AST325 Lab 3

Chris Compierchio

December 6 2021

1 Abstract

In this report, CCD images were taken to measure the positions of stars relative to the celestial coordinate system. The proper motions of asteroids were also observed. Datasets were taken from the Dunlap Institute telescope. This telescope is a 50-cm telescope located in New Mexico. It will be evident in this report that the celestial coordinates of a celestial object can be used to evaluate the object's proper motion through the analysis of surrounding stars. Various error analysis techniques were also used throughout the report.

2 Introduction

In astrometry, the proper motion of celestial objects is the measure of the observed changes in the positions of celestial objects in relation to the center of mass of the solar system. The proper motions of celestial objects can be used to determine their distance from Earth, their mass and even their speed as they fly through our solar system. In this lab, the proper motion of an asteroid is found in relation to the position of a group of stars.

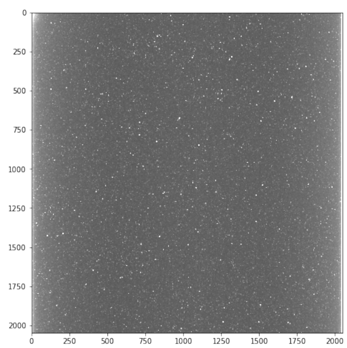
The purpose of this lab was to determine the positions of stars as well as the proper motion of a given asteroid. Data taken from the Dunlap Institute telescope was used to measure the positions of the stars in a given CCD image.These stars were then compared to that of the US Naval Observatory B-1.0 star catalog. Using this data, the position of a given asteroid was found, as well as its proper motion.

3 Observations and Data

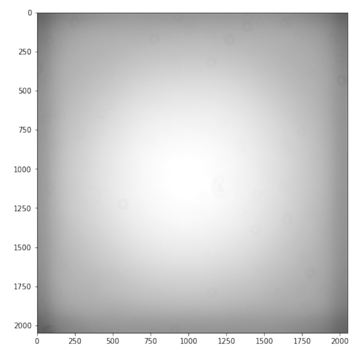
Due to the ongoing Covid-19 pandemic, the data used for this report was supplied to me by my professor.

4 Data Reduction and Methods

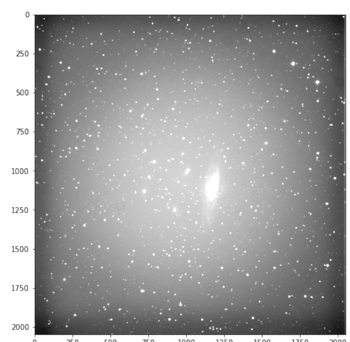
First, the image captured had to be corrected. The raw NGC-7331 file was a raw image of the galaxy, and due to some potential imperfections in the image, a flat correction had to be applied to it. A "dark" image and a "flat" image were both provided. The data in the dark image had to be subtracted from the raw data, and the data from the flat image had to be divided out of the raw data. The following are the dark, flat, and corrected images:



(a) Dark Image



(b) Flat Image



(c) Corrected Image

Figure 1: Images of NGC-7331

The stars located in the NGC-7331 image files were located using a centroiding method written in Python (see Appendix). A loop was created to find the brightest pixel in the image. The location of this pixel was recorded and placed in an array. A circle was drawn around this pixel, then the loop was started again. It is important to note that previously found pixels were not included when the loop ran again to avoid overlap. The centroids of each star were calculated using the following formulae:

$$< x >= \sum_i x_i I_i / \sum_i I_i, \; < y >= \sum_i y_i I_i / \sum_i I_i$$

The following was the resultant image:

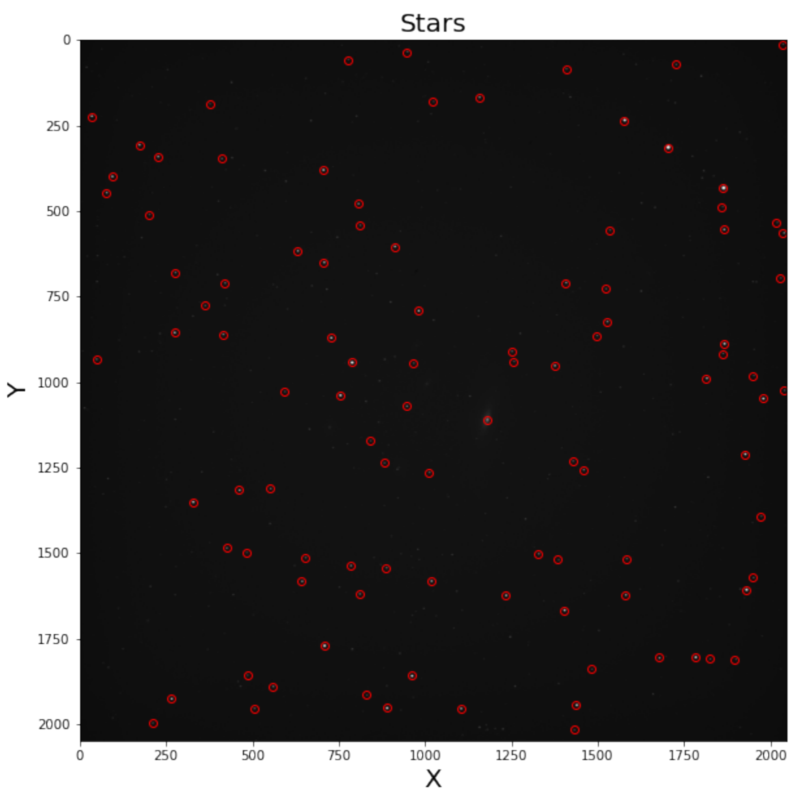


Figure 2: Locations of stars in NGC-7331 Image

Next, the USNO catalog was used to cross-correlate the stars in the NGC image. Since the USNO catalog shows stars on the celestial sphere and are measured in right ascension and declination, the pixel coordinates of the catalog were also plotted. The pixel coordinates were converted from celestial coordinates using the following formulae:

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

$$Y = \frac{-sin(\delta_0)cos(\delta)cos(\alpha -_0) - cos(_0)sin(\delta)}{cos(\delta_0)cos(\delta)cos(\alpha - \alpha_0) + sin(\delta)sin(\delta_0)}$$

The following image is the USNO catalog in celestial coordinates:

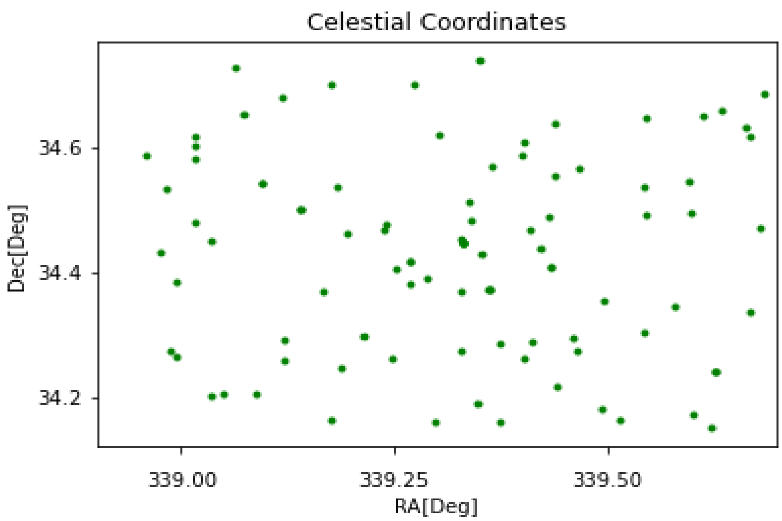


Figure 3: USNO Catalog in Celestial Coordinates

The next step was to correct the raw CCD data so that it fit the USNO data. As explained in the discussion section, the raw CCD data will have some flaws due to various reasons, and thus the data can be corrected to fit that of the USNO catalog. Below is the raw data over the USNO catalog:

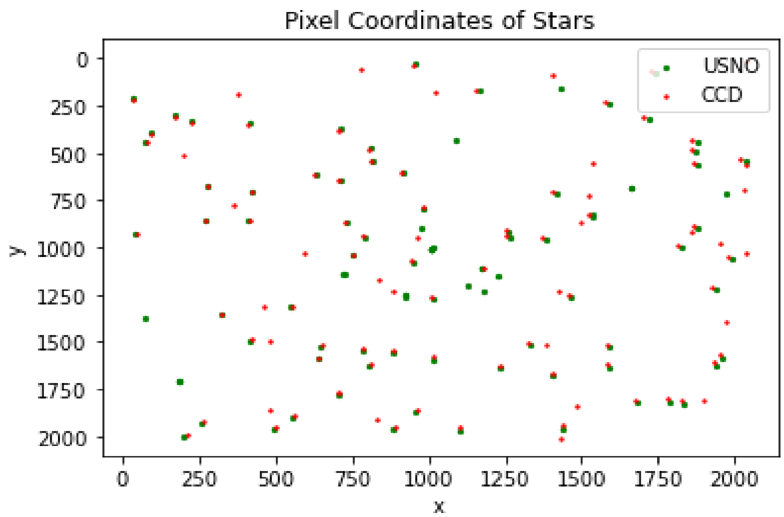


Figure 4: USNO Catalog and CCD data in Pixel Coordinates

As evident in the plot, there are a lot of outlier stars. Most of the stars from the CCD data also do not align with the USNO data, and so this could be fixed. First, removing the outlier stars gives:

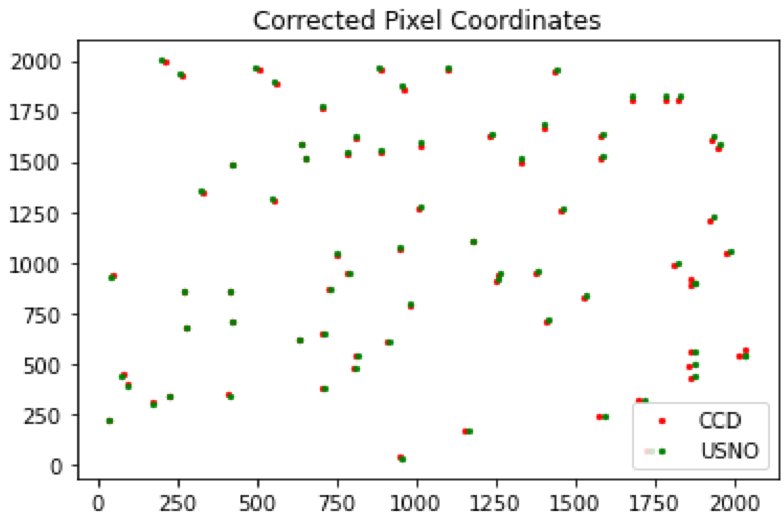


Figure 5: Corrected USNO Catalog and CCD data in Pixel Coordinates

Still, a lot of the stars are not lined up, so a correction can be performed (see appendix) to fix this and to have the raw ccd data be corrected to match that of the USNO catalog:

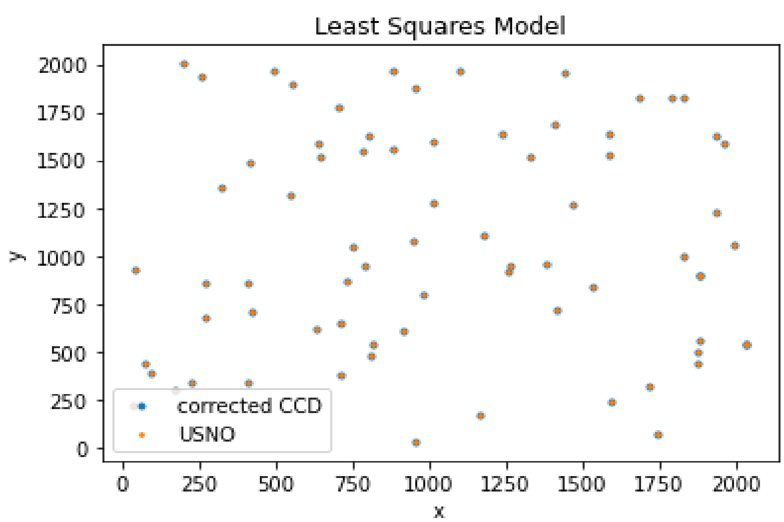


Figure 6: Corrected USNO Catalog and CCD data in Pixel Coordinates

Next, the plate constants were found using a linear least squares regression method. The following matrix was created to store the X and Y positions of the stars:

$$\mathbf{B} = \begin{pmatrix} (f/p)X_1 & (f/p)Y_1 & 1 \\ (f/p)X_2 & (f/p)Y_2 & 1 \\ \vdots & \vdots & \vdots \\ (f/p)X_N & (f/p)Y_N & 1 \end{pmatrix}$$

Figure 7: Matrix B from Lab Document Appendix 5

The matrix of plate constants was then solved for by using the following matrix formula:

$$\mathbf{c} = \left(\mathbf{B}^T\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{a}\,.$$

Figure 8: Equation c from Lab Document Appendix 5

where the matrix a contains the NGC star data. The plate constants were found to be 9.91E-1, 6.626E-3, 1.02E3, -4.67E-3, 9.89E-1, 1.02E3.

5 Data Analysis and Modelling

Not much error analysis was completed in this lab due to only some steps being fully completed, however, the reduced chi-squared statistic for the affine transformation was computed using the following formula:

$$\chi^2 = (B - ac)^T(a - Bc) \tag{1}$$

where B is shown in Figure 7, a contains the x or y values, and c contains the plate constants.

6 Discussion

For this lab, I was not able to compute the location nor the proper motion of the asteroid 26 Proserpina, however, I did get up to the point where I had to find the plate constants for NGC-7331. Plate constants are used for shear, rotation, translation, and magnification discrepancies between the measured data and the actual data. In this case, the measured data was the raw NGC-7331 data measured from the Dunlap Institute telescope and the actual data was the data taken from the UNSO catalog. This data was aligned using an affine transformation method shown in the appendix.

Due to the ongoing COVID-19 pandemic, I was not able to collect my own data, thus the errors given in this lab were very minimal, however, one could assume that some error was made throughout the measurement process. Not every telescope is perfect, and so the positions of the stars measured, as well as the position of the NGC-7331 galaxy, may not have been perfect.

In terms of calculated errors for this lab, the reduced chi-squared values were recorded for the affine transformation portion, as discussed in the Data Analysis and Modelling section. The values were about 6.187 for the x coordinates of the raw NGC data and about 23.96 for the y coordinates. These are quite high compared to a perfect value of 1, so this means the fit was somewhat poor. Perhaps the fit could have been improved by calibrating the telescope better.

7 Conclusion

To conclude, the purpose of this lab was to perform an affine transformation to find the location of stars in an image of NGC-7331. The plate constants were found to apply transformations to a raw data image of the galaxy to align the image with the USNO catalog.

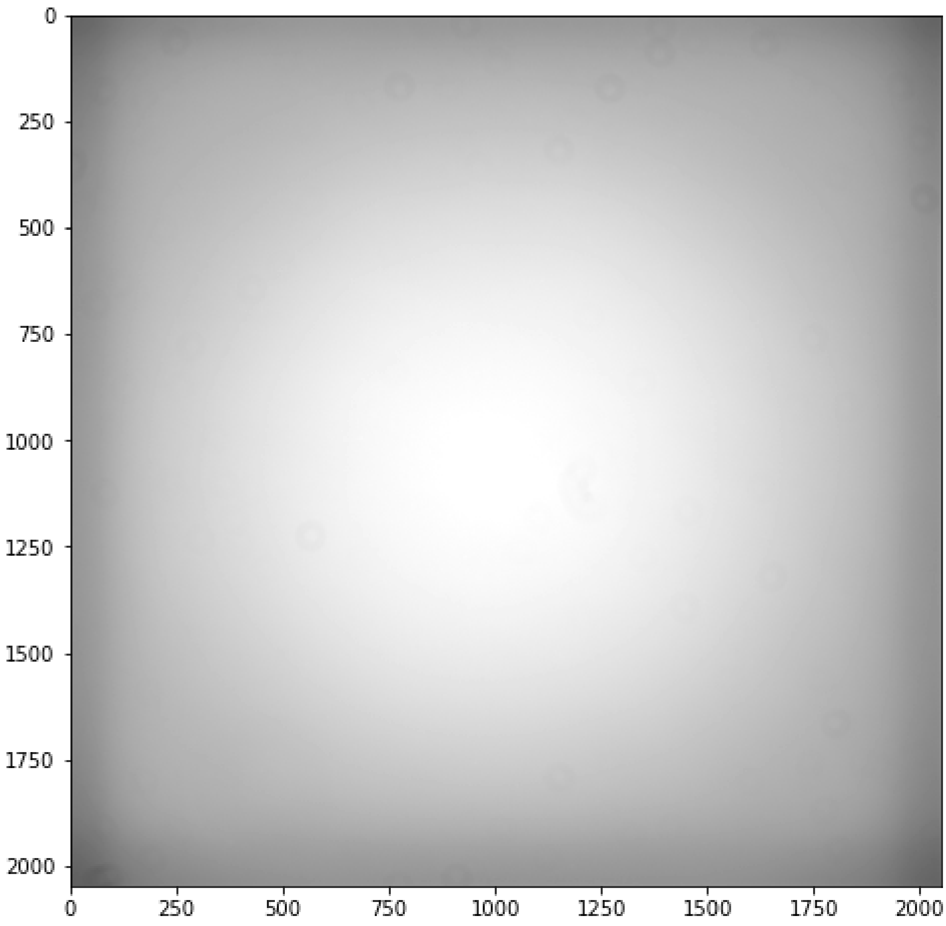
8 Appendix A

In [1]:

```
#import libraries
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt
%matplotlib inline
from scipy.stats import *
import scipy as sp
from astropy.io import fits
import urllib as url
from matplotlib.colors import LogNorm
from scipy.optimize import curve_fit
import os
from matplotlib.ticker import MultipleLocator
import urllib.request
from array import *
```

In [2]:

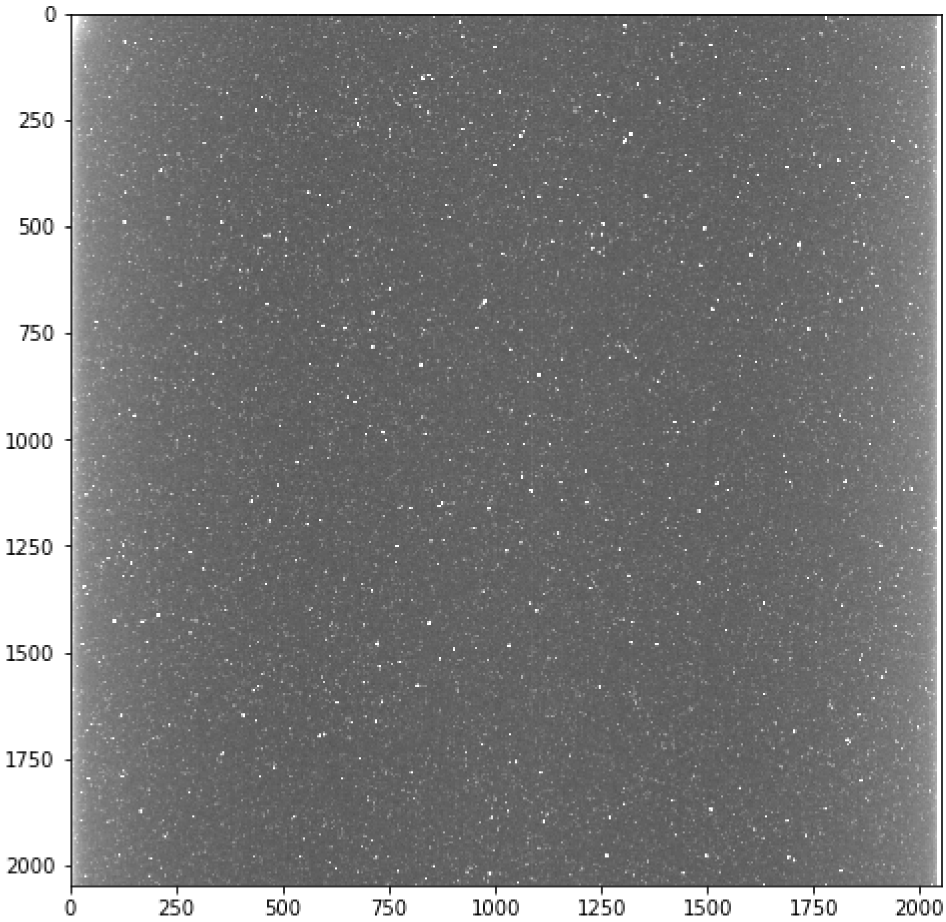
```
#open and disply the flat field data
combflat = fits.open('combflatr.fits')
header = combflat[0].header

data1 = combflat[0].data
data1.shape
plt.figure(figsize=(8, 8))
plt.imshow(data1.squeeze(), vmax=np.percentile(data1, 99), cmap="gray")
plt.show()
```



In [3]:

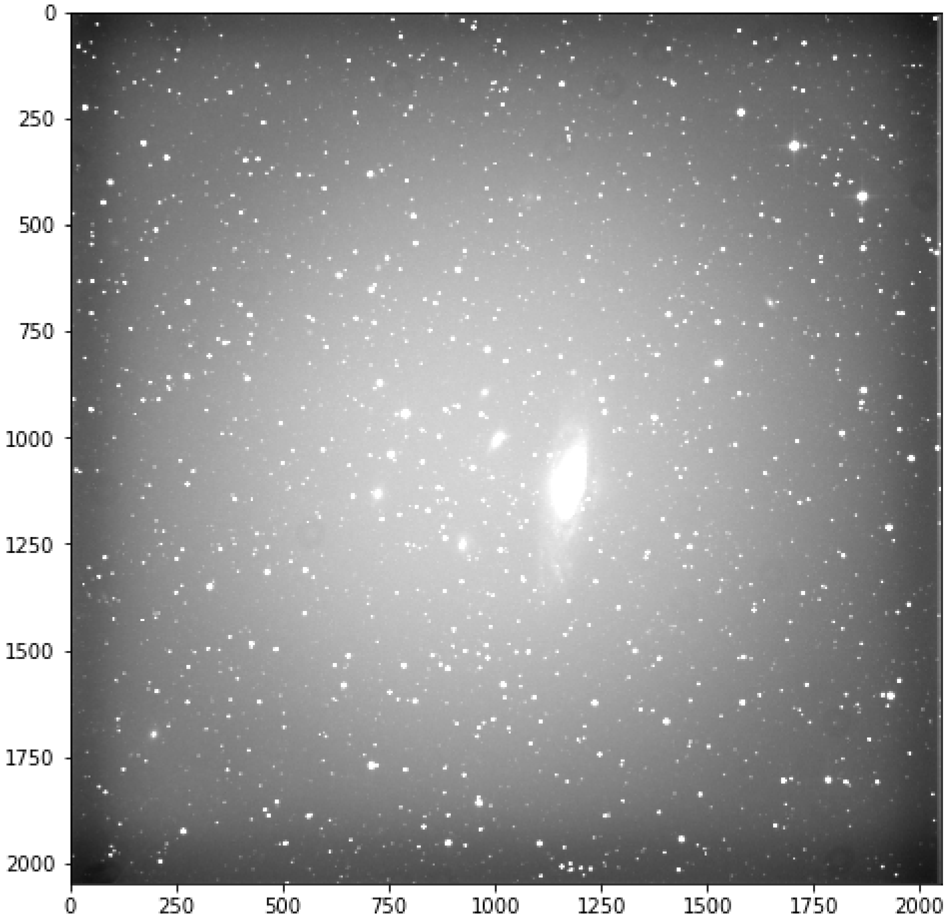
```
#open and display the dark field data
dark = fits.open('Dark-S001-R003-C003-B2.fts')
header = dark[0].header

data2 = dark[0].data
data2.shape
plt.figure(figsize=(8, 8))
```

```
plt.imshow(data2.squeeze(), vmax=np.percentile(data2, 99), cmap="gray")
plt.show()
```



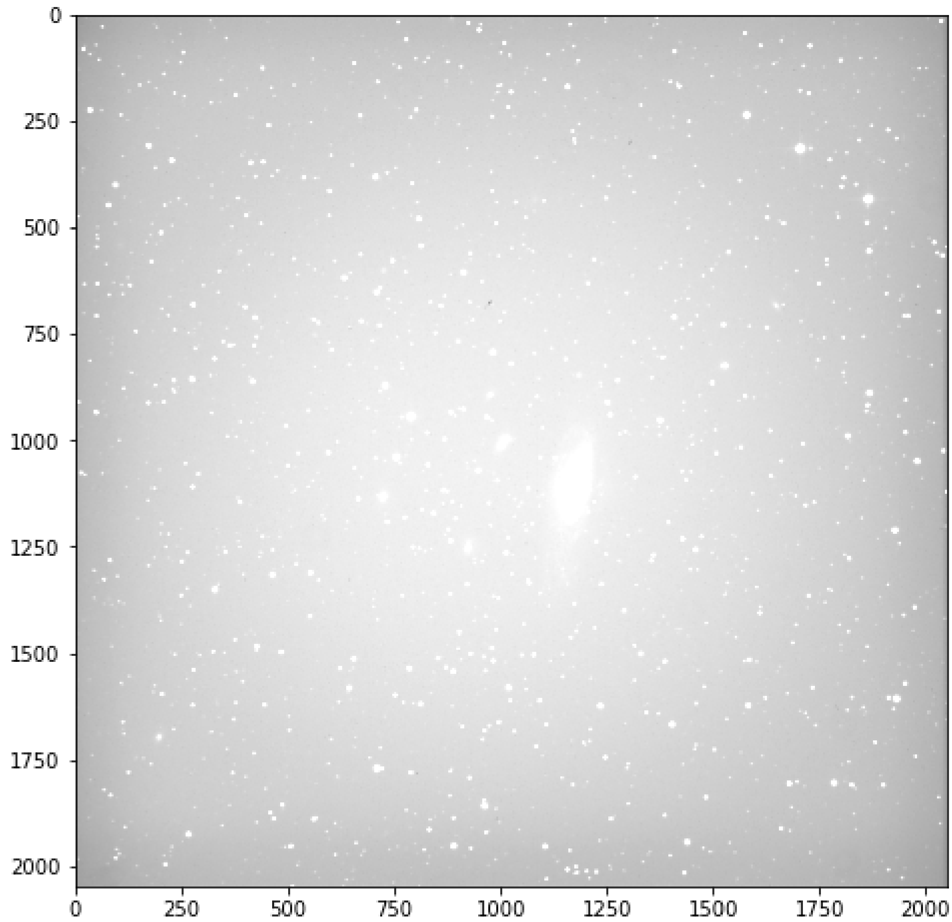
In [4]:

```
#open and display the raw NGC data

ngc = fits.open('NGC7331-S001-R001-C001-r.fts')
header = ngc[0].header

data3 = ngc[0].data
data3.shape
plt.figure(figsize=(8, 8))
plt.imshow(data3.squeeze(), vmax=np.percentile(data3, 99), cmap="gray")
plt.show()
```



In [5]:

```
#Apply data corrections

dark_corrected_image = data3 - data1
flat_image = dark_corrected_image/data2

flat_image.shape
plt.figure(figsize=(8, 8))
plt.imshow(flat_image.squeeze(), vmax=np.percentile(flat_image, 99), cmap="g
ray")
plt.show()
```



In [6]:
```
ngc[0].header
```

Out[6]:
```
SIMPLE  =                    T
BITPIX  =                   16 /8 unsigned int, 16 & 32 int, -32 & -64 rea
l
NAXIS   =                    2 /number of axes
NAXIS1  =                 2048 /fastest changing axis
NAXIS2  =                 2048 /next to fastest changing axis
BSCALE  =   1.0000000000000000 /physical = BZERO + BSCALE*array_value
BZERO   =   32768.000000000000 /physical = BZERO + BSCALE*array_value
DATE-OBS= '2011-10-13T03:18:59' / [ISO 8601] UTC date/time of exposure sta
rt
EXPTIME =   2.40000000000E+002 / [sec] Duration of exposure
EXPOSURE=   2.40000000000E+002 / [sec] Duration of exposure
SET-TEMP=  -20.000000000000000 /CCD temperature setpoint in C
CCD-TEMP=  -20.022502500000002 /CCD temperature at start of exposure in C
XPIXSZ  =   18.000000000000000 /Pixel Width in microns (after binning)
YPIXSZ  =   18.000000000000000 /Pixel Height in microns (after binning)
XBINNING=                    2 / Binning level along the X-axis
YBINNING=                    2 / Binning level along the Y-axis
XORGSUBF=                    0 /Subframe X position in binned pixels
YORGSUBF=                    0 /Subframe Y position in binned pixels
READOUTM= 'Monochrome' /        Readout mode of image
FILTER  = 'r       '          / Filter name
IMAGETYP= 'Light Frame' /       Type of image
JD      =   2455847.6381828706 /Julian Date at start of exposure
```

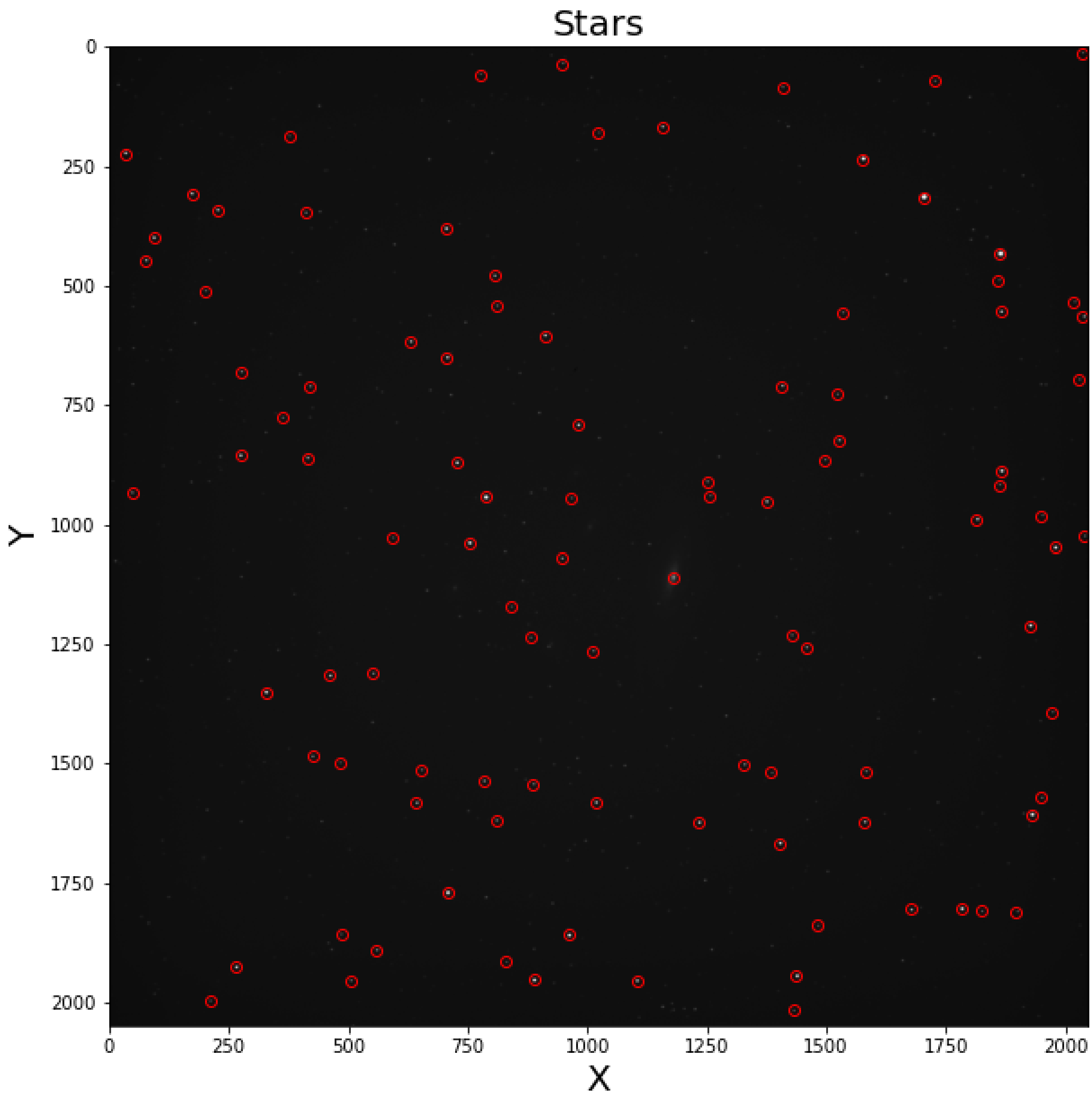
```
FOCALLEN=  0.00000000000000000 /Focal length of telescope in mm
APTDIA  =  0.00000000000000000 /Aperture diameter of telescope in mm
APTAREA =  0.00000000000000000 /Aperture area of telescope in mm^2
SWCREATE= 'MaxIm DL Version 5.15' /Name of software that created the image
SBSTDVER= 'SBFITSEXT Version 1.0' /Version of SBFITSEXT standard in effect
OBJECT  = 'NGC7331 '           / Target object name
TELESCOP= 'ACP->AstroPhysicsV2' / Telescope name
INSTRUME= 'Apogee USB/Net'     / Detector instrument name
OBSERVER= 'pearl   '           / Observer name
NOTES   = '        '
FLIPSTAT= '        '
CSTRETCH= 'Medium  ' /          Initial display stretch mode
CBLACK  =                 4758 /Initial display black level in ADUs
CWHITE  =                 6798 /Initial display white level in ADUs
PEDESTAL=                    0 /Correction to add for zero-based ADU
SWOWNER = 'Nicholas Law' /      Licensed owner of software
PIERSIDE= 'WEST    '
READMODE= 'Monochrome'
HISTORY File was processed by PinPoint 5.1.8 at 2011-10-13T03:23:09
DATE    = '13/10/11'           / [old format] UTC date of exposure start
TIME-OBS= '03:18:59'           / [old format] UTC time of exposure start
UT      = '03:18:59'           / [old format] UTC time of exposure start
TIMESYS = 'UTC     '           / Default time system
RADECSYS= 'FK5     '           / Equatorial coordinate system
AIRMASS =   1.01786463149E+000 / Airmass (multiple of zenithal airmass)
ST      = '21 46 26.71'        / Local apparent sidereal time of exp. star
t
LAT-OBS =   3.29027777778E+001 / [deg +N WGS84] Geodetic latitude
LONG-OBS=  -1.05529444444E+002 / [deg +E WGS84] Geodetic longitude
ALT-OBS =   2.28600000000E+003 / [metres] Altitude above mean sea level
OBSERVAT= 'Dunlap Institute Telescope' / Observatory name
RA      = '22 37 18.00'        / [hms J2000] Target right ascension
OBJCTRA = '22 37 18.00'        / [hms J2000] Target right ascension
DEC     = '+34 26 37.0'        / [dms +N J2000] Target declination
OBJCTDEC= '+34 26 37.0'        / [dms +N J2000] Target declination
CLRBAND = 'R       '           / [J-C std] Std. color band of image or C=C
olor
HISTORY File was processed by PinPoint 5.1.8 at 2011-10-13T03:23:12
FWHM    =   3.00714285374E+000 / [pixels] Mean Full-Width-Half-Max of imag
e star
ZMAG    =   2.13868141962E+001 / Mag zero point for 1 sec exposure
EQUINOX =               2000.0 / Equatorial coordinates are J2000
EPOCH   =               2000.0 / (incorrect but needed by old programs)
PA      =   3.59607090401E+002 / [deg, 0-360 CCW] Position angle of plate
CTYPE1  = 'RA---TAN'           / X-axis coordinate type
CRVAL1  =   3.39323757154E+002 / X-axis coordinate value
CRPIX1  =   1.02400000000E+003 / X-axis reference pixel
CDELT1  =  -3.02101567107E-004 / [deg/pixel] X-axis plate scale
CROTA1  =   3.92909599100E-001 / [deg] Roll angle wrt X-axis
CTYPE2  = 'DEC--TAN'           / Y-axis coordinate type
CRVAL2  =   3.44422583580E+001 / Y-axis coordinate value
CRPIX2  =   1.02400000000E+003 / Y-axis reference pixel
CDELT2  =  -3.02093176153E-004 / [deg/pixel] Y-Axis Plate scale
CROTA2  =   3.92909599100E-001 / [deg] Roll angle wrt Y-axis
CD1_1   =  -3.02094463788E-004 / Change in RA---TAN along X-Axis
CD1_2   =   2.07160770733E-006 / Change in RA---TAN along Y-Axis
CD2_1   =  -2.07166524840E-006 / Change in DEC--TAN along X-Axis
CD2_2   =  -3.02086073032E-004 / Change in DEC--TAN along Y-Axis
TR1_0   =   1.02400018028E+003 / [private] X-axis distortion coefficients
TR1_1   =   2.04799671349E+003
TR1_2   =   3.72920358941E-001
TR1_3   =  -1.10176001632E+000
TR1_4   =   6.14118869027E-001
TR1_5   =   1.33546162035E+000
TR1_6   =  -1.68132125691E+001
TR1_7   =  -5.28489033454E-001
```

```
TR1_8   =  -1.66768099978E+001
TR1_9   =  -8.70764421003E-001
TR1_10  =   6.74820735702E+000
TR1_11  =  -3.41773431836E-001
TR1_12  =  -1.97099962308E+000
TR1_13  =  -4.09332492302E+000
TR1_14  =  -6.97403954973E+000
TR2_0   =   1.02400000190E+003 / [private] Y-axis distortion coefficients
TR2_1   =   8.14160451750E-004
TR2_2   =   2.04800325275E+003
TR2_3   =  -1.30017295160E+000
TR2_4   =   2.02862478441E-001
TR2_5   =   1.42480374856E+000
TR2_6   =   3.95929171698E-001
TR2_7   =  -1.75225246797E+001
TR2_8   =  -1.85398173971E-002
TR2_9   =  -1.65353549125E+001
TR2_10  =   3.68000553150E+000
TR2_11  =   1.31069271342E-001
TR2_12  =   2.31093410921E+000
TR2_13  =  -1.19829063444E+000
TR2_14  =  -1.08828903572E+001
HISTORY WCS added by PinPoint 5.1.8 at 2011-10-13T03:23:12
HISTORY   Matched 239 stars from the Gray GSC-ACT Catalog
HISTORY   Average residual was 0.26 arc-seconds
PLTSOLVD=                    T / Plate has been solved by PinPoint
```

In [7]:

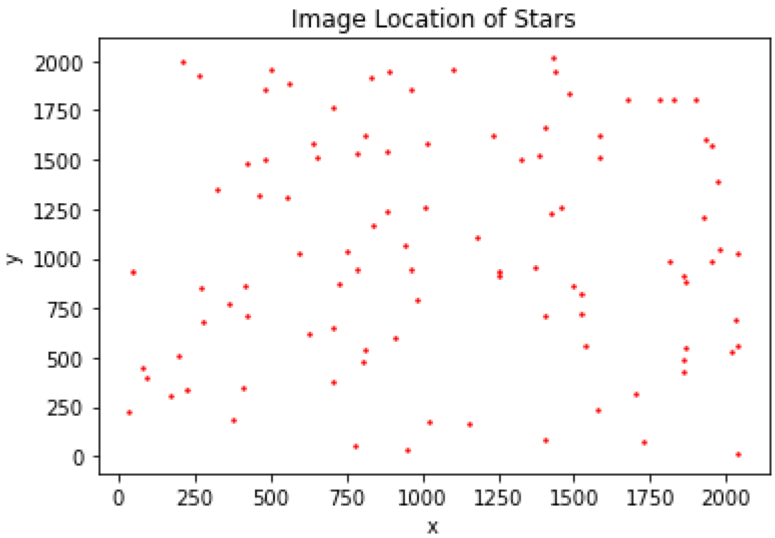
```
#function that finds centroid of image
def centroid(image):

    xvals = np.arange(image.shape[1])
    yvals = np.arange(image.shape[0])
    centroid_x = np.sum(xvals*image)/np.sum(image)
    centroid_y = np.sum(yvals*image.T)/np.sum(image)

    return np.array([centroid_y, centroid_x])

background = np.median(flat_image)

star_locations = []
BOX_SIZE = 10
im = flat_image.copy()

#find stars
for i in range(100):

    #find brightest pixel in the image
    brightest_pixel = np.array([np.where(im == np.max(im))[0][0], np.where(i
m == np.max(im))[1][0]])

    #create a "box" around that pixel
    box = im[brightest_pixel[0]-BOX_SIZE:brightest_pixel[0]+BOX_SIZE+1,
             brightest_pixel[1]-BOX_SIZE:brightest_pixel[1]+BOX_SIZE+1]
    box -= background

    #Get the centroid of that box
    c = centroid(box)

    #Map the centroid of the box back onto the orginal image
    c -= np.array([box.shape[1]//2,box.shape[0]//2])
    c += brightest_pixel

    star_locations.append(c)

    #Remove that star from the image so it isn't the brightest pixel anymore
```

```
    box[:,:] = background

#store and plot star locations
star_locations = np.array(star_locations)

plt.figure(figsize=(10,10))
plt.plot(star_locations[:,1], star_locations[:,0], 'o', color = 'r', alpha =
1, fillstyle = "none")
plt.imshow(flat_image, cmap = 'gray')
plt.title("Stars", size = 20)
plt.xlabel("X", size = 20)
plt.ylabel("Y", size = 20)
plt.show()
```



In [8]:
```
#plot star locations off of picture
plt.scatter(star_locations[:,1], star_locations[:,0], color='r', s =2)

plt.title("Image Location of Stars")
plt.xlabel("x")
plt.ylabel("y")
```

Out[8]: `Text(0, 0.5, 'y')`



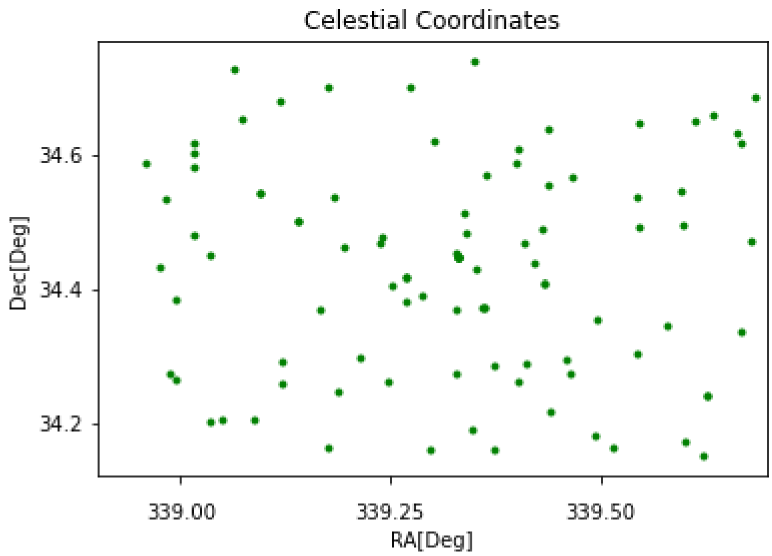
In [9]:

```
#Get the stars from the USNO catalog
def usno(radeg,decdeg,fovam):
    str1 = 'http://webviz.u-strasbg.fr/viz-bin/asu-tsv/?-source=USNO-B1'
    str2 = '&-c.ra={:4.6f}&-c.dec={:4.6f}&-c.bm={:4.7f}/{:4.7f}&-out.max=unl
imited'.format(radeg,decdeg,fovam,fovam)
    sr = str1+str2

    f = urllib.request.urlopen(sr)
    s = f.read()
    f.close()

    namecol, RAcol, DECcol, rband = 0, 1, 2, 12
    null1, null2 ='     ',''

    s1 = s.splitlines()
    s1 = s1[45:-1] # get rid of header
    name = np.array([])
    rad = np.array([]) # RA in degrees
    ded = np.array([]) # DEC in degrees
    rmag = np.array([]) # rmage

    for k in s1:
        kw = k.decode().split('\t')
        if kw[0] != '':
            name = np.append(name,kw[namecol])
            rad = np.append(rad,float(kw[RAcol]))
            ded = np.append(ded,float(kw[DECcol]))
            # deal with case where no mag is reported
            if (kw[rband] != null2) and (kw[rband] != null1):
                rmag = np.append(rmag, float(kw[rband]))
            else:
                rmag = np.append(rmag,np.nan)

    return name,rad,ded,rmag

s1 = ngc
#readpositionfromtheFITSfileandconvertRA/DECtodegrees
ras= s1[0].header['ra']
des = s1[0].header['dec']
```

```
radeg = 15*(float(ras[0:2])+float(ras[3:5])/60.+ float(ras[6:])/3600.)
dsgn = np.sign(float(des[0:3]))
dedeg= float(des[0:3])+dsgn*float(des[4:6])/60.+ dsgn*float(des[7:])/3600.
fovam=36.4
type(fovam)
#sizeofsquaresearchfieldinarcmin
name,rad,ded,rmag= usno(radeg,dedeg,fovam)
w=np.where(rmag < 13.)[0]
#selectonlybrightstarsr<15mag.
plt.plot(rad[w],ded[w],'g.')
plt.locator_params(axis='x',nbins=4)
plt.locator_params(axis='y',nbins=4)
plt.tick_params('x',pad=10)
plt.xlabel('RA[Deg]')
plt.ylabel('Dec[Deg]')
plt.title("Celestial Coordinates")
plt.ticklabel_format(useOffset=False)
#plt.axis('scaled')
plt.xlim([338.9,339.7])
```

```
<ipython-input-9-8913fb3dfb1a>:45: RuntimeWarning: invalid value encounter
ed in less
  w=np.where(rmag < 13.)[0]
```

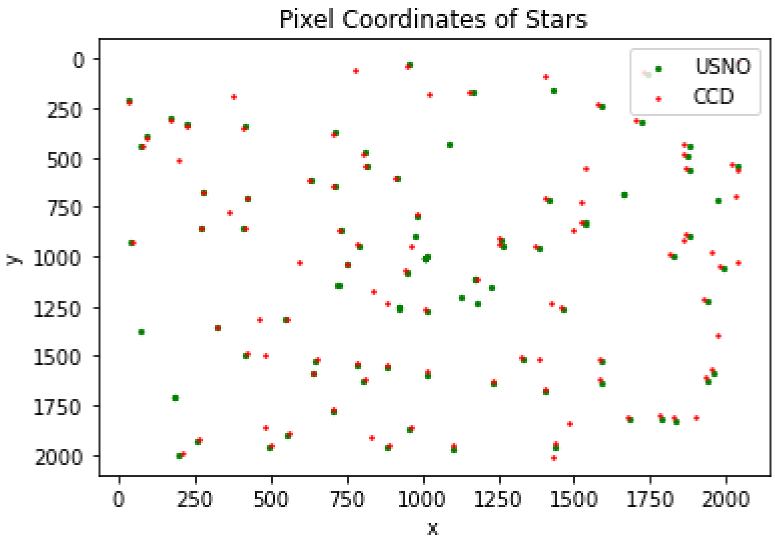
Out[9]: (338.9, 339.7)



In [10]:
```
#function that converts from celestial coordinates to standard coordinates
def standardcoords(rad, dec, a0, d0):
    ra = rad*np.pi/180
    dc = dec*np.pi/180

    a0 = a0*np.pi/180
    d0 = d0*np.pi/180

    cos_ra = np.cos(ra)
    sin_ra = np.sin(ra)
    cos_dc = np.cos(dc)
    sin_dc = np.sin(dc)

    X = -(cos_dc*np.sin(ra - a0))/(np.cos(d0)*cos_dc*(np.cos(ra - a0))+(sin_
dc*np.sin(d0)))
    Y = -(np.sin(d0)*cos_dc*np.cos(ra-a0)-((np.cos(d0)*sin_dc)))/(np.cos(d0)
*cos_dc*np.cos(ra-a0)+(sin_dc*np.sin(d0)))

    return X, Y

X, Y = standardcoords(rad[w],ded[w], radeg, dedeg)
```

In [11]:
```python
#convert from standard to pixel coordinates

f = 3454
p = 0.009*2
x0 = 1024
y0 = 1024
theta = np.pi

x=((f/p)*(X*np.cos(theta)-Y*np.sin(theta)))+x0
y = ((f/p)*(X*np.sin(theta)+Y*np.cos(theta)))+y0

x_new = list(map(lambda xi: 2048-xi, x))
x_new = np.array(x_new)

#plot the raw data over the usno data
plt.scatter(x_new,y, s = 5, color = "green", label = "USNO")
plt.title("Pixel Coordinates of Stars")
plt.xlabel("x")
plt.ylabel("y")
plt.ylim(2100, -100)

plt.scatter(star_locations[:,1], star_locations[:,0], color='r', s =2, label
= "CCD")
plt.legend(loc="best")
```

Out[11]: <matplotlib.legend.Legend at 0x1e1dc734310>



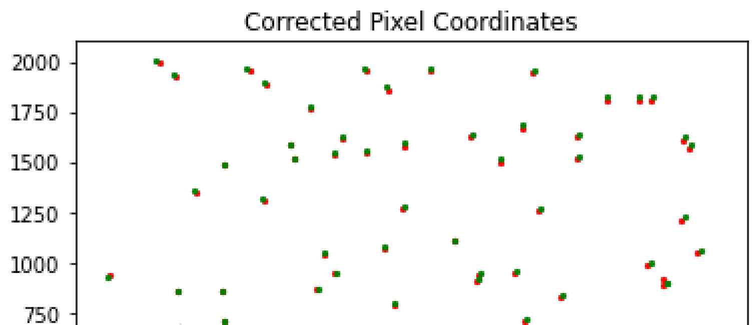
In [49]:
```python
#define the affine transformation function to solve for the proper positions
of the stars
def leastsquares(X,Y, data):
    B = []
    ax = []
    ay = []

    for i in range(X.size):
        B.append([(f/p)*X[i], (f/p)*Y[i], 1])
        ax.append(data[0][i])
        ay.append(data[1][i])

    B = np.array(B)
    ax = np.array(ax)
    ay = np.array(ay)

    Bt = B.T
    Binv = np.linalg.inv(np.dot(Bt,B))

    c_x = np.dot(Binv,np.dot(Bt, ax))
```

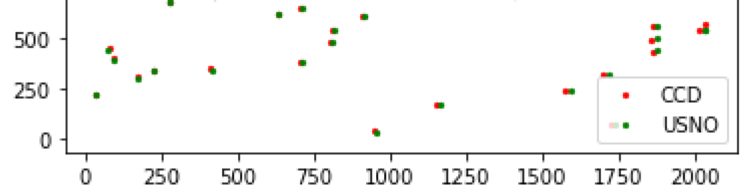
```
    c_y = np.dot(Binv,np.dot(Bt, ay))

    a_11 = c_x[0]
    a_12 = c_x[1]
    x_0 = c_x[2]

    a_21 = c_y[0]
    a_22 = c_y[1]
    y_0 = c_y[2]

    chisq_x = np.dot((ax-np.dot(B, c_x)).T, (ax-np.dot(B, c_x)))
    chisq_y = np.dot((ay-np.dot(B, c_y)).T, (ay-np.dot(B, c_y)))

    redchi_x = chisq_x/(ax.size-3)
    redchi_y = chisq_y/(ay.size-3)

    constants = np.array([a_11, a_12, x_0, a_21, a_22, y_0, redchi_x, redchi
_y])

    return constants
```

In [66]:

```
#define a function that removes outlier stars
def stars(x1,y1,x2,y2,threshold):
    x1new=[]
    y1new=[]
    x2new=[]
    y2new=[]

    for i in range (x1.size):
        for j in range (x2.size):
            if (abs(x1[i]-x2[j])<threshold) and (abs(y1[i]-y2[j])<threshold
):
                x1new.append(x1[i])
                x2new.append(x2[j])
                y1new.append(y1[i])
                y2new.append(y2[j])

                break

        return np.array([x1new, x2new, y1new, y2new])

new_stars = stars(star_locations[:,1],star_locations[:,0],x_new, y, 25)

x_ccd = new_stars[0]
y_ccd = new_stars[2]
x_usno = new_stars[1]
y_usno = new_stars[3]

#plot the raw data over the usno data with outliers removed
plt.scatter(x_ccd, y_ccd, color='r', s =5, label = "CCD")
plt.scatter(x_usno,y_usno, s = 5, color = "green", label = "USNO")
plt.legend(loc="best")
plt.title("Corrected Pixel Coordinates")
```

Out[66]: Text(0.5, 1.0, 'Corrected Pixel Coordinates')



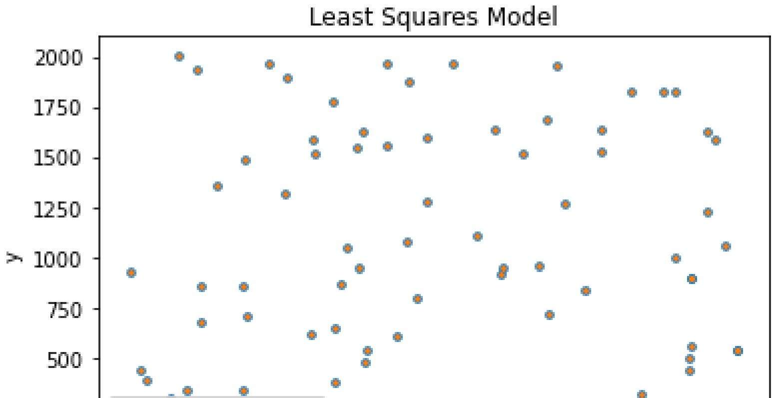


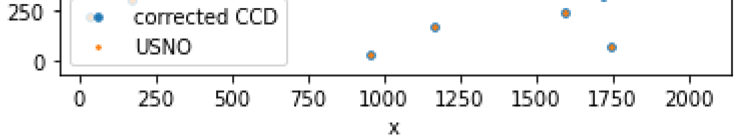
In [69]:
```
#function converts to pixel coords from standard coords
def pixel(x,y):
    p = 0.018
    f = 3454
    X = np.array(x-1024)*(p/f)
    Y = np.array(y-1024)*(p/f)
    return(X,Y)

#find plate constants
pixelxy = pixel(x_usno, y_usno)

new_data = []
new_data.append(x_ccd)
new_data.append(y_ccd)

plates = leastsquares(pixelxy[0], pixelxy[1], new_data)
plates
```

Out[69]:
```
array([ 9.90968666e-01,  6.26058014e-03,  1.01997310e+03, -4.66607055e-03,
        9.89248758e-01,  1.01942775e+03,  6.18704091e+00,  2.39578252e+0
1])
```

In [70]:
```
#use curve fit to find the correct star locations
def model_function(x,a,b,c):
    return (a*x + b*x + c)

p_optx, p_covx = curve_fit(model_function, pixelxy[0], x_usno, p0 = [0,0,150
0])
p_opty, p_covy = curve_fit(model_function, pixelxy[1], y_usno, p0 = [0,0,150
0])

xs = model_function(pixelxy[0], p_optx[0], p_optx[1], p_optx[2])
ys = model_function(pixelxy[1], p_opty[0], p_opty[1], p_opty[2])

xs_new = list(map(lambda xi: 2048-xi, xs))
xs_new = np.array(xs_new)

#plot corrected star locations over usno data
plt.scatter(xs,ys, s = 13, label = "corrected CCD")
plt.scatter(x_usno, y_usno, s = 3, label = "USNO")
plt.title("Least Squares Model")
plt.xlabel("x")
plt.ylabel("y")
plt.legend(loc="best")
```

Out[70]:
```
<matplotlib.legend.Legend at 0x1e1e3473280>
```




250
0
corrected CCD
USNO
0
250
500
750
1000
1250
1500
1750
2000
x

```
In [ ]:
```