

TALLINN UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

Liisi Soots 166764 IVSM

PAVEMENT MAPPING USING CONVOLUTIONAL NEURAL NETWORKS

Master's Thesis

Supervisors: Kristjan Korjus
PhD

Juhan-Peep Ernits
PhD

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Liisi Soots 166764 IVSM

**KÕNNITEE KAARDISTAMINE
KONVOLUTSIOONILISTE
NÄRVIVÕRKUDEGA**

Magistritöö

Juhendajad: Kristjan Korjus
PhD

Juhan-Peep Ernits
PhD

Tallinn 2018

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Liisi Soots

07.05.2018

Abstract

Autonomous vehicles are becoming more common. Therefore, with the increase of autonomous cars and robots, the need for high definition maps is increasing. Starship Technologies [1] is building robots that drive on the pavement autonomously. In the current thesis, the author makes one of the mapping processes in the company more automatic by detecting pavement edges from the images using a trained convolutional neural network. The topic was chosen to reduce the time spent on manual pavement area marking. The process is labour intensive for the company and automation is crucial for coping with the increases in demand. The hypothesis was that the process of mapping the area of pavement can be made automatic by using convolutional neural networks.

The goal of the work was to build a neural network model that can detect pavement edges. The work included creating training data from the information available, designing and training the neural network model and evaluating the results. In this thesis, previous manual annotations of pavement area were transformed to be used as the training data. The neural network model was built and evaluated with different evaluation methods. The three evaluation methods were: mean-square-error, a custom metric that took into account the subjectivity in the annotations, and a manual evaluation, where the author looked at the images and gave a rating to the output.

The significance of the prediction by the neural network was compared with a simple model which always predicts the mean result of the data. Different convolutional neural network models and simple models were compared based on the mean-square-error to choose the best model. Additionally, the results were evaluated by a person who looked through images and rated them as good or not.

The research showed that the neural network model worked quite well and exceeded the precision of the simple constant model. Based on the research, it can be concluded that the model would currently need a human to check the results, but this already results in an efficiency gain for the company and the model can be made better with further development.

In conclusion, the proposed convolutional neural network can be used for sidewalk edge detection.

This thesis is written in English and is 69 pages long, including 7 chapters, 43 figures and 6 tables.

Annotatsioon

Kõnnitee kaardistamine konvolutsiooniliste närvivõrkudega

Autonoomsed sõidukid on muutumas tänavapildis aina tavalisemaks. Koos autonoomsete autode ja robotite arvu kasvuga on tõusmas nõudlus väga täpsete kaartide järele. Starship Technologies [1] toodab roboteid, mis suudavad iseseisvalt sõita kõnniteel. Autor teeb ühe kaardistamise protsessi selles ettevõttes automaatsemaks. Töö teema valiti, et vähendada kõnnitee kaardistamiseks kuluvat aega. See protsess võib muutuda pudelikaelaks kui ettevõtte kasvab ning aina rohkem kaarte on vaja. Hüpootees on, et seda protsessi on võimalik automatiseerida kasutades konvolutsioonilisi närvivõrke.

Lõputöö eesmärk on ehitada närvivõrk, mis suudaks tuvastada kõnnitee ääri. Töös kogutakse treenimiseks vajalikud andmed, muudetakse need sobivale kujule närvivõrkude jaoks, treenitakse saadud andmetega närvivõrgu mudelit ning hinnatakse tulemusi. Selles töös kasutatakse treeningandmete koostamisel eelnevalt koostatud kõnnitee ala annotatsioone, mis on muudetud treenimiseks õigele kujule. Närvivõrgu tulemust mõõdetakse erinevate meetoditega: keskmine ruut viga, isetehtud meetrika, mis võtab arvesse subjektiivsust annotatsioonides, ja visuaalselt tulemusi hinnates. Visuaalselt hindas pilte inimene, kes on kogunud sellel alal ning andis binaarse hinnangu kas tulemus on hea või halb.

Töös kontrollitakse, kas närvivõrgu tulemus on parem kui lihtne mudel, kus alati ennustatakse kõnnitee äärte keskmist väärtust. Erineva sügavusega närvivõrgu mudeleid ja ka lihtsat mudelit võrreldakse keskmise ruut veaga, et teada saada kõige paremini töötav mudel. Lisaks hinnati tulemusi käsitsi vaadates läbi pilte ja hinnates neid kui hea või halb.

Uurimine näitas, et närvivõrgu mudel töötas väga hästi ja oli parem kui lihtne mudel. Selle uurimise tulemusena saab öelda, et praegune mudel ei automatiseeriks täielikult käsitsi annoteerimist, aga lihtsustaks seda. Mudel suudab kõnnitee ääri ennustada, kuid praegust lahendus tulemuse peaks kindlasti inimene üle kontrollima.

Kokkuvõttes, konvolutsioonilisi närvivõrke saab edukalt kasutada kõnnitee äärte tuvastamiseks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 69 leheküljel, 7 peatükki, 43 joonist, 6 tabelit.

List of abbreviations and terms

3D mapping	Pavement position and width mapping from images extracted from the video feed of robot driving
AWS	Amazon web services
CNN	Convolutional neural network
CSV	Comma-separated values
GNSS	Global Navigation Satellite System. The term includes GPS, GLONASS, Galileo, BeiDou and other regional systems
GPU	Graphics processing unit
HD map	High definition map
MSE	Mean-squared error
Node	A connection point of ways, used in 3D mapping
ReLu	Rectified linear unit function
UTF	Unicode transformation format
Way	A section of the path marked on frames during 3D mapping to indicate where the pavement is

Table of contents

1 Introduction	14
1.1 The company used in the case study	15
1.2 Mapping.....	16
1.2.1 Map creation.....	16
1.2.2 Localisation	18
1.2.3 3D mapping	19
1.3 Research goal.....	20
1.4 Approach	21
1.5 Selection of tools	21
2 Related Work.....	23
2.1 Road Detection	23
2.2 Usage of convolutional neural networks	25
3 Background.....	26
3.1 Neural Networks.....	26
3.2 Convolutional neural networks.....	27
3.3 Coordinate system changes	30
4 Approach for pavement detection.....	32
4.1 Considered methods	32
4.2 Chosen approach.....	33
5 Preparation of data.....	35
5.1 Data gathering.....	35
5.2 Calculating intersection points	37
5.3 Training dataset creation	40
5.4 Problems in data	42
6 Neural Network model construction.....	43
6.1 Data transformation	43
6.2 Augmentation	44
6.3 Model structure.....	45
6.4 Training process	47

7 Analysis of the results	48
7.1 Testing	48
7.1.1 Validation and testing during model building	48
7.1.2 Custom accuracy metric	48
7.1.3 Evaluating against simple model.....	49
7.1.4 Visual evaluation	51
7.1.5 Comparing to real process scenario.....	53
7.2 Issues with training data	54
7.3 Results	58
7.4 Future work.....	67
Conclusions	68
References	70

List of figures

Figure 1. Starship Technologies' robot	15
Figure 2. Localisation map	17
Figure 3. Map pieces	17
Figure 4. Image robot sees while driving	18
Figure 5. Multiple front cameras attached.....	18
Figure 6. Map nodes and ways	19
Figure 7. Mapping tool image	20
Figure 8 Neural network structure.....	26
Figure 9 Neuron.....	27
Figure 10. Filter example	28
Figure 11. CNNs structure.....	29
Figure 12. Image while 3D mapping	33
Figure 13. Image algorithm sees.....	33
Figure 14. Calculating intersection points	34
Figure 15. Path smoothing and construction.	34
Figure 16. Data point relations	35
Figure 17. Correct node coordinates calculation from a vector	36
Figure 18. Putting all data together	37
Figure 19. World to robot coordinate changes	38
Figure 20. Robot to camera coordinate changes.....	38
Figure 21. Camera to screen coordinate changes	39
Figure 22. Help points to calculate intersection points on an image.....	39
Figure 23. Dataset creation process	40
Figure 24. Histogram of right side values	41
Figure 25. Histogram of left side values.....	41
Figure 26. Point coordinate changes.....	44
Figure 27. MSE with different depths	45
Figure 28. Custom accuracy metric with different depths.....	45
Figure 29. MSE with different epochs	46

Figure 30. Custom accuracy metric with different epochs	46
Figure 31. Neural network structure.....	46
Figure 32. Custom accuracy metric with different thresholds.....	49
Figure 33. Constant model MSE scores in different cities	50
Figure 34. Constant model and CNN model MSE scores compared in different cities .	51
Figure 35. Image used in the visual evaluation	52
Figure 36. Annotation values and prediction values compared.....	53
Figure 37. Smoothed out prediction values	53
Figure 38. Bad input examples	54
Figure 39. Annotations and context.....	55
Figure 40. Images algorithm struggles with.	58
Figure 41. Places where algorithm gets confused	59
Figure 42. Annotations and predictions with the MSE score and frames	65
Figure 43. Annotations and predictions with the MSE score and frames	66

List of tables

Table 1. The number of training data points from each city	41
Table 2. Constant model and neural network model comparison	51
Table 3. Manual evaluation results.....	52
Table 4. Images from the annotation tool compared to the training data.....	55
Table 5. Images used in training compared to images after bug had been fixed.....	57
Table 6. MSE score and manual evaluation compared	60

1 Introduction

Autonomous vehicles are more and more common in the streets. Multiple companies like UBER [2], Lyft [3], Tesla [4], Waymo (a subsidiary of Alphabet Inc.) [5] are building cars with autonomous driving capability. Autonomous vehicles need to know very precisely where they are on the streets, where they can drive, where are the crossings, which are static and which dynamic objects. Computing the location of the car and locating objects takes a significant amount of effort at the time of driving. Static objects, for example, are lamp posts and trees; humans, pets and other vehicles, on the other hand, are dynamic. It is much safer and efficient to compute some of the parts beforehand that do not change so much - static objects. During driving comparison is made between the surroundings and prior knowledge, in that way, it can be estimated more accurately what might be dangerous for the vehicle and what not. The prior experience is represented on a map.

All of the companies building autonomous vehicles need to have some kind of custom map. The maps are custom because they depend on the sensors used in the car. Using these maps means that the vehicle does not have to rely on GNSS (Global Navigation Satellite System) or lane markings to navigate on the streets. Sensors are used to detect the moving and static objects. The static objects are located on the map so that while driving the sensors can focus on the dynamic objects like cars and people.

With the increase of developing autonomous cars, the demand for high-definition maps (HD maps) is increasing. Chief operating officer of Baidu has said that the HD maps in China will be a much more significant business than search business in Baidu today [6]. A couple of companies are focusing on providing the maps for companies building autonomous vehicles. HERE is co-working with German car companies to enable hands-free driving [7], MapBox is developing maps used by Tesla and UBER [8]. Additionally, TomTom is competing with them, having partnerships with Baidu and NVidia [9].

The maps built are three-dimensional maps, having more information than regular maps used for car navigation. The maps contain information about the location of driving lanes,

traffic lights, crossings, traffic signs, and pavements. Waymo has said, that they can locate the car with 10-centimetre accuracy [10]. Waymo uses lidar to get a detailed view of the world around it and radar to understand the speed and distance of an object. Besides, they use cameras for visual information.

Similarly, to the cars, the robots that drive around autonomously, need to have a map. Starship Technologies [1] is an autonomous robot delivery company who builds their maps so that the robots can operate on the sidewalks on their own. Like autonomous car software, it has all the information around itself gathered by the sensors on the robot. While most of the HD maps are built from the standpoint of the car road, these maps look at the world from the pavement.

1.1 The company used in the case study

Starship Technologies is a company developing self-driving delivery robots for last-mile delivery. The company was founded in 2014 by Ahti Heinla and Janus Friis. It has offices in Tallinn, London, Hamburg, Helsinki, Washington and Redwood City. The firm is working with Just Eat, Hermes, Dominos, Omniva, Wolt, Doordash, and many other organisations [1].

The robots have six wheels, drive with a speed of 6 kilometres per hour and have multiple cameras along with sensors to help them get around in the streets (Figure 1). The robots are autonomous, and when needed, they can be operated remotely.



Figure 1. Starship Technologies' robot.

One of the key reasons to choose a robot delivery is its price. Robot delivery could be cheaper than regular delivery. To make the delivery as cheap as possible one of the options is to build robots that can drive on their own. The robot would know where it is; it would avoid obstacles and could reach to the client without any human help, i.e. be autonomous.

1.2 Mapping

In this paragraph, the map creation and robot localisation process in Starship Technologies is described. To make the robot drive autonomously multiple steps need to be taken: the robot needs to have a map, the robot needs to know where it is on the map and the robot should know where it is allowed to drive. As these steps are crucial for autonomy, Starship Technologies builds their own maps, and all of the mentioned steps represent processes in the company.

1.2.1 Map creation

Starship Technologies creates custom maps for the robots. The main features on the maps are lines. The maps are 3-dimensional, containing the location of lines in the world. The lines are detected based on the change of colour in the image acquired by the cameras: house edges, pavement edges, poles, but also shadows and cars (Figure 2). The precise position of each line is detected by using multiple sequential images from a drive of a robot. Each time the robot passes a new place, the info about that place is added to the map, so next time the robot already has features to refer to and can drive in that place autonomously.



Figure 2. Localisation map. The map consists of lines that robot sees around it in a certain location. The green lines in the picture are mapped lines and the dots are mapped light sources used for night-time navigation.

Thus, the map changes a little bit each time new data is added to it (Figure 3). The coordinates in the map are relative, not absolute. It might mean that the map does not match precisely with the real world but interprets what robot sees. Such map is accurate enough to enable autonomous driving. The relative coordinates in the map can be adjusted if a new piece of data is added to it and new connections are made between parts in the map.

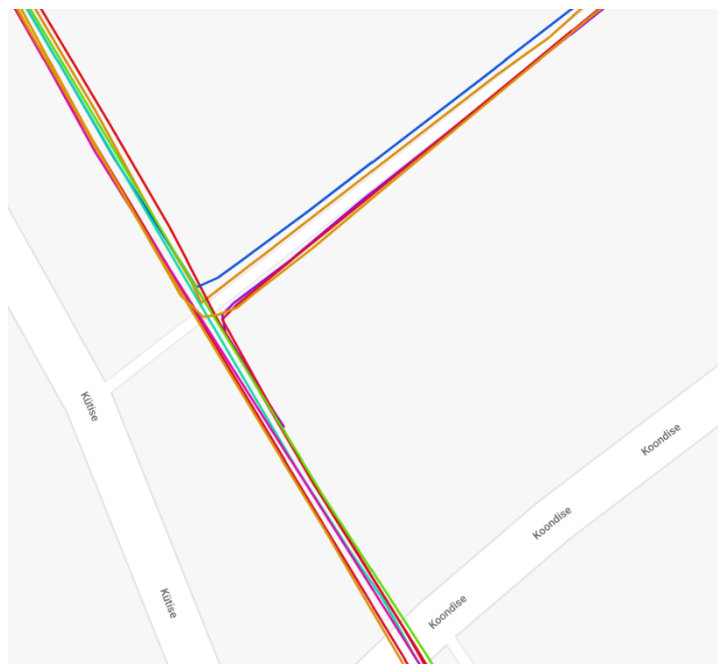


Figure 3. Map pieces. The map consists of sections added to the map. Different colours refer to different pieces.

1.2.2 Localisation

The second step of making the robot drive autonomously is to make it understand where it is. The GNSS is not accurate enough to position the robot precisely on the sidewalk. Therefore, the line-maps are used to know the robot position more accurately (Figure 4, Figure 5). The robot locates itself by matching the lines it sees around with the lines in the map. When it finds a match, the robot can start driving on its own. While the robot is moving, it continuously looks for matches to know precisely where it is.



Figure 4. Image robot sees while driving. The red lines are lines that are in the map and that robot also sees. These are used for localisation. The blue lines are lines that are in the map but the robot does not see them. The black lines are lines robot sees but are not in the map.



Figure 5. Multiple front cameras attached. All the cameras in the robot are used for the robot localisation.

1.2.3 3D mapping

The last precondition for making the robot drive on its own, is to mark where it can go and where it should not. This process is called 3D mapping. The procedure involves marking the area where the robot can drive on the image, in other words marking where the pavement or crossing is and marking the width of the sidewalk.

The 3D mapping process is the one automated in this paper by trying to detect the pavement edges from the images and marking the correct width and position of the sidewalk. Currently, the marking is done manually using tools built in-house. The tools use nodes, ways and different way types to describe the pavement (Figure 6). Nodes are points on a map, and ways are straight lines connecting two nodes. The ways can have different width according to the pavement and different types: sidewalk, crossing, driveway and car road, depending on where the way is marked.

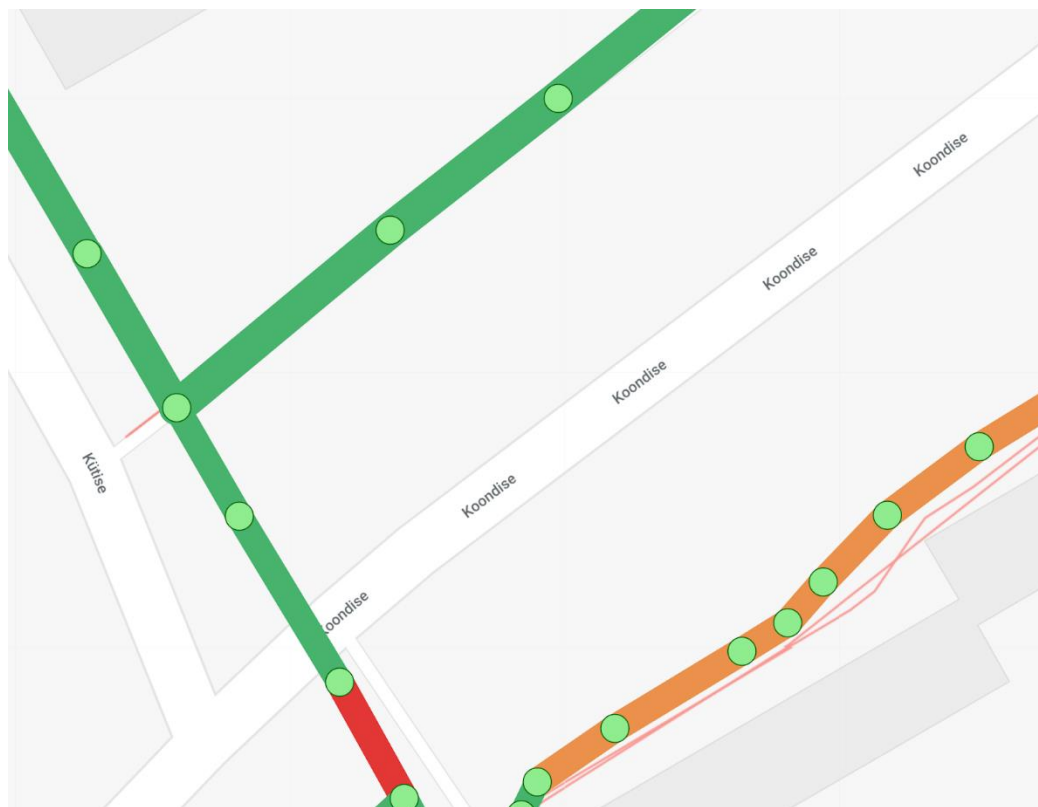


Figure 6. Map nodes and ways. Nodes are green circles and ways are rectangles between nodes. Different colour in ways indicates different way type. Green is the sidewalk, orange is car road and red is crossing. The pathway is annotated to the robot's front-camera images (Figure 7). The employees are marking the pavement centre points with nodes and drawing ways between them. With the help of the images, the way width is adjusted according to the pavement width.



Figure 7. Mapping tool image. Image of how the pavement area is currently marked on the images. The green area is marked as pavement area. The annotation consists of a node that is yellow in the front and ways that are green.

While the map creation and localisation processes are automated, the 3D mapping is currently done manually. With the growth of the company, it can become a bottleneck if the process stays the same.

1.3 Research goal

The thesis aim is to evaluate if a neural network model can be used to automate path annotations in the pavement mapping process. The further aim is to automate the path marking process and reduce the human effort with the help of the algorithm involving convolutional neural networks. The developed algorithm should reduce the cost of the 3D mapping and make it quicker.

In the scope of the thesis, the algorithm is not expected to eliminate human effort completely but to reduce it in simple mapping cases. During this investigation, the model should detect the width and direction of the sidewalk.

1.4 Approach

Neural networks [11] will be used for mapping pavements. In this work, the mapped edges will be estimated from the images. The training of the neural network might reveal the need for changes to the training dataset. The procedure can go back and forth between constructing the data in a new way and evaluating the outcome.

The data for training the model will be taken from previously manually annotated paths. At first, the available information has to be researched and understood: what info is available, what is essential and how it can be used to train the algorithm. Then the data will be assembled and prepared. It includes putting the data into the correct format that is understandable to the model. When the data is in the right form, the metrics can be defined to measure and compare different neural networks.

The evaluation of the results will be carried out based on the defined metrics and testing data that was separated from the training data during the learning process. One way to measure the accuracy is to compare the width of the previously marked road and the predicted width. A mean-square error will be used to understand how differently the algorithm places the path from the manual annotation. During the measuring, it is also relevant to take into account that the algorithm should have better results than a simple model. A simple model, for example, can always predict the mean value of the variables - always predicting constants.

The algorithm is considered successful if it can reduce the work of the mapping team and reduces the number of manual annotations needed. If the algorithm does not speed up the process, then the results can still be considered meaningful by highlighting the problems in the model.

1.5 Selection of tools

Python [12] will be used as a scripting language for data preparation. Keras library [13] with Tensorflow [14] will be used for working with neural networks. Keras is an open source API written in Python to develop neural networks. Keras was chosen as it allows to build neural networks simply and quickly. Tensorflow with Keras was chosen as it is widely used tool for building deep neural networks and other teams at the company have

had positive experiences with it. Amazon AWS [15] will be used for the computation of the model.

Pandas [16] and Numpy [17] packages are used to transform the data during neural network training. OpenCV [18] is used to prepare the data. Matplotlib [19] and Pillow [20] are used for creating diagrams and images with annotations. SciPy [21] is used for post-processing of the neural network model results.

2 Related Work

3D mapping can be automated using road detection. Road detection is essential in computer vision applications for autonomous driving cars and robots. In this paragraph previously used road detection algorithms are introduced, and other usages for convolutional neural networks are discussed.

2.1 Road Detection

The accurate width of the pavement and direction detection is quite an uncommon issue. However, there has been a lot of research in car road detection as the self-driving cars are using this functionality and the need for detecting road accurately is increasing. There are multiple approaches for identifying the street, using: texture, colour, edges and the road vanishing point [22].

Texture-based road detection approaches divide the image into multiple pieces and for each section assign if it belongs to the road or not based on the texture. This method uses segmentation to understand the road and non-road areas on an image, looking at the texture of the piece [23]. Colour-based road detection methods use RGB colour space to find the road [24] [25].

Texture-based and colour based road detections are dependent on the lighting and do not work very well with shadows, illuminations and artificial lights. To solve the latter issue Alvarez et al. [26] converted an image into illuminant-invariant feature space and then applied a model-based classifier to label pixels. This approach is better as the previous ones but still might fail due to poor lighting conditions.

One leading approach for detecting the edges of the road is vanishing point estimation [27] [28]. A vanishing point is a point at which receding parallel lines viewed in perspective appear to converge. This approach detects converging lines to vote for the most likely vanishing point. The lines are usually identified from the edges of the road.

In [27], the image was cut into multiple pieces, and for each section, a likelihood of being a road was given based on the texture values in the area. The method works very well in most of the cases but keeps struggling when a vehicle goes up or down the mountain, and there is not enough space to see and predict vanishing points.

A similar approach to the vanishing point detection has been used before for road detection where the authors computed illumination invariant images to segment the road even if the photos have a lot of shadows on them [29]. The technique relies on detecting the road edge directions to vote for the most predictable vanishing point. The issue with this kind of algorithms used in the previous work is that there might be vanishing points that are equal. So, in this case, the appearance-based road recognition is added to make the algorithm better by looking at the road colour. The algorithm works very well if the triangular road shape is visible, but it fails in more difficult situations when only one edge of the road is seen, or there are shadows on the sidewalk.

Another way of determining if a segment of the image belongs to the road or not is to use already categorised parts of the road. Similarity-based road detection compares image segments with already known road images and based on the similarity decides where the road edges are. In some cases, geometry is used to detect the road: the algorithm tries to find straight and left/right curve lines. The approach uses probability map for the road but instead of vanishing point uses different predefined road lines: soft left/right turn, sharp left/right turn, straight road and also the same things having another car in front of the autonomous vehicle [30].

As there are multiple approaches for the road detection, there are also combined models where vanishing point data is united with geometric information [29] or horizon lines [31]. The algorithms use vanishing point, then the image is segmented, and each segment is given a probability of belonging to the road. After that, the final probability map is created by combining all the vanishing point information, the section information and colour-based likelihood. The mixed versions work more accurately than just using vanishing point or road colour based estimation.

The previously described methods are not used in this work as they are older than convolutional neural network methods, and during the last years neural networks have surprised with their performance. Additionally, the methods focus on road texture, colour

but pavement structure is not so constant as the car road structure. Also, the training data creation for those methods needs more manual work, and the current data is easier to be used for the CNNs.

2.2 Usage of convolutional neural networks

Convolutional neural network (CNN) architectures are used for detection, recognition and segmentation tasks. CNNs [32] have changed the way of pattern recognition as the features do not have to be preselected but the selection of features is part of the process. The CNN approach performs well on the image recognition tasks [11] [33] and is a new phenomenon having a surprising accuracy in different classification tasks [34].

The CNNs have been trained to describe images [35] [36]. For humans it is hard to understand how the neural networks work, Google has done experiments on convolutional neural networks to understand better what is inside them. For example, they taught a neural network to create meaningful images from random points [37]. Spotify also uses CNNs to recommend music [38].

The CNNs have also been used for road detection [39] [40], for example, one approach [39] uses CNNs, then uses colour analysis to detect if the pixel belongs to the road or not and finally Naïve Bayes is applied to recognise the road areas. Another CNN approach also uses prior spatial information [40].

3 Background

Deep learning and computer vision methods will be used during this project. In this paragraph background and theoretical information are given about the methods.

3.1 Neural Networks

The human brain inspires neural network design. The network consists of layers of neurons. The layer has multiple neurons and produces an output that is input for another neurons layer. Each neuron calculates its output based on the inputs. The network has an input layer where the input data is given in for the model as arrays and output layer, which produces the result. Layers between the input and output layers are called hidden layers (Figure 8). Dense layer is a layer where each of the neurons is connected to each neuron in the next layer [41].

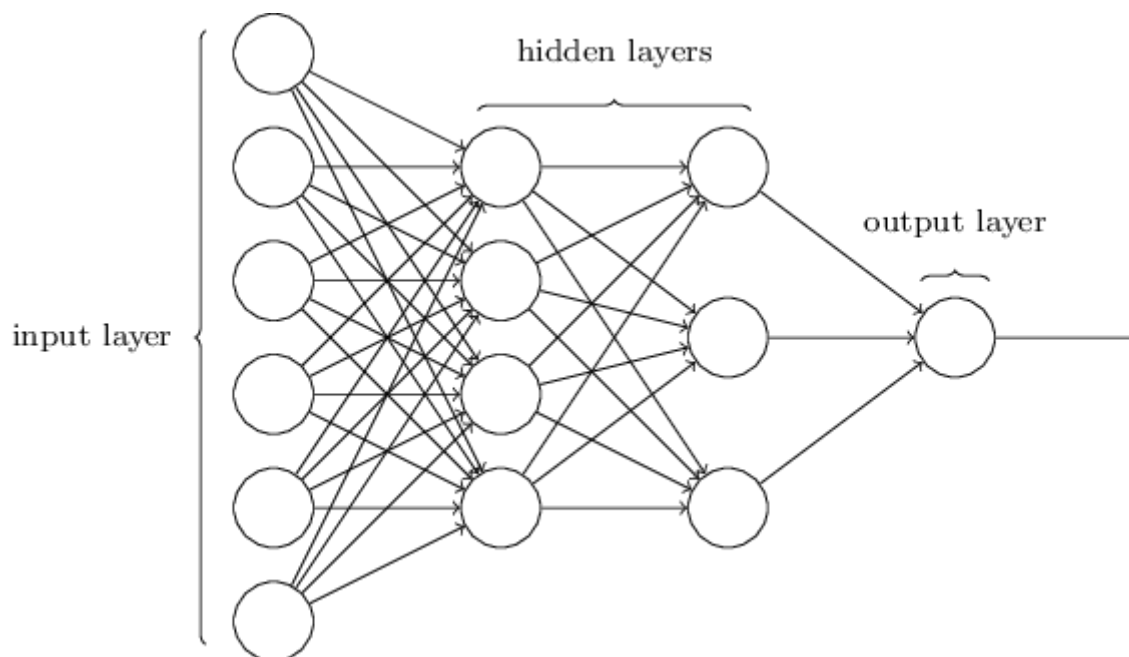


Figure 8 Neural network structure [42]. A neural network has multiple layers: input, hidden and an output layer. Output layer produces the final result.

Each neuron has inputs and based on them computes an output (Figure 9). Each input has a weight (w) that shows the importance of that input compared to other inputs in the same neuron. The neuron calculates an output based on function (1).

$$f(x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3) \quad (1)$$

The function f is non-linear and is called activation function. The function aims to bring non-linearity to the output. If an activation function is not used then, the neural network could only predict linear regression models. The activation function helps to learn more complex patterns.

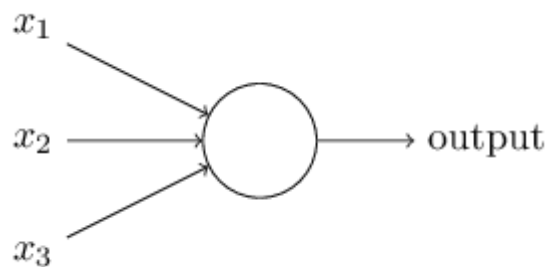


Figure 9 Neuron [42]. A neuron has multiple inputs and based on them computes an output.

An epoch in neural network context represents a number of times the neural network is trained with the same data. Batch is a set of input data feed to the neural network. Batch size represents how many input data records are fed to the neural network at once. All the training data is divided into batches, and if all the batches have been fed to the neural network, then one epoch is over.

Feedforward neural networks are networks where the output of one-layer is used as the input of the next layer. This means there are no feedback loops in the networks and data moves in one direction. Convolutional neural networks are feedforward neural networks, but recurrent neural networks use feedback loops.

3.2 Convolutional neural networks

Convolutional neural networks belong to the “deep learning” approaches. They are used for processing data that has a known grid-like topology. This kind of data is for example time intervals and image data. The convolutional network name indicates that the model employs a mathematical operation called convolution. “CNNs are neural networks that

use convolution in place of general matrix multiplications in at least one of their layers [43].”

Convolution is used to describe different concepts in math, however, in this work the concept of matrices is used. Convolution is an operation where a matrix is transformed by another matrix called kernel to a new matrix. In the example below (Figure 10), the kernel of size 3x3 is applied to the matrix, and the value of 42 is received. The answer is arrived at by multiplying all the values in the first matrix with the values in the second one, and all the values are added up. A kernel is a filter that is assigned to each pixel of the image. CNNs typically have sparse interactions which mean that the kernel is smaller than the input.

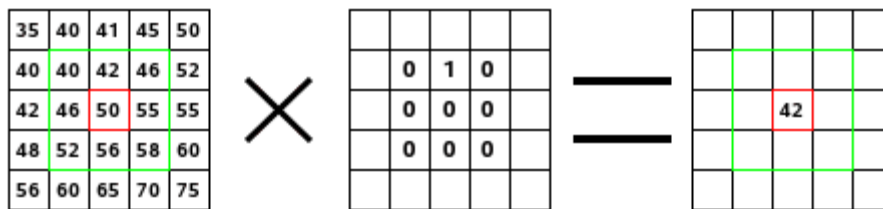


Figure 10. Filter example. The filter is added to the squares. The answer is got by multiplying all the number with the number in the filter and summing the squares up [44].

ImageNet Large Scale Visual Recognition Challenge is a competition held from 2010 where teams compete to get the highest accuracy on multiple visual recognition tasks. CNNs became widely used after 2012 ImageNet competition where Alex Krizhevsky dropped classificatory error from 26% to 15% [33]. Classification error shows the proportion of images that are classified incorrectly. It is calculated by dividing the falsely classified examples by the total number of examples.

CNNs consist of multiple layers (Figure 11) and acts similarly to the basic building blocks of the human brain – each neuron has a weight and decision point. Each layer output in CNN is the input for the next layer. The first layer in convolutional neural networks is a convolutional layer. Convolution averages out measures and the operation is defined as:

$$s(t) = (x \times w)(t) \tag{2}$$

The x refers to the input, the second argument (w) is a kernel, and the output is called feature map.

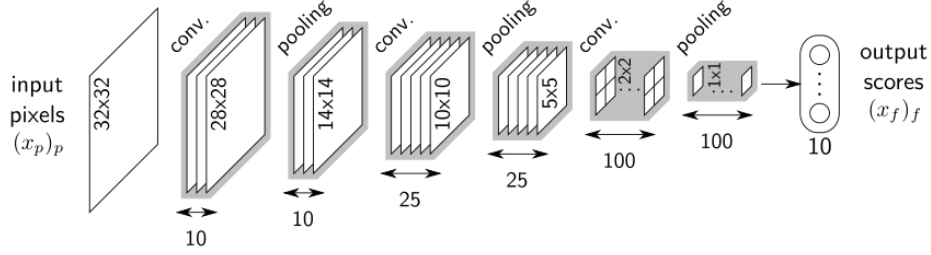


Figure 11. CNNs structure [45].

A classic CNN structure has three layers: convolution stage, detector stage and pooling stage. After each convolutional level, there is a detection level where the rectified linear unit function (ReLU) is used as an activation function (3). ReLU function is applied to all of the values in the input. ReLU is used to get rid of the vanishing gradient problem. Vanishing gradient problem means that neurons in the earlier layers learn much slower than the neurons in the later layers. This is a common problem when backpropagation is used. Backpropagation happens every time a new input has reached the end of the network, and all the neuron values are recalculated to adjust the weights.

$$f(x) = \max(0, x) \quad (3)$$

A pooling layer is used after a detector stage to downsample the data. The pooling is used to make the data space smaller and also to reduce overfitting by taking statistics of the nearby outputs. For example, max pooling takes the maximum output from the neighbour elements.

Dropout layers are used to avoid overfitting the model [46]. In the dropout layers, random neurons are dropped, and the rest of the network is trained again. This would give the model an effect that if some of the neurons are missing it still would learn to get the same result and would not learn to be dependent on only one neuron.

At the end of all the transformations in the layers, there is loss function, that is used to calculate how wrong the output currently is. The function would give a cost for each of the output depending on how far it is from the real values. The aim is to minimise the loss. For a regression problem often mean square error (MSE) is used as a loss function. Mean squared error measures the difference between the desired output and neural network prediction. The model aims to get the MSE as close to 0 as possible.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2 \quad (4)$$

Gradient Descent is an optimisation algorithm used in training neural networks. The algorithm tries to find the maximum or the minimum of a cost function. The function shows how to change x to improve the metric y [47].

3.3 Coordinate system changes

The global coordinate system is used for the maps, but the input for the neural network will be an image without the information where it is in the world. Therefore, the coordinates have to be changed from one system to another. A 3D point from the world has to be converted into a 2D point in the image. The input data has to be converted from the world coordinate system to screen coordinates.

The world coordinates are in 3D space and can be written down as formula (5), where the x' is the coordinate variable, the x , y and z are the coordinates in the world. x and y are crossed, and z represents the height and is always from the ground to up.

$$x' = (x, y, z) \tag{5}$$

Translation is changing the coordinate system origin, where I is a 3x3 identity matrix, the t is a 3x1 matrix that represents the translation of which direction the coordinates are changed (6).

$$x' = [I \mid t] \bar{x} \tag{6}$$

Additionally, to the transformation, rotation also changes the direction of the axes, where R is the 3x3 rotation matrix. The rotations are in the quaternions (7).

$$x' = [R \mid t] \bar{x} \tag{7}$$

For projecting the point to the image a calibrated camera is needed. A calibrated camera can be represented by the matrix (assuming the lens distortions have already been removed):

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Where f_x and f_y are the focal lengths, s encodes a possible skew between the sensor axes and c_x and c_y represent the centre coordinates.

Calculating 2D point from a 3D point is done with an equation:

$$X = K \cdot [R | t] \cdot x \quad (9)$$

Where X is the 2D point looked for, K is the camera matrix, R is the rotation matrix, t is translation and x is rotated 3D point coordinates in the form of $[x, y, z, 1]$ [43].

4 Approach for pavement detection

In the following section, the proposed method for solving the pavement detection tasks are considered, and alternative approaches are explained.

4.1 Considered methods

There are various ways of how to prepare the training data. The form of the data influences how good the neural network will be. The approach chosen predicts way intersection points with the lower edge of the frame. This method was selected as it involves predicting only two values from each of the images and due to that is quite easy. The technique could give predictions from each of the frames in the dataset, and it would be possible to throw away outliers after putting all the estimations together.

Alternative approaches considered involved predicting the position of a possible node ahead and the road edges near the node. This method was not chosen as it is hard to say how far forward the node should be estimated. The approach is also tricky in the corners where there is no road ahead, and the pavement is turning.

The second abandoned idea was to predict the road width and the position of the road ahead in 3D coordinates. This method is similar to the previously discussed one. Additionally, to the issues mentioned above, the approach would not be so intuitive and understandable for debugging in the images.

Another idea involved detecting the intersection points of the lower edge of the image and predicting the angle of the way, to see where it concentrates. This method had four numbers to predict, and due to that, it is harder than the chosen method.

An entirely different approach would be using segmentation [48], but it would need more precise data than there currently is. From the segmentation results, it would be harder to generate 3D paths and putting sequential frames together.

4.2 Chosen approach

The picked approach will use front camera images from the robot for detecting the intersection points of the pavement edges with the lower edge of the image. Low-resolution images will be used for the detection of pavement edges, and all the further examples in this work are using the original resolution images. The same image size is used currently in the process of 3D mapping. While the 3D mapping software allows looking at the front and side cameras of the robot at the same time (Figure 12) the algorithm will only consider the images from the front camera (Figure 13).



Figure 12. Image while 3D mapping. The centre camera and side cameras are put together. The green area marks the pavement and the yellow represents a node.



Figure 13. Image algorithm sees.

Based on the image and information where the way is, the way projection intersection with the lower edge of the frame is calculated (Figure 14). From the intersection points, the way edge coordinates can be calculated for each frame.



Figure 14. Calculating intersection points. The red points are calculated from current annotations. Right and left red points with the image are given to the algorithm as inputs.

Each of the frames in the dataset is 0.3s apart on average. The gap between the frames depends on the speed of the robot and whether it was moving or not. All of the images are overlapping with each other which means that there are multiple way widths and position estimations in the same area. From each of the image, way intersection points are calculated, and the odds are eliminated. Next, the smoothed path can be drawn by removing outliers and estimating the most likely position of the sidewalk. Finally, the nodes and ways, which are currently used in the system can be drawn on the predicted path (Figure 15). The red dots represent predictions, the crossed out dots are points removed as they do not align with other predictions. The bigger yellow circles are nodes, and the green rectangles represent the ways.

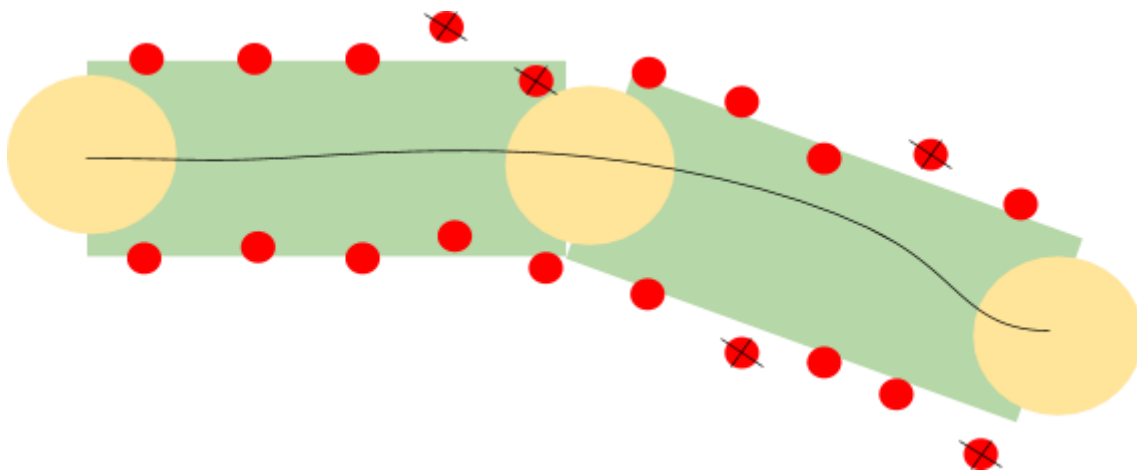


Figure 15. Path smoothing and construction. The red dots are estimations and some of them are crossed out as they are away from most of the points. The bigger yellow dots are nodes and green rectangles are ways.

80% of the 3D mapping work involves mapping sidewalks. In this thesis, we will focus on understanding the width and direction of the pavement but not the type of the way. The later will be left for the future work.

5 Preparation of data

In this paragraph the construction of data is described in detail, explaining what calculations need to be done and what filters are applied. Data preparation involves changing the representation of the data and creating the training dataset. The data is a crucial aspect of this work, and it determines the output. If the data is poorly constructed, then the predictions will not be precise.

5.1 Data gathering

Not all of the robot drives can be used for the dataset. The data can only consist of data points that are used for the map creation as these records have the exact location of the robot from the calculations done for the maps. The coordinates for the map are calculated based on the GNSS readings, odometry, and the lines robot sees itself.

Training data is calculated from multiple separate pieces: nodes, ways and precise location data (Figure 16). At first, all the information is queried from the database separately. All the nodes that have previously been 3D mapped will be used as input data for searching data points that can be used in the training dataset.

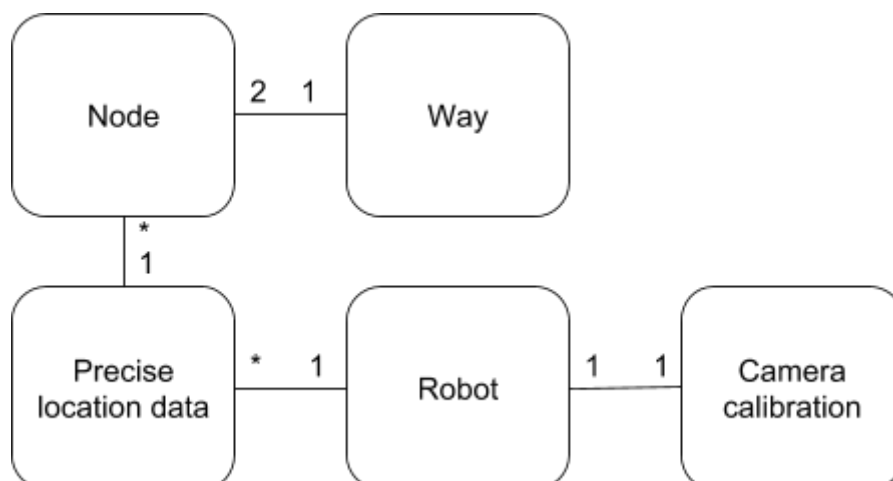


Figure 16. Data point relations.

As the map changes all the time, the correct node coordinates are not kept in the database. In the second step, the exact node coordinates have to be calculated for calculating the intersection points with the lower edge of the image. The node coordinates are not final when they are 3D mapped, but they change according to the map. In the database, the node does not have coordinates, but a vector that describes the node location and precise location data - timestamp of the robot from where the vector was drawn.

The vector never changes, only when the node is remapped to a different place. The position of the precise location might change when the map is modified. As the precise location position might change, the node position can be calculated from it with the help of the vector. The node position is a point where the vector from precise location point intersects with the ground (Figure 17).

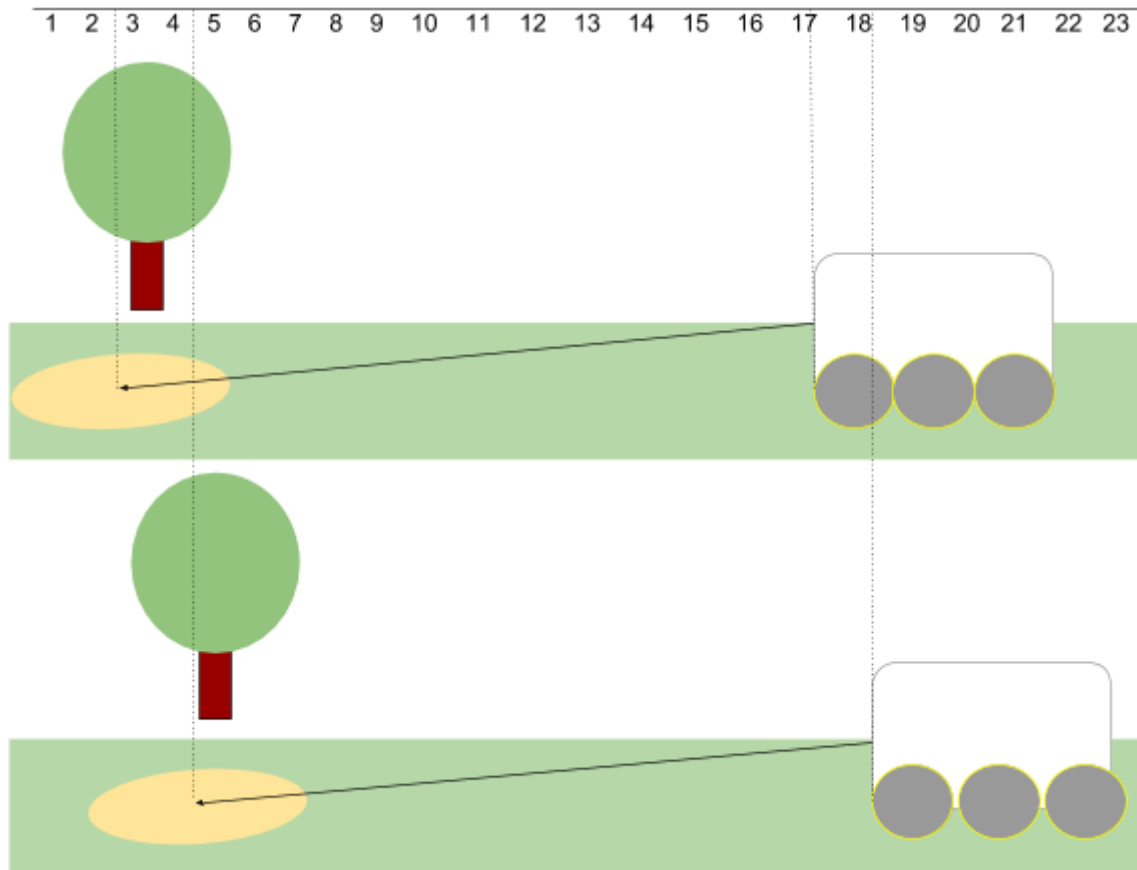


Figure 17. Correct node coordinates calculation from a vector. Precise localisation coordinates change (robot position), with that the node position also changes, but the vector remains the same.

In the next step, all of the previously gathered data is combined. Each of the precise location points needs to be associated with the node ahead and behind it and pavement width at its location. For making these relationships between data points, rectangles are created for each of the ways. The rectangle consists of way information and information about nodes it is connected to. For each of the robot location data point, it is checked to which of the rectangles the location belongs to (Figure 18). In other words, the algorithm looks for a rectangle which contains the localisation point and then creates data records with the precise location, nodes and pavement width. After all the data is bundled together, the intersection points with the lower edge of the image are calculated.

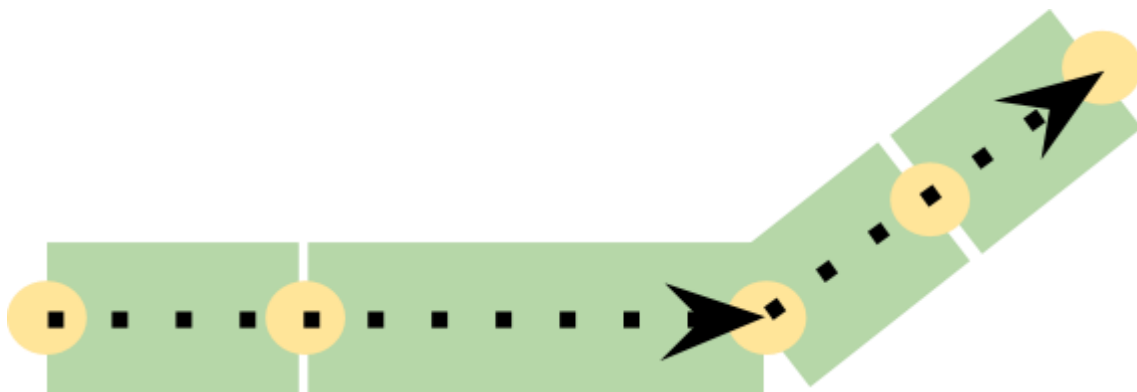


Figure 18. Putting all data together. The yellow circles are nodes, green rectangles are ways between nodes. The black dots are precise localisation points. From the picture it's seen what has to be found for one precise localisation point: node behind and ahead of the location and way width it is on.

5.2 Calculating intersection points

Each localisation point has the corresponding frame from the time the robot was driving. These images are used as an input to the model. The data will consist of images and both right and left intersection points.

To calculate the intersection points with the lower edge of the image, all the 3-dimensional coordinates have to be changed into 2-dimensional coordinates. For these calculations, the latest calibration data for robot cameras is used. The projections of the points are calculated in the order of the world to robot, robot to camera and camera to screen. In each of those steps, a calculation for the projection is performed.

In the world to robot step, the node and robot world coordinates are calculated into robot coordinates, in other words where the node is positioned from the point of view of the

robot. In that step, the coordinate axes are rotated, and the height of the map is changed to the height of the robot (Figure 19).

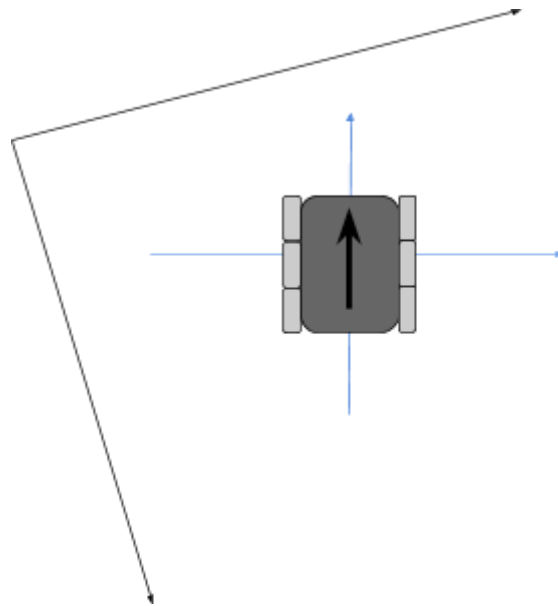


Figure 19. World to robot coordinate changes.

Next, the robot coordinates are projected to camera coordinates. Camera centre is higher than the robot centre. The coordinates are rotated to the angle camera is on the robot and the x- and y-axis are raised higher. The camera is positioned higher than the robot coordinates and is at its angle, so the coordinates have to be rotated a bit, and the height is changed (Figure 20).

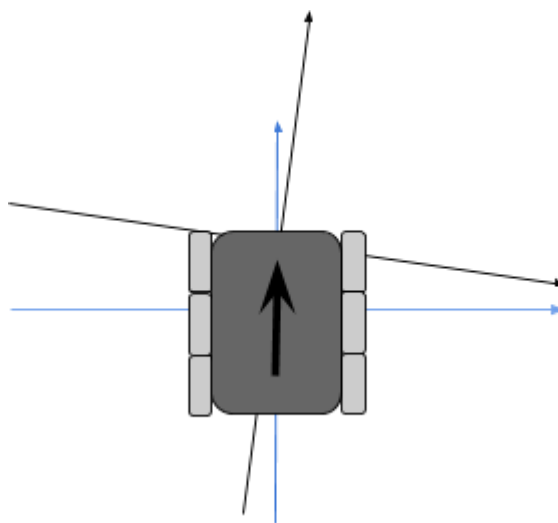


Figure 20. Robot to camera coordinate changes.

Lastly, the camera coordinates are converted into image coordinates, which means that the 3-dimensional coordinates are transformed into 2-dimensional ones. Function `project3dToPixel` from ROS [49] will be used for the transformation (Figure 21).

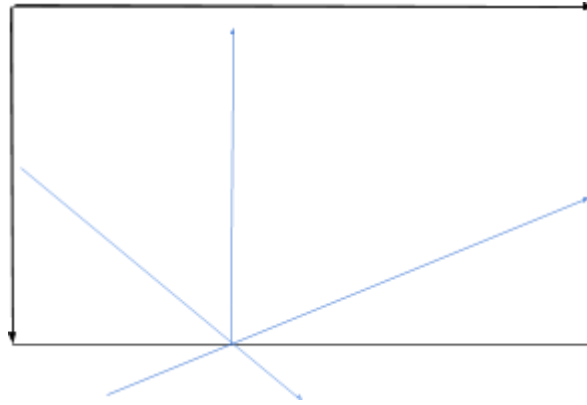


Figure 21. Camera to screen coordinate changes.

Node behind and ahead the robot has to be projected accurately on the image, to calculate the intersection points correctly. The node that is behind the robot cannot be projected very precisely. Therefore, help-points are calculated to project points only ahead of the robot on the image (Figure 22). Help-points (red points) are calculated in 3D space. First, the vector between two nodes is calculated, and then the cross-vector from the previous result is calculated. With the cross-vector and way width information, the corner points (purple points) of the way can be obtained and also the help-points coordinates can be gathered.

The corner and help-points are transformed into 2D space, and the point where the vector from purple point to red point intersects the lower edge of the image is the point used in training.

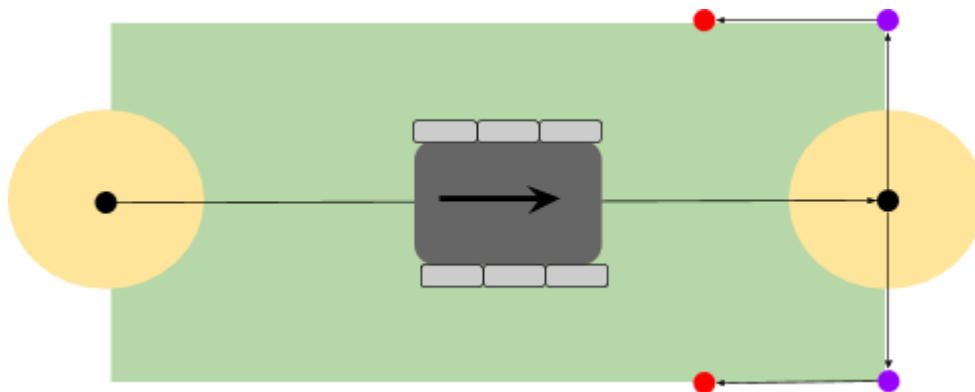


Figure 22. Help points to calculate intersection points on an image. The red points are the help points and the purple are the corner points. Black dots are the points whose location is already known.

5.3 Training dataset creation

The training data creation process consists of gathering the appropriate data from the database, downloading the image, calculating intersection points, making sure everything is right and removing invalid rows (Figure 23).

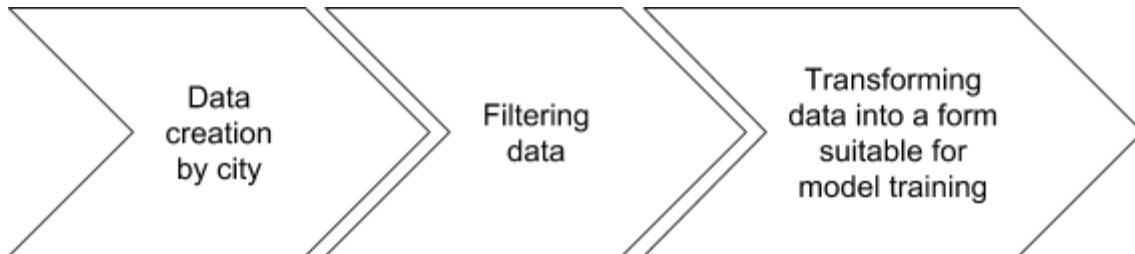


Figure 23. Dataset creation process.

Training dataset is created city by city to make all the calculation quicker. The city is divided into multiple pieces using Google Maps' tiles [50]. The tile information is already connected with the way information and can be used for dividing data into smaller pieces. During the data gathering process, each tile will have its corresponding CSV file containing this area's information. The division helps to make the process quicker and restarts more manageable as it reduces the need to recompute data records that have already been prepared.

The data gathered from the process above consists of images from the robots driving, marked by robot name and timestamp. For each "robot name and timestamp" there is a corresponding row in a CSV file. A row has info about the intersection points. Additionally, the node, way and robot location is in the data to help in the debugging process.

After the gathering process is over, all the tiles are joined to one big CSV file, and it is checked if all the data is in the correct format: each row has exactly one image and vice versa. Some of the records are removed due to the lack of information in the database or strange outputs. The outliers in the data were removed based on the previously calculated intersection points to remove the possible mistakes in the training dataset. Rows that had intersection points in the lowest or highest 5% were removed. The histograms of the remaining data can be seen in the image below (Figure 24, Figure 25) the filter point is calculated in the scale of the lower edge width of the image. Records were eliminated,

where calculated right side intersection point is less than 0.1817 or more than 1.9450, left side intersection point is less than -1.1160 or more than 0.5545.

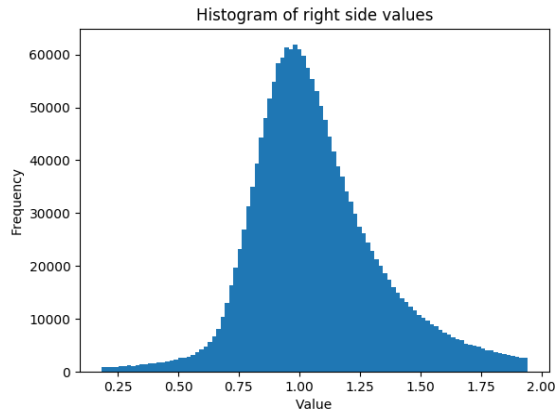


Figure 24. Histogram of right side values.

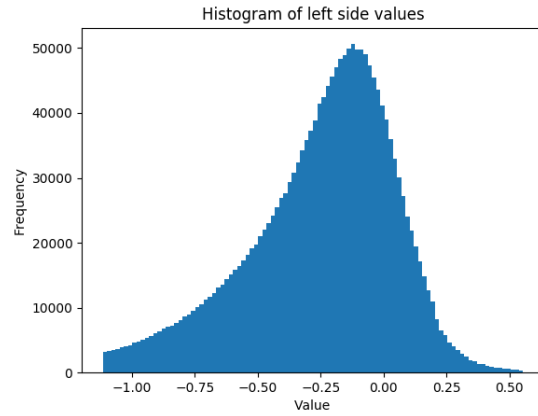


Figure 25. Histogram of left side values.

In addition to removing some data points that look wrong, night time data is excluded as it is used for creating maps for night-time driving but not for 3D mapping. The distinction of night time and daytime is made by the degree of the Sun at the current time the robot drove there. Astronomical dawn is used as a reference for day and night time as it mostly suits the mapping conditions and civil or nautical sunrises would throw away too much data that is useful and mappable. If the solar zenith angle is less than -18, then the data point is thrown away.

The training data will be stored in separate folders per city where each of the folders has one CSV file containing all the necessary information for the training. 15 different cities data is used, and in total there are 2003560 rows (Table 1).

Table 1. The number of training data points from each city.

City ID	Number of data points	City ID	Number of data points
1	108	9	101544
2	2886	10	103655
3	4268	11	146262
4	16936	12	217518
5	37867	13	218866
6	52012	14	233963
7	57067	15	744974
8	65634	Total	2003560

5.4 Problems in data

The training data might have mistakes in it. After filtering out strange data rows, there can also be mistakes in the scripts for producing the data, in the 3D mapping itself or due to robot driving up or down a curb or climbing a hill.

The problematic cases in the data are street crossings and street corners. Also sometimes 3D mapping is done differently due to poles, trees and bike roads; sometimes it is mapped further away from parking cars. The neural network does not have all of this context, and it might mean that the neural network model is not working as well in these cases. Although there are a lot of complicated situations, about 85% of the places are clear and should be easy to determine by the proposed automatic approach.

6 Neural Network model construction

In this paragraph, an overview of the neural network is given with the explanation why the model is built like it is.

6.1 Data transformation

At first, the data is read into the neural network script. The data is split into training data, validation data and testing data. Approximately 80% of the data is used for training 10% for validation and the last piece for testing. The data is not split randomly into different categories, but all the cities are assigned to one of them. Meaning that some of the towns are only used for training and others for testing or validation. The reason why the data is split like that is due to the data properties. The sequential frames in the dataset are very similar, and if there would be almost the same image in training and testing, then the results would be unfairly good. The model would then find the very identical image it was trained for and would know certainly what the correct answer is. This solution is the most conservative way for avoiding unexpectedly good results. The photos are shuffled before giving it to the neural network to get better results, and so that sequential images would not go to the neural network at once.

Training dataset is used to train the model; validation data set helps to tune the parameters in the neural network during training and the testing dataset is used to get the accuracy of the model on a completely new data.

The images in the dataset are in different sizes, but the neural network expects images in one fixed size. When reading the input to the neural network all the images are converted to the resolution 240x180 pixels. As these are colour images, then all the images have RGB values described in a list of length 3. Each of the input row has 240x180x3 attributes.

Also, the intersection points are recalculated from the pixel coordinates to relative normalised value in percent (Figure 26). The procedure is called normalisation. The recalculation is needed to unify the scale of the data.

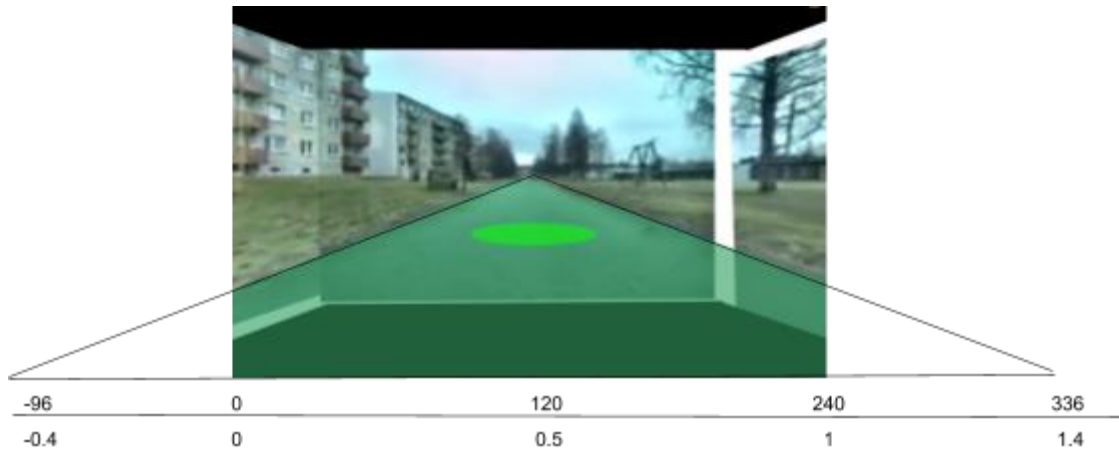


Figure 26. Point coordinate changes. The line above shows the real coordinates on the image and the lower line shows the percentual coordinates.

Currently used data set is so big that it cannot be loaded into the memory at once. Therefore, the author wrote a data generator that would only produce one batch of data at once. Each time neural network asks for a new batch the correct intersection points with normalisation and image arrays are calculated. The same procedure happens with the validation and testing data.

6.2 Augmentation

For the neural network model, it is always good to have more data and data with as much variation as possible [51]. Augmentation is a process of changing the input to be more diverse or producing more training data samples from the original data.

One augmentation approach considered was mirroring the image. Then the image arrays have to be reversed, and the intersection points have to be switched and recalculated. Mirroring is quite a popular augmentation method in the convolutional neural networks, but the author chose not to use it, as the mirroring would not give so much effect in this learning scenario. Mirroring is useful when only one-sided images are given for the network, for example in cat detection model only cats facing to the right but not to the left are in training dataset. In the current case, if all the images used had the same side towards the car road, then the mirroring would have been worth considering.

Another approach is to shift the image. The shifting augmentation is used in the current work as it gives more variety to the training data. Most of the images are quite similar as the robot is in the middle of the sidewalk and the edges are mostly at the same place. The

augmentation will help to increase the variations in the data. In this approach, new image arrays will be calculated, and the intersection points will be shifted according to the image. Shifts in pixels are chosen randomly, and the arrays describing images and intersection points are shifted according to it.

6.3 Model structure

The author approached the model construction experimentally, trying out different constructions and changing them based on how the loss function changed.

The issue solved is a regression problem, which means that the network will try to estimate a numeric value. The loss function used will be the mean square error. The measure is square of the difference between the estimation and ground truth.

In the architecture the smallest (3x3) convolution filters are used. The 3x3 filters are the smallest filter that captures the notion of left/right, up/down centre. After each convolutional layer max pooling layer is used with size (2x2). Each of the convolutional layers uses ReLu as the activation function. A dropout layer is used before the last convolutional layer with rate 75%. Finally, a dense layer with two outputs is used to get two predictions from the model. Similar architectures have been used in ImageNet competitions [52].

The author evaluated iteratively if adding depth to the neural network model improves the metric. The best depth was found to be five layers (Figure 27, Figure 28).

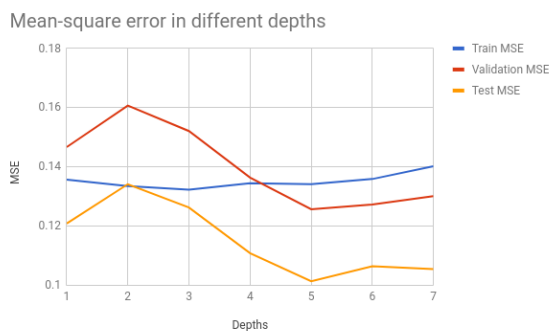


Figure 27. MSE with different depths.

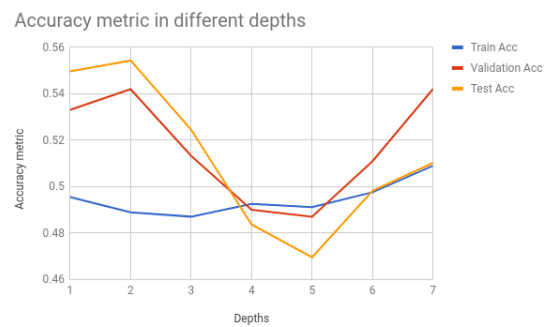


Figure 28. Custom accuracy metric with different depths.

Additionally, different epochs were analysed to get the epoch number from where the accuracy does not significantly improve anymore. It can be seen (Figure 29, Figure 30) that with the first epochs the accuracy increases significantly and from the 15th epoch onwards the mean-square-error and accuracy metric start to decrease at a lower rate. The chosen epoch rate in this work is 15. This rate was chosen as the neural network would run in reasonable time and running 30 epochs would take twice as more computing time but would not give a much better result. During the 15th epoch, the MSE score is 0.1194 and custom accuracy metric score is 0.4609.

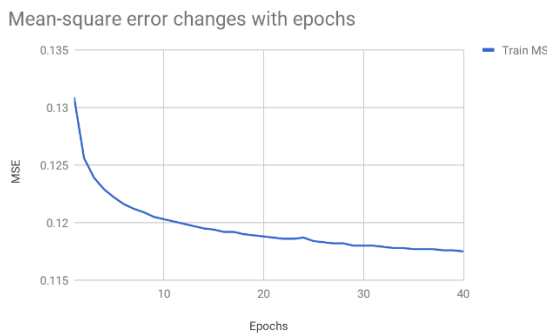


Figure 29. MSE with different epochs.

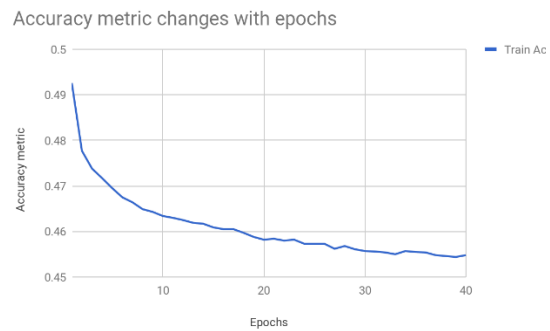


Figure 30. Custom accuracy metric with different epochs.

The final model has five convolutional layers, additionally maximum pooling layers and a dropout layer to reduce overfitting (Figure 31).

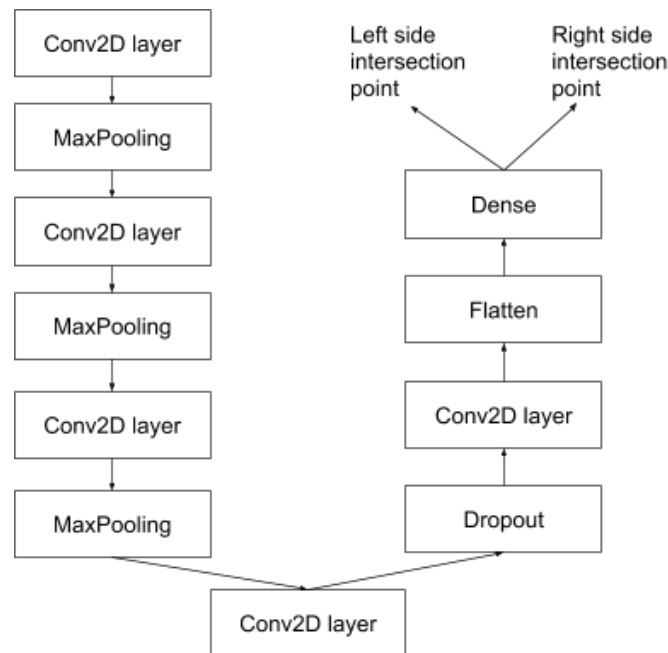


Figure 31. Neural network structure.

6.4 Training process

Neural Network training will be done in Amazon Web Services (AWS). AWS p2.xlarge with deep learning software [53] will be used for the training. All the data and scripts were moved there for the training using SSH [54]. The reason for using a server from Amazon is that the server has GPU capability, but the author otherwise does not have access to appropriately configured GPUs.

During the training, the author checked the memory usage with htop [55] and GPU usage on the server to evaluate if everything is working correctly. The models were trained with different cities distributed among training, testing and validation data, and compared afterwards.

During the training, images with prediction and annotation markings are created for evaluation processes. The visual evaluation of the images is not done on the server, and all the created images are sent back to the local computer for examination.

7 Analysis of the results

In this chapter, the analysis of the results of the current work will be provided. Additionally, tricky situations for the neural network will be described, and future work for finalising the automation of pavement mapping will be discussed.

7.1 Testing

The accuracy of the neural network model has to be evaluated to understand how well it is working. In this section, different evaluation methods used will be described with the results.

7.1.1 Validation and testing during model building

Testing of the model will be done based on the testing data that is separated from the training and validation data. The mean-square error will be used to evaluate how good the neural network model is. The smaller the MSE, the better the result. Cities are separated between testing, training and validation data. The training will be done on 13 cities, the validation will be done based on one city and testing based on another one. It is done this way so that there would not be sequential timestamp frames in training and testing. It is thus possible to avoid sequential frames of a run in a single street occurring in training and test data which would skew the outcome.

7.1.2 Custom accuracy metric

Additionally, to the mean-square error, a custom accuracy metric was constructed. The metric measures how many of the predicted intersection points are not near the annotated point. The metric correlates with MSE but gives a more understandable picture of what happens in the data.

The metric was built because the annotations used in this study are not perfect for neural networks, and also a small deviation from the annotated answer can be considered correct. The metric allows changing the limit from what point the threshold of being near is measured. If the difference between prediction and annotation was below the threshold,

then the result was correct otherwise incorrect. The threshold is given in proportion to the image size.

In the training process a threshold of 0.1 was used, this meant that if the intersection values were 0.1 off (10% of the image), they were accounted as correct. It appeared that with the final model half of the predictions were near the annotations. With threshold 0.2 already 75% of the annotations were near the annotation points (Figure 32).

Accuracy metric with different thresholds

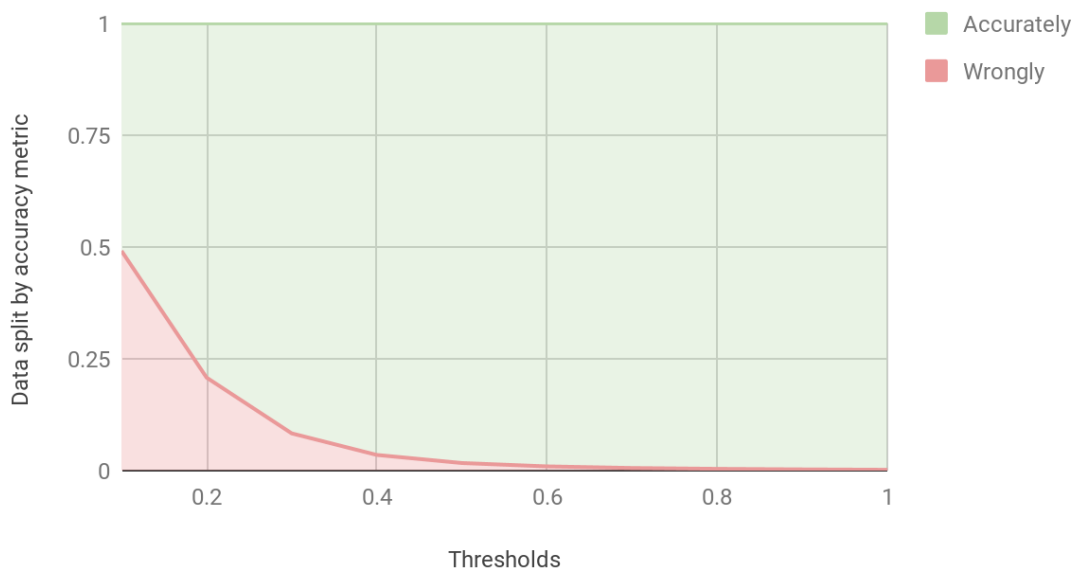


Figure 32. Custom accuracy metric with different thresholds.

7.1.3 Evaluating against simple model

The neural network model is only useful when it is better than a simple model like always assigning constant (i.e. average of the training data) as a result (from now on referred to as the constant model). In this work, the neural network model will be compared to the constant model.

Constants were found by calculating mean values for right-side and left-side intersection points. The mean values were -0.3043 for the left side and 1.1158 for the right side. The MSE score for all of the cities is 0.1784.

Additionally, custom right and left side constants were created separately for each of the cities, and the MSE scores were calculated. The results (Figure 33) show that better metrics are achieved when the city's constant is used. The MSE score for city's specific

constant model is 0.1541. The more significant takeout is that the cities have very different difficulty levels, meaning that for some cities the constant value estimation gives quite good MSE score (cities number: 1 and 9) and for some, it does not (cities number 8 and, 15). The cities that have low MSE score are more diverse and have different types of areas in them.

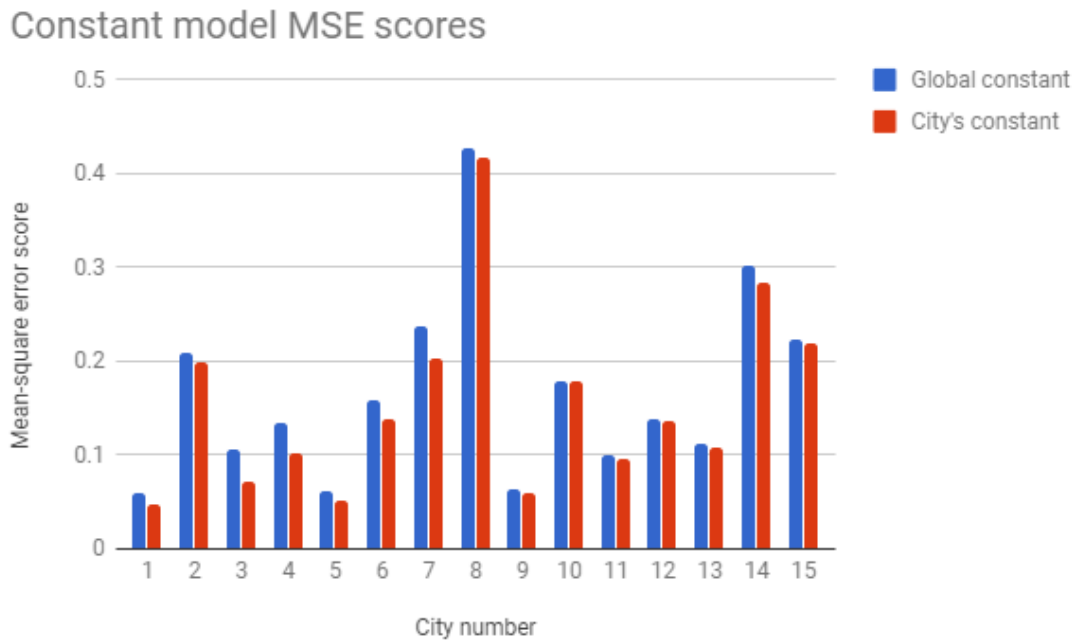


Figure 33. Constant model MSE scores in different cities.

In conclusion, the model should beat that MSE score. This would prove that the neural network model is better than randomly choosing numbers or producing constants and the model has learned something.

The comparison between the neural network and constant values showed (Figure 34) that in thirteen cities out of fifteen the neural network score beats the custom model or the results are very similar. In two of the cities, the neural network did worse than the custom model. The cities where the simple model was better than the neural network model have simple road structures, most of the pavement is very similar, with the same width. The simple model appears to be better in those cities because the neural network makes mistakes and it raises the MSE value higher.

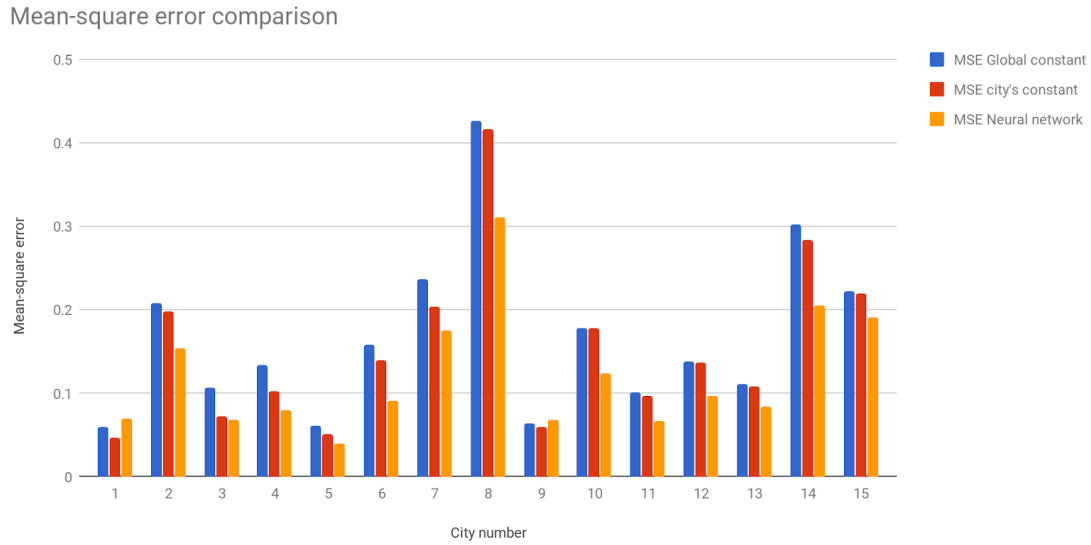


Figure 34. Constant model and CNN model MSE scores compared in different cities.

To sum up, the neural network got a smaller MSE score than city's specific constant model (Table 2). This means that the neural network model is more accurate than using the simple model.

Table 2. Constant model and neural network model comparison.

Type of model	Mean-square-error score
Neural network	0.1194
City's specific constant	0.1541

7.1.4 Visual evaluation

Mean square error is not the best validator of the quality of the model because the annotated data has mistakes in it and the annotations are not perfect. There might be other possible solutions for marking the pavement. Therefore, the author looked through the images generated by the constant model and images generated by the neural network to get another perspective of the models' performance.

The visual evaluation is performed on images where the predicted and annotated path is marked (Figure 35). In the visualisation not only the intersection points are marked, but lines are drawn to make the evaluation easier. The lines in the image are drawn between the intersection points and a point in the image centre.



Figure 35. Image used in the visual evaluation. The green is annotations and red is neural network model predictions. The destination point is picture middle point to help visualise where the sidewalk is going.

In the visual evaluation, it can be seen if the model predicted correctly with the information it had. The neural network does not have context around the image, and due to that, the network might predict pavement differently than the annotations, but the result is not wrong *per se*.

45 000 images were looked through during the visual evaluation. All the images were taken from the same city. The procedure involved counting the total number of images in the beginning, looking through all the images and deleting the ones not correct and lastly counting a total number of images in the end. The results showed that the constant model did not work as well as neural network model as it had a lot more mistakes (Table 3).

Table 3. Manual evaluation results.

Type of model	Number of images inspected	% of correct predictions by visual inspection
Neural network 1	14600	94.97
Neural network 2	6098	93.56
City's specific constant	14236	77.23
Training data	9396	86.20

Based on this evaluation it can be said that neural network model works better than the constant model. In the constant model 77% of the images were correct but 94% of the neural network model predictions were accounted as correct. From this evaluation it can also be seen that the annotations of the training data are not have very accurate (86%) and has even lower accuracy metric than the neural network model. This means that the neural network has learned from the training data that is not perfect but still inferred what is expected.

7.1.5 Comparing to real process scenario

Comparing the accuracy of only one image will not give the real result of the process as the sequential images will be learned and results linked together to form a path. Therefore, it would be fair to compare the results of sequentially attached images with the production maps

At first, the annotation data was compared with the neural network predictions (Figure 36). The graph consists of sequential intersection points over time. From the graph, it can be seen that the annotations of the right side seem to jump. This kind of jumps should not be there and is the evidence that there are mistakes made in the preparation of the training data.

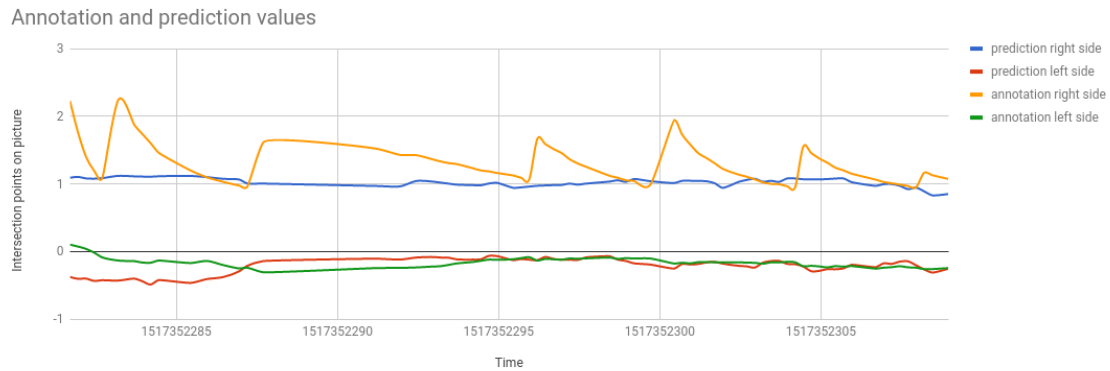


Figure 36. Annotation values and prediction values compared.

In the next step, the predicted intersection points will be smoothed out as there should not be vast differences between the sequential points. After the smoothing, a more realistic path was received (Figure 37). The paths width does not change a lot in the neural network model and also does not change in the real map either.

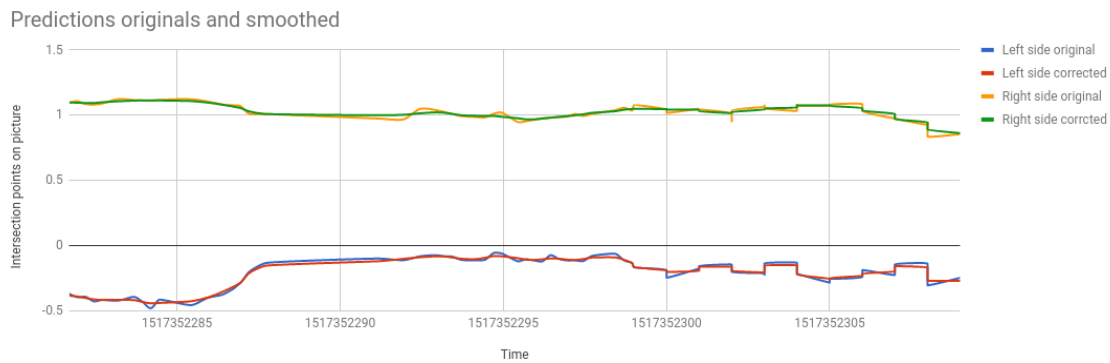


Figure 37. Smoothed out prediction values.

7.2 Issues with training data

During the training and evaluation process, some problems in the training data were discovered. It appeared that mean-square error compared to annotations is not the best indicator if the neural network works well or not. In many cases, the annotations are one, usually the best, solution for the frame. However, there might be other correct solutions for the frames.

Additionally, there are multiple cases where the training data is inaccurate (Figure 38). The errors can arise from different sources: wrong annotations, wrong calculations to intersection points and problems in the map. All those cases influence the MSE score, and therefore also some other evaluation metrics were used.



Figure 38. Bad input examples. The green lines are drawn from the intersection points to the image centre. 1) Not the whole pavement mapped. 2) Marked too wide. 3) Not marked whole pavement. 4) Marked too wide.

While the annotations were made the context of the map and surroundings was known. Therefore, the annotations could seem strange but are correct as there was something outside of the image that influenced the annotation (Figure 39). The neural network does

not have that kind of context and would mark the pavement area very wide. Due to that the MSE score and custom accuracy metric would be unreasonably high.



Figure 39. Annotations and context. In the first image it is understandable why the annotation is done like it. In the next image it is not so obvious. If there is no context then it is not understandable why the pavement width is marked so narrow.

As there seemed to be issues in the data some of the examples were compared with the annotations in the 3D mapping tool (Table 4). The results showed that the transformation from the annotations to the training data had some issues: some of the intersection points were changed.

Table 4. Images from the annotation tool compared to the training data.

Images from the annotation tool	Images from the training data

Images from the annotation tool



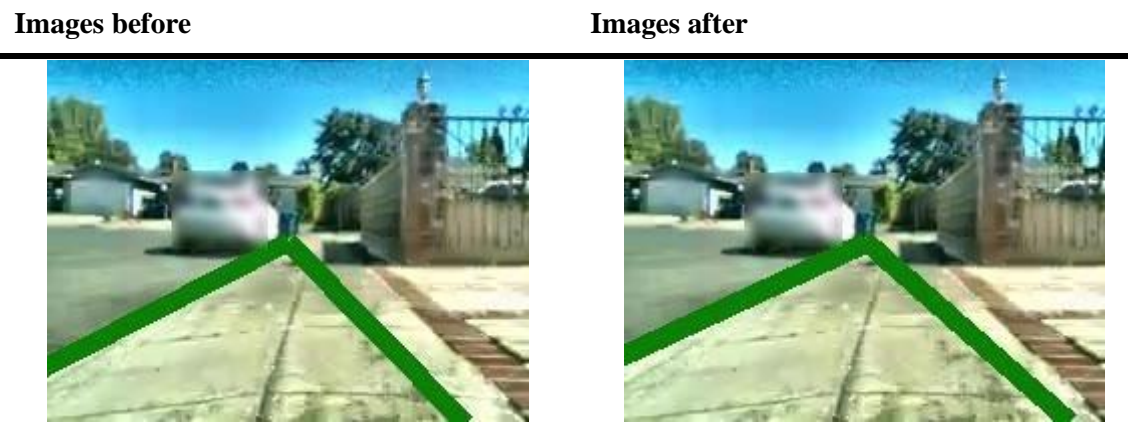
Images from the training data



During debugging at the evaluation phase an attempt was made to understand where the strange training data errors come from, as there are no such errors in the annotations. The bug was in the training data creation process where it was not taken into account that different robots have different camera resolutions. This mistake changes the intersection points a little. In the table below the before and after images with the corrected resolution can be seen (Table 5). In this thesis, the training data will not be regenerated, and it will be left for the future work.

Table 5. Images used in training compared to images after bug had been fixed. The green lines are drawn from the calculated intersection points to the centre of the image. In the left side column are images made when the bug was in the system and the right side images are the same frames when the code was correct.

Images before	Images after
	
	
	
	



7.3 Results

As an overall result it can be claimed that the neural network model is better in predicting than constant values. The result was demonstrated by manual evaluation, mean-square-error score and when looking at the neural network and constant model predictions. The model shows to be ready for further development and integration in the daily mapping process. The model works currently in the simple cases but needs a human checking when first implemented. The algorithm does not have the context of its surroundings and does not work very well on corners, curbs and ramps (Figure 40).



Figure 40. Images algorithm struggles with. Red lines are from neural network model predictions to the image centre and the green are training data points to image centre. The image number 6 is correct after all the curb climbing and turning corner, but it can be seen from the images that the curb climbing (1, 2) and turning (3, 4, 5) might not be so accurate.

Also, the model might be confused about people, temporal objects, shadows, sun and rain (Figure 41). All those issues can be solved, e.g. by using a smoothing algorithm on the results obtained by the neural network model.

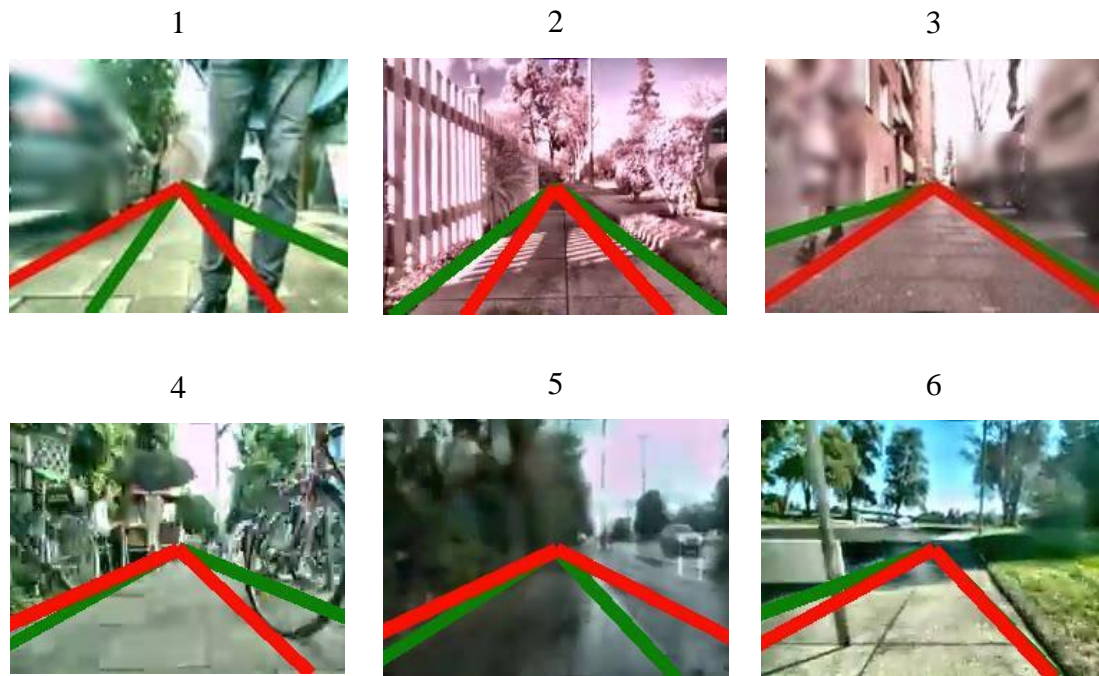


Figure 41. Places where algorithm gets confused. Red lines are from neural network model predictions to the image centre and the green are training data points to image centre. 1, 3, 4, 6) humans, obstacles ahead, 2) shadows 5) rain.

One of the biggest struggles in this work was understanding if the model works or not. The mean-square-error that was used as loss function in the model and as one of the metrics did not give a fair score for the predictions. To summarise the causes of the difficulties, an overview will be given of the problems in annotations, how it influenced the MSE score and what is the result of manual evaluation (Table 6). From these examples, it can be seen that the MSE score might not reveal the real accuracy of the predictions.

Table 6. MSE score and manual evaluation compared. Red lines are from neural network model predictions to the image centre and the green are training data points to image centre.





Image	MSE score	Manual evaluation result
	0.00645	The annotation is excellent, and the neural network model result is good.
	0.02938	Annotation could be slightly more to the right. The neural network model result is better than the annotation.
	0.03739	Annotation is good. Neural network model results are also good
	0.06618	Annotation little bit off, marked wider than it is in reality, neural network model also predicts it a little bit wider than it should be

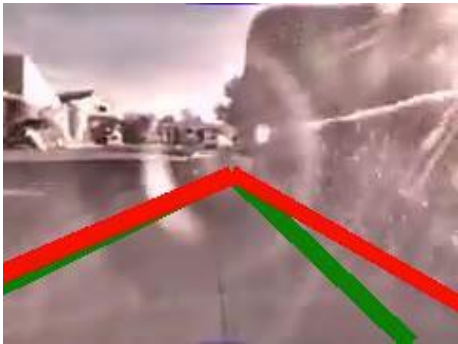



Image	MSE score	Manual evaluation result
	0.07976	Due to sun hard to understand precisely the annotations and predictions but both of them seem correct
	0.11875	Annotation is correct, also it is understandable why the prediction is a little bit different in this photo. In this case, it seems that the annotations are different due to the context that the annotation image does not have.
	0.18207	Annotation correct, neural network prediction wrong. Neural network confuses the gutter with the pavement.
	0.19690	Annotation left side is off, right side is correct. The neural network results are different. It predicts the left side better than the annotation, but the right side is off.




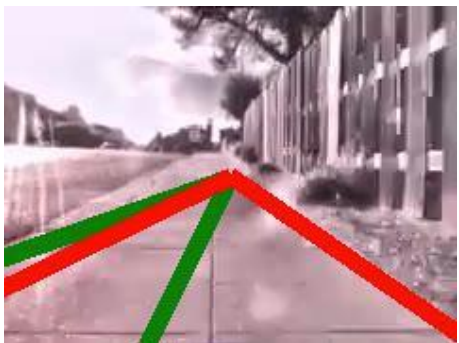


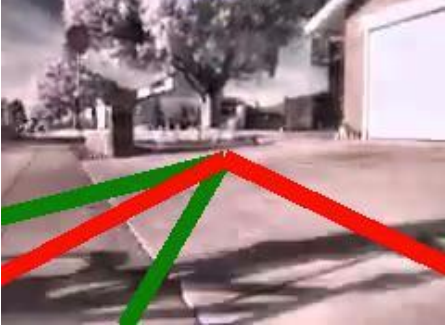
Image	MSE score	Manual evaluation result
	0.29415	Annotations right side off and left side is correct. Neural network model left side little bit off, and the right side is perfect.
	0.35370	Annotation correct, it seems it is influenced by the previous knowledge. Neural network model result also acceptable, does not have the context of the previous images. In this case, the model could be made better with more of those cases in the training dataset
	0.60776	Annotation correct, neural network prediction off
	0.62120	Annotation is wrong. Neural network prediction is perfect.

Image	MSE score	Manual evaluation result
	0.65891	Annotation is different from neural network model prediction but is also correct
	0.86148	Annotation is off. Neural network model result is correct.
	1.56034	Annotation is correct. Neural network model predicts very differently, but the mistake is understandable.

To sum up, wrong training data and the variance in results, that can be considered correct, influence the MSE score significantly. By looking the resulting images and analysing them, it can be seen that despite some problems in the training data, the neural network results are surprisingly good. In most of the cases, the neural network can predict where the sidewalk is and by combining the results of sequential images a more precise position of the sidewalk can be computed.

In the images below (Figure 42, Figure 43) it can be seen how the predictions and annotations change during driving. The examples here are taken from two different cities. In the first example, it can be seen that there seem to be mistakes in the training data, but the neural network gives a stable result. The second example also shows how the

annotations and predictions differ, but from the images, it can be seen that the neural network result is good.

Based on these results the neural network fails in predicting the edges of the path after each 10th image. After smoothing the result would be more accurate. Currently, the proposed neural network can be used for pre-mapping an area, and an employee can check the result of the neural network and make corrections if needed. This will already be quicker than the current process. The evaluation if the whole 3D mapping process can be completely automated with a new model and new features remains for the future work. Although most of the cases in the mapping can be automated there still will be situations that happen rarely or are difficult and cannot easily be mapped by the model.

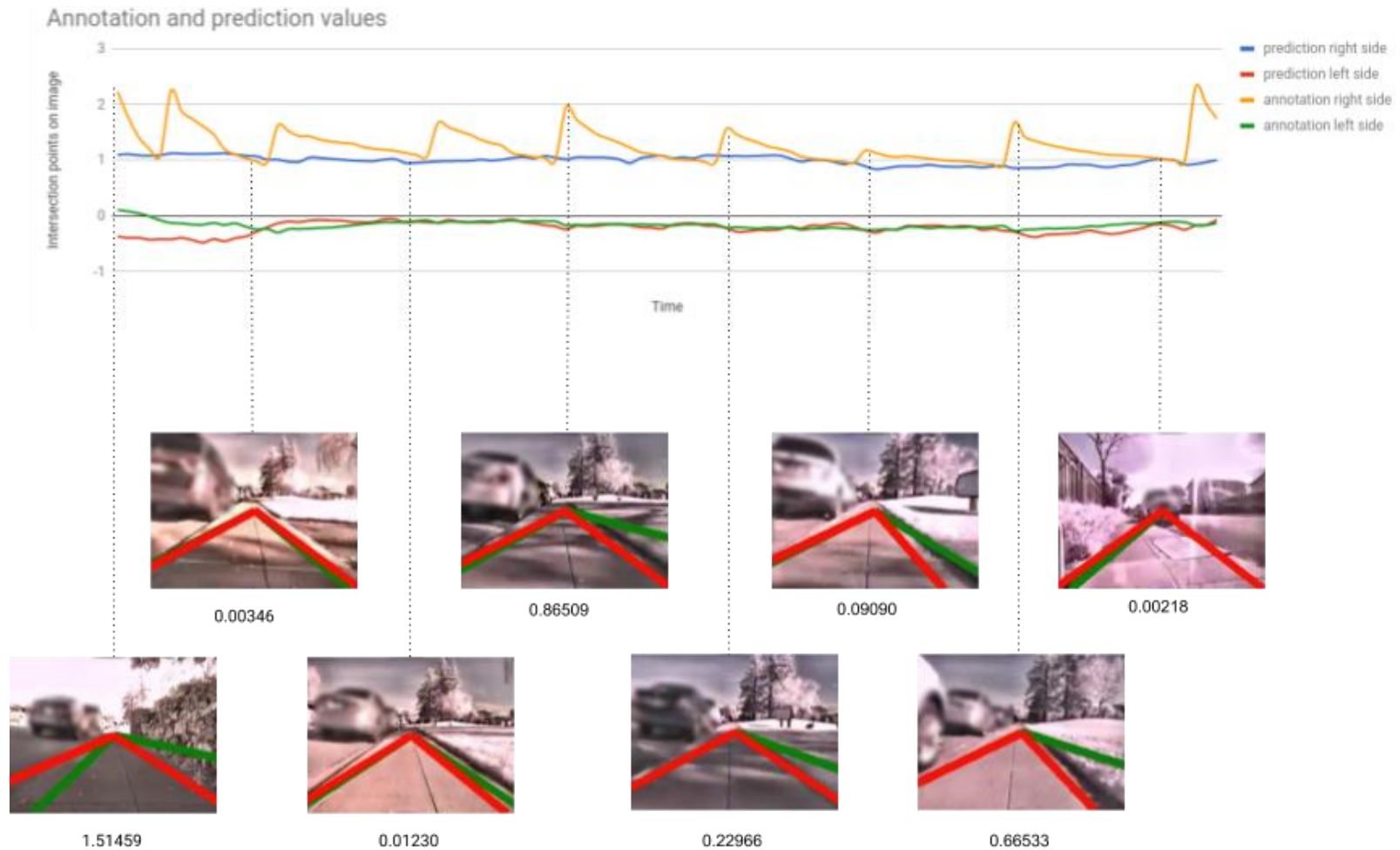


Figure 42. Annotations and predictions with the MSE score and frames. In the above figure the neural network model predictions can be seen with the annotations. For some of the timestamps an image with the corresponding lines are shown (green lines represent the annotations and red lines neural network predictions). Underneath the image MSE score for that frame is written. From these images it can be seen where the annotations go wrong, where the neural network makes mistakes and how the final results are.

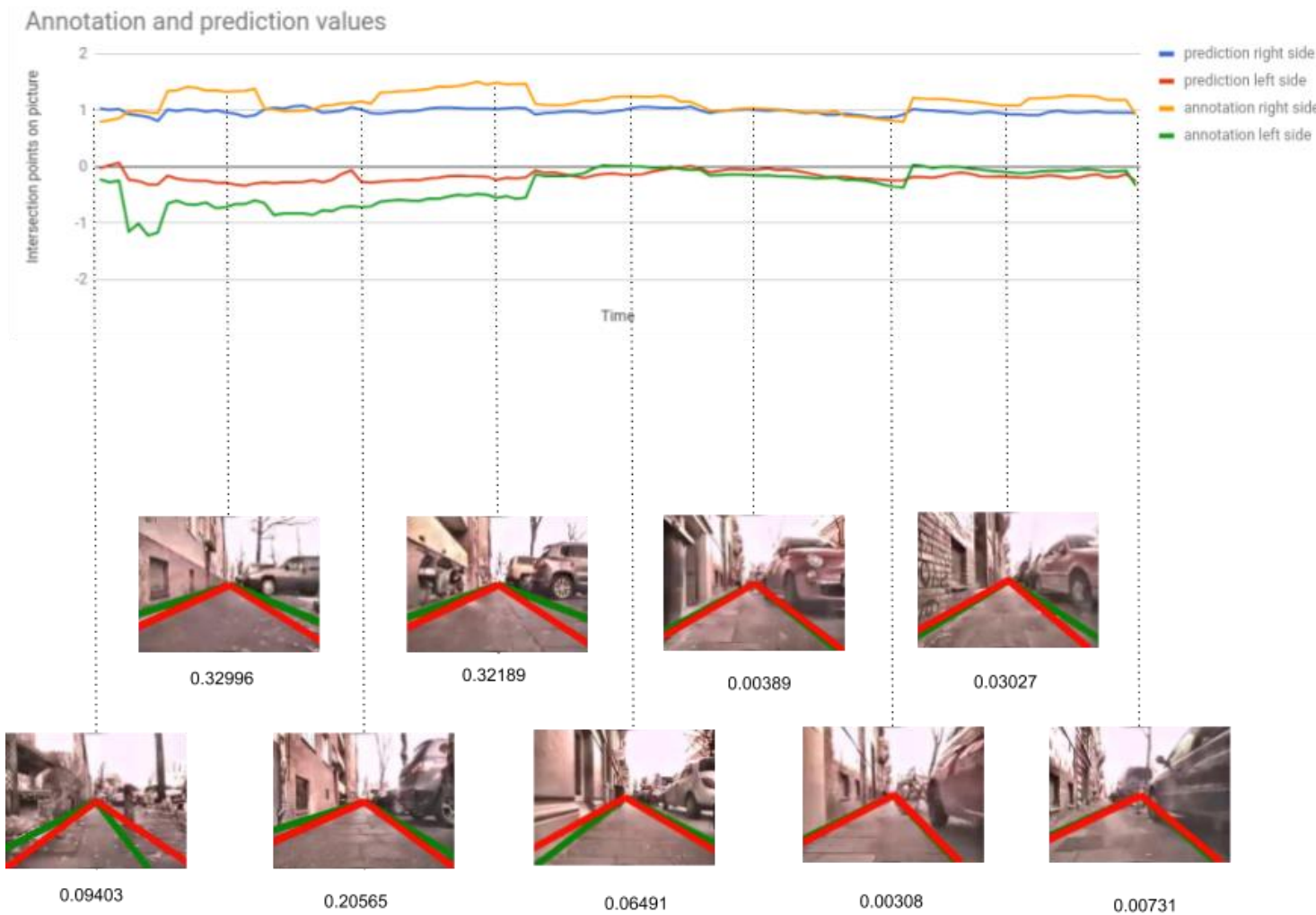


Figure 43. Annotations and predictions with the MSE score and frames. In the above figure the neural network model predictions can be seen with the annotations. For some of the timestamps an image with the corresponding lines are shown (green lines represent the annotations and red lines neural network predictions). Underneath the image MSE score for that frame is written. From these images it can be seen where the annotations go wrong, where the neural network makes mistakes and how the final results are.

7.4 Future work

The future work on the algorithm side will be to add the classification if an image is taken from the sidewalk or a crossing. This could be done with the current images. Side camera images are needed to predict if the robot is at a driveway or not.

Additionally, the algorithm should be improved continuously with new images and tuned to increase its accuracy. The training could be done with more accurate and selected annotations to remove the current problems with training data. A problem with camera calibration was detected during the work, but for the next iterations of preparing training data this problem should be eliminated. Before training a new model, it should be analysed if there are any more mistakes left in the generation process and how to get rid of them.

The scope of the work was to evaluate if it is possible to automate the 3D mapping. An algorithm has been proposed, but it is a separate project to integrate it into the everyday work process. This needs changes in the current software and is a big project on its own.

The next steps would be adding automated sequential image analysis and converting 2D intersection points to 3D coordinates. Then a custom smoothing algorithm should be created that takes into account all the mapping edge cases. After applying the smoothing algorithm, the coordinates of ways and nodes can be produced. These will be fed to the server, and results used in production.

Conclusions

The thesis aimed to evaluate the feasibility to automate the 3D mapping process. The proposed automation approach was made with the help of a neural network, that can detect the edges of the pavement. Then the results of edge detections are smoothed to improve the accuracy of the result. In this work, the collection and preparation of training data and training of the model were done.

The neural network was trained to predict pavement edge intersection points with the lower edge of the image. The neural network model performance was evaluated against a simple model which always predicts constants (an average of the training set). Constant values were calculated based on the city's mean input values. Based on the mean square error metric and the visual evaluation, it was clear that the neural network model performed better than the constant model.

Key results from the thesis:

1. Neural network predicts 94% of the pavement edges correctly
2. A method for preparing and filtering the training data is proposed
3. The network model could successfully learn even from the data that contained images that were sometimes incorrectly annotated.

The neural network model can be used in the mapping automation process. Currently, as every 10th image has a significant estimation error and after smoothing the rate can be lowered, but still, it needs a human to check the results. The current accuracy can help to reduce the work by pre-mapping new areas, and a human can look at them and make adjustments as needed.

The algorithm can be improved by having training data containing more context, improving the model further and spending more time in the model development and adding more features like distinguishing way type to it.

It could be considered for the future that, the algorithm can automate the 3D mapping in most of the cases and no human supervision is needed. Although it is likely that there needs to be some human interaction with the map as some of the mapping cases that are very rare are thus impractical to automate.

References

- [1] Technologies, Starship, "Starship Technologies," [Online]. Available: <https://www.starship.xyz/>.
- [2] UBER, "UBER webpage," [Online]. Available: <https://www.uber.com/>. [Accessed 27 4 2018].
- [3] Lyft, "Lyft self-driving vehicles," [Online]. Available: <https://www.lyft.com/self-driving-vehicles>. [Accessed 27 4 2018].
- [4] Tesla, "Tesla autopilot," [Online]. Available: <https://www.tesla.com/autopilot>. [Accessed 27 4 2018].
- [5] Waymo, "Waymo website," [Online]. Available: <https://waymo.com/>. [Accessed 27 4 2018].
- [6] R. Metz, "MIT Technology Review: Baidu sees maps for self-driving cars as bigger business than web search," 9 1 2018. [Online]. Available: <https://www.technologyreview.com/s/609936/baidu-sees-maps-for-self-driving-cars-as-bigger-business-than-web-search/>. [Accessed 27 4 2018].
- [7] HERE maps, "HERE blog," [Online]. Available: <https://360.here.com/1-million-kilometers-here-hd-live-map>. [Accessed 27 4 2018].
- [8] Mapbox, "Mapbox automotive," [Online]. Available: <https://www.mapbox.com/automotive/>. [Accessed 27 4 2018].
- [9] TomTom, "TomTom automotive," [Online]. Available: <https://automotive.tomtom.com/automotive-solutions/autonomous-driving/>. [Accessed 27 4 2018].
- [10] Waymo, "Medium: Building maps for a self-driving car," [Online]. Available: <https://medium.com/waymo/building-maps-for-a-self-driving-car-723b4d9cd3f4>. [Accessed 27 4 2018].
- [11] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85-117, 2015.
- [12] "Python," [Online]. Available: <https://www.python.org/>. [Accessed 27 4 2018].
- [13] "Keras," [Online]. Available: <https://keras.io/>. [Accessed 27 4 2018].
- [14] "Tensorflow," [Online]. Available: <https://www.tensorflow.org/>. [Accessed 27 4 2018].
- [15] "Amazon AWS," [Online]. Available: <https://aws.amazon.com/>. [Accessed 27 4 2018].
- [16] "Pandas library," [Online]. Available: <https://pandas.pydata.org/>. [Accessed 27 4 2018].
- [17] "Numpy package," [Online]. Available: <http://www.numpy.org/>. [Accessed 27 4 2018].

- [18] "OpenCV," [Online]. Available: <http://opencv-python-tutroals.readthedocs.io/en/latest/index.html>. [Accessed 27 4 2018].
- [19] "Matplotlib," [Online]. Available: <https://matplotlib.org/>. [Accessed 27 5 2018].
- [20] "Pillow," [Online]. Available: <https://pillow.readthedocs.io/en/5.1.x/#>. [Accessed 27 4 2018].
- [21] "SciPy," [Online]. Available: <https://www.scipy.org/>. [Accessed 27 4 2018].
- [22] V. H. Mistry and D. R. Makwana, "Survey: Vision based Road Detection Techniques," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 4714-4747, 2014.
- [23] J. Zhang and H.-H. Nagel, "Texture-based segmentation of road images," *Intelligent Vehicles '94 Symposium, Proceedings of the*, 1994.
- [24] . Y. He , H. Wang and B. Zhang, "Color-based road detection in urban traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 309 - 318, 2004.
- [25] B. Wang , V. Fremont and S. A. Rodriguez, "Color-based road detection and its evaluation on the KITTI road benchmark," *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, 2014.
- [26] J. M. Alvarez and A. M. Lopez, "Road Detection Based on Illuminant Invariance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 184 - 193, 2010.
- [27] . H. Kong, J.-Y. Audibert and J. Ponce, "General Road Detection From a Single Image," *IEEE Transactions on Image Processing*, vol. 19, no. 8, pp. 2211 - 2220, 2010.
- [28] . C.-K. Chang, C. Siagian and L. It, "Mobile robot monocular vision navigation based on road region and boundary estimation," *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference*, 2012.
- [29] . C. Oh, J. Son and K. Sohn, "Illumination robust road detection using geometric information," *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference*, 2012.
- [30] . J. M. Álvarez, T. Gevers and A. M. Lópe, "Vision-based road detection using road models," *Image Processing (ICIP), 2009 16th IEEE International Conference*, 2009.
- [31] J. M. Alvarez, T. Gevers and A. M. Lopez, "3D Scene priors for road detection," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010.
- [32] Y. LeCun, . B. Boser, J. S. Denker and D. Henderson, "Backpropagation applied to handwritten zip code recognition," in *Neural Computation*, Winter, 1989, p. 541–551.
- [33] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems*, vol. 1, pp. 1097-1105 , 2012 .
- [34] A. S. Razavian, . H. Azizpour, . J. Sullivan and . S. Carlsson, "CNN Features off-the-shelf: an Astounding Baseline for Recognition," arXiv:1403.6382, 2014.

- [35] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 664 - 676, 2016 .
- [36] C. Shallue, "Show and Tell: image captioning open sourced in TensorFlow," Google, 22 9 2016. [Online]. Available: <https://research.googleblog.com/2016/09/show-and-tell-image-captioning-open.html>. [Accessed 13 11 2017].
- [37] A. Mordvintsev, C. Olah and M. Tyka, "Inceptionism: Going Deeper into Neural Networks," Google, 17 6 2015. [Online]. Available: <https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html>. [Accessed 13 11 2017].
- [38] S. Dieleman, "Recommending music on Spotify with deep learning," 5 8 2014. [Online]. Available: <http://benanne.github.io/2014/08/05/spotify-cnns.html>. [Accessed 13 11 2017].
- [39] J. M. Alvarez, T. Gevers, Y. LeCun and A. M. Lopez, "Road Scene Segmentation from a Single Image," *European Conference on Computer Vision*, pp. 376-389, 2012.
- [40] C.-A. Brust, . S. Sickert and . M. Simo, "Convolutional Patch Networks with Spatial Prior for Road Detection and Urban Scene Understanding," arXiv:1502.06344, 2015.
- [41] G. E. Hinton, N. Srivastava and A. Krizhe, "Improving neural," arXiv preprint arXiv:1207.0580, 2012.
- [42] M. Nielsen, "Neural Networks and Deep Learning," [Online]. Available: <http://neuralnetworksanddeeplearning.com/chap1.html>. [Accessed 13 11 2017].
- [43] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2010.
- [44] GIMP , "GNU Image Manipulation Program user manual," [Online]. Available: <https://docs.gimp.org/en/plugin-convmatrix.html>. [Accessed 27 4 2018].
- [45] "Tutorial: Implementing Layer-Wise Relevance Propagation," 5 7 2017. [Online]. Available: <http://www.heatmapping.org/tutorial/>. [Accessed 27 4 2018].
- [46] L. S. Sterling, *The Art of Agent-Oriented Modeling*, London: The MIT Press, 2009.
- [47] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, Cambridge, Massachusetts, USA: Massachusetts Institute of Technology, 2016.
- [48] G. L. Oliveira , W. Burgard and T. Brox, "Efficient Deep Models for Monocular Road Segmentation," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4885-4891 , 2016.
- [49] "ROS: The robot operating system - project3dToPixel," [Online]. Available: http://docs.ros.org/diamondback/api/image_geometry/html/c++/classimage__geometry_1_1PinholeCameraModel.html#a30b3761aadfa4b91c7fedb97442c2f13. [Accessed 27 4 2018].
- [50] Google, "Google maps API," [Online]. Available: <https://developers.google.com/maps/documentation/javascript/coordinates>. [Accessed 28 4 2018].
- [51] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep," *arxiv.org*, 2017.
- [52] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs.CV]*, 2014.

- [53] "AWS Deep Learning AMIs," [Online].
Available: <https://aws.amazon.com/machine-learning/amis/>. [Accessed 28 4 2018].
- [54] "Connecting with AWS," [Online]. Available:
[https://docs.aws.amazon.com/AWSEC2/latest/
UserGuide/AccessingInstancesLinux.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html). [Accessed 28 4 2018].
- [55] "Htop - an interactive process viewer for Uni," [Online].
Available: <https://hisham.hm/htop/>. [Accessed 28 4 2018].