shumko_asilib_figures

July 27, 2022

1 Figure Notebook for "AuroraX, aurorax-api and aurorax-asilib: a user-friendly auroral all-sky imager analysis framework"

Before you can run this notebook, you'll need to install aurora-asi-lib via

```
python3 -m pip install aurora-asi-lib
```

1.1 If you're using Anaconda:

- Create and activate a new environment
- Install the `scipy` library via Anaconda
- Run the above pip command to install `asilib`'s other dependencies.

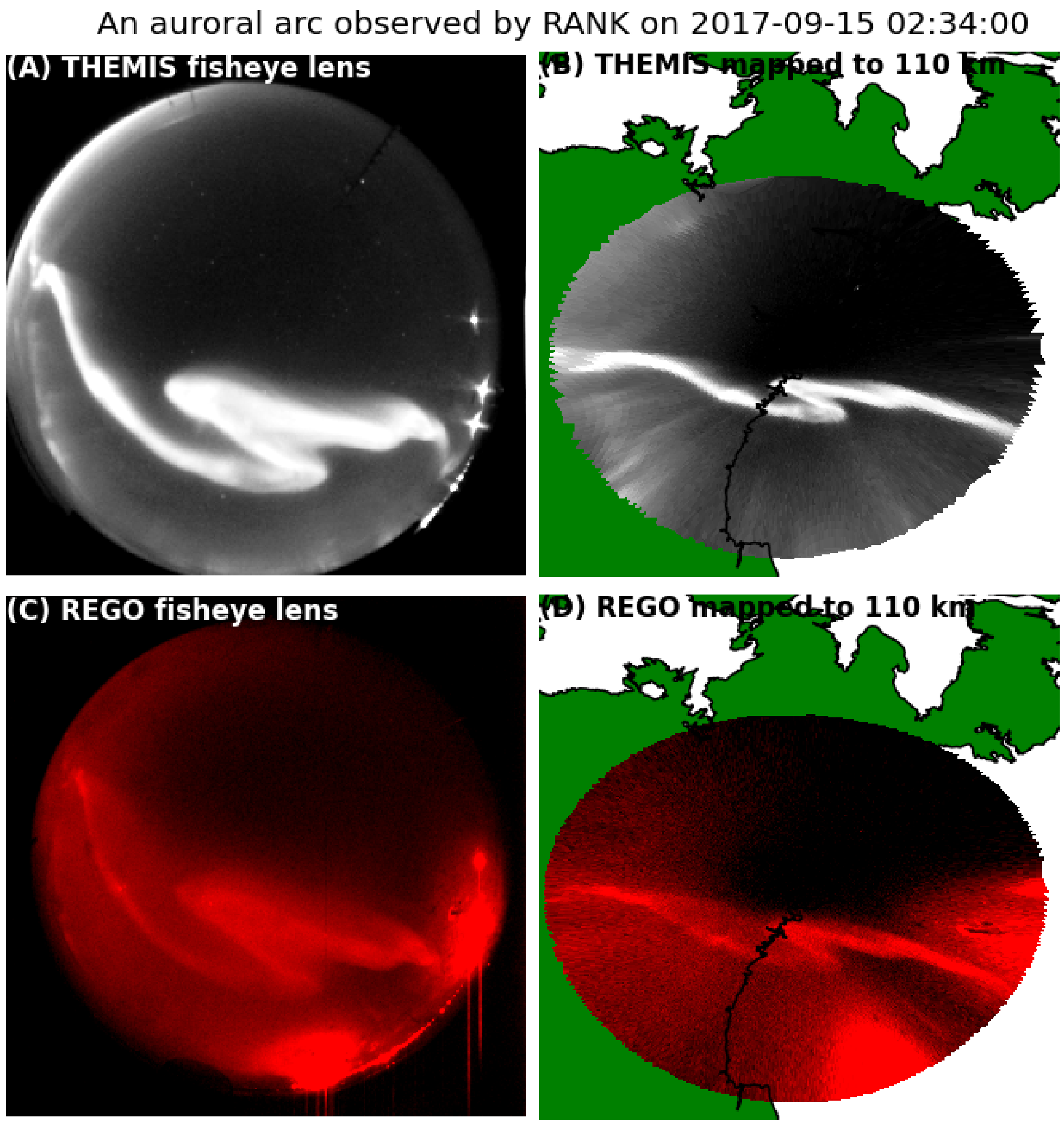
```
[ ]: from datetime import datetime, timedelta
     import string

     import matplotlib.pyplot as plt
     import matplotlib.dates
     import matplotlib.patches
     import matplotlib.gridspec
     import numpy as np
     import asilib

     print(f'asilib version: {asilib.__version__}')
```

```
asilib version: 0.12.1
```

2 Figure 2

```
[ ]: location_code = 'RANK'
     time = datetime(2017, 9, 15, 2, 34, 0)
     map_alt_km = 110
     fontsize=17

     lon_bounds = (-102, -82)
     lat_bounds = (58, 70)

     fig, ax = plt.subplots(2, 2, figsize=(10, 10))
```

```
asilib.make_map(ax=ax[0, 1], lon_bounds=lon_bounds, lat_bounds=lat_bounds)
asilib.make_map(ax=ax[1, 1], lon_bounds=lon_bounds, lat_bounds=lat_bounds)

ax[0, 0].axis('off')
ax[1, 0].axis('off')
ax[0, 1].axis('off')
ax[1, 1].axis('off')

asilib.plot_fisheye('THEMIS', location_code, time, ax=ax[0, 0], label=False)
asilib.plot_fisheye('REGO', location_code, time, ax=ax[1, 0], label=False)
asilib.plot_map('THEMIS', location_code, time, map_alt_km, ax=ax[0, 1],␣
 ↪asi_label=False)
asilib.plot_map('REGO', location_code, time, map_alt_km, ax=ax[1, 1],␣
 ↪asi_label=False)

ax[0, 0].text(0, 1, f'(A) THEMIS fisheye lens', va='top', transform=ax[0,0].
 ↪transAxes,
    color='white', fontsize=fontsize, weight='bold')
ax[0, 1].text(0, 1, f'(B) THEMIS mapped to {map_alt_km} km', va='top',␣
 ↪transform=ax[0,1].transAxes,
    color='k', fontsize=fontsize, weight='bold')
ax[1, 0].text(0, 1, f'(C) REGO fisheye lens', va='top', transform=ax[1,0].
 ↪transAxes,
    color='white', fontsize=fontsize, weight='bold')
ax[1, 1].text(0, 1, f'(D) REGO mapped to {map_alt_km} km', va='top',␣
 ↪transform=ax[1,1].transAxes,
    color='k', fontsize=fontsize, weight='bold')

plt.suptitle(f'An auroral arc observed by {location_code} on {time}',␣
 ↪fontsize=20)
plt.tight_layout()
plt.savefig('figures/fig2.jpg', dpi=300)
```


An auroral arc observed by RANK on 2017-09-15 02:34:00
(A) THEMIS fisheye lens
(B) THEMIS mapped to 110 km
(C) REGO fisheye lens
(D) REGO mapped to 110 km

```
[ ]: themis_skymap = asilib.load_skymap('THEMIS', location_code, time)
     rego_skymap = asilib.load_skymap('REGO', location_code, time)
```

```
[ ]: themis_skymap['SKYMAP_PATH']
```

```
[ ]: WindowsPath('C:/Users/mshumko/asilib-
     data/themis/skymap/rank/themis_skymap_rank_20170915_vXX.sav')
```

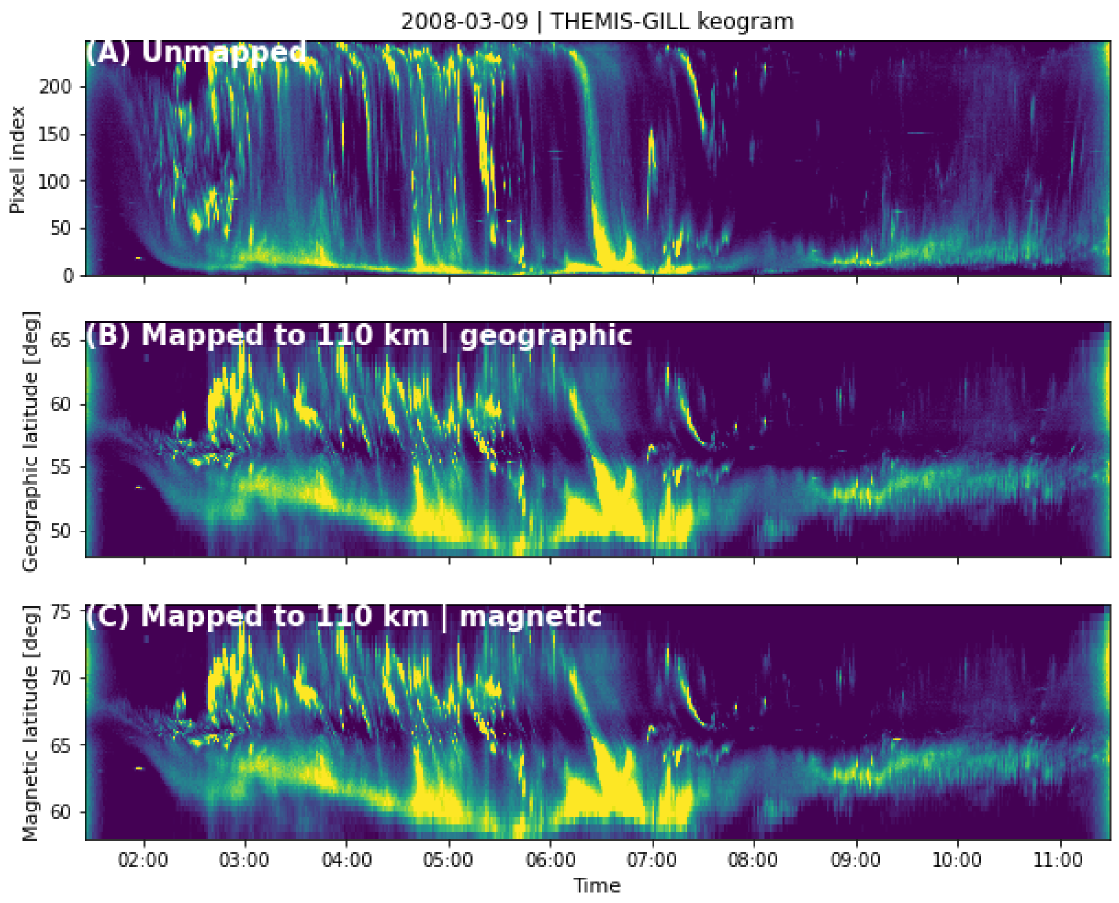
```
[ ]: rego_skymap['SKYMAP_PATH']
```

[ ]:
```
WindowsPath('C:/Users/mshumko/asilib-
data/rego/skymap/rank/rego_skymap_rank_20170817_v01.sav')
```

3 Figure 3

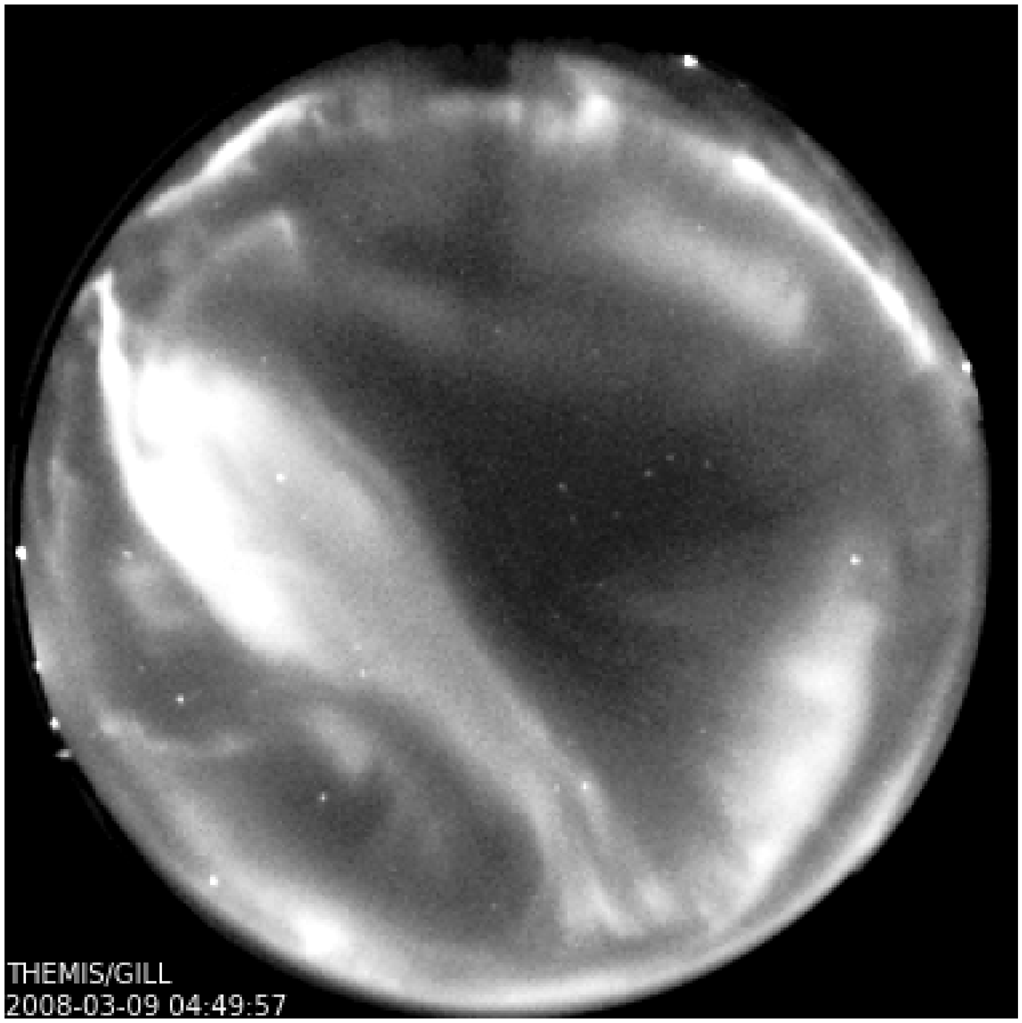
[ ]:
```
fontsize=11
fig, ax = plt.subplots(3, 1, figsize=(10, 8), sharex=True)
time_range = (datetime(2008, 3, 9), datetime(2008, 3, 10))
asilib.plot_keogram('THEMIS', 'GILL', time_range, ax=ax[0])
asilib.plot_keogram('THEMIS', 'GILL', time_range, ax=ax[1], map_alt=map_alt_km)
asilib.plot_keogram('THEMIS', 'GILL', time_range, ax=ax[2], map_alt=map_alt_km,␣
 ↪aacgm=True)
ax[-1].set_xlabel('Time', fontsize=fontsize)
ax[0].set_ylabel('Pixel index', fontsize=fontsize)
ax[1].set_ylabel('Geographic latitude [deg]', fontsize=fontsize)
ax[2].set_ylabel('Magnetic latitude [deg]', fontsize=fontsize)

fmtr = matplotlib.dates.DateFormatter("%H:%M")
ax[-1].xaxis.set_major_formatter(fmtr)
ax[1].set_title('')
ax[2].set_title('')
ax[0].text(0, 1, f'(A) Unmapped', va='top', transform=ax[0].transAxes,
    color='white', fontsize=fontsize+4, weight='bold')
ax[1].text(0, 1, f'(B) Mapped to {map_alt_km} km | geographic', va='top',␣
 ↪transform=ax[1].transAxes,
    color='white', fontsize=fontsize+4, weight='bold')
ax[2].text(0, 1, f'(C) Mapped to {map_alt_km} km | magnetic', va='top',␣
 ↪transform=ax[2].transAxes,
    color='white', fontsize=fontsize+4, weight='bold')
plt.savefig('figures/fig3.jpg', dpi=300)
```

```
c:\ProgramData\Anaconda3\envs\asilib\lib\site-packages\scipy\io\idl.py:279:
UserWarning: Not able to verify number of bytes from header
  warnings.warn("Not able to verify number of bytes from header")
c:\ProgramData\Anaconda3\envs\asilib\lib\site-packages\scipy\io\idl.py:279:
UserWarning: Not able to verify number of bytes from header
  warnings.warn("Not able to verify number of bytes from header")
c:\ProgramData\Anaconda3\envs\asilib\lib\site-packages\scipy\io\idl.py:279:
UserWarning: Not able to verify number of bytes from header
  warnings.warn("Not able to verify number of bytes from header")
```



```
[ ]: time_range = (datetime(2008, 3, 9, 4, 35), datetime(2008, 3, 9, 4, 50))
     asilib.animate_fisheye('THEMIS', 'GILL', time_range, overwrite=True)
     print(f'Movie saved in {asilib.config["ASI_DATA_DIR"] / "animations"}')
```

```
Created a C:\Users\mshumko\asilib-
data\animations\images\20080309_043500_themis_gill_fisheye directory
Movie saved in C:\Users\mshumko\asilib-data\animations
```



4 Figure 4

A conjunction montage. Lets take this one step at a time. First we define the ASI info and load the skymap file (to make the fictional satellite path overhead).

```
[ ]: asi_array_code = 'THEMIS'
     location_code = 'RANK'
     area_box_km = (20, 20)
     time_range = (datetime(2017, 9, 15, 2, 32, 0), datetime(2017, 9, 15, 2, 35, 0))
```

```
[ ]: skymap_dict = asilib.load_skymap(asi_array_code, location_code, time_range[0])
```

Create the satellite path (time, latitude, longitude, altitude) at a 500 km altitude. It is a north-south satellite track oriented to the east of the imager.

```
[ ]: n = int((time_range[1] - time_range[0]).total_seconds() / 3)  #  3 second␣
      ↪cadence.
     time = np.array([time_range[0] + timedelta(seconds=i*3) for i in range(n)])
     lats = np.linspace(skymap_dict["SITE_MAP_LATITUDE"] + 2,␣
      ↪skymap_dict["SITE_MAP_LATITUDE"] - 3, n)
     lons = (skymap_dict["SITE_MAP_LONGITUDE"] - 0.25) * np.ones(n)
     alts = 500 * np.ones(n)
     time_lla_500km = np.array([time, lats, lons, alts]).T
```

Map the satellite's altitude from 500 km to the 110 km footprint. Time is necessary to correctly evaluate the magnetic field model.

NOTE You will need to install IRBEM for the following line to run.

```
[ ]: lla_110km = asilib.lla2footprint(time_lla_500km, 110)
```

Next, map the satellite's footprint to the imager's (Azimuth, Elevation), i.e. AzEl coordinates.

```
[ ]: sat_azel, sat_azel_pixels = asilib.lla2azel(asi_array_code, location_code,␣
      ↪time_range[0], lla_110km)
```

The last step before we make the movie is to calculate what pixels are in a box_km around the satellite, to convolve it with the images to pick out the ASI intensity in that box.

```
[ ]: area_box_mask = asilib.equal_area(
         asi_array_code, location_code, time_range[0], lla_110km, box_km=area_box_km
     )
```

Calculate the mean ASI intensity in the area_box_km

```
[ ]: times, images = asilib.load_image(asi_array_code, location_code,␣
      ↪time_range=time_range)
     asi_brightness = np.nanmean(images * area_box_mask, axis=(1, 2))
     area_box_mask[np.isnan(area_box_mask)] = 0  # To play nice with plt.contour()
```

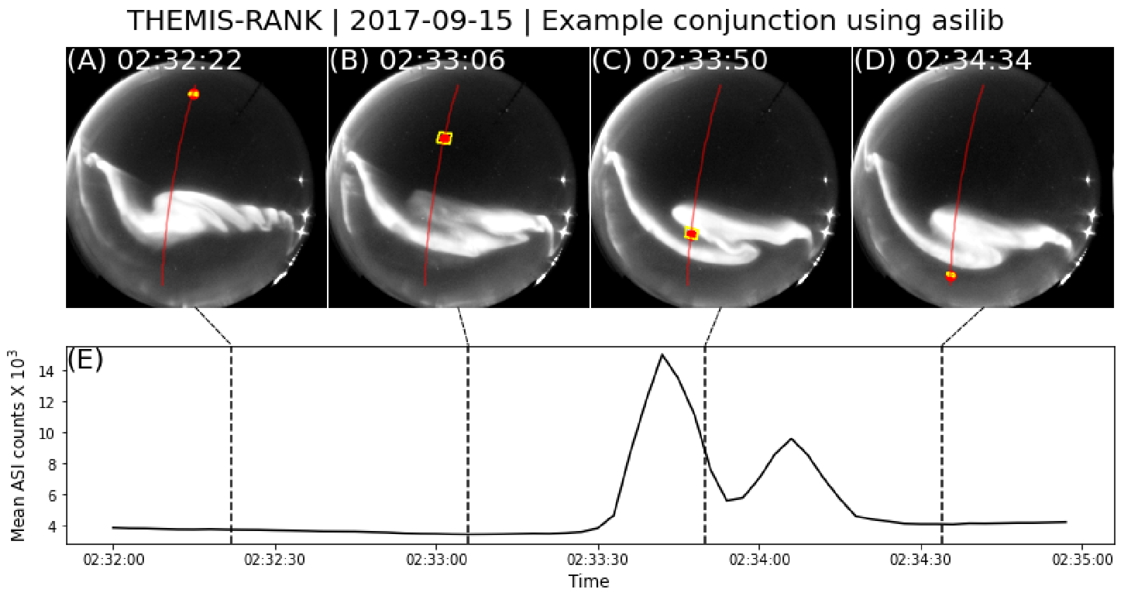
Plot preparation

```
[ ]: num_images = 4

     delta_time_s = int((time[-1]-time[0]).total_seconds()/num_images)
     montage_times = [time_range[0]+timedelta(seconds=(i+0.5)*delta_time_s) for i in␣
      ↪range(num_images)]
     formatted_times = [t.strftime('%H:%M:%S') for t in montage_times]
     subplot_labels = [f'({l}) {t}' for l, t in zip(string.ascii_uppercase[:
      ↪num_images], formatted_times)]

     # For each montage time we need to know the index in the time array so we can␣
      ↪reference
     # it later.
     downsampled_satellite_indices = np.zeros(num_images)
```

```
for i, montage_time in enumerate(montage_times):
    t_diff = np.abs([(t - montage_time).total_seconds() for t in times])
    downsampled_satellite_indices[i] = np.argmin(t_diff)
```

[ ]:
```
fig = plt.figure(figsize=(12, 6))

gs = matplotlib.gridspec.GridSpec(5, num_images, figure=fig)
ax = [fig.add_subplot(gs[:3, n]) for n in range(num_images)]
bx = fig.add_subplot(gs[3:, :])

# Draw the fisheye lens images
for i, (montage_time, ax_i, subplot_label) in enumerate(zip(montage_times, ax,␣
 ↪subplot_labels)):
    asilib.plot_fisheye(asi_array_code, location_code, montage_time, ax=ax_i,␣
 ↪label=False)
    ax_i.axis('off')

    index = int(downsampled_satellite_indices[i])
    ax_i.plot(sat_azel_pixels[:, 0], sat_azel_pixels[:, 1], 'red', alpha=0.5)
    ax_i.scatter(sat_azel_pixels[index, 0], sat_azel_pixels[index, 1], c='red',␣
 ↪marker='o', s=50)
    ax_i.contour(area_box_mask[index, :, :], levels=[0.99], colors=['yellow'])
    ax_i.text(0, 1, subplot_label, va='top', transform=ax_i.transAxes,␣
 ↪color='white', fontsize=20)

bx.plot(time, asi_brightness/1000, c='k')
bx.text(0, 1, f'({string.ascii_uppercase[num_images]})', va='top', transform=bx.
 ↪transAxes, fontsize=20)
fig.suptitle(f'{asi_array_code}-{location_code} | {montage_times[0].date()} | '
             f'Example conjunction using asilib', fontsize=20)
bx.set_ylabel(f'Mean ASI counts X $10^3$', fontsize=12)
bx.set_xlabel('Time', fontsize=12)

# ConnectAdd Guide lines between the fisheye images and the mean auroral␣
 ↪intensity subplots.
for ax_i, image_time_numeric in zip(ax, matplotlib.dates.
 ↪date2num(montage_times)):
    line = matplotlib.patches.ConnectionPatch(
        xyA=(0.5, 0), coordsA=ax_i.transAxes,
        xyB=(image_time_numeric, bx.get_ylim()[1]), coordsB=bx.transData,
        ls='--')
    ax_i.add_artist(line)

    bx.axvline(image_time_numeric, c='k', ls='--')
```

```
plt.subplots_adjust(hspace=0.17, wspace=0.01, top=0.95, bottom=0.1, left=0.09,␣
 ↪right=0.95)
plt.savefig('figures/fig4.jpg', dpi=300)
```



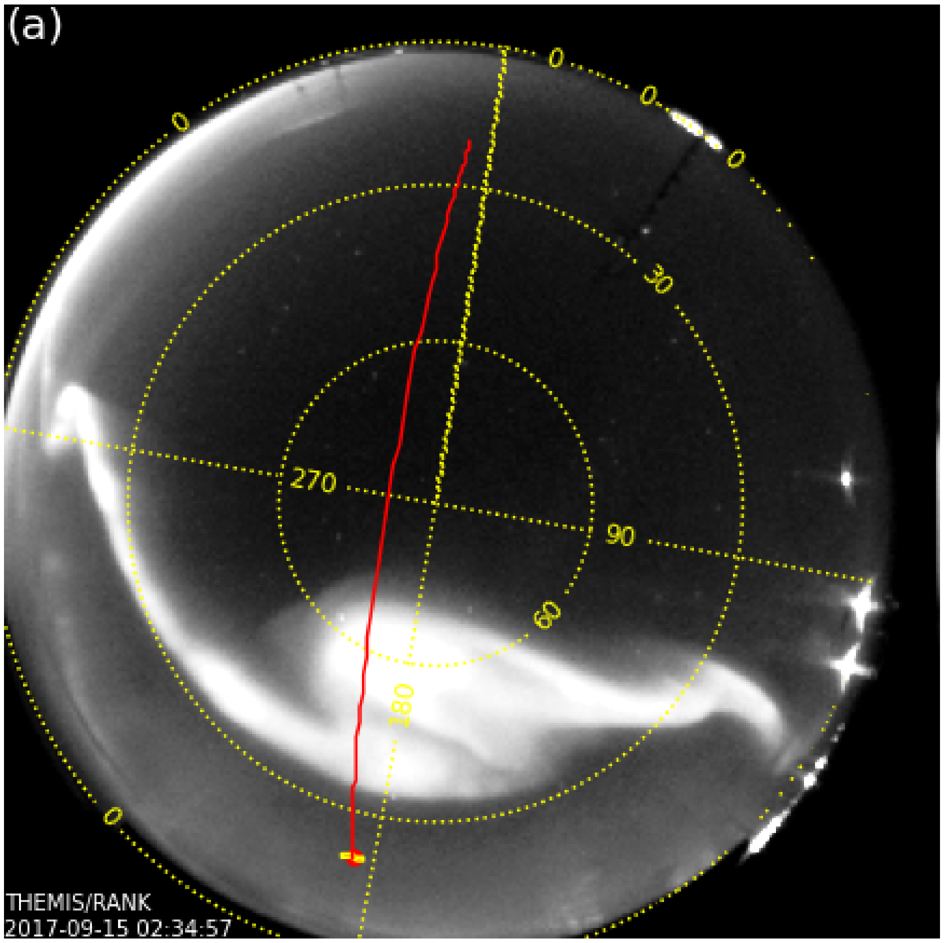
5 Movie S2

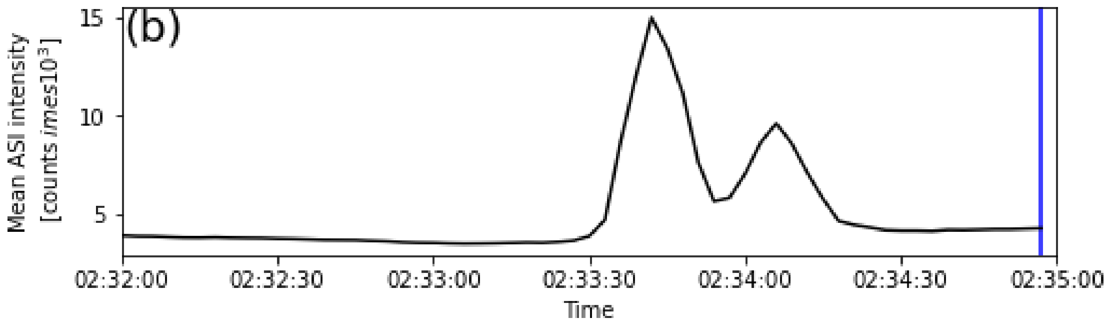
Now to make the conjunction movie.

```
[ ]: fig, ax = plt.subplots(
         2, 1, figsize=(7, 8.5), gridspec_kw={'height_ratios': [4, 1]}
     )

     # Initiate the movie generator function. Any errors with the data will be␣
      ↪raised here.
     movie_generator = asilib.animate_fisheye_generator(
         asi_array_code, location_code, time_range, azel_contours=True,␣
      ↪overwrite=True, ax=ax[0]
     )

     # Use the generator to get the images and time stamps to estimate mean the ASI
     # brightness along the satellite path and in a (20x20 km) box.
     image_data = movie_generator.send('data')

     area_box_mask_2 = asilib.equal_area(
         asi_array_code, location_code, time_range[0], lla_110km, box_km=area_box_km
     )
```

```
asi_brightness_2 = np.nanmean(image_data.images * area_box_mask_2, axis=(1, 2))
area_box_mask_2[np.isnan(area_box_mask_2)] = 0  # To play nice with plt.
 ↪contour()

for i, (time, image, _, im) in enumerate(movie_generator):
    # Note that because we are drawing different data in each frame (a unique␣
 ↪ASI
    # image in ax[0] and the ASI time series + a guide in ax[1], we need
    # to redraw everything at every iteration.

    ax[1].clear()  # ax[0] cleared by asilib.animate_fisheye_generator()
    # Plot the entire satellite track, its current location, and a 20x20 km box
    # around its location.
    ax[0].plot(sat_azel_pixels[:, 0], sat_azel_pixels[:, 1], 'red')
    ax[0].scatter(sat_azel_pixels[i, 0], sat_azel_pixels[i, 1], c='red',␣
 ↪marker='o', s=50)
    ax[0].contour(area_box_mask_2[i, :, :], levels=[0.99], colors=['yellow'])

    # Plot the time series of the mean ASI intensity along the satellite path
    ax[1].axvline(time, c='b')
    ax[1].plot(image_data.time, asi_brightness_2/1000,'k')

    ax[1].set(xlabel='Time', ylabel=f'Mean ASI intensity\n [counts $\times␣
 ↪10^3$]',
            xlim=time_range)
    ax[1].text(0, 1, '(b)', va='top', transform=ax[1].transAxes, color='black',␣
 ↪fontsize=20)

    # Annotate the location_code and satellite info in the top-left corner.
    ax[0].text(0, 1, '(a)', va='top', transform=ax[0].transAxes, color='white',␣
 ↪fontsize=20)

    plt.subplots_adjust(wspace=0, hspace=0, right=0.98, left=0.12, bottom=0.05,␣
 ↪top=0.99)

print(f'Movie saved in {asilib.config["ASI_DATA_DIR"] / "animations"}')
```

```
Created a C:\Users\mshumko\asilib-
data\animations\images\20170915_023200_themis_rank_fisheye directory
Movie saved in C:\Users\mshumko\asilib-data\animations
```


(a)
0
0
0
0
30
270
90
60
180
0
THEMIS/RANK
2017-09-15 02:34:57


(b)
Mean ASI intensity
[counts $imes10^3$]
15
10
5
02:32:00
02:32:30
02:33:00
02:33:30
02:34:00
02:34:30
02:35:00
Time