Lab 3: Astrometry from CCD Images

Diana Kossakowski [*†]
Lab Group: ∼foxy∼
Melanie Archipley, Christopher Agostino

November 11, 2015

Abstract

Using the Nickel Telescope, we were able to (i) correct the image from a FITS file taking bias and flat fields into account, (ii) find the centroids automatically, (iii) compare the centroids we found from the CCD detector with known celestial objects from a star catalog in order to find a fit between the two using a general least-squares method, and (iv) calculate the distance to the asteroid from Earth using the parallax method. I focused on the asteroid 61 Danae in the infrared because it yielded the most centroids, and I found that it is $0.85847AU$ from the Earth and 1.81617 AU from the Sun, which unfortunately, they did not appear to be close to the actual distances.

1 Introduction

The main goal of this lab is to calculate the distance to an asteroid by observing its location relative to other celestial objects provided by CCD images, and then, we convert the location to right ascension (RA, α) and declination (DEC, δ) in order to determine the parallax. I will briefly summarize the steps in reaching our end result and later describe methods as we get to each step. Data comes in the format of a FITS file which contains the header and the raw image, which we need to correct using calibration images. Next, we take the corrected image and locate the centroids of visible objects to compare and match them to a known star catalog, where the objects have units of RA and DEC. Once we have that, we can convert the star catalog objects to standard coordinates and perform a general least-squares fit so that we can map the location of the asteroid in pixels to astronomical units of RA and DEC. Finally, we can apply the parallax method to obtain the distance between the Earth and the asteroid.

Analysis

Our data is observed by the Nickel Telescope, a 40 inch (1m) reflecting telescope, located near San Jose in California that has a 2048 x 2048 CCD detector and has a field view of approximately 6.3 arcminutes by 6.3 arcminutes. The telescope itself has a 2048 x 2048 pixel CCD detector, but when the data is read in as FITS files, the pixels are binned in 2 x 2 groups, and thus we have an image that is 1024 x 1024 pixels.

[*]dkossakowski@berkeley.edu
[†]https://github.com/dianadianadiana/AY120/blob/master/Lab3/

2 FITS Images

When we acquire the data, sometimes it is not a 1024 x 1024 pixel array, but rather a 1024 x 1056 pixel array. The last 32 additional pixels are called the cover pixels and they are not actually physical pixels. They are there to provide information about correcting the image in case we did not take the bias, but otherwise, since we do have the bias provided, from now on, when I am loading the fits file, I will be ignoring the last 32 columns because they are essentially not needed in our case.

2.1 Bias, Flat Fields, and Gain

If only the images we observe are perfect and do not need any modification, would we be able to merrily dive into analyzing the data; however, that is not the case, and we must correct each image through a process of using a bias, flat, and additional corrections. Before a night of observing, one must take calibration images to account for how each pixel reacts in unsaturated and saturated situations, which we acquire from the bias and the flat fields, respectfully. Additionally, we noticed in the raw data how there are 9 "dead pixels", where they do not accurately detect photons. In order to deal with those, I took the median of the 1024 x 1024 raw image and set that value to each arbitrarily picked "dead pixel". I decided to use the average and not just set the value to zero for a more "smooth" gradient when viewing the image as well as a more "likely" value for that pixel.

2.1.1 Bias

The bias is a zero second exposure time correction that is independent of filter and its purpose is to pick up any digital noise caused by the instrument itself since in principle, no signal should be detected. Around 10 to 15 bias calibrations are recorded every night and even though one may think that the bias should not be changing drastically much between days, it does in fact change quite a bit, so we had to make sure we were applying the right bias each night. To minimize errors, we decided to take all bias files for each night and average them out. This was easily done but just summing all the bias files and dividing by the number of bias files. Just as a clarification, when I say we "averaged" the bias files (or any other set of files), what I mean is that at each pixel coordinate, we took the sum of all the values at that coordinate from the different files and then divided by the number of files. What we are NOT doing is calculating the average of each file by summing up all the values given by all the pixels and then dividing that sum by the number of pixels.

2.1.2 Flat Fields

The purpose of the flat field is to collect and obtain the best signal-to-noise ratio while avoiding over-saturation at each pixel in order to determine how each and every pixel collects light and how the relative pixel-to-pixel interaction responds to the photons hitting the detector. They are taken facing the sun either at sunset or sunrise. These flats are around five to ten seconds and must be taken in the four different filters (B, V, I, and R), where there are 3 flat fields per filter. When we do not take these into account, we can easily see an annulus or ring form in the image. Unlike with the bias images, we did not average the flat files since in the data log, they each had different notes and so we inferred that they

can't just be averaged and a different approach must be taken (explained in the following paragraph).

2.1.3 Gain

Now that we have all the tools we need, we can finally correct the image. To do so, we will introduce the concept of gain, given by equation 1.

$$corrected\ image = (R - B) * G,\ where\ G = m/(F - B),\ where\ m = avg(F - B) \quad (1)$$

where R, B, G, and F correlate to the raw image, the bias, the gain, and the flat field respectfully, and they are all 1024 x 1024 pixel arrays. The raw image is just the image we obtain from the fits file and the bias image is simply the averaged bias image, which both are corrected for the dead pixels. To get an accurate result for the gain used in the above equation on a raw image, we calculated the gain for each flat field of a certain filter and then averaged all the gains (similar to the process of averaging out the bias). We explored the possibility of just using (i) one flat field for the correction, (ii) averaging out the flat fields first and then finding the gain, and the option we ended up using is (iii) finding the gain for each flat field and then averaging out the gains. Honestly, these different approaches did not seem to make THAT much of a difference to our eyes visually, but I suppose that microscopically to the computer and for future measurements, option (iii) is more desirable because it yields the most accurate value for a gain and leaves little room for error.

With all of this said and done, I made multiple functions that would just take the fits file in interest and figure out which bias to use and which flat to use based on the file's header, and thus, beautifully correct the image, as we can see below in Figure 1. The raw is not corrected for its dead pixels and still includes all of its cover pixels. It is amazing to see how the corrected image displays many objects that were not easily seen before the modifications.

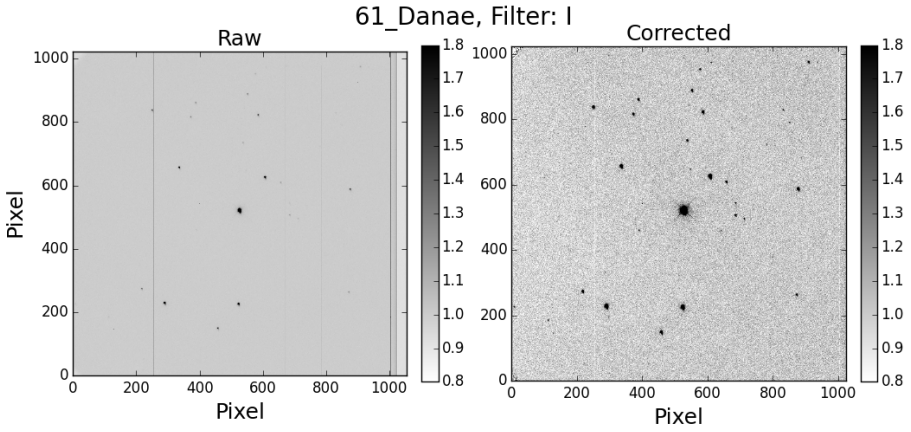

Figure 1: The raw, and corrected images are all displayed here, using Danae in the "I" filter from the night of 10/13/15.

3 Star Matching

Now that we have obtained our corrected image, we can finally have some fun and compare our images to a star catalog to reach our end goal of determining the location of our asteroid. To do so, we first need to calculate the centroids of the stars we find in our corrected image and then compare the x- and y- values to those in a star catalog, such as as the US Naval Observatory USNO-B1.0 Catalog[1], where thousands of celestial objects and their locations have been recorded. We used ALADIN[2], an interactive star atlas, to help aid us in determining if we are using the right field and orientation in the USNO catalog. We can then compare the USNO stars to our calculated centroids and then find the position and location of our asteroid.

Before I begin describing how the centroids were found, I first want to further explore how our corrected image compares to that what we can find using ALADIN by plugging in the right ascension and declination obtained from the FITS file header used to create Figure 2. Right away, we can match up key objects to see how ALADIN correlates with what we obtain from the telescope. For example, in this situation, we can match up the bottom three points in Figure 2 to the top three points in Figure 3, and thus conclude that the images from the telescope must be flipped upside down.

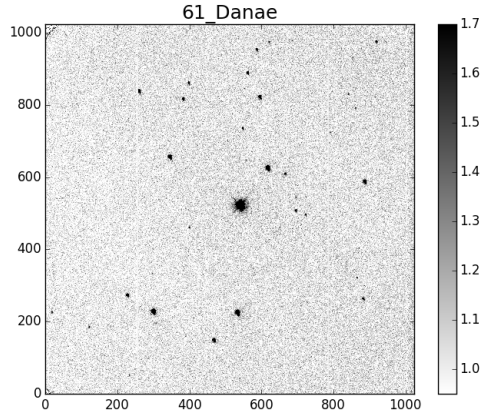

Figure 2: The corrected image for Danae in the Infrared taken on 10/13/15.

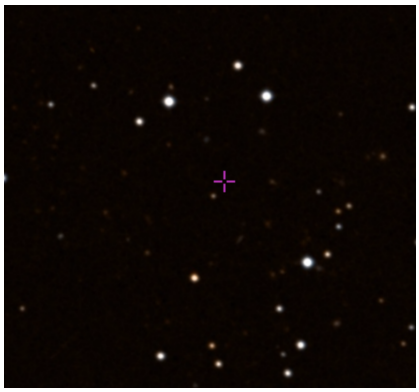

Figure 3: The ALADIN image spans roughly 6.3 arcminutes, where the *cross* indicates the RA and DEC used as the center in Figure 2.

3.1 Centroids

Though it might have been easier to manually pick out the peaks and their respective sizes, I decided to write a program that automatically determines the peaks so that I could easily apply it for any image, which would yield more and faster results in the long run. I define a peak to be the pixel that reads the highest value of intensity surrounding said pixel. Once we get these peaks, it is important to realize that the physical pixel of the peak itself is

[1]http://tdc-www.harvard.edu/software/catalogs/ub1.html
[2]http://aladin.u-strasbg.fr/

not necessarily equal to its center of intensity, so we had to calculate the true center of the intensity using centroid equations, given by equation 2.

$$x_{centroid} = \left(\frac{\sum_i^{size} I_i x_i}{\sum_i^{size} I_i} \right); \qquad y_{centroid} = \left(\frac{\sum_i^{size} I_i y_i}{\sum_i^{size} I_i} \right) \tag{2}$$

To find the peaks, I implemented a primary base condition that only the pixels that were 2.2 times above the average of the 1024 x 1024 corrected image were considered. The constant of 2.2 was arbitrarily picked after experimenting with different values; anything higher would not output enough pixels and anything below would output too many pixels. After that step, many lone pixels (including those in the corners of the CCD image) that just happened to be above the threshold needed to be discarded, so I applied many other filters to ensure that these false-positives were not considered. Then, I grouped pixels that were directly neighboring each other and out of those grouped pixel clusters, I determined which pixel in that group reads the highest value/intensity reading. The size of the peak was determined by how many pixels were in each cluster. I realize that there is room for error when determining the peak size since it is possible that the peak may in fact be 13 pixels wide rather than 8 pixels wide as my program would pick up, due to the fact that the outside pixels may have not satisfied the threshold condition of being above a certain value. However, I think it is safe to trust the given size that may be an underestimate because of the way I have calculated the centroids (where I take the peak pixel and look at the pixels that are $\pm size$ away).

To calculate the centroids, I essentially calculated the center of intensity for a square of length $2 * size$, centered at the peak pixel, denoted by (x,y). To do this, the process becomes much easier if you separate out the center of masses for x_{cm} and y_{cm} by calculating them independently. For finding the x_{cm}, you calculate multiple x_{cm}'s and then find the average. For example, you find one x_{cm} by varying x from $x = x - size$ to $x = x + size$ while keeping y at a certain pixel value. Then you would find the next x_{cm} by using a different y value, and so on, until you find all the x_{cm} for y values of $y = y - size$ to $y = y + size$ (equation 3). Likewise, we do the same thing for finding y_{cm} (equation 4).

Ideally, I would have preferred calculating the center of intensity for a circle of radius $size$ centered at (x,y) since it better mimics a star on the CCD detector, but I believe the square still does a good enough approximation, as shown in Figure 4. Finding the errors in the centroids is similar to the previous lab, through the use of error propagation, finding that the errors are negligible.

$$x_{cm} = \frac{(x_{cm,y=y-size} + x_{cm,y=y-size+1} + ... + x_{cm,y=y} + ... + x_{cm,y=y+size-1} + x_{cm,y=y+size})}{2 * size} \tag{3}$$

$$y_{cm} = \frac{(y_{cm,x=x-size} + y_{cm,x=x-size+1} + ... + y_{cm,x=x} + ... + y_{cm,x=x+size-1} + y_{cm,x=x+size})}{2 * size} \tag{4}$$

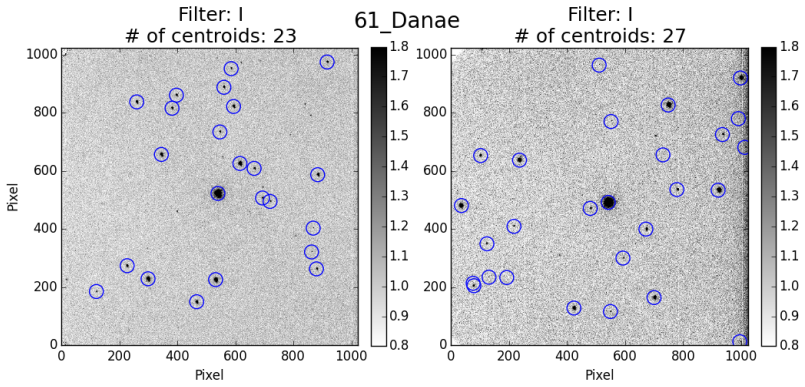

Figure 4: Both plots are the corrected images of Danae in the infrared, where the left one is from 10/13/15 and the right one is from 10/19/15. The centroids found by the automatic program are circled.

3.2 Converting Coordinates

The image we get from the CCD detector has a field of view that is roughly 6.3 arcminutes, and so, Figure 5 displays the predicted USNO catalog coordinates in units of both RA vs DEC.

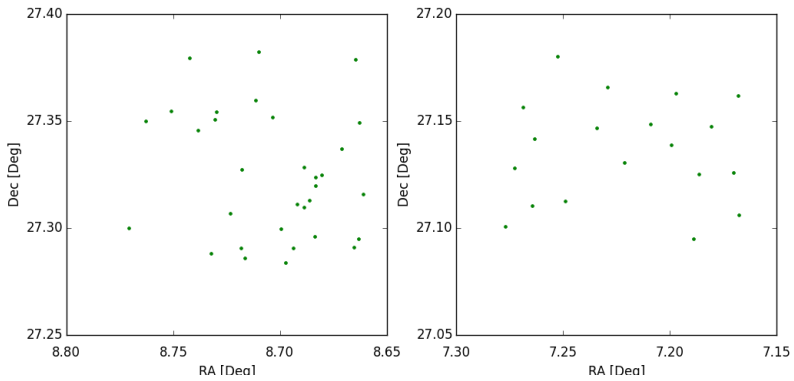

Figure 5: Both plots display the USNO catalog coordinates of magnitudes less than 18. The left and right plots correspond to Danae in the infrared on 10/13 and 10/19 respectfully.

Now that we have the centroids in units of pixels from the CCD detector, we can compare our calculated x- and y-values to the USNO catalog, where objects are in terms of right ascension and declination, so that we can convert RA and DEC to pixels and find a good fit. In order to do so, we must first convert the RA and DEC to the standard coordinates of X and Y,

$$X = -\frac{\cos\delta\sin(\alpha-\alpha_0)}{\cos\delta_0\cos\delta\cos(\alpha-\alpha_0)+\sin\delta\sin\delta_0}; \qquad Y = -\frac{\sin\delta_0\cos\delta\cos(\alpha-\alpha_0)-\cos\delta_0\sin\delta}{\cos\delta_0\cos\delta\cos(\alpha-\alpha_0)+\sin\delta\sin\delta_0} \tag{5}$$

where α_0 and δ_0 are the RA and DEC of the center of the field of view, which we obtain from the header of the asteroid FITS file. α and δ are the RA and DEC from each star position from the USNO catalog that we're considering. Even with this in mind, we need

to do another transformation to account the focal length of the telescope and the pixel size, represented by $f = 16,840mm$ and $p = 0.015mm$, respectfully.

$$x = f\left(\frac{X}{2p}\right) + x0; \qquad y = f\left(\frac{Y}{2p}\right) + y0 \tag{6}$$

where x_0 and y_0 are both 512 as they both represent the center of the CCD (half of 1024). The reason we multiply p by 2 is to account for the binning for the CCD pixels. Figures 2 and 3 have already showed how the objects from the USNO catalog must be flipped upside down. Both Figure 6 and 7 have the CCD coordinates and the USNO catalog points plotted together. The only difference between them is the magnitudes we choose to see from the USNO catalog. When the magnitude is higher, we can not really pick out a pattern between our CCD centroids and the objects from the USNO catalog, but when we allow objects with a magnitude less than 18, we can start to see a pattern emerge.

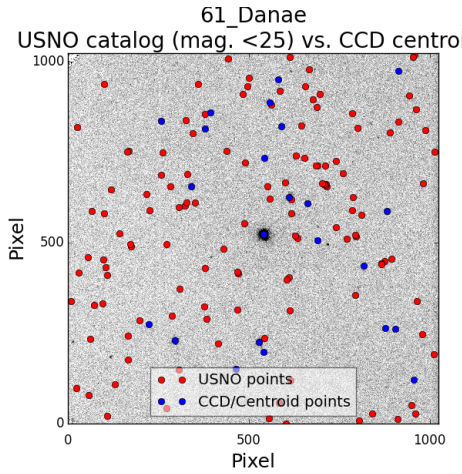

Figure 6: CCD centroids vs USNO catalog co-ordinates, magnitude less than 25. Danae, in-frared, 10/13

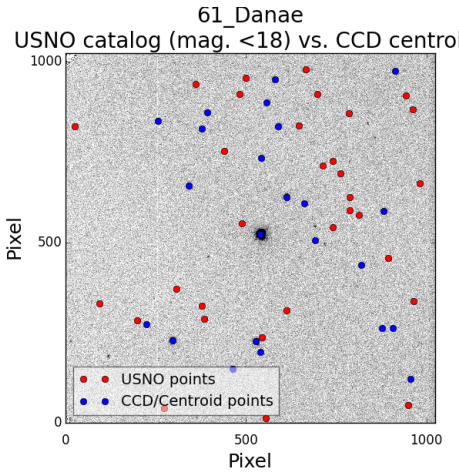

Figure 7: CCD centroids vs USNO catalog coordinates, magnitude less than 18. Danae, infrared, 10/13

Least-Squares Fitting

Looking at Figure 7, we can see how the predicted USNO catalog coordinates are off set and shifted towards the top right compared to the calculated CCD centroids. Now, we have to pair up USNO coordinates with the centroids in order to create a fit. To make this "matching" process easier, I decided to implement a user-input mechanism through Python by asking the user to match the CCD coordinate to the USNO one by labeling each point, as in Figure 8.

Initially, we took the corresponding coordinates and decided to approach the same linear-least squares method and its equations that we used in the previous lab[3], by treating x and y independently. Unfortunately, the fit was not adequate enough as we can clearly see in Figure 9, where the fit seems to missing some rotation factor. Residuals (not shown) were exhibiting the fit to be on the order of 30 pixels off.

[3]Least-Squares Fitting, J. Graham: https://drive.google.com/file/d/0B40Ynk22SiBpVlRKZFdhTnZMQkU/view

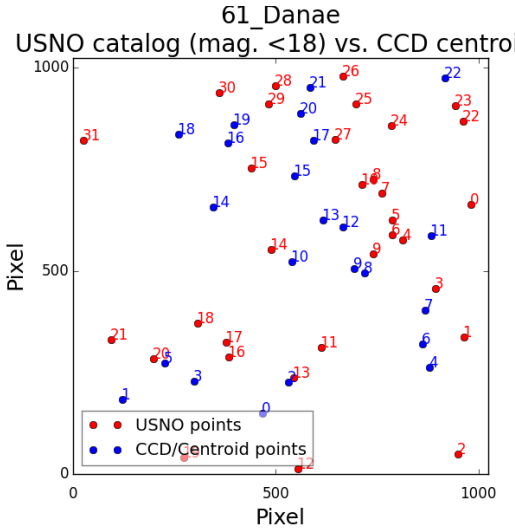

Figure 8: An annotated plot of Figure 7

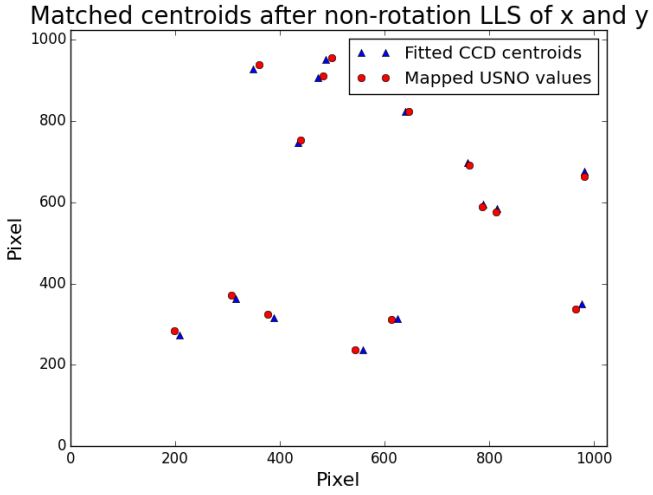

Figure 9: First linear least-squares of of Figure 7

Instead, we had to opt for a full linear transformation[4] that takes into account magnification, shear, rotation, and the translation. For a full explanation and derivation of this method, please refer back to the footnote. Therefore, we need to readjust equation 6 and incorporate θ, the rotation between the two frames to get,

$$x = f/p\,(Xcos\theta - Ysin\theta) + x0; \qquad y = f/p\,(Xsin\theta + Ycos\theta) + y0 \tag{7}$$

$$\mathbf{x} = \mathbf{TX}; \quad where\ \mathbf{X} = (X, Y, 1)\ and\ \mathbf{x} = (x, y, 1); \quad where\ \mathbf{T} = \begin{pmatrix} (f/p)a_{11} & (f/p)a_{12} & x_0 \\ (f/p)a_{21} & (f/p)a_{22} & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{8}$$

where a_{11}, a_{12}, a_{21}, a_{22}, x_0, and y_0 are called the "plate constants" which we figure out using linear least squares. a_{ij} and x_0, y_0 correspond to scale, shear and orientation of the image, and the pixel offset, respectfully. (X_i, Y_i) and (x_i, y_i) correspond to the independent and dependent variables, which are the USNO catalog coordinates and the CCD values, respectfully. We can see how the rotation offset is fixed by this general least-squares method, yielding a beautifully fit plot, with residuals ± 1 (Figure 10). We can see the different values of the plate constants in the table below, showing how they are relatively the same, yet they vary based on which fits file you are considering.

Object	a_{11}	a_{12}	a_{21}	a_{22}	x_0	y_0	$\sqrt{det(\mathbf{T})}(f/p)$	rms_x	rms_y
D 10/13	0.9901	-0.0352	0.0352	0.9909	425.040	420.237	0.99117	0.3396	0.4753
D 10/19	0.9896	-0.0360	0.0350	0.9905	498.713	482.427	0.99076	0.3396	0.4753
D 10/20	0.9899	-0.0348	0.03515	0.9903	444.915	462.051	0.99078	0.3255	0.3051

[4]General Liner-Least Squares, J. Graham: https://drive.google.com/file/d/0B40Ynk22SiBpOHBJMjRtY3JsN1E/view

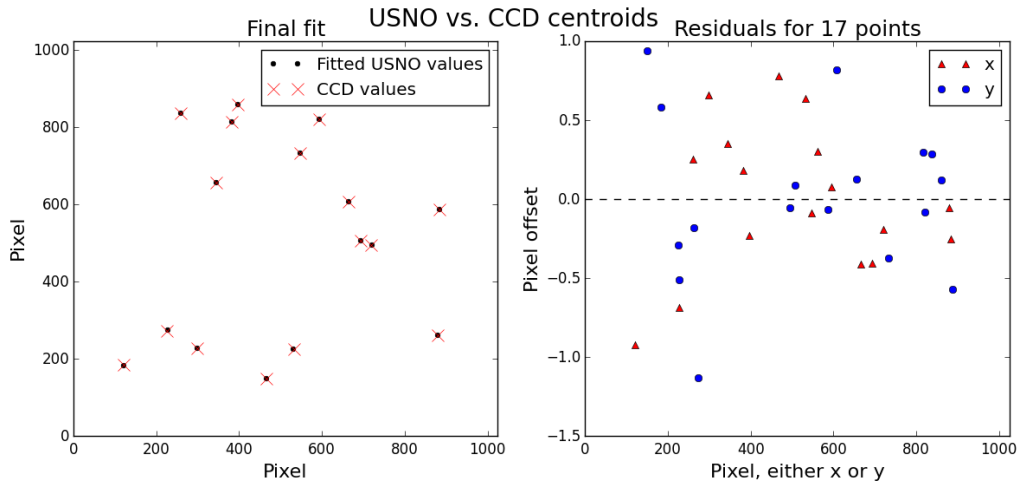

Figure 10: The left corresponds the the general linear least-squares fit; we can see how the rotation is accounted for completely. The right shows how well the fit executed with errors of ± 1 pixel.

4 Finding the Distance

We finally made it to this point, however, the difficulty is just getting started. Now we will look at the relative motion of the asteroid using data from three different days. We first used the fit we obtained in section 3.2 to calculate the RA and DEC of our asteroid from its coordinates in pixels from the CCD image; and the trajectory of the asteroid is shown in Figure 11. To see if our findings for RA and DEC were appropriate, we compared them to what JPL HORIZONS has and found little error (table below).

Object	*RA*	*RA JPL*	*DEC*	*DEC JPL*	*RA error*	*DEC error*
61 Danae 10/13	8.70125	9.0042	27.34144	27.344	3.3645%	.0093%
61 Danae 10/19	7.21409	7.4958	27.13019	27.176	3.7582%	.1685%
61 Danae 10/20	7.00514	7.2542	27.08210	27.136	3.4333%	.1986%

Using the parallax method[5], we use the RA and DEC in the table above to convert to equatorial coordinates using this relation,

$$x_{eq} = cos\alpha cos\delta; \quad y_{eq} = sin\alpha cos\delta; \quad z_{eq} = sin\delta \tag{9}$$

and then we can do a matrix transformation to find ecliptic coordinates (where $\epsilon = 23.434929111(^\circ)$ in the J2000 equinox),

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\epsilon & sin\epsilon \\ 0 & -sin\epsilon & cos\epsilon \end{bmatrix} \begin{bmatrix} x_{eq} \\ y_{eq} \\ z_{eq} \end{bmatrix} \tag{10}$$

[5]Parallax Determination for an Asteroid, J. Graham: https://drive.google.com/file/d/0B40Ynk22SiBpTWpmSVBxOGszN

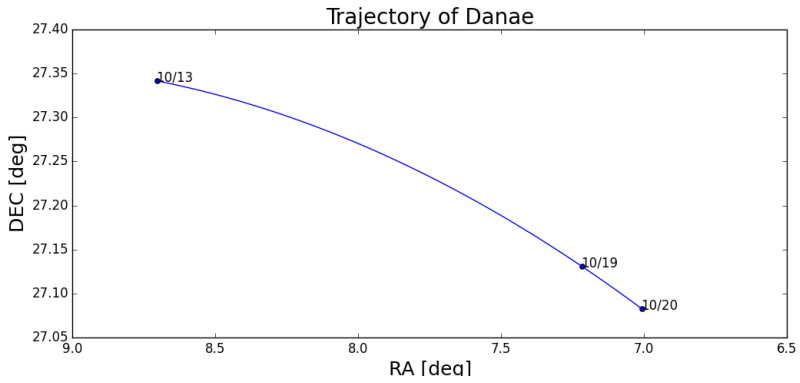

Figure 11: Plotted is the RA and DEC for corresponding 61 Danae locations.

Now, we can calculate each ecliptic position vector (from Earth) using the expression above and denote them as $\mathbf{s_1}$, $\mathbf{s_2}$, and $\mathbf{s_3}$. Next, we have to find $\dot{\mathbf{s}}$ and $\ddot{\mathbf{s}}$ using,

$$\dot{\mathbf{s}} = \frac{\tau_3(\mathbf{s_2} - \mathbf{s_1})}{\tau_1(\tau_1 + \tau_3)} + \frac{\tau_1(\mathbf{s_3} - \mathbf{s_2})}{\tau_3(\tau_1 + \tau_3)}; \quad \ddot{\mathbf{s}} = \frac{\mathbf{2}(\mathbf{s_3} - \mathbf{s_2})}{\tau_3(\tau_1 + \tau_3)} - \frac{\mathbf{2}(\mathbf{s_2} - \mathbf{s_1})}{\tau_1(\tau_1 + \tau_3)} \tag{11}$$

where τ is the time interval in days; $\tau_1 = 6.01118$ and $\tau_3 = 0.93524$. With all this information, we can determine our best guess for the distance to the asteroid from the Sun, r, and the distance to the asteroid from the Earth, ρ. We used HORIZONS[6] to generate ephemerides for solar-system bodies in order to compare our calculated answers. Using the equations below, we were iteratively guessing r and plugging it into equation 12(a) to get a value for ρ to plug it equation 12(b), and so on, until the difference between the calculated r and guessed r was less than 10^{-21}.

$$\rho = k^2 \left(\frac{1}{R^3} - \frac{1}{r^3} \right) \frac{\dot{s} \cdot (R \times s)}{\dot{s} \cdot (\ddot{s} \times s)}; \quad r^2 = \rho^2 + R^2 + 2\rho \vec{R} \cdot \vec{s} \tag{12}$$

The calculated r and ρ are found in the table below, yielding undesirable results.

Quantity	Computed Value (AU)	Value from JPL (AU)	% error
ρ	0.85847	1.61	46.678%
r	1.81617	2.55	28.777%

Conclusion

Clearly my values for r and ρ are no where near the expected values I would hope to see, and quite frankly, I am quite puzzled as to why not. The plate constants all make sense and the calculated RAs and DECs are within a 4% margin of what JPL has. I definitely am disheartened to not have the actual calculation of the asteroid's distance. Nonetheless, this lab truly proved to be challenging but rewarding as I was able to learn how to handle and manipulate FITS files effectively, calculate the centroids on the CCD detector and compare them to the star catalog, and find a way to convert pixels to RA and DEC. All of the above was done with little error, and I was able to see how we would be able to compute the distance to the asteroid which would have been truly amazing to obtain. Lastly, I am very proud that I was able to develop a program that automatically calculates the centroids based on the input of a FITS file.

[6]HORIZONS: http://ssd.jpl.nasa.gov/?horizons