Processing and Fusion of Transport platooning sensor data

Fredrik Vindenes

Master's thesis in Software Engineering

Department of Informatics,
University of Bergen

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

May 2022

Western Norway University of Applied Sciences

Abstract

In this thesis we are considering a data set which a group of researchers gathered, while conducting a transport platooning field experiment. We are assisting in the processing and fusion of the data, gathered during this experiment. We are investigating which sensors were used and what data gathering setup was used. We are investigating methods of extracting the data, formatting it and fusing it into a coherent data set, in a standardised format. We then explore different methods for filtering and extracting the data into a convenient format. We are also analysing the quality of the gathered data, using various techniques, and providing feedback on what can be improved if a similar experiment is conducted in the future. Finally we also develop some simple visualizations of the data.

Acknowledgements

First of all I would like to thank my advisor Lars Michael Kristensen, who has provided me with valuable feedback and insight during the writing of this thesis. I would also like to thank the NTNU researchers Maren Eitrheim and Markus Log, with whom I have been working during this thesis. I would like to thank them for their assistance and patience in answering my many questions. I would also like to thank Øystein Haukaas, the research assistant who helped with the quality assessment of the data. Finally I would like to thank my family and friends, who have supported and encouraged me during my work on this thesis.

Contents

Chapter 1

Introduction

1.1 Motivation

Transportation of goods is an essential part of our society. Everything from supermarkets, to hospitals relies on trucks to be able to operate. But the basic formula of how this transportation is carried out has not changed. One driver per truck driving from destination to destination. According to a 2017 study from the American Transportation Research Institute (ATRI)[1] driver wages accounts for about 40% of the overall transportation cost. There is also a significant shortage of truck drivers, both in the US and EU. It is estimated that EU lack at least 400.000 drivers as of 2021[9], and the US lacks at least 80.000[2]. This has led to several companies looking into how we can make trucking more efficient. In this thesis we are looking at one such approach: transport platooning.

1.2 Background NTNU research project

The research project and experiment to which this thesis is linked, is a collaboration between a group of researchers at NTNU and Statens Vegvesen. The purpose of the project is to assess the viability and suitability of transport platooning on rural Norwegian roads. The Finish company Ahola Transport provided the trucks, and drivers for the experiment. There were a total of 3 trucks, with 3 drivers. A support vehicle accompanied the trucks during the entire journey. The trucks were using a proprietary Scania convoy system, that relies on a combination of radar and cameras to "connect" the trucks, into a platoon. The front most truck controlled the speed and distance, between the three trucks. The drivers had to perform the steering of the trucks manually, throughout the journey. The driving took place along RV77, E6 and E10 from Junkerdal to Storjord to Narkvik to Bjørnfjell. The experiment took place on the 21st and 22nd of October 2021. On each day there were several breaks during the journey. Some parts of the drive were driven multiple times, for testing purposes [5]. Figure 1.1 shows the route driven during the field experiment.

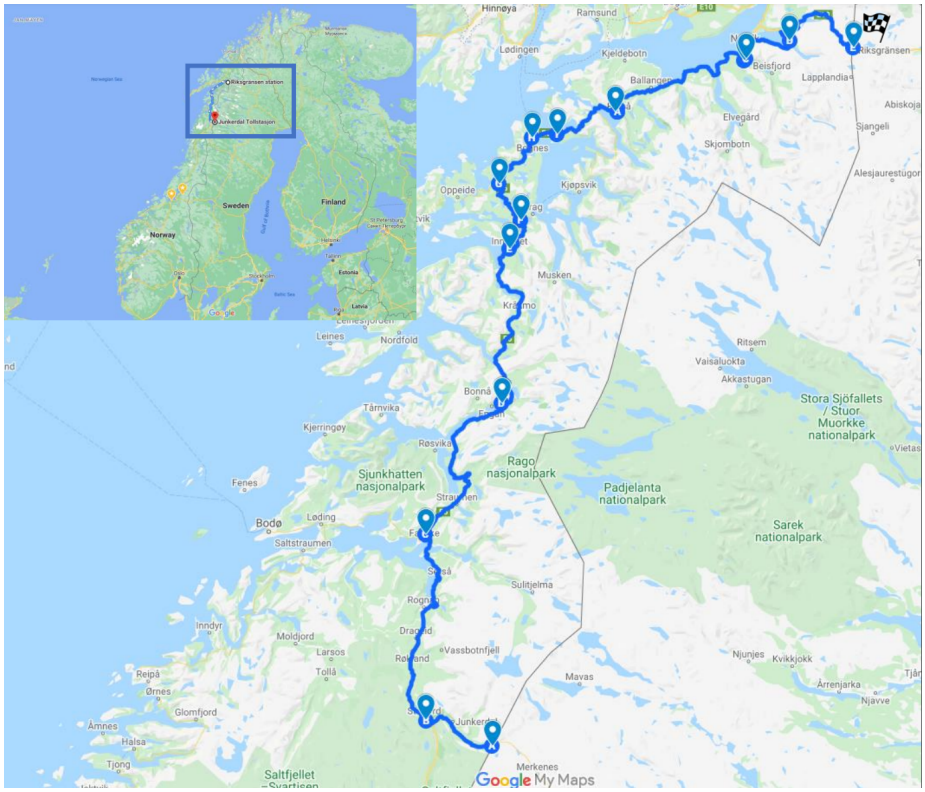

Figure 1.1: The route driven during the experiment. Credit: M.Eitrheim, M.Log et al.

1.3 Research questions

The overarching research questions that the researchers set out to answer is: Are rural Norwegian roads suitable for transport platooning, and how does a transport platoon impact the driver of a truck.

1. Does a professional driver trust the system?
   The researchers would like to figure out how much a professional driver trusts the system. This will be done by monitoring the drivers actions, and by conducting a questionnaire at regular intervals during the experiment. For this experiment, the 3 drivers were monitored and questioned by 2 researchers and 3 cameras
2. Which road features(if any) pose a challenge?
   The Researchers would like to figure out if there are any road features that causes issues for the transport platoon. They would also like to know what characterizes such features(if any). This will be part of the analysis, after the experiment
3. How does a platoon impact other road users?
   The researchers would like to know if a transport platoon effects other road users. This includes both other vehicles on the road and pedestrians along the road.
4. How can we better conduct such an experiment?
   The researcher would like to know which sensors worked well, and which did not? Do we have sufficient data to determine the viability of transport platooning on rural Norwegian roads? If not: How can we gather more relevant data?

The specific research questions considered in this thesis, and derived from the questions above, will be presented in section 1.5

1.4 Background on Platooning and Autonomous Driving

Different approaches to autonomous driving

There are three main categories of autonomous driving systems:
Vehicle to infrastructure(V2I) Vehicle to infrastructure refers to any approach where the road system itself is designed to allow vehicles to share information with the systems that operate and manage roads and traffic. Examples of such technology includes RFID readers, cameras, traffic lights, lane markers, streetlights, signs and parking meters [8]. This approach allows the infrastructure itself to help guide vehicles driving on the roads. The advantages of this approach is the technical ease of implementation, in terms of standardization, regulation and maintenance. It generally requires much less software and hardware in each vehicle, and can significantly improve traffic flow. Some of the disadvantages are that it requires expensive infrastructure to work. It also requires (some) specialized hardware in each vehicle, and is limited to operate in geographic areas, with the correct infrastructure.

Vehicle to Vehicle(V2V) Vehicle to Vehicle refers to any approach where vehicles use sensors and digital communication to coordinate among them [3]. Unlike vehicle to grid systems, these systems can drive on regular roads without the need for specialized infrastructure. One of the big advantages of this approach is the relative Ease of implementation in terms of standardization, regulation and maintenance. It also requires less software and hardware required in each vehicle, compared to full autonomy. Some of the disadvantages are that it requires multiple vehicles with compatible hardware, and still requires a driver in some situations. Each vehicle also needs to be partially autonomous(Lane-keeping, adaptive cruise control etc.).

Full autonomy Fully autonomous systems refers to any system that can autonomously drive in traffic, as if it was a regular car with a human driver.This type of system is much more flexible than the two other approaches. This makes it generally more scalable, and it works well, even if other vehicles are not autonomous. This approach is technically difficult and complex to implement. It also requires each vehicle to carry expensive hardware, in order to work.

Transport platooning

In this thesis we are analysing data gathered from the transport platooning experiment. Transport platooning is a Vehicle to Vehicle technique. In a transport platoon the front most vehicle is controlled by a human driver. Every vehicle behind the first vehicle is configured to follow the vehicle in front of it(similar to a truck with multiple wagons attached). Each vehicles(except the first) is semi-autonomous in the sense that it can keep the distance to the truck in front of it, and stay within the driving lanes. The advantage of this approach is that one can have a single driver be responsible for several trucks, thus significantly reducing the amount of drivers needed in the trucking field. One intermediate approach may be to have trucks drive in platoons between major population centres. The trucks are then piloted by human drivers inside of the cities.

1.5 Research questions (For this thesis)

The focus of this thesis is on enabling the researchers to answer the research questions posed in section 1.3, by processing and analysing the data, gathered during the experiment. We aid directly in answering how we better can conduct such an experiment. We help create a data set that will enable the researchers to answer the remaining 3 questions, by fusing all the data into one data set. We will also add some data, from a publicly available database, to aid the researcher in answering which road features may pose a challenge. Furthermore we create methods for visualizing certain data on a map. The software developed as part of this thesis will serve as a basis, from which even more advanced functionality for visualization or statistical analysis may be developed.

- Research Question 1: How can we combine all the data gathered into one data set? We will look at how the data can be formatted into one common format, and combined into one coherent and clean data set.

- Research Question 2: How do we synchronise all the gathered data into one timeline? We will be analysing the timestamps of all data sources, and determining the time difference between each of them.

- Research Question 3: How do we select, and extract segments of the data? We will look at different methods of selecting and extracting segments of data.

- Research Question 4: How good is the quality of the data? We will analyse the quality of the most important data sources, and provide feedback on the overall quality of each source.

1.6 Overview of the thesis

The goal of this thesis is to create a data processing pipeline. This pipeline takes in raw data, and produces a coherent data set, containing all the data in a standardised format. The pipeline then lets the user extract part of the data set, by filtering out the desired data. This will allow the researchers to analyse segments of the data. It will also create a clean and coherent data set, for use in further analysis. This thesis is split into 5 sub-tasks:

1. Classifying and reading the data: The first step of the thesis is to classify the different data sources. We classify the data, based on the type of information gathered by each data source(sensor). We then generate a plan for how to read the information into a usable data structure
2. Formatting the data into a common, easily usable format: The second step is to identify how the data, from the various data sources is formatted. We will convert both the timestamps, and GPS coordinates into one common format. We then create a method for converting the timestamp and location data from each individual file into this common format. The common format should be identical for all of the various data sources, and make future work with the data easy.
3. Synchronizing the data: The third step of the thesis is to synchronize all the timestamps of all the data sources into one common "Timeline". The synchronization will be carried out using a combination of algorithmic, and manual techniques, to determine the offsets between the various data sources. We will select one of the data sources to serve as the reference, from which all time offsets are calculated. The timestamps in each data source will then be adjusted, such that they match the timestamps in the reference i.e. they are "synchronized" with the Timeline.

4. Assessing the quality of the data: As part of the synchronization step, we also assess the quality of the data, from the various data sources. This is done in collaboration with the research team, that carried out the platooning experiment.

5. Filtering and extracting the data: Finally, we create two methods of filtering the data, and two methods for extracting the data into files. The filters will allow the user to precisely extract portions of the data for further inspection and analysis. We also provide some simple functionality for visualizing the data on a map.

This thesis will not include any analysis of the data itself, beyond a high level analysis of the quality of various data sources. In this thesis we will be removing some features from the data, where those features add little to no information. These features usually includes no data, redundant data, or the same data for every data entry. Except for Timestamps and coordinates, we will not be modifying or augmenting the data(with one exception, discussed in section 5.4). Since only part of the data was available at the start of the thesis, the pipeline had to be designed with a general and extendable implementation, to easily accommodate new data sources. Some of the labels of the data were altered, to better describe the data they represented. Some of the files provided were not read by our data processing software. These files contained metadata, or information not part of the data set. A few of the excluded files were part of the data set, but only contained one or a few data entries. This thesis will also not discuss the administrative tasks, or coordination with the research team at NTNU, in any significant detail.

The software described in this thesis is developed to work with Python version 3.9. The program has Pandas, Folium, OpenPyxl and MathPlotLib as dependencies. All other dependencies are part of the Python standard library. All Excel files can be opened in any Excel version supporting the XLSX format. All code discussed in this thesis is stored on github, at the following adress: https://github.com/selabhvl/transportplatoon/. The folder titled "python" contains all code relevant to this thesis.

Chapter 2

Data Sources

2.1 Sensor setup

In this chapter we provide an introduction to the data sources and sensors that were used in the transport platooning experiment. Figure 2.1 shows the setup of sensors in one of the trucks.

2.1.1 Custom radar sensors

Each of the three trucks were equipped with a custom radar device on the front. This radar setup consisted of a Raspberry Pi computer(unknown version) running the Raspbian Operating system and a uRAD Raspberry Pi v1.2 radar device. The radar device was connected to the Raspberry Pi and mounted to the front of the truck(behind the windshield). The radar device is capable of tracking up to 5 separate contacts. It is capable of tracking contacts with a range of 70 meters, and traveling at speeds up to 70 meters/second. It has an accuracy of +/- 4cm for distance, and +/- 5cm/second for velocity.

2.1.2 GoPro cameras

Each of the three trucks were equipped with 3 cameras each. These cameras were installed in the same location for all three trucks. The cameras were installed at the end of a adjustable stand, that was clamped to the inside of the vehicle(see Figure 2.1). One camera was used as a dashcam, pointing forward at the road. One camera was monitoring the steering wheel, to monitor the drivers interactions with the steering. One camera was pointed towards the accelerator and brake. The feet camera was monitoring when the driver was hovering his feet above the accelerator or brakes. Additionally, each camera contained a GPS device. This device recorded GPS coordinates along the route to the best of its ability. This data was synchronised with the video. Each camera also contained an accelerometer and a gyroscope. These devices recorded acceleration in all three axis. The 3 axis were aligned with the camera. Since all three cameras were tilted during the entire drive, these axis are not perfectly aligned with the trucks. The researchers tried to adjust these axis to roughly align with

Figure 2.1: The sensor setup for one truck. Image credit: M.Eitrheim, M.Log et al. Edited by: F.Vindenes

the trucks. As part of the video recording, the cameras also recorded audio of everything that was said during the drive. This audio recording is the basis for the transcript file, and when combined with the videos (from all 3 cameras), it is the basis for the annotation file.

2.1.3 VBOX Sport telemetry system

Each of the three trucks were equipped with a VBOX Motorsports VBOX sport telemetry system. The VBOX system records location data, and saves this data to an onboard SD card. The device has an automatic start/stop functionality. This functionality enables the device automatically, when it detects movement. This functionality turned out to be unreliable, and some of the data gathered may have been cut short because of this. Some parts of the drive were not recorded by the VBOX system in some of the trucks, because of this issue. On day 2 of the experiment, the researchers failed to properly insert the SD card into the VBOX system in truck 1. This caused all the VBOX data from truck 1 from the second day to be lost.

2.1.4 Scania Fleet Management system(Scania FMS) and Vecho Fleet Management system(Vecho FMS)

A Fleet management system is a system that records certain types of telemetry, and sends the data to the fleet operator. All 3 of the trucks used in this experiment had two separate fleet management systems installed. These systems recorded data independently of each other. The systems turned on, and started logging when the trucks were turned on. Both of the systems used a cellular data network to send the data back to the fleet operator. The fleet operator then forwarded the segments of the data relevant to this experiment, to the researchers.

2.1.5 Bittium Faros 180 Heart rate monitor

The researchers used 2 Bittium Faros 180 heart rate monitors. On day one the driver of truck 1 and the driver of truck 2 was fitted with the heart rate monitors. On day two the driver of truck 1 and the driver of truck 3 was fitted with the heart rate monitors.

2.1.6 Pupil Invisible eye-tracking glasses

Each of the drivers were fitted with Pupil Invisible eye-tracking glasses. Due to error in the setup, the data from the glasses turned out to not be usable.

2.1.7 External sensors data: Traffic enforcement camera

Statens Vegvesen operates traffic enforcement cameras, stationed at several different locations along the route, driven during the experiment. The cameras record each vehicle that passes by each of the cameras. This information is then sent to Statens Vegvesen, who passes the information on to the researchers.

2.2 Data format

In this section we outline all the different data contained in all the data sources we used in this thesis. We will be listing all data points, their associated data type, and a short description of the format(if applicable).

2.2.1 General comment about data segmentation

The researchers decide to split the data gathering into segments. These segments are based on geographical locations in Norway, and correspond to the locations where the researchers took breaks. Not all data sources uses the same type of segmentation. The first day was split into two segments: Junkerdal border crossing to Fauske and Fauske to Innhavet (end of day 1). The second day was split into three segments: Innhavet to Bogness ferry pier, Skarberget ferry pier to Narvik and Narvik to Bjørnfjell border crossing. During the ferry trip on day 2 (Bognes to Skarberget), no data was recorded. For truck 3, the GoPro camera stopped operating shortly after start on day 2. The GoPro data for the segment from Innhavet to Bognes is thus split into two sub segments.

The following is a list describing which data source uses which kind of segmentation

- Split into segments:
  Radar(Day 1 only),VBOX(varies by truck/day), GoPro(all data), NVDB data, Annotations
- Split into days:
  Radar(Day 2 only), Heart rate variability, Self report (questionnaire)
- Not Split(all data in one file):
  Fleet management system data (Scania and Vecho), Traffic enforcement camera data

2.2.2 Explanation for GoPro timing

The GoPro camera was originally designed to capture relatively short videos. In this experiment the researchers used the GoPro cameras to capture several hours of video each day. The GoPro camera therefore ended up stopping and restarting its recording at a semi-regular interval between 11-17 minutes. This interval is not consistent between the different cameras or different segments of the data. The video was thus split into shorter videos of various length. In between each video, the GoPro camera used approximately 2 seconds to start recording the next video(see section 4.2.1). This reset in the timestamps is reflected across all data sources gathered by the GoPro Camera. All data that comes directly from the GoPro cameras uses the time unit milliseconds since restart. This time unit has to be interpreted the following way: Determine when the file started recording in local time. This was done by holding a clock in front of the camera when it started recording. For each data point add the number of milliseconds to the start time of the file. When we reach a reset, we add the last time value before the reset, to the accumulated total. This total has to be added to every subsequent data point. For each restart 2050 milliseconds has to be added to the total(not including the initial start). A restart is usually characterized by a sudden drop down to 0 milliseconds. The files that were created manually by the researchers, which are based on the GoPro data(Annotations and Transcripts), uses seconds since start of the GoPro file. These files does not reset the timers, at any point during the file.

2.2.3 Radar Data

The radar data was recorded into two separate files for each truck on day one, and into one file per truck for day two. The files from day one represents the stretch from Junkerdalen to Fauske, and the strech from Fauske to innhavet. The file for the second day represents all segments driven, on that day. Radar device operating mode is described in the manual [10]. The radar data contains the following data:

Description	Datatype	Format or Unit
Operating mode of the radar device	Int	n/a
The local timestamp of the device for data entry	Datetime	YYYY-MM-DD HH:mm:ss.fff

For each contact(up to 5 contacts for each data entry) the following three data points were recorded:

Description	Datatype	Format or Unit
Distance to the contact	Float	Meters(0.01 precision)
Relative speed to the object	Float	m/s(0.1 precision)
Signal to noise ratio	Int	Decibels

2.2.4 GoPro GPS data

Description	Datatype	Format or Unit
Milliseconds since reset*	Int	n/a
Latitude	Float	Deg' North
Longitude	Float	Deg' East
Altitude	Float	Meters(WGS84)
Speed	Float	Unknown
Speed in 3D(with height)	Float	Unknown
TS	Float	Unknown
GPS accuracy	Int	0 to 9999(lower is better)
GPS satellites used	Int	n/a

*see Section 2.2.2

2.2.5 GoPro Accelerometer data

Description	Datatype	Format or Unit
Milliseconds since reset*	Float	n/a
AcclX	Float	Unknown
AcclY	Float	Unknown
AcclZ	Float	Unknown

*see Section 2.2.2

2.2.6 GoPro Gyro data

Description	Datatype	Format or Unit
Milliseconds since reset*	Float	n/a
GyroX	Float	Unknown
GyroY	Float	Unknown
GyroZ	Float	Unknown

*see Section 2.2.2

2.2.7 VBOX telemetry data

Description	Datatype	Format or Unit
GPS satellites used	Int	n/a
The local timestamp of the device for data entry	Datetime	HHmmss.ff
Latitude	Str + Float	+/- and Arcminutes*
Longitude	Str + Float	+/- and Arcminutes*
Velocity	Float	Km/h
Heading	Float	Degrees
Altitude	Float	Meters(WGS84)
Longitudinal acceleration	Float	G's
Latitudinal acceleration	Float	G's
Temperature	Float	Celsius

There are 4 more parameters relating to the battery. These were not used for telemetry due to relevancy

*Plus means North or West, Minus means South or East

2.2.8 Scania FMS data

Description	Datatype	Format or Unit
The local timestamp of the device for data entry	Datetime	YYYY-MM-DD HH:mm:ss
The odometer of the vehicle	Int	n/a
Fuel level	Int %	n/a
AdBlue level	Int %	n/a
Velocity	Mixed	Float + "km/h"
Latitude	Float	Deg' North
Longitude	Float	Deg' East

There are 4 more parameters relating to the location and vehicle status. These were not used for telemetry, due to redundancy.

2.2.9 Vecho FMS data

Data from the Vecho FMS were not used in this thesis. The data from Vecho FMS was split across multiple files, with vastly different formats. It was thus determined that the utility of the data was unlikely to make up for the effort required to integrate the data into the final data set. A preliminary assessment of the data determined that the quality and frequency of the data was low and inconsistent. This asessment was carried out by a research assistant, associated with the experiment.

2.2.10 Kubios HRV Heart rate data

Data from the Bittium Faros 180, analysed with Kubios HRV Premium software. The Heart rate data contained over 50 parameters(not listed). The data entries use a standard timestamp of the format HH:mm:ss.

2.2.11 Event log(annotations)

The researcher reviewed the videos from the GoPro cameras, and created an annotation file, for each truck, for each segment of the drive. This annotation file contains information about certain types of events that occurred during the drive. Some events are marked as point events, meaning they occurred at a certain point in time. Other events are marked with start or stop, representing the start and stop of the time period, in which the event occurred.

The file contained the following common data points:

Description	Datatype	Format or Unit
3 x video file which the information was extracted from	Strings	3 separate variables
Timestamp representing the local time, when the file started	Datetime	YYYY-MM-DD HH:mm:ss

For each data entry:

Description	Datatype	Format or Unit
Seconds since the start of the file*	Float	n/a
Event description(behaviour)	String	n/a
Event category(behavioral category)	String	n/a
Comment	String	For use by the researchers
Event marker	String	POINT, START or STOP

There are 4 more parameters containing information about the file, which the annotations was taken from. These data points were not used for telemetry, due to being identical for every data entry throughout the file.
*See Section 2.2.2

2.2.12 Transcript

Based on the videos from the GoPro cameras, the researchers created a transcript. This transcript contains all statements made by the truckers, and the researchers during the journey. The statements were made in Norwegian, English or Finish. All statements made in finish were translated into Norwegian.

Description	Datatype	Format or Unit
Start timestamp	Datetime	HH:mm:ss
Stop timestamp	Datetime	HH:mm:ss
Subject(that made statement)	String	Short form for name
Statement	String	n/a
Language	String	Language spoken
Context	String	n/a
GoPro Video	String	Filename of GoPro video
Based on camera	String	Dash,Steering or Feet

2.2.13 Self-reporting questionnaire

The questionnaire consisted of 4 questions:

- Think of the last 10 minutes. On a scale from 1 (very Low) to 10(very high), please rate your perceived workload
- Think of the last 10 minutes. On a scale from 1 (very Low) to 10(very high), please rate your perceived trust in the convoy system
- Think of the last 10 minutes. On a scale from 1 (very Low) to 10(very high), please rate your perceived safety
- Think of the last 10 minutes. On a scale from 1 (very Low) to 10(very high), please rate your perceived comfort

These questions were posed to each driver, at irregular time intervals. They were posed 11 times on day 1 and 8 times on day 2. All answers were manually recorded in an Excel-file.

2.2.14 Traffic enforcement camera data

Description	Datatype	Format or Unit
Timestamp of the camera	Datetime	YYYY-MM-DDTHH:mm:ss.fffZ
Timestamp of the server	Datetime	YYYY-MM-DDTHH:mm:ss.fffZ
Lane number	Int	n/a
Velocity	Float	Km/h
Length of vehicle	Float	Meters
Seconds between each entry(vehicle spacing)	Float	n/a
Type of vehicle	String	Specific codes used

There are 7 more parameters. These data points were identical for every data entry throughout the file.

2.2.15 NVDB data

NVDB data was extracted as part of this thesis. This data provides information about road features, along the route driven in this experiment. Further details on the contents of these files and the extraction method is provided in chapter 5. NVDB uses a system of references called Vegsystemreferanse, which are also described in Section 5.2. The NVDB data consists of one file per segment of the drive. Each file in turn consist of 6 different Excel-sheets.

Common for all sheets:

Description	Datatype	Format or Unit
Location(point)	String	Vegsystemreferanse Point
Latitude	Float	n/a
Longitude	Float	n/a
Height	Float	Meters(WGS84)
Location(Range)	String	Vegsystemreferanse Range

The first 2 Data sheets deals with horizontal and vertical curvature, the next 2 deals with tunnels and bridges, and the last 2 deals with speed limits and road width.

Horizontal curvature:

Description	Datatype	Format or Unit
Type	String	road type(shape)
Radius	Int	Meters, sign is irrelevant

Vertical curvature:

Description	Datatype	Format or Unit
Type	String	road type(shape)
Radius	Int	Meters, sign is irrelevant
Height	Float	Meters, start of segment
Height	Float	Meters, end of segment
Slope	Float	Meters, start of segment
Slope	Float	Meters, end of segment

Tunnels:

Description	Datatype	Format or Unit
Tunnel name	String	UTF-8
Length	Int	Meters
Width	Float	n/a

Bridges:

Description	Datatype	Format or Unit
Bridge name	String	UTF-8
Bridge category	String	n/a
Bridge type	String	n/a
Length	Int	Meters

Speed limit:

Description	Datatype	Format or Unit
Speed limit	Int	Km/h

Road width:

Description	Datatype	Format or Unit
Road width	Float	Meters
Road width (max)	Float	Meters
Road width (median)	Float	Meters
Road width (min)	Float	Meters
Road width (normal)	Float	Meters
Road quality(thickness)	Int	Centimeters

More than a dozen parameters were removed from the original NVDB files. This was done in consultation with the researchers. The removed parameters are primarily metadata.

Chapter 3

Fusion of Data Sources

In this task we are processing data from a transport platooning experiment. This experiment involved a large number of sensors, all of which used a idiosyncratic format to store its data. In order to work with this data, we need to format all data sources into a common format. Part of the fusion is to synchronize all data sources. Since all data sources recorded time using different methods and time zones, we need to adjust the timestamps. We would like for all of the data to follow the same timeline. The purpose of the fusion step is thus to convert all the different data sources into one clean, coherent, synchronized and data set. This data set will serve as the basis for all future work on the data.

3.1 Tools

We choose to use Python 3 as the primary language for this thesis. Python has an extensive set of libraries available, and a large community that provides support for other users. All programmers involved in this thesis appeared to be familiar with Python, and practically all processing of the data listed in this thesis was implemented, using Python only.

We choose to use Pandas as our primary library to manage the data[4]. Pandas is designed to perform well with large data sets, and has extensive support for manipulating N-dimensional data sets. Pandas is widely used for data manipulation in industry, data science and machine learning. We therefore found Pandas to be the best fit for this thesis, since the thesis involves a lot of data manipulation. Pandas uses a data structure called Data frames.

For the visualization part of the thesis we chose to use Folium[7]. Folium is a map tool, that integrates well with Python and Pandas. We chose Folium, because it offers a simple interface to visualize spatial data, on a geographical map. Folium fetches its maps from OpenStreetMaps, and adds the functionality to place markers and lines on this map. These markers can then contain all the information we need to display for each location. Thus Folium fulfils all the basic requirements we have for visualization, in this thesis.

During this thesis we used to primary types of data structures: Lists and Data frames. In this thesis Pandas Data frames are used as the primary data structure for storing, and extracting the data. Data frames has a significantly smaller footprint in the memory of the program, compared to Lists. Data frames offers a lot of built in utility for reading from, and writing to files. Data frames also offers powerful functionality in terms of filtering data, based on the values of certain columns. Lists were used in the cases where manipulation of individual data points were required. Most of the data manipulation occurred while reading the data from a file, and formatting the data into a standardised format. Thus most of the file reading functions used lists as their primary internal representation. For visualisation of annotations lists were also used, as we needed to extract data on a row by row basis. In all other parts of the program data frames were used exclusively. The conversion between List and Data frames and vice versa is relatively simple.The pandas.Dataframe() function converts from List to Data frames, and the reverse was done using Dataframe.tolist().

Advantages of data frames:

- Data frames has a better performance than lists for extracting and modifying large amounts of data.
- Data frames allows the user to move, modify or delete entire rows or columns from the data set, with one function call. It also allows for removing a range of rows or multiple columns at once.
- Data frames allows the user to access data by using column names as keys, instead of indices. This allows us to access data without needing to keep track of which column number contains which data by using keys for indexing.
- Data frames gives the user explicit control over the data types of the data, when needed.
- If the data requires no prepossessing, Pandas allows the users to read a CSV or Excel file, directly into a data frame.

Advantages of Lists:

- Lists are more flexible than data frames, in that they allow rows of different length.
- Lists uses direct indexing, which is more useful when manipulating individual data points.
- Lists allows the user to iterate through the list, without a huge performance penalty.

For working with time, we decided to use the Datetime module, which is part of Pythons standard library. The Datetime library supports various types of time arithmetic, including addition, subtraction and difference. It also allows the user to easily convert between different time representations. In this thesis we determined that we would need timestamps with millisecond precision. We therefore choose the ISO8601 format without time zones, as the standard format for timestamps. Datetime natively supports both reading and writing of ISO8601 formatted strings, thus simplifying the the reading process.

For reading Excel files we chose Openpyxl[6]. We needed a tool that would let us control the reading of each file, line by line. We also wanted to manipulate some of the data during the reading process. Although Pandas does support reading a variety of Excel files into a data frame, it gives us much less control over the reading process. We thus decided that it would be simpler to read data from Excel iteratively, rather than reading it all at once. The latter would require us to convert the data back to a list, removing labels, and then iterating through the data.

3.2 Strategy

We spent quite a bit of time discussing the strategy for how to fuse the data into one coherent data set. Since all the different data sources (except NVDB) contained a timestamp, we decided that the two modes of extraction should be based on time, and coordinates. This would require all the different data sources to be synchronized into one unified timeline. In order to synchronize all the data sources, we needed a basis to serve as the "ground truth". The data source would have to have both a timestamp and coordinates for each data point throughout the file. The data source should cover all parts of the journey. Finally the data source should have a reasonably high and consistent frequency of data entries. This would minimize the need for interpolating between data entries, when synchronizing. Two of our data sources fit this description: VBOX and GoPro GPS. We ultimately decided to use GoPro as the basis for data fusion. GoPro GPS has a frequency of 7- 8 data entries per second, and records both timestamps and coordinates. Additionally GoPro had 3 cameras for every truck. This allows us to choose the highest quality data source for each truck, for each segment. Another consideration, was that using GoPro would simplify the synchronization step for the annotation files, and the transcripts.

3.3 Fusion based on time synchronization

3.3.1 The GoPro Offsets file

The different GoPro cameras were all turned on at different times. We thus need a method of synchronising all of the different GoPro data into a common timeline. The researchers handled this issue, by holding a clock in front of the GoPro camera, when the camera was turned on. Because these videos contained sensitive information, they were not made available for this thesis. In order to determine when each GoPro file started, we needed a file that contained the starting time of all GoPro GPS files. The researchers thus created a common file, containing all the information necessary, to synchronize all the GoPro data. In this file, the data is split into groups of 3 entries. Each group represents one truck, driving one segment of the journey. There is a label for each group, that represents which segment, and which truck it represents. Each of the 3 entries in the group represents one GoPro camera inside of one truck. For each group there is a timestamp of the format YYYY-MM-DD HH:mm:ss. For each camera in the group there is a float value, that represent the number of seconds after the timestamps. In order to determine when a camera in a given group started, we take the timestamp for that group and add this offset value to it. We then have a synchronized timestamp for that camera. The timestamp determined for the GoPro GPS file is also used for the GoPro Gyro and GoPro Accelerometer files.

3.3.2 Determining the starting time of the radar data

The radar data that the researchers gathered, used the local time of the Raspberry Pi to create the timestamps. Unfortunately the local time of the raspberry pi, was not calibrated before the experiment. The timestamps on the radar data was thus several days behind the other data sources. In order to synchronize the radar data with the GoPro data, we had to make use of the annotation files. In each of the annotation files there is a data entry, that contains the data point "Radar logging" in the "behavior" column. This represents the point in time, when the radar was activated. By adding the timestamp of this data entry to the start time of the annotation file, we get the start time of the radar file. For day one, both annotations and radar data is split by segments, and by trucks. We can thus easily establish which annotation file, to associate with each radar file. For day two, the radar data is contained in one single file, for the entire day. The "Radar logging" entry, is only present in the first annotation file, for each truck on day two. For day two, we thus only use the first annotation file for each truck, to determine the start time of the radar file. It is important to note, that the first data entry in the radar file may be recorded a few seconds after the radar was enabled. The exact starting time of the radar file is stated in its filename.

3.4 Fusion based on location

3.4.1 Algorithms

We used two primary algorithms for synchronizing the data. Both of these algorithms use the Haversine method[11], to determine the distance between two points, using GPS coordinates.

The first of these algorithms we will refer to as the Correlation algorithm. The correlation algorithm takes in two data sets. Both data sets must have both a timestamp, and a position for each data entry. The algorithm then iterates through the first data set. Each data entry will be matched with the closest location in the second data set. It will then create a third data set, that contains the entire first data set, augmented with the closest matches from the second data set. By inspecting this new data set we can determine approximately how big the offset between the two data sources are. This assumes that both data sets are of high quality. If the data sets are of low quality, we can still get a reasonable idea of approximately how big the offset between the various data sources are. This algorithm comes in two varieties. The first variety assesses all data entries, while the second reduces the number of data entries, by only assessing every Nth data entry from the first data source. For data sources with a high frequency, it may be necessary to use the second of the two varieties, to reduce the running time of the algorithm.

The second algorithm we will refer to as the checkpoint algorithm. This algorithm also tries to find the time difference between two data sources. Unlike the first algorithm however, this algorithm uses the Traffic camera data as the second data source. Since the Traffic camera data is synchronized to a 3rd party server, we can use this information as a "Ground truth". The KPRecords data contains both location data, and timestamps for each of the 3 trucks. This algorithm determines when a given truck passed the traffic camera. By checking multiple files, and data sources with this algorithm, we can determine approximately how large the offset is between the files. This is done by calculating the time difference between different files, when they passed by a traffic camera. This algorithm can also be used to assess the quality and consistency between files.

The reliability of first of the two algorithms depends on the quality of the location data. It is thus a good idea to look at a large sample of data entries, to determine the time difference. Both of the algorithms perform poorly for data gathered while the trucks were travelling through tunnel. Since two of the traffic enforcement cameras were located inside of a tunnel, we should expect these two data points to deviate from the rest.

3.4.2 VBOX

The VBOX data recorded all of its data entries using a local timestamp. We ran the correlation algorithm on 2 of the VBOX files, and compared them to the corresponding GoPro files. We found that the VBOX data was approximately 2 hours ahead of the GoPro data. We used 2 hours, as the offset for all the VBOX files, on both days.

3.4.3 Scania FMS

The Scania FMS system does not store its data locally. Instead it sends the data to a server, operated by the Fleet manager. Because of this, the timestamp will be determined by the server, when the data entry is recorded. We used both the correlation algorithm with a GoPro file, and the checkpoint algorithm with the KPRecord data, to determine the offset. We determined that the Scania FMS data was about one hour behind the GoPro data. We thus decided to add a fixed offset of -1 hour to all data entries in the Scania FMS files.

3.4.4 Heart rate data

The Heart rate data did not contain any coordinate information. There were also no other information available, linking the heart rate data to any of the other data sources. We decided to consult the researchers about the timestamps, from the heart rate data. One of the researchers informed us that the heart rate monitors, started recording at 8:30 for participant 1 and 10:00 for participant 2 on the first day. Since these timestamps roughly correspond to the timestamps listed under measurement data, we decided to leave the heart rate data unchanged. We consider the heart rate data to be semi-synchronised, as there is no way for us to determine the quality of the synchronization.

3.5 Traffic camera data and KPRecords

The Traffic Camera data was split across 2 separate files. We decided to combine all of this data into one single file, to make the data easier to work with. The first file contained the names and GPS coordinates for each of the locations, where the traffic enforcement cameras were located. The second file was an Excel file that contained an overview of all the vehicles that had passed by each camera, on a given day. There was one Excel-sheet corresponding to each camera on a given day. Since each Excel-sheet contained all the vehicles that passed on a given day, we first needed to identify which data entries represented the 3 trucks. This task was handled by the researchers themselves. They marked the relevant entries, by coloring them green, in the original excel file. We created a CSV file that combined the information from the two files. For each entry we recorded location information of the traffic camera, and attached all the information recorded about a truck, for that location. We thus have 3 entries for each traffic camera. We named the file KPRecord(KP=Kjøretøyspassering= Vehicle pass).

Chapter 4

Assessment of Data Quality

In this chapter we will we taking a high level look at the quality of various data sources. One of the research questions in this thesis is "How good is the data quality". In this section we will investigate all data sources, with both location data and timestamps. This includes GoPro GPS, Vbox, and Scania FMS. We are looking at the quality of the location data, and timestamps separately. For the location data we investigate whether the coordinates are accurate. We are also interested in whether the data points are consistent and whether the frequency of the data points is consistent. For timestamps we investigate whether the timestamps are accurate, whether the frequency is consistent, and whether there are any gaps in the timeline. By assessing all of these properties we can make a subjective determination as to the quality of the data.

4.1 Location

4.1.1 GoPro GPS

All of the GoPro GPS files, were visually inspected by a research assistant, hired by the researchers. Since the route driven was well documented, it is possible to asses the quality of the location data, by looking at whether the coordinates follow the road. The research assistant used the ArcGIS software to visualize the data on a map. The research assistant went through all of the GoPro GPS files, and recorded his assessment of each file into an Excel document. The quality of each file was categorized as Good, Decent or Bad. This assessment looked at a few different properties, and assessed each of them subjectively:

- Does the coordinates follow the road, and if not, are they at least close to the road
- Does the data points have a high frequency, and is the frequency consistent
- Are there any gaps in the data, and if so how prevalent are they

This list will help the researchers pick the GoPro files with the highest quality, for each truck and segment. Based on this file, we can conclude that all trucks have at least one decent quality file, for each segment. We can also conclude that all trucks have at least one good quality file, for most segments. We can also conclude that the quality of the location data from GoPro GPS files, is generally higher for day 2 that day 1.

4.1.2 VBOX

We assessed the quality of the VBOX files by visualising them on a Folium map. Only VBOX files from day one was assessed. The first thing we noticed, was that there were a lot of markers centered on the equator. To find the source of this, we decided to inspect one of the VBOX files manually. When inspecting the data we found that there were stretches of data where the coordinates were all set to 0. It appeared that the VBOX device would record both coordinates as 0, if it could not get a return from at least 3 GPS satellites. Looking at the map we can see that there are a few gaps in the data, and that the location data seems to be of high quality, except for these areas. Figure 4.1 shows an excerpt from a Vbox file, captured on day one. Only every 100 datapoints are visualized.

4.2 Timestamps

4.2.1 GoPro GPS

Since the GoPro GPS data serves as the basis for synchronization, it is important to assess the quality of the GoPro timestamps. We would like to answer the following questions related to the GoPro GPS data:

- Are there any outlier timestamps that differ significantly from surrounding timestamps.
- Are the timestamps evenly spaced in time
- Are the timestamps strictly ascending between resets
- Does the timestamps have a consistent frequency throughout the file

The GoPro GPS data appears to have a consistent frequency of approximately 125-126 milliseconds between each data entry. This frequency appears to be mostly consistent across all cameras, resets, and segments. Exceptions to this frequency was mostly caused by data-corruption.

During the synchronization step for the GoPro data, we detected that some of the files were far out of sync. We also detected that some of the files would gradually fall out of sync, towards the end of the file. Our first hypothesis was that this might be caused by a inconsistent frequency in the timestamps. We ran the correlation algorithm between a few of the GoPro GPS files and the VBOX file for truck 1, segment 1. We could see that the period from the start of day 1 until the first reset(about 15 minutes into the journey), had relatively good and consistent overlap between the timestamps for most of the files. We thus determined that most files had consistent timestamps between resets.

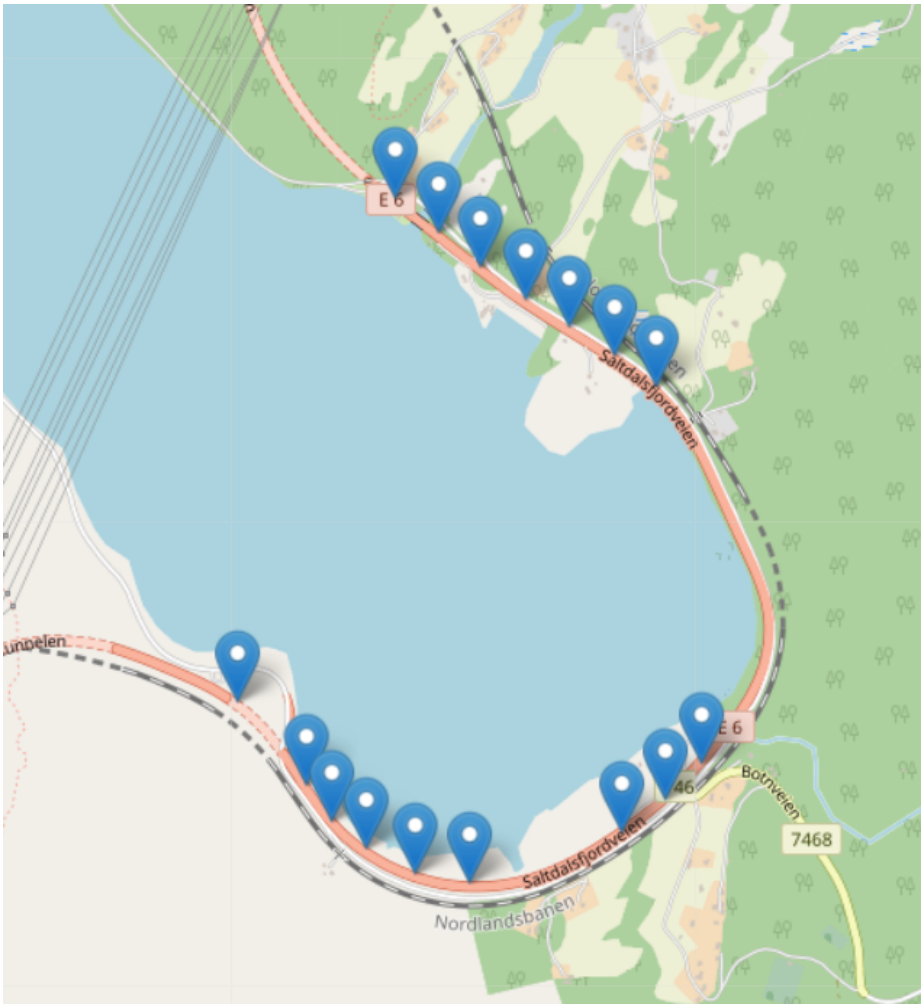

Figure 4.1: Data from a Vbox file, showing an area with poor covrage(Every 100'th datapoint shown)

The next hypothesis we wanted to investigate was that there was a delay between resets. If so we would expect the time difference to be consistent between resets, and to increase or decrease at the resets. Using the same correlation algorithm we determined that the time difference did increase around each reset. A research assistant at NTNU was tasked with figuring out how big this time difference was. He used the time difference estimation algorithm to determine that there was a difference of approximately 2050 milliseconds at almost all of the resets. This difference varied between 1810 milliseconds and 2100 milliseconds.

When compensating for this difference there were still a few files that fell significantly out of sync. The time difference in these files could vary between a few hours to a few years out of sync. Our hypothesis was that this may be caused by data corruption. We decided to search through the data to find the biggest and the smallest values. We found that some of the files contained values of 1 billion milliseconds or more, and some files contained values of -1 billion milliseconds or less. Since our implementation the time formatting algorithm relies on adding the last timestamp before a reset to the total time passed, a very big or small value just before a reset would cause the timestamps to fall out of sync. When inspecting the data manually, we found that some of the files did indeed have very big values just before a reset. We thus have to ignore these data points during the formatting process, in order to avoid the timestamps falling out of sync. Figure 4.2 shows an example of corrupted data points, with the first value before the comma of each row, representing milliseconds since start.

4.2.2 Scania FMS and Vecho FMS

We did a visual inspection of the Scania FMS data, since the frequency of data entries is relatively low in both the Scania and Vecho FMS data. We visualized the Scania FMS data on a geographical map, by placing a marker for each data entry on a map. This marker contained the timestamp for each data point. We discovered that there was a significant change in the frequency of recording inside of tunnels. Since there were quite a few tunnels on day 1, we decided to look at the data entries for day 1. We realised that Scania FMS would record a data entry, whenever it lost and then regained its connection to GPS. Scania FMS would also stop recording data when the truck was turned off. It would also reduce the frequency of recording when the truck was running, but not driving. This caused the Scania FMS data to have a variable frequency throughout the drive. Vecho FMS was inspected by a research assistant at NTNU. The conclusion from this preliminary assessment was that Vecho FMS had a highly variable frequency for data entries. This frequency varied between several times per minute, to every 10 minutes. Figure 4.3 shows an excerpt of the Scania FMS data, where the truck was driving through a tunnel on day 2.

3143,63.406948,10.4006424,322.756,0,0,1603269651873000,9999,0
3195,63.406948,10.4006424,322.756,0,0,1603269651925000,9999,0
3247,63.406948,10.4006424,322.756,0,0,1603269651977000,9999,0
3300,63.406948,10.4006424,322.756,0,0,1603269652030000,9999,0
2848,63.406948,10.4006424,322.756,0,0,1603269651578000,9999,0
2396,63.406948,10.4006424,322.756,0,0,1603269651126000,9999,0
1944,63.406948,10.4006424,322.756,0,0,1603269650674000,9999,0
1492,63.406948,10.4006424,322.756,0,0,1603269650222000,9999,0
1039,63.406948,10.4006424,322.756,0,0,1603269649769000,9999,0
587,63.406948,10.4006424,322.756,0,0,1603269649317000,9999,0
135,63.406948,10.4006424,322.756,0,0,1603269648865000,9999,0
-317,63.406948,10.4006424,322.756,0,0,1603269648413000,9999,0
-770,63.406948,10.4006424,322.756,0,0,1603269647960000,9999,0
-1222,63.406948,10.4006424,322.756,0,0,1603269647508000,9999,0
-1674,63.406948,10.4006424,322.756,0,0,1603269647056000,9999,0
-2126,63.406948,10.4006424,322.756,0,0,1603269646604000,9999,0
-2578,63.406948,10.4006424,322.756,0,0,1603269646152000,9999,0
-3031,63.406948,10.4006424,322.756,0,0,1603269645699000,9999,0
-3483,63.406948,10.4006424,322.756,0,0,1603269645247000,9999,0
-3935,63.406948,10.4006424,322.756,0,0,1603269644795000,9999,0
-4387,63.406948,10.4006424,322.756,0,0,1603269644343000,9999,0
-4840,63.406948,10.4006424,322.756,0,0,1603269643890000,9999,0
-4782,63.406948,10.4006424,322.756,0,0,1603269643948000,9999,0

Figure 4.2: A GoPro GPS file, opened in Excel, showing corrupted data points

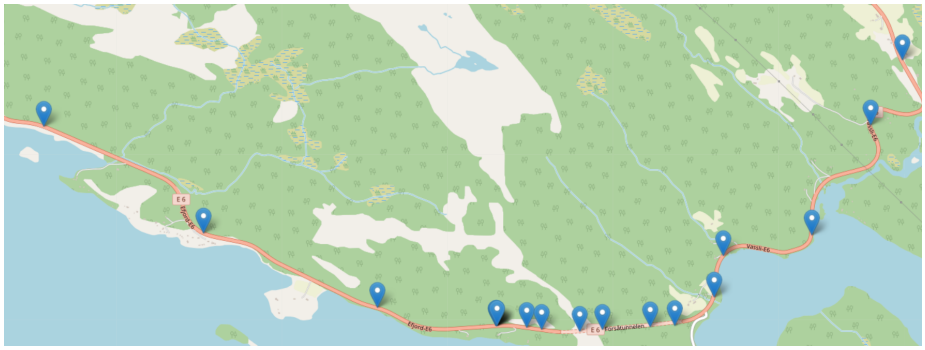

Figure 4.3: Data from a truck driving through a tunnel on day 2

Chapter 5

Integration of data from Nasjonal Vegdatabank

In order to answer which road features(if any) pose a challenge, we need information about all relevant road features, along the route driven during the transport platooning field experiment. The most comprehensive database on road features in Norway is Nasjonal Vegdatabank(NVDB). This is a publicly available database created by the Norwegian road authorities. In this thesis we use various publicly available tools to extract data from NVDB, and integrate this data into our data processing pipeline. This will allow the user to extract information about road features, alongside the information from all other data sources.

5.1 Relevance of NVDB data

Nasjonal vegdatabank (NVDB) is a publicly available database, containing information about all roads in Norway. This database contains a wide variety of information about various road features, and statistical data about segments of the road. Statens vegvesen(Norwegian road authorities) operates and maintains this database. They also provide various types of tools and interfaces to interact with the database, including a web app [13] and an open API. In this program we will be using the NVDB API to extract location information, corresponding to the NVDB data for our route. One of the research questions in this experiment is to identify which road features(if any) may pose a challenge to a transport platoon. It is thus relevant to look at road features when assessing the performance of the transport platoon in a given area. We thus decided to extract the relevant data from the NVDB, and integrate it into our data processing pipeline.

5.2 NVDB source files

A research assistant was tasked with extracting all relevant data features, from the route that was driven by the researchers. This data was extracted, such that each of the source files corresponded to the 5 segments driven on the 2 days of the experiment. For each of these segments, data was extracted in 6 categories: Horizontal curvature, Vertical curvature, Tunnels, Bridges, Speed limit and Road width. In each of these categories, the research assistant extracted all data features, the researcher determined to be relevant to the experiment.

5.3 Vegsystemreferanse

All road segments in NVDB are referred to using a system called "Vegsystemreferanse". This system uses a string, designed to be human readable, to label each segment of road in Norway. This string usually consist of road name, section, subsection(optional), and road meters. Road meters can either be specified as a point, or as a range(from a to b). Each range must be part of the same road, section and subsection. In order to extract data from NVBD we need to create a Vegsystemreferanse string, corresponding to each location we would like to extract information from. In this thesis, we determined that extracting data at a interval of 50 meters, would give us a sufficient data resolution. For each data entry in each of the Excel-sheets, we created a range of strings, representing a segment of a road given by the Vegsystemreferanse string. This range of strings represents a point every 50 meters from the start to the end of the segment. This range always include the first and the last location of a segemnt. If the segment is less that 50 meters, this range only includes the first and last location of the segment. This range of strings is the basis for extracting location information, for each data entry in the NVDB files.

5.4 Quering the NVDB API and Storing the information

In order to extract location information for each data entry, we use the NVDB API[12]. This API allows us to extract location information, by providing a Vegsystemreferanse string. We can state which format we would like the location information to be provided in. In this thesis we used "WGS84", which is the coordinate system used by GPS. The process of querying the NVDB API was done segment by segment. For each NVDB file the program iterates through all the different Excel-sheets. For each sheet the program creates a range of queries, corresponding to the Vegsystemreferanse road segment, listed in each data entry. For each data entry the program queries the NVDB, and records the location information representing that range. The program then stores all of the relevant data points for each data entry. It stores them with the location information, and the Vegsystemreferanse string for that location attached. All NVDB data is then stored in an Excel-file, with the same Excel-sheet names as the original file. This gives us a data set with both the NVDB data and GPS coordinates, for each data entry and category, for all segments of the route.

Chapter 6

Using and controlling the data processing pipeline

In this chapter we will discuss how the user can control the program. We will describe the different files the program relies upon, and how they control the behaviour of the program. We will describe the output formats, and their use cases. We also discuss the performance of the program, and how the performance is effected by the configuration.

6.1 Configuration file

The configuration file is the primary way the user can specify the data sources, and extraction of data. The configuration file is stored as a YAML file. The configuration file contains the following parameters:

- extraction mode: Lets thus user choose whether they want to extract data(True) or visualize data(False)
- filter on time: Lets the user choose whether to filter data based on time(True) or coordinates(False)
- file list: Lets the user specify the file list to be used, by entering the filename of the list.
- labels: Lets the user specify the label list to be used, by entering the filename of the list.
- output file name: Lets the user specify the output name of the data extraction file(excel only).
- output format: Lets the user specify the output format for data extraction(”excel” or ”csv”)
- start time and stop time: The start and stop time for data filtering. Must be formatted like: YYYY-MM-DD HH:mm:ss.fff

- start lat and start long: Lets the user specify the lower left corner of the "geofence" for coordinate filtering. Coordinates are specified in degrees, and are implicitly North and East
- stop lat and stop long: Lets the user specify the top right corner of the "geofence" for coordinate filtering. Coordinates are specified in degrees, and are implicitly North and East
- file types: Lets the user choose which file types to read(see Section 6.3 for more)
- trucks: Lets the user choose which trucks to read data from(see Section 6.3 for more)
- annotations file: Choose which annotation file to read data from for visualization
- reference file: Choose which GoPro or VBOX file to read data from for visualization
- map file name: Choose the filename for the map, created by the visualization algorithm

6.2 File lists

The primary file used to synchronize all data sources is the file lists. The file lists stores all the needed information about each of the files, that the program reads. This program has 2 separate file lists; one for each day of the experiment. The file lists are stored as CSV files.
The format of the file is:

- The first parameter is a string. It serves as a key , describing which data source the file belongs to. Examples of such strings include: "vbox"(VBOX data), "goprogps"(GoPro GPS data),"annot"(Annotation data) and "fms"(Scania FMS data). These strings are used to determine which function should read the given file. It also determines how the corresponding Data frame is treated during the filtering step. It is also used to determine whether a file belongs to a data source, that is supposed to be read, or filtered.
- The next parameter is an Integer, representing which truck the file belongs to. Since most files belong to exactly one truck this parameter exists for most files. This parameter is used to filter files based on trucks.
- Annotations and Heart rate data only: The next 1 or 3 parameters are positional parameters. They consist of Integers that are used to determine where in the file, certain data is located. Since some files contains data near the top of the file, that varies in length, positional parameters are required.
- For GoPro data and Radar data only: The next parameter is a timestamp, representing the time when the file starts. GoPro and radar files need this parameter for data fusion.

- All subsequent parameters, represent the path to the file, in comma separated format. The path is stored in comma-separated format, in order to make the path operating system independent. The path is joined into a string by the program, such that it works for any operating system.

6.3 Customisation of data sources and trucks

In order to give the user a greater amount of control over which data is being extracted, we decided to let the user choose which data sources and which trucks data would be extracted from. In the configuration file the user can select which data sources and trucks they wish to extract data from, by changing a boolean value that corresponds to each data source and truck. When the program launches it will pass this list of booleans as a dictionary to the file reading function. The function will then use the dictionary to determine which files to read, by using the data source type and truck number parameter, listed in the file list. Since all files that are supposed to be excluded are not read, there is no need for any other part of the program to remove any data sources. It also lets the researchers significantly increase the performance of the program, if they are only interested in a few data sources. Performance is increased by not reading the files with particularly high frequencies(GoPro,Vbox and Radar).

6.4 Output format

When extracting data from the pipeline in the "extract data" mode, all data that was not filtered out will be written to a file. There are two output formats for this mode: Excel(.xlsx) and CSV. If the user chooses Excel as their output format the program will write all the data to an Excel document. Each data frame will be written to a separate sheet, as long as it is not empty. Each sheet will use the filename variable stored next to the Data frame as the sheet name. Each filename contains information about which segment, data source and truck it comes from. There is thus no need to add any additional information to the sheet name. One drawback of writing to an Excel-file is performance. When writing particularly large data sets to an Excel file, the program seems to need a lot more time. This relation seems to be exponential between the size of the data set and time need to write all the data to the file. We thus decided to add the option to write data to a CSV file. If the user decides to write to a CSV file, the filename stored next to the data frame, will be used as the filename of the output file. Each data frame will thus be written to a separate file, if it is not empty. These CSV files can then be opened in excel, if need be. Our experience is that data sets encompassing more than 10 minutes, start taking more than a minute to extract as an excel file. We thus made the recommendation of extracting all data sets over this length as a set of CSV files. If extracting data using the "visualization" mode, the file will be output as an HTML file. This file can be opened in any web browser. The HTML file contains a map based on OpenStreetMap, with the markers representing our data, added to it.

Chapter 7

Filtering, Data Extraction and Visualization

In order to investigate sections of the data, we need tools that allows the user to extract subsets of the data. One of these tools is the filtering functions. The filtering functions allows the user to extract a subset of all the data read by the software. While the configuration files allows the user to select data sources, it does not allow the user to precisely select and extract a subset of each data source. In this chapter we describe the two different methods for extracting subsets of data. These filters are automatically applied to the data, when the software is run.

7.1 Filtering based on time

Filtering based on time is by far the simpler of the two filtering methods. All data sources except NVDB contains a timestamp for each entry. This means that we can use the same function to filter all data sources. Since all the timestamps across all the data sets were of a standardised format, we decided to use Pandas and the Datetime module for filtering based on time. The filtering works by passing three parameters to the function. Theese are the data frames(in a list), the start time and the stop time. Both the start time and stop time strings must be of the format YYYY-MM-DD HH:mm:ss.fff. Specifying the date, should in theory not be necessary, since the file lists determine the date. However due to the particular implementation of the DateTime module, specifying the date is required. The start and stop time is then converted to the pandas query format of YYYY-MM-DDTHH:mm:ss. In order to make use of the Pandas Dataframe query function, we first need to convert all the timestamps into Datetime Objects. Pandas has a built-in function for converting all timestamps in one column into Datetime Objects called toDateTime(). This function makes it possible to specify which format the function should read. The timestamp columns in all of our data frames are called "Timestamp". Because this mirrors the "Timestamp" keyword used by pandas queries, we needed to rename all the columns to "TimeS" in order to use the pandas query function.

The filtering is then done by running the query function on the entire data frame. The query function will remove all data entries except those that fall within the time range, specified by the start and stop timestamps. The data frame is then returned, irrespective if it is empty or not. Empty data frames are handled during extraction.

Since NVDB contains no time data, it is not possible to extract any NVDB data using the time filtering function. We considered implementing a method that would allow the extraction of NVDB data, using the time filtering function. This function would first extract time data from a reference file (For example a GoPro GPS file). It would then iterate through each entry of the NVDB data, and find the closest match between the reference file and NVDB, base on location. It would then copy the timestamp from the reference file into the NVDB data. There are a few downsides to this approach. First of all the quality of the NVDB timestamps would be only be as good as the reference. Secondly the algorithm would perform poorly in areas where the GPS data was of low quality, in the reference file. This would likely be a problem for the NVDB tunnel data in particular. We decided that implementing such an algorithm would be of limited use.

7.2 Filtering based on coordinates

Filtering based on coordinates makes use of both coordinate filtering and time based filtering. In order to extract data based on coordinates, the user must specify a "geofence". This geofence represents the area from which information will be extracted. In order to specify the geofence, the users must provide two locations. Since all of our data(except corrupted data entries) lies to the northeast of the equator, the coordinates implicitly represent degrees north and degrees east. The locations are specified using two float values. The first represents degrees Longitude, and the second represents degrees Latitude. The first location represents the lower left corner of the geofence. The second location represents the top right corner of the geofence. From these two points, a square is created that represents the geofence. Before the filtering takes place, all data frames are split into two categories. The first category includes all data sources that contains GPS coordinates for all data entries. The second category includes all other data sources. All data frames in category one in passed to a function that uses the Pandas data frame query function. This function takes in one data frame and 4 coordinates. Since latitude and longitude is stored as a string in the data frames, we first convert the strings into their corresponding float values. We then use the Pandas query function to filter out all data entries, that does not fall within the two longitude values. Since the route mostly travels along the north south axis, we expect that filtering by longitude first will remove more data entries, thus making the algorithm more efficient. We then use the query function to filter out all data entries, that does fall within the two longitude values. All data entries that remain are within our geofence. The data frame is thus returned(even if empty).

In order to extract information from all data sources we need to establish a time range. This time range corresponds to the time the trucks spent, within the geofence. This will usually be a GoPro GPS file. If the user selects NVDB as the only data source that contains coordinates, no data will be extracted based on time, from this algorithm. This is because it is impossible to establish a time range from NVDB data alone. In order to extract this time range, the algorithm will pick the data frame from the first category, that has the greatest number of data entries after filtering. We then take the first and last timestamp, from that data frame. These two timestamps represent the start and stop time for use in the time based filtering portion of this algorithm. We also attempted to extract the earliest and latest timestamp. This strategy often failed, because some of the GoPro files contained corrupted data entries, with incorrect timestamps. This would often cause the time range to be incorrect, since the corrupted timestamps would often be the earliest or latest values in the data. After these two values has been extracted, the algorithm will extract data from all data sources that does not contain coordinates. This part of the extraction uses the algorithm described in section 7.1.

There are some known weaknesses to the time based extraction strategy. It is possible that geofence encompasses an area, such that the trucks leave the area, and later reenters it. If so the time range will include sections, that are not included in the data extracted based on coordinates. In some cases it might be possible that the first or last data entry, in the reference file may be corrupted. If so, then the time range may differ from what we would expect. If the quality of the location information is poor, as the truck enters or leaves the geofence, this may impact the time range that is extracted.

7.3 Visualization

The second mode for extracting data is the visualization. This mode allows the user to visualize some of the data on a map. Only data sources containing coordinates for each data entry is eligible for visualization. The exception is the annotation visualization algorithm described bellow. The visualization functionality available to the user is limited to visualizing annotations. There are, however, a few algorithms available for visualising other data. These algorithms are used for testing only. These algorithms use lists instead of data frames, as they were created early on in the thesis, and has not been redesigned to work with data frames.

In this program we use Folium for visualization. Folium makes use of data from OpenStreetMap, to generate a map with geogrphical data. We then add all the markers to the map, with each marker representing one data entry. The map is then stored as a HTML file, which can be opened in any browser. The map is interactive, and lets the user click on any marker, to display the associated data. Folium also supports drawing lines between points. A general restriction for all the visualization algorithms has to do with performance. If more than 1000 markers are added to the map, performance starts to drop significantly. Around 2000 markers the performance deteriorates to the point, where using the map becomes difficult. We thus have to be careful about how many markers we add to the map. This can be done by limiting the area, the frequency or the

time range of the data, we visualize. The number of data points stored in each marker does not effect the performance at all.

The first of the testing algorithms is the basic visualization algorithm. This algorithm takes in data in a list, and plots every n'th data point on a map. The user can select how many data entries to visualize by selecting how many data entries to skip in between. This will allow us to visualise an arbitrarily large data set. For data sources with a very high frequency this algorithm lets us visualise a sample of the data, without having to sacrifice the performance of the map. This visualization strategy is not very robust, as it does not select data entries based on quality. The resulting map will have markers with a varying amount of space between them. It will also have a lot of markers around the locations, where the researchers stopped to rest.

The second of the testing algorithm is the space visualization algorithm. This algorithm also takes in a list of data, and plots a selection of data points on a map. Unlike the basic visualization algorithm, this algorithm tries to space out the markers along the route. The user must specify a limit, that determines how many meters apart the markers should be. The algorithm will add the first data entry to the map. It will then iterate through the data entries, and select the first data entry that is far enough away from the first data point(in a straight line), to exceed the limit. It will then select the first data entry that is far enough away from the second entry. It will continue using this strategy to iterate through the entire data set. This approach is susceptible to data points with low quality.

The third algorithm is available to the user. This algorithm allows the user to visualize annotations along the road. Since the annotation file does not contain any location data, we must combine it with data from another data source. It is theoretically possible to combine the annotations with any data source, but we recommend using a GoPro GPS file, or a VBOX file. GoPro GPS and VBOX files have a high frequency of data entries, which is a big advantage for this algorithm. To visualize the annotations we first create a new Data frame, that contains the annotations, and coordinates for each data entry. This is done by using the Pandas mergeAsOf function. This function is able to combine a two data frames into one, by matching each data entry in the first Data frame, with a data entry in the second Data frame. This function can match by any type of value, but in our case we are matching based on timestamps. The function also allows us to specify a tolerance. Both the timestamp and tolerance has to be a Datetime object. The function then returns a new Data frame containing all columns from both data frames, joined by their timestamps(+/- tolerance). We set the tolerance to two seconds in this function. If we wanted to use a different data source, the tolerance might need to be adjusted. Once we have the augmented Data frame, we can proceed to the visualization part. The first step is to format each data entry, such that it fits into the marker-popups. This is done by placing each data point on a separate line, next to its label. All of the data entries are then placed on the map, and saved to an HTML file. Figure 7.1 demonstrates a visualization made using the third algorithm.

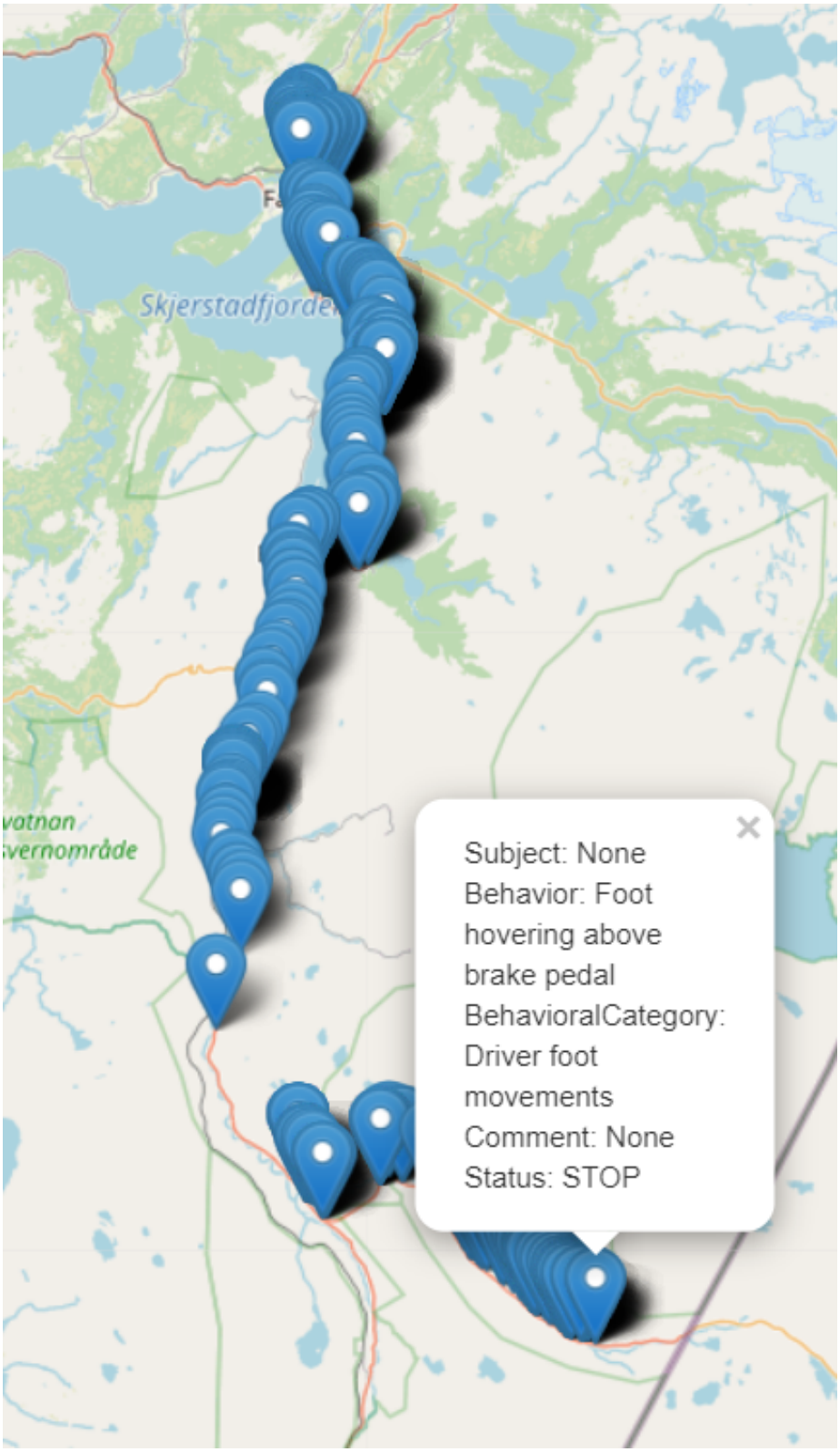

Figure 7.1: Visualization of annotations using the third algorithm. The first two algorithm produces a similar looking map

7.4 Discussion

During some parts of the experiments, the trucks drove certain parts of the route multiple times. When extracting information from these re-runs, the two different algorithms will work somewhat differently. Since filtering based on time will extract a known time range, the user will be able to extract individual sub segments. It is thus possible to extract each re-run separately. Filtering based on coordinates, will on the other hand extract all data entries, for all the re-runs in one location at once. We thus recommend exclusively using filtering based on time, when extracting information on re-runs. We recommend using extraction based on location, only when the NVBD data is required.

7.5 Applying filters

As discussed above, the filter is applied automatically, when the software is run. The filters cannot be disabled. The user must manually determine the time range. The coordinates can be determined using any map tool, that allows the user to extract GPS coordinates. Using filtering based on time, it will not filter out any data, if the specified time range starts before 2020-10-20 08:30, and ends after 2020-10-20 18:00(except some FMS data). The FMS data includes some data points from the 19th and 22nd of October, which are not relevant to the transport platooning experiment. The heart rate data starts at 8:30 which is approximately 2 hours before most other data sources. If a time range spans multiple segments of the drive, data will be extracted from all applicable segments. This also applies to geofences.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

In this thesis we have described our approach to processing data from a transport platooning experiment. We have described all the different sensors, and how they were used in this experiment.

We have looked at how we can combine all the data gathered into one data set(RQ1). This is done by creating a customized function for reading each format into a data frame. We convert various types of timestamps into ISO8601 compliant timestamps. We also convert all coordinates, into numeric only GPS coordinates, denominated in degrees north and degrees east. We also standardize the labels across all of the different data sources, and provide the user with the means to seamlessly change them, if necessary.

We have looked at how to synchronise all the gathered data into one timeline(RQ2). We decided to use the GoPro data to serve as the "ground truth", to which all other data sources would be synchronized. The transcripts and Radar data was synchronized by calculating the offset between the all the files in these data sources, and the GoPro data. The Annotation data was synchronised using the timestamp at the top of the file, as this was based on the corresponding GoPro data. The Vbox data, the Scania FMS data and the traffic enforcement camera data were synchronised by calculating the offset between them, and the GoPro data. This offset was calculated by looking at how the coordinates in the various files lined up, with the GoPro data. Finally the heart rate data was synchronised, through referring to the notes, the researchers took during the transport platooning experiment.

We have looked at how we can select, and extract segments of the data(RQ3). We created two separate filtering functions, for extracting subsets of data. The first of the filtering function extracts all data that falls within a specified time range. The second filtering function extracts all data gathered within a geographical area. We have also developed a method for selecting which data sources, trucks and day(s) to extract data from. By applying either of the filters to our data set, and selecting which data sources to use, we are able to precisely select and extract any subset of the data gathered during this experiment.

We have analysed the quality of some of the main data sources(RQ4). We have analysed the quality of the GoPro data, the Vbox data and the Scania FMS data. We have investigated the accuracy, frequency and consistency of the location data. We have also investigated the accuracy of the timestamps, and whether there are any gaps in the data, or inconsistencies in the frequency of the data.

We find that some of the lessons learned in this thesis will aid in planning future experiments, of this kind. The work described in this thesis will serve as a foundation for future work on the data. We believe that it will contribute to our understanding of how well transport platooning works on rural Norwegian roads. The data set created by this program, enables the researchers to extract and analyse segments of data from the journey. This will aid in answering the research questions posed for this research project.

8.2 Statistical analysis

One of the purposes of this thesis is to prepare a data set that can be used for analysing the performance of a transport platoon. The natural next step is thus to begin analysing the data. We are interested in investigating, whether there are any particular road features that poses a challenge to the transport platoon. It would therefore be interesting to investigate, if there are any statistical correlations between the events listed in the annotation files, any of the features in the other data sources. It could also be interesting to investigate the correlation between road features, and data quality. We could also investigate, if there is a correlation between the quality of the data across various data sources. This could be particularly interesting in areas where multiple similar data sources had low quality simultaneously. This analysis could also be a candidate for an analysis, using machine learning techniques.

8.3 Visualisations

In this thesis we have visualised annotations on a map. There is a great potential for creating other visualisations based on the data.

- One visualization that was suggested by the researchers is a correlation between the data entries in the annotation file, and certain road features from the NVDB files. An example given would be the correlation between horizontal/vertical curvature and pedal applications.
- Another possibility is to visualize locations that meet certain criteria, based on one or more data sources. An example would be locations with a significant horizontal or vertical curvature, or with a narrow road.
- A third type of visualisation could be various heat maps. These heat maps could visualize how the various road features were distributed along the route. The heat maps could also be overlaid, with the annotations.

Bibliography

[1] American Transportation Research Institute (ATRI). *An Analysis of the Operational Costs of Trucking: 2020 Update*. https://truckingresearch.org/wp-content/uploads/2020/11/ATRI-Operational-Costs-of-Trucking-2020.pdf. (Visited on Nov. 1, 2020).

[2] Inc. American Trucking Associations. *Driver Shortage Update 2021*. https://www.trucking.org/sites/default/files/2021-10/ATA%20Driver%20Shortage%20Report%202021%20Executive%20Summary.FINAL_.pdf. (Visited on Oct. 25, 2021).

[3] Dipayan N. Chowdhury et al. "A Vehicle-to-Vehicle Communication System Using Iot Approach." In: *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2018, pp. 915–919. DOI: 10.1109/ICECA.2018.8474909.

[4] Pandas dev. *Pandas*. https://pandas.pydata.org/.

[5] AT.no Editorial staff. *Statens vegvesen testet «tungbil-tog» på nordnorske veier*. https://www.at.no/transport/498905. (Visited on Oct. 26, 2020).

[6] Charlie Clark Eric Gazoni. *OpenPyXL*. https://openpyxl.readthedocs.io/en/stable/.

[7] Folium. *Python data, leaflet.js maps*. https://python-visualization.github.io/folium/.

[8] Dhaya Kanthavel, S.K.B. Sangeetha, and K.P. Keerthana. "An empirical study of vehicle to infrastructure communications - An intense learning of smart infrastructure for safety and mobility." In: *International Journal of Intelligent Networks* 2 (2021), pp. 77–82. ISSN: 2666-6030. DOI: https://doi.org/10.1016/j.ijin.2021.06.003. URL: https://www.sciencedirect.com/science/article/pii/S2666603021000105.

[9] Transport Intelligence Ltd. *European Driver Shortages*. https://www.ti-insight.com/whitepapers/european-driver-shortages/?whitepaperTitle=European%20Driver%20Shortages. (Visited on Sept. 19, 2021).

[10] Anteral S.L. *uRAD Universal Radar at 24 GHz for Raspberry Pi*. https://urad.es/wp-content/descargables/uRAD%20-%20Datasheet%20-%20RaspberryPi%20v1.2%20-%20EN.pdf.

[11] Akshay Upadhyay. *Haversine Formula – Calculate geographic distance on earth*. https://www.igismap.com/haversine-formula-calculate-geographic-distance-earth/.

[12] Statens Vegvesen. *NVDB API*. https://nvdbapiles-v3.atlas.vegvesen.no/.

[13] Statens Vegvesen. *Vegkart*. https://vegkart.atlas.vegvesen.no/.