Astronomy Lab: Report 2

Kiavash Teymoori 99100585
Alireza Talebi 400109821

Introduction

In this experiment, we captured dark frames using a Canon 1200D DSLR camera with various exposure times. A dark frame is an image taken with the camera sensor exposed to no light, which allows for the measurement of thermal noise, readout noise, and other sensor characteristics. The primary goal of this experiment is to generate a master dark frame by averaging multiple dark frames for each exposure time. This master dark frame can later be used for calibrating astronomical images by subtracting unwanted noise.

Experiment

Capturing Dark Frames

To acquire dark frames, we first ensured that the camera lens cap was securely closed to prevent any light from entering. The ISO setting was fixed at 100 to minimize sensor noise, and we captured 10 dark frames at each of the following exposure times:

- 0.001 s
- 0.01 s
- 0.1 s
- 1 s
- 10 s

Since the camera sensor can generate more noise when heated, we took precautions to prevent the camera from warming up excessively during the image capture process.

The camera stored the images in .CR2 (Canon RAW format), which is not directly compatible with the tools used for analysis. To convert the images into .FIT files, we used Siril software. However, some images were corrupted, and as a result, we only had around six usable frames per exposure time instead of ten.

Analysis

Once the dark frames were collected and converted, we processed them using Python and the Astropy library.

Grayscale conversion

```python
from astropy.io import fits
import numpy as np
import matplotlib.pyplot as plt
import os

# convert RGB FITS array to grayscale
def grayscale(arr):
    if arr.ndim == 3:  # If the array is RGB (3D)
        rgb_index = np.array([0.299, 0.587, 0.114])
        arr = np.array(arr, dtype=np.float64)
        grayscale_array = np.einsum('i,ijk->jk', rgb_index, arr)
        return grayscale_array
    elif arr.ndim == 2:  # If the array is already grayscale (2D)
        return arr
```

This function converts an RGB FITS image into grayscale by applying standard luminosity weights to the red, green, and blue channels. However, since our .FIT files were already in grayscale, this function was not necessary in our case.

Building the Master Dark Frame

```
mean_values = []
median_values = []
std_values_mean = []
std_values_median = []
variance_values_mean = []
variance_values_median =[]
mean_errors = []
median_errors = []

for fits_directory, exposure in zip(fits_directories, exposure_times):
    fits_files = [f for f in os.listdir(fits_directory) if f.endswith('.fit')]
    grayscale_arrays = []

    for file in fits_files:
        fits_path = os.path.join(fits_directory, file)
        array = fits.open(fits_path)[0].data
        array_gray = grayscale(array)
        grayscale_arrays.append(array_gray)

    grayscale_stack = np.stack(grayscale_arrays, axis=0)
    master_dark_average = np.mean(grayscale_stack, axis=0)
    master_dark_median = np.median(grayscale_stack, axis=0)

    master_dark_average_flat = master_dark_average.flatten()
    master_dark_median_flat = master_dark_median.flatten()
```

The script loops through each exposure time and loads each image, applies the grayscale conversion, and stores it in a list. The images are then stacked into a 3D NumPy array, where each layer represents a different exposure frame. we computed the master dark frame in both mean dark frame and median dark frame ways. I believe median dark frame is more suitable but i will report both of them to see the differences.
The resulting master dark frames are flattened into 1D arrays for histogram analysis.

Histogram Analysis

```
    bins = 100
    # zoom in histograms
    range_min, range_max = np.percentile(master_dark_average_flat, [0, 99])

    plt.figure(figsize=(10, 5))
    plt.title(f'Histogram of Master Dark Frame (Average) for {exposure}s')
    plt.xlabel('Intensity')
    plt.ylabel('Number of Data')
    plt.hist(master_dark_average_flat, bins=bins, range=(range_min, range_max), color='blue'
                                                )
    plt.show()

    plt.figure(figsize=(10, 5))
    plt.title(f'Histogram of Master Dark Frame (Median) for {exposure}s')
    plt.xlabel('Intensity')
    plt.ylabel('Number of Data')
    plt.hist(master_dark_median_flat, bins=bins, range=(range_min, range_max), color='
                                                magenta')
    plt.show()
```

```
    mean_values.append(np.mean(master_dark_average_flat))
    median_values.append(np.mean(master_dark_median_flat))
    std_values_mean.append(np.std(master_dark_average_flat))
    std_values_median.append(np.std(master_dark_median_flat))
    variance_values_mean.append(np.var(master_dark_average_flat))
    variance_values_median.append(np.var(master_dark_median_flat))
    mean_errors.append(mean_error)
    median_errors.append(median_error)
```

To improve visualization we zoomed in on the most relevant intensity range using the 1st and 99th percentiles. This allows us to see the overall structure of the dark frames noise distribution.

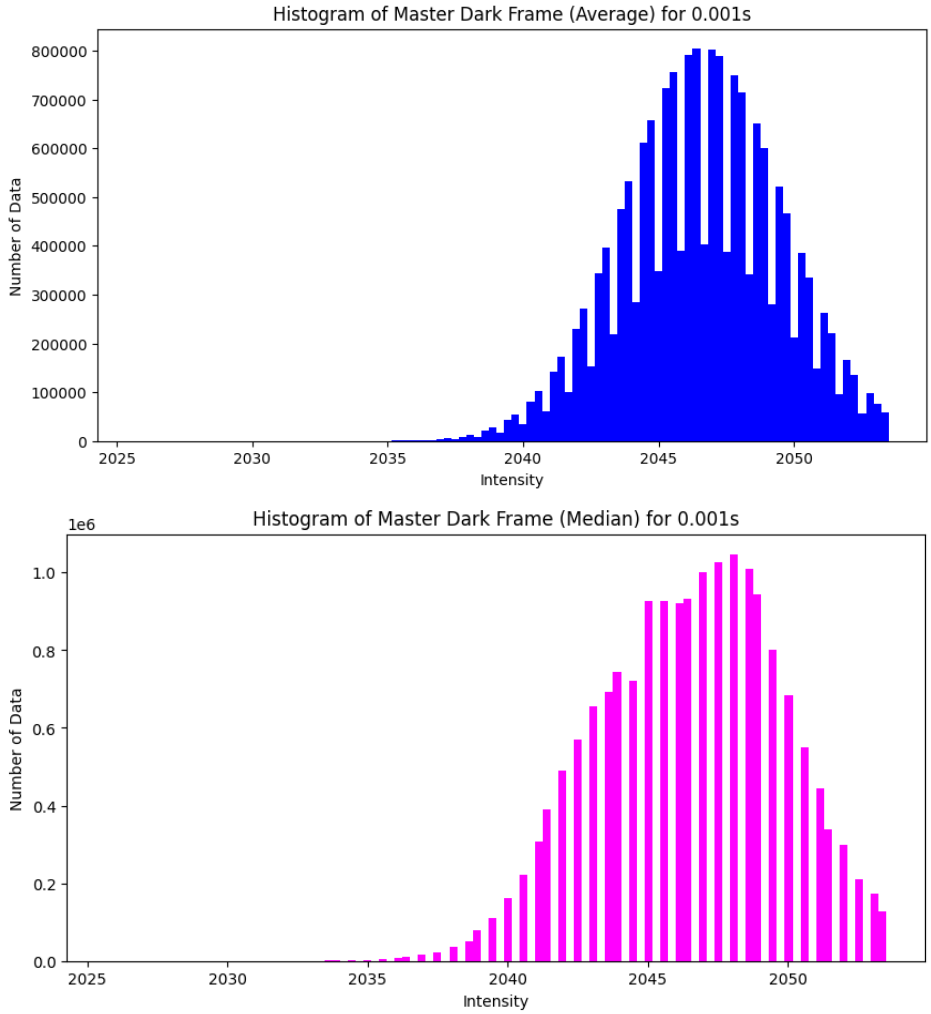

Figure 1: Number of data vs Intensity for both average and median method with exposure time = 1ms

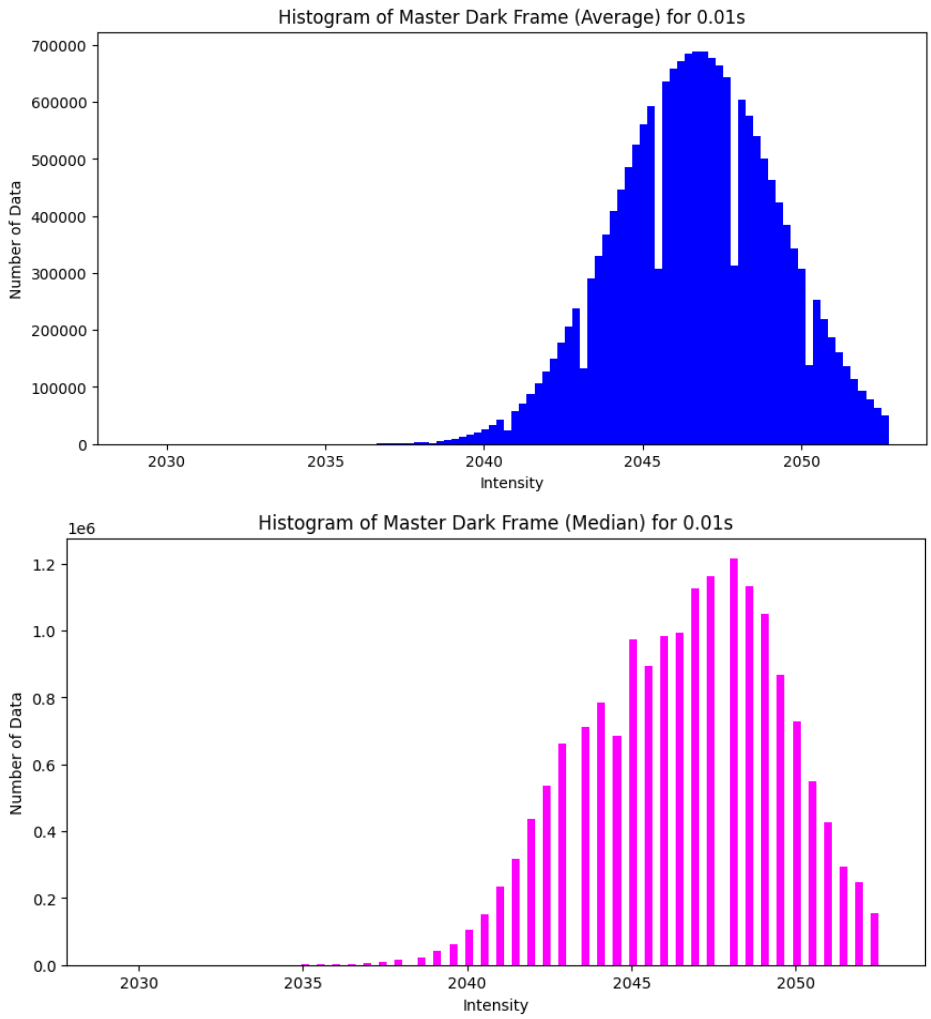

Figure 2: Number of data vs Intensity for both average and median method with exposure time = 10ms

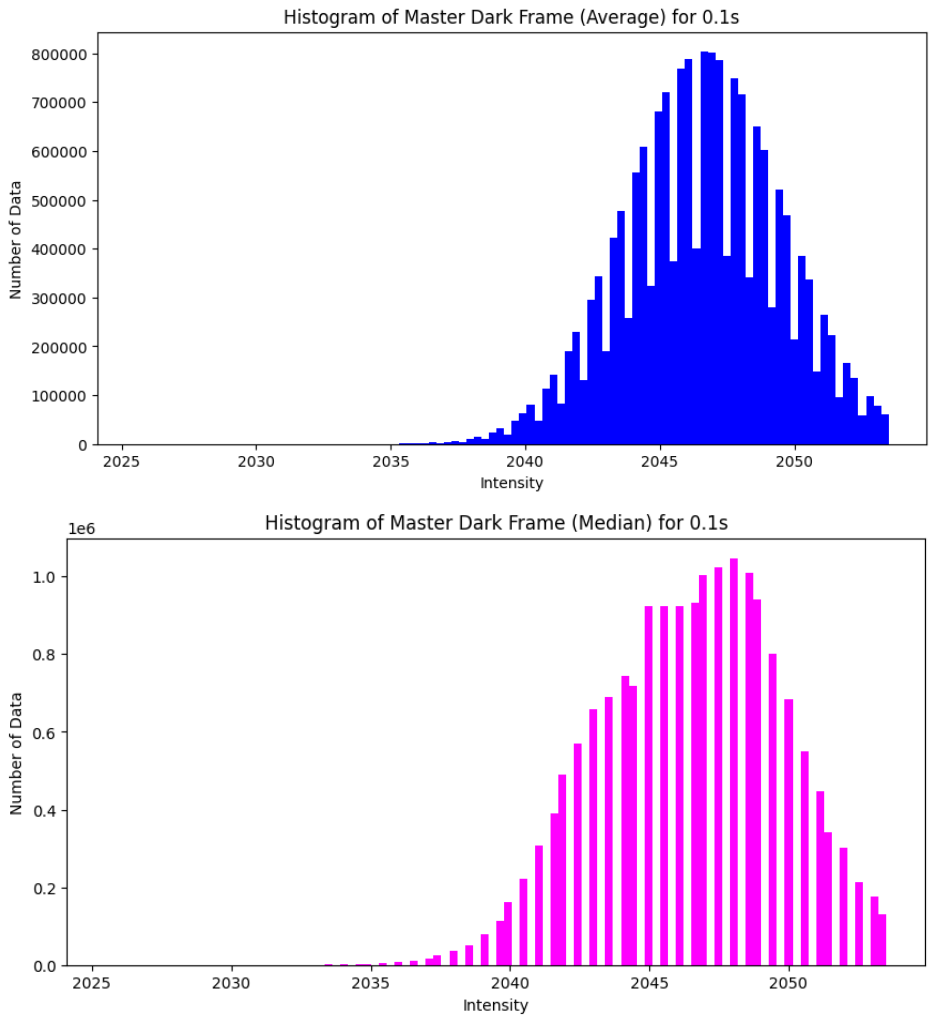

Figure 3: Number of data vs Intensity for both average and median method with exposure time = 100ms

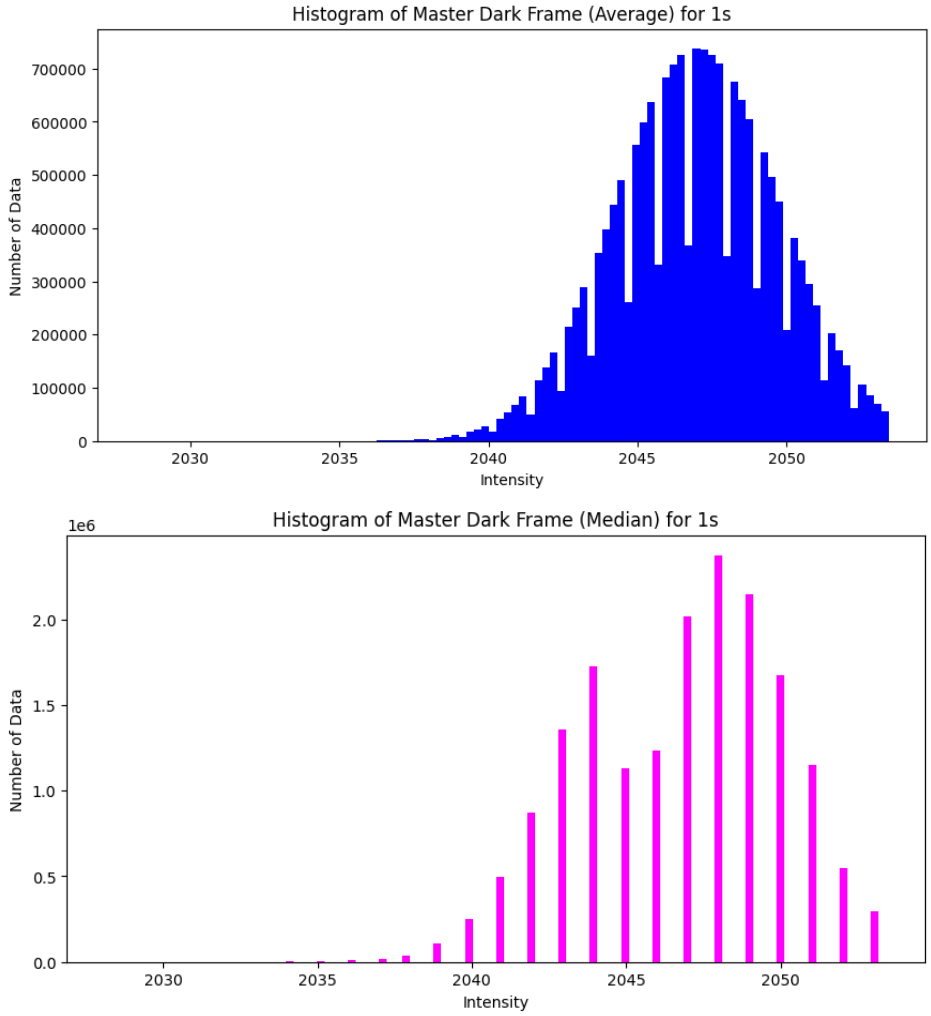

Figure 4: Number of data vs Intensity for both average and median method with exposure time = 1s

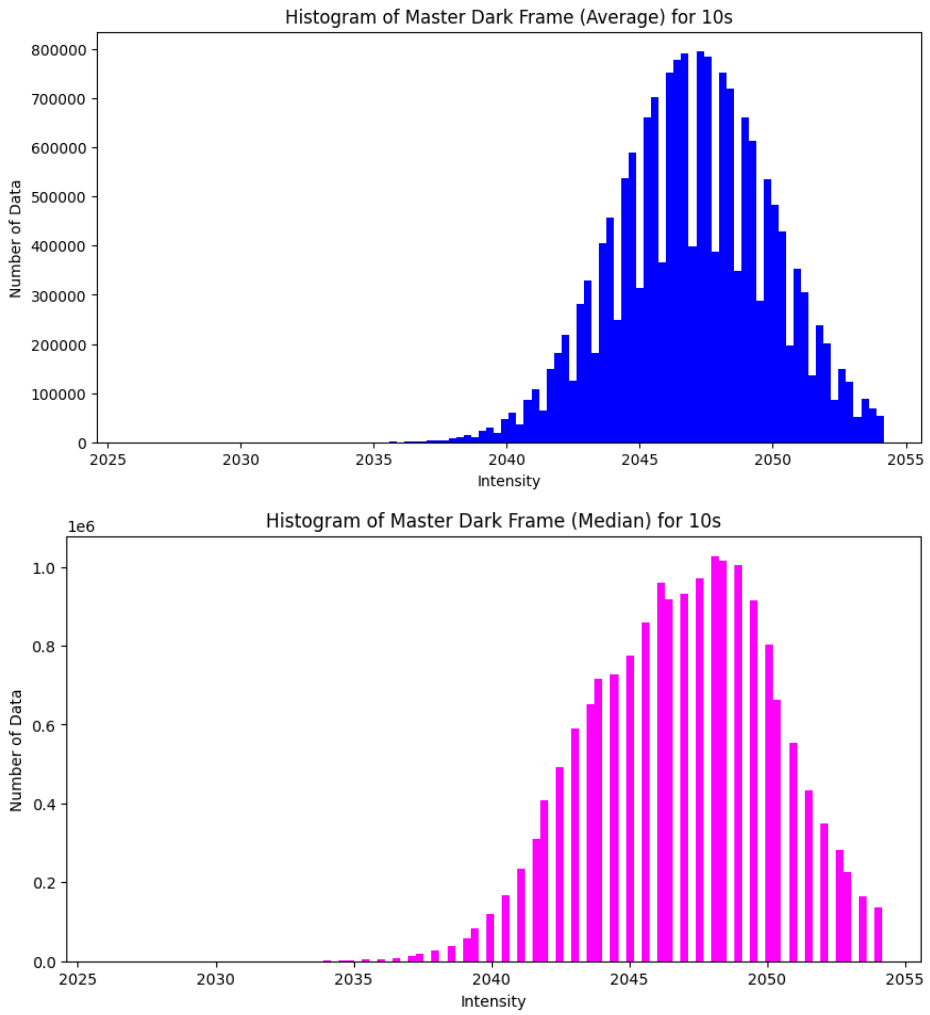

Figure 5: Number of data vs Intensity for both average and median method with exposure time = 10s

Standard Error Analysis

```
# standard error (std / sqrt(N))
N_pixels = master_dark_average_flat.size  # Total number of pixels
mean_error = np.std(master_dark_average_flat) / np.sqrt(N_pixels)
median_error = np.std(master_dark_median_flat) / np.sqrt(N_pixels)
```

For each exposure time, we computed the mean and median pixel intensity values. The uncertainty is given by the standard error, which is defined as the standard deviation divided by the square root of the number of pixels.

Exposure Time (s)	Mean Intensity ± Error	Median Intensity ± Error
0.001	2046.6356 ± 0.0007	2046.6265 ± 0.0008
0.01	2046.7367 ± 0.0006	2046.7326 ± 0.0007
0.1	2046.6386 ± 0.0007	2046.6314 ± 0.0008
1	2047.0219 ± 0.0008	2047.0212 ± 0.0009
10	2047.1070 ± 0.0021	2047.1045 ± 0.0022

Table 1: Mean and Median Intensity Values with Uncertainty for Different Exposure Times

Intensity and Variance Analysis Ideally, dark current (caused by thermal noise) should increase linearly with exposure time.

```
# Mean Intensity vs. Exposure Time
plt.figure(figsize=(10, 5))
plt.plot(exposure_times, mean_values, 'bo-', label='Mean Intensity')
plt.plot(exposure_times, median_values, 'go-', label='Median Intensity')
plt.xlabel('Exposure Time (s)')
plt.ylabel('Intensity')
plt.title('Intensity vs. Exposure Time')
plt.legend()
plt.grid(True)
plt.show()

# Mean Intensity vs. Exposure Time (log)
plt.figure(figsize=(10, 5))
plt.plot(exposure_times, mean_values, 'bo-', label='Mean Intensity')
plt.plot(exposure_times, median_values, 'ro-', label='Median Intensity')
plt.xscale('log')
plt.xlabel('Exposure Time (s)')
plt.ylabel('Intensity')
plt.title('Intensity vs. Exposure Time')
plt.legend()
plt.grid(True)
plt.show()
```

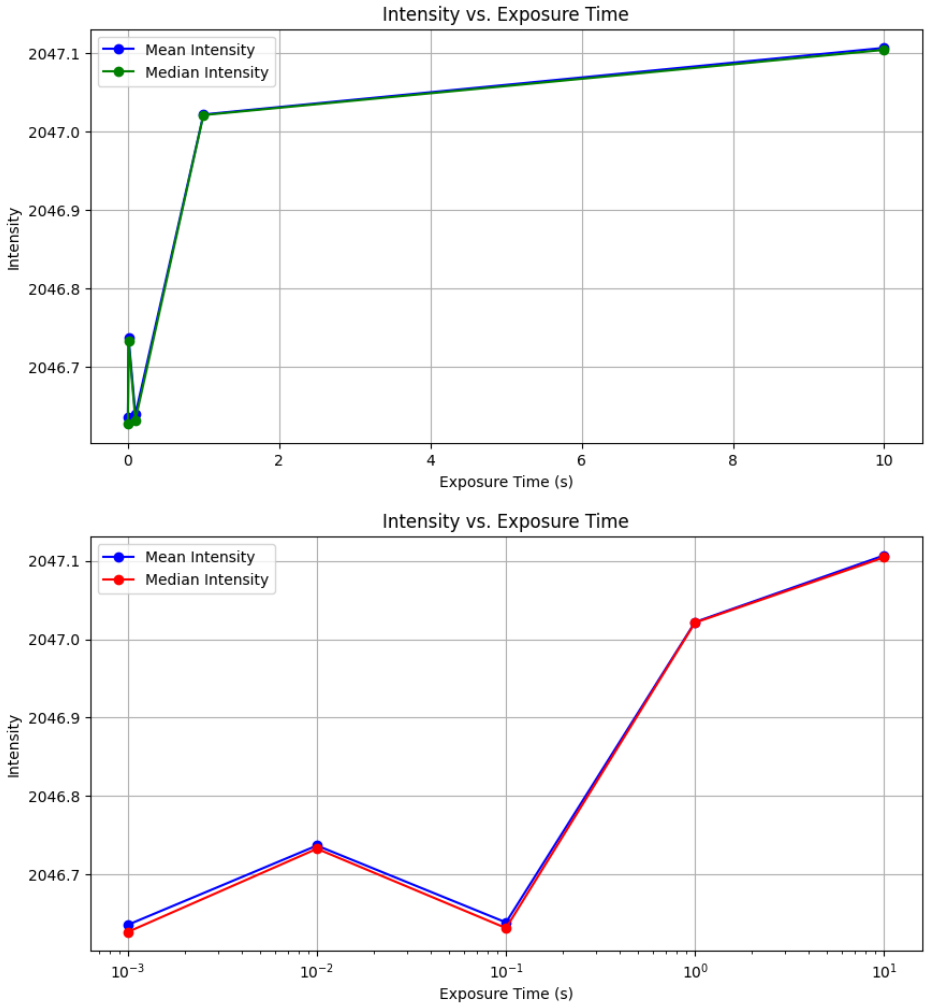

Figure 6: Mean Intensity vs. Exposure Time for both average and median method

The variance of the dark frame should also increase with exposure time.

```
# Variance vs. Exposure Time
plt.figure(figsize=(10, 5))
plt.plot(exposure_times, variance_values_mean, 'bo-', label='Variance (Mean)')
plt.plot(exposure_times, variance_values_median, 'ro-', label='Variance (Median)')
plt.xlabel('Exposure Time (s)')
plt.ylabel('Variance')
plt.title('Variance vs. Exposure Time')
plt.legend()
plt.grid(True)
plt.show()

# Variance vs. Exposure Time (Log-Log)
plt.figure(figsize=(10, 5))
plt.plot(exposure_times, variance_values_mean, 'bo-', label='Variance (Mean)')
plt.plot(exposure_times, variance_values_median, 'ro-', label='Variance (Median)')
plt.xscale('log')
plt.yscale('log')
plt.xlabel('Exposure Time (s)')
plt.ylabel('Variance')
plt.title('Variance vs. Exposure Time (Log-Log)')
```

```
plt.legend()
plt.grid(True)
plt.show()

# standard deviations
for exp, std_mean, std_median in zip(exposure_times, std_values_mean, std_values_median):
    print(f"Exposure Time {exp}s - Standard Deviation (Mean): {std_mean}")
    print(f"Exposure Time {exp}s - Standard Deviation (Median): {std_median}\n")
```

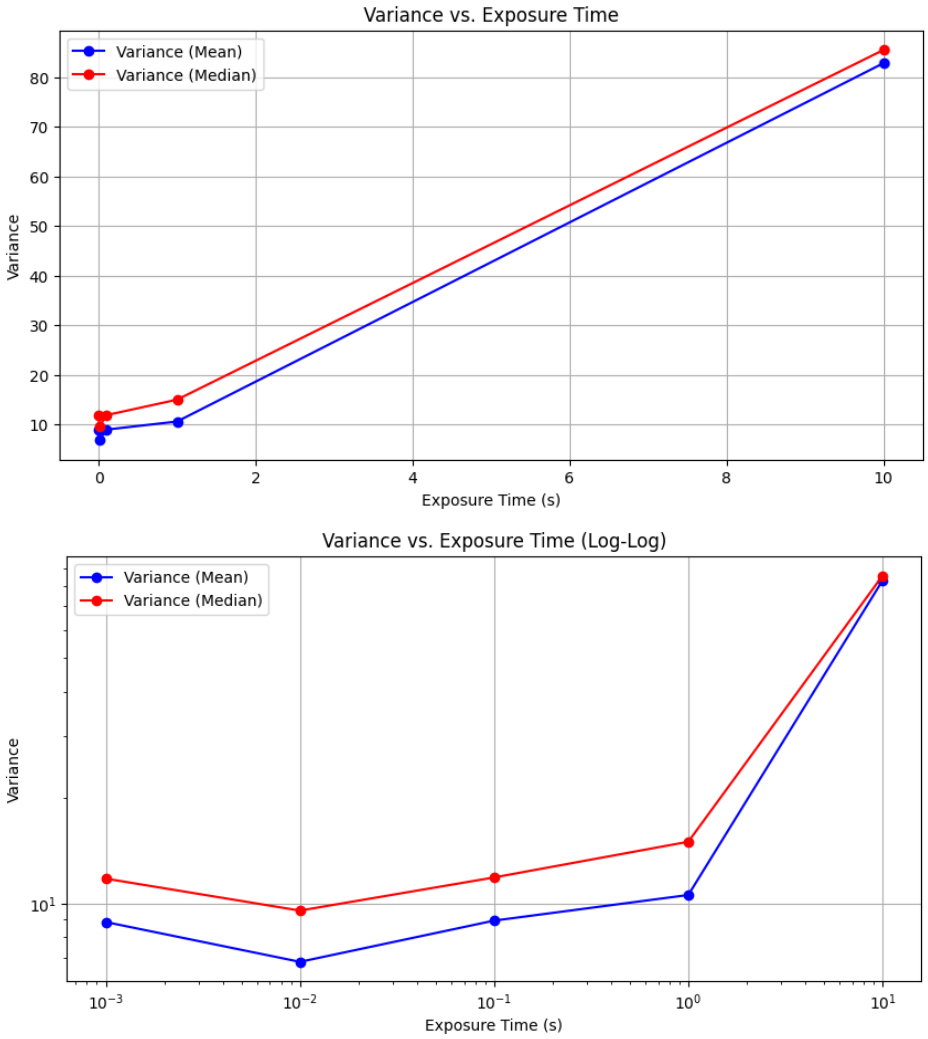

Figure 7: Variance vs. Exposure Time for both average and median method

The presence of readout noise and thermal noise influences the shape of this plot.

Question

Does the Intensity vs. Time Plot Pass Through the Origin?

No, it would not pass through the origin because the dark frame signal includes both thermal noise and readout noise, which are non-zero even at zero exposure time. If only thermal noise were present, the plot

would be linear and pass through the origin.