GNSS Performance

RDNAPTRANS

Group 8: Ming-Chieh Hu, Ákos Sárkány, Daan Schlosser, Neelabh Singh

Previously

Simple GNSS receiver: R10
- No IMU
- Accuracy: ± 1cm
- ETRS89

Perimeter corners + Cone outline

Few points on curve for better approximation

Measure points counter-clockwise for easier polygon reconstruction

Measured perimeter of 1 tile

ETRS89

```
Latitude,Longitude,Hoogte
52.0026298876,4.3747085311,43.105
52.0026465629,4.3748142976,44.334
52.0026514618,4.3748120799,44.315
52.0027046607,4.375162861,48.822
52.0027233373,4.3752353904,49.66
52.0026426272,4.3752035241,49.671
52.0026335204,4.3751427343,48.963
52.0023864649,4.3747324572,44.675
52.0023890214,4.3747284827,44.625
52.0022810481,4.3745468766,42.902
52.0022340401,4.3744684739,42.799
52.0021941216,4.3744625202,42.807
52.0021652689,4.3744842162,42.786
52.0022934354,4.3751609472,51.018
52.0023391979,4.3757677706,58.909
52.0023421909,4.3758259366,59.708
52.0023488772,4.3759058604,60.248
52.0023551156,4.3759758258,60.264
```

RD + Ellipsoidal height

```
X,Y,Z
85475.6225455622,446512.740756181,43.105
85482.9106826452,446514.494661003,44.334
85482.7660041226,446515.041786694,44.315
85506.9342544749,446520.624529409,48.822
85511.9433044575,446522.632919219,49.66
85509.6301638007,446513.684392516,49.671
85505.4420180676,446512.729415367,48.963
85476.8879173019,446485.637046995,44.675
85476.6189777484,446485.925261933,44.625
85463.9817660054,446474.087047246,42.902
85458.5254193507,446468.932456184,42.799
85458.0546970858,446464.497217537,42.807
85459.4996786975,446461.266580477,42.786
85506.1654321422,446474.877485314,51.018
85547.9031595539,446479.388205347,58.909
85551.9016938914,446479.665565496,59.708
85557.3999211347,446480.333008794,60.248
85562.2136718239,446480.960148531,60.264
```

RDNAPTRANS™
API

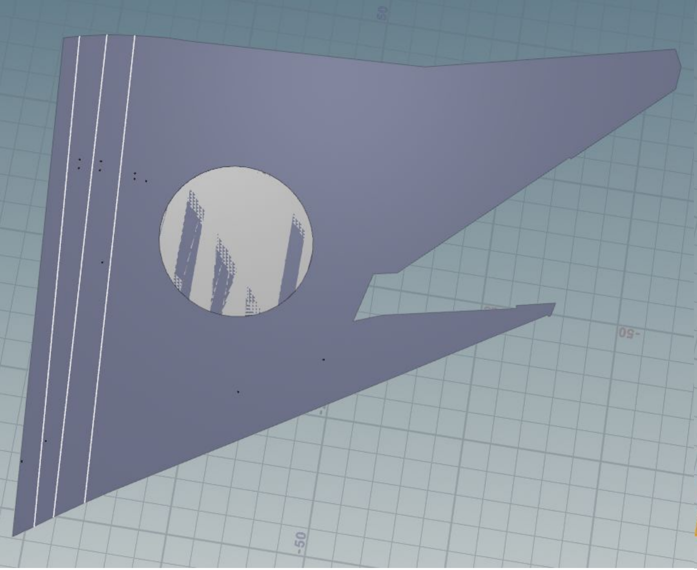
Method 1: https://github.com/GISFabriek/RdNapTransPy
Method 2: https://www.nsgi.nl/coordinatenstelsels-en-transformaties/tools/transformatie-api

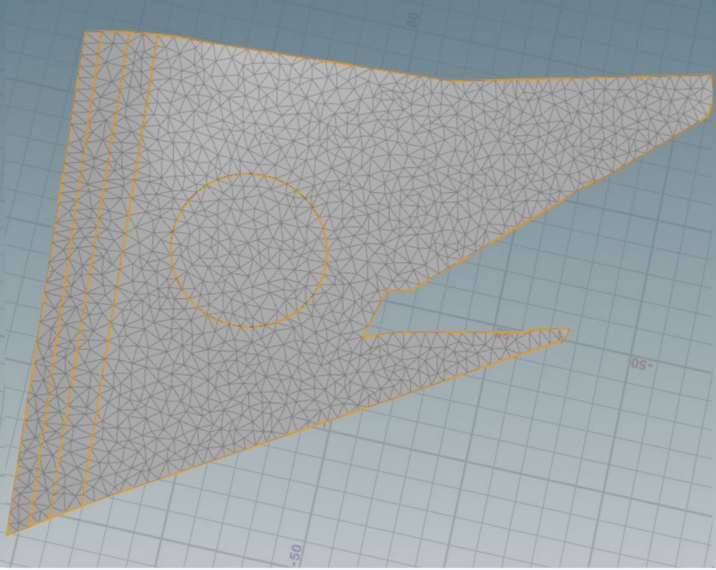
ETRS89

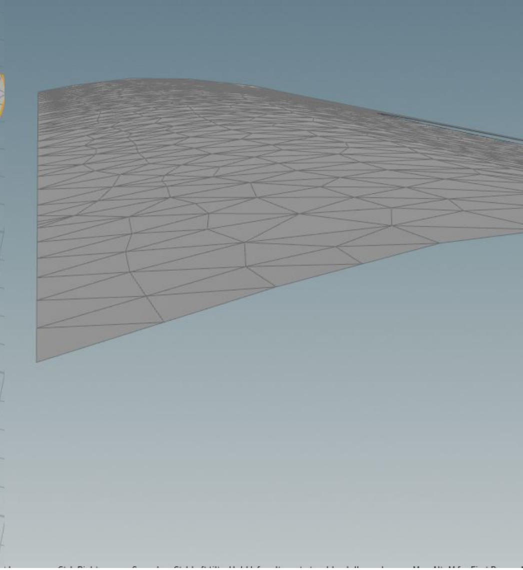
```
Latitude, Longitude, Hoogte
52.0026298876, 4.3747085311, 43.105
52.0026465629, 4.3748142976, 44.334
52.0026514618, 4.3748120799, 44.315
52.0027046607, 4.375162861, 48.822
52.0027233373, 4.3752353904, 49.66
52.0026426272, 4.3752035241, 49.671
52.0026335204, 4.3751427343, 48.963
52.0023864649, 4.3747324572, 44.675
52.0023890214, 4.3747284827, 44.625
52.0022810481, 4.3745468766, 42.902
52.0022340401, 4.3744684739, 42.799
52.0021941216, 4.3744625202, 42.807
52.0021652689, 4.3744842162, 42.786
52.0022934354, 4.3751609472, 51.018
52.0023391979, 4.3757677706, 58.909
52.0023421909, 4.3758259366, 59.708
52.0023488772, 4.3759058604, 60.248
52.0023551156, 4.3759758258, 60.264
```

RD + NAP Height

```
x, y, z
85475.62182732056, 446512.7314697935, -0.39167312733736365
85482.90997852039, 446514.48535755306, 0.8373446346662369
85482.76530001867, 446515.0324807013, 0.8183493083809231
85506.93359797099, 446520.6151613911, 5.325406174753652
85511.94265829158, 446522.6235356882, 6.163425307634745
85509.62950893283, 446513.67504873103, 6.174347222048445
85505.44135498042, 446512.7200811898, 5.466337472526511
85476.88718921754, 446485.6278689116, 1.1780931697974273
85476.61824931653, 446485.916082859, 1.1280955628582197
85463.98100986033, 446474.0779337818, -0.5950113212223521
85458.52465194237, 446468.9233615498, -0.6980578585990721
85458.05392693597, 446464.48814332957, -0.690096371079627
85459.49890941389, 446461.25752393284, -0.7111237737212982
85506.16475282407, 446474.8683021038, 7.5210108322449605
85547.90255981137, 446479.3789368716, 15.412064978309417
85551.90110183199, 446479.65628890257, 16.21106882972434
85557.3993393284, 446480.3237281091, 16.75107659502177
85562.21309864326, 446480.95087095746, 16.767083762831604
```

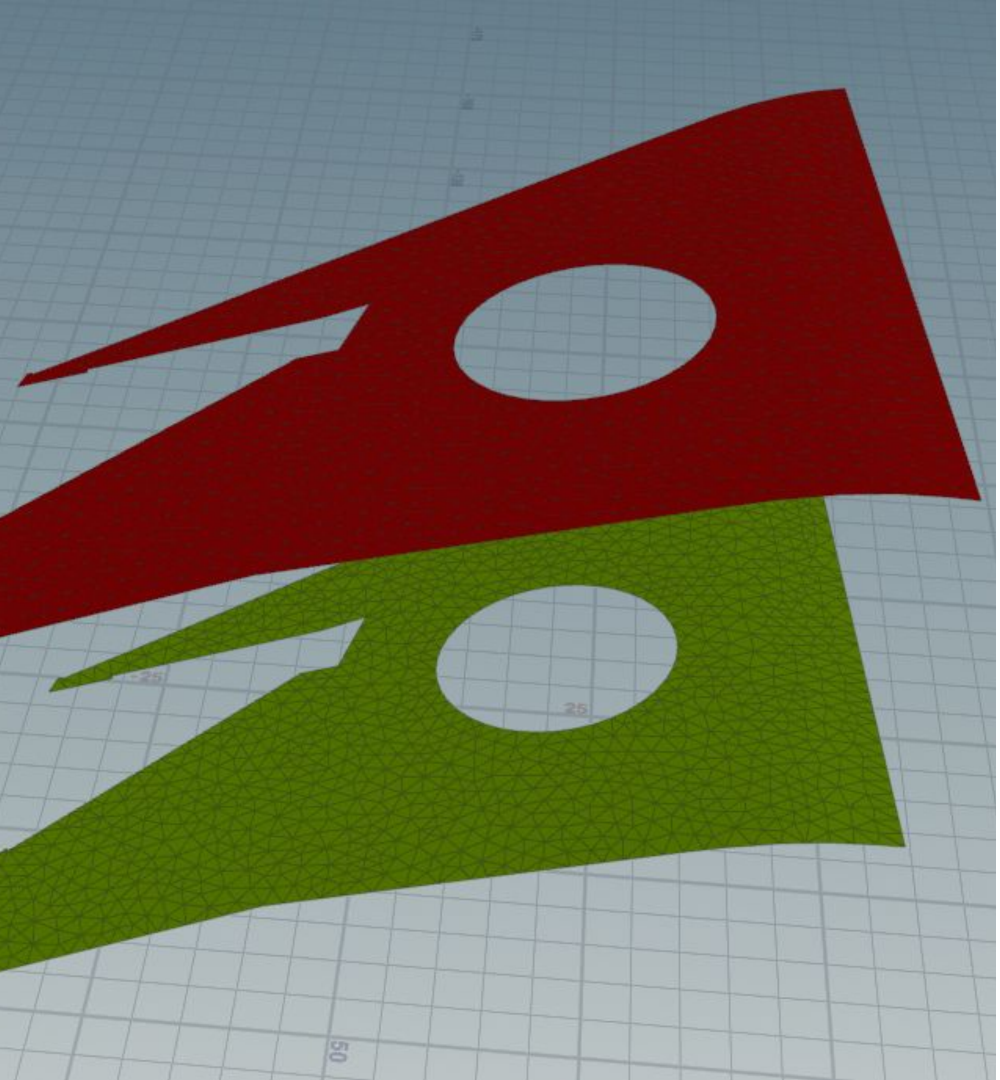






Base polygon, Circle Fitting, & Triangulation

- Project points to EPSG:28992 with RDNAPTRANS™. This way point coordinates are in meters and have the correct height.
- Connect the points on the perimeter to construct a polygon of the surface. Fit an ellipsoid to interior points. Resample to have more points. Add constraining lines along curve
- 1 concrete tile: 2 m^2, total: 128 tiles; Ventilation shaft: 5.87 m^2 → Total area to subtract: 261,87 m^2
- Constrained Triangulation to obtain surface. Smoothing to remove noise.



Results

	Surface Area	Perimeter Length
3D with Ellipsoidal Height	5015.69 m²	517.15 m
3D with NAP Height	4.998,215 m²	515.88
2D Map Projection	4.930,58 m²	511.42 m

Compare

RED:

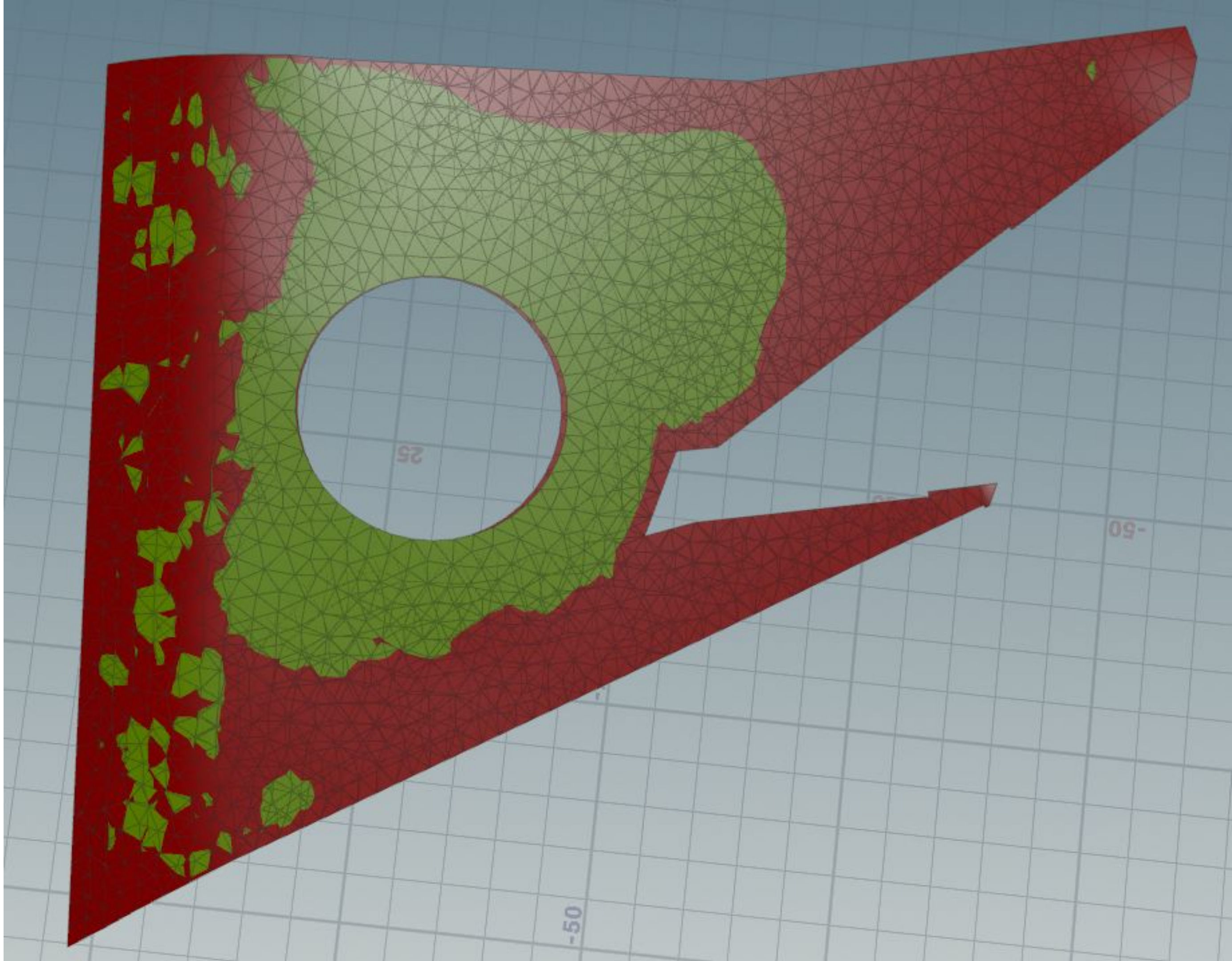
- Ellipsoidal Height

GREEN:

- NAP Height

Main difference is in fitting of oval:

- Exact height
- Exact tangent
- Exact shape



Part 6: Data Transformation: NMEA to RDNAP

Data Extraction	Height Correction	WGS84 to ETRS89	ETRS89 to RDNAP
Extract data from NMEA sentences (EPSG:4979).	Apply height correction* to get ellipsoidal height. (Applied to all the points extracted)	Perform WGS84 to ETRS89 transformation using Python.	Perform ETRS89 to RDNAP transformation using NSGI API.

*ellipsoidal height = orthometric height + geoid height

Height Correction, Issue 1

Converting coordinates using pyproj directly **does not** convert the height.

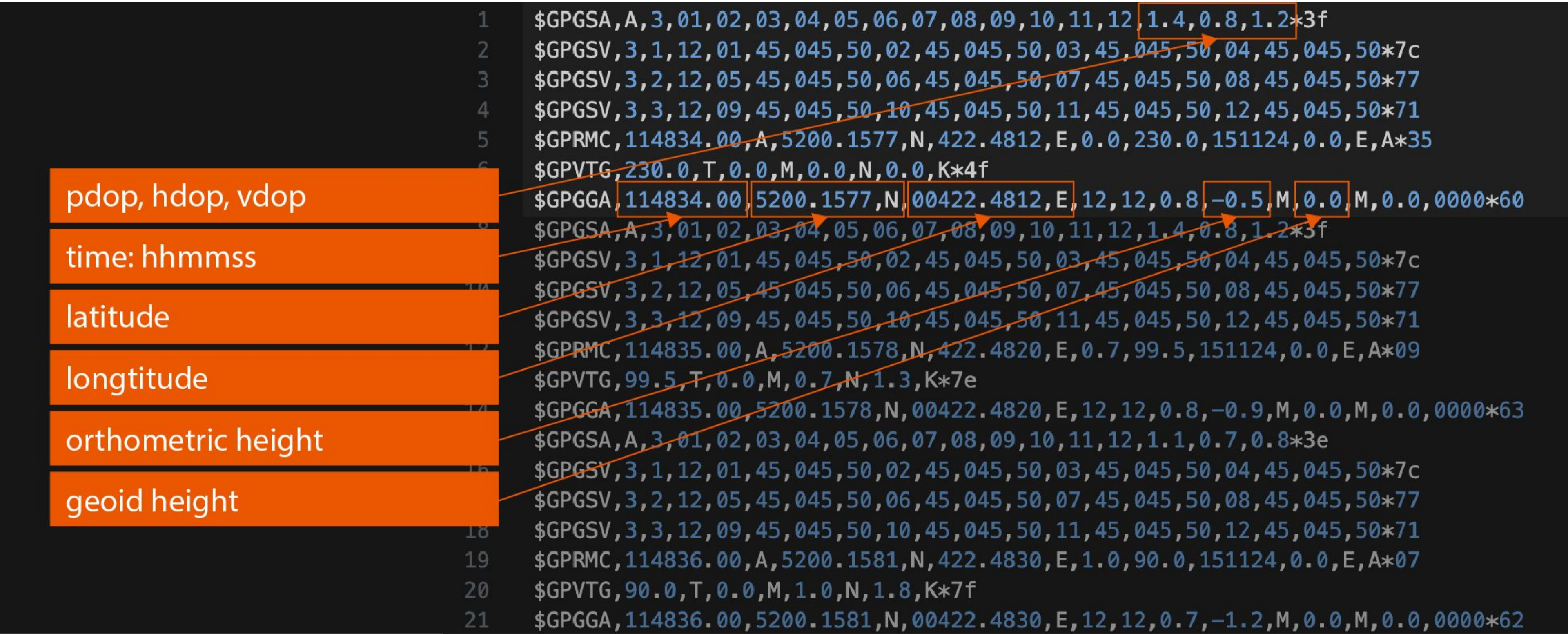
```
# THIS ONE DOES NOT CONVERT HEIGHT!
def transform_coordinates(lon, lat, height):

    transformer = pyproj.Transformer.from_crs("EPSG:4979", "EPSG:7931", always_xy=True)
    x, y, z = transformer.transform(lon, lat, height)
    return x, y, z

# Example input: WGS 84 coordinates (latitude, longitude, height)
point = [4.374686666666666, 52.002628333333334, -0.5]

# Transform the coordinates
x, y, z = transform_coordinates(point[0], point[1], point[2])

# Display the transformed coordinates
print(f"Input (EPSG:4979): Longitude={point[0]}, Latitude={point[1]}, Height={point[2]} meters")
print(f"Output (EPSG:7931): X={x}, Y={y}, Z={z}")
```

✓ 0.1s

```
Input (EPSG:4979): Longitude=4.374686666666666, Latitude=52.002628333333334, Height=-0.5 meters
Output (EPSG:7931): X=4.374686666666666, Y=52.002628333333334, Z=-0.5
```

Height Correction, Issue 2

ETRS89 uses ellipsoidal height, while the GPS data from NMEA sentences only have orthometric height and geoid height. (And for some reason the geoid height is missing)



Height Correction

Perform height correction by:

1. Using a geoid grid file to retrieve geoid height from (x, y) coordination of each point.
2. Apply the formula: orthometric height = ellipsoidal height - geoid height.

```python
def height_correction(point):
    # In a false data, 'height' is both orthometric and ellipsoidal, (h1 = H1, N1 = 0).
    longitude, latitude, orthometric_height = point[0], point[1], point[2]

    # Create a transformer using the geoid grid
    transformer = Transformer.from_pipeline(
        f"+proj=pipeline +step +inv +proj=longlat +ellps=WGS84 "
        f"+step +proj=vgridshift +grids={GEOID_GRID_PATH} "
        "+step +proj=longlat +ellps=WGS84"
    )

    # Transform coordinates
    _, _, correct_orthometric_height = transformer.transform(longitude, latitude, orthometric_height)

    # Correct data, N2 = H1 - H2, h2 = h1 + N2.
    geoid_height = orthometric_height - correct_orthometric_height
    ellipsoidal_height = orthometric_height + geoid_height

    return [longitude, latitude, ellipsoidal_height]
```

WGS84 to ETRS89

WGS84 and ETRS89 are “almost” the same, but there’s still some slight differences due to plate tectonics. Thus we perform plate tectonics corrections (using epoch) here.

```
def wgs84_to_etrs89(point_WGS):
    """Converts WGS84 coordinates (longitude, latitude, height) to ETRS89 coordinates.
    Parameters:
        point_WGS: List or tuple of [longitude, latitude, height].
    Returns:
        List of transformed coordinates [longitude, latitude, height] in ETRS89.
    """
    x, y, z = cartesian_3D_from_lon_lat(point_WGS[0], point_WGS[1], point_WGS[2])
    new_station, _ = ITRF2014_ETRF2014(
        x_coord=x,
        y_coord=y,
        z_coord=z,
        ITRF_epoch=2024.90,
        x_velocity=0,
        y_velocity=0,
        z_velocity=0,
        ETRF_epoch=2024.90,
    )
    long, lat, elev = lon_lat_from_cartesian_3D(new_station[0], new_station[1], new_station[2])
    return [long, lat, elev]
```

*reference: https://medium.com/@calebjuma27/converting-from-wgs84-to-etrs89-b51157d79c70

ETRS89 to RDNAP

Here we use NSGI API to reduce the amount of work while maintaining the accuracy level.

The Coordinate Transformation API performs coordinate transformations using the official RDNAPTRANS™ transformation procedure and other transformations defined or recommended by NSGI.

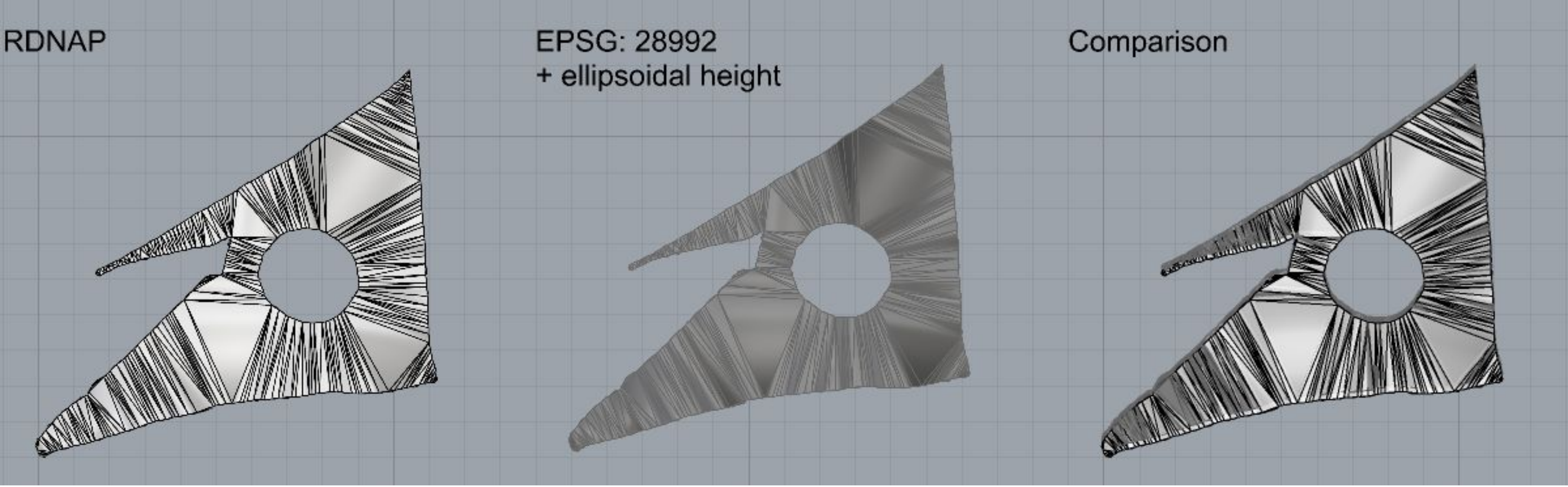
*API documentation: https://api.transformation.nsgi.nl/v2/openapi?f=html

WGS84 + Orthometric Height

```
Longitude,Latitude,Altitude
4.3746866667,52.0026283333,-0.5
4.3747000000,52.0026300000,-0.9
4.3747166667,52.0026350000,-1.2
4.3747316667,52.0026400000,-1.5
4.3747466667,52.0026466667,-1.5
4.3747616667,52.0026516667,-1.6
4.3747750000,52.0026566667,-1.8
4.3747883333,52.0026616667,-2.2
4.3748033333,52.0026650000,-2.4
4.3748116667,52.0026666667,-2.4
4.3748233333,52.0026700000,-2.3
4.3748350000,52.0026750000,-1.9
4.3748416667,52.0026800000,-2.1
4.3748500000,52.0026816667,-1.8
4.3748633333,52.0026850000,-1.6
4.3748766667,52.0026866667,-1.5
```

RD + NAP Height

```
x,y,z
85473.4754,446512.0100,-0.2931
85474.3935,446512.1827,-0.6931
85475.5456,446512.7230,-0.9931
85476.5833,446513.2649,-1.2931
85477.6236,446513.9922,-1.2931
85478.6613,446514.5341,-1.3931
85479.5846,446515.0776,-1.5931
85480.5078,446515.6211,-1.9931
85481.5430,446515.9775,-2.1931
85482.1177,446516.1550,-2.1931
85482.9240,446516.5146,-2.0931
85483.7328,446517.0597,-1.6931
85484.1983,446517.6096,-1.8931
85484.7731,446517.7870,-1.5931
85485.6938,446518.1451,-1.3931
85486.6119,446518.3178,-1.2931
```



	TIN (EPSG: 28992 + ellipsoidal height)	TIN (RDNAP)
2.5D Area (m^2)	5137.5424	5137.4389
2.5D Perimeter (m)	549.2390	549.1011
2D Projected Area (m^2)	4920.0418	4920.0252
2D Projected Perimeter (m)	529.5567	529.5563

Shape, boundaries, area, and precision are all affected by the used CRS

Part 7

Step	3	5	6
Coordinate Reference System	EPSG:28992 (RD New)	ETRS89 (GNSS-RTK to RDNAP)	WGS84 (iPhone GNSS to RDNAP)
Important details	Projected coordinate system, aka a flat 2d representation of Earth	Coordinates are relative to the European tectonic plate	Coordinates are determined from the Earth's centre, and thus does account for tectonic movement
2.5D Area (m^2)	n/a	5137.542	5137.439
2.5D Perimeter (m)	n/a	549.239	549.101
2D Projected Area (m^2)	5115.860	4920.042	4920.025
2D Projected Perimeter (m)	508.170	529.557	529.556

All projections distort in some way. EPSG:28992, since it is specifically made for usage in The Netherlands, will thus have a greater local accuracy. The other two are global systems, so we're assuming they are more distorted locally.

Precision and Idealisation

Is it all about the precision and how you conduct the measurements, or does it extend the object definition of the "grass roof of the library" as well?

Precision: Higher precision captures more accurate details.

- RTK is more suitable for tasks requiring high precision and slope considerations.
- Smartphone GNSS is useful for quick, rough estimates but less reliable for detailed analysis.

The **importance of idealization** in geospatial measurements.

Idealization: Idealization is the process of simplifying a real-world object into a geometric shape or model for easier analysis. Simplifying assumptions about the shape, slope, and boundary also impact the results.

The grass roof of the library has a complex shape with varied slopes, vegetation, and irregular boundaries. To measure and compare it, we must idealize it as a simpler shape (e.g., a polygon with straight edges).

Sources of Error and Idealization

Measurement Precision:

- RTK offers high precision, leading to detailed boundary points.
- Smartphone GNSS has lower precision, introducing noise and reducing accuracy.

Idealisation of Slope and Elevation:

- Ignoring slope (2D) underestimates the true surface area, in PDOK
- Accounting for slope (2.5D) provides a more accurate area but may vary based on how the slope is approximated.

Idealisation of Boundary Definition:

- The real boundary might be unclear (e.g., overgrown grass or uneven edges), leading to variations in how different methods interpret it.

PDOK – Idealized as a flat, projected 2D polygon using orthophoto data.

RTK Data – Detailed, with slope and elevation considered (2.5D polygon).

Smartphone Data – Simplified 2D and 2.5D polygon with less accurate boundaries.

Perimeter Differences: RTK measurements capture more details, leading to a slightly longer perimeter.

Area Differences: 2.5D area is larger due to accounting for the slope.

PDOK vs. Field Measurements: Orthophoto-based measurements (PDOK) might underestimate or overestimate due to resolution limits and lack of slope data.