Documentation for the PipelineManager

The code

UVB.py, **VIS.py** and **NIR.py** are scripts that can do the data reduction for the three arms. In these files the user can choose which recipes to run, and what files to use. To run the pipeline run (in a shell):

python UVB.py

PipelineManager.py is the code that deals with sof-files and esorex. It is not necessary to modify it to run the pipeline.

LoadRestartFile*.py can be used to load a restartfile and execute new recipes.

GetFiles.py is a Gasgano-script that organizes fits-files.

Some of the scired- and respon-recipes are unstable: some of them are untested, and others are unstable because of the pipeline. If something goes wrong in the last recipes (xsh_scired_XXX or xsh_respon_XXX) then it is usually easiest to run the last recipes from the shell (using the SOF-files generated by the script).

Packages needed

The following programs must be installed:

- Python (version 2.x.y)
- pyfits (replaces dfits/fitsort in version >5.6.0)
- The xshooter-pipeline (esorex and the recipes)
- pickle (a Python package used to create restart-files)
- zenity (optional): a gui used in GetsFiles.py

Getting started

To run the pipeline open UVB.py, VIS.py and NIR.py and do the following

- Enable the recipes you want to run: make sure that the function ".EnableRecipe(...)" is called for each of the recipes you want execute (and remove ".EnableRecipe(...)" for the recipes you don't want to run).
- Define your input-files using the SetFiles()-function. You only have to give bias- and flat-frames with the binnings and readouts you need.
- Run the scripts (Type "python UVB.py", "python VIS.py" and "python NIR.py" in a terminal)

Member functions of the PipelineManager class

SetFiles

SetFiles is used to tell the PipelineManager where the different files are located.

PipelineManager.SetFiles(SofTag, Files)

SogTag: A string with the SofTag for the files

Files: The files in a list. (relative path or full path can be used)

Example:

```
NIR.SetFiles("DARK_NIR",['DARK_NIR00.fits','DARK_NIR01.fits','DARK_NIR02.fits'])
```

DeclareNewRecipe

DeclareNewRecipe can be called in two ways.
1)
PipelineManager.DeclareNewRecipe(EsorexName)

EsorexName: The name of the Esorex recipe (as a string)

2)
PipelineManager.DeclareNewRecipe(EsorexName, SOFFileName, BinX_match , BinY_match, ReadOut_match)

EsorexName: The name of the Esorex recipe (as a string)
SOFFileName: The Name of the Sof-file (optional argument)
BinX_match: The binning of the raw-files. It should be '1', '2' or '' (optional argument)
BinY_match: The binning of the raw-files. It should be '1', '2' or '' (optional argument)
ReadOut_match: The Readout. '100', '400' or '' (optional argument)

use 2) if you have to run the same recipe for different binning/readout-combinations (e.g. With xsh_mbias or xsh_mflat). Otherwise use 1).

Always use 1) for the NIR-arm.

DeclareRecipeInputTag

Declares the Input of the recipes
PipelineManager.DeclareRecipeInputTag(SOFFileName, InputTagName, Nfiles, Binning, ReadOut)

SOFFileName: Same as in DeclareNewRecipe
InputTagName: The Sof-tag of the input file
Nfile: Number of input files (as string), or '?' (if optional). '1..n' is ok too (also with other numbers than 1)
Binning: '1x1', '1x2', '2x2', '-', 'match' or 'any'
ReadOut: '100k', '400k', '-', 'match', 'any' or '100k/400k' ('any' and '100k/400k' are identical)

For the NIR-arm Binning and ReadOut should always have the value '-' (otherwise the program will crash since binning and readout are not defined in the fits-header in the same way as for UVB and VIS)

SetRecipeOptions

Set options for the recipes:
PipelineManager.SetRecipeOptions(SOFFileName, Options)
SOFFileName: Same as in DeclareNewRecipe
Options: String with the options.

Example:
NIR.SetRecipeOptions(SOFFileName, "--sky-method=MEDIAN –background-method=median")

SetEsorexOptions

Esorex options
PipelineManager.SetEsorexOptions(SOFFileName, Options)
SOFFileName: Same as in DeclareNewRecipe

EnableRecipe

When calling EnableRecipe the recipe is added to the recipes that will be run when RunPipelineManager is executed.

PipelineManager.EnableRecipe(SOFFileName)
SOFFileName: Same as in DeclareNewRecipe

DisableEsorex

When calling DisableRecipe esorex will not be executed for the recipe. The SOF-file will still be created.

PipelineManager.DisableEsorex(SOFFileName)
SOFFileName: Same as in DeclareNewRecipe

you must have run EnableRecipe first. If not a sof-file will not be generated.

StopAfterRecipe

When this function is called the PipelineManager will stop, when the recipe with name SOFFileName is finished.

PipelineManager.StopAfterRecipe(SOFFileName)
SOFFileName: Same as in DeclareNewRecipe

PrintFilesInDictionary

This function will print all files saved in the dictionary... Good for debugging. Se how it is used in the restart-file, 'print_files.py'.

PipelineManager.PrintFilesInDictionary()

RunPipeline

RunPipeline will run the pipeline.

PipelineManager.RunPipeline()

no arguments.

ResetAllRecipes

ResetAllRecipes will reset all recipes. This function is meant to be used after restart-files are loaded.

PipelineManager.ResetAllRecipes()

no arguments.

SetOutputDir

Sets the output-directory. Default is 'Output/'

PipelineManager.SetOutputDir(OutputDir)

OutputDir: string with output-directory. e.g. 'OutVis_nosky/'

GetOutputDir

retnsur the output-directory.

PipelineManager.GetOutputDir():

no arguments

Examples

NIR-Example

An example of how to define and run a recipe:

```
#!/usr/bin/python
from PipelineManager import *

#initialization of the PipelineManager (declaration of variables, memory allocation etc)
NIR = PipelineManager()

#initialization of  'xsh_mdark' (declaration of variables, memory allocation etc)
NIR.DeclareNewRecipe('xsh_mdark')

#Two input tags are defined (copied from the pipeline manual)
NIR.DeclareRecipeInputTag('xsh_mdark', "DARK_NIR", "3", "-", "-")
NIR.DeclareRecipeInputTag('xsh_mdark', "MASTER_BP_MAP_NIR", "?", "-", "-")

#When the following command is called the recipe with name 'xsh_mdark' is added to the recipes, that will be
#executed when 'NIR.RunPipeline()' is called
NIR.EnableRecipe('xsh_mdark')

#DARK_NIR input files
NIR.SetFiles("DARK_NIR",['DARK_NIR00.fits','DARK_NIR01.fits','DARK_NIR02.fits'])

#The pipeline can be run with the following command
NIR.RunPipeline()
```

More examples are shown in NIR.py. After the execution of each recipe the output-files are automatically copied to the output-folder (declared in PipelineManager.py) and inserted into a dictionary[1], so they can be found by the next recipes.

To disable a recipe just remove the line 'NIR.EnableRecipe('xsh_mdark')'.

1 The dictionary is a member variable of the PipelineManager class.

VIS-Example

In this example bias-frames are created for the VIS-arm. Bias-frames with Binning 1x1,1x2 and 2x2 for readout 100k and 400k are produced.

```
#!/usr/bin/python
from PipelineManager import *

VIS = PipelineManager()

#A recipe is defined for each binning and readout (using two for-loops):
for ReadOut_match in ["100","400"]:
   for Binning in [["1","1"],["1","2"],["2","2"]]:
      BinX_match = Binning[0]
      BinY_match = Binning[1]

      EsorexName='xsh_mbias'
      SOFFileName='xsh_mbias_'+BinX_match+'x'+BinY_match+'_'+ReadOut_match

      VIS.DeclareNewRecipe(EsorexName,SOFFileName, BinX_match, BinY_match, ReadOut_match )
      VIS.DeclareRecipeInputTag(SOFFileName, "BIAS_VIS", "5", "any", "100k/400k")
      VIS.DeclareRecipeInputTag(SOFFileName, "MASTER_BP_MAP_VIS", "?", "match", "match")

      VIS.EnableRecipe(SOFFileName)

VIS.SetFiles('BIAS_VIS',\
['12:58.754.fits','9:15:42.309.fits','09:18:25.863.fits','09:21:09.417.fits','09:23:52.972.fits','09:26:06.594.fits','09:27:34.522.fits','09:29:02.459.fits','09:30:30.387.fits','09:31:58.315.fits','09:37:27.434.fits','09:38:11.818.fits','09:38:56.202.fits','09:39:40.556.fits','09:40:24.920.fits'])

VIS.RunPipeline()
```

Restart-files

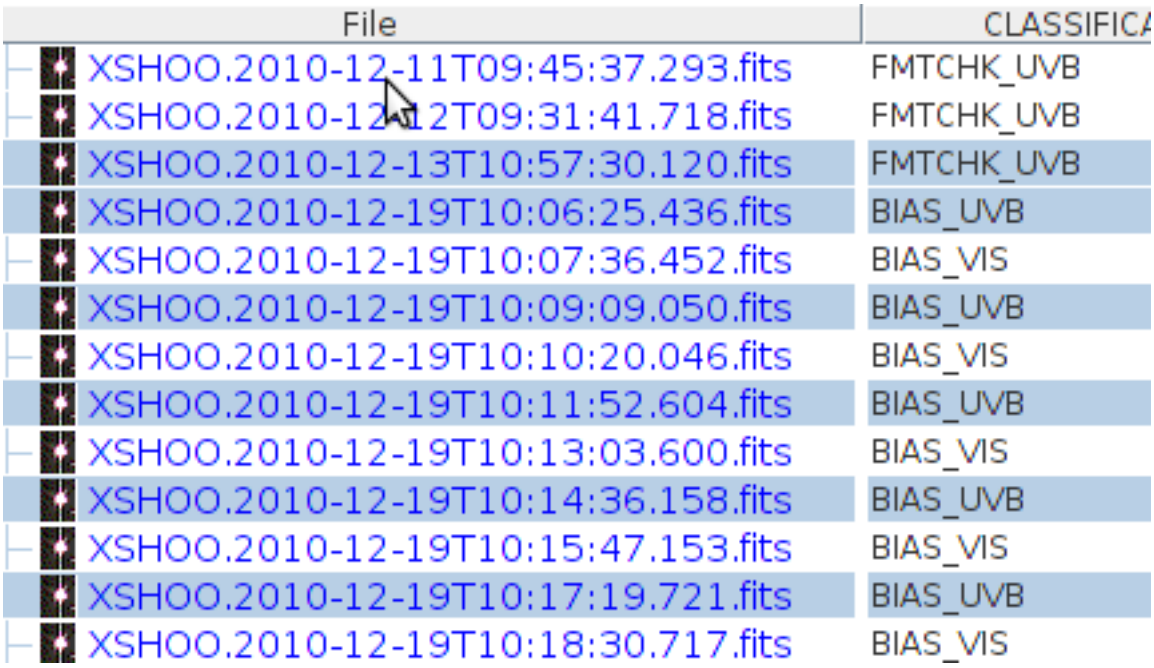
After a recipe is finished a restart-file with suffix '.restart' is created. It contains all information about the recipes and the files. It is possible to load this file (using the function, LoadRestartFile()) and start again from it. The following example shows how a recipe can be run again with a new option:

```
#!/usr/bin/python
from PipelineManager import *
#'Output/xsh_mbias_2x2_400_UVB.restart' is the restartfile generated just 'xsh_predict'
UVB = LoadRestartFile('Output/xsh_mbias_2x2_400_UVB.restart')

UVB.SetRecipeOptions('xsh_predict','--detectarclines-min-sn=2.5')

UVB.RunPipeline()
```

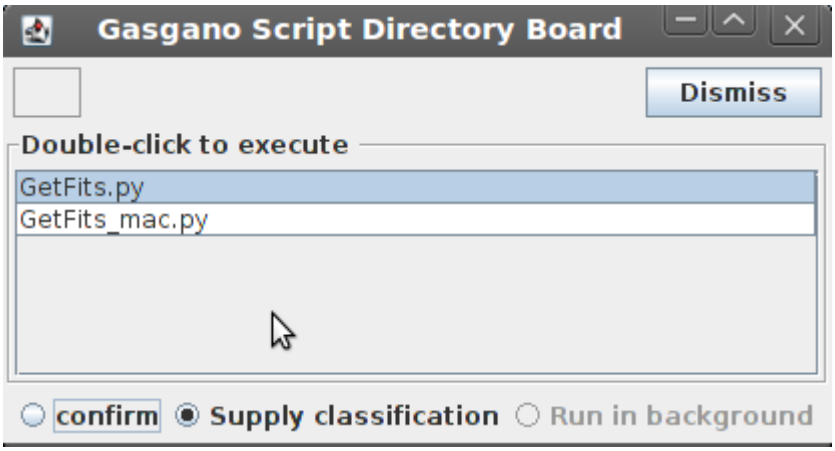
Adding files using Gasgano

The gasgano-script GetFiles.py can create the SetFiles commands in UVB.py, VIS.py and NIR.py:

- Open the script (GetFiles.py) in an editor and set the DESKTOP-variable in the code to the destination of your desktop (use the full path).
- Open Gasgano and select 'Files > Preferences...' and make sure that GetFiles.py is in the “Scripts Direcrory”.
- Select the files to be inserted into UVB.py, VIS.py or NIR.py:



- Open the script board (Tools > Script Board) and double click on GetFiles.py (Remember to enable the “supply classification” option):



- When the script is executed a popup-window will show the SetFiles-commands. (if zenity is installed)
- If you don't have zenity installed: Check the Desktop: A file named 'GetFitsOutput.txt' will now contain the SetFiles-commands.