



# **vison Documentation**

***Release 0.6+74.gf031d8f***

**Ruyman Azzollini**

**Sep 27, 2018**



# CONTENTS

<b>1</b>	<b>README</b>	<b>3</b>
<b>2</b>	<b>Installation</b>	<b>5</b>
2.1	Installation . . . . .	5
2.2	Dependencies . . . . .	7
<b>3</b>	<b>Pipeline Core</b>	<b>9</b>
3.1	Pipeline . . . . .	9
<b>4</b>	<b>Data Model</b>	<b>35</b>
4.1	Data Model . . . . .	35
<b>5</b>	<b>Analysis (Shared)</b>	<b>45</b>
5.1	Analysis (Shared) . . . . .	45
<b>6</b>	<b>Charge Injection Tools</b>	<b>49</b>
6.1	Charge Injection Tools . . . . .	49
<b>7</b>	<b>“Flat” Acq. Analysis Tools</b>	<b>51</b>
7.1	“Flat” Acq. Analysis Tools . . . . .	51
<b>8</b>	<b>Image</b>	<b>53</b>
8.1	Image Analysis . . . . .	53
<b>9</b>	<b>Monitoring (“Eyegore”)</b>	<b>55</b>
9.1	Monitoring (“Eyegore”) . . . . .	55
<b>10</b>	<b>OGSE</b>	<b>59</b>
10.1	OGSE Tools . . . . .	59
<b>11</b>	<b>Plotting</b>	<b>61</b>
11.1	Plotting . . . . .	61
<b>12</b>	<b>Point-Source Analysis</b>	<b>63</b>
12.1	Point-Source Analysis . . . . .	63
<b>13</b>	<b>Scripts</b>	<b>69</b>
13.1	Scripts . . . . .	69
<b>14</b>	<b>Support Code</b>	<b>73</b>
14.1	Support Code . . . . .	73

<b>15 Unit Testing</b>	<b>77</b>
15.1 Unit Testing . . . . .	77
<b>16 Test Scripts</b>	<b>79</b>
16.1 Charge Injection Scripts . . . . .	79
16.2 Dark Scripts . . . . .	81
16.3 Flat-Illumination Scripts . . . . .	83
16.4 Point-Source Scripts . . . . .	88
16.5 Trap-Pumping Scripts . . . . .	89
16.6 Other Test Scripts . . . . .	92
<b>17 Indices and tables</b>	<b>95</b>
<b>Python Module Index</b>	<b>97</b>

Contents:



**README**

# vison Euclid VIS Ground Calibration Pipeline

**Author** Ruyman Azzollini

**Contact** r.azzollini\_at\_ucl.ac.uk

**issue** 0.6+74.gf031d8f

**version** 0.6+74.gf031d8f

**date** Sep 27, 2018

This Python package “vison” is the pipeline that will be used at MSSL for ground calibration of the VIS detection chains (12 + 2 spares), including one ROE, one RPSU and three CCDs each.





## INSTALLATION

The package is distributed via github. The repository is hosted at:

<https://github.com/ruymanengithub/vison>

Detailed instructions:

### 2.1 Installation

#### 2.1.1 Cloning *vison* from the repository using *git*

If you don't have *git* installed in your system, please follow this [link](#) first.

Here we will follow these [instructions](#) to clone the repository to your own computer. Follow the link for instructions in other operative systems.

Step-by-step:

- Go to <https://github.com/ruymanengithub/vison>.
- Click on the green “Clone or download” button.
- In the Clone with HTTPs section, click to copy the clone URL for the repository.
- Open a Terminal.
- Change the current working directory to the location where you want the cloned directory to be made.
- Type `git clone`, and then paste the URL you copied in Step 1.

```
~$ git clone https://github.com/ruymanengithub/vison
```

- Press Enter. Your local clone will be created.

#### 2.1.2 Installation

We recommend installing the code through a *conda* environment, with a specific list of packages, so you can be sure you have all the needed dependencies.

First, if you don't have *conda* already installed in your system already, follow the instructions in this [link](#).

The screenshot shows the GitHub repository interface for 'ruymanengithub / vison'. At the top, there are navigation links for 'Features', 'Business', 'Explore', and 'Pricing'. A search bar and 'Sign in or Sign up' button are also present. Below the repository name, there are buttons for 'Watch' (1), 'Star' (0), and 'Fork' (0). A navigation bar includes 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Pulse', and 'Graphs'. The main content area is titled 'Euclid VIS Ground Calibration Pipeline' and displays repository statistics: 49 commits, 2 branches, 0 releases, 1 contributor, and GPL-3.0 license. There are buttons for 'Branch: master', 'New pull request', 'Find file', and 'Clone or download'. A list of recent commits is shown, including 'manual\_vison.pdf updated' (3 days ago), 'docs' (3 days ago), 'vison' (3 days ago), and 'LICENSE' (4 months ago).

## Installing conda and creating *vison* environment

Once you have successfully installed conda, we will create an environment that will allow you to install the pipeline and meet all its dependencies.

Step-by-Step:

- change directory to your copy of the vison repository:

```
~$ cd vison
```

- Under the 'conda' sub-folder, you will find several text files:

```
~$ cd conda
~$ ls
env-conda_vison_linux.txt  env-pip_vison.txt
```

- Then execute the following command to create a new conda environment, *vison*.

Use the OS version that may correspond in your case (by now, only linux-64 bits version available).

```
:: ~$ conda create -n vison --file env-conda_[OS].txt
```

- When prompted, type “y” and return to install the listed packages.
- Activate the new environment

```
~$ source activate vison
```

- Install the packages that are accessed via *pip*, within the conda environment:

```
~$ pip install -r env-pip_vison.txt
```

## Installing *vison*

Finally, to install the *vison* pipeline itself, we will go back to the folder we downloaded from the github repository:

```
~$ cd ../
~$ ls
conda docs LICENSE manual_vison.pdf README.md setup.cfg setup_distutils.py
↪setup.py vison
```

Then do the actual installation, via:

```
~$ python setup.py install
```

Now the vison package will be accessible from anywhere in your system, whenever you start python from within the *vison* conda environment. For example:

- open a new terminal and go to your home directory

```
~$ cd
```

- activate the vison environment:

```
~$ source activate vison
```

- start the python interpreter and import vison:

```
~$ source activate vison
~$ python
>>> import vison
>>> dir(vison)
['Eyegore', 'FlatFielding', 'Pipe', 'Report', '__all__', '__builtins__', '__doc__
↪', '__file__',
'__name__', '__package__', '__path__', '__version__', 'analysis', 'data',
↪'datamodel',
'eyegore', 'pipe', 'point', 'support']
```

## 2.2 Dependencies

Instructions to acquire a copy of the “conda” environment that provides all dependencies is included in the package. See [Installation](#) instructions for details.



## PIPELINE CORE

Pipeline master classes.

### 3.1 Pipeline

#### 3.1.1 master.py

This is the main script that will orchestrate the analysis of Euclid-VIS FM Ground Calibration Campaign.

The functions of this module are:

- Take inputs as to what data is to be analyzed, and what analysis scripts are to be run on it.
- Set the variables necessary to process this batch of FM calib. data.
- Start a log of actions to keep track of what is being done.
- Provide inputs to scripts, execute the analysis scripts and report location of analysis results.

Some Guidelines for Development:

- Input data is “sacred”: read-only.
- Each execution of Master must have associated a unique ANALYSIS-ID.
- All the Analysis must be divided in TASKS. TASKS can have SUB-TASKS.
- All data for each TASK must be under a single day-folder.
- All results from the execution of FMmaster must be under a single directory with subdirectories for each TASK run.
- **A subfolder of this root directory will contain the logging information:** inputs, outputs, analysis results locations.

Created on Wed Jul 27 12:16:40 2016

**author** Ruyman Azzollini

```
class vison.pipe.master.Pipe (inputdict, dolog=True, drill=False, debug=False, startobsid=0, processes=1)
```

Master Class of FM-analysis

```
class BF01 (inputs, log=None, drill=False, debug=False)
```

```
build_scriptdict (diffvalues={}, elvis='6.5.X')  
Builds PTCOX script structure dictionary.
```

#:param exptimes: list of ints [ms], exposure times. #:param frames: list of ints, number of frames for each exposure time. #:param wavelength: int, wavelength. Default: 800 nm. :param diffvalues: dict, opt, differential values.

**extract\_BF()**

**Performs basic analysis of images:**

- extracts BF matrix for each COV matrix

**extract\_COV()**

**Performs basic analysis of images:**

- extracts COVARIANCE matrix for each fluence

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

Analyzes the BF results across fluences.

**set\_inpdefaults** (*\*\*kwargs*)

**class Pipe.BIAS01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

BIAS01: Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:

    load ccdobj of ObsID, CCD

    with ccdobj, f.e.Q:
      produce a 2D poly model of bias, save coefficients
      produce average profile along rows
      produce average profile along cols
      # save 2D model and profiles in a pick file for each OBSID-CCD
      measure and save RON after subtracting large scale structure

plot RON vs. time f. each CCD and Q
plot average profiles f. each CCD and Q (color coded by time)
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds BIAS01 script structure dictionary.

###:param N: integer, number of frames to acquire. :param diffvalues: dict, opt, differential values.  
:param elvis: char, ELVIS version.

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

**METACODE**

```
f. each CCD:
  stack all ObsIDs to produce Master Bias
  f. e. Q:
    measure average profile along rows
    measure average profile along cols
plot average profiles of Master Bias(s) f. each CCD,Q
(produce table(s) with summary of results, include in report)
save Master Bias(s) (3 images) to FITS CDPs
```

```
show Master Bias(s) (3 images) in report
save name of MasterBias(s) CDPs to DataDict, report
```

**prep\_data()**

BIAS01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**class** Pipe.CHINJ00 (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds CHINJ00 script structure dictionary.

**Parameters** **diffvalues** – dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)**set\_inpdefaults** (*\*\*kwargs*)

**class** Pipe.CHINJ01 (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds CHINJ01 script structure dictionary.

**#:param** IDL: float, [V], value of IDL (Inject. Drain Low). **#:param** IDH: float, [V], Injection Drain High. **#:param** IG2: float, [V], Injection Gate 2. **#:param** IG1s: list of 2 floats, [V], [min,max] values of IG1. **#:param** id\_delays: list of 2 floats, [us], injection drain delays. **#:param** toi\_chinj: int, [us], TOI-charge injection. **#:param** diffvalues: dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)**meta\_analysis()**

Find injection threshold: Min IG1 Plot and model charge injection vs. IG1 Find notch injection amount.

**old\_basic\_analysis()**

Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      extract average 2D injection pattern (and save)
      produce average profile along/across lines
      measure charge-inj. non-uniformity
      measure charge spillover into non-injection
      measure stats of injection (mean, med, std, min/max, ↳
↳percentiles)

plot average inj. profiles along lines f. each CCD, Q and IG1
  save as a rationalized set of curves
plot average inj. profiles across lines f. each CCD, Q and IG1
  save as a rationalized set of curves

Report injection stats as a table/tables
```

**set\_inpdefaults** (*\*\*kwargs*)

**class** Pipe.CHINJ02 (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

Basic analysis of data. AS IT IS, REPEATS WHAT'S DONE IN THE CHECK\_DATA. CONSIDER MERGING/SKIPPING

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      load average 2D injection pattern
      produce average profile along lines
      [measure charge-inj. non-uniformity]
      [produce average profile across lines]
      [measure charge spillover into non-injection]
      measure stats of injection (mean, med, std, min/max,
↳percentiles)

[plot average inj. profiles along lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]
[plot average inj. profiles across lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]

save&plot charge injection vs. IDL
report injection stats as a table
```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

Builds CHINJ02 script structure dictionary.

#:param IDLs: list of 2 ints, [V], [min,max] values of IDL (Inject. Drain Low). #:param IDH: int, [V], Injection Drain High. #:param id\_delays: list of 2 ints, [us], injection drain delays. #:param toi\_chinj: int, [us], TOI-charge injection. :param diffvalues: dict, opt, differential values.

**extract\_data()**

NEEDED? Could be merged with basic\_analysis

**METACODE**

```
Preparation of data for further analysis:

f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      subtract offset
      extract average 2D injection pattern and save
```

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis()**

Finds the Injection Threshold for each CCD half.

**METACODE**

```
f.e.CCD:
  f.e.Q:
    load injection vs. IDL cuve
    find&save injection threshold on curve

report injection threshold as a table
```

**set\_inpdefaults** (\*\*kwargs)



**class Pipe.DARK01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis** ()

DARK01: Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      produce mask of hot pixels
      count hot pixels / columns
      produce a 2D poly model of masked-image, save coefficients
      produce average profile along rows
      produce average profile along cols
      measure and save RON after subtracting large scale structure
      save 2D model and profiles in a pick file for each OBSID-CCD

plot average profiles f. each CCD and Q (color coded by time)
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds DARK01 script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

**METACODE**

```
f. each CCD:
  f. e. Q:
    stack all ObsIDs to produce Master Dark
    produce mask of hot pixels / columns
    count hot pixels / columns
    measure average profile along rows
    measure average profile along cols

plot average profiles of Master Bias f. each CCD,Q
show Master Dark (images), include in report
report stats of defects, include in report
save name of MasterDark to DataDict, report
save name of Defects in Darkness Mask to DD, report
```

**prep\_data** ()

DARK01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [BIAS SUBTRACTION] cosmetics masking

**class Pipe.FLAT0X** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds FLAT0X script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**do\_indiv\_flats** ()

**METACODE**

```
Preparation of data for further analysis and
produce flat-field for each OBSID.
```

```
f.e. ObsID:
  f.e.CCD:

    load ccdobj

  f.e.Q:

    model 2D fluence distro in image area
    produce average profile along rows
    produce average profile along cols

    save 2D model and profiles in a pick file for each OBSID-CCD
    divide by 2D model to produce indiv-flat
    save indiv-Flat to FITS(?), update add filename

plot average profiles f. each CCD and Q (color coded by time)
```

**do\_master\_flat()**

**METACODE**

```
Produces Master Flat-Field

f.e.CCD:
  f.e.Q:
    stack individual flat-fields by chosen estimator
save Master FF to FITS
measure PRNU and
report PRNU figures
```

**do\_prdef\_mask()**

**METACODE**

```
Produces mask of defects in Photo-Response
Could use master FF, or a stack of a subset of images (in order
to produce mask, needed by other tasks, quicker).

f.e.CCD:
  f.e.Q:
    produce mask of PR defects
    save mask of PR defects
    count dead pixels / columns

report PR-defects stats
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**prepare\_images()**

FLAT0X: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [bias structure subtraction, if available] cosmetics masking

**set\_inpdefaults** (\*\**kwargs*)

**class** Pipe.**FOCUS00** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

This is just an assignation of values measured in check\_data.

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

Builds FOCUS00 script structure dictionary.

*#:param wavelength: int, [nm], wavelength. #:param exptime: int, [ms], exposure time. :param diffvalues: dict, opt, differential values.*

**filterexposures** (*structure, explog, OBSID\_lims*)

**lock\_on\_stars** ()

**meta\_analysis** ()

**prep\_data** ()

**class Pipe.MOT\_FF** (*inputs, log=None, drill=False, debug=False*)

**extract\_HER** ()

**class Pipe.NL01** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

Builds NL01 script structure dictionary.

*#:param expts: list of ints [ms], exposure times. #:param exptinter: int, ms, exposure time of interleaved source-stability exposures. #:param frames: list of ints, number of frames for each exposure time. #:param wavelength: int, wavelength. Default: 0 (Neutral Density Filter) :param diffvalues: dict, opt, differential values.*

**do\_satCTE** ()

**METACODE**

```
select ObsIDs with fluence(exptime) >~ 0.5 FWC

f.e. ObsID:
  CCD:
    Q:
      measure CTE from amount of charge in over-scan relative to_
↔fluence

f.e. CCD:
  Q:
    get curve of CTE vs. fluence
    measure FWC from curve in ADU

report FWCs in electrons [via gain in inputs] f.e. CCD, Q (table)
```

**extract\_stats** ()

Performs basic analysis: extracts statistics from image regions to later build NLC.

**METACODE**

```
create segmentation map given grid parameters

f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      f.e. "img-segment": (done elsewhere)
      measure central value
      measure variance
```

**filterexposures** (*structure, explog, OBSID\_lims*)

Loads a list of Exposure Logs and selects exposures from test PSF0X.

The filtering takes into account an expected structure for the acquisition script.

The datapath becomes another column in DataDict. This helps dealing with tests that run overnight and for which the input data is in several date-folders.

**prep\_data** ()

Takes Raw Data and prepares it for further analysis.

**METACODE**

```
f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      subtract offset
    opt: [sub bias frame]
    opt: [divide by FF]
    opt: [mask-out defects]
```

**produce\_NLCs** ()

**METACODE**

```
Obtains Best-Fit Non-Linearity Curve

f.e. CCD:
  f.e. Q:

    [opt] apply correction for source variability (interspersed_
↪exposure
    with constant exptime)
    Build NL Curve (NLC) - use stats and exptimes
    fit poly. shape to NL curve

plot NL curves for each CCD, Q
report max. values of NL (table)
```

**class** Pipe.**PERSIST01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```
f.e.CCD:
  f.e.Q:
    use SATURATED frame to generate pixel saturation MASK
    measure stats in pix satur MASK across OBSIDs
    (pre-satur, satur, post-satur)
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds PERSISTENCE01 script structure dictionary.

**Parameters**

- **exptSATUR** – int, saturation exposure time.
- **exptLATEN** – int, latency exposure time.
- **diffvalues** – dict, differential values.

**check\_data()**

PERSIST01: Checks quality of ingested data.

**METACODE**

```

check common HK values are within safe / nominal margins
check voltages in HK match commanded voltages, within margins

f.e.ObsID:
  f.e.CCD:
    f.e.Q.:
      measure offsets in pre-, over-
      measure std in pre-, over-
      measure fluence in apertures around Point Sources

assess std in pre- (~RON) is within allocated margins
assess offsets in pre-, and over- are equal, within allocated margins
assess fluence is ~expected within apertures (PS) for each frame (pre-
↳satur, satur, post-satur)

plot point source fluence vs. OBSID, all sources
[plot std vs. time]

issue any warnings to log
issue update to report

```

**filterexposures** (*structure, explog, OBSID\_lims*)**meta\_analysis()**

Meta-analysis of data.

**METACODE**

```

f.e.CCD:
  f.e.Q:
    estimate delta-charge_0 and decay tau from time-series

report:
  persistence level (delta-charge_0) and time constant

```

**prep\_data()**

PERSIST01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**PTCOX** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds PTCOX script structure dictionary.

**#:param** exptimes: list of ints [ms], exposure times. **#:param** frames: list of ints, number of frames for each exposure time. **#:param** wavelength: int, wavelength. Default: 800 nm. **:param** diffvalues: dict, opt, differential values.

**extract\_PTC()**

**Performs basic analysis of images:**

- builds PTC curves: both on non-binned and binned images

### METACODE

```

create list of OBSID pairs

create segmentation map given grid parameters

f.e. OBSID pair:
  CCD:
    Q:
      subtract CCD images
      f.e. segment:
        measure central value
        measure variance
    
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Analyzes the variance and fluence: gain, and gain(fluence)

### METACODE

```

f.e. CCD:
  Q:
    (using stats across segments:)
    fit PTC to quadratic model
    solve for gain
    solve for alpha (pixel-correls, Guyonnet+15)
    solve for blooming limit (ADU)
      convert bloom limit to electrons, using gain

plot PTC curves with best-fit f.e. CCD, Q
report on gain estimates f. e. CCD, Q (table)
report on blooming limits (table)
    
```

**set\_inpdefaults** (\*\*kwargs)

**class Pipe. STRAY00** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds STRAY00 script structure dictionary. :param diffvalues: dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)

**set\_inpdefaults** (\*\*kwargs)

**class Pipe. TP00** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

**check\_data** ()

TP01: Checks quality of ingested data.

### METACODE

```

check common HK values are within safe / nominal margins
check voltages in HK match commanded voltages, within margins

f.e.ObsID:
  f.e.CCD:
    f.e.Q.:
    
```

```

        measure offsets in pre-, over-
        measure std in pre-, over-
        measure mean in img-

    assess std in pre- (~RON) is within allocated margins
    assess offsets in pre-, and over- are equal, within allocated margins
    assess offsets are within allocated margins
    assess injection level is within expected margins

    plot histogram of injected levels for each Q
    [plot std vs. time]

    issue any warnings to log
    issue update to report

```

**filterexposures** (*structure, explog, OBSID\_lims*)

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**TP01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```

f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
    f.e.Q:
      load "map of relative pumping"
      find_dipoles:
        x, y, rel-amplitude, orientation

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for N and S)
  Counts of dipoles (and N vs. S)

```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

**extract** ()

Obtain maps of dipoles.

**METACODE**

```

f.e. id_delay (there are 2):
  f.e. CCD:
    f.e. Q:
      produce reference non-pumped injection map

f. e. ObsID:
  f.e. CCD:

    load ccdobj
    f.e.Q.:
      divide ccdobj.Q by injection map

    save dipole map and store reference

```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

**METACODE**

```

across TOI_TP, patterns:
  build catalog of traps: x,y,I-phase, Amp
  from Amp(TOI) -> tau, Pc

Report on :
  Histogram of Taus
  Histogram of Pc (capture probability)
  Histogram of I-phases (larger phases should have more traps,
                        statistically) -> check

  Total Count of Traps
    
```

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**TP02** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```

f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
    f.e.Q:
      load raw 1D map of relative pumping (from extract_data)
      identify dipoles:
        x, rel-amplitude, orientation (E or W)

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for E and W)
  Counts of dipoles (and E vs. W)
    
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

**extract** ()

Obtain Maps of Serial Dipoles.

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

**METACODE**

```

across TOI_TP, patterns:
  build catalog of traps: x,y,R-phase, amp(dwell)
  from Amp(dwell) -> tau, Pc

Report on :
    
```



```

Histogram of Taus
Histogram of Pc (capture probability)
Histogram of R-phases

Total Count of Traps
    
```

```

set_inpdefaults (**kwargs)

Pipe.catchtraceback()

Pipe.dotask (taskname, inputs, drill=False, debug=False)
    Generic test master function.

Pipe.get_execution_summary (exectime=None)

Pipe.get_test (taskname, inputs={}, log=None, drill=False, debug=False)

Pipe.launchtask (taskname)

Pipe.run (explogf=None, elvis=None)

Pipe.wait_and_run (dayfolder, elvis='6.5.X')

class vison.pipe.master.Pipe (inputdict, dolog=True, drill=False, debug=False, startobsid=0, processes=1)
    Master Class of FM-analysis

class BF01 (inputs, log=None, drill=False, debug=False)

    build_scriptdict (diffvalues={}, elvis='6.5.X')
        Builds PTCOX script structure dictionary.

        #:param exptimes: list of ints [ms], exposure times. #:param frames: list of ints, number of frames for
        each exposure time. #:param wavelength: int, wavelength. Default: 800 nm. :param diffvalues: dict,
        opt, differential values.

    extract_BF()
        Performs basic analysis of images:
        • extracts BF matrix for each COV matrix

    extract_COV()
        Performs basic analysis of images:
        • extracts COVARIANCE matrix for each fluence

    filterexposures (structure, explog, OBSID_lims)

    meta_analysis()
        Analyzes the BF results across fluences.

    set_inpdefaults (**kwargs)

class Pipe.BIAS01 (inputs, log=None, drill=False, debug=False)

    basic_analysis()
        BIAS01: Basic analysis of data.

    METACODE
    
```

```

f. e. ObsID:
f.e.CCD:

        load ccdobj of ObsID, CCD
    
```

```

with ccdobj, f.e.Q:
    produce a 2D poly model of bias, save coefficients
    produce average profile along rows
    produce average profile along cols
    # save 2D model and profiles in a pick file for each OBSID-CCD
    measure and save RON after subtracting large scale structure

plot RON vs. time f. each CCD and Q
plot average profiles f. each CCD and Q (color coded by time)

```

**build\_scriptdict** (*diffvalues*={}, *elvis*='6.5.X')

Builds BIAS01 script structure dictionary.

###:param N: integer, number of frames to acquire. :param diffvalues: dict, opt, differential values.  
:param elvis: char, ELVIS version.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

**METACODE**

```

f. each CCD:
    stack all ObsIDs to produce Master Bias
f. e. Q:
    measure average profile along rows
    measure average profile along cols
plot average profiles of Master Bias(s) f. each CCD,Q
(produce table(s) with summary of results, include in report)
save Master Bias(s) (3 images) to FITS CDPs
show Master Bias(s) (3 images) in report
save name of MasterBias(s) CDPs to DataDict, report

```

**prep\_data** ()

BIAS01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**class** Pipe.CHINJ00 (*inputs*, *log*=None, *drill*=False, *debug*=False)

**build\_scriptdict** (*diffvalues*={}, *elvis*='6.5.X')

Builds CHINJ00 script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**set\_inpdefaults** (\*\**kwargs*)

**class** Pipe.CHINJ01 (*inputs*, *log*=None, *drill*=False, *debug*=False)

**build\_scriptdict** (*diffvalues*={}, *elvis*='6.5.X')

Builds CHINJ01 script structure dictionary.

#:param IDL: float, [V], value of IDL (Inject. Drain Low). #:param IDH: float, [V], Injection Drain High. #:param IG2: float, [V], Injection Gate 2. #:param IG1s: list of 2 floats, [V], [min,max] values of IG1. #:param id\_delays: list of 2 floats, [us], injection drain delays. #:param toi\_chinj: int, [us], TOI-charge injection. :param diffvalues: dict, opt, differential values.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis()**

Find injection threshold: Min IG1 Plot and model charge injection vs. IG1 Find notch injection amount.

**old\_basic\_analysis()**

Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      extract average 2D injection pattern (and save)
      produce average profile along/across lines
      measure charge-inj. non-uniformity
      measure charge spillover into non-injection
      measure stats of injection (mean, med, std, min/max, ↵
↵percentiles)

plot average inj. profiles along lines f. each CCD, Q and IG1
  save as a rationalized set of curves
plot average inj. profiles across lines f. each CCD, Q and IG1
  save as a rationalized set of curves

Report injection stats as a table/tables
```

**set\_inpdefaults(\*\*kwargs)**

**class** Pipe.CHINJ02 (inputs, log=None, drill=False, debug=False)

**basic\_analysis()**

Basic analysis of data. AS IT IS, REPEATS WHAT'S DONE IN THE CHECK\_DATA. CONSIDER MERGING/SKIPPING

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      load average 2D injection pattern
      produce average profile along lines
      [measure charge-inj. non-uniformity]
      [produce average profile across lines]
      [measure charge spillover into non-injection]
      measure stats of injection (mean, med, std, min/max, ↵
↵percentiles)

[plot average inj. profiles along lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]
[plot average inj. profiles across lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]

save&plot charge injection vs. IDL
report injection stats as a table
```

**build\_scriptdict (diffvalues={}, elvis='6.5.X')**

Builds CHINJ02 script structure dictionary.

#:param IDLs: list of 2 ints, [V], [min,max] values of IDL (Inject. Drain Low). #:param IDH: int, [V],

Injection Drain High. #:param id\_delays: list of 2 ints, [us], injection drain delays. #:param toi\_chinj: int, [us], TOI-charge injection. :param diffvalues: dict, opt, differential values.

**extract\_data()**

**NEEDED?** Could be merged with basic\_analysis

**METACODE**

```
Preparation of data for further analysis:

f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      subtract offset
      extract average 2D injection pattern and save
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

Finds the Injection Threshold for each CCD half.

**METACODE**

```
f.e.CCD:
  f.e.Q:
    load injection vs. IDL cuve
    find&save injection threshold on curve

report injection threshold as a table
```

**set\_inpdefaults** (\*\**kwargs*)

**class Pipe.DARK01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

DARK01: Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      produce mask of hot pixels
      count hot pixels / columns
      produce a 2D poly model of masked-image, save coefficients
      produce average profile along rows
      produce average profile along cols
      measure and save RON after subtracting large scale structure
      save 2D model and profiles in a pick file for each OBSID-CCD

plot average profiles f. each CCD and Q (color coded by time)
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds DARK01 script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

**METACODE**

```
f. each CCD:
  f. e. Q:
    stack all ObsIDs to produce Master Dark
    produce mask of hot pixels / columns
    count hot pixels / columns
    measure average profile along rows
    measure average profile along cols

plot average profiles of Master Bias f. each CCD,Q
show Master Dark (images), include in report
report stats of defects, include in report
save name of MasterDark to DataDict, report
save name of Defects in Darkness Mask to DD, report
```

**prep\_data()**

DARK01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [BIAS SUBTRACTION] cosmetics masking

**class** Pipe.**FLAT0X** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds FLAT0X script structure dictionary.

**Parameters** **diffvalues** – dict, opt, differential values.

**do\_indiv\_flats()**

**METACODE**

```
Preparation of data for further analysis and
produce flat-field for each OBSID.

f.e. ObsID:
  f.e.CCD:

    load ccdobj

  f.e.Q:

    model 2D fluence distro in image area
    produce average profile along rows
    produce average profile along cols

    save 2D model and profiles in a pick file for each OBSID-CCD
    divide by 2D model to produce indiv-flat
    save indiv-Flat to FITS(?), update add filename

plot average profiles f. each CCD and Q (color coded by time)
```

**do\_master\_flat()**

**METACODE**

```
Produces Master Flat-Field

f.e.CCD:
  f.e.Q:
    stack individual flat-fields by chosen estimator
save Master FF to FITS
measure PRNU and
```

```
report PRNU figures
```

**do\_prdef\_mask()**

**METACODE**

```
Produces mask of defects in Photo-Response
Could use master FF, or a stack of a subset of images (in order
to produce mask, needed by other tasks, quicker).
```

```
f.e.CCD:
  f.e.Q:
    produce mask of PR defects
    save mask of PR defects
    count dead pixels / columns

report PR-defects stats
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**prepare\_images()**

FLAT0X: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [bias structure subtraction, if available] cosmetics masking

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**FOCUS00** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

This is just an assignation of values measured in check\_data.

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds FOCUS00 script structure dictionary.

**#:param** wavelength: int, [nm], wavelength. **#:param** exptime: int, [ms], exposure time. **:param** diffvalues: dict, opt, differential values.

**filterexposures** (*structure, explog, OBSID\_lims*)

**lock\_on\_stars()**

**meta\_analysis()**

**prep\_data()**

**class** Pipe.**MOT\_FF** (*inputs, log=None, drill=False, debug=False*)

**extract\_HER()**

**class** Pipe.**NL01** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds NL01 script structure dictionary.

**#:param** expts: list of ints [ms], exposure times. **#:param** exptinter: int, ms, exposure time of interleaved source-stability exposures. **#:param** frames: list of ints, number of frames for each exposure time. **#:param** wavelength: int, wavelength. Default: 0 (Neutral Density Filter) **:param** diffvalues: dict, opt, differential values.

**do\_satCTE()**  
**METACODE**

```

select ObsIDs with fluence(exptime) >~ 0.5 FWC

f.e. ObsID:
    CCD:
        Q:
            measure CTE from amount of charge in over-scan relative to_
↳fluence

f.e. CCD:
    Q:
        get curve of CTE vs. fluence
        measure FWC from curve in ADU

report FWCs in electrons [via gain in inputs] f.e. CCD, Q (table)

```

**extract\_stats()**

Performs basic analysis: extracts statistics from image regions to later build NLC.

**METACODE**

```

create segmentation map given grid parameters

f.e. ObsID:
    f.e.CCD:
        f.e.Q:
            f.e. "img-segment": (done elsewhere)
            measure central value
            measure variance

```

**filterexposures** (*structure, explog, OBSID\_lims*)

Loads a list of Exposure Logs and selects exposures from test PSFOX.

The filtering takes into account an expected structure for the acquisition script.

The datapath becomes another column in DataDict. This helps dealing with tests that run overnight and for which the input data is in several date-folders.

**prep\_data()**

Takes Raw Data and prepares it for further analysis.

**METACODE**

```

f.e. ObsID:
    f.e.CCD:
        f.e.Q:
            subtract offset
            opt: [sub bias frame]
            opt: [divide by FF]
            opt: [mask-out defects]

```

**produce\_NLCs()****METACODE**

```

Obtains Best-Fit Non-Linearity Curve

f.e. CCD:
    f.e. Q:

```

```

    [opt] apply correction for source variability (interspersed_
↪exposure
    with constant exptime)
    Build NL Curve (NLC) - use stats and exptimes
    fit poly. shape to NL curve

    plot NL curves for each CCD, Q
    report max. values of NL (table)

```

**class** Pipe.**PERSIST01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis**()

Basic analysis of data.

**METACODE**

```

f.e.CCD:
  f.e.Q:
    use SATURATED frame to generate pixel saturation MASK
    measure stats in pix satur MASK across OBSIDs
    (pre-satur, satur, post-satur)

```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds PERSISTENCE01 script structure dictionary.

**Parameters**

- **exptSATUR** – int, saturation exposure time.
- **exptLATEN** – int, latency exposure time.
- **diffvalues** – dict, opt, differential values.

**check\_data**()

PERSIST01: Checks quality of ingested data.

**METACODE**

```

check common HK values are within safe / nominal margins
check voltages in HK match commanded voltages, within margins

f.e.ObsID:
  f.e.CCD:
    f.e.Q.:
      measure offsets in pre-, over-
      measure std in pre-, over-
      measure fluence in apertures around Point Sources

    assess std in pre- (~RON) is within allocated margins
    assess offsets in pre-, and over- are equal, within allocated margins
    assess fluence is ~expected within apertures (PS) for each frame (pre-
↪satur, satur, post-satur)

    plot point source fluence vs. OBSID, all sources
    [plot std vs. time]

    issue any warnings to log
    issue update to report

```

**filterexposures** (*structure, explog, OBSID\_lims*)



**meta\_analysis()**

Meta-analysis of data.

**METACODE**

```
f.e.CCD:
  f.e.Q:
    estimate delta-charge_0 and decay tau from time-series

report:
  persistence level (delta-charge_0) and time constant
```

**prep\_data()**

PERSIST01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**set\_inpdefaults(\*\*kwargs)**

**class** Pipe.**PTCOX** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds PTCOX script structure dictionary.

*#:param* exptimes: list of ints [ms], exposure times. *#:param* frames: list of ints, number of frames for each exposure time. *#:param* wavelength: int, wavelength. Default: 800 nm. *:param* diffvalues: dict, opt, differential values.

**extract\_PTC()**

**Performs basic analysis of images:**

- builds PTC curves: both on non-binned and binned images

**METACODE**

```
create list of OBSID pairs

create segmentation map given grid parameters

f.e. OBSID pair:
  CCD:
    Q:
      subtract CCD images
  f.e. segment:
    measure central value
    measure variance
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

Analyzes the variance and fluence: gain, and gain(fluence)

**METACODE**

```
f.e. CCD:
  Q:
    (using stats across segments:)
    fit PTC to quadratic model
    solve for gain
    solve for alpha (pixel-correls, Guyonnet+15)
    solve for blooming limit (ADU)
    convert bloom limit to electrons, using gain
```

```
plot PTC curves with best-fit f.e. CCD, Q
report on gain estimates f. e. CCD, Q (table)
report on blooming limits (table)
```

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**STRAY00** (inputs, log=None, drill=False, debug=False)

**build\_scriptdict** (diffvalues={}, elvis='6.5.X')

Builds STRAY00 script structure dictionary. :param diffvalues: dict, opt, differential values.

**filterexposures** (structure, explog, OBSID\_lims)

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**TP00** (inputs, log=None, drill=False, debug=False)

**build\_scriptdict** (diffvalues={}, elvis='6.5.X')

**check\_data** ()

TP01: Checks quality of ingested data.

**METACODE**

```
check common HK values are within safe / nominal margins
check voltages in HK match commanded voltages, within margins

f.e.ObsID:
  f.e.CCD:
    f.e.Q.:
      measure offsets in pre-, over-
      measure std in pre-, over-
      measure mean in img-

assess std in pre- (~RON) is within allocated margins
assess offsets in pre-, and over- are equal, within allocated margins
assess offsets are within allocated margins
assess injection level is within expected margins

plot histogram of injected levels for each Q
[plot std vs. time]

issue any warnings to log
issue update to report
```

**filterexposures** (structure, explog, OBSID\_lims)

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**TP01** (inputs, log=None, drill=False, debug=False)

**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```
f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
```

```

f.e.Q:
  load "map of relative pumping"
  find_dipoles:
    x, y, rel-amplitude, orientation

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for N and S)
  Counts of dipoles (and N vs. S)

```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

**extract** ()

Obtain maps of dipoles.

**METACODE**

```

f.e. id_delay (there are 2):
  f.e. CCD:
    f.e. Q:
      produce reference non-pumped injection map

f. e. ObsID:
  f.e. CCD:

  load ccdobj
  f.e.Q.:
    divide ccdobj.Q by injection map

  save dipole map and store reference

```

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

**METACODE**

```

across TOI_TP, patterns:
  build catalog of traps: x,y,I-phase, Amp
  from Amp (TOI) -> tau, Pc

Report on :
  Histogram of Taus
  Histogram of Pc (capture probability)
  Histogram of I-phases (larger phases should have more traps,
    statistically) -> check

  Total Count of Traps

```

**set\_inpdefaults** (\*\*kwargs)

**class** Pipe.**TP02** (*inputs*, *log=None*, *drill=False*, *debug=False*)

**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```
f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
    f.e.Q:
      load raw 1D map of relative pumping (from extract_data)
      identify dipoles:
        x, rel-amplitude, orientation (E or W)

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for E and W)
  Counts of dipoles (and E vs. W)
```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

**extract** ()

Obtain Maps of Serial Dipoles.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

**METACODE**

```
across TOI_TP, patterns:
  build catalog of traps: x,y,R-phase, amp(dwell)
  from Amp(dwell) -> tau, Pc

Report on :
  Histogram of Taus
  Histogram of Pc (capture probability)
  Histogram of R-phases

Total Count of Traps
```

**set\_inpdefaults** (\*\**kwargs*)

Pipe.**catchtraceback** ()

Pipe.**dotask** (*taskname*, *inputs*, *drill=False*, *debug=False*)

Generic test master function.

Pipe.**get\_execution\_summary** (*exectime=None*)

Pipe.**get\_test** (*taskname*, *inputs={}*, *log=None*, *drill=False*, *debug=False*)

Pipe.**launchtask** (*taskname*)

Pipe.**run** (*explog=None*, *elvis=None*)

Pipe.**wait\_and\_run** (*dayfolder*, *elvis='6.5.X'*)

### 3.1.2 task.py

Generic Task (Test) Class.

Created on Tue Nov 14 14:20:04 2017

**author** Ruyman Azzollini

```

class vison.pipe.task.Task (inputs, log=None, drill=False, debug=False)

    IsComplianceMatrixOK (complidict)
    addComplianceMatrix2Log (complidict, label='')
    addComplianceMatrix2Report (complidict, label='', caption='')
    addFigure2Report (figkey)
    addFigures_ST (dobuilddata=True, **kwargs)
    addFlagsToLog ()
    addFlagsToReport ()
    addHKPlotsMatrix ()
        Adds to self.report a table-figure with HK [self.HKKeys] during test.
    addHK_2_dd ()
    add_data_inventory_to_report (tDict)
    add_inputs_to_report ()
    add_labels_to_explog (explog, structure)
    build_scriptdict (diffvalues={}, elvis='6.5.X')
    catchtraceback ()
    check_HK (HKKeys, reference='command', limits='P', tag='', doReport=False, doLog=True)
    check_HK_ST ()
    check_data (**kwargs)
        Generic check_data method
    check_stat_perCCD (arr, CCDlims, CCDs=['CCD1', 'CCD2', 'CCD3'])
    check_stat_perCCDQandCol (arr, lims, CCDs=['CCD1', 'CCD2', 'CCD3'])
    check_stat_perCCDandCol (arr, lims, CCDs=['CCD1', 'CCD2', 'CCD3'])
    check_stat_perCCDandQ (arr, CCDQlims, CCDs=['CCD1', 'CCD2', 'CCD3'])
    create_mockexplog (OBSID0=1000)
    doPlot (figkey, **kwargs)
    filterexposures (structure, explog, OBSID_lim, colorblind=False, wavedkeys=[], surrogate='')
        Loads a list of Exposure Logs and selects exposures from test 'test'.

        The filtering takes into account an expected structure for the acquisition script.

        The datapath becomes another column in DataDict. This helps dealing with tests that run overnight and
        for which the input data is in several date-folders.
    prepare_images (doExtract=True, doMask=False, doOffset=False, doBias=False, doFF=False)
    recover_progress (DataDictFile, reportobjFile)
    save_progress (DataDictFile, reportobjFile)
    skipMissingPlot (key, ref)
class vison.pipe.task.Task (inputs, log=None, drill=False, debug=False)

```

```

IsComplianceMatrixOK (complidict)
addComplianceMatrix2Log (complidict, label='')
addComplianceMatrix2Report (complidict, label='', caption='')
addFigure2Report (figkey)
addFigures_ST (dobuilddata=True, **kwargs)
addFlagsToLog ()
addFlagsToReport ()
addHKPlotsMatrix ()
    Adds to self.report a table-figure with HK [self.HKKeys] during test.
addHK_2_dd ()
add_data_inventory_to_report (tDict)
add_inputs_to_report ()
add_labels_to_explog (explog, structure)
build_scriptdict (diffvalues={}, elvis='6.5.X')
catchtraceback ()
check_HK (HKKeys, reference='command', limits='P', tag='', doReport=False, doLog=True)
check_HK_ST ()
check_data (**kwargs)
    Generic check_data method
check_stat_perCCD (arr, CCDlims, CCDs=['CCD1', 'CCD2', 'CCD3'])
check_stat_perCCDQandCol (arr, lims, CCDs=['CCD1', 'CCD2', 'CCD3'])
check_stat_perCCDandCol (arr, lims, CCDs=['CCD1', 'CCD2', 'CCD3'])
check_stat_perCCDandQ (arr, CCDQlims, CCDs=['CCD1', 'CCD2', 'CCD3'])
create_mockexplog (OBSID0=1000)
doPlot (figkey, **kwargs)
filterexposures (structure, explog, OBSID_lims, colorblind=False, wavedkeys=[], surrogate='')
    Loads a list of Exposure Logs and selects exposures from test 'test'.

    The filtering takes into account an expected structure for the acquisition script.

    The datapath becomes another column in DataDict. This helps dealing with tests that run overnight and
    for which the input data is in several date-folders.
prepare_images (doExtract=True, doMask=False, doOffset=False, doBias=False, doFF=False)
recover_progress (DataDictFile, reportobjFile)
save_progress (DataDictFile, reportobjFile)
skipMissingPlot (key, ref)

```

## DATA MODEL

Modules with classes to hold data model for inputs and outputs: exposure log, HK files, FITS files, etc.

### 4.1 Data Model

#### 4.1.1 ccd.py

Data model for Euclid-VIS CCDs (ground testing at MSSL)

Created on Fri Nov 13 17:42:36 2015

**author** Ruyman Azzollini

**class** `vison.datamodel.ccd.CCD` (*infits=None, extensions=[-1], getallextensions=False, withpover=True*)

Class of CCD objects. Euclid Images as acquired by ELVIS software (Euclid LabView Imaging Software).

The class has been extended to handle multi-extension images. This is useful to also “host” calibration data-products, such as Flat-Fields.

#### A note on Coordinates Systems:

- ‘CCD’: referenced to the first pixel readout from channel H. All 4 quadrants

in a single array, their detection nodes in the 4 “corners” of the rectangle. Same system as images are displayed on DS9. In clock-wise sense, quadrants are H (bottom-left), E (top-left), F (top-right), and G (bottom-right). - ‘Quadrant-canonical’: Quadrant coordinates system in which the first pixel is the first pixel read out (closest pixel to the readout node), and the last is the last readout. In this system, the serial pre-scan comes before the image area, and this before the serial overscan. Parallel overscan comes after image area in the parallel direction. In this system, coordinates of pixels across quadrants, for a single readout, correspond to the same point in time. Useful when doing cross-talk analysis, for example. - ‘Quadrant-relative’: quadrant coordinates system with the same relative orientation as in the ‘CCD’ system, but referenced to the ‘lower-left’ pixel of the given quadrant in such system. In this system, the readout node is in a different corner for each quadrant: lower-left for H, top-left for E, top-right for F and bottom-right for G.

**add\_extension** (*data, header=None, label=None, headerdict=None*)

Appends an extension to self (extensions are in a list).

**add\_to\_hist** (*action, extension=-1, vison=u‘0.6+74.gf031d8f’, params={}*)

**cooconvert** (*x, y, insys, outsys, Q=‘U’*)

Coordinates conversion between different systems.

**del\_extension** (*extension*)

Deletes an extension from self, by index.

**divide\_by\_flatfield** (*FF*, *extension=-1*)

Divides by a Flat-field

**do\_Vscan\_Mask** (*VSTART*, *VEND*)

**dummyrebin** (*arr*, *new\_shape*, *stat='median'*)

**extract\_region** (*ccdobj*, *Q*, *area='img'*, *vstart=0*, *vend=2086*, *Full=False*, *canonical=True*, *extension=-1*)

**get\_1Dprofile** (*ccdobj*, *Q*, *orient='hor'*, *area='img'*, *stacker='mean'*, *vstart=0*, *vend=2086*, *extension=-1*)

**get\_Q** (*x*, *y*, *w*, *h*)

**get\_cutout** (*corners*, *Quadrant*, *canonical=False*, *extension=-1*)

Returns a cutout from the CCD image, either in canonical or non-canonical orientation.

#### Parameters

- **corners** (*list (of int)*) – [x0,x1,y0,y1]
- **Quadrant** (*char*) – Quadrant, one of 'E', 'F', 'G', 'H'
- **canonical** (*bool*) – Canonical [True] = with readout-node at pixel index (0,0) regardless of quadrant. This is the orientation which corresponds to the data-readin order (useful for cross-talk measurements, for example). Non-Canonical [False] = with readout-node at corner matching placement of quadrant on the CCD. This is the orientation that would match the representation of the image on DS9.
- **extension** (*int*) – extension number. Default = -1 (last)

**get\_mask** (*mask*)

**get\_quad** (*Quadrant*, *canonical=False*, *extension=-1*)

Returns a quadrant in canonical or non-canonical orientation.

#### Parameters

- **Quadrant** (*char*) – Quadrant, one of 'E', 'F', 'G', 'H'
- **canonical** –

Canonical [True] = with readout-node at pixel index (0,0) regardless of quadrant. This is the orientation which corresponds to the data-reading order (useful for cross-talk measurements, for example). Non-Canonical [False] = with readout-node at corner matching placement of quadrant on the CCD. This is the orientation that would match the representation of the image on DS9.

**Parameters extension** (*int*) – extension number. Default = -1 (last)

**get\_region2Dmodel** (*ccdobj*, *Q*, *area='img'*, *kind='spline'*, *splinemethod='cubic'*, *pdegree=2*, *doFilter=False*, *doBin=True*, *filtsize=1*, *binsize=1*, *filtertype='mean'*, *vstart=0*, *vend=2086*, *canonical=True*, *extension=-1*)

**get\_stats** (*Quadrant*, *sector='img'*, *statkeys=['mean']*, *trimscan=[0, 0]*, *ignore\_pover=True*, *extension=-1*, *VSTART=0*, *VEND=2086*)

**get\_tile\_coos** (*Quadrant*, *wpx*, *hpx*)

Returns a dictionary with a tiling [coordinates of corners of tiles] of quadrant Q, with tiles of size wpx[width] x hpx[height].

CAUTION: Returned coordinates are Q-relative.

#### Parameters

- **Quadrant** – str, Quadrant, one of ['E','F','G','H']



- **wpx** – int, width [along NAXIS1] of tiles, in pixels.
- **hpx** – int, height [along NAXIS2] of tiles, in pixels.

**Returns** `tiles_dict = dict( wpx='Width of tiles, integer', hpx='Height of tiles, integer', llpix='Lower left corner of tiles, list of tuples', ccpix= 'Central pixel of tiles, list of tuples', Nsamps='Number of tiles, integer')`

**get\_tiles** (*Quadrant, tile\_coos, extension=-1*)

**get\_tiles\_stats** (*Quad, tile\_coos, statkey, extension=-1*)

**getsectioncollims** (*Q*)

Returns limits of [HORIZONTAL] sections: prescan, image and overscan

**getsectionrowlims** (*Q*)

Returns limits of [VERTICAL] sections: image [and vertical overscan]

**loadfromFITS** (*fitsfile, extensions=[-1], getallextensions=False*)

Loads contents of self from a FITS file.

**set\_extension** (*data, header=None, label=None, headerdict=None, extension=-1*)

Sets extension 'extension' in self.

**set\_quad** (*inQdata, Quadrant, canonical=False, extension=-1*)

**sim\_window** (*ccdobj, vstart, vend, extension=-1*)

**simadd\_flatilim** (*ccdobj, levels=None, extension=-1*)

**simadd\_points** (*ccdobj, flux, fwhm, CCDID='CCD1', dx=0, dy=0, extension=-1*)

**simadd\_poisson** (*ccdobj, extension=-1*)

**simadd\_ron** (*ccdobj, extension=-1*)

**sub\_bias** (*superbias, extension=-1*)

Subtracts a superbias

**sub\_offset** (*Quad, method='row', scan='pre', trimscan=[3, 2], ignore\_pover=True, extension=-1*)

**writeto** (*fitsf, clobber=False, unsigned16bit=False*)

**class** `vison.datamodel.ccd.CCDPile` (*infitsList=[], ccdobjList=[], extension=-1, withpover=True*)

Class to hold and operate (e.g. stack) on a bunch of CCD images. Each image (a single extension picked from each) becomes an extension in the pile.

**stack** (*method='median', dostd=False*)

**class** `vison.datamodel.ccd.Extension` (*data, header=None, label=None, headerdict=None*)

Extension Class

`vison.datamodel.ccd.cooconv_arrays_decorate` (*func*)

`vison.datamodel.ccd.test_create_from_scratch` ()

`vison.datamodel.ccd.test_load_ELVIS_fits` ()

## 4.1.2 ccd\_aux.py

Auxiliary script to ccd.py

Created on Mon Feb 19 13:14:02 2018

**author** raf

```

class vison.datamodel.ccd_aux.Model2D (img, corners=[])
    Class for 2D models of images and images sections.

    bin_img (boxsize, stat='median')

    filter_img (filtsize=15, filtertype='median', Tests=False)

    fit2Dpol_xyz (xx, yy, zz, degree=1)

    get_model_poly2D (sampling=1, pdegree=5, useBin=False)

    get_model_splines (sampling=1, splinemethod='cubic', useBin=False)

class vison.datamodel.ccd_aux.Profile1D (x, y)
    Class for 1D profiles of images and images sections.

vison.datamodel.ccd_aux.extract_region (ccdobj, Q, area='img', vstart=0, vend=2086,
                                         Full=False, canonical=True, extension=-1)

vison.datamodel.ccd_aux.get_1Dprofile (ccdobj, Q, orient='hor', area='img', stacker='mean',
                                         vstart=0, vend=2086, extension=-1)

vison.datamodel.ccd_aux.get_region2Dmodel (ccdobj, Q, area='img', kind='spline',
                                             splinemethod='cubic', pdegree=2, doFilter=False, doBin=True,
                                             filtsize=1, binsize=1, filtertype='mean', vstart=0, vend=2086,
                                             canonical=True, extension=-1)

vison.datamodel.ccd_aux.rebin (arr, new_shape, stat='mean')
    "Rebin 2D array arr to shape new_shape by averaging.

```

### 4.1.3 ccdsim.py

Methods to simulate data. Used by ccd.CCD class.

Created on Wed Apr 4 11:13:30 2018

**author** Ruyman Azzollini

```

vison.datamodel.ccdsim.sim_window (ccdobj, vstart, vend, extension=-1)

vison.datamodel.ccdsim.simadd_flatillum (ccdobj, levels=None, extension=-1)

vison.datamodel.ccdsim.simadd_points (ccdobj, flux, fwhm, CCDID='CCDI', dx=0, dy=0,
                                         extension=-1)

vison.datamodel.ccdsim.simadd_poisson (ccdobj, extension=-1)

vison.datamodel.ccdsim.simadd_ron (ccdobj, extension=-1)

```

### 4.1.4 compliance.py

Some functions to present COMPLIANCE MATRICES.

Created on Mon Apr 9 17:32:03 2018

**author** raf

```

vison.datamodel.compliance.convert_compl_to_nesteditelist (complidict)

vison.datamodel.compliance.gen_compliance_tex (indict, escape=True, caption='')

vison.datamodel.compliance.removescalars_from_dict (indict)

```

### 4.1.5 core.py

DataDict Class : holds data and results across sub-tasks of a “task” (Test). This is the CORE data-structure used to do analysis and report results.

Created on Thu Sep 21 16:47:09 2017

**author** Ruyman Azzollini

```
class vison.datamodel.core.DataDict (meta={})

    addColumn (array, name, indices)
    col_has_index (colname, indexname)
    dropColumn (colname)
    initColumn (name, indices, dtype='float32', valini=0.0)
    loadExpLog (explog)
    name_indices ()
    saveToFile (outfile, format='ascii.commented_header')

vison.datamodel.core.useCases ()
#TODO:
    # create a DataDict object from an exposure log. # add a column indexed by ObsID, CCD and Quad
    # drop a column # create a column from an operation on several columns with different dimensions
    # save to a text / excel file # save to a pickle file

class vison.datamodel.core.vColumn (array, name, indices)

    name_indices ()

class vison.datamodel.core.vIndex (name, vals=[], N=0)
```

### 4.1.6 EXPLOGtools.py

```
class vison.datamodel.EXPLOGtools.ExpLogClass (elvis='6.5.X')

    addRow (row)
    iniExplog ()
    summary ()
    writeto (outfile)

vison.datamodel.EXPLOGtools.iniExplog (elvis)
vison.datamodel.EXPLOGtools.loadExpLog (expfile, elvis='6.5.X')
    Loads an Exposure Log from file.

vison.datamodel.EXPLOGtools.mergeExpLogs (explogList, addpedigree=False, verbose=False)
    Merges explog objects in a list.

vison.datamodel.EXPLOGtools.test ()
    This Tests needs UPDATE (for data access and probably data format)
```

### 4.1.7 generator.py

Script to generate simulated data for pipeline testing purposes.

Created on Tue Aug 29 11:08:56 2017

**author** Ruyman Azzollini

```
vison.datamodel.generator.IMG_bias_gen(ccdobj, ELdict, ogse=None)
vison.datamodel.generator.IMG_chinj_gen(ccdobj, ELdict, ogse=None)
vison.datamodel.generator.IMG_flat_gen(ccdobj, ELdict, ogse=None)
vison.datamodel.generator.IMG_point_gen(ccdobj, ELdict, ogse=None)
vison.datamodel.generator.generate_Explog(scrdict, defaults, elvis='6.5.X', explog=None,
                                           OBSID=1000, date=datetime.datetime(1980,
                                           2, 21, 7, 0), CHAMBER=None)
```

Generates a fake ExposureLog from a test structure dictionary.

DEVELOPMENT NOTES:

**To be generated: (EASY)** \*ObsID, \*File\_name, \*CCD, \*ROE=ROE1, \*DATE, \*BUNIT=ADU, SPW\_clk=0?,EGSE\_ver=elvis,

**Temporal:** SerRdDel

**To be provided in defaults: (EASY)** Lab\_ver,Con\_file,CnvStart, Flsh-Rdout\_e\_time,C.Inj-Rdout\_e\_time, FPGA\_ver,Chmb\_pre,RICCD[1,2,3]T[T,B]

To be read/parsed/processed from struct: (DIFFICULT)

SerRDel?,SumWell?, IniSweep?,+etc.

```
vison.datamodel.generator.generate_FITS(ELdict, funct, filename='', elvis='6.5.X',
                                         ogse=None)
vison.datamodel.generator.generate_FITS_fromExpLog(explog, datapath, elvis='6.5.X',
                                                    CHAMBER=None)
vison.datamodel.generator.generate_HK(explog, vals, datapath='', elvis='6.5.X')
vison.datamodel.generator.merge_HKfiles(HKfiles, masterHKf)
```

### 4.1.8 HKtools.py

House-Keeping inspection and handling tools.

**History**

Created on Thu Mar 10 12:11:58 2016

**author** Ruyman Azzollini

```
vison.datamodel.HKtools.check_HK_abs(HKKeys, dd, limits='S', elvis='6.5.X')
```

Returns report on HK parameters, in DataDict (dd), compared to absolute limits.

HK Keys which have “relative” limits, always return False.

**Parameters**

- **HKKeys** – list of HK parameters, as named in HK files (without **HK\_** suffix)
- **dd** – DataDict object
- **limits** – type of limits to use, either “P” (Performance) or “S” (Safe)

- **elvis** – ELVIS version to find correspondence between HK key and Exposure Log input (commanded voltage).

**Returns report** dictionary with pairs of HK-key : Bool. True = All values for given key are within limits. False = At least one value for given key is outside limits.

`vison.datamodel.HKtools.check_HK_vs_command(HKKeys, dd, limits='P', elvis='6.5.X')`  
Returns report on HK parameters, in DataDict (dd), comparing inputs (commanded) vs. output (HK data).

HK Keys which do not correspond to commanded voltages always return 'True'.

#### Parameters

- **HKKeys** – list of HK parameters, as named in HK files (without **HK\_** suffix)
- **dd** – DataDict object
- **limits** – type of limits to use, either “P” (Performance) or “S” (Safe)
- **elvis** – ELVIS version to find correspondence between HK key and Exposure Log input (commanded voltage).

**Returns report** dictionary with pairs of HK-key : Bool. True = All values are within limits, referred to commanded value. False = At least one value is outside limits, referred to commanded value.

`vison.datamodel.HKtools.doHKSinglePlot(dtobjs, HK, HKkey, ylabel='V', HKlims=[], filename='', fontsize=10)`

Plots the values of a HK parameter as a function of time.

#### Parameters

- **dtobjs** – datetime objects time axis.
- **HK** – HK values (array)
- **HKkey** –
- **ylabel** –
- **HKlims** –
- **filename** – file-name to store plot [empty string not to save].

**Returns** None!!

`vison.datamodel.HKtools.filtervalues(values, key)`

`vison.datamodel.HKtools.iniHK_QFM(elvis='6.5.X', length=0)`

`vison.datamodel.HKtools.loadHK_QFM(filename, elvis='6.5.X', validate=False)`

Loads a HK file, or list of HK files.

Structure: astropy table. First column is a timestamp, and there may be a variable number of rows (readings).

#### Parameters

- **filename** – path to the file to be loaded, including the file itself, or list of paths to HK files.
- **elvis** – “ELVIS” version

**Returns** astropy table with pairs parameter:[values]

`vison.datamodel.HKtools.loadHK_QFMsingle(filename, elvis='6.5.X', validate=False)`

Loads a HK file

Structure: tab separated columns, one per Keyword. First column is a timestamp, and there may be a variable number of rows (readings).

**Parameters**

- **filename** – path to the file to be loaded, including the file itself
- **elvis** – “ELVIS” version

**Returns** astropy table with pairs parameter:[values]

`vison.datamodel.HKtools.loadHK_preQM(filename, elvis='5.7.07')`  
 Loads a HK file

It only assumes a structure given by a HK keyword followed by a number of of tab-separated values (number not specified). Note that the length of the values arrays is variable (depends on length of exposure and HK sampling rate).

**Parameters filename** – path to the file to be loaded, including the file itself

**Returns** dictionary with pairs parameter:[values]

`vison.datamodel.HKtools.mergeHK(HKList)`

`vison.datamodel.HKtools.parseDTstr(DTstr)`

`vison.datamodel.HKtools.parseHKfiles(HKlist, elvis='6.5.X')`

**Parameters**

- **HKlist** – list of HK files (path+name).
- **elvis** – “ELVIS” version.

**Returns** [obsids],[dtobj],[tdeltasec],[HK\_keys], [data(nfiles,nstats,nHKparams)]

`vison.datamodel.HKtools.parseHKfname(HKfname)`

Parses name of a HK file to retrieve OBSID, date and time, and ROE number.

**Parameters HKfname** – name of HK file.

**Returns** obsid,dtobj=datetime.datetime(yy,MM,dd,hh,mm,ss),ROE

`vison.datamodel.HKtools.reportHK(HKs, key, reqstat='all')`

Returns (mean, std, min, max) for each keyword in a list of HK dictionaries (output from loadHK).

**Parameters**

- **HK** – dictionary with HK data.
- **key** – HK key.

**Reqstat** what statistic to retrieve.

`vison.datamodel.HKtools.synthHK(HK)`

Synthesizes the values for each parameter in a HK dictionary into [mean,std,min,max].

**Parameters HK** – a dictionary as those output by loadHK.

**Returns** dictionary with pairs parameter:[mean,std,min,max]

## 4.1.9 inputs.py

Inputs Handling Classes and utilities.

Created on Thu Jan 11 10:34:43 2018

**author** Ruyman Azzollini

```
class vison.datamodel.inputs.Inputs (*args, **kwargs)
    Class to hold, transfer and 'document' Task Inputs.
```

### 4.1.10 QLAtools.py

Quick-Look-Analysis Tools.

#### History

Created on Wed Mar 16 11:31:58 2016

@author: Ruyman Azzollini

```
vison.datamodel.QLAtools.dissectFITS (FITSfile, path='')
vison.datamodel.QLAtools.getacrosscolscut (CCDObj)
vison.datamodel.QLAtools.getacrossrowscut (CCDObj)
vison.datamodel.QLAtools.getsectionstats (CCDObj, QUAD, section, xbuffer=(0, 0),
                                          ybuffer=(0, 0))
vison.datamodel.QLAtools.plotAcCOLcuts (dissection, filename=None, suptitle='')
vison.datamodel.QLAtools.plotAcROWcuts (dissection, filename=None, suptitle='')
vison.datamodel.QLAtools.plotQuads (CCDObj, filename=None, suptitle='')
vison.datamodel.QLAtools.reportFITS (FITSfile, outpath='')
```

### 4.1.11 scriptic.py

Classes and functions to generate ELVIS commanding scripts automatically.

Created on Wed May 24 15:31:54 2017

**author** Ruyman Azzollini

```
class vison.datamodel.scriptic.Script (defaults={}, structure={}, elvis='6.5.X')
    Core Class that provides automatic test script generation and validation.
```

```
build_cargo ()
```

Updates 'cargo' attribute. 'cargo': list of lists, each corresponding to a column in the script.

Each element in the inner lists is a register value. The first column corresponds to the column with key names.

Note: the number of frames is accumulated across columns, as ELVIS expects.

```
get_struct_from_cargo ()
```

```
load (*args, **kwargs)
```

alias method. Points to 'load\_to\_cargo'.

```
load_to_cargo (scriptname, elvis='6.5.X')
```

Loads an script from an excel file.

#### Parameters

- **scriptname** – char, script to load
- **elvis** – char, ELVIS version of script to load

**validate** (*defaults, structure, elvis='6.5.X'*)

Not sure 'validation' will work like as implemented... TODO: validate self.validate

**write** (*scriptname*)

Writes self.cargo (script) to an excel file.

**Parameters** **scriptname** – char, name of file where to write script.

`vison.datamodel.scriptic.test0()`

`vison.datamodel.scriptic.update_structdict (sdict, commvalues, diffvalues)`

Updates an script structure with common values and differential values.

**Parameters**

- **sdict** – dict, dictionary with script structure. Takes precedence over commvalues.
- **commvalues** – dict, dictionary with common values to update sdict.
- **diffvalues** – dict, dictionary with “differential” values to update “sdict”. Takes precedence over sdict and commvalues.



## ANALYSIS (SHARED)

### 5.1 Analysis (Shared)

#### 5.1.1 ellipse.py

Auxiliary module with functions to generate generalized ellipse masks.

**author** Ruyman Azzollini

**class** `vison.analysis.ellipse.TestEllipse` (*methodName='runTest'*)  
Unit tests for the ellipse module.

`vison.analysis.ellipse.area_superellip` (*r, q, c=0*)  
Returns area of superellipse, given the semi-major axis length

`vison.analysis.ellipse.dist_superellipse` (*n, center, q=1.0, pos\_ang=0.0, c=0.0*)  
Form an array in which the value of each element is equal to the semi-major axis of the superellipse of specified center, axial ratio, position angle, and c parameter which passes through that element. Useful for super-elliptical aperture photometry.

Inspired on `dist_ellipse.pro` from AstroLib (IDL).

Note: this program doesn't take into account the change in the order of axes from IDL to Python. That means, that in 'n' and 'center', the order of the coordinates must be reversed with respect to the case for `dist_ellipse.pro`, in order to get expected results. Nonetheless, the polar angle means the counter-clock wise angle with respect to the 'y' axis.

#### Parameters

- **n** – shape of array (N1,N2), it can be an integer (squared shape NxN)
- **center** – center of superellipse radii: (c1,c2)
- **q** – axis ratio r2/r1
- **pos\_ang** – position angle of isophotes, in degrees, CCW from axis 1
- **c** – boxyness (c>0) /diskyness (c<0)

`vison.analysis.ellipse.effective_radius` (*area, q=1.0, c=0.0*)  
Returns semi-major axis length of superellipse, given the area

#### 5.1.2 Guyonnet15.py

Library with functions that implement the algorithms described in Guyonnet+15. "Evidence for self-interaction of charge distribution in CCDs" Guyonnet, Astier, Antilogus, Regnault and Doherty 2015

Notes:

- I renamed “x” (pixel boundary index) to “b”, to avoid confusion with cartesian “x”.
- In paper, X belongsto [(0,1),(1,0),(0,-1),(-1,0)]. Here b is referred to as cardinal points “N”,“E”,“S”,“W”. It is linked to matrix index ib, running between 0 and 3.

Created on Thu Sep 22 11:38:24 2016

**author** Ruyman Azzollini

`vison.analysis.Guyonnet15.correct_estatic` (*img*, *aijb*)

Corrects an image from pixel-boundaries deformation due to electrostatic forces. Subtracts delta-Q.

**Parameters**

- **img** – image, 2D array
- **aijb** – Aijb matrix, 3D array

**Returns** array, img - delta-Q

`vison.analysis.Guyonnet15.degrade_estatic` (*img*, *aijb*)

Degrades an image according to matrix of pixel-boundaries deformations. Follows on Eq. 11 of G15. Adds delta-Q.

**Parameters**

- **img** – image, 2D array
- **aijb** – Aijb matrix, 3D array

**Returns** array, img + delta-Q

`vison.analysis.Guyonnet15.fpred_aijb` (*p*, *i*, *j*, *ib*)

‘The smoothing model assumes that  $a_{ij}^x$  coefficients are the product of a function of distance from the source charge to the considered boundary ( $r_{ij}$ ) and that it also trivially depends on the angle between the source-boundary vector and the normal to the boundary ( $\theta_{ij}^x$ )’

Eq. 18

**Parameters**

- **p** – parameters of the radial function (list of 2)
- **i** – pixel coordinate i
- **j** – pixel coordinate j
- **ib** – boundary index [0, 1, 2, 3]

**Returns**  $f(r_{ij})\cos(\theta_{ij}^x)$

`vison.analysis.Guyonnet15.frdist` (*i*, *j*, *ib*)

Distance from the source charge to considered boundary “b”

**Parameters**

- **i** – pixel coordinate i
- **j** – pixel coordinate j
- **ib** – boundary index [0, 1, 2, 3]

**Returns** distance  $r(ijb)$

`vison.analysis.Guyonnet15.ftheta_bound` (*i*, *j*, *ib*)

“ $\theta_{ij}^x$  is] the angle between the source-boundary vector and the normal to the boundary”.

**Parameters**

- **i** – pixel coordinate i
- **j** – pixel coordinate j
- **ib** – boundary index [0, 1, 2, 3]

**Returns**  $\theta_{i,j}^x$

`vison.analysis.Guyonnet15.fun_p(x, *p)`  
 auxiliary function to 'solve\_for\_psmooth'

`vison.analysis.Guyonnet15.generate_GaussPSF(N, sigma)`  
 Create a circular symmetric Gaussian centered on the centre of a NxN matrix/image.

`vison.analysis.Guyonnet15.get_Rdisp(img, aijb)`  
 Retrieves map of relative displacements of pixel boundaries, for input img and Aijb matrix.  
 See G15 - Eq. 6

**Parameters**

- **img** – image, 2D array
- **aijb** – aijb matrix, 3D array NxNx4

**Returns** array, relative displacements all boundaries of pixels in img

`vison.analysis.Guyonnet15.get_cross_shape_rough(cross, pitch=12.0)`

`vison.analysis.Guyonnet15.get_deltaQ(img, aijb, writeFits=False)`  
 Retrieves deltaQ map for input image and aijb matrix.

See G15 - Eq. 11

**Parameters**

- **img** – image, 2D array
- **aijb** – Aijb matrix, 3D array
- **writeFits** – save FITS file with resulting dQ map (optional)

**Returns** array, matrix with delta-Q for each pixel in img, given aijb

`vison.analysis.Guyonnet15.get_kernel(aijb)`  
 'kernel' is an array (2N-1)x(2N-1)x4. Each plane kernel[:,b] is a 2D array with the displacement coefficients aijb, in all directions around a pixel at (0,0).

**Parameters**

- **aijb** – array, matrix with displacements in 1st quadrant
- **writeFits** – save kernel to 4 FITS files

**Returns** kernel matrix, (2N-1)x(2N-1)x4

`vison.analysis.Guyonnet15.plot_map(z, ii, jj, title='')`

`vison.analysis.Guyonnet15.plot_maps_ftheta(f, ii, jj, suptitle='')`

`vison.analysis.Guyonnet15.show_disps_CCD273(aijb, stretch=5.0, peak=28571.428571428572, N=25, sigma=1.6, title='', figname='')`

`vison.analysis.Guyonnet15.solve_for_A_linalg(covij, var=1.0, mu=1.0, doplot=False, psmooth=None, returnAll=False)`

Function to retrieve the A matrix of pixel boundaries displacements, given a matrix of pixel covariances, variance, and mu.

if `var==1` and `mu==1`, it is understood that `covij` is the correlation matrix.

See section 6.1 of G15.

#### Parameters

- **covij** – array, squared matrix with pixel covariances.
- **var** – float, variance of the flat-field.
- **mu** – float, mean value of the flat-field.
- **doplot** – if True, plot the fit of the `fpred(ijb)` function
- **psmooth** – coefficients of the `fpred(aijb)` function (Eq. 18)
- **returnAll** – bool, controls return values

**Returns** if `returnAll == True`, return `(aijb, psmooth)`, otherwise return `aijb` only

`vison.analysis.Guyonnet15.solve_for_psmooth(covij, var, mu, doplot=False)`

Solving  $(p_0, p_1)$  parameters in Eq. 18 using covariance matrix and measured covariance matrix.

#### Parameters

- **covij** – array, covariance matrix
- **var** – float, variance
- **mu** – float, expected value of pixel values (“mean” of flat-field)
- **doplot** – bool, if True, plot data and best fit model

**Returns** best-fit parameters, and errors: 2 tuples of 2 elements each

`vison.analysis.Guyonnet15.test0()`

`vison.analysis.Guyonnet15.test_getkernel()`

`vison.analysis.Guyonnet15.test_selfconsist()`

`vison.analysis.Guyonnet15.test_solve()`

## CHARGE INJECTION TOOLS

### 6.1 Charge Injection Tools

#### 6.1.1 InjTask.py

Created on Wed Dec 6 15:56:00 2017

**author** Ruyman Azzollini

**class** `vison.inject.InjTask.InjTask(*args, **kwargs)`

**BROKEN\_basic\_analysis()**

Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      extract average 2D injection pattern (and save)
      produce average profile along/across lines
      measure charge-inj. non-uniformity
      measure charge spillover into non-injection
      measure stats of injection (mean, med, std, min/max, percentiles)

plot average inj. profiles along lines f. each CCD, Q and IG1
  save as a rationalized set of curves
plot average inj. profiles across lines f. each CCD, Q and IG1
  save as a rationalized set of curves

Report injection stats as a table/tables
```

**check\_data(\*\*kwargs)**

**check\_metrics\_ST(\*\*kwargs)**

TODO:

- offset levels (pre and over-scan), abs. and relative
- RON in pre and overscan
- mean fluence/signal in image area [script-column-dependent]
- med fluence/signal in image area [script-column-dependent]
- std in image area [script-column-dependent]

`get_FluenceAndGradient_limits ()`

`get_checkstats_ST (**kwargs)`

`predict_expected_injlevels (teststruct)`

`prepare_images (doExtract=True, doMask=True, doOffset=True, doBias=True, doFF=False)`

InjTask: Preparation of data for further analysis. Calls `task.prepare_images()`.

**Applies:** offset subtraction [bias structure subtraction, if available] cosmetics masking

## 6.1.2 lib.py

NEEDSREVISION

Module to provide common tools for analysis of Charge Injection acquisitions.

Created on Thu Sep 14 15:32:10 2017

**author** Ruyman Azzollini

## 6.1.3 plot.py

Charge Injection Plotting Tools.

Created on Thu Sep 14 15:39:34 2017

**author** Ruyman Azzollini

## “FLAT” ACQ. ANALYSIS TOOLS

### 7.1 “Flat” Acq. Analysis Tools

#### 7.1.1 FlatTask.py

Created on Mon Dec 4 16:00:10 2017

**author** Ruyman Azzollini

```
class vison.flat.FlatTask.FlatTask (*args, **kwargs)
```

```
    check_data ()
```

```
    check_metrics_ST (**kwargs)
```

**TODO:**

- offset levels (pre and over-scan), abs. and relative
- RON in pre and overscan
- fluence in image area [script-column-dependent]
- variance in image area [script-column-dependent]

```
    get_checkstats_ST (**kwargs)
```

#### 7.1.2 FlatFielding.py

Flat-fielding Utilities.

Created on Fri Apr 22 16:13:22 2016

@author: raf

```
class vison.pipe.FlatFielding.FlatField (fitsfile='', data={}, meta={})
```

```
    parse_fits ()
```

```
vison.pipe.FlatFielding.fit2D (xx, yy, zz, degree=1)
```

```
vison.pipe.FlatFielding.get_ilum (img,    pdegree=5,    filtsize=15,    filtertype='median',  
                                Tests=False)
```

```
vison.pipe.FlatFielding.get_ilum_splines (img,    filtsize=25,    filtertype='median',  
                                           Tests=False)
```

`vison.pipe.FlatFielding.produce_IndivFlats` (*infits, outfits, settings, runonTests, processes=6*)

`vison.pipe.FlatFielding.produce_MasterFlat` (*infits, outfits, mask=None, settings={}*)  
Produces a Master Flat out of a number of flat-illumination exposures. Takes the outputs from `produce_IndivFlats`.

`vison.pipe.FlatFielding.produce_SingleFlatfield` (*infits, outfits, settings={}, runonTests=False*)

### 7.1.3 nl.py

NEEDSREVISION

Module with tools used in NL analysis.

Created on Mon Feb 5 15:51:00 2018

**author** Ruyman Azzollini

`vison.flat.nl.fitNL` (*fluencesNL, exptimes, display=False*)

`vison.flat.nl.get_exptime_atmiddynrange` (*fluID, expID, method='spline', debug=False*)

`vison.flat.nl.test_wrap_fitNL` ()

`vison.flat.nl.wrap_fitNL` (*fluences, variances, exptimes, col\_labels, times=array([], dtype=float64), TrackFlux=True, subBgd=True*)

### 7.1.4 ptc.py

NEEDSREVISION

Module with tools used in PTC analysis.

Created on Thu Sep 14 16:29:36 2017

**author** Ruyman Azzollini

`vison.flat.ptc.fitPTC` (*means, var*)

Fits Photon Transfer Curve to obtain gain.

`vison.flat.ptc.foo_bloom` (*means, var*)

DUMMY function (PLACE-HOLDER) (Will) Finds blooming limit (where variance drops, if it does...).



## 8.1 Image Analysis

### 8.1.1 bits.py

NEEDSREVISION

Image bits analysis tools.

Created on Thu Sep 14 15:54:14 2017

**author** Ruyman Azzollini

### 8.1.2 calibration.py

Common use CDP functions / methods.

Created on Thu Nov 2 16:54:28 2017

**author** Ruyman Azzollini

```
vison.image.calibration.load_FITS_CDPs (FDict, dataclass, **kwargs)
```

Dummy function to load CDPs for all 3 CCDs. Input is of type dict(CCD1='', CCD2='', CCD3='')

### 8.1.3 cosmetics.py

Created on Wed Aug 1 11:55:12 2018

@author: Ruyman Azzollini

```
vison.image.cosmetics.get_Thresholding_DefectsMask (maskdata, thresholds)
```

### 8.1.4 covariance.py

Tools to retrieve covariance matrices for (differences of) Flat-Field images. Used in the context of Brighter-Fatter analysis, mainly.

Created on Wed Mar 7 11:54:54 2018

**author** Ruyman Azzollini

```
vison.image.covariance.f_get_covmap (sq1, sq2, N, debug=False)
```

```
vison.image.covariance.get_cov_maps (ccdobjList, Npix=4, doTest=False)
```

### 8.1.5 ds9reg.py

DS9 Regions tool.

Created on Fri May 18 15:02:07 2018

**author** raf

`vison.image.ds9reg.get_body_circles` (*X, Y, R=None, radius=6.0*)

`vison.image.ds9reg.get_body_ellipses` (*X, Y, A=None, B=None, THETA=None*)

`vison.image.ds9reg.save_spots_as_ds9regs` (*data, regfilename=None, regfile=None, reg-  
type='circle', clobber=True*)

### 8.1.6 performance.py

Performance parameters of the ROE+CCDs. Compilation of CCD offsets, offset gradients, RONs... used for checks.

Created on Wed Nov 1 09:57:44 2017

**author** Ruyman Azzollini

`vison.image.performance.get_offsets_lims` (*offsets, offsets\_margins*)

`vison.image.performance.get_perf_rdout` (*BLOCKID*)

### 8.1.7 pixbounce.py

Pixel Bounce Analysis methods.

Created on Fri Mar 9 09:50:16 2018

**author** Ruyman Azzollini

`vison.image.pixbounce.get_pixbounce_from_overscan` (*ccdobj, thresholds=None*)

Retrieves Hard Edge Respose for all Quadrants of a CCD. Uses the transition from image to overscan (along rows). Averages across rows. Input image should have high image-area fluence but not saturating. Rows can be filtered by average fluence in them via “thresholds” keyword. Do not use on images acquired with irradiated CCDs.

### 8.1.8 sextractor.py

Sextractor interface.

Created on Thu May 17 13:29:05 2018

**author** raf

**class** `vison.image.sextractor.VSExtractor` (*img=None*)

`load_catalog` (*catpath*)

`run_SEx` (*catroot, config=None, checks=None, cleanafter=False*)

`save_img_to_tmp` (*img, delete=True, close=False*)

## MONITORING (“EYEGORE”)

Tools to monitor data acquisition on real time: plots of HK, auto-updating of visual display of Exposure Log with some interactive capabilities, and display of latest images.

### 9.1 Monitoring (“Eyegore”)



Fig. 9.1: You must be Igor...

#### 9.1.1 eyegore.py

eyegore

data acquisition monitoring script for vison package.

‘- You must be Igor... - No, it’s pronounced “Eye-gore”.’

Created on Thu Feb 2 15:27:39 2017

**Author** Ruyman Azzollini

```
class vison.eyegore.eyegore.Eyegore(path, broadcast, intervals=None, elvis='6.5.X', do-  
lite=False, altpath='', doWarnings=False, dolog=False)
```

```
    setup_MasterWG()
```

```
vison.eyegore.eyegore.rsync_to_altlocalpath(path, altpath)
```

```
vison.eyegore.eyegore.rsync_to_remote(path, broadcast)
```

## 9.1.2 eyeCCDs.py

Eyegore: CCDs display.

Created on Fri Oct 13 16:16:08 2017

```
    author raf
```

```
class vison.eyegore.eyeCCDs.ImageDisplay(parent, path, elvis='6.5.X')
```

```
    gen_render()
```

```
    setup_fig()
```

## 9.1.3 eyeHK.py

Eyegore: House Keeping Monitoring.

Created on Fri Oct 13 14:11:41 2017

```
    author raf
```

```
class vison.eyegore.eyeHK.HKDisplay(root, path, interval, elvis='6.5.X')
```

```
    get_data()
```

```
    search_HKfiles()
```

```
    select_HKkeys()
```

```
class vison.eyegore.eyeHK.HKFlags(root, parent, interval=5000, elvis='6.5.X')
```

```
    MuteFlag(event)
```

```
    ResetFlag(event)
```

```
    UnmuteFlag(event)
```

```
    bind_buttons_to_methods(ix)
```

```
    changeColor(ix, color)
```

```
    isflagraised(ix)
```

```
    lowerflag(ix)
```

```
    raiseflag(ix)
```

```
class vison.eyegore.eyeHK.SingleHKplot(root)
```

```
vison.eyegore.eyeHK.sort_HKfiles(HKfiles)
```

```
vison.eyegore.eyeHK.validate_within_HKlim(val, HKlim)
```

**violation:** 0: None -1: below lower limit 1: above upper limit 2: different from limit, if limit is a single value

### 9.1.4 eyeObs.py

Eyegore: Exposure Log Monitoring.

Created on Fri Oct 13 16:22:36 2017

**author** raf

**class** vison.eyegore.eyeObs.**ExpLogDisplay** (*parent, path, interval, elvis='6.5.X'*)

**build\_elementList** ()

**get\_data** ()

**loadExplogs** ()

**search\_EXPLOGs** ()

**sortBy** (*tree, col, descending*)

sort tree contents when a column header is clicked

vison.eyegore.eyeObs.**changeNumeric** (*data*)

if the data to be sorted is numeric change to float

vison.eyegore.eyeObs.**isNumeric** (*s*)

test if a string s is numeric

### 9.1.5 eyeWarnings.py

Module to handle HK-OOL Warnings

Created on Thu Apr 19 16:09:02 2018

**author** Ruyman Azzollini

vison.eyegore.eyeWarnings.**test\_URLs** ()



OGSE stands for Optical Ground Support Equipment.

## 10.1 OGSE Tools

### 10.1.1 ogse.py

Model of the calibration OGSE

Created on Fri Sep 8 12:11:55 2017

**author** Ruyman Azzollini

```
vison.ogse.ogse.get_FW_ID(wavelength, FW={'F1': 590, 'F2': 640, 'F3': 730, 'F4': 800, 'F5':  
880, 'F6': 0})  
returns FW key corresponding to input wavelength. :param wavelength: integer, wavelength.
```





General use plotting facilities.

## 11.1 Plotting

### 11.1.1 baseplotclasses.py

vison pipeline: Classes to do plots.

Created on Mon Nov 13 17:54:08 2017

**author** Ruyman Azzollini

```
class vison.plot.baseplotclasses.Beam1DHist (data, **kwargs)
```

```
class vison.plot.baseplotclasses.BeamPlot (data, **kwargs)
```

```
    populate_axes ()
```

```
vison.plot.baseplotclasses.testBeam2ImgShow ()
```

```
class vison.plot.baseplotclasses.BasicPlot (**kwargs)
```

```
class vison.plot.baseplotclasses.Beam1DHist (data, **kwargs)
```

```
class vison.plot.baseplotclasses.BeamImgShow (data, **kwargs)
```

```
class vison.plot.baseplotclasses.BeamPlot (data, **kwargs)
```

```
    populate_axes ()
```

```
class vison.plot.baseplotclasses.BeamPlotYvX (data, **kwargs)
```

```
class vison.plot.baseplotclasses.CCD2DPlot (data, **kwargs)
```

```
class vison.plot.baseplotclasses.ImgShow (data, **kwargs)
```

```
    plt_trimmer ()
```

```
    populate_axes ()
```

### 11.1.2 figclasses.py

Created on Mon Apr 16 16:17:13 2018

**author** Ruyman Azzollini

**class** `vison.plot.figclasses.BlueScreen`

**build\_data** (*\*args, \*\*kwargs*)

**configure** (*\*\*kwargs*)

### 11.1.3 trends.py

Plotting classes shared across tasks/sub-tasks and derived from `plots.baseclasses`. They have in common that they show trends with time of some variables / stats.

Created on Fri Jan 26 16:18:43 2018

**author** raf

## POINT-SOURCE ANALYSIS

### 12.1 Point-Source Analysis

#### 12.1.1 basis.py

**author** Ruyman Azzollini

Created on Thu Apr 20 18:56:40 2017

```
class vison.point.basis.SpotBase (data, log=None, verbose=False)
```

#### 12.1.2 display.py

##### Display Library for Point-Source Analysis

Created on Fri Apr 21 14:02:57 2017

**requires** matplotlib

**author** Ruyman Azzollini

```
vison.point.display.show_spots_allCCDs (spots_bag, title='', filename='', dobar=True)
```

#### 12.1.3 gauss.py

##### Gaussian Model of Point-like Sources

Simple class to do Gaussian Fitting to a spot.

**requires** NumPy, astropy

Created on Thu Apr 20 16:42:47 2017

**author** Ruyman Azzollini

```
class vison.point.gauss.Gaussmeter (data, log=None, verbose=False, **kwargs)
```

Provides methods to measure the shape of an object using a 2D Gaussian Model.

##### Parameters

- **data** (*np.ndarray*) – stamp to be analysed.
- **log** (*instance*) – logger
- **kwargs** (*dict*) – additional keyword arguments

Settings dictionary contains all parameter values needed.

`fit_Gauss()`

## 12.1.4 models.py

### Models (Point-Like Sources)

Library module with models for processing of point-source imaging data.

**requires** NumPy

**author** Ruyman Azzollini

Created on Wed Apr 19 11:47:00 2017

`vison.point.models.fgauss2D(x, y, p)`

A gaussian fitting function where  $p[0]$  = amplitude  $p[1]$  =  $x_0$   $p[2]$  =  $y_0$   $p[3]$  =  $\sigma_{\max}$   $p[4]$  =  $\sigma_{\text{may}}$   $p[5]$  = floor

## 12.1.5 photom.py

### Aperture Photometry of point-like objects

Simple class to do aperture photometry on a stamp of a point-source.

**requires** NumPy

Created on Thu Apr 20 14:37:46 2017

**author** Ruyman Azzollini

**class** `vison.point.photom.Photometer` (*data*, *log=None*, *verbose=False*, *\*\*kwargs*)  
 Provides methods to measure the shape of an object.

#### Parameters

- **data** (*np.ndarray*) – stamp to be analysed.
- **log** (*instance*) – logger
- **kwargs** (*dict*) – additional keyword arguments

Settings dictionary contains all parameter values needed.

`doap_photom` (*centre*, *rap*, *rin=-1.0*, *rout=-1.0*, *gain=3.5*, *doErrors=True*, *subbgd=False*)

`get_centroid` (*rap=None*, *full=False*)

**TODO:** add aperture masking

`measure_bgd` (*rin*, *rout*)

`sub_bgd` (*rin*, *rout*)

## 12.1.6 shape.py

### Quadrupole Moments Shape Measurement

Simple class to measure quadrupole moments and ellipticity of an object.

**requires** NumPy, PyFITS

**author** Sami-Matias Niemi, Ruyman Azzollini

**class** `vison.point.shape.Shapemeter` (*data*, *log=None*, *verbose=False*, *\*\*kwargs*)  
Provides methods to measure the shape of an object.

#### Parameters

- **data** (*np.ndarray*) – stamp to be analysed.
- **log** (*instance*) – logger
- **kwargs** (*dict*) – additional keyword arguments

Settings dictionary contains all parameter values needed.

**circular2DGaussian** (*x*, *y*, *sigma*)

Create a circular symmetric Gaussian centered on *x*, *y*.

#### Parameters

- **x** (*float*) – *x* coordinate of the centre
- **y** (*float*) – *y* coordinate of the centre
- **sigma** (*float*) – standard deviation of the Gaussian, note that `sigma_x = sigma_y = sigma`

**Returns** circular Gaussian 2D profile and *x* and *y* mesh grid

**Return type** dict

**ellip2DGaussian** (*x*, *y*, *sigmax*, *sigmay*)

Create a two-dimensional Gaussian centered on *x*, *y*.

#### Parameters

- **x** (*float*) – *x* coordinate of the centre
- **y** (*float*) – *y* coordinate of the centre
- **sigmax** (*float*) – standard deviation of the Gaussian in *x*-direction
- **sigmay** (*float*) – standard deviation of the Gaussian in *y*-direction

**Returns** circular Gaussian 2D profile and *x* and *y* mesh grid

**Return type** dict

**measureRefinedEllipticity** ()

Derive a refined iterated polarisability/ellipticity measurement for a given object.

By default polarisability/ellipticity is defined in terms of the Gaussian weighted quadrupole moments. If `self.shsettings['weighted']` is False then no weighting scheme is used.

The number of iterations is defined in `self.shsettings['iterations']`.

**Returns** centroids [indexing stars from 1], ellipticity (including projected *e1* and *e2*), and *R2*

**Return type** dict

**quadrupoles** (*image*)

Derive quadrupole moments and ellipticity from the input image.

**Parameters** **img** (*ndarray*) – input image data

**Returns** quadrupoles, centroid, and ellipticity (also the projected components e1, e2)

**Return type** dict

**writeFITS** (*data, output*)

Write out a FITS file using PyFITS.

**Parameters**

- **data** (*ndarray*) – data to write to a FITS file
- **output** (*string*) – name of the output file

**Returns** None

## 12.1.7 spot.py

Spot Stamp Class.

Created on Thu Apr 20 15:35:08 2017

**author** Ruyman Azzollini

**class** vison.point.spot.**Spot** (*data, log=None, verbose=False, lowerleft=(None, ), \*\*kwargs*)

Provides methods to do point-source analysis on a stamp. Aimed at basic analysis:

- Photometry
- Quadrupole Moments
- Gaussian Fit

**Parameters**

- **data** (*np.ndarray*) – stamp to be analysed.
- **log** (*instance*) – logger
- **kwargs** (*dict*) – additional keyword arguments

Settings dictionary contains all parameter values needed.

**get\_photom** ()

measurements: 'apflu', 'eapflu', 'bgd', 'ebgd'

**get\_shape\_Gauss** ()

**Returns** res = dict(i0,ei0,x,ex,y,ey, sigma\_x,esigma\_x,sigmat,esigma\_y, fwhm\_x,efwhm\_x, fwhm\_y,efwhm\_y, fluence,efluence)

**get\_shape\_Moments** ()

**Returns** res = dict(x,y,ellip,e1,e2,a,b)

**get\_shape\_easy** (*method='G', debug=False*)

**measure\_basic** (*rap=10, rin=10, rout=-1, gain=3.1, debug=False*)

# TODO: # get basic statistics, measure and subtract background # update centroid # do aperture photometry # pack-up results and return

**Parameters**

- **rap** – source extraction aperture radius.
- **rin** – inner radius of background annulus.
- **rou**t – outer radius of background annulus (-1 to set bound by image area).
- **gain** – image gain (e-/ADU).

### 12.1.8 lib.py

Library module with useful data and functions for processing of point-source imaging data.

Created on Wed Apr 5 10:21:05 2017

**author** Ruyman Azzollini (except where indicated)

`vison.point.lib.extract_spot` (*ccdobj*, *coo*, *Quad*, *log=None*, *stampw=25*)

```
vison.point.lib.gen_point_mask(CCD, Quad, width=75, sources='all', coodict={'CCD2':  
    OrderedDict([('H', OrderedDict([('BRAVO', (1725.0, 1606.0)),  
    ('CHARLIE', (1131.0, 1029.0)), ('ALPHA', (554.0, 1626.0)),  
    ('ECHO', (1706.0, 435.0)), ('DELTA', (531.0, 446.0))])),  
    ('E', OrderedDict([('BRAVO', (1716.0, 1700.0)), ('CHAR-  
    LIE', (1126.0, 1124.0)), ('ALPHA', (542.0, 1725.0)),  
    ('ECHO', (1695.0, 537.0)), ('DELTA', (521.0, 551.0))])),  
    ('G', OrderedDict([('BRAVO', (1702.0, 1571.0)), ('CHAR-  
    LIE', (1139.0, 1033.0)), ('ALPHA', (534.0, 1590.0)),  
    ('ECHO', (1685.0, 394.0)), ('DELTA', (515.0, 415.0))])),  
    ('F', OrderedDict([('BRAVO', (1745.0, 1668.0)), ('CHAR-  
    LIE', (1141.0, 1144.0)), ('ALPHA', (578.0, 1686.0)),  
    ('ECHO', (1723.0, 496.0)), ('DELTA', (553.0, 522.0))])),  
    'CCD3': {'H': {'BRAVO': (1689.4, 1668.8000000000002),  
    'ALPHA': (460.6, 1668.8000000000002), 'DELTA':  
    (460.6, 417.20000000000005), 'ECHO': (1689.4,  
    417.20000000000005), 'CHARLIE': (1075.0, 1043.0)},  
    'E': {'BRAVO': (1689.4, 3754.8), 'ALPHA': (460.6,  
    3754.8), 'DELTA': (460.6, 2503.2), 'ECHO':  
    (1689.4, 2503.2), 'CHARLIE': (1075.0, 3129.0)},  
    'G': {'BRAVO': (3808.4, 1668.8000000000002), 'AL-  
    PHA': (2579.6, 1668.8000000000002), 'DELTA':  
    (2579.6, 417.20000000000005), 'ECHO': (3808.4,  
    417.20000000000005), 'CHARLIE': (3194.0, 1043.0)},  
    'F': {'BRAVO': (3808.4, 3754.8), 'ALPHA': (2579.6,  
    3754.8), 'DELTA': (2579.6, 2503.2), 'ECHO': (3808.4,  
    2503.2), 'CHARLIE': (3194.0, 3129.0)}}}, 'CCD1': {'H':  
    {'ALPHA': (460.6, 1668.8000000000002), 'CHARLIE':  
    (1075.0, 1043.0), 'DELTA': (460.6, 417.20000000000005),  
    'ECHO': (1689.4, 417.20000000000005), 'BRAVO': (1689.4,  
    1668.8000000000002)}, 'E': {'ALPHA': (460.6, 3754.8),  
    'CHARLIE': (1075.0, 3129.0), 'DELTA': (460.6, 2503.2),  
    'ECHO': (1689.4, 2503.2), 'BRAVO': (1689.4, 3754.8)},  
    'G': {'ALPHA': (2579.6, 1668.8000000000002), 'CHARLIE':  
    (3194.0, 1043.0), 'DELTA': (2579.6, 417.20000000000005),  
    'ECHO': (3808.4, 417.20000000000005), 'BRAVO': (3808.4,  
    1668.8000000000002)}, 'F': {'ALPHA': (2579.6, 3754.8),  
    'CHARLIE': (3194.0, 3129.0), 'DELTA': (2579.6, 2503.2),  
    'ECHO': (3808.4, 2503.2), 'BRAVO': (3808.4, 3754.8)}}},  
    'names': ['ALPHA', 'BRAVO', 'CHARLIE', 'DELTA',  
    'ECHO']})
```



These are pipeline scripts, not the Test Scripts (for those keep scrolling down).

## 13.1 Scripts

### 13.1.1 HKmonitor.py

**TODO** find HK files in a folder parse HK files plot HK parameters vs. time assemble all plots into a pdf file

**DEBUG**, calls nonexistent class LaTeX

Script to produce HK reports out of HK files in a folder. Aimed at quick inspection of data from Characterization and Calibration Campaigns of Euclid-VIS.

#### History

Created on Tue Mar 15 10:35:43 2016

@author: Ruyman Azzollini (MSSL)

### 13.1.2 quickds9.py

Wrap-up of ds9 to quickly load a number of images, for inspection.

#### History

Created on Thu Mar 17 13:18:10 2016

@author: Ruyman Azzollini

### 13.1.3 run\_xtalk.py

Master Script to measure and report cross-talk levels among 12 ROE channels. Takes as input a data-set composed of 3x12 CCD images, corresponding to injecting a “ladder” of signal on each of the 12 channels, using the ROE-TAB.

Created on Thu Mar 22 16:17:39 2018

**author** Ruyman Azzollini

```
vison.scripts.run_xtalk.run_xtalk(incat, inpath='', respath='', metafile='', doCompute=False)
```

### 13.1.4 vis\_cosmetics\_masker.py

Script to create cosmetics masks in VIS Ground Calibration Campaign.

Created on Wed Aug 1 11:02:00 2018

**author** Ruyman Azzollini

```
vison.scripts.vis_cosmetics_masker.do_Mask(inputs, masktype, subbgd=True, normby-  
bgd=False, validrange=None)  
vison.scripts.vis_cosmetics_masker.pre_process(FITS_list, subOffset=False,  
validrange=None)  
vison.scripts.vis_cosmetics_masker.read_OBSID_list(ff)  
vison.scripts.vis_cosmetics_masker.run_maskmaker(inputs)
```

### 13.1.5 vis\_explogs\_merger.py

Created on Fri Feb 9 15:20:01 2018

@author: raf

```
vison.scripts.vis_explogs_merger.explog_merger(ELlist, output='EXP_LOG_merged.txt',  
elvis='6.5.X')
```

### 13.1.6 vis\_genDataSet.py

Development: Creating Calibration Campaign Fake Data-set

Created on Tue Sep 05 16:07:00 2017

**author** Ruyman Azzollini

```
vison.scripts.vis_genDataSet.datasetGenerator(TestsSelector, doGenExplog, doGenHK,  
doGenFITS, outpath, elvis, CHAMBER,  
Nrows=0)  
vison.scripts.vis_genDataSet.genExpLog(toGen, explogf, equipment, elvis='6.5.X', CHAM-  
BER=None)
```

### 13.1.7 vis\_load\_DD.py

Loading a DataDict object for inspection.

Created on Wed Aug 01 10:00:00 2018

**author** Ruyman Azzollini

### 13.1.8 vis\_mkscripts.py

Automatically Generating Calibration Campaign Data Acquisition Scripts. Aimed at ELVIS.

Created on Fri Sep 08 12:03:00 2017

**author** Ruyman Azzollini

### 13.1.9 vis\_star\_finder.py

Script to find point sources in VIS Ground Calibration Campaign. Used to ‘prime’ the position tables of point-source objects.

Created on Tue Jun 12 16:09:31 2018

**author** Ruyman Azzollini

```
vison.scripts.vis_star_finder.write_ID_chart(filename, Quads, Starnames)
```

### 13.1.10 v\_ROE\_LinCalib.py

Non-Linearity Calibration of ROE (on bench).

Created on Thu Mar 15 15:32:11 2018

**author** Ruyman Azzollini

```
vison.scripts.v_ROE_LinCalib.find_adu_levels(qdata, Nlevels, debug=False)
```

```
vison.scripts.v_ROE_LinCalib.run_ROE_LinCalib(inputsfile, incatfile, datapath='',
                                             respath='', doExtractFits=True,
                                             dopolyRT=False, debug=False)
```

### 13.1.11 v\_ROETAB\_LinCalib.py

Linearity Calibration of ROE-TAB.

Created on Tue Mar 27 14:42:00 2018 Modified on Fri Sep 14 10:53:00 2018

**author** Ruyman Azzollini

```
vison.scripts.v_ROETAB_LinCalib.filter_voltage_uni(rV, filt_kernel)
```

```
vison.scripts.v_ROETAB_LinCalib.find_discrete_voltages_inwaveform(rV, levels, filtered=None,
                                                                    debug=False)
```

```
vison.scripts.v_ROETAB_LinCalib.load_WF(WFf, chkNsamp=None, chkSampInter=None)
```

```
vison.scripts.v_ROETAB_LinCalib.plot_waveform(WF, disc_voltages=[], figname='',
                                              chan='Unknown')
```

```
vison.scripts.v_ROETAB_LinCalib.run_ROETAB_LinCalib(inputsfile, incatfile, datapath='',
                                                    respath='', doBayes=False, debug=False)
```



## SUPPORT CODE

### 14.1 Support Code

#### 14.1.1 context.py

Common Values which are used by functions and classes throughout pipeline.

Created on Tue Jan 16 10:53:40 2018

**author** Ruyman Azzollini

#### 14.1.2 ET.py

Module to issue WARNING / ALERT phone calls to designated phone numbers. Uses Twilio.

'... E.T. phone home...'

Created on Thu Sep 14 10:13:12 2017

**author** raf

**class** `vison.support.ET.ET`

Class to do phone calls.

**dial\_numbers** (*url*)

Dials one or more phone numbers from a Twilio phone number.

**Parameters** `url` – char, URL with the TwiML code that Twilio uses as instructions on call.

Basically, it provides a message to be voiced, as intended.

**send\_sms** (*body*)

`vison.support.ET.grab_numbers_and_codes()`

Retrieves phone numbers and access codes necessary to make the phone calls.

#### 14.1.3 excel.py

Excel Files Interfaces.

Created on Mon Mar 26 12:07:54 2018

**author** Ruyman Azzollini

`vison.support.excel.test0()`

Just a dummy test to show we can use openpyxl

### 14.1.4 files.py

IO related functions.

**requires** PyFITS

**requires** NumPy

**author** Sami-Matias Niemi

`vison.support.files.cPickleDump` (*data*, *output*, *protocol=2*)  
Dumps data to a cPickled file.

**Parameters**

- **data** – a Python data container
- **output** – name of the output file

**Returns** None

`vison.support.files.cPickleDumpDictionary` (*dictionary*, *output*, *protocol=2*)  
Dumps a dictionary of data to a cPickled file.

**Parameters**

- **dictionary** – a Python data container does not have to be a dictionary
- **output** – name of the output file

**Returns** None

`vison.support.files.cPickleRead` (*file*)  
Loads data from a pickled file.

`vison.support.files.convert_fig_to_eps` (*filename*)  
Converts a figure to .eps. Returns new file name.

### 14.1.5 flags.py

Functions and variables related to flags for vison.

Created on Wed Sep 20 17:05:00 2017

**author** Ruyman Azzollini

`class vison.support.flags.Flags` (*indict=None*)

### 14.1.6 latex.py

Just a collection of LaTeX-generating functions for use in report.py

**History**

Created on Mon Jan 30 2017

**author** Ruyman Azzollini

`vison.support.latex.generate_header` (*test*, *model*, *author*, *reference='7-XXX'*)

`vison.support.latex.replace_in_template` (*text*, *values*)

### 14.1.7 logger.py

These functions can be used for logging information.

**Warning:** logger is not multiprocessing safe.

**author** Sami-Matias Niemi

**version** 0.3

**class** `vison.support.logger.SimpleLogger` (*filename, verbose=False*)  
A simple class to create a log file or print the information on screen.

**write** (*text*)  
Writes text either to file or screen.

`vison.support.logger.f_text_wrapper` (*msg*)

`vison.support.logger.setUpLogger` (*log\_filename, logername='logger'*)  
Sets up a logger.

**Param** *log\_filename*: name of the file to save the log.

**Param** *logername*: name of the logger

**Returns** logger instance

### 14.1.8 report.py

LaTeX - PDF Reporting Utilities.

#### History

Created on Wed Jan 25 16:58:33 2017

**author** Ruyman Azzollini

**class** `vison.support.report.Container`

**add\_to\_Contents** (*item*)

**class** `vison.support.report.Content` (*contenttype=''*)

**class** `vison.support.report.FigsTable` (*FigsList, Ncols, figswidth, caption=None*)  
Class to generate table of figures

**generate\_Latex** ()  
Generates LaTeX as list of strings

**class** `vison.support.report.Figure` (*figpath, textfraction=0.7, caption=None, label=None*)

**generate\_Latex** ()  
Generates LaTeX as list of strings.

**class** `vison.support.report.Section` (*keyword, Title='', level=0*)

**generate\_Latex** ()

**class** `vison.support.report.Table` (*tableDict, formats={}, names=[], caption=None, col\_align=None, longtable=False*)

**PENDING:**

- adjust width of table to texwidth:

```
esizebox{ extwidth}{!}{  
    ... end{tabular}}
```

- include option to rotate table to show in landscape

```
generate_Latex ()  
    Generates LaTeX as list of strings.
```

```
class vison.support.report.Text (text)
```

```
    generate_Latex ()
```

### 14.1.9 utils.py

General Purpose Utilities

Created on Tue Apr 10 15:18:07 2018

**author** Ruyman Azzollini

#### 14.1.10 vistime.py

Accessory library: time related operations

Created on Tue Oct 10 15:08:28 2017

**author** Ruyman Azzollini

```
vison.support.vistime.get_dtobj (DT)
```

```
vison.support.vistime.get_time_tag ()
```

#### 14.1.11 vjson.py

json files handling utilities.

Created on Tue Mar 27 14:25:43 2018

**author** Ruyman Azzollini

```
vison.support.vjson.dump_to_json (pydict)
```

```
vison.support.vjson.load_jsonfile (jsonfile, useyaml=False)
```

```
vison.support.vjson.save_jsonfile (pydict, jsonfile)
```



## UNIT TESTING

### 15.1 Unit Testing

#### 15.1.1 test\_ccdpile.py

Unit-testing for CCDPile class.

Created on Mon May 7 09:47:07 2018

**author** Ruyman Azzollini

#### 15.1.2 test\_ccd.py

Unit-testing for CCD class.

Created on Mon May 7 09:47:07 2018

**author** Ruyman Azzollini



## TEST SCRIPTS

These are the scripts that hold the description, execution, data validation and analysis of the tests that make the campaign. They are served by the infrastructure and tools provided by the pipeline.

**WARNING:** Currently most of the test scripts are largely meta-code, with the exception of very basic functionality used to generate acquisition scripts and validate the acquisitions, as listed in the Exposure Log, against the description of the test. The metacode has been included in the doc-strings for ease of browsing.

### 16.1 Charge Injection Scripts

#### 16.1.1 Charge Injection Scripts

##### CHINJ01

VIS Ground Calibration TEST: CHINJ01

**Charge injection calibration (part 1)** Injection vs. IG1-IG2

Created on Tue Aug 29 17:36:00 2017

**author** Ruyman Azzollini

**class** `vison.inject.CHINJ01.CHINJ01` (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds CHINJ01 script structure dictionary.

*#:param IDL: float, [V], value of IDL (Inject. Drain Low). #:param IDH: float, [V], Injection Drain High.*

*#:param IG2: float, [V], Injection Gate 2. #:param IG1s: list of 2 floats, [V], [min,max] values of IG1.*

*#:param id\_delays: list of 2 floats, [us], injection drain delays. #:param toi\_chinj: int, [us], TOI-charge injection. :param diffvalues: dict, opt, differential values.*

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Find injection threshold: Min IG1 Plot and model charge injection vs. IG1 Find notch injection amount.

**old\_basic\_analysis** ()

Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      extract average 2D injection pattern (and save)
      produce average profile along/across lines
      measure charge-inj. non-uniformity
      measure charge spillover into non-injection
      measure stats of injection (mean, med, std, min/max, percentiles)

plot average inj. profiles along lines f. each CCD, Q and IG1
  save as a rationalized set of curves
plot average inj. profiles across lines f. each CCD, Q and IG1
  save as a rationalized set of curves

Report injection stats as a table/tables
```

**set\_inpdefaults** (\*\*kwargs)

## CHINJ02

VIS Ground Calibration TEST: CHINJ02

**Charge injection calibration (part 2)** Injection vs. IDL (injection threshold)

Created on Tue Aug 29 17:36:00 2017

**author** Ruyman Azzollini

**class** vison.inject.CHINJ02.**CHINJ02** (inputs, log=None, drill=False, debug=False)

**basic\_analysis** ()

Basic analysis of data. AS IT IS, REPEATS WHAT'S DONE IN THE CHECK\_DATA. CONSIDER MERGING/SKIPPING

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      load average 2D injection pattern
      produce average profile along lines
      [measure charge-inj. non-uniformity]
      [produce average profile across lines]
      [measure charge spillover into non-injection]
      measure stats of injection (mean, med, std, min/max, percentiles)

[plot average inj. profiles along lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]
[plot average inj. profiles across lines f. each CCD, Q and IG1]
[  save as a rationalized set of curves]

save&plot charge injection vs. IDL
report injection stats as a table
```

**build\_scriptdict** (diffvalues={}, elvis='6.5.X')

Builds CHINJ02 script structure dictionary.

#:param IDLs: list of 2 ints, [V], [min,max] values of IDL (Inject. Drain Low). #:param IDH: int, [V], Injection Drain High. #:param id\_delays: list of 2 ints, [us], injection drain delays. #:param toi\_chinj: int, [us], TOI-charge injection. :param diffvalues: dict, opt, differential values.

**extract\_data()**

**NEEDED?** Could be merged with basic\_analysis

**METACODE**

```
Preparation of data for further analysis:

f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      subtract offset
      extract average 2D injection pattern and save
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis()**

Finds the Injection Threshold for each CCD half.

**METACODE**

```
f.e.CCD:
  f.e.Q:
    load injection vs. IDL cuve
    find&save injection threshold on curve

report injection threshold as a table
```

**set\_inpdefaults** (\*\*kwargs)

## 16.2 Dark Scripts

### 16.2.1 “Dark Acquisitions” Scripts

**BIAS01**

TEST: BIAS01

Bias-structure/RON analysis script

Created on Tue Aug 29 16:53:40 2017

**author** Ruyman Azzollini

**class** vison.dark.BIAS01.**BIAS01** (*inputs, log=None, drill=False, debug=False*)

**basic\_analysis()**

BIAS01: Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:

      load ccdobj of ObsID, CCD
```

```

with ccdobj, f.e.Q:
    produce a 2D poly model of bias, save coefficients
    produce average profile along rows
    produce average profile along cols
    # save 2D model and profiles in a pick file for each OBSID-CCD
    measure and save RON after subtracting large scale structure

plot RON vs. time f. each CCD and Q
plot average profiles f. each CCD and Q (color coded by time)

```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

Builds BIAS01 script structure dictionary.

###:param N: integer, number of frames to acquire. :param diffvalues: dict, opt, differential values. :param elvis: char, ELVIS version.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

**METACODE**

```

f. each CCD:
    stack all ObsIDs to produce Master Bias
    f. e. Q:
        measure average profile along rows
        measure average profile along cols
plot average profiles of Master Bias(s) f. each CCD,Q
(produce table(s) with summary of results, include in report)
save Master Bias(s) (3 images) to FITS CDPs
show Master Bias(s) (3 images) in report
save name of MasterBias(s) CDPs to DataDict, report

```

**prep\_data** ()

BIAS01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**class** vison.dark.BIAS01.**Test** (*methodName='runTest'*)

Unit tests for the BIAS01 class.

**test\_check\_data** ()

**Returns** None

## DARK01

TEST: DARK01

“Dark Current” analysis script

Created on Tue Aug 29 17:21:00 2017

**author** Ruyman Azzollini

**class** vison.dark.DARK01.**DARK01** (*inputs*, *log=None*, *drill=False*, *debug=False*)

**basic\_analysis** ()

DARK01: Basic analysis of data.

**METACODE**

```
f. e. ObsID:
  f.e.CCD:
    f.e.Q:
      produce mask of hot pixels
      count hot pixels / columns
      produce a 2D poly model of masked-image, save coefficients
      produce average profile along rows
      produce average profile along cols
      measure and save RON after subtracting large scale structure
      save 2D model and profiles in a pick file for each OBSID-CCD

plot average profiles f. each CCD and Q (color coded by time)
```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

Builds DARK01 script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

**METACODE**

```
f. each CCD:
  f. e. Q:
    stack all ObsIDs to produce Master Dark
    produce mask of hot pixels / columns
    count hot pixels / columns
    measure average profile along rows
    measure average profile along cols

plot average profiles of Master Bias f. each CCD,Q
show Master Dark (images), include in report
report stats of defects, include in report
save name of MasterDark to DataDict, report
save name of Defects in Darkness Mask to DD, report
```

**prep\_data** ()

DARK01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [BIAS SUBTRACTION] cosmetics masking

## 16.3 Flat-Illumination Scripts

### 16.3.1 Flat-Illumination Scripts

**FLAT0X**

VIS Ground Calibration TEST: FLAT0X

Flat-fields acquisition / analysis script

Created on Tue Aug 29 17:32:52 2017

**author** Ruyman Azzollini

**class** vison.flat.FLAT0X.**FLAT0X** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds FLAT0X script structure dictionary.

**Parameters diffvalues** – dict, opt, differential values.

**do\_indiv\_flats** ()

**METACODE**

Preparation of data for further analysis and  
produce flat-field for each OBSID.

f.e. ObsID:

f.e.CCD:

load ccdobj

f.e.Q:

model 2D fluence distro in image area

produce average profile along rows

produce average profile along cols

save 2D model and profiles in a pick file for each OBSID-CCD

divide by 2D model to produce indiv-flat

save indiv-Flat to FITS(?), update add filename

plot average profiles f. each CCD and Q (color coded by time)

**do\_master\_flat** ()

**METACODE**

Produces Master Flat-Field

f.e.CCD:

f.e.Q:

stack individual flat-fields by chosen estimator

save Master FF to FITS

measure PRNU **and**

report PRNU figures

**do\_prdef\_mask** ()

**METACODE**

Produces mask of defects **in** Photo-Response

Could use master FF, **or** a stack of a subset of images (**in** order  
to produce mask, needed by other tasks, quicker).

f.e.CCD:

f.e.Q:

produce mask of PR defects

save mask of PR defects

count dead pixels / columns

report PR-defects stats

**filterexposures** (*structure, explog, OBSID\_lims*)



**prepare\_images** ()

FLAT0X: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction [bias structure subtraction, if available] cosmetics masking

**set\_inpdefaults** (\*\*kwargs)

## NL01

VIS Ground Calibration TEST: NL01

End-To-End Non-Linearity Curve

Tasks:

- Select exposures, get file names, get metadata (commandig, HK).
- Check exposure time pattern matches test design.
- Check quality of data (rough scaling of fluences with Exposure times).
- Subtract offset level.
- Divide by Flat-field.
- **Synoptic analysis:** fluence ratios vs. exptime ratios >> non-linearity curve
- extract: Non-Linearity curve for each CCD and quadrant
- produce synoptic figures
- Save results.

Created on Mon Apr 3 17:38:00 2017

**author** raf

**class** vison.flat.NL01.NL01 (inputs, log=None, drill=False, debug=False)

**build\_scriptdict** (diffvalues={}, elvis='6.5.X')

Builds NL01 script structure dictionary.

#:param expts: list of ints [ms], exposure times. #:param exptinter: int, ms, exposure time of interleaved source-stability exposures. #:param frames: list of ints, number of frames for each exposure time. #:param wavelength: int, wavelength. Default: 0 (Neutral Density Filter) :param diffvalues: dict, opt, differential values.

**do\_satCTE** ()

**METACODE**

```
select ObsIDs with fluence(exptime) >~ 0.5 FWC

f.e. ObsID:
    CCD:
        Q:
            measure CTE from amount of charge in over-scan relative to fluence

f.e. CCD:
    Q:
        get curve of CTE vs. fluence
        measure FWC from curve in ADU

report FWCs in electrons [via gain in inputs] f.e. CCD, Q (table)
```

**extract\_stats()**

Performs basic analysis: extracts statistics from image regions to later build NLC.

**METACODE**

```
create segmentation map given grid parameters

f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      f.e. "img-segment": (done elsewhere)
        measure central value
        measure variance
```

**filterexposures** (*structure, explog, OBSID\_lims*)

Loads a list of Exposure Logs and selects exposures from test PSFOX.

The filtering takes into account an expected structure for the acquisition script.

The datapath becomes another column in DataDict. This helps dealing with tests that run overnight and for which the input data is in several date-folders.

**prep\_data()**

Takes Raw Data and prepares it for further analysis.

**METACODE**

```
f.e. ObsID:
  f.e.CCD:
    f.e.Q:
      subtract offset
      opt: [sub bias frame]
      opt: [divide by FF]
      opt: [mask-out defects]
```

**produce\_NLCs()**

**METACODE**

```
Obtains Best-Fit Non-Linearity Curve

f.e. CCD:
  f.e. Q:

  [opt] apply correction for source variability (interspersed exposure
  with constant exptime)
  Build NL Curve (NLC) - use stats and exptimes
  fit poly. shape to NL curve

plot NL curves for each CCD, Q
report max. values of NL (table)
```

**PTCOX**

VIS Ground Calibration TEST: PTC\_0X

**Photon-Transfer-Curve Analysis** PTC01 - nominal temperature and wavelength PTC02 - alternative temperatures / wavelengths

Tasks:

- Select exposures, get file names, get metadata (commandig, HK).
- Check exposure time pattern matches test design.
- Check quality of data (rough scaling of fluences with Exposure times).
- Subtract pairs of exposures with equal fluence
- **Synoptic analysis:** variance vs. fluence variance(binned difference-frames) vs. fluence
- extract: RON, gain, gain(fluence)
- produce synoptic figures
- Save results.

Created on Mon Apr 3 17:00:24 2017

**author** raf

**class** vison.flat.PTCOX.**PTCOX** (*inputs, log=None, drill=False, debug=False*)

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)

Builds PTCOX script structure dictionary.

*#:param* exptimes: list of ints [ms], exposure times. *#:param* frames: list of ints, number of frames for each exposure time. *#:param* wavelength: int, wavelength. Default: 800 nm. *:param* diffvalues: dict, opt, differential values.

**extract\_PTC** ()

**Performs basic analysis of images:**

- builds PTC curves: both on non-binned and binned images

**METACODE**

```
create list of OBSID pairs

create segmentation map given grid parameters

f.e. OBSID pair:
  CCD:
    Q:
      subtract CCD images
      f.e. segment:
        measure central value
        measure variance
```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Analyzes the variance and fluence: gain, and gain(fluence)

**METACODE**

```
f.e. CCD:
  Q:
    (using stats across segments:)
    fit PTC to quadratic model
    solve for gain
    solve for alpha (pixel-correls, Guyonnet+15)
    solve for blooming limit (ADU)
    convert bloom limit to electrons, using gain
```

```
plot PTC curves with best-fit f.e. CCD, Q
report on gain estimates f. e. CCD, Q (table)
report on blooming limits (table)
```

`set_inpdefaults (**kwargs)`

## 16.4 Point-Source Scripts

### 16.4.1 Point-Source Scripts

#### FOCUS00

TEST: FOCUS00

Focus analysis script

Tasks:

- Select exposures, get file names, get metadata (commandig, HK).
- Check quality of data (integrated fluxes are roughly constant, matching expected level).
- Subtract offset level.
- Divide by Flat-field.
- **Crop stamps of the sources on each CCD/Quadrant.**
  - save snapshot figures of sources.
- **for each source (5 x Nquadrants):**
  - measure shape using Gaussian Fit
- Find position of mirror that minimizes PSF sizes
- **Produce synoptic figures:** source size and ellipticity across combined FOV (of 3 CCDs)
- Save results.

Created on Mon Apr 03 16:21:00 2017

**author** Ruyman Azzollini

`class vison.point.FOCUS00.FOCUS00 (inputs, log=None, drill=False, debug=False)`

`basic_analysis ()`

This is just an assignation of values measured in check\_data.

`build_scriptdict (diffvalues={}, elvis='6.5.X')`

Builds FOCUS00 script structure dictionary.

`#:param wavelength: int, [nm], wavelength. #:param exptime: int, [ms], exposure time. :param diffvalues: dict, opt, differential values.`

`filterexposures (structure, explog, OBSID_lims)`

`lock_on_stars ()`

`meta_analysis ()`

```
prep_data ()
```

## PSF0X

TEST: PSF0X

**PSF vs. Fluence, and Wavelength** PSF01 - nominal temperature PSF02 - alternative temperatures

Tasks:

- Select exposures, get file names, get metadata (commandig, HK).
- Check exposure time pattern matches test design.
- Check quality of data (rough scaling of fluences with Exposure times).
- Subtract offset level.
- Divide by Flat-field.
- **Crop stamps of the sources on each CCD/Quadrant.**
  - save snapshot figures of sources.
- **for each source:**
  - measure shape using weighted moments
  - measure shape using Gaussian Fit
  - Bayesian Forward Modelling the optomechanic+detector PSF
- Produce synoptic figures.
- Save results.

Created on Thu Dec 29 15:01:07 2016

**author** Ruyman Azzollini

## 16.5 Trap-Pumping Scripts

### 16.5.1 Trap-Pumping Scripts

#### TP01

VIS Ground Calibration TEST: TP01

Trap-Pumping calibration (vertical)

Created on Tue Aug 29 17:37:00 2017

**author** Ruyman Azzollini

```
class vison.pump.TP01.TP01 (inputs, log=None, drill=False, debug=False)
```

```
basic_analysis ()
```

```
Basic analysis of data.
```

```
METACODE
```

```
f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
    f.e.Q:
      load "map of relative pumping"
      find_dipoles:
        x, y, rel-amplitude, orientation

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for N and S)
  Counts of dipoles (and N vs. S)
```

**build\_scriptdict** (*diffvalues={}*, *elvis='6.5.X'*)

**extract** ()

Obtain maps of dipoles.

#### METACODE

```
f.e. id_delay (there are 2):
  f.e. CCD:
    f.e. Q:
      produce reference non-pumped injection map

f. e. ObsID:
  f.e. CCD:

    load ccdobj
    f.e.Q.:
      divide ccdobj.Q by injection map

    save dipole map and store reference
```

**filterexposures** (*structure*, *explog*, *OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

#### METACODE

```
across TOI_TP, patterns:
  build catalog of traps: x,y,I-phase, Amp
  from Amp (TOI) -> tau, Pc

Report on :
  Histogram of Taus
  Histogram of Pc (capture probability)
  Histogram of I-phases (larger phases should have more traps,
    statistically) -> check

Total Count of Traps
```

**set\_inpdefaults** (\*\*kwargs)

**TP02**

VIS Ground Calibration TEST: TP02

Trap-Pumping calibration (serial)

Created on Tue Aug 29 17:38:00 2017

**author** Ruyman Azzollini**class** vison.pump.TP02.**TP02** (*inputs, log=None, drill=False, debug=False*)**basic\_analysis** ()

Basic analysis of data.

**METACODE**

```
f. e. ObsID [there are different TOI_TP and TP-patterns]:
  f.e.CCD:
    f.e.Q:
      load raw 1D map of relative pumping (from extract_data)
      identify dipoles:
        x, rel-amplitude, orientation (E or W)

produce & report:
  map location of dipoles
  PDF of dipole amplitudes (for E and W)
  Counts of dipoles (and E vs. W)
```

**build\_scriptdict** (*diffvalues={}, elvis='6.5.X'*)**extract** ()

Obtain Maps of Serial Dipoles.

**filterexposures** (*structure, explog, OBSID\_lims*)**meta\_analysis** ()

Meta-analysis of data:

Try to identify tau and pixel-phase location for each trap. Need to associate dipoles across TOI\_TPs and TP-patterns

**METACODE**

```
across TOI_TP, patterns:
  build catalog of traps: x,y,R-phase, amp(dwell)
  from Amp(dwell) -> tau, Pc

Report on :
  Histogram of Taus
  Histogram of Pc (capture probability)
  Histogram of R-phases

Total Count of Traps
```

**set\_inpdefaults** (*\*\*kwargs*)

## 16.6 Other Test Scripts

### 16.6.1 Other Scripts

#### MOT\_FF

VIS Ground Calibration TEST: MOT\_FF

**Brighter-Fatter Analysis** Using data from test PTC01 (via BF01)

Hard Edge Response in serial / parallel Bit Correlations (ADC health)

Created on Tue Jul 31 18:04:00 2018

**author** raf

```
class vison.other.MOT_FF.MOT_FF (inputs, log=None, drill=False, debug=False)
```

```
    extract_HER()
```

#### PERSIST01

VIS Ground Calibration TEST: PERSIST01

CCD Persistence test

Created on Tue Aug 29 17:39:00 2017

**author** Ruyman Azzollini

```
class vison.other.PERSIST01.PERSIST01 (inputs, log=None, drill=False, debug=False)
```

```
    basic_analysis ()
```

Basic analysis of data.

#### METACODE

```
f.e.CCD:
    f.e.Q:
        use SATURATED frame to generate pixel saturation MASK
        measure stats in pix satur MASK across OBSIDs
        (pre-satur, satur, post-satur)
```

```
    build_scriptdict (diffvalues={}, elvis='6.5.X')
```

Builds PERSISTENCE01 script structure dictionary.

#### Parameters

- **exptSATUR** – int, saturation exposure time.
- **exptLATEN** – int, latency exposure time.
- **diffvalues** – dict, opt, differential values.

```
    check_data ()
```

PERSIST01: Checks quality of ingested data.

#### METACODE



```

check common HK values are within safe / nominal margins
check voltages in HK match commanded voltages, within margins

f.e.ObsID:
  f.e.CCD:
    f.e.Q.:
      measure offsets in pre-, over-
      measure std in pre-, over-
      measure fluence in apertures around Point Sources

assess std in pre- (~RON) is within allocated margins
assess offsets in pre-, and over- are equal, within allocated margins
assess fluence is ~expected within apertures (PS) for each frame (pre-satur,
↪satur, post-satur)

plot point source fluence vs. OBSID, all sources
[plot std vs. time]

issue any warnings to log
issue update to report

```

**filterexposures** (*structure, explog, OBSID\_lims*)

**meta\_analysis** ()

Meta-analysis of data.

**METACODE**

```

f.e.CCD:
  f.e.Q:
    estimate delta-charge_0 and decay tau from time-series

report:
  persistence level (delta-charge_0) and time constant

```

**prep\_data** ()

PERSIST01: Preparation of data for further analysis. Calls task.prepare\_images().

**Applies:** offset subtraction cosmetics masking

**set\_inpdefaults** (\*\*kwargs)



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### V

vison.analysis.ellipse, 45  
vison.analysis.Guyonnet15, 45  
vison.dark.BIAS01, 81  
vison.dark.DARK01, 82  
vison.datamodel.ccd, 35  
vison.datamodel.ccd\_aux, 37  
vison.datamodel.ccdsim, 38  
vison.datamodel.compliance, 38  
vison.datamodel.core, 39  
vison.datamodel.EXPLOGtools, 39  
vison.datamodel.generator, 40  
vison.datamodel.HKtools, 40  
vison.datamodel.inputs, 42  
vison.datamodel.QLAtools, 43  
vison.datamodel.scriptic, 43  
vison.eyegore.eyeCCDs, 56  
vison.eyegore.eyegore, 55  
vison.eyegore.eyeHK, 56  
vison.eyegore.eyeObs, 57  
vison.eyegore.eyeWarnings, 57  
vison.flat.FLAT0X, 83  
vison.flat.FlatTask, 51  
vison.flat.nl, 52  
vison.flat.NL01, 85  
vison.flat.ptc, 52  
vison.flat.PTC0X, 86  
vison.image.bits, 53  
vison.image.calibration, 53  
vison.image.cosmetics, 53  
vison.image.covariance, 53  
vison.image.ds9reg, 54  
vison.image.performance, 54  
vison.image.pixbounce, 54  
vison.image.sextractor, 54  
vison.inject.CHINJ01, 79  
vison.inject.CHINJ02, 80  
vison.inject.InjTask, 49  
vison.inject.lib, 50  
vison.inject.plot, 50  
vison.ogse.ogse, 59  
vison.other.MOT\_FF, 92  
vison.other.PERSIST01, 92  
vison.pipe.FlatFielding, 51  
vison.pipe.master, 9  
vison.pipe.task, 32  
vison.plot.baseplotclasses, 61  
vison.plot.figclasses, 62  
vison.plot.trends, 62  
vison.point.basis, 63  
vison.point.display, 63  
vison.point.FOCUS00, 88  
vison.point.gauss, 63  
vison.point.lib, 67  
vison.point.models, 64  
vison.point.photom, 64  
vison.point.PSF0X, 89  
vison.point.shape, 65  
vison.point.spot, 66  
vison.pump.TP01, 89  
vison.pump.TP02, 91  
vison.scripts.HKmonitor, 69  
vison.scripts.quickds9, 69  
vison.scripts.run\_xtalk, 69  
vison.scripts.v\_ROE\_LinCalib, 71  
vison.scripts.v\_ROETAB\_LinCalib, 71  
vison.scripts.vis\_cosmetics\_masker, 70  
vison.scripts.vis\_explogs\_merger, 70  
vison.scripts.vis\_genDataSet, 70  
vison.scripts.vis\_load\_DD, 70  
vison.scripts.vis\_mkscripts, 70  
vison.scripts.vis\_star\_finder, 71  
vison.support.context, 73  
vison.support.ET, 73  
vison.support.excel, 73  
vison.support.files, 74  
vison.support.flags, 74  
vison.support.latex, 74  
vison.support.logger, 75  
vison.support.report, 75  
vison.support.utils, 76  
vison.support.vistime, 76  
vison.support.vjson, 76  
vison.test.test\_ccd, 77

`vison.test.test_ccdpile`, [77](#)

## A

add\_data\_inventory\_to\_report() (vison.pipe.task.Task method), 33, 34  
 add\_extension() (vison.datamodel.ccd.CCD method), 35  
 add\_inputs\_to\_report() (vison.pipe.task.Task method), 33, 34  
 add\_labels\_to\_explog() (vison.pipe.task.Task method), 33, 34  
 add\_to\_Contents() (vison.support.report.Container method), 75  
 add\_to\_hist() (vison.datamodel.ccd.CCD method), 35  
 addColumn() (vison.datamodel.core.DataDict method), 39  
 addComplianceMatrix2Log() (vison.pipe.task.Task method), 33, 34  
 addComplianceMatrix2Report() (vison.pipe.task.Task method), 33, 34  
 addFigure2Report() (vison.pipe.task.Task method), 33, 34  
 addFigures\_ST() (vison.pipe.task.Task method), 33, 34  
 addFlagsToLog() (vison.pipe.task.Task method), 33, 34  
 addFlagsToReport() (vison.pipe.task.Task method), 33, 34  
 addHK\_2\_dd() (vison.pipe.task.Task method), 33, 34  
 addHKPlotsMatrix() (vison.pipe.task.Task method), 33, 34  
 addRow() (vison.datamodel.EXPLOGtools.ExpLogClass method), 39  
 area\_superellip() (in module vison.analysis.ellipse), 45

## B

basic\_analysis() (vison.dark.BIAS01.BIAS01 method), 81  
 basic\_analysis() (vison.dark.DARK01.DARK01 method), 82  
 basic\_analysis() (vison.inject.CHINJ02.CHINJ02 method), 80  
 basic\_analysis() (vison.other.PERSIST01.PERSIST01 method), 92  
 basic\_analysis() (vison.pipe.master.Pipe.BIAS01 method), 10, 21

basic\_analysis() (vison.pipe.master.Pipe.CHINJ02 method), 11, 23  
 basic\_analysis() (vison.pipe.master.Pipe.DARK01 method), 13, 24  
 basic\_analysis() (vison.pipe.master.Pipe.FOCUS00 method), 14, 26  
 basic\_analysis() (vison.pipe.master.Pipe.PERSIST01 method), 16, 28  
 basic\_analysis() (vison.pipe.master.Pipe.TP01 method), 19, 30  
 basic\_analysis() (vison.pipe.master.Pipe.TP02 method), 20, 31  
 basic\_analysis() (vison.point.FOCUS00.FOCUS00 method), 88  
 basic\_analysis() (vison.pump.TP01.TP01 method), 89  
 basic\_analysis() (vison.pump.TP02.TP02 method), 91  
 BasicPlot (class in vison.plot.baseplotclasses), 61  
 Beam1DHist (class in vison.plot.baseplotclasses), 61  
 BeamImgShow (class in vison.plot.baseplotclasses), 61  
 BeamPlot (class in vison.plot.baseplotclasses), 61  
 BeamPlotYvX (class in vison.plot.baseplotclasses), 61  
 BIAS01 (class in vison.dark.BIAS01), 81  
 bin\_img() (vison.datamodel.ccd\_aux.Model2D method), 38  
 bind\_buttons\_to\_methods() (vison.eyegore.eyJHK.HKFlags method), 56  
 BlueScreen (class in vison.plot.figclasses), 62  
 BROKEN\_basic\_analysis() (vison.inject.InjTask.InjTask method), 49  
 build\_cargo() (vison.datamodel.scriptic.Script method), 43  
 build\_data() (vison.plot.figclasses.BlueScreen method), 62  
 build\_elementList() (vison.eyegore.eyJObs.ExpLogDisplay method), 57  
 build\_scriptdict() (vison.dark.BIAS01.BIAS01 method), 82  
 build\_scriptdict() (vison.dark.DARK01.DARK01 method), 83  
 build\_scriptdict() (vison.flat.FLAT0X.FLAT0X method), 84

- build\_scripdict() (vison.flat.NL01.NL01 method), 85
  - build\_scripdict() (vison.flat.PTC0X.PTC0X method), 87
  - build\_scripdict() (vison.inject.CHINJ01.CHINJ01 method), 79
  - build\_scripdict() (vison.inject.CHINJ02.CHINJ02 method), 80
  - build\_scripdict() (vison.other.PERSIST01.PERSIST01 method), 92
  - build\_scripdict() (vison.pipe.master.Pipe.BF01 method), 9, 21
  - build\_scripdict() (vison.pipe.master.Pipe.BIAS01 method), 10, 22
  - build\_scripdict() (vison.pipe.master.Pipe.CHINJ00 method), 11, 22
  - build\_scripdict() (vison.pipe.master.Pipe.CHINJ01 method), 11, 22
  - build\_scripdict() (vison.pipe.master.Pipe.CHINJ02 method), 12, 23
  - build\_scripdict() (vison.pipe.master.Pipe.DARK01 method), 13, 24
  - build\_scripdict() (vison.pipe.master.Pipe.FLAT0X method), 13, 25
  - build\_scripdict() (vison.pipe.master.Pipe.FOCUS00 method), 14, 26
  - build\_scripdict() (vison.pipe.master.Pipe.NL01 method), 15, 26
  - build\_scripdict() (vison.pipe.master.Pipe.PERSIST01 method), 16, 28
  - build\_scripdict() (vison.pipe.master.Pipe.PTC0X method), 17, 29
  - build\_scripdict() (vison.pipe.master.Pipe.STRAY00 method), 18, 30
  - build\_scripdict() (vison.pipe.master.Pipe.TP00 method), 18, 30
  - build\_scripdict() (vison.pipe.master.Pipe.TP01 method), 19, 31
  - build\_scripdict() (vison.pipe.master.Pipe.TP02 method), 20, 32
  - build\_scripdict() (vison.pipe.task.Task method), 33, 34
  - build\_scripdict() (vison.point.FOCUS00.FOCUS00 method), 88
  - build\_scripdict() (vison.pump.TP01.TP01 method), 90
  - build\_scripdict() (vison.pump.TP02.TP02 method), 91
- C**
- catchtraceback() (vison.pipe.master.Pipe method), 21, 32
  - catchtraceback() (vison.pipe.task.Task method), 33, 34
  - CCD (class in vison.datamodel.ccd), 35
  - CCD2DPlot (class in vison.plot.baseplotclasses), 61
  - CCDPile (class in vison.datamodel.ccd), 37
  - changeColor() (vison.eyegore.eyehk.HKFlags method), 56
  - changeNumeric() (in module vison.eyegore.eyehk), 57
  - check\_data() (vison.flat.FlatTask.FlatTask method), 51
  - check\_data() (vison.inject.InjTask.InjTask method), 49
  - check\_data() (vison.other.PERSIST01.PERSIST01 method), 92
  - check\_data() (vison.pipe.master.Pipe.PERSIST01 method), 16, 28
  - check\_data() (vison.pipe.master.Pipe.TP00 method), 18, 30
  - check\_data() (vison.pipe.task.Task method), 33, 34
  - check\_HK() (vison.pipe.task.Task method), 33, 34
  - check\_HK\_abs() (in module vison.datamodel.HKtools), 40
  - check\_HK\_ST() (vison.pipe.task.Task method), 33, 34
  - check\_HK\_vs\_command() (in module vison.datamodel.HKtools), 41
  - check\_metrics\_ST() (vison.flat.FlatTask.FlatTask method), 51
  - check\_metrics\_ST() (vison.inject.InjTask.InjTask method), 49
  - check\_stat\_perCCD() (vison.pipe.task.Task method), 33, 34
  - check\_stat\_perCCDandCol() (vison.pipe.task.Task method), 33, 34
  - check\_stat\_perCCDandQ() (vison.pipe.task.Task method), 33, 34
  - check\_stat\_perCCDQandCol() (vison.pipe.task.Task method), 33, 34
  - CHINJ01 (class in vison.inject.CHINJ01), 79
  - CHINJ02 (class in vison.inject.CHINJ02), 80
  - circular2DGaussian() (vison.point.shape.Shapemeter method), 65
  - col\_has\_index() (vison.datamodel.core.DataDict method), 39
  - configure() (vison.plot.figclasses.BlueScreen method), 62
  - Container (class in vison.support.report), 75
  - Content (class in vison.support.report), 75
  - convert\_compl\_to\_nesteditemlist() (in module vison.datamodel.compliance), 38
  - convert\_fig\_to\_eps() (in module vison.support.files), 74
  - cooconv\_arrays\_decorate() (in module vison.datamodel.ccd), 37
  - cooconvert() (vison.datamodel.ccd.CCD method), 35
  - correct\_estic() (in module vison.analysis.Guyonnet15), 46
  - cPickleDump() (in module vison.support.files), 74
  - cPickleDumpDictionary() (in module vison.support.files), 74
  - cPickleRead() (in module vison.support.files), 74
  - create\_mockexplog() (vison.pipe.task.Task method), 33, 34
- D**
- DARK01 (class in vison.dark.DARK01), 82
  - DataDict (class in vison.datamodel.core), 39



- datasetGenerator() (in module vison.scripts.vis\_genDataSet), 70
- degrade\_estatic() (in module vison.analysis.Guyonnet15), 46
- del\_extension() (vison.datamodel.ccd.CCD method), 35
- dial\_numbers() (vison.support.ET.ET method), 73
- dissectFITS() (in module vison.datamodel.QLAtools), 43
- dist\_superellipse() (in module vison.analysis.ellipse), 45
- divide\_by\_flatfield() (vison.datamodel.ccd.CCD method), 36
- do\_indiv\_flats() (vison.flat.FLAT0X.FLAT0X method), 84
- do\_indiv\_flats() (vison.pipe.master.Pipe.FLAT0X method), 13, 25
- do\_Mask() (in module vison.scripts.vis\_cosmetics\_masker), 70
- do\_master\_flat() (vison.flat.FLAT0X.FLAT0X method), 84
- do\_master\_flat() (vison.pipe.master.Pipe.FLAT0X method), 14, 25
- do\_prdef\_mask() (vison.flat.FLAT0X.FLAT0X method), 84
- do\_prdef\_mask() (vison.pipe.master.Pipe.FLAT0X method), 14, 26
- do\_satCTE() (vison.flat.NL01.NL01 method), 85
- do\_satCTE() (vison.pipe.master.Pipe.NL01 method), 15, 26
- do\_Vscan\_Mask() (vison.datamodel.ccd.CCD method), 36
- doap\_photom() (vison.point.photom.Photometer method), 64
- doHKSsinglePlot() (in module vison.datamodel.HKtools), 41
- doPlot() (vison.pipe.task.Task method), 33, 34
- dotask() (vison.pipe.master.Pipe method), 21, 32
- dropColumn() (vison.datamodel.core.DataDict method), 39
- dummyrebin() (vison.datamodel.ccd.CCD method), 36
- dumps\_to\_json() (in module vison.support.vjson), 76
- ## E
- effective\_radius() (in module vison.analysis.ellipse), 45
- ellip2DGaussian() (vison.point.shape.Shapemeter method), 65
- ET (class in vison.support.ET), 73
- explog\_merger() (in module vison.scripts.vis\_explogs\_merger), 70
- ExpLogClass (class in vison.datamodel.EXPLOGtools), 39
- ExpLogDisplay (class in vison.eyegore.eyegore), 57
- Extension (class in vison.datamodel.ccd), 37
- extract() (vison.pipe.master.Pipe.TP01 method), 19, 31
- extract() (vison.pipe.master.Pipe.TP02 method), 20, 32
- extract() (vison.pump.TP01.TP01 method), 90
- extract() (vison.pump.TP02.TP02 method), 91
- extract\_BF() (vison.pipe.master.Pipe.BF01 method), 10, 21
- extract\_COV() (vison.pipe.master.Pipe.BF01 method), 10, 21
- extract\_data() (vison.inject.CHINJ02.CHINJ02 method), 81
- extract\_data() (vison.pipe.master.Pipe.CHINJ02 method), 12, 24
- extract\_HER() (vison.other.MOT\_FF.MOT\_FF method), 92
- extract\_HER() (vison.pipe.master.Pipe.MOT\_FF method), 15, 26
- extract\_PTC() (vison.flat.PTC0X.PTC0X method), 87
- extract\_PTC() (vison.pipe.master.Pipe.PTC0X method), 17, 29
- extract\_region() (in module vison.datamodel.ccd\_aux), 38
- extract\_region() (vison.datamodel.ccd.CCD method), 36
- extract\_spot() (in module vison.point.lib), 67
- extract\_stats() (vison.flat.NL01.NL01 method), 85
- extract\_stats() (vison.pipe.master.Pipe.NL01 method), 15, 27
- Eyegore (class in vison.eyegore.eyegore), 55
- ## F
- f\_get\_covmap() (in module vison.image.covariance), 53
- f\_text\_wrapper() (in module vison.support.logger), 75
- fgauss2D() (in module vison.point.models), 64
- FigsTable (class in vison.support.report), 75
- Figure (class in vison.support.report), 75
- filter\_img() (vison.datamodel.ccd\_aux.Model2D method), 38
- filter\_Voltage\_uni() (in module vison.scripts.v\_ROETAB\_LinCalib), 71
- filterexposures() (vison.dark.BIAS01.BIAS01 method), 82
- filterexposures() (vison.dark.DARK01.DARK01 method), 83
- filterexposures() (vison.flat.FLAT0X.FLAT0X method), 84
- filterexposures() (vison.flat.NL01.NL01 method), 86
- filterexposures() (vison.flat.PTC0X.PTC0X method), 87
- filterexposures() (vison.inject.CHINJ01.CHINJ01 method), 79
- filterexposures() (vison.inject.CHINJ02.CHINJ02 method), 81
- filterexposures() (vison.other.PERSIST01.PERSIST01 method), 93
- filterexposures() (vison.pipe.master.Pipe.BF01 method), 10, 21
- filterexposures() (vison.pipe.master.Pipe.BIAS01 method), 10, 22

filterexposures() (vison.pipe.master.Pipe.CHINJ00 method), 11, 22  
 filterexposures() (vison.pipe.master.Pipe.CHINJ01 method), 11, 22  
 filterexposures() (vison.pipe.master.Pipe.CHINJ02 method), 12, 24  
 filterexposures() (vison.pipe.master.Pipe.DARK01 method), 13, 24  
 filterexposures() (vison.pipe.master.Pipe.FLAT0X method), 14, 26  
 filterexposures() (vison.pipe.master.Pipe.FOCUS00 method), 15, 26  
 filterexposures() (vison.pipe.master.Pipe.NL01 method), 15, 27  
 filterexposures() (vison.pipe.master.Pipe.PERSIST01 method), 17, 28  
 filterexposures() (vison.pipe.master.Pipe.PTC0X method), 18, 29  
 filterexposures() (vison.pipe.master.Pipe.STRAY00 method), 18, 30  
 filterexposures() (vison.pipe.master.Pipe.TP00 method), 19, 30  
 filterexposures() (vison.pipe.master.Pipe.TP01 method), 19, 31  
 filterexposures() (vison.pipe.master.Pipe.TP02 method), 20, 32  
 filterexposures() (vison.pipe.task.Task method), 33, 34  
 filterexposures() (vison.point.FOCUS00.FOCUS00 method), 88  
 filterexposures() (vison.pump.TP01.TP01 method), 90  
 filterexposures() (vison.pump.TP02.TP02 method), 91  
 filtervalues() (in module vison.datamodel.HKtools), 41  
 find\_adu\_levels() (in module vison.scripts.v\_ROE\_LinCalib), 71  
 find\_discrete\_voltages\_inwaveform() (in module vison.scripts.v\_ROETAB\_LinCalib), 71  
 fit2D() (in module vison.pipe.FlatFielding), 51  
 fit2Dpol\_xyz() (vison.datamodel.ccd\_aux.Model2D method), 38  
 fit\_Gauss() (vison.point.gauss.Gaussmeter method), 64  
 fitNL() (in module vison.flat.nl), 52  
 fitPTC() (in module vison.flat.ptc), 52  
 Flags (class in vison.support.flags), 74  
 FLAT0X (class in vison.flat.FLAT0X), 83  
 FlatField (class in vison.pipe.FlatFielding), 51  
 FlatTask (class in vison.flat.FlatTask), 51  
 FOCUS00 (class in vison.point.FOCUS00), 88  
 foo\_bloom() (in module vison.flat.ptc), 52  
 fpred\_aijb() (in module vison.analysis.Guyonnet15), 46  
 frdist() (in module vison.analysis.Guyonnet15), 46  
 ftheta\_bound() (in module vison.analysis.Guyonnet15), 46  
 fun\_p() (in module vison.analysis.Guyonnet15), 47

## G

Gaussmeter (class in vison.point.gauss), 63  
 gen\_compliance\_tex() (in module vison.datamodel.compliance), 38  
 gen\_point\_mask() (in module vison.point.lib), 67  
 gen\_render() (vison.eyegore.eyeCCDs.ImageDisplay method), 56  
 generate\_Explog() (in module vison.datamodel.generator), 40  
 generate\_FITS() (in module vison.datamodel.generator), 40  
 generate\_FITS\_fromExpLog() (in module vison.datamodel.generator), 40  
 generate\_GaussPSF() (in module vison.analysis.Guyonnet15), 47  
 generate\_header() (in module vison.support.latex), 74  
 generate\_HK() (in module vison.datamodel.generator), 40  
 generate\_Latex() (vison.support.report.FigsTable method), 75  
 generate\_Latex() (vison.support.report.Figure method), 75  
 generate\_Latex() (vison.support.report.Section method), 75  
 generate\_Latex() (vison.support.report.Table method), 76  
 generate\_Latex() (vison.support.report.Text method), 76  
 genExpLog() (in module vison.scripts.vis\_genDataSet), 70  
 get\_1Dprofile() (in module vison.datamodel.ccd\_aux), 38  
 get\_1Dprofile() (vison.datamodel.ccd.CCD method), 36  
 get\_body\_circles() (in module vison.image.ds9reg), 54  
 get\_body\_ellipses() (in module vison.image.ds9reg), 54  
 get\_centroid() (vison.point.photom.Photometer method), 64  
 get\_checkstats\_ST() (vison.flat.FlatTask.FlatTask method), 51  
 get\_checkstats\_ST() (vison.inject.InjTask.InjTask method), 50  
 get\_cov\_maps() (in module vison.image.covariance), 53  
 get\_cross\_shape\_rough() (in module vison.analysis.Guyonnet15), 47  
 get\_cutout() (vison.datamodel.ccd.CCD method), 36  
 get\_data() (vison.eyegore.eyeHK.HKDisplay method), 56  
 get\_data() (vison.eyegore.eyeObs.ExpLogDisplay method), 57  
 get\_deltaQ() (in module vison.analysis.Guyonnet15), 47  
 get\_dtobj() (in module vison.support.vistime), 76  
 get\_execution\_summary() (vison.pipe.master.Pipe method), 21, 32  
 get\_exptime\_atmidynrange() (in module vison.flat.nl), 52  
 get\_FluenceAndGradient\_limits() (vison.inject.InjTask.InjTask method), 49  
 get\_FW\_ID() (in module vison.ogse.ogse), 59

[get\\_ilum\(\)](#) (in module `vison.pipe.FlatFielding`), 51  
[get\\_ilum\\_splines\(\)](#) (in module `vison.pipe.FlatFielding`), 51  
[get\\_kernel\(\)](#) (in module `vison.analysis.Guyonnet15`), 47  
[get\\_mask\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_model\\_poly2D\(\)](#) (`vison.datamodel.ccd_aux.Model2D` method), 38  
[get\\_model\\_splines\(\)](#) (`vison.datamodel.ccd_aux.Model2D` method), 38  
[get\\_offsets\\_lims\(\)](#) (in module `vison.image.performance`), 54  
[get\\_perf\\_rdout\(\)](#) (in module `vison.image.performance`), 54  
[get\\_photom\(\)](#) (`vison.point.spot.Spot` method), 66  
[get\\_pixbounce\\_from\\_overscan\(\)](#) (in module `vison.image.pixbounce`), 54  
[get\\_Q\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_quad\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_Rdisp\(\)](#) (in module `vison.analysis.Guyonnet15`), 47  
[get\\_region2Dmodel\(\)](#) (in module `vison.datamodel.ccd_aux`), 38  
[get\\_region2Dmodel\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_shape\\_easy\(\)](#) (`vison.point.spot.Spot` method), 66  
[get\\_shape\\_Gauss\(\)](#) (`vison.point.spot.Spot` method), 66  
[get\\_shape\\_Moments\(\)](#) (`vison.point.spot.Spot` method), 66  
[get\\_stats\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_struct\\_from\\_cargo\(\)](#) (`vison.datamodel.scriptic.Script` method), 43  
[get\\_test\(\)](#) (`vison.pipe.master.Pipe` method), 21, 32  
[get\\_Thresholding\\_DefectsMask\(\)](#) (in module `vison.image.cosmetics`), 53  
[get\\_tile\\_coos\(\)](#) (`vison.datamodel.ccd.CCD` method), 36  
[get\\_tiles\(\)](#) (`vison.datamodel.ccd.CCD` method), 37  
[get\\_tiles\\_stats\(\)](#) (`vison.datamodel.ccd.CCD` method), 37  
[get\\_time\\_tag\(\)](#) (in module `vison.support.vistime`), 76  
[getacrosscolscut\(\)](#) (in module `vison.datamodel.QLAtools`), 43  
[getacrossrowscut\(\)](#) (in module `vison.datamodel.QLAtools`), 43  
[getsectioncollims\(\)](#) (`vison.datamodel.ccd.CCD` method), 37  
[getsectionrowlims\(\)](#) (`vison.datamodel.ccd.CCD` method), 37  
[getsectionstats\(\)](#) (in module `vison.datamodel.QLAtools`), 43  
[grab\\_numbers\\_and\\_codes\(\)](#) (in module `vison.support.ET`), 73

## H

[HKDisplay](#) (class in `vison.eyegore.eyeHK`), 56  
[HKFlags](#) (class in `vison.eyegore.eyeHK`), 56

## I

[ImageDisplay](#) (class in `vison.eyegore.eyeCCDs`), 56  
[IMG\\_bias\\_gen\(\)](#) (in module `vison.datamodel.generator`), 40  
[IMG\\_chinj\\_gen\(\)](#) (in module `vison.datamodel.generator`), 40  
[IMG\\_flat\\_gen\(\)](#) (in module `vison.datamodel.generator`), 40  
[IMG\\_point\\_gen\(\)](#) (in module `vison.datamodel.generator`), 40  
[ImgShow](#) (class in `vison.plot.baseplotclasses`), 61  
[iniExplog\(\)](#) (in module `vison.datamodel.EXPLOGtools`), 39  
[iniExplog\(\)](#) (`vison.datamodel.EXPLOGtools.ExpLogClass` method), 39  
[iniHK\\_QFM\(\)](#) (in module `vison.datamodel.HKtools`), 41  
[initColumn\(\)](#) (`vison.datamodel.core.DataDict` method), 39  
[InjTask](#) (class in `vison.inject.InjTask`), 49  
[Inputs](#) (class in `vison.datamodel.inputs`), 42  
[IsComplianceMatrixOK\(\)](#) (`vison.pipe.task.Task` method), 33  
[isflagraised\(\)](#) (`vison.eyegore.eyeHK.HKFlags` method), 56  
[isNumeric\(\)](#) (in module `vison.eyegore.eyeObs`), 57

## L

[launchtask\(\)](#) (`vison.pipe.master.Pipe` method), 21, 32  
[load\(\)](#) (`vison.datamodel.scriptic.Script` method), 43  
[load\\_catalog\(\)](#) (`vison.image.sextractor.VSExtractor` method), 54  
[load\\_FITS\\_CDPs\(\)](#) (in module `vison.image.calibration`), 53  
[load\\_jsonfile\(\)](#) (in module `vison.support.vjson`), 76  
[load\\_to\\_cargo\(\)](#) (`vison.datamodel.scriptic.Script` method), 43  
[load\\_WF\(\)](#) (in module `vison.scripts.v_ROETAB_LinCalib`), 71  
[loadExpLog\(\)](#) (in module `vison.datamodel.EXPLOGtools`), 39  
[loadExpLog\(\)](#) (`vison.datamodel.core.DataDict` method), 39  
[loadExplogs\(\)](#) (`vison.eyegore.eyeObs.ExpLogDisplay` method), 57  
[loadfromFITS\(\)](#) (`vison.datamodel.ccd.CCD` method), 37  
[loadHK\\_preQM\(\)](#) (in module `vison.datamodel.HKtools`), 42  
[loadHK\\_QFM\(\)](#) (in module `vison.datamodel.HKtools`), 41  
[loadHK\\_QFMsingle\(\)](#) (in module `vison.datamodel.HKtools`), 41  
[lock\\_on\\_stars\(\)](#) (`vison.pipe.master.Pipe.FOCUS00` method), 15, 26

lock\_on\_stars() (vison.point.FOCUS00.FOCUS00 method), 88  
 lowerflag() (vison.eyegore.eyeHK.HKFlags method), 56

## M

measure\_basic() (vison.point.spot.Spot method), 66  
 measure\_bgd() (vison.point.photom.Photometer method), 64  
 measureRefinedEllipticity() (vison.point.shape.Shapemeter method), 65  
 merge\_HKfiles() (in module vison.datamodel.generator), 40  
 mergeExpLogs() (in module vison.datamodel.EXPLOGtools), 39  
 mergeHK() (in module vison.datamodel.HKtools), 42  
 meta\_analysis() (vison.dark.BIAS01.BIAS01 method), 82  
 meta\_analysis() (vison.dark.DARK01.DARK01 method), 83  
 meta\_analysis() (vison.flat.PTC0X.PTC0X method), 87  
 meta\_analysis() (vison.inject.CHINJ01.CHINJ01 method), 79  
 meta\_analysis() (vison.inject.CHINJ02.CHINJ02 method), 81  
 meta\_analysis() (vison.other.PERSIST01.PERSIST01 method), 93  
 meta\_analysis() (vison.pipe.master.Pipe.BF01 method), 10, 21  
 meta\_analysis() (vison.pipe.master.Pipe.BIAS01 method), 10, 22  
 meta\_analysis() (vison.pipe.master.Pipe.CHINJ01 method), 11, 22  
 meta\_analysis() (vison.pipe.master.Pipe.CHINJ02 method), 12, 24  
 meta\_analysis() (vison.pipe.master.Pipe.DARK01 method), 13, 24  
 meta\_analysis() (vison.pipe.master.Pipe.FOCUS00 method), 15, 26  
 meta\_analysis() (vison.pipe.master.Pipe.PERSIST01 method), 17, 29  
 meta\_analysis() (vison.pipe.master.Pipe.PTC0X method), 18, 29  
 meta\_analysis() (vison.pipe.master.Pipe.TP01 method), 20, 31  
 meta\_analysis() (vison.pipe.master.Pipe.TP02 method), 20, 32  
 meta\_analysis() (vison.point.FOCUS00.FOCUS00 method), 88  
 meta\_analysis() (vison.pump.TP01.TP01 method), 90  
 meta\_analysis() (vison.pump.TP02.TP02 method), 91  
 Model2D (class in vison.datamodel.ccd\_aux), 37  
 MOT\_FF (class in vison.other.MOT\_FF), 92  
 MuteFlag() (vison.eyegore.eyeHK.HKFlags method), 56

## N

name\_indices() (vison.datamodel.core.DataDict method), 39  
 name\_indices() (vison.datamodel.core.vColumn method), 39  
 NL01 (class in vison.flat.NL01), 85

## O

old\_basic\_analysis() (vison.inject.CHINJ01.CHINJ01 method), 79  
 old\_basic\_analysis() (vison.pipe.master.Pipe.CHINJ01 method), 11, 23

## P

parse\_fits() (vison.pipe.FlatFielding.FlatField method), 51  
 parseDTstr() (in module vison.datamodel.HKtools), 42  
 parseHKfiles() (in module vison.datamodel.HKtools), 42  
 parseHKfname() (in module vison.datamodel.HKtools), 42  
 PERSIST01 (class in vison.other.PERSIST01), 92  
 Photometer (class in vison.point.photom), 64  
 Pipe (class in vison.pipe.master), 9, 21  
 Pipe.BF01 (class in vison.pipe.master), 9, 21  
 Pipe.BIAS01 (class in vison.pipe.master), 10, 21  
 Pipe.CHINJ00 (class in vison.pipe.master), 11, 22  
 Pipe.CHINJ01 (class in vison.pipe.master), 11, 22  
 Pipe.CHINJ02 (class in vison.pipe.master), 11, 23  
 Pipe.DARK01 (class in vison.pipe.master), 12, 24  
 Pipe.FLAT0X (class in vison.pipe.master), 13, 25  
 Pipe.FOCUS00 (class in vison.pipe.master), 14, 26  
 Pipe.MOT\_FF (class in vison.pipe.master), 15, 26  
 Pipe.NL01 (class in vison.pipe.master), 15, 26  
 Pipe.PERSIST01 (class in vison.pipe.master), 16, 28  
 Pipe.PTC0X (class in vison.pipe.master), 17, 29  
 Pipe.STRAY00 (class in vison.pipe.master), 18, 30  
 Pipe.TP00 (class in vison.pipe.master), 18, 30  
 Pipe.TP01 (class in vison.pipe.master), 19, 30  
 Pipe.TP02 (class in vison.pipe.master), 20, 31  
 plot\_map() (in module vison.analysis.Guyonnet15), 47  
 plot\_maps\_ftheta() (in module vison.analysis.Guyonnet15), 47  
 plot\_waveform() (in module vison.scripts.v\_ROETAB\_LinCalib), 71  
 plotAcCOLcuts() (in module vison.datamodel.QLAtools), 43  
 plotAcROWcuts() (in module vison.datamodel.QLAtools), 43  
 plotQuads() (in module vison.datamodel.QLAtools), 43  
 plt\_trimmer() (vison.plot.baseplotclasses.ImgShow method), 61  
 populate\_axes() (vison.plot.baseplotclasses.BeamPlot method), 61

populate\_axes() (vison.plot.baseplotclasses.ImgShow method), 61

pre\_process() (in module vison.scripts.vis\_cosmetics\_masker), 70

predict\_expected\_injlevels() (vison.inject.InjTask.InjTask method), 50

prep\_data() (vison.dark.BIAS01.BIAS01 method), 82

prep\_data() (vison.dark.DARK01.DARK01 method), 83

prep\_data() (vison.flat.NL01.NL01 method), 86

prep\_data() (vison.other.PERSIST01.PERSIST01 method), 93

prep\_data() (vison.pipe.master.Pipe.BIAS01 method), 11, 22

prep\_data() (vison.pipe.master.Pipe.DARK01 method), 13, 25

prep\_data() (vison.pipe.master.Pipe.FOCUS00 method), 15, 26

prep\_data() (vison.pipe.master.Pipe.NL01 method), 16, 27

prep\_data() (vison.pipe.master.Pipe.PERSIST01 method), 17, 29

prep\_data() (vison.point.FOCUS00.FOCUS00 method), 88

prepare\_images() (vison.flat.FLAT0X.FLAT0X method), 84

prepare\_images() (vison.inject.InjTask.InjTask method), 50

prepare\_images() (vison.pipe.master.Pipe.FLAT0X method), 14, 26

prepare\_images() (vison.pipe.task.Task method), 33, 34

produce\_IndivFlats() (in module vison.pipe.FlatFielding), 51

produce\_MasterFlat() (in module vison.pipe.FlatFielding), 52

produce\_NLCs() (vison.flat.NL01.NL01 method), 86

produce\_NLCs() (vison.pipe.master.Pipe.NL01 method), 16, 27

produce\_SingleFlatfield() (in module vison.pipe.FlatFielding), 52

Profile1D (class in vison.datamodel.ccd\_aux), 38

PTCOX (class in vison.flat.PTCOX), 87

## Q

quadrupoles() (vison.point.shape.Shapemeter method), 65

## R

raiseflag() (vison.eyegore.eyeyeHK.HKFlags method), 56

read\_OBSID\_list() (in module vison.scripts.vis\_cosmetics\_masker), 70

rebin() (in module vison.datamodel.ccd\_aux), 38

recover\_progress() (vison.pipe.task.Task method), 33, 34

removescalars\_from\_dict() (in module vison.datamodel.compliance), 38

replace\_in\_template() (in module vison.support.latex), 74

reportFITS() (in module vison.datamodel.QLAtools), 43

reportHK() (in module vison.datamodel.HKtools), 42

ResetFlag() (vison.eyegore.eyeyeHK.HKFlags method), 56

rsync\_to\_altlocalpath() (in module vison.eyegore.eyegore), 56

rsync\_to\_remote() (in module vison.eyegore.eyegore), 56

run() (vison.pipe.master.Pipe method), 21, 32

run\_maskmaker() (in module vison.scripts.vis\_cosmetics\_masker), 70

run\_ROE\_LinCalib() (in module vison.scripts.v\_ROE\_LinCalib), 71

run\_ROETAB\_LinCalib() (in module vison.scripts.v\_ROETAB\_LinCalib), 71

run\_SEx() (vison.image.sextractor.VSExtractor method), 54

run\_xtalk() (in module vison.scripts.run\_xtalk), 69

## S

save\_img\_to\_tmp() (vison.image.sextractor.VSExtractor method), 54

save\_jsonfile() (in module vison.support.vjson), 76

save\_progress() (vison.pipe.task.Task method), 33, 34

save\_spots\_as\_ds9regs() (in module vison.image.ds9reg), 54

saveToFile() (vison.datamodel.core.DataDict method), 39

Script (class in vison.datamodel.scriptic), 43

search\_EXPLOGs() (vison.eyegore.eyeyeObs.ExpLogDisplay method), 57

search\_HKfiles() (vison.eyegore.eyeyeHK.HKDisplay method), 56

Section (class in vison.support.report), 75

select\_HKkeys() (vison.eyegore.eyeyeHK.HKDisplay method), 56

send\_sms() (vison.support.ET.ET method), 73

set\_extension() (vison.datamodel.ccd.CCD method), 37

set\_inpdefaults() (vison.flat.FLAT0X.FLAT0X method), 85

set\_inpdefaults() (vison.flat.PTCOX.PTCOX method), 88

set\_inpdefaults() (vison.inject.CHINJ01.CHINJ01 method), 80

set\_inpdefaults() (vison.inject.CHINJ02.CHINJ02 method), 81

set\_inpdefaults() (vison.other.PERSIST01.PERSIST01 method), 93

set\_inpdefaults() (vison.pipe.master.Pipe.BF01 method), 10, 21

set\_inpdefaults() (vison.pipe.master.Pipe.CHINJ00 method), 11, 22

set\_inpdefaults() (vison.pipe.master.Pipe.CHINJ01 method), 11, 23

set\_inpdefaults() (vison.pipe.master.Pipe.CHINJ02 method), 12, 24

- set\_inpdefaults() (vison.pipe.master.Pipe.FLAT0X method), 14, 26  
 set\_inpdefaults() (vison.pipe.master.Pipe.PERSIST01 method), 17, 29  
 set\_inpdefaults() (vison.pipe.master.Pipe.PTC0X method), 18, 30  
 set\_inpdefaults() (vison.pipe.master.Pipe.STRAY00 method), 18, 30  
 set\_inpdefaults() (vison.pipe.master.Pipe.TP00 method), 19, 30  
 set\_inpdefaults() (vison.pipe.master.Pipe.TP01 method), 20, 31  
 set\_inpdefaults() (vison.pipe.master.Pipe.TP02 method), 21, 32  
 set\_inpdefaults() (vison.pump.TP01.TP01 method), 90  
 set\_inpdefaults() (vison.pump.TP02.TP02 method), 91  
 set\_quad() (vison.datamodel.ccd.CCD method), 37  
 setup\_fig() (vison.eyegore.eyegore.ImageDisplay method), 56  
 setup\_MasterWG() (vison.eyegore.eyegore.Eyegore method), 56  
 setUpLogger() (in module vison.support.logger), 75  
 Shapemeter (class in vison.point.shape), 65  
 show\_disps\_CCD273() (in module vison.analysis.Guyonnet15), 47  
 show\_spots\_allCCDs() (in module vison.point.display), 63  
 sim\_window() (in module vison.datamodel.ccdsim), 38  
 sim\_window() (vison.datamodel.ccd.CCD method), 37  
 simadd\_flatilim() (vison.datamodel.ccd.CCD method), 37  
 simadd\_flatilum() (in module vison.datamodel.ccdsim), 38  
 simadd\_points() (in module vison.datamodel.ccdsim), 38  
 simadd\_points() (vison.datamodel.ccd.CCD method), 37  
 simadd\_poisson() (in module vison.datamodel.ccdsim), 38  
 simadd\_poisson() (vison.datamodel.ccd.CCD method), 37  
 simadd\_ron() (in module vison.datamodel.ccdsim), 38  
 simadd\_ron() (vison.datamodel.ccd.CCD method), 37  
 SimpleLogger (class in vison.support.logger), 75  
 SingleHKplot (class in vison.eyegore.eyegore.eyehk), 56  
 skipMissingPlot() (vison.pipe.task.Task method), 33, 34  
 solve\_for\_A\_linalg() (in module vison.analysis.Guyonnet15), 47  
 solve\_for\_psmooth() (in module vison.analysis.Guyonnet15), 48  
 sort\_HKfiles() (in module vison.eyegore.eyegore.eyehk), 56  
 sortBy() (vison.eyegore.eyegore.eyehk.ExpLogDisplay method), 57  
 Spot (class in vison.point.spot), 66  
 SpotBase (class in vison.point.basis), 63  
 stack() (vison.datamodel.ccd.CCDPile method), 37  
 sub\_bgd() (vison.point.photom.Photometer method), 64  
 sub\_bias() (vison.datamodel.ccd.CCD method), 37  
 sub\_offset() (vison.datamodel.ccd.CCD method), 37  
 summary() (vison.datamodel.EXPLOGtools.ExpLogClass method), 39  
 synthHK() (in module vison.datamodel.HKtools), 42
- ## T
- Table (class in vison.support.report), 75  
 Task (class in vison.pipe.task), 32, 33  
 Test (class in vison.dark.BIAS01), 82  
 test() (in module vison.datamodel.EXPLOGtools), 39  
 test0() (in module vison.analysis.Guyonnet15), 48  
 test0() (in module vison.datamodel.scriptic), 44  
 test0() (in module vison.support.excel), 73  
 test\_check\_data() (vison.dark.BIAS01.Test method), 82  
 test\_create\_from\_scratch() (in module vison.datamodel.ccd), 37  
 test\_getkernel() (in module vison.analysis.Guyonnet15), 48  
 test\_load\_ELVIS\_fits() (in module vison.datamodel.ccd), 37  
 test\_selfconsist() (in module vison.analysis.Guyonnet15), 48  
 test\_solve() (in module vison.analysis.Guyonnet15), 48  
 test\_URLs() (in module vison.eyegore.eyegore.eyehk), 57  
 test\_wrap\_fitNL() (in module vison.flat.nl), 52  
 testBeam2ImgShow() (in module vison.plot.baseplotclasses), 61  
 TestEllipse (class in vison.analysis.ellipse), 45  
 Text (class in vison.support.report), 76  
 TP01 (class in vison.pump.TP01), 89  
 TP02 (class in vison.pump.TP02), 91
- ## U
- UnmuteFlag() (vison.eyegore.eyegore.eyehk.HKFlags method), 56  
 update\_structdict() (in module vison.datamodel.scriptic), 44  
 useCases() (in module vison.datamodel.core), 39
- ## V
- validate() (vison.datamodel.scriptic.Script method), 43  
 validate\_within\_HKlim() (in module vison.eyegore.eyegore.eyehk), 56  
 vColumn (class in vison.datamodel.core), 39  
 vIndex (class in vison.datamodel.core), 39  
 vison.analysis.ellipse (module), 45  
 vison.analysis.Guyonnet15 (module), 45  
 vison.dark.BIAS01 (module), 81  
 vison.dark.DARK01 (module), 82  
 vison.datamodel.ccd (module), 35  
 vison.datamodel.ccd\_aux (module), 37  
 vison.datamodel.ccdsim (module), 38  
 vison.datamodel.compliance (module), 38

vison.datamodel.core (module), 39  
 vison.datamodel.EXPLOGtools (module), 39  
 vison.datamodel.generator (module), 40  
 vison.datamodel.HKtools (module), 40  
 vison.datamodel.inputs (module), 42  
 vison.datamodel.QLAtools (module), 43  
 vison.datamodel.scriptic (module), 43  
 vison.eyegore.eyecCDs (module), 56  
 vison.eyegore.eyegore (module), 55  
 vison.eyegore.eyehK (module), 56  
 vison.eyegore.eyebObs (module), 57  
 vison.eyegore.eyebWarnings (module), 57  
 vison.flat.FLAT0X (module), 83  
 vison.flat.FlatTask (module), 51  
 vison.flat.nl (module), 52  
 vison.flat.NL01 (module), 85  
 vison.flat.ptc (module), 52  
 vison.flat.PTC0X (module), 86  
 vison.image.bits (module), 53  
 vison.image.calibration (module), 53  
 vison.image.cosmetics (module), 53  
 vison.image.covariance (module), 53  
 vison.image.ds9reg (module), 54  
 vison.image.performance (module), 54  
 vison.image.pixbounce (module), 54  
 vison.image.sextractor (module), 54  
 vison.inject.CHINJ01 (module), 79  
 vison.inject.CHINJ02 (module), 80  
 vison.inject.InjTask (module), 49  
 vison.inject.lib (module), 50  
 vison.inject.plot (module), 50  
 vison.ogse.ogse (module), 59  
 vison.other.MOT\_FF (module), 92  
 vison.other.PERSIST01 (module), 92  
 vison.pipe.FlatFielding (module), 51  
 vison.pipe.master (module), 9  
 vison.pipe.task (module), 32  
 vison.plot.baseplotclasses (module), 61  
 vison.plot.figclasses (module), 62  
 vison.plot.trends (module), 62  
 vison.point.basis (module), 63  
 vison.point.display (module), 63  
 vison.point.FOCUS00 (module), 88  
 vison.point.gauss (module), 63  
 vison.point.lib (module), 67  
 vison.point.models (module), 64  
 vison.point.photom (module), 64  
 vison.point.PSFOX (module), 89  
 vison.point.shape (module), 65  
 vison.point.spot (module), 66  
 vison.pump.TP01 (module), 89  
 vison.pump.TP02 (module), 91  
 vison.scripts.HKmonitor (module), 69  
 vison.scripts.quickds9 (module), 69  
 vison.scripts.run\_xtalk (module), 69  
 vison.scripts.v\_ROE\_LinCalib (module), 71  
 vison.scripts.v\_ROETAB\_LinCalib (module), 71  
 vison.scripts.vis\_cosmetics\_masker (module), 70  
 vison.scripts.vis\_explogs\_merger (module), 70  
 vison.scripts.vis\_genDataSet (module), 70  
 vison.scripts.vis\_load\_DD (module), 70  
 vison.scripts.vis\_mkscripts (module), 70  
 vison.scripts.vis\_star\_finder (module), 71  
 vison.support.context (module), 73  
 vison.support.ET (module), 73  
 vison.support.excel (module), 73  
 vison.support.files (module), 74  
 vison.support.flags (module), 74  
 vison.support.latex (module), 74  
 vison.support.logger (module), 75  
 vison.support.report (module), 75  
 vison.support.utils (module), 76  
 vison.support.vistime (module), 76  
 vison.support.vjson (module), 76  
 vison.test.test\_ccd (module), 77  
 vison.test.test\_ccdpile (module), 77  
 VSExtractor (class in vison.image.sextractor), 54

## W

wait\_and\_run() (vison.pipe.master.Pipe method), 21, 32  
 wrap\_fitNL() (in module vison.flat.nl), 52  
 write() (vison.datamodel.scriptic.Script method), 44  
 write() (vison.support.logger.SimpleLogger method), 75  
 write\_ID\_chart() (in module vison.scripts.vis\_star\_finder), 71  
 writeFITS() (vison.point.shape.Shapemeter method), 66  
 writeto() (vison.datamodel.ccd.CCD method), 37  
 writeto() (vison.datamodel.EXPLOGtools.ExpLogClass method), 39