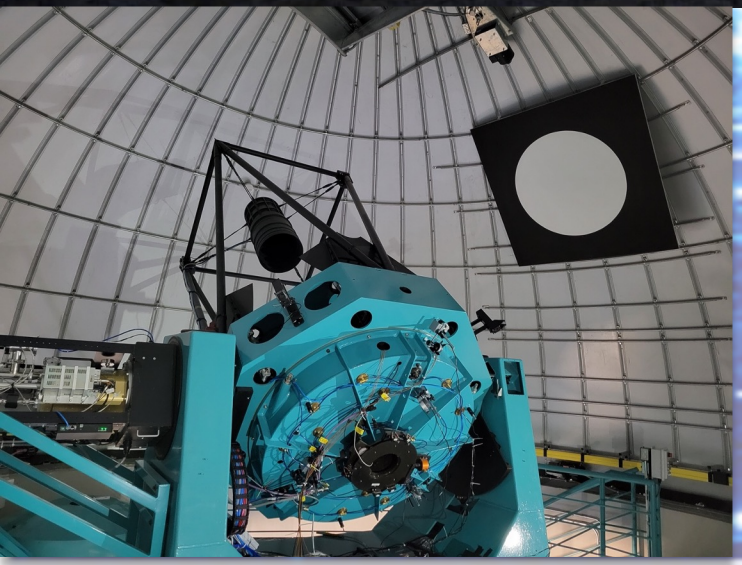
My Experience with DM stack in AuxTel Context



Sylvie Dagoret-Campagne*, Jérémy Neveu, Marc Moniez,

Laurent Le Guillou, Martin Rodriguez Monroy

Dominique Boutigny (computing with DM-stack@CC)

IJCLab & LPNHE @ IN2P3-CNRS

Part of
DM pipeline
for Auxtel analysis

Part of
DM pipeline
Used by
DP0 delegates

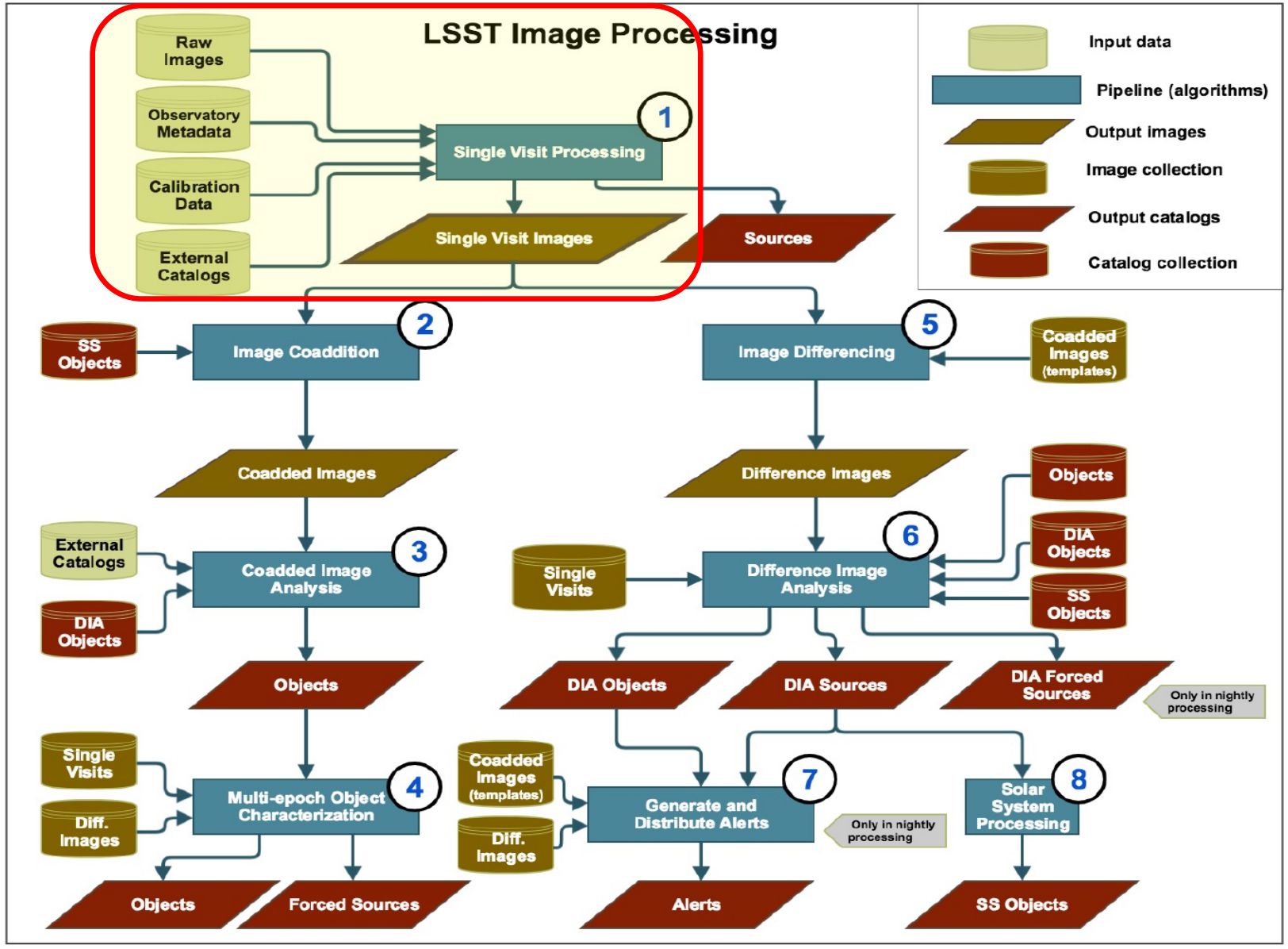
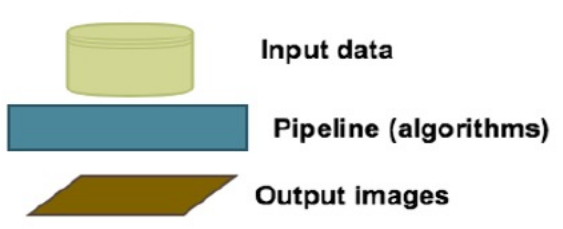


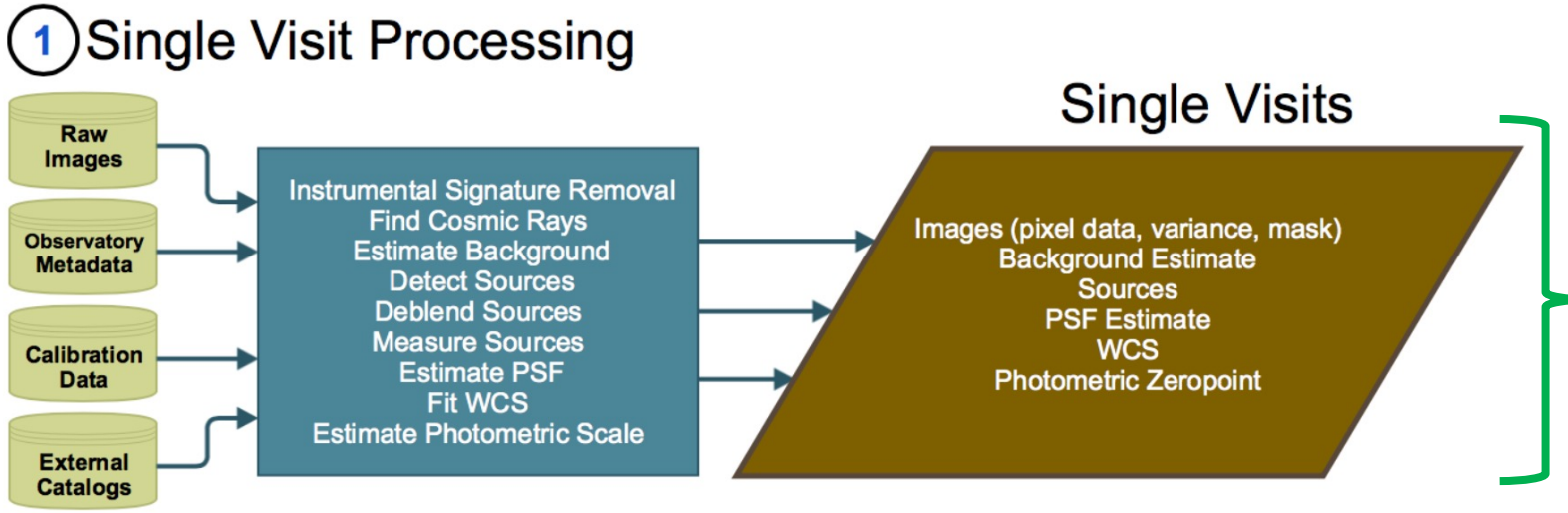
FIGURE 2: Illustration of the conceptual design of LSST science pipelines for imaging processing.

Zoom on first stage pipeline

How to interact with DM stack ?



Configuration of the Butler



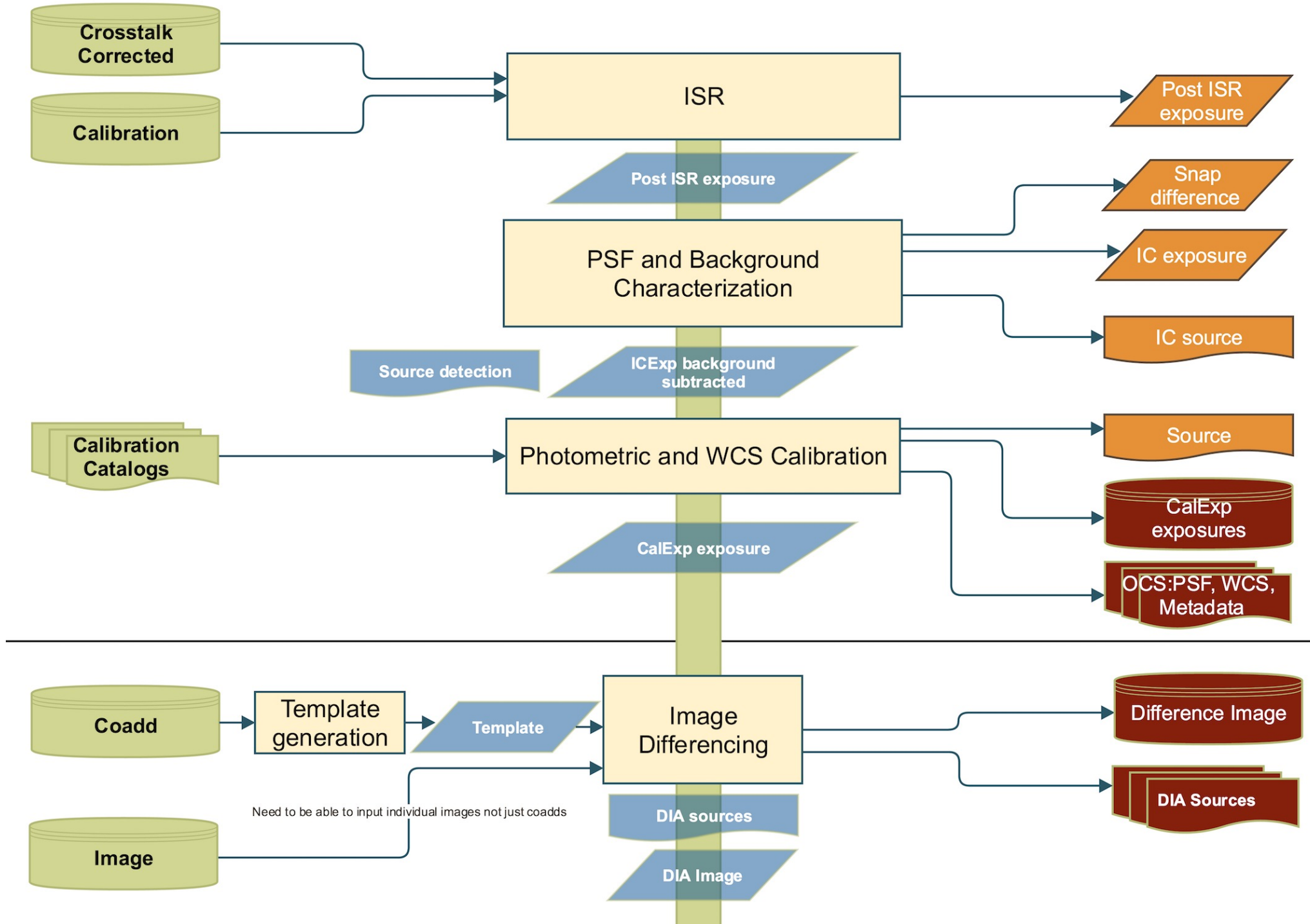
Interact with Products by Using the Butler

FIGURE 3: Illustration of the conceptual algorithm design for Single Visit Processing pipeline.

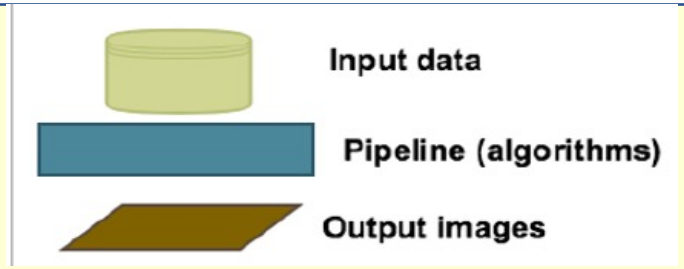
Many DM stack Tasks

- For Auxtel a processStarTask has been implemented to encapsulate a DM version of Spectractor

The Butler is a framework for generic I/O and data management.
It isolates application code from the underlying data-access implementation in terms of storage formats, physical locations, data staging, database mapping, etc.


Crosstalk Corrected
Calibration
ISR
Post ISR exposure
Post ISR exposure
Snap difference
PSF and Background Characterization
IC exposure
IC source
Source detection
ICExp background subtracted
Calibration Catalogs
Photometric and WCS Calibration
Source
CalExp exposures
CalExp exposure
OCS:PSF, WCS, Metadata
Coadd
Template generation
Template
Image Differencing
Difference Image
Need to be able to input individual images not just coadds
DIA sources
DIA Sources
Image
DIA Image

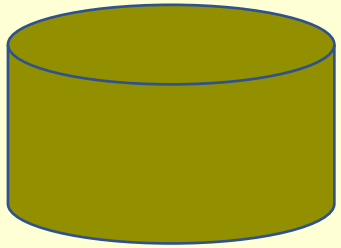
Reconstruction of Spectra with the whole DM pipeline



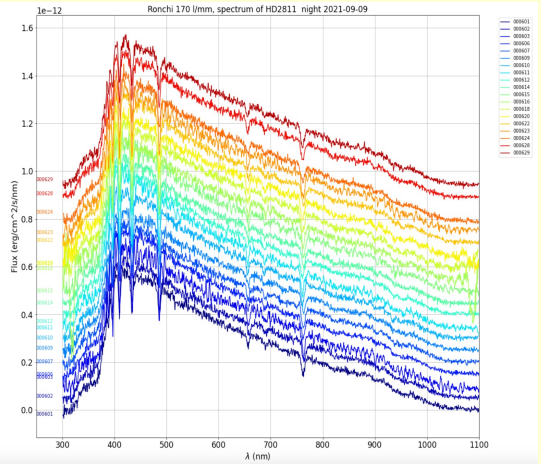
Raw images

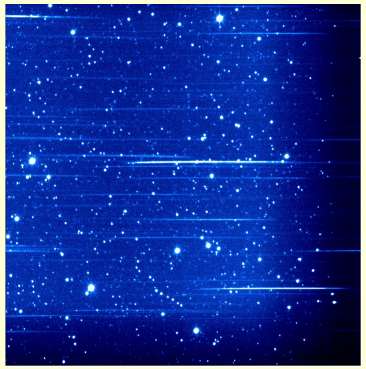
processStar
(pack atmospec)

Spectra,
Spectrogram

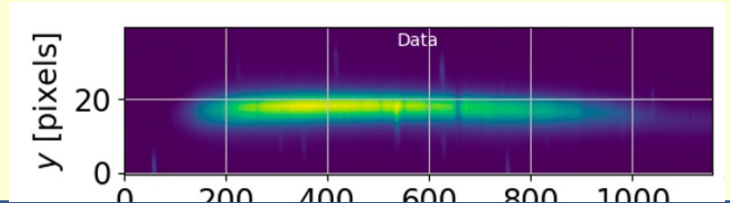


(no image rediction
No segment assembly)



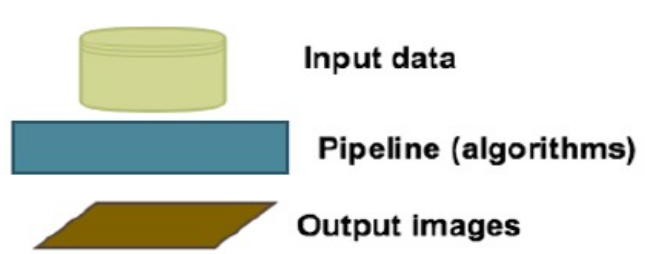


Spectractor DM version



Butler world

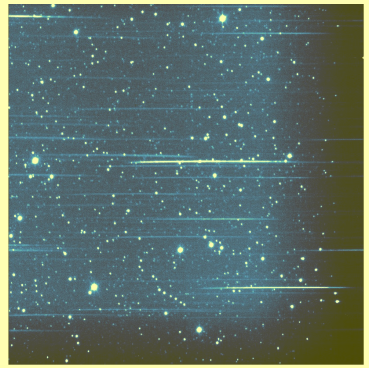
Effective Reconstruction of Spectra with DM pipeline by user



Raw images

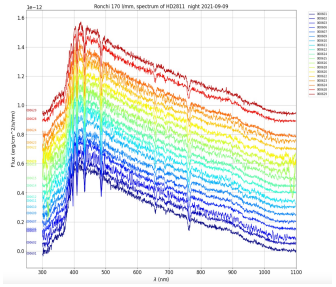
processStar
ISR

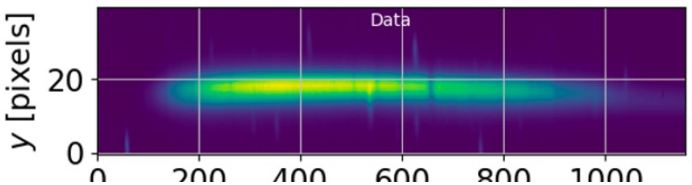
postISRCCD
image



Butler world

(no image reduction
No segment assembly)

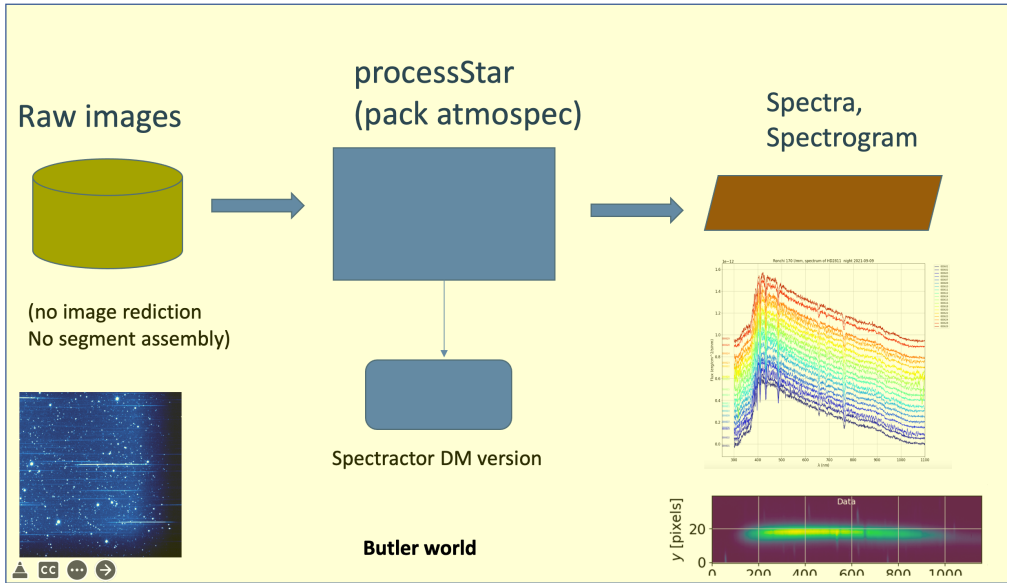




Spectractor
DESC standalone version

Why this architecture is not convenient now for commissioning

Pure DM recommendations



Pros:

- the easiest

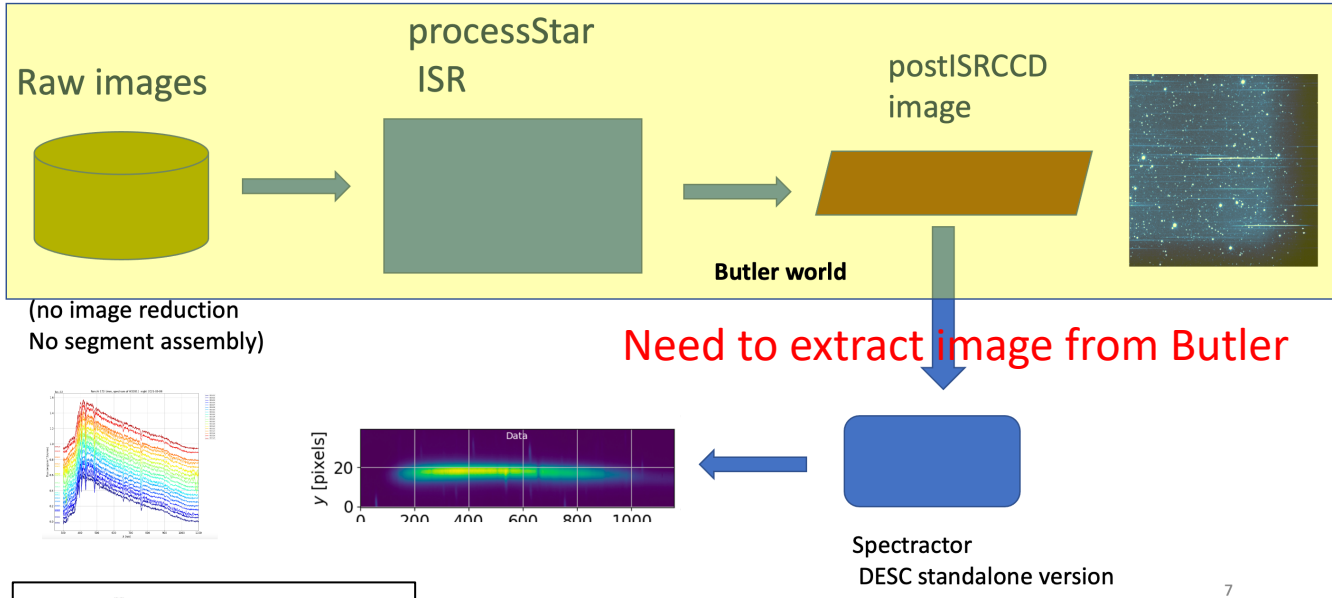
when every things works

- Don't care where information is stored
- Single DB

Cons:

- Difficult to debug,
- It is a DM « property »
- Or need to work closely with DM (but very busy)
- Hard to inject our flat

Hybrid mode non Rubin user path



Pros:

- Flexibility to work inside standalone version
- Easy to use our flat

Cons:

- Break the uniformity of the single Butler world
- Need to handle storage of external products
- Data product not useful for DM calibration (only for commissioning)

Part 1 : Auxtel at NCSA

Auxtel data at NCSA
https://lsst-lsp-stable.ncsa.illinois.edu/nb/user/dagoret/lab

Highlight : Auxtel Observations nights with a disperser

dagoret@nb-dagoret ~]$ ls /lsstdata/offline/teststand/auxTel/L1Archiver/storage

2019-03-05	2019-04-12	2019-08-20	2020-01-27	2021-01-21	2021-03-11	2021-07-13	2021-10-04	2022-02-16	20220421	
2019-03-06	2019-05-30	2019-08-21	2020-01-28	2021-01-26	2021-03-18	2021-07-16	2021-10-05	2022-02-17	20220422	
2019-03-07	2019-05-31	2019-08-27	2020-01-29	2021-01-27	2021-03-22	2021-07-20	2021-10-06	2022-02-23	20220425	
2019-03-08	2019-06-03	2019-09-04	2020-01-30	2021-02-01	2021-03-23	2021-07-27	2021-10-07	2022-02-24	20220426	
2019-03-09	2019-06-04	2019-09-15	2020-02-12	2021-02-02	2021-04-08	2021-07-28	2021-10-08	2022-03-04	20220427	
2019-03-12	2019-06-05	2019-10-17	2020-02-17	2021-02-04	2021-04-14	2021-08-04	2021-10-13	2022-03-08	20220428	
2019-03-18	2019-06-06	2019-10-24	2020-02-18	2021-02-09	2021-05-21	2021-08-05	2021-10-14	2022-03-11	20220429	
2019-03-19	2019-06-07	2019-10-31	2020-02-19	2021-02-10	2021-05-24	2021-08-16	2021-11-02	2022-03-14	20220502	
2019-03-20	2019-06-10	2019-11-04	2020-02-20	2021-02-11	2021-05-25	2021-08-17	2021-11-03	2022-03-15	20220503	
2019-03-21	2019-06-11	2019-11-06	2020-02-21	2021-02-12	2021-06-07	2021-08-18	2021-11-04	2022-03-16	20220504	
2019-03-22	2019-06-12	2019-11-08	2020-02-22	2021-02-15	2021-06-08	2021-08-20	2021-11-30	2022-03-17	20220505	
2019-03-26	2019-06-13	2019-11-13	2020-03-12	2021-02-16	2021-06-09	2021-09-02	2021-12-01	2022-03-18	20220511	
2019-03-29	2019-07-12	2019-11-14	2020-03-13	2021-02-17	2021-06-10	2021-09-03	2021-12-02	2022-03-24		
2019-04-02	2019-08-08	2019-11-18	2020-03-14	2021-02-18	2021-06-25	2021-09-07	2022-02-03	2022-03-25		
2019-04-03	2019-08-09	2019-11-19	2020-03-15	2021-03-03	2021-06-28	2021-09-08	2022-02-04	2022-03-29		
2019-04-04	2019-08-10	2019-11-20	2020-03-16	2021-03-04	2021-07-02	2021-09-09	2022-02-07	2022-04-04		
2019-04-05	2019-08-12	2020-01-17	2020-10-30	2021-03-05	2021-07-06	2021-09-13	2022-02-08	2022-04-05		
2019-04-08	2019-08-14	2020-01-21	2021-01-15	2021-03-08	2021-07-07	2021-09-22	2022-02-09	2022-04-06		
2019-04-10	2019-08-17	2020-01-22	2021-01-19	2021-03-09	2021-07-08	2021-09-30	2022-02-11	2022-04-07		
2019-04-11	2019-08-19	2020-01-23	2021-01-20	2021-03-10	2021-07-12	2021-10-01	2022-02-15	2022-04-14		

Additional packages to DM must be installed

In /home/dagoret/repos:

Additional DM compatible packages

[dagoret@nb-dagoret repos]$ ls

w_2021_02 w_2021_10 w_2021_21
w_2021_36 w_2022_09

Install additional package under repos:

In /home/dagoret/repos/w_2022_09

[dagoret@nb-dagoret w_2022_09]$ ls

- **atmospec** : the real interface to DM
- **rapid_analysis** : a kind of bookkeeping
- **Spectractor** : DM version of Spectractor standalone DESC version

In /home/dagoret/notebooks

[dagoret@nb-dagoret notebooks]$ ls -a

. .. notebook-demo system-test
.user_setups .user_setups_old
.user_setups_spring2021
.user_setups_w_2021_02
.user_setups_w_2021_10

A way to specify the Jupyter kernel

[6]:
```
from lsst.log.utils import enable_notebook_logging
enable_notebook_logging()
```

[7]:
```
from lsst.atmospec import ProcessStarTask

config = ProcessStarTask.ConfigClass()
config.doDisplayPlots = True  # show the plots in the notebook
config.spectractorDebugMode = True  # make all the debug plots along the way
config.binning = 4

# pretty minimal ISR because some things we don't have the calib products available for
# most of this would be picked up automatically if running from the command line from config files
config.isr.doLinearize = False
config.isr.doDark = False
config.isr.doFlat = False
config.isr.doFringe = False
config.isr.doDefect = True
config.isr.doCrosstalk = False
config.isr.doSaturationInterpolation = False

task = ProcessStarTask(config=config)
```

[8]:
```
from lsst.rapid.analysis.utils import runNotebook, checkStackSetup
```

```
/home/dagoret/repos/w_2022_09/rapid_analysis/python/lsst/rapid/analysis/visitCheck.py:132: FutureWarni
d has been replaced by lsst.utils.timer.timeMethod. Will be removed after v25.) -- Deprecated since ve
  def runDataRef(self, dataRef):
```

[9]:
```
checkStackSetup()
```

```
You are running w_2022_09 of lsst_distrib

Locally setup packages:
----------------------
atmospec setup at /home/dagoret/repos/w_2022_09/atmospec
rapid_analysis setup at /home/dagoret/repos/w_2022_09/rapid_analysis

Pick the image you want to run on, and the output collection you'd like the result put in
```

Official User notebook@ NCSA

Configuration of The DM pipeline

Constraint to run pipeline inside a notebook for non-DM-rubin member !

- Only change settings !

[10]:
```
dataId = {'day_obs': 20200315, 'seq_num': 192}
#outputCollection = 'u/mfl/notebook_test'
outputCollection = 'u/dagoret/notebook_test'
```

Make sure the plots appear in the notebook!

[11]:
```
%matplotlib inline
```

And that's it - let's run!

[12]:
```
result = runNotebook(dataId, outputCollection)
```

```
conda.common.io INFO: overtaking stderr and stdout
conda.common.io INFO: stderr and stdout yielding back
lsst.obs.lsst.translators.latiss WARNING: /lsstdata/offline/instrument/LATISS/storage/2020-03-15/AT_O_202003
an exposure time. Setting dark time to the exposure time.
lsst.obs.lsst.assembly WARNING: /lsstdata/offline/instrument/LATISS/storage/2020-03-15/AT_O_20200315_000192-
((minimum=(0, 0), maximum=(543, 2047)) v. (minimum=(0, 0), maximum=(575, 2047)))
lsst.isr INFO: Converting exposure to floating point values.
lsst.isr INFO: Assembling CCD from amplifiers.
lsst.isr INFO: Applying bias correction.
lsst.isr INFO: Masking defects.
lsst.isr INFO: Masking non-finite (NAN, inf) value pixels.
lsst.isr INFO: Widening saturation trails.
lsst.isr INFO: Set 6494 BAD pixels to 24.672998.
lsst.isr INFO: Interpolating masked pixels.
lsst.isr WARNING: No rough magnitude zero point defined for filter empty~ronchi90lpmm.
lsst.isr INFO: Setting rough magnitude zero point for filter empty~ronchi90lpmm: 30.940228
lsst.ctrl.mpexec.singleQuantumExecutor INFO: Execution of task 'isr' on quantum {instrument: 'LATISS', detec
lsst.characterizeImage INFO: CharacterizeImageTask initialized with 'simple' PSF.
/opt/lsst/software/stack/stack/miniconda3-py38_4.9.2-2.0.0/Linux64/meas_algorithms/gbe01a4569f+c1b4442b1a/pyt
position argument overload is deprecated and will be removed in version 24.0.  Please explicitly specify a po
  sigma = psf.computeShape().getDeterminantRadius()
/opt/lsst/software/stack/stack/miniconda3-py38_4.9.2-2.0.0/Linux64/meas_algorithms/gbe01a4569f+c1b4442b1a/pyt
position argument overload is deprecated and will be removed in version 24.0.  Please explicitly specify a po
  sigma = psf.computeShape().getDeterminantRadius()
lsst.characterizeImage.detection INFO: Detected 732 positive peaks in 145 footprints and 1025 negative peaks
lsst.characterizeImage.detection INFO: Resubtracting the background after object detection
lsst.characterizeImage.measurement INFO: Measuring 145 sources (145 parents, 0 children)
lsst.characterizeImage INFO: iter 1; PSF sigma=1.50, dimensions=(11, 11); median background=24.83
```

Specify the data to process:
- Date
- Image sequence number

Run the whole DM processing:
- Input : dataId
- Output : the collection

Constraints
- Run in notebook : Very slow
- Difficult to debug
- Buguy

- But cannot wait for a long time each image being processed by the notebook
- Run though shell or batch is not Allowed for standard users

ProcessStarTask In atmospec/python/lsst/atmospec/processStarTask.py

```
from .spectraction import SpectractorShim
class ProcessStarTaskConnections(pipeBase.PipelineTaskConnections,
                                 dimensions=("instrument", "visit", "detector")):
```

- Defines the inputs & outputs of the task

```
class ProcessStarTaskConfig(pipeBase.PipelineTaskConfig,
                            pipelineConnections=ProcessStarTaskConnections):
    """Configuration parameters for ProcessStarTask."" »
```

- Set default configuration parameters of the task

```
class ProcessStarTask(pipeBase.PipelineTask):
    """Task for the spectral extraction of single-star dispersed images.

    For a full description of how this tasks works, see the run() method.
    """
```

def runGen2(self, exp, spectractorOutputRoot, expId, sourceCentroid):

"""Calculate the wavelength calibrated 1D spectrum from a postISRCCD.
An outline of the steps in the processing is as follows:
* Source extraction - find the objects in image
* Process sources to find the x,y of the main star
* Given the centroid, the dispersion direction, and the order(s),
calculate the spectrum's bounding box
* (Rotate the image such that the dispersion direction is vertical
TODO: DM-18138)
* Create an initial dispersion relation object from the geometry
or alternative bootstrapping method
* Apply an initial flatfielding - TODO: DM-18141
* Find and interpolate over cosmics if necessary - TODO: DM-18140
* Perform an initial spectral extraction, depending on selected method
* Fit a background model and subtract
* Perform row-wise fits for extraction
* TODO: DM-18136 for doing a full-spectrum fit with PSF model
* Given knowledge of features in the spectrum, find lines in the
measured spectrum and re-fit to refine the dispersion relation
* Reflatfield the image with the refined dispersion relation

Parameters
---
exp : `afw.image.Exposure`
The postISR exposure in which to find the main star

Returns
---
spectrum : `lsst.atmospec.spectrum` - TODO: DM-18133
The wavelength-calibrated 1D stellar spectrum
"""

if self.config.doDisplayPlots: # no pdfpages backend - isn't compatible with display-as-you-go
    spectractor = SpectractorShim(configFile=configFilename,
                                  paramOverrides=overrideDict,

supplementaryParameters=supplementDict,
                                  resetParameters=resetParameters)
    result = **spectractor.run**(exp, *sourceCentroid, target, spectractorOutputRoot)

Configuration of ProcessStarTask

In atmospec/pipelines/processStar.yaml

```
description: atmospec ProcessStarTask definition.
instrument: lsst.obs.lsst.Latiss
tasks:
  isr:
    class: lsst.ip.isr.IsrTask
    config:
      # characterize performance when turning on darks as first attempt found it
      # to degrade performance (possibly due to bad darks, but take care here)
      doDark: False
      doFlat: False
      doFringe: False
      doDefect: True
      doLinearize: False
      doCrosstalk: False
      doSaturationInterpolation: False
      overscan.fitType: 'MEDIAN_PER_ROW'
      doBias: True
  characterizeImage:
    class: lsst.pipe.tasks.characterizeImage.CharacterizeImageTask
    config:
      repair.doCosmicRay: False
      doApCorr: False
      doMeasurePsf: False
      detection.includeThresholdMultiplier: 3
      # assess carefully whether turning on cosmic ray repair hurts performance
  singleStarCentroidTask: lsst.atmospec.centroiding.SingleStarCentroidTask
  processStarTask:
    class: lsst.atmospec.processStar.ProcessStarTask
    config:
      binning: 4
      doDisplayPlots: True
      doSavePlots: True
      spectractorDebugMode: True
      spectractorDebugLogging: True
```

Part 2 : Auxtel at CCIN2P3

Need to install the Butler environnement

Setting up a postgreSQL environment

gen3 relies on a postgreSQL database to handle its registry (it can in principle also use a SQLite3 database, but this solution is disfavored as it doesn't scale).

You should first open a ticket on the user support portal (https://support.cc.in2p3.fr/) to request an account on the CC-IN2P3 Rubin-LSST postgreSQL database server: ccpglsstdev.in2p3.fr

Once you have the password, you need to create a database and a schema on the postgreSQL system (be careful if you are familiar with SQL DB system the notion of database and schema is specific to postgreSQL).

Initialize the Rubin stack:

```
source /cvmfs/sw.lsst.eu/linux-x86_64/lsst_distrib/w_2022_09/loadLSST.bash setup
lsst_distrib
```

Connect to postgreSQL and run some initialization (this should be done only the first time you access to postgreSQL):

```
psql -h ccpglsstdev.in2p3.fr -p 6553 -U your_username -d postgres
>>> provide your password
   CREATE DATABASE your_username_butler1;
   \connect your_username_butler1;
   CREATE EXTENSION btree_gist;
   CREATE SCHEMA the_name_of_your_butler;
```

Please note that the whole Rubin-LSST group shares the same posgreSQL environment at CC-IN2P3, so it is important to clearly identify your database with your name.
If necessary, you can create several databases but in principle you can use only one and create as many schemas as you need (one schema per different butler).

Store your credentials

The stack needs to have access to your credentials in order to connect to your postgreSQL database. They should be kept securely in $HOME/.lsst/db-auth.yaml

```
cat > $HOME/.lsst/db-auth.yaml
- url: "postgresql://ccpglsstdev.in2p3.fr:6553/your_username_butler1"
  username: your_username
  password: xxxxxxxxxxxxxxxx
^D
chmod 600 ~/.lsst/db-auth.yaml
```

Be very careful not to forget about the last command (chmod), this file should be readable by you only !

Butler initialization

To create the butler structure, you first need to create a bootstrap file:

```
cd /sps/lsst/users/your_username/your_directory
cat > butler-seed.yaml
registry:
  db: "postgresql://ccpglsstdev.in2p3.fr:6553/your_database_name"
  namespace: "the_name_of_your_butler"
```

Then:

```
mkdir data
butler create --seed-config butler-seed.yaml --override ./data
butler register-instrument ./data 'lsst.obs.lsst.LsstCam'
```

Where LsstCam is the name of the instrument corresponding to the full Rubin-LSST focal plane. For AuxTel it should be replaced by Latiss
The instrument names can be found in the file:
https://github.com/lsst/obs_lsst/blob/main/python/lsst/obs/lsst/_instrument.py

Raw data ingestion

The command to ingest raw data in the butler is:

```
butler ingest-raws --transfer symlink -j 4  ./data path_to_the_raw_data
```

42

Instructions from Dominique to set a postgreSQL environnement At CC-IN2P3

Add the credential file

- Create the Butler top repository
- Create the ./data according Butler rules
- Ingest files (ingest commands)
  - Raw data,
  - Catalog Gaia, Pan-starrs
  - Detector calibration

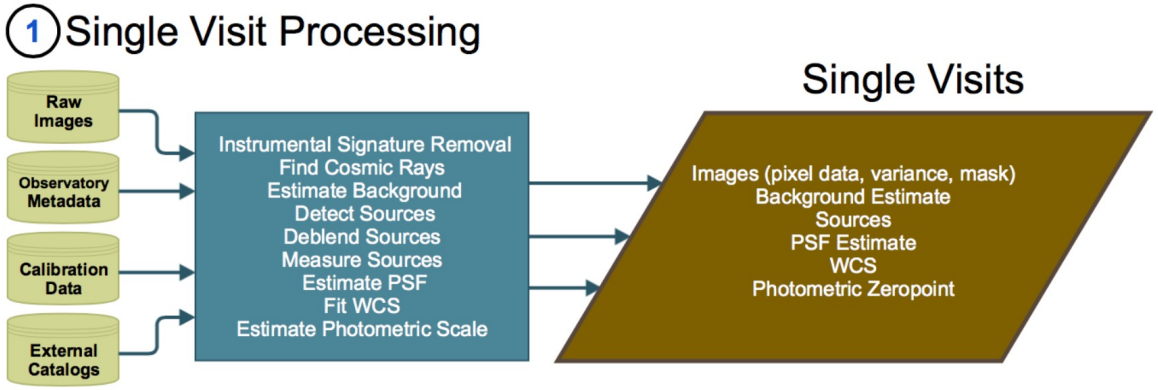


FIGURE 3: Illustration of the conceptual algorithm design for Single Visit Processing pipeline.

Setting the DM environnement

source /cvmfs/sw.lsst.eu/linux-x86_64/lsst_distrib/w_2022_09/loadLSST.bash

setup lsst_distrib

Installing DM additionnal packages

In /sps/lsst/groups/auxtel/softs/shared/auxteldm_gen3/**repos/w_2022_09/**

```
atmospec
rapid_analysis
user_setup.sh   →. For the jupyter kernel
# init the installation
```

source /sps/lsst/groups/auxtel/softs/shared/auxteldm_gen3/**repos/w_2022_09/user_setup.sh**

check installation

eups list -s | grep LOCAL

Ingestion of calibrations inside the Butler

From #in2p3-auxteldm channel

Dominique Boutigny (he/him) 14 h 54

Pour les calibrations, tu peux faire:

Astrometry calibrations

Pour les catalogues de références:

```
butler ingest-files -t direct ./data /sps/lsst/users/boutigny/auxtel-gen3/gaia_dr2_20191105 refcats gaia_dr2_20191105.ecsv
butler ingest-files -t direct ./data /sps/lsst/users/boutigny/auxtel-gen3/ps1_pv3_3pi_20170110 refcats pan-starrs.ecsv
butler register-dataset-type ./data ps1_pv3_3pi_20170110 SimpleCatalog htm7
butler register-dataset-type ./data gaia_dr2_20191105  SimpleCatalog htm7
```

Dominique Boutigny (he/him) 15 h 11

Ah desolé:

```
butler ingest-files -t direct ./data gaia_dr2_20191105 refcats /sps/lsst/users/boutigny/auxtel-gen3/gaia_dr2_20191105.ecsv
```

Dominique Boutigny (he/him) 15 h 35

C'est le même problème que pour gaia. Le path n'est pas au bon endroit

butler ingest-files -t direct ./data ps1_pv3_3pi_20170110 refcats /sps/lsst/users/boutigny/auxtel-gen3/pan-starrs.ecsv

Detector calibrations

Dominique Boutigny (he/him) 15 h 04

Il manque le path devan le deuxième `./import-calibs`

```
butler import --transfer copy --export-file /sps/lsst/users/boutigny/auxtel-gen3/import-calibs/export.yaml ./data
/sps/lsst/users/boutigny/auxtel-gen3/import-calibs -s instrument -s detector -s physical_filter
```

Ingestion of the raw images

From #in2p3-auxteldm channel

sylvie dagoret 11 h 40

Je crois

donc /sps/lsst/groups/auxtel/softs/shared/auxteldm_gen3"("0")>"butler ingest-raws --transfer symlink -j 4 ./data /sps/lsst/groups/auxtel/data/raw_ncsa/*

Dominique Boutigny (he/him) 11 h 43

Quand ce sera fini, il faudra que tu passes cette commande:

```
butler define-visits -C config.txt --collections 'LATISS/raw/all' ./data 'LATISS
```

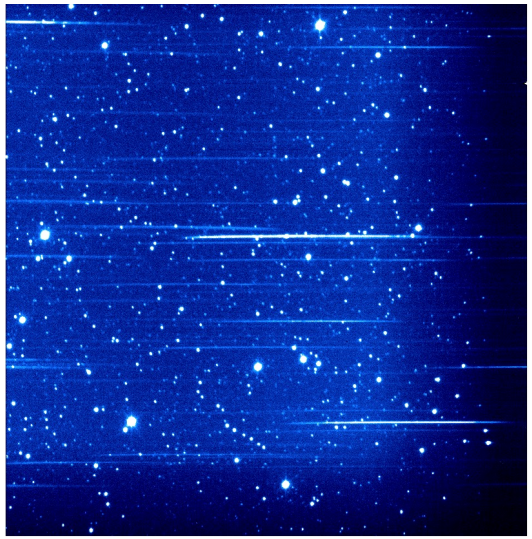
où le fichier `config.txt` contient:

```
config.groupExposures.name="one-to-one"
```

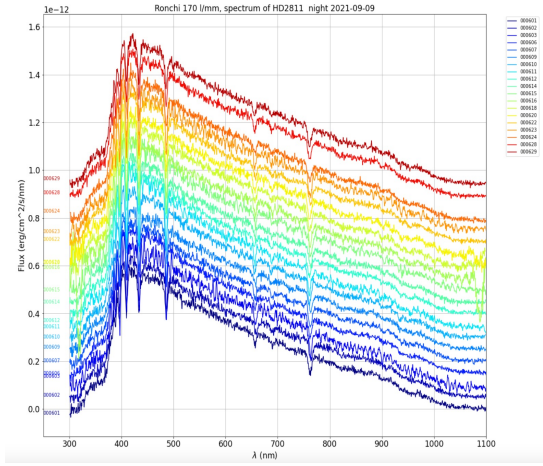
butler ingest-raws --transfer symlink -j 4 ./data /sps/lsst/groups/auxtel/data/raw_ncsa/2022-03-16/*
butler ingest-raws --transfer symlink -j 4 ./data /sps/lsst/groups/auxtel/data/raw_ncsa/2022-03-17/*
butler ingest-raws --transfer symlink -j 4 ./data /sps/lsst/groups/auxtel/data/raw_ncsa/2022-03-18/*
butler define-visits -C config.txt --collections 'LATISS/raw/all' ./data 'LATISS'

Process an image

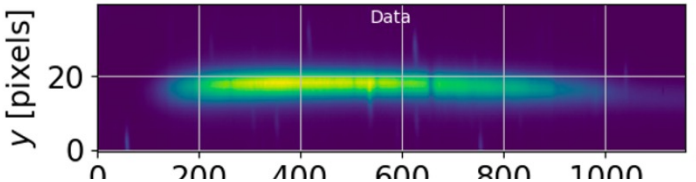
pipetask run -b ./data -p repos/w_2022_09/atmospec/pipelines/processStar.yaml -i LATISS/raw/all,refcats,LATISS/calib -o u/dagoret/first_test2 -d "exposure.day_obs=20220316 and exposure.seq_num=330 and instrument='LATISS'" --register-dataset-types --clobber-outputs



processStar task



@CCIN2P3 : lauch processStar task on preselected exposure of One night from a batch script (NCSA, notebooks)



Work around the pipetask

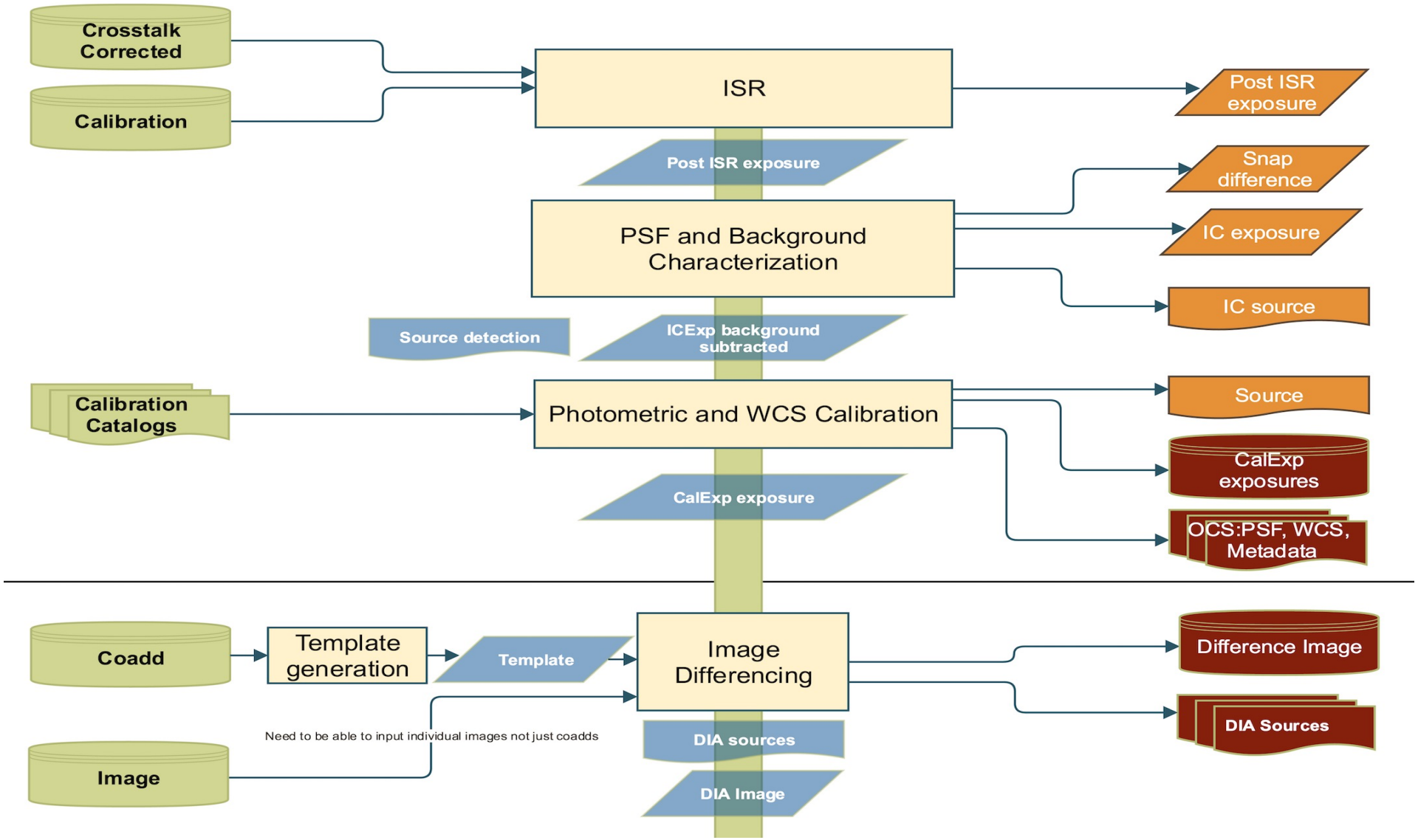
- **Book-keeping:**
  - Select the images according type to know what to process:
    - Bias, flat, disperser-no filters, disperser-filter
  - To organize the storage of the outputs

- **Extract fits images from Butler world**
  - Fits image must conform Spectractor orientation, format requirements
- **IO handling:**
  - Must define one's own « Butler » or IO conventions on disk

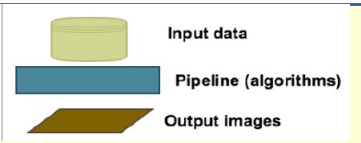
- **Run batch jobs** (one job arrays per night),

calibration (found in an old note)

https://dmtn-039.lsst.io/v/DM-8799/

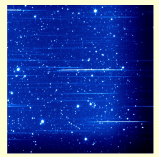


How to have impact on exposure calibration in Rubin



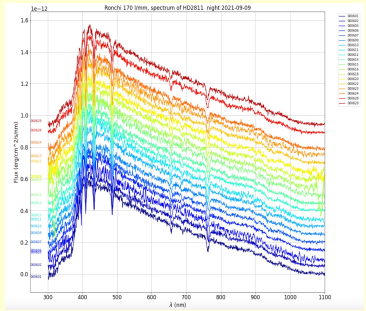
AUXTEL DATA

Raw images

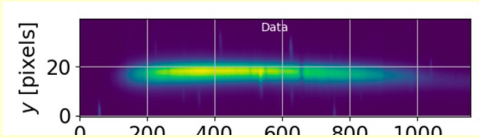


processStar
(pack atmospec)

Spectra, Spectrogram



Spectractor DM version



Design a new calibration task ?

postISR CCD

LSST DATA

Identified sources

CALEXP ?

Calibrated sources ?

Need to investigate more !

Coaddition task

Butler world

Backup

Glossary

https://confluence.lsstcorp.org/display/DM/Data+Butler+Product+Definition

Butler

The Butler is a framework for generic I/O and data management. It isolates application code from the underlying data-access implementation in terms of storage formats, physical locations, data staging, database mapping, etc.

Butler is configured by a Policy that provides configuration details, as well as by parameters provided at initialization time.

Dataset

A dataset is the persisted form of an in-memory object. It could be a single item, a composite, or a collection. Examples: `int`/`long`, `PropertySet`, `ExposureF`, WCS, PSF, `set`/`list`/`dict`.

Repository

Repository is an abstract concept that refers to data storage.

A **Dataset Repository** is a collection of datasets with an associated configuration. It may also contain metadata databases that assist with finding datasets. A repository can be versioned. Repositories can point to other repositories, providing what is in effect a search path for datasets. This allows repositories to share access to datasets without copying data and modify data without overwriting previously written data.

An **AggregateRepository** contains Repositories.

Dataset Type

A label given to a one or more datasets reflecting their meaning or usage (not their persisted representation). Each dataset type corresponds to exactly one Python type. Dataset types are used by convention by Tasks for their inputs and outputs. Examples: `calexp`, `src`, `icSrc`.

Dataset Genre

A labeled set of basic access characteristics serving as the basis for a group of dataset types, used to define new dataset types. The characteristics may include code, template strings, and other configuration data. Dataset genres are often (but not necessarily) common to all dataset types with the same Python type, making it easy for an application to select which genre is applicable to a new dataset type that it is creating.

Storage

A mechanism for reading/writing a dataset to/from an in-memory object. Particular storage methods or classes are used for writing to and reading from FITS files, databases, or other physical representations.

Policy

The policy provides configuration details for the butler framework that will access a dataset. The policy items may be set at the repository, butler subclass, and butler framework levels. Settings will be applied in that order, where the in-repository settings will have highest priority.

dataId

Scientifically meaningful key-value pairs used to indicate a dataset or datasets that should be retrieved and how datasets should be serialized.

DataRef

A `DataId` packaged with a `Butler` for access to datasets. A `DataRef` an be used with multiple dataset types (if the keys are appropriate).

DataRefSet

Logically, a set of 'DataRef's. This may be implemented as an iterator/generator in some contexts where materializing the set would be expensive. The 'DataRefSet' is usually generated by listing existing datasets of a particular dataset type, but its component 'DataRef's can be used with other dataset types.

Commonly--used data products

- raw: Raw images
- bias, dark, flat, fringe: Calibration frames
- postISRCCD: ISR applied and assembled CCD
- calexp: Calibrated exposures
- icSrc: Source catalog used for calibration
- src: Source catalog
- icMatch: Join table for “icSrc” with reference catalog

– Use Astrometry.joinMatchListWithCatalog or readMatches in meas_astrom

- deepCoadd_tempExp: Warped image
- deepCoadd: Coadd image
- deepCoadd_src: Source catalog for coadd