Import Statements

In [1]:
```
import os
import custom_cmap
from PIL import Image
from IPython.display import display

import matplotlib.pyplot as plt
import matplotlib as mpl
import pandas as pd

from moria import reduce
from pathlib import Path

from astropy.visualization import PercentileInterval
from astropy.io import fits
from astropy.visualization import LogStretch, ImageNormalize
import plotly.express as px
import numpy as np
from astropy.visualization import ImageNormalize, AsinhStretch, SqrtStretch,

plt.rcParams.update({'font.size':25})
plt.rc('text', usetex=True)
%config InlineBackend.figure_format = 'retina'
%load_ext autoreload
%autoreload 2
```

Understanding the main directory.

The directory structure for your data analysis should be organized as follows.

00.DATA
01.XYM
02.CMD
03.LOC_TRANS
04.PSF_EXTRACT
05.COORD_TRANS(OPTIONAL)
06.FIT
07.CALIBRATION
This directory structure has been set up in "MORIA/data".

All necessary scripts are included within the corresponding folders under MORIA/data. Therefore, the simplest way to run MORIA on your target is to copy all eight folders from MORIA/data to the location where you intend to perform your analysis. Begin by placing your exposures in the corresponding data/00.DATA/ directories (e.g. the /F814W and /F606W filter sub-directories).

NOTE: chmod +x program.src is a useful command to use whenever permissions are denied for a script. If any of these scripts fail, each folder in the list above (00.DATA, 01.XYM, 02.CMD, etc.) has an output "log file" for the Fortran scripts we run. If the "log file" for the step you ran shows a "permission denied" error, it is very likely that you need to run chmod _x program.src for each ".src" file used in this pipeline.

Notebook Goal

After you finish running this notebook, you will be able to use the "_flc" exposures to generate an output image stack in the F814W and F606W HST filters.

This notebook will prepare files in 00.DATA, 01.XYM and 02.CMD.

In [2]:
```
#This is the directory where you are processing your data. This does not poi
directory = os.getcwd()
```

Data Preparation

When you installed MORIA, several fortran scripts were compiled directly in "MORIA/src/fortran_compile". This step copies the fortran files into the correct directories.

In [3]:
```
reduce.data_prep_early(directory)
```

STEP 1

The cell below will convert the _flc files to _WJ2 files using run_convert_C1K1C.src These "_flc" exposures include the charge-transfer efficiency (CTE) corrections. MORIA first converts these "_flc" exposures into "_WJ2" files to convert to a full-chip coordiante system that are placed into appropriate reference geometry for subsequent steps.

In [4]:
```
reduce.run_xgf_conversion(directory)
```

```
NARG: FILENAME:           1  iffp02bxq_WJ2.fits
NARG: FILENAME:           2  iffp02byq_WJ2.fits
NARG: FILENAME:           3  iffp02c0q_WJ2.fits
NARG: FILENAME:           4  iffp02c1q_WJ2.fits
NARG: FILENAME:           5  iffp02c2q_WJ2.fits
NARG: FILENAME:           6  iffp02c3q_WJ2.fits
NARG: FILENAME:           7  iffp02c4q_WJ2.fits
NARG: FILENAME:           8  iffp02c5q_WJ2.fits
NARG: FILENAME:           9  iffp02c6q_WJ2.fits
NARG: FILENAME:          10  iffp02c7q_WJ2.fits
NARG: FILENAME:           1  iffp02bhq_WJ2.fits
NARG: FILENAME:           2  iffp02biq_WJ2.fits
NARG: FILENAME:           3  iffp02bjq_WJ2.fits
NARG: FILENAME:           4  iffp02bkq_WJ2.fits
NARG: FILENAME:           5  iffp02blq_WJ2.fits
NARG: FILENAME:           6  iffp02bmq_WJ2.fits
NARG: FILENAME:           7  iffp02bnq_WJ2.fits
NARG: FILENAME:           8  iffp02boq_WJ2.fits
NARG: FILENAME:           9  iffp02bpq_WJ2.fits
NARG: FILENAME:          10  iffp02bqq_WJ2.fits
NARG: FILENAME:          11  iffp02brq_WJ2.fits
NARG: FILENAME:          12  iffp02bsq_WJ2.fits
NARG: FILENAME:          13  iffp02btq_WJ2.fits
NARG: FILENAME:          14  iffp02buq_WJ2.fits
NARG: FILENAME:          15  iffp02bvq_WJ2.fits
NARG: FILENAME:          16  iffp02bwq_WJ2.fits
```

Step 2

Prepare the IN.* files that will be read by the scripts in 01.XYM and 02.CMD directories.

This step is important since the IN.* files in each directory are used to provide input to the backend Fortran scripts.

In [5]:
```
reduce.data_prep(directory)
```

Create MATCHUP files

In the cell below, we will achieve the following objectives:

1. Use the corresponding PSF files for each filter to create an .XYM file for every exposure. A library PSF model is used to generate the .XYM files, which contain the stellar positions and magnitudes for each HST exposure.
2. Generate the output file TRANS.xym2mat, along with a number of MAT.0* files. Note that the first line of IN.xym2mat and files beginning with 00 only define the reference coordinates used for the output. Therefore, the 00 file does not need to be a *_WJ2.xym file.
3. Keep all stars that are found in at least 80% of the exposures for each filter.

In [6]:
```
reduce.matchup_files(directory)
```

```
WARNING: VerifyWarning: Found a SIMPLE card but its format doesn't respect t
he FITS Standard [astropy.io.fits.hdu.hdulist]
```

Once the cell above finishes running, you will have the final MATCHUP files. One is located at 01.XYM/F814W/MATCHUP.F814W.XYM.02 for the F814W filter and the other is at 01.XYM/F606W/MATCHUP.F606W.XYM for the F606W filter.

We will open the MATCHUP files to inspect them further. The MATCHUP file we will open below will be for the F814W folder. You can uncomment the filename_606W line in the cell below to look at the F606W MATCHUP file.

In [7]:
```
filename_814W = Path(directory).resolve()/f"01.XYM/F814W/MATCHUP.F814W.XYM.0
#filename_606W = Path(directory).resolve()/f"01.XYM/F606W/MATCHUP.F606.XYM"

cols = ["xbar", "ybar", "mbar", "xsig", "ysig", "msig", "qbar", "Nf", "Ng",
df = pd.read_csv(filename_814W, sep=r"\s+", comment="#", header=None, names=
```

In [8]:
```
df
```

Out[8]:

	xbar	ybar	mbar	xsig	ysig	msig	qbar	Nf	Ng	Nm	Nm
0	28.0816	81.1979	-9.1099	0.0246	0.0243	0.0695	9.999	10	8	9	
1	41.7676	93.0470	-9.2503	0.0141	0.0173	0.0416	9.999	10	9	10	
2	226.1286	94.6584	-9.3269	0.0121	0.0089	0.0246	9.999	10	9	10	
3	149.2051	97.4896	-7.7537	0.0338	0.0491	0.0525	9.999	9	9	8	
4	317.4594	98.0957	-8.4755	0.0250	0.0340	0.0233	9.999	10	9	10	
...	...	...	...	...	...	...	...	...	...	...	
2254	726.6342	1127.6737	-3.8313	0.0793	0.0669	4.3817	9.999	8	8	8	
2255	945.2009	1131.1170	-9.1359	0.0325	0.0300	0.0314	9.999	10	10	9	
2256	921.8165	1133.3954	-5.6098	0.0935	0.0667	3.2060	9.999	8	8	8	
2257	873.2732	1135.0444	-4.8129	0.0289	0.0247	5.5017	9.999	8	8	8	
2258	960.0142	1137.3410	-8.5449	0.0369	0.0356	0.0197	9.999	8	8	8	

2259 rows × 17 columns

From the MATCHUP file opened above, here are the most important things to note:

1. xbar and ybar correspond to the mean x and y star positions (in pixel coordinates) for stars in the HST images.
2. mbar is the mean instrumental magnitude of the stars (either in F606W or F814W).
3. xsig, ysig, msig are the errors on positions and magnitudes.

The other rows are not necessary to complete these modules. However, for completion, these are what the other columns mean:

1. qbar: Quality of the point source fit. (9.999 = stellar source).
2. Nf: Number of frames the source is detected in.
3. Ng: Number of good detections for astrometry.
4. Nm: Number of good detections for photometry.
5. Nmin: Minimum number of images where a star had to be found in (based on how many exposures it could have been found in)
6. Nstar: Generic star number.
7. pki, pkj: Integer location of the peak pixel values.
8. pkp: Number of peaks that would be possible given the image coverage at that point in the field.
9. pkn: Minimum number of peaks required to be found (based on finding parameters).
10. pku: Number of peaks actually found.

Using the MATCHUP files, we are ready to create a stack of the HST exposures.

Step 3

This step will create a file called "outpuq.fits", which is a stack of your HST exposures. A stack image will be created in both filters (F814W and F606W), you can subsequently use it to identify the pixel location of your target.

In [9]:
```
reduce.run_output_stack(directory)
```

You are now ready to view the output stacks for both filters using DS9.

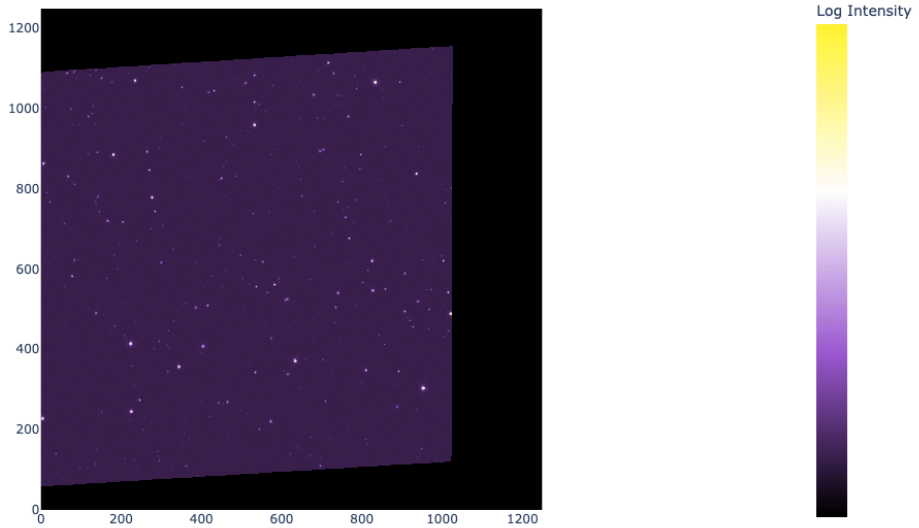
Alternatively, you can view this output stack in the jupyter notebook by running the cell below. The output stack for the F814W filter is kept in 01.XYM/F814W and the output stack for the F606W filter is kept in 01.XYM/F606W. We open the output stack for F814W below. You can uncomment the filename_606W line in the cell below to look at the F606W output stack.

The scaling can be adjusted depending on your preference by changing the argument given to "ImageNormalize" in the cell below.

In [10]:
```
###############
# READ FILES #
###############

filename_814W = Path(directory).resolve()/f"01.XYM/F814W/outputq_F814W.fits"
#filename_606W = Path(directory).resolve()/f"01.XYM/F814W/outputq_F606W.fits

##################
# OPEN FITS FILE #
```

```
###################
hdul = fits.open(filename_814W)
data = hdul[-1].data
hdul.close()
data = np.nan_to_num(data, nan=0.0, posinf=0.0, neginf=0.0)


#########################
# SCALE THE OUTPUT STACK #
#########################
norm = ImageNormalize(data, stretch=AsinhStretch(0.02))
scaled = norm(data)
nonbinary_colors = custom_cmap.nb_colors()

########
# PLOT #
########
fig = px.imshow(scaled, origin='lower', color_continuous_scale=nonbinary_col
fig.update_layout(width=700, height=700, coloraxis_colorbar=dict(title="Log
fig.show()
```

Output Stack (F814W)



In [ ]: