coordinates

May 16, 2023

1 Coordinates transforms and remapping with SunPy

Based on the SunPy examples gallery.

1.1 Getting the data we will use

We get some AIA, EUI, and SPICE files for 2022-04-02.

- EUI and SPICE directly by URL, for simplicity (otherwise one could use `sunpy.net.Fido` to find them through VSO).
- AIA from MEDOC using PySitools2 (we could also have used VSO using `sunpy.net.Fido`)

We use `parfive` for parallel download.

[1]:
```
from pathlib import Path
from datetime import datetime
import numpy as np
import matplotlib.pyplot as plt

import astropy.units as u
from astropy.coordinates import SkyCoord
from astropy.wcs import WCS
from reproject import reproject_interp
from reproject.mosaicking import reproject_and_coadd
import parfive

from sunpy.map import Map, make_heliographic_header, make_fitswcs_header,␣
    ↪all_coordinates_from_map
from sunpy.coordinates import Helioprojective, get_body_heliographic_stonyhurst
from sunraster.instr.spice import read_spice_l2_fits

from sitools2 import SdoClientMedoc

%matplotlib widget
plt.rcParams["figure.figsize"] = (10,8) # larger default figure size
```

[2]:
```
# Search for AIA 171 map at MEDOC
sdo_client = SdoClientMedoc()
```

```
aia_data_list = sdo_client.search(dates=[datetime(2022, 4, 2, 10),␣
 ↪datetime(2022, 4, 2, 10, 2)], waves=[171])
```

```
cadence parameter not specified, default value for aia.lev1 is set : cadence =
[1m]
Loading client : https://idoc-medoc.ias.u-psud.fr
2 results returned
```

[3]:
```
# dict of URLs, one file per instrument, for this tutorial
urls = {
    'aia': aia_data_list[0].url,
    'fsi': 'https://www.sidc.be/EUI/data/releases/202301_release_6.0/L2/2022/04/
 ↪02/solo_L2_eui-fsi174-image_20220402T100045611_V01.fits',
    'hri': 'https://www.sidc.be/EUI/data/releases/202301_release_6.0/L2/2022/04/
 ↪02/solo_L2_eui-hrieuv174-image_20220402T100005600_V01.fits',
    'spice': 'https://spice.osups.universite-paris-saclay.fr/spice-data/
 ↪release-3.0/level2/2022/04/02/
 ↪solo_L2_spice-n-ras_20220402T101536_V06_100664001-000.fits',
}
```

[4]:
```
# Download data to files/ directory (if they are not already there)
download_dir = Path('files')
download_dir.mkdir(exist_ok=True)
downloader = parfive.Downloader()
for instrument in urls:
    downloader.enqueue_file(urls[instrument], download_dir, f'{instrument}.
 ↪fits')
downloader.download()
```

```
Files Downloaded:   0%|          | 0/4 [00:00<?, ?file/s]
```

[4]:
```
<parfive.results.Results object at 0x7f00b0c00d00>
['files/aia.fits', 'files/fsi.fits', 'files/hri.fits', 'files/spice.fits']
```

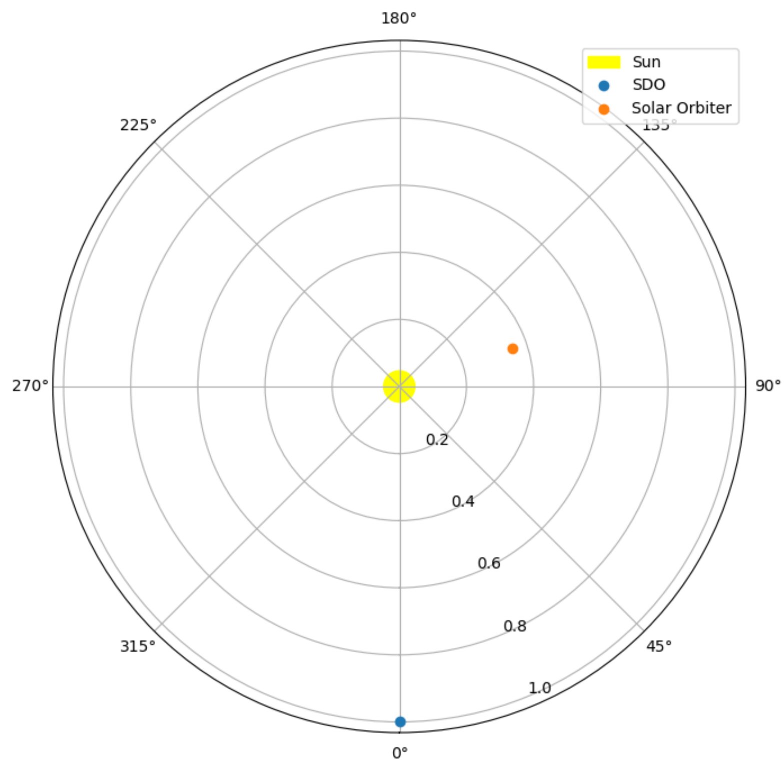
[5]:
```
# Build dictionary of maps for all image files (not SPICE)
# Resampling is optional, this is to speed up the tutorial running time
maps = {
    instrument: Map(
        download_dir / f'{instrument}.fits'
    ).resample((1024, 1024)*u.pix)
    for instrument in urls if instrument != 'spice'
}
```

1.2 Compare EUI and AIA

The viewpoints are different. Earth to Solar Orbiter longitude difference is 108.5°.

[6]:
```
fig = plt.figure()
ax = fig.add_subplot(projection='polar')
sun = plt.Circle(
    (0.0, 0.0), (10*u.Rsun).to_value(u.AU),
    transform=ax.transProjectionAffine + ax.transAxes, color="yellow",
    alpha=1, label="Sun"
)
ax.add_artist(sun)

for instrument in ['aia', 'fsi']:
    coordinates = maps[instrument].observer_coordinate
    ax.plot(coordinates.lon.to('rad'), coordinates.radius.to(u.AU), 'o',␣
  ↪label=maps[instrument].observatory)
ax.set_theta_zero_location("S")
ax.legend()
plt.show()
```



[7]:
```
maps['aia']
```

[7]:
```
<sunpy.map.sources.sdo.AIAMap object at 0x7f004d03ceb0>
SunPy Map
---------
Observatory:                 SDO
Instrument:                  AIA 3
Detector:                    AIA
Measurement:                 171.0 Angstrom
Wavelength:                  171.0 Angstrom
Observation Date:            2022-04-02 10:00:09
Exposure Time:               2.000168 s
Dimension:                   [1024. 1024.] pix
Coordinate System:           helioprojective
Scale:                       [2.3979559 2.3979559] arcsec / pix
Reference Pixel:             [513.139405 511.54712 ] pix
Reference Coord:             [0. 0.] arcsec
array([[-1.  , -0.5 , -1.5 , …, -0.75,  0.75,  0.  ],
       [ 0.  , -0.5 ,  0.25, …,  0.25,  0.5 ,  0.  ],
       [-0.5 ,  0.75, -1.25, …,  0.25,  0.5 ,  0.  ],
       …,
       [ 0.  , -0.75,  0.75, …,  0.  ,  0.25,  0.25],
       [ 1.  , -0.5 , -0.25, …,  1.  , -0.25,  0.25],
       [ 0.5 ,  0.25,  0.5 , …, -0.75, -0.25, -0.75]])
```

[8]:
```
maps['fsi']
```

[8]:
```
<sunpy.map.sources.solo.EUIMap object at 0x7f006ed6ff10>
SunPy Map
---------
Observatory:                 Solar Orbiter
Instrument:                  EUI
Detector:                    FSI
Measurement:                 174.0 Angstrom
Wavelength:                  174.0 Angstrom
Observation Date:            2022-04-02 10:00:50
Exposure Time:               10.0 s
Dimension:                   [1024. 1024.] pix
Coordinate System:           helioprojective
Scale:                       [13.18161946 13.32037335] arcsec / pix
Reference Pixel:             [516.88947368 511.5       ] pix
Reference Coord:             [-1895.81391692   790.4549648 ] arcsec
array([[0.        , 0.28369141, 0.28369141, …, 0.28369141, 0.44571686,
        0.28369141],
       [0.        , 0.41912079, 0.28369141, …, 0.54521942, 0.28369141,
        0.45016479],
       [0.        , 0.71611023, 0.82891083, …, 1.64720917, 0.55408478,
        3.4412384 ],
       …,
```

```
[0.        , 2.12841797, 2.12841797, …, 0.        , 0.        ,
 0.        ],
[0.        , 0.        , 0.        , …, 0.        , 0.        ,
 0.        ],
[0.        , 0.        , 0.        , …, 0.        , 0.        ,
 0.        ]])
```

1.3 Select a point and region on AIA map, plot them on FSI map

Helioprojective coordinates are defined for a given observer (including position and time), the corresponding frame can be obtained from the map.

We first need to ensure that the solar radius is the same in each map. We then have to adjust the radius in the AIA map so that it is consistent with the other ones.

This radius actually is the IAU official solar radius constant. Reprojections of the corona should use a different radius for the sphere on which images are projected, but this is out of the scope of this tutorial.

[9]:
```
# Print solar reference radii
for i in maps:
    print(i, maps[i].meta['rsun_ref'])
```

```
aia 696000000.0
fsi 695700000
hri 695700000
```

[10]:
```
# These are not consistent, we then adjust the radius for AIA to have the same␣
↪value as the others
original_aia_rsun = maps['aia'].meta['rsun_ref']  # keep for later
maps['aia'].meta['rsun_ref'] = maps['fsi'].meta['rsun_ref']
```

[11]:
```
# This is the IAU radius:
from astropy.constants import R_sun
print(R_sun)
```

```
  Name   = Nominal solar radius
  Value  = 695700000.0
  Uncertainty  = 0.0
  Unit  = m
  Reference = IAU 2015 Resolution B 3
```

[12]:
```
# A point on the Sun defined by its helioprojective coordinates as seen from AIA
aia_feature = SkyCoord(800 * u.arcsec, 300 * u.arcsec, frame=maps['aia'].
↪coordinate_frame)
aia_feature
```

[12]:
```
<SkyCoord (Helioprojective: obstime=2022-04-02T10:00:09.349, rsun=695700.0 km,
observer=<HeliographicStonyhurst Coordinate (obstime=2022-04-02T10:00:09.349,
```

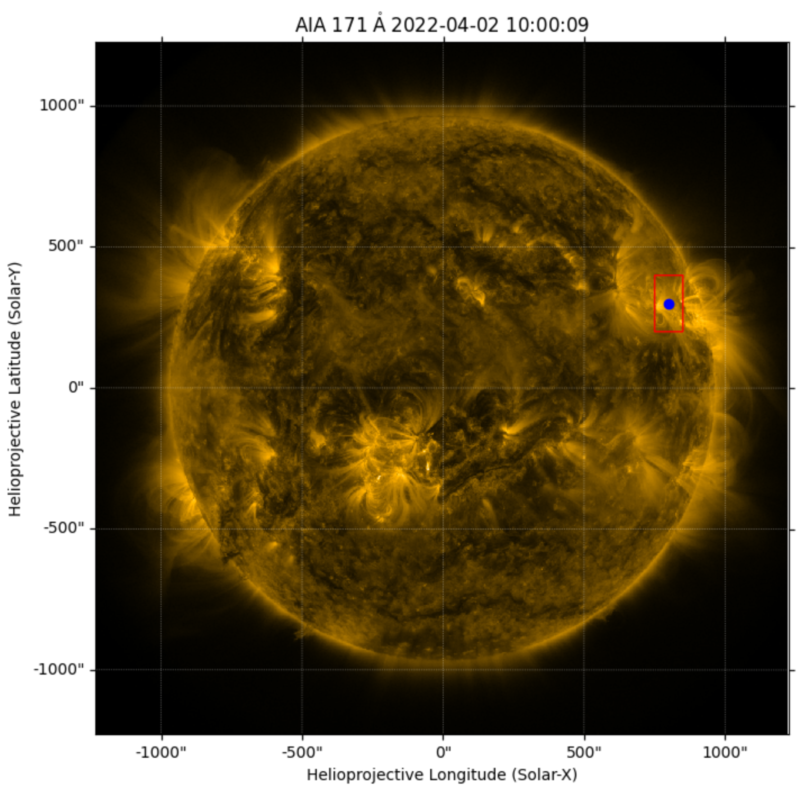
```
rsun=695700.0 km): (lon, lat, radius) in (deg, deg, m)
    (0.00717063, -6.46924727, 1.49547319e+11)>): (Tx, Ty) in arcsec
    (800., 300.)>
```

[13]:
```
# Corners of a rectangle in the same coordinates frame
aia_region = {
    'bottom_left': SkyCoord(750 * u.arcsec, 200 * u.arcsec, frame=maps['aia'].
 ↪coordinate_frame),
    'top_right': SkyCoord(850 * u.arcsec, 400 * u.arcsec, frame=maps['aia'].
 ↪coordinate_frame),
}
```

Plot point and region on AIA map.

[14]:
```
fig = plt.figure()
ax = fig.add_subplot(projection=maps['aia'])
maps['aia'].plot(axes=ax)
ax.plot_coord(aia_feature, 'bo')
maps['aia'].draw_quadrangle(**aia_region, edgecolor='r')
```

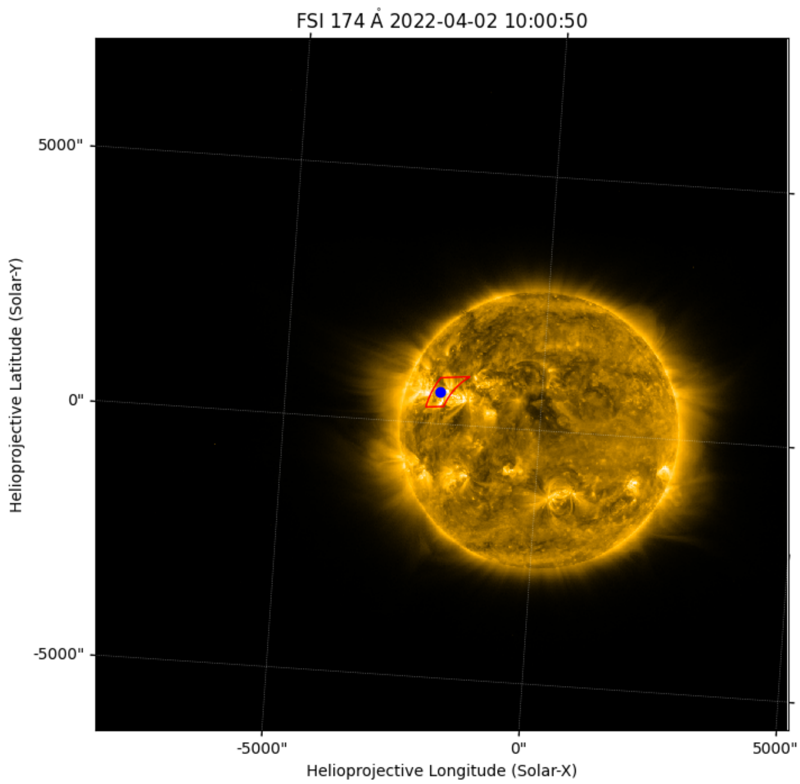
[14]: <astropy.visualization.wcsaxes.patches.Quadrangle at 0x7f004c2be140>



Plot same point and region on FSI map. Coordinate transforms are handled automagically.

[15]:
```
fig = plt.figure()
ax = fig.add_subplot(projection=maps['fsi'])
maps['fsi'].plot(axes=ax)
ax.plot_coord(aia_feature, 'bo')
maps['fsi'].draw_quadrangle(**aia_region, edgecolor='r')
```

[15]: `<astropy.visualization.wcsaxes.patches.Quadrangle at 0x7f004c2620b0>`



1.4 Reproject AIA map to FSI point of view

SunPy gallery:

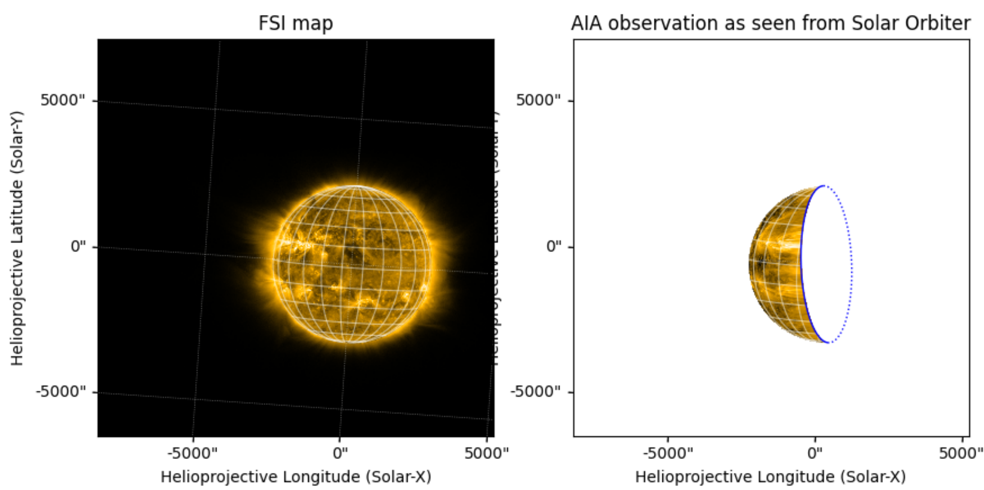
- Reprojecting images to different observers
- Reprojecting using a spherical screen

[16]:
```python
def do_plot(map1, map2, map1to2):
    'Plot map1, and map2 reprojected to the map1 view'
    fig = plt.figure(figsize=(10,5))
    ax1 = fig.add_subplot(1, 2, 1, projection=map2)
    map2.plot(axes=ax1, title='FSI map')
    map2.draw_grid(color='w', system='carrington')

    ax2 = fig.add_subplot(1, 2, 2, projection=map2)
    map1to2.plot(axes=ax2, title='AIA observation as seen from Solar Orbiter')
    map1.draw_grid(color='w', system='carrington')
    map1.draw_limb(color='blue')
```

Assuming a reprojection to the solar sphere (default):

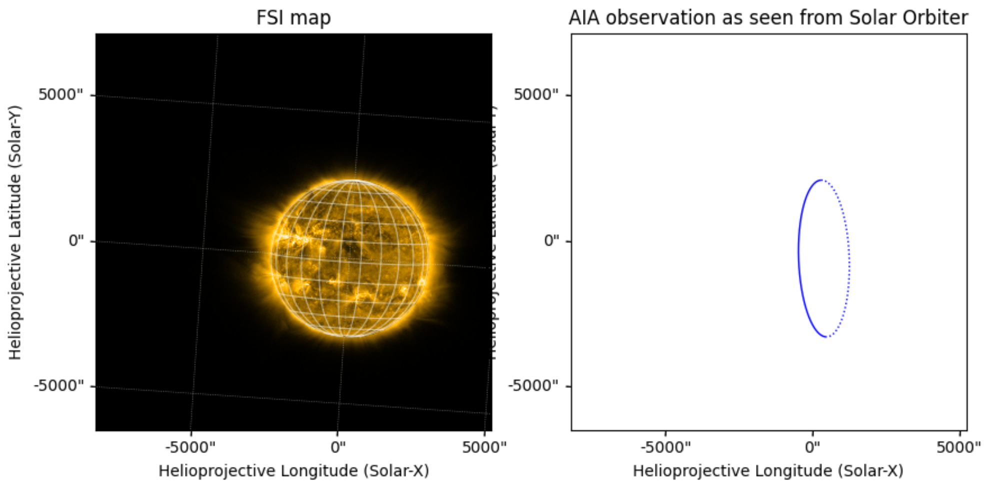
[17]:
```python
aia_viewed_from_fsi = maps['aia'].reproject_to(maps['fsi'].wcs)
# adding algorithm='adaptive' uses the DeForest (2004) algorithm:
# https://reproject.readthedocs.io/en/stable/celestial.html#adaptive-resampling
do_plot(maps['aia'], maps['fsi'], aia_viewed_from_fsi)
```



Assuming reprojection to the celestial sphere:

[18]:
```python
with Helioprojective.assume_spherical_screen(maps['aia'].observer_coordinate):
    aia_viewed_from_fsi = maps['aia'].reproject_to(maps['fsi'].wcs)
do_plot(maps['aia'], maps['fsi'], aia_viewed_from_fsi)
```

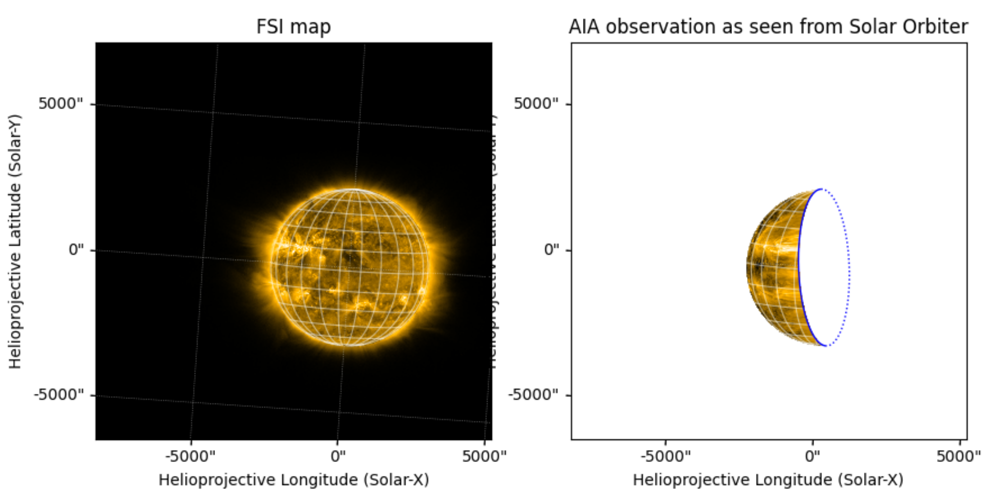
```
/home/eric/.local/lib/python3.10/site-
packages/astropy/visualization/mpl_normalize.py:180: RuntimeWarning: invalid
value encountered in divide
  np.true_divide(values, self.vmax - self.vmin, out=values)
```



We don't see the celestial sphere projection because the angle with Earth is $> 90°$ (unlike in SunPy example).

Assuming reprojection to the celestial sphere off-disk, on the solar sphere on-disk:

[19]:
```
with Helioprojective.assume_spherical_screen(maps['aia'].observer_coordinate,␣
 ↪only_off_disk=True):
    aia_viewed_from_fsi = maps['aia'].reproject_to(maps['fsi'].wcs)
do_plot(maps['aia'], maps['fsi'], aia_viewed_from_fsi)
```



Same issue here (off-disk), because of Earth-Sun-Solar Orbiter angle >90°. Better try FSI from

Mercury view (69.1°).

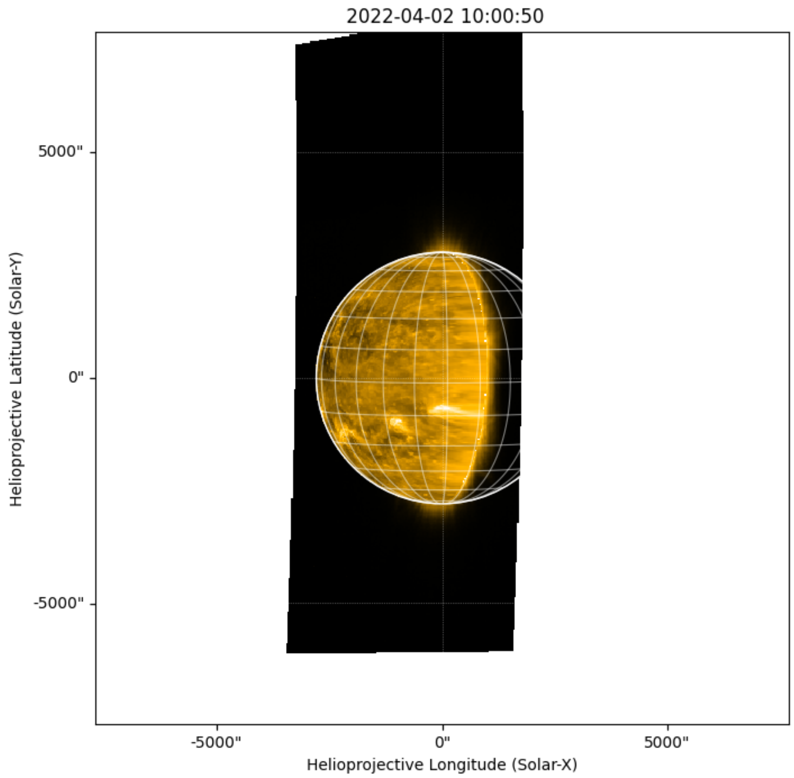
[20]:
```
mercury = get_body_heliographic_stonyhurst('mercury', maps['fsi'].date)
mercury
```

[20]:
```
<HeliographicStonyhurst Coordinate (obstime=2022-04-02T10:00:50.611,
rsun=695700.0 km): (lon, lat, radius) in (deg, deg, AU)
    (177.53816958, 2.26650076, 0.34376972)>
```

[21]:
```
# center solar disk in Mercury-based helioprojective coordinates
mercury_ref_coord = SkyCoord(
    0*u.arcsec, 0*u.arcsec,
    obstime=maps['fsi'].reference_coordinate.obstime,
    observer=mercury,
    rsun=maps['fsi'].reference_coordinate.rsun,
    frame="helioprojective"
)
header = make_fitswcs_header(
    (512, 512),
    mercury_ref_coord,
    scale=[30, 30] * u.arcsec / u.pix,
)
with Helioprojective.assume_spherical_screen(maps['fsi'].observer_coordinate,␣
 ↪only_off_disk=True):
    fsi_viewed_from_mercury = maps['fsi'].reproject_to(header)
```

[ ]:
```
plt.figure()
fsi_viewed_from_mercury.plot()
fsi_viewed_from_mercury.draw_grid()
fsi_viewed_from_mercury.draw_limb()
```

[ ]:
```
(<matplotlib.patches.Circle at 0x7f00456c1a50>, None)
```



1.5 Build Carrington map from AIA and FSI

SunPy gallery:

- Creating Carrington maps
- Creating a full-Sun map with AIA and EUVI

[27]:
```
shape_out = (720, 1440)
```

[28]:
```
# Check reference solar radius (again)
for i in maps:
    print(i, maps[i].meta['rsun_ref'])
```

```
aia 696000000.0
fsi 695700000
hri 695700000
```

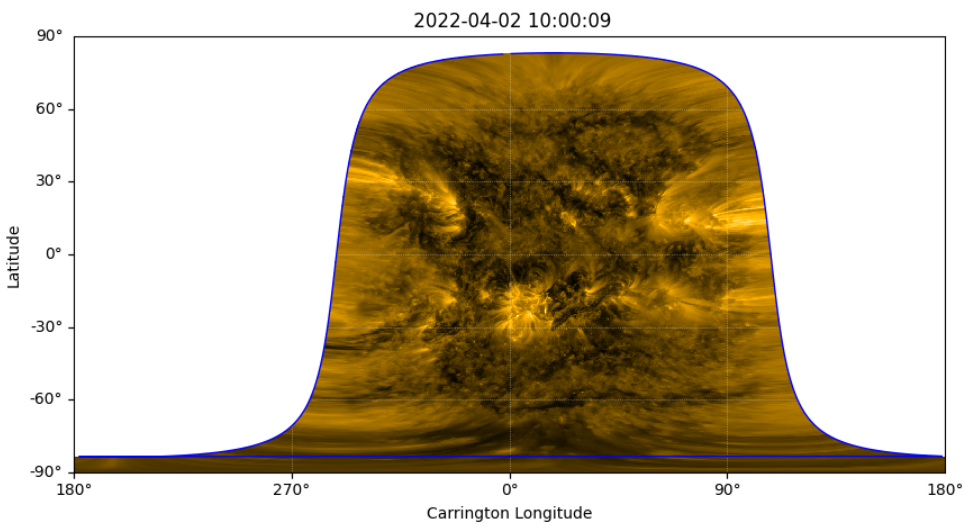
[29]:
```
# Back to the original AIA reference solar radius (or not; now confused about␣
 ↪what should be done)
maps['aia'].meta['rsun_ref'] = 695700000 # original_aia_rsun
```

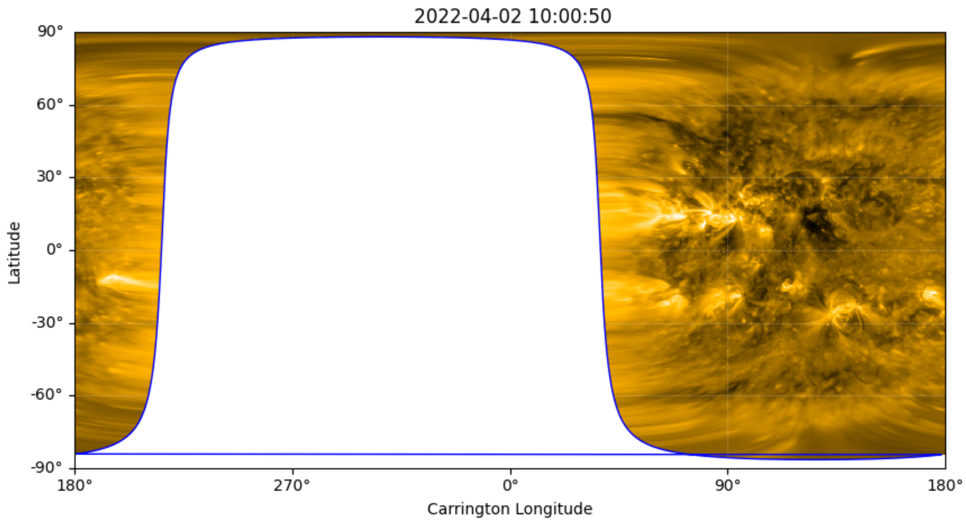
```
for i in maps:
    print(i, maps[i].meta['rsun_ref'])
```

```
aia 695700000
fsi 695700000
hri 695700000
```

[30]:

```
for instrument in ['aia', 'fsi']:
    header = make_heliographic_header(maps[instrument].date, maps[instrument].
 ↪observer_coordinate, shape_out, frame='carrington')
    carr_map = maps[instrument].reproject_to(WCS(header))
    plt.figure(figsize=(10,6))
    carr_map.plot()
    maps[instrument].draw_limb(color='b')
```





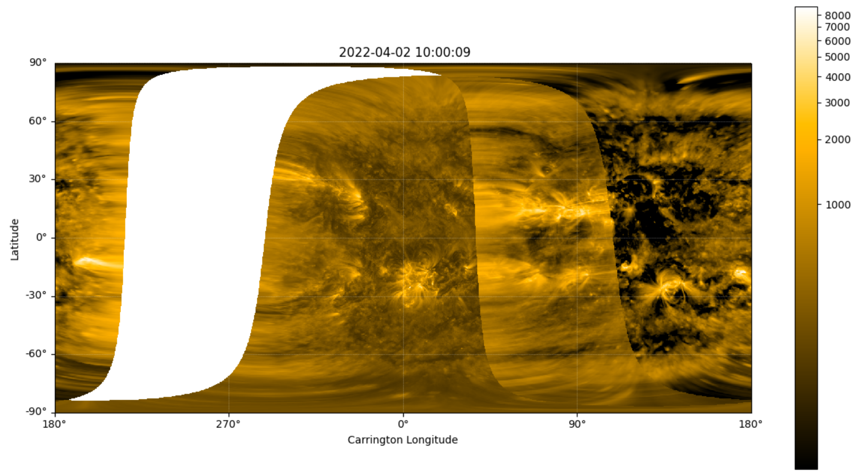
Now combine both projected images into one map.

[31]:
```
header = make_heliographic_header(maps['aia'].date, maps['aia'].
  ↪observer_coordinate, shape_out, frame='carrington')
array, footprint = reproject_and_coadd(
    [maps[instrument] for instrument in ['aia', 'fsi']],
    WCS(header),
    shape_out,
    reproject_function=reproject_interp,
    match_background=True, background_reference=0
)
```

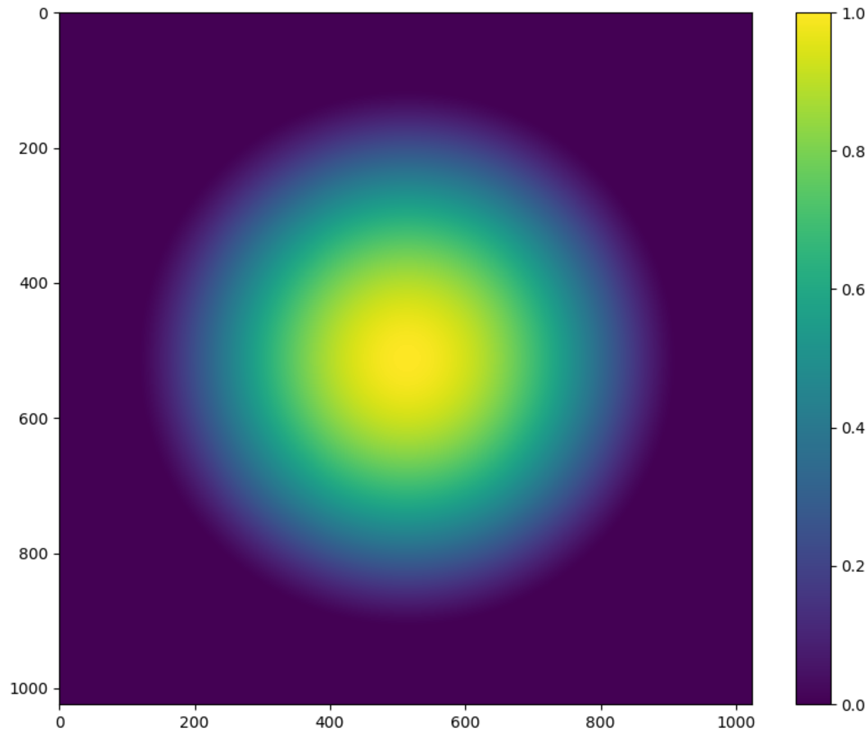
```
WARNING: SunpyUserWarning: The conversion of these 2D helioprojective
coordinates to 3D is all NaNs because off-disk coordinates need an additional
assumption to be mapped to calculate distance from the observer. Consider using
the context manager `Helioprojective.assume_spherical_screen()`.
[sunpy.coordinates.frames]
WARNING: SunpyUserWarning: The conversion of these 2D helioprojective
coordinates to 3D is all NaNs because off-disk coordinates need an additional
assumption to be mapped to calculate distance from the observer. Consider using
the context manager `Helioprojective.assume_spherical_screen()`.
[sunpy.coordinates.frames]
```

[32]:
```
carr_map = Map((array, header))
carr_map.plot_settings = maps['aia'].plot_settings
fig = plt.figure(figsize=(15, 8))
```

```
ax = fig.add_subplot(projection=WCS(header))
im = carr_map.plot(axes=ax, vmin=100)
plt.colorbar(im, ax=ax)
plt.show()
```



[33]:
```
# Same with weights
coordinates = tuple(map(all_coordinates_from_map, [maps[i] for i in ['aia',␣
↪'fsi']]))
weights = [coord.transform_to("heliocentric").z.value for coord in coordinates]
weights = [(w / np.nanmax(w)) ** 3 for w in weights]
for w in weights:
    w[np.isnan(w)] = 0
plt.figure()
plt.imshow(weights[0])
plt.colorbar()
plt.show()
```

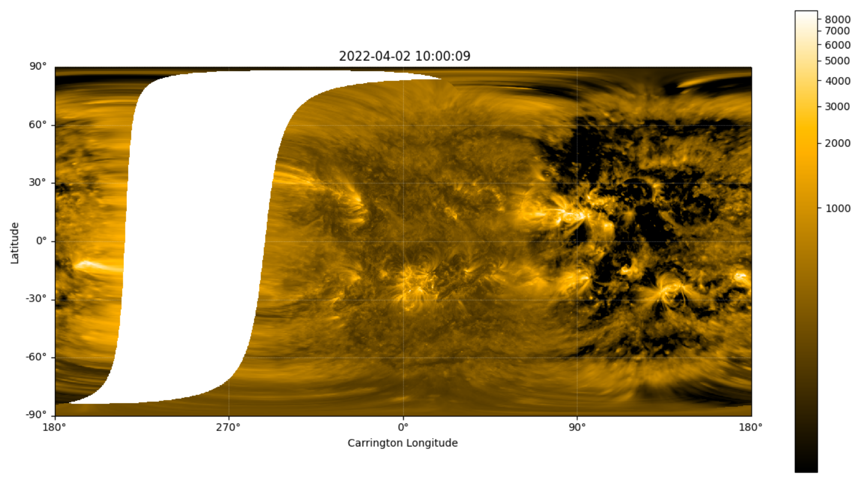


[34]:
```
header = make_heliographic_header(maps['aia'].date, maps['aia'].
 ↪observer_coordinate, shape_out, frame='carrington')
array, footprint = reproject_and_coadd(
    [maps[instrument] for instrument in ['aia', 'fsi']],
    WCS(header),
    shape_out,
    input_weights=weights,
    reproject_function=reproject_interp,
    match_background=True, background_reference=0
)
```

```
WARNING: SunpyUserWarning: The conversion of these 2D helioprojective
coordinates to 3D is all NaNs because off-disk coordinates need an additional
assumption to be mapped to calculate distance from the observer. Consider using
the context manager `Helioprojective.assume_spherical_screen()`.
[sunpy.coordinates.frames]
WARNING: SunpyUserWarning: The conversion of these 2D helioprojective
coordinates to 3D is all NaNs because off-disk coordinates need an additional
assumption to be mapped to calculate distance from the observer. Consider using
```

the context manager `Helioprojective.assume_spherical_screen()`.
[sunpy.coordinates.frames]

[35]:
```
carr_map = Map((array, header))
carr_map.plot_settings = maps['aia'].plot_settings
fig = plt.figure(figsize=(15, 8))
ax = fig.add_subplot(projection=WCS(header))
im = carr_map.plot(axes=ax, vmin=100)
plt.colorbar(im, ax=ax)
plt.show()
```



1.6 Plot SPICE FOV on FSI

[36]:
```
raster = read_spice_l2_fits(str(download_dir / 'spice.fits'))
```

```
WARNING: FITSFixedWarning: CROTA = 3.42495827694 / [deg] S/C counter-clockwise
roll rel to Solar N
keyword looks very much like CROTAn but isn't. [astropy.wcs.wcs]
WARNING: FITSFixedWarning: 'datfix' made the change 'Set MJDREF to 59671.427508
from DATEREF.
Set MJD-OBS to 59671.427508 from DATE-OBS.
Set MJD-BEG to 59671.427508 from DATE-BEG.
Set MJD-AVG to 59671.432366 from DATE-AVG.
Set MJD-END to 59671.437223 from DATE-END'. [astropy.wcs.wcs]
```

[37]:
```
raster
```

[37]:
```
<ndcube.ndcollection.NDCollection object at 0x7f0045f307c0>
NDCollection
------------
Cube keys: ('Mg IX 706 - Peak', 'N IV 765 - Peak', 'Ne VIII 770 - Peak', 'Ly-
gamma-CIII group (Merged)', 'Ly Beta 1025 (Merged)', 'O VI 1032 - Peak')
Number of Cubes: 6
Aligned dimensions: [1.0 830.0 160.0] pix
Aligned physical types: [('time',), ('custom:pos.helioprojective.lon',
'custom:pos.helioprojective.lat'), ('custom:pos.helioprojective.lon', 'time',
'custom:pos.helioprojective.lat')]
```

[38]:
```
window = raster['Ne VIII 770 - Peak']
window
```

[38]:
```
<sunraster.spectrogram.SpectrogramCube object at 0x7f0045307d30>
SpectrogramCube
---------------
Time Period: ['2022-04-02 10:15:39.204' '2022-04-02 10:29:33.954']
Instrument axes: ['raster scan' 'spectral' 'slit' 'slit step']
Pixel dimensions: [  1  50 830 160] pix
Longitude range: [-2389.18369022 -1699.93853085] arcsec
Latitude range: [ 115.3313573  1061.94038124] arcsec
Spectral range: [7.67734135e-08 7.72512468e-08] m
Data unit: adu
```

[39]:
```
# Use rebin to create intensity map from spectrogram (data cube)
intensity = window.rebin((1,50,1,1))[0, 0, :, :]
# bug: y axis becomes incorrect when binning over y?
maps['spice'] = Map((intensity.data, intensity.meta))
```

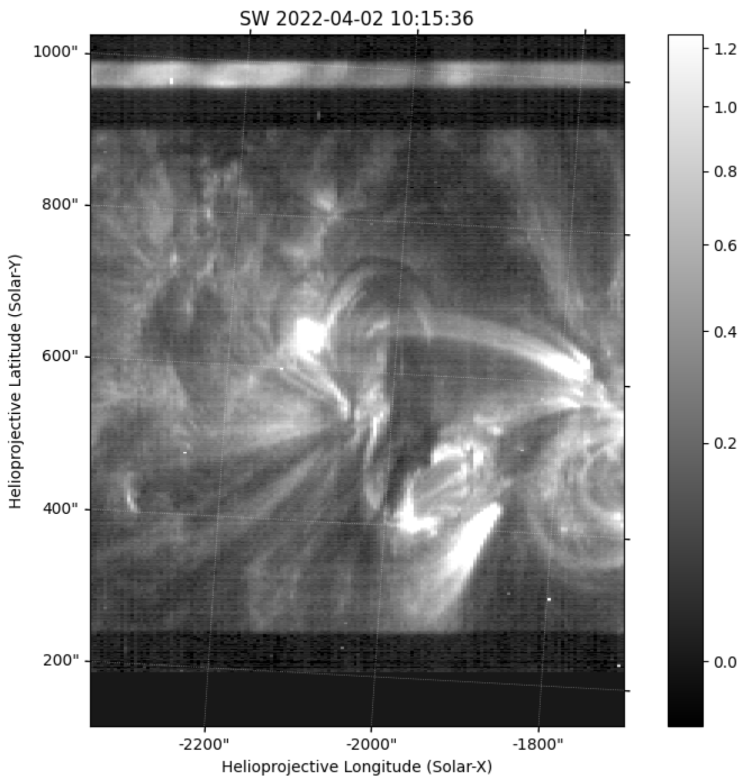
[40]:
```
intensity
```

[40]:
```
<sunraster.spectrogram.SpectrogramCube object at 0x7f00452997b0>
SpectrogramCube
---------------
Time Period: ['2022-04-02 10:15:39.204' '2022-04-02 10:29:33.954']
Instrument axes: None
Pixel dimensions: [830 160] pix
Longitude range: [-2389.18369022 -1699.93853085] arcsec
Latitude range: [ 115.3313573  1061.94038124] arcsec
Spectral range: None
Data unit: adu
```

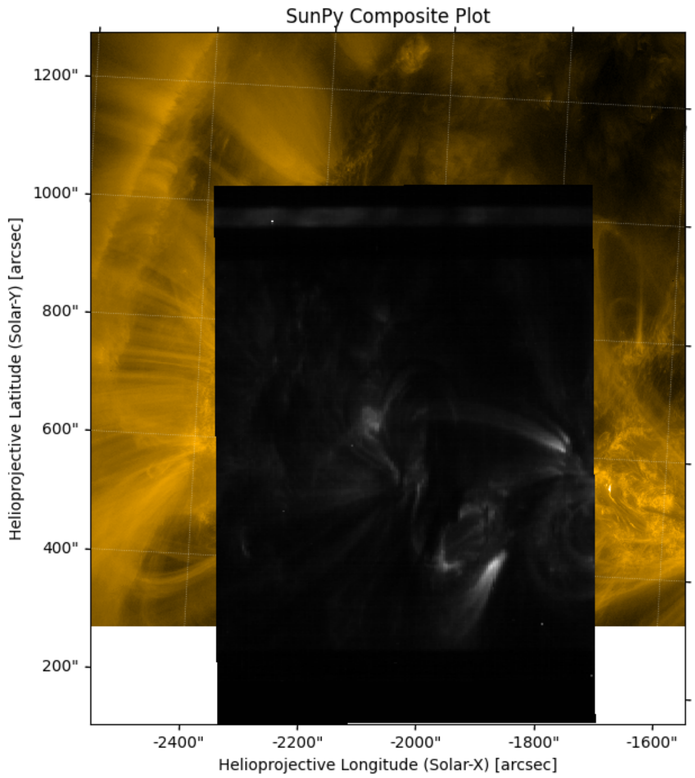
[41]:
```
# For better image value normalization
from astropy.visualization import SqrtStretch, AsymmetricPercentileInterval,␣
↪ImageNormalize
norm = ImageNormalize(
```

```
    intensity.data,
    interval=AsymmetricPercentileInterval(1, 99),
    stretch=SqrtStretch()
)
```

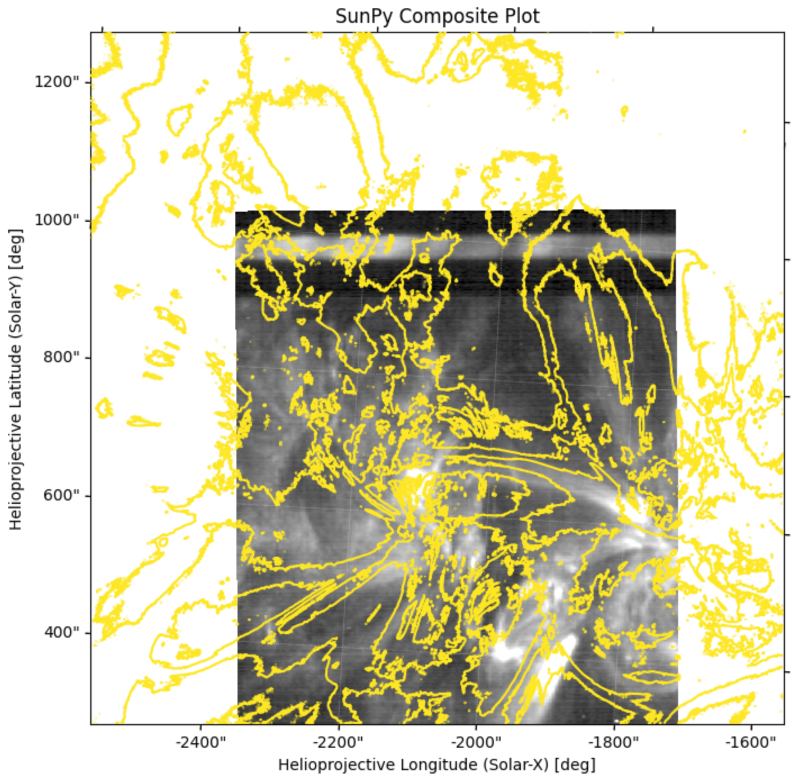
[42]:
```
plt.figure()
maps['spice'].plot(norm=norm, aspect=1/4)
plt.colorbar()
plt.show()
```



[43]:
```
# Composite map (need to understand how to apply value normalization to SPICE␣
↪image)
comp_map = Map(maps['hri'], maps['spice'], composite=True)
plt.figure()
comp_map.plot()
plt.show()
```



[44]:
```
# Composite map with HRI contours
map_spice2hri = maps['spice'].reproject_to(maps['hri'].wcs)
comp_map = Map(map_spice2hri, maps['hri'], composite=True)
comp_map.set_levels(index=1, levels=[0, 1000, 2000, 5000]*u.ct/u.s)
plt.figure()
comp_map.plot(norm=norm)
plt.show()
```



[45]:
```
from mpl_animators import ArrayAnimatorWCS
from astropy.visualization import AsinhStretch, ImageNormalize
map_sequence = Map(maps['hri'], map_spice2hri, sequence=True)
sequence_array = map_sequence.as_array()
norm = ImageNormalize(vmin=0, vmax=3e4, stretch=AsinhStretch(0.01))

sequence_array[..., 1] *= 4000 # rescale SPICE values to EUI values
```

[46]:
```
m = map_sequence[0]
wcs = WCS(naxis=3)
wcs.wcs.crpix = u.Quantity([0*u.pix] + list(m.reference_pixel))
wcs.wcs.cdelt = [1] + list(u.Quantity(m.scale).value)
wcs.wcs.crval = [0, m._reference_longitude.value, m._reference_latitude.value]
wcs.wcs.ctype = ['index'] + list(m.coordinate_system)
wcs.wcs.cunit = [u.dimensionless_unscaled] + list(m.spatial_units)
wcs.wcs.aux.rsun_ref = m.rsun_meters.to_value(u.m)
print(wcs)
```

```
WCS Keywords
```

```
Number of WCS axes: 3
CTYPE : 'index'  'HPLN-TAN'  'HPLT-TAN'
CRVAL : 0.0  -2145.537942339939  763.9551990562587
CRPIX : 0.0  447.5  479.5
PC1_1 PC1_2 PC1_3  : 1.0  0.0  0.0
PC2_1 PC2_2 PC2_3  : 0.0  1.0  0.0
PC3_1 PC3_2 PC3_3  : 0.0  0.0  1.0
CDELT : 1.0  0.984  0.984
NAXIS : 0  0
```

[47]:
```
wcs_anim = ArrayAnimatorWCS(sequence_array, wcs, [0, 'x', 'y'], norm=norm).
    ↪get_animation()
plt.show()
```