Lab Report 4: Astrometry

Brianna Liz Binoy
Group Z
February 10, 2024

ABSTRACT

Astrometry is a branch of astronomy that deals with the measurement of the positions and motions of celestial objects in the sky. In this lab, we used CCD Astrometry to calculate the proper motion of the 26 Prosperina asteroid, located in our Solar System's asteroid belt. Using the USNO B1.0 Catalog, we determined the standard coordinates of the stars in the sky images taken. We utilized both standard and pixel coordinates to derive a transformation matrix, enabling the conversion of standard coordinates to pixel coordinates. This transformation matrix was used to extract the magnification, shear, and rotaional offset of the images, to determine the rough pixel coordinates of the asteroid for each observation day. Using centroiding, accurate pixel coordinates were determined and were converted into Right Ascension (RA) and Declination (DEC) for each day. Python's inbuilt curve-fit function was then used to calculate the asteroid's velocity. The calculated velocity was estimated to be 60 ± 20 arcseconds per hour.

1. Introduction

Asteroid astrometry is a branch of astronomy dealing with the precise measurement of the location and motions of asteroids in the sky. By tracking the apparent movements of asteroids against the reference stars, astronomers can determine their orbits, velocities, and other characteristics. Astrometry is used in cataloging asteroids, predicting their future positions, and giving insights into potential threats to Earth from impacts. The underlying motivation of this paper is to conduct astrometric observations aimed at measuring the positions of stars relative to their celestial coordinate system and to analyze the proper motion of the asteroid 26 Prosperina.

Asteroids are small, rock-like objects that orbit our Sun. They are the remnants of our solar system formation and are mostly found near the asteroid belt; which is the region between Mars and Jupiter. Studying asteroids helps scientists understand the history of the solar system. Asteroids are also potential for future space exploration missions.

The USNO Catalog (United States Navy Observatory Catalog) is a catalog of celestial objects compiled by the United States Naval Observatory (USNO). It contains positions, proper motions, and magnitudes for over 1 billion stars, galaxies, and other celestial objects. The USNO B1.0 catalog is widely used by astronomers and researchers for a variety of purposes, including astrometry, photometry, and celestial navigation. In this experiment, the USNO B1.0 Catalog was used to identify the RA and DEC of the stars in the images and to compare them with the pixel coordinates.

When an image of stars centered at (α_0, δ_0) is taken with CCD detectors, the celestial coordinates of the stars (α, δ) are projected onto the tangent plane of a CCD. These projected coordinates are found using the following 2 equations:

$$X = -\frac{\cos\delta \sin(\alpha - \alpha_0)}{\cos\delta_0 \cos\delta \cos(\alpha - \alpha_0) + \sin\delta \sin\delta_0} \tag{1}$$

$$Y = -\frac{\sin\delta_0 \cos\delta \cos(\alpha - \alpha_0) - \cos\delta_0 \sin\delta}{\cos\delta_0 \cos\delta \cos(\alpha - \alpha_0) + \sin\delta \sin\delta_0} \tag{2}$$

where X is the standard x coordinate and Y is the standard y coordinate.

To convert the standard (X,Y) coordinates to pixel coordinates, the following equations are used for an ideal CCD camera:

$$x = \left(\frac{f}{p}\right) X\cos(\theta) - \left(\frac{f}{p}\right) Y\sin(\theta) + x_0 \tag{3}$$

$$y = \left(\frac{f}{p}\right) X\sin(\theta) + \left(\frac{f}{p}\right) Y\cos(\theta) + y_0 \tag{4}$$

where f is the focal length of the camera and p is the pixel size. In this experiment, the coordinates of the USNO B1.0 Catalogs were converted to pixel coordinates to measure the celestial coordinates of the stars in the CCD images.

These equations can be written using a transformation matrix as shown below:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} (f/p)a_{11} & (f/p)a_{12} & x_0 \\ (f/p)a_{21} & (f/p)a_{22} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{5}$$

where the a_ij terms are called the plate constants, referring to the scale magnification, shear and orientation of the image and the x_0 and y_0 are offsets in the pixels.

In this lab, this matrix is used to convert the standard coordinates of the asteroid to pixel coordinates.

The paper commences with an overview of the experimental setup and methods employed to eliminate noise in the images originating from dark currents and systematic variations in pixel sensitivity across the CCD image sensors. Subsequently, it provides an outline of the methods utilized to determine the velocity of asteroids. Section 5 will then present the plots utilized in obtaining the asteroid's velocity. Lastly, the discussion aims to elaborate on the data inspection process, concluding with some final remarks.

2. Data and Observation

3 CCD image files of the Earth sky containing 26 Proserpina were taken for 4 nights using the 50-cm robotic telescope from Dunlap Institute of Telescope (DIT). It has a focal length of 3454 mm. The telescope was equipped with a TE-cooled Kodak KAF-16803 CCD detector array with 4096 x 4096 pixels and provided a field view of approximately 0.53 arcseconds per pixel. This telescope is situated on Mt. Joy in the Saramental Mountains of New Mexico, USA. The USNO- B1.0 Catalog was used to compare the CCD pixel coordinates of the stars in the images to the known positions of the stars. The data are from scans of photographic plates from the Palomar 48-inch Schmidt telescope taken for various sky surveys during the

last century. The USNO-Bl.0 catalog provides all-sky coverage down about V = 21 mag and astrometric accuracy of 0.2 arcseconds (in J2000 equinox).

3. Data Reduction

To avoid readout noise, all the CCD pixels were binned; each 2 × 2 group of pixels in the original array was represented in the FITS file by a single value equal to the average of these four. This also reduced the image size but did not affect the overall image clarity.

To remove noise from the 3 images of the first day, dark, bias, and flat frames were taken on the same day. The dark frames were taken with the same exposure time as the light frames (original asteroid images) of 60 seconds to accurately represent the noise characteristics of the sensors during the time the light frames were captured. After that, the medians for the dark and bias frames were taken to produce master dark and master bias frames. The master dark frame was then subtracted from each asteroid image. The master bias frame was subtracted from each flat to remove any electronic defects before the flats were individually normalized, averaged by taking the median, and finally re-normalized so that the median of the final product was unity.

After obtaining all frames, it was observed that the sky images were inverted. This can be seen in Figure 11b. The high-intensity spot in the top left corner of the master dark frame is also seen in the raw image, but in the bottom right corner, indicating that the raw image was flipped. After flipping the raw sky images, they were cleaned by subtracting the master dark and normalizing by the master flat; normalizing by the master flat frame corrected for the non-uniform CCD pixel sensitivity problem and the rings that appeared on the images due to the dust grains on the filter. The code for this calculation is under Appendix 6.12. This process was repeated for the sky images captured on the subsequent three observation days. Separate median frames were calculated for each observation day to accommodate any additional noise specific to that observation day, thereby ensuring an accurate representation of noise characteristics and tailored noise reduction. Appendix 6.10 displays example raw and corrected images.

After cleaning the image files, it was noticed that almost all image files still contained a bright line on their right edge. This may be because the CCD pixels on that edge are damaged and cannot transfer the electrons when the image is read out. To remove this, we simply cropped the images, and the final cleaned images were saved in a file.

4. Methodology

4.1. Source Detection

After cleaning all 12 image files, the centroids of all the stars in each image were calculated.

A Python function was created to set a threshold intensity to only identify positions on a sky image where the intensities are above value. The threshold intensity was chosen based on a plot like in Figure **??**. All points on the image above that intensity were identified and marked; this indicated a star or an asteroid. An example of this image is under Appendix. The code for this is under 6.12.

After that, the centroids of the bright points were calculated using the Python code under Appendix. To ensure our centroiding algorithm was accurate, a function was created to randomly select 5 stars and

they were found. The equation used for finding the centroids of each star is as follows:

$$< x >= \frac{\Sigma_i x_i I_i}{\Sigma_i I_i}$$
$$< y >= \frac{\Sigma_i y_i I_i}{\Sigma_i I_i}$$

where x_i and y_i are the x and y pixel values and I_i is the intensities.

Any overly saturated stars were masked out of the images as they could have led to inaccuracies in the centroid calculations; this is due to the distortion caused by saturation artifacts. Masking out these stars led to more accurate centroid measurements.

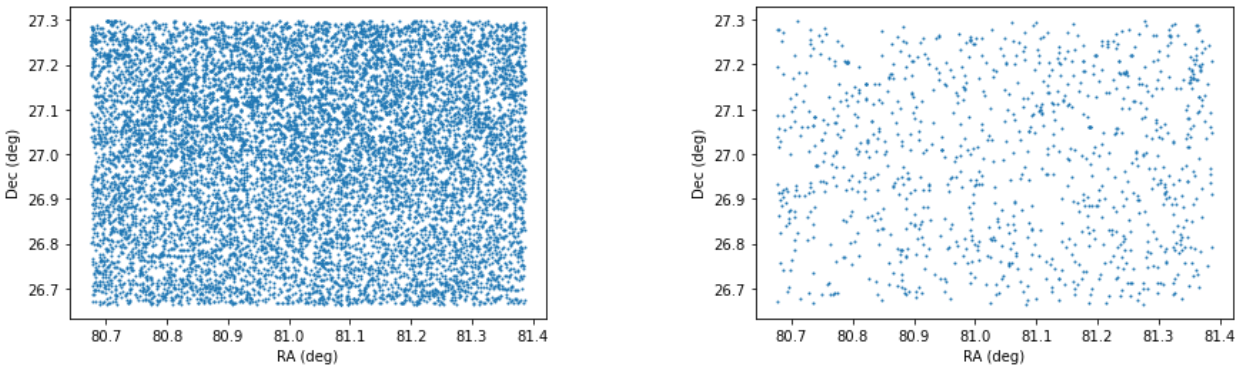
After the centroids were calculated, they were saved as a Pandas data frame in Python. Any stars that were not aligned (with a distance threshold of 6 pixels) were removed from the list of centroids.

This was then repeated for each observation day.

4.2. Finding the Celestial Coordinates of our Stars

To figure out the Right Ascension and Declination of each of our saved centroids/stars in the images, their positions were compared to the USNO B.1 Catalog stars.

After extracting the USNO star coordinates with the same center of field as the sky image, and selecting a region of width 36 arcminutes, stars of magnitude brightness higher than 17 were filtered out as shown in Figure 2. The RA and DEC values were converted to degrees.



(a) USNO Stars

(b) USNO Stars of magnitude less than 17

Fig. 2.—: USNO star coordinates centered at RA 05h24m07.53s and DEC +26°58'50.5". There are 1021 stars with a magnitude less than 17.

The USNO coordinates were then converted to standard planar coordinates using the Equation 1 and Equation 2. Then, equations 4 and 4 were used to convert the standard planar coordinates to pixel coordinates with image shape 2048 x 2048. The focal length used was 3454mm and the pixel size used was

0.009*2mm, accounting for the binning. The rotation offset was set to -89°and this was empirically determined. The Python function that was written for this transformation is under Appendix. An example of one of the cross-matched images is shown in Figure 3.

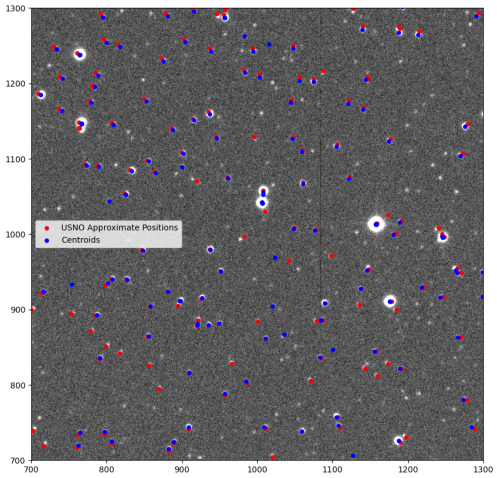


Fig. 3.—: Cross-matching for a part of one of the images from 19/01/2012.

As we can see from Figure 3, although USNO stars and our CCD centroids have the same patterns, there is a small offset. Inspection of the stars also reveals a magnification and that the two patterns are rotated concerning each other. The residuals of this image is plotted in Figure 4.

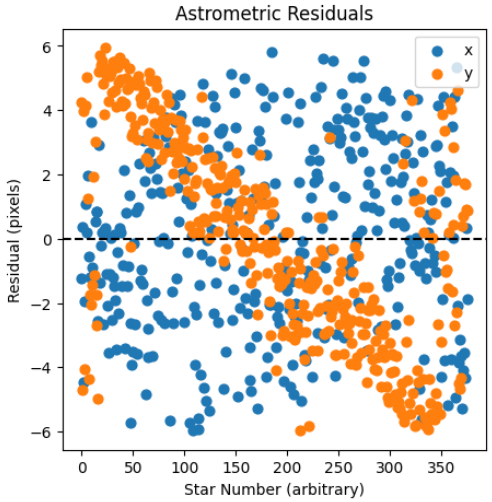


Fig. 4.—: Astrometric Residuals of a part of one of the images from 19/01/2012 for the cross-matching function: It is evident that the further we move away from the center, the larger the residuals are becoming. This is because the function works less effectively for stars further from the center.

The residual plot shows that the function gets less accurate for stars further away from the center. This

is also obvious, as we can see in Figure 4 that the the alignment of the blue and red dots is less accurate nearer to the edges than the center. This may also be because the centroiding equation for the stars calculated in Section 4.1 is not very accurate.

4.3. Plate-Solving

To solve for the magnification and offset errors, we determined the plate constants using least squares fit.

By obtaining the pixel centroids of our ccd images and the approximate USNO standard coordinates of those stars, a function to create the transformation matrix to transform pixel coordinates to standard coordinates was run. This function is under Appendix. Since the Transformation matrix is of the 5, each element containing the factor f/p in the matrix was divided out and the plat constants were found. The final mean plate constants for each of the observation days are printed in 6.4.

After calculating the chi-squared value for each of our plate constant matrices using the equation :

$$\chi^2 = (a - Bc)^T (a - Bc) \tag{6}$$

where a is pixel values, B is the matrix and c is the plate constants.

The final chi-squared value was found to be 0.02 suggesting a very small discrepancy between the USNO coordinates and CCD coordinates. Our plate constant values for each of the days are as follows

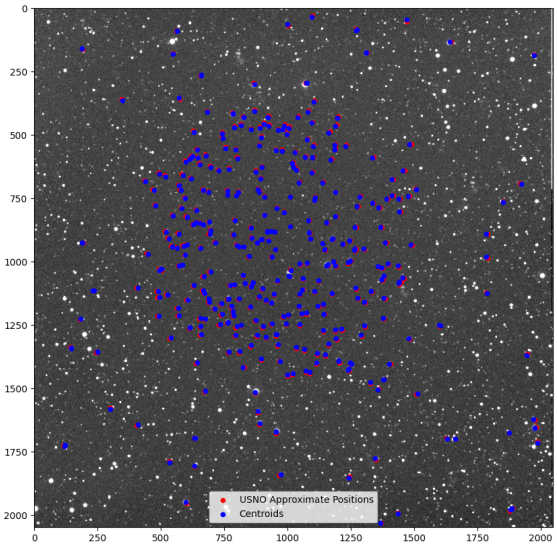
The uncertainties of these values were calculated by using the formula:

$$\delta x = \frac{\sigma}{\sqrt{n}} \tag{7}$$

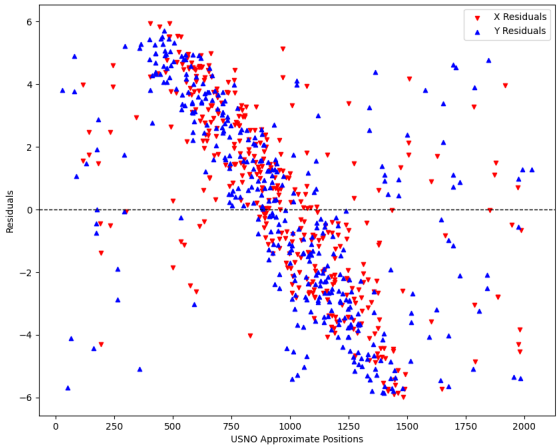
and the standard deviation was found using the equation:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}} \tag{8}$$

The before and after plots of an image from January 19th, 2012 using plate solving and their residuals are below:

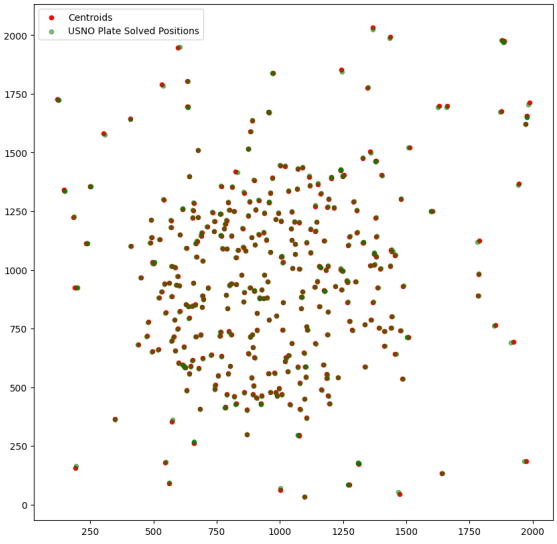


(a) Plot of Stars

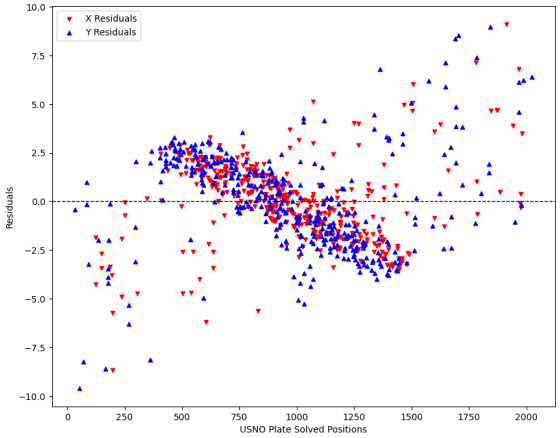


(b) Residuals of the pixels

Fig. 5.—: Image of the sky from January 19, 2012 before plate solving and the residual plot beside it. There is a huge variation for the values away from the centre pixel.



(a) Plot of Stars



(b) Residuals of the pixels

Fig. 6.—: Image of the sky from January 19, 2012 after plate solving and the residual plot beside it. There is much less variation for the values away from the centre pixel.

After plate-solving, the magnitudes of the residuals of the x and y pixels decreased for all days, indicating the impact of plate-solving on converting standard coordinates to pixel coordinates. However, the code still does not work properly for stars further away from the center. This may be because of the centroiding code for that was written earlier, which did not account for the stars near the edges of the images properly.

4.4. Asteroid Positions

Using the rough estimates of the asteroid RA and DEC coordinates by analyzing the images using ALADIN for each of the 4 observation days, the coordinates were converted to standard coordinates using Equation 1 and Equation 2 and the function in 6.7. The pixel coordinates were then determined using the plate constants obtained from the previous code (6.4) for each of the observation days . The values obtained were then run in the Python functions to see where the asteroids were in the images, marked, and then their centroids were calculated for each day using code under Appendix. The centroids were then converted back to RA and DEC coordinates to get the accurate positions of the asteroid on all 4 days. The plot of the Asteroid positions is below.

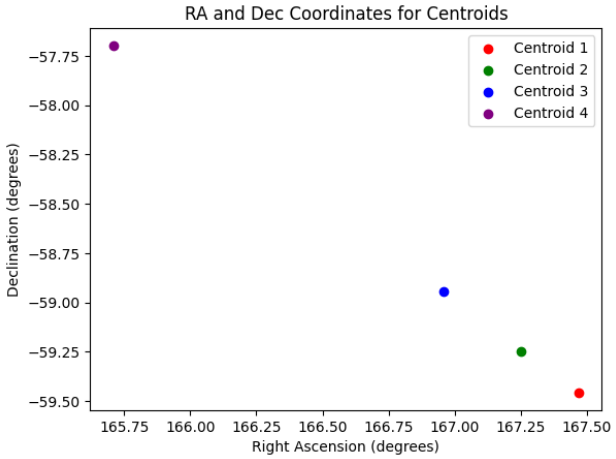
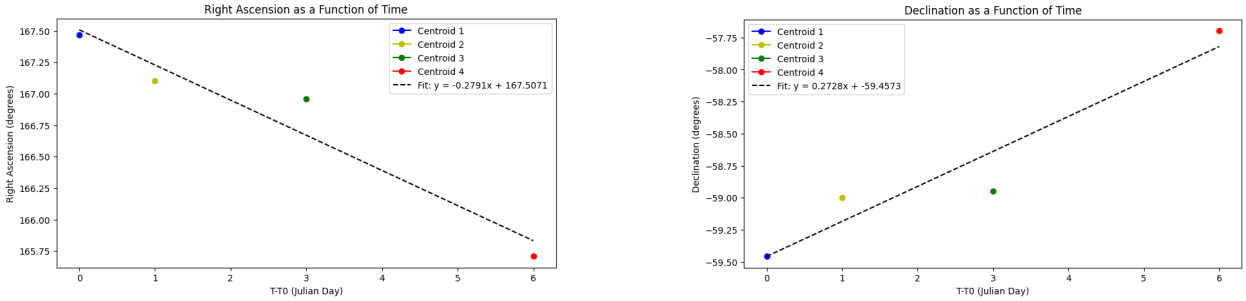


Fig. 7.—: The right ascension and declination of the asteroid on all four observation days

5. Data Analysis

After calculating the positions of the asteroid, its velocity was obtained. This was done by plotting right ascension and declination as a function of time, and curve fitting the data to get the gradients, as shown in Figure 8.



(a) Plot of RA as a function of Time

(b) Plot of DEC as a function of time

Fig. 8.—: Plots of Right Ascension and Declination of the asteroid on all 4 days as a function of time in days.

The values of the gradients and intercepts were found from the outputs of the polyfit function, as can be seen in .

The velocity of the asteroid in arcseconds per hour was computed to be 60 ± 20 arcseconds/hour. The computations for these values are under Appendix 6.8 and Appendix 6.9.

6. Discussion & Conclusion

Nasa's Open Data Portal records the value of 26 Prosperina to be 600 arcseconds/day. Upon comparison to our value, this value is 2.3 times smaller. Many inconsistencies in our calculations were overlooked which have caused us to have a much smaller value.

Despite having removed the dark and bias noises and dividing all the images by the flat-field frames, noise was still present in the data. This was evident because many of the stars in the images appeared non-circular, which is not characteristic of stars.

6.1. Centroiding

One way to enhance our code is by refining the centroiding calculations. Currently, the centroiding code doesn't account for background noise and struggles with fewer circular objects, resulting in offsets in many star centroids. Moreover, when stars are closely positioned, our centroiding code erroneously treats them as a single entity. Additionally, the code fails to accurately calculate the centroids of overly saturated stars, leading to inaccuracies due to saturation artifacts.

A potential improvement is to integrate a Gaussian Smoothing Filter. This filter can effectively reduce noise in the images while preserving brighter spots, thus yielding more precise centroids. To prevent stars from being incorrectly considered as a single entity, we could adjust the width of the boxes in the centroiding function.

Furthermore, stars that are oversaturated could be masked out of the image during centroid calculations. These adjustments would significantly enhance the accuracy and reliability of the centroiding process,

ensuring that the calculated positions better reflect the true locations of the stars in the images.

6.2. Plate Constants

Regarding plate constants, discrepancies in velocity compared to values from the Data Portal may also stem from the transformation matrix obtained from centroids and approximate USNO coordinates. Although the chi-squared value was relatively small, residual plots for the images after plate solving suggest that the code's accuracy diminishes for stars farther from the field's center. Moreover, the code was initially written concerning an image file from January 19th, 2012. As the center of the field varies across observation days, the code becomes less accurate for dates further from January 19th. Notably, the code exhibited the least accuracy on January 25, 2012, with the transformation matrix elements showing the greatest disparity for this date.

Another aspect to consider is that the code was designed for a single image and merely rerun for other images. Improving this process involves implementing a for-loop function for each day's images, allowing for the averaging of values such as centroids and plate constants for each day. This approach would enhance the accuracy of the final velocity estimation for the asteroid.

Lastly, there could have been some other effects caused by optical distortion or anamorphic magnification of the telescope and CCD camera system that was not accounted for while conducting plate solving.

6.3. Conclusion

The main focus of this paper was to find the final velocity of the 26 Prosperina asteroid using CCD Astrometry. 12 images were taken between 19th January 2012 to 25th January 2012. A centroiding code was created to calculate the centroids of the stars in each image. Although the centroiding code did not give very accurate centroids of the stars in the images, the plate-solving matrix came out with a very small chi-squared value. However, the plate-solving matrix was less accurate for stars further away from the center, reflecting the error in the centroiding function. Using rough estimates of the asteroid positions on all four observation days, the centroids were then calculated using the plate-solving matrix we obtained, and the proper motion of the asteroid was found.

Appendix

6.4. Final Plate Constants

$$\text{plates\_1} = \begin{bmatrix} -190833.55902242 & 3587.43183624 & 1015.13348155 \\ 3449.05766971 & 190976.50356884 & 1042.63742992 \end{bmatrix}$$

$$\text{plates\_2} = \begin{bmatrix} -191787.87568368 & 3284.60318849 & 1014.46952208 \\ 3017.64964738 & 191504.34382733 & 1042.1546029 \end{bmatrix}$$

$$\text{plates\_3} = \begin{bmatrix} -191870.21546091 & 3394.72916352 & 1015.04458741 \\ 3321.39673498 & 191600.91246643 & 1042.5020863 \end{bmatrix}$$

$$\text{plates\_4} = \begin{bmatrix} -191738.41841866 & 3309.87661568 & 1015.57662374 \\ 3548.63018819 & 191797.56956724 & 1042.98139081 \end{bmatrix}$$

6.5. Python code to finding the T matrix

```
def get_XY(ra, dec, alpha0, delta0):
    ra_rad = np.radians(ra)
    dec_rad = np.radians(dec)
    alpha0_rad = np.radians(alpha0)
    delta0_rad = np.radians(delta0)


    X = -(np.cos(dec_rad) * np.sin(ra_rad - alpha0_rad)) / (
            (np.cos(delta0_rad) * np.cos(dec_rad) * np.cos(ra_rad - alpha0_rad)) + np.sin(delta0_rad) *

    Y = -((np.sin(delta0_rad) * np.cos(dec_rad) * np.cos(ra_rad - alpha0_rad)) - np.cos(delta0_rad) * np
            (np.cos(delta0_rad) * np.cos(dec_rad) * np.cos(ra_rad - alpha0_rad)) + np.sin(delta0_rad) *
    return X,Y

def get_T(ccd_x, ccd_y, usno_X, usno_Y): #ccd_x and ccd_y are pixel coordinates of our stars and usno_X
    # a_11, a_12, a_21, a_22, x_0, y_0 = params
    # X_transformed = a_11 * usno_X + a_12 * usno_Y + x_0
    # Y_transformed = a_21 * usno_X + a_22 * usno_Y + y_0

    #usno_x = B*c where c looks like (a_11, a_12, x_0)
    p = 0.009*2 #Remember, the pixels are binned in 2x2 squares
    f = 3454
    ratio = f/p
    B = []
    for i in range(len(usno_X)):
      l = [ratio*usno_X[i]]
      B.append(l)
    B = np.array(B)
    #c = (#transpose of B* B)^-1*(transpose of B)*a
    transpose_B = B.transpose()
    result = np.dot(transpose_B,B)
    inverse_result = np.linalg.inv(result)

def get_T(ccd_x, ccd_y, usno_X, usno_Y):
  # for i in range(len(usno_X)):
```

```
  a = np.array([usno_X[0], usno_Y[0]])
  b = np.array([np.array(ccd_x)[0], np.array(ccd_y)[0]])
  # print(a.shape)
  # print(usno_X)

  T = np.linalg.lstsq(a.T,b.T)
  usno_x = np.dot(T, usno_X[0])
  usno_y = np.dot(T, usno_Y[0])
  # print(T)
  # a = [ratio*usno_X[1], ratio*usno_Y[1], 1]
  # b = [ratio*ccd_x[1], ratio*ccd_y[1]]
  # T = np.linalg.lstsq(a,b)
  # print(T)
    # print(b)
  return T, usno_x, usno_y
```

6.6. centroiding

```
def centroid(image):
    """
    Calculate the centroid of an image.

    Parameters:
    - image: 2D numpy array representing the image.

    Returns:
    - x_center, y_center: Coordinates of the centroid.
    """
    # Generate coordinate grids
    y, x = np.indices(image.shape)

    # Calculate the total intensity
    total_intensity = np.sum(image)

    # Calculate the x and y centroid coordinates
    x_center = np.sum(x * image) / total_intensity
    y_center = np.sum(y * image) / total_intensity

    return x_center, y_center
```

6.7. Asteroid code

:

```
# Plotting right ascension as a function of time
plt.figure(figsize=(10, 5))
plt.plot(0, centroid1_ra_deg, label='Centroid 1', marker='o', linestyle='-')
plt.plot(-3, centroid2_ra_deg, label='Centroid 2', marker='o', linestyle='-')
plt.plot(3, centroid3_ra_deg, label='Centroid 3', marker='o', linestyle='-')
plt.plot(6, centroid4_ra_deg, label='Centroid 4', marker='o', linestyle='-')

# Adding labels and title
plt.xlabel('T-T0 (Julian Day)')
plt.ylabel('Right Ascension (degrees)')
plt.title('Right Ascension as a Function of Time')

# Adding legend
plt.legend()

# Show the plot
plt.show()

# Plotting declination as a function of time
plt.figure(figsize=(10, 5))
plt.plot(0, centroid1_dec_deg, label='Centroid 1', marker='o', linestyle='-')
plt.plot(-3, centroid2_dec_deg, label='Centroid 2', marker='o', linestyle='-')
plt.plot(3, centroid3_dec_deg, label='Centroid 3', marker='o', linestyle='-')
plt.plot(6, centroid4_dec_deg, label='Centroid 4', marker='o', linestyle='-')
# Adding labels and title
plt.xlabel('T-T0 (Julian Day)')
plt.ylabel('Declination (degrees)')
plt.title('Declination as a Function of Time')

# Adding legend
plt.legend()

# Show the plot
plt.show()

#plotting the centroids of each asteroid position for all four days

def convert_to_radec(X, Y):
    ra_rad = alpha0_rad + np.arctan2(X, -Y)
    dec_rad = np.arcsin(np.sin(delta0_rad) * np.cos(np.arctan2(X, -Y)) - np.cos(delta0_rad) * np.sin(np
```

```
    return ra_rad, dec_rad

# Function to convert radians to degrees
def radians_to_degrees(radians):
    return np.degrees(radians)

# Convert (X, Y) coordinates to (ra, dec) coordinates in radians
centroid1_ra_rad, centroid1_dec_rad = convert_to_radec(centroid1_X, centroid1_Y)
centroid2_ra_rad, centroid2_dec_rad = convert_to_radec(centroid2_X, centroid2_Y)
centroid3_ra_rad, centroid3_dec_rad = convert_to_radec(centroid3_X, centroid3_Y)
centroid4_ra_rad, centroid4_dec_rad = convert_to_radec(centroid4_X, centroid4_Y)

# Convert radians to degrees
centroid1_ra_deg = radians_to_degrees(centroid1_ra_rad)
centroid1_dec_deg = radians_to_degrees(centroid1_dec_rad)

centroid2_ra_deg = radians_to_degrees(centroid2_ra_rad)
centroid2_dec_deg = radians_to_degrees(centroid2_dec_rad)

centroid3_ra_deg = radians_to_degrees(centroid3_ra_rad)
centroid3_dec_deg = radians_to_degrees(centroid3_dec_rad)

centroid4_ra_deg = radians_to_degrees(centroid4_ra_rad)
centroid4_dec_deg = radians_to_degrees(centroid4_dec_rad)

# Print (ra, dec) coordinates for all four centroids
print(f"Centroid1 (ra, dec): ({centroid1_ra_deg}, {centroid1_dec_deg})")
print(f"Centroid2 (ra, dec): ({centroid2_ra_deg}, {centroid2_dec_deg})")
print(f"Centroid3 (ra, dec): ({centroid3_ra_deg}, {centroid3_dec_deg})")
print(f"Centroid4 (ra, dec): ({centroid4_ra_deg}, {centroid4_dec_deg})")
```

6.8. Computation of Velocity of Asteroid

COMPUTATION OF VELOCITY OF ASTEROID:

$$\begin{aligned}\text{The velocity in Degrees per day} &= \sqrt{(0.2728)^2 + (-0.2791)^2} \\ &= \sqrt{0.0744 + 0.07797} \\ &\approx \sqrt{0.15237} \\ &\approx 0.3903 \text{ degrees/day}\end{aligned}$$

$$\begin{aligned}\text{Converting to arcseconds per day} &= 0.3903 \times 3600 \\ &= 1405.08 \text{ arcseconds/day}\end{aligned}$$

$$\begin{aligned}\text{Converting arcseconds per hour} &= \frac{1405.08}{24} \\ &\approx 58.54 \text{ arcseconds/hour}\end{aligned}$$

6.9. Computation of Uncertainty of Velocity of Asteroid

$$uncertainty = \sqrt{\left(\frac{0.0527}{0.2791}\right)^2 + \left(\frac{0.0586}{0.2728}\right)^2}$$

$$\begin{aligned}\text{Converting to arcseconds} &= uncertainty \times 60 \\ &\approx 20 \text{ arcseconds/hour}\end{aligned}$$

6.10. Images Obtained during Reduction

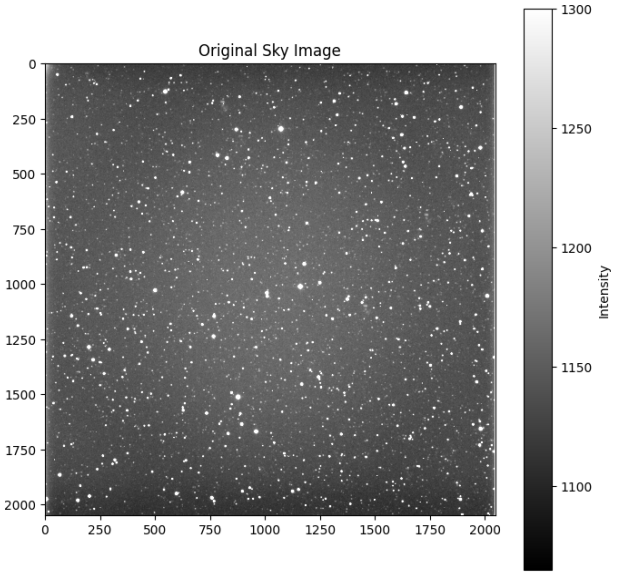


Fig. 9.—: Original Sky Image

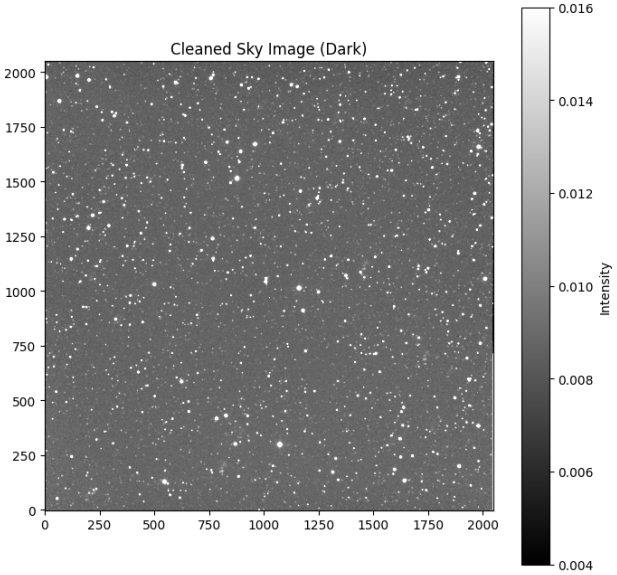
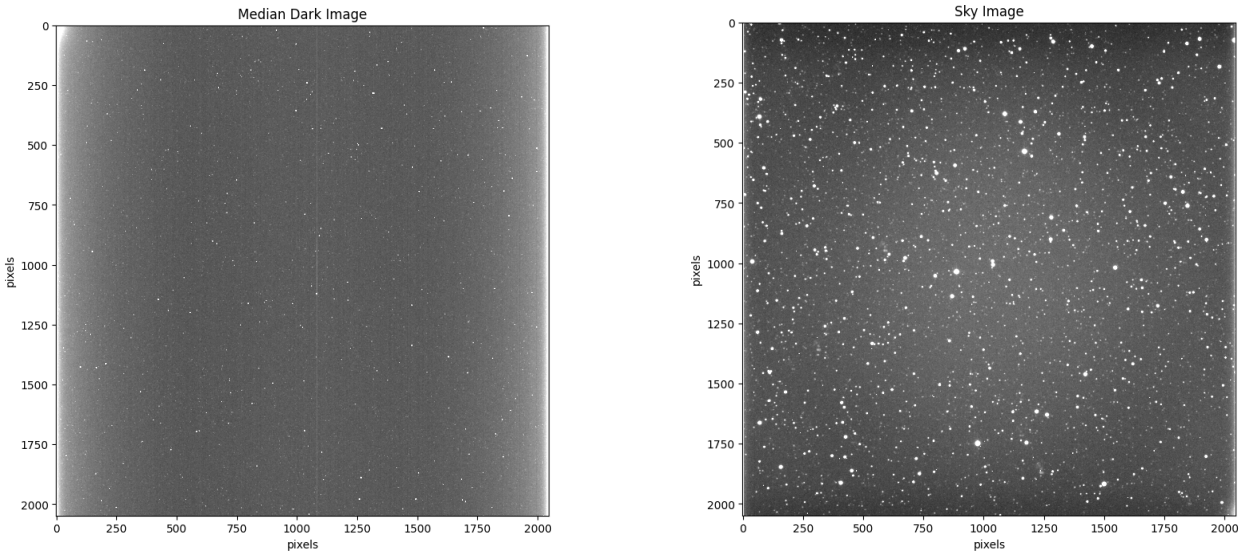


Fig. 10.—: Cleaned Data



(a) Name of Figure A

(b) Name of Figure B

Fig. 11.—: Median Dark Image vs Flipped Image.

6.11. How Threshold Intensity was chosen

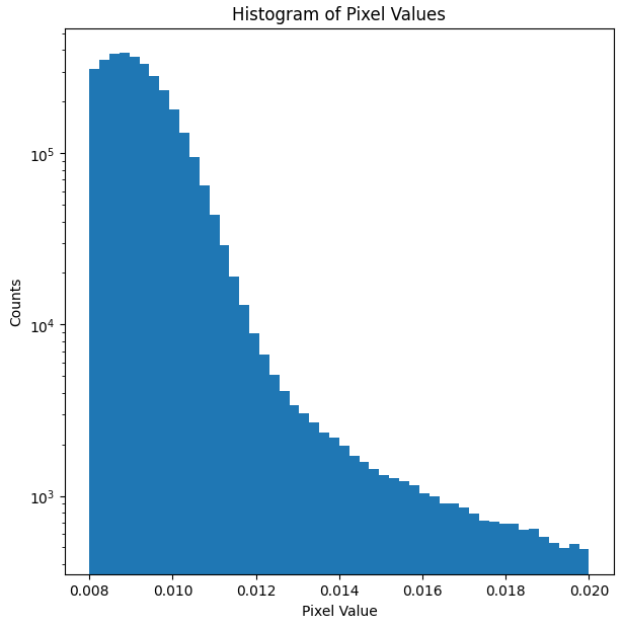


Fig. 12.—: Counts vs Pixel Values for Determining thershold intensity for choosing stars to centroid

6.12. Code for Cleaning Data

```
for i in range(len(asteroid_files_19)):
  header, img, center_of_field = read_file(asteroid_files_19[i], verbose = False, sky_image=True)
  img = img[::-1,::-1]
  clean_img_no_dark = (img - median_bias_files[0] )/(median_flat_files[0])
  clean_img_dark = (img - median_dark_files[0])/(median_flat_files[0])
  plot_image(img, title =  "Original Sky Image", log = True)
  #plot_image(clean_img_no_dark, title =  "Cleaned Sky Image (No Dark)", log = True)
  #plot_image(clean_img_dark, title =  "Cleaned Sky Image (Dark)", log = True)
  hdu = fits.PrimaryHDU(clean_img_dark)
  hdul = fits.HDUList([hdu])
  hdul.writeto(f'/content/drive/MyDrive/26Proserpina/Cleaned/cleaned_20120119_{i}.fits', overwrite=True


  plt.figure(figsize=(8, 8))
  plt.imshow(clean_img_dark, cmap='gray', vmin=0.004, vmax=0.016, origin = 'lower')
  plt.title( "Cleaned Sky Image (Dark)")
  plt.colorbar(label='Intensity')
  plt.show()


# Now we want to find the brightest points in the image for the normal stars so we can center them and
# Pick a value from the histogram we just plotted that you think may exclude the background but include
threshhold = 0.017
mask = np.ones_like(region)
masked_region = region * mask
# Find pixel positions where the intensity of the pixel is greater than the threshhold
bright_points = np.array(np.where(masked_region > threshhold))

# Remember to be careful of how indexing works in python
# The first index is the y position, the second index is the x position
xs = bright_points[1]
ys = bright_points[0]

plt.figure(figsize=(7,7))
plt.imshow(region, vmin=vmin, vmax=vmax, cmap='gray', origin='lower')
plt.title('Region of the Image');
plt.scatter(xs,ys,marker='x',color='red',label='Bright Points')
```

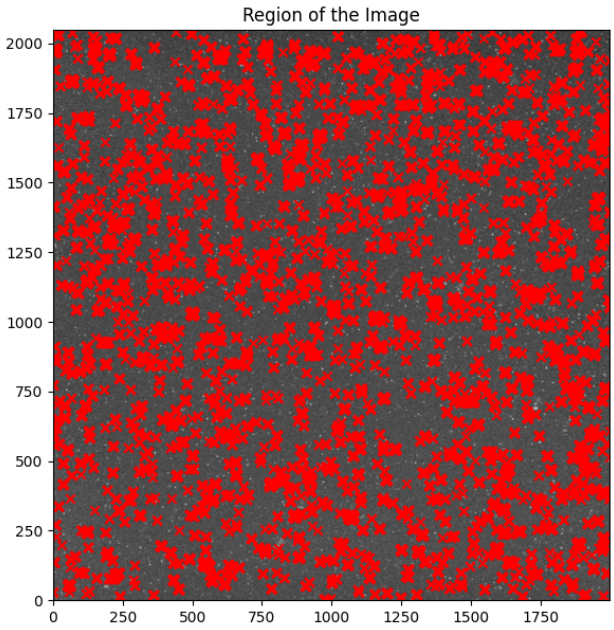


Fig. 1.—: Red marks indicate all the stars that were identified