Copilot

I see, thank you for the clarification. You're using the convention commonly used in mathematics and physics, where `theta` is the azimuthal angle in the xy-plane, and `phi` is the polar angle from the positive z-axis. This is sometimes referred to as the physics convention for spherical coordinates.

In this case, we can adjust the rotation matrices accordingly. Here's the updated code:

```python
from scipy.spatial.transform import Rotation as R

def project_ellipsoid(pts, center, theta_e, phi_e, theta_c, phi_c):
    # Translate the ellipsoid to the origin
    pts -= center

    # Rotate the ellipsoid to its desired orientation
    r = R.from_euler('ZY', [theta_e, -phi_e], degrees=False)
    pts = r.apply(pts)

    # Rotate the points in the opposite direction of the plane's normal
    r = R.from_euler('ZY', [-theta_c, phi_c], degrees=False)
    pts = r.apply(pts)

    # Project the points onto the xy-plane
    projected_pts = pts[:, :2]

    return projected_pts
```

In this code:

- `R.from_euler('ZY', [theta_e, -phi_e], degrees=False)` creates a `Rotation` object that first rotates around the z-axis by `theta_e` and then around the y-axis by `-phi_e`. This aligns the ellipsoid with the axes.
- `r.apply(pts)` applies the rotation to the points.
- The same process is repeated to rotate the points in the opposite direction of the plane's normal.

This should give you the projected points on the plane. Note that this assumes the plane is initially the xy-plane (z=0), and the ellipsoid's longest axis is initially aligned with the z-axis. If that's not the case, you would need to adjust the rotations accordingly.

Also, please note that the `Rotation` class uses radians for the angles, not degrees. If your angles are in degrees, you can convert them to radians with `np.radians()`, or set `degrees=True` in the `from_euler()` call. 😊