CHALMERS
UNIVERSITY OF TECHNOLOGY

Safe and enjoyable routes for motorcycle riders, with dynamic hazard warnings

Using graph algorithms and deep learning

Master's thesis in Computer science and engineering

Andreas Carlsson, Jonathan Krän

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Safe and enjoyable routes for motorcycle riders, with dynamic hazard warnings

Using graph algorithms and deep learning

ANDREAS CARLSSON, JONATHAN KRÄN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2020

Supervisor: Stefan Candefjord, Department of Electrical Engineering
Advisor: Niklas Ohlsson, Detecht Technologies AB
Examiner: Bengt Arne Sjöqvist, Department of Electrical Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A gravel road in Dalsland, Sweden. Photo by Andreas Carlsson.

Gothenburg, Sweden 2020

Abstract

Motorbike riders who drive for recreational purposes often seek routes that are optimized for other purposes than the common factors of the length of the route and the time it takes to ride it. The primary aim of this study was to find what such factors could be and to design a routing system that takes these factors into account.

A survey was conducted which showed that curvature and the surrounding scenery of a route are two such factors that are important for this type of motorbike riders. The curvature of the road was calculated by using circumscribed circles to determine the curvature of road segments. Scenery images were sourced from the Google Static Street View API and classified using transfer learning on a pre-trained model of the convolutional neural network ResNet-50.

Seven routing profiles were developed which use a combination of the curvature and scenery data in different ways, for instance by preferring high curvature and scenery that contains water. Each routing profile required the creation of a pre-processed graph and was static in the sense that parameter changes could not be made without additional pre-processing. The routing was performed using the GraphHopper routing engine and OpenStreetMap was the source of all geographical data.

The secondary aim was to explore methods of increasing road safety for this type of motorbike rider. A dynamic warning concept was proposed that can warn the user of potentially dangerous upcoming road sections, such as sharp curves and road works.

The routing system is capable of suggesting routes in Sweden in near real time (less than 50 ms) based on road curvature and surrounding scenery. We conducted a qualitative evaluation, consisting of six deep interviews, of the routing system and the dynamic warning concept. The findings from these interviews suggest that our routing system performs worse than implementations by competitors regarding curvature. However, our system has the advantage of also considering surrounding scenery. The interviewees were mostly positive towards the dynamic warning concept.

The two main ways to improve the routing system appear to be to further increase the preference of curvature when routing and to use more images of surrounding scenery.

In conclusion, this thesis shows that there is promise in using curvature and scenery for routing systems that target motorbike riders who drive for recreational purposes. Additionally, initial results show that dynamic warnings could provide a safer motorbiking experience if implemented in an appropriate way.

Keywords: preference-based routing, curvature, road scenery, routing problem, path finding, road safety, motorbiking.

Acknowledgements

We would like to thank our supervisor Stefan Candefjord for his continued support and commitment throughout the project. We want to express our gratitude to Võ Thị Ngọc Mỹ and Tuyen Ngo for their moral support and encouragement. Additionally we thank Niklas Ohlsson and Tobias Goldman, founders of Detecht Technologies AB, for their valuable input during the project. Finally, we would like to give extra thanks to Tuyen Ngo for her guidance and help with the illustrations in the report.

Andreas Carlsson and Jonathan Krän, Gothenburg, June 2020

Contents

List of Figures

Glossary

Accuracy – Number of correct predictions divided by the number of total predictions made

API - Application Programming Interface

AUC – Area Under the Curve, a metric in machine learning that defines the area under the ROC curve

Bee-line distance - The straight line distance between two points on a map

Class – The label assigned to an image from the network

CNN – Convolutional Neural Network, a type of neural network suited for image classification.

Confusion matrix – A matrix containing the actual class of an item (rows) and the class predicted by a machine learning model (columns)

Content tag – A tag assigned to an image that describes the content in a more detailed way compared to the label

Danger value A value representing the danger of an upcoming coordinate, based on the curvature of that coordinate, the current speed of the driver and the distance remaining to that coordinate.

DNN – Deep Neural Network, a neural network with multiple layers between the input and output layers.

DRL – Deep Residual Learning

GPS – Global Positioning System

Label – The overall content of an image

Labeller – The person assigning a label to an image, setting the ground truth for that image

Loss – A metric for how well a model is able to classify a data set. Lower is better.

ML – Machine Learning

Node – A vertex in a graph (given by its longitude and latitude).

ODbL – Open Database License

OSM – Open Street Map, a collaborative project aiming to make geographical data available under an open source licence (ODbL).

ResNet – A CNN model that uses skip-connections.

RNN - Recurrent Neural Network, a class of artificial networks.

ROC – Receiver Operating Characteristic curve

Way - An ordered list of nodes in the Open Street Map data.

WHO – World Health Organization

1

Introduction

Routing in maps is a thoroughly studied topic with many approaches suggested for finding optimal routes. Normally, optimality is measured in terms of short and efficient routes. However, some groups of road users might prefer different route properties than these. Motorbike riders that drive for recreational purposes are one such group that might prefer other types of routes, not necessarily the shortest [1]. We make a distinction between different groups of motorbike riders; in many countries, the motorcycle is mainly used as a mean of transportation, while in other countries it is mainly used for recreational purposes. We suspect that this latter group of motorbike riders are sometimes willing to sacrifice time efficiency for route quality and enjoyability.

The aim of this thesis is to develop a routing system that uses the preferences of recreational motorbike riders to recommend enjoyable routes. We hypothesise that the curvature of the road and its surrounding scenery are two important factors for how enjoyable a route is perceived to be. Using curvature as a routing parameter has been done by others, but using the surrounding scenery of the road as a parameter is a novel approach to the best of our knowledge.

A secondary aim is to provide safety information to riders in the form of dynamic warnings that can alert the driver as they are driving along a route, for instance if they are approaching a curve with excessive speed or if there is an ongoing road work ahead. Warnings like this can be beneficial not only for the driver but also for other people and vehicles sharing the road. For example, if a driver gets a chance to slow down before approaching a road work it will also increase the safety for the people working on the construction site.

1.1 Background

Routing, i.e. path finding, amounts to solving the shortest path problem. The shortest path problem is to find a path between two vertices in a graph such that the sum of the weights along its edges are minimized. We will refer to such a path as an *optimal path*. In many cases, optimality is defined in terms of the shortest or fastest route between two vertices. The shortest path problem is a well-studied

problem and there are many proposed algorithms for solving it [2; 3]. The problem is easy for small graphs, but it quickly becomes a challenging problem in terms of required time as the graph size increases [4]. One algorithm for solving the shortest path problem is Dijkstra's algorithm [2]. It is a well-known algorithm for path finding, and many later approaches are based on this algorithm.

Algorithms used for finding optimal routes between two locations generally use a graph representation of the road network. Graphs of the world's road network are available online, like those provided by *OpenStreetMap* and *Google Maps*. These graphs are used by services and projects such as *GraphHopper* and the *Open Source Routing Machine* for routing in real road networks. While the default modes of these projects optimize for short and fast routes, their algorithms can be adapted to optimize for different criteria instead.

As mentioned earlier, recreational motorbike riders will likely not consider the shortest route to be the best route. These riders are often less interested in driving the shortest or fastest route between two locations. Instead, they might want an *enjoyable* route. What makes a route enjoyable depends on several factors and can differ between riders. It is not clear what optimality of a route means for this group. There are likely shared preferences though, and curvy routes are generally appreciated [5]. The surrounding scenery is possibly another important factor for how enjoyable a route is perceived to be.

Riding a motorbike is often considered to be significantly more dangerous than other means of transportation [6]. There are several reasons for this and while the behavior of the individual riders can be a significant risk factor [7], the type of road in question also plays a significant role [8].

Recommending routes that are optimized based on parameters like curvature and surrounding scenery could affect the safety of an average route. In particular, recommending enjoyable routes could put motorbike riders at greater risk. For example, assume that motorbike riders are found to enjoy curvy roads. Using curvy roads as a routing parameter could result in more dangerous roads since up to 17% of all motorcycle accidents occur while in a curve [9]. It would therefore be important to warn drivers, for instance by advising them to be extra cautious as they approach a particularly curvy road segment or an area with ongoing road works.

According to a study from WHO released in 2015 [10], road traffic injuries is the number one cause of death amongst people aged 15-29 years. Data from the same report suggests that almost a quarter of all traffic accidents world-wide with a deadly outcome was with a motorbike. Moreover, motorcycle users are among the more vulnerable road users [11]. Despite motorcycle users only making up a fraction of road users, they represent a significant amount of fatal accidents. Song et al. [11] state that "Although motorcycles made up only 3% of all registered vehicles in the U.S. in 2012, they accounted for 15% of all traffic fatalities and 18% of all occupant fatalities". Motorcycle riders are often not able to perform a crash-

avoidance maneuver in time [12].

1.2 Related work

While there are many other studies on the problem of preference-based routing, the problem appears to be relatively unexplored for the motorbike rider target group. Novack et al. [13] present a preference-based routing system with the goal of creating pleasant routes for pedestrians within a city environment. Parameters for the route recommendation include occurrence of green areas, social spaces and less trafficked streets. Routes were created using OpenStreetMap data with a cost function based on these parameters. In their evaluation, they found that people generally perceived their routes to be preferable to the shortest route.

On the other hand, real-time routing is a thoroughly explored topic. Luxen and Vetter [14] explore how data from OpenStreetMap [15] can be used to build a real-time routing engine. They evaluate the performance on both a computationally limited handheld device and on a more powerful server and show that it is feasible to perform offline routing on a handheld device. Their solution for handheld devices make use of contraction hierarchies and as well as other preprocessing steps to accomplish real time performance on handheld devices.

There is also some previous work on warning system for motorbike riders. M. Song et al. [11] tested several different warning interface displays for motorcycle riders. Auditory, visual and haptic feedback modes were considered. They found that their test group preferred a combination of auditory and haptic feedback for conveying warning information. Moreover, they state that auditory feedback is easy to implement but was lacking in terms of conveying directional information. Riders in their study were able to easily distinguish between handlebar vibration and haptic warnings from their system.

1.2.1 Commercial alternatives on the market

There are several commercial products for route planning aimed at motorcyclists. We took a closer look at some of the alternatives. Figure 1.1 shows the graphical interfaces for these alternatives.

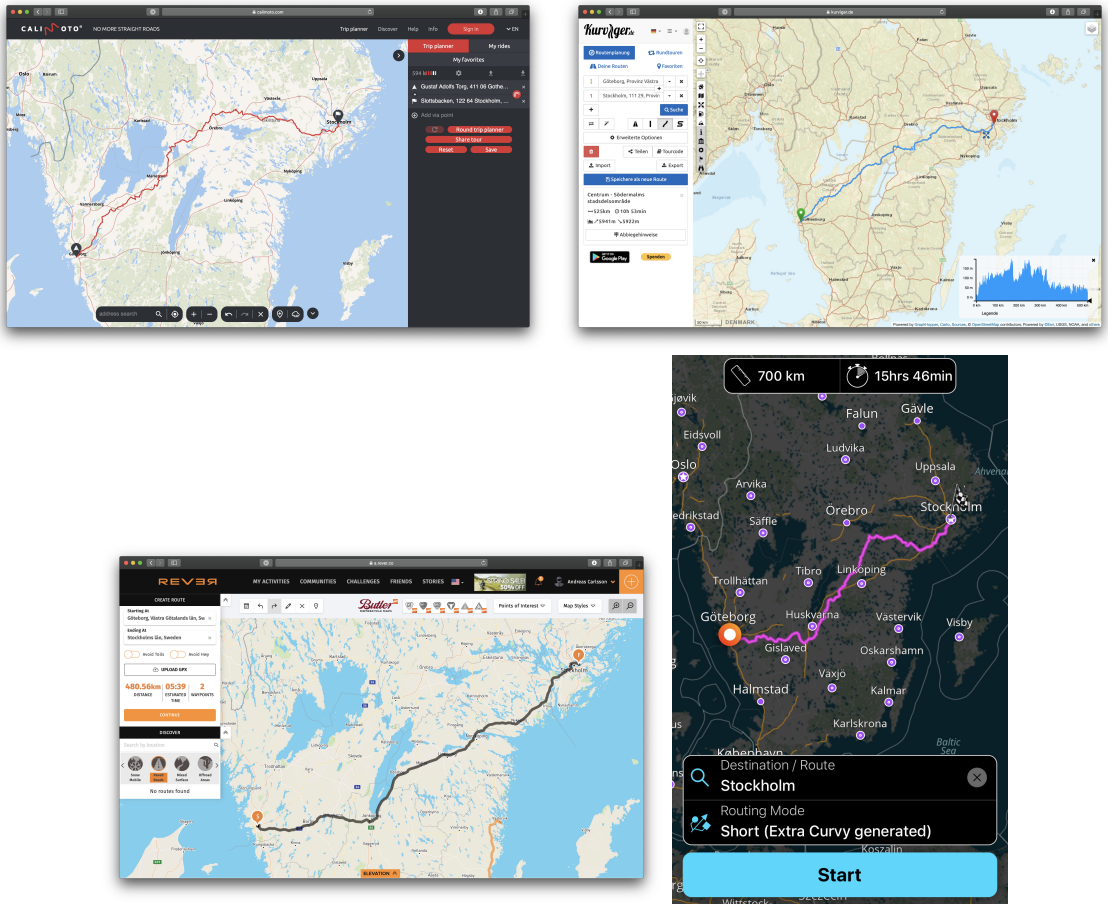


Figure 1.1: The interface of the respective trip planners in the comparison.

Calimoto's trip planner [16] offers routing with intermediate stops and planning round trip routes with a given length. An interesting feature of the Calimoto trip planner is the possibility to set a certain level desired curvature between each intermediate stop. In order to export or save a route the user has to be logged in. An interesting feature is their connection to weather data where it is possible to display expected precipitation along the route. Calimoto also gives a ranking for how high curvature a route has (see Figure 1.2).

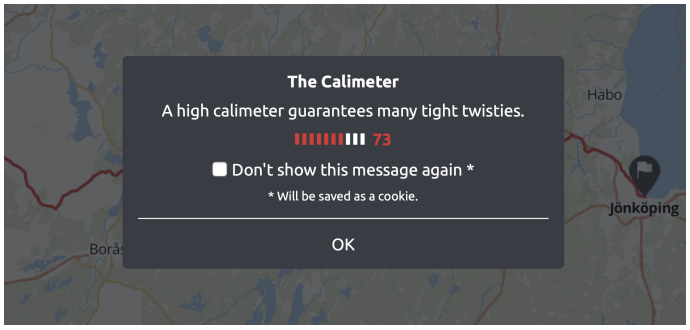


Figure 1.2: The ranking of curvature for a specific route.

Kurviger [17] is an online planning tool for finding routes with varying degrees of curvature. The curvature parameter has three possible values, and it is possible to include or exclude highways from routing. Kurviger uses GraphHopper [18] for routing and OpenStreetMap [15] for map data. Routes can be imported and exported in a variety of formats. One unique part of Kurviger is their choice to display altitudes along the route. Similarly to Calimoto [16], Kurviger also has an option to search for round trip routes.

Rever [19] is a venture capital backed startup founded by Justin Bradshaw and Mark Roebke in 2015. They are based in Eagle, Colorado in the United States. Their main product is an app in which you can track your routes and they also offer an online trip planner. Rever's distinguishing feature is perhaps their focus on offering vetted roads as part of their paid plans. There are options to avoid tolls and highways, but no option for finding roads with a certain level of curvature.

Scenic [20] is an iOS-application where it is possible to search route based on how fast they are to drive, the distance of them or the efficiency. They also include an option to generate a "curvy route", where the level of curvature can be set to one of three levels. There are also options to avoid unpaved roads, narrow roads and avoiding to drive the same road twice. The routing engine makes use of Kurviger's [17] algorithm.

1.3 Aim

The main aim of this thesis is to develop a routing algorithm for finding enjoyable and safe routes for motorbike riders. The routing algorithm should make use of the curvature of the road and its surrounding scenery. Moreover, the algorithm should be adaptable to different preferences, such as preferring high curvature or a particular type of scenery. The use of surrounding scenery for routing is perhaps the main contribution as it has not been done before to the best of our knowledge.

A secondary aim is to provide additional safety information to motorbike riders. This will be done by giving dynamic warnings to drivers as they are approaching a section of the route which might be dangerous.

1.4 Scope

This project only includes and uses data from the geographic area of Sweden (a total area of around $450\,000\,\mathrm{km}^2$). However, all approaches can be extended to include a larger area assuming availability of data.

2

Theory

2.1 OpenStreetMap graph structure

A road network can be seen as a graph consisting of edges and vertices. In the OpenStreetMap graph format, vertices (called nodes) are divided into tower nodes and pillar nodes (see Figure 2.1). A **tower node**, also called a junction, has either one, three, or more outgoing edges. If a tower node has one edge it corresponds to the end of a one-way-street. The other case, with three or more edges, corresponds to a junction from which several ways are possible. **Pillar nodes** are nodes that hold information about the road between junction nodes. The pillar nodes are not necessary for routing since they only contain two edges. Assuming we have reached a pillar node using one of those edges, there is only one remaining edge that can be traversed. By ignoring pillar nodes when routing, the time it takes to create a route between two points can be significantly decreased. The main purpose of the pillar nodes is to correctly map the physical location of a road to its representation in the graph.

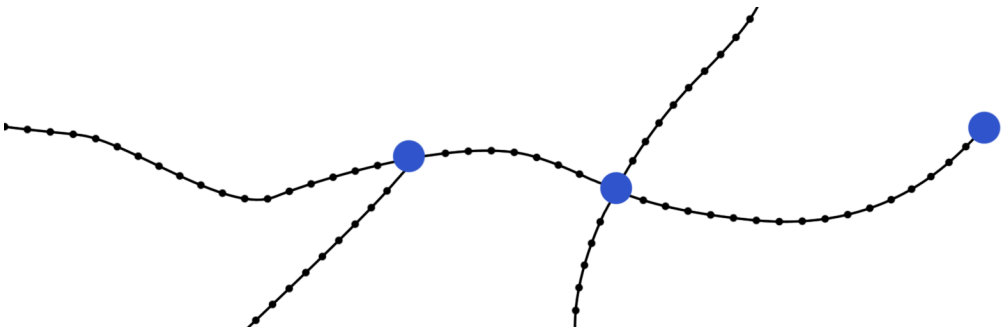


Figure 2.1: The blue circles represent tower nodes and the small black circles represent pillar nodes. Tower nodes correspond to junctions and the end of a one-way streets. Pillar nodes correspond to positions along roads (they define road segments).

2.2 Routing in graphs

Routing consists of finding a path in a graph. There are many algorithms for this problem, two of the most common ones being *Dijkstra* and *A**. The GraphHopper routing engine has support for both these algorithms, including modifications like contraction hierarchies (see section 2.5), which can speed them up.

2.2.1 A*

A commonly used algorithm for finding the shortest way between two vertices in a graph is A* (A star) [21]. It is also often used to find the shortest paths in road networks [22]. A* is optimal in the sense that it always finds the best solution to the problem given that the heuristic used when evaluating a path never overestimates the cost. This is expressed as having a heuristic that is admissible. The heuristic used in this thesis is the pythagorean distance between two nodes, i.e the bee-line distance.

2.3 Haversine distance

Calculating the distance between two coordinates is important both when determining the road curvature (see section 2.4) and when sampling points for scenery images (see subsection 3.2.1).

The Haversine formula is a formula that can be used to calculate the distance between two points on a sphere [23]. The distance between two coordinates in latitude and longitude is given by

$$d = 2r \arcsin \sqrt{\sin^2\left(\frac{\theta_2 - \theta_1}{2}\right) + \cos(\theta_1)\cos(\theta_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)} \tag{2.1}$$

where r is the radius of the sphere, the first coordinate is (θ_1, λ_1) and the second coordinate is (θ_2, λ_2) (θ is latitude and λ is longitude).

2.4 Determining road curvature

According to our user survey (see section 3.1), road curvature is an important factor for how enjoyable a route is to motorbike riders. It is not clear how best to measure road curvature. Two approaches for measuring road curvature are presented here.

2.4.1 Bee-line distance to actual distance ratio

One approach to finding the curvature of a road section is to compare the ratio of the bee-line distance between two tower nodes with the distance for also visiting each

one of the intermediate pillar nodes. The latter one is the actual distance one needs to go from one tower node to the other. By dividing the actual distance with the bee-line we get a ratio that we can use as a measurement of curvature. A downside with this approach is that it does not take the actual curves into consideration. Figure 2.2 exemplifies this problem. The left and the right road in Figure 2.2 have the same distance between the nodes A and B, but the road to the right contains many more curves than the road to the left.

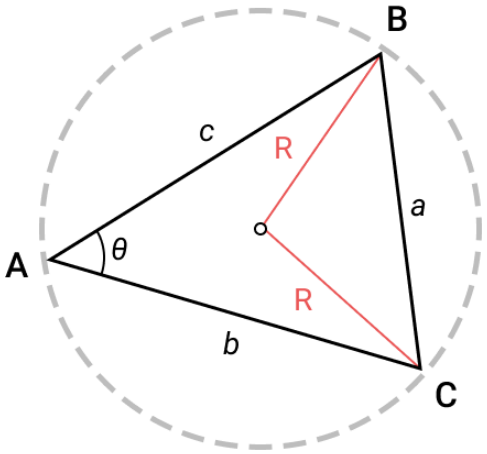
Figure 2.2: The problem that arises when using bee-line to actual distance ratio for determining curvature.

2.4.2 Circumscribed circles

Another way of calculating curvature is by using circumscribed circles. It differs from the bee-line approach in that it considers the curvature of segments of three nodes along the road section. This allows it to correctly solve for the problem in Figure 2.2.

The circumscribed circle of a triangle is defined as the unique circle that passes through all three vertices of the triangle [24]. Figure 2.3b illustrates the circumscribed circle of a triangle.

(a) A triangle with corners in three of the geographical points along a road. Map © Mapbox 2020 and © OpenStreetMap 2020.



(b) A triangle with its corresponding circumscribed circle.

Figure 2.3: Circumscribed circles in road segments.

To use circumscribed circles for calculating road curvature, we form triangles using groups of three points along a road segment (see Figure 2.3a). The radius R of the inscribed circle can be computed by the formula

$$R = \frac{abc}{2bc \cdot \sin\theta} = \frac{abc}{\sqrt{(a+b+c)(a+b-c)(a-b+c)(-a+b+c)}} \tag{2.2}$$

Where a, b and c are the lengths of the sides of the triangle and θ is the inscribed angle. The radius is used as a measurement of how sharp a certain road section bends. A large radius indicates a large θ which means that the section is relatively straight. A small radius means that θ is small and that the road section has a sharp bend.

2.5 Contraction Hierarchies

Basic path finding techniques such as Dijkstra's algorithm and A^* are not fast enough to provide real-time routing in the original OpenStreetMap road graph of Sweden. Routing on that graph using these techniques can take seconds even when the origin and destination points are relatively close to each other. With the origin and destination being far apart, the routing can take tens of seconds, which is too slow for practical use. However, the routing speed of these algorithms can be improved by preprocessing the graph that they use for routing.

Contraction hierarchies is a preprocessing technique that can be used to speed up shortest path queries in road networks [25]. The technique takes advantage of the hierarchical nature of road networks. When constructing the contraction hierarchy (CH) graph G_{CH}, the nodes of the original graph G are ordered by an estimate of the

gain that will be had if a node is contracted. Initially, $G_{\text{CH}} = G$. Contraction of a node means that it is removed from G_{CH} without removing any shortest paths from G_{CH} that are present in the original graph G. The contraction is done by replacing paths $\langle u, v, w \rangle$ by a new path $\langle u, w \rangle$.

When routing, the graph $G_{\text{CH}} = (V, E)$ is split into two graphs called the upward graph $G_{\uparrow} := (V, E_{\uparrow})$ with $E_{\uparrow} := \{(u, v) \in E : u < v\}$ and the downward graph $G_{\downarrow} := (V, E_{\downarrow})$ with $E_{\downarrow} := \{(u, v) \in E : u > v\}$. The node comparisons use the node contraction order. Shortest path queries from s to t consist of running a variant of bidirectional Dijkstra shortest path search. A forward search is performed in $G_{\uparrow}$ and a backward search in $G_{\downarrow}$. If, and only if, there exists a shortest path between s and t in G, these searches will meet at a node with the highest order of the nodes present in a shortest path from s to t.

2.6 Nearby point search (R-Tree)

A common problem when dealing with geographical data is searching for nearby points around a given point. The routing process presented in this thesis requires n-nearest point queries. It uses this type of query both for interpolating scenery values and for sampling points for scenery images.

The R-Tree is a data structure commonly used for handling the n-nearest search problem. The data structure is a height-balanced tree whose leaf nodes contain pointers to objects. Its structure makes efficient spatial search possible. [26].

2.7 Scenery

We hypothesize that the surrounding scenery of a route can be used as a factor for determining how enjoyable and safe the route is. For example, a road surrounded by dense forest can be more dangerous to ride compared to a road surrounded by open fields. Figure 2.4 shows some example images of the scenery around roads.

(a) Example of scenery

(b) Example of scenery

(c) Example of scenery

(d) Example of scenery

Figure 2.4: Examples of scenery images along different routes. Images © Google 2020.

2.8 Machine learning for scenery classification

The routing algorithm needs sceneries as numerical values. Therefore, the scenery images have to be transformed from images to some numerical representation. This transformation can be performed using machine learning techniques [27].

2.8.1 Convolutional Neural Networks

A convolutional neural network (CNN) is a class of deep neural networks that makes use of convolutions as input to the neural network. In the context of deep machine learning, a convolutional layer is a set of filters, or kernels, that perform a set of operations on the input data and and produce an output in the shape of a vector. A kernel in a convolutional layer is a matrix of scalars that are multiplied with the input vector. One classic example of a kernel is one that triggers on contours in an image [28]. Figure 2.5 shows the basic architecture of a CNN.

These kinds of networks are particularly useful when the input data has a spatial relationship. Images contain this kind of spatial structure; when dealing with images, it is generally beneficial to use a method that does not treat pixels that are distanced in the same way as pixels that are close to each other. CNN's has proved to work very well for this specific task [29]. The convolutional layers will provide feature extraction (thus treating pixels that are close together differently from pixels that are far apart) that can be used as input to the fully connected layers.

The input data generally passes through a number of convolutional layers, and are then flattened to a one-dimensional vector before being fed to a set of fully connected layers. These fully connected layers are then trained using stochastic gradient descent. There exists several methods for doing this learning [30]. In this thesis we make use of a the Adam optimizing algorithm, which is an extension to stochastic gradient with adaptable learning rate [31].

The last layer of the network is a so called softmax layer [32] that transforms the score of each class to a probability between 0 and 1. The softmax function takes $z_k, k = 1...K$ as input and gives calculates to probability σ for a given class c as

$$\sigma(z_c) = \frac{e^{z_c}}{\sum_{i=1}^{K} e^{z_i}} \tag{2.3}$$

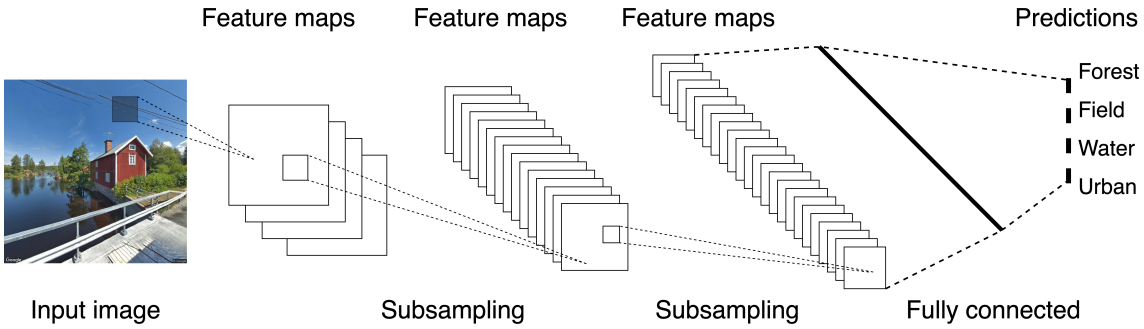


Figure 2.5: Simplified illustration of the convolutional neural network architecture used in this thesis. Street View image © Google 2020.

2.8.1.1 ResNet-50

ResNet50 [33] is a 50 layers deep convolutional network that is commonly used for image recognition. The network uses residual learning which is a technique that makes it easier to train deeper networks. The technique is implemented using *skip connections* which allows the network to skip certain layers. This skipping over layers also helps mitigate the problem of vanishing gradients, thus making it easier to train the network [33].

2.8.1.2 ImageNet

When training a neural network, a large amount of training data is generally needed in order to reach satisfying performance. An image data set commonly used for training networks is ImageNet, which is an open data set [34] consisting of over 14 million images. The ResNet50 network used in this thesis was pretrained on a subset of the ImageNet data set where its images had been divided into 1000 categories.

2.8.2 Transfer learning

ResNet50 is the neural network used in this thesis to classify scenery images. This neural network has over 23 million trainable parameters. Training all of these from a random initialization would require extensive computing power and a large set of training data. To avoid having to retrain the whole network, one can make use of transfer learning. Transfer learning is an area in machine learning where knowledge gained on one problem set is adapted for use on a different set of data that is similar to the original data. It is useful when training a model from scratch is infeasible, either due to computational restraints or due to lack of training data. In this thesis, both of these constraints are applicable (see section 5.2). In transfer learning, the weights of every layer except the last few ones are frozen. Their weights are left unchanged, and training occurs only on the last few layers that are not frozen.

2.8.3 Inter-rater reliability

Training a neural network requires labelled data. Such labelling tasks often benefit from having several people label the same data set [35]. There are several reasons

for this. For instance, two people might perceive one image in different ways depending on their background. It is also probable that mistakes are made during the classification process. As such, the classification results can likely be improved by combining the classifications of several people. One approach for determining the degree of agreement of the classifications is Krippendorff's alpha [36]. This metric gives an indication on the level of agreement as $\alpha \in [0, 1]$, where a higher α corresponds to a higher reliability. This value is calculated as

$$\alpha = 1 - \frac{D_o}{D_e} \tag{2.4}$$

where

$$D_o = \frac{1}{n} \sum_{r \in R} \sum_{k \in R} \delta(c, k) \sum_{u \in U} m_u \frac{n_{cku}}{P(m_u, 2)}$$

$$D_e \frac{1}{P(n, 2)} \sum_{c \in R} \sum_{k \in R} \delta(c, k) P_{ck}$$

and

$$P_{ck} = \begin{cases} c \neq k, & n_c n_k \\ c = k & n_c(n_c - 1) \end{cases}$$

D_o := Observed disagreement between two classifiers.
D_e := Expected disagreement by chance.
$\delta(v, v')$:= 1 if $v = v'$, 0 otherwise
U := A set of items, (i.e. images in the training set)
u := A single item from the set U
R := The set of all classes
P_{ck} := The permutation function

$\alpha = 1$ means that there is perfect reliability, which in this context means that the people labelling images are in complete agreement.

3

Methods

3.1 User survey

In order to understand what matters the most to motorcyclists searching for routes, a survey in the format of an online questionnaire was conducted. The target group for the survey should ideally include a representative sample of all motorcyclists that drive for recreational purposes. We made the assumption that randomly sampling users from the Detecht user base provides an adequate approximation for this group. The participants for the survey were drawn from the Detecht user database, from which 40 000 people were randomly selected. These people were sent an email with a link to the survey along with a short explanation of the purpose of the survey.

3.1.1 Constructing the survey

Survey questions can be structured either as free response questions or by using a fixed set of options that can be selected for a question. Using a fixed set of options is tempting as it simplifies the evaluation process, but there are criticisms of both approaches [37]. If the response categories are too strictly defined we might miss out on valuable information that the respondent is unable to relay. On the other hand, if the options are too free, the result of the survey will be harder to evaluate. Since the survey was conducted via a web page the construction of the questions was even more important [38]. The survey had to be exhaustive enough to give valuable information, while also being easy and convenient for the respondents to complete.

The resulting survey used fixed options for all questions except the last one, for which the respondent was able to provide any additional information or thoughts. The survey consisted of 20 questions allowing respondents to complete the survey within a few minutes.

For a number of the fixed option questions, the respondent could choose between *strongly disagree* and *totally agree*. To encourage respondents to make a definitive choice, these questions did not include a neutral option. The possible options were *strongly disagree*, *disagree*, *agree*, and *strongly agree*.

Some demographics are considered to be at higher risk when riding motorbikes,

in particular young male adults [39]. Because of this, the survey also collected information about the gender and age of the respondent. No other personal data was collected except for gender and age, in order to secure anonymity and thereby receiving as honest responses as possible. If a submission to the survey was possible to connect to a specific individual, the respondents might be encouraged to provide data of how they want to be perceived rather than how they actually feel [40].

3.1.2 Analyzing the results

The survey was open during a six-week period and received 383 responses in total. The majority of respondents were males above the age of 40 who had more than 10 years of driving experience. Nearly 40% (153 respondents) of the respondents had been involved in an accident and 18.8% (72 respondents) had been in an accident so severe that they had to visit the hospital. The vast majority of responses were from male respondents older than 40 years. Due to the lack of data, it was not possible to draw any valuable conclusions from the gender or age of the respondents.

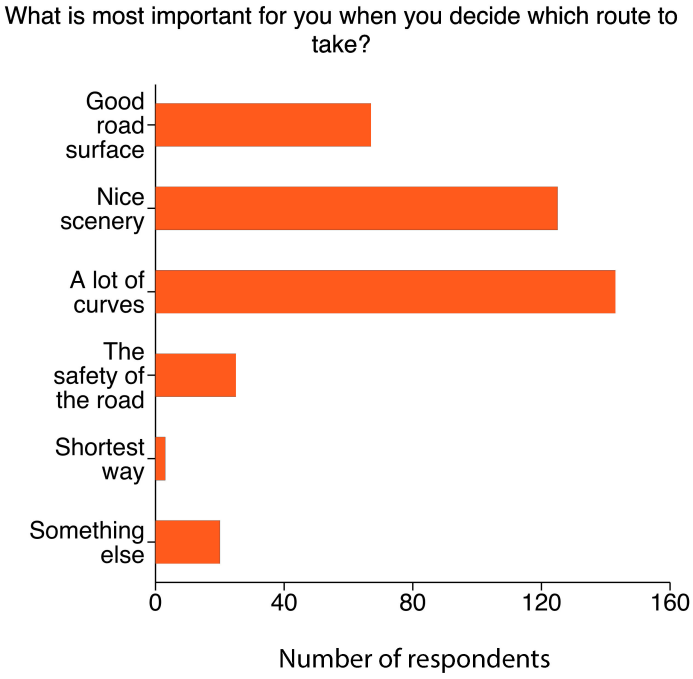


Figure 3.1: Curvature and scenery turned out to be the two most important factors according the respondents of the survey.

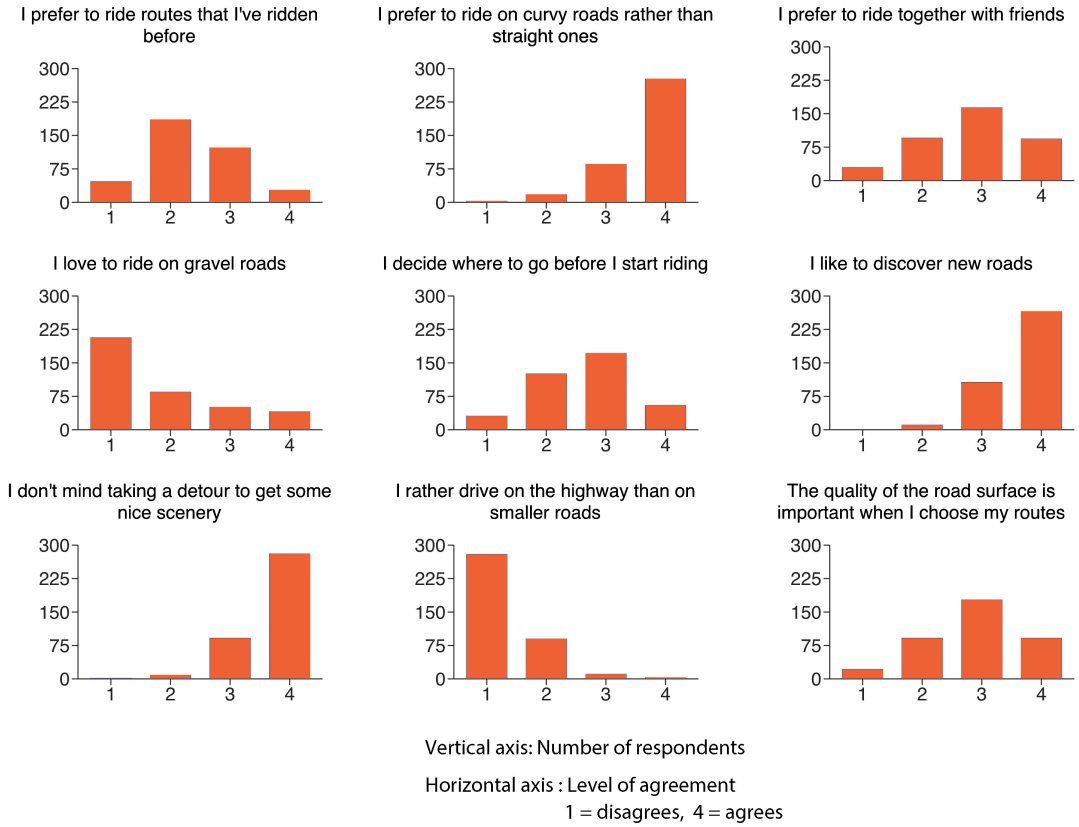


Figure 3.2: To find out the respondents' preferences, they were asked to give a rating between 1 and 4 for nine different statements. 1 stands for "disagree" and 4 stands for "totally agree".

The results of the survey was used to decide what to focus on. As can be seen in Figure 3.1 and Figure 3.2, the two most important factors are the road curvature and the surrounding scenery.

Figure 3.3: Examples of images for the chosen categories: forest, field, water and urban. Images © Google 2020.

3.2 Scenery classification

The scenery classification is based on images drawn from the Google Street View Static API. The surrounding scenery of a road can be used both to recommend enjoyable and safe roads. For instance, drivers might prefer a surrounding landscape rich in lakes and rivers. The scenery can also be used to recommend safer roads by prioritizing roads with clear sight e.g. by penalizing sharp curves in dense forest areas. Based on these considerations, the four categories **forest**, **field**, **water** and **urban** were chosen. These categories were chosen because they broadly represent the different types of scenery in Sweden. Figure 3.3 shows examples images for the four categories.

3.2.1 Selecting coordinates for scenery image sampling

Ideally, all available scenery images would be used for scenery classification. However, as the images were all taken from the Google Street View Static API, it was not possible to use all of them. Using all of the images would require too many resources, both computational and economical. Due to these restrictions (mainly the economical one), the amount of images that could be used for scenery classification was limited to around 60000.

Because of this restriction, a subset of all images available on Google Street View had to be selected while covering as much area as possible. It is reasonable to only use images that were taken near roads. If they were not taken near a road, they are unlikely to make a noticeable difference for the proximity-based routing algorithm. Because of this, the locations for sampling images to be used for scenery classification were selected as a subset of available nodes in the road network.

Two main methods of selecting sampling points were evaluated. In the first approach, nodes are sampled uniformly random. While this method leads to an unbiased selection of points, it has several shortcomings. With this approach, many more nodes would be sampled from areas with dense road networks like cities than from less dense areas (see Figure 3.4). Moreover, since there is no mechanism that keeps nodes apart from each other, many samples are wasted as they are geographically close to other samples and thus their surrounding scenery will not vary significantly.

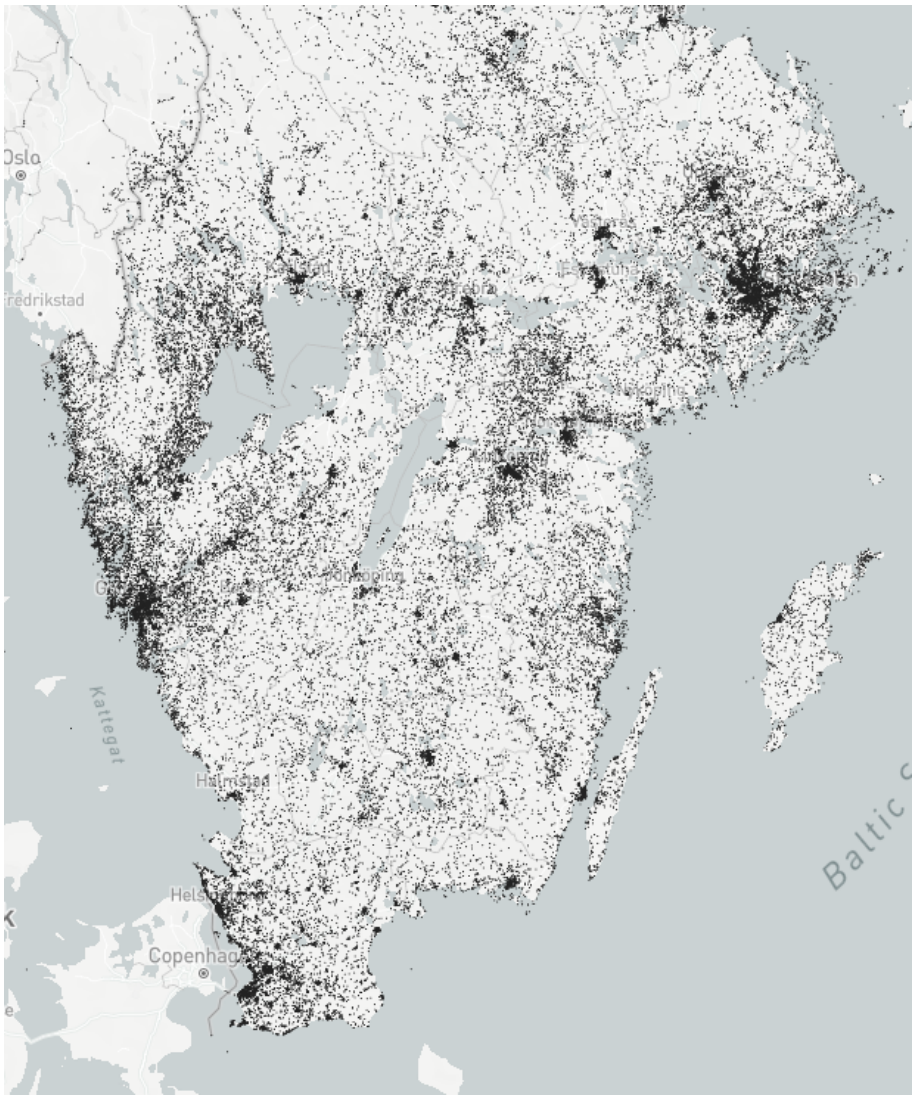


Figure 3.4: The problem in which many more nodes are sampled in areas like cities than elsewhere. Each black dot corresponds to one sample. Map © Mapbox 2020 and © OpenStreetMap 2020.

In the second approach, the bounding box of the map is subdivided into rectangular areas (Figure 3.5) with each area having side lengths 0.05 longitude and 0.05 latitude. The probability of sampling any given node was set to be proportional to the density of nodes in its containing area and was computed as

$$P(\text{Select current node from box } i) = \frac{N}{K \cdot \log(n_i)}$$

where N is the total amount of nodes to sample, K is the amount of boxes in the grid, and n_i is the total amount of nodes in box i.

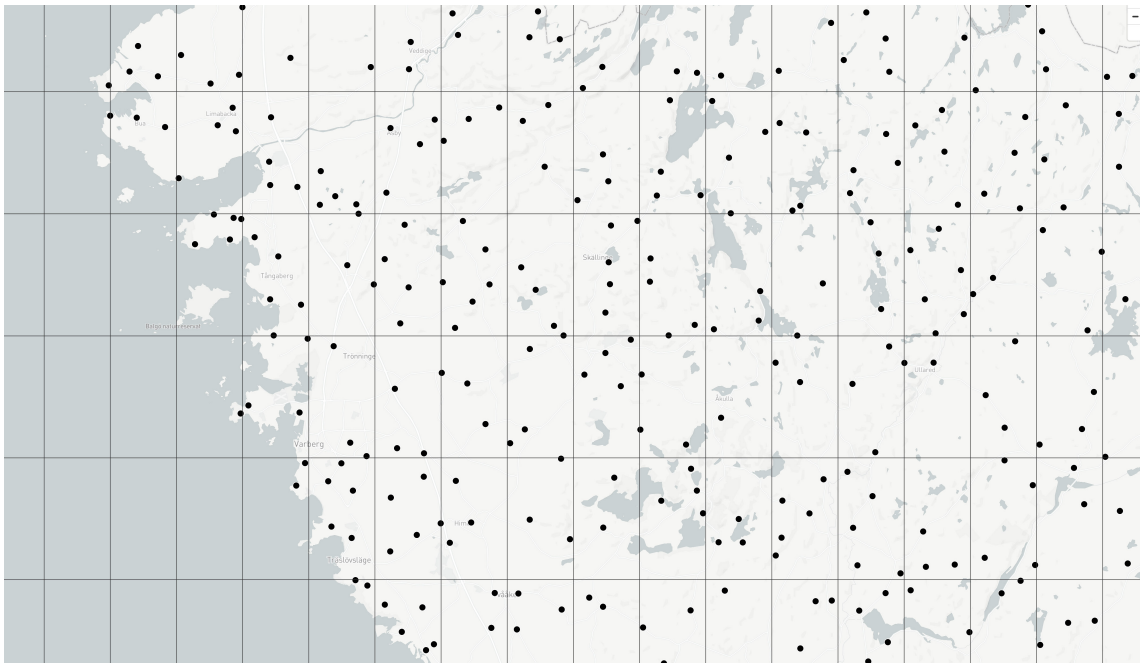


Figure 3.5: The grid overlay with a side length of 0.05 in latitude and 0.05 in longitude. The main purpose of using a this approach is to prevent the algorithm from choosing nodes that are too close together in a dense area, for example a city. Map © Mapbox 2020 and © OpenStreetMap 2020.

This approach solved the issue where boxes with low node density often would not receive any samples at all.

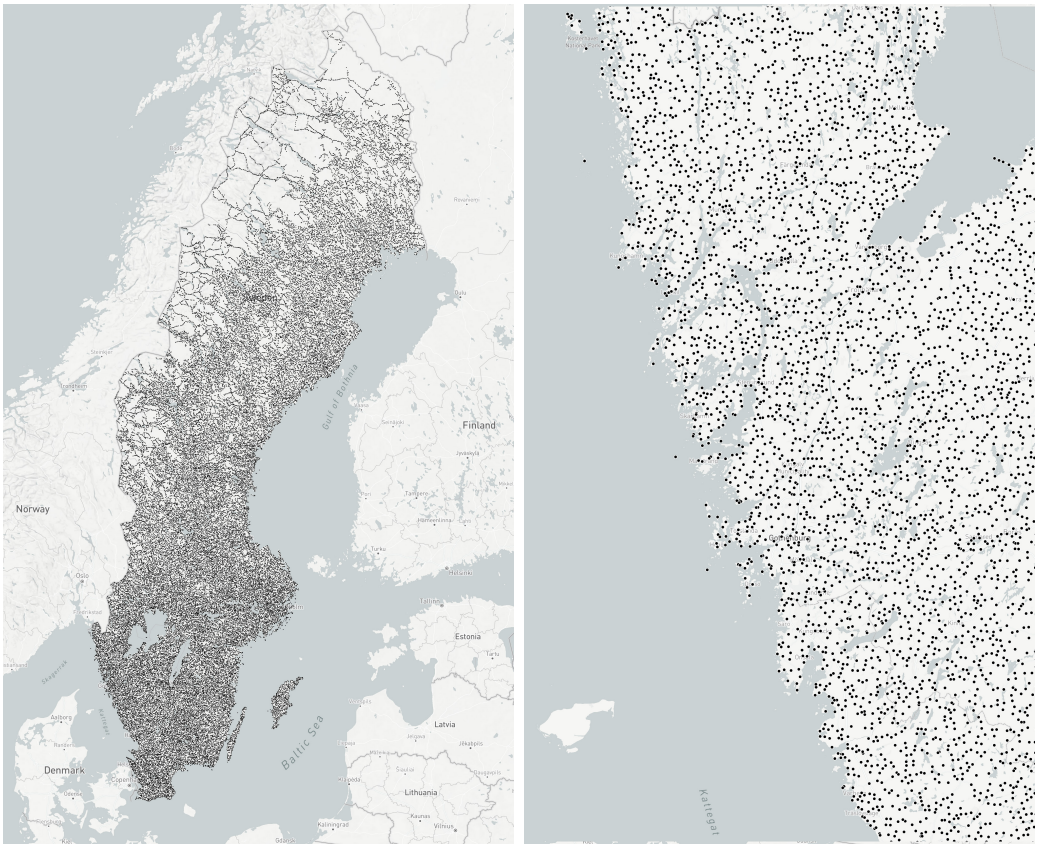


Figure 3.6: Sampled points by using the grid sampling method. Maps © Mapbox 2020 and © OpenStreetMap 2020.

Figure 3.6 shows the clustered result of the grid sampling approach. The nodes are somewhat evenly distributed, with urban areas containing more nodes than rural areas.

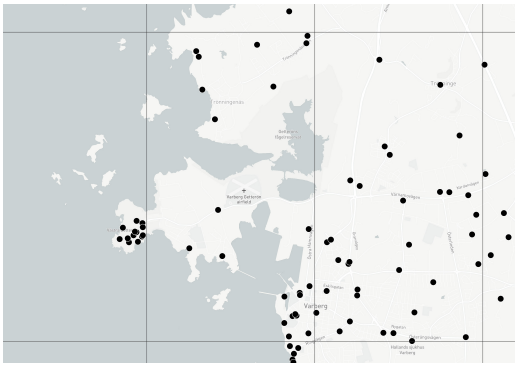


Figure 3.7: Splitting the map into grid sections caused problems in some areas. For example, areas that contained coast line would see a large amount of samples in a small area because most of the grid area is water. Map © Mapbox 2020 and © OpenStreetMap 2020.

This approach improved the sampling, but its results still exhibit the problem of nodes being too close to each other. This was especially apparent in areas where most of the box area consisted of water (see Figure 3.7). In order to space out samples more evenly, two distance-related factors were added to the sampling probability function. Two commonly used formulas for calculating distances on spheres and ellipses respectively are the Haversine (section 2.3) formula and Vincenty's formula [23]. The Haversine formula was chosen for distance calculations because it has a faster average running time [23]. The first factor p_{prox} uses the distance between current and the previous node in the way. With

$$\begin{aligned} n_{\text{prev}} &:= \text{Previous node in the way.} \\ n_{\text{curr}} &:= \text{Current node being considered.} \\ n_{\text{nearest}} &:= \text{Nearest already sampled node.} \\ D &:= \text{The maximum distance (2 kilometers)} \end{aligned}$$

We have

$$p_{\text{prox1}} = \min(15, \exp \text{haversine}(n_{\text{prev}}, n_{\text{curr}}))$$

The second factor uses the distance between the current node and the nearest already sampled node:

$$p_{\text{prox2}} = 2 \cdot \left(\frac{4}{(1 + \exp(-0.5 \cdot \text{haversine}(n_{\text{curr}}, n_{\text{nearest}})))} - 2 \right)^3$$

Figure 3.8 illustrates how the second probability factors impacts the sampling likelihood. The final probability of sampling a node in box i is computed as:

$$P(\text{Select current node from box } i) = p_{\text{prox1}} \cdot p_{\text{prox2}} \cdot \frac{N}{K \cdot \log(n_i)}$$

Using these modifications, the samples were more evenly distributed across the area. Figure 3.9 shows an example of sampling with these modifications and how the changes resulted in a more even distribution of samples, with Figure 3.10 showing an example of the improvement in areas consisting primarily of water (compare to Figure 3.7).

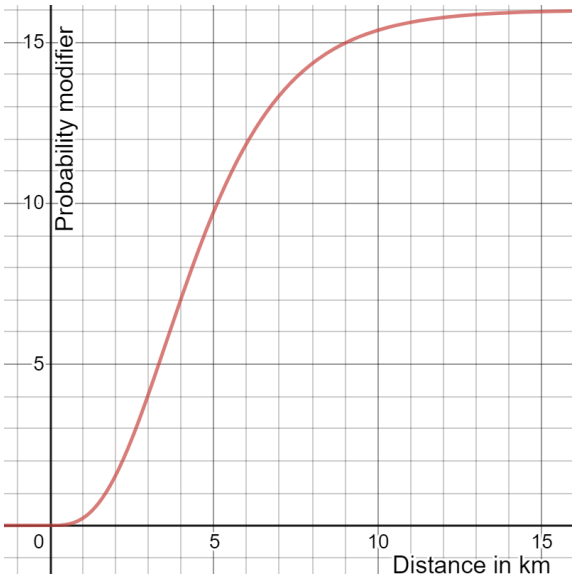


Figure 3.8: p_{prox2}: The probability of sampling a node increases with the distance to the last node.

Figure 3.9: An example of what the final sampling method produced. Each black dot corresponds to one sample. Map © Mapbox 2020 and © OpenStreetMap 2020.

Figure 3.10: Areas with lots of water no longer contained too many samples (each black dot corresponds to one sample). Map © Mapbox 2020 and © OpenStreetMap 2020.

3.2.2 Collecting additional data for under-represented categories

When images were sampled from locations given by this sampling scheme, forest and field images were heavily over-represented while locations close to water were under-represented. With only samples from this method, there were not enough water images present for the network to be able to properly classify water images. There are several reasons for why we encountered this problem (see chapter 5). Unbalanced data is a common problem in machine learning [41] and several methods exist to deal with this, for example redistributing the data [42].

We used a slightly modified version of the sampling approach described earlier to increase the amount of water images in the data set. The OSM data contains tags that can be used to filter out water nodes. For the sampling of points from which to sample images of (hopefully) water, we included only ways and relations that fulfill: Either the tag $natural = water$ is present, or the tag $water = v$ is present, where $v \notin \{basin, wastewater\}$. Except for this change, the sampling scheme was the same as the one described above (see subsection 3.2.1). This sampling scheme was used to collect additional images that contain water, augmenting our first data set to better represent water scenery.

3.2.3 Direction and field of view

When using the Google Street View Static API, it is possible to set the desired direction and field of view of the image. Because the scenery on the left and the

right side of the road might look different, we fetched one image in each direction. In order to get imagery that resembles the driver's field of view we calculated the direction of the road and fetched one image pointed 50° to the left and 50° to the right. The field of view was set to 100°. With this approach we were able to retrieve images that covered an angle of 100° (see Figure 3.11).

Figure 3.11: Images are fetched to the left and to the right of the driving direction. Images © Google 2020, map © Mapbox 2020 and © OpenStreetMap 2020.

For images sampled from the water points we made use of a slightly different approach. Since water is typically only present on one side of the road, for example if the road runs along a lake, these images are fetched using a different method. The direction of the fetched image was set to the angle between the water node and the nearest node in the Street View API that contained imagery (see Figure 3.12). Using this method, we only had to fetch one image for each water node.

Figure 3.12: Images that were sampled from the water node data-set had their direction pointed towards the water node in order to capture more water. Images © Google 2020, map © Mapbox 2020 and © OpenStreetMap 2020.

3.2.4 Classifying imagery

Image classification was done by feeding the images through a convolutional neural network that for each image outputs the probability of it belonging to each class. The nodes in the graph were then tagged with these scenery probabilities. These probabilities were stored in the format shown in Table 3.1 and Listing 3.1.

Name	Value	Description
forest	[0, 1]	A measure of the confidence that the image contains forest.
field	[0, 1]	A measure of the confidence that the image contains field.
water	[0, 1]	A measure of the confidence that the image contains water.
urban	[0, 1]	A measure of the confidence that the image contains urban features (e.g. houses).

Table 3.1: Numerical representation of the scenery in an image

```
{
    "forest": 0.85,
    "field": 0.1,
    "water": 0.05,
    "urban": 0.02
}
```

Listing 3.1: An example of scenery representation in JSON to be used by the routing algorithm.

Three different approaches were considered: training a network from scratch, using a pre-trained model and using a pre-trained model in combination with transfer learning. The first option was ruled out due to the extensive computational resources that would be required to train a network from scratch [43].

The second option we considered was to use a ResNet18 [**?**] network pre-trained on the Places365 [44] data set. The network outputs a probability score for 365 classes. A few of these classes were relevant when looking at sceneries along a route, while most of them were irrelevant (for example "praying", "eating" and "stressful"). One approach of handling these categories is to create a mapping from the predictions made by the network to the relevant classes of forest, field, water and urban. This approach was ruled out partly due to the large amount of irrelevant classes and partly because the training data contains few images similar to the ones taken from Google Street View.

The third and final option was to use a pre-trained model and make use of transfer learning in order to adapt the model to our data (see subsection 2.8.2). By using

a pre-trained network, ResNet50 [**?**], the time it takes to train a model can be significantly reduced [45].

3.2.5 Obtaining and categorizing training data

While there is an extensive amount of images available from Google Street View, the coverage of Sweden is not perfect. We therefore had to make sure that there was imagery available at the points that had been selected from the sampling before making the actual request for downloading them.

Another aspect that had to be taken into account was the cost of using the Google Street View Static API. A trade-off had to be made for downloading enough images to get meaningful data while staying within budget. Ideally, one would like to fetch images from every available point to get as accurate result as possible. Since this proved to be too costly, we had to limit the amount of data (see subsection 5.2.2).

The images were stored both locally and in a storage bucket in Google Cloud. For every image downloaded, we also saved an entry in a document database containing information about the coordinates of the image, which heading the camera was pointed at and what label that should be associated with the image.

The accuracy of convolutional neural networks generally improves significantly with larger training sets [46]. As such, it would have been ideal to classify as many images as possible. We had to make a trade-off between the number of images to label and the time we could spend on it. The number of images were selected so that the image labelling could be finished within one working day if we spent five seconds per image, the total number of images labelled ended up being 4774.

Since labelling images is a somewhat subjective task it was also desirable to have several people labelling the same images. To simplify the labelling task, we built a web frontend where images could be fetched from our backend, labelled by a person and stored back to the database. This allowed us to ask several people for help with the labelling task. The difference in how different people labelled the data was measured with Krippendorff's alpha [47] (see subsection 2.8.3).

Figure 3.13 shows the web frontend which was built using React [48] with Semantic UI [49]. Firebase [50] was used as the backend for storing the labelling results.

Apart from the image itself, the speed limit (where available) was also shown for each image. The speed limit was shown because certain images would benefit from the user having access to additional data. The user were given the choice to classify the image into one of the categories *forest*, *field*, *water*, *urban* and *miscellaneous*. Images categorized as *miscellaneous* were not used for training the network. The users could also tag an image with additional information as well as adding their own tags. These tags ended up not being used in the final product due to time constraints, see chapter 5.

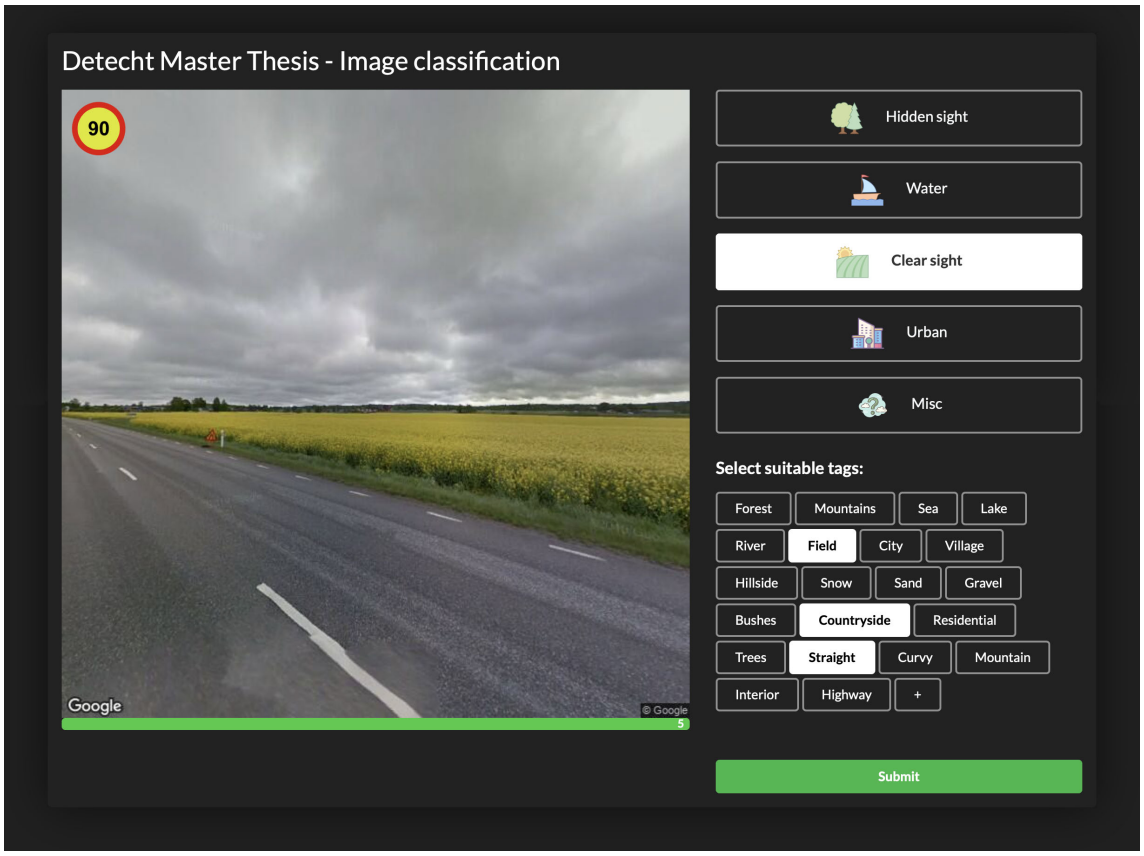


Figure 3.13: The frontend for image classification. The highest allowed speed is shown in the top left corner of the image. To the right of the image the labeller is able to select one of five categories. Image © Google 2020.

3.2.6 Training the network

The implementation of the network was mainly done in PyTorch [51] where ResNet50 is included as a pre-trained model. Before feeding the images to the network, a number of random transformations was applied on the training set. The images were first scaled down to 256x256 pixels, then rotated a random amount between -15 and +15 degrees. Each image where then flipped horizontally with a probability of 0.5 before the center 214x214 pixels of the image was cut out (see Figure 3.14). The reason for doing this transforming and distorting the images was to prevent over-fitting and making the network generalize better, even on a relatively small training set. It also contributes to making the feature detection part of the network more versatile [52].

During training, the weights of all layers were frozen except for the last layer, which was replaced with a new set of layers. The final layer of this new set of layers was a softmax layer with four outputs, one for each scenery category. The final softmax layer produces values in the range $[0, 1]$ which can be interpreted as a probability of the image belonging to that category [32].

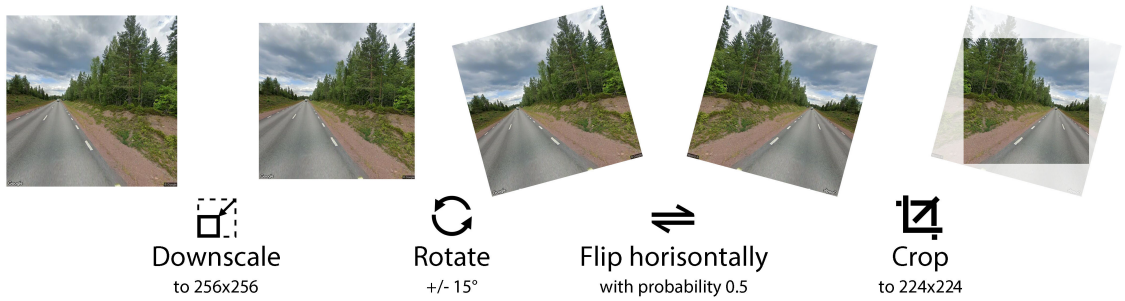


Figure 3.14: Transformations are applied to images in the training set in order to make the network generalize better. Images © Google 2020.

The subset of the data that had been manually labelled was split into a train, test and validation set using a 60% of the images for training, 20% for test and 20% for validation. The network was trained for 20 epochs and then evaluated on the validation set which was used to tune the hyper parameters [53]. Figure 3.15 and Figure 3.16 show the results of this training.

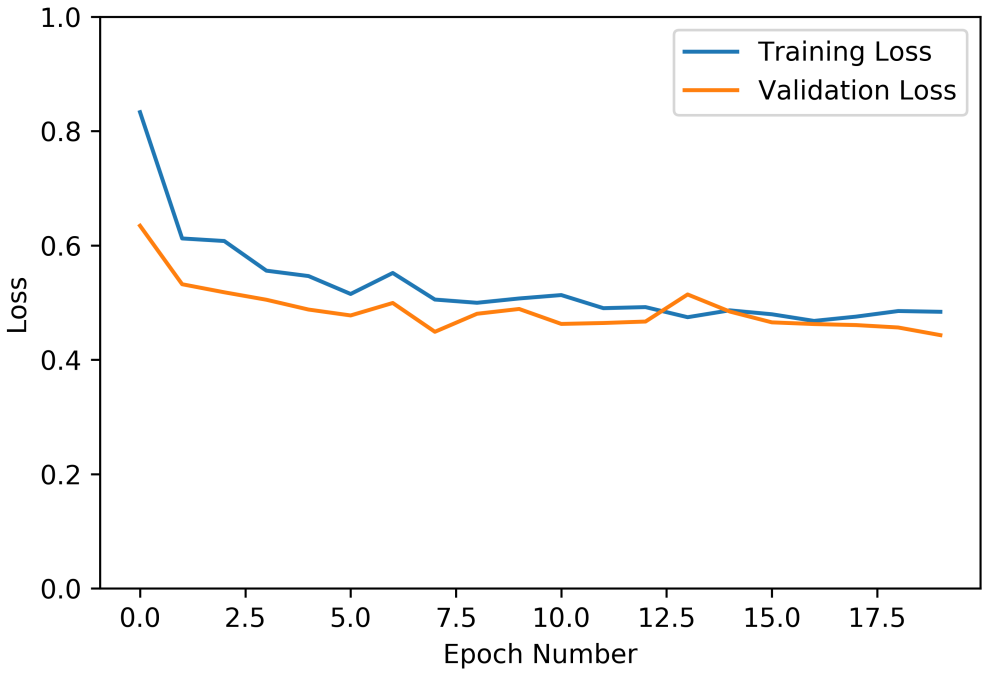


Figure 3.15: Loss curves of the model for image classification.

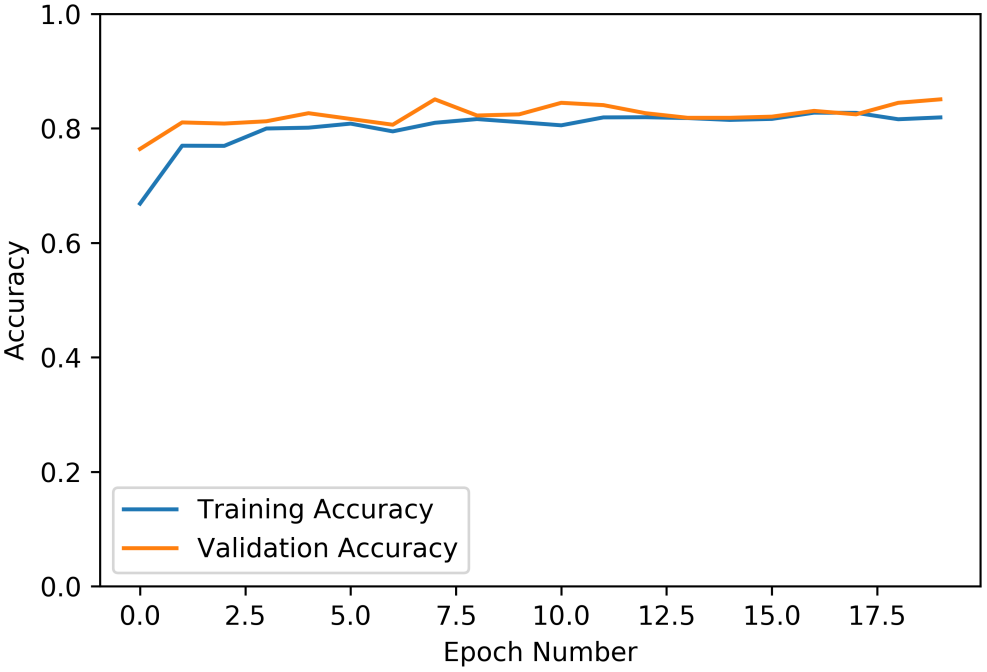


Figure 3.16: Accuracy curves of the model for image classification.

3.3 Incorporating scenery and curvature data

OSM data for Sweden was downloaded from the Geofabrik website. This data had to be augmented with curvature and scenery data so that the routing algorithm could access it. Because of the size of the OSM data for Sweden, it cannot all be stored in memory at the same time on a typical machine. Moreover, the ways and relations of this data only hold reference ID's to their nodes. A lookup has to be performed to retrieve information about a node such as its coordinates. The lookups were made possible by storing the OSM data in a PostgreSQL database with the PostGIS extension. The tool osm2pgsql [54] was used to load the OSM data into the PostgreSQL database.

3.3.1 Adding curvature to the OSM data

A curvature data point is a value $c \in \mathbb{R}^+$ with higher values being better, i.e. more curvature. The curvature value for a way is calculated by segmenting the way into node triples $\{(n_0, n_1, n_2), (n_1, n_2, n_3), \ldots, (n_{k-2}, n_{k-1}, n_k)\}$ where k is the amount of nodes in the way. These node triples are then used to calculate the curvature using the radius of their circumscribed circle. If the circumcircle radius of a segment exceeded 10 000 meters, that segment was considered to be straight. Listing 3.2 contains pseudo code for how the curvature value of a way is calculated.

```
1  total_distance = 0
2  weighted_distance = 0
3
4  prev_segment_length = 0
5  prev_radius = 10 000
6
7  for i = 1, k:
8      n2 = nodes[i - 1]
9      n3 = nodes[i]
10
11     segment_length = haversine_distance_on_earth(n2, n3)
12     total_distance += segment_length
13
14     if (i == 1)
15         weighted_distance += segment_length
16         prev_segment_length = segment_length
17         continue;
18
19     n1 = nodes[i - 2]
20
21     base_length = haversine_distance_on_earth(nodes[i-2],
           nodes[i])
22     radius = min(10 000, circumcircle_radius(
```

```
23          prev_segment_length,
24          segment_length,
25          base_length
26      )
27
28      weighted_distance +=
29          segment_length * get_curvature(min(radius,
               prev_radius))
30
31      prev_radius = radius
32      prev_segment_length = segment_length
33
34  way_curvature = weighted_distance / total_distance
```

Listing 3.2: Pseudo code for calculating curvature. The *way_curvature* value is used as the curvature value for the way.

3.3.2 Adding scenery to the OSM data

A scenery data point is a vector of real values $[s_0, s_1, \ldots, s_T], s_t \in [0, 1]$ where T is the amount of scenery categories. The scenery data of a way is a combination of the scenery data along its nodes. Since only a fraction of all nodes have scenery data associated with them (see subsection 3.2.1), the sceneries for nodes without associated scenery data were computed as a combination of the at most three[1] nearest[2] nodes containing scenery data.

Let N_w be the nodes of way w. With S being the set of the at most three nearest scenery samples, the value V_n^t for scenery tag t for a node n is computed as $V_n^t = \max_{s \in S} s^t$, where s^t denotes the value for scenery category t at node s. For a complete way w, the scenery category value $\hat{V}_w^t$ for scenery category t is computed as the average of the scenery category values of its nodes:

$$\hat{V}_w^t = \frac{1}{|N_w|} \sum_{i=1}^{|N_w|} V_i^t$$

The final scenery of the way is $U_w = \{\hat{V}_w^1, \hat{V}_w^2, ..., \hat{V}_w^T\}$. This set of values are used when routing for the different routing profiles that rely on scenery data (see subsection 3.4.3).

Figure 3.17 shows an overview of the locations that were given scenery data using this approach.

[1] There may be fewer than three if the nearest scenery samples are too far away.

[2] The nearest node queries were performed using an R-Tree implementation called *jsi*, available at `https://github.com/aled/jsi`.

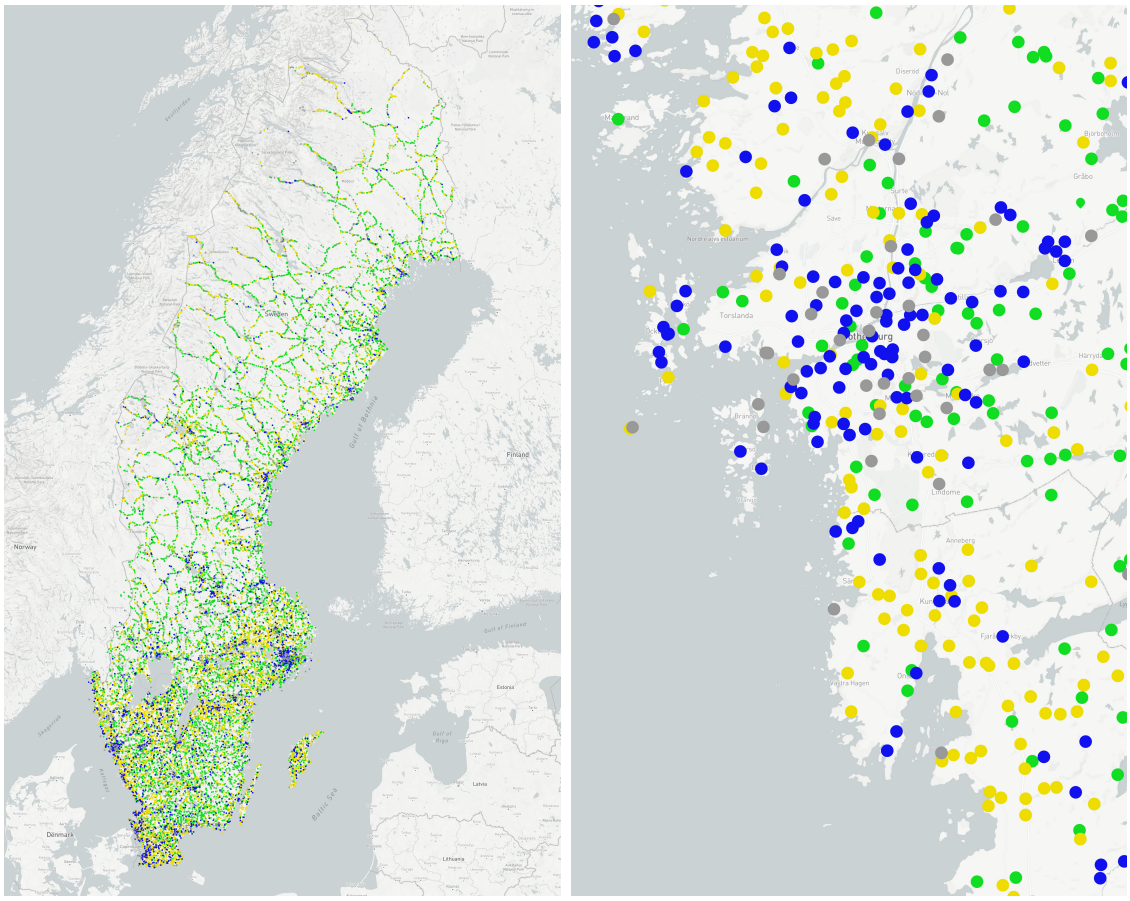


Figure 3.17: Locations with scenery data. Green points correspond to forest, yellow points to field, blue points to water and grey points to urban. The rightmost image shows a zoomed-in view of the scenery locations around Gothenburg. Maps © Mapbox 2020 and © OpenStreetMap 2020.

3.4 Routing

3.4.1 Choice of routing engine

There are several routing services available online such as Google Routes and Mapbox Directions. These services are suitable for general routing, but the requirements of this project called for more customizability, hence the need for open source software. At the time of writing, there were two main open source routing engines; Project OSRM written in C++ and Graphhopper written in Java. Graphhopper was chosen because of the authors familiarity with the Java language. The GraphHopper routing engine supports the three routing modes *flexible mode*, *speed mode* and *hybrid mode*. The flexible mode allows for more customization on a request basis, but queries for points that are far apart can take minutes to complete. The speed mode makes use of *contraction hierarchies* [55] and is around 10 times faster [56]. The speed mode for routing has been used throughout the project.

3.4.2 Route weighting function

The routing algorithm tries to minimize the total weight of the route. The final weight w for an edge is calculated as

$$\begin{aligned} w_r &= \frac{\text{road\_distance}}{\log(\text{road\_max\_speed})(0.5 + \text{priority})} \\ w &= f(\text{curvature}) \cdot g(\text{scenery}) \cdot w_r \end{aligned} \tag{3.1}$$

where $f := \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ (a function of curvature), $g := \mathit{Scenery} \longrightarrow \mathbb{R}^+$ (a function of scenery), and priority is a modifier based on the road type of the OSM way. The logarithm of the speed is used to decrease the impact of highways on the weight, as we generally want to avoid highways. The functions f and g can be adjusted based on preference. For instance, they could be chosen to reward a curvy road that does not encounter many urban areas. The road type of a way is given in OSM by the string value r_v of the *highway* key[3]. The priority is calculated based on the sets in Listing 3.3 as[4]:

Listing 3.3: The avoid set and the prefer set for the priority parameter.

```
avoidSet = {'highway', 'secondary', 'tertiary'}

preferSet = {
    'motorway', 'trunk', 'motorroad', 'residential',
    'living_street', 'service'
}
```

$$x = \begin{cases} 0, & \text{for } r_v \in \text{avoidSet} \\ 7, & \text{for } r_v \in \text{preferSet} \\ 4, & \text{otherwise} \end{cases}$$

$$\text{priority} = \frac{x}{7}$$

3.4.3 Routing profiles

Routing in GraphHopper is done based on profiles. A profile consists of a vehicle configuration and a weighting configuration. The vehicle configuration contains vehicle-specific routing parameters. Some examples are default speeds for different

[3]The OSM highway key is described here.

[4]These priority values were used because they are standard priority values for GraphHopper routing (see PriorityCode).

road types, penalties for road surface, and road accessibility information (for instance, pedestrian paths should be excluded from motorbike routing). In order to speed up routing, GraphHopper performs preprocessing once for each profile. Because this preprocessing takes some time, it places a limit on the amount of profiles that can be used in practice. Through experimentation, we settled on the following profiles that use the curvature and scenery data (c is curvature, w_r is the regular weight given in Equation (3.1), and S_c is the scenery value for scenery category c):

1. Very low curvature, $w = \frac{8}{c^3} \cdot w_r$
2. Low curvature, $w = \frac{4}{c^3} \cdot w_r$
3. Medium curvature, $w = \frac{2}{c^3} \cdot w_r$
4. High curvature, $w = \frac{1}{c^3} \cdot w_r$
5. Forest with high curvature, $w = \frac{1}{c^3} \cdot \frac{6}{\exp(3 \cdot S_{\text{forest}})} \cdot w_r$
6. Field with high curvature, $w = \frac{1}{c^3} \cdot \frac{6}{\exp(3 \cdot S_{\text{field}})} \cdot w_r$
7. Water with high curvature, $w = \frac{1}{c^3} \cdot \frac{6}{\exp(3 \cdot S_{\text{water}})} \cdot w_r$

A preprocessed graph was generated for each of these profiles using contraction hierarchies. The preprocessing for these profiles took around 15 minutes in total.

3.5 Safety aspects

Motorcycle accidents are commonly caused by road user mistakes, both from the motorcycle driver and from other road users [57]. A study on motorcycle accidents in the UK found that junctions are a place where other road users are particularly bad at detecting motorcycles. Motorcycle drivers themselves are more likely to cause an accident in curves or when trying to overtake another vehicle [57]. The study points at education and driver awareness being an important tool to reduce accidents. In addition to this, it also mentions the ability of the rider to plan ahead as important, especially when the rider is approaching a curve.

Exploring ways of changing the behavior of motorcycle riders and other road users is beyond the scope of this study. Therefore, we instead focus on warning the rider when potentially dangerous road sections are approaching, such as curves and junc-

tions. Additional factors that can be taken into account for the dynamic warnings system are discussed in chapter 7.

The warnings can be relayed to the driver in different ways. Some possible ways of doing it are auditory, haptically, and visually. Out of these three ways, haptic and auditory feedback is more effective and also preferred by riders over visual feedback [11; 58].

3.5.1 Curvature warnings

When approaching a curve it is important for the driver to adapt their speed accordingly so that it can be passed safely. To determine whether or not to give a warning, we looked at three factors: the current location of the driver, the curvature of the upcoming road section and the speed at which the driver is moving. The location of the driver and its speed can be obtained via the GPS-module in the user's phone. Upcoming coordinates are given by the routing engine and updated continuously.

For each update of the coordinates a "danger"-value is calculated as follows:

$$\text{danger} = \frac{\text{speed}^{W_s}}{W_c \cdot \text{radius} + \text{distance}}$$

$$W_s := \text{Weighting of current speed.} \tag{3.2}$$
$$W_c := \text{Weighting of curve radius.} \tag{3.3}$$

For the purpose of testing the algorithm we tried a few different approaches for how to calculate the danger-value and what the constants should be. While being usable for testing the interface, the calculation of the danger-value is subject for future work and the constants will have to be tuned in order for the algorithm to be useful in real life situations. The curve radius weight W_c was set to 1.8 and the speed weighting W_s was set to 2.0. More about this under subsection 5.2.3.

To test how this works in a real life setting we implemented a simple version of it in the Detecht app (see Figure 3.18). The features were only visible to selected test users.

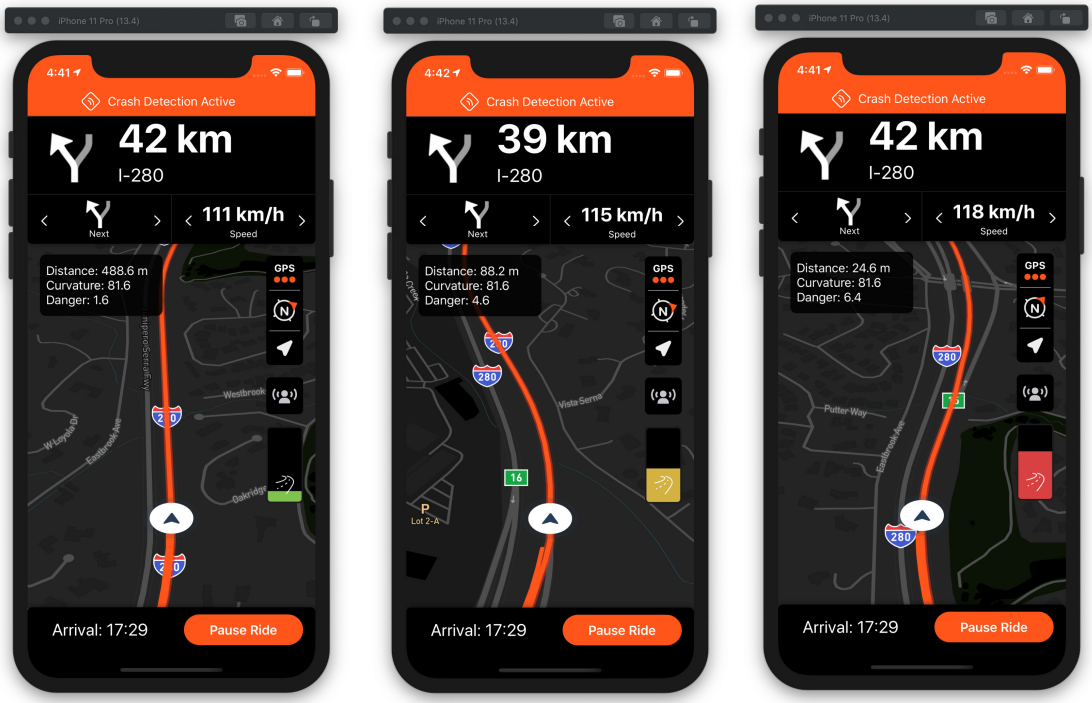


Figure 3.18: Example of curvature warning in the app. The box in the upper left corner of the map shows the values that the danger-value is based on, together with the current speed. These values are not intended to be shown to the end user but are only used for reference during development. The screenshots show three different levels of danger; low, medium and high. These danger levels are represented as green, yellow and red in the bar on the right-hand side. The height of the bar also indicates the current amount of danger as described earlier. Maps © Mapbox 2020 and © OpenStreetMap 2020.

3.5.2 Nearby road works

Road networks all over the world undergo constant transformation. New roads are built and existing roads are being repaired and rebuilt. For a motorcyclist, entering an area with an ongoing construction might pose increased danger as well as making the driving more difficult. By utilizing an open API from Trafikverket, road warnings are fetched and displayed to the user as shown in Figure 3.19.

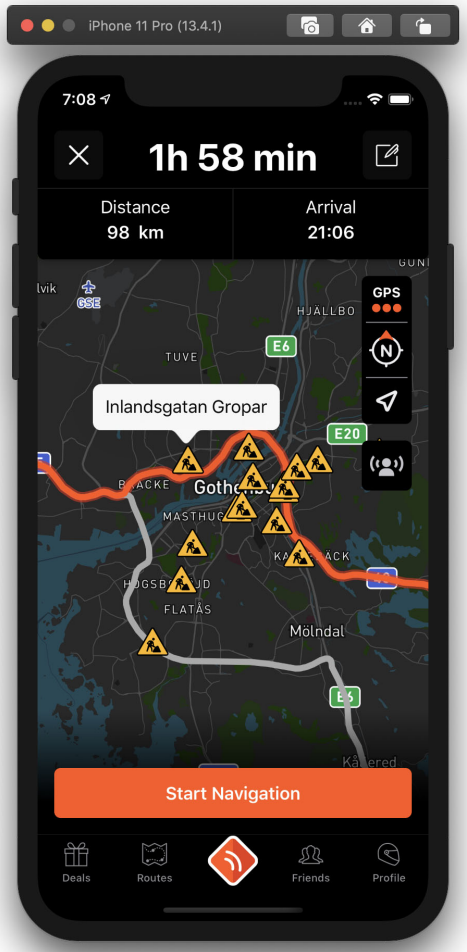


Figure 3.19: A prototype of road warnings. Only warnings in close proximity to the route are shown. Details are shown when the user taps a warning symbol. Map © Mapbox 2020 and © OpenStreetMap 2020, Icons © Icons8 https://icons8.com/.

4

Results

4.1 Scenery image classification

4.1.1 Agreement between the labellers

The agreement between labellers (n = 2) was measured by Krippendorff's α. A high value indicates a high level of agreement. As discussed in section 5.1, some images are difficult to label correctly since they can fall into several categories. The result for Krippendorff's α, 0.631, is to be considered as rather bad which we believe is due to the difficulty of labelling the content of images into only one categorys.

Percent Agreement	**Krippendorff's α (nominal)**	**Agreements**	**Disagreements**	**Total cases**
75.7%	0.631	1209	389	1598

4.1.2 Size of training set

Due to time and monetary constraints we had to limit the number of images used for training. The results, presented in and Figure 4.1, show how the accuracy (in %) of scenery classification varies with the size of the data set used for training the neural network. A number of training sessions were carried out in order to find out whether or not a bigger training set would yield more accurate results. It turned out this was not the case; increasing the number of images used for training only has a marginal positive effect on the accuracy.

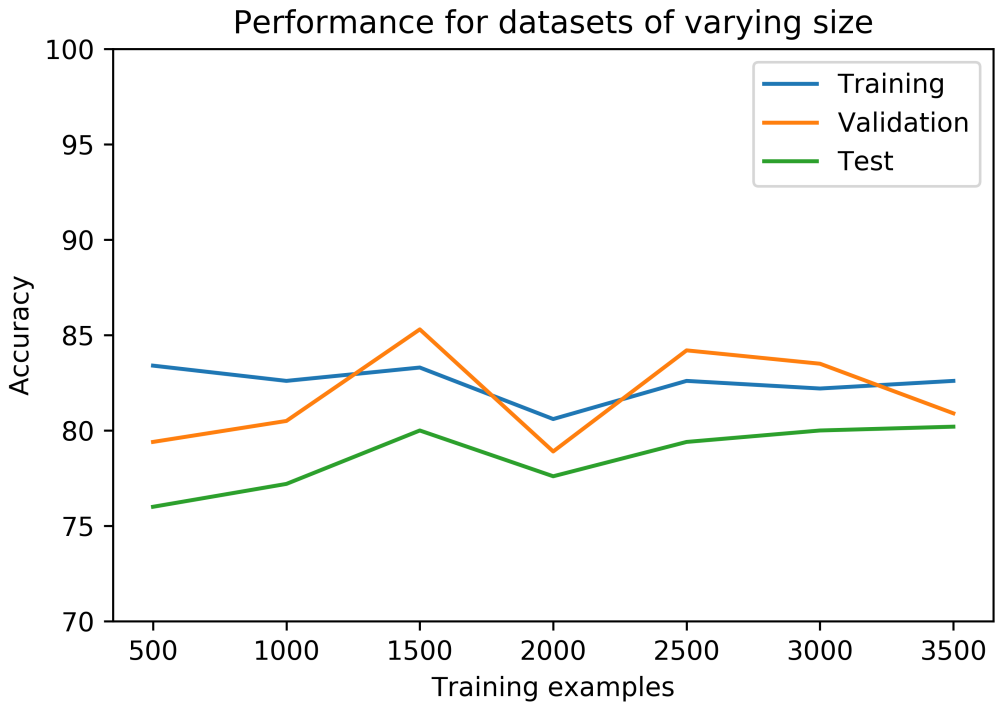


Figure 4.1: Performance of the network when using data sets of varying size.

4.2 Routing

The routing was performed on a 64-bit Windows 10 machine using the Intel(R) Core(TM) i5-8400 CPU. On this machine, the routing runs in real time for each profile. The longest routes possible, i.e. from the southernmost part of Sweden to the northernmost part, are calculated in less than 100 milliseconds for any of the profiles. Table 4.1 shows typical routing times for routes of varying length.

Table 4.1: Running times for routing with any profile where the resulting route has a certain length.

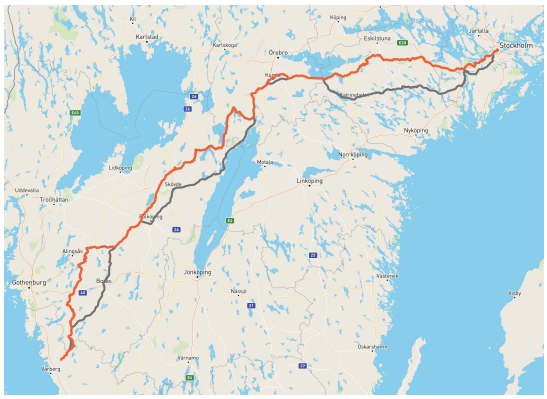
	5 km	50 km	500 km	1500 km
Any routing profile	∼1 ms	∼2 ms	∼20 ms	∼30 ms

4.2.1 Comparing different level of curvature

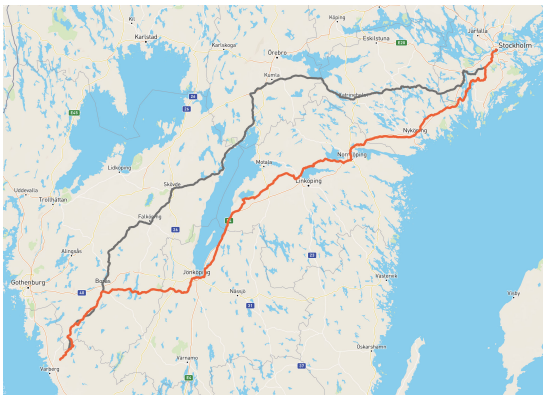
As described in subsection 3.4.3, the curvature weighting parameter can be adjusted to a desired level of curvature. Figure 4.2 shows comparisons between the fastest route in grey and routes with varying level of curvature in orange.

(a) Routing profile: High curvature

(b) Routing profile: Medium curvature



(c) Routing profile: Low curvature



(d) Routing profile: Very low curvature

Figure 4.2: Different routes from Gothenburg to Stockholm, each route created using one of the curvature routing profiles. The orange line in each figure corresponds to the curvy route, and the gray line corresponds to the fastest route. Maps © Mapbox 2020 and © OpenStreetMap 2020.

4.2.2 Comparison to other routing services

A number of comparisons were done between routes generated using the algorithms in this thesis and existing routing services on the market. We chose to compare against two popular services, namely Calimoto [16] and Kurviger [17]. The route used in these comparisons starts in Halmstad and ends in Uppsala. This route was chosen due to the many alternative routes that are available between these cities.

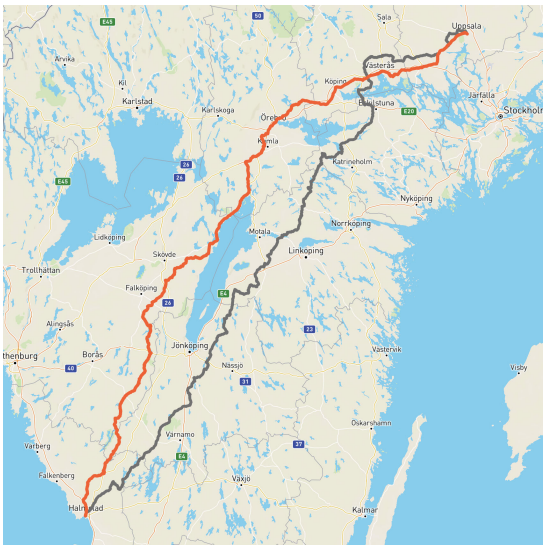
4.2.2.1 Calimoto comparison

Figure 4.3 compares our curvature routes with routes from Calimoto created with their "super twisty route" option enabled.

(a) Routing profile: High curvature

(b) Routing profile: Medium curvature

(c) Routing profile: Low curvature



(d) Routing profile: Very low curvature

Figure 4.3: Our curvature routes compared to the route generated by Calimoto, between Halmstad and Uppsala. The orange line corresponds the route generated by our algorithm and the gray line is the route generated by Calimoto. Maps © Mapbox 2020 and © OpenStreetMap 2020.

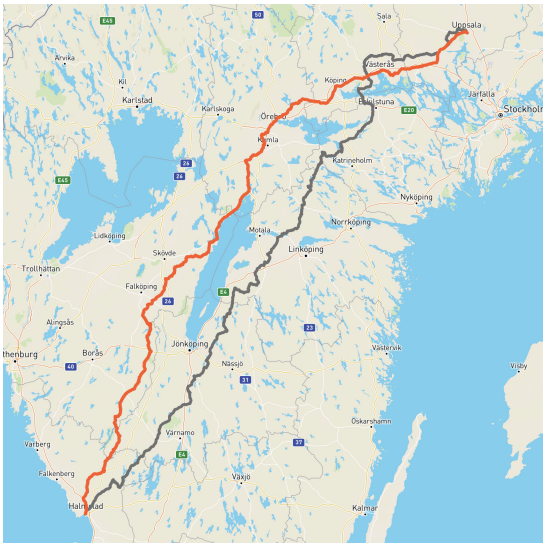
4.2.2.2 Kurviger curvature comparison

The route from Kurviger was created using the "Extra curvy" option. Figure 4.4 shows comparisons between the route generated by Kurviger and routes generated using our curvature profiles.

(a) Routing profile: High curvature

(b) Routing profile: Medium curvature

(c) Routing profile: Low curvature



(d) Routing profile: Very low curvature

Figure 4.4: A route from Gothenburg to Stockholm, calculated using four different routing profiles. The orange line corresponds to a curvy route, and the gray line corresponds to the fastest route. Maps © Mapbox 2020 and © OpenStreetMap 2020.

4.3 User evaluation of routes

In order to evaluate routes generated by the algorithm, we invited six users that had previously shown interest in being beta testers of the Detecht app. Each person was shown 7 different routes between two locations and were asked to select their top three choices from these routes.

Since some of the alternatives are very similar we also gave them the option to let several route alternatives have the same ranking in the list. The participant was asked to rate the routes based on how likely they were to ride each route. For example, if the user thought that the forest-route and the field route was the best routes and could take either one of these, he or she could assign both with a ranking of 1. Table 4.2 shows the results of the user evaluation.

A common opinion from the users in the test group was that they often took many more factors into consideration when choosing which route to take. Examples of these were the road surface, amount of traffic on the road, weather conditions and the time of day they are going for a ride.

	Forest	Field	Water	Curvature	Fastest	Kurviger	Calimoto
Uddevalla Strömstad	3	2	2	2		1	1
Helsingborg Kristianstad	2	3		3		1	1
Kungälv Varberg	2	2	2	2	3	1	1
Stockholm Uppsala	1				3		2
Malmö Ystad		3	3			2	1
Örebro Södertälje	1	2	2	2		3	1
Göteborg Jönköping		3	2			1	

Table 4.2: Top three route choices by users for different routes, with 1 being the top choice.

Table 4.3: Warning system feedback from the participants in the survey.

	Positive	**Negative**
Visual	Easy for the driver to understand and correctly interpret warnings.	Might draw attention from the road and make risky situations even more risky.
Audible	Can be used to give detailed feedback about the specific danger approaching, for example a sharp curve or a road work.	Requires the driver to use a headset. Can be distracting if it is poorly implemented.
Haptic Phone	Could be used to give feedback without distracting the attention from the driver	Might be difficult to feel. Only works if the driver carries the phone close to their body, i.e. not if it is attached to handlebar.
Haptic Watch	Always close to body so the haptic feedback would possible to feel even when the phone is mounted on the handlebar. Could work well in conjunction with other types of warnings.	Might also be difficult to perceive while driving. Difficult to feel the difference between a curvature warning and other types of haptic feedback. Creates a need for the user to be well aware of how the warning system works in order to interpret the warnings.

4.4 User evaluation of warnings

To evaluate warnings, a prototype was implemented in the Detecht app and shown to test users. Their evaluation of the system was done at the same time as they evaluated the different routes. While more extensive testing is required before deploying such warnings systems, the initial feedback showed that it would be beneficial to have access to dynamic warnings. One common feedback was that a warning system like this would give them more time to plan their driving and avoid making use of hard breaking which might itself cause an accident.

One important aspect of the warning systems was how to deliver the warnings to the users. Four different ways were suggested: visually on screen, audible via headphones, haptic feedback via the phone and finally haptic feedback through a smartwatch. The feedback from the participants in the survey was compiled into a table of positive and negative aspects for each solution (see Table 4.3).

5

Discussion

The main aim of this study was to create a routing system based on enjoyability factors. Results from the initial survey helped confirm our suspicions that curvature and surrounding scenery were both important factors for an enjoyable route. The number one most important factor for the users was that a route contained "a lot of curves". As shown in the results, we have created a routing system that takes curvature and scenery into account. In the user evaluation, where our routes were compared to those of kurviger.de and Calimoto, most users preferred the kurviger.de and Calimoto routes.

A common opinion among the participants of the evaluation was that our routes often included larger and straighter routes while the routes of the other services favored smaller, more curvy ones. This suggests that the algorithm should perhaps prefer curvy routes to a higher degree. It also suggests that larger roads should receive higher penalties. For instance, with the current algorithm, the weights of roads that are tagged as highway, secondary, or tertiary is doubled. Better results might be achieved by further increasing the weight penalties of such roads.

The participants were positive to the concept of using the surrounding scenery as a factor for routing. However, this positivity did not translate to a preference of routes that used scenery as a factor for routing. There are numerous reasons for why this might be the case. We could identify four possible causes:

1. **Lack of data** - The data used is too sparse leading to not being enough scenery data points to make a significant impact on the routing.

2. **Poor classification** - The accuracy of the classifier might be too small leading to unreliable scenery data.

3. **Poor algorithm** - The algorithm might not use the data in an adequate way.

4. **External factors** - The classification of the routes was correct but other factors such as heavy traffic or road conditions make them undesirable to ride.

Of these, poor classification is likely the least valuable to try to improve. With the

method we used, we were able to achieve an accuracy of around 80 %, which means that the gains that can be made here are limited. Furthermore, while the algorithm can undoubtedly be improved, its efficiency is limited by the amount of available data. That leaves lack of data as the obvious issue to address first. With the current amount of scenery data, many route nodes do not have any nearby scenery. Adding more data here should improve the routing that is based on scenery.

The external factors could definitely be improved, for example by taking into consideration the current traffic situation and evaluating the road conditions.

While only a handful of people participated in the route evaluation, its results do indicate that further improvements are necessary to perform better than existing services.

5.1 Image labelling

Labelling the images turned out to be a more complex task than we first anticipated. While many of the images clearly consisted of forest, water or free-sight (see Figure 5.1), there are also a significant amount of images in the training set that might be placed somewhere in between these.

Figure 5.1: Examples of images that are easy to classify. Images © Google 2020.

Figure 5.2: Examples of images that are difficult to classify. Images © Google 2020.

Figure 5.2 shows three images that are more difficult to classify. The first image is easy for a human to classify as field. However, with a tree obstructing a large portion of the image, there is a risk that the network will wrongfully classify this as forest. For the second image we have a road running along the water but only a small portion of the water is visible in the image. We believe that this kind of images are difficult to classify both for a human labeller and for the neural network. One could argue that the image can be classified both as forest and water. The last image contains two types of scenery; forest on the left side on the road, and field on the right side of the road.

5.2 Limitations

5.2.1 Geographical area

Only data from the geographical area of Sweden was used in this project. We chose Sweden due to our familiarity with its landscape and some of its roads. This allowed us to analyze roads that we had driven before and make rough estimates of the quality of our results in the early stages of the project. Using a larger area such as all of Europe would be possible, but doing so would add a considerable amount of time to the preprocessing of graphs. It would also complicate the scenery classification by bringing in a more diverse range of possible sceneries.

5.2.2 Scenery classification

The scenery classification is based on an image data set that consists of images taken along Swedish roads. The Google Street View Static API was used to retrieve this image data set, mainly because it is a large data set. However, only a subset of this data set could be used due to Google's pricing plans. Around 60 000 images were sampled from this data set and used to determine the surrounding scenery of roads across Sweden.

There was a limitation in how many images we would be able to use for the training set, mainly due to the time it took to classify each image manually. As described in subsection 4.1.2, the performance of the model does increase with a larger training set, but the increase seems to level out. We limited ourselves to labelling 4774 images and we don't believe that using a larger data set would have made a significant impact on the performance of the model.

Ideally one would like to have at least two different persons labelling the images. If the labels do not match up the labellers could go back and re-evaluate them. Due to time constraints we only had multiple labellers for a subset of the training data.

5.2.3 Curve warnings algorithm

The curve warning algorithm currently only works if the user drives along a predetermined route. If the user makes a detour from this route or for some reason is

unable to follow it the route must be re-calculated before it is possible for the driver to receive curve warnings. The main reason for doing this implementation was so that we can receive a list of upcoming coordinates. A more generalized solution could take the current position and heading of the user and calculate a probability for that the driver would be riding towards a certain coordinate.

6

Conclusion

The aim of this research was to create a routing system that could recommend routes based on enjoyability factors such as curvature and the surrounding scenery of the route. We have proposed a system that can recommend enjoyable routes to motorbike riders based on the curvature of the route and its surrounding scenery. The system makes use of OpenStreetMap data and images from Google Street View. While the results we show are based on data from within Sweden, the routing system can be adapted to work in other geographical areas as well. The evaluation of the results indicates that our solution performs worse than existing alternatives. The comments from participants of the final evaluation suggest that the routing algorithm should perhaps favor roads with high curvature even more than it already is. The initial results of using surrounding scenery for routing is promising, but we suspect that more data is necessary to make significant improvements to routing by using images of the surrounding scenery of the road.

We have also developed a concept for informing riders about safety aspects of their route, such as if they are approaching a curve too quickly, as well as information about road works along the route. Initial results shows promise but need more evaluation before being implemented into a real world product.

7

Future work

We suggest that the first step for continuing this work is to investigate whether easy gains can be made in routing quality. As we see it, the primary avenues for such gains are an even more favorable weighting for roads with high curvature, and using more scenery data to hopefully improve the quality of the scenery-based routing. Finally, these changes should be evaluated more thoroughly than the evaluation we were able to produce for this work, in order to increase confidence in the viability of this routing approach.

In addition to these possible improvements, there are several ways in which this work can be scaled up or otherwise improved:

Expanding the area The geographical area of this project is limited to the Swedish road network (see subsection 5.2.1). It is highly likely that road networks in different parts of the world have other aspects that need to be taken into consideration when designing a routing algorithm. A suggestion for future work is to explore how to scale up to larger areas.

Adding more types of scenery Only commonly occurring sceneries in Sweden were taken into consideration for the system that classifies the surroundings of a road. In other areas of the world this system might therefore not be applicable. For this system to be applicable to other geographical areas, it would have to be changed. This change could either be to include more types of scenery, or to adapt the scenery used for training of the neural network based on the area of interest.

Additional routing factors When evaluating the results of the routing algorithm, it became evident that drivers often take into consideration more factors than curvature and scenery. These factors included for example road surface, amount of traffic, the current weather and time of day. A suggestion for future work is to include these in the routing engine in order to generate more tailored route choices.

Dynamic warnings There are numerous additional factors that can be taken into account for the dynamic warnings system. Apart from looking at the curvature of the road in two dimensions, we could also factor in the change in elevation of the

road since this place a role in what the driver can see. Additionally, when the driver passes over the top of a hill, the force of the vehicle against the road surface will be lower than when driving on a flat road.
The current weather conditions can also play an important role. If the dynamic warning system had awareness of the current weather conditions it would be possible to warn the user when the drivers of approaching traffic experienced back-light. Using the information about current and previous precipitation in combination with road elevation would allow the system to warn for potentially dangerous water retention along the route.
Apart from road works along the route, it might also be valuable to warn the user when he or she passes through an area with schools or hospitals so they can be more aware of the surroundings.

More thorough evaluation More thorough tests and evaluations using a larger test group would be useful for further evaluating the results and how they compare to other existing solutions. It proved difficult to evaluate the routes in an objective manner due to the vast amount of factors that drivers take into account when choosing a route.

If the routing system was put into use, another way of evaluating the system would be to track how often users choose routes generated by our algorithm. If users often choose our routes, it is indicative of that the routing system performs well.

Bibliography

[1] R. Frash Jr, J. Blose, W. Smith, and K. Scherhag, "A multidisciplinary marketing profile of motorcycle tourists: explorers escaping routine to find flow on scenic routes," *Tourism Recreation Research*, vol. 43, no. 4, pp. 432–444, 2018.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, pp. 269–271, Dec. 1959.

[3] G. Gallo and S. Pallottino, "Shortest path algorithms," *Annals of Operations Research*, vol. 13, no. 1, pp. 1–79, 1988.

[4] M. Potamias, F. Bonchi, C. Castillo, and A. Gionis, "Fast shortest path distance estimation in large networks," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, (New York, NY, USA), p. 867–876, Association for Computing Machinery, 2009.

[5] B. Leder, "Parameters and characteristics of motorbike rider related route determination," *Vienna, Austria, University of Applied Sciences Technikum Wien*, 2011.

[6] O. Njå and S. M. Nesvåg, "Traffic behaviour among adolescents using mopeds and light motorcycles," *Journal of Safety Research*, vol. 38, no. 4, pp. 481 – 492, 2007.

[7] M. T. Yousif, A. F. M. Sadullah, and K. A. A. Kassim, "A review of behavioural issues contribution to motorcycle safety," *IATSS Research*, 2019.

[8] F. Malin, I. Norros, and S. Innamaa, "Accident risk of road and weather conditions on different road types," *Accident Analysis Prevention*, vol. 122, pp. 181 – 188, 2019.

[9] S. Hecker, A. Liniger, H. Maurenbrecher, and D. Dai, "Learning a curve guardian for motorcycles," *IEEE Intelligent Transportation Systems Conference (ITSC)*, 07 2019.

[10] World Health Organization, *Global status report on road safety 2015*. World Health Organization, 2015.

[11] M. Song, S. McLaughlin, and Z. Doerzaph, "An on-road evaluation of connected motorcycle crash warning interface with different motorcycle types," *Transportation Research Part C: Emerging Technologies*, vol. 74, pp. 34 – 50, 2017.

[12] F. Biral, M. Da Lio, R. Lot, and R. Sartori, "An intelligent curve warning system for powered two wheel vehicles," *European Transport Research Review*, vol. 2, no. 3, pp. 147–156, 2010.

[13] T. Novack, Z. Wang, and A. Zipf, "A system for generating customized pleasant pedestrian routes based on openstreetmap data," *Sensors*, vol. 18, p. 3794, 11 2018.

[14] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, (New York, NY, USA), p. 513–516, Association for Computing Machinery, 2011.

[15] OpenStreetMap contributors, "Openstreetmap," 2020. [Online]. Available: https://www.openstreetmap.org, Accessed on: Feb 07, 2020.

[16] Calimoto GmbH, "Calimoto trip planner," 2020. [Online]. Available: https://calimoto.com/en/motorcycle-trip-planner, Accessed on: Apr 15, 2020.

[17] TTI GmbH - TGU Kurviger, "Kurviger," 2020. [Online]. Available: https://kurviger.de, Accessed on: Apr 30, 2020.

[18] GraphHopper, "Graphhopper routing engine," 2020. [Online]. Available: https://github.com/graphhopper/graphhopper, Accessed on: Feb 05, 2020.

[19] Rever, "Rever moto, inc.," 2020. [Online]. Available: https://rever.co, Accessed on: Apr 15, 2020.

[20] ScenicApp, "Scenic," 2020. [Online]. Available: https://kurviger.de, Accessed on: Apr 30, 2020.

[21] R. BELLMAN, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[22] W. Zeng and R. L. Church, "Finding shortest paths on real road networks: the case for A*," Apr. 2009.

[23] H. Mahmoud and N. Akkari, "Shortest path calculation: A comparative study for location-based recommender system," in *2016 World Symposium on Computer Applications Research (WSCAR)*, pp. 1–5, March 2016.

[24] M. Srivastav, "Circumcircle and incircle of a triangle with its impact in development of skill," *International Journal of Mathematical Archive ISSN 2229 – 5046*, vol. Vol. - 6, No. - 6 (2015): June-2015:, pp. 69–75, 06 2015.

[25] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction hierarchies: Faster and simpler hierarchical routing in road networks," in *Experimental Algorithms* (C. C. McGeoch, ed.), (Berlin, Heidelberg), pp. 319–333, Springer Berlin Heidelberg, 2008.

[26] M. Hadjieleftheriou, Y. Manolopoulos, Y. Theodoridis, and V. J. Tsotras, "R-trees: A dynamic index structure for spatial searching," in *Encyclopedia of GIS* (S. Shekhar, H. Xiong, and X. Zhou, eds.), pp. 1805–1817, Cham: Springer International Publishing, 2017.

[27] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, 11 2015.

[28] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[29] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018.

[30] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[31] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2014.

[32] Y. Ollivier, "Riemannian metrics for neural networks," *CoRR*, vol. abs/1303.0818, 2013.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[35] S. Kotsiantis, I. Zaharakis, and P. Pintelas, "Machine learning: A review of classification and combining techniques," *Artificial Intelligence Review*, vol. 26, pp. 159–190, 11 2006.

[36] K. Krippendorff, "Computing krippendorff's alpha-reliability," 2011. [Online]. Available: https://repository.upenn.edu/asc_papers/43, Accessed on: Jun 17, 2020.

[37] W. Foddy, *Constructing Questions for Interviews and Questionnaires: Theory and Practice in Social Research*. Cambridge University Press, 1993.

[38] K. L. Manfreda, Z. Batagelj, and V. Vehovar, "Design of Web Survey Questionnaires: Three Basic Experiments," *Journal of Computer-Mediated Communication*, vol. 7, 04 2002. JCMC731.

[39] J.-T. Wong, Y. Chung, and S.-H. Huang, "Determinants behind young motorcyclists' risky riding behavior," *Accident; analysis and prevention*, vol. 42, pp. 275–81, 01 2010.

[40] W. C. Schmidt, "World-wide web survey research: Benefits, potential problems, and solutions," *Behavior Research Methods, Instruments, & Computers*, vol. 29, pp. 274–279, Jun 1997.

[41] J. M. Johnson and T. M. Khoshgoftaar, "Survey on deep learning with class imbalance," *Journal of Big Data*, vol. 6, no. 1, p. 27, 2019.

[42] R. A. Bauder and T. M. Khoshgoftaar, "The effects of varying class distribution on learner behavior for medicare fraud detection with imbalanced big data," *Health Information Science and Systems*, vol. 6, no. 1, p. 9, 2018.

[43] Y. You, Z. Zhang, C. Hsieh, and J. Demmel, "100-epoch imagenet training with alexnet in 24 minutes," *CoRR*, vol. abs/1709.05011, 2017.

[44] B. Zhou, A. Khosla, À. Lapedriza, A. Torralba, and A. Oliva, "Places: An image database for deep scene understanding," *CoRR*, vol. abs/1610.02055, 2016.

[45] M. Hussain, J. Bird, and D. Faria, "A study on cnn transfer learning for image classification," in *Annual UK Workshop on Computational Intelligence*, 06 2018.

[46] S. Uchida, S. Ide, B. Iwana, and A. Zhu, "A further step to perfect accuracy by training cnn with larger data," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 405–410, 10 2016.

[47] F. Neto, R. Torkar, R. Feldt, L. Gren, C. Furia, and Z. Huang, "Evolution of statistical analysis in empirical software engineering research: Current state and steps forward," *Journal of Systems and Software*, vol. 156, 07 2019.

[48] Facebook Inc., "React js," 2020. [Online]. Available: https://reactjs.org, Accessed on: Apr 20, 2020.

[49] Semantic Org, "Semantic ui," 2020. [Online]. Available: https://react.semantic-ui.com, Accessed on: Apr 30, 2020.

[50] Google, "Firebase," 2020. [Online]. Available: https://firebase.google.com, Accessed on: Apr 20, 2020.

[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8024–8035, Curran Associates, Inc., 2019.

[52] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, 2019.

[53] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *CoRR*, vol. abs/1811.12808, 2018.

[54] OpenStreetMap, "osm2pgsql," 2020. [Online]. Available: https://github.com/openstreetmap/osm2pgsql, Accessed on: Jun 17, 2020.

[55] GraphHopper, "Edge-based contraction hierarchies," 2020. [Online]. Available: https://github.com/graphhopper/graphhopper/blob/master/docs/core/edge-based-ch.md, Accessed on: Feb 26, 2020.

[56] GraphHopper, "Now flexible routing is at least 15 times faster," 2017. [Online]. Available: https://www.graphhopper.com/blog/2017/08/14/flexible-routing-15-times-faster/, Accessed on: Feb 26, 2020.

[57] D. D. Clarke, P. Ward, C. Bartle, and W. Truman, "In-depth study of motorcycle accidents," *Road Safety Research Rep*, vol. 54, 2004.

[58] V. Huth, F. Biral, Óscar Martín, and R. Lot, "Comparison of two warning concepts of an intelligent curve warning system for motorcyclists in a simulator study," *Accident Analysis Prevention*, vol. 44, no. 1, pp. 118 – 125, 2012. Safety and Mobility of Vulnerable Road Usears: Pedestrians, Bicyclists, and Motorcyclists.