Assignment: Automated track infrastructure recognition using vibration analysis

In this study, we installed a suite of sensors to collect high-resolution data from train movements, which includes both rail vibration and Global Positioning System (GPS) measurements. Two rail vibration sensors were mounted on the train, Sensor 1 on the left side and Sensor 2 on the right, providing complementary vibration signals. The GPS unit simultaneously recorded latitude, longitude, and speed, while also logging the number of connected satellites to assess data quality. In addition to sensor data, infrastructure event information (including turnouts, joints, and bridges) was sourced from the primary control system. However, these events are not exclusively associated with the track currently used by the train; they sometimes correspond to nearby sub-tracks, necessitating an additional filtering step to isolate events that the train actually encountered.

The main goal of our project is to develop a robust algorithm capable of detecting track infrastructure events, specifically turnouts, joints, and bridges, directly from the rail vibration signals. This capability would facilitate more effective and automated monitoring of track conditions in real time.

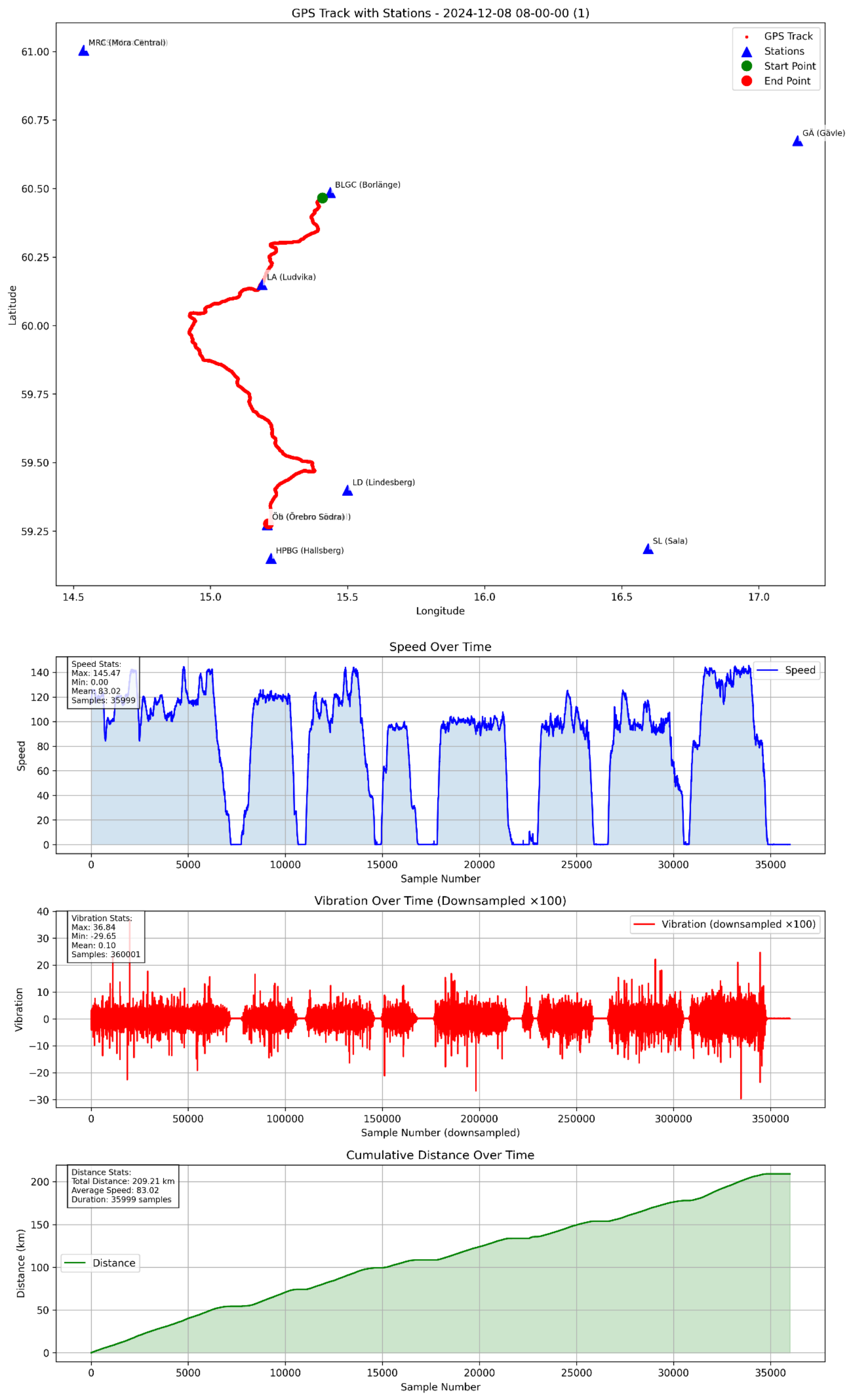

Figure 1. Train track data between two stations: This figure presents multiple aspects of the train journey, including the track between two stations, speed variations over time, vibration data from accelerometer sensors, and the cumulative distance travelled over time.

Code 1:

The script is used to load infrastructure information (including turnouts, joints, and bridges) and plot them on the map.

Code 2 :

The script is an end-to-end prototype for exploring vehicle telemetry: it prompts the user (via Tkinter file dialogs) to load five CSV files containing latitude, longitude, two vibration channels and speed, converts each into a Pandas DataFrame with a synthetic timestamp column, merges latitude and longitude into a GPS table and the two vibration channels into a single vibration table, then slices the vibration signal into 10-second windows with a 500 Hz sampling rate. Using Plotly and Dash, it renders an interactive web dashboard where the left panel shows every GPS point on an OpenStreetMap basemap and the right panel initially shows an empty vibration plot; when a user clicks a point on the map, the callback retrieves the corresponding vibration segment (or the last available segment if the index is out of range) and overlays the two vibration channels against time, effectively linking spatial position to time-synchronous vibration data for quick visual inspection.

Tasks to complete:

1. **Grade 3 : Mapping**
   - Complete Code 1, load the three CSV files (From Data 1 folder), draw the map, save a static image. Explain.
   - Using Code 2, select only the GPS/VIB files data (From Data 2 folder) that belongs to the same track in code 1, draw & save images. Explain.
2. **Grade 4 : Labelling**
   - Keep only the points actually travelled (Bridge, Rail-joint, Turnout).
   - Label every vibration segment with the corresponding categorical label (Bridge, Rail-joint, Turnout, and other).
3. **Grade 5 : Classification**
   - Extract numerical features from every vibration segment.
   - Train several classical ML models and at least one deep-learning model.
   - Report and compare the metrics, keep the best model.

Note: The data in the **Data 2** folder was collected from real-life train movements. Consequently, some GPS data may be inaccurate. This aspect is intentionally included in the assignment to expose students to the challenges of working with real-world data, including noise and inconsistencies.

Additionally, the vibration data has a high computational cost. Loading and processing it on low-performance computers may cause issues such as slow performance or crashes.

The report

You have to submit a report on this assignment. The report should contain:

1) Heading
2) Name of the participants
3) Problem description
4) Your observations and reflections

The assignment will be considered complete, when the student has demonstrated the working code to the teacher, uploaded the code to GitHub repository, and submitted the report on canvas.