Observation Planning Part 3

1. Where is my target?
2. When can I observe my target?
3. How do I know when I've found my target (make a finder chart)?
4. How long do I need to observe?
5. **How will I calibrate my data?**

Index

How will I calibrate my data?

To make a photometric measurement, we need to know how efficient our instrument really is. In theory, every photon collected in our telescope goes to the CCD and is converted to electrons, but in practice several factors prevent this.

Q1

What are the factors that limit our ability to collect the light? i.e., what prevents us from counting every photon? Consider instrumental and atmospheric effects.

Rayleigh scattering, air glow, light pollution, zodical light, dark current, mie scattering, what filters you're using, exposure time, cloud cover, the time of day, air mass, telescope apature, CCD, hot pixles

Calibration: Photometric Standards

To calibrate our image, we can use reference stars with known brightness to infer how much light is lost on the way to our image.

Landolt photometric standard stars are the best standards to use because they've been selected to be non-variable and have been carefully calibrated.

However, the AAVSO (American Association of Variable Star Observers) has provided some really nice tools for obtaining standard star locations:

https://www.aavso.org/apps/vsd/stdfields (https://www.aavso.org/apps/vsd/stdfields)

We can also retrieve catalogs from Vizier, the Centre de Données astronomiques de Strasbourg services for catalogs.

For simplicity, we'll adopt this latter approach.

In [1]:
```
from astroquery.vizier import Vizier
import numpy as np
```

You can use the `Vizier.get_catalogs` method in astroquery to grab the data from a catalog whose precise name you already know

In [2]:
```
Vizier.ROW_LIMIT = 10000
landolt_catalog = Vizier.get_catalogs('J/AJ/146/131/standards')
```

Vizier actually returns a list of tables:

In [3]:
```
landolt_catalog
```

Out[3]:
```
TableList with 1 tables:
	'0:J/AJ/146/131/standards' with 16 column(s) and 349 row(s)
```

But we want to look at only the first of these:

In [4]:
```
landolt_tbl = landolt_catalog[0]
landolt_tbl.show_in_notebook(display_length=5) # you can change the display_lengt
```

Out[4]:

Table length=349

Show 5 entries Search:

idx	Name	N	__Vmag_	__B-V_	__U-B_	__V-R_	__R-I_	__V-I_	No	Nn	RAJ2000	DEJ2000	
			mag	mag	mag	mag	mag	mag			"h:m:s"	"d:m:s"	
0	GD 2B		13.279	0.588	-0.001	0.350	0.349	0.697	19	10	00 07 25.484	+33 19 00.17	J000725
1	GD 2A		14.853	0.912	0.684	0.530	0.462	1.002	17	8	00 07 26.174	+33 18 19.18	J000726
2	GD 2	N	13.802	-0.295	-1.192	-0.142	-0.171	-0.313	26	13	00 07 32.261	+33 17 27.62	J000732
3	GD 2C		13.314	0.619	0.081	0.360	0.357	0.718	17	8	00 07 32.355	+33 20 14.69	J000732
4	GD 2E		15.188	0.575	0.076	0.339	0.323	0.652	7	4	00 07 36.675	+33 17 41.73	J000736

Showing 1 to 5 of 349 entries First Previous 1 2 3 4 5 …70 Next Last

The table above contains two columns that specify the RA and Dec coordinates, `RAJ2000` and `DEJ2000`.

We'd like to be able to plot these and use them in some calculations below, so we'll turn them into `astropy.coordinates` objects. Note that we need to do this because most software tools don't know how to read sexagesimal labels as numbers. Anyway, the conversion process is easy:

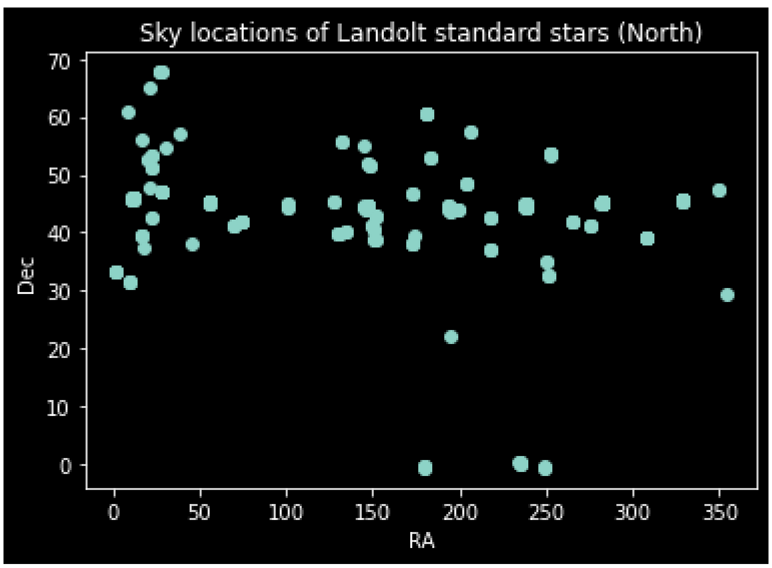
In [5]:
```
from astropy import coordinates, units as u
```

In [6]:
```
landolt_coords = coordinates.SkyCoord(landolt_tbl['RAJ2000'], landolt_tbl['DEJ20(
```

We can visualize the *sky coverage* of these standard stars in RA and Dec:

In [7]:
```
import pylab as plt
plt.style.use('dark_background') # if you're using a light background, you shoulc
```

In [8]:
```
_=plt.plot(landolt_coords.ra, landolt_coords.dec, 'o')
_=plt.xlabel("RA")
_=plt.ylabel("Dec")
_=plt.title("Sky locations of Landolt standard stars (North)")
```

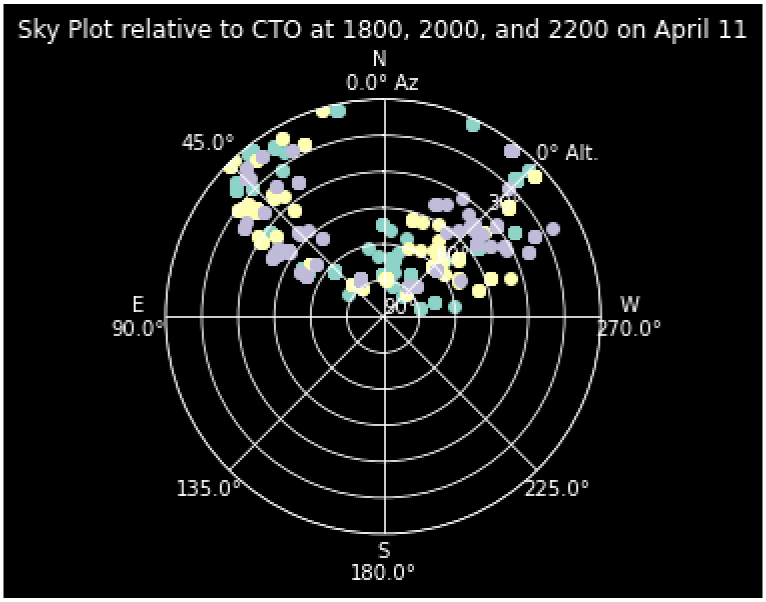


What if we want to figure out where these are in altitute and azimuth (alt/az) relative to us, the observer? The Sky Plot feature from astroplan is good for that! Of course, we need to specify the observatory first:

In [9]:
```
from astroplan import Observer
from astropy import units as u # shortcut
CTO = Observer(location=coordinates.EarthLocation(lat=29.643018, lon=-82.349004*u
               timezone='EST',
               name='University of Florida Campus Teaching Observatory',
              )
```

In [10]:
```
from astroplan.plots import plot_sky
from astropy.time import Time
```

In [11]:
```
_=plot_sky(target=landolt_coords, observer=CTO, time=Time('2021-04-11 18:00:00'))
_=plot_sky(target=landolt_coords, observer=CTO, time=Time('2021-04-11 20:00:00'))
_=plot_sky(target=landolt_coords, observer=CTO, time=Time('2021-04-11 22:00:00'))
_=plt.title("Sky Plot relative to CTO at 1800, 2000, and 2200 on April 11")
```



There are a lot of targets that go straight overhead! There aren't so many in the South, but that's because we picked a standard star catalog that is meant for the north; if we wanted stars further south, we could use ["II/183A/table2" (http://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=II/183A/table2)](http://vizier.u-strasbg.fr/viz-bin/VizieR-3?-source=II/183A/table2) instead.

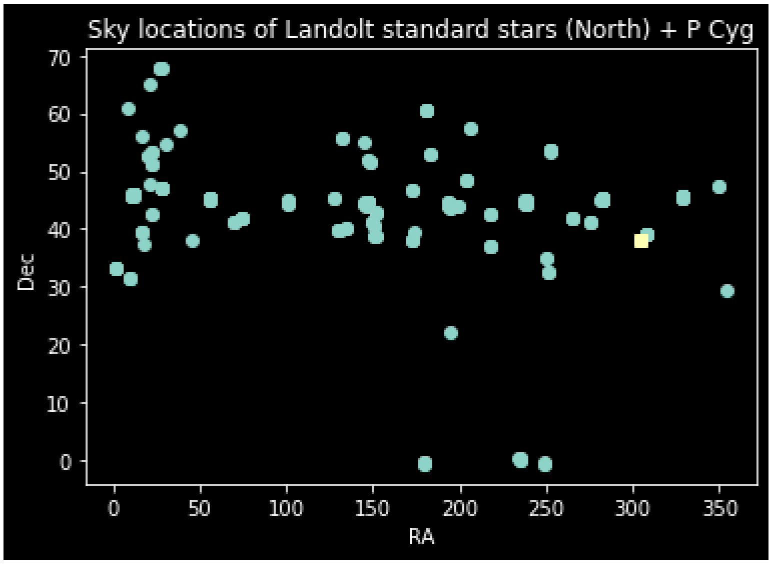
To find these table IDs, you can just search on Vizier for the name "Landolt standards".

Let's say we want to observe P Cygni. How do we find the closest standard(s)?

First, we can simply overplot it on our RA/Dec plot

In [12]:
```
pcyg_coord = coordinates.SkyCoord.from_name('P Cygni')
```

In [13]:
```
_=plt.plot(landolt_coords.ra, landolt_coords.dec, 'o')
_=plt.plot(pcyg_coord.ra, pcyg_coord.dec, 's')
_=plt.xlabel("RA")
_=plt.ylabel("Dec")
_=plt.title("Sky locations of Landolt standard stars (North) + P Cyg")
```



Find closest calibrator

[Index](#)

How can we figure out which calibrator source from the catalog is the closest one?

We can calculate the distance between P Cygni and each of the stars in the Landolt catalog. That distance is the *angular separation* on the sphere:

$$\theta = \cos^{-1}[\sin(\delta_1)\sin(\delta_2) + \cos(\delta_1)\cos(\delta_2)\cos(\alpha_1 - \alpha_2)]$$

In practice, you don't want to do this yourself, as there can be numerical issues when calculating these values near the poles (see the article on [Great Circle distances (https://en.wikipedia.org/wiki/Great-circle_distance)](https://en.wikipedia.org/wiki/Great-circle_distance)). Thankfully, astropy's coordinates provide a `separation` tool to calculate this for us.

In [14]:
```
distances_from_pcyg_to_standards = pcyg_coord.separation(landolt_coords)
```

We can then find which of these is closest by taking the minimum:

In [15]:
```
np.min(distances_from_pcyg_to_standards)
```

Out[15]: $2^\circ 37' 43.74122076''$

The closest standard star is about 2 degrees away. Which star is it, though? We can use `np.argmin` to obtain the index corresponding to that minimum value.

In [16]:
```
index = np.argmin(distances_from_pcyg_to_standards)
index, landolt_coords[index]
```

Out[16]:
```
(305,
 <SkyCoord (ICRS): (ra, dec) in deg
     (307.40492917, 39.28821667)>)
```

So now we have its location. Can we find out more about the star, like its name and brightness?

Since `landolt_coords` has the same length and order as the `landolt_table` above, yes! We can use the same index:

In [17]:
```
landolt_tbl[index]
```

Out[17]: *Row index=305*

Name	N	__Vmag_	__B-V_	__U-B_	__V-R_	__R-I_	__V-I_	No	Nn	RAJ2000	DEJ2000	
		mag	mag	mag	mag	mag	mag			"h:m:s"	"d:m:s"	
str12	str1	float32	float32	float32	float32	float32	float32	int16	int16	str12	str12	
GD 391F		12.500	0.544	0.048	0.329	0.331	0.660	10	5	20 29 37.183	+39 17 17.58	J

Great! We've found our standard star, and we know it has a visual magnitude $V_{mag} = 12.5$!

In [18]:
```
print("V Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'])
```

```
V Magnitude of standard star:  12.5
```

We can also determine its magnitude in the B and R bands using the colors in the table. Note that the titles of the columns tell you what they contain: except for the V-band, the columns show *colors*, i.e., delta-magnitudes. The `B-V` column is the `B-V` color. If you want to obtain the B color, you just do `B-V + V = B`.

In [19]:
```
print("B Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] + landolt_tb
print("R Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] - landolt_tb
```

```
B Magnitude of standard star:  13.044
R Magnitude of standard star:  12.171
```

Now that we've selected this star, we need to go back through and do the same planning exercises for it as for the targets:

1. Where is my calibrator? (we answered this a few cells ago)
2. When can I observe my calibrator? (the same time as my target!)
3. How do I know when I've found my calibrator? (make a finder chart)
4. How long do I need to observe?

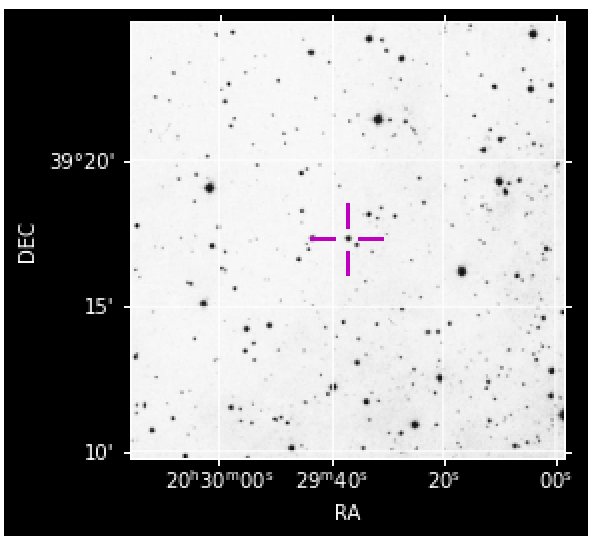
Finder Chart for Calibrator

Index

Since we've done (1) and (2), let's do (3) and (4):

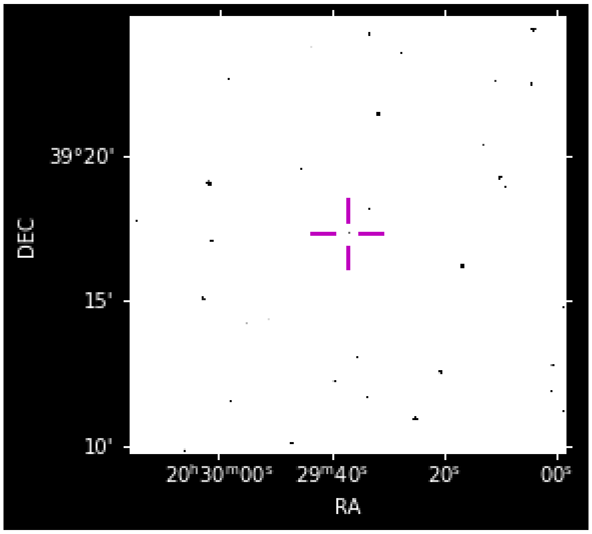
Recall how you made finder charts from Observation Planning Exercise 1:

In [20]:
```
from astroplan.plots import plot_finder_image
ax, hdu = plot_finder_image(landolt_coords[index], survey='DSS', fov_radius=15*u.
```



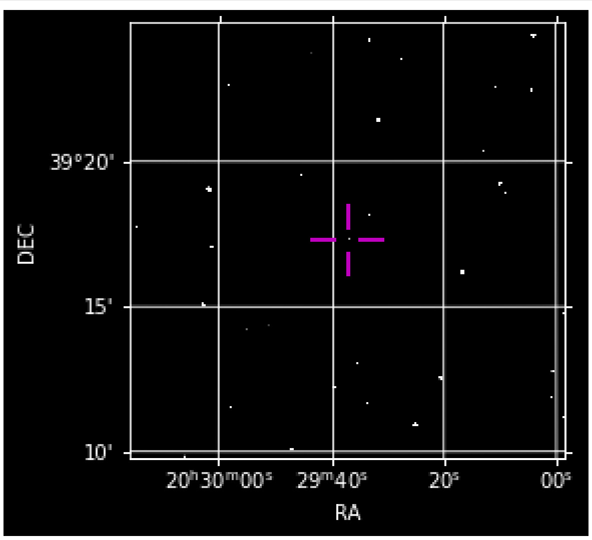
If you've been on an observing run, you know that these finder charts are hard to use, and sometimes it's better to show them more saturated. You can do that by specifying `style_kwargs`:

In [21]:
```
ax, hdu = plot_finder_image(landolt_coords[index],
                            survey='DSS',
                            fov_radius=15*u.arcmin, grid=False, reticle=True,
                            style_kwargs={'vmin':18000, 'vmax':18100})
```



You can also change the colorscale if you want something that looks a little more like the night sky:

In [22]:
```
ax, hdu = plot_finder_image(landolt_coords[index], survey='DSS',
                            fov_radius=15*u.arcmin, grid=False,
                            reticle=True, style_kwargs={'vmin':18000, 'vmax':181(
```



Exposure time for calibrator

[Index](#)

How long do we need to observe? We use the same technique as Observation Planning Exercise 2.

The zero points are:

- [3600 Jy for V-band (http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.V&&mode=browse&gname=Generic&gname2=Bessell#filter)](http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.V&&mode=browse&gname=Generic&gname2=Bessell#filter)
- [4000 Jy for B-band (http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.B&&mode=browse&gname=Generic&gname2=Bessell#filter)](http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.B&&mode=browse&gname=Generic&gname2=Bessell#filter)
- [2400 Jy for I-band (http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.I&&mode=browse&gname=Generic&gname2=Bessell#filter)](http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.I&&mode=browse&gname=Generic&gname2=Bessell#filter)

In [23]:
```
standard_star_vmag = landolt_tbl[index]['__Vmag_']
vmag_zeropoint = 3600*u.Jy
snu_standard = vmag_zeropoint * (10**(-standard_star_vmag/2.5))
standard_star_vmag, snu_standard
```

Out[23]:
```
(12.5, <Quantity 0.036 Jy>)
```

Using the telescope area and the filter properties, we now determine how much energy we receive from the star (which we calculated in the previous notebook):

- The V filter has a central wavelength of 5504 Angstroms
- The V filter has a width of "about" 1000 Angstroms (for [this filter (http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.V&&mode=browse&gname=Generic&gname2=Bessell#filter)](http://svo2.cab.inta-csic.es/theory/fps/index.php?id=Generic/Bessell.V&&mode=browse&gname=Generic&gname2=Bessell#filter), the width is 893 Angstroms, but we'll stick with the order-of-magnitude approximation for now)
- We are calculating the area of a 14-inch telescope

In [24]:
```
v_filt_wav = 5504*u.AA
v_filt_freq = (v_filt_wav).to(u.Hz, u.spectral())
v_filter_width = 1000*u.AA
A_CTO = (np.pi*(14/2 * u.imperial.inch)**2).to(u.cm**2)
standard_ergs_per_s = snu_standard * A_CTO * v_filt_freq*(v_filter_width/v_filt_w
standard_ergs_per_s.to(u.erg/u.s)
```

Out[24]: $3.5381785 \times 10^{-8} \ \frac{\mathrm{erg}}{\mathrm{s}}$

In [25]:
```
from astropy import constants
```

As before, we want to know the *number of photons* we will receive per second, so we convert the above energy to a photon rate.

In [26]:
```
standard_phot_per_s = (standard_ergs_per_s / (constants.h * v_filt_freq)).decompo
standard_phot_per_s
```

Out[26]: $9803.5063 \ \frac{1}{\mathrm{s}}$

Then, we want to account for the inefficiences in our telescope: the average filter efficiency, the CCD's quantum efficiency, and the loss from atmospheric absorption (noting, of course, that the atmospheric loss is calculated for zenith, so it could be worse than this!)

In [27]:
```
filter_efficiency = 0.75
quantum_efficiency = 0.7
atmosphere_loss = 0.2
received_fraction = filter_efficiency * quantum_efficiency * (1-atmosphere_loss)
received_fraction
```

Out[27]: 0.41999999999999993

As before, we need to determine how much our signal will be spread out. We use the same $\sigma = 2$ " PSF and 0.5 " pixel scale

In [28]:
```
psf_area = 2 * np.pi * (2*u.arcsec)**2
psf_area
```

Out[28]: $25.132741 \ \mathrm{arcsec^2}$

In [29]:
```
pixel_scale = 0.5*u.arcsec/u.pixel
psf_area_pixels = psf_area * pixel_scale**-2
psf_area_pixels
```

Out[29]: $100.53096 \ \mathrm{pix^2}$

In [30]:
```
count_rate_per_pixel = standard_phot_per_s / psf_area_pixels * received_fraction
count_rate_per_pixel
```

Out[30]: $40.957258 \ \mathrm{\frac{1}{s\,pix^2}}$

We want our signal-to-noise ratio to be at least as good as our target, ideally better, since we will be comparing these measurements (taking their difference) and therefore their noise will add in quadrature again. We can set a target SNR = 100 as before

Recall that the readnoise on the sum is the square root of the sum of the individual pixel read noise:

$$\sigma^2_{RN,sum} = \Sigma_i \sigma^2_{RN,i} = N\sigma^2_{RN}$$
$$\sigma_{RN,sum} = \sqrt{N} \cdot \sigma_{RN}$$

In [31]:
```
readnoise_per_pix = 10*u.adu/u.pix
readnoise_sum = psf_area_pixels**0.5 * readnoise_per_pix
readnoise_sum
```

Out[31]: $100.26513 \ \mathrm{adu}$

Recall the equation (from Observation Planning Part 2) for the target signal as a function of SNR:

$$S = SNR^2 \pm \frac{\sqrt{SNR^4 + 4\sigma_{RN}^2 SNR^2}}{2}$$

In [32]:
```
SNR = 100
target_signal = SNR**2 + (SNR**4 + 4*readnoise_sum.value**2*SNR**2)**0.5 / 2
target_signal
```

Out[32]: 21204.060197753017

In [33]:
```
integration_time_including_readnoise = target_signal / standard_phot_per_s / rece
integration_time_including_readnoise
```

Out[33]: 5.1497756 s

Exercise

Index

Repeat this exercise below for:

- (1) the B and R filters. How long do you need to expose the standard star in each of those filters to get to SNR = 100?
- (2) A standard star near your target. Use one of the targets we observed (e.g., M13 or M57).

Note that for most of this exercise, you don't need to re-calculate much of the above. Think about which items you do need to re-calculate. Answer these questions before you do the exercise.

Do you need to recalculate or re-enter:

- The star's magnitude? Yes
- The star's spectral flux density? Flux density? Yes
- The star's count rate? yes
- The PSF area? no
- The total readnoise? no
- The telescope area? no
- The filter center? yes
- The filter width? yes
- The goal SNR? no
- The target signal for the goal SNR? no
- The goal integration time? yes
- The star location? no

Answer each of the above with a "Yes" or "No". The first is answered "Yes" for you - we're looking at a different filter for part (1) and a different target for part (2), so you definitely need to have a different value for the magnitude.

Then, in new cells below here, complete the exercise:

In [34]:
```
print("B Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] + landolt_tb
print("R Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] - landolt_tb
```

```
B Magnitude of standard star:  13.044
R Magnitude of standard star:  12.171
```

B

In [35]:
```
standard_star_bmag = landolt_tbl[index]['__Vmag_'] + landolt_tbl[index]['__B-V_']
bmag_zeropoint = 4000*u.Jy
snu_standard_b = bmag_zeropoint * (10**(-standard_star_bmag/2.5))
standard_star_bmag, snu_standard_b
```

Out[35]:
```
(13.044, <Quantity 0.02423595 Jy>)
```

In [36]:
```
## central wavelength b 4400
## b filter width 1000
b_filt_wav = 4400*u.AA
b_filt_freq = (b_filt_wav).to(u.Hz, u.spectral())
b_filter_width = 1000*u.AA
A_CTO = (np.pi*(14/2 * u.imperial.inch)**2).to(u.cm**2)
standard_ergs_per_s_b = snu_standard_b * A_CTO * b_filt_freq*(b_filter_width/b_fi
standard_ergs_per_s_b.to(u.erg/u.s)
```

Out[36]: $3.7272529 \times 10^{-8}\ \frac{\mathrm{erg}}{\mathrm{s}}$

In [37]:
```
standard_phot_per_s_b = (standard_ergs_per_s_b / (constants.h * b_filt_freq)).dec
standard_phot_per_s_b
```

Out[37]: $8255.9072\ \frac{1}{\mathrm{s}}$

In [38]:
```
count_rate_per_pixel_b = standard_phot_per_s_b / psf_area_pixels * received_fract
count_rate_per_pixel_b
```

Out[38]: $34.491672\ \frac{1}{\mathrm{s\,pix^2}}$

In [39]:
```
integration_time_including_readnoise_b = target_signal / standard_phot_per_s_b /
integration_time_including_readnoise_b
```

Out[39]: $6.1151193\ \mathrm{s}$

R

In [40]:
```
standard_star_rmag = landolt_tbl[index]['__Vmag_'] - landolt_tbl[index]['__V-R_']
rmag_zeropoint = 3000*u.Jy
snu_standard_r = rmag_zeropoint * (10**(-standard_star_rmag/2.5))
standard_star_rmag, snu_standard_r
```

Out[40]: (12.171, <Quantity 0.04061827 Jy>)

In [41]:
```
r_filt_wav = 6400*u.AA
r_filt_freq = (r_filt_wav).to(u.Hz, u.spectral())
r_filter_width = 1000*u.AA
A_CTO = (np.pi*(14/2 * u.imperial.inch)**2).to(u.cm**2)
standard_ergs_per_s_r = snu_standard_r * A_CTO * r_filt_freq*(r_filter_width/r_fi
standard_ergs_per_s_r.to(u.erg/u.s)
```

Out[41]: $2.9525386 \times 10^{-8} \ \frac{\mathrm{erg}}{\mathrm{s}}$

In [42]:
```
standard_phot_per_s_r = (standard_ergs_per_s_r / (constants.h * r_filt_freq)).dec
standard_phot_per_s_r
```

Out[42]: $9512.591 \ \frac{1}{\mathrm{s}}$

In [43]:
```
count_rate_per_pixel_r = standard_phot_per_s_r / psf_area_pixels * received_fract
count_rate_per_pixel_r
```

Out[43]: $39.741867 \ \frac{1}{\mathrm{s\,pix^2}}$

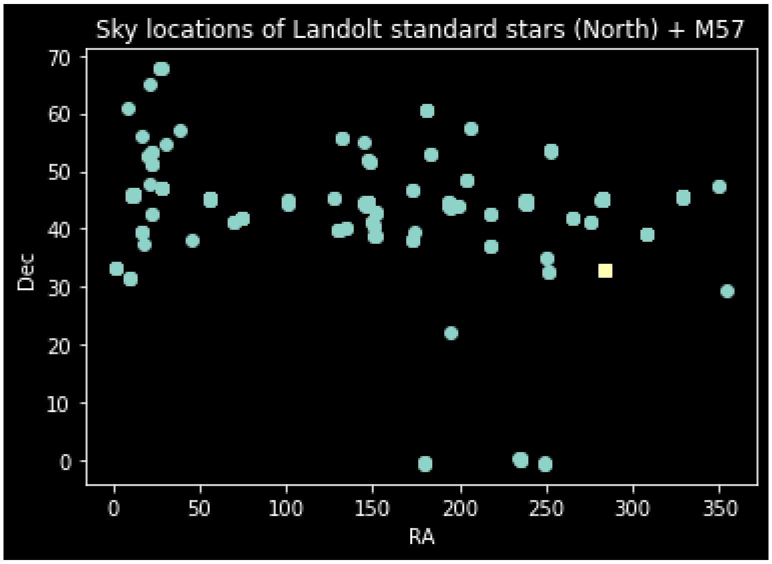
In [44]:
```
integration_time_including_readnoise_r = target_signal / standard_phot_per_s_r /
integration_time_including_readnoise_r
```

Out[44]: $5.3072667 \ \mathrm{s}$

M57

In [45]:
```
M57_coord = coordinates.SkyCoord.from_name('M57')
```

In [46]:
```
plt.figure()
_=plt.plot(landolt_coords.ra, landolt_coords.dec, 'o')
_=plt.plot(M57_coord.ra, M57_coord.dec, 's')
_=plt.xlabel("RA")
_=plt.ylabel("Dec")
_=plt.title("Sky locations of Landolt standard stars (North) + M57")
```



In [47]:
```
distances_from_M57_to_standards = M57_coord.separation(landolt_coords)
```

In [48]:
```
np.min(distances_from_M57_to_standards)
```

Out[48]: $10^{\circ}00'34.40821255''$

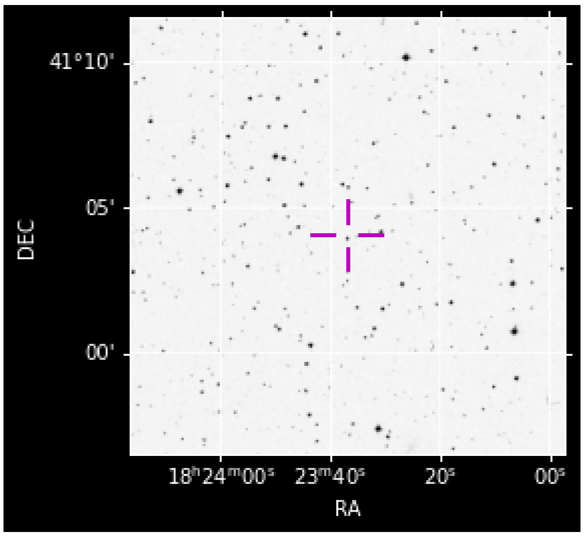
In [49]:
```
index = np.argmin(distances_from_M57_to_standards)
index, landolt_coords[index]
```

Out[49]:
```
(276,
 <SkyCoord (ICRS): (ra, dec) in deg
     (275.90421667, 41.067375)>)
```

In [50]:
```
print("V Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'])
print("B Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] + landolt_tb
print("R Magnitude of standard star: ",landolt_tbl[index]['__Vmag_'] - landolt_tb
```

```
V Magnitude of standard star:  14.288
B Magnitude of standard star:  14.208
R Magnitude of standard star:  14.324
```

In [51]:
```
ax, hdu = plot_finder_image(landolt_coords[index],
                            survey='DSS',
                            fov_radius=15*u.arcmin,
                            grid=False, reticle=True)
```



In [52]:
```
standard_star_vmag = landolt_tbl[index]['__Vmag_']
vmag_zeropoint = 3600*u.Jy
snu_standard = vmag_zeropoint * (10**(-standard_star_vmag/2.5))
standard_star_vmag, snu_standard
```

Out[52]:
```
(14.288, <Quantity 0.00693589 Jy>)
```

V

In [53]:
```
v_filt_wav = 5504*u.AA
v_filt_freq = (v_filt_wav).to(u.Hz, u.spectral())
v_filter_width = 1000*u.AA
A_CTO = (np.pi*(14/2 * u.imperial.inch)**2).to(u.cm**2)
```

In [54]:
```
standard_ergs_per_s = snu_standard * A_CTO * v_filt_freq*(v_filter_width/v_filt_w
standard_phot_per_s = (standard_ergs_per_s / (constants.h * v_filt_freq)).decompo
count_rate_per_pixel = standard_phot_per_s / psf_area_pixels * received_fraction
```

In [55]:
```
integration_time_including_readnoise = target_signal / standard_phot_per_s / rece
integration_time_including_readnoise
```

Out[55]: 26.729347 s

B

In [56]:
```
standard_star_bmag = landolt_tbl[index]['__Vmag_'] + landolt_tbl[index]['__B-V_']
bmag_zeropoint = 4000*u.Jy
snu_standard_b = bmag_zeropoint * (10**(-standard_star_bmag/2.5))
b_filt_wav = 4400*u.AA
b_filt_freq = (b_filt_wav).to(u.Hz, u.spectral())
b_filter_width = 1000*u.AA
```

In [57]:
```
standard_ergs_per_s_b = snu_standard_b * A_CTO * b_filt_freq*(b_filter_width/b_fi
standard_phot_per_s_b = (standard_ergs_per_s_b / (constants.h * b_filt_freq)).dec
count_rate_per_pixel_b = standard_phot_per_s_b / psf_area_pixels * received_fract
```

In [58]:
```
integration_time_including_readnoise_b = target_signal / standard_phot_per_s_b /
integration_time_including_readnoise_b
```

Out[58]: 17.865087 s

R

In [59]:
```
standard_star_Rmag = landolt_tbl[index]['__Vmag_'] - landolt_tbl[index]['__V-R_']
Rmag_zeropoint = 3000*u.Jy
snu_standard_R = Rmag_zeropoint * (10**(-standard_star_Rmag/2.5))
R_filt_wav = 6400*u.AA
R_filt_freq = (R_filt_wav).to(u.Hz, u.spectral())
R_filter_width = 1000*u.AA
standard_ergs_per_s_R = snu_standard_R * A_CTO * R_filt_freq*(R_filter_width/R_fi
standard_phot_per_s_R = (standard_ergs_per_s_R / (constants.h * R_filt_freq)).dec
count_rate_per_pixel_R = standard_phot_per_s_R / psf_area_pixels * received_fract
```

In [60]:
```
standard_ergs_per_s_R = snu_standard_R * A_CTO * R_filt_freq*(R_filter_width/R_fi
standard_phot_per_s_R = (standard_ergs_per_s_R / (constants.h * R_filt_freq)).dec
count_rate_per_pixel_R = standard_phot_per_s_R / psf_area_pixels * received_fract
```

In [61]:
```
integration_time_including_readnoise_R = target_signal / standard_phot_per_s_R /
integration_time_including_readnoise_R
```

Out[61]: 38.55416 s

In [ ]: