UAS Positioning and Navigation

Alex Mejlevang Ellegaard - aelle20@student.sdu.dk
Anders Lind-Thomsen - andli20@student.sdu.dk
Peter Skibdal Frydensberg - pefry20@student.sdu.dk
Thomas Therkelsen - ththe20@student.sdu.dk

November 2024

4 Exercises

4.1 Record GNSS track

The GNSS track is recorded using the `hello_drone.py` ROS2 script which has been modified to write its data to a file. In the dataset we can see that the altitude measurements are very wrong going down almost 80 meters from the start of the walk. This might be because of interference from the metal buildings (although we would expect this to also affect the longitude and latitude measurements), it could also be because when between the two buildings the only satellites which can see us is directly above us, which would results in inaccuracies in the latitude measurements, also known as DOP.

4.2 Convert coordinates to UTM

The data is converted from geodetic to UTM with the `transverse_mercator.py` and `utm.py` scripts which have been explained in earlier assignments.

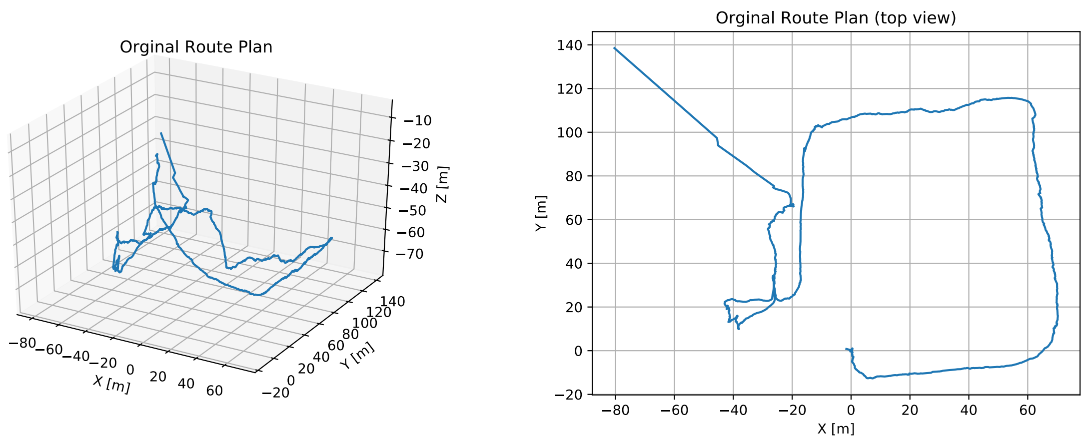

Figure 1: Original data in UTM format

The data in figure 1 is plotted relative to the start position at $x, y = (0, 0)$.

4.3 Remove outliers

We remove outliers in the data by removing any xy-pairs with a speed of above a certain threshold (in our case 50 m/s) with respect to the previous xy-pair. Since we are only comparing two data points at a time, the algorithm will be fooled if there are two noisy measurements in a row where the noise is in the same direction. The algorithm could be made more robust by using more data points.

In order to test the algorithm, a set of 10 random outliers has been added to the data, which can be seen in figure 2. The outlier removal algorithm is then used on the noisy data which results in the plot in figure 3.

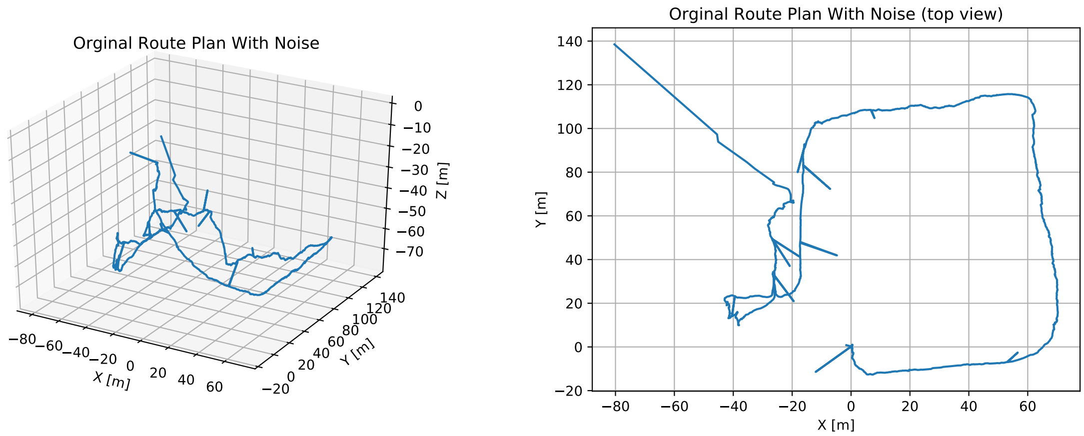

Figure 2: Original data with added outliers

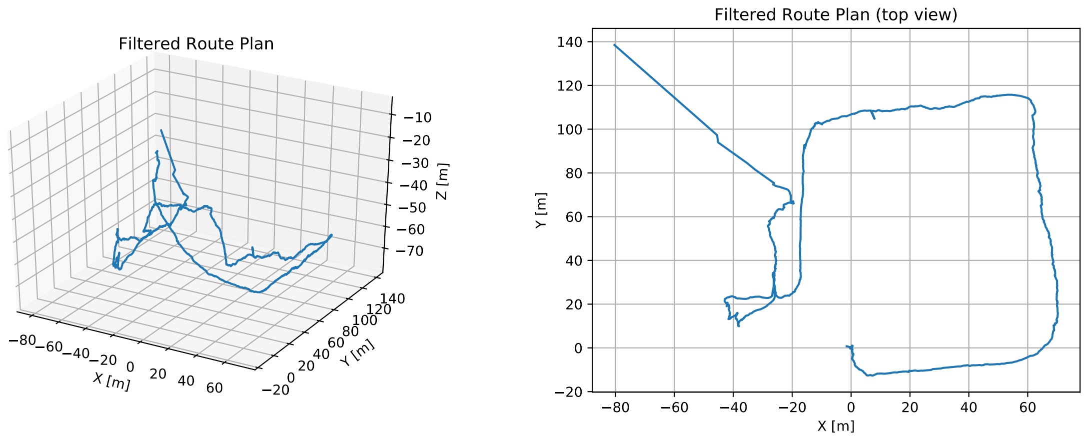

Figure 3: Data after removing outliers

As can be seen, the algorithm removes most of the added outliers except for the one at around (10, 110) in figure 3.

4.4 Simplify the track

To simplify the track so it does not contain the full 5000 entries we use the Ramer–Douglas–Peucker algorithm to reduce the number of points to 100. This algorithm reduces the number of points while keeping the general shape of the track by recursively dividing the track while keeping the points furthest from the linear path segment and removing other insignificant points.

The algorithm can be seen in action in figure 4, where the data has been simplified from over 5000 points to a little over 50.

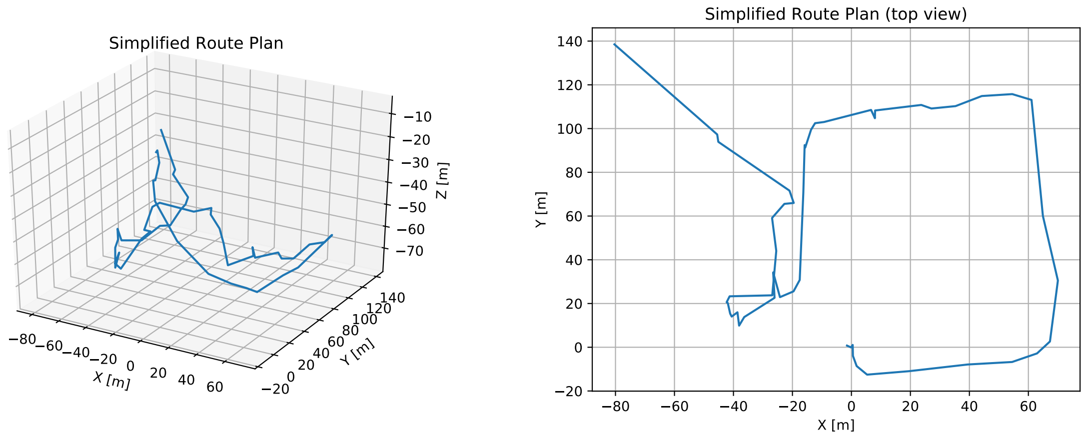

Figure 4: Data after applying the simplification algorithm

The effect of the algorithm can easily be seen on the ridge from (60, 0) to (60, 110) in figure 4 when compared to the same ridge in figure 3.

Additionally, a feature has been added to the python class which outputs the corresponding KML map to the simplifed data so it can be reviewed in google maps. A screenshot from google maps can be seen in figure 6.

Note that the path looks noisy and strange in the left part of the map. This is due to us walking inside Building 44 of SDU Tek with the GPS.

4.5 Convert to geographic coordinates.

The track is converted to geographich coordinates using the same procedure as in lecture 2.

4.6 Export as a route plan to QGroundControl software.

The route plan using the lattitude and longitude data has been made so it can be imported to QGroundControl Software. A screenshot of the imported mission plan is shown in figure ??

Figure 5: Data after smoothing with 3D Cubic Hermetic Splines

4.7 Fixed wing optimization

In order to optimize the route plan for a 3D version of the provided Cubic Hermetic Splines algorithm has been implemented. The simplified data from figure 4 is smoothed and plotted in figure 5.

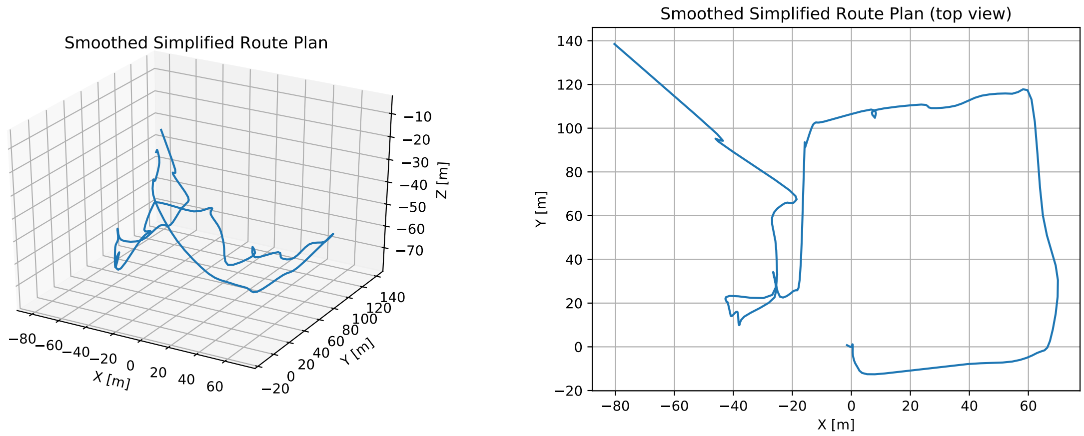

Figure 6: Data after smoothing with 3D Cubic Hermetic Splines

Note that there are very few sharp edges compared to figure 4. However, everything comes at a cost which in this case is having more points (here about 200).