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

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()

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

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**()

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)

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]

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

SUPPORT CODE

4.1 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

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

V

R

S

T

V

W