STScI | SPACE TELESCOPE SCIENCE INSTITUTE

Instrument Science Report WFC3 2021-02

Reducing Drift and Shift (DASH) Data Using wfc3_dash and Accompanying Notebook Workflow

C. Martlin, R. O'Brien, I. Momcheva, M. Gennaro

Updated: November 30, 2022
Originally Published: January 11, 2021

Abstract

DASH mode enables large, shallow WFC3/IR mosaics within one orbit by dropping to gyros for guiding on all exposures after the first exposure (see Section 6.12.5 in the Instrument Handbook). Due to dropping to gyros, the FLT/FLC pipeline output images created by the calibration pipeline are unusable for typical science cases as all sources will be "smeared" across the detector. While the general process of how to derive science-ready data from DASH observations is described in the paper that introduces DASH (Momcheva, et. al 2016), the resources found at https://github.com/spacetelescope/WFC3Library are an interactive and adaptable version of instructions and scripted workflow. In this report we provide a detailed walkthrough of the wfc3_dash package and the suggested DASH processing workflow described in a Jupyter Notebook. Each step in the workflow has parameters that can be adjusted within the package. Further, individual steps can be substituted out if the user has a different method in mind for that portion of processing. Between the Jupyter Notebook workflow and the detailed information below a user will be able to create a

customized pipeline to create individual science ready images from each DASH exposure image.

---

1 Introduction

The purpose of this Instrument Science Report is to guide users of the Drift And SHift (DASH) data-taking method through using the new 'wfc3_dash' package and Jupyter notebook workflow to reduce their DASH data. While the general process of how to derive science-ready data is described in the paper that introduces DASH (Momcheva, et. al 2016), the resources found at `https://github.com/spacetelescope/WFC3Library` are an interactive and adaptable version of instructions and scripted workflow.
'dash.ipynb' - is a Jupyter Notebook that provides an in-depth walk through of using the wfc3_dash package to turn an individual DASH IMA - a calibrated intermediate IR mutliaccum image - into a stacked single image ready for science analysis. Steps 2 and 3 are currently in production and will provide users with workflows to create more complex science products; these are discussed further in the conclusion in section 5.

2 Important Links

- wfc3_dash package - `https://github.com/spacetelescope/WFC3Library`
- dash.ipynb - `https://github.com/spacetelescope/WFC3Library/blob/master/notebooks/dash/dash.ipynb`
- DrizzlePac Notebooks for aiding a user in alignment of individual reads (step 3 in the workflow) - `https://github.com/spacetelescope/notebooks/tree/master/notebooks/DrizzlePac`

3 DASH Pipeline

3.1 Pipeline Basics

The wfc3_dash package currently consists of the main script 'reduce_dash.py' which also utilizes the helper functions found within 'utils.py'. We don't currently have any online script documentation, such as ReadtheDocs, available but all of the functions within reduce_dash have fully prepared docstrings. For example, figure 1 shows the full docstring for the function 'fix_cosmic_rays' that can be found in 'reduce_dash.py' and is used in step 3.6. That step uses this function to remove cosmic rays from the individual difference files created from the original IMA image.

```
def fix_cosmic_rays(self, rm_custom=False, flag=None, **lacosmic_param):
    '''
    Resets cosmic rays within the seg maps of objects and uses L.A.Cosmic
    to find them again.

    Parameters
    ----------
    self : object
        DashData object created from an individual IMA file.
    rm_custom : bool
        Specifies whether or not the user would like to remove custom flags
        within the boundaries of sources, as defined by the segmentation map
        created from the original FLT.
    flag : int
        Specifies flag the user would like the remove within the boundaries
        of sources.
    lacosmic_param : dic
        Dictionary of the L.A.Cosmic parameters that users may want to specify.
        If not set, then presets are used.

    Output
    ------
    Fixed for cosmic rays diff files : fits
        Same diff files created in split_ima that have now been corrected
        for cosmic ray errors.
    '''
```

Figure 1: Example of a docstring of a function used within reduce_dash.py.

The main script used for reducing DASH data in our example pipeline is 'reduce_dash.py'. In the second figure we include a flowchart graphic to demonstrate the workflow of the notebook and the order of the functions from 'reduce_dash.py' that will allow a user to properly create a science ready image. Below are listed all the available functions within reduce_dash.py along with their input parameters and the default settings. This list follows the same order as the workflow flowchart in Figure 2 and maps to that workflow directly.

- main(ima_file_name = None, flt_file_name = None, align_method = None, ref_catalog = None, drz_output=None, subtract_background = False, wcsname = 'DASH', threshold = 50., cw = 3.5, updatehdr=True, updatewcs=True, searchrad=20., astrodriz=True, cat_file = 'catalogs/diff_catfile.cat')
- split_ima(self)
- make_pointing_asn(self)
- create_seg_map(self)
- diff_seg_map(self, cat_images=None, remove_column_names=True, nsigma=1.0, sig=6.0, npixels=5)
- subtract_background_reads(self, subtract=True, reset_stars_dq=False)
- fix_cosmic_rays(self, rm_custom=False, flag=None, **lacosmic_param)
- align(self, subtract_background = True, align_method = None, ref_catalog = None, create_diff_source_lists=True, updatehdr=True, updatewcs=True, wcsname = 'DASH', threshold = 50., cw = 3.5, searchrad=20., astrodriz=True, cat_file='catalogs/diff_catfile.cat', drz_output=None, move_files=False)
- move_files(self)

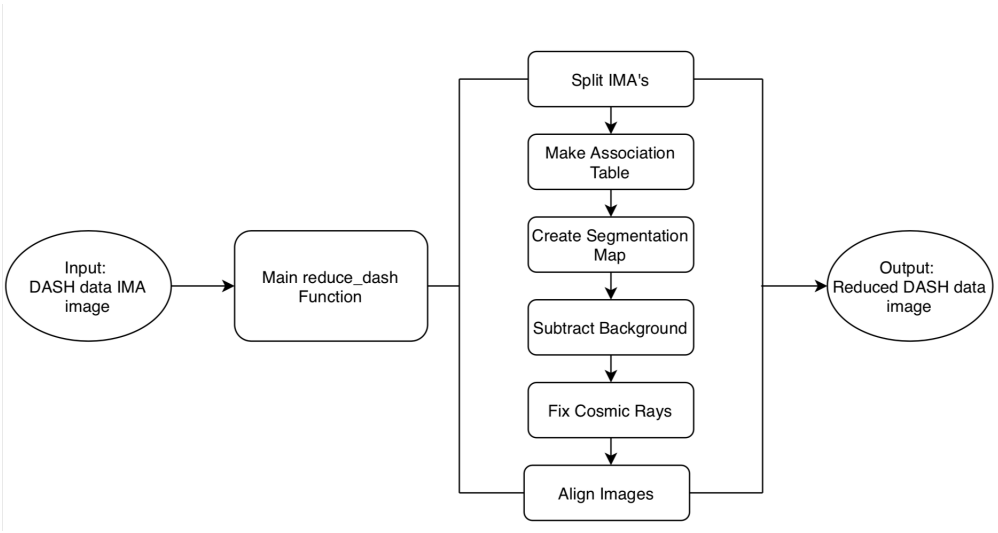

Figure 2: Flowchart of the functions within reduce_dash and their typical order - this is the workflow followed within the Jupyter Notebook walkthrough.

All functions used within our DASH data reduction pipeline have several parameters that have default values usually good for a first run for most DASH observations, but it's likely that some parameters will need to be tested to find the best fit for a user's particular observations. Further, the fact that these functions are run in distinct steps in our workflow lends the pipeline to be adaptable to a user's individual data reduction practices. For example, if a user has a different method for removing cosmic rays it is possible to skip fix_cosmic_rays( ), insert their own step to run on the individual IMA difference files (diff files), and then continue the notebook workflow afterwards.

3.2 Installation

Installation instructions are available on the GitHub page: `https://github.com/spacetelescope/WFC3Library`. Users will need to install the 'wfc3_dash' package as well as the 'Lacosmicx' package to fully run the pipeline; instructions for how to install both are available in the previously mentioned link.

4 Notebook 1 - The Ins and Outs

4.1 General User Workflow Overview

The main purpose of this notebook example is to aid users in creating a full workflow to produce science ready data from the IMA fits files of DASH data. Due to the lack of fine guidance, the FLT/FLC images created by the pipeline aren't appropriate for analysis as the sources will be smeared across the detector, as can be seen in the IMA read outs images below in Figure 3. Therefore, to create a science image with un-smeared sources a user needs to take the difference images of the IMA reads (the non-destructive readouts taken over the course of the observation) and stack them to create a full image. Examples of these

difference files can be seen below in Figure 4. This will allow the sources within that image to have a full readout time, similar to how the FLT/FLC output files work for non-DASH IR data.

The following sections will walk through the process provided in the notebook from data download to the creation of the final science image. This will be following the workflow from the flowchart in figure 2. Users can adapt the notebook to run on their own data or simply use it as an example for setting up a script or pipeline of their own for running on larger data sets. While the first several steps are general and easy to adapt to any data, it's important to note that many of the later steps - from the background subtraction onwards - will need to be specialized for every users specific data and the default values used in the example, while a good starting point, may need to be tested and updated to provide successful final science images.

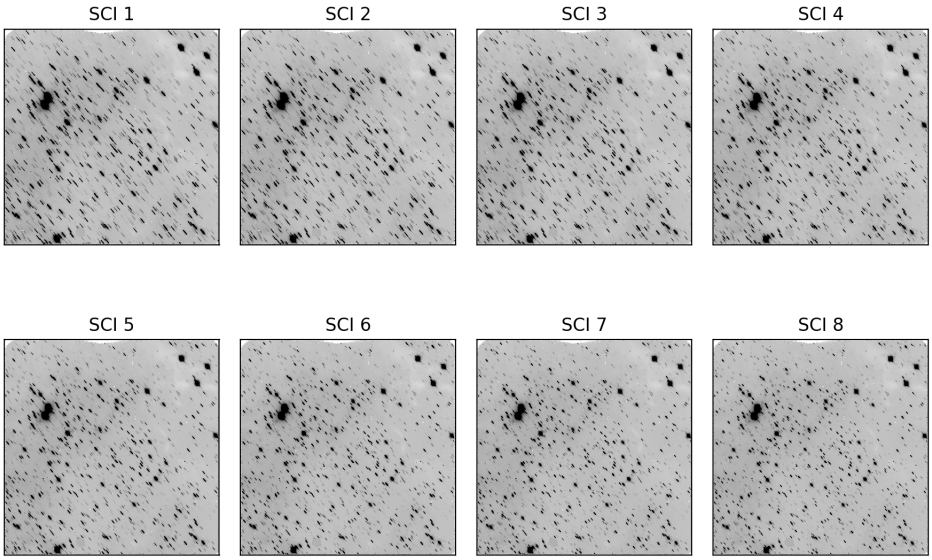

Figure 3: Example of the first portion of the IMA readout images used to build the FLT/FLC image. The blurring of the sources is seen strongly in the first several extensions as they are the addition of all previous extensions before them.

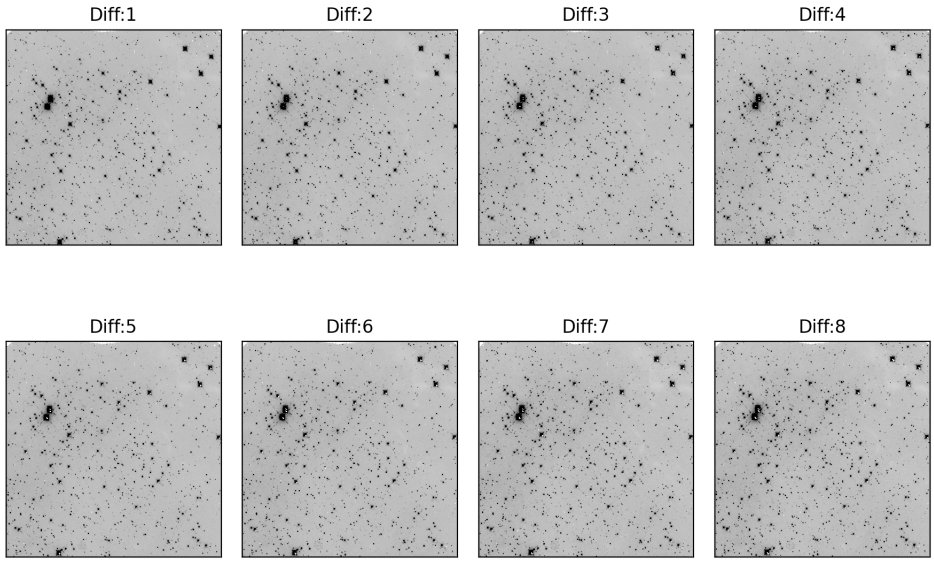

Figure 4: Example of the first half of difference images made from an IMA file. We no longer see the built up smear as in the usual IMA read outs and will be using these diff files to create our file science image.

4.2 Step 2. Data Download

For the workflow exercise within the first notebook we will be using publicly available data from proposal 15238 and will be using astroquery.mast package to download the needed IMA and FLT files. Information about how to install astroquery and how to use astroquery.mast can be found here `https://astroquery.readthedocs.io/en/latest/index.html`.

We have decided to only download a single observation for the example to keep the workflow as simple as possible. With astroquery it is possible to download large numbers of observations and any file type that is available through the usual MAST portal. In order to use the wfc3_dash workflow in the Jupyter Notebook you will need the IMA intermediate calibrated product and the FLT file pipeline product for each DASH observation that you want to reduce. You can obtain your observations using the same astroquery method demonstrated in the notebook or through the MAST portal, as long as the files end up in an accessible path to your Jupyter Notebook.

4.3 Step 3.1. Dash Object Overview

- Function used:

  DashData(ima_file_name, flt_file_name)

- Input file or object:

  IMA and FLT path for a single exposure

- Output file or object:

DashData object for that observation

a. Create a DashData object using the path to the IMA file we have downloaded above

```
In [ ]: myDash = DashData(localpathtofile+'ima.fits', flt_file_name=localpathtofile+'flt.fits')
        print(myDash.root)
```

Figure 5: Example of running step 3.1 from the workflow in the Jupyter Notebook.

The basis of the reduce_dash script is a class object called 'DashData' which allows us to store all associated information from the IMA and FLT files of a DASH observation. reduce_dash.py first creates the DashData object within the main( ) function then feeds that to each following function to be manipulated and updated appropriately. Most functions listed below will just use the DashData object to keep track of the different portions of the observation it needs to use to run - for example the path or filename of the IMA file - but some functions, such as split_ima( ) or make_pointing_asn( ), may also update or add a new value to the DashData object for use by another function later on.

While it is possible to run the entire pipeline using the single reduce_dash.main( ) function, we use the example notebook to demonstrate to users how to use each reduce_dash function individually to enable checking the results of each step before moving on to the next step in the process. This is important as each observation may need individual parameters for sensitive steps such as creating a segmentation map or aligning your final calibrated difference files to each other. This also allows the user to decide to swap in their own method for a specific step of the workflow, for example if they have a package or function able to create a more appropriate segmentation map they can decide on step 3.4 to comment out the use of myDash.create_seg_map( ) and instead feed their segmentation files into the next step of the pipeline. We will review in each step below the input files and output files of each step to ensure ease of substituting in alternative methods.

4.4 Step 3.2. Create diff files from IMA Reads

- Function used:

  myDash.split_ima( )

- Input file or object:

  myDash object with IMA and FLT paths

- Output file or object:

  1. myDash object updated with Difference filenames
  2. FITS files of the difference files between the IMA readouts

This step is where our pipeline differs from the standard HST pipeline. Normally the IMA readouts are stacked to create the final FLT image a user will work with. This isn't appropriate with DASH data though as each readout appears more smeared than the one

b. Create diff files

```
In [ ]: myDash.split_ima()
```

Figure 6: Example of running step 3.2 from the workflow in the Jupyter Notebook.

before it and when stacked together the data is unusable. But we are able to use the small time differences between read outs to create a fully stacked image using Astrodrizzle! This first step creates the difference images from adjacent reads which are short enough to not have much slew. Then we are able to take those differences reads and use the rest of the pipeline to stack them together to create a deeper and useful final science image.

4.5 Step 3.3. Create An Association File

- Function used:

  myDash.make_pointing_asn( )

- Input file or object:

  myDash object with IMA, diff files, and FLT paths for a single observation

- Output file or object:

  1. ASN FITS file with an updated header/list of the difference files from the IMA.
  2. Updates myDash object with the filename of the ASN file.

c. Create an association file

```
In [ ]: myDash.make_pointing_asn()
```

Figure 7: Example of running step 3.3 from the workflow in the Jupyter Notebook.

Association files are used within the HST pipeline and MAST data archive to aid users and pipelines in keeping related data organized, so we employed a similar method here. We take a step to create an association file that links together information about the IMA file, the FLT file and the set of difference files we created in the step prior to this one. This helps ensure we track and provide all needed information to later steps and allows us to be sure we are aligning all difference reads properly. This file can also be used outside the pipeline to aid in tracking intermediate files in case a user decides to develop individualized calibration steps. Examples of steps a user may decide to personalize for their data are listed below in steps 3.4 to 3.5.

4.6 Step 3.4 Create Segmentation Map

- Function used:

  myDash.create_seg_map( )

- Input file or object:

  myDash object with IMA, diff files, and FLT paths for a single observation.

- Output file or object:

  1. A FITS file of the segmentation image
  2. A *.dat file of all the sources and various centroid information

d. Create Segmentation Map

```
In [ ]: myDash.create_seg_map()
```

Figure 8: Example of running step 3.4 from the workflow in the Jupyter Notebook.

The segmentation map used in this workflow is created from the FLT file and is used to aid in the background subtraction step and the cosmic ray rejection step by providing the placement of the sources within the difference files. If a user has a seperate method for making a segmentation map to find their sources then they can skip this step and instead provide the *seg.fits files to the following steps.

4.7 Step 3.5. Subtracting Background

- Function used:

  myDash.subtract_background_reads(self, subtract = True, reset_stars_dq = True)

- Input file or object:

  1. myDash object with IMA, diff files, and FLT paths for a single observation.

  2. Subtract parameter as a bool where 'False' means the background calculated is not subtracted from the files, but is written to the header as 'MDRIZSKY' and 'True' means the background is written to the header and subtracted from the image.

  3. reset_stars_dq parameter as a bool where 'True' uses the segmentation map to reset pixels within sources that have been marked previously with the data quality flag (4096) for a cosmic ray back to zero, essentially reseting any cosmic ray flags that fall within a source.

- Output file or object:

  1. Difference FITS files that have been background subtracted.

This step uses the segmentation maps to find the median background value of each individual difference file (excluding the sources by using the segmentation map) and, if 'subtract = True', subtracts that value from the image to account for the background.

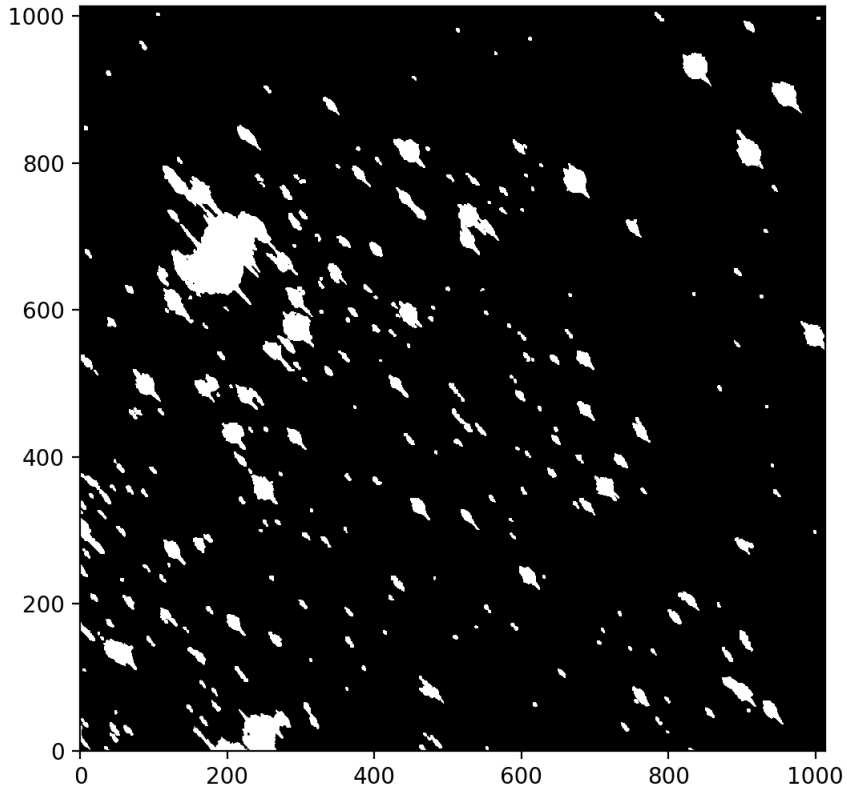

Figure 9: Example of a segmentation map as shown in the Jupyter Notebook.

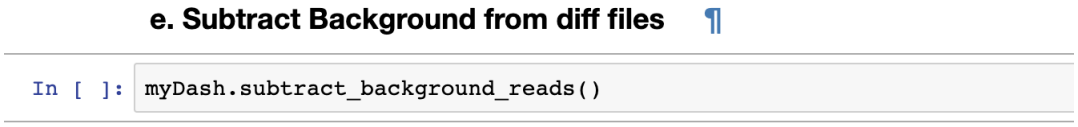

Figure 10: Example of running step 3.5 from the workflow in the Jupyter Notebook.

4.8 Step 3.6 Fixing Cosmic Rays

- Function used:

  myDash.fix_cosmic_rays(self, rm_custom=False, flag=None, **lacosmic_param)

- Input file or object:

  1. myDash object with IMA, diff files, and FLT paths for a single observation.

  2. rm_custom is a bool parameter set to 'False' by default. When 'True' a user may provide custom DQ flags they would like to remove from within sources.

3. flag can be set to an integer and should be a specific DQ prime number, such as 4096 which stands for Cosmic Rays, that the user would like removed if inside of a source as defined by the segmentation map.

4. lacosmic_param is a dictionary of the LAcosmic parameters a user can change from the hardcoded defaults.

- Output file or object:

1. Difference FITS files that have been corrected for cosmic rays and possibly have an update to the DQ array for pixels within sources.

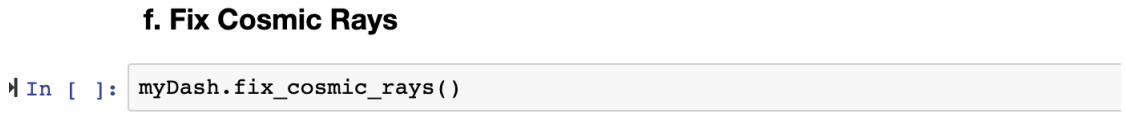

Figure 11: Example of running step 3.6 from the workflow in the Jupyter Notebook.

This step uses the LAcosmic software to identify cosmic rays within the difference files to remove them from the image by marking them in the data quality array with the cosmic ray flag. This step also has the ability to allow a user to reset any data quality flag within the sources in the difference image to allow using all values within those pixels.

4.9 Step 3.7. Aligning Difference Reads to Each Other

There are many more parameters available to update or change within the myDash.align function which is where the most specialized work a user will have to implement within their DASH pipeline work. This is due to the complexities of TweakReg and AstroDrizzle. Below we will go over the uses of myDash.align within the Jupyter Notebook example. Once you have successfully aligned your IMA difference files you have completed the pipeline and have a final science image equivalent to the FLT FITS file that is usually provided by the calibration pipeline.

The most important thing to note is that a user must first test their TweakReg alignment by setting 'astrodriz = False' and 'updatehdr = False'. Only once the TweakReg results show good alignment should the user then run myDash.align with 'astrodriz = True', as in step 3.8.

4.9.1 TweakReg

- Function used:

myDash.align(updatehdr = False, updatewcs = True, astrodriz = False)

- Input file or object:

1. updatehdr parameter is a bool where False means the header of the difference files will not be updated. User will want to set to True once their TweakReg results are good.

2. updatewcs is a boolian where False means the world coordinate system (wcs) is not updated in the difference file headers.

3. astrodriz is a boolian where False means AstroDrizzle is not run on the difference files. While testing TweakReg this should always be set to False.

- Output file or object:

  TweakReg fitting results

g. Align reads to each other

```
In [ ]: myDash.align(updatehdr=False, updatewcs=True, astrodriz=False)
```

Figure 12: Example of running step 3.8 from the workflow in the Jupyter Notebook.

TweakReg - a function available through AstroDrizzle - is one of the most important steps of this workflow as it will allow the user to ensure a good alignment between the diff files of the IMA image. However, specified information on how best to run this task has already been documented here `https://github.com/spacetelescope/notebooks/tree/master/notebooks/DrizzlePac`. Further, if a user has specific issues with TweakReg the HST Help Desk is available for questions and suggestions here `https://stsci.service-now.com/hst`. Only once a user has good shift results from TweakReg should they proceed to running the myDash.align( ) function with 'astrodriz' = True, as in the step below.

4.9.2 AstroDrizzle

- Function used:

  myDash.align(threshold = 20.)

- Input file or object:

  Threshold parameter which is default set to 20.

- Output file or object:

  A *drz.fits image which is the combination of the IMA diff files and is the final science product

To begin, there are many more AstroDrizzle parameters that have default settings within the myDash.align function, but for this basic walkthrough example those default settings work well and so we only adjust the Threshold parameter. Users should note that when testing on their own data they can begin with the default parameters and then adjust according to Astrodrizzle suggestions found within the Astrodrizzle documentation or Notebook examples. A user's updates to these parameters can be done in the same manner as the example Threshold update, by providing the parameter within the myDash.align( ) function - as long as the astrodriz parameter is set to 'True', which is the default.

Users needing more experience or help with AstroDrizzle should consult the documentation available here `https://drizzlepac.readthedocs.io/en/latest/astrodrizzle.html` and the notebooks available here `https://github.com/spacetelescope/notebooks/tree/master/notebooks/DrizzlePac`.

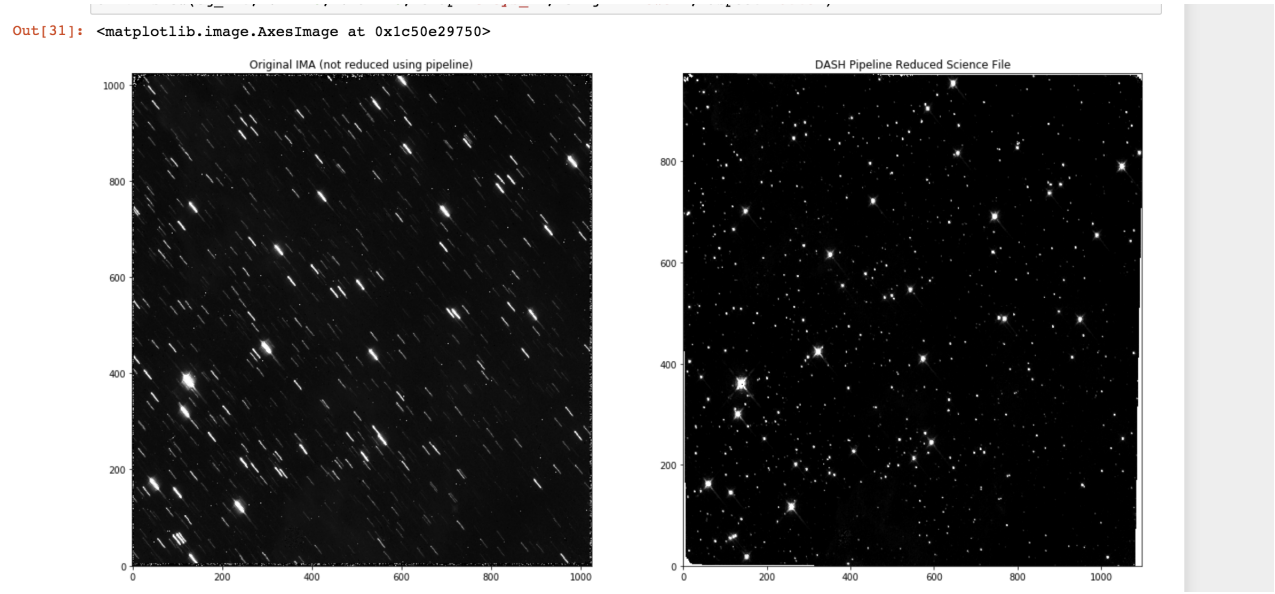

Figure 13: Example of the original pipeline FLT on the left which demonstrates the smearing of the sources compared to the final science image created by the DASH pipeline on the right where the sources are no longer smeared.

4.10 Advanced Options

As a reminder, this pipeline and the workflow demonstrated in the Jupyter Notebook is shown as an example of how to properly process DASH data, but mileage will vary greatly depending on the individual data. One reason this workflow was created to aid users in being able to swap in their own methods for every step. If the way our CR rejection is set up doesn't fully catch all cosmic rays in a users data they should feel free to replace that step with their own method and then continue with the rest of the workflow. Further, every run of TweakReg and AstroDrizzle will need individual testing of parameters which users may need to consult DrizzlePac documentation or notebooks to fully understand. The wfc3_dash package does allow users to run tests with TweakReg and AstroDrizzle without updating their files and the Jupyter notebook shows how to do this and then once the results look good the user will do a final run of both to create their final science image.

5 Conclusion

This first notebook is the most poignant as it walks the user through taking the intermediate DASH IMA file and creating a science ready final product. With the use of the wfc3_package

and the Jupyter Notebook workflow we are hoping to ease the process of creating these science ready images from a DASH observation.

Currently, two more notebooks are in development to follow this first one. The second notebook will be aiding users in using an outside catalogue to align multiple DASH IMA images from a single visit together - it will be able to take all IMA difference files from the same visit and create science data using whatever outside catalogue a user provides. Following that, a third workflow will be created to take the result of the first two notebooks and use Tweakback to apply the shifts from the second onto the individual diff images from the first and then drizzle all the diff images from the various exposures together into a mosaic of multiple exposures.

References

Dressel, L. 2019, 'Wide Field Camera 3 Instrument Handbook, Version 12.0'

Momcheva, I. et. al. 2016. "A New Method for Wide-Field Near-IR Imaging with the Hubble Space Telescope." https://arxiv.org/pdf/1603.00465.pdf