ZTF24aahgqwk in NGC 3443

Light Curve Notebook

This notebook begins with the 36 stacked images produced by our Calibration Notebook, and produces a light curve, consisting of 18 Sloan r' and 18 Sloan g' data points.

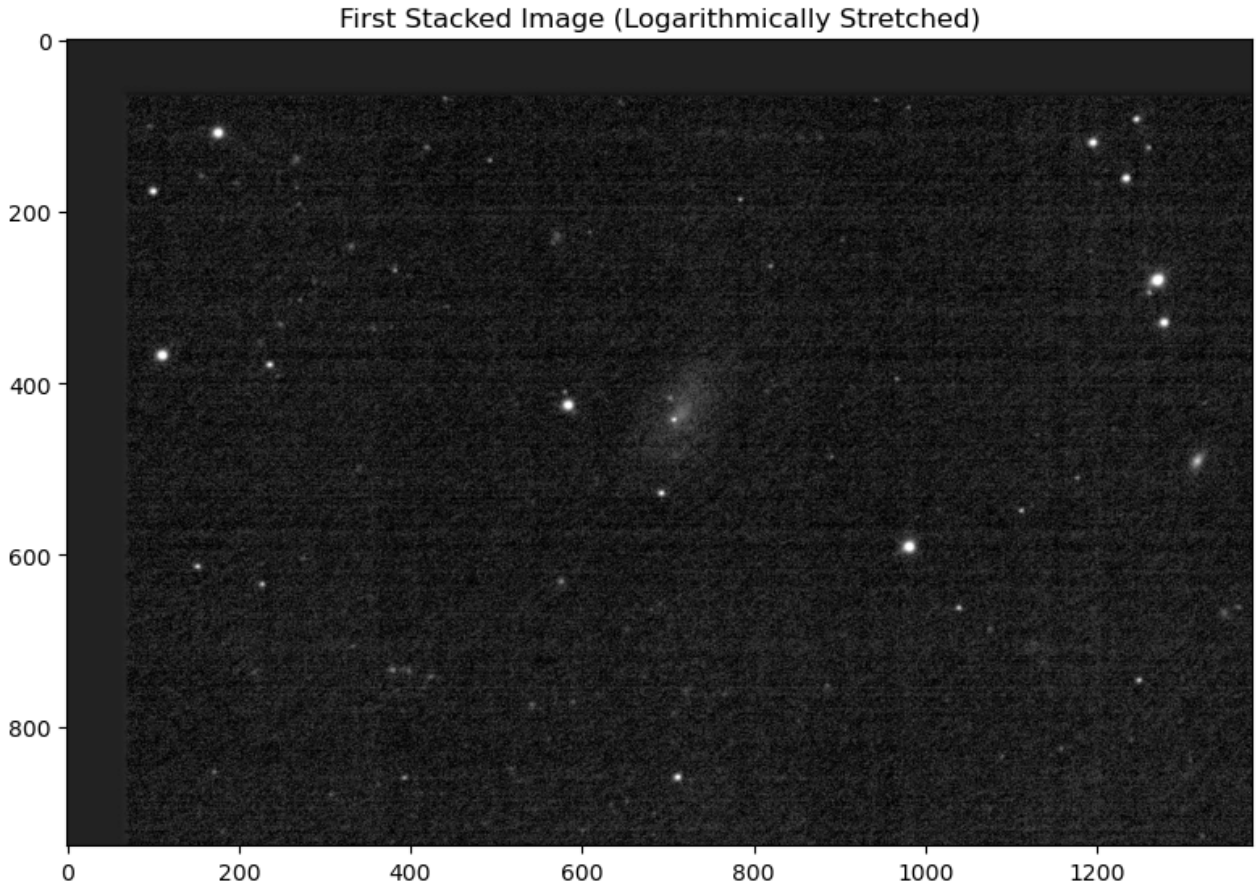
See a least squares notebook for background for the method used below.

In [10]:
```
import os
import numpy as np
from scipy.optimize import least_squares
from astropy import units as u
from astropy.nddata import CCDData
from astropy.io import fits
from ccdproc import ImageFileCollection
import astroalign as aa
import matplotlib.pyplot as plt
%matplotlib inline
from math import log10, floor

# THIS COMMENT IS THE LONGEST A LINE CAN BE AND STILL RENDER COMPLETELY WHEN
# THIS COMMENT IS 72 CHARACTERS WITHOUT COUNTING THE NEWLINE AT THE END.

# This notebook needs to be able to find the stacked images.

home_directory = os.path.expanduser('~')
supernova_project_directory = os.path.join(home_directory, 'Projects', 'supe
stacked_directory = os.path.join(supernova_project_directory, 'analyses', 'Z

# The 36 images are in the stacked directory. There were 18 observation sess

# filters

filters = ['r', 'g']
filter_full_names = ["Sloan r'", "Sloan g'"]

# observation dates (UTC)

observation_dates = [
    '2024-03-20',
    '2024-03-21',
    '2024-03-23',
    '2024-03-27',
    '2024-04-02',
```

```
    '2024-04-03',
    '2024-04-04',
    '2024-04-06',
    '2024-04-10',
    '2024-04-11',
    '2024-04-13',
    '2024-04-17',
    '2024-04-21',
    '2024-04-22',
    '2024-04-23',
    '2024-04-29',
    '2024-04-30',
    '2024-05-02'
]

def file_for_date_and_filter(date, filter):
    return os.path.join(stacked_directory, date + '-' + filter + '_stacked.f

def stacked_image_for_date_and_filter(date, filter):
    file = file_for_date_and_filter(date, filter)
    return CCDData.read(file, unit=u.adu)

# Log stretch utility

def log_stretch_transform(black_point, saturation_range):

    log_saturation_range = log10(saturation_range)

    def fn(pixel_value):
        pixel_value -= black_point
        if pixel_value <= 1.0:
            return 0
        else:
            log_pixel_value = log10(pixel_value)
            if log_pixel_value >= log_saturation_range:
                return 255;
            else:
                return floor(256 * log_pixel_value / log_saturation_range)

    return fn
```

Display a Representative Image

In [21]:
```
first_image = stacked_image_for_date_and_filter('2024-03-20', 'r')

# Log stretch

stretch_function = log_stretch_transform(26, 50)
stretch_transform = np.vectorize(stretch_function)

stretched_image = stretch_transform(first_image.data)

# Display the image

fig, axes = plt.subplots(1, 1, figsize=(8, 8))

axes.imshow(stretched_image, cmap='gray')
axes.set_title("First Stacked Image (Logarithmically Stretched)")

plt.tight_layout()
plt.show()
```



In [22]:
```
subframe = stretched_image[380:579, 600:799]

# Display the representative subtracted dark

fig, axes = plt.subplots(1, 1, figsize=(8, 8))

axes.imshow(subframe, cmap='gray')
axes.set_title("200x200 Subframe of First Stacked Image (Logarithmically Str

plt.tight_layout()
plt.show()
```