level fitting code is used (and extended) by both algorithmic components, which can then be run straightforwardly in sequence.

8.7.3.4 Hybrid Models In simultaneous fitting, the model used to fit one object can affect the quality of the fit to its neighbor, making it important the the best model be used for each object. We explicitly fit both Moving Point Source Models and Galaxy Models to every object, however, precisely because we do not expect to always be able to securely classify objects well enough to know which model is better (and we certainly do not expect to be able to classify them before fitting these models). In simultaneous fitting, trying all possibilies leads to a combinatorial explosion of model-fitting jobs (fitting each of two models to N objects leads to 2^N fits).

Given that both of these models have a static point source as a limit, and classification is hardest at this limit, making the right classification for neighbors may not be critical; misclassified objects would still end up being fit with a model that is broadly appropriate for them. Even in this case, we would still have $2N$ fitting objects when fitting an N-object blend with two models: every object would be fit twice as the "primary" object (with both models), and then twice for each of its neighbors. Given that each of these fitting jobs still involves fitting N objects, this still results in a scaling of approximately $2N^2$ (clever optimizers and samplers could probably reduce this, with a cost in code complexity and development time).

Another option would be to fit both models simultaneously, by introducing a higher-level boolean parameter that sets the type. Sampling from this hierarchical model is not significantly difficult than sampling from either of the original models if naive samplers are used, but optimizers and samplers that rely on derivative information will likely have trouble dealing with the discrete type parameter. It may be possible to define a smooth transition between the two models through the static point source limit they share, though this would likely require customization of the optimizer and sampler as well. Sampling with this sort of hybrid model would naturally produce samples from both models in the proportion weighted by the marginal probability of those models, which is essentially ideal (assuming sampling is considered useful for Moving Point Source Models). Using an optimizer with hybrid models would result in a result for just the best-fit model, which is somewhat less desirable.

8.8 Spatial Models

In many areas we will need to represent spatial models. This will include models fit to sparse and non-uniformly sampled data. We will support fitting Chebyshev polynomials and splines. We will also support regression techniques like Gaussian Processes.

8.9 Background Estimation

8.9.1 Single-Visit Background Estimation

Background estimation will be done on the largest scale feasible first. In the case of Alert Production, this may be on the size of a chip. In DRP, we expect this to be on a full focalplane. An initial low order estimate will be made on a large scale. Each chip will be divided into subregions. For each subregion, a robust average of the non-masked pixels will be computed. All values for all chips will be fit by an appropriate function (see §8.8). This will provide a low order background estimation in focal plance coordinates. Note that this can only be done if the Instrument Signature Removal is very high fidelity. Any sharp discontinuity could cause problems with fitting a smooth function.

A higher order background model can be computed per chip. First, the low order background is subtracted from the image. The non-masked pixels will again be binned on a finer grid avoiding bright objects. The median in

each bin is fit by an appropriate function. In practice, this process will likely be iterative.

In the case of Alert Production, there will be no full focalplane model since we expect to process only a single chip in each thread. In this case, we constrain the background with the available un-masked pixels without removing a global background first. Note that image differencing is still possible even in the scenario where there are no unmasked pixels in the science image. The background can be modeled as a part of the PSF matching process. We will want to do background modeling and subtraction in Alert Production when possible because we will want to do calibrated photometry. Even though these measurements are not persisted for science use, they will be very useful for debugging and QA.

If there are so few un-masked pixels in the entire focalplane that even a low order global background is impossible to model, background modeling may need to be iterated with a procedure that models and subtracts stars (for example, see the BootstrapImChar pipeline in DRP).

Crowded fields and composition:
Requirements include working in crowded fields. I think estimating a full focalplane model is the best we can do. If there are no unmasked pixels in the entire FoV, I don't think there is much we can do. I didn't explicitly talk about composition of background models, but this takes that into account by allowing a global model to be subtracted from the single chip image before a higher order model is fit.

8.9.2 Coadd Background Estimation

A variant of Single-Visit Background Estimation that is run on coadds to model the remaining background that is not removed during background matching. This includes the observational background from the reference image

8.9.3 Matched Background Estimation

A variant of Single-Visit Background Estimation for use on difference images produced in Background Matching. This will be able to operate on full visit scales with much less concern for oversubtracting bright objects, which may allow it to use qualitatively different algorithms. It may also be able to use models specifically designed to subtract specific ghosts or stray light patterns.

8.10 Build Background Reference

8.10.1 Patch Level

Background-matching each `CoaddTempExp` to a reference exposure performs comparably to fitting the offsets to the N(N-1)/2 difference images, however the co-add quality will depend on the quality of the reference image. Choosing a reference image on a per-patch basis is as simple as choosing the `CoaddTempExp` that maximizes coverage and is the highest weighted component in the chosen weighting scheme: e.g. minimum variance, optimum point source SNR.

Coverage is defined as the fraction of non-NaN pixels in the `CoaddTempExp`. NaN pixels arise in `CoaddTempExp`s because of gaps between the chips and edges of the visit focal planes. The camera design specifications indicate a 90% fill factor, and thus approximately 10% of pixels will be NaN due to chip gaps. The SNR of the background can be estimated from either the `CoaddTempExp`s themselves, using the variance plane of pixels without the source detection bit mask flagged, or from calibration statistics such as the zero point (a proxy for transparency). In the limiting case that all `CoaddTempExp`s have the same coverage, finding the best reference image reduces to the problem of weighting epochs in co-addition.

For example, the reference image that minimizes the variance in the co-add is the minimum variance `CoaddTempExp`, and the reference image that maximizes SNR in coadd point source measurements is the `CoaddTempExp` with the maximum $T_i^2/\mathrm{FWHM}_i^2\sigma_i^2$, where T_i is the normalization factor necessary to put the input `CoaddTempExp`s on the same photometric system (a proxy for the atmospheric transparency), and σ_i^2 the average variance of the pre-scaled exposure. By combining one of these statistics with coverage, we can construct an objective/cost function that relates the importance of coverage and sky-background, and can select a visit that minimizes that quantity objective function.

8.10.2 Tract Level

Constructing reference images for tract-sized co-adds follows the same principle, but requires maximizing the SNR/coverage of a large mosaic constructed from multiple visits. Algorithms for mosaicking partially overlapping images have been well established [e.g. **?** **?**]. By mosaicking visits, applying additive scalar or sloping offsets to calexps, we can generate a tract-sized reference image. Algorithms for selecting visits to construct these fall on a spectrum

of computational expense. On the less expensive side is a greedy algorithm which starts with a "best" (as defined above) visit chosen at the center of the tract. Visits can be chosen, scaled, and added one by one in the vicinity, moving outwards. Another option is to choose a small set of visits that completely cover a tract without gaps, which can b cast as a constrained convex optimization problem[15], and mosaic them using standard mosaicking techniques. Finally, the most expensive option would use all the visits to simultaneously tie the visits together using all overlaps while background matching.

- 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.

8.11 PSF Estimation

8.11.1 Single CCD PSF Estimation

Single CCD PSF estimation needs to be run in both Alert Production and in in Data Release Processing. In Alert Production it will be the final PSF model for both direct and difference image measurement. In Data Release Processing, it will be used as an initial bootstrapping step to start off image characterization. We do not intend to include chromatic effects in the PSF at the single CCD estimation phase.

The first step is to select a set of suitable stars to use as PSF exemplars. This can be done by finding clusters in second moment space. In production, we expect that an external catalog with PSF candidates that have been show to be non-varying and isolated will produce better results.

Once a set of candidate stars is selected each star is fit by a set of appropriate basis functions: e.g. shapelets. The PSF is approximated by

$$P = \Sigma_n c_n \Psi_n$$

where Ψ_n is the n^{th} basis function, and c_n is the coefficient for that basis function. We can solve for the coefficients in the least squares sense using a QR decomposition or similar technique. We then have an estimate of the PSF at several locations on the chip. For each of the coefficients we can fit 2D Chebyshev polynomials to each coefficient to model the spatial variation in

[15] probably

each component (see 8.8). By interpolating the fit coefficients, we can derive an estimate of the PSF at any point in the chip.

The order of the spatial model cannot excede number of PSF exemplars in the frame. If there is only a single PSF candidate star, we will assume the PSF is constant across the CCD. In the case of no PSF candidate stars, we will assume a double Gaussian PSF with width set by observation metadata: e.g. FWHM from the guiding system.

Do we need something more complex?:
We can get arbitrarily complex, but I don't thinkn we need a more complex system in the baseline until we show this won't work. Some have considered using Gaussian process regression for this, but I don't know that anyone has formalized that.

8.11.2 Wavefront Sensor PSF Estimation

- 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.
- Assumed to be at least mostly contributed by Telescope and Site.

8.11.3 Full Visit PSF Estimation

- Decompose PSF into optical + atmosphere (and maybe sensor).
- Constrain model with stars, telemetry, and wavefront data.
- Wavelength-dependent.
- Used in *ImChar in DRP.

- Must include some approach to dealing with wings of bright stars (not trivial)

8.12 Aperture Correction

- Measure curves of growth from bright stars (visit-level, at least in DRP)
- Correct various flux measurements to infinite (CCD-level). Stellar measurements are easy, galaxies are hard (impossible to do formally correctly?).
- Probably want sequence of apertures to agree asymptotically?
- Propagate uncertainty in aperture correction to corrected fluxes; covariance is tricky.

8.13 Astrometric Fitting

8.13.1 Single CCD

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

AP will need to do reasonably good astrometeric calibration on single frames in order to do the relative warping between the template and science images. We have seen that the kernel matching algorithm can take out bulk astrometric errors up to a significan fraction of a pixel. However, we would like to avoid making the kernel matching deal with astrometric errors. Astrometric errors between the science and template coordinate systems should be less than 15mas. We will use the internal reference catalog used in DRP as the reference catalog. This will be based on astrometry from an external source and will be extended using high quality measurements on coadds from DRP.

We can project the per-chip coordinates into a flat (Gnomonic) projection in the focalplane. This will take out the bulk of the optical distortion. Next, we will use a matching algorithm like that outlined in [**?**]. Once we match, a 2D polynomial solution will be fit to minimize the residual.

[**Dependency:**
This introduces a dependency on DRP's internal reference catalog not capture elsewhere.]

8.13.2 Single Visit

Full visit astrometric fitting will be done as a bootstrapping step toward higher quality calibration in DRP. All measurements in the visit will be projected to a tangent plane, taking into account all knowledge of the sensor arrangement and optics. The reference catalog (likely the DRP reference catalog) will be projected to the same tangent plane.

Sources will be matched, again using a [**?**] like algorithm. Once the reference and observations are matched, a multi-component WCS will be fit. We expect the components will be related to residuals on the optical model and will included a component to account for the Von Karman turbulence.

8.13.3 Joint Multi-Visit

In the case where there are multiple visits overlapping the same part of the sky, e.g. a patch, we can leverage multiple realizations to beat down the random contribution of the atmosphere to get a better estimate of the optical model and the atmospheric contribution per visit.

The catalogs are stacked and matched using a multi-matching algorithm like OPTICS. At this point, the measurements can be matched to an external catalog for the purposes of absolute astrometry. With all measurements in hand, a multi-component WCS is fit to all measurements at the same time on order to minimize the residual from the mean position for each object.

Joint astrometric fitting must be able to work both with and without an external reference catalog (while only producing relative results in the latter case, of course).

The first run of this algorithm in DRP may be responsible for correcting source positions to account for DCR. Later runs should be given as input centroids measured with a PSF model that includes (and hence corrects for) DCR, and hence should not need to make this correction.

8.14 Photometric Fitting

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 spatial variation for clouds) to all CCDs simultaneously after matching to reference catalog.

- Need for chromatic dependence unclear; probably driven by AP.
- Might be possible to use a "nightly zeropoint" if calibration fields are taken (e.g., during twilight)

8.14.3 Joint Multi-Visit

In joint photometric calibration, all observations within a tract are combined to jointly estimate the best possible measurement of the relative flux of each source and a spatially-varying gray photometric scaling for each visit.

This includes:

- Deriving SEDs for calibration stars from colors and reference catalog classifications.
- Utilizing additional information from wavelength-dependent photometric calibration built by the calibration products production to convert observed flux to a flux through a standard atmosphere.
- Fitting zeropoint and possibly perturbations to all CCDs on multiple visits simultaneously after matching to reference catalog.

The first step is to solve:

$$m_{i,j} = m_i + z(x, y)_j \tag{2}$$

where $m_{i,j}$ is an observed magnitude of star with true magnitude m_i on observation j, with position-dependent photometric scaling z. Naively, solving Equation 2 would involve simply computing the pseudo-inverse of the sparse m_{ij} matrix. Unfortunately, the inverse of a large sparse matrix is a large dense matrix. Thus one must use iterative solvers such as the LSQR algorithm (a conjugate gradient-type solver) to find the best-fitting values of m_i and the (model-dependent) parameters of z_j.

It is possible that the calibration products production may not produce adequate wavelength-dependent photometric calibration (especially for the atmosphere), requiring this to be included in the fit as well.

8.14.4 Large-Scale Fitting

8.14.4.1 Global Fitting Global photometric fitting (as opposed to tract-level Joint Multi-Visit Photometric Fitting) will most likely not be run in Data Release Production, as we expect to be able to use the Gaia catalogs to calibrate between tracts. The suitability of the Gaia catalogs for this purpose

still needs to be confirmed in detail (it depends on our ability to predict LSST colors from Gaia BP/RP spectra). Even if Gaia is used, a global photometric fit may still be useful as a QA tool (run after DRP) to verify the quality of the global photometric.

Tract-level photometric calibration leaves a "floating zeropoint" in the solution (if you add X to all the m_i's, and -X to all the z_j's the solution is the same). If one solves regions of the sky independently, then the floating zeropoints T_j of each tract need to be matched:

$$p_{i,j} = p_i + T_j \tag{3}$$

This represents a full-survey sequence point.

One open issue is that it's not clear what uncertainties to put in for the different $p_{i,j}$'s (unlike the observed magnitudes where it's relatively easy to calculate a reasonable uncertainty). One must also come up with a method for computing the uncertainties on the returned best-fit parameters.

After solving for all the magnitudes, and merging all the tract-level zeropoints, there's still the final floating zeropoint (in each filter) that needs to be removed. One possibility is to use spectrophotometric White Dwarf standards to set the overall photometric zeropoint since they have spectra that should be theoretically calculated to millimag precision. There's also speculation that Gaia BP/RP spectra could provide a good way to do the flux calibration.

8.14.4.2 Interim Wavelength-Dependent Fitting Most LSST software development will be tested on precursor datasets, such as images taken from the HSC and DECam instruments, and these lack the detailed characterization of wavelength-dependent photometric transmission effects planned for LSST. Even when LSST commissioning data and the LSST atmospheric monitoring telescope are available, the Gaia catalog and monochromatic flats may not be. In order to exercise and commission the pipelines before the full system is operational we will have to include fitting for these chromatic effects in a global photometric fit that can be run on important precursor datasets. This will not obtain the same level of accuracy as the full LSST system, but it will provide wavelength- and spatially-dependent photometric calibrations that are sufficiently accurate to test our ability to apply those corrections to our measurements. When the full LSST photometric calibration system is operational, we expect this functionality to continue to play a

role in QA even though it will probably not be run as part of Data Release Production. Obviously, having this capability also serves as risk mitigation in case some aspects of the planned photometric calibration system fail to perfom adequately.

8.15 Retrieve Diffim Template for a Visit

In difference imaging a major contributor to the quality of the difference image is the choice of template. We expect that the DRP template generation algorithm will be quite complex. It will potentially involve synthesizing multiple monochromatic templates that will be used to model the effects of DCR.

Ideally, the retrieval will be to select the correct bounding box from the correct master template for the current observation. In the simplest implementation, we would build reference templates on a grid of hour angle and positions. If we need a more complicated algorithm for generating reference templates, we expect the template generation algorithm will provide an algorithm to interpret the templates.

8.16 PSF Matching

The essence of image subtraction is to astrometrically register the science image $S(x, y)$ and template image $T(x, y)$, and then match their point spread functions (PSFs) of so that they may be subtracted pixel by pixel. The PSFs are the time–averaged transfer functions of a point source through the Earth's atmosphere, telescope optics, and into the silicon of the detector before being read out. We assume that the science image can be modeled as a convolution of the template image by a PSF–matching kernel $\kappa(u, v; x, y)$, i.e., $S = \kappa \otimes T$. (Indices u, v indicate that the kernel itself is a 2–dimensional function, which varies as a function of position x, y in the image; during convolution and correlation there is an implicit summation over u, v.) Then the difference image, upon which new or variable sources are detected, is given by $D = S - (\kappa \otimes T)$.

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.

The current implementation of the PSF matching algorithm is summarized in detail by Becker, et al. (2013) (`http://ls.st/x9f`). We model the PSF–matching kernel by decomposing it into a set of basis functions $\kappa(u,v) = \sum_i a_i \kappa_i(u,v)$ [30], where the coefficients are determined via ordinary least-squares estimation:

$$\begin{aligned} C_i &\equiv (\kappa_i \otimes T); \\ b_i &= \sum_{x,y} \frac{C_i(x,y) S(x,y)}{\sigma^2(x,y)}; \\ M_{ij} &= \sum_{x,y} \frac{C_i(x,y) C_j(x,y)}{\sigma^2(x,y)}; \\ a_i &= M_{ij}^{-1} b_j. \end{aligned} \tag{4}$$

$\sigma^2(x,y)$ is the per–pixel variance stored in the `variance` plane of each LSST `exposure`. To generate a spatially varying model for the kernel, we further decompose the relative weights of the basis coefficients a_i into spatially-varying low-order polynomials, i.e. $\kappa(u,v;x,y) = \sum_i a_i(x,y)\kappa_i(u,v)$. We also allow for a spatially-varying differential background between the two images $b(x,y)$ that may be fit for using a low–order polynomial [30, 31]. The image difference is then $D(x,y) = S(x,y) - T(x,y) \otimes \kappa(u,v;x,y) - b(x,y)$. There are many choices of baisis functions, e.g. regularized delta functions. However, the basic algoirthm remains unchanged.

The basis functions $\kappa_i(u,v)$ are a degree of freedom in this problem. Following [30], we use a set of nGauss = 3 Gaussians, each with a different width σ_i, and each modified by a Laguerre polynomial to a given order (see below). Following more recent studies studies [e.g. 32], we parameterize these different Gaussian widths via a single ratio β, such that $\sigma_{i+1} = \beta \times \sigma_i$ with $\beta = 2.0$. (We note that all constants are defined by `Config` variables and may be adjusted on a per-use basis). We set the overall scale for the σ by noting that, under the assumption that the PSFs of the images are Gaussian (σ_S for the science image and σ_T for the template image), the σ_κ of the matching kernel should be simply $\sigma_\kappa^2 = \sigma_S^2 - \sigma_T^2$. We use this canonical width for the central Gaussian in the basis sequences (i.e., $\sigma_{i=2} \equiv \sigma_\kappa$ when using three Gaussians bases). Each of the three default

kernel basis functions are modified by Laguerre polynomials up to order degGauss = $[4, 2, 2]$, respectively. This results in a total number of (non-spatially varying) bases of $\sum_i^{\mathrm{nGauss}}(\mathrm{degGauss}_i + 1) \times (\mathrm{degGauss}_i + 2)/2$, or 27 given the aforementioned defaults.

A spatially-invariant matching kernel $\kappa(u, v)$ is determined separately for image substamps centered on multiple kernel candidates across the image. The kernel candidates are selected using the `DiaCatalogSourceSelector` to query the appropriate reference catalog for appropriate sources to use for PSF matching. This selector allows the user to specify the brightness and color range of the objects, toggle star or galaxy selection, and to include variable objects or not. Sources are vetted for signal-to-noise and masked pixels (in both the template and science image). The matching (spatially-invariant) kernel models $\kappa_j(u, v)$, determined for each kernel candidate j as described above, are examined and filtered by various quality measures. The resulting ensemble of filtered kernel models is used to constrain the spatially-varying kernel model $\kappa(u, v; x, y)$ by fitting the spatially-varying basis kernel coefficients $a_i(x, y)$ with a N^{th}-order 2-dimensional Chebyshev polynomial. This results in the final full spatial solution $\kappa(u, v; x, y) = \sum_i a_i(x, y)\kappa_i(u, v)$, which may be evaluated at each location (x, y) in the image for convolution.

Detection on the difference image occurs through correlation of $D(x, y)$ with the science image's PSF, yielding optimally filtered detection image $D'(x, y) = D(x, y) \circ PSF_S(u, v; x, y)$ where $\circ$ denotes correlation (currently the DM stack uses convolution instead of correlation). The values of the pixels in $D'(x, y)$ provide a maximum likelihood estimate of there being a point source at that position. Detection occurs by simply finding pixels that are more than $N \times \sigma$ above the square root of the per–pixel variance. Alternatively, the science image may be *pre-convolved* with a kernel similar to its PSF, in which case the resulting difference image $D(x, y)$ is already filtered for detection. This latter option has the additional advantage that it help us avoid requiring em deconvolution of the template image, in cases when the template has a wider PSF than the science image.

8.16.2 PSF Homogenization for Coaddition

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

In PSF-matched coaddition, input images are convolved by a kernel that matches their PSF to a predefined constant PSF before they are combined. This so-called "model PSF matching" uses the PSF-matching algorithm

described in the previous section to match the PSF *model* from an exposure to a pre-determined template (e.g., a constant-across-the-sky double Gaussian) PSF model. For this task, we realize each PSF model into an exposure-sized grid, and then utilize those as kernel candidates as input for the PSF matching algorithm described above.

8.17 Image Coaddition

- 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

8.18 DCR-Corrected Template Generation

Refraction by the Earth's atmosphere results in a dispersion of an astronomical image along the "parallactic angle". This amplitude of this dispersion depends on the spectral energy distribution (SED) of the source and the refractive index of the atmosphere. Differential chromatic refraction (DCR) refers to the SED dependent refraction within a given photometric passband. For the airmass range of the LSST and its filter complement the amplitude of the DCR could be up to 1.1 arcsec in the u band and 0.8 arcsec in the g band. Image subtraction templates that do not account for DCR will result in dipoles in the subtracted images.

The baseline approach for minimizing DCR induced dipoles in image differences is to selected coadded templates that are close in airmass . This will identify three airmass bins (XXX where is this defined) from which PSF matched coadds will be generated.

Given the sensitivity of the number of false positives to the astrometric accuracy of the registration of images and the dependence of this astrometric accuracy on DCR we plan to define an interpolation scheme for generating DCR corrected templates.

8.18.1 Refraction from the Atmosphere

Refraction is dependent on the local index of refraction of air $n_0(\lambda)$ at the observatory and, as a function of wavelength is given by,

$$\begin{aligned} R(\lambda) &= r_0 n_0(\lambda) \sin z_0 \int_1^{n_0(\lambda)} \frac{dn}{n\left(r^2 n^2 - r_0^2 n_0(\lambda)^2 \sin^2 z_0\right)^{1/2}} \\ &\simeq \kappa(n_0(\lambda)-1)(1-\beta)\tan z_0 - \kappa(1-n_0(\lambda))\left(\beta - \frac{n_0(\lambda)-1}{2}\right)\tan^3 z_0 \end{aligned} \tag{5}$$

with z_0 the zenith distance.

8.18.2 Generating a DCR Corrected Template

Given a set of observed images, $O(x, z)$, at an airmass of z, and assuming that we know the wavelength dependence of the refraction, we can model the corresponding image at the zenith (or any other airmass), $I(x, 0)$. For simplicity, we will consider only a single row of a sensor as comprising an image, that the direction of the DCR is aligned along the row, and that the PSF is constant between images.

The impact of DCR is to move flux between pixels as a function of airmass and wavelength. Refraction, $R(\lambda, z)$, can be treated as a shift operator or a convolution, $D(\lambda, z)$, and is known given the refractive index of the atmosphere. If we consider that the zenith image can be decomposed into a linear sum of images as a function of wavelength, i.e.,

$$I(x', 0) = \sum_\lambda I(x', 0, \lambda) \tag{6}$$

then the observed set of images are given by,

$$O(x, z) = \sum_\lambda I(x', \lambda) \otimes D(\lambda, z) \tag{7}$$

Solving for $I(x, \lambda)$ becomes a regression problem that can be solved for by minimizing

$$\chi^2 = \sum_x (O(x) - \sum_\lambda I(x', \lambda) \otimes D(\lambda))^2 \tag{8}$$

There are a number of possible approaches for finding the "zenith" image. The convolution can be written as a transfer matrix, T, where the elements of the matrix correspond to the fraction of pixel x' that maps to pixel x in the observed image. Under this mapping, we can write the linear equations as $TI = O$ and by inverting the matrix solve for I.

While, T, is clearly sparse the number of terms that must be solved for given the number of wavelengths λ that I is decomposed into, means that we require a heavily regularized regression. The initial implementation for the DCR corrected template will invert the linear equations assuming smoothness between adjacent pixels and as a function of wavelength, by adopting first and second order finite difference matrices. A prototype implementation has been demonstrated for the 1D case.

A second approach will be to forward model the problem by iteratively updating I based on a set of observations O.

For each approach the number of wavelength bins that $I(x)$ can be decomposed into will depend on the number of observations at different airmass. The assumption of a constant PSF will clearly not hold for the LSST observations but can be incorporated within the convolution Equation 7 or addressed through a separable, wavelength dependent, PSF convolution. The robustness of these techniques will need to be evaluated for low signal-to-noise sources and in the presence of scattered light and artifacts.

8.19 Image Decorrelation

8.19.1 Difference Image Decorrelation

In situations where the signal-to-noise in the template image is not insignificant (e.g., when the template is constructed by co-addition of a small number of exposures), the resulting image difference will contain autocorrelated noise arising from the convolution of the template with the PSF matching kernel prior to subtraction. This will result in inaccurate estimates of thresholds for `diaSource` detection if the (potentially spatially-varying) covariances in the image difference are not properly accounted for.

A viable alternative in the case of noisy template images is to construct a difference image with a flat noise spectrum, like the original input images [33, 34]. This simply involves multiplying the image difference by a term which removes its frequency dependence,

$$D(k) = [S(k) - \kappa(k)T(k)] \sqrt{\frac{\sigma_S^2 + \sigma_T^2}{\sigma_S^2 + \kappa^2(k)\sigma_T^2}}, \tag{9}$$

where S is the science image, T is the template, σ_S^2 and σ_T^2 are their respective variances, and κ is the PSF-matching kernel which, when convolved with the template, matches the PSF of the template to that of the science image. κ may be solved for (in real space) as described in Section 8.16. Then the multiplication by the square-root term in Equation 9 may be interpreted as applying a post-image-differencing convolution kernel which "removes" the pixel-wise correlation which was added by convolution of the template by the PSF-matching kernel. The PSF of the resulting decorrelated difference image ϕ_D then equals the PSF of the science image ϕ_S, convolved with the post-differencing kernel:

$$\phi_D(k) = \phi_S(k) \sqrt{\frac{\sigma_S^2 + \sigma_T^2}{\sigma_S^2 + \kappa^2(k)\sigma_T^2}}. \tag{10}$$

We are investigating this approach and have shown that, for idealized situations, the resulting image differences are statistically indistinguishable from those generated using the "Proper image subtraction" technique proposed by [34].

Issues arising from complications often seen in real-world data such as spatially-varying PSFs and/or poorly-evaluated matching kernels, spatially variable backgrounds and/or noise, and possibly non-Gaussian or heteroschedastic noise need to be further evaluated. Such tests are currently underway on simulated and real data. These tests could highlight the advantages of the method proposed here over the proposal of [34], including: no requirement for accurate measurement of the PSFs of the science or template images, and thus the ability to to account for errors in astrometric alignment and to directly model spatially varying differential PSFs.

8.19.2 Coadd Decorrelation

- Fourier-space/iterative deconvolution of likelihood coadds, as in DMTN-15.

- Need to test with small-scale research before committing to this approach.

8.20 Star/Galaxy Classification

8.20.1 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.

8.20.2 Multi-Source S/G

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

8.20.3 Object Classification

- Best classification derived from multifit and possibly variability.

8.21 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. [25], 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.

8.21.1 Characterization of Periodic Variability

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

$$y(t) = ct + \sum_{i=1}^{3} \sum_{j=1}^{4} y_i(t|jf_i) \tag{11}$$

$$y_i(t|jf_i) = a_{i,j} \sin(2\pi j f_i t) + b_{i,j} \cos(2\pi j f_i t) + b_{i,j,0} \tag{12}$$

  where i sums over fundamentals and j over harmonics.

- Use iterative application of the generalized Lomb-Scargle periodogram, as described in [25], 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.

8.21.2 Characterization of Aperiodic Variability

In addition to the periodic variability described above, we follow [25] 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.

$$\text{Kurtosis} = \frac{n(n+1)}{(n-1)(n-2)(n-3)} \sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{s} \right)^4 - \frac{3(n-1)^2}{(n-2)(n-3)} \tag{13}$$

$$s = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \overline{x})^2} \tag{14}$$

- Standard deviation.
- The fraction of magnitudes which lie more than one standard deviation from the weighted mean.

- Welch-Stetson variability index J [27], 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

$$P_k = \delta_i \delta_j \tag{15}$$

$$\delta_i = \sqrt{\frac{n}{n-1}}\frac{\nu_i - \overline{\nu}}{\sigma_\nu}, \tag{16}$$

where n is the number of observations of the object, and ν_i its flux in observation i. Following the procedure described in Stetson [27], the mean is not the simple weighted algebraic mean, but is rather reweighted to account for outliers.

- Welch-Stetson variability index K [27], 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 ratio of $F_{5,95}$ to the median.

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

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

8.22 Proper Motion and Parallax from DIASources

Every time we observe another apparition of a DIAObject, we have an opportunity to update/improve the proper motion and parallax models. The DIASources are associated with the current best model from the DIAObject. The proper motion and parallax are then refit using the new observation.

Do we actually want to do this:
I had a conversation about this with Colin. In reqlity we can't do as good a job with proper motion and parallax in nightly processing as we can in DRP. It's true that we would have no estimate of the proper motion or parallax until the first release if we do not calculate it in nightly, but I'd argue that before the first release we don't have the baseline to calculate an accurate anyway. Further, the measurement in DRP can be much better since we can do it as part of joint astrometric fitting. If we don't measure pm and parallax in nightly, we could still use the DRP measurement in the associated DRP object for association.

8.23 Association and Matching

Association between an external catalog of sources with objects detected from an LSST visit is critical to many aspects of the nightly and data release processing. External catalogs may come from photometric or astrometric standards (e.g. catalogs from GAIA), from previous LSST observations (e.g. Objects), or from catalogs derived from previous observations (e.g. the ephemerides of moving sources).

For cross-matching to reference catalogs the algorithm must be able to account for variation in scale and rotation of the field, and for optical distortions within the system. It must be fast and robust to errors and capable of matching across different photometric passbands.

For association with previous LSST observations the algorithms will need to be probabilistic in nature, must account for cases where the image quality results in the blending of sources, must work at high and low Galactic latitude, and must allow for sources with varying and variable SEDs.

Algorithmic components in this section will typically (but perhaps not always) delegate to the N-Way Matching software primitives, which provide spatial indexing and data structures for simple spatial matches.

8.23.1 Single CCD to Reference Catalog, Semi-Blind

Given a set of sources detected on a single sensor, and a corresponding reference catalog we adopt a simple pattern matching algorithm to cross-match between the catalogs. We assume that the sources detected on the sensor have approximate positions given by the telescope's initial estimate of a WCS, that we know the plate scale of the system, and that positional errors are available for both the sensor and reference catalog.

Cross-matching is undertaken using the Optimistic Pattern Matching (B) algorithm of Tabur [**?**]. The algorithm defines an order m (default 6) size $m-1$ acyclic connected tree as the pattern to match between catalogs. The details of the LSST implementation of the Optimistic Pattern Matching (B) algorithm are as follows:

- An optical distortion model is subtracted from the positions of the detected sources on the sensor to define a gnomonic plane projection.
- Detected sources are ordered in descending brightness and the brightest n stars (default = 50) are selected, I.
- Given the extent of the sensor (defined from the initial WCS) the brightest n sources are extracted from the reference catalog, R.
- The pairwise distances and position angles between the reference catalog sources, R, are measured (generating $n(n-1)/2$ pairs).
- For the m brightest sources in I the length of the edges and the position angles of the edges of the graph are calculated (with the brightest source the center of the graph). Angles are relative to the sensor orientation and distances are in angular coordinates.
- A binary search identifies all pairs in R with a length matching the length of the edge containing the brightest two sources (within a tolerance that has a default of 3 arcsec). The difference in position angle between the reference and source edges is assumed to be due to the rotation of the sensor relative to the reference catalog and this position angle difference is accounted for in all remaining matches.
- The remaining $m-2$ edges are compared to the pairs in R and a match is assumed if the edge length and postion angles are within the tolerances (defaults 3 arcsec and 1 degree respectively).

- If $m-1$ matches are not found the search repeats; initially for the remaining matches to the length of the edge corresponding to the brightest two sources and then for a new graph that excludes the brightest object from I and finds the next brightest m sources.

Once $m-1$ edges match, an initial verification step is performed where a gnomonic projection of the reference catalog is fit to the matched stars. Given a consistent fit to the gnomonic projection all sources and reference objects are matched in sensor coordinates (sorted by brightness) with the brightest source in I selected when two sources are within the match tolerance (default 3 arcsec).

For the case of no WCS for the sensor or a significant error in the WCS ($>$ 3 arcsec), a blind matching will be undertaken using the algorithms in astrometry.net.

8.23.2 Single Visit to Reference Catalog, Semi-Blind

For single visit cross-matching matches all sources within a focal plane will be matched to the reference catalog, R. This case is a modification of the original Tabur algorithm. It must account for significant distortions across the focal plane, and for deviations from a Euclidean space when due to the gnomonic projection.

Modifications from the single sensor cross matching are:

- Given a model for the postions and orientations of the sensors on the focal plane, sensor coordinates are transformed to focal plane coordinates
- The focal plane coordinates are corrected for the optical distortion model to provide a Euclidean space

8.23.3 Multiple Visits to Reference Catalog

AUTHOR: Jim

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

8.23.4 DIAObject Generation

Assuming that all DIAObject positions been propagated to the MJD of the visit (including proper motion and the generation of ephemerides for SSObjects) association of a DIASource with a DIAObject simplifies to the probabilistic assignment of a DIASource to a DIAObject.

We define this assignment in terms of the Bayes Factor, B, that defines the ratio of the probability that the observed data, D, is more likely given a model, H, that the DIASource and DIAObject are matched, than for a model K, where the sources do not match.

$$B(H,K|D) = \frac{p(D|H)}{p(D|K)} \tag{17}$$

see Budavari and Szalay [35].

Assuming a normal distribution for positional uncertainties the Bayes Factor is given by,

$$B(H,K|D) = \frac{sinh(w)}{w} \prod_{i=1}^{n} \frac{w_i}{sinh(w_i)} \tag{18}$$

with

$$w = |\sum_{i=1}^{n} w_i \bar{x_i}| \tag{19}$$

with x_i the 3D unit vector for a position on a sphere, and $w = 1/\sigma^2$ with σ the uncertainty on the position.

For the case of two sources and small uncertainties on the positions this simplifies to

$$w = \sqrt{(}w_1^2 + w_2^2 + 2w_1 w_2 \cos(\phi) \tag{20}$$

and

$$B = \frac{2}{\sigma_1^2 + \sigma_2^2} \exp(-\frac{\phi^2}{2(\sigma_1^2 + \sigma_2^2)}) \tag{21}$$

with ϕ the angle between the positions.

For all pairs of sources within a given tolerance the Bayes Factor will be calculated and the source with the largest Bayes Factor assigned to the DIAObject. For sources above the Bayes Factor threshold that were not assigned, the Bayes Factor and DIAObject ID will be persisted in the DIASource. Thresholds for the Bayes Factor will be derived from simulations.

An extension to Bayes Factor association that accounts for unknown proper motions is also possible [**?**]

8.23.5 Object Generation

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

8.23.6 Blended Overlap Resolution

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

8.24 Raw Measurement Calibration

- 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.

8.25 Ephemeris Calculation

Ephemeris calculation for the purpose of association in the nightly pipelines and for attribution and precovery in dayMOPS will require an indexing algorithm as well as a numerical integration phase. The JPL Horizons page reports $700,000$ asteroid orbits. This is far too many to run forward for every observation we will take. Thus, we will need to predict which bodies are likely to cross an aperture on the sky.

There are tools that allow for orbit prediction. As a baseline, we suggest using the OOrb (https://github.com/oorb/oorb). Regardless of the tool we use in production, it will need the following features:

- `Propagation`: Take a set of orbits and do the full numerical integration forward/backward in time to produce a new set of orbital elements
- `Prediction`: Produce a set of topocentric positions for a given set of objects at a particular time

In order to make spatial lookup of the orbits of interest fast, we will checkpoint the location of every Solar System object at the beginning, middle and end of each upcoming night. The checkpointing will involve saving topocentric positions for all Solar System objects and saving the propagated orbital parameters at the end of the night. We cannot precompute this for the duration of the survey because we will find new objects and we will update orbits of known objects. This computation will be done daily as part of the preparatory work for nightly observing. This is not a large computational challenge and is pleasingly parallel.

During nightly processing, ephemeris prediction will be carried out on the objects that may intersect the visit in question. For spatial filtering, all objects will be assumed to move linearly over half the night. The on-sky visit aperture with an appropriate buffer to account for the maximum acceleration of a Solar System object over about 4 hours will determine which objects potentially fall in the exposure. For those few thousand objects, precise ephemerides will be calculated for the purpose of association.

8.26 Make Tracklets

Tracklets are the building blocks of orbits. The process of linking observations is to pair up all observations that are within some distance of each other given a maximum on sky velocity. For any source, tracklets can be found by looking in circular apertures in subsequent visits with the radius of the circular aperture growing with time by $v_{max}dt$ for v_{max} in appropriate units. In practice we will follow [39] and build KD-trees on detections from each visit. KD-trees allow fast range searches. Linking up tracklets simply involves a series of range searches on available visits.

The number of tracklets goes up as $O(n^2)$ where n is the number of images covering a region in a given time span. However, many of the tracklets are degenerate (i.e. for an object moving slowly across the sky, it is possible that the beginning, ending and every other image in between could be within the velocity cut). These degenerate tracks are "collapsed" by computing a velocity vector for each tracklet. The tracklets are then binned in speed, perpendicular distance from a reference location, and direction. Similar to a Hough transform, degenerate tracklets will tend to accupy similar bins. Bins with multiple tracklets will be used to reduce the tracklets to the longest linear tracklet consistent with the tracklets.

When tracklets are collapsed, we gain more information about the collapsed tracklet since we have multiple observations of it. This allows some tracklets to be dismissed as spurious linkages. Any observation that deviates from the linear fit to the collapsed tracklet by a threshold amount will be discarded as spurious.

8.27 Attribution and Precovery

Precovery is the process of adding 'orphan' DIASources, those that do not belong to a SSObject or DIAObject, to a SSObject. Any time an SSObject's

orbital parameters change significantly, it's possible that DIASources not associated previously could now match. The process is to calculate ephemerides backward in time from the earliest observation as far as is possible given the uncertainty in the orbit. These ephemerides are compared to the orphan DIASources. If a match is found, a new orbit is fit and if the new orbit is a better fit than the old one, the SSObject is updated with the new fit.

Attribution is the process of adding tracklets to known SSObjects. For a given time window, topocentric ephemerides are calculated for all SSobjects that could potentially intersect any of the images in that window at the observation times of each of the images. These ephemerides are then compared to the tracklets in the time window. If any of them match in location and velocity, a new orbit is calculated. If the new orbit is better than the old one, the tracklet is tagged as being part of that SSObject and the SSOBject is updated with the new orbital parameters.

Since both attribution and precovery involve updating the SSObject, this process is recursive. The cadence of the recursion will be daily. Since we run attribution and precovery at least once during every run of the moving object pipeline, there is little need to recurse on shorter timescales.

8.28 Orbit Fitting

Given a database of tracklets not associated with any SSobject, we will look for tracks that match physical orbits.

Finding tracks is a tricky problem. We will follow the approach presented in [39]. All except the fastest moving bodies will have linear motion over a night; however, this is not true over the LSST discovery window of 30 days. In order to have high quality candidate tracks, we require three tracklets per track. Since there are limits to how fast Solar System objects can move and also how fast the can accelerate, we can build a KD-tree on the tracklets in a given observation in velocity and position. Given a node, this implies an acceleration for nodes in other trees. Since we require at least one support tracklet between any two endpoint tracklets, we can discard any nodes that do not have at least on matching node between them. With this in mind, we search for pairs of tracklets that match the velocity and acceleration cuts and are on different nights. If there is also at least one node between them in time (and on a different night than either of the endpoints) that also pass the velocity and acceleration criteria, all nodes are searched for tracks.

In order to validate candidate tracks, a quadratic fit to the orbit is

attempted with higher order topocentric effects due to reflex motion of the Earth included. These effects depend on the distance of the object from the Earth, so the range is fit for as part of the fitting process. For tracks with sufficiently good χ^2, the tracks are passed on to an orbit fitter. As above, there are tools to fit for orbital parameters given a set of observations. We will use these as our final orbit determination.

8.29 Orbit Merging

Bin all SSObjects of interest in orbital parameter space, by building a tree on the SSObject database. If there are any orbits that are sufficiently close in parameter space, they will be merged into a single orbit and the SSObject database updated.

9 Software Primitives

9.1 Cartesian Geometry

- Geometry in image, focal plane coordinate systems.
- Includes continuous (floating point) and discrete (integer) versions of some things; integer versions refer to entire pixels, which makes them somewhat different.
- May need augmented versions of some classes to allow them to know what coordinate system they're in.
- May need augmented versions of some classes to store uncertainty.
- All classes need to be persistable. Some need to be persistable to individual records (via e.g. FunctorKeys)
- All classes have counterpart Spherical classes related to them by WCS transforms.

9.1.1 Points

- Needs sensible handling of arithmetic operators. Currently implemented by making Extent a separate class, adding CoordinateExpr for element-wise comparisons – but those aren't the only options.
- Need continuous (PointD) and discrete (PointI) versions.
- 3-d continuous Point/Extent also useful, especially in representing unit vectors on the sphere. May not need to be the same template class (and maybe it shouldn't be, if it simplifies our code).
- Probably need to make these immutable (or have an immutable version) at least in Python so they can be exposed as properties.
- Needs to be persistable to individual records in the table library.
- Probably needs augmented version with uncertainty.
- Probably needs augmented version with coordinate system.

9.1.2 Arrays of Points

- Need containers for Points that work well in both C++ and Python – more than just a naively-wrapped `std::vector` would provide (in terms of NumPy interoperability, mostly). Probably something based on ndarray, translating to a NumPy array with x and y fields?
- Unclear if we need a container with dynamic size. Could probably use `std::vector` and Python `list` while building arrays, then freeze into a fixed, viewable array.
- Probably needs augmented version with coordinate system (all points in same coordinate system).
- Should look into what Astropy does here.

9.1.3 Boxes

- Need continuous (BoxD) and discrete (BoxI) versions, with different relationships between min, max, and dimensions.
- Probably need to make these immutable (or have an immutable version) at least in Python so they can be exposed as properties.
- Needs to be persistable to individual records in the table library.
- Spherical counterpart is actually Spherical Polygon.

9.1.4 Polygons

- Only continuous version needed.
- Mostly used to represent large-scale masks (regions around bright stars, vignetted regions).
- Needs to support rasterization to mask and/or footprint.
- Needs to support efficient topological operation and predicates with other Polygons, Points, and Boxes (probably not Ellipses).

9.1.5 Ellipses

- Only continuous version needed.
- Mostly used to represent source/object shapes.
- Needs to support many different ellipse parameterizations.
- Needs to support fast evaluation of elliptically-symmetric functions (via computing the generating affine transform)
- Need version that knows its position and one that doesn't.
- Needs to support rasterization to mask and/or footprint
- May need an immutable version in Python (not yet certain).
- May need an augmented version with uncertainty.

9.2 Spherical Geometry

The spherical geometry library is a dependency of the database as well as applications, it includes fundamental types that are logically present in database tables (as groups of fields), and some geometry classes are important for spatial indexing.

- Geometry on the sky
- All positions and distances are Angles; need type safety for angle unit.
- May need augmented versions of some classes to allow them to know what coordinate system they're in.
- May need augmented versions of some classes to store uncertainty.
- All classes need to be persistable. Some need to be persistable to individual records (via e.g. FunctorKeys)

9.2.1 Points

- Needs sensible handling of arithmetic operators. Point/Extent split probably an even better idea here.
- Probably need to make these immutable (or have an immutable version) at least in Python so they can be exposed as properties.
- Needs to be persistable to individual records in the table library.
- Probably needs augmented version with uncertainty.
- Probably needs augmented version with coordinate system.

9.2.2 Arrays of Points

Same requirements as Cartesian Arrays of Points.

9.2.3 Boxes

- Not obvious we need this at all.
- Defined on long/lat grid, so not a box in any Cartesian projection.
- Needs special handling for poles?

9.2.4 Polygons

- Connecting points with great circles is probably sufficient, even if this only approximately maps to Cartesian polygons in most projections; we will have very few Cartesian polygons that extend beyond the size of one CCD, and for those great circles should be fine.
- Needs to support efficient topological operation and predicates with other Polygons, Points, and Boxes (probably not Ellipses).
- May need to support rasterization to some spherical pixelization scheme (e.g. HTM), but those requirements are probably driven more by database.

9.2.5 Ellipses

- Doesn't need to be a true spherical geometry - we really just need a Cartesian ellipse with angular position and size, defined via a gnomonic plane projection centered on the ellipse. All spherical ellipses will be small enough that we don't have to worry about the topology of large ellipses.
- Probably needs augmented version with uncertainty.

9.3 Images

9.3.1 Simple Images

- Conceptually just a numpy array + xy0
- Still need to fix xy0 behavior on iterators/locators
- Constness is a mess
- Need more Pythonic interface to templates.
- Needs FITS import/export in addition to some round-trip internal representation. May need FITS roundtrip.

9.3.2 Masks

- Should not rely entirely on bits in integer images; consider extending to include:
  - a container of Footprints (actually PixelRegions).
  - a container of Polygons or other geometries.
- May want to switch from compile-time number of bits (`Array<uintN,2>`) to dynamic (`Array<uint8,3>`).
- Can we do anything to fix confusing semi-singleton mask plane dict behavior, while getting the functionality we want?
- Also all requirements of simple images.

9.3.3 MaskedImages

Includes components:

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.

Uncertainty 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 covariance.

Other notes:

- Want to support constant mask and uncertainty, probably via single-pixel images with zero strides.
- Want NumPy-like view of all three planes. Probably a new object that implements array interface without inheriting from numpy.ndarray.
- Also all requirements of simple images.

9.3.4 Exposure

Includes components:

MaskedImage Image, mask, uncertainty.

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 and position. Some PhotoCalibs may represent global calibration and some may represent relative calibration.

CameraGeom Object describing the detector this image corresponds to, if applicable. Could go on a subclass of Exposure for sensor-level images.

CoaddInputs Table(s) describing the inputs that went into this coadd. Could go on a subclass of Exposure for sensor-level images.

VisitInfo Additional metadata about visit (including pointing and and time information).

Other notes:

- Probably missing some components in the above list.
- Want to forward more MaskedImage operations to Exposure (so we don't have to say getMaskedImage() all the time).
- Need to be able to persist and pass around non-image components separately.
- Need to integrate ValidPolygon component in current design with Mask.
- Needs FITS import/export in addition to some round-trip internal representation. May need FITS roundtrip.

9.4 Multi-Type Associative Containers

- Replacement(s) for PropertyList/PropertySet.
- Needs to be more Pythonic; more like dict or OrderedDict.
- Need a variant that can be used to round-trip FITS headers.

9.5 Tables

All classes need round-trip internal persistence and FITS, ASCII, SQL import/export.

9.5.1 Source

- In-memory data structure for Source, DIASource, ForcedSource tables.
- Can have (Heavy)Footprint attached.
- Always has ID, coord (at least conceptually; may be computed on-the-fly).
- Has slots.

9.5.2 Object

- In-memory data structure for Object, DIAObject.
- Must be able to represent information from multiple bands and coadd flavors (array fields? nested rows of another type?)
- Needs to have multiple (Heavy)Footprints attached.
- Needs to have join to table of Monte Carlo samples.
- Maybe just want to be able to attach arbitrary objects?
- Has slots.

9.5.3 Exposure

- Want to be able to store all non-image Exposure components in a single record.

9.5.4 AmpInfo

- Used to record electronic parameters for amplifiers in Camera Descriptions.

9.5.5 Reference

- Need table class for (external) reference catalogs.
- Has a lot in common with Source and Object, but needs fewer attachments, and typically is in calibrated units instead of raw units.

9.5.6 Joins

- Need an in-memory representation of relationships (one-many, many-many, maybe one-one) between tables.
- Need pointer-like behavior (e.g. for one-many, a Record looks like it has another Catalog as one of it fields)
- Used to represent outputs of N-Way Matching.
- Used to store samples with Object tables.
- Used to related ForcedSource to Object and DIASource to DIAObject.

9.5.7 Queries

- Need basic SQL-WHERE-like query support, at least in Python.
- A concrete use case is in for use as source selectors for e.g. PSF candidates.
- Could maybe delegate this to Pandas and/or Astropy, use NumPy expressions.
- May need to support string expressions (supplied as configuration parameters, for instance).
- Actually being able to write SQL could be very nice. In-memory sqlite back-end? Some other third-party SQL parser, with our own (numpy-compatible) storage backend?

9.5.8 N-Way Matching

- 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 identification 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 to derive 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.

9.6 Footprints

All classes need to be persistable (usually as components of larger data structures such as tables or masks).

- Footprint itself includes both Spans and Peaks, representing a detection.
- Footprints are guaranteed to be contiguous.
- Concept is fine, class itself needs a lot of cleanup.

9.6.1 PixelRegions

- Very lightweight data structure that is just a container of Spans - represents just a pixel region.
- Needs large suite of topological operations.
- May be noncontiguous.

9.6.2 Functors

- Run functions on each pixel in a PixelRegion
- Needs to support unary, binary, maybe ternary?
- Needs to support modifing arguments in-place and returning them.
- Abstracts whether pixels are from a 2-d image or a flattened 1-d array.

9.6.3 Peaks

- Needs to record position, rough flux.
- Needs to be extensible to also hold at least flags.
- Needs very low overhead; will have many, many peaks.
- Current implementation uses custom table class, but is a bit clunky.

9.6.4 FootprintSets

- Specialized container for Footprints.
- Because Footprints are guaranteed contiguous, most topological operations are here instead (as they have the potential to merge or split Footprints).
- Needs better interoperability with table library, which is also a kind of container of Footprints.

9.6.5 HeavyFootprints

- A Footprint with its own pixels, stored as a flattened 1-d array.
- May sometimes need mask and uncertainty as well, may not.
- Definitely need a version that doesn't have mask and uncertainty.

9.6.6 Thresholding

- Low-level operations for finding above-threshold regions and peaks within them (on MaskedImages as well as Images).
- Should decompose into operations that just find above-threshold regions (as PixelRegions), operations that just find Peaks within PixelRegions, and a higher-level operation to do both, returning a FootprintSet.

9.7 Basic Statistics

- Various robust statistics for central tendency and distribution widths, measured on 2-d and 1-d arrays.
- Needs to be able to make use of mask and uncertainty arrays.
- Needs to work on 2-D Images and MaskedImages
- Needs to work on stacks of aligned pixels for coaddition.

9.8 Chromaticity Utilities

All classes need to be persistable (usually as components of larger data structures such as tables or camera descriptions).

9.8.1 Filters

- One or more classes that represent the complete wavelength-dependent throughput of the system and all of the multiplicative components that comprise it (actual filter curves, sensor QE, etc.).
- Needs to be able to handle position-dependence as well, including coordinate transformations of position dependency (from e.g. filter coordinate system to focal plane to individual sensors).
- Need concrete classes that are mostly fixed with a parameters to represent highly-variable aspects (e.g. atmospheric absorption)

- Probably need another class to represent a telescope or survey's set of filters.

9.8.2 SEDs

- One or more classes that represent object spectra.
- Needs interoperability with filter classes (integrate to yield fluxes, ...?)
- Defines canonical approach to inferring SED from colors (which requires a library of canonical SEDs)
- Used to evaluate PSF and PhotoCalibs.

9.8.3 Color Terms

- Low-order approximations to mapping between different filter systems.
- Unclear (to jbosch) whether we'll use these at all in LSST production pipelines, but definitely needed for work with precursor data.

9.9 PhotoCalib

Needs to be persistable (usually as components of larger data structures such as Exposure).

- Spatially- and wavelength-dependent photometric calibration.
- May be relative or absolute.
- Needs to represent rescalings somehow (change from flats-for-backgrounds to monochromatic object flats).
- Needs to hold its own uncertainty (may not be just one number).
- May ultimately be a hierarchy of classes, intead of just one.
- Probably needs to hold a Filter. This is mostly just convenience; if it doesn't have one it needs to be passed one to be used.

9.10 Convolution Kernels

Probably needs to be persistable, but only to ease persistence of higher-level objects that may be built on top of them.

- Supports spatially-varying convolution with a variety of tricks for special kernels (e.g. spatially varying linear combinations of fixed kernels, kernels separable in x and y).
- Must support correlation as well.
- Closely related to PSFs, but kernels are not wavelength-dependent, and PSFs are. Not clear whether difference imaging kernels should actually be Kernels (they could be more like PSFs if they're wavelength-dependent).
- Includes support for image warping with both Lanczos and PSF-like kernels.
- Need to be able to compose Kernels.
- Needs to support approximation on different spatial scales (smoothly varying kernels need not be fully evaluated at every pixel).

9.11 Coordinate Transformations

- Need general system for 2-d coordinate systems and transformations, including both spherical and Cartesian systems.
- Transforms must be composable; conceptually we have a graph with coordinate systems as nodes and transforms as edges.
- Needs close integration with geometry libraries.
- Needs very lightweight implementations of affine/linear transforms.
- Needs interoperability with Image xy0 concept.
- Needs serialization to both internal (round-trippable) formats and import/export to standard external formats. Ideally the internal format would also be at least somewhat external (i.e. shared with Astropy).

- Needs to be able to at least export to standard FITS WCS (with some approximation).
- Coordinate tranforms are not wavelength-dependent.
- See also DMTN-10.

9.12 Numerical Integration

- Standard basic numerical integration based on Gaussian quadrature: can probably just wrap an external library.
- Unclear whether we also need any differential equation integration.
- May need specialized routines for computing multivariate Gaussian and/or Student's t CDFs for Monte Carlo sampling.

9.13 Random Number Generation

- Just need standard distributions and generators provided by most external RNG libraries.
- Need to design carefully with parallelization primitives to ensure deterministic results when running in parallel.

9.14 Interpolation and Approximation of 2-D Fields

- Unified interface to spline, polynomial, inverse-distance approaches to representing 2-d fields.
- Used for at least backgrounds and aperture corrections, maybe PSF modeling, WCS, other things.

9.15 Common Functions and Source Profiles

- Library of 2-d functions used for PSFs and galaxy profiles. Sersics, Gaussians, Moffats, etc.
- Maybe delegate to GalSim (would probably require contributing required features to GalSim)?

9.16 Camera Descriptions

- What we call CameraGeom, but it's more than geometry.
- Geometry is built on top of Coordinate Transformations library.
- Electronic descriptions built on top of AmpInfo Tables.
- Throughput descriptions built on top of Chromaticity Utilities

9.17 Numerical Optimization

- Linear least-squares fitting with and without constraints, with and without Bayesian priors.
- At least some nonlinear fitting with and without Bayesian priors (extension of Levenberg-Marquardt probably). May need to handle some limited constraints as well.
- Could invest a lot of effort early in this and do it well; this would retire risk elsewhere. Or we can do this as-needed and probably spend less effort overall, but may find ourself blocked at inconvenient times by lack of hard-to-implement features.

9.18 Monte Carlo Sampling

- Need modern MCMC sampler. Could probably use external code, but it's not entirely clear we can afford to do this in Python.
- Need adaptive importance sampling from mixture distributions and MCMC chains.

9.19 Point-Spread Functions

- Includes aperture corrections.
- Includes characterization of extended wings of PSF.
- Wavelength-dependent.
- Must support coaddition of PSF models.

- May need know its uncertainty, and be able to sample PSF realizations form this.

9.20 Warping

We need functions to warp regularly gridded data to a new grid in an arbitrary coordinate system with flux conservation.

9.21 Fourier Transforms

We will need to calculate discrete Fourier transforms on images in both directions.

9.22 Tree Structures

Maybe just having a KD-tree we can use in C++ and Python will be enough, however we likely will ndeed a C++ accessible version since there are already at least two C++ implementations currently in the stack.

9.23 Tools

KSK: When going through the algorithmic components, I noticed many tools we will need. By tools, I really just mean a well known algorithm that we can apply as a black box to data for a particular purpose. It's possible this should go someplace else. I'm open to suggestions.

- Periodogram – This will likely also require some sort of data type to hold the periodogram
- Hough transform and Canny algorithm – We may need one or both of these. We may also need variants on the standard implementation.
- General linear algebra framework including on sparse matrices.
- Data discovery – Simply ask questions like: "What data are at this location in this repository" **this is likely a middleware requirement**
- Reference catalog on disk representation for fast localization **this is also most likely a middleware requirement**

- orbit propagation
- orbit prediction
- orbit fitting

10 Glossary

ADU Analogue Digital Unit, also commonly called DN, the unit of measure of an analogue-to-digital-converter.

AP Alert Production

API Applications Programming Interface

CBP Collimated Beam Projector

CCOB Camera Calibration Optical Bench

CPP Calibration Products Pipeline

CR Cosmic Ray, strictly speaking, a cosmic ray *muon*, but in the context of ISR, used to collectively refer to any tracks caused by subatomic particles or nuclear interactions.

CTE Charge Transfer Efficiency

CTI Charge Transfer Inefficiency; 1 − CTE

DAC Data Access Center

DAQ Data Acquisition

DMS Data Management System

DR Data Release.

DRP Data Release Production

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

ISR Instrument Signature Removal

mask An integer bitmask used to convey information about a particular pixel, footprint, region, etc.

map A spatially varying scalar value representing a varying quantity e.g. coverage.

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

PTC Photon Transfer Curve, method for measuring the gain of a CCD from the variance in a flat-field image.

QE Quantum Efficiency

REB Readout Electronics Board, the unit of electronics used to readout three CCDs, *i.e.* one third of a raft.

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.

T&S Telescope and Site team

TBD To Be Determined

TCS Telescope Control System

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.

WBS Work Breakdown Structure

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] P. O'Connor, **Crosstalk in multi-output CCDs for LSST**, JInst 10 05 C05010 (2015).

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

[24] 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.

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

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

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

[28] 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).

[29] 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).

[30] C. Alard and R. H. Lupton, **A Method for Optimal Image Subtraction**, ApJ 503, 1996 325–331 (1998).

[31] C. Alard, **Image subtraction using a space-varying kernel**, A&A 144, 2 363–370 (2000).

[32] , H. Israel, H., F. V. Hessman, and S. Schuh, **Optimising optimal image subtraction**, Astronomische Nachrichten 328, 16–24.

[33] N. Kaiser, **Addition of Images with Varying Seeing**, Pan-STARRS Document Control, PSDC-002-011-xx (2004).

[34] B. Zackay, E. O. Ofek, and A. Gal-Yam, **Proper image subtraction – optimal transient detection, photometry and hypothesis testing**, ApJ, *Submitted* (2016).

[35] Budavári, T. and Szalay, A. S., **Probabilistic Cross-Identification of Astronomical Sources**, ApJ, 679, 301-309 (2008)

[36] Canny, J., **A Computational Approach to Edge Detection**, IEEE Trans. Pattern Anal. Mach. Intell., 8, 6, 679–698 (1986)

[37] Borncamp, David & Lim, P., L., **Satellite Detection in Advanced Camera for Surveys/Wide Field Channel Images**, STSCI Instrument Science Report, ISR 16-01, (2016)

[38] Galamhos, C., Matas, J., & Kittler, J., **Progressive probabilistic Hough transform for line detection**, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1999)

[39] Kubica, Jeremy, Masiero, Joseph, Moore, Andrew, Jedicke, Robert, & Connolly, Andrew, **Variable KD-tree Algorithms for Spatial Pattern Search and Discovery**, Proceedings of the 18th International Conference on Neural Information Processing Systems, NIPS'05, 691–698 (2005)