

mosaics

July 24, 2023

1 Plotting and analyzing ASI mosaics

```
[ ]: from datetime import datetime

import matplotlib.pyplot as plt
import matplotlib.colors
import aacgm2
import numpy as np
import scipy.interpolate

import asilib
import asilib.asi
import asilib.map

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

asilib version: 0.18.1

We first create an `asilib.Imagers()` object consisting of TREx-RGB `asilib.Imagers()` defined a list of `location_codes`.

```
[ ]: asilib.asi.trex.trex_rgb_info()
```

```
[ ]:      array location_code      name  latitude  longitude
0  TREx_RGB      ATHA    Athabasca    54.60    -113.64
1  TREx_RGB      FSMI    Fort Smith   60.03    -111.93
2  TREx_RGB      GILL      Gillam    56.38     -94.64
3  TREx_RGB      LUCK    Lucky Lake  51.15   -107.26
4  TREx_RGB      PINA      Pinawa    50.26    -95.87
5  TREx_RGB      RABB    Rabbit Lake  58.23   -103.68
```

```
[ ]: time = datetime(2021, 11, 4, 7, 3, 51)
location_codes = ['LUCK', 'RABB', 'PINA', 'GILL']
map_alt = 110
min_elevation = 10
```

```
[ ]: _imagers = []
```

```

for location_code in location_codes:
    _imagers.append(asilib.asi.trex.trex_rgb(location_code, time=time,
    ↪alt=map_alt))

asis = asilib.Imagers(_imagers)

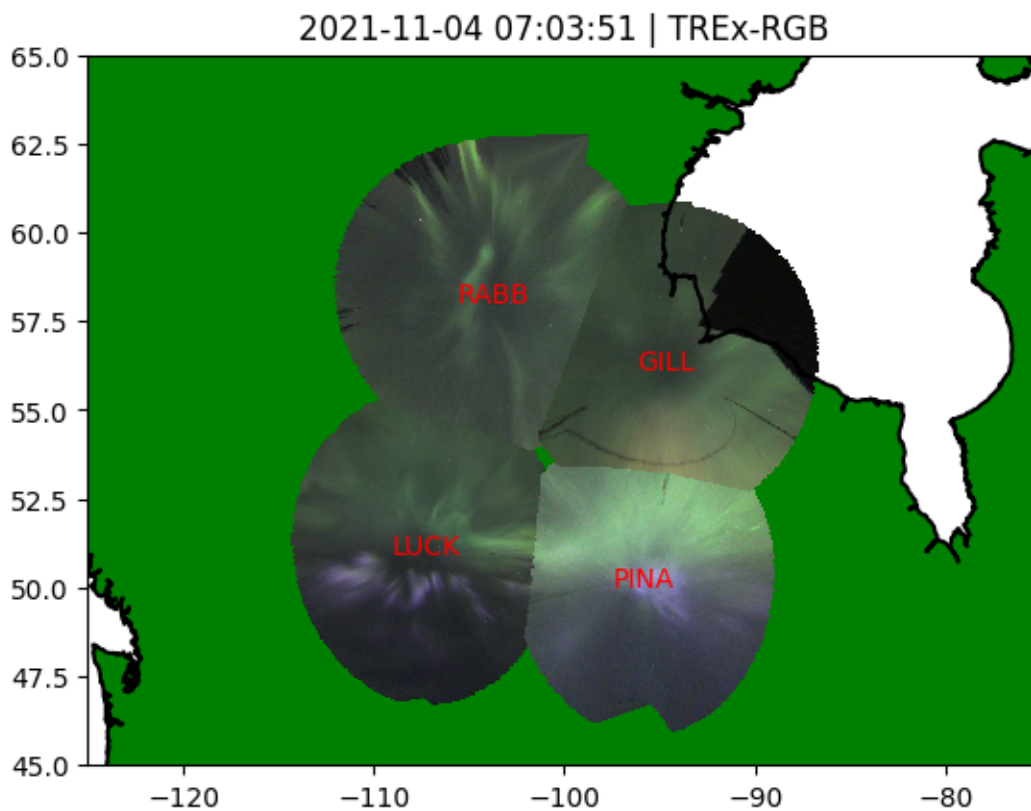
```

Plot a TREx-RGB mosaic with and without AACGM magnetic latitude contours.

```

[ ]: lon_bounds=(-125, -75)
lat_bounds=(45, 65)
ax = asilib.map.create_simple_map(lon_bounds=lon_bounds, lat_bounds=lat_bounds)
asis.plot_map(ax=ax, overlap=False, min_elevation=min_elevation)
plt.title(f'{time} | TREx-RGB');

```



```

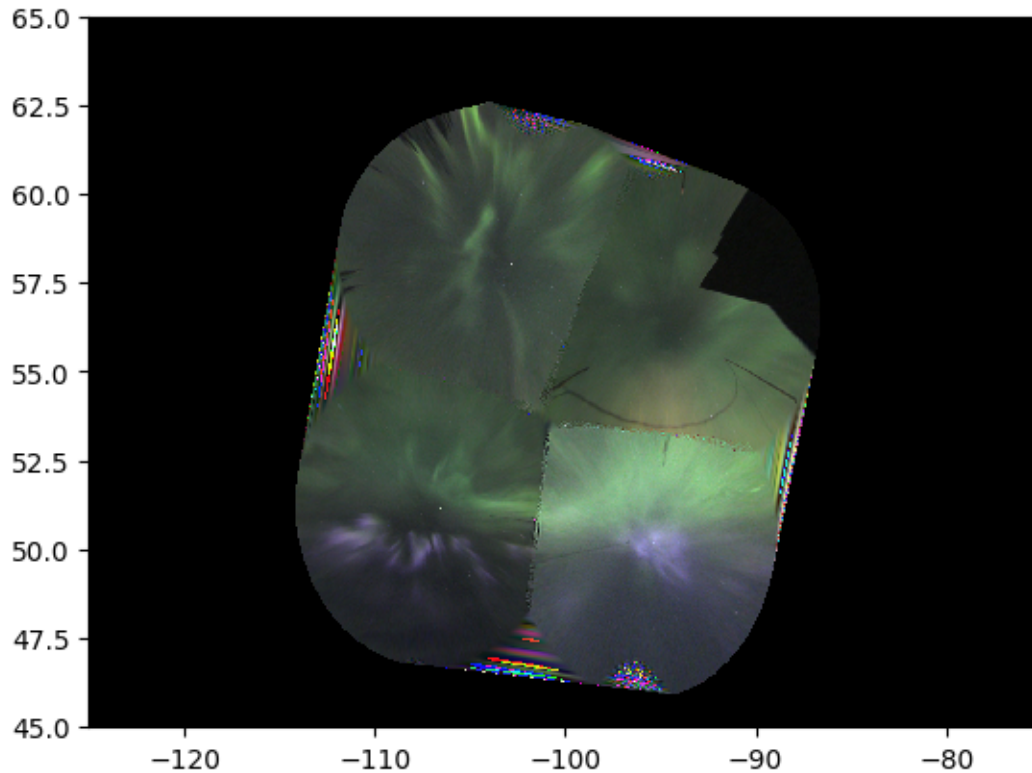
[ ]: lat_grid, lon_grid = np.meshgrid(np.linspace(*lat_bounds), np.
    ↪linspace(*lon_bounds, num=51))
# Need to pass flattened arrays since aacgm2 does not work with n-D arrays.
aacgm_lat_grid, aacgm_lon_grid, _ = aacgm2.wrapper.convert_latlon_arr(
    lat_grid.flatten(), lon_grid.flatten(), 110, time, method_code='G2A'
)
aacgm_lat_grid = aacgm_lat_grid.reshape(lat_grid.shape)

```



```
[ ]: plt.pcolormesh(lon_grid2, lat_grid2, rgb_grid/255) # Need to divide by 255 to indicate RGB channels to pcolormesh.
```

```
[ ]: <matplotlib.collections.QuadMesh at 0x1ac0f8e0890>
```

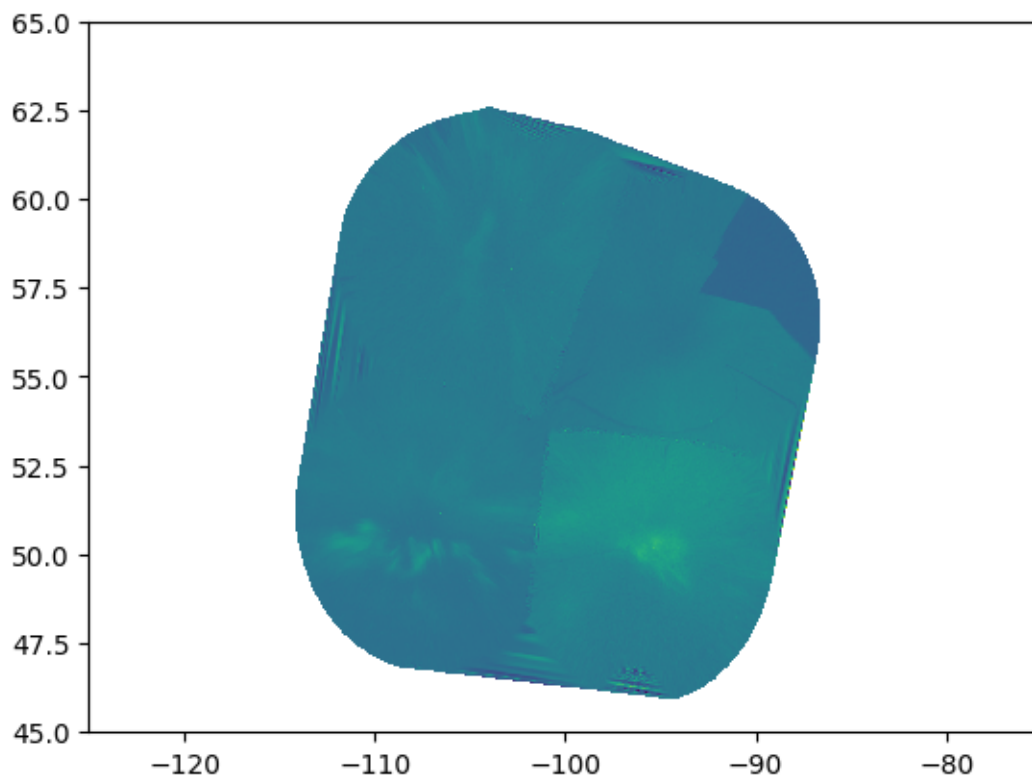


Looks pretty good except some of the edges! Now, how about the individual colors and green-blue ratio?

First, the blue grid:

```
[ ]: plt.pcolormesh(lon_grid2, lat_grid2, b_grid)
```

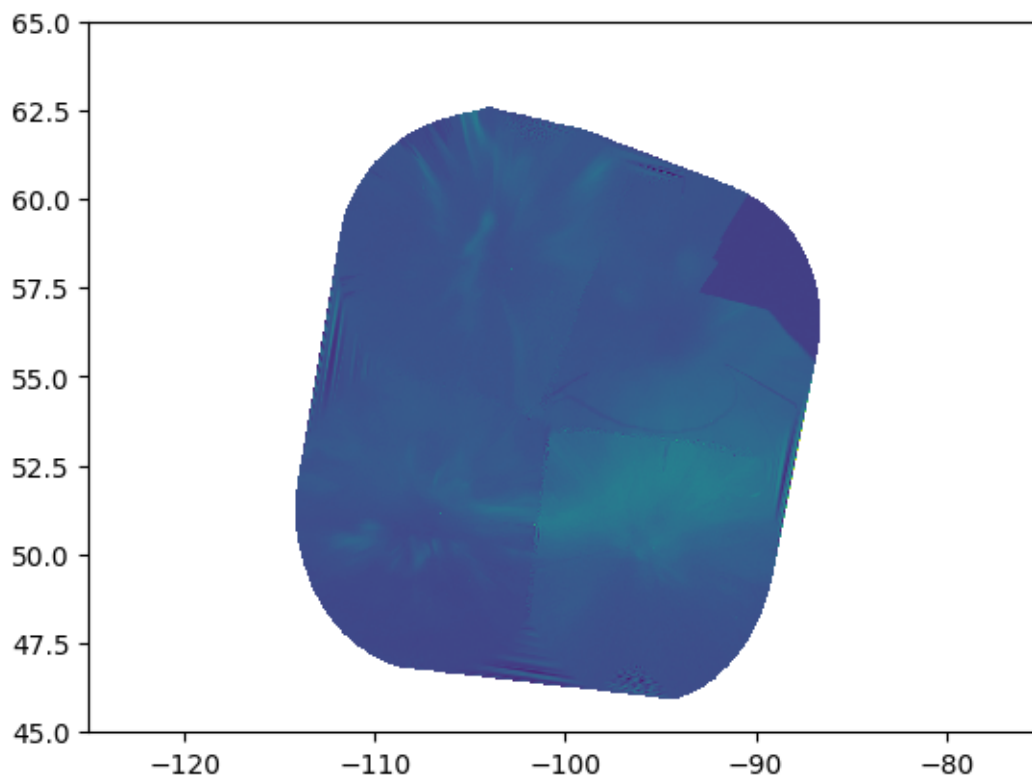
```
[ ]: <matplotlib.collections.QuadMesh at 0x1ac0f7efad0>
```



And the green grid:

```
[ ]: plt.pcolormesh(lon_grid2, lat_grid2, g_grid)
```

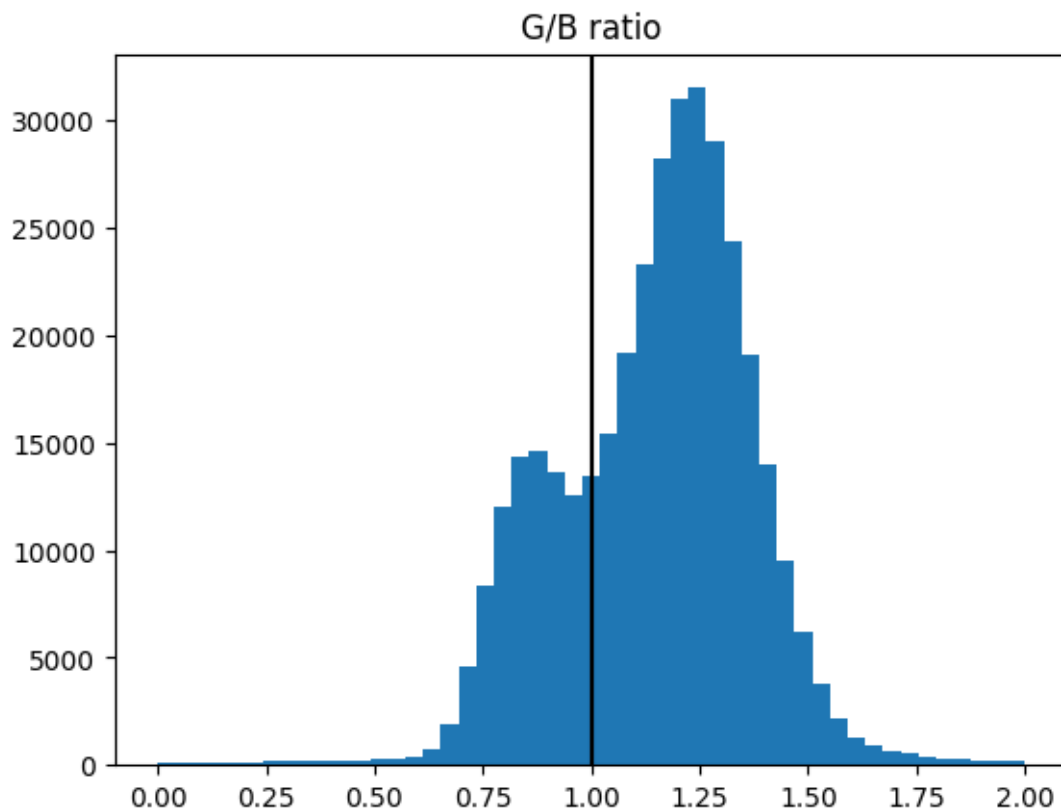
```
[ ]: <matplotlib.collections.QuadMesh at 0x1ac0f7dee10>
```



And the ratio:

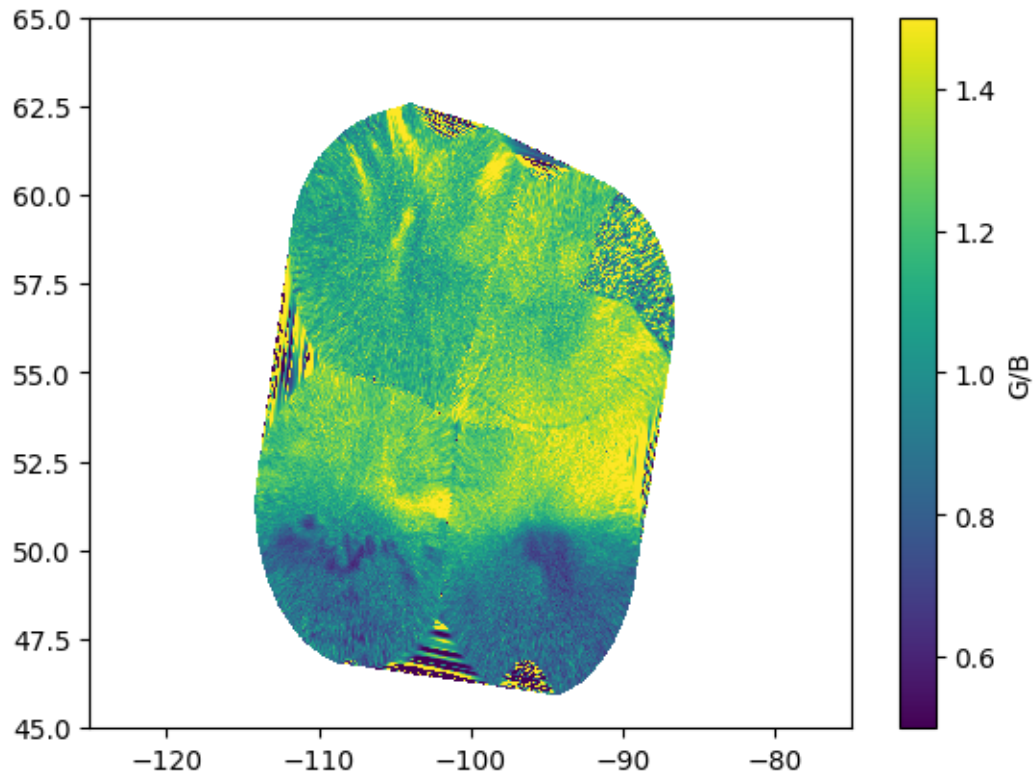
```
[ ]: plt.hist((g_grid/b_grid).flatten(), bins=np.linspace(0, 2));  
      plt.title('G/B ratio')  
      plt.axvline(1, c='k')
```

```
[ ]: <matplotlib.lines.Line2D at 0x1ac0f79c210>
```



```
[ ]: plt.pcolormesh(lon_grid2, lat_grid2, g_grid/b_grid, vmin=0.5, vmax=1.5)
     plt.colorbar(label='G/B')
```

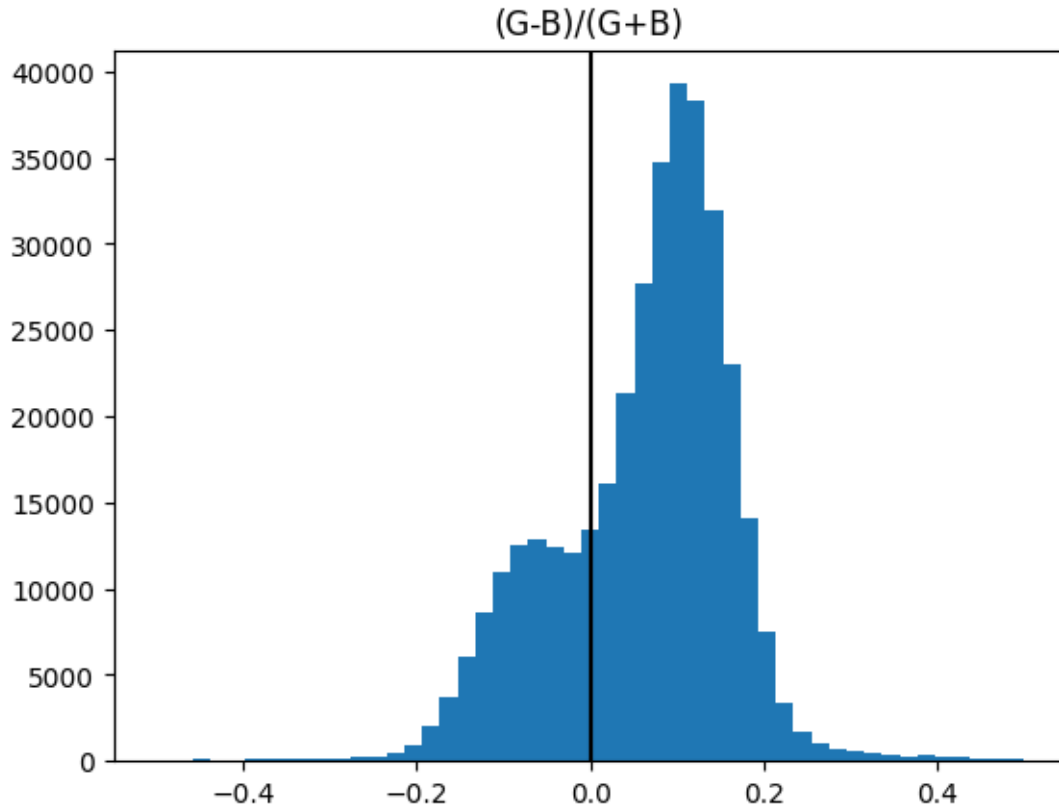
```
[ ]: <matplotlib.colorbar.Colorbar at 0x1ac0ec14cd0>
```



Normalized ratio

```
[ ]: plt.hist(((g_grid-b_grid)/(g_grid+b_grid)).flatten(), bins=np.linspace(-0.5, 0.
↪5));
plt.title('(G-B)/(G+B)')
plt.axvline(0, c='k')
```

```
[ ]: <matplotlib.lines.Line2D at 0x1ac0ecb3c90>
```

```
[ ]: ax = asilib.map.create_simple_map(lon_bounds=lon_bounds, lat_bounds=lat_bounds,
    ↪ land_color='grey')
mesh = ax.pcolormesh(lon_grid2, lat_grid2, (g_grid-b_grid)/(g_grid+b_grid),
    ↪ vmin=-0.2, vmax=0.2, cmap='PiYG')
plt.colorbar(mesh, label='(G-B)/(G+B)')
cs = plt.contour(lon_grid, lat_grid, aacgm_lat_grid, colors='k')
ax.clabel(cs, inline=True, fontsize=10, fmt=lambda x: f'$\lambda = \circ{x}$')
ax.set(
    title=f'{time}\nInterpolated TReX-RGB | $\lambda_{AACGM}$ contours',
    xlabel='Geo longitude [deg]',
    ylabel='Geo latitude [deg]'
);
```

