PROJ 201 FINAL REPORT

Project Title: Astrophysical Image Processing using James Webb Space Telescope Observations

Name, Surname & ID of group members:

Kutluhan Aygüzel - 29563

Lütfi Canberk Ilısal - 29501

Ozan Parlayan - 29283

Supervised by: Ersin Göğüş

Month, day, year: 04/01/2023

Abstract

The James Webb Space Telescope (JWST), an advanced astronomical space observation equipment targeted towards infrared wavelengths, represents a significant advancement in astronomy history. The James Webb Space Telescope's sensors, including MIRI, NIRCam, NIRISS, and NIRSpec, may collect remarkable data and images from deep space (NASA, 2022). The project is focused on processing this special data, which comes in the FITS file format from the database Mikulski Archive for Space Telescope (MAST), and interpreting this data to display the stunning images mostly collected from the equipment on the James Webb Space Telescope: NIRcam and MIRI. The primary sources of data used in the project's basic framework are the MIRI and NIRCam instruments.

Through the use of software environments such as Jupyter Notebook IDE, it has been made easier to read data, obtain images from the James Webb Space Telescope and process the data to project modified images. It was necessary to select the right coding environment in addition to the right coding language and libraries. Python is used as the coding language for this project. Python wouldn't be able to handle the demands of processing the data by itself. In order to retrieve, process, modify and project the data, comprehensive libraries such as Astropy, NumPy, Matplotlib, SciPy are used with its imported subtools and modules.

In this project, firstly, information of the FITS files are shown and extension of the FITS file is chosen; in the second part, cutout process is initialized based on the properties of the given image HDU; in the third and fourth parts, images of chosen NIRCam and MIRI extensions are degraded and smoothed by using a matrix technique; at last, in the fifth part, final image of the project which is the ratio of the two different images is projected.

Introduction

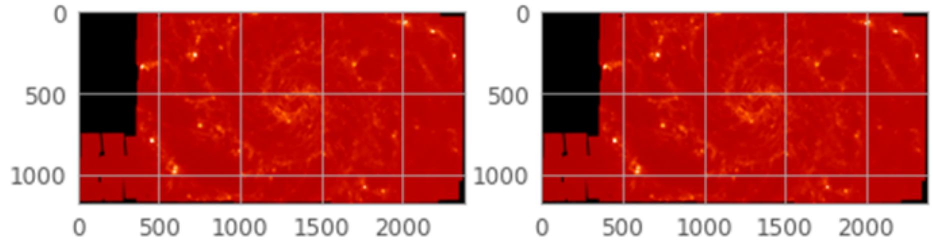
The James Webb space telescope was launched on 7/12/22. The main mission of James Webb Webb's four instruments, the Near-Infrared Camera (NIRCam), Near-Infrared Spectrograph (NIRSpec), Mid-Infrared Instrument (MIRI), and Fine Guidance Sensor/Near InfraRed Imager and Slitless Spectrograph (FGS/NIRISS), may observe anything from objects in our solar system to the early cosmos (NASA, 2022). Our research is focusing on MIRI and NIRCam instruments and the data they are collecting.

The Mid-Infrared Instrument (MIRI), the first instrument in the project, was created by the MIRI Consortium, a team of scientists and engineers from European nations, a group from the Jet Propulsion Lab in California, and scientists from a number of American universities (NASA, 2022). With a wavelength range of 5 to 28 microns, the mid-infrared portion of the electromagnetic spectrum is visible to MIRI's camera and spectrograph. Its sensitive detectors will let it to identify faint comets, freshly developing stars, and the redshifted light of distant galaxies (NASA, 2022). Wide-field, broadband imaging will be provided by the MIRI camera, and medium-resolution spectroscopy will be made possible by the spectrograph, revealing new physical information about the far-off objects it will view. The MIRI's standard working temperature is 7 K. Webb is equipped with a cutting-edge "cryocooler" for cooling MIRI's detectors. The “cryocooler” is crucial because measurements have a smaller margin of error at lower temperatures. There is a two-step procedure instead: The instrument is cooled to 18K with a pulse tube precooler and to 7K with a Joule-Thomson loop heat exchanger.(NASA, 2022).

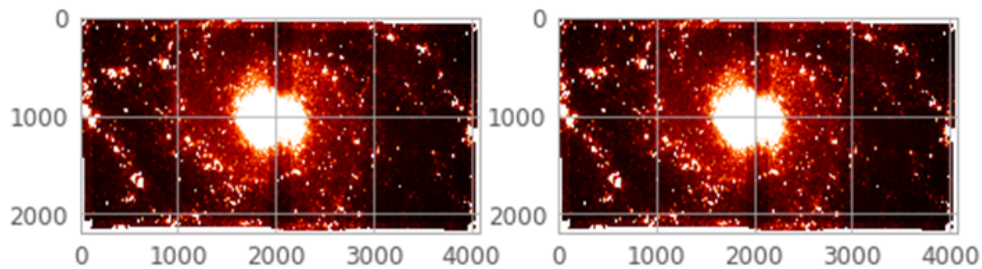
Another instrument used in the project is The Near Infrared Camera, created by Lockheed Martin and the University of Arizona, is another important sensor. Webb's main imager, NIRCam, will have a wavelength range of 0.6 to 5 microns. The number of stars in nearby galaxies, young stars, and the earliest stars and galaxies in the process of formation will all be detected by NIRCam. Coronagraphs, which are tools that let astronomers take photographs of extremely faint objects around a center light source, like star systems, are part of NIRCam. By obstructing the light from a brighter object, NIRCam's coronagraph enables observation of a nearby, dimmer object (NASA, 2022).

Light from the sky is collected by Webb's mirrors and directed at the scientific equipment. Before the light is eventually focused into the detectors, the devices either filter it or spectroscopically disperse it. Every device has a unique detector. In the detectors, photons are taken in and eventually transformed into the electronic voltages that we can measure. Both near-IR HgCdTe and mid-IR Si:As detectors begin the detection process at the same point. A photon that is incident on a semiconductor is absorbed, producing mobile electron hole pairs. These migrate until they arrive at a location where they can be collected under the effect of inherent and added electric fields (Wired, 2022). In order to read a Webb detector's pixels more than once before resetting them, Webb's detectors are utilized. This allows us to average several non-destructive reads instead of just one, which lowers read noise. Another application of Webb is the ability to detect "jumps" in signal level—a telltale sign that a cosmic ray has disrupted a pixel—by utilizing numerous samples of the same pixel. When a cosmic ray has disrupted a pixel, it is possible to apply a correction in ground-based processing to recover a large portion of the scientific value of the afflicted pixel (Wired, 2022).

The JWST actually looks at two ranges of infrared light: the near infrared and mid-infrared. The mid-range infrared is often associated with heat, and that is mostly true. The reason why the JWST looks at infrared light is the Doppler effect. Doppler effect can be observed within the light but since the speed of light is super fast (3 x 108 m/s), the effect is not noticeable in many situations (Wired, 2022). However, because of the expansion of the universe, just about all of the galaxies that it is seen from Earth are moving away from us. So to us, their light appears to have a longer wavelength. It is called a redshift, meaning the wavelengths are more red because they are longer (Wired, 2022).



The MIRI image obtained in the project.



The NIRCam image obtained in the project.

Why JWST uses infrared light has another good reason, that is it's difficult to get an unobstructed view of far-away celestial objects thanks to the gas and dust that are the detritus from old stars. These can scatter visible light more easily than they can infrared wavelengths. Essentially, infrared sensors are able to see through these clouds better than visible light telescopes can (Wired, 2022).

Another perspective on this project that could be provided is the using space coordinates which are RA and DEC. Understanding of the positions of the celestial objects is facilitated by these objects RA and DEC values. As longitude and latitude are to the surface of the Earth, RA (right ascension) and DEC (declination) are to the sky. While Dec measures north/south directions, like latitude, RA corresponds to east/west directions (like longitude). In fact, RA is timed in hours, minutes, and seconds (Saintonge, 2022).

How the team managed to understand the space coordinates was using a module called WCS. Geometric changes between one set of coordinates and another are described by world coordinate systems (WCSs). The mapping of an image's pixels onto the celestial sphere is a typical application. The mapping of pixels to spectrum wavelengths is a further typical use. Furthermore, in the project, the libraries of numpy and matplotlib helped the displaying of the images in the project (Astropy, 2022).

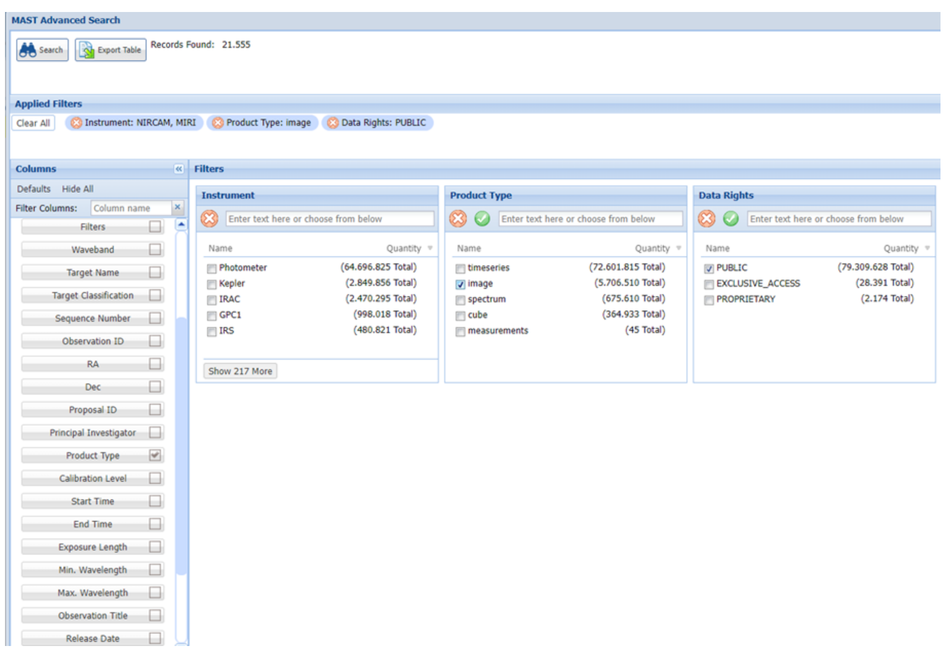
In this research, our purpose is to find the hardness ratio by comparing two photos taken by MIRI and NIRCam, which were obtained by MAST Archive, with different wavelengths in the same area of the space by using Python, Astropy and other libraries. To make it, a code of 6 parts was written by using the Jupyter Notebook IDE which is the most efficient based on our research because other coding IDEs like PyCharm or Python IDLE could not handle processing with FITS files, unlike Jupyter Notebook. Additionally, Jupyter Notebook IDE is reached by a platform called Anaconda Navigator which also provides many other coding environments. When the Jupyter Notebook is launched by Anaconda Navigator, it starts a local server named "http://localhost:8888/tree" and when the Jupyter Notebook is initialized, it reaches the documents in the PC by using an interface in the browser. After that, to create a coding notebook in any directory of the computer is available.

In the Jupyter Notebook, FITS files are handled and the images inside of these files manipulated and modified in a way that hardness ratio could be obtained.

Methods & Materials

MAST Archive

To obtain FITS files, the MAST archive is used . In order to find appropriate FITS files, 3 filters are used. First is Instrument which is used for NIRCAM and MIRI instruments of James Webb Space Telescope, second is product type. As the team works with images the product type should be image. Final filter is data rights. Public FITS files are handled in this project.



After that process, different NIRCAM and MIRI FITS files with same RA and DEC are chosen and downloaded to obtain image of the same space area.

	Actions	Observation T...	Mission	Provenance Name	Instrument	Project	Waveband	RA	Dec
1		science	JWST	CALJWST	NIRCAM	JWST	Infrared	12:26:51.296	+21:52:19.97
2		science	JWST	CALJWST	NIRCAM	JWST	Infrared	12:26:51.296	+21:52:19.97
3		science	JWST	CALJWST	MIRI	JWST	Infrared	12:26:51.296	+21:52:19.97

Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® Distribution that allows you to launch applications and manage conda packages, environments, and channels without using command line interface (CLI) commands. (https://docs.anaconda.com/navigator/index.html#:~:text=Anaconda%20Navigator%20is%20a%20desktop,line%20interface%20(CLI)%20commands.)

Jupyter Notebook

Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience. Through the web-based application, users can create data visualizations and other components of a project to share with others via the platform. (https://www.jetbrains.com/datalore/?source=google&medium=cpc&campaign=16346810819&term=jupyter%20notebook%20python&content=642229598821&gclid=CjwKCAiAh9qdBhAOEiwAvxIok_DFc1-w2omkaMGPi2HLYeULTfi7wegEcZYG5vfLNb7K3CFVx_VjrRoCGYsQAvD_BwE)

Code (See Appendix B)

First Part of Code:

First NumPy package is used because NumPy is a fundamental package for scientific computing (see Appendix A). For the purpose of visualization of the FITS file, firstly matplotlib.pyplot (see Appendix A) package is used for data visualization and graphical plotting. Additionally, for better visualization astropy.visualization.astropy_mpl_style (see Appendix A) package is also integrated.

```
plt.style.use(astropy_mpl_style)
```

```python
from astropy.io import fits
from astropy.visualization import ZScaleInterval
import matplotlib.pyplot as plt
import numpy as np
from astropy.utils.data import get_pkg_data_filename
from astropy.visualization import astropy_mpl_style
from astropy.nddata import Cutout2D
```

After visualization settings, astropy.utils.data.get_pkg_data_filename (see Appendix A) is used for retrieving the data file of NIRCAM and MIRI.

```python
from astropy.utils.data import get_pkg_data_filename
```

After taking the name of NIRCAM and MIRI FITS files from the user, get_pkg_data_filename() is used for retrieving the data file of both NIRCAM and MIRI.

```python
nircam = input("Please enter the name of the NIRCAM FITS file you want to open: ")
miri = input("Please enter the name of the MIRI FITS file you want to open: ")
```

```python
nircam_file = get_pkg_data_filename(nircam)
miri_file = get_pkg_data_filename(miri)
```

The astropy.io.fits (see Appendix A) package provides access to FITS files.

```python
from astropy.io import fits
```

fits.open() is used for accessing both HDU lists for NIRCAM and MIRI.

```python
hdulist_nircam = fits.open(nircam)
hdulist_miri = fits.open(miri)
```

Then .info() is used for displaying Primary HDU for both NIRCAM and MIRI. Other lines for displaying information in an understandable style for the users.

```
print("\n" + "\n")
print("NIRCam")
print(hdulist_nircam.info())
print("\n" + "\n")
print("----------------------
print("\n" + "\n")
print("MIRI")
print(hdulist_miri.info())
print("\n")
```

After displaying Primary HDUs, it is asked to users which image extension they want to see. Also, there is a warning for users that extension numbers must be the same because every extension has a different imaging type.

```
print("Extension numbers must be same.")
extension_nircam = int(input("Which extension image do you want to see in NIRCAM? "+"\n"))
extension_miri = int(input("\n" + "Which extension image do you want to see in MIRI? "+"\n"))
```

Firstly, the HDU from the given extension taken from the user is opened, then the HDU for accessing the header of the given extension is used.

```
hdu_nircam = hdulist_nircam[extension_nircam]
hdu_miri = hdulist_miri[extension_miri]
```

```
header_nircam = hdu_nircam.header
header_miri = hdu_miri.header
```

In order to pull image data from given extensions, fits.getdata() function is used. This function takes two parameters, first one is data file and second one is data file's extension.

```
nircam_image_data = fits.getdata(nircam_file, ext=extension_nircam)
miri_image_data = fits.getdata(miri_file, ext=extension_miri)
```

ZScaleInterval is used for manipulating images for better projection (See Appendix A).

```
from astropy.visualization import ZScaleInterval
```

```
z = ZScaleInterval()
```

Then ZScaleInterval.getlimits() (see Appendix A) is used for determining minimum and maximum values based on NIRCAM or MIRI image data.

plt.subplots() takes two parameters, nrows and ncols, which are the number of rows/columns of the subplot grid.

.imshow() takes three parameters, first image data and min and max values determined by ZScaleInterval. imshow() method returns an image in a window.

```
z_nir1,z_nir2 = z.get_limits(nircam_image_data)
fig_nir, axes_nir = plt.subplots(nrows = 1, ncols =2)
axes_nir[0].imshow(nircam_image_data, vmin = z_nir1, vmax = z_nir2)
axes_nir[1].imshow(nircam_image_data, vmin = z_nir1, vmax = z_nir2)
```

```
z_miri1, z_miri2 = z.get_limits(miri_image_data)
fig_miri, axes_miri = plt.subplots(nrows = 1, ncols =2)
axes_miri[0].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
axes_miri[1].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
```

Second Part of Code:

Assigning x and y values which are obtained by a given extension of FITS file for NIRCAM and MIRI image.

```
x_miri_center = 1300.003016955477
y_miri_center = 589.933975818463

x_nircam_center =1900.575878822971
y_nircam_center =1089.325394456016
```

Assigning position values of NIRCAM and MIRI cutout center.

```
position_nircam = (x_nircam_center, y_nircam_center)
position_miri = (x_miri_center, y_miri_center)
```

Assigning the CutOut2D size for NIRCAM and MIRI. In order to reach these values, first, the nominal pixel area in steradians (PIXAR_SR) in the given extension's header file is found. The steradian is the unit of solid angle in the International System of Units (https://en.wikipedia.org/wiki/Steradian).

Dividing PIXAR_SR of MIRI and PIXAR_SR of NIRCAM gives us the nominal pixel area ratio of MIRI and NIRCAM.

PIXAR_SR of MIRI is 2.84403609523084E-13

PIXAR_SR of NIRCAM is 9.34E-14

Nominal pixel area ratio of MIRI and NIRCAM is 3.0450065259430835117773019271949

Square root of nominal pixel area ratio of MIRI and NIRCAM is
1.7449947065659206496919411701204

The square root of the nominal pixel area ratio of MIRI and NIRCAM is taken to obtain the pixel length ratio of MIRI and NIRCAM. For simplicity, the pixel length ratio is taken as 1.7 which means 10 MIRI pixel for 17 NIRCAM pixel length for x and y axis.

Also, in the header file of MIRI and NIRCAM, it is seen that the number of pixels in the axis x and y direction of MIRI is 1024 and the number of pixels in the axis x and y direction of NIRCAM is 2048. Maximum common multiple of 10 and 17 that do not exceed the number of pixels of MIRI or NIRCAM is 102. Thus, 10*102 = 1020 pixels for MIRI and 17*102=1734 pixels for NIRCAM.

```
size_nircam = (1734,1734)
size_miri = (1020,1020)
```

astropy.nddata.Cutout2D() (see Appendix A) creates a cutout object from a 2D array.

```
from astropy.nddata import Cutout2D
```

Cutout2D() takes three parameters, image data, cutout center and cutout size then returns the edited image.

```
cutout_nircam = Cutout2D(nircam_data, position_nircam, size_nircam)
cutout_miri = Cutout2D(miri_data, position_miri, size_miri)
```

Third Part of Code:

Edited NIRCAM image that is obtained consists of a 1734x1734 matrix. For the downgrading operation first, a new 1734x102 matrix is created and added each and every 17 consecutive pixel values and divided by 17 then, assigned these new values to the new matrix.

In the second part, the same operation performed lately created a 1734x102 matrix, then a 102x102 matrix was obtained, which means our downgraded NIRCAM image.

Also, at the same time, the maximum density value of the 102x102 matrix is determined in order to create a normalized image later.

```
new_nircam = [[0 for y in range(102)] for x in range(1734)]
for k in range (1734):
    for j in range(102):
        sum_nircam = float(0)
        for i in range(j * 17, (j + 1) * 17):
            sum_nircam += float(cutout_nircam.data[k][i])
        new_density_nircam = sum_nircam / 17
        new_nircam[k][j] = new_density_nircam

new_new_nircam = [[0 for y in range(102)] for x in range(102)]
maxvalue_nircam = 0


for c in range(102):
    for b in range(102):
        sum2_nircam = float(0)
        for a in range(b * 17, (b + 1) * 17):
            sum2_nircam += float(new_nircam[a][c])
        new_density2_nircam = sum2_nircam / 17
        new_new_nircam[b][c] = new_density2_nircam


        if maxvalue_nircam < new_density2_nircam:
            maxvalue_nircam = new_density2_nircam
```

Fourth Part of Code:

The same procedure followed for MIRI but the only difference is that at NIRCAM, 17 pixels are selected to add up, however at MIRI, 10 pixels are selected for the downgrading process.

```
new_miri = [[0 for y in range(102)] for x in range(1020)]
for k in range (1020):
    for j in range(102):
        sum_miri = float(0)
        for i in range(j * 10, (j + 1) * 10):
            sum_miri += float(cutout_miri.data[k][i])
        new_density_miri = sum_miri / 10
        new_miri[k][j] = new_density_miri

maxvalue_miri = 0

new_new_miri = [[0 for y in range(102)] for x in range(102)]
for c in range(102):
    for b in range(102):
        sum2_miri = float(0)
        for a in range(b * 10, (b + 1) * 10):
            sum2_miri += float(new_miri[a][c])
        new_density2_miri = sum2_miri / 10
        new_new_miri[b][c] = new_density2_miri


        if maxvalue_miri < new_density2_miri:
            maxvalue_miri = new_density2_miri
```

Fifth Part of Code:

New matrices are created for the normalization process, then every MIRI and NIRCAM density value is divided by their maximum value, finally assigned to new matrices.

```python
new_new_nircam2 = [[0 for y in range(102)] for x in range(102)]
new_new_miri2 = [[0 for y in range(102)] for x in range(102)]
sum1 = float(0)
for i in range (102):
    for j in range (102):
        new_new_miri2[i][j] = new_new_miri[i][j] / maxvalue_miri
        new_new_nircam2[i][j] = new_new_nircam[i][j] / maxvalue_nircam
```

Sixth Part of Code:

For the final image, MIRI and NIRCAM matrices are divided and assigned to new matrices.

```python
final_image = [[0 for y in range(102)] for x in range(102)]
for i in range (102):
    for j in range (102):
        l = new_new_miri2[i][j] / new_new_nircam2[i][j]
        final_image[i][j] = l
```

Results

In this project, as it aimed in the proposal report, the density of images for James Webb Telescope's Miri and NirCam instruments was found.

In this research, our purpose is to find the hardness ratio by comparing two photos taken by MIRI and NIRCam with different wavelengths in the same place via Python and Astropy library.

To be able to find the hardness ratio of the FITS file images, there must be some former steps before calculating it. First thing that should be done is including the FITS files into the project. To be able to include FITS files into the project, the files must be in the same directory in your computer with the project you opened in the Jupyter Notebook. In the following step files are read and HDU informations are displayed so that the user can choose the Image HDU extension that will be used in the continuing steps.

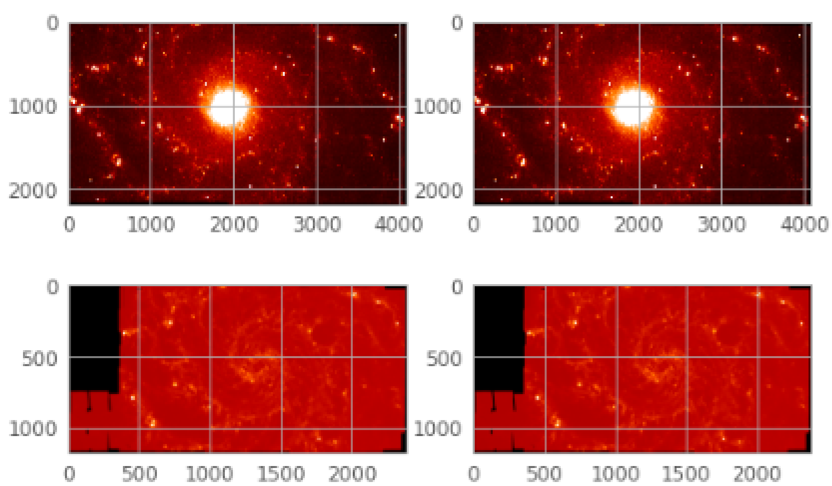
```
NIRCam
Filename: jw02107-o040_t018_nircam_clear-f335m_i2d.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     369   ()
  1  SCI           1 ImageHDU        75   (4079, 2190)   float32
  2  ERR           1 ImageHDU        10   (4079, 2190)   float32
  3  CON           1 ImageHDU         9   (4079, 2190)   int32
  4  WHT           1 ImageHDU         9   (4079, 2190)   float32
  5  VAR_POISSON    1 ImageHDU         9   (4079, 2190)   float32
  6  VAR_RNOISE    1 ImageHDU         9   (4079, 2190)   float32
  7  VAR_FLAT      1 ImageHDU         9   (4079, 2190)   float32
  8  HDRTAB        1 BinTableHDU    816   8R x 403C   [23A, 5A, 3A, 48A, 7A, 13A, 6A, 5A, 7A, 10A, 4A, L, D, D, D, D, 32A, 48A,
70A, 11A, 2A, D, 47A, D, 10A, 12A, 23A, 23A, 26A, 11A, 5A, 3A, 3A, 2A, 1A, 2A, 1A, L, 14A, 13A, 2A, 26A, 20A, 27A, 10A, K, L,
L, L, L, 7A, 7A, 5A, D, D, D, D, D, D, 27A, D, D, D, 6A, 8A, 1A, 4A, 5A, 5A, L, D, D, D, D, D, D, D, D, D, D, D, D, 4A, D, D,
D, D, D, D, D, D, D, K, 5A, 9A, D, D, D, D, D, D, D, D, D, 7A, D, D, K, K, D, D, K, K, D, D, K, K, K, K, K, D, D, D, D, D, D,
D, D, K, K, L, L, K, K, D, D, D, D, D, D, D, 4A, K, K, K, K, K, K, D, D, D, D, 4A, D, D, K, D, K, D, D, D, D, D, D, D, D, D, D,
D, D, D, D, D, D, D, 7A, 10A, D, D, D, D, D, D, D, D, D, D, D, D, D, 10A, 10A, D, D, D, D, D, D, D, D, D, D, D, D, K, K, D, 4A,
K, K, K, D, 4A, K, K, K, D, 4A, K, D, D, K, 27A, 27A, 10A, D, D, D, D, D, D, D, 9A, 27A, D, D, D, D, D, D, D, 8A, 14A, 33A, D,
D, 3A, 3A, D, 33A, 3A, 39A, D, D, 41A, 33A, 3A, 3A, 3A, 3A, 3A, D, 33A, 3A, 3A, 3A, D, D, 38A, 33A, 3A, 3A, D, 35A, 35A, D, 38
A, D, 3A, D, D, D, D, 39A, D, D, D, 3A, D, 38A, D, 40A, 37A, D, D, D, 3A, D, D, D, D, D, 8A, D, D, D, D, D, 8A, 8A, D, D, D, D,
D, 8A, D, 7A, 7A, D, D, 7A, 8A, D, D, 8A, D, D, D, 8A, D, 8A, 8A, 8A, 8A, D, D, D, 8A, 8A, D, D, D, D, 8A, D, 8A, D, D, D, 5A,
D, L, 6A, D, D, D, D, 4A, D, D, D, K, D, D, D, D, D, D, 12A, 12A, D, 3A, 3A, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D,
117A, D, D, D, D, D, D, K, D, D, D, D]
  9  ASDF          1 BinTableHDU     11   1R x 1C   [36706B]
None


-----------------------------------------------------------------------------------------------------------------------
```

```
MIRI
Filename: jw02107-o039_t018_miri_f1130w_i2d.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
  0  PRIMARY       1 PrimaryHDU     349   ()
  1  SCI           1 ImageHDU        75   (2379, 1178)   float32
  2  ERR           1 ImageHDU        10   (2379, 1178)   float32
  3  CON           1 ImageHDU         9   (2379, 1178)   int32
  4  WHT           1 ImageHDU         9   (2379, 1178)   float32
  5  VAR_POISSON    1 ImageHDU         9   (2379, 1178)   float32
  6  VAR_RNOISE    1 ImageHDU         9   (2379, 1178)   float32
  7  VAR_FLAT      1 ImageHDU         9   (2379, 1178)   float32
  8  HDRTAB        1 BinTableHDU    816   12R x 403C   [23A, 5A, 3A, 48A, 7A, 13A, 6A, 5A, 7A, 10A, 4A, L, D, D, D, D, 32A, 48
A, 70A, 11A, 2A, D, 47A, D, 10A, 12A, 23A, 23A, 26A, 11A, 5A, 3A, 3A, 2A, 1A, 2A, 1A, L, 12A, 6A, 2A, 26A, 20A, 27A, 10A, K, L,
L, L, L, 7A, 7A, 5A, D, D, D, D, D, D, 27A, D, D, D, 4A, 8A, D, D, 6A, D, D, D, D, D, D, D, 4A, D, D, D, D, D, 3A, 4A, D, D, D,
D, D, D, D, D, D, K, 5A, 9A, D, D, D, D, D, D, D, D, D, 6A, D, D, K, K, D, D, K, K, D, D, K, K, K, K, K, D, D, D, D, D, D, D,
D, K, K, L, L, K, K, D, D, D, D, D, D, D, 4A, K, K, K, K, K, K, D, D, D, D, 12A, D, D, K, D, K, D, D, D, D, D, D, D, D, D, D,
D, D, D, D, D, D, D, 7A, 10A, D, D, D, D, D, D, D, D, D, D, D, D, D, 10A, 11A, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D,
D, D, D, D, D, D, D, D, D, D, D, D, D, K, 27A, 27A, 10A, D, D, D, D, D, D, D, 9A, 27A, D, D, D, D, D, D, D, 8A, 14A, 31A, D, D,
3A, 3A, D, 31A, 3A, 37A, D, D, 39A, 31A, 3A, 3A, 3A, 3A, 3A, D, 31A, 3A, 3A, 3A, D, D, 36A, 31A, 3A, 3A, D, D, 33A, D, 36A, D,
3A, D, D, 32A, 31A, 37A, D, D, D, 3A, D, D, D, D, D, D, D, D, 3A, D, D, D, D, D, 8A, D, D, D, D, D, 8A, 8A, D, D, D, D, 8A, 8A,
D, 7A, 7A, D, D, 7A, 8A, D, 8A, 8A, D, D, D, 8A, D, D, 8A, 8A, 8A, D, 8A, 8A, 8A, 8A, D, D, D, D, D, D, 8A, D, D, D, 5A, D, L,
6A, D, D, D, D, 4A, D, D, D, K, D, D, D, D, D, D, 12A, 12A, D, 3A, 3A, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, D, 117A,
D, D, D, D, D, D, K, D, D, D, D]
  9  ASDF          1 BinTableHDU     11   1R x 1C   [36428B]
None


Extension numbers must be same.
Which extension image you want to see in NIRCAM?
1

Which extension image you want to see in MIRI?
1
First image is NIRCam image
Last image is MIRI image
```



After these steps, to find the hardness ratio, the areas of the images of MIRI and NIRCam needed to be determined and these areas must refer to the area of that hardness ratio desired to be calculated. The area is calculated by carefully considering the pixel values of both images. In the continuation, the center value of this area is put into the code manually by assigning the values into the variables of *x_miri_center*, *y_miri_center* for MIRI image and *x_nir_center*, *y_nir_center* for NIRCam image. After that, the values of cutout sizes must be assigned to the values of *size_nircam* and *size_miri* but when assigning values, the amount of pixels per RA and DEC values must be considered carefully. In this project, the sample values for the mentioned variables above are:

x_miri_center = 1300.003016955477

y_miri_center = 589.933975818463

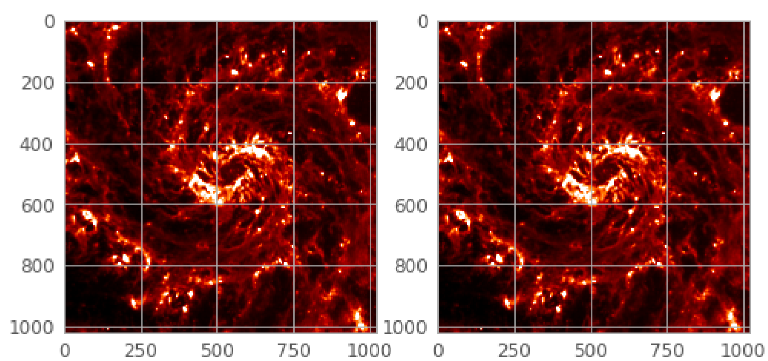
x_nir_center = 1900.575878822971

y_nir_center = 1089.325394456016

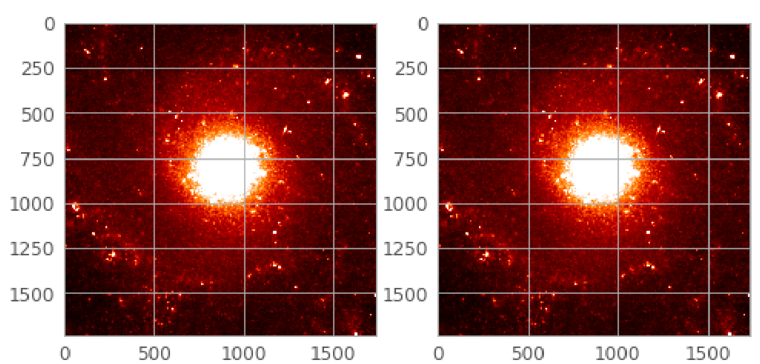
size_nircam = (1734,1734)

size_miri = (1020,1020)

After completing all these steps, run the code and the cutout images will be projected on the interface of Jupyter Notebook as below:

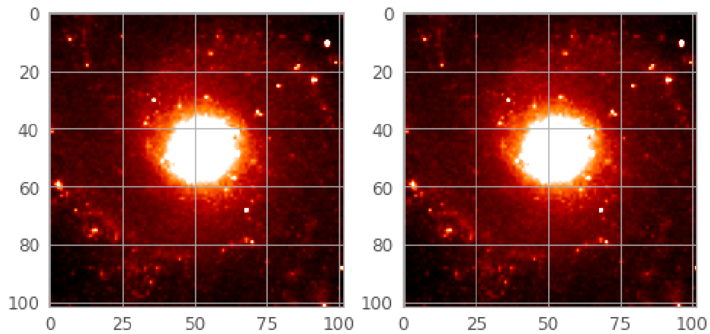


The CutOut 2D MIRI Image

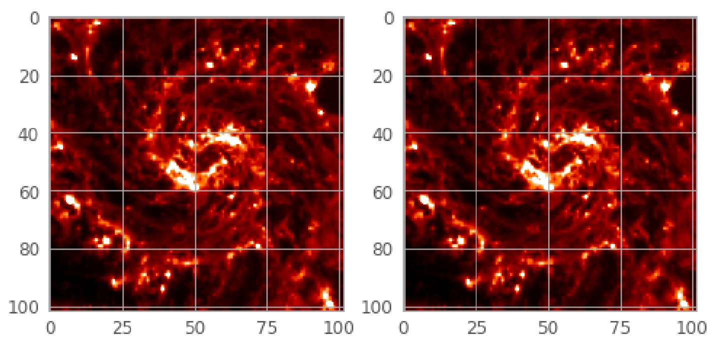


The CutOut 2D NIRCam Image

However, the pixel values, in other words the matrix sizes of images are not the same. To have the hardness ratio, the matrix sizes have to be the same because, otherwise this kind of a hardness ratio is not correct. Therefore, another step is applied which are the **Third Part of Code** and **Fourth Part of Code** (see Appendix B). In these steps, the smoothing (degrading) process for the matrices is applied. By smoothing the matrix, the average of the matrix values are calculated and assigned to a new matrix. This process is applied for both images and the new matrix that values are assigned into has the same size for both images. As a result, two different images with the same matrix sizes could be obtained.



The degraded NIRCam CutOut2D Image

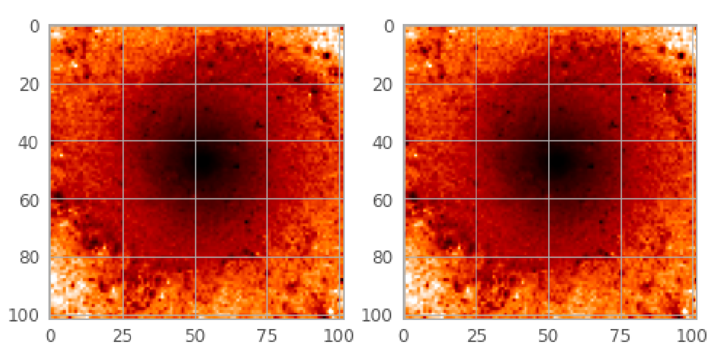


The degraded MIRI CutOut2D Image

Before the last step the matrix values for both matrices are divided by their maximum values to have a normalized matrix.

As a last step, these two matrices are divided with each other; thus, a matrix which can represent the hardness ratio is obtained. Even the image of this matrix is obtained.

The matrix of hardness ratio and its projection



Discussion and Conclusion

The conclusion is that after dividing the two matrices, a final matrix and its projection is obtained. In this projection, it can be clearly seen that the center of the image is dark and it has a spiral pattern towards the center. What this image is showing to us is that when two matrices are divided with glowy centers in them, the center values of this new division matrix would be lower than other parts of the matrix because before the division, the two cut out images have the centers with high density values as a result of glowy image center. In other words, the values of the center area of these two cut out and degraded images has less variance than other parts of the image and two images' center values are closer because they both glow and have high density values. Consequently, the result of the division of two matrices gives lower values in the center area of the new matrix. Because of this situation, the center part of the new image has lower density values which means less glow in the center part.

When considering other parts, the variance of the values increases because MIRI and NIRCam images represent mostly different values except their center values. Like it is said, the center parts of the two images are glowing as a result of the heat that is stemming from the high interaction between the atoms in space as a result of the gravitational force. However, other parts of the images are not glowing for a specified, absolute area unlike the image center, the matrix values in the other parts of both images vary with noticeable spiral patterns in the images. Because of this variation, the colors of this new matrix come up with the colors of mostly red, orange and white colors with some black spots on which means there are some high matching of the values in the two matrices.

Another conclusion that can be made by using the central color of this new image is that, if different FITS files are imported and if the final image would come up with all glow, it could mean the glowy parts of the imported FITS files are not matched, if there is a glowy part for each image like a center of a spiral. Also, if there is a pattern in each imported FITS image and it could be seen as it is seen in the original images, then it means that the final matrix is sufficient to have an idea about whether the aligning process of the two images is successful or not.

The conclusion that can be made of the hardness ratio by looking at this final image is that the dark parts are actually representing the high energy areas in the given content and the brighter parts of the images show the low energy levels. How this conclusion is made is that the gravitational force causes the pulling of any object around it if there is no stronger gravitational force in space. Another thing for gravitational force is that when it pulls the objects around, it results in making the objects (atoms) interact with each other at the molecular level and emitting heat and energy in space. This energy and heat is detectable by JWST as a result of its structure and it can be seen from the images. JWST detects these high energy and heat and in the images, these high energy and heat is seen as glow. As a result, when calculating and projecting the hardness ratio with the final image, the darker areas represent higher energy areas and brighter areas represent lower energy areas in space.

So, the obtained final image can give an idea about the hardness ratio of the two images that is showing the same area in space and coming from different JWST instruments, MIRI and NIRCam.

References

Astropy. (2022). *World Coordinate System.* Retrieved November 4, 2022, from https://docs.astropy.org/en/stable/wcs/index.html

Astropy. (2022). Getting Started. Retrieved November 4, 2022, from https://docs.astropy.org/en/stable/index.html

Saintonge, A. (2022). *What are Ra and Dec?* Retrieved November 4, 2022, from http://curious.astro.cornell.edu/about-us/112-observational-astronomy/stargazing/technical-questions/699-what-are-ra-and-dec-intermediate#:~:text=RA%20(right%20ascension)%20and%20DEC,hours%2C%20minutes%2C%20and%20seconds

Nasa. (2022). Mid-Infrared Instrument (MIRI). Retrieved November 3, 2022, from https://webb.nasa.gov/content/observatory/instruments/miri.html

Nasa (2022). Near Infrared Camera (NIRCAM). Retrieved November 3, 2022, from

https://webb.nasa.gov/content/observatory/instruments/nircam.html

Nasa (2022). Infrared Detectors. Retrieved November 3, 2022, from https://webb.nasa.gov/content/about/innovations/infrared.html

Wired. (2022). The Physics of the James Webb Space Telescope. Retrieved November 3, 2022, from https://www.wired.com/story/the-physics-of-the-james-webb-space-telescope/

Nasa (2022). The James Webb Space Telescope Instruments. Retrieved November 3, 2022, from: https://www.nasa.gov/mission_pages/webb/instruments/index.html

The Astropy Developers (12 Dec 2022). ZScaleInterval. Retrieved January 4, 2023, from: https://docs.astropy.org/en/stable/api/astropy.visualization.ZScaleInterval.html

The Astropy Developers (24 Sep 2015). FITS File handling. Retrieved January 4, 2023, from: https://het.as.utexas.edu/HET/Software/Astropy-1.0/io/fits/index.html#:~:text=The%20astropy.,to%20store%20images%20and%20tables.

Hunter, J., Dale, D., Firing, E., Droettboom, M. & Matplotlib development team (2012-2022). Retrieved January 4, 2023, from: https://matplotlib.org/stable/tutorials/introductory/pyplot.html

NumPy Developers (2008-2022). What is NumPy? Retrieved January 4, 2023, from: https://numpy.org/doc/stable/user/whatisnumpy.html

The Astropy Developers (12 Dec 2022).get_pkg_data_filename. Retrieved January 4, 2023, from: https://docs.astropy.org/en/stable/api/astropy.utils.data.get_pkg_data_filename.html

The Astropy Developers (05 Jul 2017). Data Visualization. Retrieved January 4, 2023, from: https://eteq.github.io/astropy/visualization/index.html

The Astropy Developers (12 Dec 2022).Cutout2D. Retrieved January 4, 2023, from: https://docs.astropy.org/en/stable/api/astropy.nddata.utils.Cutout2D.html

The Scipy community (11 May 2014). scipy.ndimage.interpolation.rotate. Retrieved January 4, 2023, from: https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.interpolation.rotate.html

Appendix A

The Methods of Libraries

ZScaleInterval is a kind of visual interpretation tool for FITS file images and useful for manipulating in a good way in the phase of image projection.

ZScaleInterval.getlimits() is a method of ZScaleInterval that returns the minimum and maximum value in the interval based on the values provided.

astropy.io.fits() is a module that provides access to FITS files.

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

numpy(NumPy) is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

astropy.utils.data.get_pkg_data_filename retrieves a data file from the standard locations for the package and provides a local filename for the data.

astropy.visualization provides functionality that can be helpful when visualizing data. This includes a framework for plotting Astronomical images with coordinates with Matplotlib (previously the standalone wcs axes package), functionality related to image normalization (including both scaling and stretching), smart histogram plotting, RGB color image creation from separate images, and custom plotting styles for Matplotlib.

astropy.visualization.astropy_mpl_style improves some settings over the matplotlib default style.

astropy.nddata.Cutout2D creates a cutout object from a 2D array. The returned object will contain a 2D cutout array.

scipy.ndimage.interpolation.rotate() rotates an array.

Appendix B

First Part of Code

```
#imported modules
from astropy.io import fits
from astropy.visualization import ZScaleInterval
import matplotlib.pyplot as plt
import numpy as np
from astropy.utils.data import get_pkg_data_filename
from astropy.visualization import astropy_mpl_style
from astropy.nddata import Cutout2D
from scipy.ndimage.interpolation import rotate

z = ZScaleInterval() #assigning a module into a variable
plt.style.use(astropy_mpl_style) #making MatPlotLib use different visualization style

nircam = input("Please enter the name of the NIRCAM FITS file you want to open: " + "\n")
#entering the NIRCam file name
miri = input("Please enter the name of the MIRI FITS file you want to open: " + "\n") #entering
the MIRI file name

nircam_file = get_pkg_data_filename(nircam) #retrieving the data file of NIRCam
miri_file = get_pkg_data_filename(miri) #retrieving the data file of MIRI

hdulist_nircam = fits.open(nircam) #opening NIRCam FITS file
hdulist_miri = fits.open(miri) #opening MIRI FITS file

#displaying the information in an understandable style
print("\n" + "\n")
print("NIRCam")
print(hdulist_nircam.info()) #showing the Primary HDU information of NIRCam
print("\n" + "\n")
print("----------------------------------------------------------------------------------------------------------
---------------")
print("\n" + "\n")
print("MIRI")
print(hdulist_miri.info()) #showing the Primary HDU information of MIRI
print("\n")
```

```
#entering the extension numbers
print("Extension numbers must be same.")
extension_nircam = int(input("Which extension image do you want to see in NIRCAM? "+"\n"))
#defining the HDU extension of NIRCam
extension_miri = int(input("\n" + "Which extension image do you want to see in MIRI? "+"\n"))
#defining the HDU extension of MIRI

hdu_nircam = hdulist_nircam[extension_nircam] #opening the HDU from the given extension of NIRCam
hdu_miri = hdulist_miri[extension_miri] #opening the HDU from the given extension MIRI

header_nircam = hdulist_nircam[extension_nircam].header #obtaining header of the given NIRCam extension
header_miri = hdulist_miri[extension_miri].header #obtaining header of the given MIRI extension

nircam_image_data = fits.getdata(nircam_file, ext=extension_nircam) #pulling the image data from the given NIRCam extension
miri_image_data = fits.getdata(miri_file, ext=extension_miri) #pulling the image data from the given MIRI extension

#projecting the NIRCam Image HDU
z_nir1,z_nir2 = z.get_limits(nircam_image_data) #getting the min and max values of the data
fig_nir, axes_nir = plt.subplots(nrows = 1, ncols =2) #defining subplots
axes_nir[0].imshow(nircam_image_data, vmin = z_nir1, vmax = z_nir2) #projecting image
axes_nir[1].imshow(nircam_image_data, vmin = z_nir1, vmax = z_nir2) #projecting image

#projecting the MIRI Image HDU
z_miri1, z_miri2 = z.get_limits(miri_image_data) #getting the min and max values of the data
fig_miri, axes_miri = plt.subplots(nrows = 1, ncols =2) #defining subplots
axes_miri[0].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2) #projecting image
axes_miri[1].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2) #projecting image

print("First image is NIRCam image" + "\n" + "Last image is MIRI image")

#name of the FITS files
#jw02107-o040_t018_nircam_clear-f335m_i2d.fits
#jw02107-o039_t018_miri_f1130w_i2d.fits
```

Second Part of Code

```
#CutOut2D initializing
x_miri_center = 1300.003016955477 #assigning the x value of center of MIRI image
y_miri_center = 589.933975818463 #assigning the y value of center of MIRI image

x_nircam_center =1900.575878822971 #assigning the x value of center of NIRCam image
y_nircam_center =1089.325394456016 #assigning the y value of center of NIRCam image

print("Your cutout NIRCam center is", "(", x_nircam_center,",", y_nircam_center, ")", "and
MIRI cutout center is",
    "(", x_miri_center,",", y_miri_center, ")") #printing center values to inform the user

nircam_data = fits.getdata(nircam_file, ext = extension_nircam) #pulling data from NIRCam
FITS from given extension
miri_data = fits.getdata(miri_file, ext = extension_miri) #pulling data from MIRI FITS from
given extension

position_nircam = (x_nircam_center, y_nircam_center) #assigning position values of NIRCam
cutout center
position_miri = (x_miri_center, y_miri_center) #assigning position values of MIRI cutout center

size_nircam = (1734,1734) #assigning the CutOut2D size for NIRCam
size_miri = (1020,1020) #assigning the CutOut2D size for MIRI

cutout_nircam = Cutout2D(nircam_data, position_nircam, size_nircam) #Cutting out the
NIRCam FITS file
cutout_miri = Cutout2D(miri_data, position_miri, size_miri) #Cutting out the MIRI FITS file

print("Here is your cutout images")
print("First one is MIRI" + "\n" + "Last one is NIRCam")

#projecting the MIRI cutout image and NIRCam cutout image
z_miri1, z_miri2 = z.get_limits(miri_image_data)
fig_miri, axes_miri = plt.subplots(nrows = 1, ncols =2)
axes_miri[0].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
axes_miri[1].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
```

```
z_miri1, z_miri2 = z.get_limits(cutout_miri.data)
fig_mir, axes_mir = plt.subplots(nrows = 1, ncols =2)
axes_miri[1].imshow(cutout_miri.data, vmin = z_miri1, vmax = z_miri2)
axes_miri[0].imshow(cutout_miri.data, vmin = z_miri1, vmax = z_miri2)

z_nir1, z_nir2 = z.get_limits(cutout_nircam.data)
fig_nir, axes_nir = plt.subplots(nrows = 1, ncols =2)
axes_nir[1].imshow(cutout_nircam.data, vmin = z_nir1, vmax = z_nir2)
axes_nir[0].imshow(cutout_nircam.data, vmin = z_nir1, vmax = z_nir2)
```

Third Part of Code

```
#degrading the values of NIRCam image
#degrading NIRCam in y axis
new_nircam = [[0 for y in range(102)] for x in range(1734)] #initializing a matrix
for k in range (1734):
    for j in range(102):
        sum_nircam = float(0) #initializing the NIRCam sum value
        for i in range(j * 17, (j + 1) * 17):
            sum_nircam += float(cutout_nircam.data[k][i]) #summing up the values
        new_density_nircam = sum_nircam / 17 #assigning the value of new density matrix
        new_nircam[k][j] = new_density_nircam #assigning the last version matrix to the new
density matrix

new_new_nircam = [[0 for y in range(102)] for x in range(102)] #initializing a new matrix
maxvalue_nircam = 0 #initializing the max value of NIRCam data

#degrading NIRCam in x axis
for c in range(102):
    for b in range(102):
        sum2_nircam = float(0) #initializing the second NIRCam sum value
        for a in range(b * 17, (b + 1) * 17):
            sum2_nircam += float(new_nircam[a][c]) #summing up the values
        new_density2_nircam = sum2_nircam / 17 #assigning the value of new density matrix
        new_new_nircam[b][c] = new_density2_nircam #assigning the last version matrix to the
new density matrix

        #finding the max value of the NIRCam data densities
        if maxvalue_nircam < new_density2_nircam:
```

```
        maxvalue_nircam = new_density2_nircam

print("Max value of NIRCam image data is ", maxvalue_nircam, "\n") #printing the max
NIRCam data value

print("Here is your degraded NIRCam image with the original cutout NIRCam image")

#projecting the degraded cutout NIRCam image and original cutout NIRCam image
z_nir1, z_nir2 = z.get_limits(cutout_nircam.data)
fig_nir, axes_nir = plt.subplots(nrows = 1, ncols =2)
axes_nir[1].imshow(cutout_nircam.data, vmin = z_nir1, vmax = z_nir2)
axes_nir[0].imshow(cutout_nircam.data, vmin = z_nir1, vmax = z_nir2)

z_nir1, z_nir2 = z.get_limits(new_new_nircam)
fig_nir, axes_nir = plt.subplots(nrows = 1, ncols =2)
axes_nir[1].imshow(new_new_nircam, vmin = z_nir1, vmax = z_nir2)
axes_nir[0].imshow(new_new_nircam, vmin = z_nir1, vmax = z_nir2)
```

Fourth Part of Code

```
#degrading the values of MIRI image
#degrading MIRI in y axis
new_miri = [[0 for y in range(102)] for x in range(1020)] #initializing a matrix
for k in range (1020):
    for j in range(102):
        sum_miri = float(0) #initializing the MIRI sum value
        for i in range(j * 10, (j + 1) * 10):
            sum_miri += float(cutout_miri.data[k][i]) #summing up the values
        new_density_miri = sum_miri / 10 #assigning the value of new density matrix
        new_miri[k][j] = new_density_miri #assigning the last version matrix to the new density
matrix

maxvalue_miri = 0 #initializing the max value of MIRI data

new_new_miri = [[0 for y in range(102)] for x in range(102)] #initializing a new matrix
for c in range(102):
    for b in range(102):
        sum2_miri = float(0) #initializing the second MIRI sum value
        for a in range(b * 10, (b + 1) * 10):
            sum2_miri += float(new_miri[a][c]) #summing up the values
```

```
        new_density2_miri = sum2_miri / 10 #assigning the value of new density matrix
        new_new_miri[b][c] = new_density2_miri #assigning the last version matrix to the new
density matrix

        #finding the max value of the NIRCam data densities
        if maxvalue_miri < new_density2_miri:
            maxvalue_miri = new_density2_miri

print("Max value of MIRI image data is ", maxvalue_miri, "\n") #printing the max value of the
matrix
print("Here is your degraded MIRI image with the original cutout MIRI image")

#projecting the original cutout MIRI image and degraded cutout MIRI image
z_miri1, z_miri2 = z.get_limits(miri_image_data)
fig_miri, axes_miri = plt.subplots(nrows = 1, ncols =2)
axes_miri[0].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
axes_miri[1].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)

z_miri1, z_miri2 = z.get_limits(new_new_miri)
fig_mir, axes_mir = plt.subplots(nrows = 1, ncols =2)
axes_miri[1].imshow(new_new_miri, vmin = z_miri1, vmax = z_miri2)
axes_miri[0].imshow(new_new_miri, vmin = z_miri1, vmax = z_miri2)

z_miri1, z_miri2 = z.get_limits(miri_image_data)
fig_miri, axes_miri = plt.subplots(nrows = 1, ncols =2)
axes_miri[0].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)
axes_miri[1].imshow(miri_image_data, vmin = z_miri1, vmax = z_miri2)

z_miri1, z_miri2 = z.get_limits(cutout_miri.data)
fig_mir, axes_mir = plt.subplots(nrows = 1, ncols =2)
axes_miri[1].imshow(cutout_miri.data, vmin = z_miri1, vmax = z_miri2)
axes_miri[0].imshow(cutout_miri.data, vmin = z_miri1, vmax = z_miri2)
```

Fifth Part of Code

```
#normalization of MIRI and NIRCAM image
new_new_nircam2 = [[0 for y in range(102)] for x in range(102)]  #initialize a new matrix
new_new_miri2 = [[0 for y in range(102)] for x in range(102)]  #initialize a new matrix
for i in range (102):
    for j in range (102):
```

        new_new_miri2[i][j] = new_new_miri[i][j] / maxvalue_miri #assigning the last version matrix to the new density matrix
        new_new_nircam2[i][j] = new_new_nircam[i][j] / maxvalue_nircam #assigning the last version matrix to the new density matrix

Sixth Part of Code

```
#creating the final image
final_image = [[0 for y in range(102)] for x in range(102)] #initialize a new matrix
for i in range (102):
    for j in range (102):
        l = new_new_miri2[i][j] / new_new_nircam2[i][j] #dividing every value of MIRI to NIRCam values
        final_image[i][j] = l #assigning the division result to a variable

#projecting the final image
z_final1, z_final2 = z.get_limits(final_image)
fig_final, axes_final = plt.subplots(nrows = 1, ncols =2)
axes_final[0].imshow(final_image, vmin = z_final1, vmax = z_final2)
axes_final[1].imshow(final_image, vmin = z_final1, vmax = z_final2)
```