Documentation

How to create your own difference images from Hubble’s WFC3 (UVIS) or ACS

Hi, this is a brief overview on what programs to run and how to use the pipeline on plhstproc1/2 for making difference images, specified for Hubble’s WFC3-UVIS, a camera with two CCDs (UVIS 1 and UVIS 2) and ACS.

The input (\_flc.fits) files look something like this:

ibsf15akq\_flc.fits

The first letter indicates what camera this is from, “i” means WFC3. “j” ACS. Three for the proposals specified letters follow the first one, here “bsf” for proposal 12499. Then we have the visit number, as not all images are taken at once, all images from one session in a given orbit are called one visit. Three time specific characters, here “akq” and the images extension “flc.fits” follow the number. Flc images are calibrated and flattened images, which we prefer over flt, only flattened images.

We now want to subtract two sets of images from one another, therefore we dump all the images in one folder and start running the following scripts (I created an alias for jpy aliased to the echoing the path to the jpy scripts):

/ifs/cs/projects/armin1/hstpipe/v20.0/photpipe/pythonscripts/HSTWFC3UVIS\_jhein/

This is equivalent to cdpy/HSTWFC3UVIS\_jhein/

# 1. sort\_images\_by\_filter.py

On the images we run sort\_images\_by\_filter.py from this folder. It sorts the images into different directories depending on their header keyword “FILTER” and creates this keyword, if not available (some HST images rather have two keywords, FILTER1 and FILTER2, and one of them has CLEAR1/2L which is not applicable). Every Filter should be handled separately now.

# 2. Tweakreg.py - Tweakreg

We now align the images using tweakreg to an outside catalogue. The script is called Tweakreg.py. It goes through two lists, VIS and FILTER, and aligns all images with this visit in their name and this Filter in their header to a given catalogue. As there can always be a problem, it is recommended to make a back up eg. in the same directory as bu/. We can give a set of parameters here:

|  |  |  |  |
| --- | --- | --- | --- |
| -trinput | str | ‘flc.fits’ | General criteria for scanning through the images used in the search for Filters and the images which are finally used |
| -VIS | str | Empty list | Values can be given in a white space separated list, eg.:  -VIS 01 10 20 42  The program will run on these images only, and every set separately. Not giving any parameters is automatically creating a list of visits between 01 and 20 |
| -FILTER | str | Empty list | Limits the Filter one wants to run through, one can by hand specify which Filters are to use, eg. After sorting them one can give–FILTER F775W  Giving an empty list will result in the program scanning through all the images in the directory, looking for all possible Filters. It takes a minute of time but is still quite useful and avoids typing errors from the user |
| -mode | str | Jakob | If somebody has another idea how to sort the images and how to run them an elif statement can be added:  elif mode== MyMode:  run this instead  So far only mode Jakob exists |
| -dir | str | ./ | runs the program on the given directory, it is not recommended to change this though, depending on the environment it will puke if there is a dot or a capital letter in the directory path, rather cd into the directory and run the program with -dir ./ |
| -tr\_refcat | str |  | specifies the catalogue used, values in degrees for ra,dec (hexagesimal), with the columns:  ra, dec, mag, e\_mag  1:23:45.678 -9:87:65.432 15 0.001  … |

The recommended environment is ssbdev. It is not recommended to change any tweakreg parameters, if you need to go to the script and change them in there. To look up the possible parameters go to the website:

<http://stsdas.stsci.edu/stsci_python_sphinxdocs_2.13/drizzlepac/tweakreg.html>

Right now we use something specified, that works for a txt file with ra, dec, mag (band), dmag (band) from cutting down a given catalogue to that information.

The original catalogue for 30dor can be found here:

<https://archive.stsci.edu/prepds/30dor/Preview/observations.html>

The catalogues for each brick of the PHAT survey here:

<https://archive.stsci.edu/prepds/phat/datalist.html>

|  |  |
| --- | --- |
| files= dir + (element from list of images with specified Filter) + trinput | Image input is specified |
| refimage = None  refcat= tr\_refcat | We want to have a catalogue  Can be specified by user input |
| interactive=False  clean=True | It can be extremely annoying to press return for every image, so we avoid it. Furthermore, Drizzlepac has the advantage, that it will give you any files created in between, the problem is, that we don’t need any of them |
| updatewcs=True  updatehdr=True  wcsname=‘UPDATEWCS’ | We want to update the wcs information in the header, the new name for it is called UPDATEWCS, the name is not important later on |

The input files are moved to a new Folder FILTER\_VISIT/ and output is moved to png/ and coo/ depending on their extension.

Tweakreg.py aligns the images, including every image extension; UVIS1 is generally in fits extension 4 or in extension 1, UVIS2 in the other. Both are finally aligned independent of the other. As there are a lot of stars to align to, in the images and the catalogue, this can always take a while.

# 3. Astrodrizzle.py

This is a buggy script, which can be left out, it drizzles the images together and while doing so we take advantage of the cosmic ray reduction, which is only one step of the many. It is not recommended to run it at all, in CPFIX this will be corrected anyways. For those who really want to do this anyways I recommend using the TEAL interface, further information can be found at:

<http://stsdas.stsci.edu/stsci_python_sphinxdocs_2.13/drizzlepac/astrodrizzle.html>

# 4. FCNUM.py / DIFFVAL.py (other possibility is compare\_images\_1dir.py, if possible use this one)

# a) FCNUM

The name rename originally was used as here depending on which CCD is in which extension, two images would have been created, one with UVIS1 and one with UVIS2 in extension 1. This has been changed, as it takes way too much disk space to do this. This rather assigns field center numbers (FCNUM). FCNUM is used for the difference imaging; in the pipeline field centers are determined (mkfieldfile4swarp\_HST.pl) for the SWARP stage. We here look at the aligned images and assign FCNUM depending on the center of UVIS1. We also wet MAINCCD to the UVIS number in extension 1.

For the test, if two images are more or less in the same field we have to give a grade of uncertainty, which is the length of the shorter side of one image, which is converted into degrees and gives the upper limit for the angle of separation between the two center points. Otherwise one can specify the maximum search radius; these are the parameters:

|  |  |  |  |
| --- | --- | --- | --- |
| -radius | float | None | This can be used to limit the search radius, a small value would be 0.001, using the short side of an image (1024 pixels, 0.04 arcsec/pix => radius=0.0113778) automatically calculates the radius |
| -onlyshow | bool | False | Give –onlyshow to not apply the changes, but to show what would be done. |
| -f | bool | False | Give -f to disable interactivity |
| -survey | str | None | Give –survey to set the keyword survey, used in rename\*\*\*.pl, if you don’t give it In the first place you are asked to set it |

At the end a plot will be shown, with the center points of the images. One color belongs to one field center, but not necessarily, as the color code repeats itself after some iteration (seven or eight). Using the rainbow\_colorcode (©) would be an option, but the difference in color for many fields is rather small. The plot in rainbow colors is showing the order of the images.

# b) DIFFVAL.py (is not necessary any more, was used for 30dor!)

This value is used to create the right directory and the tmpl for the difference imaging. It finds diffvalues for the images in one field taken the most time apart and with the highest exposure time, then moves the images into ./1 ./-1 which will be SURVEY\_FILTER and SURVEY\_FILTER\_tmpl lateron

# 5. Rename\_jhein.pl / Rename\_jhein\_ACS.pl / rename\_jhein\_30dorbkp.pl

The data can now be put anywhere, preferably in the rawdata\_orig/ directory (cdraw/../rawdata\_orig/) so it can always be recovered easily. From there we link the files into the rawdata directory (cdraw), as this is where the pipeline will look for them. The files are renamed and get a special name and directory, depending on their project ID, their Fieldcenter, their Instrument, Survey, Diffval etc. Every image is linked twice with CCD1 and 2, so cpfix can extract the data for UVIS1 from the one and 2 from the other directory without going into a loop. The links are also not copies of the files and take not a lot of disk space. To run rename\_jhein.pl just type it while being in the right directory.

The original file is located in jpl (cdperl/HSTWFC3UVIS/) but the one being executed is in cdperl. Parameters are –onlyshow as in FCNUM.py and indir /outdir can be specified too.

Currently the output directory is SURVEY\_FILTER (\_tmpl if using the one for 30dor)

# 6. The Pipeline

This so far has only been the initialization; we now have the images in the cdraw directory.

And with FINDNEWIM the pipeline looks for new images, CPFIX puts them into the cdwork directory. The images are run like normal reduction images are though the normal stages and afterwards though the diffim stages which are:

|  |  |
| --- | --- |
| Red:  FINDNEWIM  CPFIX  SIP2PV  MKSATMASK  SWARP  DOPOT  ABSPHOT | Diff:  MATCHTEMPL  DIFFIM |

Pipeline stages are run the following way:

pipeloop.pl –red directory CCDs

pipeloop.pl –diff directory1 directory2 CCDs (here the second directory would besubtracted from the first one)

To see where the single images are use:

pipeview.pl –red directory CCDs (or –diff…)

pipestats.pl –red directory CCDs (or –diff…)

Specifications for pipeloop are:

|  |  |  |
| --- | --- | --- |
| -stage | Specifies the stages | -stage FINDNEWIM  -stage CPFIX-ABSPHOT |
| -redo  -redobad | Force to redo (all/failures) the stage, otherwise it will only take the input from the previous stage | -redo  -redobad |
| -k | Specify any stage specific parameter in cdcofig/ pipeline.params | -k CONDOR\_MAX\_CPU 128  -k SW\_NX 10000  -k SW\_NY 10000 |
| -condor | Send request to condor cloud, makes it faster due to CPU splitting, but the output will not be visible, this works great when knowing that everything should works |  |

# 6.1.1 FINDNEWIM

FINDNEWIM look into cdraw to check if there are new images, nothing else.

# 6.1.2 CPFIX

CPFIX uses the output of FINDNEWIM to copy these images to cdwork, it reads out which CCD is the MAINCCD and saves the data accordingly. It looks for the SCI, ERR and DQ extensions and gives an error if they are not as wished.

It also uses the DQ extension to mask cosmic rays, hot pixel,…

For the flc files the extensions can look somewhat confusing, with certain better-corrected areas, this is because the cosmic ray reduction has also been done (by overlapping images).

Necessary for this stage are the files following two files in cdconfig UVIS1.zpt.txt and UVIS2.zpt.txt. And the not yet existing equivalent, the ACS zero points.

# 6.1.3 SIP2PV

Converts inside the header .SIP to .PV, a stage that is quickly done, normally should not raise any errors ever, as it is a copy of a general sip2pv converter.

# 6.1.4 MKSATMASK

Masks out bright start, whose light “spilled over” into neighboring pixel

# 6.1.5 a) mkfieldfile4swarp\_HST.pl

THIS IS NOT PART OF THE PIPELINE. In cdconfig there is a file called DEFAULT.fieldcenters, necessary for SWARP, giving a field center for the target name and the CCD which is determined by the mkfieldfile4swarp\_HST.pl script with the following syntax:

mkfieldfile4swarp\_HST.pl FILENAME DIRECTORY(as in the pipeloop) CCDmin CCDmax

-useRADECfromheader (uses ra dec from header, key words can be specified, default are RA, DEC)

-rootdir INDIR (changes input directory)

-workdir (= -rootdir ‘/workspace’)

More information with mkfieldfile4swarp\_HST.pl –h

Output file goes into the current directory, move it or copy the content to the DEFAULT.fieldcenters file in cdconfig, as this is the place where the information is taken from. It seems like an unnecessary step, but is actually extremely important, as we here adjust every field center (previously defined by FCNUM!) to the sphere we are looking at. The images themselves look like rectangles, but there is an error while assuming that they actually are.

My personal recommendation is to link all the files from both directories (SURVEY\_FILTER/\*/ and SURVEY\_FILTER\_tmpl/\*/) into a new directory to have as little difference as possible between them

# 6.1.5 b) SWARP

It projects the image into the sky. The square’s center is determined by the field center. The output is a sw.fits file. Mask, Noise and Weight file are also given

# 6.1.6 DOPHOT

The sw.fits file from SWARP is taken now and converted into a catalogue (.dcmp), which can be looked at using ds9mark.pl

# 6.1.7 ABSPHOT

This catalogue is compared again to an outside catalogue, preferably the one used in Tweakreg.py, to correct for any magnitude shifts. It has to be placed in cddata/absphot/ MEANINGFULNAME / photcode/ field.photcode.cat

# 6.2.1 MATCHTEPL

Using pipeloop.pl –diff dir dir\_tmpl <CCDS>

It is the –diff equivalent to FINDNEWIM, it will link the necessary files to

# 6.2.2 DIFFIM

Makes difference images by subtracting sw.fits images