CHALMERS
UNIVERSITY OF TECHNOLOGY

Algorithms for Robust Path-Planning

Master's thesis in Engineering Mathematics and Computational Science

Thomas Jonsson Damgaard
Jens Åkerlund

Department of Space, Earth and Environment
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis

Algorithms for Robust Path-Planning

Thomas Jonsson Damgaard
Jens Åkerlund

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Space, Earth and Environment
Chalmers University of Technology
Gothenburg, Sweden 2020

Algorithms for Robust Path-Planning
Thomas Jonsson Damgaard
Jens Åkerlund

Supervisors:
Claes Andersson, Department of Space, Earth and Environment, Chalmers
Anders Averö, Saab AB
Patrik Ellrén, Carmenta AB
Mikael Rittri, Carmenta AB
Håkan Warston, Saab AB

Examiner:
Claes Andersson, Department of Space, Earth and Environment, Chalmers

Master's Thesis
Department of Space, Earth and Environment
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Road network of Gothenburg, Sweden, visualized using Carmenta Engine. Road data provided by © OpenStreetMap contributors [1]. Raster data and imagery provided by © Lantmäteriet [2].

Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Algorithms for Robust Path-Planning
Thomas Jonsson Damgaard
Jens Åkerlund
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

Route optimization is a commonly studied field of optimization resulting in path-planning algorithms. In this project, alternative route generation and robustness analysis were conducted for common road networks and off-road terrains. This was done using Open Street Map data and high-resolution terrain data provided by Vricon and Lantmäteriet. Alternative routes were generated using a constructed 2-way search algorithm. Robustness analysis was split into a physical robustness-index and a non-physical robustness simulation. The generated routes conform to the constructed robustness-index, which is sensitive to weather conditions when traversing terrain. The simulation is applied to the generated routes to visualize additional penalty times where the user can identify critical points along said route. This is done using a robustness simulation which creates obstacles along the chosen path and calculates the additional time needed. The final result is a complete Java program connected to Carmenta Engine, a map engine provided by Carmenta. The resulting program showed conceptually promising routes, providing reasonable alternatives while accounting for current weather conditions.

Keywords: route optimization, alternative routes, robustness algorithms, off-road trafficability, high resolution raster data, GIS.

Acknowledgements

We would like to first thank Anders Averö, Patrik Ellrén, Mikael Rittri, and Håkan Warston for helping us out throughout the entire project with lots of valuable discussions, feedback and ideas that we could apply to our project. We would also like to thank Claes Andersson for being our supervisor from Chalmers, who guided us on the right track along the project.

We would also like to thank Vricon for providing high resolution raster data.

Finally, we would like to thank Saab AB and Carmenta AB for giving us this opportunity and all the encouragement from all the coworkers!

We would also like to thank family and friends for that constant support throughout the years of studying at Chalmers!

Thomas Jonsson Damgaard
Jens Åkerlund
Gothenburg, Sweden, June 2020

Contents

List of Figures

List of Tables

1

Introduction

Route optimization is a broad and intensively studied field of optimization where the objective is to find the best path on a given graph concerning many different aspects, most commonly used is travel-time minimization. Saab and Carmenta are two companies located in Gothenburg, Sweden, who both partly work with path and mission planning products that need to deliver vehicle paths that meet certain criteria, notably for military purposes [6, 7]. Less common criteria used might be taking into account exposure to enemy surveillance or avoiding mine-fields/threats. With the recent introduction of greatly improved geographical-data, with post spacing resolutions of 0.5 m [8, 9], one might ask if this additional and very detailed information could be used to further improve path-finding algorithms used today. One aspect could be to further study the robustness of a generated route.

Robustness, in this thesis, will be used as a term to describe both whether or not a given path is traversable in practice and how stable it is to obstacles appearing along the path. A path could be given a robustness index that would indicate how likely the path would yield a successful relocation concerning different dynamical aspects and could also be given a histogram over the potential additional penalty times visualized on the map to identify critical parts. Taking into account for example weather conditions or the sturdiness of traversed roads will hopefully lead to a better understanding of how one can better generate safe paths.

1.1 Background

This thesis project is a continuation based on three previously completed master's theses and one completed bachelor's thesis at Saab, where high-resolution data from Vricon has been analyzed with the purpose of improving route planning and mission planning in a military context. Vricon is a company specializing in stereophotogrammetry to create high-resolution geospatial data [10]. Vricon is partly owned by Saab and offers solutions in intelligence analysis, military operations, telecom, emergency preparedness, and precision 3D registration [11].

The first project was completed by Isaksen and Brusevold [12] at Ostfold University College, in collaboration with Saab Surveillance in Halden. They were tasked with the mission to study line-of-sight algorithms to better understand where radar units or missile units should be placed to achieve maximum working range.

The second project was conducted by Nilsson [13] at Linköping University, in collaboration with Saab Dynamics. The goal was to reconstruct the heights of buildings within the Vricon data set. Here the high-resolution data provided by Vricon was used for a different purpose than mission/route planning. However, progress was still made to better understand what usefulness high-resolution data could provide.

The third master's thesis, with the closest resemblance to this thesis, was conducted by Larsson [14] at Chalmers University of Technology together with Saab Surveillance in Gothenburg. The project was based on route planning for the radar system ARTHUR, a defense unit that tracks enemy missile trajectories in the air to determine the location of impact and the location of origin, meaning where the hostile missile was launched. Travel time between deployment sites is a very important aspect when planning missions for the ARTHUR system. The system needs to be redeployed frequently to minimize the risk of being engaged when detected by hostile forces. Additionally, the horizon, as viewed from the deployment site was a critical aspect, when minimizing the reaction time when trying to detect hostile fires. The short reaction time is vital when trying to maximize the time to warn for impacting fire. Larsson used stochastic optimization, a Genetic Algorithm (GA) to be precise, to optimize the route planning.

The fourth and final master's thesis was conducted by Johansson and Karlsson [15] at Chalmers University of Technology together with Saab Surveillance in Gothenburg. Their task was to provide an algorithm for finding the best grouping spot for a complete anti-air defense company, with many different units, particularly the sensor unit UndE23 and fire units EldE97 and EldE98. The algorithm needs to be optimized such that all the units can communicate and work at high efficiency. The same GA, provided by Larsson [14], was used for this problem.

1.2 Purpose

The first goal of this project was to analyze vehicle routes in 3D terrain and develop a robustness function that provides each route with a robustness index. The robustness index would be used to describe how likely a relocation mission will succeed in practice. Additionally, a robustness simulation was to be constructed that studies a given route for weaknesses measured in potential penalty times if problems occur along the way. Given a robustness function and a robustness simulation, the second goal of this project was to generate alternative routes that take into account the robustness index and the best available route can be determined by mission planners. Hopefully, the robustness index and simulation could potentially lead to increased knowledge in path-planning where one takes other factors, other than travel-time, into account. The final result was to be visualized in Carmenta Engine, a map engine provided by Carmenta. The robustness function and routing ought to be implemented in Java as a stand-alone product.

1.3 Limitations

The project is centered around the robustness function which is determined based on many different factors. One clear limitation is the number of factors that the authors could take into account for the implemented program. A good amount of factors were studied with theoretical models with reasonable precision some of which would later be implemented into the final product. However, some did not make it into the final product and only stays as theoretical models, some of which are implemented into separate Java classes. The accuracy in these models is very low and should only be seen as conceptual factors that a path-finding program could take into account if high precision data is provided.

Due to the amount of time spent studying potential factors, the total amount of time spent implementing it into the final product was limited. Thereby, some parts were never implemented and the overall product lacks some finishing touches, mostly to its GUI and functionality. For the probject that is not a big concern as a whole since the implementation should only be seen as a proof of concept. Limiting implementation time led to minimal performance improvements.

The final product could not be tested in practice, which limits the amount of statistical analysis that could be performed on the final product. Instead, the results would only be judged as either reasonable or not reasonable by the authors and supervisors.

1.4 Problem Formulation

This project can be split into two goals. Firstly, define algorithms and methods that describe how robust a chosen route is with respect to dynamical events such as unforeseen obstacles and changes in weather conditions. Secondly, define methods and algorithms for generating alternative routes.

2

Theory

In this chapter, all theories used in this project will be presented. Most of the theory surrounds optimization problems, computer science algorithms, and physics. Most of the theory presented here will be further developed and put into context in chapter 3.

2.1 Success Probability Modelling

For a complete path of several sub-segments $i \in N$ the probability for successful traversal of the entire path can be computed as a product of all sub-segment probabilities $p_i \in [0, 1]$

$$\text{success} = 1 - \prod_{i \in N} (1 - p_i). \tag{2.1}$$

2.1.1 Normal Distribution function and CDF

The following is true for one continuous variable X but could be generalized to multiple continuous stochastic variables $\vec{X} = (X_1, X_2, ..., X_n)^T$. If a randomized variable X is defined to be normal distributed, it can be written as $X \sim N(\mu, \sigma^2)$. Here μ is the expected value of X [16] and σ is the variance [17]. Its probability function (density function) is given as [18],

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{2.2}$$

It has a cumulative distribution function (CDF), which is used to calculate the probability of the stochastic variable X to be within a certain (continuous) interval. It can be expressed as,

$$P(X \leq x) = \Phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx. \tag{2.3}$$

2.2 Multi-Objective Optimization

A multi-objective optimization problem is an optimization problem that is centered around multiple objective functions [19]. The variable of interest could be bounded

strictly using polyhedrons, while the objective functions that are assumed to be minimized/maximized would consist of at least two or more functions. A typical mathematical description for these types of problems would be [19],

$$\begin{array}{cc} \text{minimize} & (f(x)_1, f(x)_2, ..., f(x)_k) \\ \text{s.t} & x \in X, \end{array} \tag{2.4}$$

where x is a set of variables bounded by a polyhedron space X and $k \geq 2$. Using a scalarizing approach, for some unique cases, one could translate a multi-objective optimization problem into a single optimization problem using a set of weighted constants w_i [19],

$$\begin{array}{cc} \text{minimize} & \sum_i^k w_i f(x)_i \\ \text{s.t} & x \in X. \end{array} \tag{2.5}$$

The scalarizing method is a priori method which means that by approximating the original problem as a single optimization problem, one could yield a set of Pareto points. The Pareto points are used to describe potential optimal points in the original problem [20]. The best qualified Pareto points are associated with the Pareto optimal points (Pareto efficient points).

2.2.1 Pareto Optimality

Given a set of Pareto points $\mathcal{P}$, one could describe Pareto optimality [20] as a subset of $\mathcal{P}$ that contains the dominating Pareto points. First, let's define a dominating Pareto point. Given two Pareto points $p_1, p_2 \in \mathcal{P}$ let's assume (WLOG) that their i-th values are strictly different while all the their other values are equal. So, $p_1 = (f_{11}, f_{12}, ..., f_{1i}, ..., f_{1n})$ and $p_2 = (f_{21}, f_{22}, ..., f_{2i}, ..., f_{2n})$. That means that $f_{1k} = f_{2k}$, $\forall k \neq i$ and $f_{1i} \neq f_{2i}$. If the Pareto points are associated to a minimization problem, the lesser of the two values will be the host of the dominating Pareto point, let's assume that $f_{1i} < f_{2i}$. This means that p_1 is dominating p_2 and is qualified to be among the final set of Pareto optimal points $\tilde{\mathcal{P}} \subseteq \mathcal{P}$. Although, if $f_{1j} > f_{2j}$ for some value $j \neq i$ (assumption one is no longer true), then point p_1 does no longer dominate point p_2 since they have one value each that is different and considered "better" than the others.

2.3 Route Optimization Algorithms

Route optimization algorithms have been studied extensively since the beginning of computer science. Route optimization typically involves some nodes connected by edges. The goal is to find the best route from a start node to a destination node. The best route is the route with minimum/maximum cost, depending on the exact problem. The cost of a route can be modeled as the sum of traversing all edges or nodes of the route, but more complex models could be applied.

2.3.1 Options for Describing Traversable Environments

There are several ways of describing traversable environments. Some options are the following:

- Pre-vectorized road networks,
- Classification and DSM rasters,

Pre-vectorized networks reduce the search space by containing vectors of existing edges. These vectors can include a lot of information, such as speed limits, ground type, and bearing capacity, or maximum vehicle height. They could even include time-dependent information such as current traffic and delays.

Classification and DSM rasters can be used to construct a searchable space. A raster consists of regularly spaced values. One example is a digital surface model (DSM) which describes the elevation of the terrain. This can then be used to detect obstacles or steep hills that cannot be traversed. One extra piece of information is classification rasters which contain information on the ground type, such as open, low vegetation, high vegetation, building, etc.

2.3.2 Dijkstra's Algorithm

Dijkstra's algorithm, created by Edsger W. Dijkstra, finds the shortest path between two nodes in a graph. The graph is required to have non-negative weight/distance edges between all vertices [21].

```
add start node with distance 0 to min-priority queue q
construct empty set visited
construct empty map distance to store the
distance between start and all nodes in N
while q not empty do
    n <- q.pop()
    if n in visited then
        continue
    end if
    add n to visited
    for each node p connected to n do
        d <- w+dist(n, p)
        if d < distance(p) then
            add p to q with priority d
        end if
    end for
end while
```

The worst-case runtime depends on the queue implementation used. For an ordinary binary heap, the worst-case runtime: $O((|E| + |V|) \log(|V|))$. This method also uses $O(|V|)$ auxiliary space. The algorithmic runtime can be improved to

$O(|E| + |V| \log(|V|))$ However, this doesn't always result in higher performance in reality [22].

2.4 Monte Carlo Simulation

Monte Carlo simulation is built upon the principle of combining randomness in input samples to achieve (hopefully) deterministic results based on probability distributions. It is considered a stochastic process (random process). A deterministic result is a result where the same input will return a consistent output, regardless of the randomness built-in, a deterministic conclusion can be drawn from the input all the time. The general way of performing a Monte Carlo method can, in short, be described by four steps [23],

1. Define a domain of possible inputs.
2. Generate inputs randomly from a probability distribution over the domain.
3. Perform a deterministic computation on the inputs.
4. Aggregate the result.

The Monte Carlo simulation is best used on physics-related or pure mathematical problems where no other clear deterministic approach is available.

2.5 Collision detection

To detect certain key events collision detection models and algorithms are sometimes used as key observatories. In the following project, two key collision detection models were used, a circle and line model and a polygon and line model. The polygon and line model is exactly just a modified version of the line and line detection model with the assumption of segmented lines instead of infinite long lines.

2.5.1 Line-Circle

Assume that there exists a circle centered in the origin with a radius r that can be expressed as $x^2 + y^2 = r^2$ and that there exists a line expressed by $ax + by = c$. Then there exist two conditions [24] where the line would intersect the circle, either tangentially or as a secant line. The following conditions are,

$$r^2(a^2 + b^2) - c^2 > 0 \tag{2.6}$$

$$r^2(a^2 + b^2) - c^2 = 0, \tag{2.7}$$

where (2.6) is the secant line condition and (2.7) is the tangent line condition.

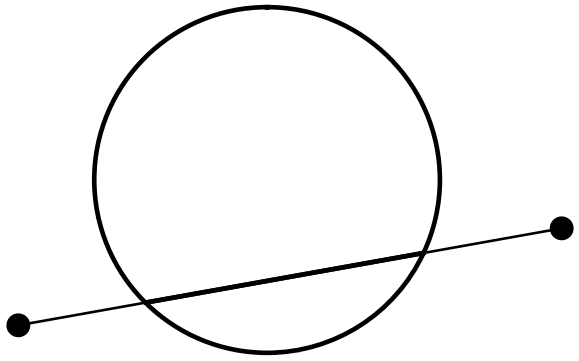

Fig. 2.1: Illustration of collision between a line segment and circle.

2.5.2 Line-Polygon

There is no direct line-polygon intersection matrix, but one could re-state the problem differently. A line-polygon intersection problem could instead be seen as a n-times line-line intersection for the segmented case (not infinitely long lines) where the polygon would consist of n many segmented lines. Given two segmented lines L_1 and L_2 defined as,

$$L_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + t \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix}, \; L_2 = \begin{pmatrix} x_3 \\ y_3 \end{pmatrix} + u \begin{pmatrix} x_4 - x_3 \\ y_4 - y_3 \end{pmatrix}$$

where $u, t \in [0, 1]$, L_1 is defined by the points $(x_1, y_1)^T$ and $(x_2, y_2)^T$ and L_2 is defined by the points $(x_3, y_3)^T$ and $(x_4, y_4)^T$. We can now compute the intersection as [25],

$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} x_1 + t(x_2 - x_1) \\ y_1 + t(y_2 - y_1) \end{pmatrix} \text{ or } \begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} x_3 + u(x_4 - x_3) \\ y_3 + u(y_4 - y_3) \end{pmatrix} \tag{2.8}$$

where,

$$t = \frac{(x_1 - x_3)(y_3 - y_4) - (y_1 - y_3)(x_3 - x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{2.9}$$

and,

$$u = -\frac{(x_1 - x_2)(y_1 - y_3) - (y_1 - y_2)(x_1 - x_3)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}. \tag{2.10}$$

After a line-line intersection has been performed for all lines one can obtain all of the intersection points between the line and the polygon. Additionally, one could calculate the distance the line is inside the polygon given two intersection points.

2.5.3 Normalized Worst-Case

Each simulation indicates what kind of additional penalty time (if possible) is needed to perform the re-routing process from the obstacle point. To better visualize the problematic parts among a given set of routes, a coloring scheme was applied. The colors were normalized with respect to the worst-case scenario, for the complete set

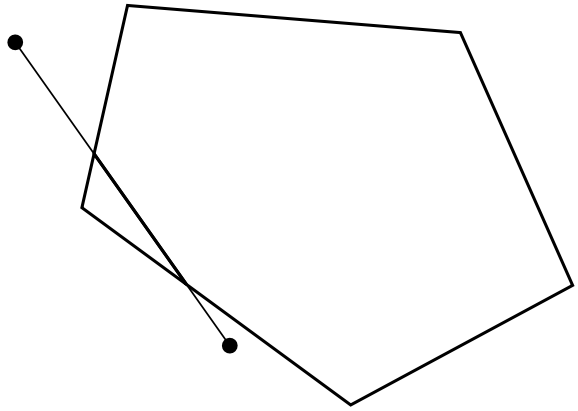

Fig. 2.2: Illustration of collision between a line segment and polygon.

of all routes. Given a set of penalty time t_{ij} for path $i \in \mathcal{I}$ at sample point $j \in \mathcal{J}$, the following normalizes each point,

$$\bar{t_{ij}} = \frac{t_{ij}}{\max_{i \in \mathcal{I}, j \in \mathcal{J}}(t_{ij})} \tag{2.11}$$

2.6 Classical Systems and Energy

Classical systems are defined as systems influenced by the physical laws described by Isaac Newton [26]. They can be expressed in different reference systems but are not influenced by relativity. Given an inertial reference system, the coordinates expressing the reference system are said to be fixed and do not change like the case for a non-inertial reference system. A typical coordinate system is the Cartesian coordinate system [27]. An important measured physical property, typically expressed in all types of physics, is energy [28]. The conservative law of energy states that the total energy E in a fixed system cannot change over time,

$$\Delta E = 0. \tag{2.12}$$

This law sometimes plays a central roll in expressing how systems behave after an event. It is only true for certain cases, one particular case is where there is no work done by the system. Work is defined as $W = f\Delta l$ where f is an applied force and Δl is a displacement.

2.6.1 Kinetic Energy

When selecting the most optimal path when traversing off-road environments the elevation data could be used as a key aspect when generating such an optimum. If one minimizes the energy, using kinetic energy and potential energy that is sensitive to vertical height changes, the most optimal way for a vehicle to move could be very similar to how a particle would behave.

In a classical system for a non-rotating object moving at velocity $\vec{v}$ its kinetic energy can be expressed as [29],

$$E_k = \frac{1}{2}mv^2, \tag{2.13}$$

where m is the mass of the object. Given a potential energy term $V(\vec{r})$ that depends on the placement $\vec{r}$, one could study the change in kinetic energy E_k. The total (conserved) energy in the closed system can be expressed as,

$$E = E_k + V(\vec{r}). \tag{2.14}$$

3

Methods

This section describes various sub-robustness models that were implemented. It also describes how these models were combined to give a complete robustness index to a given route.

3.1 Theoretical Methods

In this chapter, all of the theoretical models studied and used for this project will be presented. The models cover everything from Newtonian-mechanics, probability theory, terramechanics, and much more. They are all used partly to describe the robustness of a selected road.

The robustness of an entire path can be modeled in several ways:

- The maximum risk of all segments of the path. For example, the maximum slope could be the deciding factor for mission success. If the segment with the greatest slope is feasible, all other segments should work as well.
- The multiplicative success probability. If every segment i of the path has success rate p_i, the total success rate will be a product of all segment probabilities.
- Additive risk.

3.1.1 Definition of Robustness

Robustness is a word that could mean many different things. For this project, the word robustness is used to describe both whether a chosen path is traversable or not and how stable the chosen path is with respect to small obstacles appearing along the way.

The physical factors are said to be the factors that answer more or less whether a path is traversable or not. They can be seen as theoretical maximums of physical properties such as the weight bearing capacity of a path or whether a vehicle can climb a certain angle of steepness. They will exist as layers of information that build up the region of interest, the map. However, these physical factors are not static and are sensitive to changes in temperature and weather conditions.

The non-physical factors are said to be related to what options exist along a chosen path. It expresses how stable the path is with respect to small obstacles appearing

and potentially blocking optional routes ahead. If one or more sub-paths of the complete path were to be closed/removed, is it possible to re-route from those closed off sub-paths and if so, what is the penalty? The penalty will be measured in additional time to reach the final destination, assuming such a path exists.

The final robustness will take into account both of these different factors, physical and non-physical. The individual factors that determine the complete robustness will also be weighted differently to better suit different types of scenarios.

3.1.1.1 Robustness-Index

To better understand the physical robustness, a robustness-index was constructed. The robustness-index is defined as a continuous value between zero and one, $I \in [0, 1]$. It is a value determined by the robustness of each sub-factor, $\mathcal{R} := \{\text{factor } 1, \text{factor } 2, ...\}$. Their individual robustness is defined as $I_i \in [0, 1]\ \forall i \in \mathcal{R}$. Multiple attempts have been performed to define the relationship between the individual factors and the final robustness index.

The first solution was to define it as the product of each individual robustness factors,

$$I = \prod_{i \in \mathcal{R}} I_i, \tag{3.1}$$

Since the individual factors could be of different importance, another attempt was made to instead apply a set of weighted constants $w_i \in \mathbb{R} \setminus \{\pm\infty\}$ to determine the weighted product of all factors,

$$I = \left[\prod_{i \in \mathcal{R}} I_i^{w_i} \right]^{\frac{1}{W}}, \tag{3.2}$$

$$W = \sum_{i \in \mathcal{R}} w_i. \tag{3.3}$$

Finally, the last attempt made was simply to pick the worst individual robustness factor to describe the final robustness factor,

$$I = \min_{i \in \mathcal{R}}(I_i).$$

Each of the factors can be seen as a "physical" risk, either one is close to the theoretical limit of that factor or have reached a total amount of exposure. For the case of $I \approx 1$ the route is deemed to be very robust. For the opposite case, $I \approx 0$, the route is deemed to not be robust and is said to be non-traversable.

3.1.2 Terramechanics - Understanding Off-road Trafficability

Terramechanics is the study of soil and its properties, where some of the main properties of interest in a soil type are internal friction angle ϕ, cohesion force c, and density γ [30]. Terramechanics is important to understand for off-road trafficability. Most notable work within this field came from Mohr-Coulomb, Bekker, and Terzaghi. For this report, the exact behavior of each type of soil is not of importance, rather the possibility of accurately expressing one of them is sufficient since one could easily expand this area of robustness if one would like to know the exact behavior of the soil of interest. A soil type's structure is well defined by the USDA Soil Texture Triangle (see Fig.3.1). The theoretical models used in this report for off-road-robustness were a mix of accurately describing mathematical expressions and tables used to better bound the problem, limiting the time spent studying different soils.

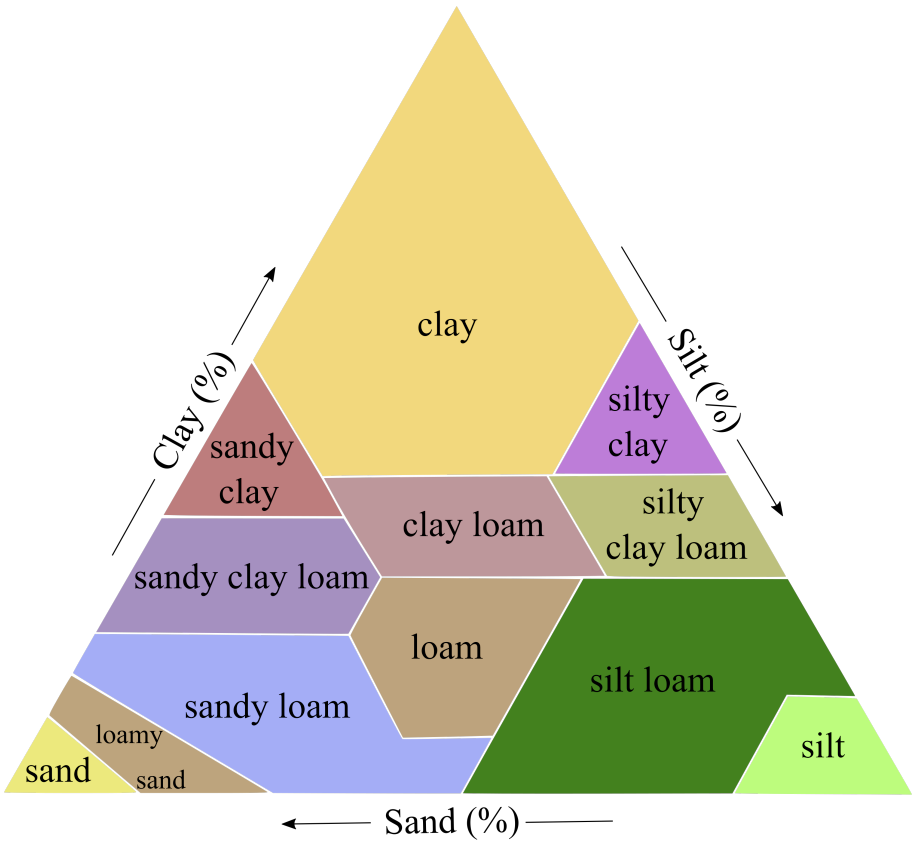

Fig. 3.1: Illustration of the soil-texture diagram as seen in a report by Groenendyk et al. [3]. The soil-texture diagram is more commonly known as the USDA Soil Texture Triangle. It defines a soil with respect to what percentage of sand, silt, and gravel that it contains. This triangle-diagram is widely used in terramechanics to classify the different types of soil-textures.

3.1.2.1 Mohr-Coulomb failure criterion

The sheer strength failure criterion for soils was first expressed by Mohr-Coulomb [31] and can be expressed as,

$$\tau = c + \sigma \tan \phi, \tag{3.4}$$

where τ is the soil shear strength, c is the soil cohesion force, σ is the applied pressure force on the soil and ϕ is the internal friction angle. This relation is studied very closely to determine when a soil type reaches its maximum potential before failure (e.g used for bearing capacity).

3.1.2.2 Bearing Capacity of Different Soils for Vehicles

The bearing capacity model determines the maximum amount of pressure that can be applied to a soil by a foundation without failure/sinkage. The formula used for this project is based on Terzaghi's formula [32], which states that for continuous foundations,

$$q_f = cN_c + \sigma N_q + 0.5B\gamma N_\gamma, \tag{3.5}$$

where q_f is the maximum bearing capacity before failure, c is the cohesion force of the soil, σ is the applied force that can be expressed as $\sigma = D\gamma$ where D is the depth of the foundation, γ is the density of the soil, B is with width of the foundation and N_c, N_q and N_γ are all constants given by Terzhagi. The constants are calculated using,

$$N_q = \frac{e^{2\pi(0.75-\phi/360)\tan(\phi)}}{2\cos^2(45+\phi/2)}$$

$$N_c = 5.14,\ \phi = 0$$

$$N_c = \frac{N_q - 1}{\tan\phi},\ \phi > 0$$

$$N_\gamma = \frac{2(N_q+1)\tan\phi}{1+0.4\sin 4\phi}.$$

Note that the internal friction angle ϕ is measured in degree not in radian and the last term N_γ is actually an approximated version created by Coduto [32]. Although, this formula depends on a depth D that describes the depth of the foundation. For a vehicle with a tire, there is no foundation going into the ground (unless one allows sinkage).

Simon Y. Oloo et al. published a paper "Bearing capacity of unpaved roads" (1997) [33] where they modified Terzaghi's failure criterion to better suit a vehicles bearing capacity. The foundation depth was set to zero, yet the contact area of the vehicle still implied some sort of depth. The new model is then,

$$q_f = cN_c + 0.5B\gamma N_\gamma, \tag{3.6}$$

where one term has been removed due to depth being equal to zero, $\sigma = D\gamma \rightarrow \{D = 0\} \rightarrow \sigma = 0$. Other than that the model is identical to Terzaghi's model.

3.1.2.3 Soil Cohesion Force and Internal Friction Angle

The cohesion force c and the internal friction angle ϕ in the soil are two important properties that define a soil type and are unique for each type of soil and are assumed in this report to be dependent on moisture content w and temperature T. The cohesion force is mostly present in clay-mixed soils and describes the shear strength of a rock or soil that is independent of interparticle friction [34]. The internal friction angle describes the friction shear resistance in a soil [35]. For this report, the water content (moisture content) in a soil type will be the dominant dynamical factor and changes both the internal friction angle and the cohesion force. The temperature factor will be neglected for temperatures above 0°C and this assumption is based on a report written by Anh-Minh Tang et al. [36] where only minimal changes in cohesion force and friction angle was observed for temperatures around $5 - 40$°C. Though, for temperatures below 0° the soil enters a frozen state where temperature now becomes a dominant factor. This case is not neglected in this report.

3.1.2.4 Soil Properties with Respect to Moisture Content

Moisture content is one key attribute that drastically changes properties of soils. It mainly decreases the shear strength of the soil. Moisture content is thereby a very important aspect to study and is a key dynamic factor for this project. Yifan Dong et al. investigated the change in shear-strength soil properties with respect to change in water content in the black soil region of Northeastern China [37] for some type of soils (mostly silty-clay-loam Fig. 3.1). The result indicated that it was possible to create a function of cohesion force or internal friction angle with respect to water moisture levels w (pressured in %) with some precision. The final expressions found were,

$$\begin{aligned} \phi(w) &= 34.35e^{-w/8.69} + 27.52 \ (R^2 = 0.87) \\ c(w) &= \begin{cases} 18.47w - 20.85 \ (R^2 = 0.98) & w < 12 \\ -257.92\ln(w) + 908.95 \ (R^2 = 0.98) & w \geq 12. \end{cases} \end{aligned} \tag{3.7}$$

Note that the precision is worse for the friction angle $\phi(w)$ compared to the cohesion force $c(w)$.

Two other soil studies were analyzed to compare with the one mentioned above. Red soil [38] located closer by in the southern part of China and researched by Yue Zhang et al. Gypseous soil [39] located further away in the middle east researched by Moataz A. Al-Obaydi. The red soil samples had very similar behavior to the black soil samples at higher moisture content levels. However, the red soil had an overall stronger shear strength with a larger window of allowed soil moisture content before reaching failure with said vehicle used in measurements. The Gypseous soil also had (though based on fewer sample points) a similar shear strength behavior where the peak performance was reached in the region of 10-15% moisture content. The significant difference between these soils is that the Gypseous soil had an overall worse bearing capacity. It had a much shorter interval of allowed soil moisture content w measured for said vehicle. Thereby, soil moisture content could be of higher importance when traversing regions where soils have similar behaviour to the Gypseous soil where the window of allowed soil moisture content is shorter.

3.1.2.5 Frozen Soil Properties

How different soils behave when entering a frozen/solid state is not an easy task to explain, although, The authors behind the book "Frozen Ground Engineering" [40] does an excellent job defining the most importance changes that occur. In general, the soil cohesion force c increases whenever the soil enters a phase transition to solid-state. A report by Xiangtian Xu et al. [41] determined that the increase in bearing capacity is not only due to the phase transition itself, the increase is also due to the increase in compactness of the soil. They studied a silty-clay soil with different level of compactness before freezing it down where they found that the cohesion force increased much more for the lesser compact soil samples when entering the frozen state, meaning that whenever the soil enters the phase transition it's not just the ice that increases the cohesion force, it's the change in structure that increases the overall cohesion force as well. The increase in bearing capacity when entering a frozen state could be measured to be up to a factor of 10-100 times higher depending on the type of soil and its original compactness.

The author of the aforementioned book presents a formula for determining the cohesion force $c(t, \theta)$ which depends on the time-to-failure time (creep-rate) t and a temperature $\theta = -T$ (Kelvin). It is said that the total strain on the frozen soil is given by $\epsilon = \epsilon_0 + \epsilon_c$ where ϵ_0 is an initial strain on the soil at time 0 and ϵ_c is a delayed strain, known as the creep strain, that occures at the creep-time c. The strain level is related to the uniaxial stress strength σ that is applied to the soil. This term is present in the Mohr-Coulomb failure stress line, $\tau = c(t, \theta) + \sigma \tan \phi(t, \theta)$ where τ is the shear strength in the soil and $\phi(t, \theta)$ is the internal friction angle in the soil which is also dependent on failure rate t and temperature θ. The complete relationship for the cohesion force can be described according to the author as,

$$c(t, \theta) = \frac{\sigma(t, \theta)}{2\sqrt{N_\phi}}, \tag{3.8}$$

where N_ϕ is a flow value defined by,

$$N_\phi = \frac{1 + \sin \phi(t, \theta)}{1 - \sin \phi(t, \theta)}.$$

The stress $\sigma(t, \theta)$ is related to the strain ϵ mentioned above by,

$$\sigma(t, \theta) = \sigma_{c\theta} \left(\frac{\dot{\epsilon}}{\dot{\epsilon}_c} \right)^{1/n},$$

where $\dot{\epsilon} = d\epsilon/dt$, n is an experimental creep exponent and $\sigma_{c\theta}$ is the temperature-dependent creep modulus corresponding to the creep-rate $\dot{\epsilon}_c$. This formula can be used to calculate how the cohesion coefficient for different soils changes as temperature decreases. Note that as the parameter θ increases, the temperature decreases due to their relation $\theta = -T$. Typical values were provided by the author for some soils.

3.1.2.6 Frozen Lake

During the winter season and given some long-term weather and temperature conditions, crossing a frozen lake with a vehicle is considered a possibility and can be a promising short-cut, if and only if the frozen lake's bearing capacity P_{lake} can withstand the vehicle's wheel-to-ground pressure P_{vehicle}, $P_{\text{lake}} > P_{\text{vehicle}}$. An empirically proven formula that is often used to approximate P_{lake} is Gold's Formula that was first introduced in 1971 by Dr. Lorne Gold [42, 43]. Given a depth h that measures the thickness of the ice in m and a safety constant A ranging from 3.5-7.0 kg/cm^2, Gold's formula says that the maximum weight (kg) of the truck P is given by,

$$P_{\text{lake}} = Ah^2. \tag{3.9}$$

The safety factor A [42, 43] measures the total amount of risk the individual is ready to take while crossing the ice, a lower value of A means that a smaller amount of risk is taken. Estimating the quality of the ice by giving A its smallest possible value, taking the least amount of risk, is an appropriate approach for this report when trying to maintain a good robustness index.

3.1.2.7 Estimating Wheel Sinkage Near Bearing Capacity Failure Based on Mohr-Coulomb Failure

The following is a model purposed by the authors as a possible way of measuring the total wheel sinkage and is used as an experimental formula. Combining Terzaghi's original bearing capacity formula and the modified version created and tested by Simon Y. Oloo et al. (Section 3.1.2.2), one could try and estimate the potential sinkage depth D a foundation would have, if allowed. In general, a wheel cannot be allowed to have a sinkage depth, since it is assumed (just like in Oloo's report) to be situated on the top layer of the surface, nonetheless a simple modification allows one to estimate the potential sinkage depth D,

$$D = \frac{q_f - cN_c - 0.5B\gamma N_\gamma}{\gamma N q}, \tag{3.10}$$

where this is a simple rewritten form of Terzaghi's original bearing capacity formula with the exact same corresponding terms. Comparing Terzaghi's original expression with the modified version, the leftover term is σN_q where $\sigma = D\gamma$ where D is the depth of the foundation. A rough estimation for the potential depth in sinkage for a wheel could possibly be estimated using this formula, but no report of similar content was found about this.

If a vehicle were to travel across an unpaved road of known soil type, the first layer of the surface is penetrated by the wheel due to the wheel to ground pressure being stronger than the bearing capacity of the first layer. This formula was used to make small experiments around the possible sinkage depth.

3.1.2.8 Calculating the Robustness for Unpaved Roads

The robustness was measured in success rate p, which is simply $p = 1 - r$ where r is the risk of failure. For unpaved roads, the bearing capacity is one aspect of robustness and in this reported no additional sinkage was allowed apart from the initial

depth used to estimate the contact area between the wheel and the ground. Thereby the absolute bearing capacity q_f found in Terzaghi's formula must be greater than the wheel to ground pressure p_{vehicle}. The bearing capacity, as stated above, depends on temperature T, water content w, and of course, soil type which makes this value dynamic while the wheel to ground pressure remains relatively static. One way of calculating the success rate is to assume that the value q_f follows a normal distribution $q_f \sim N(\mu, \sigma^2)$ with some expected value μ and variance σ. The expected value is assumed to be the value given by the formula q_f 3.6 and the variance σ will be approximated as small artificial errors in water content w and temperature T, though the estimated error in picking the wrong soil type is neglected. The precision is assumed to be completely accurate. The final distribution becomes $q_f \sim N(q_f(w,T), q_f(\Delta w, \Delta T)^2)$. To calculate the success rate, one would like to see how far away the wheel to ground pressure p_{vehicle} is to the maximum bearing capacity (assuming it's not already above mentioned limit). The final expression becomes,

$$P_{\text{success}} = 2\max(0, P(X \leq q_f) - P(X \leq p_{\text{vehicle}})), \tag{3.11}$$

where $P(X \leq x) = F_X(x) = \int_{-\infty}^{x} f_X(t)dt$ and $f_X(t)$ is the normal distribution function behaving like $X \sim N(q_f(w,T), q_f(\Delta w, \Delta T)^2)$ and X is a stochastic value from said distribution.

3.1.3 Vehicle Model

This chapter describes the models used for determining the vehicle's properties, e.g pressure from wheel to the ground, and maximum slope it can climb. The vehicle model is based on terrain-vehicle Scania SBA 111/111S where radar unit ARTHUR is mounted on.

3.1.3.1 Maximum Slope

For the case of traverse direction the maximum slope θ depends on the friction between the tire and the ground, μ. It can be considered the slipping angle and is not static due to the ground friction not being static. Using standard Newtonian mechanics applied to a vehicle climbing an angle θ we get that,

$$\sum F_x = 0 \rightarrow \mu N \cos\theta = N \sin\theta \rightarrow \theta = \arctan\mu. \tag{3.12}$$

For the case of the sideways direction the maximum slope θ depends on the geometry of the vehicle and its center of mass. This could be considered the tipping angle and is considered static due the geometry of the vehicle not changing. For this model, the center of mass is said to be in the center where the vehicle is approximated using a box with width b and height h. The box is placed at an angle θ from the ground and is shifted an angle ϕ from the horizontal plane. Using geometry one can show that the angle ϕ has the relation $\tan\phi = \frac{h/2}{b/2} = h/b$. Using that $\phi + \theta = \pi/2$ one can find an expression for θ. The final expression is simply $\theta = \arctan(b/h)$, where the maximum angle θ now is determined by the geometry of the vehicle.

3.1.3.2 Wheel to Ground Pressure

Mellgren proposed in 1980 a nominal ground pressure formula for vehicles with wheels or vehicles with tracks [44]. The purpose was to create a rather simple formula that is independent of the wheel pressure itself, the soil properties and that would allow a 15% sinkage of the wheel diameter. The wheel to ground pressure is very much dependent on the contact area A between the wheel and the ground, which is easily estimated using the formula created by Mellgren, even for the case of a tracked vehicle, at the cost of some precision. The final nominal ground pressure is given by,

$$N^k_{\text{NominalGroundPressure}} = \frac{G_k}{rb}, \tag{3.13}$$

where G_k is the wheel load for wheel k on the vehicle, r is the radius of the wheel and b is the width of the vehicle.

Another method of calculating wheel to ground pressure was used in this report as an alternative. Simon Y. Oloo et al. used in their report a modified version of the Portland Cement Association (PCA 1984) contact area equation where they assumed the contact area between the wheel and the ground to be a rectangle where the width and the height were determined using the wheel load and the wheel pressure [33]. Compared to the Mellgren equation, this one creates the contact area A using the wheel load P and the wheel pressure P_t where the wheel pressure now becomes a factor with additional information. For a vehicle axle with 2 wheels attached at each side, the final formula proposed is,

$$A = \frac{P}{2P_t},$$

where P is the total axle load, P_t is the wheel pressure and the factor 2 comes from the fact that there are 2 wheels attached to this axle. As mentioned above, the area is assumed to be a rectangle, $A = bh$ where,

$$h = 0.8712\sqrt{\frac{A}{0.5227}},$$

$$b = 0.6h.$$

The final contact area (wheel-to-ground) pressure becomes,

$$P_e = \frac{P}{2hb}.$$

Both of these formulas have been used and compared in this report.

3.1.3.3 Calculating Robustness in Maximal Slope

The models expressed in Section 3.1.3.1 describe the theoretical maximum slope a vehicle can climb or lean sideways. For the case of leaning, the final theoretical maximum value is a function with respect to the width and height of the vehicle. For the case of the maximum climbing angle, it is dependent on the friction between the wheel and the ground.

J.Y Wong provided a table [5] of typical values of the friction constant for different types of wet/dry roads. These values were used to calculate the theoretical maximum slope. A normal distribution was assumed for the measured slope with expected value μ and variance σ, $\theta \sim N(\mu, \sigma^2)$. The expected value μ is said to be the measured angle θ and σ was approximated to be some small artificial error $\Delta\theta$ such that the normal distribution becomes $\theta \sim N(\theta, (\Delta\theta)^2)$.

The final expression for the success rate P_{success} is given by,

$$P_{\text{success}} = 2\max(0, P(X \leq \theta) - P(X \leq \theta_{\max}), \tag{3.14}$$

where $P(X \leq x) = F_X(x) = \int_{-\infty}^{x} f_X(t)dt$ and $f_X(t)$ is the normal distribution function behaving like $X \sim N(\theta, (\Delta\theta)^2)$ and X is a stochastic value from said distribution.

The same success rate expression is used to calculate the robustness in the sideways leaning angle that is solely dependent on the geometry of the vehicle for its theoretical maximum.

3.1.3.4 Vehicle Velocity

For a given vehicle there exists a maximum velocity speed that drastically changes depending on what kind of ground surface it travels on, dry or wet conditions, and at what climbing slope angle. The standard maximum velocity of the vehicle is (usually) provided by the manufacturer where perfect conditions are assumed. By perfection conditions, the author refers to dry asphalt as ground surface and at a flat climbing angle. For this study, the velocity $v(t, \theta)$ of the vehicle is defined to be dependent on the ground surface $t \in \mathcal{T}$ and climbing angle θ. The angle θ is the climbing angle that the vehicle traverses and has values between $\theta \in [0, \theta_{tw}^{\max}]$ where $\theta_{tw}^{\max}$ is calculated using the models presented in Section 3.1.3.1 and depends on the moisture level w (wet/dry conditions) and terrain $t \in \mathcal{T}$. The maximum angle value $\theta_{tw}^{\max}$ depends partly on the different ground surfaces t used by J.Y Wong [5] and they in turn are used to construct the set $\mathcal{T} = \{\text{Asphalt, Concrete, ...}\}$.

Given a set (provided by the manufacturer of the vehicle or estimated by the driver) of initially estimated velocities $v(t, 0)$ of the vehicle at $\theta = 0$ for all the different ground surfaces $\forall t \in \mathcal{T}$, the final velocity model becomes,

$$v(t, \theta) = v(t, 0)\cos\left(\frac{\pi}{2}\frac{\theta}{\theta_{tw}^{\max}}\right). \tag{3.15}$$

This model will not be tested nor verified in this report. It servers as an outline for the change of velocity with respect to change in θ for different terrains $t \in \mathcal{T}$. It is important when studying the total traveling time for a vehicle following a path. Though this study mainly focuses on the robustness of roads rather than minimizing traveling time but a comparison between traveling time and robustness for different paths will be made in this report.

3.1.4 The Change in Water Content

Water content plays a significant role in the bearing capacity equation. To estimate the change in water content (or moisture level) in a soil, the water balance equation was studied with some detail. In this chapter, the water equation is showcased and explained.

3.1.4.1 The Water Equation

The general water balance equation is expressed as,

$$P = R + E + \Delta S, \tag{3.16}$$

where ΔS is the change in water storage, P is the precipitation, E is the evapotranspiration and R is the streamflow [45]. The term evapotranspiration means the evaporation of water due to heat from the sun and the plant transpiration of water due to the vegetation. The streamflow is a term that describes how water flows away from a point through its geometry (e.g a point on a downward hill will see water flowing away from that point at a rapid rate). All of these terms are measured in unit water content per time unit.

3.1.4.2 Water Equation in Detail

The water balance equation 3.16 is the most common way of expressing the balance of water in a system, however, that expression leaves out some details. To better understand in detail how the expression works we can re-write it as,

$$\frac{dS}{dt} = P(t) - E(t, \sigma T^4) - R(t, \Omega). \tag{3.17}$$

The water equation has now turned into a differential equation with respect to time t, σT^4 is energy per area unit where σ is Stefan–Boltzmann constant, T is temperature and Ω describes the geometry. The term σT^4 comes from the fact that there is an energy balance equation included in the water balance equation, where earth is assumed to behave according to the theory of black-body radiation [46]. There have been multiple studies whose purpose have been to find the exact correlation between the water balance equation and the energy balance equation [47, 48]. The main idea is the study the net radiation between the sun and earth to study how water evaporates. The energy balance equation is usually expressed as,

$$R_{\text{net}} = (1 - \alpha)K \downarrow + \sigma(\epsilon_{\text{sky}} T_{\text{sky}}^4 - \epsilon_{\text{earth}} T_{\text{earth}}^4),$$

where $(1 - \alpha)K \downarrow$ is the incoming energy from the sun minus the reflected part and $\sigma(\epsilon_{\text{sky}} T_{\text{sky}}^4 - \epsilon_{\text{earth}} T_{\text{earth}}^4)$ describes the black body radiation difference between the atmosphere and the surface where ϵ equals an area of exposure. This result is then applied to the water balance equation where later this is included in the evapotranspiration term mentioned above.

3.1.4.3 Worst Case Change in Water Content

For this report, the more detailed water balance equation is only partly experimented with. A simplified version of the water equation is used only to predict the potential worst-case changes in soil moisture if precipitation of any kind is present during a relocation where the vehicle drives on unpaved roads. It's used as a guide to measure the potential change Δw used for the bearing capacity (see Section 3.1.2.8) where w is the moisture level in a soil type.

To begin, one needs to determine the region of interest, at what depth the soil moisture change is of importance. This model assumes that the top layer of a soil T is of importance at a depth $d_T = 0.1$m. In a report written by Francisco José Del Toro-Guerrero et al. [49] it was stated that the soil moisture content changes more drastically closer to the top layer of a soil type while remaining more consistent at lower depths. The model will thereby only be accurate shortly after precipitation and represent the spike in water content, much like the ones seen in Toro-Guerrero's report. The following expresses the short term change in water content,

$$\Delta w(d_w, \rho_T) = 100\frac{d_w \rho_w}{d_T \rho_T}, \tag{3.18}$$

$$d_w = \int_{t_0}^{t_1} \dot{d}_w dt,$$

where $\Delta w(d_w, \rho_T)$ is the change in water content (measured in percentage) that depends on the two variables d_w which is the total precipitation (measured in length m) and ρ_T is the dry unit density of the soil T. The variable d_w is given by integrating $\dot{d}_w$ (which is the precipitation measured in m/hr) over some change in time $\Delta t = t_1 - t_0$. The term d_T is the aforementioned soil depth (which is assumed to be 0.1m) and ρ_W is the density of water.

Annually the total summation of change in water storage in the soil remains mostly zero unless a climate change is in effect, nonetheless during different seasons you might see dry/wet soil. The initial value for the current water storage w_0 can be provided by weather-data collecting sites such as SMHI [50]. The final soil moisture content is given by $w = w_0 + \Delta w$.

3.1.5 Exposure to Hostiles

One potential robustness is the idea of avoiding being exposed to hostile enemies while relocation in enemy territory. In this report, different models are presented to deal with different kinds of scenarios of potential exposure. None of these models are verified nor have been previously used in real-life situations. The idea is built around Carmenta Engine's own "exposure-to-threat" function where the authors' models should be seen as a potential improvement.

3.1.5.1 Carmenta Engine's Exposure Function

Assuming the line of sight is clear and not hindered by any obstacles in the way (e.g trees or mountains) the function creates a 2D circle on a 2D-above projection of a

region where a certain risk is taken if a vehicle enters the exposed area. The vehicle takes a penalty when entering that area but it is not directly correlated to time of exposure, meaning a vehicle could enter the exposed area and stay there for an unlimited time without taking an additional penalty. The penalty taken is constant over the entire exposed area and is not dependent on any distance from the source.

The purpose of the authors' models is to build upon this function and expand it to different scenarios where the distance to the source might affect the penalty (lesser chance of being exposed the further away you are) and including time under exposure where the risk increases the longer a vehicle exists within the exposed area.

3.1.5.2 Human Exposure

The first initial model is based on human eye-sight. The model works both for the case of the naked human eye or for the case of when binoculars are used to enhance the vision. Assume that the eye can see a maximum distance of $r_{\max}$ where this distance is greatly increased if the use of binoculars is present. The complete area of exposure is then on a 2D-circle seen from a 2D-above projection of a region with radius $r_{\max}$ and area $A_{\text{exposed}} = \pi r_{\max}^2$. If the source is placed in the center $\vec{p}_{\text{source}}$ of the circle, the exposure level will be non-zero on the circle unless any obstacle obscures the line of sight (e.g a mountain/trees covering the view) from the center point to the outer circle. The authors propose three different distance-risk models where the risk decreases with the distance from the source. All of these models are based on the assumption that the risk per second is 1.0 at the center of the circle and drops to 0.5 at a distance $r_{\max}/2$ away from the center. Below the different models are presented.

Given a 2D position $\vec{p}_{\text{current}}$ for the vehicle, a source position $\vec{p}_{\text{source}}$ and a maximum sight distance $r_{\max}$. The distance between the source and the vehicle is given by $r = ||\vec{p}_{\text{current}} - \vec{p}_{\text{source}}||$. The risk per unit time is now given by,

$$f_{\text{risk}}(r, r_{\max}) = kr + m = 1 - \frac{r}{r_{\max}} \tag{3.19}$$

$$f_{\text{risk}}(r, r_{\max}) = \beta \frac{1}{r^{\alpha}} = \frac{1}{r^{-\frac{1/2}{r_{\max}/2}}} \tag{3.20}$$

$$f_{\text{risk}}(r, r_{\max}) = \alpha e^{-\beta r} = e^{\frac{2}{r_{\max}} \ln(\frac{1}{2}) r}, \tag{3.21}$$

where function (3.19) is created using the standard linear function $kr + m$, function (3.20) is based on typical source radiation function $\beta \frac{1}{r^{\alpha}}$ and function (3.21) is based on the decreasing exponential function $\alpha e^{-\beta r}$. Each of these equations express different fall off of risk per unit time with respect to distance r between source and the vehicle. Observe that the risk per unit time is 0 if you are outside the circle for all the functions, $f_{\text{risk}}(r, r_{\max}) = 0,\ r > r_{\max}$. This is due to the fact that the vehicle is outside the exposure zone.

The total risk is calculated using,

$$r_{\text{TotalRisk}} = \int_{t_0}^{t_1} \int_{C} f_{\text{risk}}(||\vec{p} - \vec{p}_{\text{source}}||_{L2}, r_{\max}) d\vec{p} dt, \tag{3.22}$$

where the first integral is the line-integral on the curve C describing the path that the vehicle takes inside the exposed area and the second integral is the time-integral where the vehicle enters the exposed area at t_0 and leaves at t_1. The total robustness is calculated as,

$$p_{\text{success}} = 1 - r_{\text{TotalRisk}}.$$

3.1.5.3 Exposed to Radar

Compared to the human eyesight, a radar is assumed to have a constant risk-of-exposure per unit time over the exposed area, it does not decrease as the distance between the vehicle and source increases. Assume that the exposed area (just like in Carmenta Engine) is a 2D-circle on a 2D-above projection of a region on a map. The maximum range of exposure is, assuming that the radar isn't obscured by an object, a distance r_{max}. The model purposed by the authors to describe the risk per unit time is a projection of the vehicle on a circle where the vehicle cover some radius r, and the radar updates the view of the area at some frequency f. The further away the vehicle is, the smaller the radius r it covers on the circle and the period time for the radar is simply $T = 1/f$. In the worst case, the vehicle is spotted immediately, while in the best case the radar takes a total of $2\pi - L/(2\pi r)$ radius on a unit circle in the distance before it finds it, where L is the length of the side of the vehicle that is exposed and r is the distance between the vehicle and the radar. The radar scans the other side of the circle before finally reaching the part of the circle that the vehicle covers. The authors' model is based on the second scenario.

First one needs to calculate which side of the vehicle the radar sees which in worst case is the longest side, either the width of the vehicle l_w or the length of the vehicle l_l. Assume the radar is located at a point $\vec{p}_{\text{source}}$ and the vehicle is located at a point $\vec{p}$ traveling in a (normalized) direction $\vec{v}$. The distance vector between the source point and the vehicle $\vec{p}_{\text{source}\to p} = \vec{p} - \vec{p}_{\text{source}}$ will act as a direction creating an angle θ between itself and the direction of which the vehicle is traveling. Using the scalar product, the angle θ is calculated as,

$$\theta = \arccos\left(\frac{\vec{v} \cdot \vec{p}_{\text{source}\to p}}{||\vec{p}_{\text{source}\to p}||}\right).$$

The distance L projected towards the vehicle, as mentioned above, is now equal to,

$$L(\theta) = \sqrt{l_w^2 \cos^2(\theta) + l_l^2 \sin^2(\theta)},$$

where once again l_w is the width of the vehicle, l_l is the length of the vehicle and θ is the angle calculated above. Given this projection we can now approximate how many radiuses of a unit circle the vehicle covers,

$$R_{\text{covered}} = 2\pi - \frac{L(\theta)}{2\pi r},$$

where L is the projection length given above and $r = ||\vec{p} - \vec{p}_{\text{source}}||$ is the distance between the vehicle and the source. Note that this formula breaks down in the center of the circle where the radius $r \leq L/(2\pi)^2$, meaning that it's only valid for

$r > L/(2\pi)^2$. This will be handle later on where the risk per time unit will be equal to 1 for this illegal region.

The risk per time unit is now given by,

$$r_{\text{risk}} = \begin{cases} \frac{\omega}{R_{\text{covered}}} & r > \frac{L}{(2\pi)^2} \\ 1 & \text{otherwise} \end{cases} \tag{3.23}$$

where $\omega = 2\pi f$ is the angular velocity and f is the frequency of which the radar updates the exposed region. The total risk and robustness can be calculated with a line integral on a path C where the direction changes according to Ω during a time $t_0 \to t_1$,

$$r_{\text{TotalRisk}} = \int_\Omega \int_C \frac{\omega}{2\pi - \frac{L(\theta)}{2\pi||\vec{p}-\vec{p}_{\text{source}}||}} d\vec{p} d\theta \int_{t_0}^{t_1} dt, \tag{3.24}$$

assuming you are in the allowed region ($r > L/(2\pi)^2$) described above else the integral is simply,

$$1 \cdot \int_{t_0}^{t_1} dt.$$

The robustness is $P_{\text{success}} = 1 - r_{\text{TotalRisk}}$. Note that if the path C covers both the allowed region and the illegal region, one can split up the region using two different integrals that both can be calculated for each region. The sum of these two integrals will be the final result.

3.1.6 Total Initial Robustness

The total initial robustness for each path will be given by a Bayesian distribution from all of the physical robustness factors. The physical robustness factors include bearing capacity, maximum slope, and exposure to hostiles. These will all be weighted differently since for some cases exposure to enemies might be of more importance than maximum slope. The final expression is,

$$P_{\text{FinalRobustness}} = \prod_{i \in \mathcal{I}} P_i^{w_i}, \tag{3.25}$$

where the set $\mathcal{I}$ includes all of the above mentioned physical robustness factors, P_i is the robustness for factor i and w_i is the weighted term for factor i. This can be changed into a sum using exponential and logarithmic rules,

$$P_{\text{FinalRobustness}} = \exp\left[\sum_{i \in \mathcal{I}} w_i \log P_i\right] \tag{3.26}$$

After the initial robustness, the robustness value for the given path will be updated after the path has been tested further using Monte-Carlo simulation to see how robust the path is with respect to the graph G that it exists on.

3.2 Software Methods

This section will describe the software methods used throughout the project. This regards data loading and pre-processing but also robustness simulations.

3.2.1 Data Loading and Pre-Processing

For this project, the data was loaded in two steps. Firstly the OSM data is loaded. Afterward, the raster information is parsed and all important features are extracted. The complete data can be saved in a compressed and serialized memory representation. Since the input data is rather big, pre-processing all valuable data is advantageous since the subsequent loading times will be quicker.

3.2.1.1 Parsing OSM Data

The OSM data was downloaded from `https://www.openstreetmap.org/` and could be parsed into the application. OSM data includes OSM nodes connected by OSM ways. The ways could describe road networks or other vector information such as buildings, land segments, or public transport routes. The complete input XML data is parsed by the software and all the desired ways are saved. For this purpose, only the road networks are of interest and are saved. Some roads also include information about maximum speed which was also saved and taken into account for route finding.

3.2.1.2 Loading Big Rasters Without Hitting Memory Limits

The raster data used for this project is of spatial resolution 0.5 m which requires a lot of computations and memory for searching. For this reason, a tile-based parser and downsampler were implemented.

The parser utilizes the fact that the data is split into separate tiles that are much smaller in size ($\sim$ 150 MB). The tiles are parsed one by one, and appropriate terrain nodes are created where the terrain is classified as a traversable class. An illustration of raster tiles and traversable nodes can be seen in Fig. 3.2. The classes are provided according to the LAS 1.4 standard [51] and the classes deemed traversable were initially "2 Ground" and "11 Road Surface". For certain weather conditions, "9 Water" could be used for frozen lakes that could be traversed if ice depth is sufficient. The nodes are created with a lower resolution than the input raster which reduces the number of nodes that have to be traversed when searching for routes. This reduction loses some information from the original data but can still work as an approximation of which terrain that is traversable. Between the identified nodes, terrain edges can be constructed. The edges can still take the complete high-resolution raster into account, finding a possible route between the nodes if such a route exists. In this project, a line edge between the nodes was used but more sophisticated obstacle avoidance could be used at the cost of higher pre-computational time. The exact obstacle avoiding routes can be saved to disk, but are not necessarily loaded into memory at all times. The exact node-to-node path can be loaded on demand once the edge is contained in a route that should be displayed to the end-user.

One example of parsed terrain nodes and edges is seen in Fig. 3.3.

3.2.1.3 Merging of Data

With the OSM and raster data loaded the two can be merged into one continuous search graph. This was done by looking at all OSM nodes and finding the closest

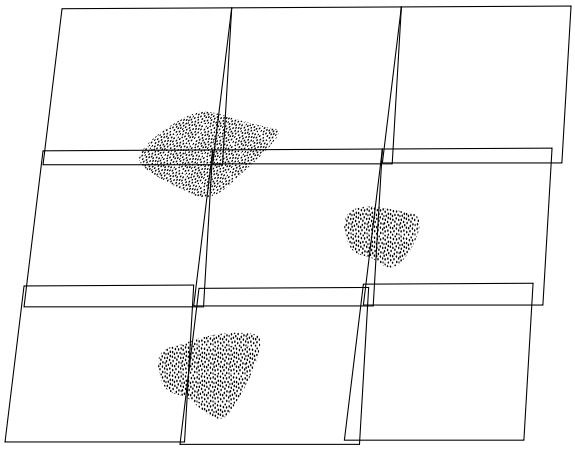

Fig. 3.2: Illustration of the raster parser implemented. The complete input raster is comprised of several smaller raster tiles. The tiles are parsed one by one in order to stay within memory limits. For every tile possible terrain nodes and edges are identified and saved.

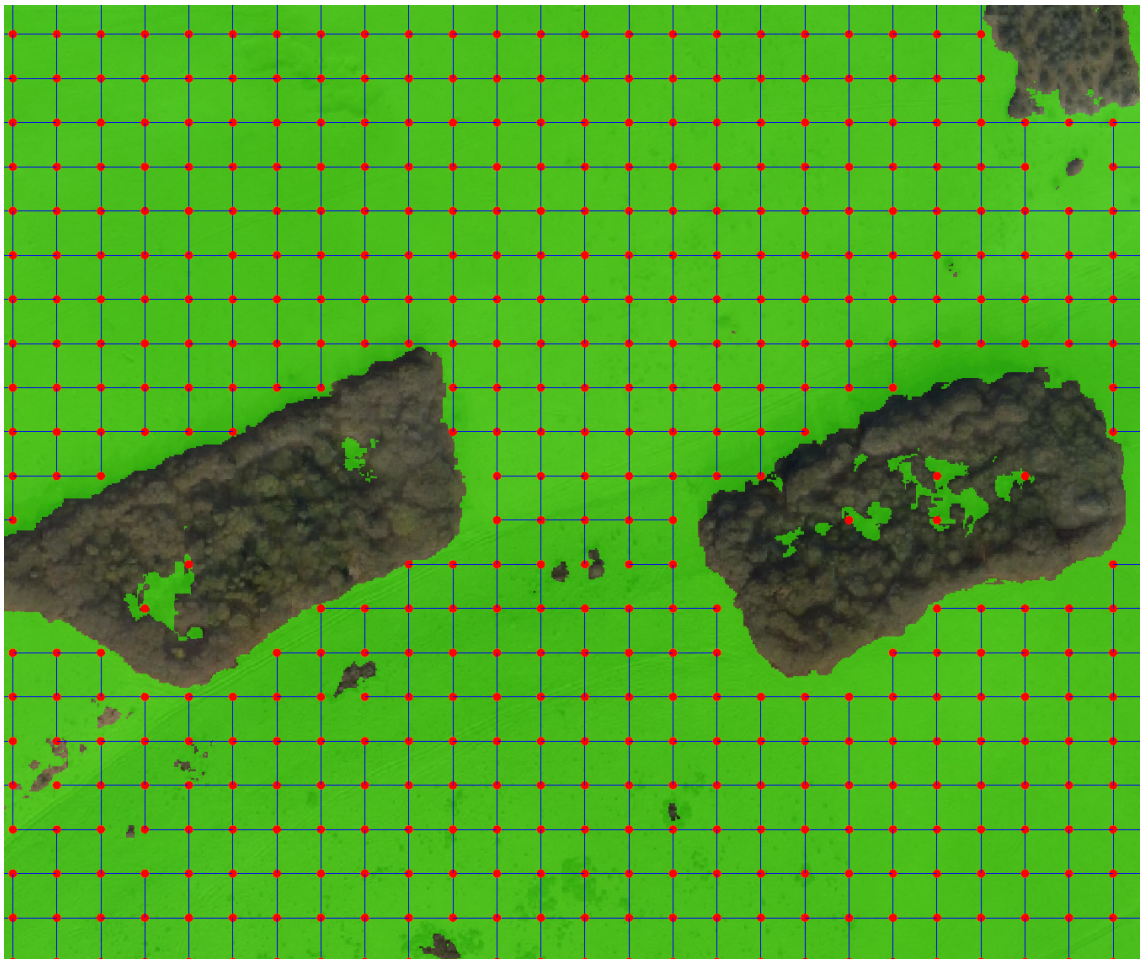

Fig. 3.3: Traversable terrain nodes and edges parsed from a higher resolution raster. The original raster or of spatial resolution 0.5 m whereas the approximated model is resampled to 20 m.

raster nodes. If the path between the nodes was traversable an edge could be added. This can be seen in Fig. 3.4.

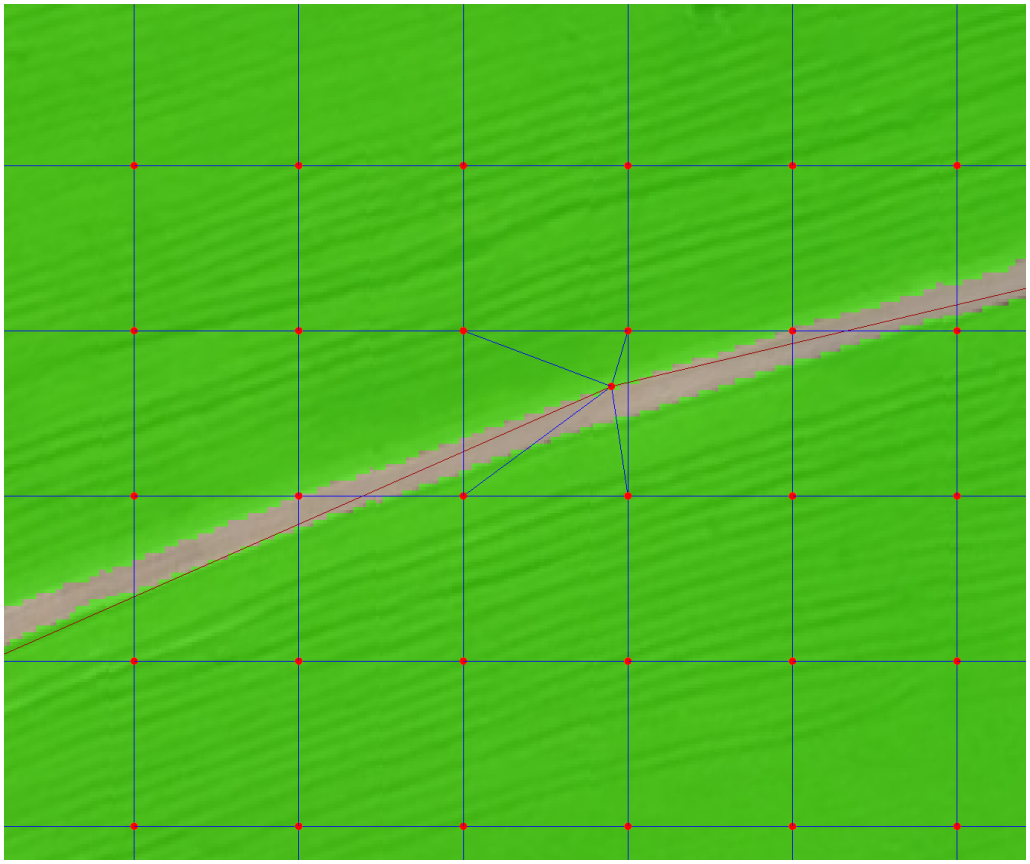

Fig. 3.4: Edges connected between OSM and raster data. The points in a grid formation represent the traversable nodes in terrain and all connected edges represent traversable terrain edges. The middle includes a road which can be seen by the raster classification layer and also the OSM edges. The OSM node in the middle is connected to 4 nearby raster nodes.

3.2.2 Two-sided Search for Generating Alternative Routes

For finding more than one route a two-sided search method was implemented. The method is based on the theory described in "Alternative Route Techniques and their Applications to the Stochastics on-time Arrival Problem" [52]. Given a start node, the traversable network is searched forwards using Dijkstra's algorithm until reaching the destination node. The search is then continued for a multiple cost of the shortest route. A similar search is done backward from the destination node. For both searches, the cost of reaching all intermediate nodes are saved and then joined with the result of the other search. This gives two sets of found points by both searches where the intersection gives possible *constraint points* as illustrated in Fig. 3.5.

Once all common points are found the complete routes can be constructed. Here the routes are picked by firstly minimizing total cost. The routes are also required

to be unique and have a maximum fraction of nodes used by other already accepted routes.

3.2.2.1 Complexity Analysis for Finding Alternative Routes

The worst-case time complexity of the two searches is $\mathcal{O}((V + E) \log V)$ where V is the number of vertices/nodes in the search space and E is the number of edges. The intersecting sets can be found in $\mathcal{O}(V)$ using a $\mathcal{O}(1)$ hash map. The complexity of filtering found candidate constraint points depends on home many routes are wanted. For a maximum of k routes, every route has to be compared to the previously accepted routes. This requires $\mathcal{O}(kV)$ time, but the computation also depends on the length of found routes. As the vertex length of a route is $\mathcal{O}(V)$, we in total get $\mathcal{O}(kV^2)$.

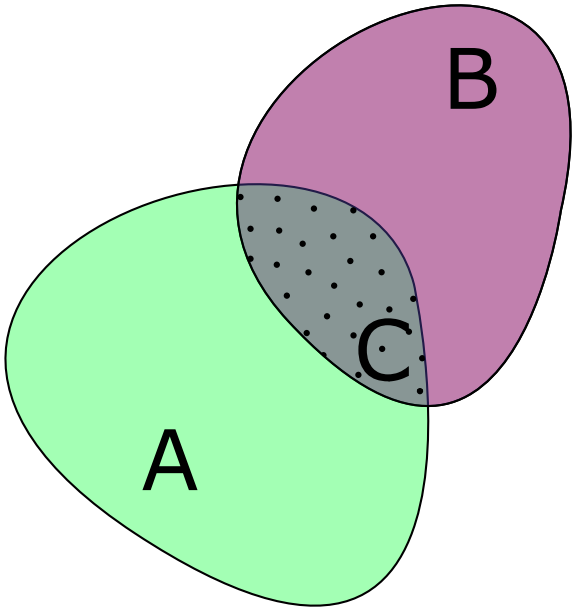

Fig. 3.5: Illustration of points found by a 2-search method where the set A and B are the points found by the first and second search. The intersection C denotes the points reached by both searches. The size of the set C can be modified by extending the search ranges for both searches. This reaches more points for both searches, assuming the search graph is larger.

3.2.3 Robustness Simulation

The robustness of a given route can be evaluated based on what happens if a segment of the route is blocked. In a military context, this could be due to enemy roadblocks or terrain destruction. In civilian situations, the same analysis could be made for evaluating the impact of traffic jams or accidents. The impact of such events can be approximated by simulating the impact of placing roadblocks along the path. In this project, this was done by starting with a complete route and adding circles blocking the road. The algorithm starts with previously traversed edges of zero cost.

This simplifies the re-searching by providing a quick shortcut for the initial path. The cost of the new search represents the additional cost for the obstructed path.

3.2.3.1 Sample Points

The sample points used for the simulation were placed either randomly for statistical purposes or equidistantly to understand the complete road and search for weaknesses. A sample point consisted of a 2D-circle with an arbitrary radius. Any road crossing by the circle would be temporarily eliminated. In other words, any edge E that would intersect with the circle (calculated using collision detection) would have its edge removed from the respective nodes V.

3.2.3.2 Complexity Analysis for Robustness Simulation

When using equally spaced points the number of sample points increases with V assuming a constant sample interval. The cost of re-running the routing is as stated previously $\mathcal{O}((|V|+|E|)\log(|V|)$ giving a total worst-case time complexity for simulation of $\mathcal{O}(|V|(|V|+|E|)\log(|V|)$. However, the simulation time per sample highly depends on the time penalty as the original route is already shortcutted. If the additional search is short, the simulation time per point is small.

3.2.4 Software Implementation

The methods were implemented in several parts

- Data parser for loading OSM and raster data,
- Route engine to generate and simulate routes between two points,
- Carmenta Engine map presentation for displaying generated routes and visualizing robustness.

The data parsing, route generation, and route simulation were implemented in Java as described above. The route simulation used a Java threading pool for simultaneous simulation. As the task of route simulation is highly parallel the theoretical performance improvement is big.

3.2.4.1 Carmenta Engine map presentation for displaying generated routes and visualizing robustness

For visualization Carmenta Engine was used. Carmenta Engine provides a simple to use SDK for presenting maps and map data. The design process was done using Carmenta Studio, which provides a simple way of constructing user interfaces. A screenshot of the used Carmenta px file can be seen in Fig. 3.6. Additionally, Java Swing was used for drawing user interfaces alongside the map presentation.

3.2.4.2 Geotools for Data Parsing and Distance Calculations

Geotools was used for parsing raster data and for calculating distances. The raster parser was used as a backend to the implemented tile-based parser, providing in-

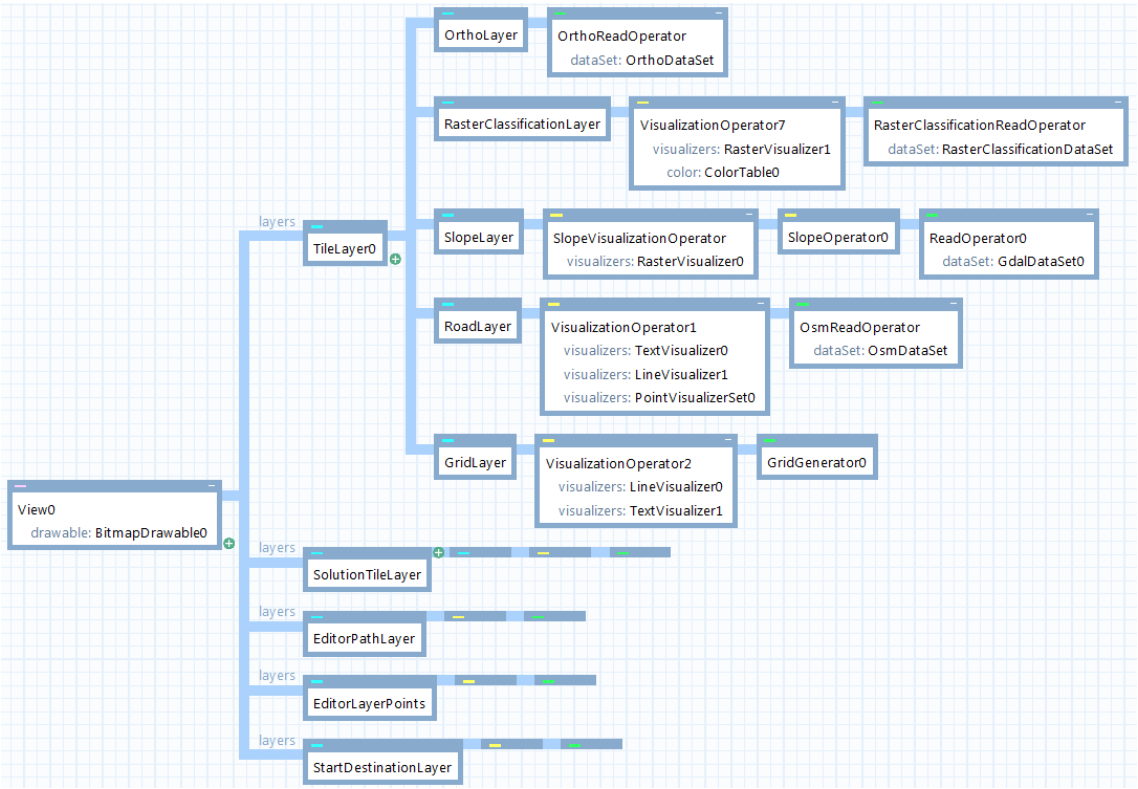

Fig. 3.6: Structure of px file used by Carmenta Engine for map presentation. The rendering is split into several layers such as road network, ortho photogrammetry, and generated routes.

terpolated height and classification queries. Geotool's gedetic distance calculator internally uses GeographicLib, which is based on *Algorithms for geodesics* [53]. By using this library, manual distance calculations for longitude-latitude coordinates were avoided.

3.2.4.3 Java Serialization

After parsing the input data the internal representation can be saved to avoid having to recompute everything again. This was done using Java's built-in serialization. As the provided serializer recursively builds the structure, initial attempts resulted in stack overflow as the search space gives a high recursion depth. This was fixed by breaking all pointers and instead giving all nodes and edges unique ids that can later be resolved when searching.

3.2.4.4 MATLAB

The mathematical software MATLAB was used mainly as a tool to simulate parts of the physical robustness such as maximum slope, total exposure, and bearing capacity. If somewhat successful results were given by the simulation, that was sound and reasonable, the MATLAB code would later be translated into Java classes and functions. Calculations, approximations, and regressions were conducted using part of MATLAB's own toolboxes. The MATLAB simulations were only a tiny part of the project and were mostly used as a check for mathematical errors.

3.2.5 Graphical User Interface (GUI)

To help the user navigate maps and routes, a graphical user interface GUI was created with the purpose of being simple-to-use. Read below for a complete function list. During this project, Carmenta Engine was used primarily to visualize maps, routes, and more. The user interface can be seen in Fig. 3.9. The weather options can also be changed by pressing the Config button, providing the options according to Fig. 3.8.

3.2.5.1 Pre-processing Tool

The data can be preprocessed using the interface as seen in Fig. 3.7. The layers of information needed is the road-map provided by OpenStreetMap [1] and the terrain-data provided by Lantmäteriet [2] and Vricon [4].

3.2.5.2 Generate Routes

To generate routes, load the preprocessed data. Simply press the generate button to acquire computer-optimized routes with respect to travel-time and robustness. Generated routes will be shown in the menu to the left. Right-click any of the routes to delete them or edit them. The editing mode allows the user to see what checkpoint is used for the generated route, changes to that route can be made by the user. Additional checkpoints can be provided by the user if the route does not fulfill certain key aspects. Additional checkpoints are generated by holding the Alt-key

and pressing on any of the edges visualized by the dashed lines. Press CANCEL if one wishes to leave the editor-mode. A display of the editing interface can be seen in Fig. 3.10.

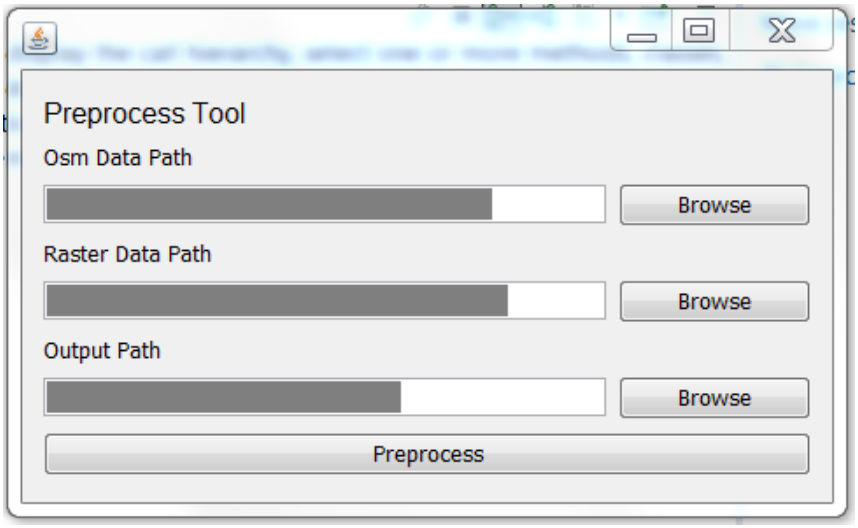

Fig. 3.7: Graphical user interface for the pre-processing tool. Here the user can browse for Osm and raster data and finally select an output path.

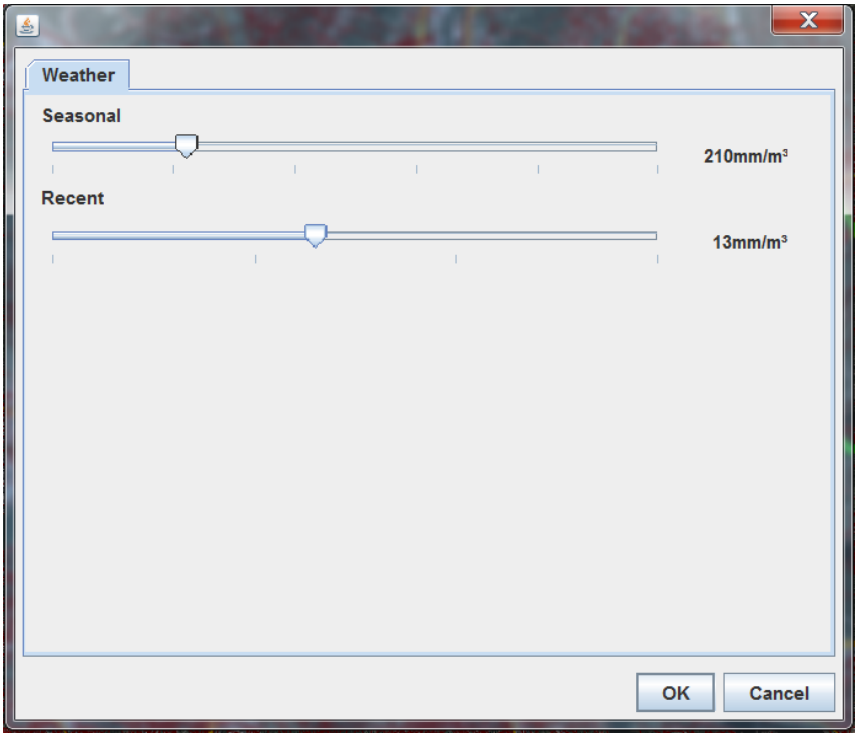

Fig. 3.8: Graphical user interface for changing weather conditions used for routing. The weather options includes seasonal weather conditions and recent precipitation.

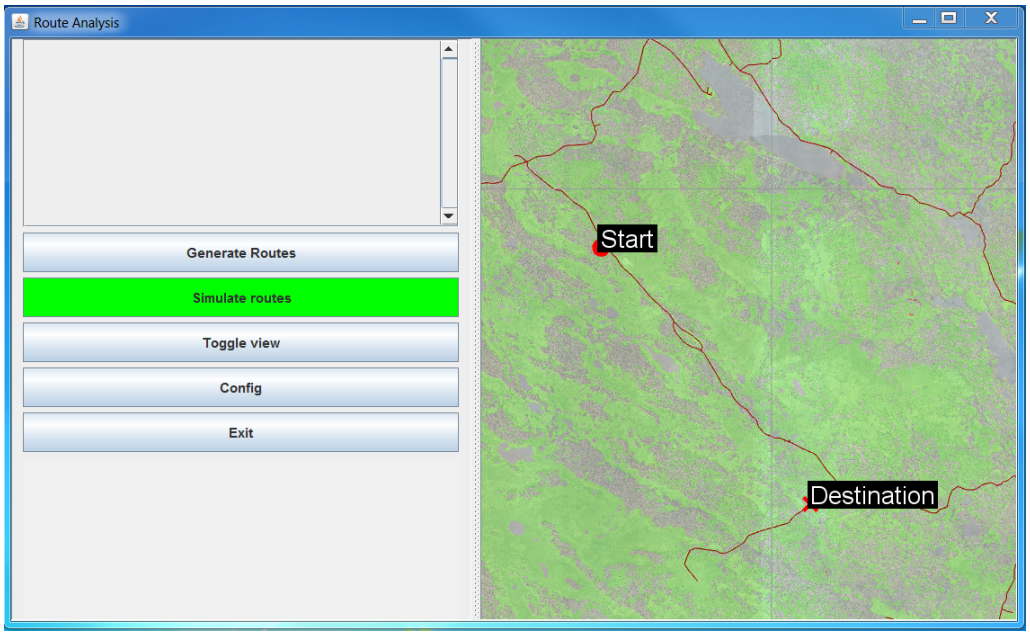

Fig. 3.9: Graphical user interface for route generation and analysis.

3.2.5.3 Penalty Time and Histogram

Given a set of routes visualized in the program (either generated by the program or created using the customized-route option), the routes can be simulated. After running the simulation the view can be toggled to show the simulated robustness. Each line segment of the route indicates a fictionalized obstacle that blocks the vehicle from traveling on the path. The colors indicate the additional time needed by the car (penalty-time) to find an alternative route and reach the final destination. Each route is present inside a list, select any of the routes to get a better view of the individual route. If one has selected a route within the list, a histogram will be present at the bottom-left-side of the program. The histogram indicates what type of penalty times are present for a selected route which gives the user an understanding of how stable a route is.

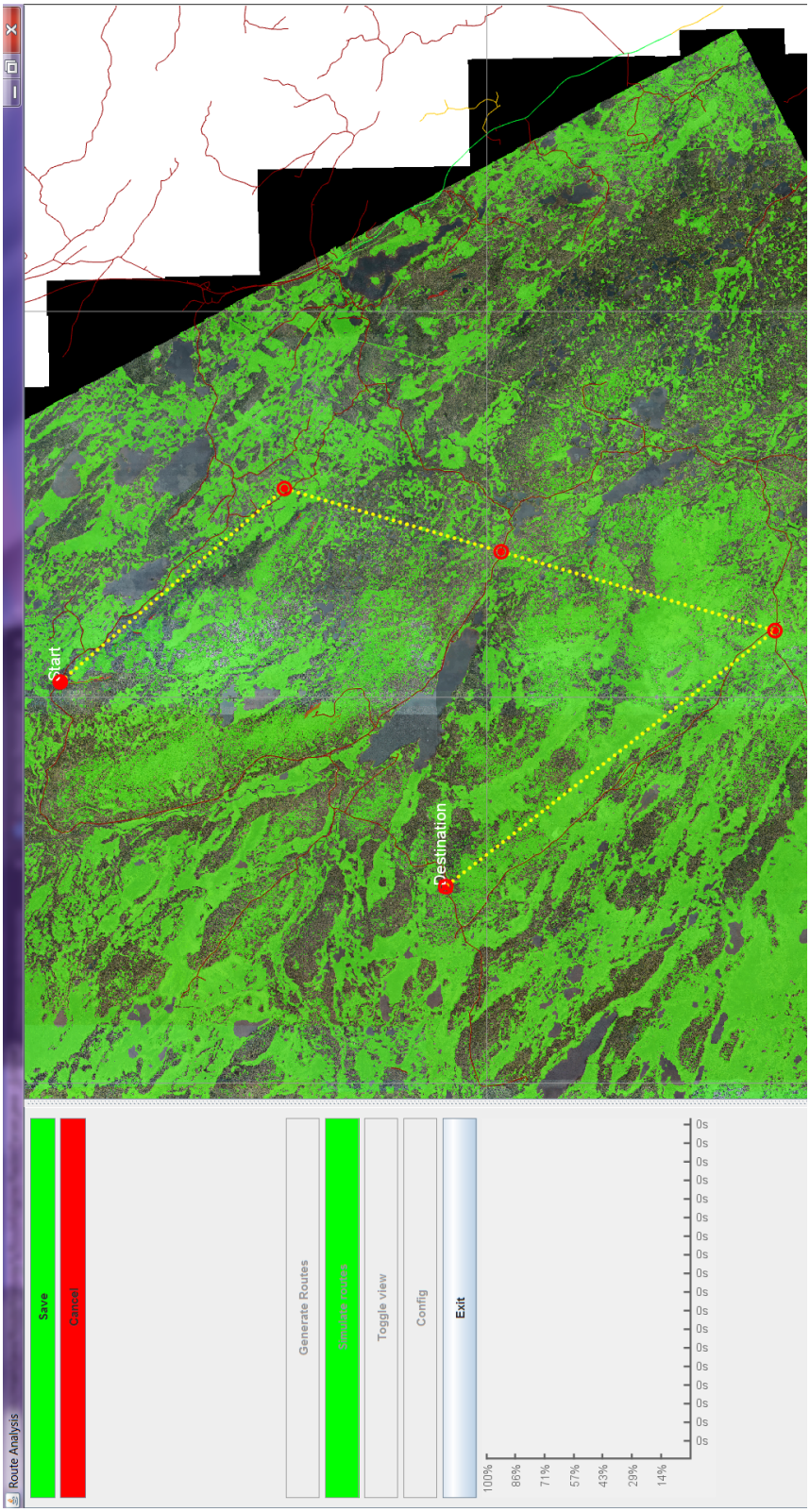

Fig. 3.10: Graphical User Interface (GUI) for modifying routes. The route starts from the bottom start position, indicated by "Start" and ends at the destination position, similarly indicated by "Destination". The route is forced to go through some points along the path which can be modified. They can be moved or deleted or a new point could be added between two existing points. Once the route is saved the new fastest route constrained to go through the added points is generated.

4

Results

The following chapter presents the results acquired during this master's thesis. The results center around the implemented path-finding program.

4.1 The Robustness-Index

The final robustness is a combination of the implemented and well-studied simulation test together with the theoretical robustness-index only partly implemented. The results from the simulation, after it's performed by a user on a generated path, are presented in a histogram to the user showcasing the different penalty-times. In addition, a color-scheme is used to visualize where on the path the most problematic parts are. This was never translated into an index.

The implemented aspects of the robustness-index are the bearing capacity and the maximum slope. The only dynamical event that changes these properties is the weather, which consists of the seasonal moisture level and total rainfall within the last 24 hours. Typical values of the robustness-index are either 0 or close to 1. This indicates that the implemented robustness-index behaves more like a true/false statement, that checks whether a vehicle can traverse chosen path.

Ground Type	Asphalt	Concrete	Gravel	Off road	Snow	Ice
Maximum Slope (Dry)	36.8°	36.8°	28.8°	33.0°	8.5°	4.0°
Maximum Slope (Wet)	24.2°	34.9°	-	21.8°	8.5°	4.0°

Table 4.1: Table illustrating the maximum traversable slope angle studied truck-vehicle 3.1.3 can climb given a set of different ground classifications and weather conditions. The result is achieved using measured values [5] inserted into approximated theoretical model (3.12).

The non-implemented factors for the robustness-index are simulated using MATLAB and translated partly into Java classes not yet implemented into the final program. Typical values are shown to vary a lot, especially compared to the above-mentioned robustness values that behaved more like a binary value.

4.2 Alternative Routes

The 2-way search proved to work quite well for generating alternative routes. One example of alternative routes can be seen in Fig. 4.1. The search has successfully found the 3 roads connecting the northern and southern parts of Gothenburg, Sweden. The quality of the alternative routes is hard to quantify, but in many cases, the alternative routes in ordinary road networks were reasonable and could have been generated by a human operator.

In terrain environments, the generated routes can look quite similar. The routes are forced to by unique by not using the same nodes. But for terrain, the number of alternative routes with minor modifications can be quite high as a terrain segment is made of several discrete nodes. This can be seen in Fig. 4.2. For cases like the one seen in the figure, it would've been more informative to get a path using the road network instead of crossing the terrain.

4.3 Simulation of Routes

The route simulation gives reasonable results when testing various scenarios. For parts of a route where there are few nearby alternatives, the route is highly penalized by road obstructions. Simulation of road networks can be seen in Fig. 4.3 with the different routes seen in Fig. 4.1. The critical areas of the routes, such as the bridges and tunnel, are denoted in the visualization.

When taking terrain into account, parts of the road which are nearby open terrain areas are delayed less than parts where the re-routability is low. This can be seen in Fig. 4.4 where terrain alternatives are available along some parts of the route.

4.4 Weather Effects

When generating alternative routes and performing simulations, the weather conditions proved conceptually to be important. See Fig 4.4 and Fig 4.5 for a typical case scenario. The soil used for this case was the Black soil. Given good weather conditions, a moisture level below 25%, and dry roads, the vehicle would have access to multiple choices when working around obstacles and trying to minimize the traveling time. Given poor weather conditions, moisture level above 30%, and wet roads, the vehicle would not have access to the terrain due to weak bearing capacity and not be able to traverse steep hills due to weak friction. This removes choices available for the vehicle, which resulted in increased penalty time when facing an obstacle.

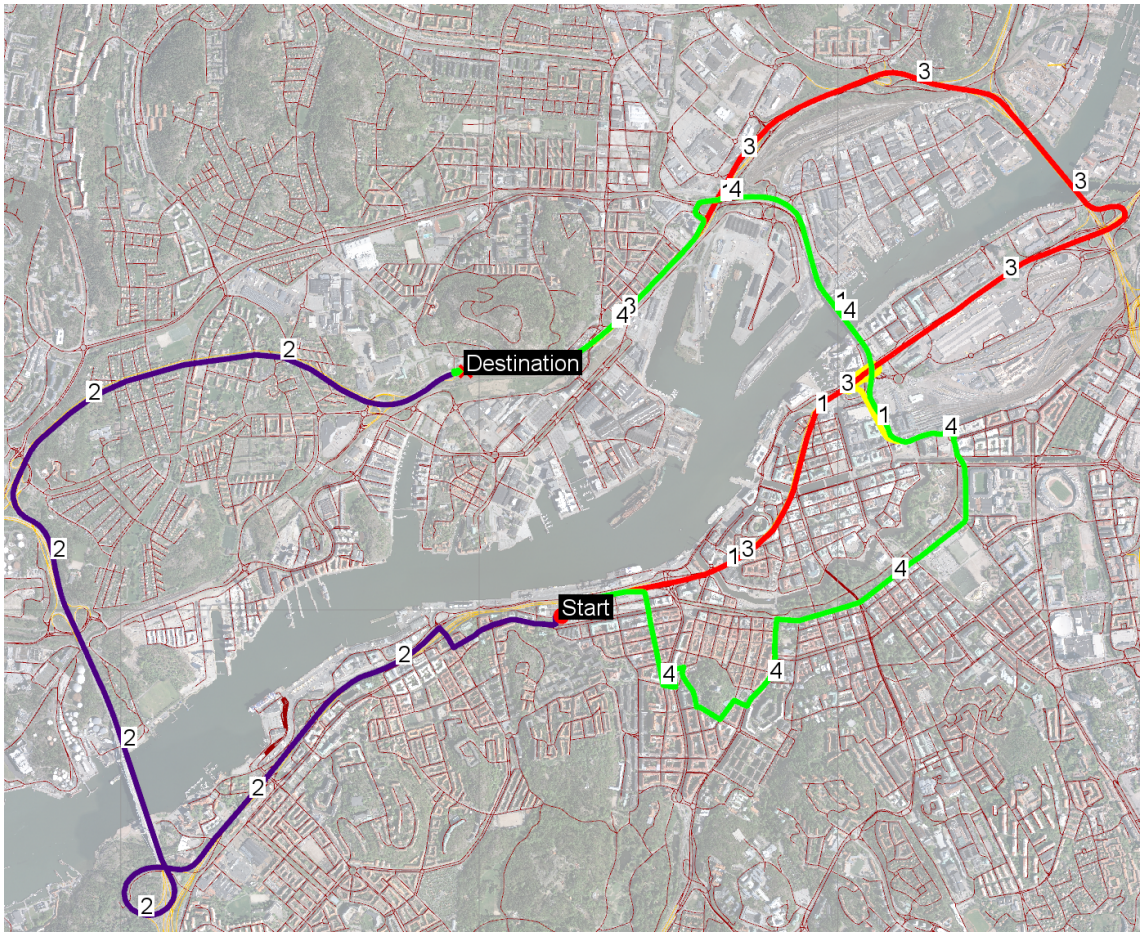

Fig. 4.1: Map visualizing an urban city enviornment, Gothenburg, in Sweden. Alternative routes generated by a 2-way search algorithm are visualized on the map. The different routes are indicated with a label displaying a number and are given different colors. The first route generated is given the label "one" and the second route is given the label "two" and so forth. The first route has the shortest traveling time t_1. The other generated routes are bounded by a scalar factor $c = 2$ of the shortest path, $t_i \leq ct_1$, $i \in \{2, 3, ...\}$, such that the path-planning program doesn't look for unreasonable options exceeding traveling time. Each of the routes are also unique from one another based on the intersection of the traversed nodes with a minimum of 50% unique nodes. In this case, the path-finding program is able to find all three different options of crossing the river, using two bridges and a tunnel.

Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

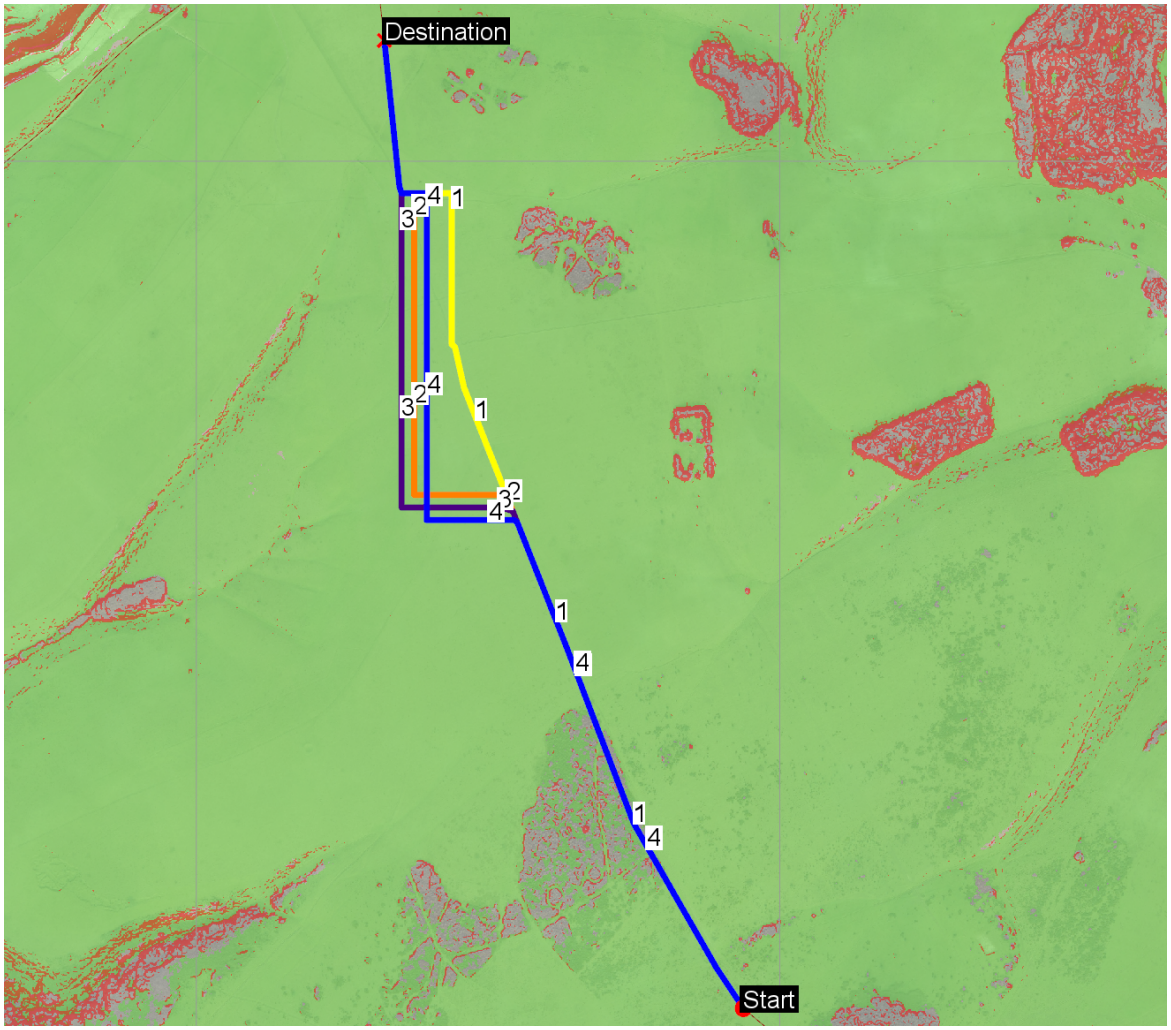

Fig. 4.2: Example of routes generated by a 2-way search method. Each route is given a label with a number expressing in what order they have been generated. Each route is also given a color. The first route, labeled with the number "one", has the shortest traveling time compared to the other generated options. Each alternative does not exceed a traveling time of two-times the traveling time of the shortest alternative. Each route is also unique based on the intersection of each traversed node, where each route needs to have a minimum of 50% unique nodes compared to all other routes. The start and destination are placed on two roads that are *not* connected by a road network near the divergence of the alternatives. The routes instead use the terrain in slighly different ways to complete the path.

Road data provided by © OpenStreetMap contributors [1].
Raster data and imagery provided by © Vricon [4].
Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

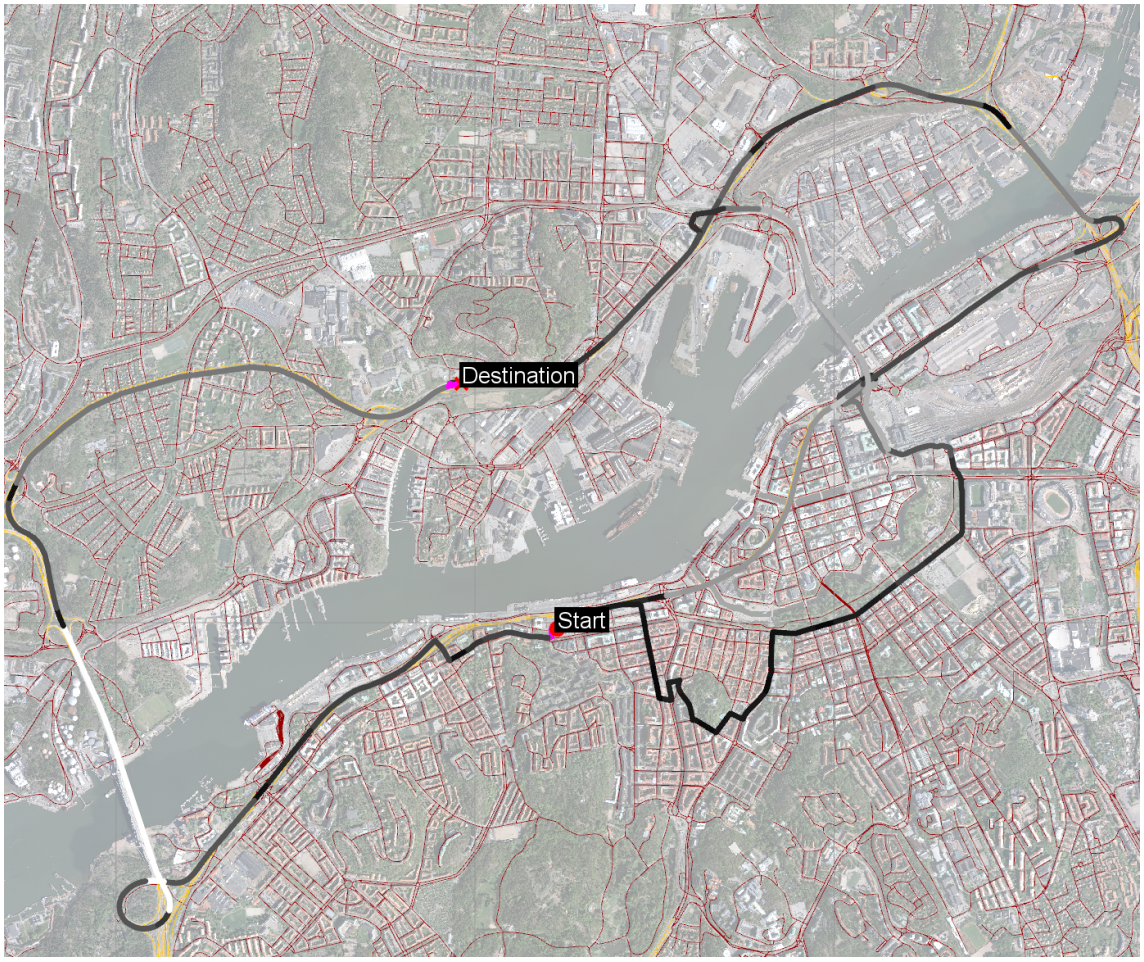

Fig. 4.3: Map visualizing an urban city enviornment, Gothenburg, in Sweden. A set of generated routes are displayed on the map. The solution from the robustness-simulation is displayed using a color-scheme and the colors are applied directly on-top of the alternative routes. Lighter colors of the routes indicate areas with a higher penalty while parts indicated with darker colors have less penalty. It is indicated by the simulation that the problematic areas for this city are the bridges and tunnel used to cross the nearby river when going from start to destination.

Road data provided by © OpenStreetMap contributors [1].
Raster data and imagery provided by © Lantmäteriet [2].
Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

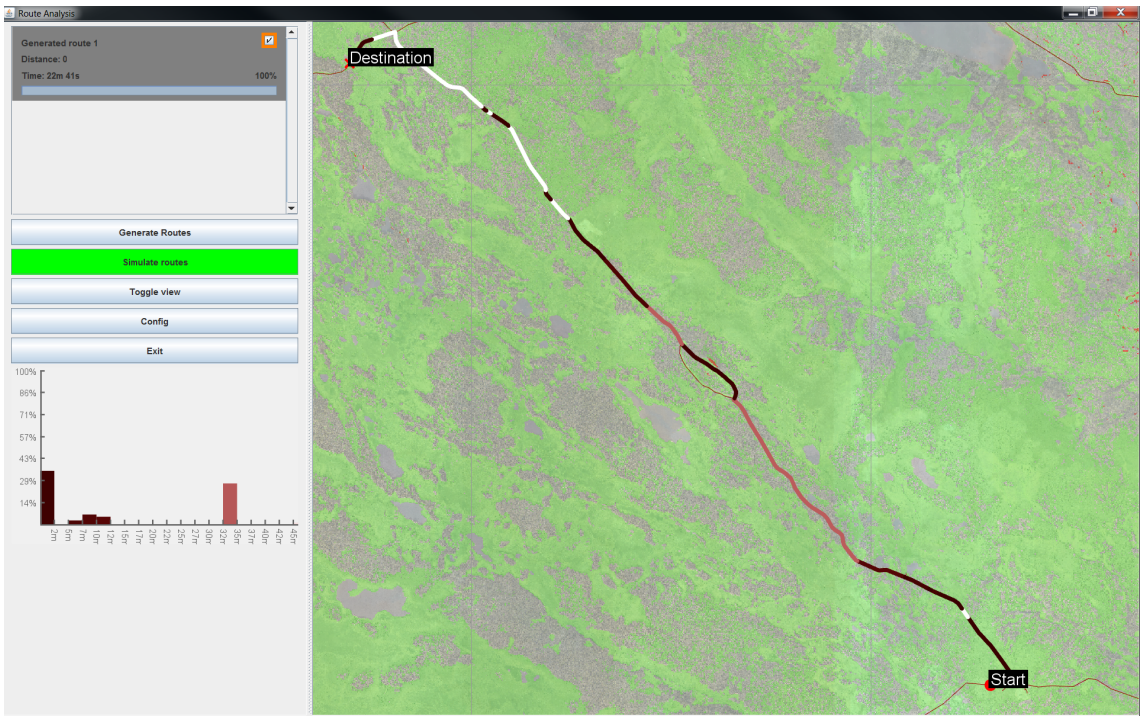

Fig. 4.4: Route generated by the software with good weather conditions. For roadblocks along the path, alternative terrain detours are available. The route starts from the bottom start position, indicated by "Start" and ends at the destination position, similarly indicated by "Destination". The route is viewed in robustness viewing mode which colors the route according to the delay in case of obstacles. Low delays are colored in dark colors whereas relatively high delays are colored in light colors. For parts of the route where the re-routing wouldn't finish quickly enough the road is colored magenta in the actual program. The delays are also shown in the histogram (bottom left), with the frequency i.e. how often a certain delay is seen.

Road data provided by © OpenStreetMap contributors [1].
Raster data and imagery provided by © Vricon [4].
Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

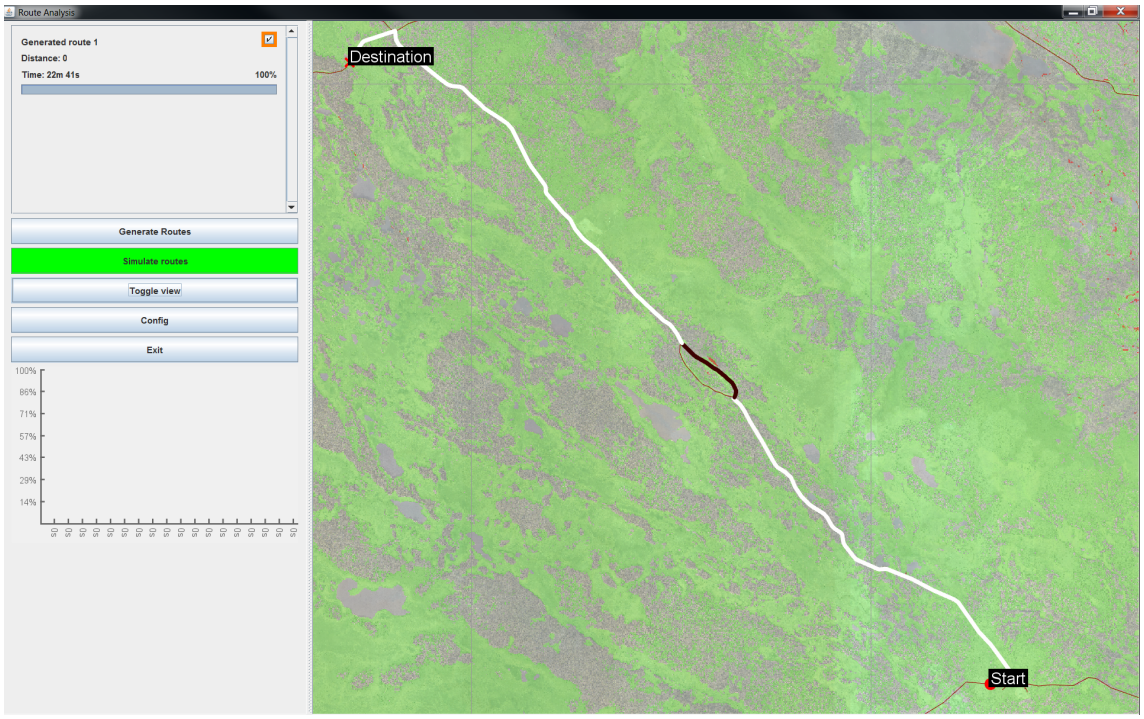

Fig. 4.5: Route generated by the software with poor weather conditions. For roadblocks along the path, alternative terrain detours are not available. This results in severe delays in some parts of the road. The route starts from the bottom start position, indicated by "Start" and ends at the destination position, similarly indicated by "Destination". The route is viewed in robustness viewing mode which colors the route according to the delay in case of obstacles. Low delays are colored in dark colors whereas relatively high delays are colored in light colors. For parts of the route where the re-routing wouldn't finish quickly enough the road is colored magenta in the actual program. The delays are also shown in the histogram (bottom left), with the frequency i.e. how often a certain delay is seen.

Road data provided by © OpenStreetMap contributors [1].
Raster data and imagery provided by © Vricon [4].
Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

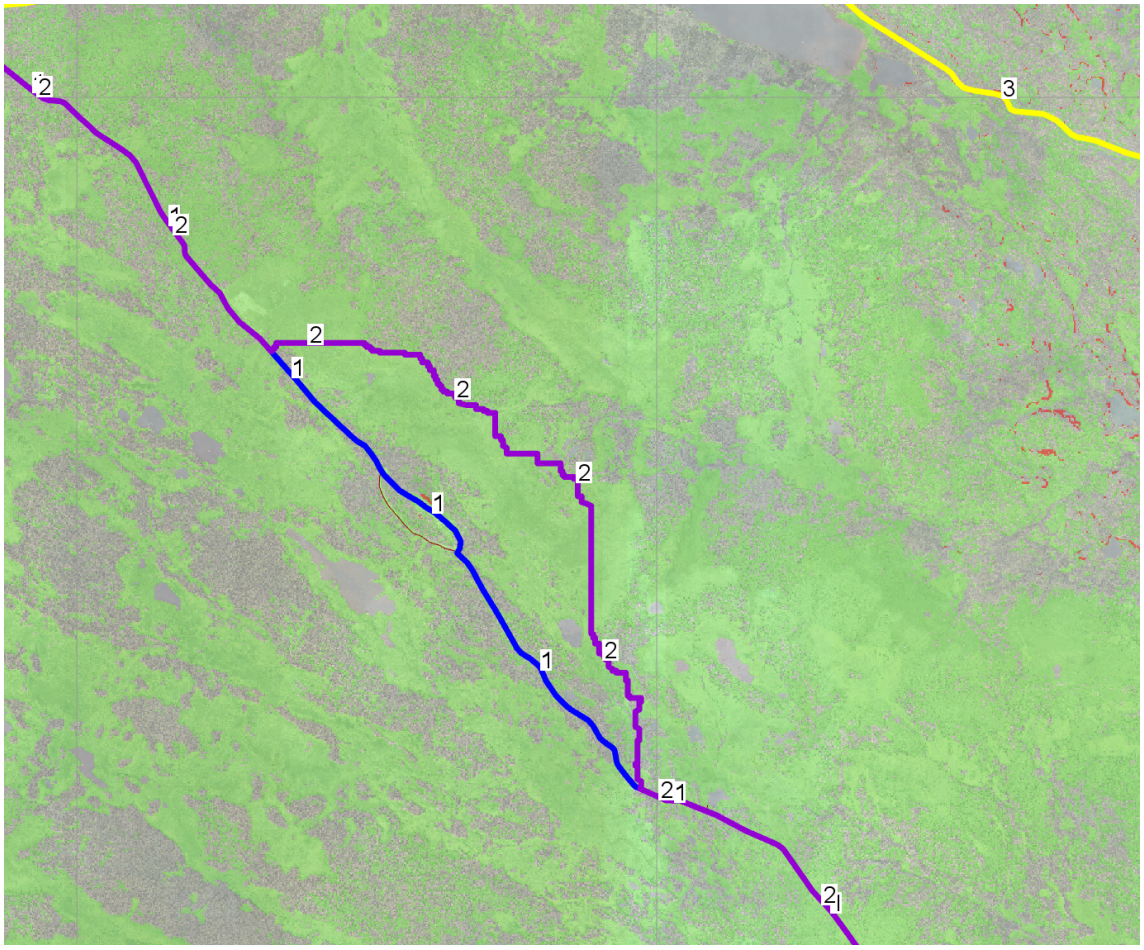

Fig. 4.6: Route generated by the software with good weather conditions. The path-finding program has access to the terrain. One of the generated routes traverses the terrain and is color in purple labeled 2. The other route, which doesn't traverse the terrain, is colored in blue and labeled 1. The off-road route makes sure not to traverse any illegal obstacle, however, the chosen off-road path might not be optimal and should instead be seen as just an alternative.

Road data provided by © OpenStreetMap contributors [1].
Raster data and imagery provided by © Vricon [4].
Note: for this image the display settings have been changed for the report to be readable in printed format. The actual map display is slightly different.

5

Discussion

In this chapter a discussion will be presented that covers all of the implemented and non-implemented ideas that came about during this project. The main focus is the results that are presented in the previous chapters.

5.1 Physical robustness

The physical robustness properties, implemented in the final product or not, showed promising modeling possibilities but with questionable accuracy. The purpose of studying these properties was to try and define at what conditions (if any) the chosen vehicle could in fact traverse on an unpaved road. The bearing capacity of the ground, the maximum traversable slope angle, and the threat of exposure became the three main physical properties. The first two were assumed to be dependent on the type of ground and the weather. Exposure would depend on a guess of what type of system that would be monitoring an area.

5.1.1 Maximum slope and bearing capacity

The maximum slope angle and the bearing capacity were studied for a different set of ground types. The maximum slope would decrease depending on the type of ground and if the ground was wet or not, something that wasn't easy to model. Instead, they use a threshold for the latest rain was used which proved to be a decent guess. If there had been more than a number of millimeters in rain for the past 24 hours, the ground could be assumed to remain wet at the top-level assuming the temperature is not abnormally hot. The main problem was this threshold that would switch the maximum slope instantly, there were no continuous values that account for "half"-wet ground, etc. Thereby the wetness-index was a sensitive parameter. Potentially one could interpolate the maximum friction of dry and wet ground to find intermediate values. Although, there is no clear indication what type of interpolation that should be used, since the relation is most likely not linear.

For bearing capacity, no real-life testings were made and thereby the main trust lied in the papers researched about each specific soil (Sec.3.1.2.4). The three studied soils were the Black and Red soil located in China and the Gypseous soil located in the middle east. Combining the properties of these soils at different moisture levels with the modified bearing capacity model (Eq.3.6) that accounted for vehicles on

unpaved roads, the result showed some fascinating numbers. The Red and Black soil behaved quite similarly, the moisture level acted as a cut-off value where if the moisture level reached 30% or above the vehicle could no longer traverse the soil without the risk of sinking. The Red soil had a higher cut-off value compared to the Black soil. The Gypseous soil, located in a different region, had much different behavior. If the soil was too dry, the vehicle could sink into the ground as well, creating an interval where the vehicle could safely traverse the soil. This shouldn't come as a surprise since the climate is very different in the middle east compared to in China. It was clear from the observations that moisture level (from seasonal groundwater storage and recent rainfalls) would have an impact on whether the vehicle could traverse safely off-road.

If the temperature drops below 0°C the bearing capacity would increase drastically due to the ground freezing (3.1.2.5). Even lakes could become traversable according to Gold's formula (3.9) if and only if the ice is thick enough. Thereby knowing if the ground is frozen or not proved to be crucial if one wanted to make improvements when routing, a lake could become an excellent short-cut. Previously non-traversable soil could become traversable meaning off-road is now a possibility. Though, since the top level of the ground is frozen, the maximum slope angle decreases significantly. The freezing phenomena have a trade-off, improved stability of the ground with the cost of it having to be quite flat to ensure a successful relocation.

The bearing robustness-index describes whether the vehicle will sink into the soil, however, the vehicle is already assumed to have a 15% sinkage from the Mellgren formula (see eq 3.13). The angle robustness-index indicates whether the vehicle can traverse the current angle, which is approximated using a gradient calculator used on the elevation data. These are all theoretical values that indicate some sort of limitation. To create an index out of this, it was assumed that the parameters were normally distributed with some kind of error. In general, there were no error bars available in the researched papers, which meant that fake error bars were simulated. Since no real validation has been made from the research in this project, these simulated errors could be completely false, and should instead be seen as an attempt at creating uncertainty which yields a robustness index used for route planning. The assumption that all of the parameters are normally distributed is sort of reasonable but could be completely wrong as well. The best way to look at the robustness index of the bearing capacity and the maximum slope is to view it as an indication that the chosen route might have some sort of risk or weakness. It's a warning rather than an absolute truth, nothing is measured with high certainty.

5.1.2 Weather condition

The weather, together with the current climate/season, was chosen as the main dynamical event that could drastically change the map and thereby the routing. The complete understanding of the weather lies in the hydrological model. Simplifications had to be made since the hydrological model is far too difficult to describe in great detail (see section 3.1.4.2).

The two main factors taken into account were the current water storage available in

the ground, which depends on the climate and season, and then the additional water storage that comes from precipitation. Since the bearing capacity and maximum slope angle were both centered around the top layers of the soil, the depth of interest was kept rather short instead of deep. This meant that precipitation could have a big impact on the current water storage since the water didn't have to sink that deep into the ground. The final product needed only the current water storage (for example provided by Swedish SMHI) and the total precipitation within the last 24 hours as it's input to estimate the total water moisture content in the ground for each specific soil. For the maximum slope angle, it wasn't as easy, since it relied on the top layer being wet. To determine if the ground remains wet after a certain amount of time isn't easy, so a simple threshold was applied to the current temperature and the total amount of rainfall within the last 24 hours. But for this project there existed no "half"-wet ground or any continuous value in between, so the wetness-index was simply a binary value, which made it very sensitive.

5.1.2.1 Implementation of the dynamical map

For the implementation, in practice, the weather became a set of values inserted into the config folder that the user could provide the program using local weather forecasts. The routes could look very different, the terrain could become an option, a steeper hill could be traversable or a frozen lake could become an excellent short-cut. All would depend dynamically on what the user could provide the program. For further improvements, this type of data should be inserted into the program using weather forecasting maps, terrain maps that specify soil and hydrological maps. A great source for most of these maps was found to be SMHI. The bearing robustness was applied to each off-road terrain node where for the sake of simplification, only one soil was chosen for all the off-road nodes. This is not true in reality, where the soil could differ greatly between different types of forests and fields. The gradient was applied to each edge, using the closest neighbor approximation. An approximation that would still provide the most basic of information, such as if a hill, tree, or building is in the way.

The robustness-index isn't of much interest for these types of physical properties, either you can relocate yourself through the forest or you cannot. The only time it might be interesting is if they reach the maximum slope one can climb, where one could take some sort of risking going up a steep hill on a rainy day. In general, it behaves more like a true or false statement, a binary value, rather than anything continuous where the risk indicates some sort of higher reasoning. The weather simply justifies whether off-road trafficability or hill-climbing is possible or not.

5.1.3 Exposure

Our exposure models are not verified nor tested and are not based on any researched models. These models should be seen as general guesses for what real-life exposure might behave like. Nonetheless, they do serve one excellent purpose, as a penalty function where the robustness decreases over time as a vehicle enters an exposed area. The only research done about these models was the maximum range of sight

from the human eye. Not much could be taken from it apart from our sight decreasing over distance and would also be greatly decreased if a fog of any kind was present.

The exposure models were never implemented into the final product, the theoretical models were analyzed and went through small testing where approximations were testing for accuracy. It is still unknown to the authors whether these models would make sense in any path-planning applications, however, the one found in the Carmenta engine seemed to have delivered no-go zones and small penalty zones. Given a budget, one could now exist for a small-time within any of the exposed areas within the Carmenta Engine. Our models would have been centered around this proposed result. Further improvements in routing could potentially be a consequence of being able to search within no-go zones tied to a budget.

The implementations would have been done using polygons. Exposure objects have been written in Java, one generic abstract class combined with multiple exposure classes ranging from human-eye exposure to radar-style exposure. This robustness index would potentially have been of more interest compared to the above-mentioned bearing and slope angle based robustness index. Instead of becoming a true/false scenario, the user could create a budget scenario, where the route could be partly inside an exposed area, such that the total risk doesn't exceed a certain level. This way, the traveling time could be minimized further if a short-cut existed inside an exposed area.

5.2 Non-physical Robustness

A simulation algorithm was created to visualize and measure the non-physical robustness. The non-physical robustness centers around the aspect of multiple-choice as the vehicle travels along the selected route. If an unexpected obstacle were to block the current path, would the vehicle be able to re-route from a point, and at what cost (penalty time)? Given a set of alternative routes, running this simulation proved positive when it comes to visualizing the problematic areas for each route, giving (sometimes) a clear hierarchy of which route that would on average have the smallest penalty time. Thereby the most robust route out of a set of alternative routes could be selected by the user based on the potential penalty time.

5.2.1 Robustness Simulation Advantages and Disadvantages

The simulation method implemented uses the initial route as a baseline route. The initial route is discounted which results in a search that grows in computational time with the delay. The advantage is that the entire route doesn't have to be re-routed again but only the new segment avoiding the obstacle. The disadvantage is that the newly found route could make adjustments early in the route for obstacles that are further along the path. One possible assumption would be that the user continues along the planned route until an obstacle is reached and then tries to find an alternative one, meaning that there is a cost for actually attempting the first route from the start.

5.2.2 Visual Representation and Histogram

One of the strong features that resulted from the simulation was the visual aspect. The penalty time followed a color-scheme which made it easy for the user to identify the safe part and the more challenging parts of a selected path. When presented with multiple routes, each route could be analyzed based on this penalty times along the path shown in color. Combining the route with a complete histogram that indicated how often a bad sub-path would occur, the user could then chose the best option based on total travel time and potential penalty time. This result could have many applications. One aspect would be finding the most robust way for an ambulance unit in a traffic-heavy city, where the shortest path combined with the least amount of potential penalty time might be the best option easily illustrated using this program.

5.2.3 Simulating the dynamic map

The simulation seemed to work well with the dynamical side of the map. If the route had access to the terrain, it could create small detours around the virtual obstacle and minimizing the penalty time. Changing the weather conditions from good to poor would result in the terrain not being accessible and the penalty times would be much greater. If substantial differences are observed, the user must take into account the weather conditions more carefully before a relocation should begin.

5.3 Alternative Routes

As the alternative route generation seems to work well for road routes, the approach is recommended to use when trying to quickly generate alternatives. The computational complexity in comparison to other methods is quite low as the search is only done twice. However, the search distance used in this project is a bit further than the shortest path to give more alternatives. For this very reason the number of computations is not exactly double but a bit higher. For this project the simulation takes vastly more time than the generation of routes, so the total time taken is barely affected by this decision. For real-time or near-real-time applications the additional search factor should probably be lowered, at least for routing operations with a long total distance.

The routes presented by the software for terrain routing were not too different. This is since routes are constrained to be unique by looking at the percentage of common nodes between all found routes. As the terrain routes use slightly different nodes in the same area, they are very similar without being similar according to the internal comparison used. This could be improved by not looking at the exact nodes visited and instead compare the rough areas traversed. By constructing a grid the closest grid points along the route can be filled, either by binary values indicating if the grid cells have been used or by other continuous distributions w.r.t. distance. The new route candidates can calculate grid cell values to see if the two routes use the same areas of the map. Other metrics could also be used such as Hausdorff distance [54].

5.3.1 Pareto Optimal Alternatives

Another way of generating alternative routes would be to solve a bi-objective optimization problem where one minimizes the traveling time while maximizing the robustness-index. Translating the bi-objective optimization problem into a single objective optimization problem using a scalarizing approach would yield a set of Pareto points. Varying the scalars would generate different Pareto points, where a subset of optimal Pareto points could be generated. Each optimal Pareto point would represent an optimal route, multiple optimal Pareto points would result in multiple different routes.

5.4 Implementation

The implementation was done as a stand-alone java-based program that communicated with Carmenta Engine to visualize maps and other graphical components. This was done such that the algorithms created in this project could be applied with any sort of map-visualizing based program and not just Carmenta Engine. This section discusses each implementation by itself and points out key choices made and comments on the overall user experience and performance.

5.4.1 Java implementation

The Java implementation is based on object-oriented programming. To be as flexible as possible, abstract classes were chosen as a top layer that would later extend real classes. For example, there would exist a generic node class that describes basic information a node needs. Then two real types were made, OSM nodes and terrain nodes, that would extend this abstract class to simplify and better structure the coding aspect.

5.4.2 Carmenta Engine Experience

Carmenta Engine made it very easy to present maps to the end-user. The SDK has many interfaces to common data types such as GeoTIFF and Shape used in this project. Since the software can pre-cache large rasters and a GPU accelerated renderer, the performance is really good.

5.4.3 Program - GUI

The graphical user interface was designed to be as simple and user-friendly as possible. All of the buttons are available on the side with the map visualizer from the Carmenta Engine taking up most of the space. The config folder creates a pop-up window where the user can browse for different map-data or change the current weather conditions. All the buttons have easily understood functions such as generating routes, simulating routes, or changing views. Built-in functions from Carmenta Engine were used as frequently as possible to simplify the task of creating a well-functional GUI and proved to be very beneficial for this project.

Once routes are generated they will appear inside a list. This way of presenting the solution was chosen because it felt like the user should have access to all of the routes and directly see the differences of travel time and robustness. Colors were chosen as the primary way of differentiating the routes.

If the user selects one route and then right clicks it, the user can enter an editor mode. This might not be intuitive for the user, but we found it to be the best solution with minimal clicks. While in editing mode, the user must select the nodes describing the checkpoints used for creating the route. They can be moved after having been selected, and more checkpoints can be added by holding the Alt key. All of these functions were taking directly from Carmenta Engine and from our own experience it seemed to work pretty smooth. After that, the user can either accept the edited route or throw it away using the cancel button.

Given a set of routes presented in a list, the user now has access to the simulation button, which changes color once active to indicate that the program is busy. Since the user had to be given some feedback that the simulation had started, changing the color of the simulation button from green to red seemed like a rather quick and easily understood signal to the user. The user can track the process by studying the processing bar next to each route that also comes with an estimated time. After the simulation is down, the user has access to the "change view" button, which seems rather odd at first-hand. If the button is pressed the user can now see the penalty-times along each road visualized using a color-scheme. Since it appeared very cluttered, we chose to highlight each solution if the user selects that solution within the list, lowering the opacity of all the other solutions.

No actual user-tests were performed during this project and instead the only feedback presented to us was our own and partly our supervisors. This area could be furthered improved given a set of data from user-feedback. Colors were used as often as possible to provide the user with key information, updates, or warnings. It was deemed to be the most effective and intuitive way of communicating with the user.

5.5 Future work

The following section comments on all of the possible future improvements that would hopefully benefit our implementation.

5.5.1 Bi-Objective Path-finding

One of the main improvements to be made is to change the path-finding algorithm to minimize traveling-time and maximize the robustness-index as a bi-objective optimization problem. Due to a limited amount of time, the current solution minimizes the traveling-time and prints the final robustness while not being affected by it. When finding alternatives, the current solution has a budget of search-space and uniqueness which results in a set of alternative solutions. A better way of presenting alternative solutions would be to print the Pareto optimal points generated from solving the bi-objective optimization problem. A budget for search-space and unique-

ness among the different paths would become constraints. The user could generate their paths by selecting checkpoints, but the program still only optimizes for traveling time given these checkpoints. We believe the implementation of bi-objective optimization to be a crucial improvement because it could potentially lead to much better alternatives for the user.

The Pareto optimal points given by the bi-objective optimization problem could become alternative routes if implemented with care and in an efficient way. Having said that, such points might not exactly be unique unless one solves the optimization problem using a different set of weighted constants. Thereby a variation in the use of weighted constants might result in better alternatives.

Our current Dijkstra's algorithm could still be used if the bi-objective optimization problem is transformed into a single objective optimization problem. Otherwise, it needs to be rewritten as a multi-dimensional Dijkstra's algorithm which might not be an easy task. Using different weights would allow the Dijkstra's algorithm to make different choices, which leads to possibly different routes. Combining this with the 2-way-search algorithm, that generates alternative routes, might provide the user with lots of great alternatives, as long as the user isn't overwhelmed with them. Picking the right one to print as a result to the user is still a difficult task.

5.5.2 Optimizing Route In Terrain - Kinetic Operator

Using a kinetic operator, one could potentially optimize the route in an off-road environment. The kinetic operator expresses the energy in the system and is applied to the motion of the vehicle and its change in position. Assuming the law of conservation of energy, the kinetic operator could optimize the behavior of the vehicle when choosing the best optimal terrain route.

For example, if a hill is present the current algorithm would only ask the following question, is it too steep? If not, the program will, without consequence, climb the hill. But using a kinetic operator one could optimize the terrain routing where the hill now becomes a negative motion for the vehicle. The operator will then try to avoid the hill to minimize the loss of energy. That way, the program might select the path with additional care. Keeping the velocity as high as possible would have a positive impact on traveling time.

This might improve the choices made in the terrain, although, it might come at a cost in runtime and performance. Some of this could be done in the pre-processing stage using pre-computed values. Additionally, one could recreate the terrain network, in the pre-processing phase, such that the only path in the terrain is the one the kinetic operator deems best. However, that would limit the number of alternative routes provided by the terrain, which for this project is a key feature.

5.5.3 Current Dynamical Events and Soils

Currently, the main studied dynamical event is the weather, more specifically the current water storage in the ground, rainfall, and temperature. Most affected prop-

erties are the maximum traversable slope angle and the bearing capacity of the ground, including water, such as a lake, which at a freezing state could become a short-cut if used. The temperature aspect never made it into the final product, though models have been studied and implemented into MATLAB simulations and Java classes. Changing the friction of the ground by assuming it is in a wet condition created more no-go zones where too steep areas became unreachable. Increasing the moisture in the soil had an equal effect where the soil could become unavailable for the user. Despite calculating a normally distributed error inside the parameters, the behavior is more or less a binary value, where parts of the map are either traversable or no-go zones. Additional studies and accurate errors might change this behavior where a user instead takes a risk entering the off-road environment for a period of time, or climbing a steep hill. As it stands now, there isn't much to optimize, unless you are given the choice of two different off-road environments where one can withstand the current wetness/dryness better than the other.

Additional soil types might be of interest to study and is necessary to create an authentic program, having said that, the authors believe their study about three different soil types covers well enough the boundaries of how soils behave with respect to different moisture levels. Moreover, it is important to point out that the soil properties are impacted by more features other than moisture level and temperature, such as compactness which could have huge impact on bearing compactness. The soils studied indicated that, translating the results into the modified bearing capacity equation, compactness could increase the overall capacity by many factors. A study indicated, that temperatures above freezing temperature had minimum effect on soil properties, in most cases, and neglecting it will probably not leave out a large error, though it is a potential improvement in the model.

5.5.4 More Dynamical Events

Additional events not yet implemented would be hostile areas, blown up bridges and additional weather properties such as temperature. Temperatures below 0° has been studied using the theory of frozen ground physics, but nothing is so far implemented into the final program. Only theoretical studies have been made with simulations written in MATLAB to study the bearing capacity of ice at different levels of depth. Additionally this MATLAB code has been translated into Java classes. The events that would occur would be increased bearing capacity for every soil, which now would additionally include lakes etc. A frozen soil would however contribute to a lower maximum traversable slope since the first layer of ground is assumed to be covered in snow or ice which has a lower wheel to ground friction coefficient.

Hostile areas would be visualized using polygons written as a layer on top of the map. Some models have been created in hopes of illustrating the risk of exposure when entering such a territory, none of which is implemented into the final product.

5.5.5 Improved Exposure Model

The exposure models needs to be re-worked with additional research related to line-of-sight from the human eye and the behaviour of radar systems and how they locating targets. Additionally, another type of threat present for military usage would be entering vantage points where enemy hostiles have a clear sight advantage or height advantage. Using line-of-sight algorithms on elevation data would potentially indicate if the path-finding algorithm selects (in its search for a route) a sub-path that would include such a dangerous vantage point for hostiles. For example, valleys would be a clear dangerous zone due to the disadvantage in height difference if enemy hostiles were to be present.

5.5.6 Additional Classes and Classification

Currently only a limited amount of terrain classes are used inside the program. These classes are assumed to be accurately classified and behave nicely enough for a vehicle to traverse under the right conditions. The terrain is assumed to be smooth. If rough terrain is present inside the used classes this information is not acknowledged by the robustness-index.

Vricon provided the terrain data with the classes and classification. Using additional classes, such as lakes with frozen conditions, would increase the options of off road routes. Additional options slows down the performance but could yield better alternative routes. With improved classification, resulting in more classes, better routes could potentially be generated with path-planning algorithms at the cost of performance. Vricon provides resolutions down to the scale of 50 cm, sadly not much time was spent during this project to analyze how detailed terrain could be useful. Due to performance and stability issues the terrain was almost immediately scaled up, leaving out a great deal of detail.

5.5.7 Additional Vehicle Models and Customization

Currently the only studied vehicle is a truck-like vehicle with the purpose of having a radar system mounted on top of it. The model expressing the vehicle are so far very simplified. But additional types of vehicles could be studied and selected by the user. Currently the vehicle is written into the source code and not provided by the user. Another type of vehicle that could be used are tracked vehicles, such as tanks. They will behave differently for example in off road environments and have a different maximum slope it can traverse.

5.5.8 Scaling up and scaling down

Given a very detailed high-resolution raster data one of the main negative aspects of studying it are the (very likely) performance issues. Either very excessive run-time or “out of memory”-issues were two common reoccurring challenges for this project. To solve this a re-scaling was done to downsize the overall data, which minimizes the use of memory and the search space for the algorithms which minimizes the run-time. However, the downside of scaling down the data is the loss of information, key

details that for some areas of interest could be vital. For example, when studying one part of a map where trees and road-width are of interest then high-resolution raster data can provide very accurate details needed to perform the correct measurements, if the data is down-scaled such scenarios are lost. Part of the work is to identify key areas where high-resolution data is needed. Thereby, an algorithm that knows when to scale down the data and scale up the data could improve the overall accuracy with the current implementation. One key scenario where this could be vital is when performing a simulation in the terrain. Although not much time was spent on this algorithm, the authors believe this to be the main key-area where high resolution data could become vital for the path-planning program.

5.5.9 Improved stability and performance

One big concern when using large amounts of data is concurrent memory usage. During this project data sizes had to be adjusted to fit into memory. Java being a garbage collected language removes some memory management requirements from the developer but optimal usage is not guarantied. Using a language like C++ would give further opportunities to analyze memory strategies and better allocate system resources. The current implementation assumes the entire search volume is contained in memory, which is not always required. A similar tile based approach could be applied for searching as well, leaving some parts in memory uncompressed or on disk. When searching the tiles could be loaded and searched momentarily, reducing the concurrent memory usage. Such an implementation would require care as to which tiles to be loaded in order to get good performance. When simulating routes a new route has to be found avoiding the virtual obstacle. If the new route only has a few areas with detours the routing endeavor could be contained to only a few tiles - possible limiting the number of tiles to be loaded.

The current implementation is written in Java. Even though the current implementation is multi-threaded, the performance could most likely be improved. This could either be improved by more carefully planning shared memory or by rewriting the software in a language like C++.

5.5.10 Improved sampling points and stochastic analysis

The current simulation selects a certain amount of equidistant sampling points. A clear improvement would be to implemented a better logic for selecting sampling points. For example, selecting equidistant points along a bridge doesn't provide additional information, at least compared to the additional processing time. Instead one could use one, or perhaps a few, point(s) to describe the entire bridge, which in general should have a pretty high penalty-time. Such a logic could be constructed using classification of the roads and determine if a road changes drastically along its path or if the penalty-time is consistent along its complete path. This would possibly lead to a fewer amount of points visualizing the exact same result. That alone could decrease the run-time and increase the performance of the program.

Additionally, one could extract sampling points from a complete map stochasti-

cally using some uniform distribution. Having selected a set of stochastic points, one could perform statistical analysis. The simulation would change into an actual complete Monte Carlo simulation. The expected penalty-time and expected variance of penalty-time would indicate to the user if the selected route is robust or not, in theory. However, the current implementation tests the complete route, provides the user with a histogram over all the penalty-times and visualizes them to the user on the map. This, we believe, is an already great way of showing how robust and stable a chosen route is with respect to unforeseen obstacles occurring along routes.

5.5.11 Fixing the Simulation

The current simulation delivers quite good visual results at a decent runtime, yet, improvements are needed to further improved the overall robustness analysis. Since the current solution is that the path restarts at the beginning, where the previously used path is free up until the break-point and afterwards, it does not need to redo all the computation and can rely a lot on the previously made solution path. The clear benefit is faster runtime for the simulation, which at its core is a problem that is NP-complete. However, some problems occur along the approximation.

For example, if one places a break-point before or after the bridge, the solution is identical and the penalty time is identical for the entire bridge. This is simply not correct since if the break-point if after the bridge, one can assume that the driver has already crossed the bridge and now needs to drive back across the bridge which can be seen as additional penalty time. Since the entire bridge will be given an identical color, our current solution works great to visualize the problem, yet, in some extreme cases it results in a completely false solution if the bridge is extremely long. A clear improvement would be to add this additional penalty time if the current delay is equal to the previous delay and the exact same solution is made, giving a more continuous color-scheme for a problematic area.

Additionally, to better understand the simulation, the user should be provided by a visual representation of how the path-finding program solves the break-point task. In other words, if the program inserts an obstacle somewhere on the map, the obstacle point should be selectable. When the point is selected the alternative route that the program has used should be visualized for the user such that he/she knows how to react given the obstacle. At the current state of the program, these solutions are saved but never visualized to the user.

5.5.12 Improved user-experience

No public testings were conducted to acquire documented feedback about the current user-experience. Although the GUI itself isn't a central part this project, improvements could be made to simplify the usage and extend the functionality. First off, the option to create a route should be implemented since currently it's only possible to edit a route generated by the program. The option to add hostile areas with the use of polygons. Improved warning signs applied to routes if simulation results in very long penalty times. The config-options, where the user inserts weather param-

eters, should be replaced with graphical maps provided by the user with weather forecasting information. Additionally it should be possible for the user to insert map-data to the program directly, as of right now this is done inside the source code and visualized in Carmenta Engine.

5.6 Real-Life Testing

A great way to verify or at least get an understanding of the purposed route by the path-planning program, is to perform real-life testing. The routes that travel through the OSM-layer are very likely to be reasonable, although, the off road routes are not easy to understand. The physical and non-physical robustness properties are not validated with real-life experiments either, which would be important moving forward. The assumption of the uncertainty in parameters that are assumed to be normal distributed is more of a conceptual approach than anything real, however, real-life errors would potentially improve the understanding of the robustness-index. By errors in parameters the authors is referring to the physical and non-physical robustness proprieties that are sensitive to parameters such as soil moisture content and temperature.

The downside of real-life testing is the potential economical aspect, where it could become costly to try and perform a relocation using mainly this path-finding program. Testing the physical properties should be done experimentally with labs and not by actually driving off road when bad weather conditions are present. A more cost-efficient approach would be to have an experienced user try the program and based on previous relocation experience using other path-finding algorithms the user can provide essential feedback. If the feedback is positive then there is some truth to the current programs ability to locate and analyse a route. If the feedback is negative the experienced user might have additional, not yet studied in this project, aspects of robustness that needs to be added into the program.

5.7 Terrain Approximation and Contraction Hierarchies

The traversal time of a terrain segment can be pre-computed for a given set of points. Instead of traversing the segment for each shortest path query the travel time can instead be found in a table. This can drastically reduce the number of nodes to be traversed. An example of this can be seen in Fig. 5.1 where the terrain area has been approximated for some points along the border. This is a similar type of simplification to contraction hierarchies where some parts of graph are precomputed [55].

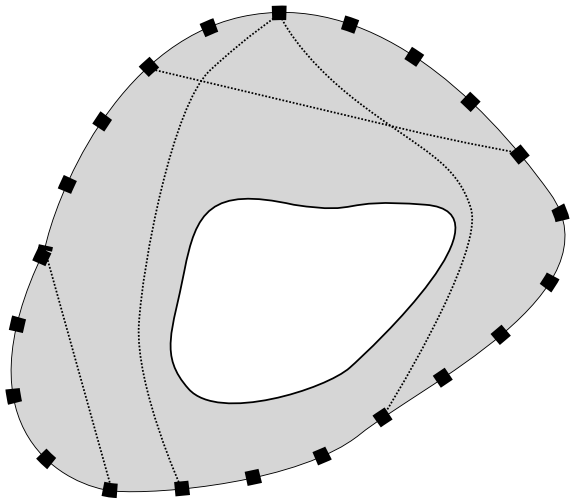

Fig. 5.1: Method of approximating terrain traversal times. The search time for traversing internal points of a traversable area, such as terrain, can be removed. When finding a route between two edges of the area the visited nodes in between can be pre-computed. This saves memory usage, since fewer nodes needs to be stored in memory, but also compution time. This also leaves the option of performing a higher resolution between edges to give a more accurate traversal time.

6

Conclusion

The results produced from this project are valid from a conceptual perspective. The authors have been able to provide detailed and reasonably well-performing algorithms that indicate some ability to generate alternative routes and deliver some kind of understanding of robustness. The theoretical methods constructed seem to work well within the framework of this project and are shown to produce reasonable results. Whether these artificial results are accurate for real-life scenarios is yet to be tested, however, conceptually they work well enough as the first proof of concept. It has been somewhat proven theoretically that it is possible to provide a sense of robustness for a specific route and also in most cases generate unique alternatives.

Bibliography

[1] OpenStreetMap contributors; 2020. https://www.openstreetmap.org.

[2] Lantmäteriet; 2020. https://www.lantmateriet.se/.

[3] Groenendyk D, Ferré T, Thorp K, Rice A. Hydrologic-Process-Based Soil Texture Classifications for Improved Visualization of Landscape Function. PloS one. 2015 06;10:e0131299.

[4] Vricon; 2020. https://www.vricon.com/.

[5] Wong JY. Theory of ground vehicles. Wiley; 2008. Available from: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=cat07470a&AN=clc.01dab8b432e5470fbc7278b3326ef6a2&site=eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[6] Carmenta. Carmenta Engine 5.13 released; 2020. Accessed 2020-01-31. Available from: https://carmenta.com/en/news/carmenta-engine-5-13-released/.

[7] Saab. The tool for comprehensive operational planning. Saab; 2020. Accessed 2020-01-31. Available from: https://saab.com/land/c4i/tactical-c2-systems/cbr-commanders-briefing-tool/.

[8] Vricon. DTM GLOBAL BARE EARTH MODEL WITH 0.5M RESOLUTION; 2017. Accessed 2020-01-31. Available from: https://www.vricon.com/wp-content/uploads/2017/06/Vricon_DTM_FINAL.pdf.

[9] Vricon. DSM GLOBAL ELEVATION MODEL WITH 0.5M RESOLUTION; 2017. Accessed 2020-01-31. Available from: https://www.vricon.com/wp-content/uploads/2017/06/Vricon_DSM_print.pdf.

[10] Tramz M. Beyond Google Earth, How One Company Wants to See the World in 3D. Time; 2015. Available from: https://time.com/3902025/vricon-3d-mapping/.

[11] Vricon. Company Overview;. Accessed 2020-01-31. Available from: https://www.vricon.com/company/overview/.

[12] Isaksen FA, Brusevold KN. Mission Planning with HighResolution Geographical Data. Østfold University College; 2018.

[13] Nilsson M. Building Reconstruction of Digital Height Models with the Markov Chain Monte Carlo Method. Linköping University; 2018. Accessed 2020-01-31. Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-148886.

[14] Larsson K. Deployment Planning for Artillery Hunting Radar Systems Using High-Resolution Geospatial Data. 2018;Accessed 2020-01-31. Available from: https://hdl.handle.net/20.500.12380/256419.

[15] Johansson E, Karlsson S. Deployment of air defense. Chalmers University of Technology; 2019. Accessed online: August 31, 2020, https://hdl.handle.net/20.500.12380/300448.

[16] Wikipedia contributors. Expected value — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 29-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Expected_value&oldid=959196308.

[17] Wikipedia contributors. Variance — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 29-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Variance&oldid=959037389.

[18] Wikipedia contributors. Normal distribution — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 29-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Normal_distribution&oldid=959529084.

[19] Wikipedia contributors. Multi-objective optimization — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 19-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Multi-objective_optimization&oldid=950036949.

[20] Wikipedia contributors. Pareto efficiency — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 29-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Pareto_efficiency&oldid=959469260.

[21] Wikipedia contributors. Dijkstra's algorithm — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 30-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Dijkstra%27s_algorithm&oldid=957555961.

[22] Larkin DH, Sen S, Tarjan RE. A Back-to-Basics Empirical Study of Priority Queues; 2014.

[23] Wikipedia contributors. Monte Carlo method — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 25-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Monte_Carlo_method&oldid=957705820.

[24] Wikipedia contributors. Intersection (Euclidean geometry) — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 23-April-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Intersection_(Euclidean_geometry)&oldid=945630805.

[25] Wikipedia contributors. Line–line intersection — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 25-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Line%E2%80%93line_intersection&oldid=954429641.

[26] Wikipedia contributors. Classical mechanics — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 27-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Classical_mechanics&oldid=954201841.

[27] Wikipedia contributors. Cartesian coordinate system — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 27-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Cartesian_coordinate_system&oldid=956433671.

[28] Wikipedia contributors. Energy — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 27-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Energy&oldid=955430906.

[29] Wikipedia contributors. Kinetic energy — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 27-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Kinetic_energy&oldid=955637612.

[30] Muro T, O'Brien J. Terramechanics : land locomotion mechanics. Balkema; 2004. Available from: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=cat07470a&AN=clc.65ed57dcd6714163bb37619fc177077a&site=eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[31] Labuz ZA J F. Mohr–Coulomb Failure Criterion; 2012. Available from: https://doi.org/10.1007/s00603-012-0281-7.

[32] Das BM. Principles of foundation engineering (6th ed.). Thomson India; 6th edition (July 26, 2006); 2007.

[33] Oloo S, Fredlund DG, Gan J. Bearing capacity of unpaved roads. Canadian Geotechnical Journal - CAN GEOTECH J. 1997 06;34:398–407.

[34] Wikipedia contributors. Cohesion (geology) — Wikipedia, The Free Encyclopedia; 2017. [Online; accessed 23-March-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Cohesion_(geology)&oldid=794824970.

[35] Geotechdata info. Angle of Friction; updated 2013. [Online; accessed 23-March-2020]. Available from: http://geotechdata.info/parameter/angle-of-friction.html.

[36] Tang AM, Pereira JM, Hassen G. Effect of temperature on the shear strength of soils and soil/structure interface. Canadian Geotechnical Journal. 2016 01;53.

[37] Dong Y, Wu Y, Yin J, Wang Y, Gou S. Investigation of Soil Shear-Strength Parameters and Prediction of the Collapse of Gully Walls in the Black Soil Region of Northeastern China. Physical Geography. 2013 05;32:161–178.

[38] Zhang Y, Zhong X, Lin J, Zhao D, Jiang F, Wang MK, et al. Effects of fractal dimension and water content on the shear strength of red soil in the hilly granitic region of southern China. Geomorphology. 2020;351. Available from: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=edselp&AN=S0169555X19304477&site=eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[39] Al-Obaydi M. Effect of Moisture Content on Shear Strength parameters of Gypseous Soil o. 1999 04;6:60–71.

[40] Andersland Orlando B, Ladanyi B. Frozen Ground Engineering (2nd Edition). John Wiley and Sons; 2017. Available from: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=edsknv&AN=edsknv.kpFGEE000S&site=eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[41] Xu X, Wang B, Fan C, Zhang W. Strength and deformation characteristics of silty clay under frozen and unfrozen states. Cold Regions Science and Technology. 2020;172. Available from: http://search.ebscohost.com/login.aspx?direct=true&AuthType=sso&db=edselp&AN=S0165232X19305981&site=eds-live&scope=site&custid=s3911979&authtype=sso&group=main&profile=eds.

[42] Health I, Association S. Best Practices for Building and Working Safely on Ice Covers in Ontario; 2014. [Online; accessed 31-March-2020]. Available from: http://www.ihsa.ca/PDFs/Products/Id/IHSA029.pdf.

[43] Department of Transportation of the Government of the Northwest Territories. GUIDELINES FOR SAFE ICE CONSTRUCTION 2015; 2015. [Online; accessed 31-March-2020]. Available from: https://www.inf.gov.nt.ca/sites/inf/files/resources/0016-001_norex_ice_road_constr._web.pdf.

[44] Porvinsky T, Pentek T, Bosner A, Stankic I. In: Ecoefficient Timber Forwarding on Lowland Soft Soils; 2012. .

[45] Wikipedia contributors. Water balance — Wikipedia, The Free Encyclopedia; 2019. [Online; accessed 24-March-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Water_balance&oldid=926467677.

[46] Wikipedia contributors. Black-body radiation — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 26-March-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Black-body_radiation&oldid=947080100.

[47] Enkhbayar Dandar, Maarten W Saaltink, Jesús Carrera, Buyankhishig Nemer. A surface model for water and energy balance in cold regions accounting for vapor diffusion; 2017. [Online; accessed 26-March-

2020]. Available from: https://www.hydrol-earth-syst-sci-discuss.net/hess-2016-659/hess-2016-659.pdf.

[48] Abby Frazier and Hla Htun. Water and Energy Balance; 2014. [Online; accessed 26-March-2020]. Available from: https://www.ctahr.hawaii.edu/LittonC/PDFs/680_Discussion_Water.pdf.

[49] Francisco José Del Toro-Guerrero TKSHBR Enrique R Vivoni, Vázquez-González R. Variations in Soil Water Content, Infiltration and Potential Recharge at Three Sites in a Mediterranean Mountainous Region of Baja California, Mexico. 2018;[Online; accessed 7-April-2020]. Available from: https://www.mdpi.com/2073-4441/10/12/1844.

[50] SMHI. Vattenbalans; 2020. [Online; accessed 7-April-2020]. Available from: https://www.smhi.se/vadret/vadret-i-sverige/vattenbalans.

[51] for Photogrammetry & Remote Sensing TAS. LAS SPECIFICATION VERSION 1.4 – R13 15 July 2013; 2013. [Online; accessed 30-May-2020]. Available from: https://www.asprs.org/wp-content/uploads/2010/12/LAS_1_4_r13.pdf.

[52] Kobitzsch MH. Alternative Route Techniques and their Applications to the Stochastics on-time Arrival Problem; 2015.

[53] Karney CFF. Algorithms for geodesics. Journal of Geodesy. 2013 Jan;87(1):43–55. Available from: https://doi.org/10.1007/s00190-012-0578-z.

[54] Wikipedia contributors. Hausdorff distance — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 30-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Hausdorff_distance&oldid=952461286.

[55] Wikipedia contributors. Contraction hierarchies — Wikipedia, The Free Encyclopedia; 2020. [Online; accessed 31-May-2020]. Available from: https://en.wikipedia.org/w/index.php?title=Contraction_hierarchies&oldid=953487028.