import numpy as np

import matplotlib.pyplot as plt

import sunpy.map

from sunpy.coordinates import frames

from astropy.coordinates import SkyCoord

import astropy.units as u

from astropy.io import fits

from sklearn.cluster import DBSCAN

from reproject import reproject\_interp

from matplotlib.colors import LogNorm

# ==========================================

# 1) FILE PATHS

# ==========================================

aia\_bg\_fits    = "/content/aia.lev1\_uv\_24s.2013-10-26T013807Z.1700.image\_lev1.fits"

aia\_flare\_fits = "/content/aia.lev1\_uv\_24s.2013-10-26T014319Z.1700.image\_lev1.fits"

hmi\_field\_fits = "/content/hmi.B\_720s.20131026\_014800\_TAI.field.fits"

# ==========================================

# 2) LOAD AIA & HMI DATA

# ==========================================

field = fits.getdata(hmi\_field\_fits)

hmi\_map = sunpy.map.Map(hmi\_field\_fits)

aia\_map = sunpy.map.Map(aia\_flare\_fits)

aia\_bg  = sunpy.map.Map(aia\_bg\_fits)

# ==========================================

# 3) FIND FLARE LOCATION, AIA FOOTPOINTS

# ==========================================

flare\_hgs = SkyCoord(1\*u.deg, 4\*u.deg,

                     frame=frames.HeliographicStonyhurst,

                     obstime=aia\_map.date)

flare\_hpc = flare\_hgs.transform\_to(

    frames.Helioprojective(observer=aia\_map.observer\_coordinate)

)

flare\_px = int(aia\_map.world\_to\_pixel(flare\_hpc).x.value)

flare\_py = int(aia\_map.world\_to\_pixel(flare\_hpc).y.value)

zoom = 60

x1, x2 = flare\_px - zoom, flare\_px + zoom

y1, y2 = flare\_py - zoom, flare\_py + zoom

flare\_region = aia\_map.data[y1:y2, x1:x2] - aia\_bg.data[y1:y2, x1:x2]

flare\_region[flare\_region < 0] = 0

flare\_norm = np.uint8((flare\_region / flare\_region.max()) \* 255)

threshold = np.percentile(flare\_norm, 99.5)

bright\_px = np.column\_stack(np.where(flare\_norm > threshold))

clustering = DBSCAN(eps=6, min\_samples=2).fit(bright\_px)

labels = clustering.labels\_

unique = np.unique(labels[labels != -1])

if len(unique) < 2:

    raise ValueError("Could not find two valid footpoints in AIA data.")

fp\_aia\_candidates = []

for lbl in unique[:2]:

    cluster = bright\_px[labels == lbl]

    mean\_pix = np.mean(cluster, axis=0).astype(int)

    fp\_world = aia\_map.pixel\_to\_world((mean\_pix[1] + x1)\*u.pix,

                                      (mean\_pix[0] + y1)\*u.pix)

    fp\_aia\_candidates.append(fp\_world)

# ==========================================

# 4) REPROJECT HMI FIELD -> AIA WCS

# ==========================================

hmi\_field\_map = sunpy.map.Map(field, hmi\_map.meta)

hmi\_field\_reproj\_data, \_ = reproject\_interp(

    hmi\_field\_map, aia\_map.wcs, shape\_out=aia\_map.data.shape

)

hmi\_field\_reproj = sunpy.map.Map(hmi\_field\_reproj\_data, aia\_map.meta)

# ==========================================

# 5) FILTER FOOTPOINTS BY HMI FIELD

# ==========================================

filtered\_fp = []

for wcoord in fp\_aia\_candidates:

    pixcoord = hmi\_field\_reproj.world\_to\_pixel(wcoord)

    x\_pix = int(pixcoord.x.value)

    y\_pix = int(pixcoord.y.value)

    if 0 <= x\_pix < hmi\_field\_reproj.data.shape[1] and \

       0 <= y\_pix < hmi\_field\_reproj.data.shape[0]:

        B\_val = hmi\_field\_reproj.data[y\_pix, x\_pix]

        if B\_val > 400:

            filtered\_fp.append(wcoord)

if len(filtered\_fp) == 2:

    fp1\_world, fp2\_world = filtered\_fp

    print("Using footpoints that satisfy HMI filter (B>400 G).")

else:

    fp1\_world, fp2\_world = fp\_aia\_candidates

    print("Fewer than 2 footpoints met HMI threshold → using raw AIA footpoints.")

# ==========================================

# 6) MONTE CARLO UNCERTAINTY PROPAGATION

# ==========================================

N = 1000

aia\_res = aia\_map.scale.axis1.value  # arcsec/pixel

sigma\_pix = 1.5

sigma\_arcsec = sigma\_pix \* aia\_res

fp1\_tx\_samples = np.random.normal(fp1\_world.Tx.value, sigma\_arcsec, N) \* u.arcsec

fp1\_ty\_samples = np.random.normal(fp1\_world.Ty.value, sigma\_arcsec, N) \* u.arcsec

fp2\_tx\_samples = np.random.normal(fp2\_world.Tx.value, sigma\_arcsec, N) \* u.arcsec

fp2\_ty\_samples = np.random.normal(fp2\_world.Ty.value, sigma\_arcsec, N) \* u.arcsec

fp1\_samples = SkyCoord(Tx=fp1\_tx\_samples, Ty=fp1\_ty\_samples, frame=fp1\_world.frame)

fp2\_samples = SkyCoord(Tx=fp2\_tx\_samples, Ty=fp2\_ty\_samples, frame=fp2\_world.frame)

separations\_arcsec = fp1\_samples.separation(fp2\_samples).to(u.arcsec).value

separations\_Mm = (separations\_arcsec \* 725) / 1000  # Mm

loop\_lengths\_Mm = (np.pi \* separations\_Mm) / 2

fp\_dist\_Mm   = np.mean(separations\_Mm)

fp\_dist\_std  = np.std(separations\_Mm)

loop\_len\_Mm  = np.mean(loop\_lengths\_Mm)

loop\_len\_std = np.std(loop\_lengths\_Mm)

# ==========================================

# 7) CREATE SUBMAP AROUND ONE FOOTPOINT

# ==========================================

subradius = 120 \* u.arcsec

bottom\_left = SkyCoord(fp1\_world.Tx - subradius,

                       fp1\_world.Ty - subradius,

                       frame=aia\_map.coordinate\_frame)

top\_right   = SkyCoord(fp1\_world.Tx + subradius,

                       fp1\_world.Ty + subradius,

                       frame=aia\_map.coordinate\_frame)

aia\_sub       = aia\_map.submap(bottom\_left=bottom\_left, top\_right=top\_right)

hmi\_field\_sub = hmi\_field\_reproj.submap(bottom\_left=bottom\_left,

                                        top\_right=top\_right)

# ==========================================

# 8) PLOTTING (Linear Scale Only)

# ==========================================

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(1, 1, 1, projection=aia\_sub)

# Plot AIA and HMI

aia\_sub.plot(axes=ax, cmap='hot', title=False)

hmi\_field\_sub.plot(axes=ax, cmap='Blues', alpha=0.1, title=False)

# Footpoint markers

ax.plot\_coord(fp1\_world, marker='o', color='red', markersize=14,

              markeredgecolor='black', markeredgewidth=1.5, label='Footpoint 1')

ax.plot\_coord(fp2\_world, marker='o', color='blue', markersize=14,

              markeredgecolor='black', markeredgewidth=1.5, label='Footpoint 2')

# Loop-length annotation

ax.text(0.02, 0.02,

        f"Loop Length ≈ {loop\_len\_Mm:.2f} ± {loop\_len\_std:.2f} Mm",

        transform=ax.transAxes,

        fontsize=16,

        color='white',

        bbox=dict(facecolor='black', alpha=0.6, boxstyle='round,pad=0.3'))

# Legend & axis styling

leg = ax.legend(loc='upper right', fontsize=16, facecolor='white')

leg.get\_frame().set\_alpha(0.9)

ax.set\_xlabel("Solar X (arcsec)", fontsize=18, fontweight='bold')

ax.set\_ylabel("Solar Y (arcsec)", fontsize=18, fontweight='bold')

ax.tick\_params(axis='both', labelsize=16, width=1.3)

ax.coords[0].set\_format\_unit(u.arcsec)

ax.coords[1].set\_format\_unit(u.arcsec)

plt.tight\_layout()

plt.savefig("flare\_footpoints\_comparison\_linear.png", dpi=600, bbox\_inches='tight')

plt.show()

# ==========================================

# 9) PRINT MEASUREMENTS

# ==========================================

print(f"\nFootpoint separation: {fp\_dist\_Mm:.2f} ± {fp\_dist\_std:.2f} Mm")

print(f"Loop length (semicircle): {loop\_len\_Mm:.2f} ± {loop\_len\_std:.2f} Mm")

(Use the code below when the solar flare is near the limb of the sun)

import numpy as np

import matplotlib.pyplot as plt

import sunpy.map

from sunpy.coordinates import frames

from astropy.coordinates import SkyCoord

from sunpy.coordinates import Heliocentric

import astropy.units as u

from astropy.io import fits

from sklearn.cluster import DBSCAN

from reproject import reproject\_interp

# ==========================================

# 1) FILE PATHS

# ==========================================

aia\_bg\_fits    = "/content/aia.lev1\_uv\_24s.2013-10-28T043007Z.1700.image\_lev1.fits"

aia\_flare\_fits = "/content/aia.lev1\_uv\_24s.2013-10-28T044119Z.1700.image\_lev1.fits"

hmi\_field\_fits = "/content/hmi.B\_720s.20131028\_044800\_TAI.field.fits"

# ==========================================

# 2) LOAD AIA & HMI DATA

# ==========================================

field = fits.getdata(hmi\_field\_fits)

hmi\_map = sunpy.map.Map(hmi\_field\_fits)

aia\_map = sunpy.map.Map(aia\_flare\_fits)

aia\_bg  = sunpy.map.Map(aia\_bg\_fits)

# ==========================================

# 3) FIND FLARE LOCATION, AIA FOOTPOINTS

# ==========================================

flare\_hgs = SkyCoord(71\*u.deg, 8\*u.deg,

                     frame=frames.HeliographicStonyhurst,

                     obstime=aia\_map.date)

flare\_hpc = flare\_hgs.transform\_to(

    frames.Helioprojective(observer=aia\_map.observer\_coordinate)

)

flare\_px = int(aia\_map.world\_to\_pixel(flare\_hpc).x.value)

flare\_py = int(aia\_map.world\_to\_pixel(flare\_hpc).y.value)

zoom = 120

x1, x2 = flare\_px - zoom, flare\_px + zoom

y1, y2 = flare\_py - zoom, flare\_py + zoom

flare\_region = aia\_map.data[y1:y2, x1:x2] - aia\_bg.data[y1:y2, x1:x2]

flare\_region[flare\_region < 0] = 0

flare\_norm = np.uint8((flare\_region / flare\_region.max()) \* 255)

threshold = np.percentile(flare\_norm, 99.5)

bright\_px = np.column\_stack(np.where(flare\_norm > threshold))

clustering = DBSCAN(eps=6, min\_samples=2).fit(bright\_px)

labels = clustering.labels\_

unique = np.unique(labels[labels != -1])

if len(unique) < 2:

    raise ValueError("Could not find two valid footpoints in AIA data.")

fp\_aia\_candidates = []

for lbl in unique[:2]:

    cluster = bright\_px[labels == lbl]

    mean\_pix = np.mean(cluster, axis=0).astype(int)

    fp\_world = aia\_map.pixel\_to\_world((mean\_pix[1] + x1)\*u.pix,

                                      (mean\_pix[0] + y1)\*u.pix)

    fp\_aia\_candidates.append(fp\_world)

# ==========================================

# 4) REPROJECT HMI FIELD -> AIA WCS

# ==========================================

hmi\_field\_map = sunpy.map.Map(field, hmi\_map.meta)

hmi\_field\_reproj\_data, \_ = reproject\_interp(

    hmi\_field\_map, aia\_map.wcs, shape\_out=aia\_map.data.shape

)

hmi\_field\_reproj = sunpy.map.Map(hmi\_field\_reproj\_data, aia\_map.meta)

# ==========================================

# 5) FILTER FOOTPOINTS BY HMI FIELD

# ==========================================

filtered\_fp = []

for wcoord in fp\_aia\_candidates:

    pixcoord = hmi\_field\_reproj.world\_to\_pixel(wcoord)

    x\_pix = int(pixcoord.x.value)

    y\_pix = int(pixcoord.y.value)

    if 0 <= x\_pix < hmi\_field\_reproj.data.shape[1] and \

       0 <= y\_pix < hmi\_field\_reproj.data.shape[0]:

        B\_val = hmi\_field\_reproj.data[y\_pix, x\_pix]

        if B\_val > 400:

            filtered\_fp.append(wcoord)

if len(filtered\_fp) == 2:

    fp1\_world, fp2\_world = filtered\_fp

    print("Using footpoints that satisfy HMI filter (B>400 G).")

else:

    fp1\_world, fp2\_world = fp\_aia\_candidates

    print("Fewer than 2 footpoints met HMI threshold → using raw AIA footpoints.")

# ==========================================

# 6) MONTE CARLO UNCERTAINTY PROPAGATION with 3D Correction

# ==========================================

N = 1000

aia\_res = aia\_map.scale.axis1.value  # arcsec per pixel

sigma\_pix = 1.5

sigma\_arcsec = sigma\_pix \* aia\_res

# Generate perturbed coordinates

fp1\_tx = np.random.normal(fp1\_world.Tx.value, sigma\_arcsec, N) \* u.arcsec

fp1\_ty = np.random.normal(fp1\_world.Ty.value, sigma\_arcsec, N) \* u.arcsec

fp2\_tx = np.random.normal(fp2\_world.Tx.value, sigma\_arcsec, N) \* u.arcsec

fp2\_ty = np.random.normal(fp2\_world.Ty.value, sigma\_arcsec, N) \* u.arcsec

fp1\_samples = SkyCoord(Tx=fp1\_tx, Ty=fp1\_ty,

                       frame=frames.Helioprojective,

                       obstime=aia\_map.date,

                       observer=aia\_map.observer\_coordinate)

fp2\_samples = SkyCoord(Tx=fp2\_tx, Ty=fp2\_ty,

                       frame=frames.Helioprojective,

                       obstime=aia\_map.date,

                       observer=aia\_map.observer\_coordinate)

# Transform to 3D heliocentric Cartesian

fp1\_3d = fp1\_samples.transform\_to(Heliocentric)

fp2\_3d = fp2\_samples.transform\_to(Heliocentric)

# Filter out off-disk NaNs

valid = ~np.isnan(fp1\_3d.x.value) & ~np.isnan(fp2\_3d.x.value)

fp1\_3d\_valid = fp1\_3d[valid]

fp2\_3d\_valid = fp2\_3d[valid]

if len(fp1\_3d\_valid) < 10:

    raise ValueError("Too many invalid footpoint samples due to off-limb projection.")

# Compute separations and loop lengths

deltas = fp2\_3d\_valid.cartesian - fp1\_3d\_valid.cartesian

sep\_km = deltas.norm().to(u.km).value

sep\_Mm = sep\_km / 1e3

loop\_Mm = (np.pi \* sep\_Mm) / 2

fp\_dist\_Mm   = np.mean(sep\_Mm)

fp\_dist\_std  = np.std(sep\_Mm)

loop\_len\_Mm  = np.mean(loop\_Mm)

loop\_len\_std = np.std(loop\_Mm)

print(f"Valid MC samples used: {len(fp1\_3d\_valid)} / {N}")

# ==========================================

# 7) CREATE SUBMAP AROUND ONE FOOTPOINT

# ==========================================

subradius = 120 \* u.arcsec

bl = SkyCoord(fp1\_world.Tx - subradius,

              fp1\_world.Ty - subradius,

              frame=aia\_map.coordinate\_frame)

tr = SkyCoord(fp1\_world.Tx + subradius,

              fp1\_world.Ty + subradius,

              frame=aia\_map.coordinate\_frame)

aia\_sub       = aia\_map.submap(bottom\_left=bl, top\_right=tr)

hmi\_field\_sub = hmi\_field\_reproj.submap(bottom\_left=bl, top\_right=tr)

# ==========================================

# 8) PLOTTING (Linear Scale Only)

# ==========================================

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(1, 1, 1, projection=aia\_sub)

# Plot AIA & HMI

aia\_sub.plot(axes=ax, cmap='hot', title=False)

hmi\_field\_sub.plot(axes=ax, cmap='Blues', alpha=0.1, title=False)

# Footpoint markers

ax.plot\_coord(fp1\_world, marker='o', color='red', markersize=14,

              markeredgecolor='black', markeredgewidth=1.5, label='Footpoint 1')

ax.plot\_coord(fp2\_world, marker='o', color='blue', markersize=14,

              markeredgecolor='black', markeredgewidth=1.5, label='Footpoint 2')

# Loop-length annotation

ax.text(0.02, 0.02,

        f"Loop Length ≈ {loop\_len\_Mm:.2f} ± {loop\_len\_std:.2f} Mm",

        transform=ax.transAxes,

        fontsize=16,

        color='white',

        bbox=dict(facecolor='black', alpha=0.6, boxstyle='round,pad=0.3'))

# Legend & styling

leg = ax.legend(loc='upper right', fontsize=16, facecolor='white')

leg.get\_frame().set\_alpha(0.9)

ax.set\_xlabel("Solar X (arcsec)", fontsize=18, fontweight='bold')

ax.set\_ylabel("Solar Y (arcsec)", fontsize=18, fontweight='bold')

ax.tick\_params(axis='both', labelsize=16, width=1.3)

ax.coords[0].set\_format\_unit(u.arcsec)

ax.coords[1].set\_format\_unit(u.arcsec)

plt.tight\_layout()

plt.savefig("flare\_footpoints\_3D\_linear.png", dpi=600, bbox\_inches='tight')

plt.show()

# ==========================================

# 9) PRINT FINAL MEASUREMENTS

# ==========================================

print(f"\nFootpoint separation (3D): {fp\_dist\_Mm:.2f} ± {fp\_dist\_std:.2f} Mm")

print(f"Loop length (semicircle, 3D): {loop\_len\_Mm:.2f} ± {loop\_len\_std:.2f} Mm")