UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Dmytro Zabolotnii

Automatic Road Boundaries Extraction for High Definition maps

Master's Thesis (30 ECTS)

Supervisor: Naveed Muhammad, PhD
Supervisor: Dmytro Fishman, MSc

Tartu 2021

Automatic Road Boundaries Extraction for High Definition maps

Abstract:
Autonomous Vehicles (AV) research moves forward and promises to create a safer and more efficient driving process than the vast majority of humans can achieve. However, just as humans, Autonomous Vehicles still rely on the maps of their surroundings to conduct most of their operational sub-tasks. These maps are enriched with a large quantity of additional information for a more accurate representation of the natural world, earning the common name of High Definition (HD) Map. The rapid increase of the field's popularity also brings a great deal of attention to the HD maps creation and maintenance. Still, to this day, almost all HD maps are created using many human hours of expert labor, raising their cost and creating barriers to broader adoption. In this work, we research recent advancements of HD maps automatic creation and apply novel methods to extract road information, namely road boundaries. We strive to create an automatic system capable of extracting the necessary information from LIDAR data from vehicles deployed in urban conditions, with a high degree of accuracy and tolerance to externalities, such as weather conditions or road construction details. In order to evaluate the system, we use the publicly available Nuscenes dataset and compare automatically created road boundaries with provided manually drafted ground truth. The system achieves a precision score of 0.62 and a recall score of 0.31 at the distance tolerance of 40 cm.

Keywords:
autonomous driving, high definition maps, computer vision, point cloud's processing

CERCS:
T111: Imaging, Image Processing; P176: Artificial Intelligence

Automaatne tee servade leidmine täppiskaartide jaoks.

Lühikokkuvõte:
Isejuhtivate sõidukite uurimisvaldkond areneb edasi ja tõotab luua turvalisema ning efektiivsema sõidu protsessi kui suurem osa inimesi saavutada suudaksid. Sarnaselt inimestele sõltuvad ka isejuhtivad sõidukid jätkuvalt oma lähiümbrust kirjeldavatest kaartidest opereerimiseks vajalike ülesannete lahendamisel. Nendesse kaartidesse on lisatud palju informatsiooni kirjeldamaks ümbritsevat maailma ja neid kutsutakse täppiskaartideks. Antud valdkonna populaarsuse kiire kasv on toonud palju tähelepanu ka täppsikaartide loomisele ja haldamisele. Siiski on tänapäevalgi vaja täppiskaartide tegemiseks palju inimtunde tööd, mis tõstavad nende hinda ja tekitavad barjääre laiemaks kasutuselevõtuks. Käesolevas uurimistöös uuritakse valdkonna viimaseid saavutusi täppiskaartide automaatsel loomisel ja rakendatakse uudseid meetodeid tee info, antud juhul siis teeservade, hankimiseks. Käesolevas töös loodi automaatne süsteem, mis oleks võimeline selle info hankima autodele paigaldatud LIDARiga linnakeskkonnas kogutud andmetest väga kõrge täpsusega ja oleks seejuures tolerantne ekstreemsete oludele suhtes nagu erinevad ilmastiku tingimused ja tee ehitus. Süsteemi hindamiseks kasutatakse avalikult kättesaadavaid Nuscenes andmeid ning võrreldakse nende põhjal automaatselt loodud tee servasid andmetega kaasas olevate käsitsi joonistatud tee servadega. Väljatöötatud süsteemi täpsus lubatud kaugustolerantsi 40 cm juures on 0.62 ja saagis 0.31.

Võtmesõnad:
autonoomne juhtimine, kõrglahutusega kaardid, arvutinägemine, punktpilve töötlemine

CERCS:
T111: Pilditehnika; P176: Tehisintellekt

Contents

1 Introduction

1.1 Introduction

Contrary to the popular idea, Autonomous Driving (AD) is a pretty old scientific field with the first research attempts started in the 1920s [Sen26], and the first autonomous vehicles tested in the 1980s, in the well-known Navlab [THKS88] or Autonomous Land Driven Vehicle (ALV) [KTW86]. Nevertheless, even one hundred years later, we are only reaching the point of introducing autonomous vehicles in limited settings - long-track hauling [SS18], or predefined route taxis [Lit17]. Like nuclear fusion, it proverbially remains "always ten years away," which shows the extreme difficulty of the task and perhaps overzealous optimism of the researchers. Unlike nuclear fusion, AD research is available to a much wider selection of research groups and has exploded in popularity in the last ten years, both in the minds of the general public and prospective scientists. However, the question remains: what is the difference between ALV, modern Tesla, or Waymo autonomous vehicle, and hypothetical perfect self-driving car of the future? The answer is the degree of autonomy.

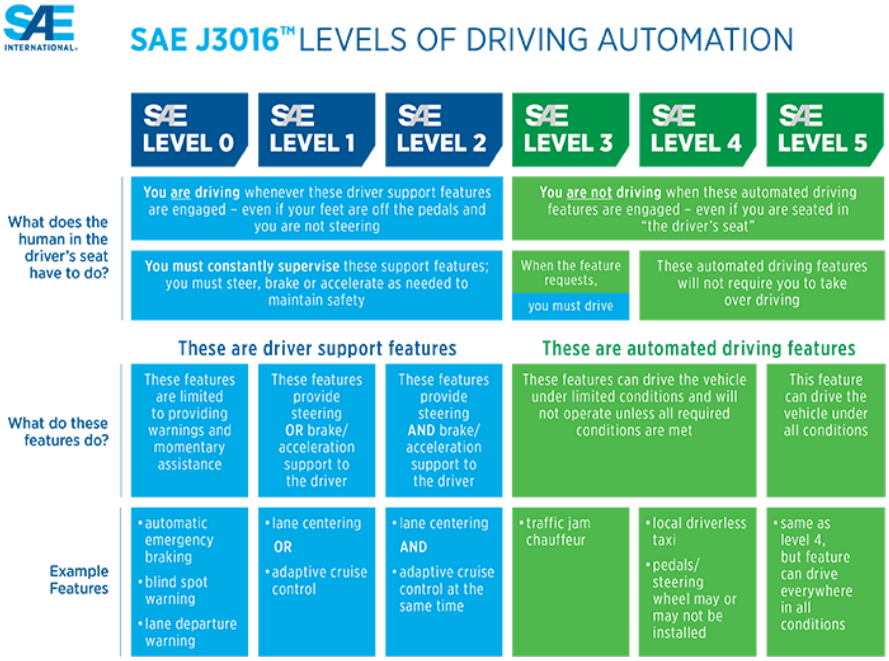

Figure 1. SAE introduced levels of automation [SAE16]

There are many approaches to measuring the degree of autonomy, with the most popular being Level 0 (fully manual) to Level 5 (fully autonomous), introduced by The Society of Automotive Engineers (SAE) [SAE16]. Classifying existing autonomous

vehicle solutions can be non-trivial, but even most optimistic researchers would not call even the best available systems to be beyond Level 4 (fully autonomous under certain restrictions) [TK17]. Still, some industry leaders believe in the full implementation of Level 5 autonomy of 2030 [Ske18], while others are much less optimistic and believe there are still many challenges: technological, societal, and legal. [LZH+18].

Regardless of the predictions, there are many approaches to reach Level 5 autonomy and are usually divided into two broad categories. The first category is the classical modular approach, borrowing from traditional robotics science. Every sub-task of Autonomous Driving: sensing, perception, planning, control is governed by a semi-independent module, based on the classical robotic algorithms or machine-learning-based solution [LTZG17]. A second category is an end-to-end approach that usually uses a complex neural network to process data from sensors and directly outputs car control values. While being quite different on the surface, both approaches try to mimic human driver behavior, either by directly training on human driver data or breaking the process of the driving into simpler sub-tasks and solving them with efficient approaches.

As such, it is expected that autonomous vehicles rely on map data of their immediate surroundings for navigation, just as humans relied on the maps through most of the written history. Due to the complexity of the self-driving sub-tasks, such as localization and decision-making, the necessary maps are usually much more information-dense than the normal navigational ones. Such maps, named High-Definition (HD) maps (or High-Precision maps), often expand in physical dimension (to include data in 3D space) and information dimension (to include qualitative data, such as traffic symbols meaning), or even in temporal dimension (to include traffic light schedule or predicted weather conditions), and include all the contemporary map information at centimeter-level precision. The creation and utilization of HD maps is one of the key challenges for autonomous driving research [SH16]. Due to the overall complexity and sheer amount of raw data that needs to be processed, existing HD maps are created offline semi-manually using a massive amount of expensive human-expert labor. While there is some recently published research on the automatic creation of the HD maps' elements [Ma20], the topic remains relatively obscure and not under the spotlight from the general autonomous vehicle research community.

The task of generating an entire HD map is too broad for a single research paper, so we had to limit it to a generation of a single HD-map element. After additional research, we selected the road boundary: the border separating drivable and non-drivable areas for Autonomous Vehicles. Despite being necessary information for many tasks in Autonomous Driving, it is still a non-trivial task to extract road boundaries from immediate sensor information, especially in urban settings. This is caused by wide variation in the possible ways of defining non-drivable area border: all possible shapes of curbs (see Figure 3), different colored markings, just no visible demarcation at all. Additionally, in urban settings, it is common for the non-drivable area to be present

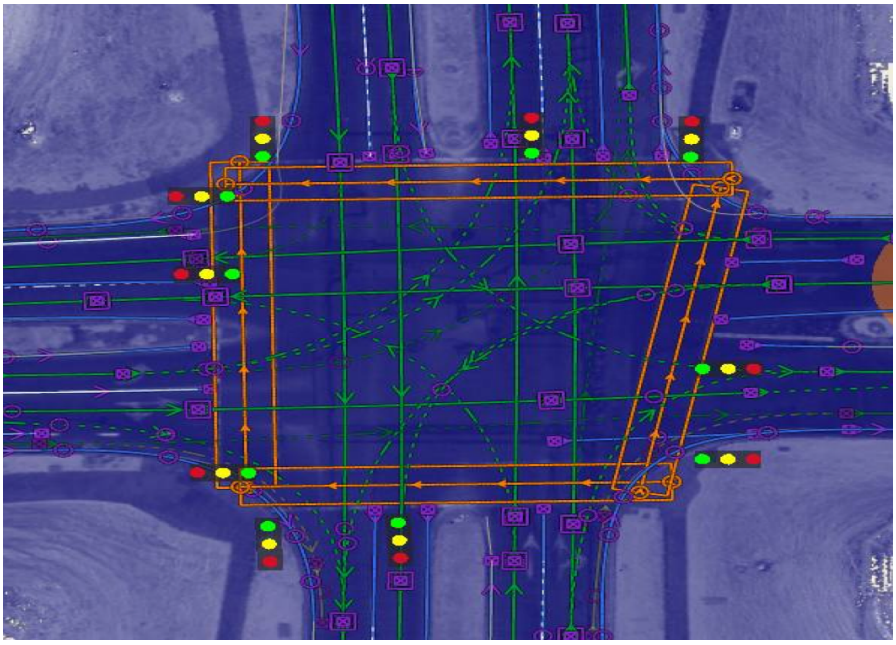

Figure 2. Showcase HD map demonstrating additional encoded information by Waymo [Way17]

inside the road itself, in the form of different decorations, pedestrian safe zones, or entire pedestrian-only boulevards. As such, having this information prerecorded in the HD map as viable geometry that only needs to be projected on the sensor/vehicle frame of view is much more preferable.

1.2 Contributions

Our contributions are three-fold. Firstly, we create a comprehensive pipeline converting one of the existing Autonomous Driving perception datasets - Nuscenes [CBL+20] into one of the only publicly available datasets suitable for further HD map-related research and our current problem. Secondly, we adopt an existing solution to the automatic generation of road boundaries of highways [LHM+19] and test its performance in a complex urban setting. Thirdly, we propose a simple heuristic-based method allowing us to complete gaps in our road boundaries predictions and use it together with other morphological operations to refine raster predictions from the neural network from the previous step into one-pixel wide skeletons that can be easily converted to vector format.

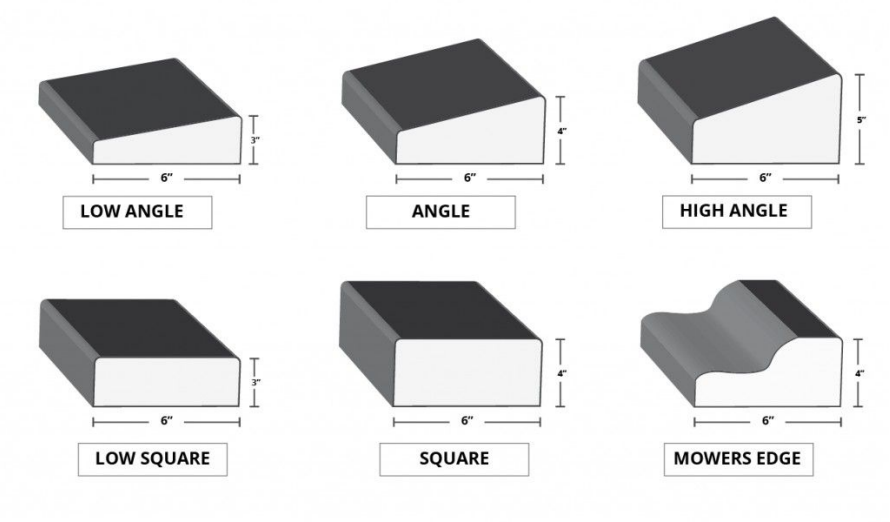

Figure 3. Different types of the curbs [Dis20]

1.3 Structure of the thesis

The thesis is comprised of 5 chapters beyond this Introduction chapter. Related works chapter will cover the most recent works in the autonomous generation of different HD maps elements, specifically road boundaries. Data collection and pre-processing chapter will extensively cover existing Autonomous Driving datasets and their possible usage for HD map research and describe in detail all stages of the pre-processing pipeline of the Nuscenes dataset. Stage 1 chapter will cover a deep neural network-based solution that regresses from recorded LIDAR sensor data to raster representation of road boundaries, while Stage 2 chapter will cover refinement of raster data to final form. Finally, Conclusions chapter will outline the general conclusion of the paper, summarize the results and provide vectors for future research work.

2 Related Works

2.1 Automatic mapping solutions using aerial photography

Automatic creation of the maps' elements such as transportation routes is the task that was researched long before the current resurgence of the interest in Autonomous Vehicles. Starting from the 70s [BT76], satellite photography data has been used for the specific task of identifying roads and integrating generated data into Geographic Information Systems (GIS). Early approaches worked almost exclusively with aerial or satellite image data. The common theme of those approaches was to rely on using classical Computer Vision algorithms such as edge detection for rough initial detection of pixels belonging to the road and specifically developed algorithms for road following that allow to filter out pixels that do not belong to the road afterward [AFZAW99]. Road following algorithms is broken into several categories according to [AFZAW99], such as edge trackers, pattern matched based trackers, and region-based trackers. Most of these algorithms are built upon several extrinsic assumptions that are found in the physical constructions of the roads, for example, that roads boundaries are parallel to each other and width between them is constant, their curvature is not extreme, or that the pixel intensity is relatively uniform inside road as compared to outside. It is easy to see that these assumptions are not accurate for all the different possible road structures. To compensate for this, the combinations of the different road following algorithms are used [MJD88]. Finally, generated and cleaned up road segments are usually incomplete and do not cover the entire road structure, so different completion methods based on geometric properties were introduced to fill in gaps [MLBS97] [SMR97]. These early solutions did not target or reached the level of accuracy necessary for HD maps and usage of their generated map data in Autonomous Vehicles but still laid a valuable groundwork for later solutions. The example of the prediction by one of the algorithms is shown in the Figure 4.

The advent of neural networks and their usage across the computer vision field created many new approaches to the road boundaries detection problem. [MLU17] combines the convolutional neural network with graph theory to map road topology, while [MH10] combines supervised and unsupervised methods in order to segment the road itself. Alternatively, conditional random field-based methods [WMZS15] [MZWLS14] were applied to extract and identify pixels with a high likelihood to be roads, and topology is extracted afterward. These methods work with high-quality aerial images in urban conditions and achieve significant metric improvement over previous attempts while still not reaching the level of human experts. However, even high-quality aerial maps have pixel values ranging .25-1.5 meters, requiring sub-pixel precision to successfully integrate the extracted information in HD maps, which these and analogous methods do not achieve. The example of the prediction by one of the algorithms is shown in the Figure 5.

Figure 4. Example of partial road boundary extraction from aerial photo imagery in rural setting from Mayer et.al. [MLBS97]

2.2 Automatic mapping solutions using LIDAR point clouds

Sequentially, low precision can be resolved with the switch out of aerial photography as the main source of the primary data. The main candidates for such a switch are camera imagery and, more recently, LIDAR sensor data. Camera-based approaches were prevalent in the early 2000s and continue to find usage today; however, the data source still has many restrictions, such as the lack of depth information, the lack of 360-degree coverage, and limited coverage of obstacles. LIDAR sensor data using Mobile Laser Systems (MLS), on the other hand, has no such restrictions and reaches a degree of density over 10000 points on square meter, which is impossible for airborne approaches to reproduce [CWL$^+$19]. If installed on the car, the MLS system comprehensively covers 360-degree space around the car up to an effective range of the sensor (usually 50-60 meters, with some newer models claiming up to 400 meters visibility [Lit21]). If the vehicle is traveling across a specific route, combinations of all LIDAR scans taken at 30-40 Hz covers the entirety of the road and most of the relevant objects of interest nearby, allowing to theoretically extract any information: road boundaries, markings, line demarcations, signs, traffic lights, etc. relevant to the creation of the HD map,

Figure 5. Road topology extracted by Mattyus et.al. [MLU17]. Green denote segmented pixels belonging to road, red - extracted road center line, blue - road completion segments generated by A* algorithm

with extremely high precision. In fact, this is how most HD maps are created in some commercial companies such as Waymo, Here, or TomTom: semi-automatic extraction of relevant information from LIDAR data and combined with normal digital maps and satellite imagery data.

Still, automatic extraction of the roads' interesting features from LIDAR data is a challenging task that has been researched only recently. One of the reasons is that commercial sensors were costly until the late 2010s limiting the potential researchers lacking necessary funds. This also leads to most high-quality and quantity LIDAR datasets being private, even if used in public research. Another reason is that while LIDAR data is precise and dense in theory for normal road conditions, in practice, it also has several detriments [Ma20]. Firstly, it is massive and unstructured, limiting classical approaches to analysis, object recognition, and segmentation used for imagery data, requiring usage and development of the novel methods. Secondly, point density is not uniform in both point density and intensity values, requiring advanced normalization and interpolation. Finally, some road information is just not caught in all different scans due to immovable obstacles such as the parked vehicles, and removing dynamic objects is a challenge.

Despite that, there is a number of recent works using LIDAR data as a primary or an

auxiliary data source for the automatic extraction of various road elements. The most notable works for the road boundaries extractions are listed below. Homayounfar et al. [HMLU18] use a hierarchical recurrent neural network to estimate road boundaries directly from sensor feed in real-time. Real-time processing suffers from obstacles and is not entirely suitable for creating an exact map. Bai et al. [BMH+18] fuses LIDAR and camera data and processes them inside CNN for the efficient and precise lane predictions from which boundaries can be extracted naively. Liang et al. [LHM+19] present a comprehensive offline solution of extracting road boundaries from the combined LIDAR scans using CNN and refinement of them to vector form using RNN. Achieving precise results in the rural setting (achieving over 0.9 accuracy metric, with accuracy threshold of 5 cm) and being relatively lax to input data requirements (requiring only LIDAR data, without auxiliary camera or aerial imagery), it was the primary inspiration for this master thesis. Most recently, Ma et al. [Ma20] presents a very complex solution for offline road boundary generation, combining several convolutional deep neural networks with the generative adversarial network reaching state-of-art in road boundary extraction and completion problem. However, the sheer complexity of the solution and the unavailability of the used dataset (that also requires high-quality satellite data as auxiliary) made it impossible to reproduce or expand the said solution.

3 Data collection and pre-processing

3.1 Datasets

As we have established in the previous section, almost all recent Road Boundary extraction solutions heavily rely on Machine Learning. Since its inception, the backbone of Machine Learning was the massive amount of data that is often collected and labeled separately. The quality and quantity of the dataset often influence the final result more than the quality of the solution itself. Additionally, it is vital to have reliable data organization and management, as even small mistakes between different versions can lead to profound divergence in the course of research [SSSM13].

Fortunately, a massive surge of publicly available datasets was established in the last decade, with the most well-known earliest example of Kitti dataset [GLSU13]. These datasets were created to help prospective researchers solve problems directly in the Autonomous Driving field, such as the agent movement prediction or scene flow, or adjacent tasks in the Computer Vision field, such as scene segmentation, object classification, etc. Competitions between the solutions to these tasks hosted by the authors of such datasets have attracted numerous new researchers and remain important events and stimulation of the Autonomous Driving scientific community. All data is usually recorded from sensors mounted on the moving vehicle during some pre-defined routes. Earlier datasets, such as Kitti, consist primarily of the camera sensor recorded raw data. However, with the broader acceptance of the LIDAR technology in the industry and the increase of public funding and interest, all recent Autonomous Driving focused datasets also include LIDAR and sometimes RADAR data. As the previously reviewed research focused on the autonomous generation of HD maps' elements mainly uses the camera, LIDAR, airplane/satellite photos data (or a combination of three), extensive research was done to find a suitable dataset for our purpose. Unfortunately, while there is a plethora of datasets with a massive amount of raw input data that can be easily pre-processed to our needs (i.e., additional information about segmented dynamic objects that can be removed easily), there are not that many that have pre-labeled output data, namely road boundaries geometry.

More specifically, for a dataset to be suitable for the developed solution, it should meet the following three pre-conditions:

1. It must include pure LIDAR data that can be cross-referenced with GPS or local coordinates at every frame, and have calibration information/transformation matrix for sensors, to project data into global map view;

2. It must include annotations of all the dynamic objects, i.e., pedestrians/cars, so that the LIDAR point cloud can be effectively cleaned. While the separate pre-processing solution for segmentation can be used (for example, Point

Pillars[LVC+19]), it introduces another layer of error and complexity into the pre-processing pipeline, and there are enough datasets that satisfy this condition;

3. It must include information on the road boundaries encoded in raster or preferably vector form. The geometries must be aligned with point clouds coordinates without error and be highly accurate to real-world road boundaries. While it is always possible to extract road boundaries from the OSM or Google Maps, the degree of the accuracy will be nowhere near current research results, and that is before training our model on this data, which will further decrease the accuracy of the output.

Keeping in mind these three conditions, we reviewed all publically available big-size datasets created for Autonomous Driving research of the last decade. Summary of the nine datasets of interest is available in Table 1. The table includes information about the following categories: scenes amount (scenes are defined as an independent collection of the sensor data during the car trip), amount of camera-generated RGB images, amount of LIDAR generated Point Clouds, amount of annotated frames (frames with annotated objects of interest, either via bounding boxes or segmentation of point cloud), whether there is any underlying map information located in the dataset, which of defined conditions dataset fulfilled (partial fulfillment is denoted with *), year and location of the dataset. All of the data is extracted from cited sources and cross-referenced with a summary table from [CBL+20], with some information missing/hard to estimate from sources explanation.

As can be seen, from the analyzed datasets, only three of them support all three conditions at least partially. This is likely caused by HD maps being under-focused in general Autonomous Vehicle research or extreme labor-intensiveness of manually labeling even well-organized data. Additionally, there are some non-public datasets not mentioned in the table, such as the datasets used in [Ma20] and [LHM+19]. Information on their quantity and quality is limited, and the authors of the relevant research did not answer our inquiry on whether they plan to release data to the community. Combining all these factors shows that there is much work to be done for the availability of the data for autonomous HD map generation, and hopefully, the situation will change in the following years.

Among the three datasets supporting all conditions, Lyft L5 and Argoverse support condition number three (encoded HD map) only partially. In both cases, the HD map information is encoded in the raster image: a single image covering all represented layers in Lyft L5 case and a separate image for the driving area layer in Argoverse case. For the Lyft L5, different HD map layers represented in one raster image sometimes overlap each other, which makes processing and isolating the road boundary layer extremely difficult. While the situation is better with Argoverse, both datasets are relatively small in size compared to Nuscenes, which completely satisfies all three conditions and is much larger, leading to choosing Nuscenes as the only dataset of choice for the paper.

Table 1. Summary table of the recent Autonomous Driving datasets

Dataset	**Scenes**	**Images**	**PCs**	**Ann. Frames**	**Map**	**Conditions**	**Year**
Nuscenes [CBL+20]	1K	1.4M	400K	40K	Yes	1+2+3	2019
Lyft L5 [KUH+19]	366	323K	43K	46K	Yes	1+2+3*	2019
Argoverse [CLS+19]	113	490k	44k	22k	Yes	1+2+3*	2019
Kitti [GLSU13]	22	15K	15K	15K	No	1+2	2012
ApolloScape [HCG+18]	-	144K	0	144K	No	1*+2	2012
Oxford [MPLN17]	100+	11M	3M	0	No	1	2014
H3D [PMGC19]	160	83K	27K	27K	No	1+2	2019
Waymo [SKD+20]	1K	1M	200K	200K	No	1+2	2019
Audi [GKM+20]	-	41K	12K	12K	No	1+2	2020

Nuscenes [CBL+20] is one of the recent Autonomous Driving focused datasets released in 2019. It is one of the biggest datasets by the sheer size and demonstrates remarkable quality - boasting over 1000 scenes with interesting urban scenarios, traffic situations, and wild animal activities. Dataset is still primarily focused on classical segmentation and agent prediction task, introducing a much more extensive variety of label categories - up to 23 classes - than its competitors. However, unlike almost all other datasets, it includes a fully vectored HD map with high precision and deep layers - going from zoomed out the drivable area to specific pedestrian crossing and lane segments areas. Figure 6 shows an example of a small part of HD maps encoded in Nuscenes, with all 11 layers. With the data collection locations being primary urban settings, it makes a natural choice for our research thesis. Unfortunately, on closer inspection, the dataset quality is not always suitable for our task, and after manual filtering out of bad samples, we are left with a dataset half of its size - a little over 500 scenes. A more detailed description of the pre-processing pipeline of the Nuscenes dataset will be described in the following section.

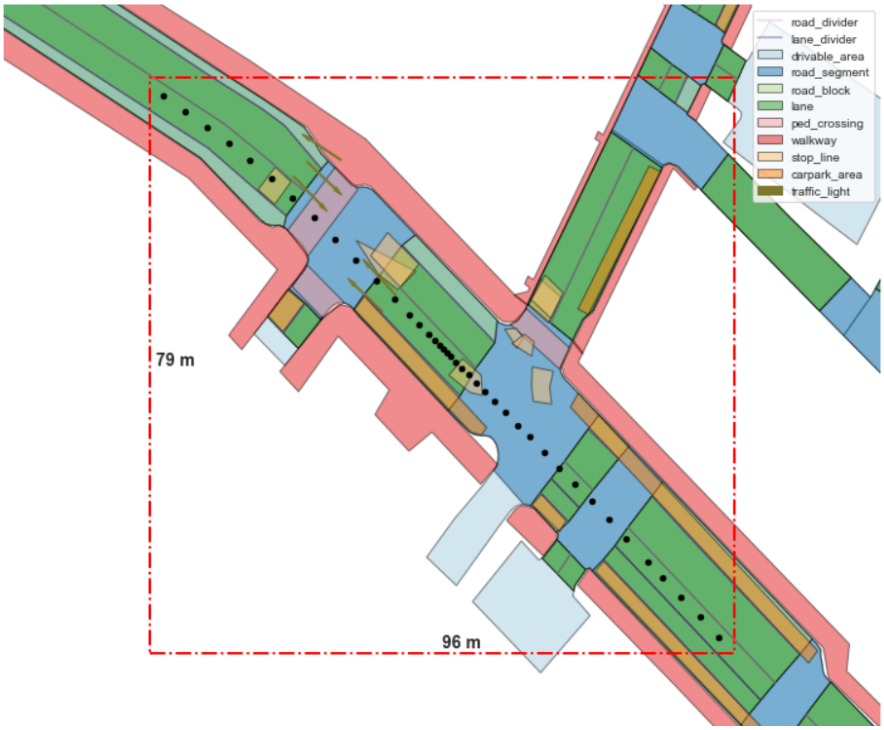

Figure 6. Nuscenes HD map projection to Bird-Eye-View with all available layers

3.2 Pre-processing pipeline

Nuscenes database is organized using pseudo-SQL principles and provides the top-down schema that allows easy and organic access to the information. The schema is presented in Figure 7 and covers all of the data. The database itself is not implemented in any SQL or non-SQL format, and instead, authors provide Python-based API that allows access of raw data through simple primary/foreign keys (called tokens in the context of Nuscenes) based GET operation.

The scene is the largest granular object in the Nuscenes database. One scene denotes 20-30 seconds of continuous sensor data on the physical level, following a car moving a specific trajectory. In some cases, the car movement is interrupted or slowed for periods of the track because of traffic rules, and in rare cases, the vehicle is standing still during the whole scene. As such, trajectories vary in length, which leads to variable areas covered by vehicle sensors, with sizes ranging from 0.01 to 0.025 square kilometers. Contemporary works on HD-mapping seek to have sub-10 cm accuracy [LHM+19], meaning that one pixel should cover less than 5 pixels. With the assumption that one pixel represents 4cm by 4cm, one Nuscenes scene area presented from Birds-Eye-View will be the size of several million pixels. This is a suitable size for CNN-based processing

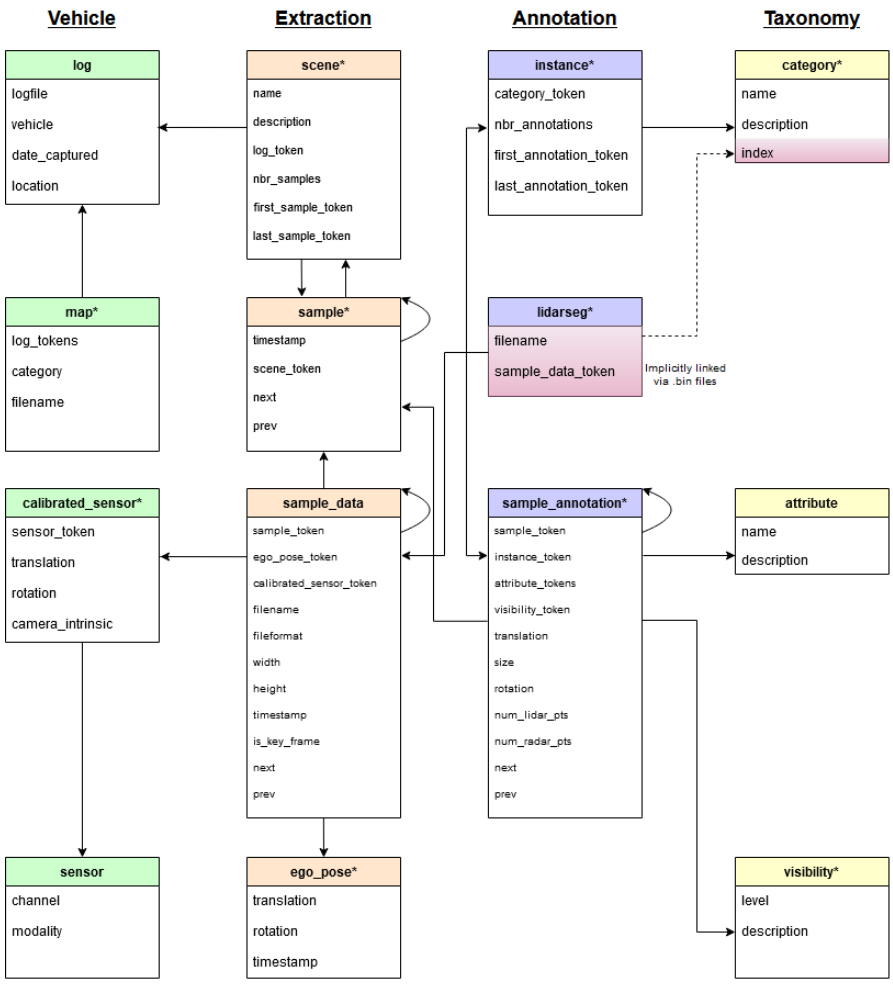

Figure 7. Nuscenes database schema [CBL+20]

and will allow extraction of smaller size patches for the augmentation later in the pipeline. Due to these reasons, we decide on the direct conversion of one scene -> one image for our dataset.

The scene holds multiple foreign tokens to samples belonging to it. A sample on the physical level represents one frame of the sensor data with given synchronized timestamps. A sample holds tokens to sample data containing information from every sensor separately, together with relevant calibrations and transformation matrices for sensors and the vehicle itself. In addition, the sample contains token to sample annotation, which has segmented boxes of every object of interest, together with their classification label. The coordinates of all sensor and object data are local and in line with the provided map data geometries coordinates, so no additional adjustment to align sensor data to the

map is needed.

Following [LHM+19], we want to extract all the LIDAR data for a scene available for the input images of our boundary extractor and all the map data. Naively, we iterate over all samples inside a scene, and after transformation of LIDAR cloud from sensor frame to vehicle frame to global frame for a given sample, we conjugate them all into a single point cloud for a scene. To clean LIDAR data from unnecessary points for road boundary detection, we employ several techniques. Firstly, we establish the minimal and maximal effective range on the sensor frame and remove all points outside of our specified range. This allows us to remove the points belonging to the sensor's carrier and points detected far away that inflate image size while being too scarce to provide significant information for the Neural Network. Secondly, we remove all points with a z-axis value higher than 0.5 meters on the vehicle frame. This allows a fast and elegant solution to remove almost all non-ground points while saving information about road boundary curbs and like and not being reliant on any complicated ground point filtering algorithm. Finally, on the global frame, we filter out all points belonging to dynamic objects, i.e., cars or pedestrians, as we are not interested in them. Dynamic objects in Nuscenes are encoded in bounding boxes, and while these boxes often include some ground points, there is no problem, as we accumulate ground points from all the different samples.

The final result of this filtering and accumulating process is the LIDAR point cloud that contains mostly ground points and covers all the areas traversed in the scene. The following steps encode information from the LIDAR cloud in the Birds-Eye-View - namely, intensity and z-axis values. For the intensity, we can project values to pixel cells on the 2D grid by discarding z-axis values of the point cloud. However, as both our observations and [Ma20] show, on the low pixel cell size (4cm x 4cm in our case), the point cloud is often not dense enough, leaving a lot of undesirable gaps between pixels. While there are several interpolation methods we could use, most of them under-perform or do not compute within memory or time restrictions when the number of data points goes into millions, as in our case. One of the workable methods is Inverse Distance Weighting [ZPRA99]. To optimize IDW for our conditions, we encode pixel cells and LIDAR points in the GPU-powered KD-trees [MM99], and use KD-trees for closest neighbor lookup with limited range, and k=9. Using these closest neighbors' intensity values and distance to the pixel cell, we calculate its intensity, even if no points are projected in the pixel cell itself. For the z-axis, we project the maximum z-axis value from all points into the cell and then calculate the Sobel gradient value by calculating Sobel derivatives in both directions (filter size=11) and taking the square root of the sum of their squares. Finally, we encode the Sobel gradient value in the pixel cell. For both of the images, we also establish a minimum bounding box that is dependant on effective LIDAR distance from vehicle trajectory and transform the images to it. The entire pipeline for the input images is presented in Figure 8.

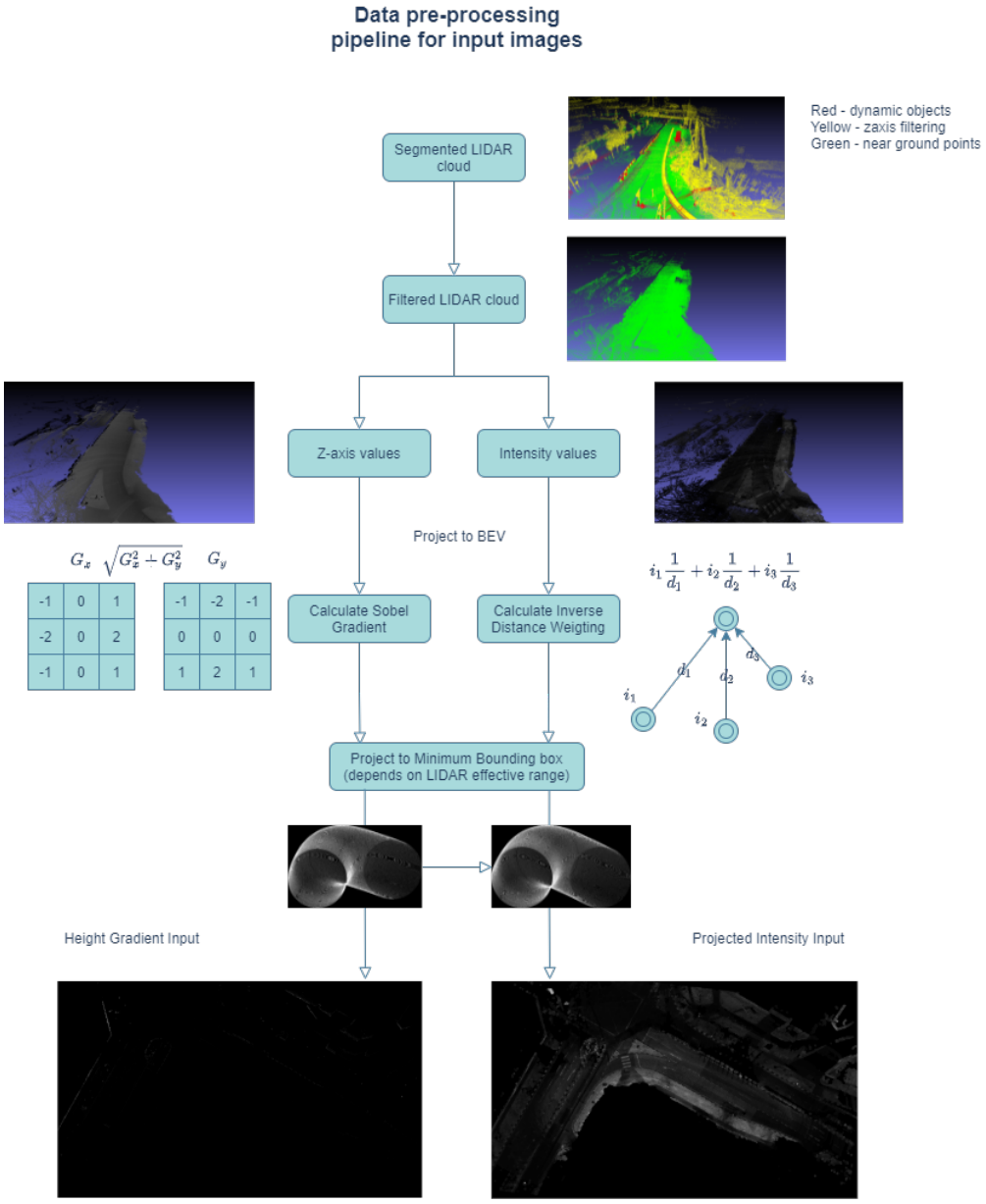

Figure 8. LIDAR data encoding pipeline

Analogously, we want to encode HD-map boundaries data into images that will serve as regressor targets. We use Nuscenes provided HD map in WKT format geometry, and extract the "driveable area" layer from it, already projected into Bird-Eye-View and having the exact local coordinates as our LIDAR information. Next, we transform it into Multi-Line, where every line represents a single boundary. Sometimes the lines encoded in Nuscenes are only fragments of a single line (boundary), so we double-check if lines touch at their endings and connect them if necessary. From here on, we can easily encode all the regressor target outputs denoted in [LHM+19]. First, we encode road boundaries endpoints using line endpoints and set the closest 4 pixels to them. Next, to encode the road boundary itself, we create two separate output images - one encoding distance to road boundary, another direction to road boundary. To generate these images, we populate the pixel cell grid with points and use the geometry library to find the closest point to each of them on our road boundary Multi Line. Given two points, we have a vector. In the first output, we encode the inverse of the vector distance in every pixel. In the second output, we record the direction of the normalized vector in every pixel. To specify vector two values, we use RGB images and write their absolute values in two channels while using third to hash both signs of their X and Y values. Lastly, some minimal bounding box projection for inputs is used to reduce final image size and maximize useful area. The entire pipeline can be seen in Figure 9.

The whole pre-processing is written in Python and, as such, not perfectly optimized. Still, as the pipeline is programmed to process each scene separately, the lazy parallelization of invoking multiple instances to process multiple scenes simultaneously was used. The targeting pixel size of 4cm by 4cm, processing of whole Nuscenes database into our dataset takes around 16 hours on the server cluster. This time includes the generation of one set of images for every Nuscenes scene and around 20 smaller patches for them for data augmentation. The final dataset size is 20 patches x 750 scenes = 15000 images, but after manual filtering of 250 non-workable Nuscenes scenes (i.e., very sparse point clouds, wrong/incomplete HD map), we are left with around 10000 images. Factually, the real size of the dataset is even smaller, as several scenes' routes are in the same geographical location, making parts of their point cloud overlap. While the point clouds are still recorded during different rides, using potentially different sensors, the overlap between static areas is significant. Using this dataset, we manually split it into training/validation (3:1) sets to avoid location overlap as much as possible for our neural network implementation discussed in the next section.

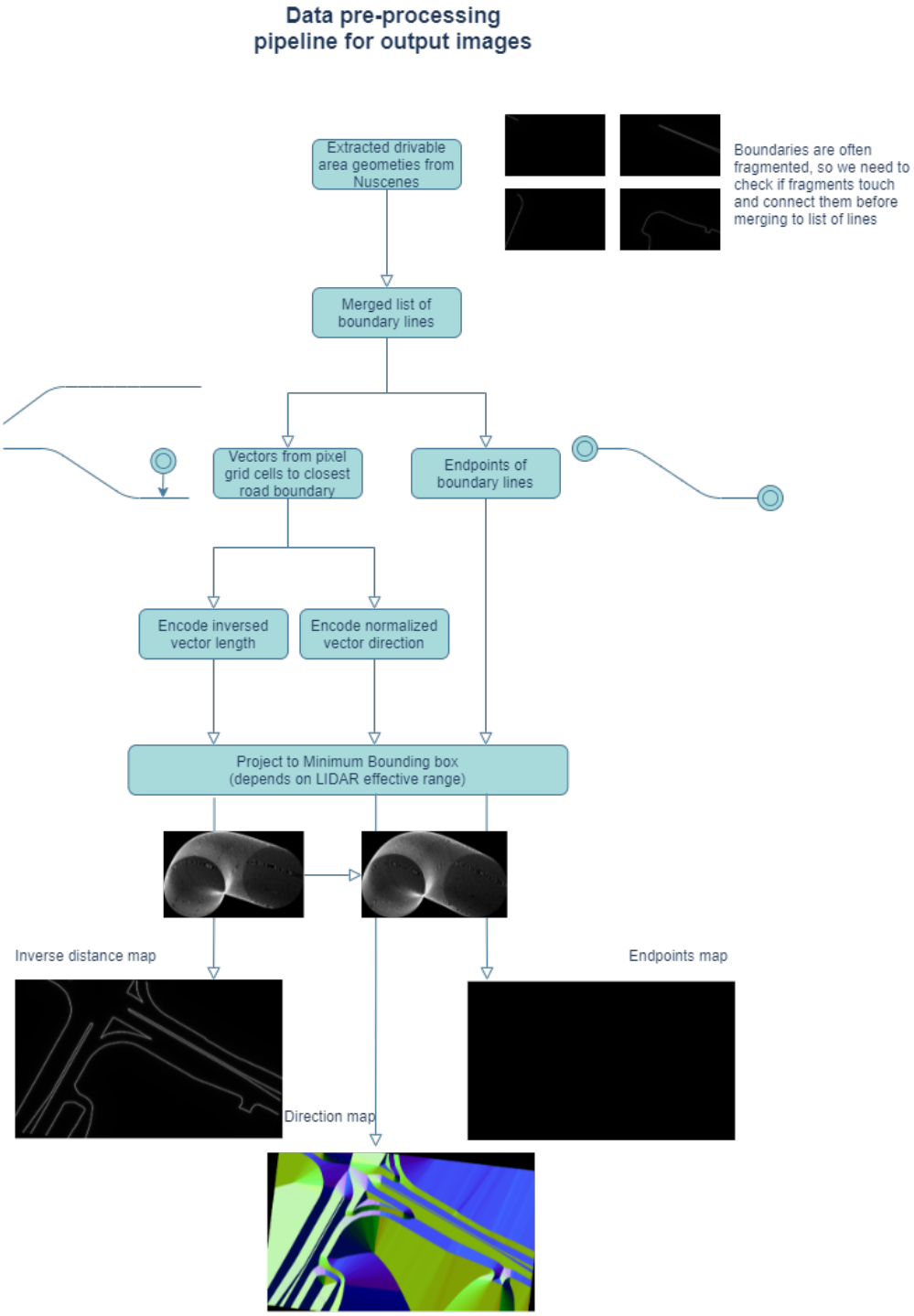

Figure 9. HD map data encoding pipeline

4 Stage 1 Architecture

4.1 Network Architecture

The goal of Stage 1 is to extract all useful information from a LIDAR point cloud and convert it to the road boundaries estimations that can be further refined in Stage 2. To this goal, we re-implement neural network architecture used by Liang et al. [LHM+19], which by itself is a variation of classical Feature Pyramid Network architecture [LDG+17]. The architecture was chosen over the analogous works reviewed in Related Works due to the following reasons:

1. It is one of the only available architectures that use only LIDAR point cloud projected on Birds-Eye-View as main input, while still optionally allowing satellite/camera imagery as auxiliary data for minor gain. Nuscenes dataset and publicly available datasets do not have satellite or comprehensive 360-degree camera data included, so this part is not realized in re-implementation.

2. Feature Pyramid Network is the practical and straightforward architecture that saw successful application across many Computer Vision problems. Several improvements have been proposed for the base structure since the original introduction, and a few are implemented in our re-implementation.

3. The network is robust because of the skip connections and specialized in processing large-size images that consist the majority of our processed Nuscenes dataset.

On the base level, Feature Pyramid Network is the deep convolutional network that is divided into two stages: encoder and decoder. The encoder stage features convolutional layers grouped into several layers, and inside, every layer is grouped into several units. Feature map size is kept static across the layer but reduced 2x2 by going every layer up, instead of depth increases every layer to represent more rich spatial information. On the implementation side, we have five layers, and we use two ResNet or ResNext [XGD+17] units in every layer and achieve downscale by using strides. This is another significant difference to Liang et al. implementation (in addition to dropping auxiliary camera input), as their implementation achieves a reduction of feature map size by using dilated convolutional neural network layer [YK15]. Using an equal amount of memory, implementation using dilation ends up much shallower in our tests and produces the worse results. Exemplary comparison on the training data can be seen in figure 10.

The decoder stage is organized into several layers, with a 2x upscale between layers and a simple addition skip connection to the layer from the encoder stage with exact feature map size. Every decoder layer holds the so-called head unit, which starts with a convolutional layer with fixed depth branching into three different convolutional layers - one for every output. After branching, all three outputs from every layer are upscaled to

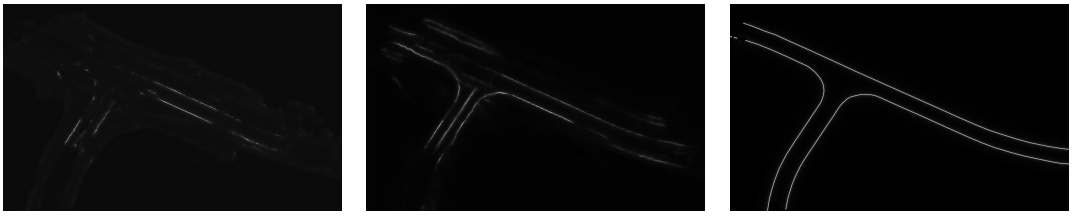

Figure 10. Comparison between implementation using dilation(on the left) and implementation using strides (in the center). Ground truth of the data image presented on the right

original image resolution, after which matching output branches are concatenated and reduced to the depth of their output. Every single convolutional operation is followed by the optional batch normalization and ReLU activation layers. The implemented structure allows the pixel to pixel regression output with final linear activation layer and classification output by putting an additional sigmoid activation layer in the end. The overall architecture of FPN model is presented in the figure 11.

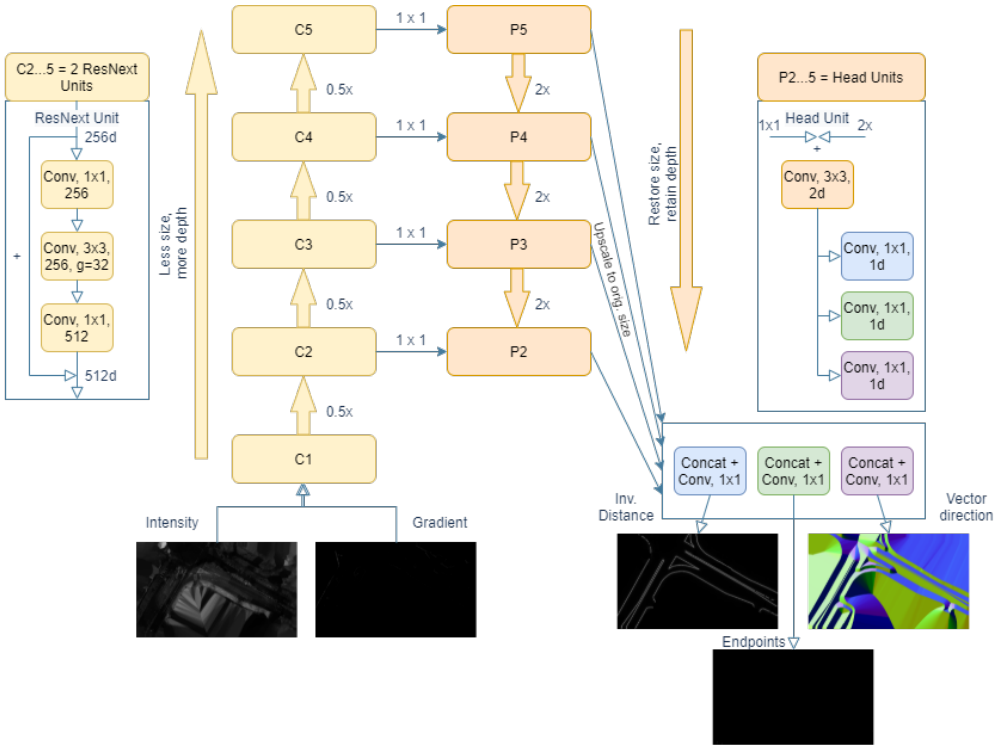

Figure 11. Stage 1 architecture model

4.2 Practical implementation details

Baseline architecture was implemented in Keras using Tensorflow backend, and trained on GPU nodes on High-Performance-Cluster provided by the University of Tartu, containing Tesla V100 GPUs, with a memory limit of 32 GB. The network was trained using Adam optimizer with a learning rate of 1e-3 and batch size of 4 and trained in 4 different versions by default: without batch normalization and regularization factor, with batch normalization and without regularization factor, without batch normalization, and with regularization factor of 1e-4, with batch normalization and with regularization factor of 1e-4. We normalize input images to mean 0, std 1 before feeding them to the neural network and resize both input and output images. In addition to that, we explored and compared several variations of the baseline architecture to iron out small unspecified details of original implementation and potentially improve the result. The loss function heavily depends on the used variation and will be explained there. Several binary categories can classify these variations:

1. Strides versus Dilation: As mentioned previously, the original implementation uses dilation to expand feature map depth, while our uses strides. Theoretically, strides implementation contains more parameters and slower to train and features more depth in the layers with the same memory limit. Dilation implementation contains fewer parameters and faster to train, and can feature equal depth only with drastically increased memory consumption. In practice, with the same memory limit of 32 GB, the dilation results are vastly inferior both visually and in terms of the in-training loss.

2. One output versus Multi-output: Parent implementation uses three different outputs as notified - distance, endpoints, and direction maps. In [LHM+19], the loss necessary to train a multi-output model is the weighted sum of mean squared error losses for distance and endpoints and cosine similarity loss for direction maps. In our early tests, we implemented this to the word, however, the resultant trained neural network provided horrible predictions in areas where the input projected LIDAR point cloud did not have any information (i.e., outside of LIDAR effective range). This applied for both direction and distance outputs and was seemingly caused by neural network training getting confused in similar areas during training - i.e., input data is uniformly zero, while output direction map is non-zero and similar to when input is non-zero. As we do not use the same algorithm for the second stage as Liang et al., training with only one output necessary for our approach - distance map was tested and reduced specified phenomena significantly. Initially, we used root mean squared error loss (RMSE) on the distance map ground truth compared to neural network output for the one output approach.

$$\sqrt{(y_{true} - y_{pred})^2}$$

3. Weighted loss vs. normal loss: while the phenomena described in the previous point were significantly reduced with the adoption of one output architecture, it did not disappear completely. Estimating the reverse distance to road boundary can be crudely regressed into classification tasks with two classes: nearby boundary with high regression value, the outside boundary with regression value close to zero, or foreground and background classes. Following this, we have a classical problem with many false positives for the underrepresented class, in a problem where false positives are extremely unwanted. As such, we implement weighting penalizing false positives much more than false negatives to combat this problem. While original output data is in continuous pixel form values 0-255, we assign foreground class to pixels with a value of 21+, representing the distance to the boundary of less than 40 cm. Then we can use straightforward weighted RMSE function:

$$\sqrt{\frac{\sum_{i=1}^{n} w_i (y_{true} - y_{pred})^2}{\sum_{i=1}^{n} w_i}},$$

where specific value of weight is equal for every pixel that belongs to the class in the image and found by experimentation,

$$w_{background} = 0.9, w_{foreground} = 0.1$$

As such, the final architecture used for all the subsequent testing has only one output, is implemented with strides convolutional neural network, and has weighted RMSE loss in addition to all the basic architecture technical details. We train the specified architecture is mentioned four mods for 20-30 epochs. Unfortunately, as can be seen in Figures 12, 13, validation loss stops decreasing around Epoch 5-10, while training loss continues to decrease afterwards. It is a direct sign of overfitting, and all of the experimental architecture variations show a similar trend. The temporary conclusion is that the presented dataset is not sufficiently big or diverse for the neural network to continue learning besides the initial episode. However, as shown in the Dataset selection section, we have little alternative, and the only choice was to proceed with an evaluation based on the results we have.

Further results are extracted using batch normalization model only, as tests showed regularization factor introducing unnecessary false positives, while no normalization or regularization lead to background pixels having values very close to foreground pixels. See the Figure 14 for comparison.

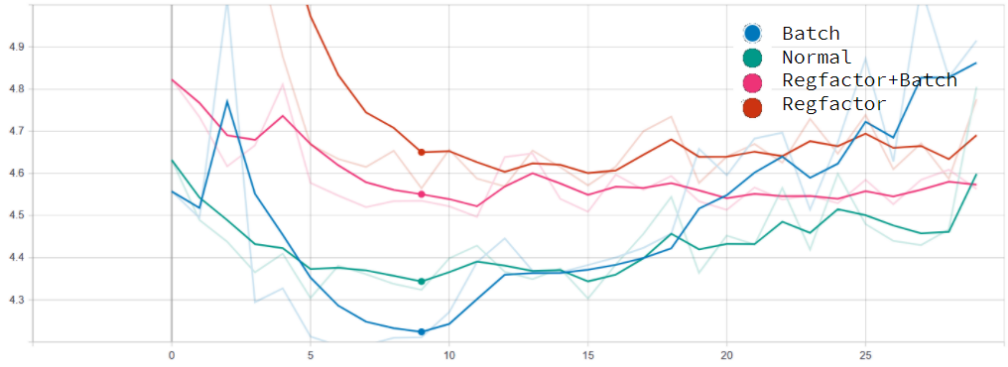

Figure 12. Stage 1 architecture validation loss

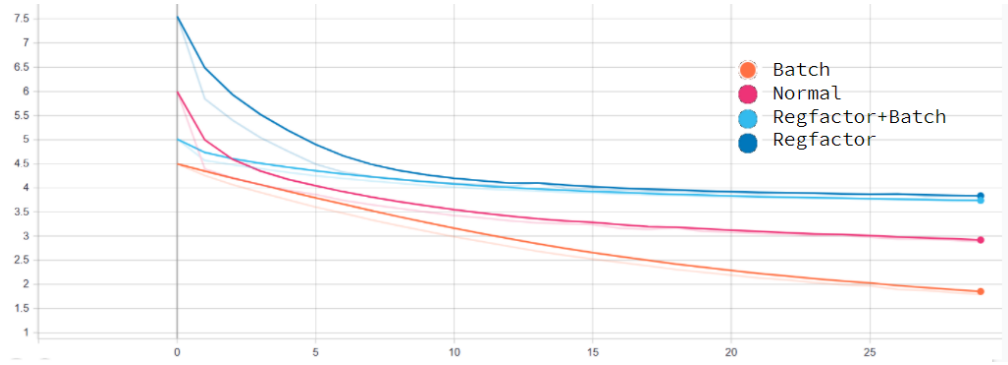

Figure 13. Stage 1 architecture training loss

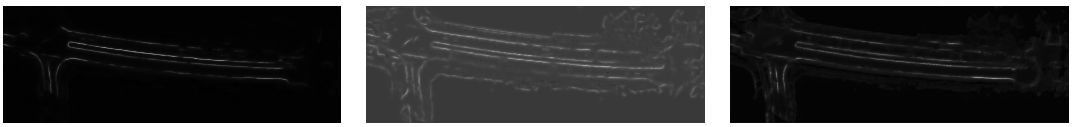

Figure 14. Comparison between validation data output on architecture with batch normalization (left), without batchnorm or regfactor(center), with batchnorm and regfactor(right)

4.3 Evaluation metric and intermediate results discussion

As the automatic road boundary extraction problem is quite novel, there are no universally agreed metrics. To have a point of reference, we implement the metric used by Liang et al. [LHM+19], which is very similar to what is used by [Ma20]. The main idea of the metric is to re-implement the basic confusion matrix for two classes - road boundary and rest (foreground and background), however, add threshold tolerance to the meaning of truth. If, for example, threshold tolerance is 20 cm, every pixel in the ground truth image

counts as True Positive if there is at least one predicted pixel in the radius of 20cm, and False Negative if there is none. We calculate every positively predicted pixel as False Positive if there are no ground truth pixels present in the tolerance radius. All other pixels are treated as true negatives. Figure 15 provides visual representation. Such metric is easy to calculate and provides useful derivatives of precision, recall, and f1 metric when we care only for foreground class, and they will be used in the final evaluation of stage 2. The only negative is that with a large enough threshold, one predicted foreground pixel can satisfy multiple pixels on multiple road boundaries, which should be avoided by lowering the threshold. Fortunately, even in urban conditions, there is a significant gap between road boundaries, measured in meters, and most evaluations in [LHM+19] [Ma20] deal with 10-40 cm thresholds.

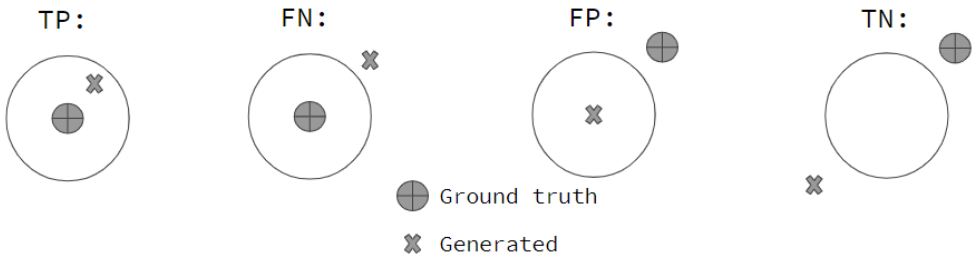

Figure 15. Confusion matrix with threshold visual representation

The first step to applying this metric is to transform the continuous output of neural network regressor into discreet output containing two classes only. As the output decodes distance to the closest suspected road boundary, thresholding seems a natural choice. Initially, a simple percentage thresholding and Otsu thresholding (that finds optimal static percentage threshold for every image automatically) [KOA92] were attempted. However, after brute-forcing simple percentage thresholding on training data, the best result achieved with the threshold of 0.043 of the maximum value in the image, with the tolerance of 40 cm was average precision, recall, and f1 score on the level of about 0.45, with Otsu thresholding giving slightly better result, but providing several extremely bad outlier images where more than 70 percent of the image were identified as foreground class. The alternative Gaussian adaptive thresholding was adopted and quickly surpassed both in metrics, giving precision level of 0.48, recall level of 0.44, and f1 score of 0.46, and looking more suitable for future Stage 2 refinement, see the figure 16. Validation data scores using adaptive thresholding are significantly worse, at precision 0.38, recall 0.34, f1 score 0.356. Evidently, the final results of stage 1 are nowhere near close upper .9 values for recall/precision/f1score reported in [LHM+19] using the same tolerance range of 40cm. While Liang et al. scores are not directly comparable due to their testing being done primarily outside of the urban setting, Ma et al. [Ma20] reports similar results

in his work done in crowded cities' streets, given the same tolerance range.

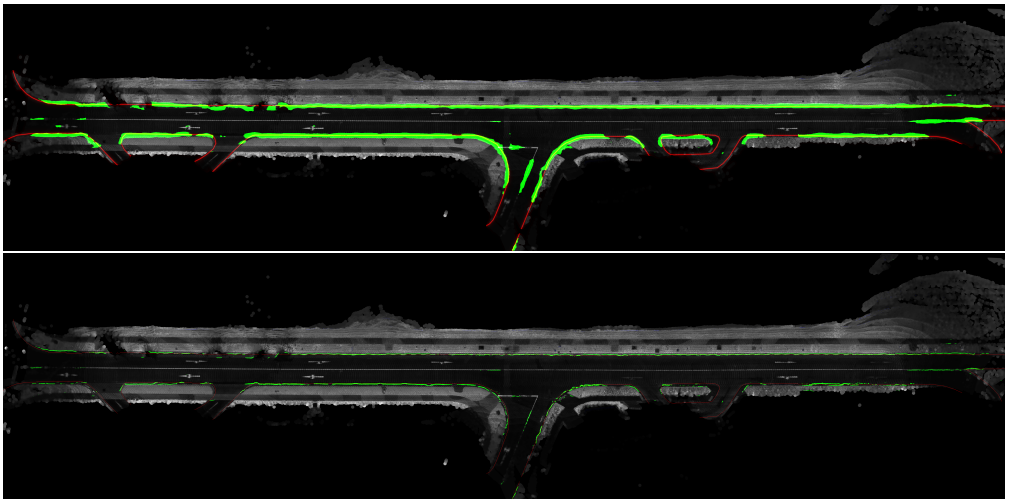

Figure 16. Comparison between Otsu (top) and adaptive Gaussian thresholding(bottom)

5 Stage 2 Architecture

5.1 Morphological-based cleaning

The main goal of Stage 2 is to refine our predictions from Stage 1 and accomplish the following three objectives: remove remaining false positives, make raster data suitable for vectorization, and complete missing gaps between road boundaries segments. Adaptive Gaussian Thresholding is the first step in the process and partially fulfills the first objective. Most contemporary approaches [Ma20] [BMH$^+$18] usually rely on additional Machine Learning based solution for these purposes. However, we cannot implement any similar solutions, as they often depend on amortized learning: where a dataset of the new supervised training process is constructed from the validation part of the old. In our case, the validation part of our converted dataset is tiny, including only 70 scenes and even less variety if counting only completely separate locations. Therefore we fall back on more classical Computer Vision-based approaches - mainly morphology operations, because of their robustness of working with binary data representing different pre-determined shapes.

The developed refinement process consists of many primitive morphology operations implemented in SciPy or Scikit-image Python packages, chosen for stated objective achievement. All the operations are performed only on the foreground class pixels, generated by initial Adaptive Gaussian Thresholding. Additionally, we use our specially created module for road gap completion, described in section 5.2. The process itself is linear and consists of the following steps:

1. Apply Adaptive Gaussian Thresholding to generate initial binary image and clear some of the false positives remaining.
2. Create a slightly dilated mask on all regions not containing any information from the original intensity image, as the neural network could not predict road boundaries without any information.
3. Dilate thresholded image slightly in order to close small gaps between potential road boundary segments.
4. Skeletonize the image to extract a one-pixel wide representation that can be easily turned into a vector format.
5. Apply negative information mask and remove all predictions in its range.
6. Prune short branches from created skeleton by identifying longest branch (solution to this step is adapted from open source code by [KR16]).
7. Image connected regions are segmented, and very small segments are deleted to clean up an image.

8. Developed completion technique is used and connects nearby segments that form almost a straight line on their endpoints and adjacent to endpoints pixels to complete road boundaries.

9. Image connected regions are segmented, and small segments are deleted to finish cleaning up an image.

The resulting image contains many one-pixel wide skeletons without any side branches representing road boundary segments, allowing easy conversion to a vector format like Multi Line. Additionally, the implemented algorithm ensures that straight gaps between road boundaries are complemented, however, the algorithm is not perfect and does not complete all the gaps, as well as does not differentiate between true and false positive curves, completing them nondiscriminatory. Visualization of the entire process is present in Figure 17.

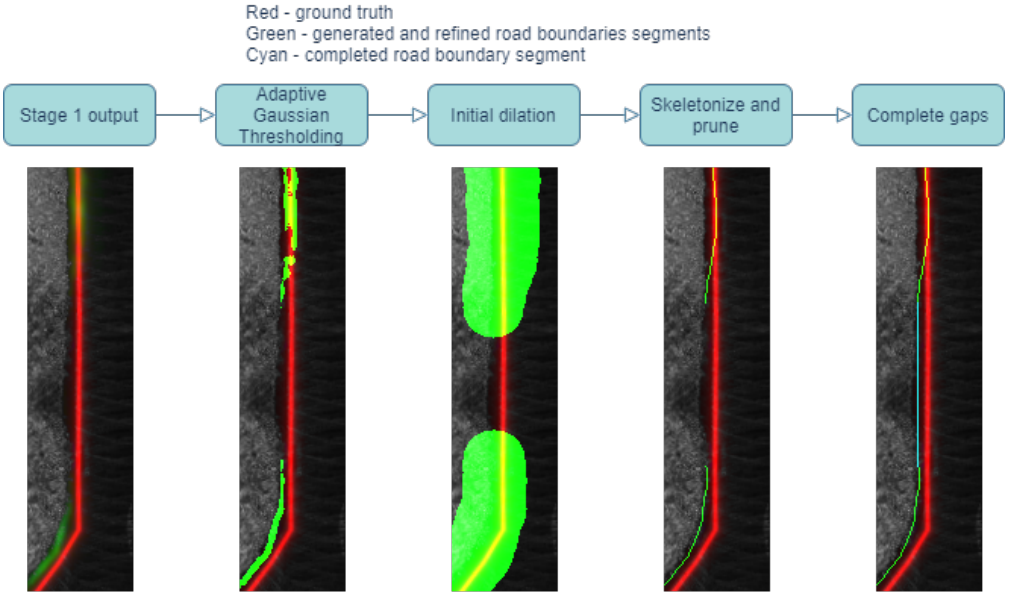

Figure 17. Morphological-based refinement and cleaning process

5.2 Road segment completion module

Road boundary gap completion is a specific case of contour completion task, which is a non-trivial problem with most state-of-art solutions relying on Machine Learning, for example - Conditional Random Fields based solutions [RFM08] [SSGK14] - to solve it. Unfortunately, no open-source solution suitable for our case has been found, and

re-implementing another complex algorithm has been deemed outside of the scope of the thesis due to time constraints. Instead, we implement a simple geometry-based solution that relies on the domain assumptions about road boundaries. The primary assumption is that we can assume road boundaries to be extremely smooth in most of their segments, excluding sharp turns on the crossroads, or in case our road boundary is internal inside the road and forms a closed polygon. This exception disqualifies treating the problem as the simple line fitting problem that can be solved by Hough transform [IK88]. We will still focus on completing only the linear-like gaps while considering that the total curve can still have turns, see Figure 17 for example.

To achieve completion of a mostly linear gap between two disjoint road segments $\acute{A}$ and $\acute{B}$, we first isolate the closest endpoints on the segments: A_1 and B_1. Endpoints here are defined as segment pixels that have only 8-way neighbor pixels belonging to the same segment, while the segment, in general, is defined as a collection of foreground class pixels where one can travel from any pixel in the segment to any other using 8-way connections through only pixels that have foreground class. From here on, "pixel" will refer to foreground class pixel unless noted otherwise. Due to morphological processing before this point, the road segments that are not loops are guaranteed to have only two endpoints. Next for endpoints A_1 and B_1 we recurrently find $X \approx 40$ adjacent pixels that lay on their respective road boundaries, and define "extended endpoints" sets of points $A = A_1...A_2$ and $B = B_1...B_2$. From here on, we treat the problem of gap completion as line fitting of the $A \cup B \cup A_1B_1$ point set.

However, even sets A or B separately often do not resemble line, so to judge if the entire union can be regressed to a line-like connection, we encase it into a minimum area bounding rectangle [FS75] and find if the smaller side of the created rectangle σ is less than ε, where ε is heuristically found minimal acceptable size measured in pixels. Naively, it is easy to see that if σ approaches zero, then the entire bounding rectangle can be regressed to a line without losing the information. More formally, the minimum bounding rectangle encasing $A \cup B \cup A_1B_1$ is at least as big as the minimum bounding rectangle encasing A_1, A_2, B_1, B_2 (because it has to enclose at least those four points at minimum). We can treat it as a geometry problem in classical Euclidean space (refer to Figure 18). Assuming that the beams A_2A_1 and B_2B_1 intercept, we get a quadrilateral with the minimum bounding rectangle of $A“A‘B‘B“$ that can be degraded into trapezium $A“A_1B_1B“$ without significant loss of information. When we can approximate our union as a line, then the intersection angle of beams at point C, $\gamma = \alpha + \beta$ is nearing π, and the larger of α, β is nearing $\frac{\pi}{2}$. As the $B_2B‘ = B_1B_2 \cos\alpha$, it is nearing zero 0 giving $B_1B_2 \approx X$, thus giving us easy metric for gap filling. If the beams do not intersect, we can use a line formed by midpoints of their bounding rectangle smaller sides to mirror one of the beams to guarantee that the beam and mirrored beam intersect, while the minimum bounding box is unchanged. If, after mirroring, beams still do not intersect, then we have the specific case of $A_2A_1B_1B_2$ forming the rectangle, in which case σ will

be significantly bigger than ε.

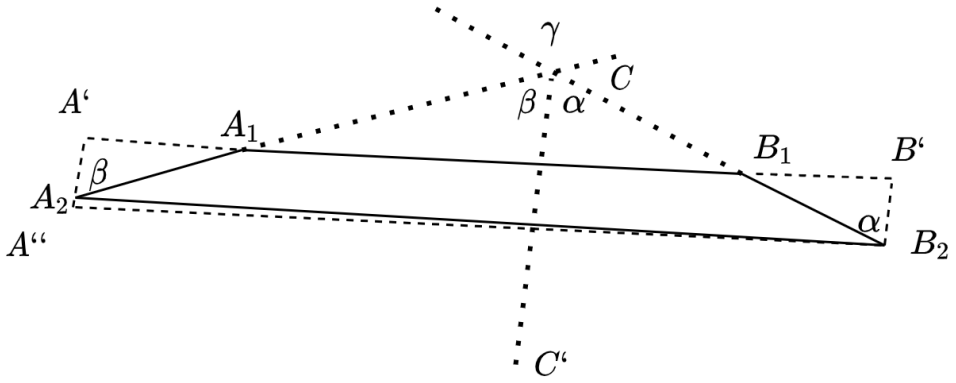

Figure 18. Simplified representation of road gap completion problem

The completion algorithm itself is simple and tries to connect all the isolated road segments pairwise using the established criteria. First, it finds all the isolated road segments using [GBC10] 8-way connected components algorithm. For each connected component, it identifies its endpoints and drops all segments without two endpoints. It extends the endpoint by looking up pixels belonging to the same segment in the extended neighborhood. Finally, it sorts segments by overall pixel length and calculates pairwise distances from every endpoint to all others not belonging to the same segment. The main loop consists of the algorithm iterating every segment starting from largest and trying to connect it with all other segments one-by-one starting from closest by distance, using pairwise endpoints that are closest. The algorithm puts both extended endpoints into a minimal bounding rectangle and calculates its sides. Then, the algorithm compares the smaller side σ versus heuristically established $\varepsilon = 6$, and the larger side versus maximum connecting range of 500 pixels, connecting road segments if both values are smaller by drawing a line between endpoints. If the algorithm connects two endpoints, then neither can be connected again and if the algorithm connected both endpoints of the given segment, it skips to the next pre-emptively. The complexity of the main loop is, therefore, $O(n^2X)$, where n is the amount of the segments (as we iterate over them square times) and X is the total amount of points in $A \cup B$ (as the linear time solution for minimum area rectangle exists [FS75]) and is negligible as we have 20-30 segments and 100 total points at union at most. The complexity of pre-main loop operations is hard to estimate due to considerable reliance on inbuilt vectorized operations from OpenCV Python package. Overall, the algorithm is only a small (less than ten percent) part of the total pipeline execution time.

5.3 Experimental results and discussion

In section 4.3, we have established a simple confusion matrix-based metric to evaluate the result of rasterized prediction. We are using the same metric for stage 2 evaluation as we still have raster data as the final output with a tiny change - ground truth data is skeletonized before comparison with Stage 2 output data. Following [Ma20] and [LHM$^+$19] we consider thresholds of 20 and 40 cm, and in this section, 40 cm is used as default. All the data used for showing the results in this section is from the validation part of the dataset used in Stage 1.

To comprehensively visualize Stage 2 results, we overlay final refined road boundary pixel output denoted colored in green, road gap completions created by algorithm from section 5.2 in cyan, thinned ground truth road boundaries colored in red, original LIDAR point cloud projected to BEV in greyscale, and auxiliary z-axis gradient in blue. The entire pipeline of reading Stage 1 output, processing it through Stage 2, and calculating metrics takes 90 seconds on average. Eighty to ninety percent of the processing time is taken by finding the longest branches in the skeletons and common dilation operations substeps. Overall performance is not bad for the pixel accuracy of 4cm that it provides, and there is no real-time requirement. Additionally, it can be easily parallelized by processing every image in a separate thread, cutting total time significantly.

As reported in section 4.3, we had low calculated metrics, and viewing through visualization confirms scarcely slight improvement over them. The worst part is the wide variety over the results, with some being quite good, that can be adapted into actual HD map only with small expert final touches, and some being extremely bad, recognizing only a small part of the actual road boundary and creating more false positives than can be negated. To illustrate, we include some of the best examples on the Figure 19, some average examples on the Figures 20 and 21, and some of the worst examples on the Figure 22.

The best results have high robustness correctly predicting most of the outer boundary and some of the more complex boundary structures while leaving almost no false positives, reaching a recall value of up to 0.8 and a precision value of 0.9. The average result category consists of most of the output dataset, where the process correctly recognizes at least some parts of the boundary. However, it often leaves many false positives, and in this case, the completion routine often connects false positives segments, further decreasing precision. Still, precision ranges on levels from 0.5 to 0.7 and recall from 0.3 to 0.4. The worst result category includes horrible results, with network and refinement essentially learning nothing about road boundaries and spawning more false-positive curves than actual road boundaries segments. Overall, on the subjective discrimination, the resultant output set contains 1:6:3 ratios of best:average:worst results. The analysis shows that while the architecture definitely learns useful features and refines them accurately in some cases, it is very far from having competency in all different situations, further proving that the initial training set is incompetent for the given goal.

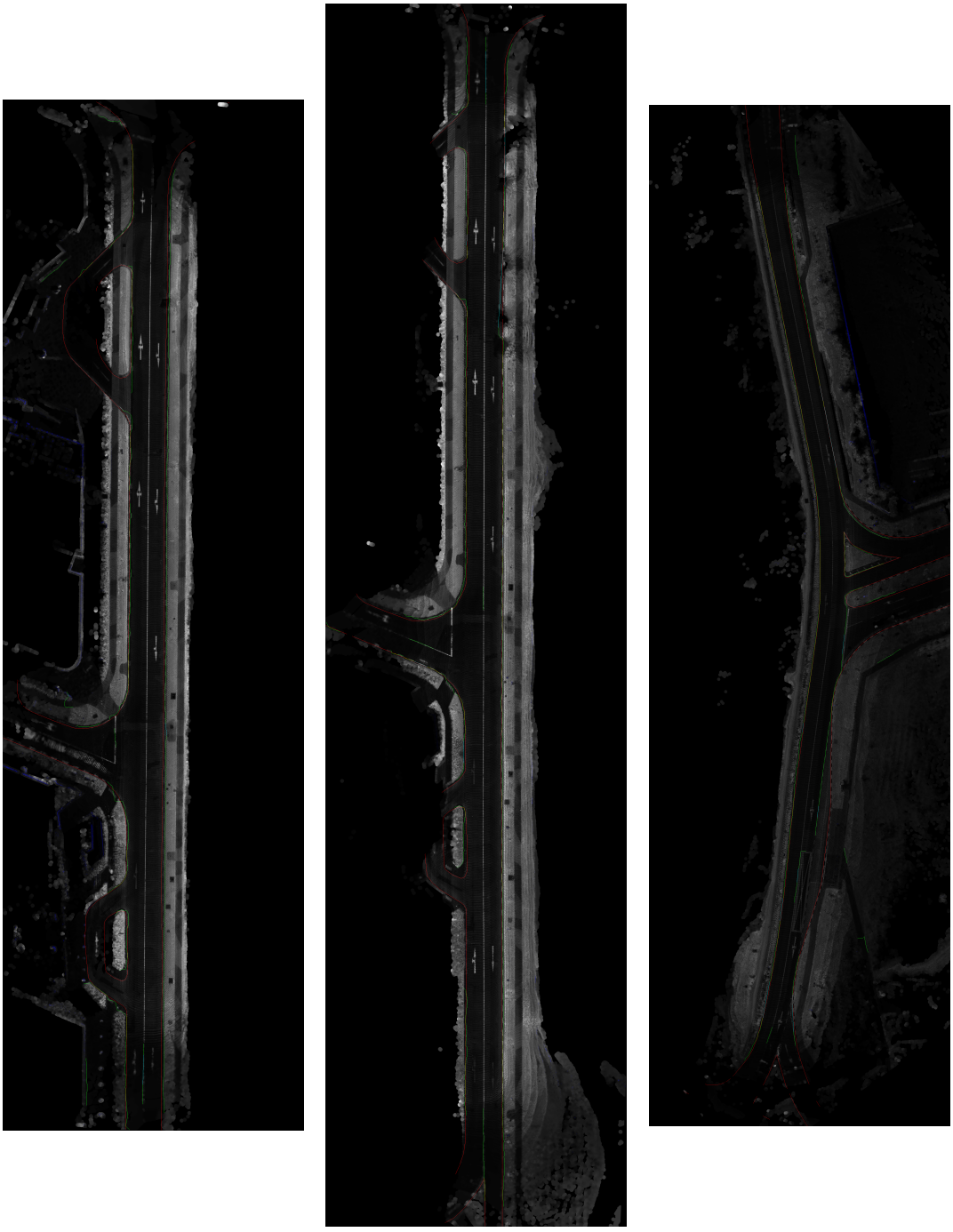

Figure 19. Best results of the entire pipeline. Note: Best viewed in digital form, zoom if necessary

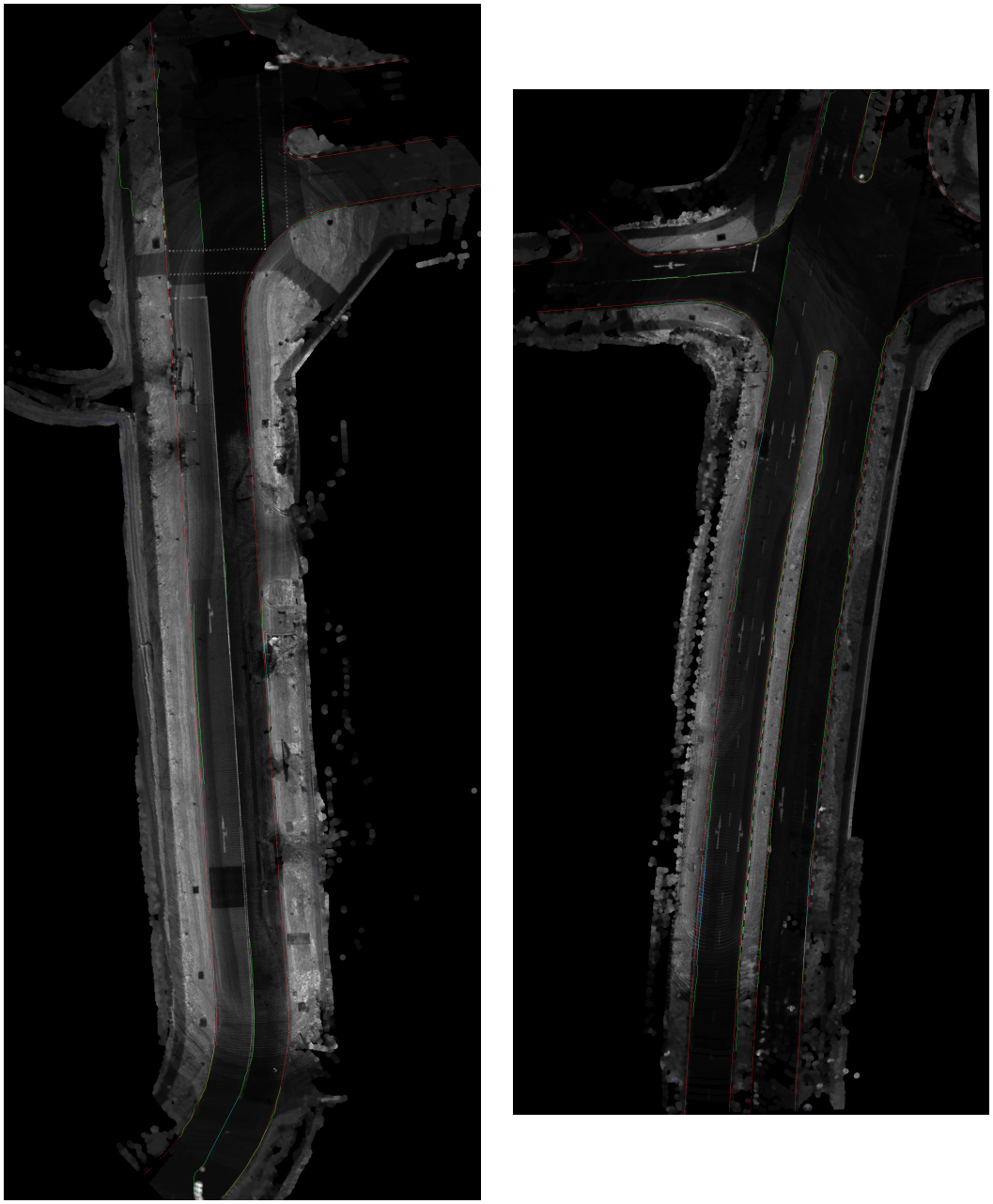

Figure 20. Average results of the entire pipeline. Note: Best viewed in digital form, zoom if necessary

Figure 21. Average results of the entire pipeline (cont.). Note: Best viewed in digital form, zoom if necessary

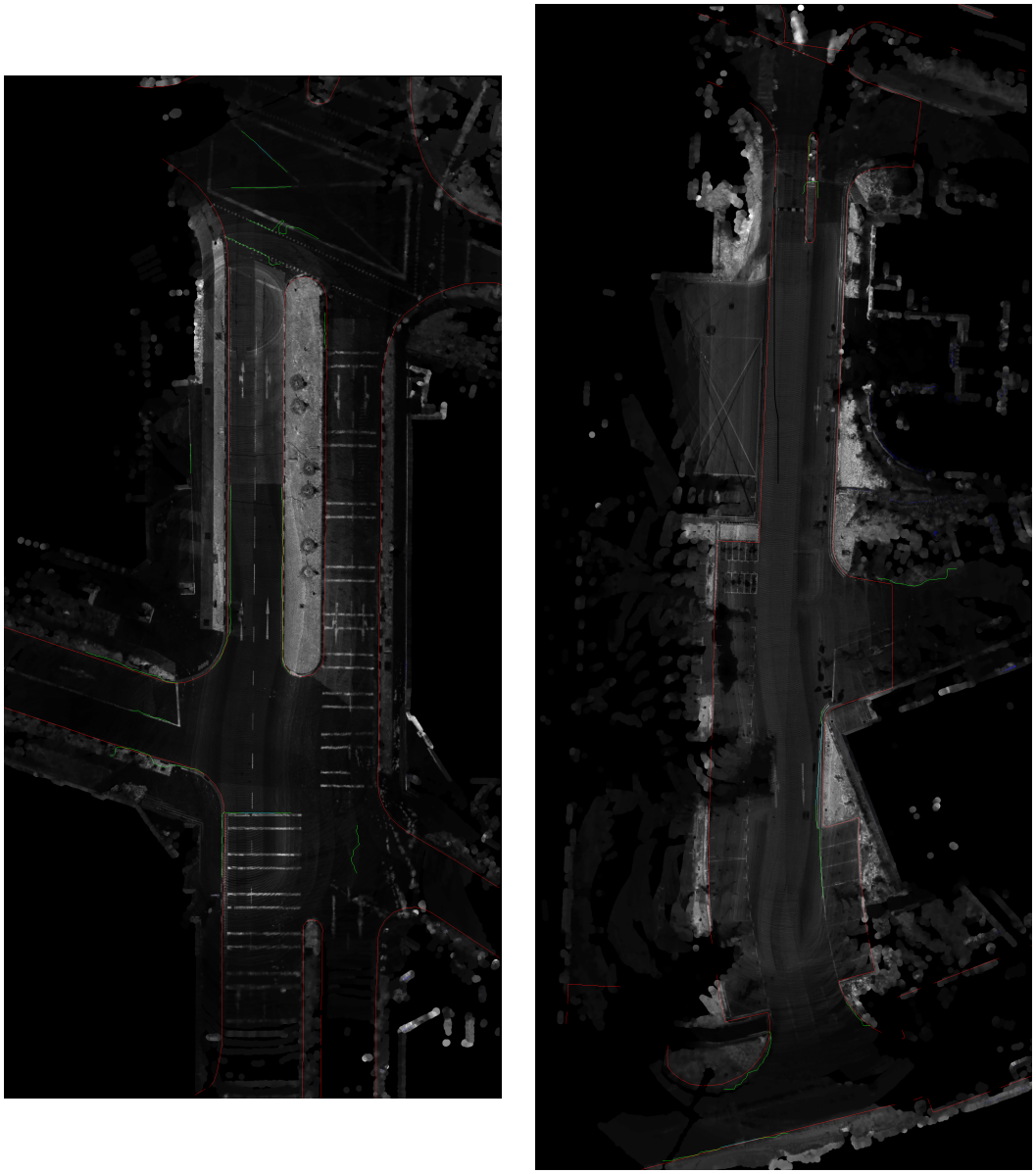

Figure 22. The worst results of the entire pipeline. Note: Best viewed in digital form, zoom if necessary

5.4 Final results comparison

The relative novelty of the task combined with non-standard metrics restricts the number of contemporary works we can compare our results. Additionally, only Ma et al. [Ma20] performs experimentation on the data similar to ours, i.e., complex urban setting, though he includes evaluations of several older methods on his private dataset, the most recent one included here [ZLG+17]. However, this is not a problem by itself, as our results do not reach the state-of-the-art results by a long margin. Still, in Table 2, we compare our method with and without the road completion module to the contemporary works from the last five years covering the same problem. Additionally, we include the 90th percentile, roughly representing an average of our best predictions our architecture is capable of, representing the theoretical best our method can achieve for comparison.

Table 2. Comparison table of the final results

Solution	Precision		Recall	
	20	40	20	40
Stage1+Stage2	0.44	0.62	0.19	0.28
Stage1+Stage2 +Gap Completion	0.44	0.62	0.22	0.31
Stage1+Stage2 +Gap Completion 90th perc.	0.7	0.9	0.45	0.58
Distance Transform [BMH+18]	0.85	0.96	0.84	0.94
CNN+RNN [LHM+19]	0.82	0.93	0.82	0.93
CNN+Adversial GAN [Ma20]		0.9		0.91
Voxel-based non-ML [ZLG+17]		0.89		0.91

While even the 90th percentile does not quite achieve the state-of-art results, we can still draw some positive conclusions. First of all, as was reiterated earlier, the architecture does learn valuable features, and the refinement process is effective enough not to worsen and sometimes improve convolutional neural network predictions. Secondly, implemented road boundary gap completion module is practical and increases our recall, almost at no total cost to the precision. It would be even more effective with better baseline input, as heuristic constraints could be relaxed when there is less fear of false positives in general. Third, we confirm the non-triviality of the given task and poor suitability of even the best publically available datasets for HD map-related research.

6 Conclusions

In this thesis, we present an overview of the automatic road boundary generation problem using LIDAR point cloud data as a primary input. We provide extensive research of publically available datasets suitable for the task and create a custom pipeline converting Nuscenes, one of the most well-known Autonomous Driving related datasets, into a dataset suitable for future work in High-Definition mapping research. We re-implement a variant of Feature Pyramid Network for the purpose of the road boundaries generation and experiment with the training process, proposing several minor improvements to the architecture. We build a morphologically-based refinement process of the CNN output raster data into the form suitable for vector operations, including the denoising process and a simple custom-designed module for road boundaries gap completion. Finally, we evaluate our converted dataset using a complete pipeline and present visualized final results together compared to contemporary works.

Our obtained results do not measure up to the state-of-art research. We still extract valuable lessons about the research goal, and our general conclusion is that the architecture will be much more successful with higher quality and quantity dataset. We confirm the viability of the road boundaries gap completion module and hope that our codebase can be used in future research on the topic.

The future work can be focused on two primary directions. First, the absence of a high-quality dataset for the task is a significant obstacle not resolved in the thesis. A solution would be to congregate several existing datasets and manually fix their quality problems or create a completely separate dataset from scratch. Second, implementing and developing more complex Machine Learning based solutions, such as Generative Adversarial Networks, will help alleviate dataset problems, but only to a degree.

7 Acknowledgements

This thesis was funded by Autonomous Driving Lab, a collaboration project between the University of Tartu and Bolt. I want to give sincere appreciation to my supervisors, Naveed Muhammad and Dmytro Fishman, for guiding me through the process of this long, grueling, and fulfilling project. Special thanks to Edgar Sepp for helping me to translate Abstract to the Estonian language.

References

[AFZAW99] M-F Auclair-Fortier, Djemel Ziou, Costas Armenakis, and Shengrui Wang. Survey of work on road extraction in aerial and satellite images. 1999.

[BMH+18] Min Bai, Gellert Mattyus, Namdar Homayounfar, Shenlong Wang, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Deep multi-sensor lane detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3102–3109. IEEE, 2018.

[BT76] Ruzena Bajcsy and Mohamad Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, (9):623–637, 1976.

[CBL+20] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[CLS+19] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[CWL+19] Yiping Chen, Shiqian Wang, Jonathan Li, Lingfei Ma, Rongren Wu, Zhipeng Luo, and Cheng Wang. Rapid urban roadside tree inventory using a mobile laser scanning system. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(9):3690–3700, 2019.

[Dis20] Engineering Discoveries. Different types of curbs and dimension, Jan 2020.

[FS75] Herbert Freeman and Ruth Shapira. Determining the minimum-area encasing rectangle for an arbitrary closed curve. *Communications of the ACM*, 18(7):409–413, 1975.

[GBC10] Costantino Grana, Daniele Borghesani, and Rita Cucchiara. Optimized block-based connected components labeling with decision trees. *IEEE Transactions on Image Processing*, 19(6):1596–1609, 2010.

[GKM+20] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S Chung, Lorenz Hauswald, Viet Hoang Pham,

Maximilian Mühlegg, Sebastian Dorn, et al. A2d2: Audi autonomous driving dataset. *arXiv preprint arXiv:2004.06320*, 2020.

[GLSU13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[HCG+18] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018.

[HMLU18] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Hierarchical recurrent attention networks for structured online maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3417–3426, 2018.

[IK88] John Illingworth and Josef Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.

[KOA92] Takio Kurita, Nobuyuki Otsu, and N Abdelmalek. Maximum likelihood thresholding based on population mixture models. *Pattern recognition*, 25(10):1231–1240, 1992.

[KR16] Eric W Koch and Erik W Rosolowsky. Filfinder: Filamentary structure in molecular clouds. *Astrophysics Source Code Library*, pages ascl–1608, 2016.

[KTW86] Takeo Kanade, Chuck Thorpe, and William Whittaker. Autonomous land vehicle project at cmu. In *Proceedings of the 1986 ACM fourteenth annual conference on Computer science*, pages 71–80, 1986.

[KUH+19] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 perception dataset 2020. `https://level5.lyft.com/dataset/`, 2019.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[LHM+19] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Shenlong Wang, and Raquel Urtasun. Convolutional recurrent network for road boundary extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9512–9521, 2019.

[Lit17] Todd Litman. *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[Lit21] Zach Little. Seeing the future: How argo lidar changes the self-driving game, May 2021.

[LTZG17] Shaoshan Liu, Jie Tang, Zhe Zhang, and Jean-Luc Gaudiot. Computer architectures for autonomous driving. *Computer*, 50(8):18–25, 2017.

[LVC+19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[LZH+18] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach, Md E Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 751–766, 2018.

[Ma20] Lingfei Ma. Road information extraction from mobile lidar point clouds using deep neural networks. 2020.

[MH10] Volodymyr Mnih and Geoffrey E Hinton. Learning to detect roads in high-resolution aerial images. In *European Conference on Computer Vision*, pages 210–223. Springer, 2010.

[MJD88] David M McKeown Jr and Jerry L Denlinger. Cooperative methods for road tracking in aerial imagery. In *Proceedings of the 1988 DARPA IUS Workshop*, pages 327–341, 1988.

[MLBS97] Helmut Mayer, Ivan Laptev, Albert Baumgartner, and Carsten Steger. Automatic road extraction based on multi-scale modeling, context, and snakes. *International Archives of Photogrammetry and Remote Sensing*, 32(Part 3):106–113, 1997.

[MLU17] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017.

[MM99] Songrit Maneewongvatana and David M Mount. It's okay to be skinny, if your friends are fat. In *Center for geometric computing 4th annual workshop on computational geometry*, volume 2, pages 1–8, 1999.

[MPLN17] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[MZWLS14] Javier A Montoya-Zegarra, Jan D Wegner, L'ubor Ladickỳ, and Konrad Schindler. Mind the gap: modeling local and global context in (road) networks. In *German Conference on Pattern Recognition*, pages 212–223. Springer, 2014.

[PMGC19] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9552–9557. IEEE, 2019.

[RFM08] Xiaofeng Ren, Charless C Fowlkes, and Jitendra Malik. Learning probabilistic models for contour completion in natural images. *International journal of computer vision*, 77(1-3):47–63, 2008.

[SAE16] SAE. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. *SAE International,(J3016)*, 2016.

[Sen26] Milwaukee Sentinel. Phantom auto'will tour city. *The Milwaukee Sentinel*, page 4, 1926.

[SH16] Heiko G Seif and Xiaolong Hu. Autonomous driving in the icity—hd maps as a key challenge of the automotive industry. *Engineering*, 2(2):159–162, 2016.

[SKD+20] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[Ske18] Jean-Paul Skeete. Level 5 autonomy: The new face of disruption in road transport. *Technological Forecasting and Social Change*, 134:22–34, 2018.

[SMR97] Carsten Steger, Helmut Mayer, and Bernd Radig. The role of grouping for road extraction. In *Automatic Extraction of Man-Made Objects from Aerial and Space Images (II)*, pages 245–256. Springer, 1997.

[SS18] Peter Slowik and Ben Sharpe. Automation in the long haul: Challenges and opportunities of autonomous heavy-duty trucking in the united states. *The International Council on Clean Transportation*, 2018.

[SSGK14] Nathan Silberman, Lior Shapira, Ran Gal, and Pushmeet Kohli. A contour completion model for augmenting surface reconstructions. In *European Conference on Computer Vision*, pages 488–503. Springer, 2014.

[SSSM13] Martin Shepperd, Qinbao Song, Zhongbin Sun, and Carolyn Mair. Data quality: Some comments on the nasa software defect datasets. *IEEE Transactions on Software Engineering*, 39(9):1208–1215, 2013.

[THKS88] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1988.

[TK17] Eric R Teoh and David G Kidd. Rage against the machine? google's self-driving cars versus human drivers. *Journal of safety research*, 63:57–60, 2017.

[Way17] Waymo. Building maps for a self-driving car, Aug 2017.

[WMZS15] Jan Dirk Wegner, Javier Alexander Montoya-Zegarra, and Konrad Schindler. Road networks as collections of minimum cost paths. *ISPRS Journal of Photogrammetry and Remote Sensing*, 108:128–137, 2015.

[XGD+17] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[YK15] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[ZLG+17] Dawei Zai, Jonathan Li, Yulan Guo, Ming Cheng, Yangbin Lin, Huan Luo, and Cheng Wang. 3-d road boundary extraction from mobile laser scanning data via supervoxels and graph cuts. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):802–813, 2017.

[ZPRA99] Dale Zimmerman, Claire Pavlik, Amy Ruggles, and Marc P Armstrong. An experimental comparison of ordinary and universal kriging and inverse distance weighting. *Mathematical Geology*, 31(4):375–390, 1999.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Dmytro Zabolotnii**,
*(*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,

   Automatic Road Boundary Extraction for High Definition maps,
   *(*title of thesis)

   supervised by Naveed Muhammad and Dmytro Fishman.
   *(*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.

Dmytro Zabolotnii
13/05/2021