vison Documentation

Release 0.1

Ruyman Azzollini

Apr 28, 2017

CONTENTS

Contents:

Author Ruyman Azzollini

Contact r.azzollini_at_ucl.ac.uk

issue 0.1

version 0.1

date Apr 28, 2017

This Python package "vison" is the pipeline that will be used at MSSL for ground calibration of the VIS detection chains, including one ROE, one RPSU and three CCDs.

CHAPTER

ONE

INSTALLATION

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

https://github.com/ruymanengithub/vison

Detailed instructions: TBW

1.1 Dependencies

A copy of the conda environment that provides all dependencies will be included in the package.

CHAPTER

TWO

PIPELINE CORE

2.1 Pipeline

2.1.1 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 directory (TBC).
- 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

contact r.azzollini_at_ucl.ac.uk

class vison.pipe.master.**Pipe**(*inputdict*, *dolog=True*)
Master Class of FM-analysis

run()

2.1.2 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*)

CHAPTER

THREE

DATA MODEL

3.1 Data Model

3.1.1 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*)
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.

add_extension(*data*, *header=None*, *label=None*, *headerdict=None*)

divide_by_flatfield(*FF*, *extension=-1*)
Divides by a Flat-field

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_stats (*Quadrant, sector='img', statkeys=['mean'], trimscan=[0, 0], extension=-1*)

getsectioncollims (*QUAD*)
Returns limits of sections: prescan, image and overscan

set_quad (*inQdata*, *Quadrant*, *canonical=False*, *extension=-1*)

simadd_flatilum (*levels={'H': 0.0*, *'E': 0.0*, *'G': 0.0*, *'F': 0.0}*, *extension=-1*)

simadd_points (*flux*, *fwhm*, *CCDID='CCD1'*, *dx=0*, *dy=0*, *extension=-1*)

simadd_poisson (*extension=-1*)

simadd_ron (*extension=-1*)

sub_bias (*superbias*, *extension=-1*)
Subtracts a superbias

sub_offset (*Quad, method='row', scan='pre', trimscan=[3, 2], extension=-1*)

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

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

vison.datamodel.ccd.**test_create_from_scratch** ()

vison.datamodel.ccd.**test_load_ELVIS_fits** ()

3.1.2 EXPLOGtools.py

class vison.datamodel.EXPLOGtools.**ExpLogClass** (*elvis='5.7.04'*)

addRow ()

iniExplog ()

writeto (*outfile*)

vison.datamodel.EXPLOGtools.**iniExplog** (*elvis*)

vison.datamodel.EXPLOGtools.**loadExpLog** (*expfile*, *elvis='5.7.04'*)
Loads an Exposure Log from file.

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

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

3.1.3 HKtools.py

House-Keeping inspection and handling tools.

History

Created on Thu Mar 10 12:11:58 2016

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

vison.datamodel.HKtools.**HKplot**(*allHKdata*, *keylist*, *key*, *dtobjs*, *filename=''*, *stat='mean'*)
Plots the values of a HK parameter as a function of time.

Parameters

- **allHKdata** – HKdata = [(nfiles,nstats,nHKparams)]
- **keylist** – list with all HK keys.
- **key** – selected key.
- **tdeltahour** – time axis.

Returns None!!

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

vison.datamodel.HKtools.**iniHK_QFM**(*elvis='6.0.0'*)

vison.datamodel.HKtools.**loadHK_QFM**(*filename*, *elvis='5.8.X'*)
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
- **form** – format of HK file, given by version of "ELVIS"

Returns dictionary 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
- **form** – format of HK file, given by version of "ELVIS"

Returns dictionary with pairs parameter:[values]

vison.datamodel.HKtools.**parseHKfiles**(*HKlist*, *elvis='5.7.07'*)

Parameters **HKlist** – list of HK files (path+name).

Returns [obsids],[dtobjs],[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*)

Synthetizes 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]

3.1.4 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=''*)

CHAPTER

FOUR

ANALYSIS (SHARED)

4.1 Analysis (Shared)

4.1.1 ellipse.py

Auxiliary module with functions to generate generalized ellipse masks.

author Ruyman Azzollini

contact r.azzollini@ucl.ac.uk

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*, *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 in '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)
- **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=0*, *c=0*)
Returns semi-major axis length of superellipse, given the area

CHAPTER

FIVE

MONITORING (“EYEGORE”)

5.1 Eyegore

5.1.1 eyegore.py

CHAPTER

SIX

POINT-SOURCE ANALYSIS

6.1 Point-Source Analysis

6.1.1 FOCUS00

VIS Ground Calibration 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

contact r.azzollini_at_ucl.ac.uk

`vison.point.FOCUS00.`**`basic_analysis_FOCUS00`**(*DataDict*, *Report*, *inputs*, *log=None*, *debug=False*)

Performs basic analysis on spots: - 2D Gaussian Model shape measurements - Quadrupole Moments shape measurements

vison.point.FOCUS00.**filterexposures_FOCUS00**(*inwavelength*, *explogf*, *datapath*, *OBSID_lims*, *structure={'Ncols': 6, 'col6': {'Mirr_pos': 70.2, 'Exptime': 10.0, 'N': 2}, 'col4': {'Mirr_pos': 70.0, 'Exptime': 10.0, 'N': 2}, 'col5': {'Mirr_pos': 70.1, 'Exptime': 10.0, 'N': 2}, 'col2': {'Mirr_pos': 69.8, 'Exptime': 10.0, 'N': 2}, 'col3': {'Mirr_pos': 69.9, 'Exptime': 10.0, 'N': 2}, 'col1': {'Mirr_pos': 69.8, 'Exptime': 0, 'N': 5}}*, *elvis='5.7.04'*)

Loads a list of Exposure Logs and selects exposures from test FOCUS00.

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.

vison.point.FOCUS00.**generate_Explog_FOCUS00**(*wavelength*, *struct*, *elvis='6.0.0'*, *date=datetime.datetime(1980, 2, 21, 7, 0)*)

vison.point.FOCUS00.**generate_FITS_FOCUS00**(*wavelength*, *explog*, *datapath*, *elvis='6.0.0'*)

vison.point.FOCUS00.**generate_Fake_FOCUS00**(*wavelength*, *date=datetime.datetime(1980, 2, 21, 7, 0)*, *rootpath=''*)

Generates Fake FOCUS00 data

vison.point.FOCUS00.**generate_HK_FOCUS00**(*explog*, *datapath*, *elvis='6.0.0'*)

vison.point.FOCUS00.**get_FOCUS00_structure**(*wavelength*)

vison.point.FOCUS00.**get_basic_spot_FOCUS00**(*stamp*, *x0*, *y0*, *log=None*, *debug=False*)

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

vison.point.FOCUS00.**get_shape_spot_FOCUS00**(*stamp*, *x0*, *y0*, *method='G'*, *log=None*, *debug=False*)

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

vison.point.FOCUS00.**meta_analysis_FOCUS00**(*DataDict*, *RepDict*, *inputs*, *log=None*)

Analyzes the relation between PSF shape and mirror position.

vison.point.FOCUS00.**prep_data_FOCUS00**(*DataDict*, *Report*, *inputs*, *log=None*, *debug=False*)

Takes Raw Data and prepares it for further analysis. Also checks that data quality is enough.

vison.point.FOCUS00.**run**(*inputs*, *log=None*)

Test FOCUS00 master function.

6.1.2 PSF0X

VIS Ground Calibration 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

contact r.azzollini_at_ucl.ac.uk

vison.point.PSF0X.**basic_analysis_PSF0X**(*DataDict*, *RepDict*, *inputs*, *log=None*)
Performs basic analysis on spots: - Gaussian Fits: peak, position, width_x, width_y

vison.point.PSF0X.**bayes_analysis_PSF0X**(*DataDict*, *RepDict*, *inputs*, *log=None*)

Performs bayesian decomposition of the spot images:

- optomechanic PSF and detector PSF.

Also measures R2, fwhm and ellipticity of "extracted" detector PSF.

vison.point.PSF0X.**filterexposures_PSF0X**(*inwavelength*, *explogf*, *datapath*, *OBSID_lims*, *structure={'Ncols': 6, 'col6': {'Exptime': 18.0, 'N': 3}, 'col4': {'Exptime': 10.0, 'N': 10}, 'col5': {'Exptime': 15.0, 'N': 4}, 'col2': {'Exptime': 1.0, 'N': 20}, 'col3': {'Exptime': 5.0, 'N': 18}, 'col1': {'Exptime': 0, 'N': 5}}*, *elvis='5.7.04'*)
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.

vison.point.PSF0X.**meta_analysis_PSF0X**(*DataDict*, *RepDict*, *inputs*, *log=None*)
Analyzes the relation between detector PSF and fluence.

vison.point.PSF0X.**prep_data_PSF0X**(*DataDict*, *RepDict*, *inputs*, *log=None*)
Takes Raw Data and prepares it for further analysis. Also checks that data quality is enough.

vison.point.PSF0X.**run**(*inputs*, *log=None*)
Test PSF0X master function.

6.1.3 basis

Created on Thu Apr 20 18:56:40 2017

author Ruyman Azzollini

contact r.azzollini _at_ ucl.ac.uk

class vison.point.basis.**SpotBase**(*data*, *log=None*)

6.1.4 display

Created on Fri Apr 21 14:02:57 2017

author Ruyman Azzollini

contact r.azzollini _at_ ucl.ac.uk

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

6.1.5 gauss

Doing Aperture Photometry of an object

Simple class to do Gaussian Fitting to a spot.

requires NumPy, astropy

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

Created on Thu Apr 20 16:42:47 2017

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

class vison.point.gauss.**Gaussmeter**(*data*, *log=None*, ***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()

6.1.6 models

FM-Calib. Campaign.

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

Created on Wed Apr 19 11:47:00 2017

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

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

A gaussian fitting function where p[0] = amplitude p[1] = x0 p[2] = y0 p[3] = sigmax p[4] = sigmay p[5] = floor

6.1.7 photom

Doing Aperture Photometry of an object

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

requires NumPy

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

Created on Thu Apr 20 14:37:46 2017

author Ruyman Azzollini

contact r.azzollini _at_ ucl.ac.uk

class vison.point.photom.**Photometer**(*data*, *log=None*, ***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*)

6.1.8 shape

Measuring a shape of an object

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

requires NumPy

requres PyFITS

author Sami-Matias Niemi, Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

class vison.point.shape.**Shapemeter**(*data*, *log=None*, ***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

class vison.point.shape.**TestShape**(*methodName='runTest'*)
Unit tests for the shape class.

6.1.9 spot

Created on Thu Apr 20 15:35:08 2017

@author: raf

class vison.point.spot.**Spot**(*data*, *log=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.

6.1.10 lib.py

FM-Calib. Campaign.

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

contact r.azzollini_at_ucl.ac.uk

vison.point.lib.**fwcentroid**(*image*, *checkbox=1*, *maxiterations=30*, *threshold=1e-05*, *halfwidth=35*, *verbose=False*, *full=False*)

Implement the Floating-window first moment centroid algorithm chosen for JWST target acquisition.

See JWST-STScI-001117 and JWST-STScI-001134 for details.

This code makes no attempt to vectorize or optimize for speed; it's pretty much just a straight verbatim implementation of the IDL-like pseudocode provided in JWST-STScI-001117

image [array_like] image to centroid

checkbox [int] size of moving checkbox for initial peak pixel guess. Default 1

maxiterations [int] Max number of loops. Default 30

halfwidth [int] Half width of the centroid box size (less 1). Specify as a scalar, or a tuple Xhalfwidth, Yhalfwidth. Empirical tests suggest this parameter should be at *least* the PSF FWHM for convergence, preferably some small factor larger

threshold [float] Position threshold for convergence

(ycen, xcen) [float tuple] Measured centroid position. Note that this is returned in Pythonic Y,X order for use as array indices, etc.

-Marshall Perrin 2011-02-11

vison.point.lib.**get_Gauss_spot**()

measurements:'i0_G','ei0_G','flu_G','eflu_G', 'x0_G','ex0_G','y0_G','ey0_G', 'sx_G','esx_G','sy_G','esy_G', 'e1','e2','e','R2'

vison.point.lib.**get_Quadrupoles_spot**()
measurements:'e1','e2','e','R2'

vison.point.lib.**get_photom_spot**(*stamp*, *x0*, *y0*)
measurements:'apflu','eapflu','bgd','ebgd'

CHAPTER

SEVEN

SUPPORT CODE

7.1 Support Code

IO related functions.

requires PyFITS

requires NumPy

author Sami-Matias Niemi

contact r.azzollini_at_ucl.ac.uk

vison.support.files.**cPickleDump**(*data*, *output*)
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*)
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.

Euclid-VIS Calibration Programme Pipeline: vison

Reporting Utilities.

History

Created on Wed Jan 25 16:58:33 2017

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

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

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**(*Title=''*, *level=0*)

generate_Latex()

class vison.support.report.**Table**(*tableDict*, *formats={}*, *names=[]*, *caption=None*)

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()

Just a collection of LaTeX templates for use in report.py

History

Created on Mon Jan 30 2017

author Ruyman Azzollini

contact r.azzollini_at_ucl.ac.uk

vison.support.latex.**generate_header**(*test*, *model*, *author*)

These functions can be used for logging information.

Warning: logger is not multiprocessing safe.

author Sami-Matias Niemi

contact r.azzollini_at_ucl.ac.uk

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.**setUpLogger**(*log_filename*, *loggername='logger'*)
Sets up a logger.

Param log_filename: name of the file to save the log.

Param loggername: name of the logger

Returns logger instance

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

V

H

I

L

M

P

Q

R

S

T

V

W