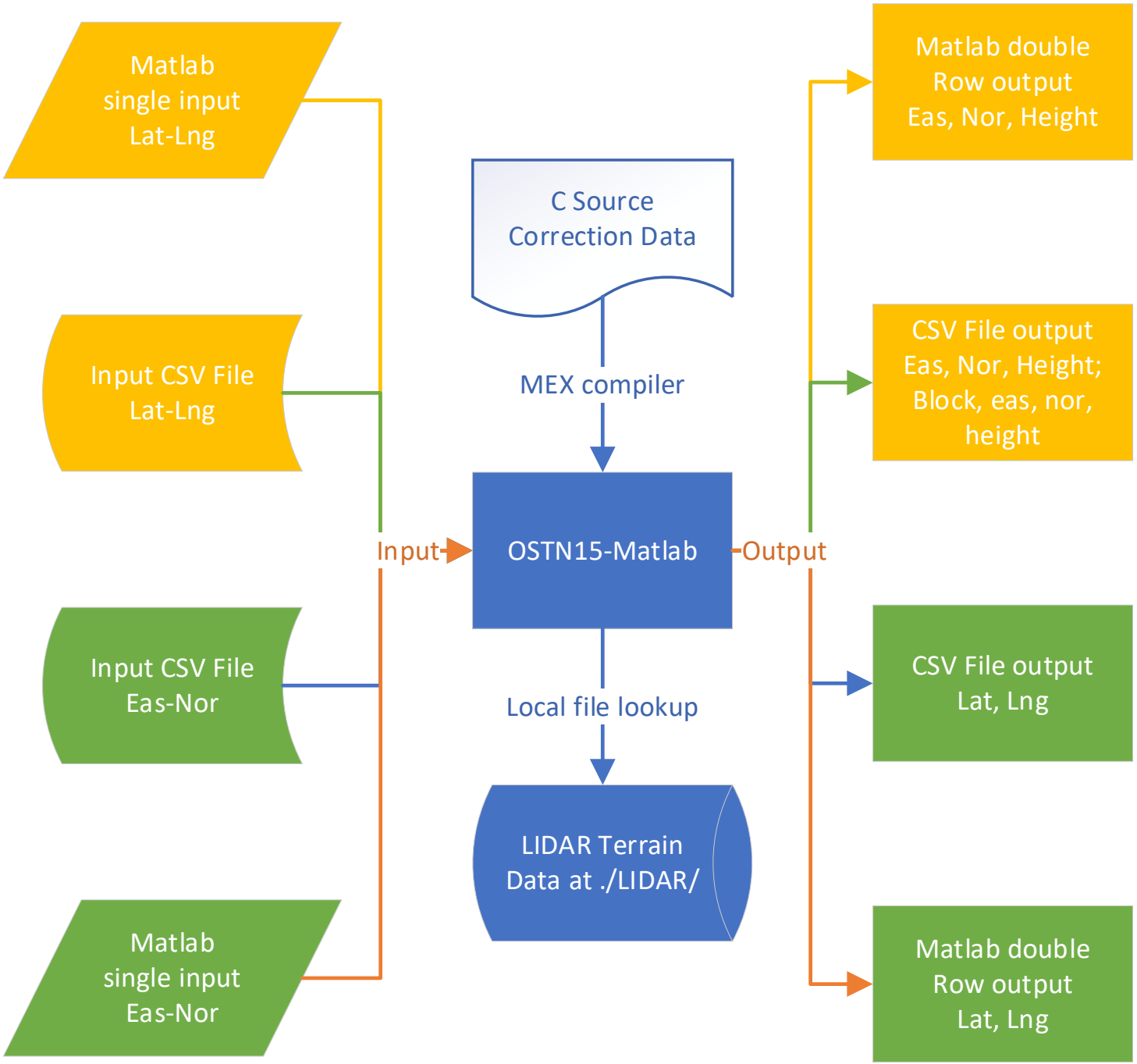
OSTN15-Matlab Conversion Implementation & Altitude Inference

Tony Chen

Goal

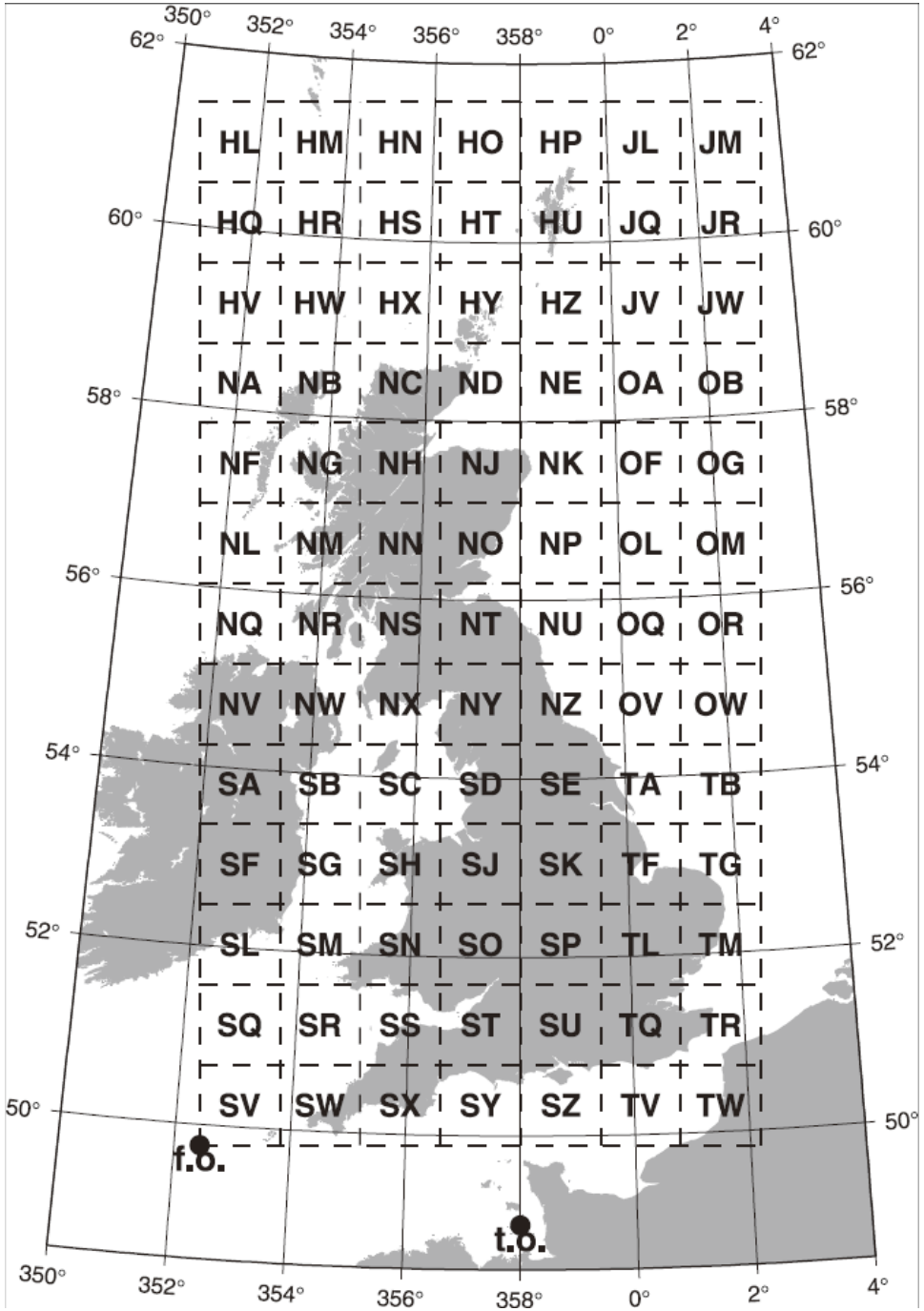
- To replace the awkward and time-demanding QGIS implementation of coordinate conversion from OSGB 1936 to WGS 84
- At the same time, to find out the heights/altitude of a given GPS-acquired route in a easier way
- All methods to be usable at one command from MatLab, trying to be compatible to older versions used by the group
- Finally, by comparing between results, we can form a method of correction for the barometer of the phone, forming intrinsic altitude data

Data Flow


Matlab
single input
Lat-Lng
Input CSV File
Lat-Lng
Input CSV File
Eas-Nor
Matlab
single input
Eas-Nor
C Source
Correction Data
MEX compiler
OSTN15-Matlab
Local file lookup
LIDAR Terrain
Data at ./LIDAR/
Input
Output
Matlab double
Row output
Eas, Nor, Height
CSV File output
Eas, Nor, Height;
Block, eas, nor,
height
CSV File output
Lat, Lng
Matlab double
Row output
Lat, Lng

The OSTN15 Model

- Details of the OSGB1936 System/The National Grid
  - Coordinates are represented by how many metres away from the false origin(f.o.), from the east and north axis, called Eastings and Northings
  - For every 100 kilometre, two letters are assigned for a quicker location
  - Has an geoid(a ellipsoid-or, a sphere-like surface but a little bit flatter) slightly different from the global system(GPS/WGS84)



The OSTN15 Model

- The Model
- 1. Translating between ETRS89 Latitude-Longitude and Easting-Northing
  - Within the kilometre square where the point falls, a bilinear interpolation is used to obtain the exact transformation value for the point from the values at the four corners of the kilometre square. The GRS80 ellipsoid is used, with National Grid projection parameters.
  - The inverse transformation (OSGB36 to ETRS89) is accomplished by an iterative procedure, i.e. repeatedly using the last obtained value to calculate until the value variate only within a given toleration.

The OSTN15 Model

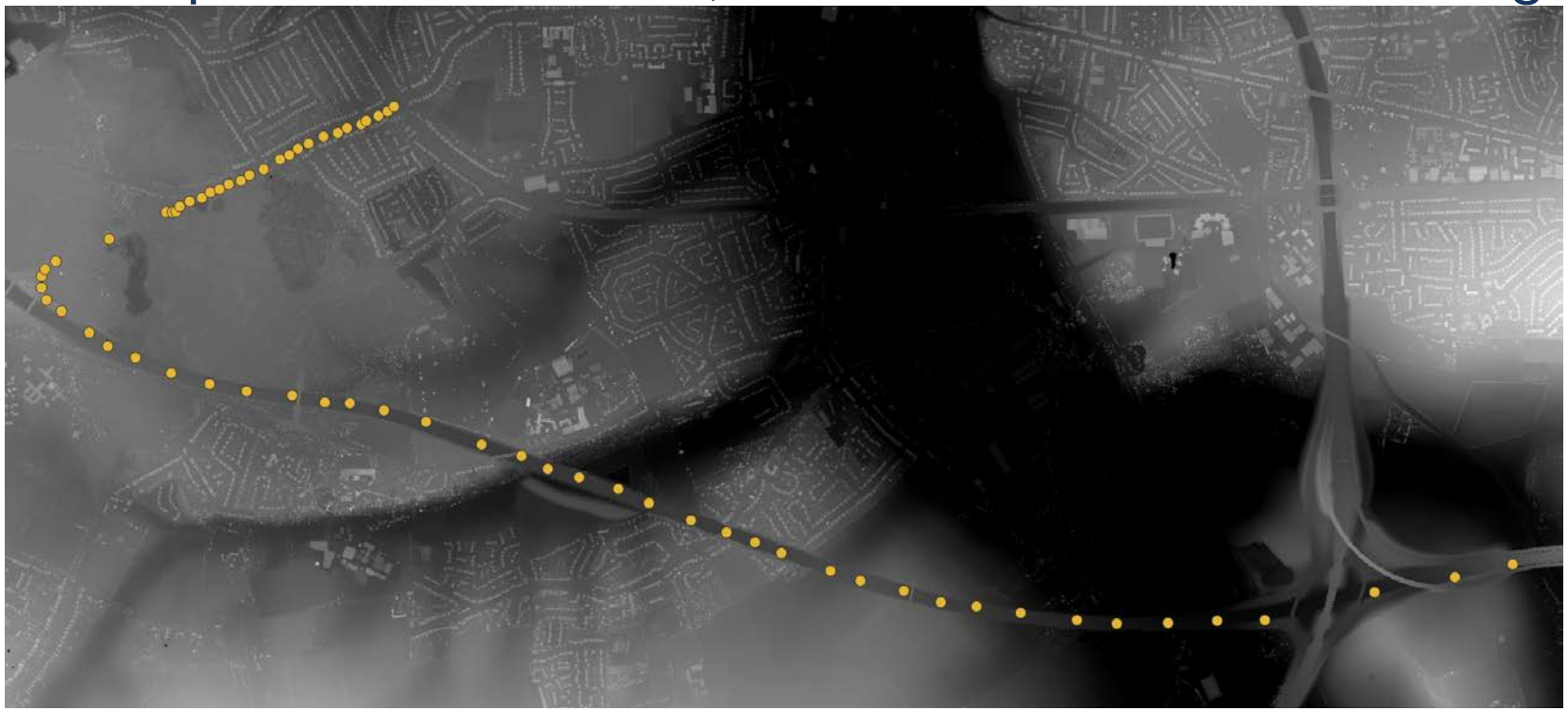
- 2. Convert the ETRS89 Easting-Northing (x, y) obtained to a National Grid Easting-Northing (e, n)
  - The easting and northing shifts from a given data file are added to the x and y coordinates, respectively. This works as an average shift between its nearest neighbors, i.e. an interpolation between them.
  - To translate back, an iterative method is still used, repeatedly calculating using the interpolation method above until the value changes within a given tolerance, since the interpolation information is typically unknown.
- Accuracy: Within Great Britain, 0.1m

The OSTN15 Model

- Accuracy
  - We Selected a series of points on a highway in Block TQ57, By Google Earth, and got its GPS/WGS84 Coordinates.

The OSTN15 Model

- Accuracy
  - We input the converted OSGB36 Data towards QGIS 3.2, above the HSM Data picture from LIDAR, to see if the routes fits the highway.

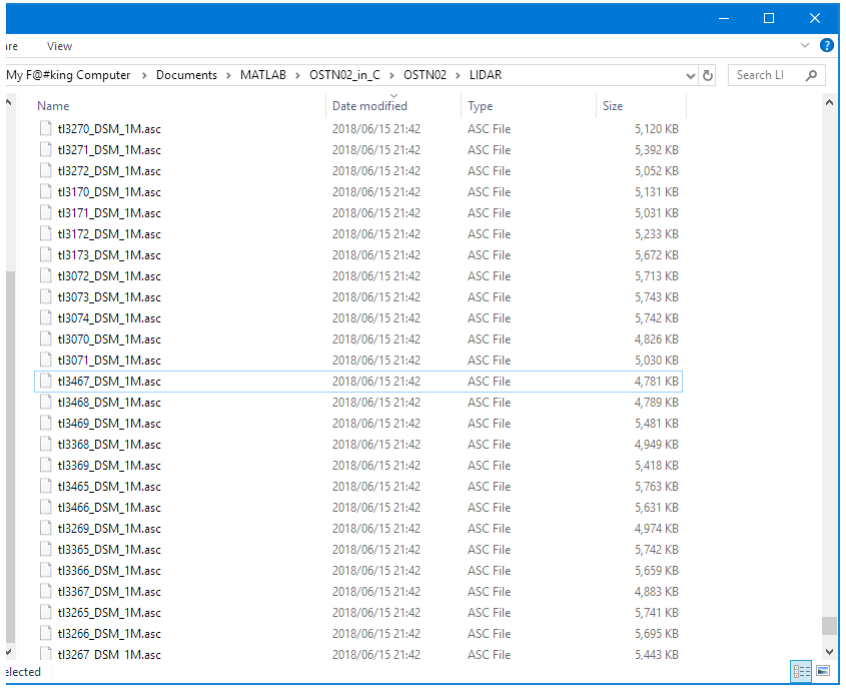


The LIDAR DSM Data

- LIDAR is an airborne mapping technique, which measures the height of the terrain and surface objects on the ground through the use of a scanning laser.
- The DSM Data reserves ground objects such as bridges and buildings, while the DTM Data, another type of data that the LIDAR holds, does not, causing trouble when a truck crosses a bridge.
- The Environment Agency holds datasets in resolutions of 50cm, 1m and 2m; 1m is used, concerning size and miss rate concerns.
- Accuracy: Vertical 5cm, Horizontal 40cm

The LIDAR DSM Data

- Loading files for lookup
  - Files are saved in the ‘LIDAR’ directory inside the whole package
  - Current data covers a route shown in the next page
  - Run the package first, and by checking error messages you will know what parts of data are needed to be downloaded
  - Data at http://environment.data.gov.uk/ds/survey/#/ ,after download, decompress at the LIDAR directory



Implementation

- Original code is a series of C files.
- MatLab has an API usable by C, and has a compiler (called **mex**) to convert C files into a MatLab executable binary library (ending with .mexw64 in 64-bit Windows systems), then this C implementation can be called and executed.

Implementation

- The program provides conversion for both ways.
  - WGS84/GPS/ETRS89 Latitude & Longitude → OSGB36 Easting & Northing:
    - Can be used to transfer a single set of coordinates.
    - Can also load a file of a series of coordinates, and output two files in the working directory of the Matlab Library.
      - Raw output file with easting and northing simple numerical values only. This can be loaded by any GIS map;
      - Refined output file with the double-letter block the point within, first offset digit, direction within the 10km block, and an offset inside double-letter blocks, finally height. These sets of information helps locate what file to use in LIDAR.

Implementation

- OSGB36 Easting & Northing → WGS84/GPS/ETRS89 Latitude & Longitude:
  - Can be used to transfer a single set of coordinates.
  - Can also load a file of a series of coordinates, and output one file in the working directory of the Matlab Library.
    - Raw output file with easting and northing simple numerical values only. This can be loaded by any GIS map. Reverse conversion will not offer height data, since this is estimated to be not quite useful.

Using OSTN15-Matlab

- How to Use?
  - For conversion between one set of Latitude-Longitude/Easting-Northing: [a, b, c] = OSTN15_Matlab('gps-to-grid',lat,lon) or [a,b] = OSTN15_Matlab('grid-to-gps',eas,nor)
  - For conversion between batches of Latitude-Longitude/Easting-Northing: OSTN15_Matlab('gps-to-grid'/'grid-to-gps', filename)
  - If you ever have no clues how to use: type OSTN15_Matlab directly, and a guide and a test program will be run.
- To check for height the LIDAR data should be present. Otherwise a -9999 will show as a result.

Using OSTN15-Matlab

- Input & Output Format

realGpsForTong.csv × realGPSForTong-Raw.csv

```
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26338
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
51.473,0.26339
```

Input: Lat-Lon

realGPSForTong-Raw.csv × realGpsForTong.csv

```
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557290.365381,177321.865016,5.567000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
557291.059812,177321.886484,4.737000
```

Raw output: Eas, Nor, Elevation

realGPSForTong-Output.csv × realGPSForTong-Raw.csv

```
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57290.365381,77321.865016,5.567000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
TQ57NE,57291.059812,77321.886484,4.737000
```

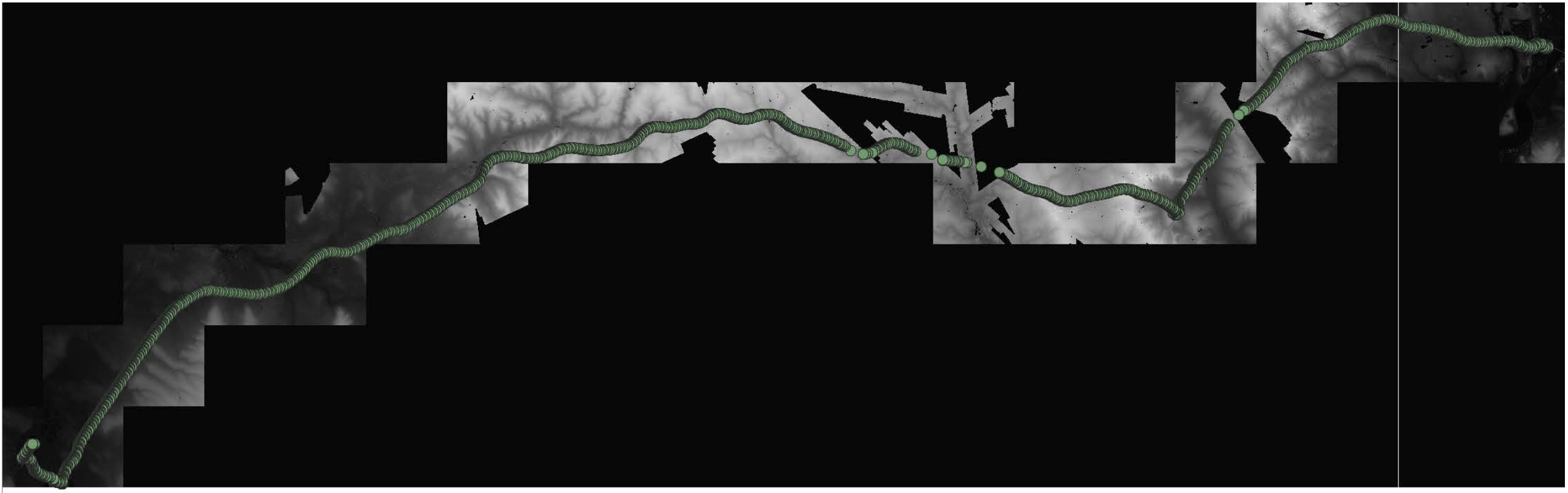
Final output: Block, Eas offset, Nor offset, Elevation

Testing & Results

- The route consists of 9000+ points, while errors may contain due to precision problems when opening the .csv file from Excel.
- The route can be displayed in Google Earth as this: (London to the right)

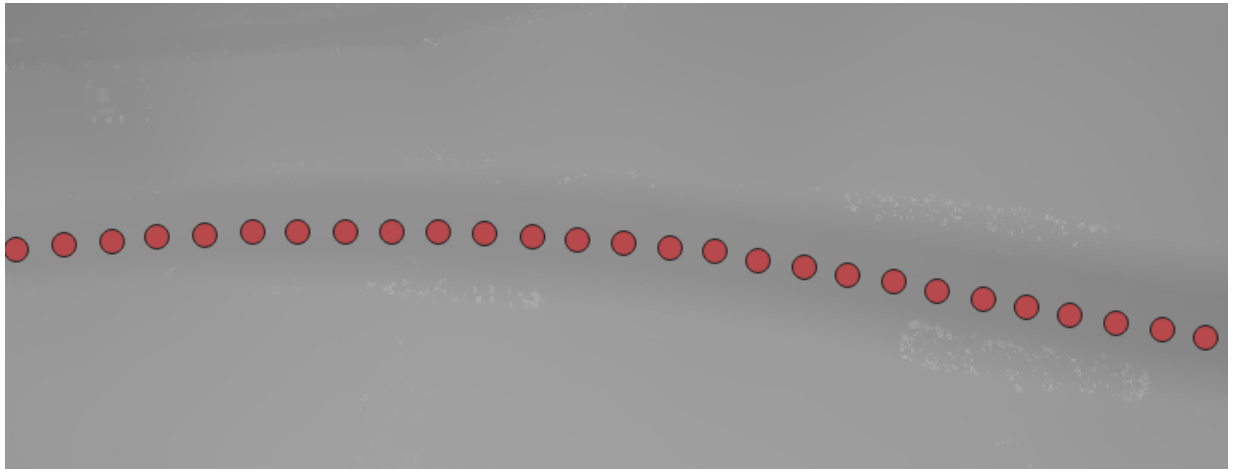
Testing & Results

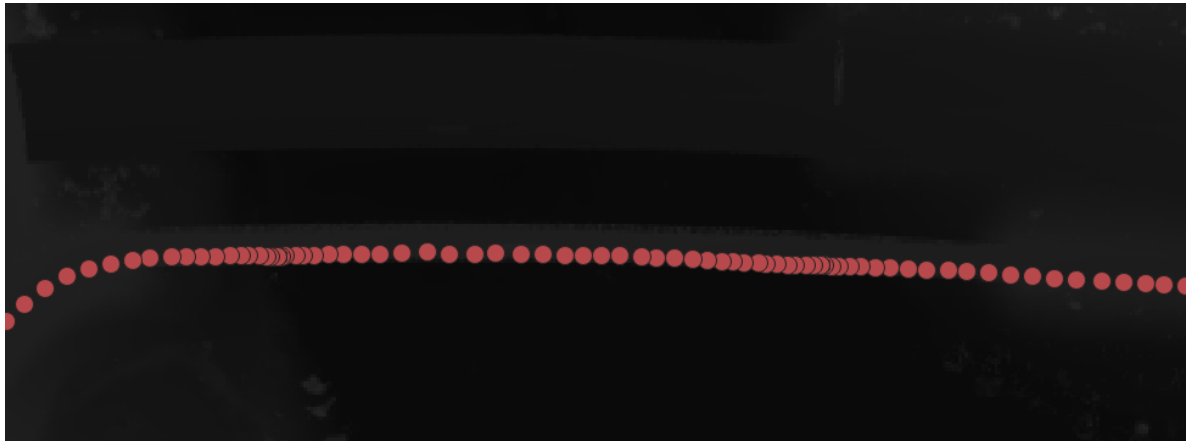
- This is the route displayed in QGIS:

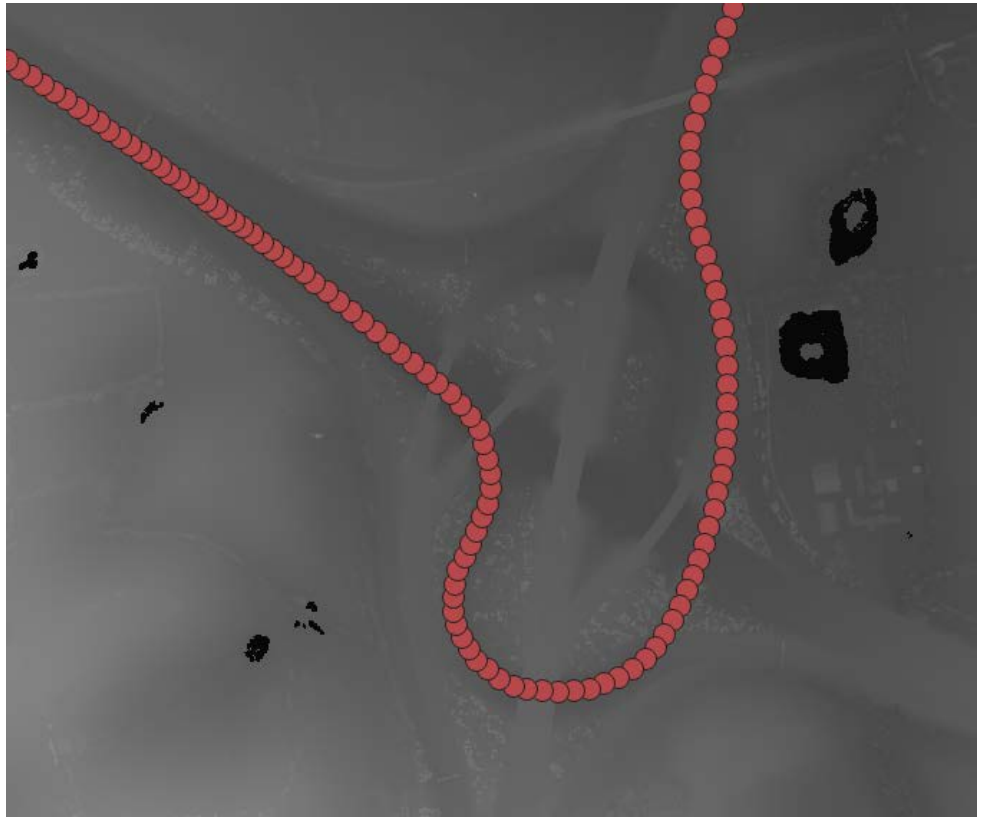


Testing & Results

- Details
  - This should be precise enough.
  - However, there are problems of data lacking to some point.

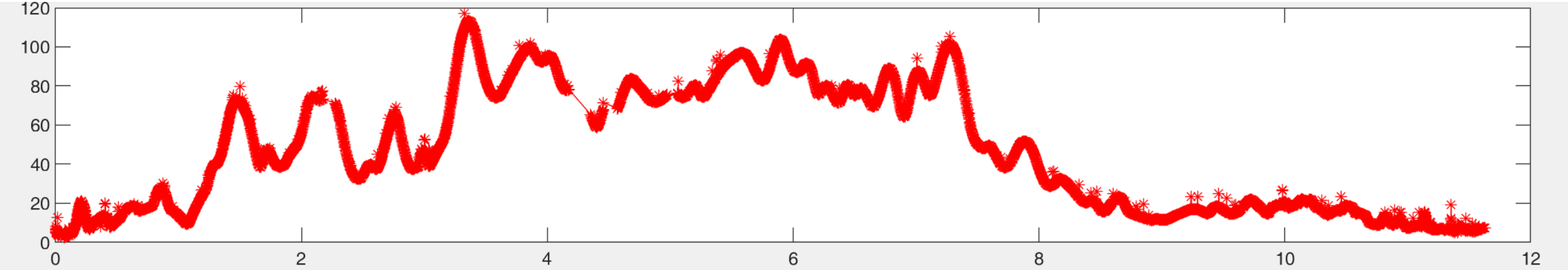






Testing & Results

- Elevation results from LIDAR Data
  - Data are shown by accumulated Euclidean distance between data points.
  - There are eccentric data points, since sample coordinate points may locate on bridges across highways or tunnels



Testing & Results

- Elevation Data from Google Earth
  - Data are shown by distances measured by Google Earth itself.
  - Data results nearly same