Deblurring and Enhancing Astrophotographs Using PSF Based Digital Image Processing

Group Number: 19

Weekly Report 4

Group Members:

- Hetanshu Parmar
- Purvish Parekh
- Tarjani Patel
- Vrushti Patel
- Ansh Rathva

1 Introduction

To meet the project objective, we applied Wiener filters and deconvolution to the images and then applied contrast stretching to the obtained output, to get better visibility. However, as discussed in the previous report, the problem of blurring and low resolution still persists. Hence, in order to improve the quality of the output, we use a method called Stacking alongside filtering and deconvolution. Stacking involves aligning or combining 3 images of the same region with varying wavelengths. It helps in reducing noise and significantly affects the PSR i.e. Peak Signal Ratio. More importantly, it helps model the true PSF of an image which helps model noise better as compared to the Gaussian PSF since it would be subjective. Our focus will now be on stacking as it will help retain the information of the original captured image and in color mapping as well. It is expected to provide a better quality output than the previously applied methods.

2 Methodology

During the fourth week, we focused on further enhancing the output image quality by adding colors. For creating colored images, we referred to the techniques used for the Hubble Space Telescope (HST) and the James Webb Space Telescope (JWST). The fundamental technique they use involves using three or more images of the same regions with different wavelengths and stacking them on top of each other to create a final colored image. For simplicity, we have used a set of three different images, each capturing approximate wavelengths of 430nm, 555nm, and 700nm, representing the colors blue, green, and red, respectively.

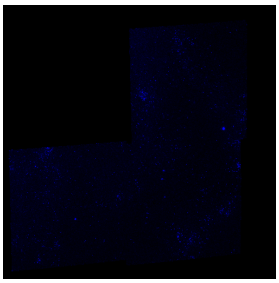

Figure 1: Blue Image

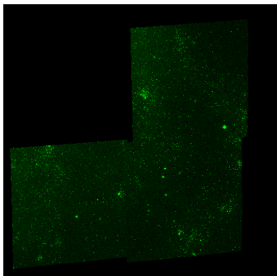

Figure 2: Green Image

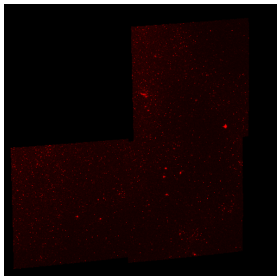

Figure 3: Red Image

Those three images are then processed individually through the previously refined pipeline, which comprises contrast-stretching, a Wiener filter, and Richard-Lucy deconvolution. The three individually processed images are then stacked on top of each other, providing data for respective color channels, resulting in a colorful RGB image. The individual images are shown below.

Below is the code for the image processing pipeline, which includes contrast stretching, Wiener filtering, and Richard-Lucy deconvolution for individual blue, green, and red images, and stacking them to create the final colored image.

2.1 Code:

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from skimage import img_as_float, restoration, exposure,
↪  transform
from numpy.fft import fft2, ifft2
from astropy.io import fits

def make_preview(img, max_dim=2048):
    h, w = img.shape[:2]
    scale = min(1.0, float(max_dim) / max(h, w))
    if scale < 1.0:
        return transform.resize(img, (int(h * scale), int(w *
        ↪  scale)), anti_aliasing=True)
    return img.copy()


# ========== Utility Functions ==========

def read_fits(path):
    with fits.open(path) as hdul:
        data = None
        header = None
```

```
            for hdu in hdul:
                if getattr(hdu, 'data', None) is not None:
                    data = hdu.data
                    header = hdu.header
                    break

            if data is None:
                raise ValueError(f"No image data found in any HDU of
                ↪  {path}")

        if hasattr(data, 'filled'):
            data = data.filled(np.nan)

        data = np.asarray(data)

        if data.dtype == object:
            try:
                data = data.astype(np.float64)
            except Exception:
                flat = []
                for v in data.flat:
                    try:
                        flat.append(float(v))
                    except Exception:
                        flat.append(0.0)
                data = np.array(flat,
                ↪  dtype=np.float64).reshape(data.shape)

        data = np.nan_to_num(data, nan=0.0, posinf=0.0, neginf=0.0)

        data = data.astype(np.float64, copy=False)
        return img_as_float(data)

    def gaussian_psf(size=7, sigma=1.5):
        ax = np.linspace(-(size-1)/2, (size-1)/2, size)
        xx, yy = np.meshgrid(ax, ax)
        psf = np.exp(-(xx**2 + yy**2) / (2 * sigma**2))
        psf /= psf.sum()
        return psf

    def wiener_filter(img, kernel, K=0.0001):
        kernel /= np.sum(kernel)
        dummy = fft2(img)
        kernel_fft = fft2(kernel, s=img.shape)
        wiener = np.conj(kernel_fft) / (np.abs(kernel_fft)**2 + K)
        result = np.abs(ifft2(dummy * wiener))
```

```
    return result

def preprocess_channel(img, psf, K=0.0001, num_iter=30):
    wiener_out = wiener_filter(img, psf, K)
    p2, p98 = np.percentile(wiener_out, (60, 98))
    stretched = exposure.rescale_intensity(wiener_out,
    ↪  in_range=(p2, p98))

    deconv = restoration.richardson_lucy(stretched, psf,
    ↪  num_iter=num_iter, clip=True)
    p2, p98 = np.percentile(deconv, (60, 98))
    final = exposure.rescale_intensity(deconv, in_range=(p2,
    ↪  p98))
    return final

# ========== Load FITS Images (update paths accordingly)
↪  ==========
path_B = "C:/Users/pnpar/Downloads/Image_Stacking_Data/5/j96r230⌋
↪  11_435_drc.fits"  # Blue channel
path_G = "C:/Users/pnpar/Downloads/Image_Stacking_Data/5/j96r230⌋
↪  11_555_drc.fits"  # Green channel
path_R = "C:/Users/pnpar/Downloads/Image_Stacking_Data/5/j96r230⌋
↪  11_658_drc.fits"  # Red channel
img_B = read_fits(path_B)
img_G = read_fits(path_G)
img_R = read_fits(path_R)

# ========== PSF and Processing ==========
psf = gaussian_psf(size=7, sigma=1.5)

B_proc = preprocess_channel(img_B, psf)
G_proc = preprocess_channel(img_G, psf)
R_proc = preprocess_channel(img_R, psf)

# ========== Stack into RGB ==========
rgb = cv2.merge((B_proc, G_proc, R_proc))

# ========== Display ==========
plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.imshow(make_preview(img_R), cmap='gray')
plt.title("Raw HST (F814W - Red)")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(make_preview(img_G), cmap='gray')
```

```
plt.title("Raw HST (F555W - Green)")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(make_preview(img_B), cmap='gray')
plt.title("Raw HST (F435W - Blue)")
plt.axis('off')
plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 5))
plt.subplot(1, 3, 1)
plt.imshow(make_preview(R_proc), cmap='gray')
plt.title("Processed HST (F814W - Red)")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(make_preview(G_proc), cmap='gray')
plt.title("Processed HST (F555W - Green)")
plt.axis('off')

plt.subplot(1, 3, 3)
plt.imshow(make_preview(B_proc), cmap='gray')
plt.title("Raw HST (F435W - Blue)")
plt.axis('off')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
plt.imshow(make_preview(rgb))
plt.title("True-Color HST Composite (Wiener + RL)")
plt.axis('off')
plt.show()

# ========== Save ==========
cv2.imwrite("HST_ColorComposite.png", cv2.cvtColor((rgb *
↪  255).astype(np.uint8), cv2.COLOR_RGB2BGR))
```

3 Results

The shown image is the result generated by processing RGB wavelength intensity data separately and combining the data into the RGB channels of the colored image.

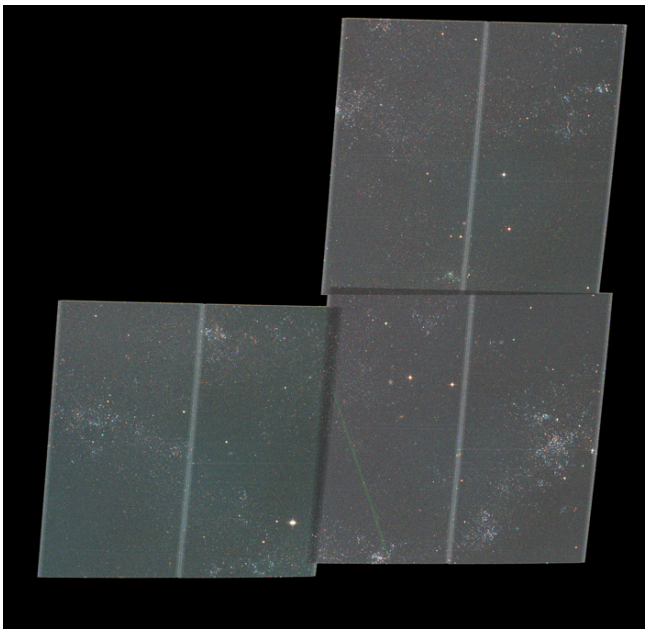

Figure 4: Raw Input Image

4 Problems Faced:

- The problem of partial noise amplification persists
- During the stacking operations, we faced some challenges due to the resolution and shape of the raw .fits file. We had to find a set of three different images that have data for three different wavelengths. However, some images had a different shape and resolution in the .fits file. That resulted in misaligned stacked images. In the given images, the problems encountered during the image alignment process for the stacking operation are illustrated. This error was partially removed by altering `numpy.dstack(B, G, R)` function with `cv2.merge(B, G, R)` function, which better handles the shape and the channel of images, and by removing the errornous

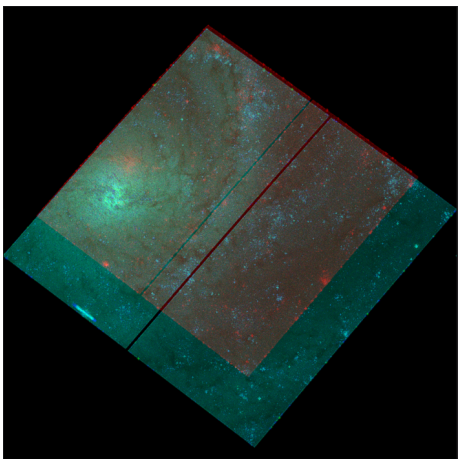

Figure 5: Differing Sizes of Images

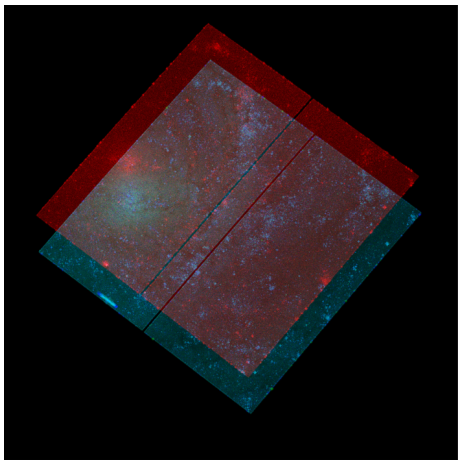

Figure 6: Misalign Images

5 Next Set of Work:

In the upcoming week, we plan to:

- Expand the data set containing information on three different wavelengths
- Increase the robustness of the merging operations to avoid any misalignment and shape errors
- The current pipeline is a computationally expensive operation and takes a significant amount of time; therefore, try to optimize it

6 Conclusion

For this week, we aimed to create fully processed and clean full-colored images using similar image stacking methods employed in HST and JWST. We processed images of RGB filters and put their data into the RGB channel of the output image. Next, we will expand our data set to include multiple sets of RGB wavelength filter images and develop a robust algorithm to handle alignment errors in stacking operations.

References

[1] E. J. Groth, "A pattern recognition program for cosmic ray rejection on CCD images," *The Astronomical Journal*, vol. 118, no. 4, pp. 1764–1776, 2006. [Online]. Available: https://iopscience.iop.org/article/10.1086/510117/pdf)

[2] Mikulski Archive for Space Telescopes (MAST), "MAST Portal." [Online]. Available: https://mast.stsci.edu/portal/Mashup/Clients/Mast/Portal.html

[3] NASA, "How are Webb's full-color images made?," [Online]. Available: https://science.nasa.gov/mission/webb/science-overview/science-explainers/how-are-webbs-full-color-images-made/