University of Tartu
Institute of Computer Science
Conversion Master in IT Curriculum

Edgar Sepp

Creating High-Definition Vector Maps for Autonomous Driving

Masters' Thesis (15 ECTS)

Supervisor: Tambet Matiisen, MSc

Tartu 2021

Creating High-Definition Vector Maps for Autonomous Driving

Abstract:

Autonomous driving holds many promises for transportation - increased safety, lower costs, and less burden to the environment. In light of some recent accidents, it is clear that the technology is not fully ready yet, and the robustness and research in the area need to be increased.

Most of the autonomous driving solutions rely on high-definition maps (HD maps) - specialized lane-level maps with very high locational accuracy. Mobile mapping cars (specially equipped vehicles with sensors for map data collection) by big mapping companies are used to collect the data for creating HD maps. Along with required data processing the creating and keeping the HD maps up to date in a changing world is very costly. Availability of the HD maps would considerably lower the bar for adopting autonomous driving at large.

To the best of the author's knowledge, there are no freely available HD maps for self-driving available for Estonia. To be able to conduct research experiments with the University of Tartu's Autonomous Driving Lab (UT ADL) self-driving platform, such maps had to be created. Several available tools for creating the maps and existing data sources were reviewed. The custom workflow was created for mapping and a tool to convert the HD vector map to Autoware vector map format was created. Finally, quantitative measures about time estimates needed to create the HD vector maps and their usage in UT ADL were given.

Keywords:

high-definition maps, autonomous driving, Autoware

CERCS: P170 (Computer science, numerical analysis, systems, control)

Täppiskaartide loomine isejuhtivatele sõidukitele

Lühikokkuvõte:

Isejuhtivate sõidukite tehnoloogiale pannakse suuri lootusi transpordi valdkonnas: kasvav turvalisus, kulude ja keskkonnamõjude vähenemine. Mõnede viimase aja õnnetuste valguses on aga selge, et tehnoloogia ei ole selleks veel päris valmis ja selle töökindlust ning taset on vaja tõsta.

Suurem osa isejuhtivatest tehnoloogiatest kasutavad spetsiaalseid kõrge lahutuse, suure asukohatäpsuse ja sõiduradade detailsusega vektorkaarte - täppiskaarte. Neid kaarte toodavad enamasti suured kaardifirmad, kasutades spetsiaalselt selleks kohandatud sensoritega varustatud mobiilseid kaardistusautosid. Kogutud andmete järeltöötlus ja täppiskaartide ajakohasena hoidmine pidevalt muutuvas keskkonnas muudab sellise kaardistusprotsessi üpris kulukaks. Selliste kaartide olemasolu muudaks isejuhtivate tehnoloogiate laiema kasutuselevõtu tunduvalt lihtsamaks.

Autorile teadaolevalt Eesti kohta vabalt kättesaadavaid isejuhtivatele tehnoloogiatele mõeldud täppiskaarte olemas ei ole. Võimaldamaks Tartu Ülikooli isejuhtivate sõiduki labori isejuhtiva autoga vastavaid eksperimente läbi viia tuli vastavad täppiskaardid luua. Töös antakse ülevaade erinevatest töövahenditest ja andmeallikatest. Analüüsi tulemusena loodi kohandatud töövoog kaardistuseks ja programm täppiskaardi konverteerimiseks *Autoware vector map* formaati. Täppiskaartide loomiseks kulunud aja, mahu ja nende kasutamise osas tuuakse välja ka kvantitatiivsed näitajad.

Võtmesõnad:

Täppiskaardid, isejuhtivad sõidukid, Autoware

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1. Introduction

Autonomous driving (AD) holds the promise to radically lower the costs of transportation, increase safety, and through sharing of the vehicles also make it less of a burden for the environment [1], [2]. Technology is getting cheaper and more accessible, lowering the bar for research in the field. More and more research groups are involved and autonomous cars are included in many strategic plans and roadmaps. Some recent accidents show that the optimal approach is not here yet and the robustness of the state-of-the-art in autonomous driving needs to be increased [3].

One fundamental separation between approaches is modular and end-to-end driving. The modular approach divides self-driving into smaller logical modules and one of the important parts of the approach is using a special high-definition map (HD map). End-to-end driving solves autonomous driving usually with one big neural network where control commands are calculated directly from sensory inputs. The proponents of this approach have been against using HD maps as they are very costly to make and maintain. Still more examples of modular approach and relying on HD maps exist. The best examples that have reached the farthest in this field, like Waymo[1] and Cruise[2], which are testing with fully driverless cars, are using HD maps.

HD maps are lane-level maps with very high locational accuracy providing a lot of information about road geometry, different traffic regulating elements, and road surroundings. These can be part of the localization model of HD maps [4]. It means that maps and their features could be used as a reference for locating the self-driving vehicle using its perception [5]. Autonomous driving needs very precise localization (a few cm level accuracy is needed) to be able to utilize HD maps to their full potential and navigate complicated urban traffic. HD maps can be used to extend the vision beyond the normal sensory range and concepts like electronic horizon [6] are introduced and included in behaviour planning based on the semantic information included in HD maps.

High-definition maps do not exist for Estonia to the best of the author's knowledge. At least these maps are not freely available. To conduct research in autonomous driving in Estonia these maps are needed and the only option is mapping. Mapping is mainly done by big companies using fleets of mobile mapping systems (MMS) - cars that are expensive to set up and maintain [38]. Mapping is collecting the needed data, processing it, and packaging it into the necessary format. Big companies use a lot of automation and AI to do the mapping. Setting up these pipelines from zero is mainly laborious manual work and therefore costly, needing also a lot of time. Constant changes in the environment and the need for being up-to-date to guarantee safety puts in place the requirements that have not been there before. One way to tackle it seems to be constant monitoring through crowdsourcing [7]. Additional requirements for HD maps are dictated by the software and hardware platform used, data quality needs, and resources available. Availability of the maps would lower the costs and difficulties of getting started with the self-driving car experiments.

[1] https://blog.waymo.com/2020/10/waymo-is-opening-its-fully-driverless.html
[2] https://www.theverge.com/2020/12/9/22165597/cruise-driverless-test-san-francisco-self-driving-level-4

The current thesis investigates what is necessary and how to create high-definition vector maps for the self-driving platform at the Autonomous Driving Lab (ADL) of the University of Tartu. The most optimal solution for current requirements is sought that would help to start autonomous driving experiments with the ADL's platform on the street.

The goals of this thesis:

1. Overview of available data sources, tools, and existing HD map formats that could be used in mapping workflow.
2. Develop a mapping workflow that meets the needs of the ADL platform
    a. define reality and data model of collected data;
    b. develop a conversion script to the necessary HD vector map format.
3. Give initial quantitative estimates about mapping workflow and future perspectives.

2. Background

This section will give an overview of autonomous driving and its relation to maps. An introduction to high-definition (HD) maps - special maps for enabling autonomous driving - is given. HD maps can be used only when an autonomous vehicle can locate itself on the map, so the main localization techniques are introduced along with the concepts used. At the end of the section, a short review of different mapping methods is given and main HD vector map formats are introduced.

2.1. Autonomous Driving

Autonomous driving (AD) can be best explained with the image from SAE (Society of Automotive Engineers) in Figure 1, where 5 levels of automation are described [8]. Levels below 3 can be considered as Advanced Driver Assistance Systems (ADAS) that can help in some situations, but the driver must constantly supervise and take control to maintain the safety of the driving. Examples of ADAS are automatic emergency braking, forward collision warning, blind-spot monitoring, lane departure warning, and lane-keeping assistance systems. Only levels 4 and 5 do not require fallback to the driver. In the case of level 4 AD is possible only in specified conditions called Operational Design Domain (ODD) of the system.

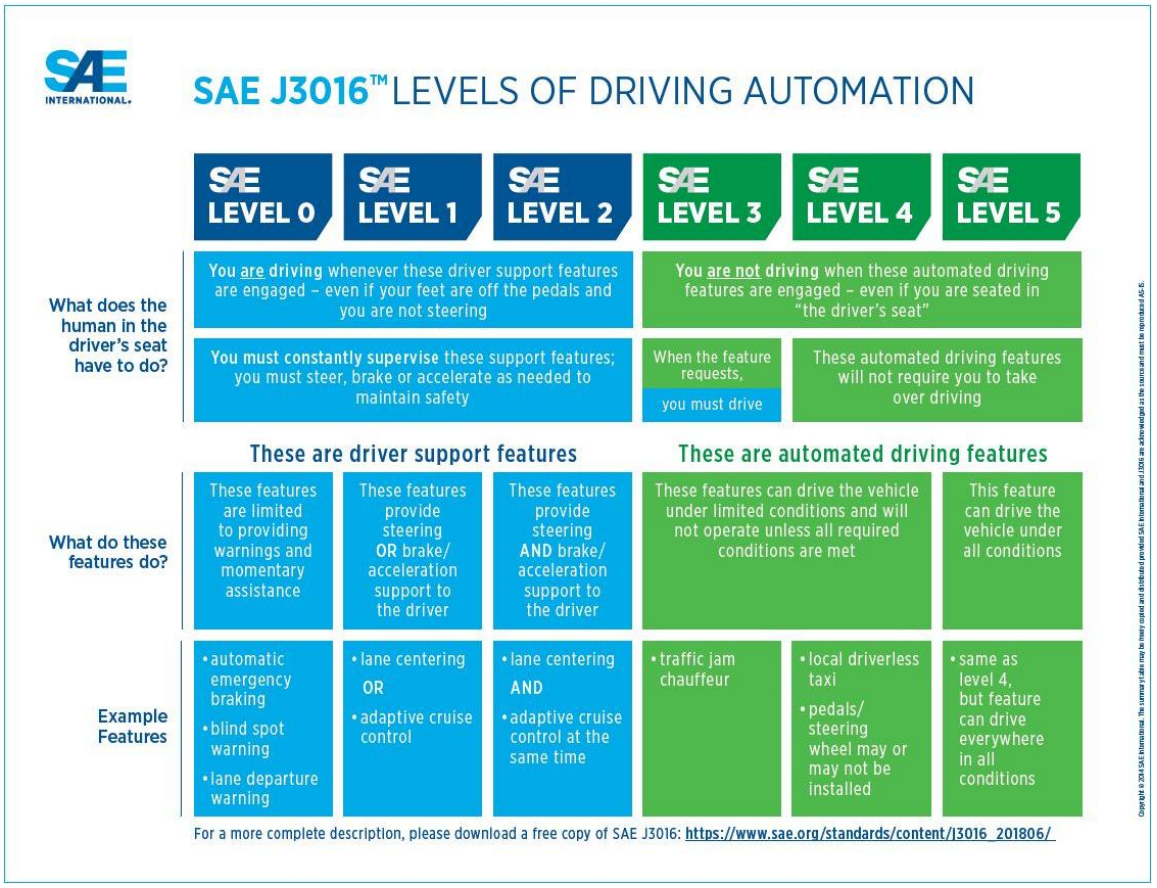

Figure 1. Five levels of driving automation by SAE[3].

[3] https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic

Main expectations related to AD and ADAS are fewer accidents, reducing driving stress, emissions, and the possibility for current mobility-impaired to be able to use cars [1]. Similar hopes are brought out also by Montgomery et al. [2]: road casualty reduction, less energy consumed, traffic congestion mitigation, and increased productivity at the expense of the time that was spent on doing the driving. ADAS could help prevent a combined total of 40% of all passenger-vehicle crashes [9]. In contrast to previous hopes, several accidents related to AD have happened that undermine the trust and adoption of AD technology. According to Yurtsever et al. (2020), the technology is not fully ready yet and the robustness of the state-of-the-art needs to be increased [3].

There are two general approaches to solve the full AD: modular and end-to-end approach. In the case of the modular approach, the tasks of AD are divided into separate modules: perception, localization, mapping, planning, decision making, and vehicle control [3]. The modular approach is considered simpler to develop because the subtasks can be independently tested and debugged. The major disadvantage of this approach is error propagation through the pipeline (imagine the misclassification of the object that might not trigger the correct response at later planning stages) [10]. In the case of an end-to-end approach, the control commands for the self-driving car and its motion are generated directly from sensory inputs using some sort of machine learning approach (usually neural networks). The biggest problems with this approach are the interpretability of the actions, lack of hardcoded safety measures [11] and with few intermediate outputs, it is difficult or impossible to understand why the model misbehaves [12].

While it is obvious that AD would not be possible without the sensors that help to solve the perception tasks there has been quite a big confrontation about the special maps for enabling AD - high-definition maps (HD maps). It could be aligned with two approaches to solve the AD: the modular approach needs HD maps and end-to-end driving does not. Of course, it is not so straightforward as there are many different approaches to solve the AD problem and there is no agreed optimal solution [3].

2.2. Maps for Autonomous Driving

End-to-end AD company Wayve has said on their webpage[4]: "*We're building artificial intelligence capable of driving in any urban environment, anywhere in the world. To do this, we are pioneering end-to-end deep learning which can drive in complex, never-seen-before environments… Traditional systems rely on expensive Lidar sensors, HD maps and heavy testing in a local area. This means they overfit to their geofenced area and can't generalize to multiple cities.*" In a blog post by Lyft (another company developing AD but with a modular approach), we can read[5]: "*Maps are a key component to building self-driving technology.*" Everybody agrees that making maps and keeping them up to date is costly, but still, a lot more map-based approaches can be found. It is understandable that big mapping companies

[4] https://wayve.ai/

[5] https://medium.com/lyftself-driving/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758 d6

like HERE[6] and TomTom[7] support the idea of maps playing an essential part in unlocking the solution to AD, but there are also other supporters that do not make money with mapping (Lyft, Waymo). What is not so definite and agreed upon is what these maps should contain and what formats should be used. What is common though is that the map for AD is not referred to as a simple map, but a high-definition map (HD map). Sometimes also the term Highly Automated Driving map (HAD map) is used [13]. In the current thesis, the term HD map is used.

Another interesting axis is where at one end there are big automotive industries and original equipment manufacturers (OEM) and at the other end, there are research groups. While research groups mostly rely on open-source and free tools having all very customized setups, approaches, and research questions. For small research groups, the map data from big providers are locked inside the navigation systems. It is not possible to use maps with the proprietary format from navigation systems for their AD experiments because there is also no support from the software side. The automotive industries depend on more closed and proprietary hardware and software when trying to scale up the solutions. Hence much more initiative is also put on agreeing on the standards, like Navigation Data Standard Association[8].

HD maps can be used if a vehicle can be localized relative to a map and for that sensors are needed. Sensors are necessary also because AD requires the precise and robust perception of the environment [14]. Different sensors can be used for localization, the most common solutions are based on Global Navigation Satellite Systems (GNSS) and lidar, both fused with Inertial Measurement Unit (IMU) and odometry data, additionally, also radars and cameras are used [15]. Sensors are mainly divided into two categories: exteroceptive and proprioceptive. Exteroceptive sensors are used to acquire information about the surrounding environment of an ego vehicle. Sensors in this group are camera, lidar, radar, ultrasonic sensors, and GNSS. These sensors retrieve information about obstacles (cars, pedestrians), but also buildings, drivable area, traffic lights, and signs. Proprioceptive sensors continuously measure and monitor the state of the ego-vehicle itself. Information about speed, acceleration, location, and yaw are constantly measured by wheel encoders, IMU, and tachometers.

HD Map can be treated as a sensor enabling the car to see further and compensate for bad weather conditions or sensor failure, this concept is introduced as Electronic Horizon [6]: "*The map data, e.g. road geometry (curvature, slope), number of lanes, speed limits, etc. is provided on a vehicle bus system as the so-called Electronic Horizon.*" The importance of the map is supported also by the Navigation Data Standard Association: "*maps should serve as a common location reference, the common canvas on which vehicle and infrastructure sensor data are combined to paint a realistic picture of the world. The map helps the car understand the world around it. And the map also includes the data that vehicle and infrastructure sensors cannot provide. An example of such non-detectable data are rules of the road that are not posted on a sign*" [16]. In the presentation by Mobileye[9] it is said that HD map can also

[6] https://360.here.com/2015/04/16/autonomous-cars-can-understand-real-world-map/
[7] https://www.tomtom.com/products/hd-map/
[8] https://nds-association.org/
[9] https://www.youtube.com/watch?v=HPWGFzqd7pI

help with features that are really hard to get with just perception: like stopping and yielding points, common speed, lane and traffic light association. HD map also adds contextual awareness and helps to decide if there is an overpass, obstacle, or the sign above the road.

AD needs robust perception because this is the foundation for higher-level systems: path planning, control algorithms, object detection, decision making [14]. Whether the HD maps are part of the AD depends on the approach, but there are definitely more examples relying on the HD maps.

2.3. High-Definition maps

Maps to support full AD (level 3 to 5) are called high-definition (HD) maps. The concept of HD maps emerged in 2010 growing out of Mercedez-Benz's research plan to make a fully autonomous drive on the 125th anniversary of Bertha Benz's drive[10]. They took the detailed road maps from that time (ADAS maps), added a lane model (lane level mapping and connectivity with added stop lines, crosswalks, traffic lights) and localization layer (visual cues for the car to help with localization) and that was the foundation for HD maps. In 2013 they made the drive as Mercedes Benz S-Class S 500 completed the same 103 km long trip as Bertha Benz fully autonomously using an HD map [17]. The map format used is called lanelets and is introduced more thoroughly in [18].

There are ADAS maps for navigation that can provide information about the steps needed to reach the destination and assist in some driving tasks and then there are HD maps that provide information about the surrounding with centimeter-level precision [14]. A good overview of the differences between HD and ADAS maps is brought out in the TomTom webpage (Table 1).

According to [4] HD map content is divided into 3 models all consistently geo-referenced:

- Road model - used for general strategic planning, like navigation.
- Lane model - used for perception and tactical planning (guidance), includes detailed and feature-rich lane level data
- Localization model - help to localize self-driving vehicles on the map.

The main task of the road model would be routing: generate a global plan from start to endpoint. The topological structure is a priority and centimeter-level accuracy is not that important [19]. For routing the road network can be envisioned as a directed graph, where nodes are interconnections between lanes and edges represent road segments. In the case of local planning (for example, guiding the car over an intersection, avoiding obstacles, stop for traffic lights) centimeter-level accuracy is needed. According to some approaches, lane level data should be represented as curves, arcs and splines (or some other mathematical formula) [20], not as a sequence of points. Typical tasks constantly solved during AD would be where the ego vehicle is relative to the lane or what is the curvature of the turn and by having the

[10] https://360.here.com/the-evolution-of-the-hd-live-map

lane as a mathematical function these calculations could be more efficient than having a lot of points and another aspect could be less amount of memory needed for storing the lanes.

An accurate lane model can help only if the localization accuracy relative to the lane network is at the same level [19]. This is not an easy task to localize the ego vehicle with the precision of a few centimeters having six degrees of freedom (XYZ, roll, pitch, yaw). The correct pose is needed to accurately determine the field of view of the sensors [4]. The localization model of HD maps is meant to help to solve this problem by having detailed and precise landmark feature layers that can be used in the localization process. The process itself is more thoroughly discussed in the following localization chapter but simply put: real-time data from onboard vehicle sensors is fused with map data and the exact position of the ego vehicle in relation to all landmarks is found[11]. Examples of these localization models can be Road DNA by TomTom and HD Localization Model by Here [4]. As AD is expanding and operating in more complicated environments more new features are added [7]. Adding new features means mapping and it is considered to be quite an expensive process. Big mapping companies usually have fleets of special mapping cars (Mobile Mapping systems - MMS) that are used to drive all the public roads and collect the data.

Table 1. Comparison of ADAS and HD maps based on TomTom[12].

	ADAS map	HD map
Locational accuracy	Few meters	Few centimeters
Lane level information	Only at intersections specifying possible turns	All throughout the map
Additional information	Road level ● Traffic signs ● Speed restrictions ● The curvature of turns at intersections ● Slope of road	Lane level ● Traffic signs ● Speed restrictions ● Lane geometry (including curvature and slope) ● Road furniture
Level of attribution	Minimal: location and value/meaning	Very detailed: location, height, width, meaning, category, subcategory, position relative to the road, color, text, lane associations

Constant remapping in some large rural areas might be impractical due to fast changes, like vegetation in the roadsides [21]. It can severely affect localization if point cloud, which is not up to date, is used. Point cloud in such a case is really not practical. The other known problem with point clouds is the storage problem. They need a lot of memory to store all the data. The concept of “sparse” maps is opposed to high density maps and according to Ma et

[11] https://www.intellias.com/solving-the-challenges-of-hd-mapping-for-smart-navigation-in-autonomous-cars/
[12] https://www.tomtom.com/blog/autonomous-driving/adas-map-vs-hd-map/

al. [22], their sparse maps need orders of magnitude less storage and can perform at the same level utilizing lanes, traffic signs, and vehicle dynamics for localization.

Lyft offers 5 level HD map view in their blog[13]. They divide the map into five layers:

- Base map layer (standard definition map - road network);
- Geometric map layer - this is all about the 3D view of the world, trajectories, ground model, voxelized geometric map (more efficient than using processed point cloud);
- Semantic map layer - builds on top of the geometric map and adds semantic objects (intersections, crosswalks, stop signs, parking spots, bus lane, cycle lane…). Lanes, stopping, yielding points, connectivity graphs at intersections;
- Map priors layer - add behavioral aspects about the static and dynamic elements of the map with the option to have time and weekday separation, some examples are traffic light cycles, parking behavior in street, traffic flow speed;
- Real-time layer - used to read/write real-time traffic information and observations.

All layers must be perfectly aligned and indexed that would allow efficient parallel lookups in the current location and neighborhood. The geometric layer is all about the 3D representation of the world to which the semantic layer adds the understanding and meaning about the physical and static parts of the world. Map priors are used to modify the behavior of the ego vehicle. An example could be the standing car in front of the ego vehicle. If there is a pedestrian crossing right after the standing car the selected behavior might lean towards stop and wait, but if there are parking spots on the roadside next to it then overtaking maneuver might be chosen instead. As we can see HD maps are not just about locational accuracy and mapping the road network. HD maps can be much more - they can be treated as a sensor providing a bigger range than other sensors in the car [6]. All this information can be used in AD and it adds safety and more comfort for passengers.

Imagine AD without a map: sensors providing real-time information out of what 3D model of the surrounding needs to be constructed and driving is performed. It is a complicated process and needs a lot of processing: volume of data, the accuracy of object detection, disturbances like weather, illumination (day, night), and different traffic situations. HD maps can be good prior knowledge about the environment (knowing the drivable area, differentiating between obstacle or sign) if localization works correctly. Localization itself is also quite complicated and GNSS only is also not a very reliable solution, therefore maps can also help in solving localization [5]. It is still challenging for the perception to get information about lane merges and connections in the intersection, traffic light and sign associations, yielding and stopping points and HD map can help also in these cases [3].

Data quality of HD maps is not only having lane level detailedness and few centimeter levels of locational accuracy. There are a lot more aspects needed to assess the quality of HD maps and it can be done by looking at the ISO standards for geographic information[14] with some explaining examples:

[13] https://medium.com/lyftself-driving/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6

[14] https://www.iso.org/standard/32575.html

- completeness (are all the features of every feature class present);
- logical consistency (is the road network topology correct, one-way streets digitized in the correct direction);
- positional accuracy (accuracy of absolute location);
- temporal accuracy (is the dataset up to date or does it reflect the situation 1 year ago, is it different among feature classes);
- thematic accuracy (wrong attribute values: path for pedestrians is attributed as a road for cars).

2.3.1. Localization

We can distinguish between relative and absolute localization. Relative localization means that the vehicle is localized using information about its surroundings. To avoid collisions the relative localization accuracy is more important than absolute localization accuracy [23]. This information is acquired using measurements by sensors, usually distances from other objects. If these objects are previously mapped features then by matching these measurements with map information we could localize the ego vehicle also on the map. Location on the map (the coordinates) can be viewed also as absolute location. The most common way of acquiring absolute location is to use the Global Navigation Satellite System (GNSS).

Localization can be done using active or passive sensors. Active sensors emit some energy and measure the reflected energy (lidars, radars, ultrasonic sensors). Lidar is one of the most used sensors in AD and also used for localization. The downside of it is the high implementation cost, sensitivity to weather conditions, high power consumption, and the large capacity of the output point cloud data that sets high demands for processing [24]. On the other hand, it is one of the main and most accurate sensors for perception tasks. Passive sensors detect reflected electromagnetic radiation from other sources, like the sun and the examples are cameras. Cameras are easy to mount and considerably less expensive than lidars and they can represent the color and texture of all the objects in the image [24].

Localization of vehicle for AD demands highly accurate, preferably six degree of freedom (xyz coordinates and pitch, roll, yaw angles) localization, which is crucial for defining the Field of View (FOV) of the sensors [4]. According to Yurtsever et al. [3], there are 3 main ways for localization: GNSS-IMU fusion, Simultaneous Localization and Mapping (SLAM), and a priori map-based localization.

Classical **GNSS and IMU-based approaches** can provide an accuracy of 2-3 meters and this is not enough for AD. Accuracy can be increased using base station solutions like differential GNSS or Real-time kinematic (RTK) corrections, this will increase the accuracy up to 2-5 cm. The problems will arise when the signal loss from the satellites, signal multipath interference, poor sky visibility when infrastructure is blocking the view, and atmospheric conditions that affect the GNSS signal quality appear [24]. RTK GNSS increases the accuracy considerably but the availability of the RTK service will be the issue [25]. RTK GNSS is more vulnerable to signal degradation and loss of integrity in urban areas and recovering from this can be difficult. GNSS signal degradation can happen also because of

dense vegetation [26]. One of the solutions is to use dead reckoning (DR) for the time when GNSS RTK is not available. DR is usually implemented using IMU and odometry data to derive the relative direction and distance driven from the last known location. This on the other hand might quickly get out of control when positional errors start to accumulate [26]. It is generally agreed that RTK GNSS can produce needed positional accuracies, but it is not reliable on its own. The general solution is to add additional sensors (modalities) and combine the results, like wheel odometry data and lidar measurements [27]. Measurements from different sensors can be merged together using the Kalman filter or other methods. The downside of the RTK GNSS-IMU DR localization solution is the considerably high cost of the sensors needed for implementation [26].

SLAM type of localization does not need to have a priori knowledge about the environment. It also means that premade map is not necessary, so these methods could potentially work everywhere. On the other hand, high computation needs make them less efficient than localizing using existing maps [28]. There is another problem using SLAM - the localization tends to drift over time and when returning to the same place the generated map is not overlapping. Different “loop closure” techniques are used to address the issue, prerequisite is of course the ability to correctly recognize the revisited places [23]. As the origin of SLAM algorithms is robotics and more from indoor mapping they are facing the following challenges in the context of AD [25]:

- accuracy,
- scalability (high computing needs),
- availability (need for immediate localization even if there is no first passage, it means that some premade map is needed),
- recovery (ability to localize in large scale map in case of failure),
- updatability (identify change between the map and current observation and be able to correct the map),
- dynamicity (handling the dynamic objects).

So we still need an initial map because we need the localization straight away and also without an existing map, the recovery of SLAM would be a very hard problem to solve. DR and odometry or some other modality should be added otherwise it is not possible to recover (continue using the same map coordinates). In robotics, the problem is ḱnown as kidnapped robot problem and according to that robot might believe it is somewhere else than it really is and the recovery from that is very important in the context of AD [29].

SLAM localization would give us coordinates relative to map frame coordinates, but if there is no prior map then these coordinates are arbitrary. Therefore SLAM is used more in mapping and when considering the localization it is part of the localization pipeline to include more modalities and provide options to obtain localization information from different sources. The most commonly used sensor for SLAM localization is lidar.

A priori map-based localization assumes an existing map and localization is done using online readings from sensors and matching them with the map. There are two approaches:

- landmark search and

- point cloud matching.

Landmarks can be any kind of feature (lane markings, poles, signs, curbs, buildings), but to be able to localize enough landmarks is needed. In the case of landmarks, they have to be identifiable with online sensory information. Map-based localization is vulnerable to changes in the environment, sensor readings will deviate from the information on the map. Additional information such as GNSS, IMU, and odometry data can be added and fused with the system [30] to help in locations where there are fewer landmarks available. Very common is to use GNSS data as an initial pose estimate.

In the case of point cloud localization, an online scanned point cloud is matched with an existing point cloud map. Matching is performed using 3D registration-based (like Iterative Closest Point [31] or Normal Distribution Transform (NDT) [32] algorithms), 3D feature-based or 3D deep learning-based methods. 3D feature-based methods are current state-of-the-art due to their proven stability in real-life situations [33]. In the case of point cloud matching lidar is needed making the system quite expensive to implement.

A very popular topic is to find cheaper alternatives to localization, like using cheap GNSS, combined with camera images and doing the landmark matching [34]. It is often called 2D to 3D localization - match 2D sensor data (camera images) to 3D features (real world) and obtain a localization [3]. There is an approach using stereo camera setup and visual odometry matching it with lidar intensity image [35]. Another setup uses inverse perspective mapping of camera images to produce the top view. Whitelines and road markings are extracted from there and using GNSS location as initial estimate features are matched with existing HD map and lane level accuracy is achieved [24]. A similar approach with a monocular camera is suggested by [26].

The main purpose of using cameras and maps is to acquire lane-level high precision vehicle absolute localization with minimal cost. Another advantage compared to point cloud localization is that maps are not so heavy. Instead of millions of points, we could use much fewer features (less points) and solve the memory problem. In one experiment these “sparse maps” were used with lanes, traffic signs, and vehicle dynamics to localize robustly with respect to a sparse semantic map. They validate the effectiveness of their method on a new highway dataset consisting of 312km of roads and show that the proposed approach is able to achieve 0.05m lateral accuracy and 1.12m longitudinal accuracy on average while taking up only 0.3% of the storage required by previous LiDAR intensity-based approaches [22].

2.3.2. Coordinate frames and (reference) systems

Maps contain spatial information, they essentially store locations of different features. HD maps are no different, all the features have locations, and the location is represented by a set of coordinates. There are global coordinate systems as used by GNSS, that have coordinates defined with latitude, longitude and elevation (above ellipsoid). These angular coordinates are very inconvenient to perform distance calculations and therefore cartesian coordinates are used for such purpose. The problem with cartesian coordinates is that the Earth is round and it is not possible to create a 2D map of the Earth without distortions. Distortions can be kept

minimal by choosing a suitable map projection for the necessary area. That is also the reason why there are many different map projections and different coordinate reference systems (CRS) used around the world.

Coordinates are very important also in robotics, because we want to know what is around the robot, how far it is etc. and this can be achieved by having everything around the robot transformed into one coordinate frame. In robotics, all the frames use the right-hand rule to define the axis for the coordinate systems. For base_link frame, the coordinates would be x - forward, y - left, and z up. Quite often the map coordinates will be to locate the base_link in the map frame and this is the task for localization.

The map frame is a world fixed frame. The location of the ego-vehicle in the map frame is discrete, meaning that location jumps every time the new measurements are processed and a new location is calculated. This frame is suitable for positioning an ego-vehicle for long-term global reference. The usual convention in the map frame is x - east, y - north and z - up. Odom frame is also world fixed, but continuous frame based on wheel odometry or IMU data. It is suitable for short-term referencing because of its continuous nature. Earth frame is used in robotics in case multiple robots operate in different map frames. Then individual map frames will be linked to earth frame (static transforms) and so the robot locations could be transformed into each other's map frames. In the case of one robot, there is no need for an earth frame. The coordinate frame called earth is the origin of ECEF[15] - earth-centered, earth-fixed coordinate system.

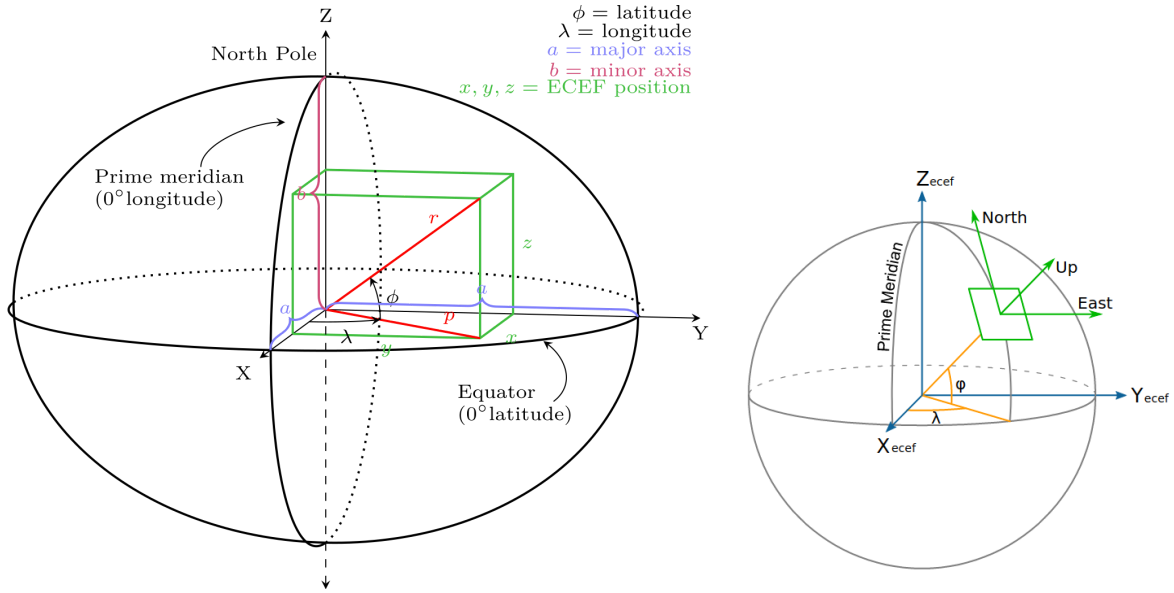

Figure 2. Example of linking global and local coordinates: left[13] - the relationship between ECEF (xyz) cartesian coordinates and Geodetic coordinates (latitude, longitude), right[16] - defining a local coordinate system (East North Up - ENU) relative to geodetic coordinates.

[15] https://en.wikipedia.org/wiki/ECEF
[16] https://www.ros.org/reps/rep-0105.html

GNSS gives us the location in geodetic coordinates that is not the same thing as geocentric coordinates used in ECEF. GNSS produces measurements in the WGS84[17] system (World Geodetic System, format: BLH, B - Latitude, L - Longitude, H - height above ellipsoid). We would need the information about the ellipsoid to transform these coordinates to ECEF coordinates (XYZ - distances from Earth center). The relationships between ECEF and the geodetic system can be seen in Figure 2. A simple way to define a local coordinate system would be to take the global coordinate (either in ECEF xyz system or use geodetic coordinates) and use it as an origin for a local map frame called ENU - East North Up. Local map frame coordinates would be defined from that origin point using a right-hand rule that would result in East pointing to x, North to y, and z to upward direction. The problem with that solution is degrading accuracy of the coordinates when moving away from the origin point. It might not be a problem when operating in small areas, but it is not the case with AD. That is why usually some existing map coordinate systems are used instead of simple local ENU frames. To fix a map coordinate reference system, the following needs to be defined:

- geodetic reference system;
- an ellipsoid (has parameters about its size and orientation relative to the Earth center);
- projection surface (type - plane, conic, cylinder - and how it is positioned relative to the ellipsoid);
- origin of the coordinate system on the projection surface along with necessary parameters.

Estonian national coordinate system[18] is using GRS80 ellipsoid and Lambert Conformal Conic Projection. The resulting projection is called LAMBERT-EST and is defined by the following parameters:

- Geodetic Reference System: ETRS89. It is considered identical to the WGS84 system.
- Reference ellipsoid: GRS80
- Standard parallels:
    - BS=58°00'
    - BN=59°20'
- Central meridian:
    - L0=24°00'00"

Based on LAMBERT-EST projection Plane Rectangular Coordinate System L-EST97 is defined:

- x-axes is collinear with LAMBERT-EST central meridian.
- Geodetic coordinates of the origin point:
    - B0=57°31'03.19415"
    - L0=24°00'
- Rectangular coordinates of the origin point:
    - x0= +6 375 000m

[17] https://epsg.io/4326
[18] https://geoportaal.maaamet.ee/eng/Spatial-Data/Map-Sheet-Indexes-and-Coordinate-Systems/L-EST-Coordinate-System-p360.html

    - y0=+500 000m
- epsg code[19]: 3301

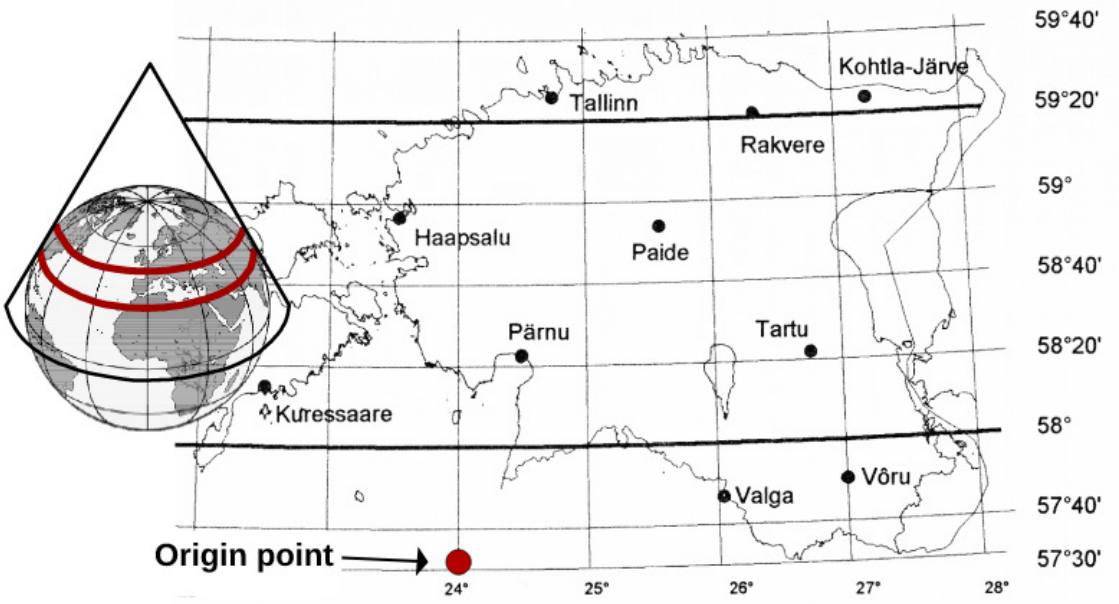

Figure 3. The conic projection used to transform Estonia into a flat surface (2 standard parallels are darker) and the origin point for L-EST97 CRS indicated with a red dot [36].

L-EST97 might be a very good CRS to use in Estonia, because of already existing map data that uses these coordinates and minimizing distortions of map coordinates that arise when projecting round Earth to a 2D map. Considering AD and the wish to drive also outside of Estonia we might instead use some global cartesian coordinate system. As an example, Universal Transverse Mercator (UTM) and Military Grid Reference System (MGRS) are popular choices and also used in Autoware[20]. Sometimes different map data can be in different coordinate systems: point cloud map is using MGRS since it gives the flexibility of choosing scale and coordinate detailedness and UTM system is used for vector map data [37]. What coordinate system to use depends on the application, the main thing is to keep track of them. Knowing the CRS of the data gives an option to transform the data to other coordinate systems.

2.4. Mapping methods

The process of mapping does not consist only of data collection, the data needs to be processed and the final (HD vector) map needs to be pieced together. Sometimes the necessary data might be already collected (open datasets, crowdsourcing) and the collection

[19] https://epsg.io/3301
[20] https://discourse.ros.org/t/autoware-coordinate-systems/10662/14

step might not be necessary. In general, we can distinguish three steps in mapping: data acquisition (driving the route and using the sensors to collect the data, gather existing data or combine both), data processing (get from raw data to usable data) and finally map creation (manual refinement and confirmation of data) [38] [7].

Mapping for HD vector maps is usually done with MMS (Mobile Mapping System) cars [38] having a lot of sensors to record the data. Global companies like Here[21], TomTom[22], and Google have entire fleets of these cars. Main sensors used are: GNSS, lidar(s), camera(s), odometry, IMU, radar(s). It requires a lot of processing afterward and automation is the key here. According to [39] HD map creation still requires a lot of human input which is time-consuming and prone to errors. In the Baidu Apollo course lectures, they claim that 90% of the mapping is automated[23] and their approach includes data sourcing (MMS car fleet), data processing, object detection, manual verification, different map products (HD map, localization map, point cloud map).

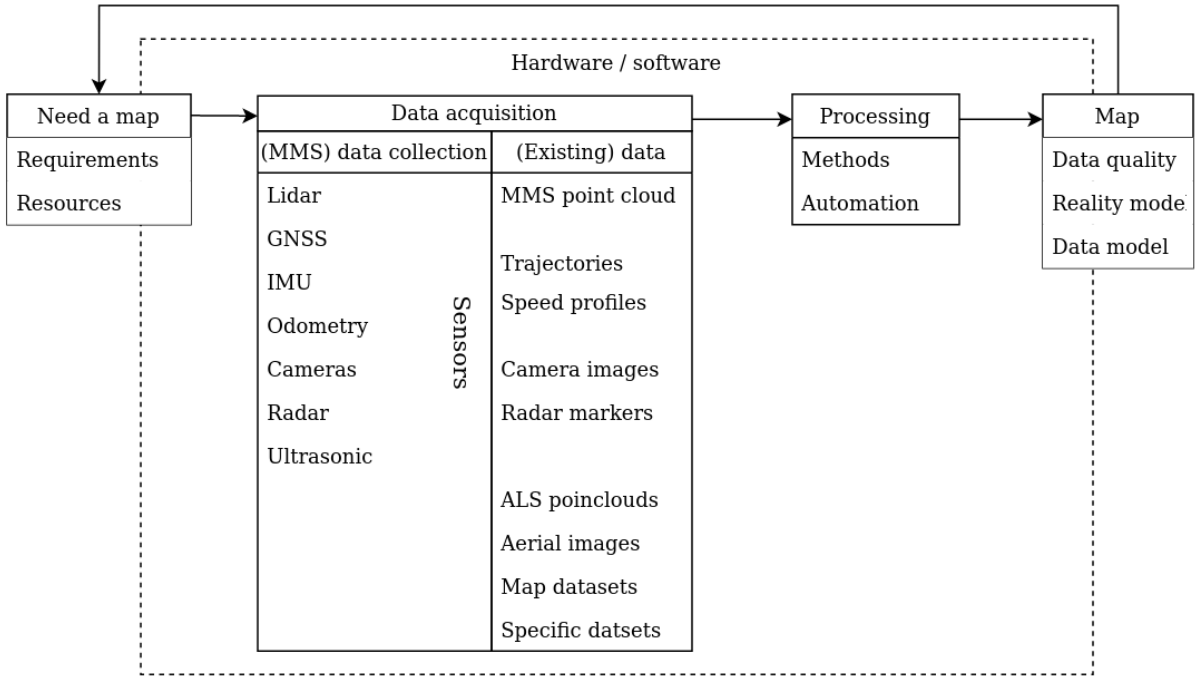

Figure 4. Schematic of the mapping process.

Special MMS cars are expensive to set up. To escape these costs company called Lyft[24] decided to use its self-driving car fleet. It can be argued how much less equipped is their autonomous car compared to MMS car, but the benefits they bring out are quite relevant:

- better utilization of autonomy fleet with no extra costs and development for MMS;

[21] http://here.heresf.acsitefactory.com/platform/mapping/map-data
[22] https://www.tomtom.com/blog/autonomous-driving/how-we-make-our-hd-maps/
[23] https://apollo.auto/devcenter/coursevideo.html?target=2_20
[24] https://medium.com/lyftself-driving/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6

- same data logs, so there is no difference if the car is driving or mapping (data is there for both);
- perception can help label the data and aid in change detection.

Another solution is to use cars with less precise sensors (normal cars) since most of the ADAS-capable cars today are equipped with quite many sensors (cameras, radars, ultrasonics, GNSS). As the collected data from these cars are not so accurate a lot of data would be needed and therefore crowdsourcing is suggested [7]. Such an approach is also used by Mobileye and their solution is called Road Experience Management[25]. Typically involves data collection and uploading to some platform that takes all the observations and synthesizes them into usable data.

Crowdsourcing seems to be one of the key features making collecting and keeping the map data up to date cheaper and quicker. Examples of crowdsourcing can be found also from Comma.ai[26] having a fleet of EON and GrayPanda users to collect the road data and build an accurate representation of the highways throughout the US - the data is also for sale[27]. Tesla is collecting the map data[28] although the CEO of Tesla Elon Musk has said that high precision lane maps are not a good idea[29]. Mapillary[30] collects street-level imagery to extract map data. Even global mapping companies offer options for their users to make edits in the maps like Here Map Creator[31]. Baidu Apollo has also released its data sourcing tools to the public so that anyone can participate in map creation. And classical example of crowdsourced data is of course OpenStreetMap[32].

Mapping might have very different strategies. Let's take a big mapping company with a fleet of MMS cars, their mission might be: drive through all the roads and collect all the data possible. It is their mission as a service provider and gaining an advantage among competitors. Totally different mapping strategy might be for a research institution that has an AD platform with a certain software framework that supports a specific HD map format with only a limited amount of features and hardware framework that is capable of utilizing only part of the map functionality, but they have modified the code to test some novel localization method based on radar markers they have collected in some novel way. The previous theoretical example is here to illustrate the complexity and the very many aspects of mapping that is dependent on the requirements and available resources:

- Requirements:
    - Positional accuracy (2D, 3D, at the level of few meters or few centimeters);
    - What features / layers are needed (their level of attribution);
    - What format is needed.
- Resources:

[25] https://www.mobileye.com/our-technology/rem/
[26] https://medium.com/@comma_ai/hd-maps-for-the-masses-9a0d582dd274
[27] https://comma.ai/services/data-sales
[28] https://www.geoawesomeness.com/tesla-entering-map-making-business-what-does-it-mean-for-the-industry/
[29] https://www.forbes.com/sites/bradtempleton/2019/05/20/elon-musk-declares-precision-maps-a-really-bad-idea-heres-why-others-disagree/
[30] https://www.mapillary.com/update-maps
[31] https://mapcreator.here.com/
[32] https://www.openstreetmap.org

- Data (is the data available or needs to be collected, the area needed to cover);
- Money (is there money to buy the data, do the mapping yourself);
- Time (For what time is needed, is it possible), long and short term objectives;
- Skills (setting up the workflows and pipelines for automation).

Data collected with MMS cars can be divided into 2 categories: HD Map data (lidar point clouds and camera images) that will be transformed into an HD map and auxiliary data (different logs: GNSS, IMU, wheel odometry) that helps to create the map, but won't be part of it [40]. Although some of the auxiliary data could be also used in the HD map if collected appropriately (like reference speeds for the lanes).

Lidar point clouds provide direct 3D location information and geometry with high precision. The location from camera images can be acquired from triangulation and after 3D reconstruction methods like structure from motion[33]. Both sources are used to extract semantic information and attribute data for features in the HD map. Mapping is mostly done after the data collection process because the accuracy relies on very accurate pose estimates of the data collection vehicle [40], but this is not always possible in real-time that is why the post-processing of auxiliary data is used to correct the possible errors by fusing together all the logs [41].

It would take a lot of time to create an HD map from the massive amounts of collected high-dimensional data, using manual labor (as can be seen also reported later in this thesis). There is a quite big overlap in using ML for solving problems for autonomous driving and mapping tasks, for example, object detection and labeling (perception) is very similar to a mapping task (adding semantic knowledge to features) [40]. Therefore ML has been used a lot to solve also different mapping tasks. For example, lidar intensity images combined with camera images were used to derive lane topology and lane boundaries of complex highways [42] with the help of ML. Point clouds with segmented camera images were used to remove clutter objects from point clouds in an automated way [43]. In another example semantic information is extracted from images and added to an HD map, extracting locations from aligned point cloud [44]. Lidar and camera are used to map road boundaries and lanes in [39]. Only camera images are used along with auxiliary data to detect lane locations and lane markings (reflect traffic rules) and used for extracting semantic information [45]. Lidar without camera images is used to extract road boundary in [46].

There are also interesting examples of mapping where the data is not from the MMS car. One example is Mapillary - a crowdsourcing platform where users can upload street-level imagery and traffic-related map elements are extracted. They have published an open dataset about traffic signs for detection and classification on a global scale [47]. It has 400 manually annotated traffic sign classes with 105K street-level images. Another example is based on open-source map data found in OpenStreetMap (OSM). The data from OSM was used to create HD vector maps for Baidu Apollo [48]. They estimated the accuracy for maps as low and one of the reasons for that is too generalized features (features are represented by too few points).

[33] https://en.wikipedia.org/wiki/Structure_from_motion

Another interesting dataset is aerial images. Quite often this data is collected by a national mapping agency, it might be processed into orthophotos, has good enough resolution and accuracy, and is open data. This is the case in Estonia where orthophotos with 10 cm ground resolution are available for urban areas. Aerial images and aerial lidar scanning data are for road segmentation using deep convolutional neural networks [49]. In their later approach, they have advanced the method to produce also the road network graph [50]. An example of combining OSM data with aerial images to map lanes by using a novel image segmentation algorithm [51] is a good example of combining different datasets to make it easier for ML algorithm to achieve the elements needed.

Mapping can not handle everything and some of the tasks are still hard to get by mapping according to Mobileye[34]:

- priority - which lane has priority over which lane;
- traffic lights crosswalk and lane associations;
- stopping and yielding points;
- drivable path;
- common speed.

2.5. Existing High-Definition map formats

Maps for humans are mainly interpreted visually and there is a limit to how much information can be transferred like that. An AD system can not understand the world and do the planning in the same way as humans, the map needs to be efficient and quickly available at the code level for different calculations and decision making. In the following subchapters the following HD map formats will be introduced:

- Autoware vector maps
- OpenDrive
- Navigation Data Standard
- Lanelet2

2.5.1. Autoware Vector Maps

Autoware.ai is the world's first "All-in-One" open-source software for autonomous driving technology[35]. One of the planning nodes there is OpenPlanner and it is used also by Autonomous Driving Lab (ADL) in the University of Tartu. OpenPlanner supports the simplified version of Aisan vector map format that is usually called as Autoware vector map format and for clarity reasons Autoware vector map is also used in this text. There is not much documentation about it and the questions about it are quite common in forums[36]. Answers to the previous question confirm what is also written on the page of Vector Map Builder[37] that this format is originating from Aisan Technology proprietary ADAS map

[34] https://www.youtube.com/watch?v=HPWGFzqd7pI
[35] https://www.autoware.ai/
[36] https://discourse.ros.org/t/documentation-on-vector-map-format/8282
[37] https://tools.tier4.jp/feature/vector_map_builder/

format in a bit simplified form. Therefore also the documentation is not available. The best source to understanding the format is actually the user guide[38] for Vector Map Builder that requires registering a user account and logging in.

In general, the format consists of a lot of thematically separated csv files that are related to each other using ids. According to [52] there can be 32 different csv files or categories of how the information of the vector map is distributed: point, vector, line, area, pole, box, dtlane, node, lane, wayarea, road edge, gutter, curb, white line, stop line, zebrazone, crosswalk, road mark, road pole, road sign, signal, street light, utility pole, guardrail, sidewalk, drive-on-portion, crossroad, side strip, curve mirror, wall, fence, and rail crossing. Among those files, the minimum set necessary to be able to simplest drives with a car are point, node, lane, and dtlane csv files.

2.5.2. OpenDrive

The standard was originally published in 2005 by VIRES Simulationstechnologie GmbH, but in 2018 the standards were adopted by Association for Standardization of Automation and Measuring Systems (ASAM)[39]. OpenDrive's main use was related to simulation applications. OpenDrive[40] is an open file format organized in a hierarchical structure and serialized in an XML format. The format is technology and vendor independent.

The format describes exact road geometry, includes surface properties, markings, signposting and logical properties such as lane types and directions. OpenDrive has one reference line and lanes are built around that.

Chinese company Baidu has adopted and modified the OpenDrive format for their open-source AD framework called Apollo. Their contributions are making it more suitable also for real-life AD and the format is referred to as Apollo OpenDrive.

2.5.3. Navigation Data Standard

Navigation Data Standard[41] (NDS) is the de facto standard between automotive OEMs, map data providers, and manufacturers of navigation devices. Full NDS is not an open format, full specification is available to NDS association members. For NDS association members there are also some tools provided. NDS is used currently in over 30 automotive brands globally [16].

NDS uses SQLite database file format and is very heavy (a lot of feature classes and attributes) that is brought out in the formats review by Autoware.auto working group discussion[42]. Figure 5. provides some insight into the data stored in NDS format. In the case

[38] https://tools.tier4.jp/vector_map_builder/user_guide/
[39] https://www.asam.net/news-media/press-releases/detail/?tx_news_pi1%5Bnews%5D=48&cHash=726c911525f72ede5ade9e755c3c8369
[40] https://www.asam.net/standards/detail/opendrive/
[41] https://nds-association.org/
[42] https://discourse.ros.org/t/autoware-maps-and-map-formats-working-group-meeting-minutes-24-july-2019/10059

of NDS format, the navigation data and navigation software are separated to provide flexibility to end-users. There are also initiatives like NDS.Live[43] format that would make it possible to stream map data live from the internet.

OpenLaneModel[44] is a reduced set of data from NDS specific for AD and the specs are there available after registering on their website.

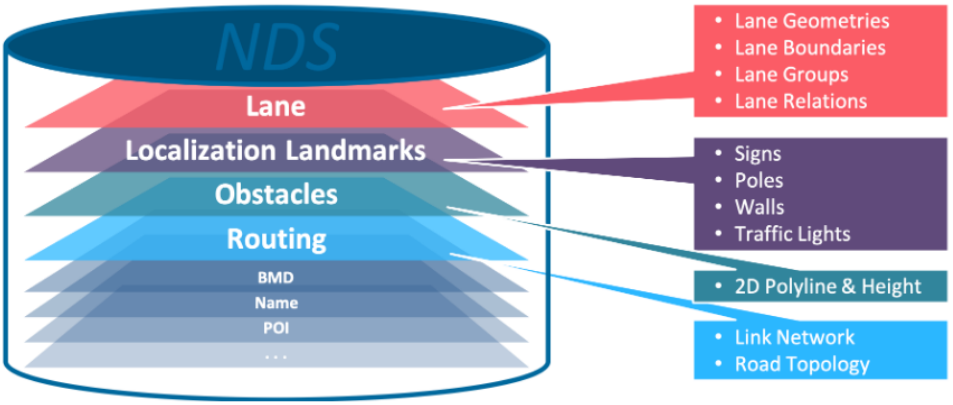

Figure 5. Example layers from the NDS database and some selected features they contain [16].

2.5.4. Lanelet2

Lanelet2[45] is a C++ library to handle map data for AD. Flexibility and extensibility are the main principles to handle the upcoming challenges of future maps for AD. The format is XML based OSM data format built on top of lanelets. There are free editors available, like JOSM[46]. Lanelet2 is much more lightweight than NDS. As stated in [13], Lanelet2 is not just for some isolated applications like localization or motion planning, but for all potential tasks that AD could need from HD maps.

There is a map part that uses six different primitive types to describe all the map data (Points, linestrings, polygons, lanelets, areas, and regulatory elements). Separate from map data (format) there is also a software framework and API that enable the use of different traffic rules for different user types. It can be different for cars, cyclists, and emergency vehicles resulting in totally different routing schemes (for example allowing emergency vehicles to drive in the opposite lane).

There is a format, software framework, and API that allows geometry-based search (like finding closest), generate different routing graphs specific to pedestrians, cars, or cyclists

Architecture of Lanelet2:

- physical layer - real observable elements /mapped features

[43] https://nds-association.org/nds-live-the-new-generation-of-map-data-format/
[44] https://olm.nds-association.org/
[45] https://github.com/fzi-forschungszentrum-informatik/Lanelet2
[46] https://josm.openstreetmap.de/

- relational layer - map features are connected to lanes, areas and traffic rules
- topological layer - neighbourhood relationships and connections forming a network of potentially passable regions depending on the road user and situation.

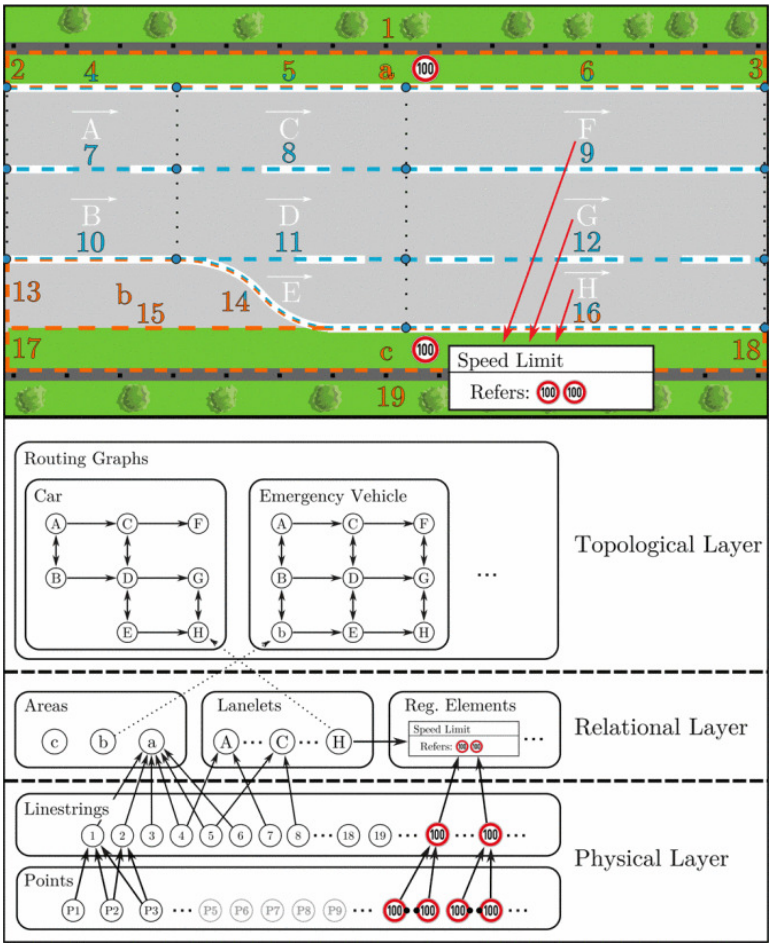

Figure 6. Map example for a highway road (enclosed by guardrails) and the resulting map structure. Lanelets have capital letters, areas lowercase letters, and linestrings have numbers, image from [13].

2.5.5. Comparison of HD vector map formats

Table 2. compares some aspects of formats and availability of documentation, making it easy or difficult to start using the format. If there is no software framework and API available for the format all needs to be implemented making it really hard to adopt the format. Table 3 is adopted from the Autoware maps and map formats working group, where the selection of the main map format for Autoware.auto was discussed.

Table 2. Comparison of HD vector map formats.

	Autoware Vector Map	(Apollo) OpenDrive	Lanelet2	Navigation Data standard
File format	csv text files	XML file	XML file	SQLite database
Documenta-tion	Lacking	Lacking (need to register and ask)	Available	Full documentation assumes membership
Open-source	No	Yes	Yes	No
API	Implemented inside ROS	Lacking	C++, Python bindings	C++, Java

Table 3 evaluates different formats a bit more from an industry perspective. Specifically would bring out the category "Adoption of the Format" from the table. When adoption in the open-source community would be considered then NDS would be in authors' opinion somewhere around 1 and the marks for Autoware vector map and Lanelet2 would be raised to 2 and 3 respectively.

Table 3. Comparison of HD vector map formats by Autoware Maps and Formats working group[47].

Criteria	Autoware Vector Map	OpenDrive	Lanelet (OSM XML)	NDS
Ease of Creation	3 (There are multiple available tools)	1 (not many tools to write map)	4 (They are node based, which is probably easier than defining continuous curves)	2 (Seems difficult to create all information)
Tools	3 (Read/Write for ROS software is already implemented as Autoware package)	4 (Many different tools, but majorly for simulation, not for Autonomous Driving Software)	3 (There are many OSM tools, but not much for Lanelet2)	4 (Many tools are available after purchasing the license)
Adoption Of Format	1 (Not publicly used by other companies)	4 (Many automotive and tier-1 companies, now being standardized by ASAM)	2 (Some map vendors providing maps in Lanelet2 format)	5 (Created by major automotive and tier-1 companies, standardized under NDS Association)

[47] https://discourse.ros.org/t/autoware-maps-and-map-formats-working-group-meeting-minutes-24-july-2019/10059

Relation to Production Systems	1 (No plans for connecting to production systems)	3 (Not directly used in production environment but can be converted to NDS which is favored by many OEMs)	1 (Lanelet is relatively new)	5 (same reason as above)
Expressive-ness	2 (Small amount of flexibility)	4 (traffic sign id is not clear in countries outside Germany)	3 (Does have room to extend format, but has less information compared to OpenDrive at the moment)	5 (the detailed specification is closed, but is expected to be very high considering adoption and relation to production system)
Interchangea-bility	2 (MR in gitlab for conversion from Opendrive and Lanelet)	3 (bidirectional convertor to NDS)	3 (Has MR in gitlab, also lanelet1 and OpenDrive converter available)	2 (bidirectional converter to OpenDRIVE)
Accessibility	2	4	5	1 (requires a license)
Total	14	23	21	24

3. Methodology

In this chapter the methodology of achieving the main task: create an HD map that would enable AD capabilities for the University of Tartu (UT) self-driving platform. HD map format is dependent on the software used and its features have to enable the tasks specified in the ODD of the platform. Therefore the overview of the platform is given at the beginning of this chapter. It is followed by the review of the possible tools for map creation and also available data sources for the HD map. Having gathered all the background information a workflow developed by the author of this thesis is introduced at the end parts of this chapter. Custom HD map with its reality and data model and converter script that will transform the created HD map into the correct format needed for the platform are introduced at the end of this chapter.

3.1. Autonomous driving platform

Autonomous driving (AD) platform in UT Autonomous Driving Lab (ADL) is bought from AutonomousStuff[48]. It consists of a Lexus RX450h car that has an already built-in drive-by-wire system. Additionally, the Spectra computer and following sensors to support highly automated driving tasks were installed by AutonomouStuff:

- Allied Vision Mako cameras
- NovAtel PwrPak7 GNSS and IMU
- Delphi ESR 2.5 radar
- Velodyne VLP-32C lidar.
- Pacmod - hardware and software solution that enables controlling the car steering and speed
- Velodyne VLP-16 lidar - added later

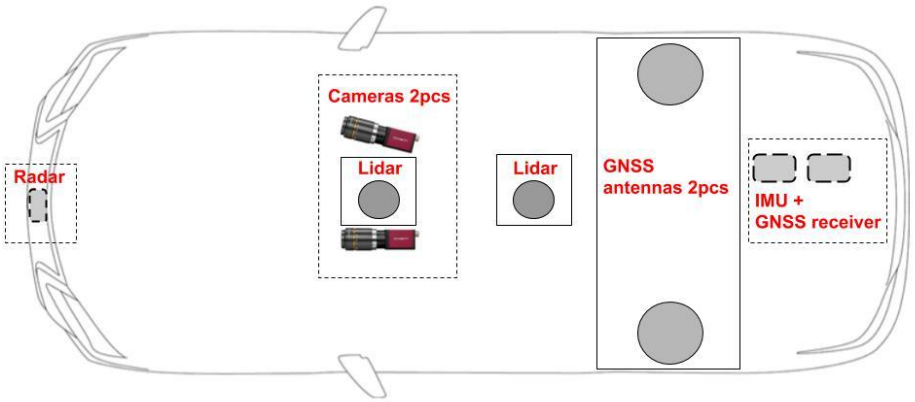

Figure 7. Schematic view of sensor placement.

[48] https://autonomoustuff.com/

Main components of the software platform:

- Ubuntu Linux operating system
- Robot Operating System (ROS)[49] middleware layer
- Autoware.ai[50] - an open-source software solution for AD, first introduced in [53]

Autoware consists of a lot of nodes that will be launched. Each node will have its specific task, for example, there are nodes for doing the localization, removing ground points from the raw lidar point cloud, clustering the remaining points for object detection, car trajectory selection, global planning etc. Each of these nodes subscribes to the necessary topic(s) and publishes their results to another topic(s). Based on the information in these topics control commands for the ego vehicle are derived by the controllers that are sent through the Pacmod to the car's CAN bus where the actuators will execute these commands.

Localization can be carried out using GNSS-IMU with RTK corrections. As an academic institution, the RTK correction service[51] from Estonian Land Board is free of charge for ADL. Localization using lidars has been also tested using the point cloud maps generated in ADL and by EyeVi Technologies[52]. NDT-matching algorithm was used to test point cloud localization. Currently, the GNSS-INS localization with RTK corrections has been more reliable and this is the main localization method used.

The GNSS system gives the absolute location in the WGS84 system (latitude, longitude, and elevation above the ellipsoid). These coordinates are good to provide a location on Earth, but not very suitable for example distance calculations due to their angular measurements relative to the plane of the prime meridian and equatorial plane. To make calculations simpler cartesian coordinates are needed and that is why Estonian coordinate system L-EST97 is used. That is why our localization node is configured to output coordinates in the L-EST97 system and it means that our maps also need to use that system.

For planning (global route planning, local planning), behaviour selection and for longitudinal control, OpenPlanner is used [54]. OpenPlanner consists of several running nodes that are responsible for several specific tasks, like regulating speed when following other cars, deciding on actions based on detected traffic light status, brake when a pedestrian crosses the road etc. To perform these tasks OpenPlanner needs the HD map that will be used as a common reference frame to all the information from perception nodes and planning tasks. Localization will produce the location of the car (base link) in the map coordinates and all the objects detected by lidar or radar (other vehicles, pedestrians) will be also transformed to a map coordinate system, so all the planning and decision making happens using the map coordinates and including information from the HD map. OpenPlanner has to fulfill the following tasks:

- subscribes to current location information and locates ego vehicle on the map;
- global route planning - path with the minimal cost from start to endpoint will be found;

[49] https://www.ros.org/
[50] https://www.autoware.ai/
[51] https://geoportaal.maaamet.ee/eng/Spatial-Data/Geodetic-Data/Geodetic-Networks/ESTPOS-p671.html
[52] https://eyevi.tech/

- local trajectory planner - generates smooth trajectories on a local scale that will be inputted to trajectory following algorithms;
- considers HD map features like stop lines, stop signs, traffic lights and acts upon them according to the set rules;
- subscribes to detected obstacle topic and looks if the obstacles are blocking the lane and reacts accordingly (different behaviour states: forward, follow, stopping);
- use additional HD map features if behavior enabled (for example curbs can be treated as obstacles for the ego vehicle).

OpenPlanner can use 2 formats for HD maps: Autoware vector map and kml file designed specifically for OpenPlanner. Both of the formats do not have much information nor documentation available.

The operational design domain (ODD) is usually a description of the conditions and limits the environments where the AD system is meant to properly operate. In the case of the ADL platform the initial requirements can be summarized with the following points:

- AD takes place only on the demo routes with an HD map being present.
- There is always a safety driver ready to take over the control. No full autonomy is expected from the ADL self-driving platform.
- Traffic situations that are meant to be handled are demo route specific:
    - follow the traffic flow and maintain reasonable distance with vehicles in front;
    - recognize traffic lights and act accordingly;
    - perform right turns in intersections with traffic lights (where there is no need to give way);
    - handle regulated and unregulated pedestrian crossings.

Requirements for the HD map:

- lane level detailedness for mapped features;
- locational accuracy around 5-10 cm: the car must be able to drive autonomously and stay in the lane in straights and curves (lateral control)
- locations for traffic lights and stop lines;
- for longitudinal speed control, velocity must be mapped;
- for routing the topology must be correct;
- features that must be mapped: curbs, lanes, white lines, traffic lights, pedestrian crossings, intersection, road signs, stop lines;
- smooth curvature of the trajectories so the drive would be smooth (lateral control);
- usage of indicator lights;
- L-EST97 coordinates are used.

3.2. Assessing available datasets

To be able to prepare maps faster by using already existing data possible datasets were reviewed. When relying on different datasets the information about CRS used is essential to be able to align the data in space. Estonian national CRS is L-EST97 and all the state offered datasets most likely use it. GNSS usually outputs coordinates in the WGS84 system and the same is used by OSM data. Some other datasets might be using some UTM-related projection and coordinates. It is important to keep track of the CRS systems used to be able to align data from different sources.

3.2.1. Estonian Topographic Database

Estonian Topographic Database[53] (ETD) is the basis for the Estonian National Land Information system. Its main objectives are:

1. *to manage and organise the production of topographic spatial data and topographic products and maps covering the whole territory of Estonia;*
2. *to provide the society with up-to-date and high-quality data, data services and maps produced on the basis of these data;*
3. *to provide state and local government databases with topographic spatial data for objects managed in their respective databases;*
4. *to implement the directive of the European Parliament and of the Council establishing an infrastructure for spatial information in the Community.*

Mapped real-world features are defined in the reality model and the data model specifies what attributes are added and how the features are stored in the database. Both are available in the form of feature catalog[54] that is unfortunately only in the Estonian language.

The scale of data is meant to satisfy scales 1:10000 and 1:5000 and all mapped data is stored as vector points, lines, or polygons with coordinates in the L-EST97 system. Every element has its unique ID and can be associated with respective registers like Road register, Address Register etc. All the data in ETD is open data starting from 01.07.2018[55].

More interesting in our context is the road data. According to the reality model roads are represented in the database as areas with centerlines. Inside the city road area also includes the pedestrian walkways at the roadside. The centerline of the road has parameters for width (without pedestrian walkways at the roadside), driving direction, type, and road surface among other attributes, but no information about the number of lanes. Therefore it is very hard to extract lane-level information out of that data. An example of centerlines and wayarea from ETD is shown in Figure 8.

[53] https://geoportaal.maaamet.ee/eng/Spatial-Data/Estonian-Topographic-Database-p305.html
[54] https://geoportaal.maaamet.ee/docs/ETAK/ETAK_reaalsusmudel.pdf?t=20200306125656
[55] https://geoportaal.maaamet.ee/docs/Avaandmed/Licence-of-open-data-of-Estonian-Land-Board.pdf

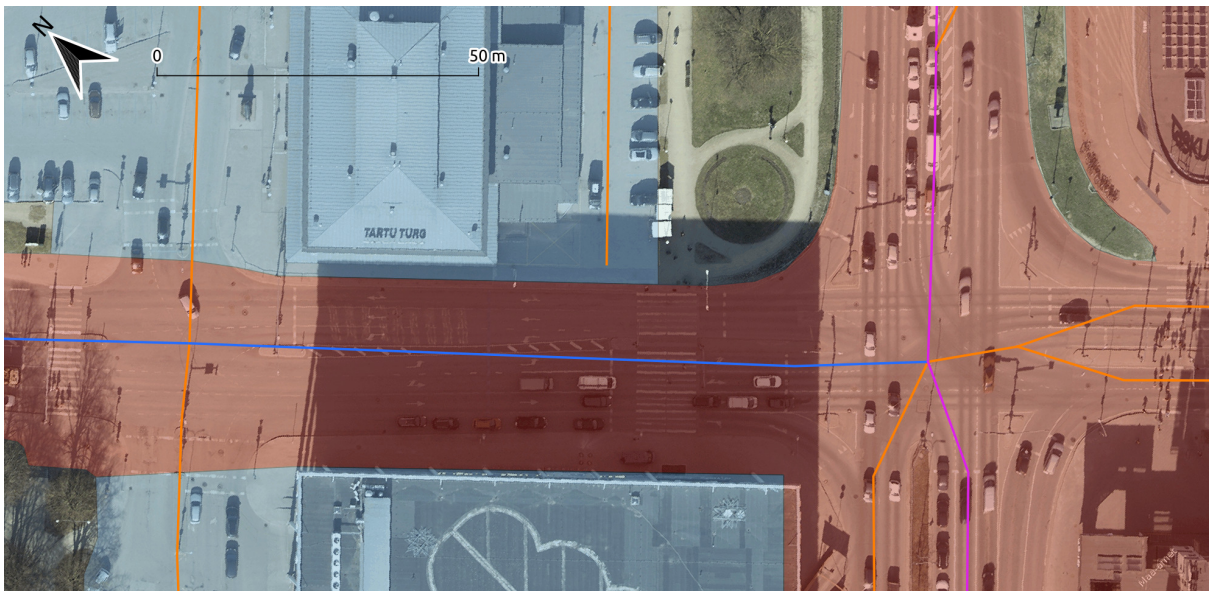

Figure 8. ETD data on top of Orthophoto. Way area (red - traffic area, blue - parking area) and centerlines of the streets (colors different categories), Map data: Estonian Land Board 2021.

Way area from ETD could be used as a way area for the Autoware vector map. The fact that it also includes pedestrian areas next to the road is in this case an advantage as it is used to filter out what objects need to be tracked and pedestrians are among them (they can accidentally step in front of the car and by tracking them already in the pedestrian area we could detect such behavior earlier).

Figure 9. Wayarea from ETD around Delta building, map data: Estonian Land Board 2021.

As an overall estimation, the ETD dataset is too general for AD. For HD maps we would need a much more detailed level in the thematic separation of features, more feature classes (specifically concentrating on lane level elements needed for driving (signs, traffic lights, curbs, road markings, gutter, rails, poles etc), and higher locational accuracy.

3.2.2. Orthophotos[56] from Estonian Land Board

Estonian Land Board (ELB) started to collect aerial images and processing them into orthophotos (aerial photos where distortions caused by terrain relief, camera tilt, and central projection are removed) in 2002[57]. Currently, they are covering half of the Estonian territory within one year. The resulting orthophotos have a Ground Sampling Distance (GSD) of 20-25 cm in the countryside and 10-16 cm in urban areas. An example orthophoto from Riia-Turu crossing in Tartu can be seen in Figure 10. For humans, some of the features are easily seen and recognizable: lanes, whitelines, stoplines, curbs, road edges, pedestrian areas, wayareas. Even poles could be identified and therefore some traffic light and sign locations can be assumed. On the other hand, the processing to extract some of the features automatically is not so trivial. Several solutions exist that was also referenced in the chapter about mapping methods.

Figure 10. Orthophoto from Riia-Turu intersection, map data: Estonian Land Board 2021.

[56] https://geoportaal.maaamet.ee/eng/Spatial-Data/Orthophotos-p309.html
[57] https://geoportaal.maaamet.ee/eng/Spatial-Data/Orthophotos/Orthophoto-metadata-by-year-p350.html

Absolute locational accuracy for orthophotos is in best cases[58] 0.5 GSD in X and Y and 1 GSD in Z coordinate. In our experience, there was a very good match when plotting the trajectories of the car on top of the orthophoto. There is a very good potential to extract the lane information from orthophotos in Estonia. Since urban areas are mapped yearly then we could have very high-resolution data (10 cm GSD) for intersection in Figure 10 from 2018, 2019 and 2020.

3.2.3. Elevation Data from Estonian Land Board

Along with aerial image collection also Aerial Laser Scanning (ALS) is performed and as a result, digital elevation models (DEM) and 3D ALS point clouds are produced. This dataset is also open data and available for everybody. Elevation data can be used to automate setting the elevation for the features in the HD vector map. It is very easy to do using DEM. Most detailed DEM that ELB offers have 1 m GSD.

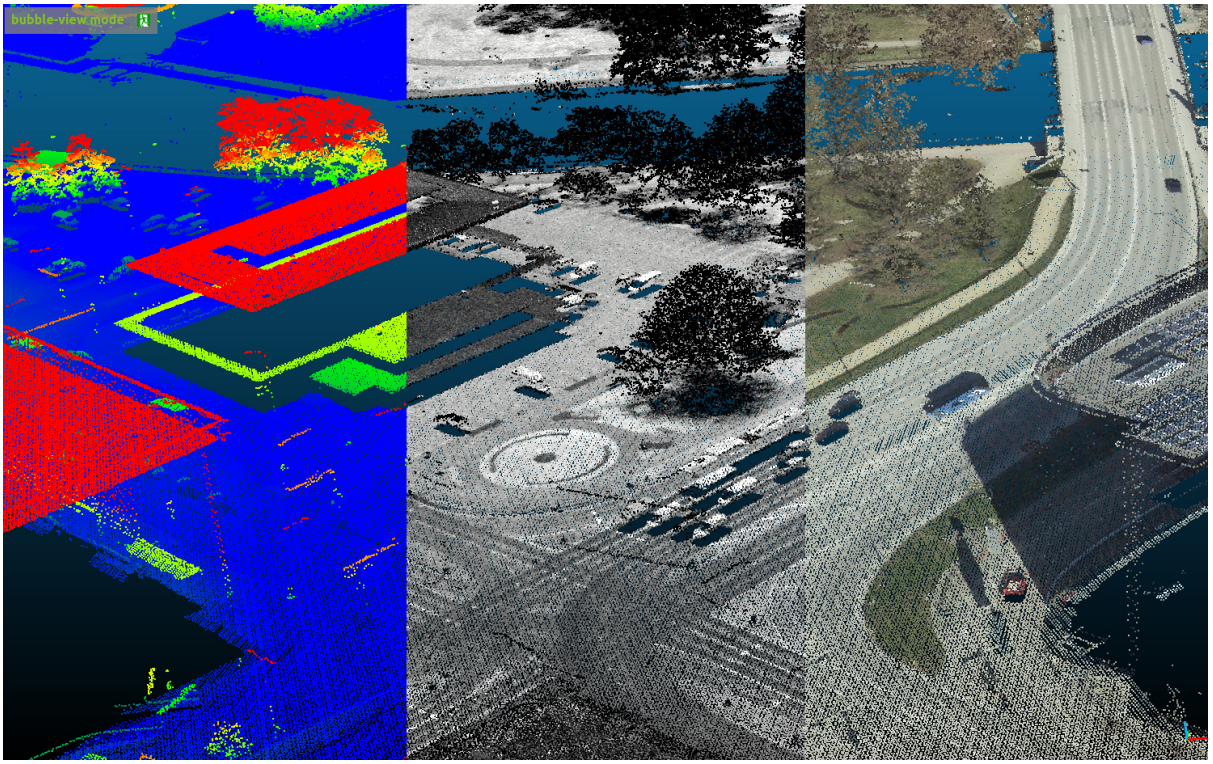

Figure 11. ALS point cloud data from Estonian Land Board 2020. Same point cloud with different coloring: left - elevation (z value), center - intensity, right - RGB color.

Measurement accuracy for ALS point clouds depends on measurement height, angle, and the surface characteristics. Some numerical evaluations in case of hard flat surfaces from 1000 m height in x,y coordinates is 9 cm and in z coordinate is 7 cm[59]. These point clouds are not so dense and feature-rich as point clouds collected from MMS systems, but they might have

[58] https://geoportaal.maaamet.ee/docs/Tutvustus2012_v3.pdf?t=20120125115647
[59] https://geoportaal.maaamet.ee/docs/Tutvustus2012_v3.pdf?t=20120125115647

some advantages. One of the advantages could be the better absolute accuracy of coordinates. We know that MMS systems in urban environments can have bad GNSS quality, hence they rely also on IMU when putting together the point cloud that has the commonly known problem of drift over time. Loop closure algorithms are used to compensate for that. So ALS point clouds or orthophotos could be used to perform automatic georeferencing of MMS point clouds [55].

ALS data could be also used together with orthophotos in one workflow so they could complement each other and as a result, a better pipeline for extracting necessary features could be developed. In the mapping section, some examples are brought. The potential of this dataset could be quite good.

3.2.4. OpenStreetMap[60]

"*OpenStreetMap is built by a community of mappers that contribute and maintain data about roads, trails, cafés, railway stations, and much more, all over the world.*[61]"

The times when OSM was just for some mapping enthusiasts are over. An increasing proportion of edits are made from corporations and their main focus has been on road networks, while non-corporate mappers are more editing buildings and points-of-interest [56]. As of March 2020, nearly 17% of the global road network (measured per kilometer) was edited by a corporate data-team member [57]. From the blogpost[62] by Joe Morrison we can read his opinion on why the big players like Facebook, Amazon, Apple, and Microsoft are actively collaborating and updating the OSM data. One of the reasons is purely economic - paying to google is for map data is expensive and by contributing to OSM and using that they can enrich their own map data. There are also these interesting examples of very targeted mapping, one of them happened when Tesla released its smart summon functionality. It works much better when the car has some knowledge about the parking lots, so the surge of mapping the parking ailes in big parking lots in North America happened [57].

One of the most asked questions about the OSM data is about its accuracy? There is research by Haklay from 2010 [58] and Barron [59] from 2014 and they both agree that this seemingly very simple question is actually hard to answer. In order to answer the question the following information and data are needed:

- ground truth data to compare it with (how correct is the ground truth data? the outcome will be accuracy relative to that);
- accurate for what? what is the task we are solving: geo-coding, searching POI's, map-applications, routing, AD;
- accuracy as data quality aspects: positional accuracy, attribute accuracy, logical consistency, completeness of the data, semantic accuracy, temporal quality.

[60] https://www.openstreetmap.org
[61] https://www.openstreetmap.org/about
[62] https://joemorrison.medium.com/openstreetmap-is-having-a-moment-dcc7eef1bb01

Haklay compared the data in OSM against Ordnance Survey data, but the results are by now out of date. Barron was suggesting a framework and did not provide actual numbers. There are also more recent studies to evaluate OSM accuracy, but they wouldn't have much use in the current context because the data in OSM can be in a very different quality level also regionally.

There is one recent example from Norway where Gran used OSM data in his Master Thesis to create an HD vector map for the Apollo framework. His conclusion was that it might not have all the data, attributes might be lacking and it might not be up to date, but it still has some useful information and for example, the road network topology was acquired from there as it was not so conveniently accessible from Norvegian national road data bank [48].

So the data quality of OSM should be assessed with specific tasks in mind. In our case, we would like to solve the AD task within our demo route. Requirements for the map are described in the previous section. When looking at the features that can be mapped in OSM then we could have quite a lot of useful data from there. All available features and their possible tags (reality and data model) can be accessed on OSM Map Features page[63]. Let's look at some examples:

- Each lane is not separately mapped, but the number of lanes can be defined. The tag oneway could be set to true. If a road is not a one-way road then backward and forward lanes can be separately defined. Mapped roads from OSM can be seen in Figure 12. The problem here is that we don't have information about lane widths, sometimes lane number is not specified, so automatic generations is quite problematic. Interesting is also if the digitized road is symmetric with respect to lanes or driving directions.

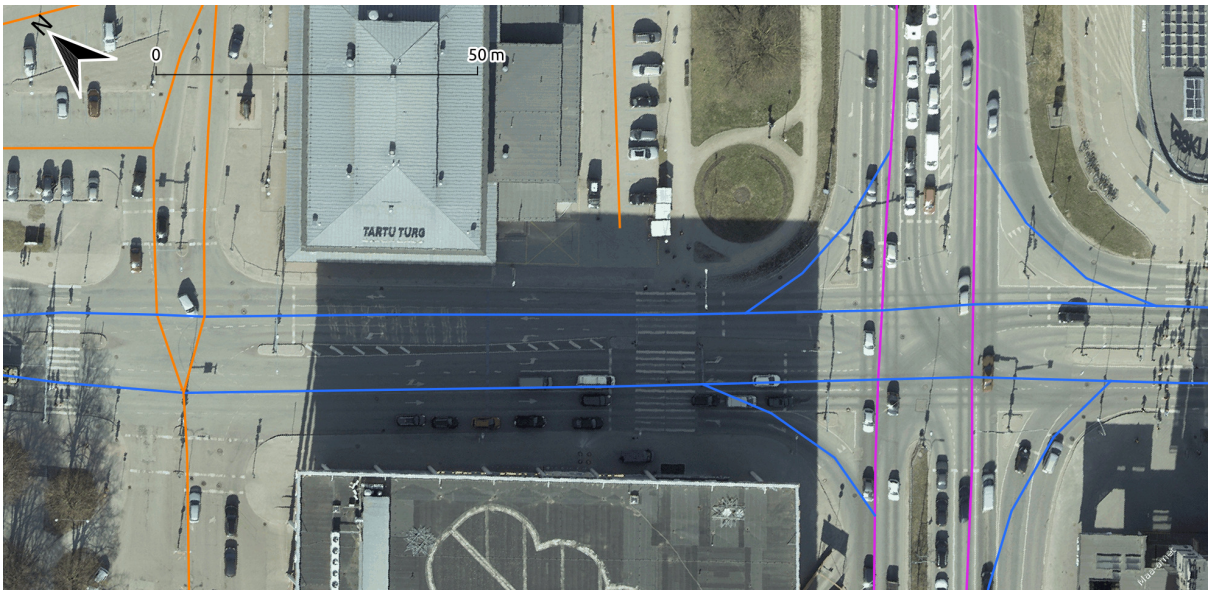

Figure 12. OSM road centerlines plotted on top of the ELB orthophoto.

- There is an option to map pedestrian crossings and these are mostly available in our demo route locations.

[63] https://wiki.openstreetmap.org/wiki/Map_features

- Traffic lights in OSM are associated with nodes and they use the same node that is used also in representing road geometry and located where the stopline is. So if there are 3 lanes in parallel, but it is digitized as one road then there will be only one point. In reality, there might be 3 traffic lights for all the lanes in different places (right-hand side of the road, hanging above the road, across the intersection). No such information is present in OSM, so we can not get actual locations for traffic lights that are needed for our camera-based traffic light recognition.
- In very few cases there are stopping or yielding points marked on the road, but the main problem is again the missing lane level detailedness of that information.
- Curb information is not present and very hard to reconstruct. We can assume that it is between road and sidewalk, but without width information and if lane information is missing then it is very hard to reconstruct it with the required 5-10 cm level accuracy.
- Road markings are not present.
- Routing and topology information can be acquired at the general road level, but as we are struggling to reconstruct the lane level road network we will have problems also with smooth trajectories. Yes, they can be made smooth, but not reflect reality.
- Maximum allowed speeds are there in OSM data, but for speed profile to be used in longitudinal control they are directly not suitable, since speeds need to decrease in curves. The option is to add some other control mechanism for example based on the curvature of the trajectory to calculate the speed.
- There is more additional information that can be used or integrated for creating HD maps, like semantic information about parking places, bus stops.

In general, there are some features from OSM that could be used, but the automatic creation of lane level map data suitable for AD needs is not so straightforward and possible to quickly implement. The solution from [48] also makes some simplifications by taking just the road network, but the width of the road and the number of lanes are additionally manually specified.

3.2.5. Mapillary

Mapillary[64] is a company that collects street-level imagery from its users and creates maps out of them. Images are processed and a multitude of features are extracted using different computer vision and deep learning-based algorithms. All the extracted data, like traffic signs, traffic lights, road markings, etc are georeferenced and are available through the Mapillary platform. To download the data organization-level account is needed and the area of interest needs to be defined. The accuracy of the dataset depends on many things. As Mapillary uses Structure from Motion (SfM)[65] techniques to create 3D scenes out of uploaded images and coordinates for the features that come from these scenes then it is clear that more images provide better reconstruction and better locational accuracy.

[64] https://www.mapillary.com/
[65] https://github.com/mapillary/OpenSfM

Another important factor is the location accuracy of the images since this will be used as one of the starting points in creating the 3D scene. Since the images are captured using mobile devices and quite often while moving and in the city environment where there is not the best reception for GNSS signal then interesting deviations can be seen. Figure 13 displays some traffic signs (pedestrian crossing, speed limit) and traffic lights in the Keskpark green area. It is clearly the result of one wrongly placed track. When clicked on the image we can clearly identify that the actual location is next to Kaubamaja on Uueturu street facing towards river Emajõgi, so the traffic signs and lights should be actually at the intersection of Uueturu and Vabaduse puiestee.

Mapillary dataset is a bit raw yet to be taken and used for AD and HD maps. This might not hold for some other locations, like highway sections with more accurate localization and more images. In general, Mapillary provides data about all traffic signs and an impressive list of map objects, like lane markings, poles, manholes, traffic lights, street lights, banners, barriers etc. Additionally, objects are detected from images, and also this data is available.

Figure 13. Mapillary[66] placing traffic signs and lights in Keskpark in Tartu city center. Traffic signs and lights in the red rectangle should be placed inside the green rectangle.

66

https://www.mapillary.com/app/?lat=58.37939450286123&lng=26.725971079399145&z=17.6684711269959&menu=false&signs=true&points=true&mapFeature%5B%5D=object--traffic-light--general-upright&line=true&pKey=lyQEMplgqT2EIxH4RVZskH&focus=map&mapStyle=Mapillary+streets&x=NaN&y=NaN&zoom=0&detections=true

3.2.6. Other data sources

- Recorded trajectories by the Autonomous Driving Lab self-driving platform, can be used for automating lane generation, in the current thesis they were used as a reference for manual digitizing.
- TraffEst (the company managing most of the traffic lights in Tartu) - no database with exact locations of traffic lights. API access to online traffic light information was provided.
- Tartu City Government - has 3D information about the topography and houses that could be helpful for building a simulation environment. Also, spatial information related to city planning and asset management tasks.
- Tallinn City Government - our data needs for mapping the Tallinn demo route were so small that we did not ask.
- Commercial map data providers: were not considered, would have to pay for the data. Estonian companies: Regio, Reach-U; Global: Here, TomTom.
- EyeVi Technologies - point cloud map, they would also have road surface orthophotos and much denser point clouds that could be used for object detection and mapping.
- Estonian Road Administration - most of their data is outside of the cities - state-owned highways. Inside the city, road data is mainly managed by local municipalities.
- Free satellite images (for example Sentinel data) - most probably not enough to satisfy locational accuracy. Quite costly high-resolution images from commercial providers could be an option, but were not considered in this thesis. More useful for countryside areas (road visibility not blocked by houses or other objects).

3.2.7. Comparison of data sources

It is quite hard to estimate the data quality and suitability of discussed data sources in general, with no set purpose and specific requirements in mind. For that reason comparison of different data sources will be carried out with respect to the current needs of UT ADL. It means that the comparison reflects our current approach and ODD that might change in the future.

One very important factor for current choices in mapping is quite short demo routes and therefore the area to be mapped is also very small. It allows manual digitization as one viable option for creating the map. In the future, we definitely need to make use of the datasets discussed and the following comparison (Table 4) is taking that into consideration. Datasets are assessed with respect to features as they are defined in the Autoware vector map format and the following scale is used for evaluation:

0 - information is not there;
1 - information is incomplete, not worth trying;
2 - partial information, needs processing with heuristics
3 - full information, not in a suitable form (e.g. images), needs costly postprocessing

4 - full information, relatively easy processing (e.g. conversion of points to curves)
5 - full information in a suitable form

Table 4. Evaluation of extracting the needed data for Autoware vector map feature layers from reviewed data sources.

Autoware Vector Map feature layers	ETD	Orthophotos	ALS from ELB	OSM	Mapillary	Other: MMS point cloud	Other: recorded trajectories
lane	2	3	2	2	0	3	4
dtlane	2	3	2	2	0	3	3
whiteline	0	3	2	0	2	3	2
signaldata	0	1	1	1	2	3	0
roadsign	0	1	1	1	2	3	0
pole	0	1	1	0	2	3	0
curb	0	3	3	1	2	3	1
roadedge	1	3	3	1	2	3	1
stopline	0	3	2	2	2	3	0
intersection	2	3	3	2	2	3	2
crosswalk	1	3	2	3	2	3	0
wayarea	5	3	3	3	2	3	2

Main conclusions:

- No lane-level vector data was freely available. Other than collected trajectories from drives;
- No data where it could be easily and automatically derived;
- There are some potential datasets that can be used, but more sophisticated processing would be needed to develop (orthophotos, point clouds);
- There are some potentials of combining different data sources that might lead to some more accurate results;
- Car imagery and point cloud along with car perception could be used and integrated into mapping pipeline;
- Some not so important data could be easily acquired, like wayarea from Estonian topographic database or buildings;
- When combining them data quality issues (thematic, positional, temporal accuracy, completeness and logical consistency) need to be carefully considered.

3.3. Available tools for creating Autoware vector maps

In the following section, an overview of available tools meant for creating Autoware vector maps is given.

3.3.1. Vector Map Builder

Vector map builder[67] is referenced from Autoware github[68] page as one of the Autoware map tools[69] that is made available by TierIV[70]. Drawing of the vector map assumes point cloud and the map is drawn on top of it in 3D view over the web browser. The requirement of point cloud is also brought out in [52] and additionally, it is added that this tool is not reliable (it might not be available due to heavy load by other users, since it is an online tool and shared by many users).

The requirement of needing the point cloud means that the created vector map will have coordinates relative to the point cloud and if the point cloud does not have good geometry and is not aligned with any CRS then it is really hard to use it with other datasets.

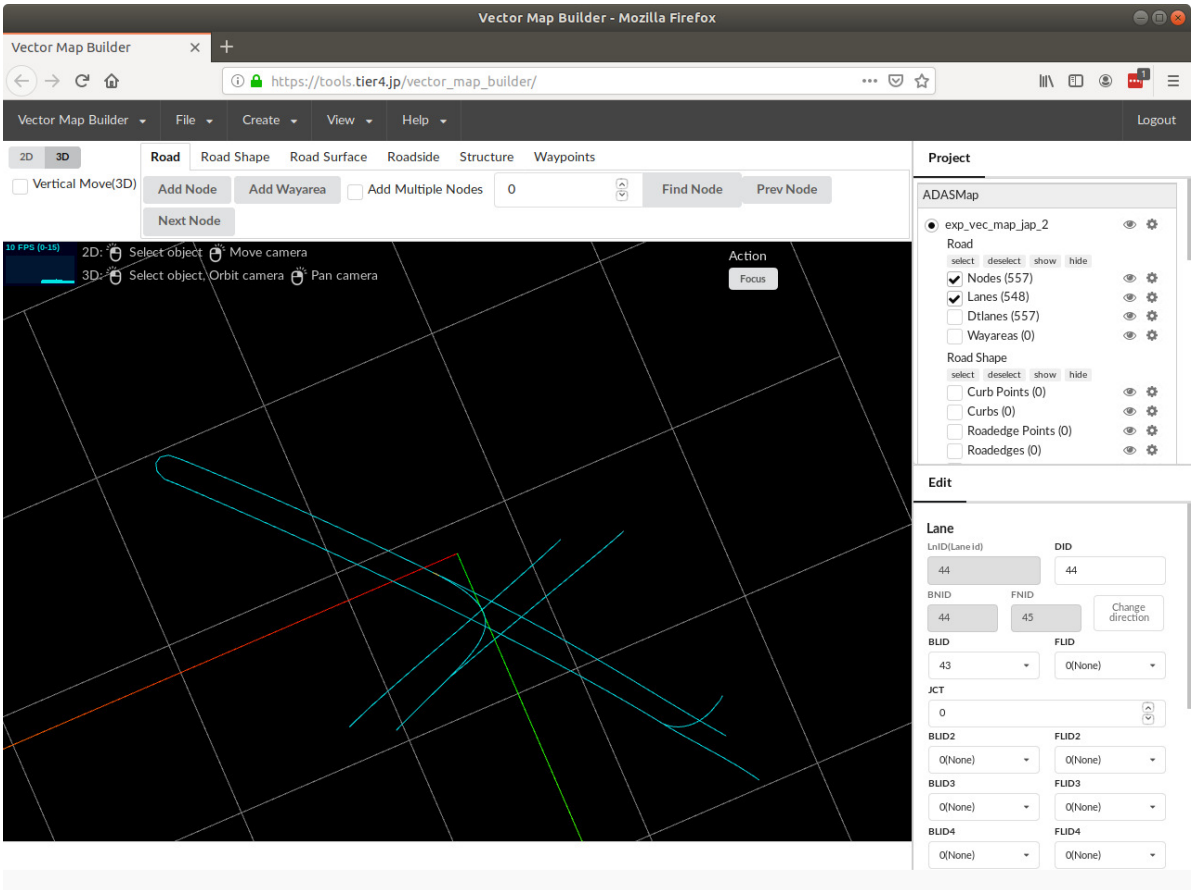

Figure 14. The user interface of the Vector Map Builder.

[67] https://tools.tier4.jp/feature/vector_map_builder/
[68] https://github.com/Autoware-AI/autoware.ai
[69] https://tools.tier4.jp/
[70] https://tier4.jp/en/

The user interface (UI) of the tool can be seen in Figure 14 with some lanes drawn and attributes of the lanes that can be manually edited. Drawing the features is done by adding one point at a time and therefore very time-consuming.

3.3.2. MapToolbox

MapToolbox is a Unity plugin for making Autoware vector maps[71]. At the time only a general description and a tutorial video were available. The tool itself is quite simple so the lack of documentation was not restricting the usage of the tool. Vector map drawing is based on top of the point cloud and no support for different coordinate systems or different data sources. Only a limited amount of different features can be added: lanes, white lines, road edges, curbs, stop lines and traffic lights. In general, it can be quite enough for simpler maps. The big downside is the necessity to have a point cloud and the whole Unity to be installed.

Drawing the features was quite convenient, but all the digitized lines were splines (no straight lines). Snapping the features was also possible that makes creating a connected network much easier. Sometimes maps were not working when exported, but no options for validation were present.

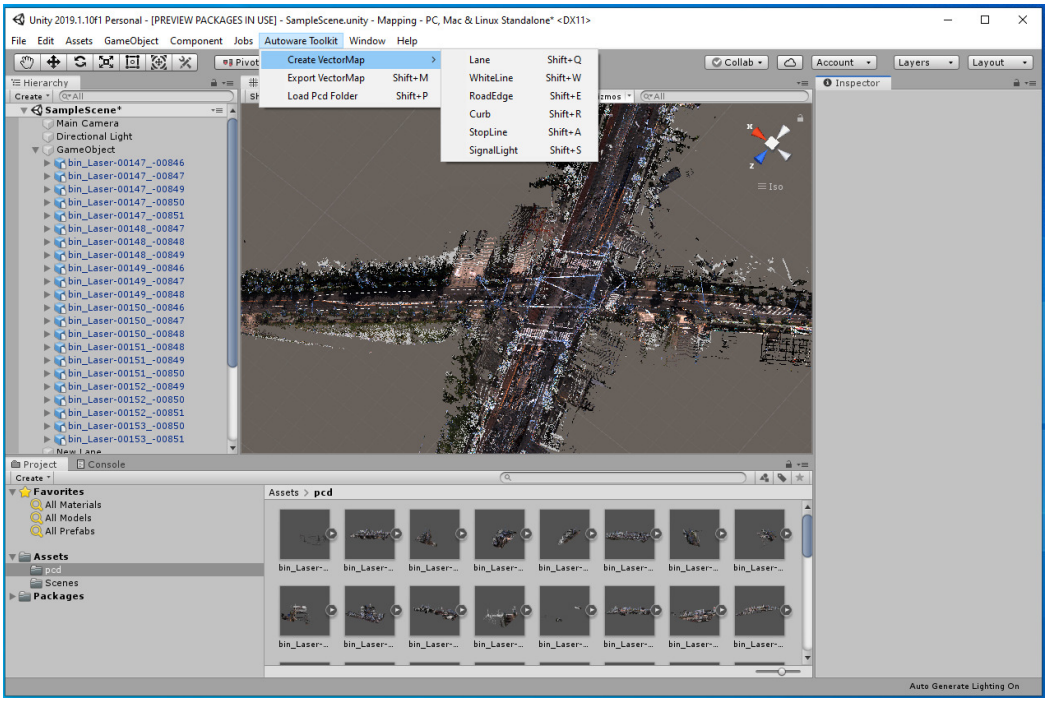

Figure 15. The user interface of Unity-based vector map creation tool MapToolbox.

3.3.3. Unity LGSVL Map annotation Tool

Unity LGSVL Map annotation Tool[72] is quite a heavy tool. It requires Unity and the LGSVL simulator to be installed. Map editing or scene annotating is suggested while running the

[71] https://github.com/autocore-ai/MapToolbox
[72] https://www.lgsvlsimulator.com/docs/map-annotation/

simulator as a Unity project in a Windows environment. It was tested a bit with example Unity scenes that were available with the LGSVL simulator, but the process was quite slow and complex. To make a new map there must be a Unity scene for that area and the solution just didn't seem to be reasonable to create a new map. As a positive side, it supported exporting annotated vector map to 4 different formats.

3.3.4. Assure Maps

Assure Maps mapping tool is available from github page[73]. The creator of this program is also the author of the OpenPlanner in Autoware.ai. During the testing, the software was not very mature. Also currently it supports only a limited amount of features for AutoWare vector map format (lanes, waypoints, road lines, traffic lights, stop lines, traffic signs, road boundaries, crossings), but still sufficient to do the routing and satisfy the basic requirements also for ADL platform. The main downside is again relying on the point cloud when drawing a map. This tool supports loading more different HD vector map formats (OpenPlanner kml, Autoware Vector map, OpenDrive, lanelet2) than other simple tools. There seems to be added support for storing also the CRS information when saved to .kml or .osm format. Still can't see this tool to be used when bigger areas would need to be mapped.

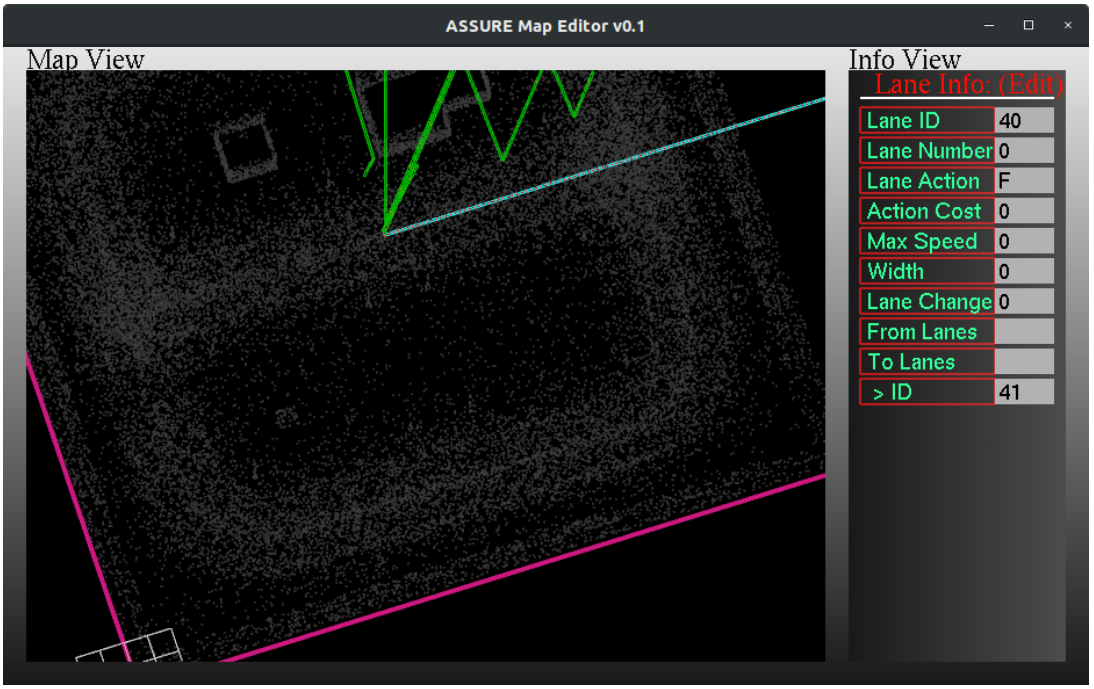

Figure 16. User Interface of the Assure mapping tool

3.3.5. Simple Vector Map Tool

Simple Vector Map Tool [52] is meant to fulfill minimum requirements for a vector map so that it can be used in Autoware.ai and OpenPlanner for path planning. It means that only the

[73] https://github.com/hatem-darweesh/assuremapingtools

following files will be created: point.csv, node.csv, dtlane.csv, lane.csv. The workflow is illustrated in Figure 17.

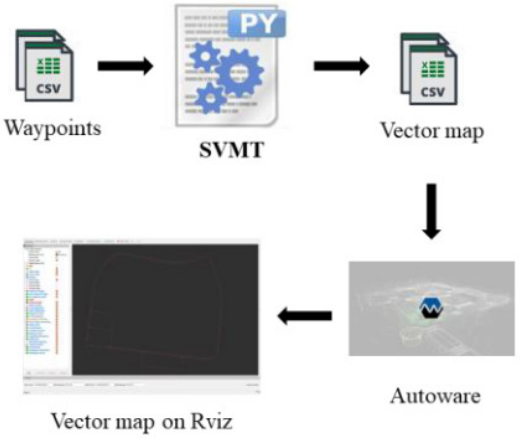

Figure 17. The workflow of Simple Vector Map Tool [52].

This tool assumes that each lane has a separate waypoint file and connects these waypoints into a lane network that is suitable for routing an additional file that specifies the connections that need to be created. Adding other features like stop lines or traffic lights is not possible using this tool. The code for the tool can be accessed in github page[74].

3.3.6. Comparison of tools for creating Autoware vector map

A short comparison of different tools is shown in Table 5. Some additional explanation for the following fields is given:

- Supported features (features that can be created using this tool):
  - Minimal: lane, dtlane, point, node;
  - Medium (adding to minimal): stop line, traffic light, curb, crosswalk;
  - High (adding to medium): boundary, intersection, pedestrian path, wayarea
  - Most: all that is included in previous and some more
- Time estimation - subjective relative estimation of how much time it would take to create the same map with different tools.

Table 5. Comparison of available tools.

Tool Name	Vector Map Builder	Map toolbox	LGSVL Map Annotation Tool	Assure Maps	SVMT
Software requirements	web browser	Unity and a plugin	Unity, LGSVL simulator, plugin	Assure Mapping tool	Python

[74] https://github.com/wntun/Autoware/blob/master/SimpleVectorMapperTool.py

Access	Free (account needed)	Free (account needed)	Free (account needed)	Free	Free
Open-source	No	No	No	No	Yes
Map creation	One point/node at the time	Draw lines / points	Draw lines / points	Draw lines / points	no
UI type	3D view	3D view	3D View	3D View	No UI
Starting point / assumptions	pcd map	pcd map	Unity scene	pcd map	Waypoint files
Automation / scripting	No	No	Semi	No	Semi
Support for multiple CRS	No	No	No	No	No
Documentation	Minimalistic	Minimalistic	Available	Available	Available
Save / export HD map formats	- Autoware vector map - Waypoint file	- Autoware vector map - Lanelet2	- Autoware vector map - Lanelet2 - OpenDrive - Apollo OpenDrive	- Autoware vector map - OpenPlanner kml file - Lanelet2	- Autoware vector map
Load / import HD map formats	- Autoware vector map - Waypoint file		- Autoware vector map - Lanelet2 - OpenDrive - Apollo OpenDrive	- Autoware vector map - OpenPlanner kml file - Lanelet2 - OpenDrive	- Waypoint file
Supported features	Most	Medium	High	High	Minimal
Point cloud support	yes (pcd files)	yes	yes	yes	no
Time estimation	Very slow	medium	slow	medium	medium
Potential hand-ling large maps (city scale)	very low	low	low	low	low

3.4. Custom workflow for creating the HD maps

Considering our needs and requirements from the platform side, reviewing available datasets and analyzing the tools a custom workflow for creating the HD vector maps was created. Comparison of map requirements and the available tools for Autoware vector map creation showed that none of the tools satisfy all the requirements (see Table 6). All the reviewed tools have at least 3 cases where the requirements are not met. Additionally, all the tools, except

Simple Vector Map Tool, have a 3D environment for drawing features that is consistent with the requirement of having a point cloud or 3D scene as a basis for the drawing. It is good when all the details are drawn straight to their correct heights, but for larger-scale drawing it makes the process a lot slower and is not so ideal also for bigger areas.

Table 6. Compare the tools with respect to HD map requirements (+ - meets the requirements, +/- - can handle with some inconveniences, - - does not meet the requirements; the tools: Vector Map Builder - VMB, Map Toolbox - MTB, LGSVL Map Annotation Tool - LGSVL, Assure Maps - AM, Simple Vector Map Tool - SVMT).

HD map requirement	Extended explanation	VMB	MTB	LGSVL	AM	SVMT
Achieve 5-10 cm level accuracy	All tools capable	+	+	+	+	+
Create lane-level detailed road network	No suitable existing dataset, would need to create from scratch	+	+	+	+	+
Minimum set of features needed	curbs, lanes, whitelines, traffic lights and signs, crosswalks, intersection, stoplines	+	+	+	+	-
Topologically correct lane network	Would need snapping during digitizing procedure and tools to check the topology	+/-	+/-	+/-	+/-	+/-
Use custom speed profile	Modify / edit attribute values	+	-	+	+	+
Add custom attributes	Could be useful for customized mapping to make it more faster or convenient	-	-	-	-	-
Smooth trajectories for car	Ability to create/use scripts that would automate some tasks	-	-	-	-	+/-
Support for L-EST97 and other CRS's	Data might be in different CRS, so it would be good to be able to support multiple CRS's or make coordinate transformations	-	-	-	-	-
Create HD map from scratch	There might be no data available for the area, but still a map is needed. Most of these tools assume the presence of point cloud or 3D scene in order to start.	-	-	-	-	+

3.4.1. QGIS

During the first tests, the need for a simple HD map that could be used with our AD platform in some remote area with enough space and without disturbing other vehicles and pedestrians was obvious. Possible locations were scouted using well-known map portals like Google

maps[75] and a web map from Estonian Land Board called XGis[76]. The latter one has orthophotos with much better resolution. When the trajectories of the car were plotted on top of the orthophotos seemed to match very well and it gave an idea to draw the map on top of them. As a result, a decision was made to build a workflow of creating the maps using QGIS[77], an open-source Geographic Information System (GIS) to handle spatial information. The author was already previously familiar with the software and was aware of the availability of Web Mapping Service[78] (WMS) from the Estonian Land Board that offers the capability to use all recent and historical orthophotos as a background map in the QGIS environment without the need for downloading.

Some considerations supporting QGIS as the tool for creating HD vector maps:

- open-source and freeware with the big active community;
- easy to use with Estonian free data sources and WMS services;
- support for a large number of spatial information file formats and databases;
- support for map projections and different coordinate systems;
- a lot of existing tools and plugins (checking the topology, smoothing linestrings,...);
- integrated with Python, relatively easy scripting;
- could easily handle HD vector maps covering large areas;
- can be upscaled to using spatial databases like PostGIS as backend and data storage;

One of the negative aspects was the lack of support for large 3D point clouds and 3D map editing. On the other hand, it was evident from tests with other tools that drawing the features in 3D is slower than in 2D. It was also quite clear that the third dimension (z coordinate) for the Autoware vector map format can be easily automatically added from other sources during the conversion step. QGIS was also used in [24] for road linking points and data organization.

3.4.2. Development of map creation procedures

As it is quite unintuitive to create the Autovare vector map straight away (lines and polygons are broken apart into points and spread up into different csv files), it was decided that the optimal way is to create our own map that would be later converted to Autoware vector map format. This allowed us to make the process of map digitizing to be much more convenient and as easy as possible. Still, certain rules have to be followed in order to make possible the automatic conversion of the digitized map. In the field of geoinformatics the rules what to digitize and how are typically fixed in reality and data models:

- Reality model - a simplified description of the real-world features that are mapped. Available in appendix II.
- Data model - description of how these features will be represented and mapped in the database, see appendix III.

[75] https://www.google.com/maps/
[76] https://xgis.maaamet.ee/maps/
[77] https://www.qgis.org/en/site/
[78] https://www.ogc.org/standards/wms

- Presentation model - defines the cartographic visualization of the features, so that the reality model can be displayed as a map. The presentation model is important for humans to make it visually more interpretable. Different visualizations can be used for the HD vector map and since they have no effect on how it is converted to Autoware vector map format then it is not thoroughly documented in this thesis. Only a general description aside from the maps in the form of the legend is given.

All the data is maintained in the Estonian coordinate system L-EST97, as all the source data is available in these coordinates. Whenever there is a need, coordinates can be transformed to any other CRS. The data is stored in shapefiles[79] - vector data format that can store geometry and related attribute data of the features, originally developed by Esri[80].

3.4.3. Steps for map creation in QGIS

1. Data collection in the field: collecting trajectories of the mapped area with the car
2. Mapping the features (defined in reality model) by following the rules in the data model.
    - collected trajectories are plotted on top of the orthophotos (orthophotos are displayed in QGIS using Web Mapping Service from ELB, different years are available);
    - attribute data is added for the features during the mapping process;
    - speed values are taken from collected trajectories;
    - Traffic light heights are measured from the point cloud and then checked and tested on the rosbags.
3. Applying the script with smoothing and generalization for lanes. The script is built with QGIS model builder and includes the following:
    - Tool *Smooth*: with 2 iterations and allowed offset value of 0.25 m;
    - Tool *Simplify*: distance-based simplification with Douglas-Peucker algorithm and tolerance of 0.03 m;
4. Validating the topology of the road network. QGIS core plugin Topology Checker[81] is used with a rule that lanes must not have dangles.
5. Script to prepare lanes for exporting (adding the intersection IDs and splitting them to 1m sections).

After these 5 steps are completed, the map is ready to be converted into Autoware vector map format.

[79] https://en.wikipedia.org/wiki/Shapefile
[80] https://www.esri.com/en-us/home
[81] https://docs.qgis.org/3.4/en/docs/user_manual/plugins/core_plugins/plugins_topology_checker.html

3.5. Converter to Autoware vector map

The converter takes map layers (each layer has its own shapefile) and converts them to Autoware vector map format. Converter assumes that data in shapefiles are formatted according to reality and data model. The converter is written in Python and the following libraries are used: pandas[82], geopandas[83], numpy[84], fiona[85], rasterio[86], os, glob.

The general structure and the relations between the files used in the Autoware vector map format are shown in Figure 18. It is possible to create and utilize more relations, also more tables could be used (for example roadedge.csv, utilitypole.csv), but at the current state of the converter, only the features of the Autoware vector map visualized in Figure 18 are used.

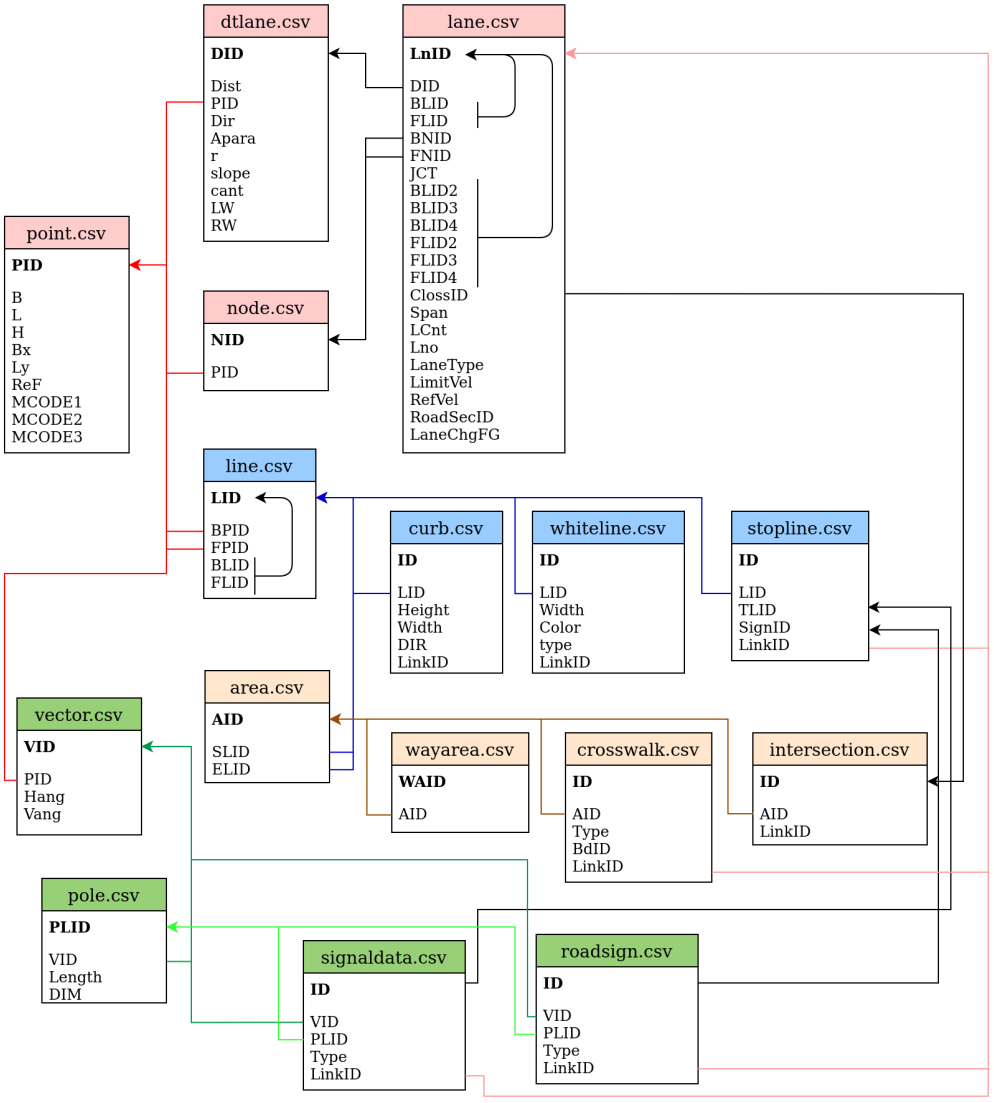

Figure 18. File structure and relations between the features used in AutoWare vector map format.

[82] https://pandas.pydata.org/
[83] https://geopandas.org/
[84] https://numpy.org/
[85] https://pypi.org/project/Fiona/
[86] https://rasterio.readthedocs.io/en/latest/intro.html

3.5.1. Converter functionality

In this section step-by-step functionality of the converter is described. A description follows the order of how the conversion from HD vector map to Autoware vector map format is done. HD vector map layers digitized in QGIS (map layers defined by data and reality model - *.shp files) are written into Autoware vector map format (*.csv files) following the steps described below and visualized in Figure 19:

1. Use **lane.shp** and write related data to respective csv files:

- **point.csv**:
  - geometry is extracted (list of x and y coordinates in L-EST97 CRS) and point.csv is filled with all individual points;
  - overlapping points from the start and end of the lanes are added only once;
  - Initially, 0 elevation is used for lanes as all following elements that have height are mapped relative to lane heights (height from ground);
- **node.csv**: point.csv to node.csv relation is created;
- **dtlane.csv**: attribute data from lane.shp describing centerlines is extracted;
- **lane.csv**: lane connections (topological network for routing) are created.

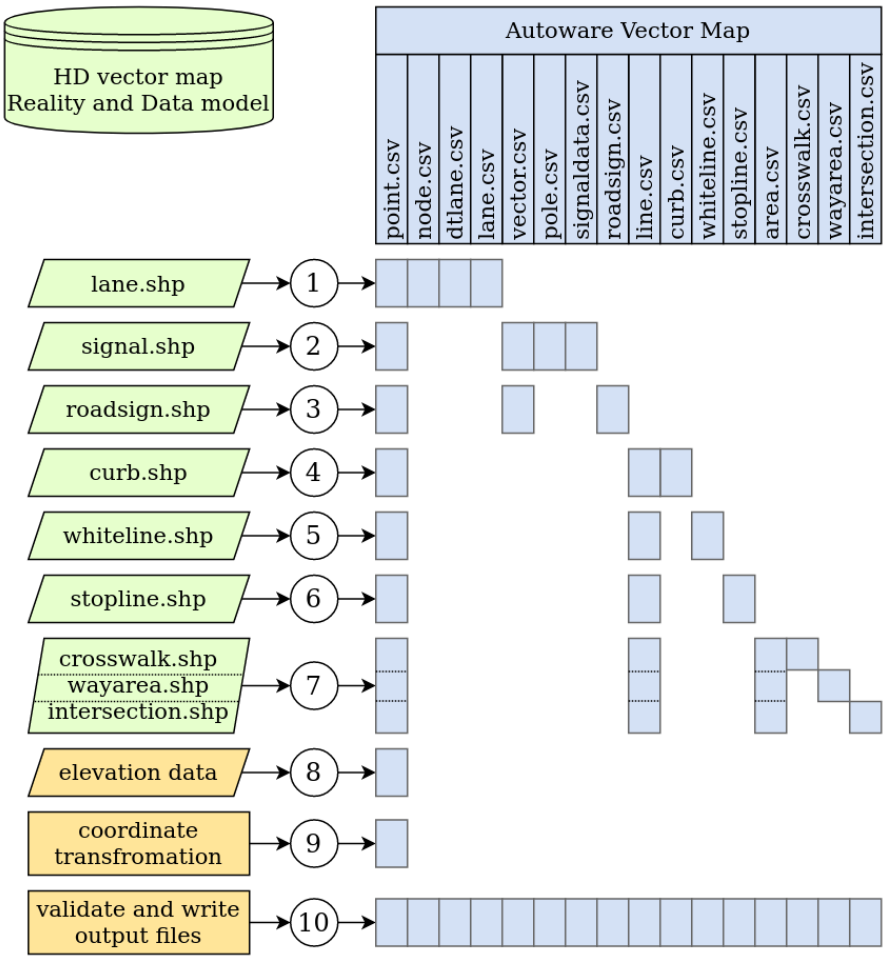

Figure 19. Converter steps in writing the data into Autoware vector map format.

2. Use **signal.shp** (traffic lights) iterate over all elements and fill the data to:

- **pole.csv**: for each traffic light a pole is created;
- **point.csv**: extract the location of each traffic light lamp;
    - PID - point ID incremented respectively;
    - fill in Bx and Ly coordinates for point locations;
    - parse elevations from Heights in signal.shp and add to point.csv H;
- **vector.csv**: each lamp will be described as a vector;
    - link vector to correct point ID;
    - extract horizontal and vertical angles from signal.shp Hang and Vang;
- **signaldata.csv**:
    - link each lamp with the correct pole and vector;
    - add lamp color from signal.shp.

3. Use **roadsign.shp** to fill the data in the following csv files:

- **point.csv**: similar logic as in 2. point.csv;
- **vector.csv**: similar logic as in 2. vector.csv;
- **roadsign.csv**: link to correct vector and add roadsign Type from roadsign.shp.

4. Use **curb.shp** to fill the data in the following csv files:

- **point.csv**: extract all the points from line strings and add to pints.csv;
- **line.csv**: add new line elements;
- **curb.csv**: link with correct line and add Height and Width from curb.shp.

5. Use **whiteline.shp** to fill the data in the following csv files:

- **point.csv**, **line.csv** - similar logic as in point 4;
- **whiteline.csv**: link with correct line and add Width, Color and Type from whiteline.shp.

6. Use **stopline.shp** to fill the data in the following csv files:

- **point.csv**, **line.csv** - similar logic as in point 4;
- **stopline.csv**: link with correct line and use stopline middle point to link it with a traffic light, stop sign and lane ID's.

7. Use **crosswalk.shp**, **wayarea.shp** and **intersection.shp** layers to fill the data in the following csv files:

- **point.csv**, **line.csv**: similar logic as in point 4;
- **area.csv**: create areal elements and increment ID's, link with line.csv;
- **crosswalk.csv**: create table and link with the area and add a reference to lane;
- **wayarea.csv**: create table and link with the area;
- **intersection.csv**: create area and link with the area;

8. **Add correct elevation to points.csv**. In this step, all features are lifted to their correct heights using DEM with a 1m ground sampling distance from Estonian Land Board.

- The elevation is extracted for all the points in point.csv files using rasterio library and coordinates in Bx,Ly

- Extracted elevation values are added to existing values in H attribute in point.csv.

9. **Perform x,y coordinate transformation**. Cartesian coordinates used in Bx and Ly are currently in L-EST97 CRS. These coordinates are quite big (6 and 7 digits) and Rviz (visualization software for ROS) has problems visualizing so big coordinates, so the following transform is performed:

- The shift for easting is -650000 and for northing is -6465000
- Coordinates in **point.csv** Bx and Ly are updated by applying the shift

10. **Perform checks** on values and data types and **write all the output** files to the specified directory.

3.5.2. HD vector map testing

A converted Autoware vector map is then tested before it is used in real-life traffic. Tests include the following steps:

1. testing in OpenPlanner simulation:
    a. mainly map loading and routing is tested;
    b. traffic lights and stopping before stop lines;
    c. simulated obstacles can be added;
2. parking lot tests, here the Autoware vector map is transformed (necessary shift and rotation are applied to coordinates in point.csv) to match the location of the parking lot, Figure 20. Parking lot tests are used for:
    a. speed profile verification;
    b. lateral control (keeping the trajectory);
    c. tests with obstacles;
3. the HD vector map is ready to be tested on the streets.

Figure 20. Tallinn demo route lanes transformed into ERM parking lot in Tartu for testing, background orthophoto from Estonian Land Board.

Source code for map converter is available upon request.

4. Results

4.1. Examples of vector maps

Several maps have been created using the developed workflow. Some maps are fairly simple to test specific scenarios in the parking lot. Two bigger maps that have been created are the Tartu demo route map (Figure 21) and Tallinn demo route map (Figure 22). Some parts of the Tartu demo route map are drawn by the IT students Karl-Johan Pilve (Narva mnt starting from Delta roundabout towards Raadi and the Ermi street to ERM B-parking lot) and Krister Looga (Muuseumi tee, Roosi and Puiestee streets) with supervision from author.

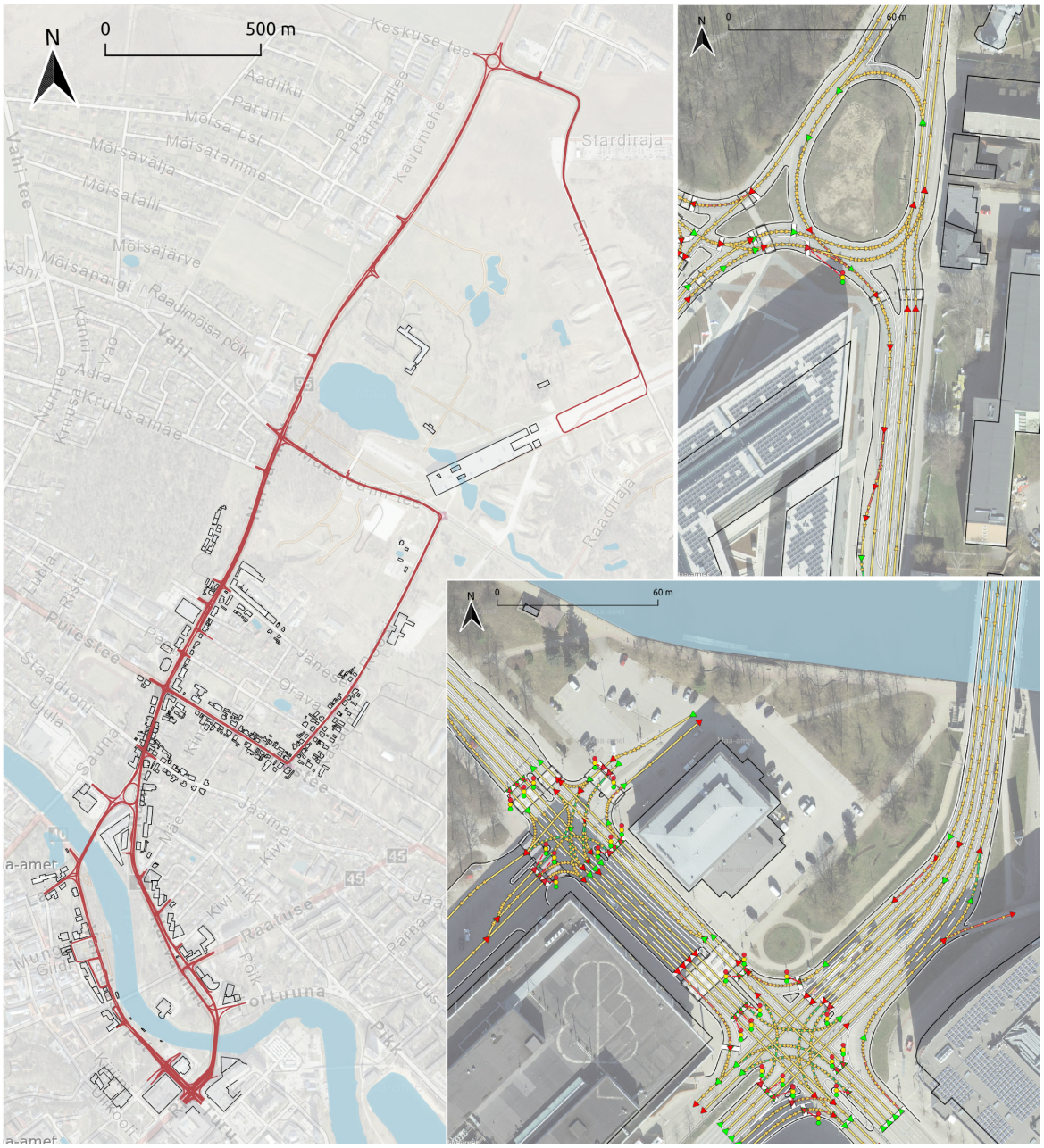

Figure 21. Area covered with HD map in Tartu. All the extent of the lanes on the left side and detailed outcrops on the right side. Background orthophoto from Estonian Land Board.

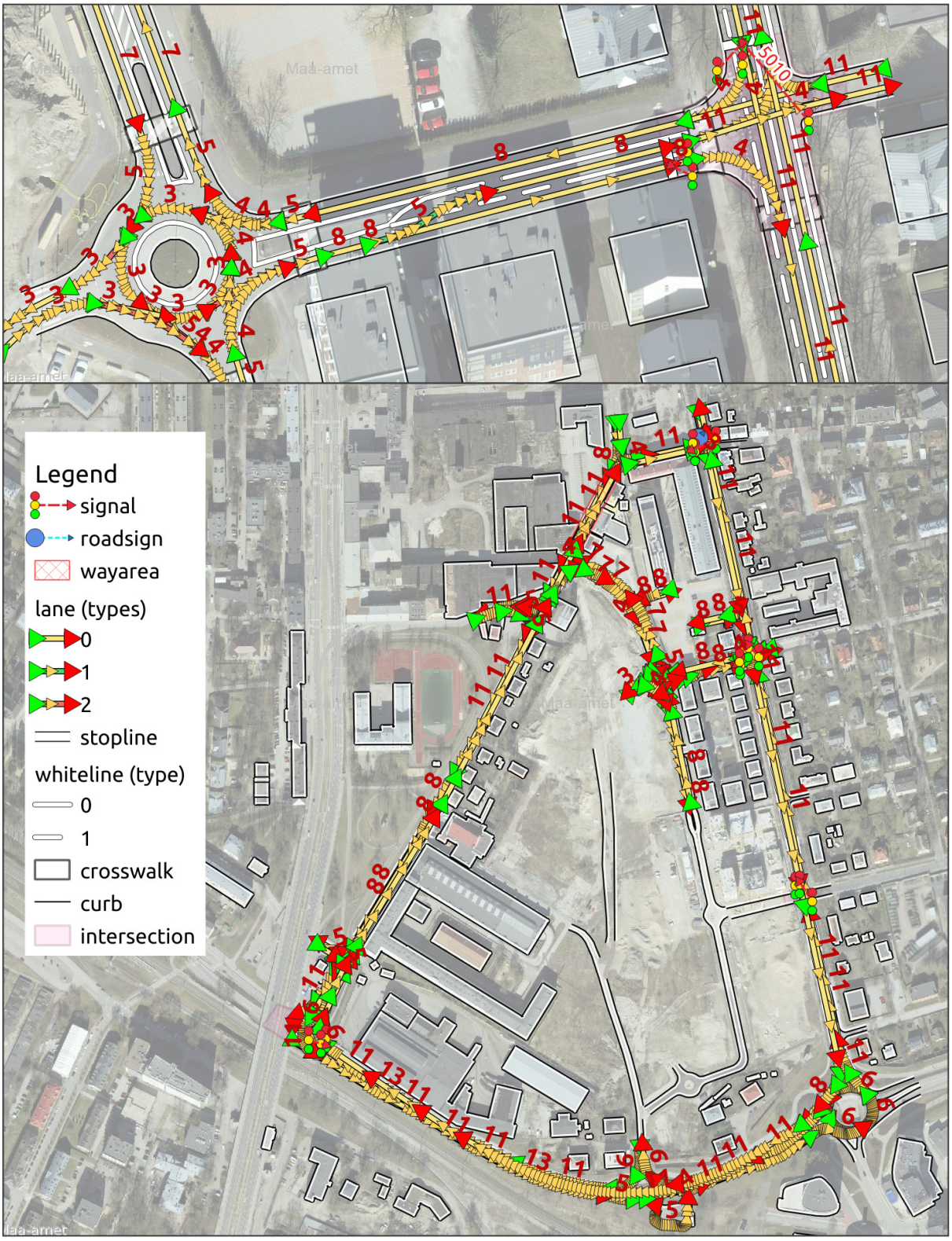

Figure 22. Tallinn demo route map, upper: central part of demo route Tiiu street, lower: Overview of all the Tallinn demo route. Background orthophoto from Estonian Land Board.

Maps converted to Autoware vector map are usually visualized using RViz[87] - that is 3D visualization tool for a Robot Operating system (ROS). Some examples of how the map is visualized in ROS can be seen from Figures 23 to 25.

Figure 23. The upper part of Tallinn demo route vector map in Rviz. On the right side of the image also a point cloud map made by EyeVi Technologies[88] is visualized.

[87] https://github.com/ros-visualization/rviz
[88] https://eyevi.tech/

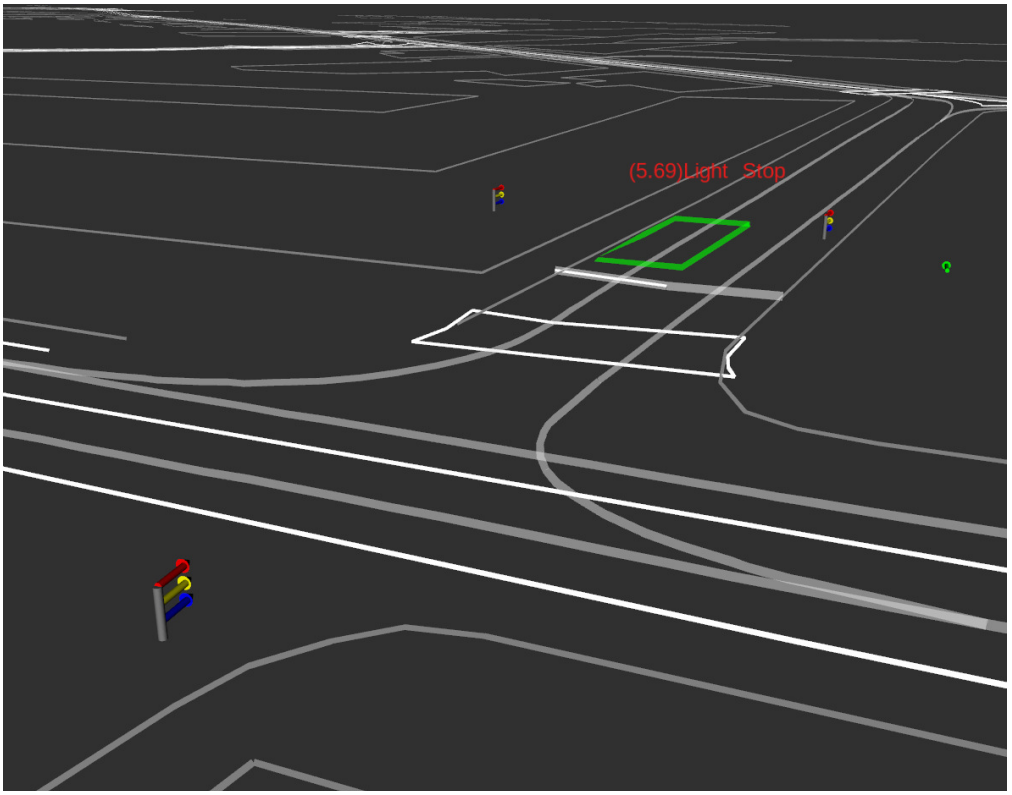

Figure 24. A traffic light can be seen on the bottom left corner represented by 1 vector for each lamp. The green rectangle shows the position of the ego-vehicle and the red text denotes its state.

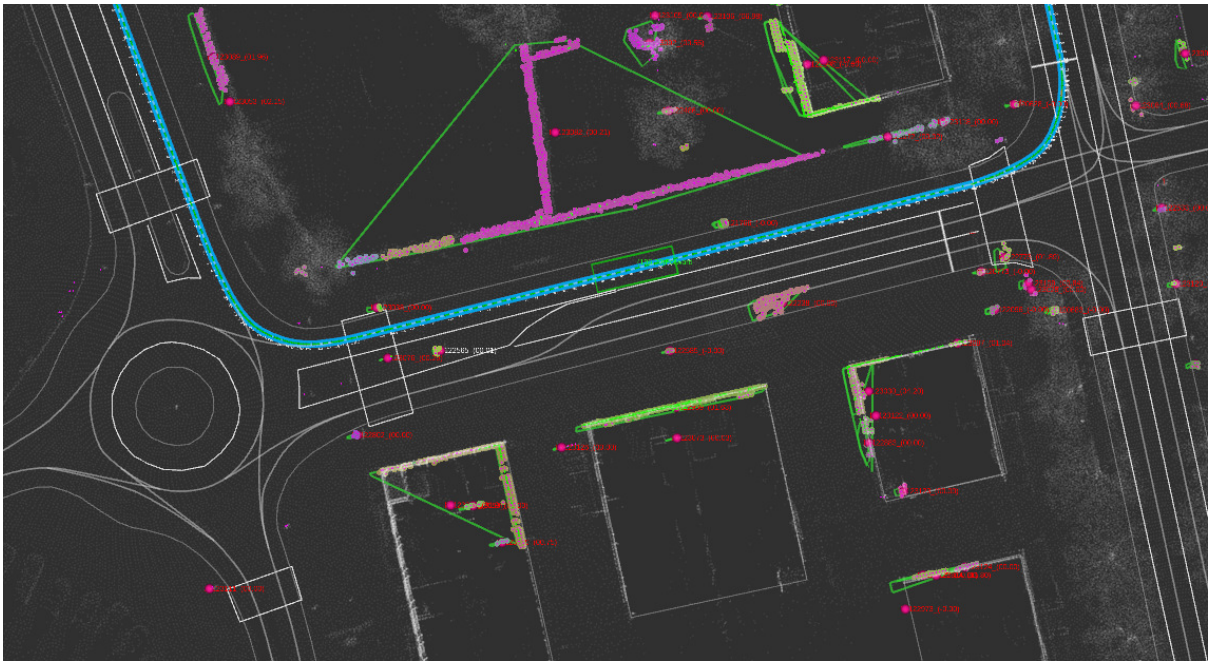

Figure 25. The actual situation during the demo drive where ego-vehicle (green rectangle) is approaching a small roundabout, other green lines are drawn around detected objects.

4.2. Quantitative measures

Table 7 compares some quantitative aspects of the created HD vector maps. It is difficult to estimate the time spent on doing the Tartu demo route map, because the workflow and digitizing concepts were also developed at the same time. It means that some parts of the map might have been partly remapped and techniques automating some tasks might have not been there yet. Also, the map was created in several stages adding features and complexity incrementally. Main adjustments after testing consist of improving the positioning of traffic lights, adjusting speeds, corrections to trajectories. Important to note is that not all the features in the area are mapped, but only the ones required to drive the designated route. The biggest difference is most probably with the traffic lights, in the case of pedestrian crossings, curbs and whitelines most of them should be there. For example, all the traffic lights when driving the same street but in the opposite direction (oncoming lane) are missing, mainly because they were not needed for our demo route.

Table 7. Quantitative measures of the HD vector maps created.

	Tartu demo route	Tallinn demo route
Time it took	h	h (47)
Data collection with car	not recorded	4
Map creation time (Initial map)	not recorded	35 (+/-5)
Adjustments after testing	not recorded	8
Total distance of	km	km
Lanes	30.5	7.1
Whitelines	26.1	4.9
Curbs (include buildings)	42.8	20.3
Total number of	count	count
Traffic signs and lights	101	15
Crosswalks	77	19
Trajectories of turning maneuvers	214	52
Points in point.csv	48411	12872
Additional measures		
Man hours per lane km (initial map creation)	hard to tell	5 h/km
Man hours per lane km (Final map)	hard to tell	6.6 h/km

During the period of 19/02/2020-27/01/2021, there are 715 recordings of different drives with ADL self-driving car (Table 8). Most of these drives have used Autoware vector maps created with the workflow developed in this thesis.

Table 8. A total number of recorded drives and autonomous drives in Tartu and Tallinn demo routes with UT ADL self-driving car.

	no	km
All recorded drives	715	873.9
Only autonomous drives	608	782.3
Autonomous drives in Tallinn demo route	39	55.4
Autonomous drives in Tartu demo route ● (exclude rosbags with keywords: raadi, lombi, sulaoja, tallinn, tll, tln) ● additionally exclude drives shorter than 1 km	405 269	585.3 535.9

Table 9 gives a more detailed comparison of the autonomous drives in the demo routes. One of the main measures used currently in AD is the number of disengagements per km. It is quite high in our case, but here we have to keep in mind that this number also includes disengagements encountered during development and testing, and also disengagements due to situations not in our ODD, for example giving way to pedestrians waiting at pedestrian crossings.

Table 9. Comparison of autonomous drives in Tallinn and Tartu demo routes.

	Tartu demo route	Tallinn demo route
Total number of drives	269	39
Total distance (km)	535.9	55.4
Total time (h)	24.16	2.62
Number of disengagements	931	143
Km per disengagement	0.58	0.39
Distance driven autonomously (%)	0.93	0.89
Time driven autonomously (%)	0.87	0.85

Estimates for mapping bigger areas

In the following section, the estimate of 6.6 h per lane km from the Tallinn demo route map is used to calculate the time needed to map the whole of Estonia, Tallinn and Tartu city. For calculations, all the road data from the Estonian topographic database is used with some of the road types excluded, see Table 10.

Table 10. The total amount of roads and roads selected to calculate map creation time estimates. Data from Estonian topographic database.

	km	Note	km
1st level highways	1907.4		1907.4
2nd level highways	2495.9		2495.9
3rd level highways	12593.3		12593.3
Other state highways	0.7		0.7
Ramps and connections	124.0		124.0
Other roads	63711.6	24749 km of dirt roads excluded	38962.6
Streets	7125.5		7125.5
Light traffic road	1851.9	excluded	0.0
Footpath	22472.1	excluded	0.0
Total	112282.4		63209.4

The same exclusion logic as in the Table 10 was also used for extracting roads for Tartu and Tallinn city areas (exclude light traffic roads, footpaths, and dirt roads from class other roads). Since the time estimate for map creation in Table 7 is in h per lane km then road kilometers were converted to lanes using a multiplier, see Table 11. As a result lane kilometers were acquired and used in the estimated time calculation.

Table 11. Length of roads in Estonia, Tallinn and Tartu and its conversion to lanes.

	Roads (km)			Lane conversion multiplier	Lanes (km)		
	Estonia	Tallinn	Tartu		Estonia	Tallinn	Tartu
1st level highways	1907.4	50.5	10.2	4	7629.6	202.0	40.9
2nd level highways	2495.9	4.6	9.5	3	7487.7	13.7	28.6
3rd level highways	12593.3	11.2	0.4	2	25186.6	22.4	0.7
Other state highways	0.7	0.0	0.0	2	1.4	0.0	0.0
Ramps and connections	124.0	0.7	1.4	2	248.0	1.4	2.8
Other roads	38962.6	136.6	11.0	2	77925.2	273.3	22.1
Streets	7125.5	1392.8	448.4	4	28502.0	5571.1	1793.5
Total	63209.4	1596.4	480.9		146980.5	6083.8	1888.5

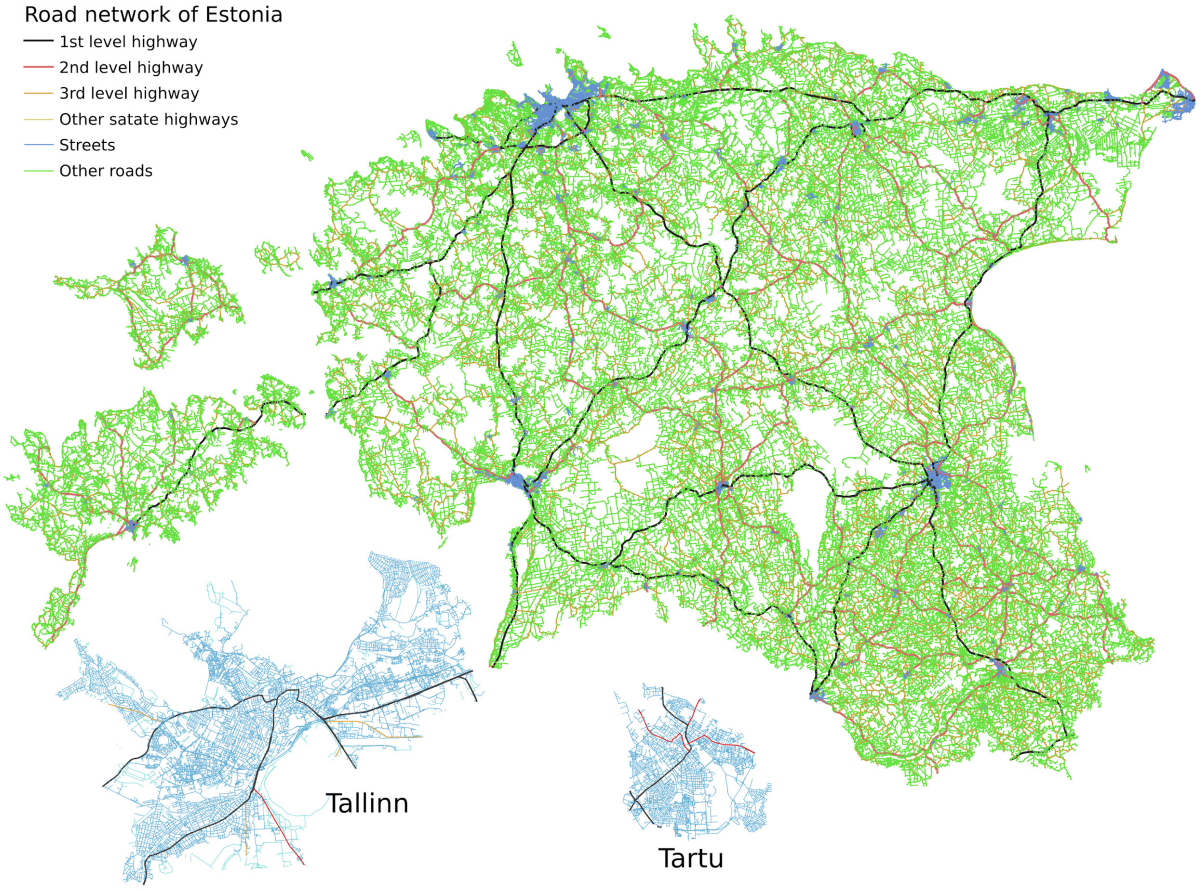

Figure 26. Road network of Estonia, Tallinn and Tartu cities. Road data from Estonian topographic database.

Following numbers are used to make the calculation for time estimates needed to map entire Estonia, Tallinn and Tartu city:

- Lane kilometers from Table 11;
- 2086 working hours per year for 1 person
- 6.6 h per lane km needed to create the final map

Table 12. Time estimates for mapping Estonia, Tallinn and Tartu city.

	Total hours needed (h)	Total person years
Estonia	970071.3	465.0
Tallinn	40153.3	19.2
Tartu	12464.1	6.0

5. Discussion

The developed workflow of creating HD vector maps for Autoware.ai has been used now to create several maps for UT ADL self-driving car. The workflow (digitizing maps according to specified reality and data model and converter) has proven its ability to fulfill the needs specified in ODD of UT ADL. On the other hand, it is just the start when considering the map making for AD. Current maps are covering quite small areas and most of the map creation process has been manual. The main challenges for wider adoption are how to scale up the approach so that bigger areas with shorter times could be covered with the HD vector map. It will never be the goal for UT ADL to map entire Estonia or enter the mapping business, but conducting research in this area definitely is. Also, the AD experiments of UT ADL might grow more ambitious (broaden the scope of ODD) and extend into other areas that might need new feature types and mapping workflow would need to be updated. Mapping will definitely be one of the cornerstones and enabling factors for autonomous driving. The key to making maps faster and cover bigger areas lies in the automation of data collection and processing so that less effort would be needed for map creation.

Technical issues

All the reviewed tools for creating the Autoware vector map were based on digitizing the features in 3D view on top of the point cloud or existing 3D scene. It was quickly realized that drawing in 2D was much faster, especially when elevation is added later using a digital elevation model. Controlling the 3D view and the xyz when digitizing, especially with large point clouds is quite inconvenient. Point clouds are very useful in local scales acquiring the details like determining traffic light height above the road or attached to the pole on the roadside. Reviewed tools did not have much support for other data sources like orthophotos that were freely available from Estonian Land Board and proved to be very useful for digitizing the lanes. Orthophotos are much easier to understand and work with than point cloud.

If there is no prior knowledge and data about the environment it might be a good idea to make a point cloud and draw a map based on that so they would match perfectly. Then the collected point cloud could be also used as a source for localization and an HD vector map for controlling the planning logic. If the area gets bigger or different data sources are used to compose a map then some common reference system for aligning the data would be needed. The problem with point clouds is the geometrical instability on the global scale. They provide very good relative accuracy, but when GNSS-IMU or SLAM-based localization fails during the data collection or starts to drift then these errors must be corrected during point cloud map assembly, but these are not trivial tasks. So if map data is digitized on top of point-cloud it might be hard to align it later with data from other sources and vice versa. Aligning the assembled point cloud with the vector map is again not a trivial task and might need quite demanding processing and potentially expensive software.

When the first HD maps were created and tested an interesting problem with Rviz (visualization node for ROS and Autoware) and large coordinates of the L-EST97 system

was discovered. Coordinates in front of the Delta building in Tartu have the following values: X=6474917.36, Y=659341.25. HD vector map with these coordinates was not displayed correctly in Rviz, after some investigation it was found that Rviz uses float32 type for coordinates[89] and X value simply does not fit into that. Smaller coordinates are suggested by selecting the origin closer to the operating point. To solve this additional shift was applied to coordinates, so the x and y values would be smaller.

In geodesy, the spatial reference systems use different conventions naming the axis. It is the same also with the L-EST97 coordinate reference system (CRS) where the North pointing axis is called x and the East pointing axis is called y. The axis naming is usually reversed in robotics and needs to be taken into account when converting maps from HD vector map format to Autoware vector maps.

Another map-related problem that made us do some reconfiguration of the original localization node was the heading angle. The heading angle is very important for the low-level controller to know in which direction the self-driving vehicle is currently driving. The heading angle is measured from the North in a clockwise direction and it needs to be aligned with the heading angle of the CRS used. In our case, the L-EST97 system is used, but the heading angle provided by the GNSS system uses WGS84. The heading angle of these systems is aligned on the central meridian used by the CRS system (location of the origin point) but starts to deviate when moving away from it. It was corrected by adding the location-based calculation that would correct the deviation between the heading angles so that the heading angle from the GNSS would match the CRS system heading angle.

Autoware vector map format is not publicly available and documentation is lacking. That made creating the conversion tool a challenge. Traffic lights provide a good example. A self-driving vehicle would need to know that there is a traffic light ahead on its path, but how is this information retrieved? There is an option to link traffic lights with LinkID to a lane (lane ID will be added to traffic light LinkID value). Additionally, there is an option to add traffic light Id to stopline.csv and the stopline will be linked with the lane. The question raised here is, which link is used? No official documentation to clarify it, so the only reliable way is to actually go through the source code of Autoware and search for the implementation. Several such problems were raised during the converter creation making it hard to understand if the error is from the converter side or misunderstanding about the Autoware vector map format or how its use is implemented at the code level.

Time for map creation

It took 6.6 h per lane km to create an HD vector map for the Tallinn demo route. It is quite time-consuming. The main reason for that is the manual digitization of the HD vector map features. Since the length and area where the demo routes lie were quite small the manual drawing was an acceptable way of creating the map.

The developed workflow relies on orthophotos and the collected trajectories. During the first tests when trajectories were recorded and plotted on top of the orthophotos a very good

[89] https://github.com/ros-visualization/rviz/issues/1091#issuecomment-298695876

locational match was seen. Estonia has very good coverage with orthophotos (from 2017 yearly coverage of urban areas with 10 cm ground resolution) that are free for use, so they provide excellent material for drawing the lanes. Current manual digitization could be replaced with some automatic processing (not trivial and needs to be developed) that could give a very good win in terms of time spent drawing the lanes. Recorded trajectories would still be needed for velocity profiles.

Another approach with lanes could be automating lane generation from trajectories. Currently, with only one car it would not be a feasible solution, because it would take quite a lot of time to drive through all the possible trajectories. To make the trajectories more reliable and eliminate some traffic situations (overtaking standing car) several trajectories from different times (minimize GNSS errors) and weather conditions (effects to GNSS and difference of summertime vs wintertime trajectories) should be used. With one car it is not an option, but definitely a feasible solution with a car fleet.

Most probably a combination of these will be used. For example, the following sequence of actions: extract lanes from orthophotos, validate correctness with recorded trajectories and update speed profiles, train a model to derive a speed profile for lanes based on curvature and traffic situations and update lanes with speed profiles not with recorded trajectories. When these pipelines are set up for lanes and other features it would considerably decrease the time needed to make an HD vector map. The current example with the Tallinn demo route shows that only 8.5% of the time was data collection with the car and most of the time 91.5% was spent manually digitizing and correcting the data. In the future, these percentages should be reversed, mainly by decreasing the amount spent on the map digitizing and using automated workflows instead.

UT ADL has no ambition to become a mapping company, so most probably it will not be achieved there, but definitely, various automation possibilities will be discussed, researched and also implemented.

Optimizing the HD vector map creation

In this section different options to make the mapping procedure more automated and faster will be suggested:

- Optimizing data collection using a car. Currently, only trajectories with speed profiles, point cloud and camera images were used. Some processing was used to visualize the data more conveniently, so it would be more helpful in the map creation process, but no automatic feature extraction was used. It is possible to develop a specific workflow for data collection or even some simple added procedures might make the later mapping easier:
    - add recording the blinker status and attach it as an attribute to lanes - automatic blinker status mapping
    - adding additional recorded features or create nodes with additional functionality to help in mapping tasks - approach towards MMS car setup

- Processing collected data - automate the workflows (currently very few mapping related processing are automated):
    - automatic lane generation based on trajectories
    - speed profile generation from trajectories or using the curvature of the trajectory
    - traffic light mapping - automatic point cloud map generation and object recognition
    - traffic lights, stop signs and other signs - point cloud and camera-based methods. Standard heights could be used as initial input.
    - stoplines - camera and trajectory data along with speed could be helpful;
- Use data from additional sources:
    - curbs, road markings/whitelines, stop lines could be extracted from orthophotos and point clouds using some machine learning pipelines;
    - lanes also from orthophotos and by adding trajectories of the car we get an additional understanding of systematic errors;
    - lane width (LW, RW) calculation from orthophotos.
- Fusing different data sources to make it more precise or easily extractable:
    - trajectories collected with car and trajectories extracted from orthophotos;
    - lane width from orthophoto and car camera or (point cloud).
- Preparing/converting the map to HD vector map
    - optimizing some requirements in the data model: simplify drawing (draw spaghetti lane model and use topology to correct it during the conversion with added validation scripts
    - smoothing trajectories - partly automated - can be made more user friendly and faster
    - some features might not be necessarily extracted at all. like lane width (LW, RW), it might be possible to automate its calculation from other collected map data - central trajectory, curbs and whitelines.
    - checking the topology rules - currently mostly only routing perspective, but also other elements: crosswalks, signs, traffic lights
- Export the map
    - maybe some of the previous things can be automated in this step
    - more validation functionality can be added to spot mistakes earlier.
- Migrate HD vector map to PostGIS database
    - will add options from more sophisticated topology rules that can be used in validation and creating relations between elements
    - Option for multiple users editing the same data
    - QGIS can still be used as graphical UI for data browsing and editing

Future plans

Autoware vector map format is supported by Autoware.ai, but the development of it is more or less halted[90] and the efforts are directed towards the new version called Autoware.auto. The main map format for Autoware.auto is Lanelet2. That is definitely making us look into options of converting our HD vector map also to other formats. This might bring some additional requirements for data collection or reorganize how we currently store the mapped data, meaning that the data model and the reality model need to be adjusted accordingly.

Our current HD vector map is quite minimalistic, considering all the semantic information and landmark features that could be included. When comparing it to Lyft's understanding of HD vector maps[91] it almost does not have any semantic information and almost misses the map priors layer (we have only speed profiles) and some information about real-time data (traffic lights in Tartu demo route are accessed over web API). Semantic data and map priors layer could be really useful to make cars' behaviour more sophisticated when making decisions about the behaviour. It might mean that in the future when expanding the operational design domain of the UT ADL platform new features are necessary and the reality model and data model need to be adjusted accordingly.

As a research institution, UT ADL is not interested to start with large-scale mapping and HD vector mapping production. Mainly we are interested in building competence and then focusing on some real-life problems in mapping and its automation that could be solved by developing new methodologies. To support other research fields in UT ADL also making maps richer in sense of features and added semantic information could be another direction.

Once the HD vector map of the area is created it is not actually finished. Changes happening in the environment might quite quickly make it out of date, so the change detection and remapping methods are also very important tasks that need to be addressed. As introduced in the text there are quite many crowdsourcing solutions used in this field and when pairing this with a fleet of cars the potential of crowdsourcing becomes much more real. Another closely related research question is also change detection and its integration into the mapping pipeline. The change could be detected between two mapping drives or identified by comparing the car's perception and HD map.

[90] https://discourse.ros.org/t/end-of-life-dates-for-autoware-ai/13750

[91] https://medium.com/lyftself-driving/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6

6. Conclusion

The general motivation behind this thesis was to help the University of Tartu Autonomous Driving Lab's (UT ADL) self-driving platform on the streets to conduct real-life autonomous driving experiments. High-definition maps (HD maps) play an important part in this by enabling global and local planning functionality and providing a common reference frame for fusing map features with outputs from perception. A more specific task for this thesis was to review the available datasets and tools to create HD vector maps and develop a suitable workflow for UT ADL.

Requirements of the UT ADL regards to HD vector map were defined with respect to the operational design domain (ODD), hardware and software framework used in the self-driving platform. Available data sources were reviewed and their suitability for HD vector map creation was analysed. There was no data with lane-level information available. It could be potentially acquired, but with sophisticated processing with applied heuristics. Only the Autoware vector map feature layer named wayarea could be easily acquired from the Estonian topographic database provided by the Estonian Landboard.

All the tools reviewed were limited in one way or another. The best tool offered for Autoware vector map (supported most of the features) creation is web-based Vector Map Builder, but it lacks convenience and has accessibility issues. All the tools started to have performance issues when the maps got bigger and the lack of support for other vector formats and coordinate reference systems made integrating data from different sources difficult. One of the most used formats for HD maps - Navigation Data Standard - is not free and not supported by the software framework used in the UT ADL platform.

As a result, a custom workflow for mapping was created based on QGIS software (freeware and open source geographic information system) and a data conversion tool was developed. Map data was digitized and stored as described in the newly defined reality model and data model. The conversion tool was then used to convert map data into Autoware vector map format. Creating an intermediate map dataset gives us a lot of flexibility to include different data sources and develop additional tools to convert the HD map into other formats or add automating tools for data collection, processing or validation. HD vector map formats used in autonomous driving systems are optimized for the efficiency in the car (access the features in code level) and lack visualization options to make the maps easily understandable for humans. The intermediate map layer is specifically oriented for human use and supports easy map creation and editing.

Currently, only small areas have been mapped and almost no automation in creating the map features was used. It took 6.6 h per lane km to create a Tallinn demo route map (total of 47 h). From a future perspective in the context of HD vector map creation, the automation of different procedures is crucial. With current mapping speeds, it would take 465 years to map the whole of Estonia (estimated of 147000 km of lanes) for 1 person and 6 years for Tartu city (1888 km of lanes). It is hard to speculate how much faster the mapping can be, but as brought out in the text the Baidu Apollo has automated 90% of the mapping pipeline, so there is a lot of potential to make it faster.

The current workflow was used to drive autonomously 39 times in the Tallinn demo route and in a total of 55.4 km and 89% of that was in autonomous mode. The number of times and the amount in kilometers are much higher for the Tartu demo route: 269 autonomous drives with 536 km and 93% of it in autonomous mode. These numbers confirm that the developed workflow and HD vector maps created have proven their ability to support UT ADL and its self-driving platform to conduct autonomous driving on the streets.

7. Acknowledgement

First and foremost, I would like to express my thanks to my supervisor and employers for giving me the most valuable support possible - the time! I'm very grateful to the supervisor also for the support and discussions throughout writing the thesis.

I can't stress enough the importance of the family who has suffered without complaints during this time and instead of making it harder given me the motivation to do it. I thank you all for that!

Also, I want to thank my Mother! I know it meant a lot to you and thank you for reminding me that. It really helped to set the focus.

This work was funded by a cooperation project between the University of Tartu and Bolt "Applied Research on Development of Autonomous Driving Lab for Level 4 Autonomy".

8. References

[1] T. J. Crayton and B. M. Meier, "Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy," *J. Transp. Health*, vol. 6, pp. 245–252, Sep. 2017, doi: 10.1016/j.jth.2017.04.004.

[2] W. D. Montgomery, R. Mudge, E. L. Groshen, S. Helper, J. P. MacDuffie, and C. Carson, "America's Workforce and the Self-Driving Future: Realizing Productivity Gains and Spurring Economic Growth," Securing America's Future Energy, Washington, DC, USA, Technical Report, 2018.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A Survey of Autonomous Driving: Common Practices and Emerging Technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020, doi: 10.1109/ACCESS.2020.2983149.

[4] R. Liu, J. Wang, and B. Zhang, "High Definition Map for Automated Driving: Overview and Analysis," *J. Navig.*, vol. 73, no. 2, pp. 324–341, Mar. 2020, doi: 10.1017/S0373463319000638.

[5] W. Kühn, M. Müller, and T. Höppner, "Road Data as Prior Knowledge for Highly Automated Driving," *Transp. Res. Procedia*, vol. 27, pp. 222–229, Jan. 2017, doi: 10.1016/j.trpro.2017.12.011.

[6] C. Ress, A. Etemad, D. Kuck, and J. Requejo, "Electronic Horizon - Providing Digital Map Data for ADAS Applications," 2016, doi: 10.5220/0001508000400049.

[7] C. Kim, S. Cho, M. Sunwoo, and K. Jo, "Crowd-Sourced Mapping of New Feature Layer for High-Definition Map," *Sensors*, vol. 18, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/s18124172.

[8] SAE, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," 2016.

[9] A. J. Benson, B. C. Tefft, A. M. Svancara, and W. J. Horrey, "Potential Reductions in Crashes, Injuries, and Deaths from Large-Scale Deployment of Advanced Driver Assistance Systems (Research Brief)," AAA Foundation for Traffic Safety, Washington, D.C., 2018.

[10] R. McAllister *et al.*, "Concrete Problems for Autonomous Vehicle Safety: Advantages of Bayesian Deep Learning," pp. 4745–4753, 2017.

[11] L. Chi and Y. Mu, "Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues," *ArXiv170803798 Cs*, Aug. 2017, Accessed: Feb. 04, 2021. [Online]. Available: http://arxiv.org/abs/1708.03798.

[12] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A Survey of End-to-End Driving: Architectures and Training Methods," *IEEE Trans. Neural Netw. Learn. Syst.*, pp. 1–21, 2020, doi: 10.1109/TNNLS.2020.3043505.

[13] F. Poggenhans *et al.*, "Lanelet2: A high-definition map framework for the future of automated driving," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2018, pp. 1672–1679, doi: 10.1109/ITSC.2018.8569929.

[14] C. Fernández, D. Fernández-Llorca, and M. A. Sotelo, "A Hybrid Vision-Map Method for Urban Road Detection," *Journal of Advanced Transportation*, Oct. 30, 2017. https://www.hindawi.com/journals/jat/2017/7090549/ (accessed Feb. 04, 2021).

[15] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A Survey of the State-of-the-Art Localization Techniques and Their Potentials for Autonomous Vehicle Applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018, doi: 10.1109/JIOT.2018.2812300.

[16] P. Hubertus, M. Schleicher, F. Klebert, G. Horn, and M. Junker, "The Benefits of a Common Map Data Standard for Autonomous Driving," Navigation Data standard,

White Paper, 2019. Accessed: Feb. 17, 2021. [Online]. Available: https://nds-association.org/wp-content/uploads/2019/06/NDS-White-Paper__Benefits-of-Map-Data-Standard-for-Autonomous-Driving.pdf.
[17] J. Ziegler *et al.*, "Making Bertha Drive—An Autonomous Journey on a Historic Route," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 2, pp. 8–20, Summer 2014, doi: 10.1109/MITS.2014.2306552.
[18] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, Jun. 2014, pp. 420–425, doi: 10.1109/IVS.2014.6856487.
[19] A. Chen, A. Ramanandan, and J. A. Farrell, "High-precision lane-level road map building for vehicle navigation," in *IEEE/ION Position, Location and Navigation Symposium*, May 2010, pp. 1035–1042, doi: 10.1109/PLANS.2010.5507331.
[20] A. Schindler, G. Maier, and F. Janda, "Generation of high precision digital maps using circular arc splines," in *2012 IEEE Intelligent Vehicles Symposium*, Jun. 2012, pp. 246–251, doi: 10.1109/IVS.2012.6232124.
[21] T. Ort, L. Paull, and D. Rus, "Autonomous Vehicle Navigation in Rural Environments Without Detailed Prior Maps," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2040–2047, doi: 10.1109/ICRA.2018.8460519.
[22] W.-C. Ma *et al.*, "Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization," *ArXiv190803274 Cs*, Aug. 2019, Accessed: Nov. 18, 2019. [Online]. Available: http://arxiv.org/abs/1908.03274.
[23] R. Zekavat and R. M. Buehrer, "Localization for Autonomous Driving," in *Handbook of Position Location: Theory, Practice, and Advances*, IEEE, 2019, pp. 1051–1087.
[24] J. M. Kang, T. S. Yoon, E. Kim, and J. B. Park, "Lane-Level Map-Matching Method for Vehicle Localization Using GPS and Camera on a High-Definition Map," *Sensors*, vol. 20, no. 8, Art. no. 8, Jan. 2020, doi: 10.3390/s20082166.
[25] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Trans. Intell. Veh.*, vol. 2, no. 3, pp. 194–220, Sep. 2017, doi: 10.1109/TIV.2017.2749181.
[26] Z. Xiao, D. Yang, T. Wen, K. Jiang, and R. Yan, "Monocular Localization with Vector HD Map (MLVHM): A Low-Cost Method for Commercial IVs," *Sensors*, vol. 20, no. 7, Art. no. 7, Jan. 2020, doi: 10.3390/s20071870.
[27] J. Levinson, M. Montemerlo, and S. Thrun, *Map-Based Precision Vehicle Localization in Urban Environments*. 2007.
[28] A. Ranganathan, D. Ilstrup, and T. Wu, "Light-weight localization for vehicles using road markings," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 921–927, doi: 10.1109/IROS.2013.6696460.
[29] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, Illustrated edition. Cambridge, Mass: The MIT Press, 2005.
[30] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor Fusion-Based Low-Cost Vehicle Localization System for Complex Urban Environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1078–1086, May 2017, doi: 10.1109/TITS.2016.2595618.
[31] L. Wang, Y. Zhang, and J. Wang, "Map-Based Localization Method for Autonomous Vehicles Using 3D-LIDAR **This work is supported in part by the National Natural Science Foundation of China under Grant No. 61473209.," *IFAC-Pap.*, vol. 50, no. 1, pp. 276–281, Jul. 2017, doi: 10.1016/j.ifacol.2017.08.046.
[32] N. Akai, L. Y. Morales, E. Takeuchi, Y. Yoshihara, and Y. Ninomiya, "Robust localization using 3D NDT scan matching with experimentally determined uncertainty and road marker matching," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1356–1363, doi: 10.1109/IVS.2017.7995900.

[33] M. Elhousni and X. Huang, “A Survey on 3D LiDAR Localization for Autonomous Vehicles,” *ArXiv200600648 Cs*, Nov. 2020, Accessed: Feb. 04, 2021. [Online]. Available: http://arxiv.org/abs/2006.00648.
[34] S. Hecker, D. Dai, and L. Van Gool, “End-to-End Learning of Driving Models with Surround-View Cameras and Route Planners,” *ArXiv180310158 Cs*, Aug. 2018, Accessed: Feb. 04, 2021. [Online]. Available: http://arxiv.org/abs/1803.10158.
[35] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman, “Distraction suppression for vision-based pose estimation at city scales,” in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 3762–3769, doi: 10.1109/ICRA.2013.6631106.
[36] H. Jürgenson, “Eesti kaardisüsteemi matemaatiline alus,” *Geodeet*, vol. 8, no. 32, 1995.
[37] M. Mizutani, M. Tsukada, Y. Iida, and H. Esaki, “3D maps distribution of self-driving vehicles using roadside edges,” Okinawa, Japan, Nov. 2020, Accessed: Feb. 23, 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02997520.
[38] I. Puente, H. González-Jorge, J. Martínez-Sánchez, and P. Arias, “Review of mobile mapping and surveying technologies,” *Measurement*, vol. 46, no. 7, pp. 2127–2145, Aug. 2013, doi: 10.1016/j.measurement.2013.03.006.
[39] M. Elhousni, Y. Lyu, Z. Zhang, and X. Huang, “Automatic Building and Labeling of HD Maps with Deep Learning,” *ArXiv200600644 Cs*, May 2020, Accessed: Feb. 04, 2021. [Online]. Available: http://arxiv.org/abs/2006.00644.
[40] J. Jiao, “Machine Learning Assisted High-Definition Map Creation,” in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, Jul. 2018, vol. 01, pp. 367–373, doi: 10.1109/COMPSAC.2018.00058.
[41] M. Schreiber, C. Knöppel, and U. Franke, “LaneLoc: Lane marking based localization using highly accurate maps,” in *2013 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2013, pp. 449–454, doi: 10.1109/IVS.2013.6629509.
[42] N. Homayounfar, J. Liang, W. Ma, J. Fan, X. Wu, and R. Urtasun, “DAGMapper: Learning to Map by Discovering Lane Topology,” *2019 IEEECVF Int. Conf. Comput. Vis. ICCV*, 2019, doi: 10.1109/ICCV.2019.00300.
[43] J. Lee *et al.*, “Automated Algorithm for Removing Clutter Objects in MMS Point Cloud for 3D Road Mapping,” *Sensors*, vol. 20, no. 15, Art. no. 15, Jan. 2020, doi: 10.3390/s20154076.
[44] D. Paz, H. Zhang, Q. Li, H. Xiang, and H. Christensen, “Probabilistic Semantic Mapping for Urban Autonomous Driving Applications,” *ArXiv200604894 Cs*, Sep. 2020, Accessed: Feb. 01, 2021. [Online]. Available: http://arxiv.org/abs/2006.04894.
[45] L. Tang, T. Deng, and C. Ren, “Semantic Information Extraction of Lanes Based on Onboard Camera Videos,” *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 42.3, pp. 1643–1649, Apr. 2018, doi: 10.5194/isprs-archives-XLII-3-1643-2018.
[46] J. Zhao, X. He, J. Li, T. Feng, C. Ye, and L. Xiong, “Automatic Vector-Based Road Structure Mapping Using Multibeam LiDAR,” *Remote Sens.*, vol. 11, no. 14, Art. no. 14, Jan. 2019, doi: 10.3390/rs11141726.
[47] C. Ertler, J. Mislej, T. Ollmann, L. Porzi, G. Neuhold, and Y. Kuang, “The Mapillary Traffic Sign Dataset for Detection and Classification on a Global Scale,” in *Computer Vision – ECCV 2020*, Cham, 2020, pp. 68–84, doi: 10.1007/978-3-030-58592-1_5.
[48] C. W. Gran, “HD-Maps in Autonomous Driving,” NTNU, 2019.
[49] B. Parajuli, P. Kumar, T. Mukherjee, E. Pasiliao, and S. Jambawalikar, “Fusion of aerial lidar and images for road segmentation with deep CNN,” in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, New York, NY, USA, Nov. 2018, pp. 548–551, doi: 10.1145/3274895.3274993.

[50] B. Parajuli, A. R. Choudhury, and P. Kumar, “Reconstructing Road Network Graphs from both Aerial Lidar and Images,” 2020, pp. 1737–1746, Accessed: Feb. 01, 2021. [Online]. Available: https://openaccess.thecvf.com/content_WACV_2020/html/Parajuli_Reconstructing_Road_Network_Graphs_from_both_Aerial_Lidar_and_Images_WACV_2020_paper.html.
[51] P. Fischer, S. M. Azimi, R. Roschlaub, and T. Krauß, “Towards HD Maps from Aerial Imagery: Robust Lane Marking Segmentation Using Country-Scale Imagery,” *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 12, Art. no. 12, Dec. 2018, doi: 10.3390/ijgi7120458.
[52] W. N. Tun, S. Kim, J. Lee, and H. Darweesh, “Open-Source Tool of Vector Map for Path Planning in Autoware Autonomous Driving Software,” in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Feb. 2019, pp. 1–3, doi: 10.1109/BIGCOMP.2019.8679340.
[53] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada, “An Open Approach to Autonomous Vehicles,” *IEEE Micro*, vol. 35, no. 6, pp. 60–68, Nov. 2015, doi: 10.1109/MM.2015.133.
[54] H. Darweesh *et al.*, “Open Source Integrated Planner for Autonomous Navigation in Highly Dynamic Environments,” *J. Robot. Mechatron.*, vol. 29, no. 4, pp. 668–684, Aug. 2017, doi: 10.20965/jrm.2017.p0668.
[55] M. Javanmardi, E. Javanmardi, Y. Gu, and S. Kamijo, “Towards High-Definition 3D Urban Mapping: Road Feature-Based Registration of Mobile Mapping Systems and Aerial Imagery,” *Remote Sens.*, vol. 9, no. 10, Art. no. 10, Oct. 2017, doi: 10.3390/rs9100975.
[56] J. Anderson, D. Sarkar, and L. Palen, “Corporate Editors in the Evolving Landscape of OpenStreetMap,” *ISPRS Int. J. Geo-Inf.*, vol. 8, no. 5, Art. no. 5, May 2019, doi: 10.3390/ijgi8050232.
[57] J. Anderson and D. Sarkar, “Curious Cases of Corporations in OpenStreetMap,” Jul. 04, 2020, doi: 10.5281/zenodo.3923050.
[58] M. Haklay, “How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets,” *Environ. Plan. B Plan. Des.*, vol. 37, no. 4, pp. 682–703, Aug. 2010, doi: 10.1068/b35097.
[59] C. Barron, P. Neis, and A. Zipf, “A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis,” *Trans. GIS*, vol. 18, no. 6, pp. 877–895, 2014, doi: https://doi.org/10.1111/tgis.12073.

Appendix

I. Glossary

AD - Autonomous driving

ADAS - Advanced driver assistance systems

ADL - Autonomous Driving Lab

ALS - Aerial laser scanning

CRS - Coordinate reference systems

DEM - Digital elevation model

DR - Dead reckoning

ECEF - earth centered earth fixed

ELB - Estonian Land Board

FOV - Field of view

GNSS - Global navigation satellite system

HAD maps - Highly autonomous driving maps

HD map - High-definition map

IMU - Inertial measurement unit

MGRS - Military grid reference system

MMS - Mobile mapping system

NDS - Navigation Data Standard

NDT - Normal distribution transform

OEM - Original equipment manufacturer

ODD - Operational design domain

OSM - OpenStreetMap

ROS - Robot operating system

RTK - Real-time kinematic

SAE - Society of Automotive Engineers

SD maps - Standard-definition maps

SLAM - Simultaneous localization and mapping

UTM - Universal Transverse Mercator

WMS - Web map service

II. Reality model for HD vector map

Attribute	type	Unit / color	Explanation
Lane *the line representing the lane, trajectory that cars will follow in normal driving conditions.*			
LW	double	meters	Left width of the lane
RW	double	meters	Right width of the lane
LimitVel	int	m/s	Maximum velocity
RefVel	int	m/s	Suggested velocity
Speed	int	m/s	Speed taken from speed profile
LaneType	int		0 - straight, 1 - left turn, 2 - right turn
ClossId	int		Id of the intersection
Whiteline *road markings that separate lanes, mark the roadsides and safety areas on the roads*			
Width	double	cm	width of the wihteline
Color	string		W - white, Y - yellow
type	boolean		0 - continuous, 1 - dashed whiteline
Roadsign *stop sign at the roadside*			
Id	int		Id of the road sign
Hang	double	degrees	Horizontal angle, clockwise from North
Vang	double	degrees	Vertical angle, 0 - facing up, 180 - down
Height	double	m	Height from ground level to sign center
Type	int		denotes behaviour for stop sign 0 - don't stop; 1 - stop and go; 2 - stop and wait for go command
Signal (Traffic lights) *traffic light consists of up to 3 individual lamps of different color (2 - green, 3 - yellow, 1 - red)*			
Id	int		Id of the traffic light

group_id	int		Id of the traffic light group (traffic lights for the same maneuver)
Hang	double	degrees	Horizontal angle, clockwise from North
Vang	double	degrees	Vertical angle, 0 - facing up, 180 - down
Lights	string		Order of traffic light colors, default “1,3,2”
Heights	string	meters	Heights for individual traffic lights from ground to center of individual lights, default: “2.4,2.6,2.8”
Stopline *the location where the car should stop if the traffic light is red or giving the way to other cars in an intersection or to pedestrians crossing the crosswalks*			
Id	int		Stop line id
Curb *an edging built along a street to form part of a gutter*			
Height	double	cm	Height of the curb
Width	double	cm	Width of the curb
building	boolean		0 - simple curb, 1 - denotes the building
Crosswalk *the area marked for pedestrians crossing a street or road*			
Id	int		id of the crosswalk
Type	int		crosswalk subtype (pedestrian, bicycle)
BdID	int		Id of the crosswalk area border
LinkID	int		Link to crosswalk id that formes one group (border)
Intersection *area where two or more streets intersect*			
ClossId	int		Id of the intersection
Wayarea *driveable area for the cars*			
Id	int	id	Id of the wayarea

III. Data model for HD vector map

Geometry	Requirements for digitizing	
Lane		
linestring	direction	must follow the driving direction
	startpoint endpoint	must be snapped to connected lane start and endpoints to form a topologically correct network that is usable also for routing
	middle points	must form a smooth trajectory that is suitable for the car to follow
Whiteline		
linestring	direction	not important
	startpoint endpoint	should be snapped with following or preceding whitelines
	middle points	should follow the digitized features
Roadsign and **Signal** (Traffic lights)		
line	startpoint	location of the object in xy plane (z is taken from attributes)
	endpoint	location of stopline it is linked to - snapped to central point of stopline
Stopline		
linestring	direction	not important
	startpoint endpoint	visually in correct location
	middle point	only 1 middle point is allowed and it must be snapped to a lane endpoint
Curb		
linestring		no particular guidelines for digitizing; should follow real world features
Crosswalk		
polygon		should be digitized to cover the whole area of the feature

		no specific requirements for digitizing
Intersection		
polygon		the intersection area is usually aligned with stoplines before the intersection covering also crosswalk areas.
		no specific requirements for digitizing
Wayarea		
polygon		the wayarea includes all the drivable area for the cars and also areas where pedestrians can walk; it can be used to filter objects
		no specific requirements for digitizing

IV. Autoware vector map tables, its' fields and filling rules

point.csv		
Field	Explanation	Filled with
PID	Point ID	Generated automatically and incremented when elements added
B	Latitude, WGS84	Not used
L	Longitude, WGS84	Not used
H	Height / elevation	Elevation of the point (above sea level)
Bx	Easting	y coordinate from point (in L-EST97 easting is y coordinate)
Ly	Northing	x coordinate from point (in L-EST97 northing is x coordinate)
ReF		Not used
MCODE1		Not used
MCODE2		Not used
MCODE3		Not used

node.csv		
Field	Explanation	Filled with
NID	Node ID	Equal to PID
PID	Point ID	PID from the point.csv

dtlane.csv describes mainly geometric attributes of the lane		
Field	Explanation	Filled with
DID	dtlane ID	Automatically generated
Dist	Distance of the lane	1.0 - lanes are 1m long sections
PID	Point ID	respective point ID (beginning of lane) is used
Dir		Not used
Apara		Not used
r		Not used
slope		Not used
cant		Not used
LW	Left width from centerline	LW from lane.shp layer attributes
RW	Right width from centerline	RW from lane.shp layer attributes

lane.csv topology and routing related attributes for lanes		
Field	Explanation	Filled with
LnID	Lane ID	Automatically generated
DID	dtlane ID	linked with respective dtlane ID
BLID	Before Lane ID	previous incoming lane ID
FLID	Forward Lane ID	following lane ID
BNID	Before Node ID	Node ID at the beginning of the lane
FNID	Forward Node ID	Node ID at the end of the lane
JCT	branching pattern type	not used
BLID2, BLID3, BLID4	Before lane IDs (merging)	if multiple lanes branch or merge into the lane the respective ID's are filled in these fields.
FLID2, FLID3, FLID4	Forward lane IDs (branching)	
ClossID	Intersection ID	respective intersection ID from clossId in lane.shp
Span	Accumulating distance between lane merging and branching points	Accumulated distance is calculated
LCnt	Number of lanes in total for same direction	Not used
Lno	Lane number	Not used
LaneType	center / left / right	Based on our custom modification LaneType is used to indicate blinkers
LimitVel	maximum allowed velocity	LimitVel from lane.shp layer attributes
RefVel	suggested velocity	RefVel from lane.shp layer attributes
RoadSecID	inside intersection is 0, adjacent lanes have same roadsection ID	Not used
LaneChgFG	flag if lane change is allowed	Not used

pole.csv		
Field	Explanation	Filled with
PLID	Pole ID	Automatically generated and incremented
VID	Vector ID	Automatically generated and incremented
Length	Length	constant used
Dim	Dimension	constant used

vector.csv		
Field	Explanation	Filled with
VID	Vector ID	Automatically generated and incremented
PID	Point ID	link to respective point ID
Hang	Horizontal angle	Hang from signal data layer
Vang	Vertical angle	Vang from signal data layer

signaldata.csv		
Field	Explanation	Filled with
ID	signaldata ID (each lamp separately)	signal ID is used with lamp color code concatenated to it
VID	Vector ID	link to respective point ID
PLID	Pole ID	Link to correct pole ID
Type	Lamp color	Import lamp color from signal.shp Lights field
LinkID	reference to Lane ID	correct lane ID is extracted using signal.shp line endpoint location

roadsign.csv		
Field	Explanation	Filled with
ID	signaldata ID (each lamp separately)	signal ID is used with lam color code concatenated to it
VID	Vector ID	link to respective point ID
PLID	Pole ID	Not used
Type	roadsign behavior	value from roadsign layer Type attribute
LinkID	reference to Lane ID	correct lane ID is extracted using roadsign data line endpoint location

line.csv		
Field	Explanation	Filled with
LID	Line ID	Automatically generated
BPID	Before (start) point ID	Link with correct PID
FPID	Forward (end) point ID	Link with correct PID
BLID	Before line ID (0 if none)	link with the previous line
FLID	Forward line ID (0 if none)	link with the following line

curb.csv		
Field	Explanation	Filled with
ID	Curb ID	Automatically generated and incremented
LID	Line ID	Link to correct line ID
Height	Height in cm	use Height value from curb layer
Width	Width in cm	use Width value from curb layer
DIR		Not used
LinkID	reference to lane ID	Not used

whiteline.csv		
Field	Explanation	Filled with
ID	Curb ID	Automatically generated
LID	Line ID	correct line ID
Width	Width in cm	use whiteline Width
Color	Color	use whiteline Color
Type	Type - dashed, continuous	use whiteline Type
LinkId	reference to lane ID	Not used

stopline.csv		
Field	Explanation	Filled with
ID	Curb ID	Automatically generated
LID	Line ID	correct line ID
TLID	Traffic light ID	Use location (stopline middle point) to retrieve trafficlight ID, stopsign ID and lane ID
SignID	Stopsign ID	
LinkId	reference to lane ID	

area.csv		
Field	Explanation	Filled with
AID	area ID	Automatically generated
SLID	Start line ID	ID of the first line of the area
ELID	End line ID	ID of the last line of the area

crosswalk.csv		
Field	Explanation	Filled with
ID	crosswalk ID	taken from crosswalk layer Id field
AID	area ID	link to the correct area
Type	0-border, 1-striped pattern, 2-bicycle pass	Not used
BdID	border ID	Not used
LinkID	reference to lane ID	link to lane using location from respective area's first line endpoint

wayarea.csv		
Field	Explanation	Filled with
WAID	wayarea ID	taken from wayarea layer Id field
AID	area ID	link to the correct area

intersection.csv		
Field	Explanation	Filled with
ID	intersection ID	taken from intersection layer ClossId field
AID	area ID	link to the correct area
LinkID	reference to lane ID	Not used

V. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Edgar Sepp**,

(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Creating High-Definition Vector Maps for Autonomous Driving,

   (title of thesis)

   supervised by Tambet Matiisen.

   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Edgar Sepp

11/03/2020