NTNU – Trondheim
Norwegian University of
Science and Technology

Hierarchical Path Planning for Ground Vehicles

Marius Thoresen

Master of Science in Cybernetics and Robotics

Submission date:	July 2015
Supervisor:	Oddvar Hallingstad, ITK
Co-supervisor:	Solveig Bruvoll, FFI

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Path planning for autonomous terrain vehicles is performed with terrain data of different levels of detail. Some of the data are prior knowledge with coarse resolution, while some are sensed data of the surroundings with finer resolutions. The vehicle always needs a path in the most detailed terrain available to be able to traverse the terrain. Performing complete path planning using the most detailed terrain data is often not possible as detailed terrain data only becomes available as the vehicle moves, or that the data sets are too large for practical computations.

Performing path planning using the coarse terrain data can provide a basis for performing path planning using the higher detail terrain, which can improve performance of calculations. The combination of the path planning in different terrain is a type of hierarchical path planning.

In this report, a method for hierarchical path planning using terrain data is presented. Using terrain data in different resolutions, a hierarchy of terrain data is created, with the coarsest data being the highest level, and the finest data being the lowest level in the hierarchy. In the hierarchy, path planning is performed in the highest level, and the result is used to reduce the data of the lower levels before performing path planning.

Experiments using real terrain data are performed for comparing the hierarchical path planning to conventional path planning, both with respect to computational efficiency and also optimality and similarity between the paths. Additional experiments are performed for investigating possible improvements to the method presented.

The results show that hierarchical path planning can improve computation efficiency at the expense of optimality of the paths. In the experiments, as much as 5 times increase in computation efficiency were achieved on average, depending on the parameters chosen. The hierarchical paths are in most cases close to optimal, but large deviations does occur. The optimality is also dependent on the parameters chosen, and there is a trade-off between computational efficiency and optimality.

Sammendrag

Ruteplanlegging for autonome terrengkjøretøy blir utført med terrengdata med ulikt detaljnivå. Noe av dataen er forhåndskjent data med grov oppløsning, mens noe av dataen er sanset data fra de umiddelbare omgivelsene med finere oppløsning. Kjøretøyet trenger alltid en rute for de mest detaljerte terrengdataene som er kjent for å kunne bevege seg gjennom terrenget. Å utføre en komplett ruteplanlegging med de mest detaljerte terrengdataene som finnes er ofte ikke mulig fordi detaljert terrengdata kun blir tilgjengelig når kjøretøyet beveger seg, eller at de detaljerte terrengdataene er for store til praktiske beregninger.

Å utføre ruteplanlegging ved å bruke den grove terrengdataen kan gi en basis for å utføre ruteplanlegging for den fine terrengdataen. Kombinasjonen av ruteplanlegging i ulike terrengtyper er en form for hierarkisk ruteplanlegging.

I denne oppgaven blir en metode for å utføre hierarkisk ruteplanlegging presentert. Ved å bruke terrengdata av ulike oppløsninger, et hierarki av terrengdata blir laget, der den groveste terrengdataen er øverst i hierarkiet og den fineste dataen er nederst. I hierarkiet blir ruteplanlegging utført i det høyeste nivået, og ruten som blir funnet brukes til å velge ut et nabolag rundt denne ruten i nivået under. Ruteplanlegging blir deretter utført i nivået under, innenfor det reduserte nabolaget for å finne en detaljert rute.

Eksperimenter som bruker virkelige terrengdata er utført for å sammenligne den hierarkiske urteplanleggingen med konvensjonell ruteplanlegging, både med hensyn på beregningseffektivitet og optimalitet og likhet mellom rutene. I tillegg er det utført eksperimenter for å undersøke mulige forbedringer av metoden som er presentert.

Resultatene viser at den hierarkiske ruteplanleggingen can øke redusere beregningstiden, men på bekostning av optimalitet for rutene. I eksperimentene er så mye som 5 ganger økning i effektivitet oppnådd i gjennomsnitt, avhengig av parametre valgt for beregningene. De hierarkiske rutene er i de fleste tilfeller nær optimale, men store avvik oppstår i noen av tilfellene. Optimaliteten avhenger også av parametrene valgt, og det er en avveining mellom effektive beregninger og optimalitet.

Preface

This master's thesis concludes my M.Sc. studies in Engineering Cybernetics at Norwegian University of Science and Technology (NTNU). The work has been performed in Trondheim during the spring 2015.

I would like to thank my supervisors at FFI, Solveig Bruvoll and Vegard Kvernelv. They have been of great help in working with this thesis, and have provided valuable guidance and feedback all throughout the process. I would also like to thank my main supervisor at NTNU and UNIK, Professor Oddvar Hallingstad for this great opportunity of working with FFI on my thesis.

Trondheim, July 2015
Marius Thoresen

Table of Contents

Abstract **i**

Sammendrag **iii**

Preface **v**

Table of Contents **ix**

List of Figures **xiii**

1 Introduction **1**
- 1.1 Problem description . . . 1
- 1.2 Report structure . . . 2

2 Path planning **5**
- 2.1 Path planning theory . . . 5
  - 2.1.1 Terrain representation . . . 6
  - 2.1.2 Path planning algorithms . . . 7
- 2.2 A path planning framework . . . 8
  - 2.2.1 Terrain analysis . . . 9
  - 2.2.2 Creating a graph from terrain . . . 13
  - 2.2.3 Path planning using A* . . . 17

3 Hierarchical path planning **19**
- 3.1 Previous work . . . 20
- 3.2 Hierarchical path planning in terrain . . . 22
  - 3.2.1 Proposal for new method . . . 22
  - 3.2.2 Method evaluation . . . 23
  - 3.2.3 Extensions of the method . . . 24

4 Path analysis **25**
- 4.1 Path cost . . . 25
- 4.2 Computational cost . . . 26

4.3 Path distance metric 26
4.3.1 Area calculations for specific graph types 28

5 Experiment setup **33**
5.1 Hierarchical path planning framework 33
5.2 Terrain analysis with real data 34
5.2.1 Terrain types 34
5.2.2 Elevation 40
5.3 Path planning implementations 46

6 Experiments **49**
6.1 Experiment 1: Path planning in different terrain resolutions 50
6.1.1 First trial: 1 m, 2 m and 10 m resolutions 51
6.2 Experiment 2: Hierarchical path planning 58
6.2.1 Experiment setup 58
6.2.2 Hierarchical paths using 2 m terrain 60
6.2.3 Hierarchical paths using 10 m terrain 62

7 Extended experiments **69**
7.1 Experiment 3: Hierarchical path planning in flat terrain 69
7.1.1 Experiment setup 70
7.1.2 Distance between paths 71
7.1.3 Relative costs 72
7.2 Experiment 4: Comparison of 4 edge and 8 edge graphs 74
7.2.1 2 m terrain 74
7.3 Experiment 5: Multiple level hierarchy 80
7.3.1 Combining 10 m, 2 m and 1 m terrains 80
7.3.2 Results 81

8 Discussion **85**
8.1 Hierarchical path planning 85
8.1.1 2 m hierarchical path planning 85
8.1.2 10 m hierarchical path planning 86
8.1.3 Infeasible hierarchical paths 86
8.1.4 Overall performance of the hierarchical path planning 87
8.2 Improved hierarchical path planning 88
8.2.1 Flat terrain 88
8.2.2 8 edge graphs 89
8.2.3 Multiple level hierarchy 90
8.3 Summary 91

9 Conclusion **93**
9.1 Further work 94

Bibliography **95**

Appendix **97**

A Greens theorem in area calculations **99**

A.1 Green's theorem 99

A.2 Greens area formula 99

A.3 Line integral for straight edges 100

A.4 Line integral for diagonal edges 101

B MATLAB implementations **103**

B.1 Terrain analysis 103

B.2 A* search algorithm 105

B.3 Path evaluation 107

B.4 Hierarchical path planning implementation 109

List of Figures

2.1 General undirected weighted graph. Illustration taken from [13]. 7
2.2 Geometry of the slope and the gradient 12
2.3 Graph representation of grid structure with 4 edges per vertex 13
2.4 Another graph representation of 4 grid structure 15
2.5 Distances in 4 grid paths. 16
2.6 Distances in 8 grid paths. 16

3.1 Illustration of graphs in different hierarchy levels for houses. Illustration taken from [13]. 19

4.1 Area between two intersecting paths 28
4.2 Path intersection in the wrong order 29
4.3 Edges of a 4 edge graph 29
4.4 Diagonal edges 30

5.1 A sample showing terrain types with 1 m resolution 36
5.2 Problems with naive downsampling of terrain 38
5.3 Downsampling from 1 m to 10 m 39
5.4 Height data with 1 m resolution 41
5.5 Slopes in 1 m resolution 42
5.6 Slopes 1 m resolution, quantized into 3 levels. Blue: $\theta < 15°$, Yellow: $15° \leq \theta < 25°$ and red $\theta \geq 25°$ 42
5.7 Slopes 10 m resolution, quantized into 3 levels. Blue: $\theta < 15°$, Yellow: $15° \leq \theta < 25°$ and red $\theta \geq 25°$ 43
5.8 Detailed area taken from figures 5.6 and 5.7. Narrow details in (a) have dissappeared entirely in (b), while the larger features at the bottom are preserved. 44
5.9 Detailed area taken from figures 5.6 and 5.7. Most features are preserved, but there is a tendency of areas that are red in (a) appear as yellow in (b). . 45
5.10 Terrain types for the same area as previously 46
5.11 Terrain and slopes data combined into speed map 47

6.1 Full 26 km by 26 km terrain 50

6.2 Close to identical optimal paths in 1 m, 2 m, and 10 m terrain resolutions. 52
6.3 Optimal paths in 1 m terrain and 10 m terrain, with large distance between the paths 52
6.4 Computation times of optimal paths in 1 m, 2 m and 10 m resolutions. $N = 30$ cases. 53
6.5 Relative computation times of optimal paths in 2 m and 10 m resolutions, relative to the computation time of the 1 m optimal paths. $N = 30$ cases. . 54
6.6 Distance between 1 m optimal path and 2 m and 10 m optimal paths. N = 30 cases. 55
6.7 Costs of optimal paths in 1 m, 2 m and 10 m 56
6.8 Relative cost of 2 m and 10 m optimal paths, relative to 1 m optimal path . 57
6.9 Detailed view of relative cost of 2 m optimal paths, relative to 1 m optimal path 57
6.10 Creating graphs in fixed radius neighborhood of graphs 59
6.11 Cost of 1 m optimal path and 2 m hierarchical paths. Three outliers are removed from the 10 m radius paths, as no path were found in these cases 61
6.12 Relative costs of 2 m hierarchical paths, compared to 1 m optimal path cost. Three infeasible outliers removed from the 10 m radius path. 61
6.13 Relative costs of 2 m hierarchical paths, with outlier removed from 50 m and 100 m radius paths. and 3 outliers removed from the 10 m radius path 62
6.14 Computation time of 2 m hierarchical paths and the 2 m optimal path . . . 63
6.15 Relative computation time of the the 2 m hierarchical paths compared to the 2 m optimal path 63
6.16 Average distance between 2 m hiearchical paths, and 1 m optimal paths . 64
6.17 Cost of 1 m optimal path and 10 m hierarchical paths. 1 infeasible path removed from 100 m path, and 6 infeasible paths removed from 50 m path 65
6.18 Relative cost of 10 m hierarchical paths and 1 m optimal path. 1 infeasible path removed from 100 m path, and 6 infeasible paths removed from 50 m path. 65
6.19 Computation time of 2 m hierarchical paths and the 2 m optimal path . . . 66
6.20 Relative computation time of 10 m hierarchical paths compared to the 2 m optimal path 67
6.21 Average distance between 10 m hiearchical paths, and 1 m optimal paths . 67

7.1 Slopes in a flat terrain selection 70
7.2 Flat terrain: all distances 71
7.3 Flat terrain: distances with outlier removed 72
7.4 Flat terrain: Costs of the 10 m optimal path, and 1 m hierarchical path, relative to the 1 m optimal path 73
7.5 Computation times of 1 m optimal paths and 2 m hierarchical paths, 4 and 8 edge graphs. 74
7.6 Relative computation time of 2 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs. 75
7.7 Average distance between 1 m optimal paths and 2 m hierarchical paths, 4 and 8 edge graphs 76

7.8 Relative costs of 2 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs. 76
7.9 Computation times of 1 m optimal paths and 10 m hierarchical paths, 4 and 8 edge graphs. 77
7.10 Relative computation times in 10 m terrain, 4 and 8 edge graphs. 78
7.11 Average distance between 1 m optimal paths, and 10 m hierarchical paths, 4 and 8 edge graphs 78
7.12 Relative costs of 10 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs. 79
7.13 Computation time of multiple level paths and 10 m hierarchy paths. Single (S) denotes the 10 m hierarchical path, and multiple (M) denotes the paths multiple level paths. 82
7.14 Relative computation time of multiple level paths compared to single 10 m hierarchy paths. 82
7.15 Relative cost of multiple level paths compared to 10 m hierarchical paths . 83

A.1 An area R with boundary $\mathcal{C}$ 100

Chapter 1

Introduction

1.1 Problem description

Unmanned vehicles are seeing increased interest, and several types of autonomous vehicles has been made possible by advances in autonomy research. A type of unmanned vehicles is vehicles capable of traversing terrain outside. They have many potential uses, especially when they can be made autonomous. Mine clearing, reconnaissance and transportation is some examples of uses where unmanned vehicles can be of benefit.

For unmanned terrain vehicles to become autonomous, they must have the ability to automatically plan paths through terrain based on the available terrain information. The path planning consists of two processes. The first is using available raw terrain information to create a suitable terrain representation for the path planning purposes. Secondly, with a well defined terrain representation, the best path trough the terrain can be found using search algorithms with the terrain representation.

The initial path planning is be done using prior knowledge of the terrain. The terrain data available does not contain detailed information about the immediate surroundings of the vehicle, but rather a coarse terrain data over larger areas. Path planning in this terrain yields a global path to a desired target. When finer and more detailed terrain data becomes available, a new detailed path needs to be found. The fine terrain data may be real-time data gathered as the vehicle moves, or may be available initially, but are too large to efficiently be used for path planning.

The terrain representations with different levels of detail is a type of hierarchy where the coarse terrain is the high level, and the finer terrain are the lower levels in the hierarchy. A path planning procedure first finding a path in the a level, and using this as a basis for finding a detailed path in a lower level is a type of hierarchical path planning.

Using the such a hierarchical path planning rather than using a detailed terrain representation, can give increases in path planning performance, as the path planning in a sparsely sampled terrain is faster than a more detailed and more densely sampled terrain. The objective of this thesis is be to find and implement a hierarchical path planning procedure. To achieve this, the objective can be divided into the following subgoals:

1. Consider existing methods for path planning, and implement a specific method for automatic terrain analysis and path planning.

2. Find a method for performing hierarchical path planning using terrain representations of different levels of detail.

3. Perform experiments using hierarchical path planning, and compare the results to conventional path planning based on a single terrain level.

1.2 Report structure

The structure of the report is described here. The report is divided into the following chapters:

Path planning is divided into two parts. The first part provides a brief review of path planning theory and literature. The second part describes a path planning framework for performing automatic terrain analysis and path planning, and is an original contribution.

Hierarchical path planning is also divided into two parts. The first part is an introduction to the concept of hierarchical path planning, and provides a review of the earlier work on the subject. The second part is a proposal for a new hierarchical path planning method for terrain. A short discussion of possible advantages and disadvantages along with possible improvements is included. This method is the main contribution of this report.

Path analysis describes metrics for comparing and evaluating paths found using hierarchical and conventional path planning. These methods are needed for analyzing experiment results. A new contribution is a metric for distance between paths based on the area between two generic paths.

Experiment setup describes how specific terrain data is used with the framework described in the path planning chapter. This includes parameters for the terrain analysis, and implementation using these.

Experiments is the chapter containing the main results for the hierarchical path planning method. The experiments compare the hierarchical path planning to conventional path planning using the same cases for a direct comparison.

Extended experiments is a chapter containing three additional experiments for further investigating the potential of the hierarchical path planning. The first is an experiment investigating how hierarchical path planning can be made more reliable, and the next two are ideas for improving the performance of the method.

The **discussion** contains a discussion about the results from the experiments and the value of an hierarchical approach for the path planning.

The **conclusion** summarizes the contributions of this report and the results obtained. It also fatures suggestions for future work with hierarchical path planning.

Chapter 2

Path planning

2.1 Path planning theory

Path planning is the process of finding a path between a start point and an target point in a space. For a ground vehicle, this space will be the earth surface. An example of such path planning is the GPS guidance frequently used in personal vehicles, which uses path planning for finding the shortest route from the current location to any destination. With recent development of autonomous road going cars, an system for path planning is an essential part. The path planning in these situations are restricted to roads, but path planning can also be used for planning movement in the terrain. Path planning is also widely used in video games combined with artificial intelligence (AI) for characters to move around in games [13]. Many innovations in path planning are therefore a result of the video game industry.

Performing path planning is key component for any kind of autonomous vehicle, and there is a large number of techniques in use, and it is a field of increasing interest. Path planning can also be of use in manned ground vehicles, as it can aid the operator in finding the best path available. Path planning as vehicle aiding in terrain is widely used in military applications today [4].

Path planning can be divided into two parts, global and local path planning. Global path planning is concerned with finding long range paths, and uses all available terrain data for finding the optimal path. Due to the possibility of large acquired data, the path planning can be a slow process. In general, the path planning is not required to run in real time, but can be run before a mission [8]. Being able to perform global path planning in real time does however add flexibility in case the global path need to be recalculated, for example due to large deviations between a priori terrain data, and the local terrain data acquired by sensors.

The local path planning is the processed of using the sensor data about the environment to create a path in the world as it is sensed. The local path planning will handle tasks like collision avoidance and stability, and will include non-holonomic constraints such as limited movement due to wheel placement [12]. It will also incorporate the path from the global path planning to reach the overall target. For autonomous vehicles, the the local path planning will be the basis for the vehicle control, and is therefore essential to run in real time. For path planning used for aiding drivers in movement, there might not exist a local path planning, and the global path is the only one used for aiding.

Path planning has the ability to find the optimal path between the start and the target points. The first step for finding this is to define what type of situation the path planning is for, and what measure constitutes the optimal path. The objective can be defined from a variety of parameters, or can be a combination of many. The objectives can for example be to find the least time consuming path to traverse which is a simple measure, but useful in many situations. The most fuel saving path or the shortest path in distance are also other general measures that are used. With path planning in terrain, other parameters such as high accessibility and smoothness of path can be interesting as well. With path planning for military purposes, other criteria like avoiding detection, finding cover or finding positions with good overview of an area [18],[4] are also important. The different criteria can also be combined to find paths that have several of these properties.

2.1.1 Terrain representation

In order to use terrain data for path planning, a method for representing the terrain data available into a structure that can be used for finding the optimal path.

Two dimensional graphs

A way of using the terrain data for path planning, is to make the 3 dimensional terrain in the form of a 2 dimensional graph where the terrain is projected onto the horizontal plane. The graph consists of a set of vertices, and weighted edges between the vertices. An illustration of a general weighted undirected graph can be seen in figure 2.1. The vertices represents physical locations, and the edges represents possible movement between two vertices. Only movement along edges are permitted, and the cost of moving along an edge is given by the edge weight [13]. The world coordinates of each point is given as vertex properties. As the graph is a projection of a 3 dimensional sphere onto a 2 dimensional plane, distortion occurs, and the length of the graph edges are not exactly equal to the real world distance. The world coordinates associated with the edges should be used for distance calculations in the graph for accurate results [3].

There are several ways of representing the terrain as a graph structure, but a commonly used and simple method is cell decomposition. The world is in these methods divided into a set of representing areas where each cell contains the characteristics about the corresponding real world area. The commonly used method is to divide the world into a grid of quadratic cells, where each cell will contain terrain data for the specific cell. This graph structure is also known as a tile graph. This information can for example be elevation, terrain type, roughness and accessibility [8],[3]. Depending on the characteristics,

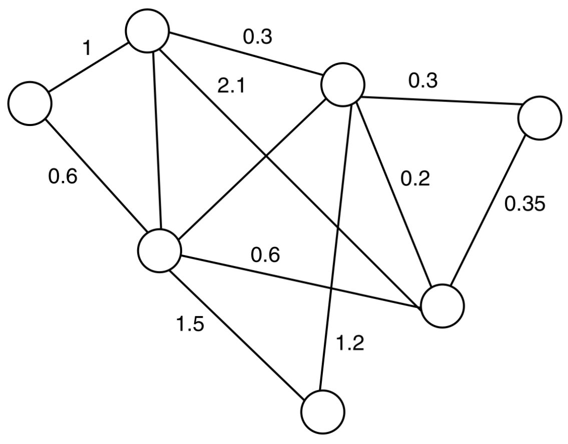


Figure 2.1: General undirected weighted graph. Illustration taken from [13].

some cells are traversable, while others are treated as obstacles. The tile graph is often a simple structure to create, as terrain data often are given in the format of a grid of cells. Other structures than the tile graphs can also be used, like for example quad trees which separates between areas of high and low details, allowing more efficient storage [16].

Other major methods for representing the world are roadmap methods. Roadmap methods finds keypoints in the world, and describes how to move between these and creates a complete terrain representation. Some roadmap methods include visibility graphs, Voronoi diagrams and probabilistic roadmaps. They will not be considered further in this report, but more information on these methods can be found in [8].

Terrain analysis

With a graph structure in place, the edge weights have to be computed. The edge weights represents the cost of moving between two vertices. The weights therefore have to be set according to the objective of the path planning. For the fastest path, the weights are based on time consumption for traversing an edge, and is found using the distance between the edges and the speed that can be held. For short distances and high speed, the weight is set low, and for long distances and low speeds, the weight is set high. The speed depend on the terrain data, with for example roads and open fields allow high speed, while forest forces the speed down. The slopes can also be used for the speed calculations, as flatter terrain allows higher speeds, while steep terrain is inaccessible. Inaccessible terrain can be treated by setting the speed to 0 and setting an infinitely large edge weight [3].

2.1.2 Path planning algorithms

With the terrain represented as graphs, the path planning can be performed with shortest path algorithms from mathematical graph theory. A common algorithm for the general shortest path problem is Dijkstra's algorithm from 1959 [7]. Dijkstra's algorithm is both a complete and an optimal algorithm, meaning that it is guaranteed to find a solution if one exists, and the solution found is guaranteed to be the optimal solution [6]. A problem with Dijkstra's algorithm is that it does not only find the path between the start vertex and the end vertex, in the process it also finds the shortest path to all other vertices. As the other paths aren't interesting for the path planning purpose, it is wasteful because it performs more calculations than is necessary for the path planning [13].

A* algorithm

The A* is a generalization of Dijkstra's algorithm designed for finding paths between two nodes, and not the general shortest path problem from graph theory. The A* algorithm and algorithms based on it are the most used algorithms in the game industry and in robotics [13],[8]. The A* algorithm uses a heuristic function for predicting which vertices that gives the shortest path to the target vertex. The heuristic speeds up the search, but depending on which heuristic chosen, the algorithm may return a suboptimal path [13]. The A* algorithm is always complete, and it is optimal in cases where the heuristic is chosen so that it never overestimates the remaining cost between any vertex and the target [10]. Choosing a heuristic that finds potentially suboptimal paths can make the A* path planning quicker to perform, and it is therefore a trade-off between computation speed and optimality of the paths returned.

2.2 A path planning framework

In this report, a framework for performing path planning using terrain data will be implemented. Only global path planning using a priori terrain data will be considered. No real time sensor data will be used, and the method implemented is not supposed to run in real time, and there are therefore no hard computation limits on the path planning. A result of this is that large problem sizes and long computation times can be acceptable for the purposes of experimentation.

The framework will consist of three main parts:

Terrain analysis A form of terrain analysis needs to be implemented. This will take raw terrain data and make useful information for the required tasks of the terrain vehicle in question. The terrain analysis will be different depending on the goals for the path planning.

Graph generation The terrain will be represented as a graph structure. Path planning can therefore be performed with shortest path algorithms for graphs. The weight of the edges will be set based on the terrain information gathered through the terrain analysis.

Path planning The path planning itself will be performed using a standard A* implementation. A* has good performance, and can be designed for returning the optimal path.

The three parts are in principle independent from each other, but the A* algorithm will benefit from choosing a heuristic according to the graph structure and terrain. The effectiveness of the A* algorithm will therefore depend upon the choice of the graph and the heuristic choice.

2.2.1 Terrain analysis

Terrain analysis provides a way of converting raw terrain data into information that can be utilized for path planning. The terrain analysis will have to be based on a goal for the overall path planning in order to extract the necessary information.

A simple goal for performing path planning is to find the least time consuming path to traverse, the fastest path. This is a good choice as it is a goal that is a applicable to a wide range of vehicles in different situations. It also features simple computations compared to some of the more advanced goals, such as maintaining cover while moving. The terrain analysis will therefore have the goal of determining speed based on the terrain.

The speed a vehicle can maintain depends on several factors. Most important is the vehicle in question. Different sizes, different power outputs and different traction varies greatly depending on the vehicle type. For the purposes of these experiments, the terrain analysis is based on how a single imagined vehicle could perform. This vehicle can be compared to an ATV and has somewhat similar properties. ATVs has good mobility in a variety of terrain types. On roads and in open terrain they can move relatively fast, and also have the ability to traverse rougher terrain with reduced speeds. They are therefore a good choice for path planning experiments intended for repeatability.

The terrain data consist of two types of data, elevation data and terrain types. The data is sampled from the real world in an evenly spaced grid pattern. The data can be viewed as a projection into a 2D plane as viewed from above. It is assumed that the elevation and terrain types data are of same resolution, and if this is not available, this must be obtained by first.

Terrain types

The terrain types data contains information about the terrain category of each cell in the terrain grid. Different terrain types allows different speeds while some terrain types are inaccessible altogether. The speeds is therefore chosen to be equal for all cells Choosing the speeds needs to be done manually depending on the situation and the type of vehicle.

Elevation data

With the terrain types data, elevation data is also be used for the terrain analysis. The data contains the elevation above sea level for individual points in the terrain, organized as cells in a grid in the same manner as with the terrain types. With accurate elevation measurements, slopes can also be calculated for the terrain which is utilized for path planning.

Movement with elevation

With the addition of the elevation data, the speed model derived from the terrain data can be modified to take the elevation profile into account. There are two main ways the elevation data affects how a terrain vehicle moves through the terrain:

Increased distances With elevation differences between cells, the distance between is not equal to the horizontal cell size, but is a function of the height difference as well.

Steep slopes With large slopes, terrain vehicles are not be able to traverse the terrain, or might only do so a reduced speeds.

Both of these effects are included in the terrain analysis performed.

Modified cell distance

With an elevation difference between cells, the real distance between the cells can be calculated using the Pythagorean theorem. With two adjacent cells with u, and v, with elevation $h(u)$ and $h(v)$, the elevation difference is

$$\Delta h = h(u) - h(v) \tag{2.1}$$

With a spatial terrain resolution d_0, the distance between two adjacent cells u and v is

$$d(u, v) = \sqrt{d_0^2 + (\Delta h)^2} \tag{2.2}$$

For small slopes, the elevation difference Δh is relatively small compared to the terrain resolution, and the approximation

$$d(u, v) \approx d_0 \tag{2.3}$$

can be used.

The terrain resolution d_0 is in reality not the true distance between the cells, as the cells are obtained from a projection of the spherical real world terrain onto a 2 dimensional terrain representation. The elevation of the terrain is accounted for, but the projection causes a degree of distortion. For calculating distances accurately, the coordinates of each cell should be used. For smaller distances however, this difference is small, and the nominal cell size d_0 is used as an approximation.

Go	Vehicle can maintain regular terrain speed
Go slow	Vehicle can move at a slow speed if the terrain type is feasible
No go	The terrain type is infeasible

Table 2.1: The 3 types of regions in different slopes

Slopes and inaccessible cells

In addition to the increased distance, slopes also affects if ground vehicles are able to move across the terrain itself. With too large slopes, vehicles are not able to traverse the terrain. This is simply due because the traction is not good enough for the slopes. Also, terrain vehicles can in some cases move through terrain, but are only able to do so with a reduced speed if the slopes are not too large. With small slopes, it can be assumed that the vehicle can traverse the terrain with the regular speed given by the terrain type.

This division can be implemented by assigning each cell into one of three possible categories, depending on the slope of the cell. These categories are shown in table 2.1.

The slope limits for the Go-slow and No-go regions can vary with the terrain type, as different terrain gives different traction. In a real situation, the roughness of terrain can also affect the overall traction, as a smooth surface with a certain traction can be easier to traverse than a rough surface with the same traction.What kind of vehicle that is used also affects the results.

These factors makes the problem a complicated with many rules for calculating which speeds are maintainable for each terrain type. In addition, the speed a vehicle can go at slow speed can also vary with the terrain. To reduce the complexity of the problem, two assumptions are made.

1. The slope limits for the Go-slow and No-go regions are the same for all terrain types.
2. The reduced speed in the Go-slow region are the same for all terrain types.

In many situations, the given terrain vehicle would have the ability of increasing the performance above the assumptions, but the limits are set so there is a margin in certain situations. This ensures that the paths found would be feasible in a real situation, even though higher speeds and mobility could be reached.

The method for determining the speed in a cell can be summarized as follows: If the terrain speed of a cell is given as $V_0(u)$, and the slow speed is given as V_{slow} the speed of a cell $V(u)$ with a slope θ is given as

$$V(u) = \begin{cases} 0 & \text{No go} \\ \min(V_0(u), V_{slow}) & \text{Go slow} \\ V_0(u) & \text{Go} \end{cases} \tag{2.4}$$

Calculating the slope of a cell

For calculating the overall slope at a point, the gradient can be used. The direction of the gradient vector $\nabla h(x,y)$ gives the direction of the maximum increase of a function $h(x,y)$, while the length of the gradient, $|\nabla h(x,y)|$ gives the maximum rate of change of $h(x,y)$. The maximum rate of change gives the size of the overall slope in a point, as seen in figure 2.2.

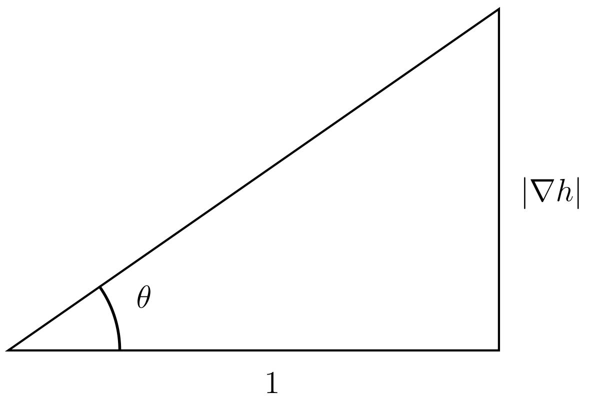


Figure 2.2: Geometry of the slope and the gradient

The gradient of a scalar field h is defined as

$$\nabla h(x,y) = \frac{\partial h}{\partial x}\hat{x} + \frac{\partial h}{\partial y}\hat{y} \tag{2.5}$$

As h is a discretely spaced map with spacing $\Delta x = \Delta y = d_0$, the partial derivatives are the differences in x and y direction, divided by the spacing. Therefore, a specialized variant used for the elevation is

$$\nabla h(x,y) = \frac{\Delta h_x}{d_0}\hat{x} + \frac{\Delta h_y}{d_0}\hat{y} \tag{2.6}$$

The length of the gradient is just the length of the vector

$$\begin{aligned} |\nabla h(x,y)| &= \sqrt{\left(\frac{\Delta h_x}{\Delta x}\right)^2 + \left(\frac{\Delta h_y}{\Delta y}\right)^2} \\ &= \frac{1}{d_0}\sqrt{(\Delta h_x)^2 + (\Delta h_y)^2} \end{aligned} \tag{2.7}$$

From figure 2.2, the relationship between the gradient and the slope angle θ is

$$\tan\theta = |\nabla h| \tag{2.8}$$

Using the method of 2.4, the terrain types data and the elevation data concludes the terrain analysis, and a complete map of the speeds for each cell in the terrain data is obtained.

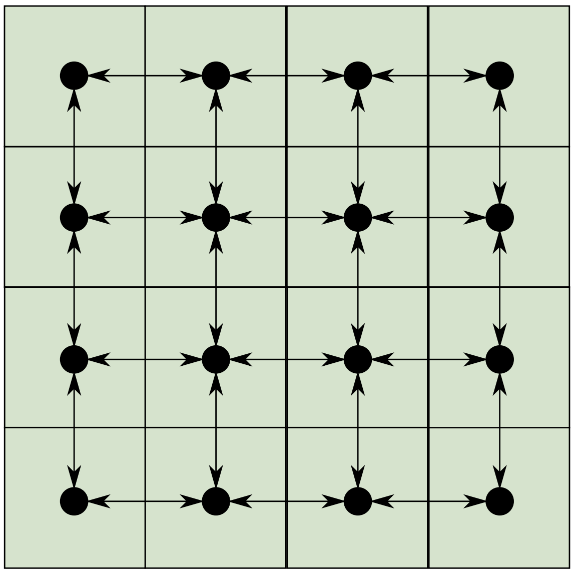


Figure 2.3: Graph representation of grid structure with 4 edges per vertex

2.2.2 Creating a graph from terrain

With a terrain analysis in place that can produce speeds for a terrain, creation of graphs can be done. With the terrain analysis being done with grid structures of data, the simplest type of graph for representing this data is a graph where each cell in the terrain data are represented by a vertex in the graph. From the terrain analysis, the speeds was defined for each cell. In a graph structure, the movement is not tied to the vertex itself, but to the edges between cells. The speed information from the individual must therefore be converted into information for moving between cells instead.

Edge types

The simplest graph variant for a grid graph is a structure where all the vertices have edges to the vertices above, below and to the sides. This means in total 4 edges per vertex, except for the boundary vertices. This type of graph representation is illustrated in figure 2.3. As this is the simplest graph structure, this is the main choice for the path planning framework. Another possiblity would be to include diagonal edges, giving each vertex 8 edges instead of 4 edges. The advantages and disadvantages of both structures is discussed further in section 2.2.2.

Assigning weights

The goal of the path planning determines how the weights are set, and in this case the objective is to find the fastest path between two vertices. Setting the weight of the edges as the time needed to traverse the edges, the cost which is the sum of all weights for a path is the total time taken to traverse an entire path. The path with the lowest cost is be he path taking least time to traverse.

The time needed to traverse a single edge is given by the distance and speed between the vertices.

$$time = \frac{distance}{speed} \tag{2.9}$$

The distance between two vertices is calculated with (2.2). The speeds are however not given between two vertices, but rather for each vertex. The solution is therefore to calculate the time with (2.9) for each of the vertices, and take the average. For two vertices u and v, with the speeds given as $V(u)$ and $V(v)$, and the distance between them given as $d(u, v)$, the weight $w(u, v)$ is calculated as

$$w(u, v) = \frac{1}{2}\left(\frac{d(u, v)}{V(u)} + \frac{d(u, v)}{V(v)}\right) \tag{2.10}$$

The weight is independent of the direction of the edge, the same weight is applied to both the edge going from u to v, $E(u, v)$ and the edge going from v to u, $E(v, u)$. The graph is in reality an undirected graph as all edges can be traversed in both directions with the same weight.

In the cases where the terrain is infeasible and the speed is set to zero, which gives a division by zero. To avoid this, instead of setting the infeasible speeds to exactly zero, the value is set close to zero, for example in the order of magnitude 10^{-12} or even smaller. This means that if one of the vertices is infeasible, the average weight is high, in the order of 10^{12}. The feasible edges has weights around 0.1 to 10, and thus path costs for feasible paths is nowhere near the cost of the infeasible edges.

Comparison of 4 vs 8 edge per vertex

As mentioned earlier, the reason for choosing a four edge per vertex as in figure 2.4a is that this is the simplest structure. Another possibility would be to include diagonal edges, as illustrated in figure 2.4. In the real world, a vehicle would have the ability to move in any direction. By including the diagonal edges, the ability to move in more directions means the paths is closer to movement in the real world. An alternative to this would be smoothing of the 4 edge paths, which is often preferred in practice, but it is not certain that this is as good results.

Distance measurements for 4 and 8 edge graphs

In the real world which allow movement in all directions, the distance between two points $\mathbf{p}_1 = [x_1, y_1]$ and $\mathbf{p}_2 = [x_2, y_2]$ is given by the euclidean distance, the l_2 norm. [12].

$$d(\mathbf{p}_1, \mathbf{p}_2) = ||\mathbf{p}_2 - \mathbf{p}_1||_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{2.11}$$

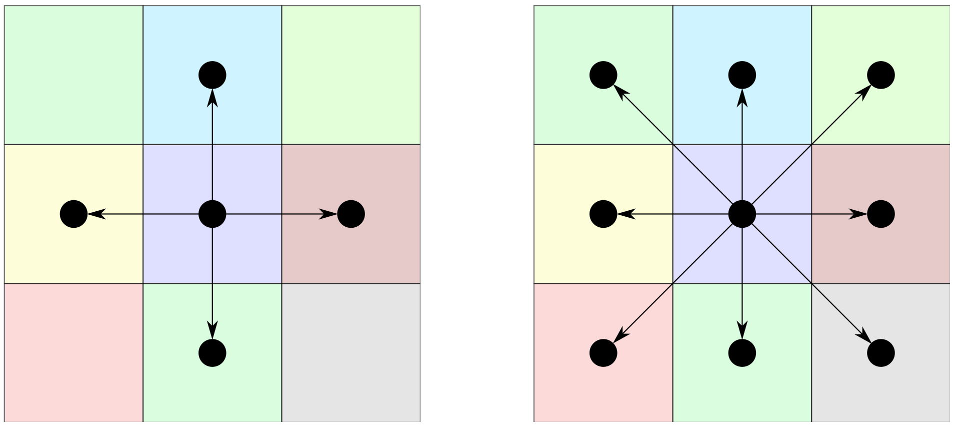

(a) 4 edges per vertex **(b)** 8 edges per vertex

Figure 2.4: Another graph representation of 4 grid structure

With the 4 edge vertices, only movement in north, south, east and west directions are allowed. The distance between two vertices is then given by the manhattan distance, the l_1 norm

$$d(\mathbf{p}_1, \mathbf{p_2}) = \|\mathbf{p_2} - \mathbf{p}_1\|_1 = |x_2 - x_1| + |y_2 - y_1| \tag{2.12}$$

In the case where the vertices are located directly in north-south or east-west directions of each other, the l_1 distance is the same as the l_2 distance, but in other cases, the l_2 distance is always shorter. The largest difference occurs when the vertices are located $45°$ from one of the l_1 edge directions.

For distances in l_2 space, the shortest path between two points is a straight line, and all other paths are necessarily longer. In l_1 space, the shortest path doesn't need to be a straight line, and it does not need to be unique as seen in figure 2.5. In l_2 space, the green path is the optimal path between the red vertices, while in l_1 space, the 3 paths in black are all of equal length, and there doesn't exist any shorter paths. This means that optimality of paths doesn't imply that the paths are close to each other. This is potentially an issue for hierarchical path planning, as it is based on the notion that the optimal paths in one resolution should be in the neighborhood of the optimal path in another neighborhood.

Introducing graphs with diagonal edges in addition to the 4 regular edges reduces this effect. The case from figure 2.5 would allow the green optimal path in l_2 space to be a feasible path, and this would be the unique optimal path. The same issue does however arise when the angle between two vertices is such that it is offset $22.5°$ from any of the existing edge directions. This can be seen in figure 2.6, where several optimal paths between two vertices are shown, in addition to the green optimal path in l_2 space. Due to the increased resolution in directions, the distances between the optimal path in l_2 space, and the optimal paths in the 8 edge graph are relatively smaller compared to the distances in the 4 edge graph. It is therefore more likely that the optimal paths in the 8 edge graphs is

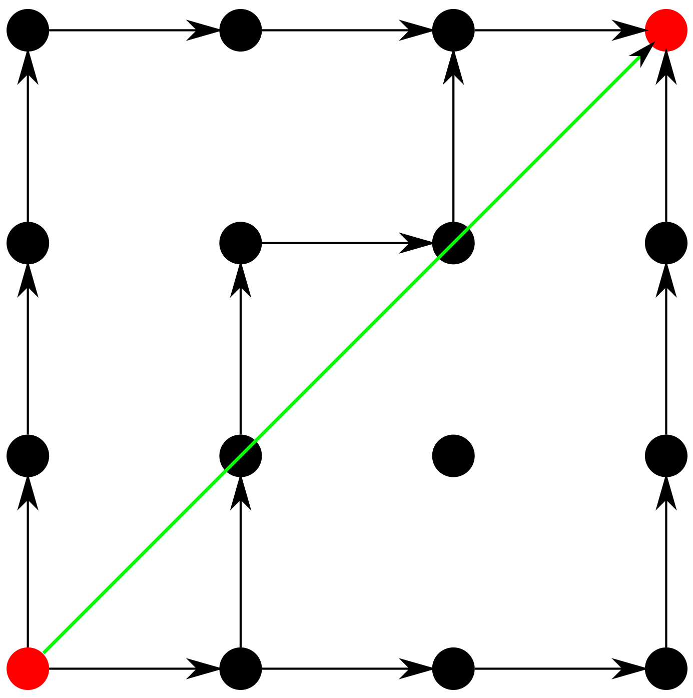


Figure 2.5: Distances in 4 grid paths.

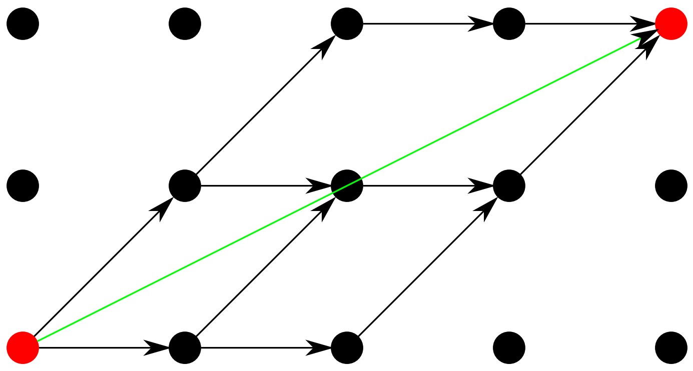


Figure 2.6: Distances in 8 grid paths.

closer to each other, and potentially increases the reliability of hierarchical path planning.

The 4 edge graph type have the advantage that it only has half the edges of the 8 edge graph type, and path planning will therefore be considerably faster in the 4 edge structure. The issue of paths being expected to be further apart in the 4 edge than in the 8 edge graphs can also be circumvented by increasing the radius around the path. This increases the number of edges in the search, but this could potentially be doubled and still achieve the same performance as the 8 edge graphs, and the increased radius might lead to just as good or better results.

As the 4 edge graph structure is the simplest structure and has the fewest edges meaning highest performance speed, this is the preferred choice for the path planning framework. The 8 edge graphs has some potential advantages, and it is left as a possible alternative in the path planning framework.

2.2.3 Path planning using A*

The A* path planning algorithm will be used for the actual path planning. Depending on the heuristic chosen, there is a trade off between running time and optimality. In this implementation, optimality of paths is desired, as it makes the comparison of paths possible as the best paths are always found. Calculation speed is not as important, as this implementation is not used in any real-time applications where the calculation time is critical. Keeping the running times to a minimum is however desirable for practical purposes, as it makes working with data quicker and more efficient.

To ensure that the optimal path and not a near optimal path is found, a heuristic is chosen that always underestimates the remaining cost.

For the 4 edge graph, the minimum distance left between the current vertex $\mathbf{p}_c$ and the target vertex $\mathbf{p}_t$ is given by the l_1 norm

$$d_1(\mathbf{p}_c, \mathbf{p}_t) = |x_t - x_c| + |y_t - y_c| \tag{2.13}$$

The fastest speed possible would be if the terrain type was roads, with a speed $V_{max} =$ 30 km/h $= 8.33$ m/s. The smallest possible time taken between any point is therefore

$$h(\mathbf{p}_c, \mathbf{p}_t) = \frac{d_1(\mathbf{p}_c, \mathbf{p}_t)}{V_{max}} \tag{2.14}$$

and this is chosen as the heuristic.

For the 8 edge graph type, the l_1 distance can overestimate the distance, but the l_2 distance always gives a lower bound of the distance:

$$d_2(\mathbf{p}_c, \mathbf{p}_t) = \sqrt{(x_t - x_c)^2 + (y_t - y_c)^2} \tag{2.15}$$

The heuristic is chosen as

$$h(\mathbf{p}_c, \mathbf{p}_t) = \frac{d_2(\mathbf{p_c}, \mathbf{p_t})}{V_{max}} \tag{2.16}$$

The A* algorithm is implemented in MATLAB, and with a binary heap min queue for increased performance. The graph implementation does not separate between the different terrain resolution, and so the exact same algorithm is run with all data sets. Computation times and other performance measures for the A* algorithm are therefore directly comparable.

With the path planning algorithm describe, the path planning framework described in section 2.2 is complete:

The overall goal of the path planning is finding path that minimizes the time taken between a start and a target point.

Terrain analysis The terrain analysis will use terrain types data and elevation data for providing the speeds for a terrain selection.

Graph generation The terrain will be represented as a tile graph with either a 4 edge or an 8 edge configuration. Weights are set from the speeds found using the terrain analysis.

Path planning The path planning is performed using a standard A* implementation, with heuristics chosen to ensure optimality of paths.

Chapter 3

Hierarchical path planning

Hierarchical path planning is to use a hierarchy structure for use in path planning. The higher levels in the hierarchies will contain less information and details, while the information will increase for lower levels.

An example of this can be path planning in an area which include houses with interiors, and is seen in figure 3.1. The illustration show how a group of houses can be divided into a hierarchy. The highest level is the outside view that shows how the houses are connected through a sparesely sampled graph with only 4 vertices. The next level has a more densely sampled graph, as all the rooms in the house is included in the graph. The the number of vertices increases, and the second level in the hierarchy therefore includes more details and information than the highest level. In the lowest level of the hierchy, the space is even more densely sampled, and includes possible movement for each individual, and the size of the complete is therefore many times greater than the higher hierarchy levels.

For path planning in the example graph, the lowest level is needed for a complete route that can be traversed, for example by walking. However, the lowest level graph will contain information that is not useful for performing the path planning such as the rooms in houses that are not visited, and especially the interior of each room. Using the hierarchies for path planning allows for more efficient use of the graphs.

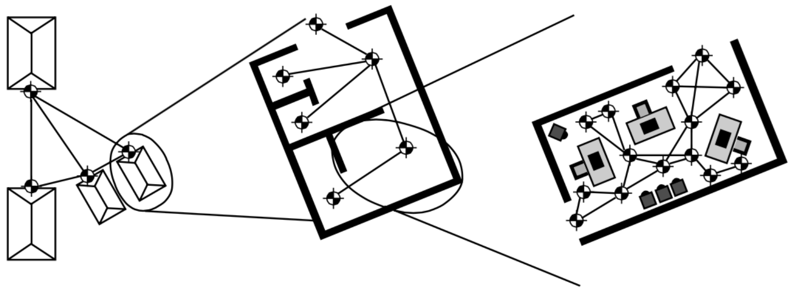


Figure 3.1: Illustration of graphs in different hierarchy levels for houses. Illustration taken from [13].

Using path planning in the highest level, the path found here can be used for excluding the parts of the second level graph that are not in the path. Performing path planning can then be performed in the second hierarchy level for finding a graph that shows which rooms that will be visited. Using this path, all other rooms can be excluded from the lowest level hierarchy, and the lowest level graph has been reduced to a minimum. This reduced graph can therefore be used to find a high detail path, and as the graph used is smaller than the original low level graph, the path planning will have been faster.

There advantage of this method is that the computation time will decrease, but a disadvantage is that we can no longer be certain that the overall shortest path has been found. Details such as shortcuts between rooms can have been lost in the process of creating the mid level graph, and excluding rooms from this can mean that path with lower costs will be excluded from the lowest level hierarchy. However, the path found will presumably not be far from optimal, and so the net result is that a close-to optimal path that is computed faster than with conventional path planning.

Another advantage of hierarchical path planning, is that it can split a complex problem into smaller subproblems. A coarse path can be divided into subsections, and for each of the subsections, the optimal low level path can be found. This is an advantage for use in real time applications, as a coarse path can be obtained fast, and then the vehicle can begin traversing the path as soon as the detailed path of the first subsection is found, and find the next parts of the path while moving. The principles has been assumed to be for the global path planning, but this principle shows how the global path planning can be tied together with the local path planning.

3.1 Previous work

Hierarchical path planning has previously been explored, especially for use in video games where path planning in general is widely used. In these situations, path planning is task required to be performed in real time, and computation time is more critical than finding the best paths, near optimal paths is often good enough for the purposes.

An early hierarchical approach to path planning is to use quadtrees to decompose terrain into a hierarchy [16]. Initially, the map is composed of only 4 blocks. If all the terrain within each block is of the same type such as walkable or non-walkable, the cell remains unchanged. If the terrain within a cell is not homogeneous, the cell is decomposed into 4 new blocks, and so on. Path planning will be performed by moving between the middle of the cells, and the paths are therefore suboptimal. The graph can potentially be greatly reduced in size, with large reductions in computation speed. The method works well for graph with large areas of the same type, but with high detail levels the reduction level, and therefore the effectiveness will decrease. The method works best if the terrain types can be divided into few categories, such as in games where the maps consists of two categories, walkable and non-walkable terrain. With real world terrain, different terrain types and terrain elevation means increased complexity, and quadtrees can not necessarily be expected to give large performance increases.

There are several variants of the A* algorithm that uses principles of hierarchical path planning. The Hierarchical Path-Finding A* (HPA*) is a version of A* for grid based maps [2]. This algorithm has a low level map with fine resolution, and an abstraction is used for creating a higher hierarchy level. The abstraction groups the cells into rectangular clusters of cells. Path planning is used with a graph consisting of the clusters to find the optimal path in this hierarchy level. With the clusters found, the algorithm finds the best way of moving between the clusters, and a final path is obtained. The method reports paths within 1 % degradation of path quality, with up to 10-fold improvement in computation speed for simulations based on mazes with only walkable/non-walkable terrain, from maps of real world games.

The HPA* algorithm has been used by others for further improvements, such as [11] which introduces several methods for increasing performance such as lazy edge computation and faster path smoothing. Other refinements includes a version of the HPA* algorithm called Hierarchical Annotated A* (HAA*) [9], which uses the same principles as the HPA*, but allows a single high level path to be used for multiple entities with different terrain perceptions with respect to speed over different terrain types, and different sets of infeasible terrain.

The Partial-Refinement A* algorithm (PRA*) [17] is a hierarchical path planning algorithm that uses a hierarchy division for finding a high level path, and uses the path for finding partial paths, and eventually a full path for the low level. The path planning itself is performed similarly to the HPA* algorithm, but instead of using a predetermined cluster pattern, the algorithm creates clusters based on similarity between cells, and the clusters are therefore of variable size. A similar method with a different method for state abstraction is also seen in [5]. The partial paths makes it possible for an entity to move before the path planning is complete, which means the time critical part of the path can be calculated fast, while the complete path can be postponed.

The Hierarchical Terrain Representation for Approximately Shortest Paths (HTAP) [14] is a method that is somewhat similar to the HPA* algorithm. It uses a multiple layer hierarchy created by downsampling the original terrain in small grids, repeated multiple times. At the desired hierarchy level, the path planning is performed, and a path is found. The path planning is repeated again in the level below, but only for the cells that belong to the higher level path. This process is repeated until a path in the lowest level of the hierarchy is found. For a variety of different terrains, consisting of images and mazes, the HTAP algorithm rarely performs worse than 1.3 times the cost of the optimal path, and usually better than 1.1 times the optimal cost. The computation times are two orders of magnitude faster than the conventional A* algorithm.

Most of the developments of hierarchical path planning comes from the video game industry, due to the need for quick path planning algorithms, and all of the hierarchical path planning methods offers increased performance, while maintaining near-optimal paths.

3.2 Hierarchical path planning in terrain

Hierarchical path planning methods have seen some use already. For the use in games, the problem with path planning can be said to have a lower bound of the level of details, as a map in a video game will have a finite resolution, and which means that there exists some global optimal path that can be found using path planning.

For the hierarchical algorithms that exists, the algorithms rely on having a well defined lowest level, and the higher hierarchy levels use abstractions of the lowest level, either directly, or through repeated abstractions. A disadvantage of these methods is that there is no logical extensions for extending the problems with more detailed terrain data and lower hierarchies. If the path planning is to be repeated with the new lowest hierarchy level, the entire path planning will have to be repeated. The results already obtained are left obsolete, without any means of using these results for the new path planning.

For terrain vehicles, and especially autonomous terrain vehicles which rely on local path planning as well as the global path planning, the level of details that can be included in the path planning is much higher, and there will never be a lowest level. The lower levels of the path planning might not be available when the path planning begins either, as data gathered by sensors can be added to the terrain representation.

Because of this, the ability to use multiple hierarchy levels together that does not rely on an abstraction of the lowest level is desirable.

3.2.1 Proposal for new method

For a terrain with terrain data given in multiple resolutions, a hierarchical structure can be defined, with the finest resolution being the lowest level, and the coarsest resolution is the highest level. The different resolution terrain data may originate from the same low level terrain data, but it does not have to do so. Nonetheless, as all levels are a representation of the same terrain, paths from a common start to a common endpoint can be found in all different resolutions.

As all terrain levels are different representations of the same terrain, it seems likely that the paths in the different terrain levels will be close to each other. A path found in a higher level could therefore possibly be used for aiding in finding the optimal path in a lower level.

A way of using the higher level path is to define a neighborhood around this path, and find the shortest path within this neighborhood in the lower level. If the optimal path in the lower level is within this neighborhood, the search should yield the global optimal path. If the global optimal path is not entirely within this neighborhood, a local optimal path will instead be found, which will be suboptimal compared to the global optimum. However, the cost of this local optimal path can still be quite close to the optimum, and can be good enough. The method is somewhat similar to the HTAP algorithm [14], but the difference is that all higher hierarchies are built from the lowest level. It also only searches for a low level path within the children of the higher level path instead of in a neighborhood, greatly reducing the flexibility in case of terrain differences between the terrain levels.

The selection of the neighbor can be done several ways. The simplest method will be a fixed radius around the high level path. The radius for such a path would need to be determined by experiments. More advanced alternatives could be to increase radius where there is more uncertainty in the terrains, and decrease the distances where the terrains are similar. This could potentially increase performance and quality of the paths, compared to a fixed radius approach.

3.2.2 Method evaluation

The overall goal of this thesis is to investigate if this proposed method can give increased performance over conventional path planning using fine terrain resolution. There are several possible challenges that needs investigation in order to determine if this is the case, and if the method can yield any contributions, or if the performance will be too low to have any practical uses.

Computation time

For the hierarchical approach, there are several factors that determine the efficiency of the path planning. A conventional approach will only have to run the path planning once, but in a low level terrain which is time consuming. The hierarchical approach proposed will instead need to perform 3 steps:

1. Perform path planning in the high level terrain.
2. Create a reduced size low level graph.
3. Perform path planning in low level reduced size graph.

The size of the neighborhood searched, and the relative difference in the terrain resolutions will determine how fast the hierarchical path planning will be. How large improvements that can be expected will however be needed to be tested with experiments.

Optimality

Another aspect that needs to be determined is how often the hierarchical path planning gives the optimal solution. In the cases it does not, it is also interesting to see how far from the optimum the path costs will be. As each of the hierarchies aren't necessarily derived from the others, there is potentially differences between the terrain levels, that can yield poor results. How often this happens is also an interesting aspect of the path costs.

Resolutions

Which relative resolution levels that will yield the best results is an aspect that also needs experimentation. Terrains with little difference in resolution will have smaller differences in the terrain, and should give better paths, but can lead to large computation times. In the other end, with large differences in the resolutions, the path planning of the high level can be performed quickly, but the path planning in the low level is still time consuming, and so the difference should not be too large either.

3.2.3 Extensions of the method

The mentioned challenges can be considered the main parts for providing a complete experiment of the effectiveness of the proposed method. However, there are also some ways of improving the method further.

Multiple hierarchies

As seen in [14] and [17], several hierarchy levels were combined for the best performance. The same principle can be applied here also. With a hierarchy with 3 or more levels, the first level can be used for finding a hierarchical path in the next layer. This path can further be used for finding a hierarchical path in the next level, all the way down to the lowest level. The size of the graphs at each level can possibly be reduced for each level, resulting in an overall faster path planning, while maintaining much of the reliability of the regular 2-level hierarchical path planning.

If using high resolution terrains for the finding a path basis can improve accuracy, then it could be an idea to use even lower resolution terrain data to find an very rough path first, and use this to find a mid-level path which finally will be used to find the final path. If using high resolution paths in fact will increase accuracy, then using multiple levels could improve the speed of finding these greatly.

4 or 8 edge graphs

With tile graph as proposed in the path planning framework in section 2.2, the simplest choice for the graph structure is the 4 edge graph type. However, as discussed in section 2.2.2, the 8 edge graph can have an advantage in that paths between the same start and end points, might be located closer to each other. For hierarchical path planning, this could be an advantage as it could mean that the hierarchical path obtained more likely will be the optimal path as well. This could lead to increased path quality, but possibly at the expense of increased computation time.

Chapter 4

Path analysis

Comparing paths is necessary to evaluate results for path planning, and to compare the obtained hierarchical paths with the conventional paths. Optimality of hierarchical paths is one aspect that is needed to determine how successful the hierarchical approach is. Comparing computation time is also needed to determine if the hierarchical path planning will provide improvements in performance, and how much this will be.

As the hierarchical path planning is based around the assumption that an optimal path in a fine resolution terrain will be close to an optimal path in a coarser resolution, a measurement for how close or similar two paths are is also needed. In section 4.3, such a metric is proposed.

The three measurements used are:

- Path cost of an entire path
- Computational cost, the time taken to find a path
- A distance metric for paths based on area between them

4.1 Path cost

The cost of a specific path is a valuable measure to compare the difference between to paths. The cost of a path is the sum of the weights of the edges in the path. For a path consisting of a subset of the graph edges $S \subset E$, with weight w, the total cost is given by

$$Cost = \sum_i w_i, \quad i \in S \tag{4.1}$$

The weights are defined by how long time it takes to traverse an edge, hence the cost is independent of terrain resolution. The costs of paths are therefore directly comparable, even if they are in different terrain resolutions. However, as discussed in section 2.2.1, differences are expected when comparing paths of different resolutions due to differences in the terrain data.

4.2 Computational cost

One of the main reasons for using hierarchical path planning is that it can provide increase in computational performance, and a measure of this is therefore useful. This report proposes a method for potentially reducing the computational cost by considering multiple hierarchy levels, and ways of measuring this is therefore essential. Computation time is a simple measurement suitable as a way of measuring the computations. The graph types and the path planning algorithms are always the same, both across hierarchy levels and also both for full graphs or reduced graphs found using a fixed radius distance from another path. The computation time can therefore be comparable for different kind of paths that are found.

Of all the calculations that are done, including processing the terrain data and creating the actual graphs, it is mostly the actual path planning that is of interest. The preprocessing of the data can be done beforehand and is therefore not critical. An exception is the creation of the fixed radius graphs used for hierarchical path planning which occurs after a path is found. This should therefore be included when finding taken into account when comparing hierarchical path planning with conventional path planning.

4.3 Path distance metric

When investigating different paths between the same start and endpoints, how similar paths are can be intuitive when examined visually. However, to quantify this is not necessarily as simple, and a consistent metric is needed to measure this.

One method for comparing how far away two paths are from each is to find the area between the paths. If two paths are far apart, the area between the paths will be large. Conversely, if the paths are close to each other, the area will be small. The area between two curves can be useful, but it makes it hard to compare paths of different length, as a certain area for a short path means a much larger relative area than the same area for two longer paths.

If two paths are located at a constant distance from each other, the area between the paths will simply be

$$Area = distance \cdot path\ length$$

Equivalently, the distance between the paths is

$$distance = \frac{Area}{path\ length}$$

If the distance varies, the formula will not give a correct distance at any point on the curve, but the area divided by the path length can still be used as a measurement of the average distance between the curves. If the curves are of different length which is likely the case, then the length of the shortest path will be used. This measurement will be a consistent measure for how far apart two curves are. It is also independent of the length of the curves, so the average distance between two curves will be directly comparable with other cases.

Calculating the area of a region R can be difficult to perform directly, and an alternative to a direct calculation is to use Green's area fomula.

$$\text{Area of } R = \frac{1}{2} \oint_{C} -y\, dx + x\, dy \tag{4.2}$$

Greens area formula allows the area of a region R with the boundary C, to be calculated as a curve integral along the boundary C. As the area is always positive, the integral can be evaluated in either direction and then take the absolute value. Green's area formula is derived from Green's theorem for the plane, see Appendix A.1 and A.2 for details. As this provides a convenient solution to finding the area within a curve, this is used further in this report.

For the purpose of the area between two paths, this special case of Green's theorem can be applied. By reversing the direction of one of the paths, the two paths combined form a closed curve, as the paths have the same bounding a region R.

A requirement for the theorem is that the boundary C is a simple closed curve. As the paths have the same start and end points, the boundary will be closed. If the paths crosses each other, the boundary is not a simple closed curve. However, as the paths crosses, the region R will be divided into subregions, each which has some simple closed curve as a boundary. The boundary will consist of parts of the two paths, and by finding the point where the paths crosses each other, the boundary for the smaller subregion can be found. With the boundary for each of the subregions, Green's theorem can be used for finding the areas of each of the subregions between each intersection separately, and taking the sum to find the total area. If the paths are identical between some points along the curves, every point can be considered an intersection. But using Green's theorem will not be a problem as the area will be zero, so these parts of the curve will not affect the answer.

This is illustrated in figure 4.1, showing two paths between a start and an end point. The paths crosses each other forming subregions with parts of the paths as the boundary. The complex shape of the bottom right region show how a direct area calculation could be hard to implement. Given only the paths, it could be hard to determine which parts of in the plane are inside and outside of the boundary. Taking the curve integral for each of the three subregions is however simple. For the middle region, the curve orientation of the curve is reversed compared to the other two, but taking the absolute value solves the problem.

If the order of the intersection points is different for two specific paths, the corresponding subregions becomes more complex. This is illustrated in figure 4.2. For path a, the

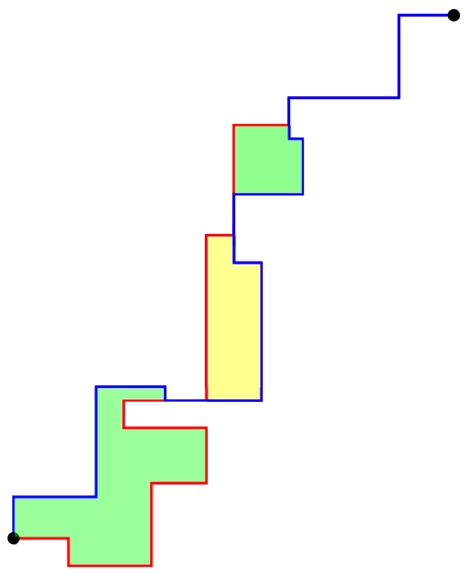


Figure 4.1: Area between two intersecting paths

intersections are in the order $a_1,\ a_2$ and a_3, and for the path b, the intersections are in the order of $b_1,\ b_2$ and b_3, but as $a_1 = b_3$, $a_2 = b_2$ and $a_3 = b_1$, the intersections are not in the same order for both paths. This creates problems for area calculations, as the boundaries of the subregions are more complex. The region in the bottom left does not have a boundary consisting of two sections of the paths between a start and an endpoint, instead it consists of multiple sections of the paths.

A way of avoiding these problems is to ignore the inner paths between the first and last intersections, and instead calculate the entire area by defining the boundary between the start point, point a_1, the end point, point a_3 and back to the start point. This can be used in a general setting, by finding a the first and last points that are not in order, and ignore the points in between.

4.3.1 Area calculations for specific graph types

For the graph types in question, namely the 4 edge per vertex and the 8 edge per vertex graphs discussed in section 2.2.2, each edge is be either horizontal, vertical, or diagonal, when viewed in a xy-coordinate system. It is therefore possible to calculate these line integrals directly.

Area for 4 edge graph

For a 4 edge graph per vertex, the edges of a vertex is shown in figure 4.3. Note that the xy-coordinate system is defined as shown in the figure, and not in the regular cartesian coordinate system. The reason is purely for implementation purposes, as MATLAB works with image coordinates defined in this way.

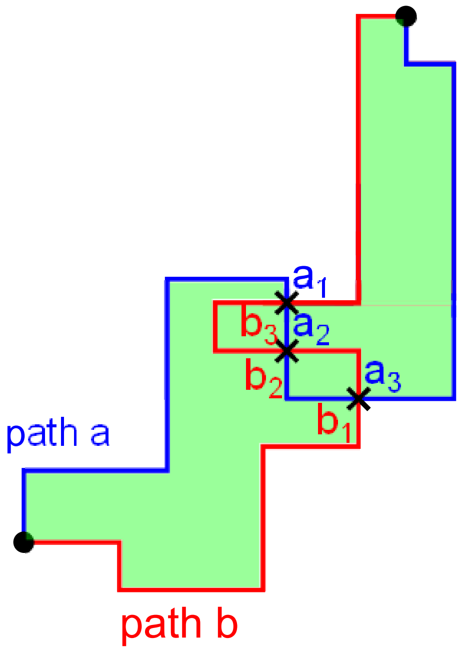


Figure 4.2: Path intersection in the wrong order

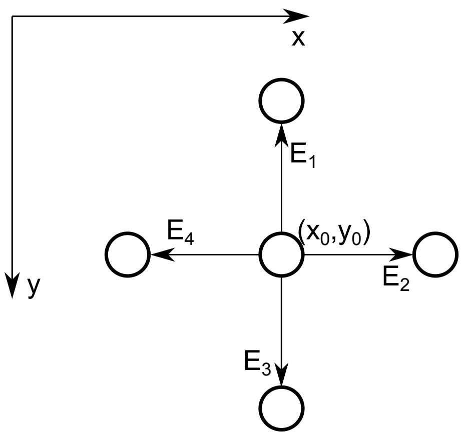


Figure 4.3: Edges of a 4 edge graph

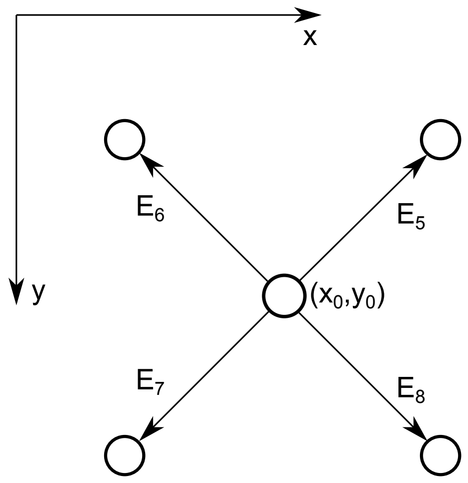


Figure 4.4: Diagonal edges

There are 4 different edges as shown in figure 4.3. E_1 is the north edge, E_2 is the east edge, E_3 is the south edge, and E_4 is the west edge. For an edge E_j, if we denote the integral over this edge W_j then

$$W_j = \frac{1}{2} \int_{E_j} -y\,dx + x\,dy \tag{4.3}$$

is calculated in appendix A.3, and the results are

$$W_j = \frac{1}{2} \int_{E_j} -y\,dx + x\,dy = \begin{cases} -\frac{1}{2}x_0 \cdot d & j = 1 \\ -\frac{1}{2}y_0 \cdot d & j = 2 \\ \frac{1}{2}x_0 \cdot d & j = 3 \\ \frac{1}{2}y_0 \cdot d & j = 4 \end{cases} \tag{4.4}$$

The area of a simple region will then be the sum of the all the edge integrals

$$A_R = \left| \sum_i W_i \right| \tag{4.5}$$

for the edges E_i along the boundary.

Area for 8 edge graph

For the 8 edge graph, edges are also added in both northwest-southeast and northeast-southwest directions. The edges from the 4 edge types are still present, but 4 other edges types are added, shown in figure 4.4. The integrals over each of the edges are

$$W_j = \int_{E_j} -y\,dx + x\,dy = \begin{cases} (-x_0 - y_0)d & j = 5 \\ (-x_0 + y_0)d & j = 6 \\ (x_0 + y_0)d & j = 7 \\ (x_0 - y_0)d & j = 8 \end{cases}$$

The calculation of the integrals are shown in Appendix A.4. As with the 4 edge paths, the area is calculated with

$$A_R = \left| \sum_i W_i \right| \tag{4.6}$$

for the edges E_i along the boundary.

Average distance

The path length is a trivial calculation, as it will be the sum of the individual edge lengths. These are $l = d_0$ for straight edge and $l = \sqrt{2} \cdot d_0$ for a diagonal edge. The total path length L for a path is then given by

$$L = \sum_i l_i \tag{4.7}$$

The average distance D is then given by

$$D = \frac{|\sum_i W_i|}{\sum_i l_i} \tag{4.8}$$

Implementation of the path distance metric in MATLAB is included in appendix B.3.

With some consistent measure of how similar two paths are, it will be possible to analyze if hierarchical path planning using one path as a basis for the another one could be a good idea. Intuitively, if two paths in different terrain resolutions are similar, it makes sense that the one in the higher resolution could be found using the neighborhood of the low resolution path as a basis.

Chapter 5

Experiment setup

In the previous chapter, a method for performing hierarchical path planning as an alternative to conventional path planning was proposed. The hierarchical approach has the potential of being yielding faster performance of the path planning. With the method proposed, the path found will not necessarily be the global optimal path, but rather a local optimal path.

To investigate if the proposed method really can provide improved performance, experiments that implements the hierarchical approach must be performed, along with the regular path planning for comparison. The main focus of the experiments will be any decrease in computation time and how close the hierarchical paths are to the global optimal paths.

The methods discussed in this chapter can be used for general terrain data. When performing experiments in practice however, a selection of terrain data must be made. The results will necessarily depend on the terrain data used, but hopefully the results can be used to draw conclusions for a general case, and possibly develop some methods for evaluating the success of the hierarchical path planning.

5.1 Hierarchical path planning framework

For performing hierarchical path planning, a framework for the path planning is needed. Necessary for performing any path planning is terrain data. Three major components needs to be implemented for the experiments.

Terrain analysis A form of terrain analysis needs to be implemented. This will take raw terrain data and make useful information for the required tasks of the terrain vehicle in question. The terrain analysis will be different depending on the goals for the path planning.

Graph generation The terrain will be represented as a graph structure. Path planning can therefore be done with shortest path algorithms for graphs. The weight of the edges will be set based on the terrain information gathered through the terrain analysis.

Path planning The path planning itself will be performed using standard path planning algorithms as the terrain is just a regular weighted graph. For the hierarchical path planning, a method for taking a path in one level of the hierarchy and finding the neighborhood of this path is also required. With this neighborhood, a new graph only allowing movement within this neighborhood must be used.

The overall goal of the experiments will be to investigate the hierarchical path planning principles. The focus is not on making the path planning itself necessarily as realistic or run with as high performance as possible. Obtaining a consistent framework that easily can be replicated in other settings and with variety of different terrain data set will be the focus. As long as all terrain analysis, graph generation and path planning are performed in a manner similar to common practices, the principles and conclusions of this investigation can be transfered to other applications.

5.2 Terrain analysis with real data

The terrain data consist of two types of data, elevation data and terrain types. The data is sampled from the real world in an evenly spaced grid pattern. The data can be viewed as a projection into a 2D plane as viewed from above. The terrain data consists of elevation data and terrain types in equal resolutions. The data are stored in GeoTIFF files, a version of the TIFF image files that is georeferences and therefore contains information about real world coordinates of the data.

5.2.1 Terrain types

The terrain data describes the terrain type of each individual cell in the grid. The terrain types are divided into 14 categories listed in table 5.1. The colors in the table shows the colors of the terrain data as seen in the GeoTIFF file [15]. An example of a selection from the terrain data is shown in figure 5.1. This sample has a resolution of 1 m, which is the highest resolution available in the provided data.

Using only 14 categories means utilizing the terrain data is simple, as the information is limited. There is also a tendency for the terrain data to feature relatively large connected areas of the same type, and few small details. The exception are roads and streams which always are in line form, not as areas. Large connected areas contributes to making the path planning more predictable, as this likely means small differences between different resolutions.

A disadvantage can be that the generalization can lead to large deviations from theoretical path planning in a priori data, to paths that can be applied in the real world. In a real life situation, the specific way of choosing the parameters would be. Therefore, performing terrain analysis with this type of terrain as a basis might not lead to the most accurate results when comparing to the real world.

Table 5.1: Terrain Categories

	Color code			Color	Description
	R	B	G		
Areas	0	128	128	🟦	Farmland
	128	128	128	▨	Dried out river or lake
	128	255	255	🟦	Industrial area
	0	0	255	🟦	Lake
	128	0	64	🟪	Wetlands
	0	0	0	⬛	Ocean areas
	255	255	255	⬜	Open areas
	64	128	128	🟦	Quarry
	0	0	160	🟦	Rivers (large)
	0	168	0	🟩	Sports fields
	0	128	64	🟩	Forest
	255	255	0	🟨	Residential area
Lines	0	0	255	🟥	Road
	64	0	160	🟪	Stream

For the purposes of experimenting with hierarchical path planning, this will not affect the results in anyway, as the highest resolution available will be considered the truth. It is however important to keep in mind that certain details can have been lost when transferring theoretical paths to the real world.

Inaccessible terrain types

First part of analyzing the terrain, is to determine which types of terrain is inaccessible for traversing with a terrain vehicle. All kinds of water would be infeasible. Of the categories in table 5.1, this is oceans, lakes, rivers and streams. Small streams could potentially be possible to cross in some instances for a real vehicle, but this is not possible to determine from the terrain data alone. Defining the streams to be inaccessible means highest consistency, and ensures realistic results.

Urban and industrial areas must be assumed to be inaccessible as well. Areas like this typically contain buildings which are not represented in the terrain data, and can feature industrial equipments, fences and other objects that make the areas inaccessible. These areas are therefore treated as inaccessible in general to avoid any issues.

The last terrain category that is considered infeasible is quarries. These typically contain large slopes, high drops and will often feature large stone blocks, piles of rock and heavy industrial equipment. An ATV could typically traverse a quarry, but the terrain data available would likely not be enough, and is therefore considered infeasible.

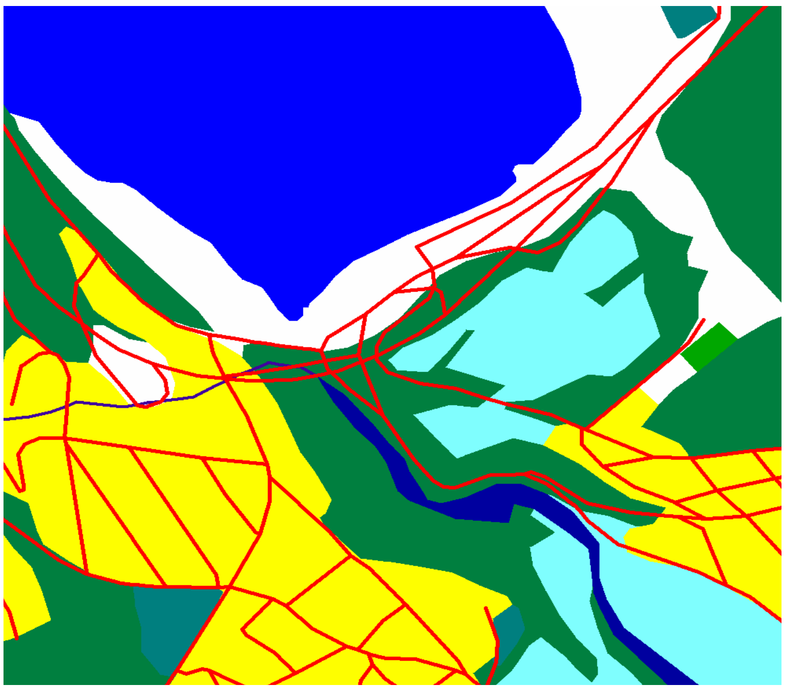


Figure 5.1: A sample showing terrain types with 1 m resolution

Terrain speeds

The remaining terrain categories will be considered accessible, and the speed possible for each of the terrain types will vary. On roads, the vehicle will be able to maintain maximum speed without difficulty. Other terrain types like sports fields consists of flat areas with trimmed grass and allows high speeds. Farmland and wetland will usually be open areas with rougher surface and therefore not as high speeds can be held. Forests can in general be traversed with a terrain vehicle, but only with low speeds. Exact speeds for different terrain types will vary greatly with the type of vehicle in question. For example, wetlands could be inaccessible for some types of vehicles, while others could move through with high speeds.

In table 5.2, speeds for the different terrain types are chosen. These values are based on [18], where terrain data has been used in a similar way. The exact values that are chosen are also not crucial to get accurate, as long as they appear realistic, and are chosen in a similar manner as in other applications.

Terrain resolutions

The provided terrain types data has a 1 m spatial resolution of the terrain, meaning the size of the grid cells is 1 m by 1 m. The elevation data which will be considered in section 5.2.2, is available in the 1 m resolution, and also in 2 m, 10 m, and 50 m resolutions. When experimenting with using different resolutions for path planning, these 4 resolutions will

Table 5.2: Speeds in different terrain types

Terrain type	**Speed**
Farmland	20 km/h
Dried out river or lake	20 km/h
Industrial area	0 km/h
Lake	0 km/h
Wetlands	15 km/h
Ocean	0 km/h
Open areas	15 km/h
Quarry	0 km/h
Rivers (large)	0 km/h
Sports fields	25 km/h
Forest	5 km/h
Residential area	0 km/h
Road	30 km/h
Stream	0 km/h

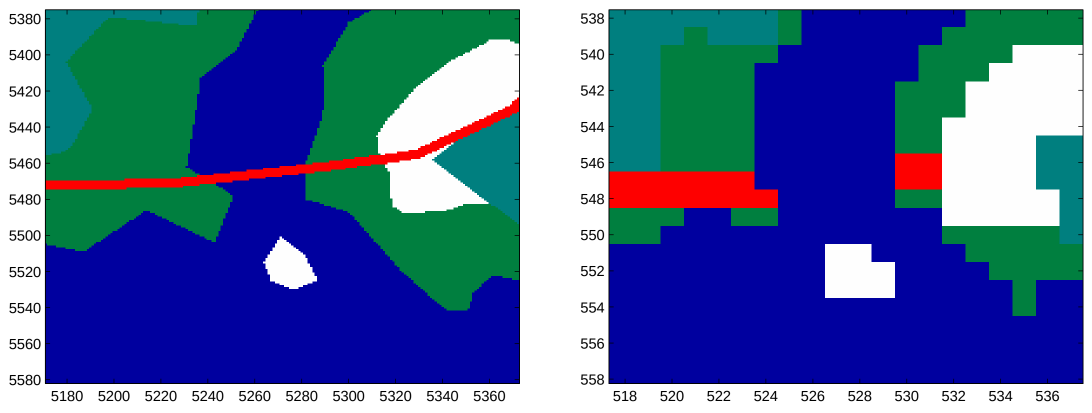
be the hierarchy levels. As the terrain types is only available in the highest resolution, a downsampling is required for using the terrain data at other levels.

Downsampling terrain types

Downsampling terrain types must be performed in a way that preserves important features of the terrain. This is especially important as for example narrow terrain features can disappear altogether. If these are significantly different than the surrounding terrain, this can impact path planning through these areas. One example here could be a bridge over water. The bridge provides a narrow path through an otherwise infeasible area. As the path is narrow, a naive downsampling algorithm could make the bridge disappear, or make holes in it, both which would leave the bridge as a whole infeasible to traverse.

For path planning purposes, this effect will be especially critical. As bridges over rivers typically are the only crossing nearby, making these infeasible will mean potentially large deviations between paths. Another similar example are streams which in are long stretches which are infeasible in otherwise feasible terrain. A stream can not be crossed, but a naive downsampling can leave holes in the stream allowing a crossing through feasible terrain. The end result would be the same, large differences between the the true path and the path in the coarse terrain.

An illustration of the problem is shown in figure 5.2 where a terrain is downsampled from 1 m resolution to 10 m resolution. Each 10$\times$10 m block in the fine terrain will be a single cell in the coarse terrain. A naive, but reasonable method for downsampling is to choose the most frequent terrain type within each block. With this type of downsampling, the downsampled version of figure 5.2a is showed in figure 5.2b. Comparing the figures, a noticeable feature is how the bridge in the fine terrain is missing in the coarse terrain, and



(a) Terrain in 1 m resolution **(b)** Terrain downsampled to 10 m resolution

Figure 5.2: Problems with naive downsampling of terrain

thus leaving the river to appear infeasible to cross.

A solution is to create a hierarchy of terrain types, where some terrain types take precedence over others. This allows for the most important features to be maintained. Most important are the infeasible terrain types and the roads which provides paths through infeasible terrain. The other types of feasible terrain consists of larger areas, and the effects of downsampling here will only change the terrain features along the borders between different terrain types. This can lead to narrow strips where the terrain speeds are differ from the fine terrain, or switch feasible or infeasible terrains. The net effect will be that some areas of some terrain types will be slightly larger, and others slightly smaller, and this should only lead to small errors in the end result.

In the priorities, all the infeasible areas are considered equal as the resulting speed that is set will be 0 km/h in all the cases. The remaining terrain types all have non-zero speeds, and the priorities will not significantly affect the result. A choice is made to let the faster terrain types have higher priority, but this is of little significance. The choice of priorities of the feasible terrain types will only have a minor effect on the results, and other orders could have been chosen. A table showing all the priorities is shown in table 5.3.

The algorithm for downsampling is simple. As mentioned in section 5.2, the downsampling will always be done with a integer factor N, and therefore, all cells in an N-by-N block will be downsampled into a single cell in the new coarse resolution. With the defined priorities, the new cell is chosen to be the terrain type with the highest priority within the block. This will ensure that no elements of higher priorities will be lost in the process of downsampling.

The downsampling will introduce errors in the overall speeds in the terrain due to two effects. First a tendency for faster terrain to take precedence over slower, and thus increasing areas with faster terrain. Second, a tendency of the infeasible areas to take precedence over feasible terrain and thus decreasing the overall speed. The net effect of both will be small as this will be an issue only at the border between terrain types, but it shows that

Table 5.3: Priorities of terrain types when downsampling

Priority	Terrain type	Speed
1.	Road	30 km/h
2.	Industrial area	0 km/h
	Stream	0 km/h
	Lake	0 km/h
	Ocean	0 km/h
	Quarry	0 km/h
	Rivers (large)	0 km/h
	Residential area	0 km/h
3.	Sports fields	25 km/h
4.	Farmland	20 km/h
	Dried out river	20 km/h
5.	Open areas	15 km/h
	Wetlands	15 km/h
6.	Forest	5 km/h

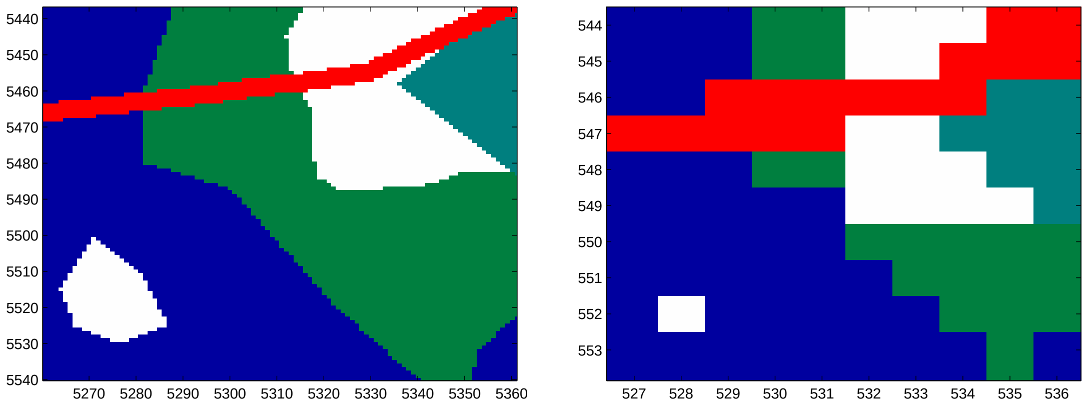


Figure 5.3: Downsampling from 1 m to 10 m

inaccuracies are to be expected when downsampling terrain.

In figure 5.3 an example of a downsampling from 1 m resolution to 10 m resolution. Important features such as the road, and water are preserved, and covers more area than before. Less important features can therefore shrink, such as the small island to the left in the terrain selection.

An implementation of the downsampling in MATLAB is included in the Appendix in Code B.1.

5.2.2 Elevation

In addition to the terrain types data, detailed elevation data is also provided. The data sets contains the elevation above sea level for individual points in the terrain, organized as cells in a grid in the same manner as with the terrain types. A difference from the terrain types is that the where the terrain types are segmented into 14 categories, the elevations are given as floating point values, measured with 0.1 m vertical resolution. With accurate elevation measurements, slopes can also be calculated for the terrain which will be utilized for path planning.

Elevation data resolution

The elevation data provided have a spatial resolutions of 1 m, 2 m and 10 m. When performing experiments with multiple hierarchy levels, these resolutions will be the resolutions used. All the resolutions are divisible by all the lower resolutions. An advantage with these resolutions is that all points in a fine resolution terrain is clearly defined to belong to a single point in any coarser resolution.

A transfer the other way around from a coarse to a fine terrain resolution will however be ambiguous. Any point in a coarse terrain can correspond to any point within the corresponding block of cells. For example, a single point in the 2 m resolution could mean any of 4 points in the 1 m terrain, and so it does not make sense to move from a coarse terrain to a fine terrain directly. Avoiding moving from coarse to fine resolution altogether will circumvent the problem, and this is therefore not a real issue.

An example of height data visualized can be seen in figure 5.4. The height data are in 1 m resolution. The displayed area has it's lowest elevation around 35 m above sea level, and shows a small hill with the highest elevation of around 150 m.

Slopes

Using the specific elevation data provided, the method of section 2.2 can be used for calculating the slopes and combining these with the terrain types data. Setting the limits for the slopes of the Go, Go slow and No go regions are based on [3], where a similar use of elevation data is utilized. The limits for the No-go region are chosen as $25°$, and slopes larger than this is considered infeasible. The Go-slow region is chosen to be the slopes between $15°$ and $25°$. The Go-slow speed is chosen to be 5 km/h. The slope limits here are the overall slope of a cell, and not just the slope in the direction of which the vehicle is moving. This is a choice that is made to make the calculations and data needed simpler. For visualization purposes it also makes more sense as a the state of a single cell is

The specific method for determining the speed in a cell will therefore be summarized as follows: If the terrain speed of a cell is given as $V_0(u)$, the speed of a cell $V(u)$ with a

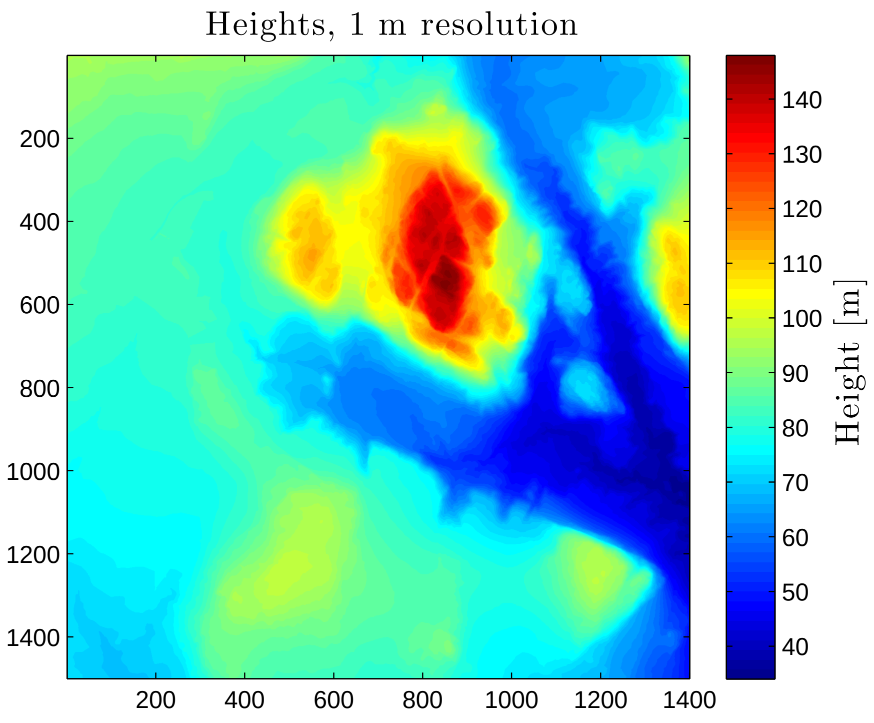


Figure 5.4: Height data with 1 m resolution

slope θ is given as

$$V(u) = \begin{cases} 0 & 25^\circ \leq \theta \\ \min(V_0(u), 5\text{ km/h}) & 15^\circ \leq \theta < 25^\circ \\ V_0(u) & \theta < 15 \end{cases} \tag{5.1}$$

There is one exception to this rule, which is roads. Roads are often found in the forms of bridges, or can for example be located right next to ditches, embankments and other terrain features that locally gives large slope measurements. Small errors in elevation measurements can lead to road sections being measured to have too high slopes, and road sections that are feasible can according to the model appear as infeasible. As roads in almost all cases are traversable for an ATV like vehicle, roads with slopes larger than 25° in are assumed to be in the Go-slow category instead of the No-Go category. This means that roads are always feasible in the developed speed model.

The implementation of the slopes calculation can be seen in Code B.2. The slopes for the area shown in figure 5.4, is shown in figure 5.5. The color indicates the slope angle, with dark being 0°, that is flat terrain. The maximum slope in the displayed area is dark red which is close to 80°, a nearly vertical drop.

Implementation of the quantization of the slopes can be seen in Code B.3. The same area shown with the slopes quantized to the three different regions of (5.1) can be seen in figure 5.6.

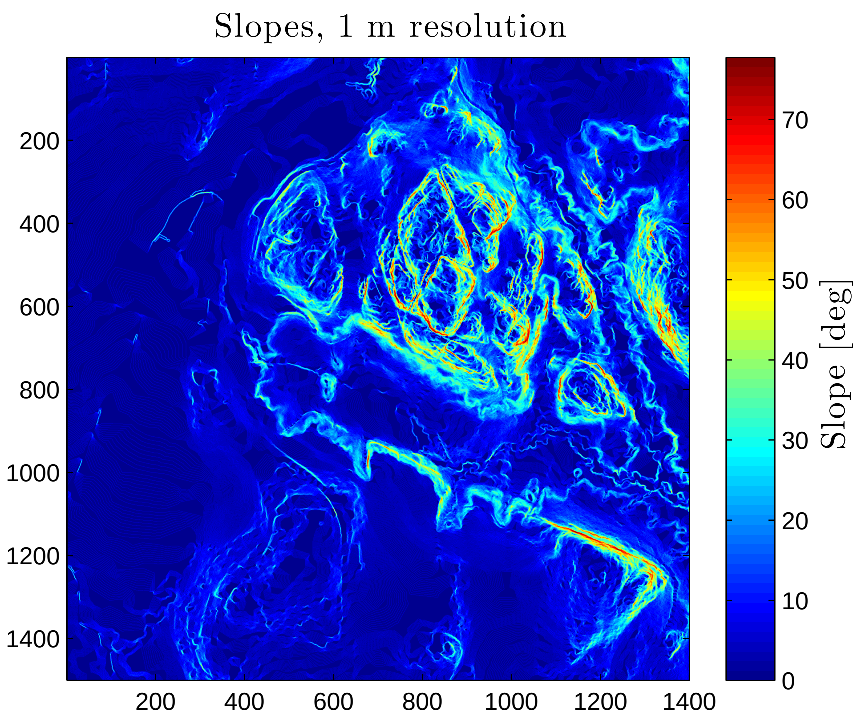


Figure 5.5: Slopes in 1 m resolution

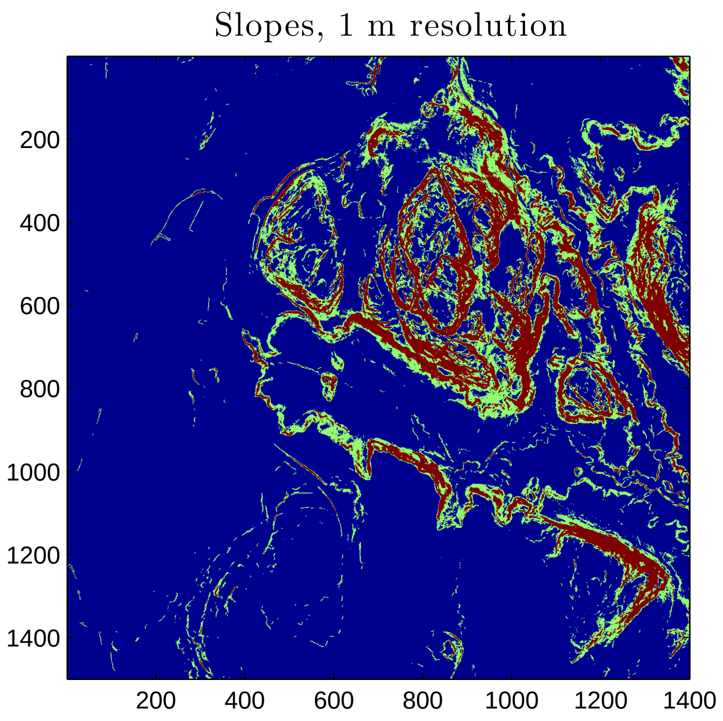


Figure 5.6: Slopes 1 m resolution, quantized into 3 levels. Blue: $\theta < 15°$, Yellow: $15° \leq \theta < 25°$ and red $\theta \geq 25°$

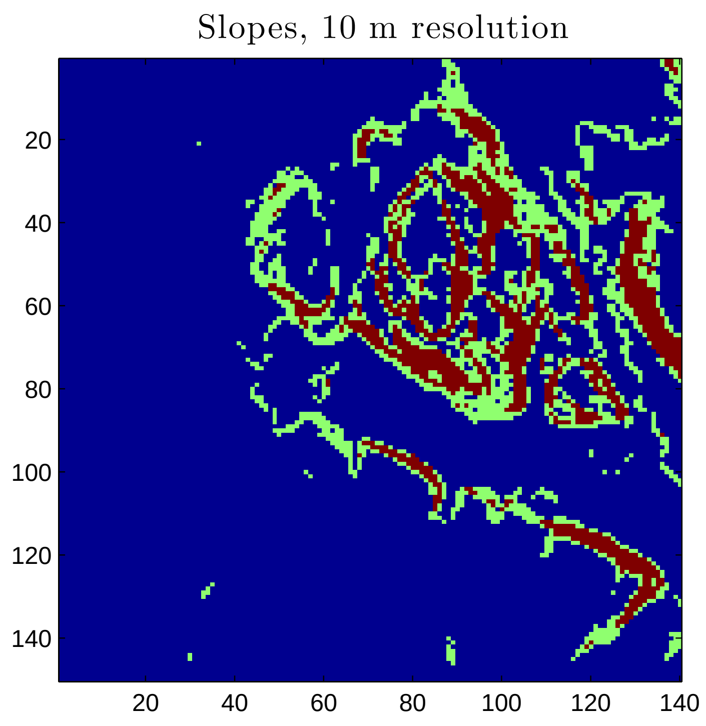


Figure 5.7: Slopes 10 m resolution, quantized into 3 levels. Blue: $\theta < 15°$, Yellow: $15° \leq \theta < 25°$ and red $\theta \geq 25°$

Slopes in different resolution

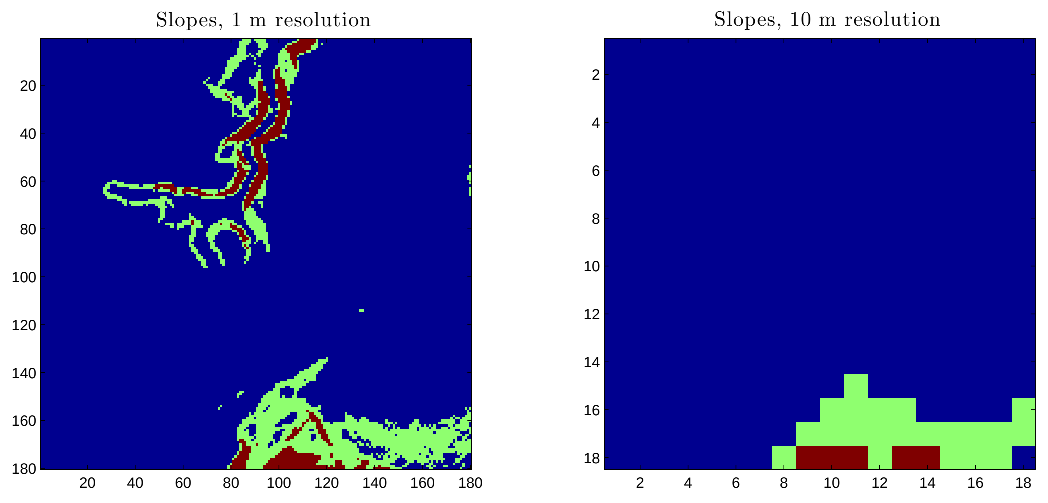
As the elevation data is available in different resolutions, the method of downsampling needed for the terrain types is not necessary. Instead, the same method of calculating slopes can be used with elevation data of any resolution. Avoiding downsampling means the slopes calculated will be the same whether a coarse resolution is the only available, or if a finer resolution is the basis.

In figure 5.6, the quantized slopes is shown with a 1 m resolution. Using elevation data with 10 m resolution, the slopes for the same area is shown in figure 5.7.

Detail loss for coarse resolutions

The detail level of the slopes data is high compared to the relatively low detail level in the terrain types. As seen in the terrain types sample in figure 5.1. Slopes with the same 1 m resolution is shown in figure 5.6, and by inspection it can be seen that the slopes has complex and narrow patterns. This will pose a problem when comparing to slopes data with lower resolution for slope details narrower than the new resolution. An example can be seen when comparing the 1 m slopes with the 10 m slopes for the same area seen in figure 5.7, where many narrow details has been lost.

A detailed example of this effect can be seen in figure 5.8, which is taken from the areas



(a) 1 m resolution **(b)** 10 m resolution

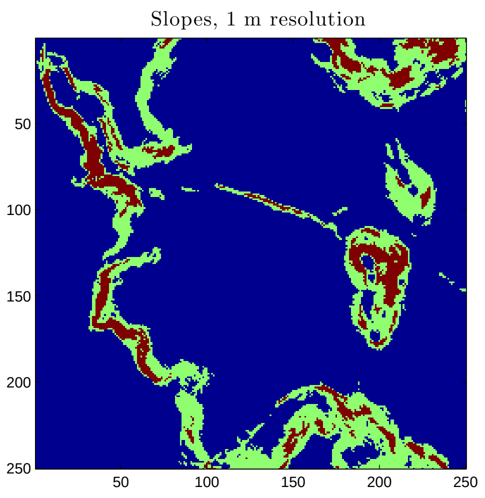
Figure 5.8: Detailed area taken from figures 5.6 and 5.7. Narrow details in (a) have dissappeared entirely in (b), while the larger features at the bottom are preserved.

of figure 5.6 and 5.7. Here, a series of narrow details in the 1 m terrain are completely lost in the 10 m resolution, and the area appear as flat with slope less than 15°.

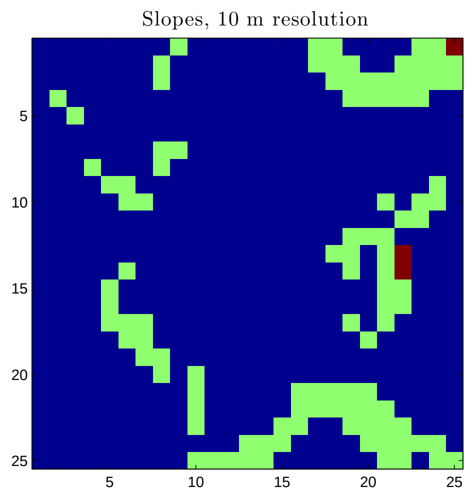
For wider details, they may not disappear altogether from the downsampling, but rather have a tendency to appear less steep in the 10 m terrain. An example of this effect can be seen in figure 5.9. Here, the Go-slow areas are for the most part preserved, but the No-go areas are not there in the 10 m resolution.

A consequence of these effects is that it allows for paths that appear feasible in the 10 m terrain to be infeasible in the higher resolution terrain, as it can show infeasible terrain as feasible. This can potentially cause problems when performing hierarchical path planning, as this essentially is to find paths in coarse resolution terrain and use these as a basis for path planning in a fine resolution terrain. If the paths in the coarse resolution can not be trusted to be feasible in finer resolutions, then the finer resolution paths must be longer, which could mean that entirely different paths will be better overall.

A comparison of the percentage of the terrain that are in each category shows that for this area, the relative distribution in both the terrain are very close as seen in table 5.4, with a small decrease in the steepest areas. That indicates that there isn't possible to draw any clear conclusions of whether one of the resolutions yields overall smaller or larger slopes than the other. This makes it harder to tell where the differences in paths in the fine and the coarse terrain will arise, as it can't be predicted accurately where the coarse terrain is missing significant details.



(a) 1 m resolution



(b) 10 m resolution

Figure 5.9: Detailed area taken from figures 5.6 and 5.7. Most features are preserved, but there is a tendency of areas that are red in (a) appear as yellow in (b).

	1 m	**10 m**
$0 \leq \theta < 15°$	81.1 %	82.6 %
$15 \leq \theta < 25°$	10.4 %	10.4 %
$25 \leq \theta$	8.5 %	7.0 %

Table 5.4: Distribution of the relative quantity of area in each of the slopes categories.

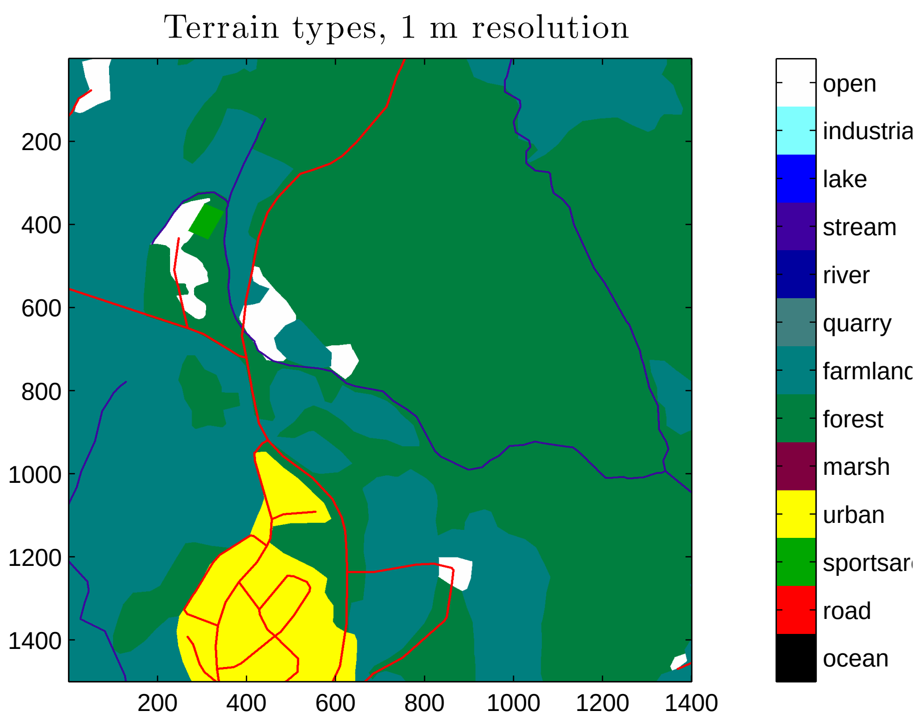


Figure 5.10: Terrain types for the same area as previously

Combining terrain types and slopes

With the method from (5.1), the terrain speed data, and the slopes can be combined into a new map consisting with the new speed values. For the area previously shown for the height data, the terrain types are given in figure 5.10. Using this terrain data, the combined speed data is shown in figure 5.11. This map of the speeds for a terrain vehicle will be the basis for creating a graph that can be used for path planning. The creation of this speed map is implemented in Code B.4.

5.3 Path planning implementations

The implementation of a graph structure is straightforward, and details on how this can be done can be found in [6]. The graph implementation in MATLAB that is used is therefore not included in this report. The implementation of the A* algorithm in MATLAB is included in the appendix as Code B.2. The version shown is assuming a 4 edge graph structure, but a version using an 8 edge graph structure has also been implemented. The differences are minor, and it is therefore not included here.

The essence of the hierarchical path planning is the implementation of the

```
createMask(graph, path, radius)
```

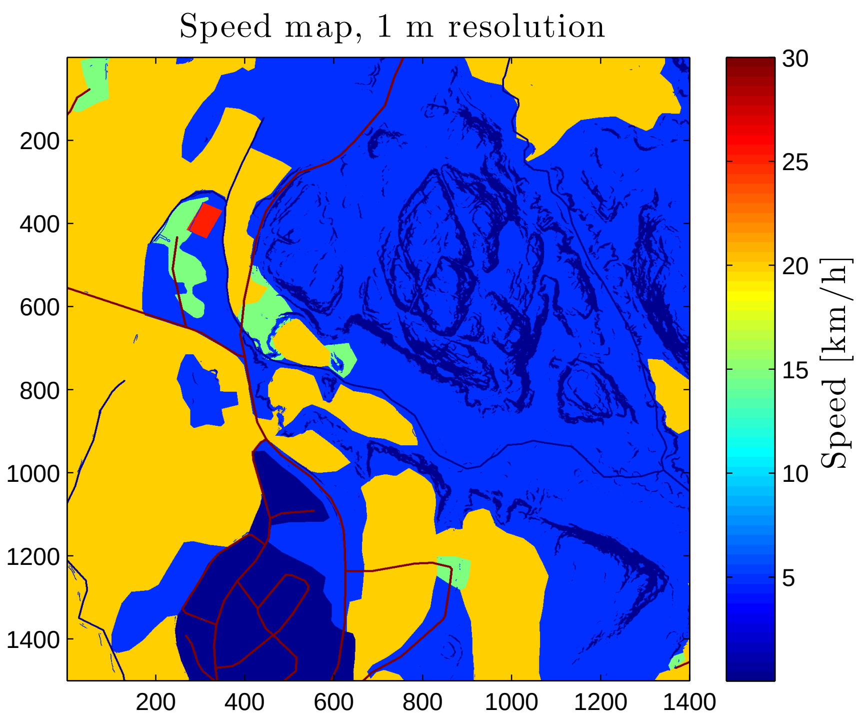


Figure 5.11: Terrain and slopes data combined into speed map

function in Code B.7. Given a graph and a path in this graph, and a desired path radius, this function creates a mask of the area within the given radius of the specified area. This mask is then used for creating a graph, only including the masked areas. This is supported directly by the graph class, and all other processes such as terrain analysis and path planning are exactly as before.

Performing hierarchical path planning is therefore achieved by performing path planning in a coarse terrain, use the path found to create a mask containing the area surrounding this, and using this mask to create a new graph in finer terrain for the area covered by the mask. Performing path planning in this new graph then gives a hierarchical path.

Chapter 6

Experiments

In chapter 2, a framework for performing automatic terrain analysis and path planning was proposed. In chapter 5, this framework was used with terrain data to make path planning experiments with different terrain resolutions, using both elevation and terrain type data. Investigating how path planning compares between the different terrain resolutions is an experiment that can provide insight into how the different terrain resolutions can be combined using hierarchical path planning. Experiments will therefore be conducted to investigate this.

In chapter 3, a method for performing hierarchical path planning using multiple terrain resolutions was proposed. The method has the potential of improving the computation performance significantly over conventional path planning. The method uses the highest terrain resolution available for path planning, while at the same time working with a reduced data set for increased performance. The method will be investigated in this chapter with a series of experiments to see if an increase performance can be achieved while maintaining near-optimal paths.

This chapter contains two experiments, where the second builds on the first one:

1. Optimal path planning in 1 m, 2 m and 10 m terrain resolutions.
2. Hierarchical path planning in 1 m terrain, using 2 m and 10 m optimal paths.

The first experiment is a comparison of the optimal paths in each of the terrain resolutions, and serves to provide a baseline for use for the hierarchical path planning experiments. The second experiment is the main experiment for testing the hierarchical path planning. The same cases as in experiment 1 is used for finding hierarchical paths, and the hierarchical paths will be compared to the optimal paths.

The terrain available for experiments is a 26×26 km area covering Larvik, Sandefjord and the surrounding areas. A full view of the area can be seen in figure 6.1. The terrain features large farmland areas near the coast, and more forest further away from the coast. Two large residential areas can be seen. The southern most is Larvik, while the one a little east of the center is Sandefjord.

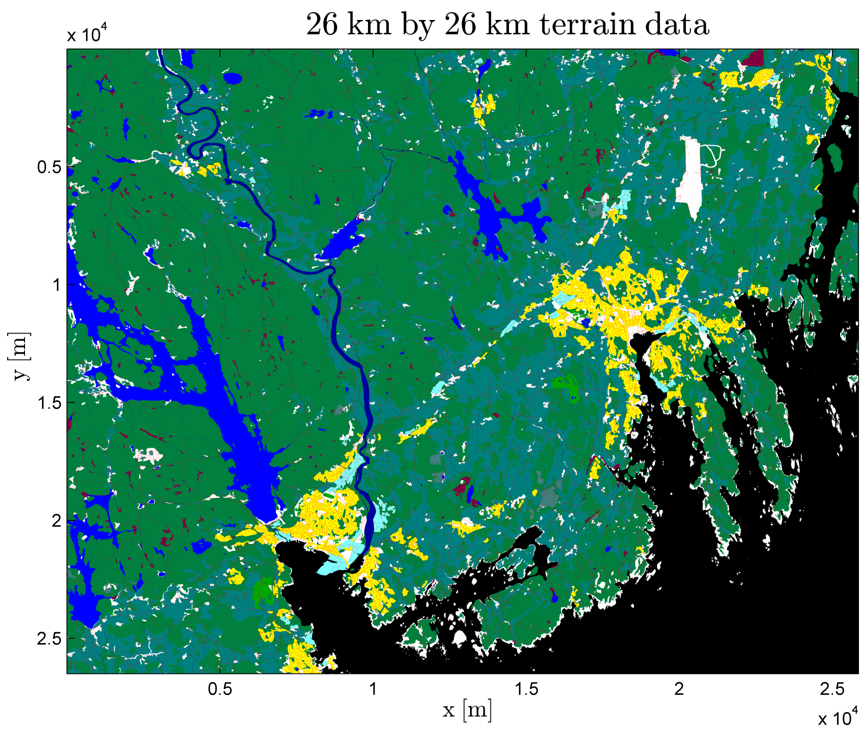


Figure 6.1: Full 26 km by 26 km terrain

Heights data for the same area is also provided. Terrain analysis and path planning is implemented according to chapter 5. The system used for simulations is a Dell Optiplex 9020, with Intel Core i7 3.6 GHz processor and 16 GB RAM.

All results from experiments are presented in boxplots. This allows detailed information to be presented in a simple way. The red center mark is the median of the data sets. The lower and upper bounds of the blue box marks the 25th and 75th percentile, which means 50 % of the data set is within these boxes. The shiskers extend to the most extreme data points not considered outliers, based on the data set. Outliers are plotted individually as red marks.

6.1 Experiment 1: Path planning in different terrain resolutions

The first experiment is an experiment that will form a basis for performing hierarchical path planning. The experiment will take a look at the differences in paths found in all the resolutions. This will also provide insight into some of the questions raised in the chapter 5.

Initially, we want to do path planning in each of the terrains resolutions with the same start and end points. From this simple experiment, we will compare the optimal paths in each of the terrain resolutions. What is of interest is how far apart the paths are and if there is any difference in the optimal paths, which path is the shorter one. Trials here will repeated many times over, with randomly selected start and end points. This way, we can use statistics to see if the results are consistent and if not not, how much variation is there in the data. Also important is to establish the computation time for each of the terrains, to see how much increase in computation time changes as terrain resolution will increase.

6.1.1 First trial: 1 m, 2 m and 10 m resolutions

For the first trial, the terrain resolutions 1 m, 2 m and 10 m is used. Path planning are done with random start and target points for paths. First, a start point is chosen at random from the whole 26×26 km area. Another point is chosen as the target point. It is selected by choosing a random direction from the point, and choosing a random distance between 2 and 4 km. The resulting point will be the target point. A subselection of the large 26×26 km terrain is chosen by finding the rectangle spanned by the start and end point in the 1 m resolution, and adding a 1000 m padding for the rectangle. To ensure that the region will be uniquely defined in the 2 m and 10 m terrain, the limits are chosen so that they are divisible by 10. All the coordinates from the 1 m terrain can therefore be converted to 2 m and 10 m terrain by dividing by 2 and 10 respectively.

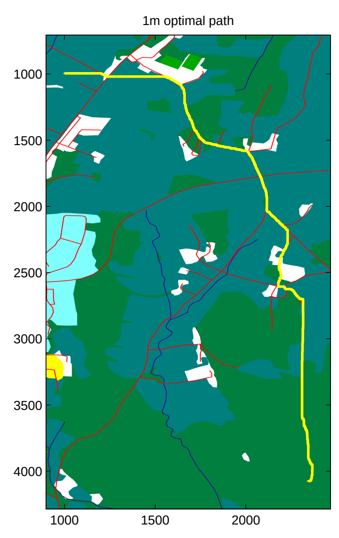
With the subregions defined in all terrain resolutions, the respective elevation and terrain types data can be used to create a speed map, as shown in figure 5.11. Generic graphs are created of the sizes of the subregions, and the speed maps are used to assign weights to the edges.

Path planning between the start and end points is used for finding the optimal paths in each of the terrain resolution. For the experiment, the trial is repeated $N = 30$ times. Paths from one of the cases are shown in figures 6.2a, 6.2b and 6.2c. Here, the optimal paths in all terrains are almost exactly the same. This is a good illustration of a case where hierarchical path planning would work very well, as the optimal path in the 1 m terrain always is within a very small radius of both the 2 m terrain and 10 m terrain.

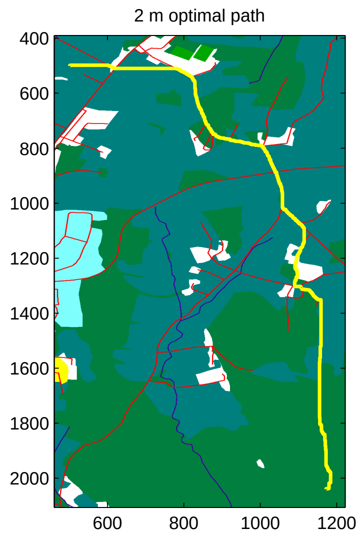
Another example where hierarchical path planning would not work as well can be seen in figure 6.3. Here, the optimal paths in the 1 m terrain and 10 m terrain are located very far apart. Any radius around the 10 m path that would cover the 1 m path would be so large that it would cover most of the original terrain selection, making the hierarchical approach essentially equal to the conventional path planning. Despite the large distance between the paths, it could still be possible that a near optimal path could be found in the neighborhood of the 10 m path. Further investigations into this can be seen in section 6.2.

Computation time

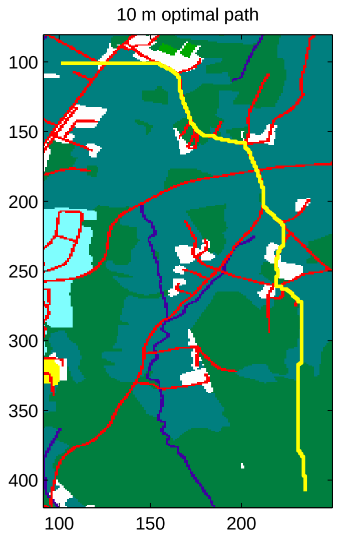
The first variable measured is the computation time. The time measured the time used for the actual path planning algorithm to run. The computation times for each of the terrains are given in figure 6.4. There is large spreads in the computation times, but as the start



(a) 1 m optimal path

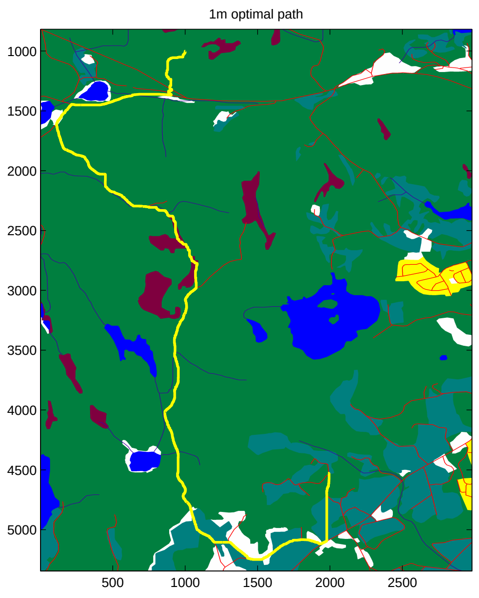


(b) 2 m optimal path

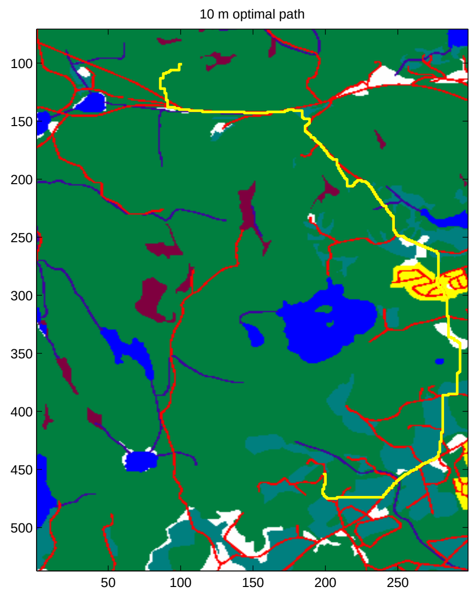


(c) 10 m optimal path

Figure 6.2: Close to identical optimal paths in 1 m, 2 m, and 10 m terrain resolutions.

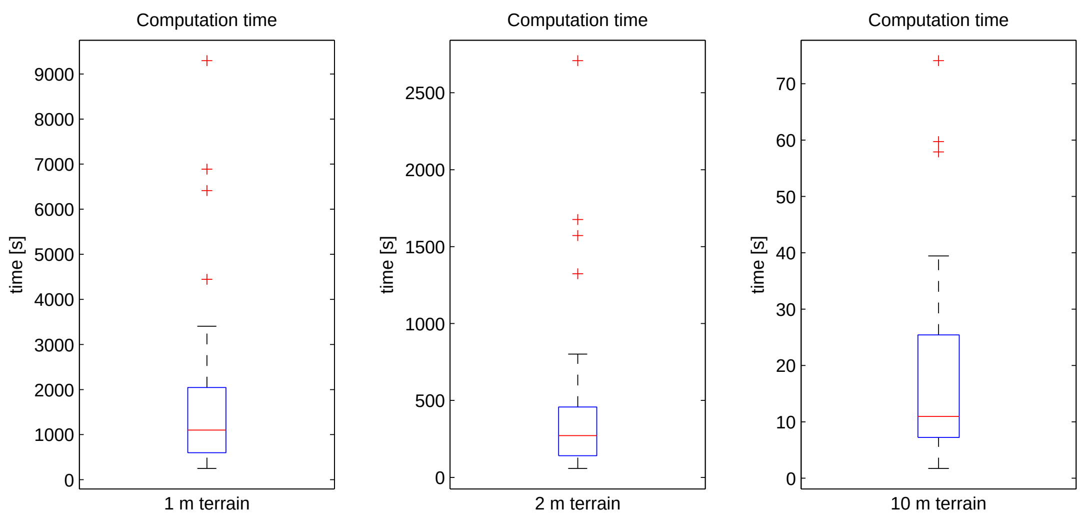


(a) 1 m resolution



(b) 10 m resolution

Figure 6.3: Optimal paths in 1 m terrain and 10 m terrain, with large distance between the paths



(a) 1 m **(b)** 2 m **(c)** 10 m

Figure 6.4: Computation times of optimal paths in 1 m, 2 m and 10 m resolutions. $N = 30$ cases.

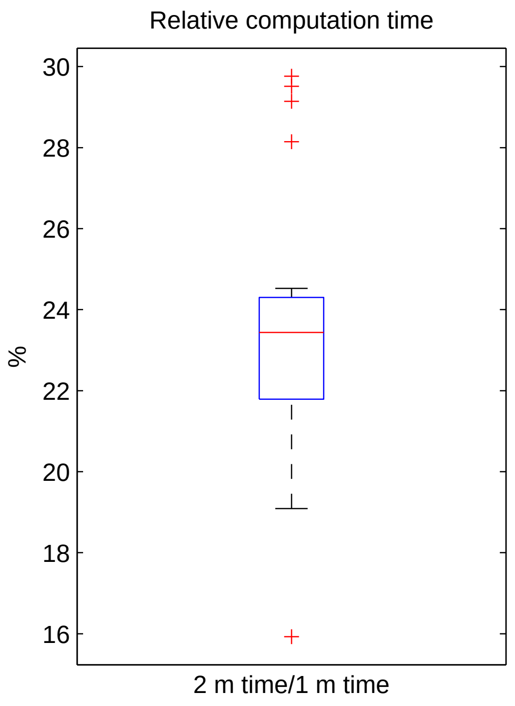
and endpoints are chosen randomly, the computation times for each individual case is not expected to match the others which causes the spread. The figure illustrates well the large differences in computation time between the terrains, with the 1 m terrain time almost 100 times larger than the 10 m computation time.

In figure 6.5, the relative computation times of the 2 m and 10 m optimal paths compared to the 1 m optimal paths is shown for each path planning case. The 2 m optimal paths has computation times around 22-24 % of the 1 m optimal paths. As seen there is a large spread in the values however, with relative computation time spanning from 16 % to 30 %. For the 10 m paths, the relative computation times are around 1 % of the 1 m computation times, with a spread spanning from 0.5 % to 2 %. An small observation that can be made is that for the given resolutions, there is a close to linear relationship between problem size and computation times. The 2 m terrain have 25 % of the number of vertices and edges of the 1 m terrain, and the 1 m terrain have 1 % of the number of vertices and edges of the 1 m terrain.

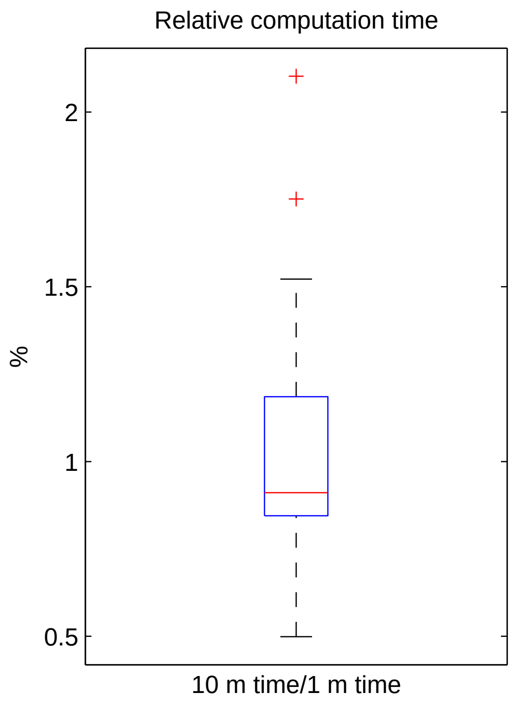
Distance between paths

The distance between paths in different terrain resolutions will be important for performing hierarchical path planning. If the distance between the optimal paths in the 1 m terrain and the lower resolution terrains are small, it will be more likely that the 1 m optimal path will be within a certain radius of the paths in the lower resolution terrains.

The distances between the 1 m optimal path and the 2 m and 10 m optimal paths is shown in figure 6.6. The distance between the 1 m and 2 m paths appear to be close to 0, and in fact turns out to be exactly 0 for all the cases. The 1 m paths when transfered to the 2 m



(a) 1 m vs 2 m



(b) 1 m vs 10 m

Figure 6.5: Relative computation times of optimal paths in 2 m and 10 m resolutions, relative to the computation time of the 1 m optimal paths. $N = 30$ cases.

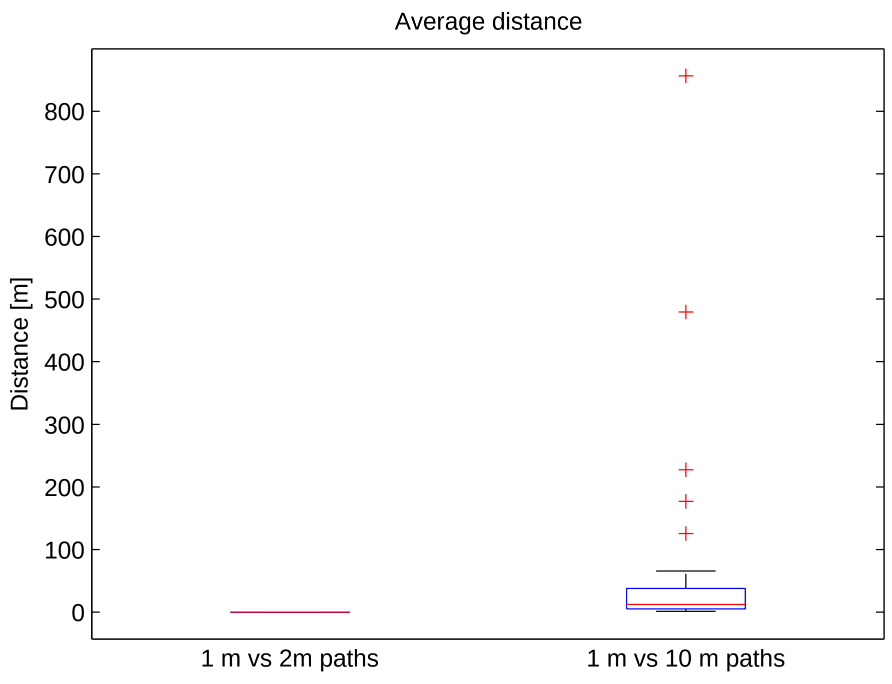


Figure 6.6: Distance between 1 m optimal path and 2 m and 10 m optimal paths. N = 30 cases.

terrain are therefore identical to the 2 m paths.

In these cases, the 1 m optimal path could be found using any radius around the 2 m optimal path. With a very small radius, the path planning in the 1 m terrain could be performed very fast, most likely much faster than the 2 m optimal path planning, and therefore give overall increase in performance. Experiment 2 will go further into this question.

The 10 m optimal paths and the 1 m optimal paths are however not 0. The median is close to 0, and in most cases are below 100 m average distance. This is an indication of cases where a fixed radius path planning could yield good results. There are however 5 outliers with large distances between the paths, which shows that paths in different terrain resolutions are not necessarily close. This can translate to suboptimal results using hierarchical path planning as optimal paths aren't in the neighborhood. How far from optimum can however not be determined from the distances alone.

Path cost

For the paths found, the total cost of a path will be the time used to traverse the path. The path costs are therefore measured in seconds. The costs of paths is therefore comparable between different resolutions. However, the terrains will necessarily be somewhat different as some speed information gets lost in lower resolutions.

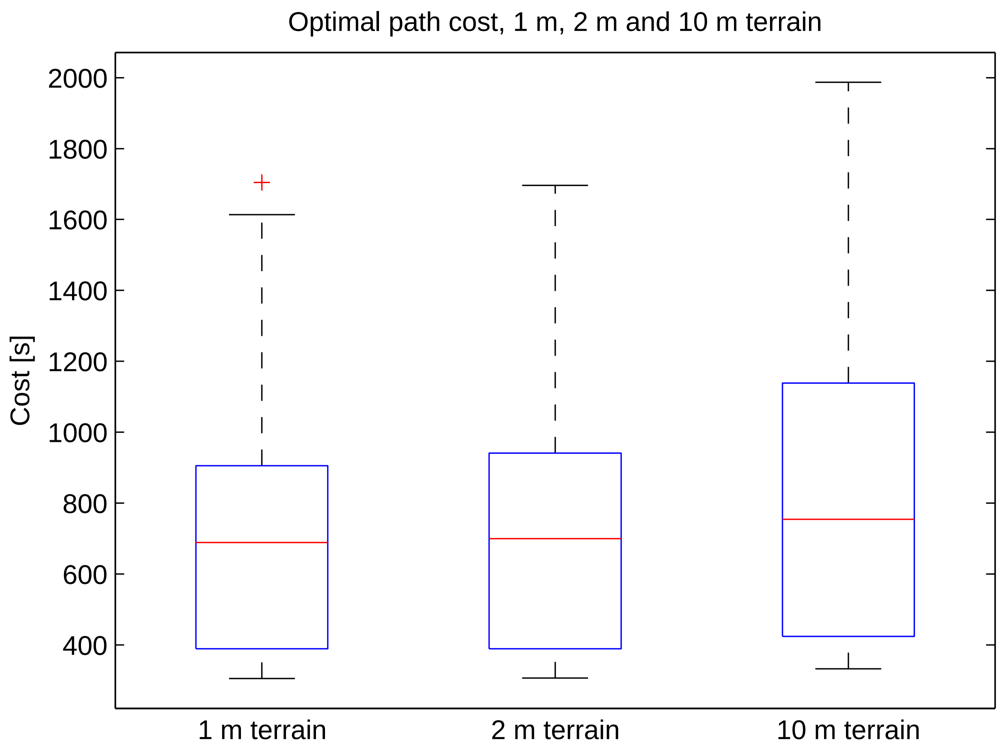


Figure 6.7: Costs of optimal paths in 1 m, 2 m and 10 m

For the paths in the first trials, the path costs are shown in figure 6.7. The 1 m and 2 m costs appear similar, while the 10 m costs are a little higher. The costs of the 2 m and 10 m relative to the 1 m optimal paths are seen in figure 6.8. The 2 m paths are close to the costs of the 1 m paths, while the 10 m path costs in general are larger, and they have a larger spread in relative increase.

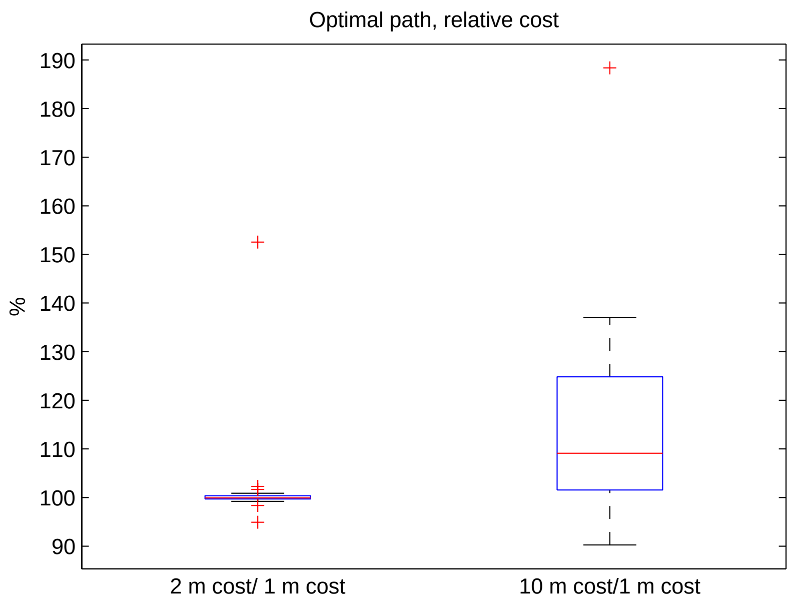


Figure 6.8: Relative cost of 2 m and 10 m optimal paths, relative to 1 m optimal path

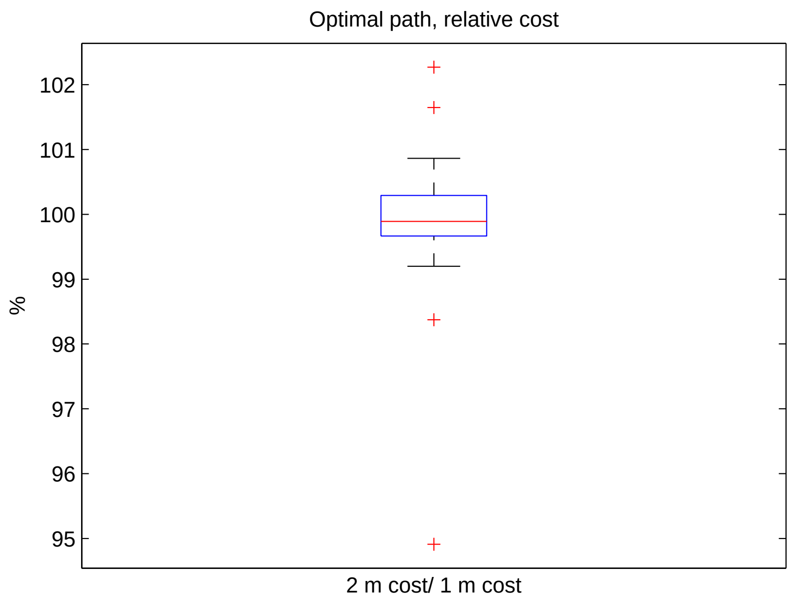


Figure 6.9: Detailed view of relative cost of 2 m optimal paths, relative to 1 m optimal path

6.2 Experiment 2: Hierarchical path planning

The second experiment will be the main experiment for testing the hierarchical path planning principles discussed in chapter 3. The experiments will use the same cases as in experiment 1, and build directly on the results found in this experiment. The path planning performed in experiment 1 found optimal paths in the 1 m, 2 m and 10 m terrain. As the path planning in the 2 m and 10 m terrain were considerably faster than the 1 m terrain, the idea is to use the paths in the 2 m and 10 m terrain and create a new graph in the 1 m terrain in the neighborhoods of these paths. This will reduce the size of the 1 m graph, which means that faster path planning can be achieved.

6.2.1 Experiment setup

For both the 2 m terrain and the 10 m terrain, graphs are created in a fixed radius distance around the corresponding optimal paths. For each of the terrain, 3 fixed radii are chosen, to see how the radii affects the results.

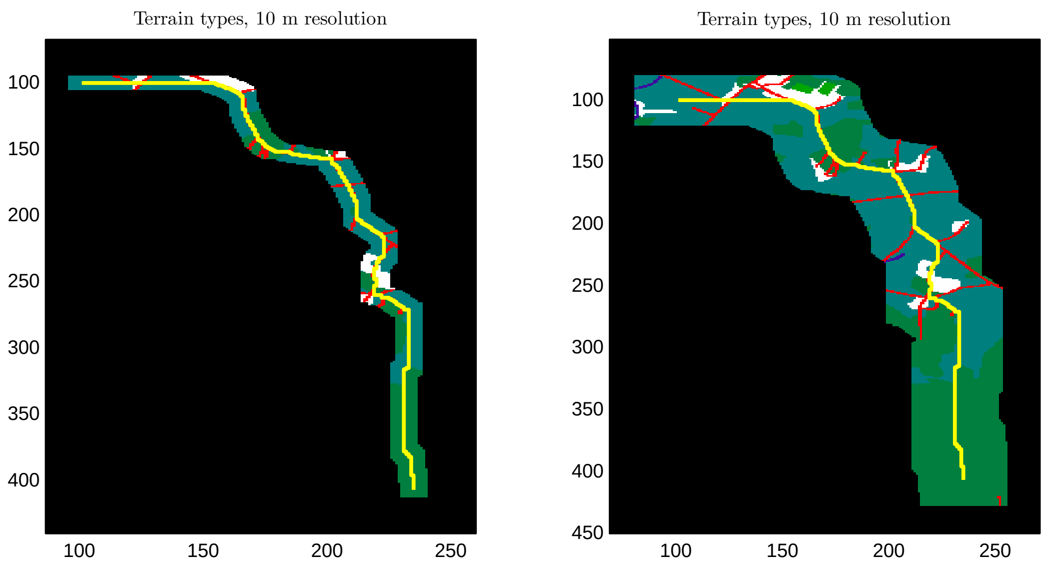
From experiment 1, it was seen that the 2 m optimal paths were close to identical to the 1 m optimal paths. A consequence of this is that it allows for a small radius to be used, as the 1 m optimal path will be within this radius. The radii chosen are therefore 10 m, 50 m and 100 m. The 10 m radius has the risk of being to small, but it can be interesting to see in how many cases the hierarchical path will be within this radius. The small radius also means a small graph, potentially making the path planning algorithm run fast. With the close distances between the paths seen in experiment 1, there does not appear to be need for any radius larger than 50 m, but the 100 m radius is included for comparison. If this is true, the 50 m and 100 m radius should yield very close results.

For the 10 m radius, the path distances were larger, but the majority had an average distance of less than 100 m. The 10 m radius would likely be to small to be of any use, and the radii is instead increased to 50 m, 100 m and 200 m. These radii will not cover all the distances shown in figure 6.6, and in these cases it will be interesting to see how the path costs are affected.

An illustration of how the reduced graph is created using a path found in a coarse resolution terrain is shown in figure 6.10, for 50 m and 200 m radius. The large reduction in area can be seen, especially in the 50 m terrain, where the graph only covers a small fraction of the original area, shown in black.

Computation time

In addition to the path planning itself which is expected to be faster than the path planning in the full terrain, creating the reduced graphs also needs to be taken into account. These are created based on the path specific to the case, the reduced graphs can not be created a priori. The total computational cost for a hierarchical path that can be compared with an



Terrain types, 10 m resolution
Terrain types, 10 m resolution

(a) 50 m neighborhood **(b)** 200 m neighborhood

Figure 6.10: Creating graphs in fixed radius neighborhood of graphs

optimal path will therefore be given as

$$\begin{aligned}\text{Computation time} &= \text{Optimal path planning in coarse resolution}\\ &+ \text{creation of reduced graph}\\ &+ \text{path planning in reduced graph}\end{aligned} \tag{6.1}$$

Comparing the improvement in computation time is done for each specific case with the relative computation time, K_T. This is the ratio between the total computation time in (6.1) of the hierarchical path planning, and the computation time of the optimal path planning.

$$K_T = \frac{\text{Hierarchical computation time}}{\text{Optimal computation time}} \tag{6.2}$$

Path cost

The path costs are also compared in a similar way to the computation time, with the relative path cost K_C defined as

$$K_C = \frac{\text{Hierarchical path cost}}{\text{Optimal path cost}} \tag{6.3}$$

The path cost will always be equal to, or larger than 1. A relative path cost equal to 1 is the case where the hierarchical path is the optimal one.

Distance measurement

The distance between the hierarchical paths and the optimal path defined in (4.8) is also a variable that will be measured. This variable is however comparable between the cases, as it does not depend on the length of the paths, but have been normalized with respect to the path length.

6.2.2 Hierarchical paths using 2 m terrain

The hierarchical paths based on the 2 m terrain is the experiments that are expected to have the largest resemblance between the hierarchical paths and the conventional paths. This is due to the closeness between the 1 m and 2 m optimal paths from experiment 1.

Path cost

The first aspect of the hierarchical path planning is if the hierarchical paths are the optimal paths, and if not how far they are from the optimal path. The costs of the optimal paths in 1 m terrain together with the 2 m hierarchical paths is shown in figure 6.11. The 50 m and 100 m paths have at least one measurement larger than any in the optimal path. The 10 m radius appears to have lower costs than the optimal paths, but this is due to three infeasible cases where outliers with infinite cost were removed.

A box plot showing the relative increase in path costs compared to the optimal path is shown in figure 6.12. The same three outliers are removed from the 10 m radius, leaving paths with close to 0 increase in cost. The 50 m and 100 m paths also have a single outlier, while the other cases are close to zero. A detailed plot with the outlier removed is shown in figure 6.13. The 10 m radius path only have cost increases of less than 0.2 %, while the 50 m radius also have single outlier with a 1.2 % increase in path cost. This is a small increase in cost, but the other values have less than 0.1 % increase in cost, and it is therefore considered an outlier. This outlier is however not in the 100 m radius path, which only have increases of less than 0.1 % of the optimal path.

Computation times

The computation times of the 2 m hierarchical paths is shown in figure 6.14. As seen, the computation times are reduced compared to the optimal path. The reduction varies with the path radius, but all show computation times compared to the optimal path planning.

The relative computation times of the 2 m hierarchical paths compared to the 1 m optimal paths are shown in figure 6.15. The 10 m paths have computation times centered around 25-30 % of the 1 m optimal path planning. Considering that the 2 m optimal path planning had computation times of around 20-25 % of the 1 m optimal path, this is close to the lower bound of what's achievable.

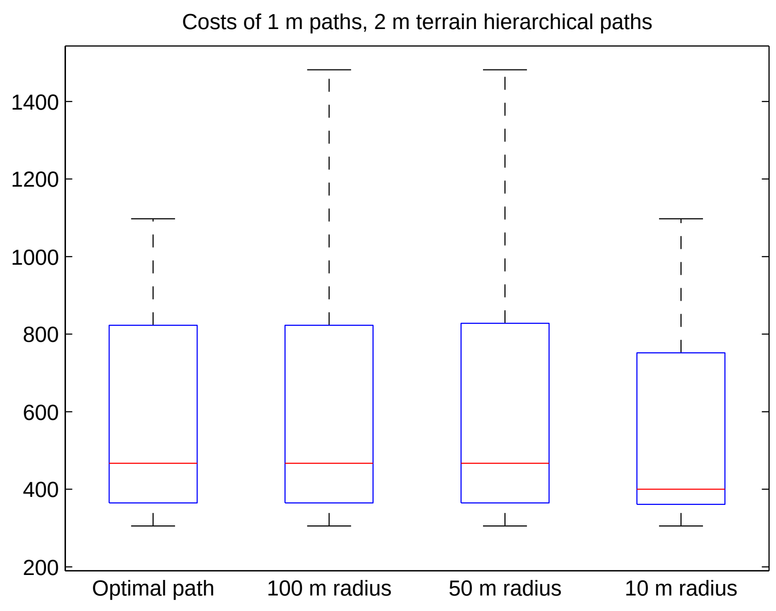


Figure 6.11: Cost of 1 m optimal path and 2 m hierarchical paths. Three outliers are removed from the 10 m radius paths, as no path were found in these cases

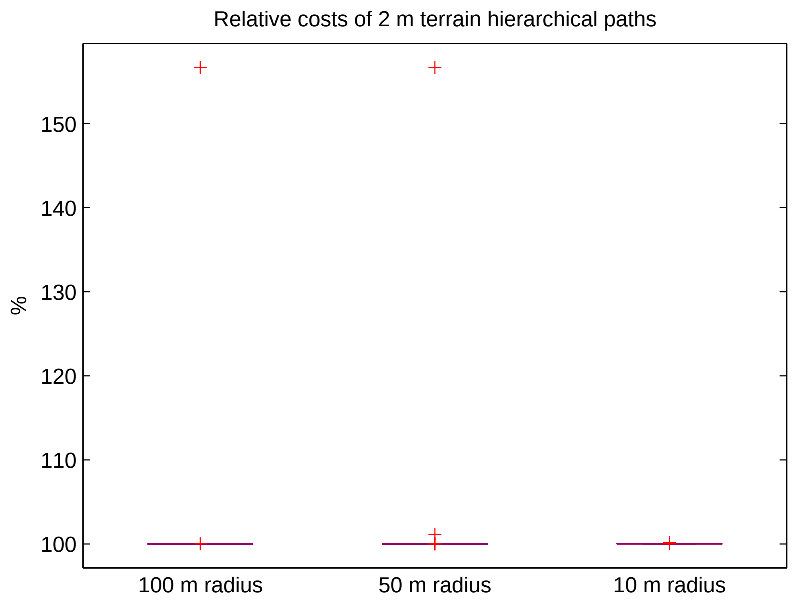


Figure 6.12: Relative costs of 2 m hierarchical paths, compared to 1 m optimal path cost. Three infeasible outliers removed from the 10 m radius path.

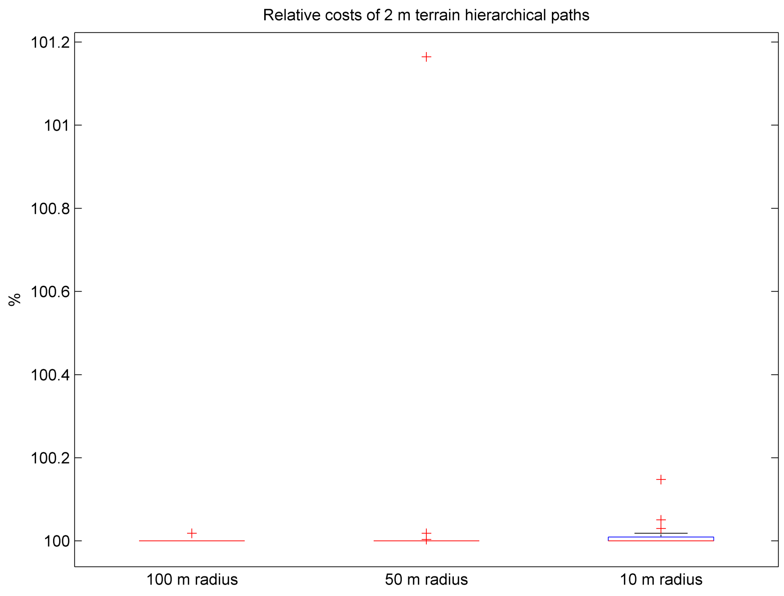


Figure 6.13: Relative costs of 2 m hierarchical paths, with outlier removed from 50 m and 100 m radius paths. and 3 outliers removed from the 10 m radius path

For the 50 and 100 m radius paths, the costs increases, with the 50 m having the median at 35 % of the optimal cost, and the 100 m radius having the median at 45 %. Both values are improvements to the optimal path planning, and with expected cost close to the optimal paths.

Path similarity

The average distances between the hierarchical paths and the optimal curves are shown in figure 6.16. The distances in most of the cases are very close to 0, but there are some outliers. For the 50 and 100 m radius, the largest outlier is also the same outlier that caused a 50 % increase in costs. However, the second outlier with average distance of 40 m can not be seen in the costs plot as an outlier. This shows that large distances does not necessarily mean differences in cost. For the 10 m radius paths, average distances close to, but not equal to 0 can be seen. The same pattern were also seen in the relative costs, but the differences were negligible with increased costs below 0.2 %. The measured distances here are therefore not important, as they lead to insignificant change.

6.2.3 Hierarchical paths using 10 m terrain

The optimal paths in the 1 m terrain and the 10 m terrain are at risk of being located further apart than the 1 m and 2 m optimal paths, as seen from figure 6.6. Yet, large distance

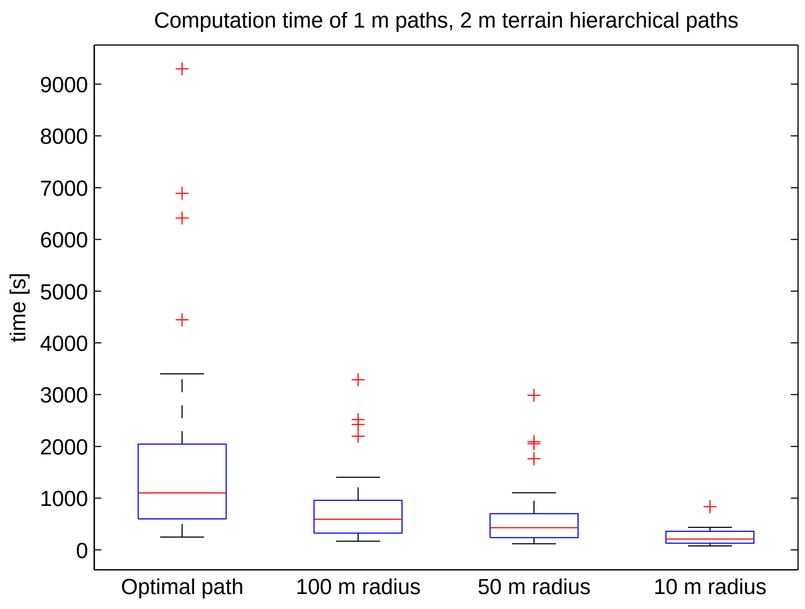


Figure 6.14: Computation time of 2 m hierarchical paths and the 2 m optimal path

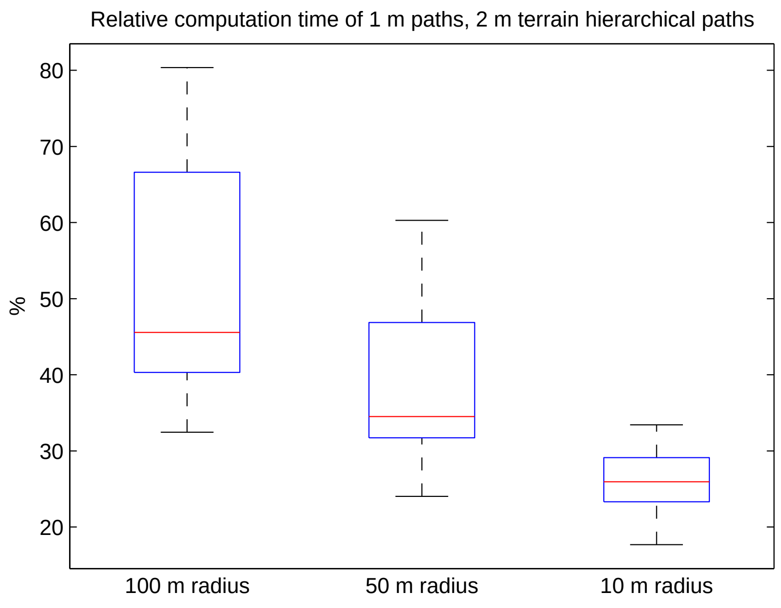


Figure 6.15: Relative computation time of the the 2 m hierarchical paths compared to the 2 m optimal path

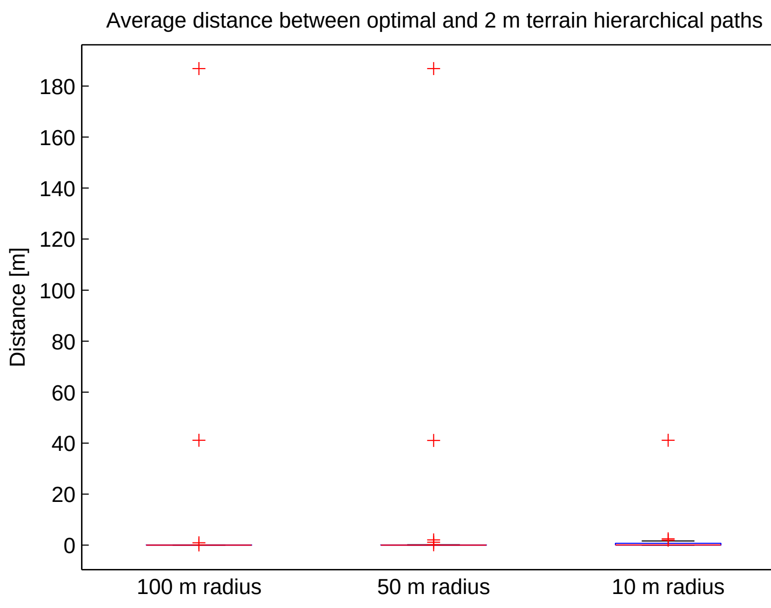


Figure 6.16: Average distance between 2 m hiearchical paths, and 1 m optimal paths

between curves does not imply large difference in cost as seen from the 2 m hierarchical paths, and the costs of the 10 m hierarchical paths can still be close to optimal.

Path costs

The path costs of the hierarchical paths based on the 10 m terrain are shown in figure 6.17. For the 50 m path costs, there are 6 paths that were infeasible with infinite path costs that were removed. For the 100 m paths, 1 path was infeasible. Besides these paths, the path cost looks similar, with a slight increase increase in path cost the smaller the radius are. The 50 m path costs appear smaller than the the other, but this is due to the removal of some of the paths from the results.

A plot of the relative path costs of the hierarchical paths compared to the optimal paths is seen in figure 6.18. The same infeasible paths are also removed here. For the 200 m path, there are a tight group of 25 paths with costs within 0.5 % of the optimal path. The remaining 5 can be seen spread with costs from 2 % to 12 % increase. For the 100 m radius, the 75th percentile is still below 2 % increase, and the median is at 0 increase in cost. Except for the infeasible paths, the other paths have increase below 12 % also. The 50 m radius have as mentioned 6 paths that were infeasible. The median is not at 0 as with the others, and the 75th percentile is higher than 3 % increase. Still, there are no measurements with higher values than 12 % increase in cost.

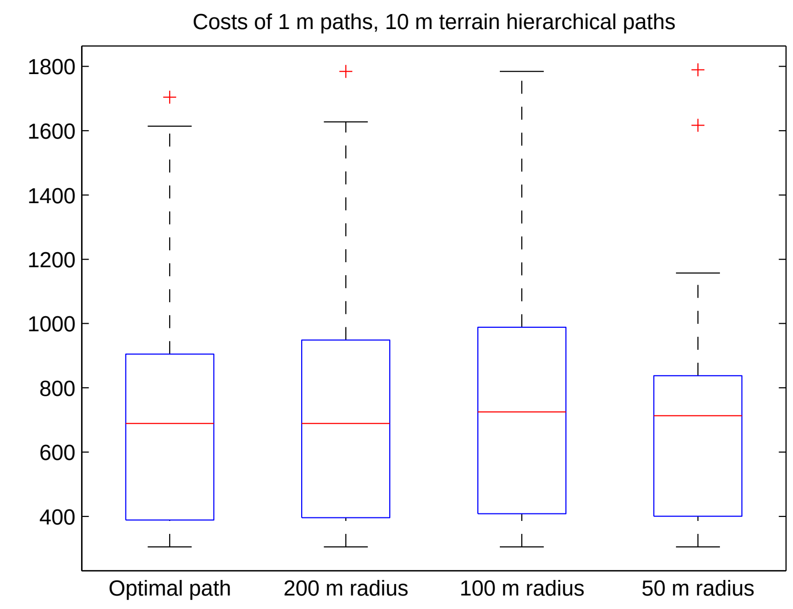


Figure 6.17: Cost of 1 m optimal path and 10 m hierarchical paths. 1 infeasible path removed from 100 m path, and 6 infeasible paths removed from 50 m path

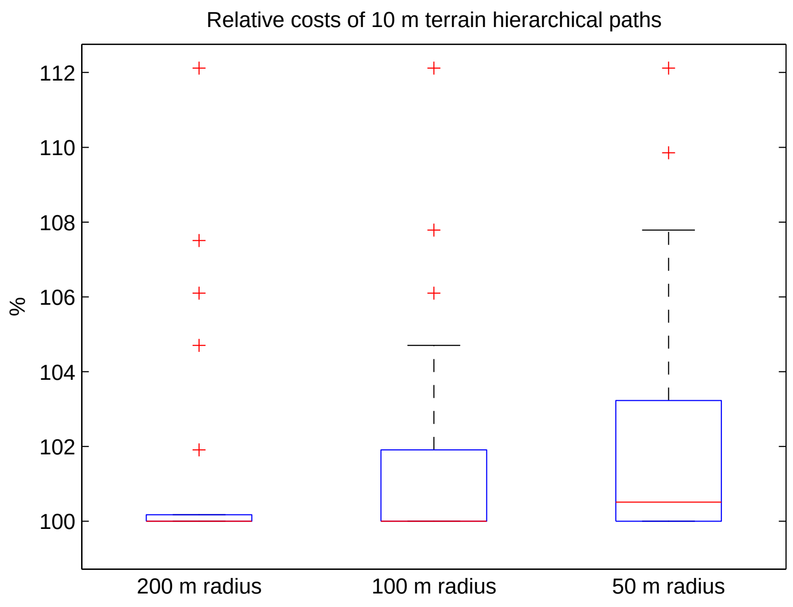


Figure 6.18: Relative cost of 10 m hierarchical paths and 1 m optimal path. 1 infeasible path removed from 100 m path, and 6 infeasible paths removed from 50 m path.

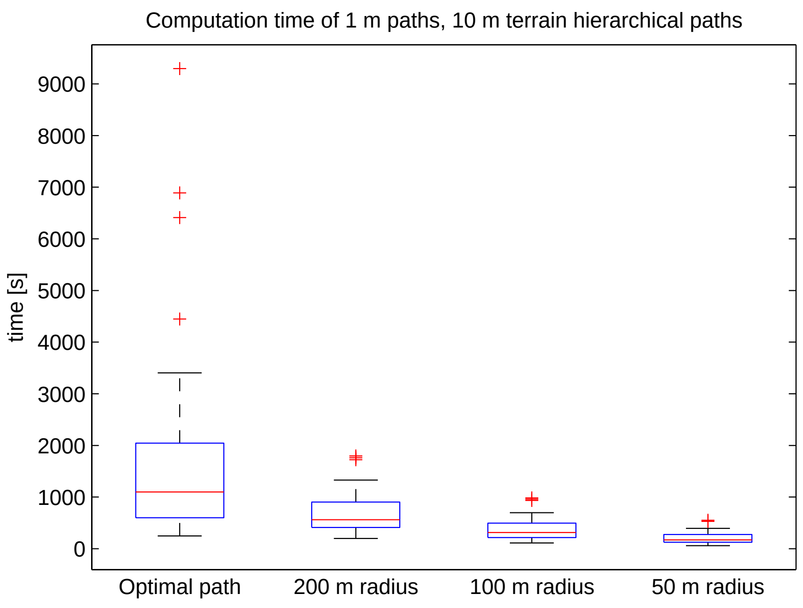


Figure 6.19: Computation time of 2 m hierarchical paths and the 2 m optimal path

Computation time

The total computation times for the 10 m hierarchical paths are shown in figure 6.19. The results look similar to the 2 m computation times. The relative computation times of the hiearchical paths compared to the optimal paths can be seen in figure 6.20. The 200 m radius have the majority of computation times between 40 % and 75 % with median at 50 % of the optimal paths. There are also computation times spanning from only 15 % to 110 %, meaning from an improvement by a factor of 6, to a computation time that is worse than the optimal path planning. The relative computation times decreases with the decrease in radius, and for the 100 m radius, the majority of the measurements are located between 20 % and 45 % with a median of approximately 30 %. There is also a further large decrease in computation times for the 50 m radius, with the majority of the measurements between 10 % and 25 % and with a median of approximately 15 %.

Path distances

The average distances between the optimal paths and the 10 m hierarchical paths are figure 6.21. Unlike the hierarchical paths from the 2 m terrain, where the paths were very close to each other with distances below 5 m except for 2 outliers, there is a large spread in the distances for the 10 m hierarchical paths. The distances does not depend on the path radii, as only minor differences can be seen between the different paths. The distances are however similar to the distances seen between the 10 m optimal paths and the 1 m optimal paths from experiment 1 and figure 6.6.

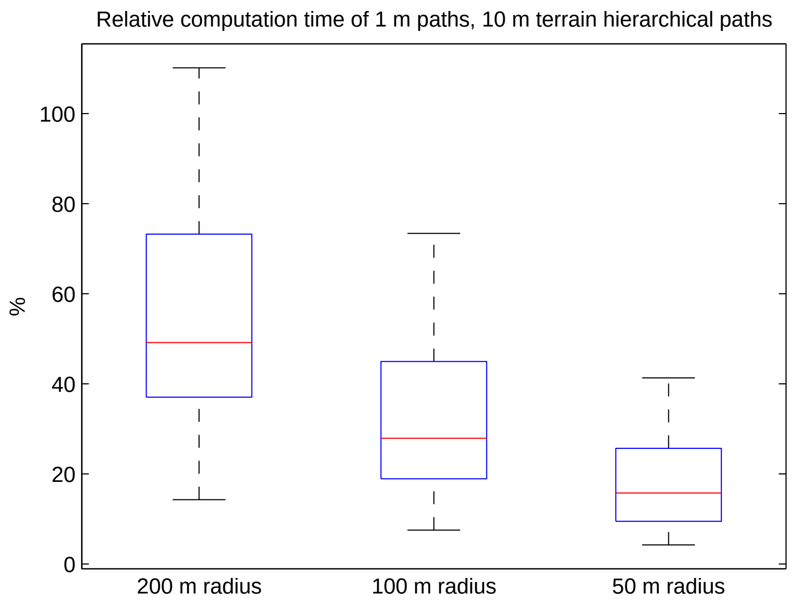


Figure 6.20: Relative computation time of 10 m hierarchical paths compared to the 2 m optimal path

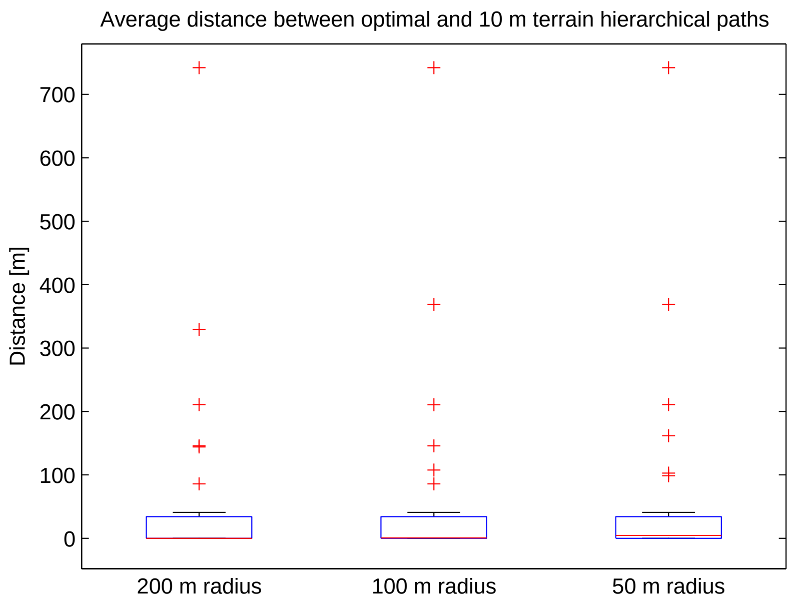


Figure 6.21: Average distance between 10 m hiearchical paths, and 1 m optimal paths

Chapter 7

Extended experiments

With the experiments of chapter 6, a basis for the hiearchical path planning has been established. There is still some possibilities for improving the results of the path planning.

In this chapter, the method of chapter 6 will be modified in two different ways to see how the method could potentially be improved.

The first experiment is an experiment that uses the methods of experiment 2, but will use only flat terrain selections. The goal of this experiment is to study the results of the hierarchical path planning in the flatter terrain, and to see if the flatter terrain has any affect on the paths in different terrain resolutions

The second experiment is to use 8-edge graphs instead of the 4-edge graph used until now. THe motivation for this experiment is given in section 2.2.2, and the hypothesis is that 8-edge graphs can lead to smaller distances between paths in different terrain resolutions, and thus improve the optimality of paths found using the method.

The third experiment is motivated by the results of the 2 m hierarchical paths of experiment 2. The 1 m and 2 m optimal paths were in most cases identical, and this can be utilized. As the 1 m path can be expected to be close to the 1 m optimal path, instead of using the 10 m terrain to find a path in the 1 m terrain neighborhood, the 2 m terrain can be used instead, yielding faster computation. The 2 m paths can thereafter be used to find the final 1 m path. The reduced size of each of the subproblems can yield lower overall computation times.

7.1 Experiment 3: Hierarchical path planning in flat terrain

The second experiment is an experiment that will investigate how the flatness of the terrain will affect the the path planning. The hypothesis to be tested is derived from section 5.2.2, where the coarser resolution terrain is missing slopes details that exist in the finer terrain.

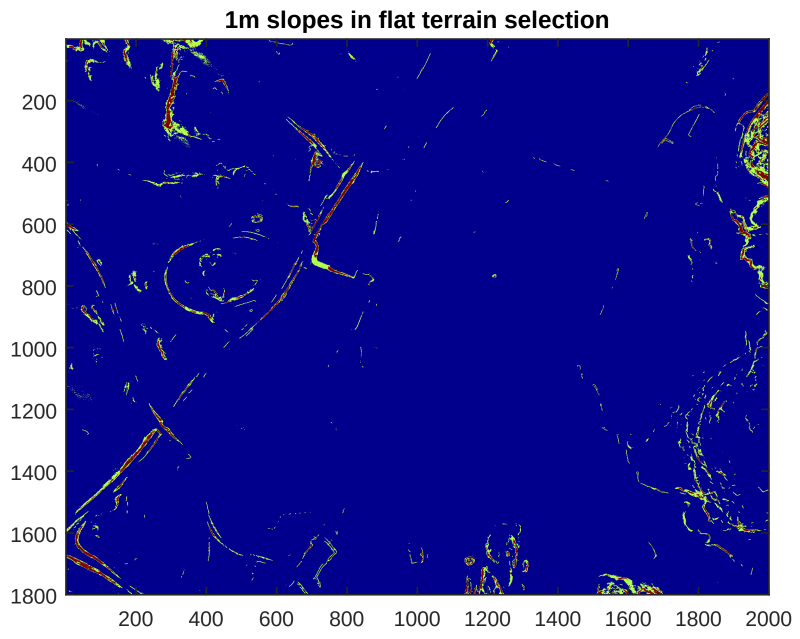


Figure 7.1: Slopes in a flat terrain selection

The hypothesis is that if the terrain is flat, the difference between the different resolution terrains will be smaller, and the cost and distance between paths in different terrain will be smaller. The terrain types don't have the same level of details, and is this is handled with the downsampling algorithm discussed in section 5.2.

7.1.1 Experiment setup

The goal of the experiment is to investigate if flat terrain will increase the optimality of the hierarchical path planning. Experiment 3 and figure 6.16 shows that the 2 m hierarchical paths in most cases are close to optimal, and little could be gained in flat terrain. The 10 m terrain hierarchical paths are further from the optimal paths, both in distance and in cost, and has the potential for improvement. This experiment will therefore compare hierarchical paths based on the 10 m terrain to the 1 m optimal paths.

The terrain selections are chosen manually from the available terrain to find the flattest areas suitable for testing. The start and target points are then chosen from within these subsections. An example on one of these selections can be seen in figure 7.1, which contains smaller areas of steep terrain compared to the example selection shown in figure 5.6.

The hierarchical radius will be set to 100 m, as this showed reasonable performance in experiment 3, both regarding time and optimality. The measured variables will be the distance between the paths and the cost for the 1 m optimal, 10 m optimal and the hierarchical path based on the 10 m optimal path. The experiment will be repeated for a total

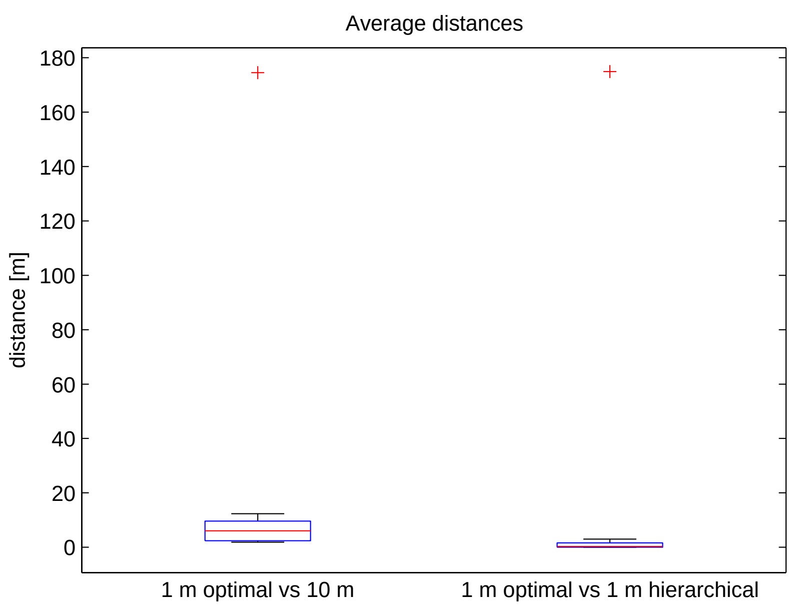


Figure 7.2: Flat terrain: all distances

of $N = 12$ cases. As these are manually selected, these cases are not used in any of the other experiments which are all chosen at random.

7.1.2 Distance between paths

The first variable measured is the distance between paths, and this can be seen in figure 7.2. As seen, there is a single outlier where the 1 m optimal path, and the 10 m optimal paths have an average distance of $D = 175$ m. The same outlier is also seen in the hierarchical paths.

A plot of the distances with the outliers removed is shown in figure 7.3. Compared to the distances of the regular hierarchical paths of figures 6.16 and 6.21, the distances between the 1 m and the 10 m optimal paths are greatly reduced, showing that the optimal paths are closer to each other in the flat terrain. The distances vary from 2 m to 12 m, all of which are well within the hierarchical path radius of 100 m. The optimal paths in the 1 m terrain is therefore expected to be within the radius of the 10 m optimal paths, which will make the hierarchical paths the optimal paths.

The distances between the 1 m optimal paths and the 1 m hierarchical paths are smaller, with all average distances at $D = 3$ or less, with the median being $D = 0$ distance. A distance of 0 means that the paths are identical, and thus the costs will be equal.

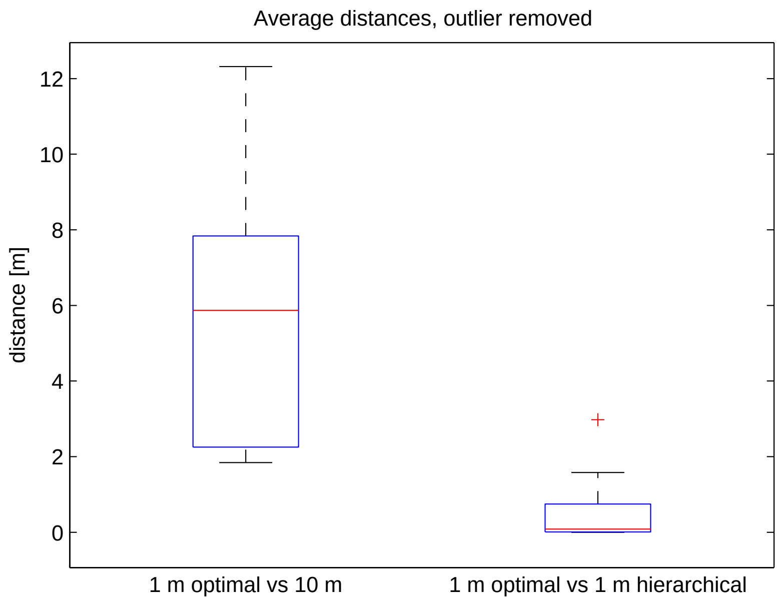


Figure 7.3: Flat terrain: distances with outlier removed

7.1.3 Relative costs

The relative costs of the 10 m optimal paths and the 1 m hierarchical paths compared to the 1 m optimal path are shown in figure 7.4. The 10 m have costs centered around 98 % - 100 % of the cost of the 1 m path costs, with some outliers further away. However, all the hierarchical path costs except for one outlier, are exactly identical to the 1 m optimal path cost. From the distances, it was seen that some of the paths were identical, but not all. This means that there are different paths that have identical costs. As discussed in section 2.2.2, different paths can be optimal with equal costs, and the paths in this experiment is a result of this.

The outlier of the hierarchical paths have an increase of 0.8 % in cost, which in practice is a negligible difference. This outlier is also the same outlier as in the distances. This shows how a large distance between curves does not necessarily mean a difference in cost.

The overall results of the experiments shows that for the flat terrain, the paths in different terrain resolutions can be expected to be close to each other. The hierarchical paths will therefore be the optimal paths in these cases. In the cases where the paths differ, the difference in cost between the terrains will be relatively small, and the hierarchical paths are expected to be close to optimal in these cases anyway.

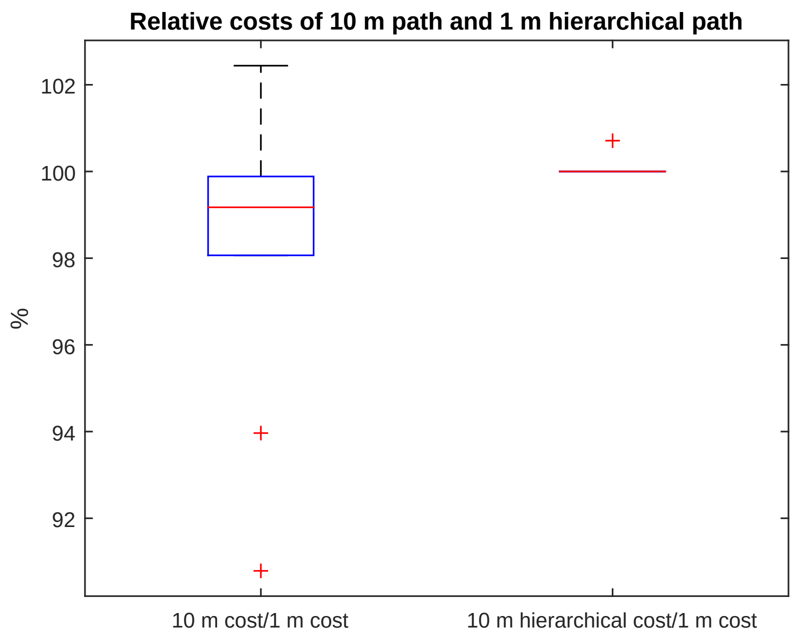


Figure 7.4: Flat terrain: Costs of the 10 m optimal path, and 1 m hierarchical path, relative to the 1 m optimal path

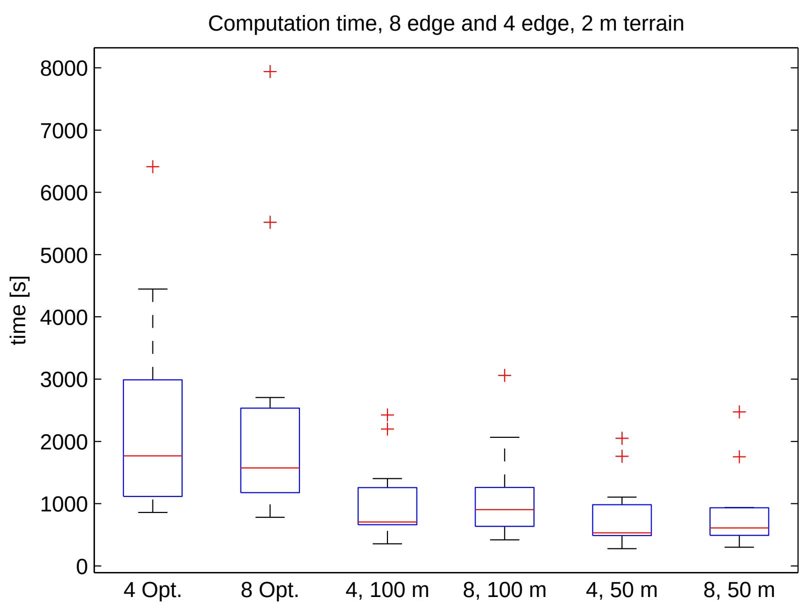


Figure 7.5: Computation times of 1 m optimal paths and 2 m hierarchical paths, 4 and 8 edge graphs.

7.2 Experiment 4: Comparison of 4 edge and 8 edge graphs

The following experiment will look at how 8 edge graphs compare to the 4 edge graphs.

The experiment is performed by creating 8 edge graphs for a subset of the cases of experiment 2, and repeating the path planning. The results of the 8 edge path planning can be compared directly with the results of experiment 2. The primary purpose of the experiment is to determine if the distance between graphs in different terrain resolutions is smaller than for the 4 edge graphs. In relation to the path distance, how the distance influence the cost of the 8 edge hierarchical paths compared to the optimal 1 m 8 edge path is also interesting. Due to the double number of edges of the 8 edge graph, the computation time for the 8 edge graph can be affected.

7.2.1 2 m terrain

The computation time for the 2 m terrain is seen in figure 7.5, both for the 4 edge terrain and for the 8 edge terrain. The 4 edge results are a subset of the results from experiment 2, while the 8 edge results are the results for the same cases with 8 edge graphs. The results show that the computation times are in general similar to the computation times of the 4 edge terrain, despite the increased number of edges.

The relative computation time of the hierarchical paths compared to the optimal paths in each of the graph types are shown in figure 7.6. The results shows that the relative com-

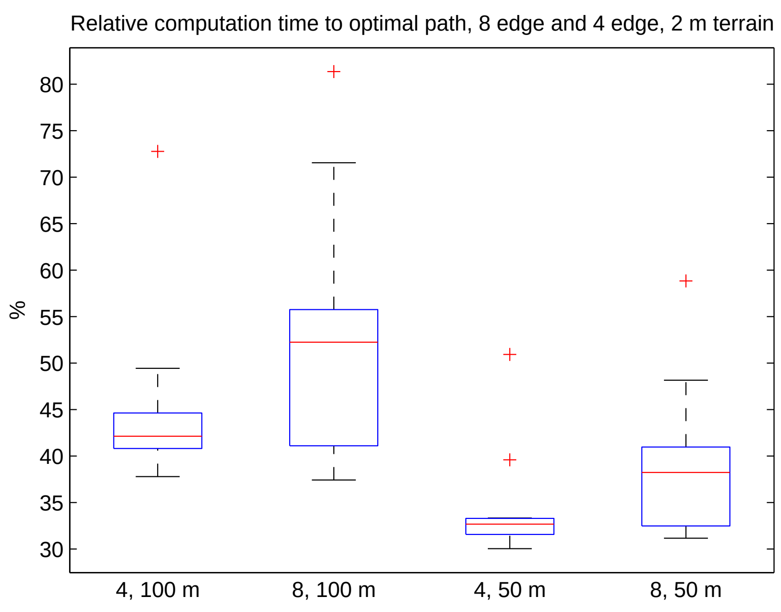


Figure 7.6: Relative computation time of 2 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs.

putation times of the 8 edge hierarchical paths are larger than the corresponding relative computation times in the 4 edge terrain. The relative computation times also have a larger spread than the 4 edge times.

In the 2 m terrain, the average distance between the hierarchical paths and the 1 m optimal paths for both the 4 and 8 edge graphs are shown in figure 7.7. The 4 edge graphs have 0 distance between the hierarchical paths, except for an outlier in both the cases which means the paths found are the optimal paths. The 8 edge paths have small distances of 10-20 m between the paths in most of the cases, which is a worse result than in the 4 edge case.

The relative cost of both the 4 edge and 8 edge hierarchical paths in the 2 m terrain are shown in figure 7.8. The results corresponds with the average distances for the graphs, which showed that the hierarchical paths were identical to the optimal paths, and therefore have has equal cost. The 8 edge paths were not identical to the optimal paths, which shows in the relative costs. The costs of the hierarchical paths are around 20 % higher than the costs of the hierarchical paths, which is a relativly large increase in cost. In none of the cases were the costs identical, which means that the optimal path were never found in the neighborhood of a 2 m optimal path. The path radius does not matter either, as the results are similar for both the 50 m and 100 m paths.

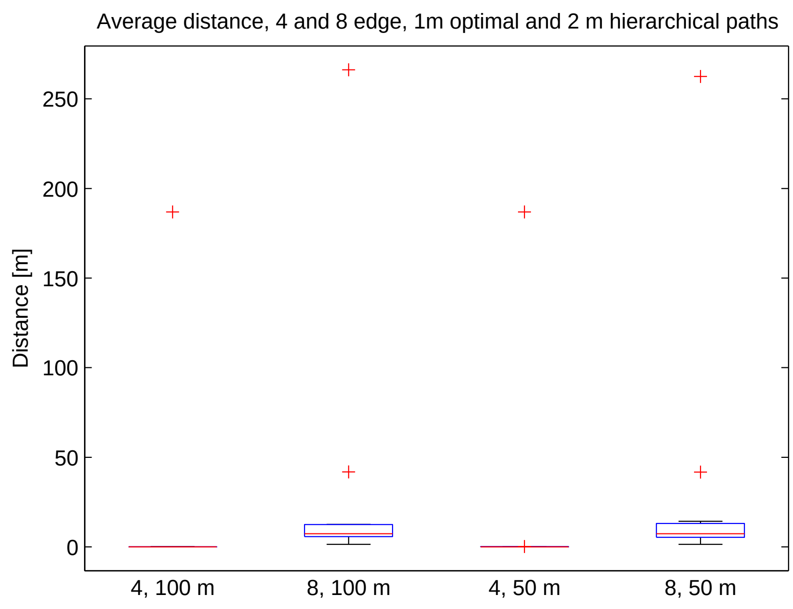


Figure 7.7: Average distance between 1 m optimal paths and 2 m hierarchical paths, 4 and 8 edge graphs

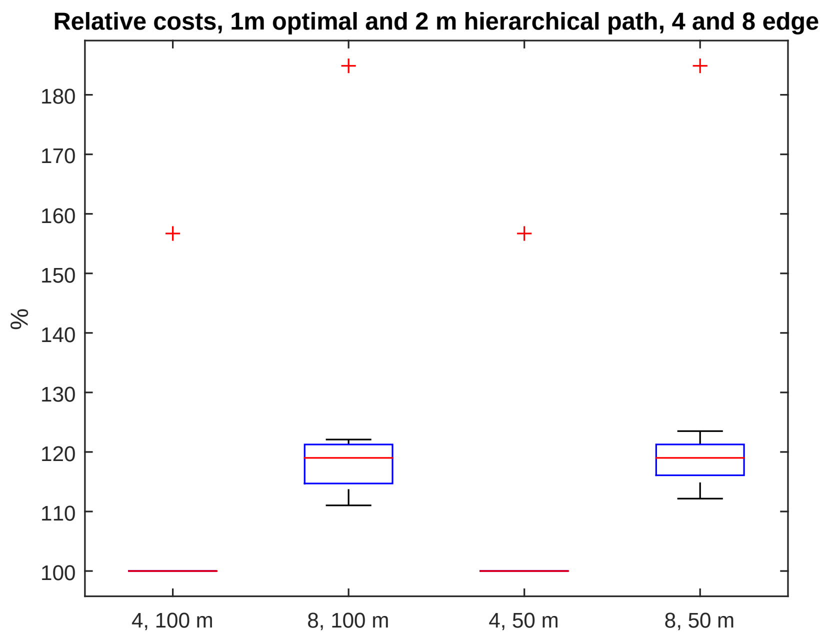


Figure 7.8: Relative costs of 2 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs.

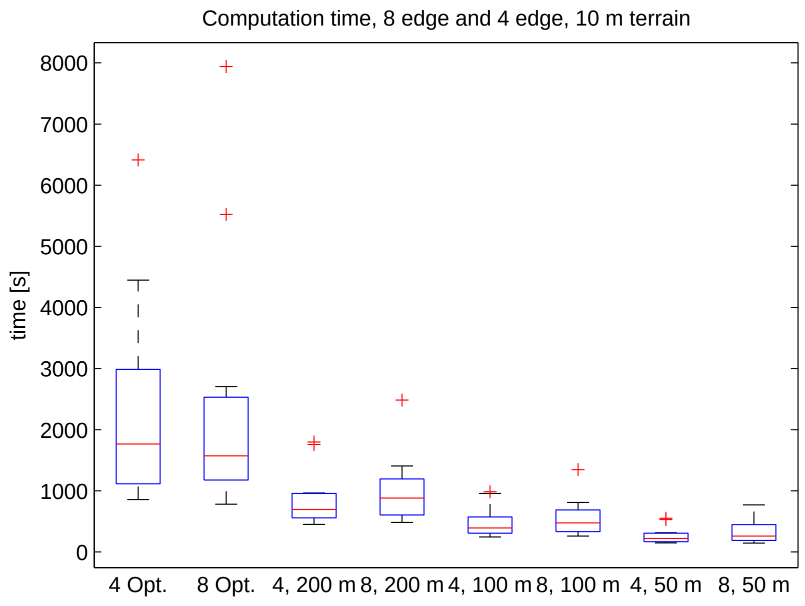


Figure 7.9: Computation times of 1 m optimal paths and 10 m hierarchical paths, 4 and 8 edge graphs.

10 m terrain

The computation times of the 10 m hierarchical paths shows the same tendencies as the 2 m hierarchical paths, with the computation times being similar for the the 4 and 8 edge graphs, with the 8 edge slightly higher. The relative computation times of the hierarchical paths compared to the optimal paths are closer for the 10 m hierarchical paths than for the 2 m terrain, and both have equal spread in the relative computation times, but the 8 edge computation are still higher than the 4 edge times.

The distances between the 10 m hierarchical paths and the 1 m optimal paths are seen in figure 7.11. The 8 edge distances are significantly smaller than for the 4 edge graphs, which in contrast with the distances for the 2 m hierarchical paths. The 4 edge graphs have distances between 0 and 200 m, while the 8 edge have distances lower than 40 m, with the median close to 0 m distance, except for two outliers.

The relative costs of the 10 m hierarchical paths for the 4 and 8 edge paths are shown in figure 7.12. As in the case of the 2 m hierarchical paths, the relative costs of the paths have relative costs of 15-20 % more than the optimal paths, with all the paths having at least 10 % increase in cost. The 4 edge paths on the other hand have costs closer to optimal, with 10 % increase in cost at highest and median less than 1 % increase from the optimum.

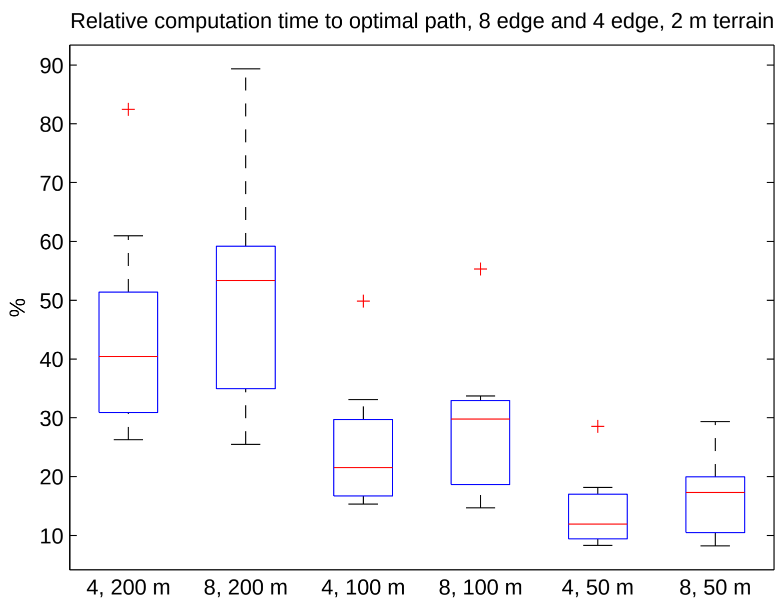


Figure 7.10: Relative computation times in 10 m terrain, 4 and 8 edge graphs.

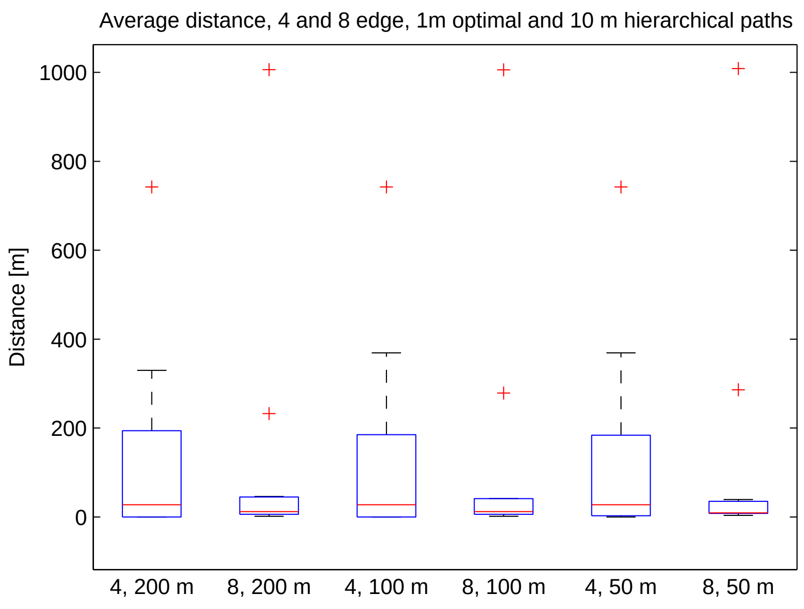


Figure 7.11: Average distance between 1 m optimal paths, and 10 m hierarchical paths, 4 and 8 edge graphs

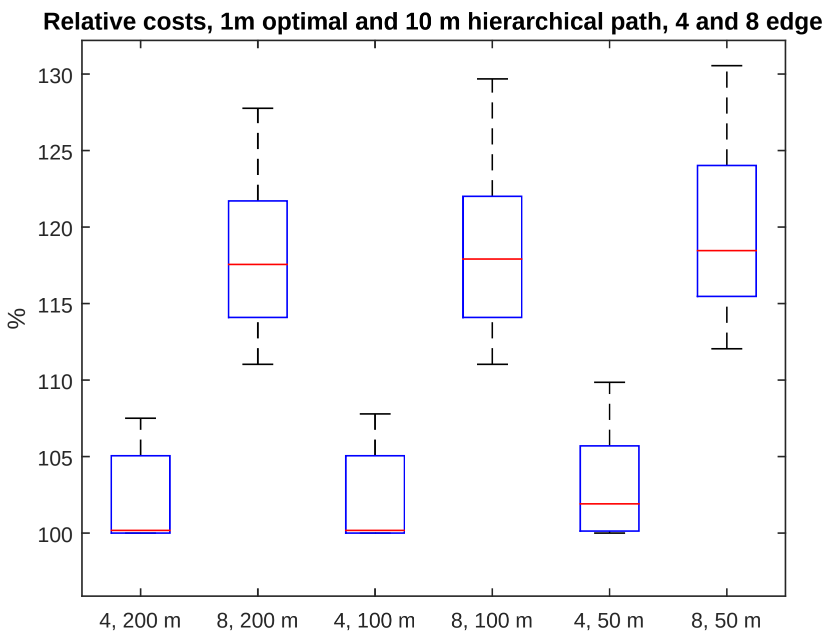


Figure 7.12: Relative costs of 10 m hierarchical paths compared to 1 m optimal paths, 4 and 8 edge graphs.

7.3 Experiment 5: Multiple level hierarchy

From experiment 1 in 6.1, it was observed that in all cases of the path planning, the 1 m optimal paths and the 2 m optimal paths were identical. Further, in experiment 3 the consequence of this observation was seen, as the hierarchical paths based on the 2 m terrain with any fixed radius gives the optimal path. In the same experiments, the reduction in computation cost was not large. The optimal path planning in the 2 m terrain has a computation cost of approximately $\frac{1}{4}$ of the path planning in the 1 m terrain. In addition to this computation cost, there is also a cost of creating the hierarchical graphs, and as this cost depends on the size of the path radius and can be high. On top of that, the path planning in the reduced 1 m terrain is performed, taking a significantly amount of time. There is a significant reduction in time, but this can be improved.

As seen, all the hierarchical paths are optimal, and the increased radius is therefore not needed. With the 10 m radius, the creation of the 1 m terrain and the path planning in the reduced 1 m graphs is quick compared to the 2 m path planning. The overall reduction in computation time is significant, and from the experiments the paths are highly reliable. The 10 m hierarchical path planning is more unreliable, but it is in the order of 100 times faster than the 1 m path planning, as seen in experiment 1, figure 6.5a.

There are two important effects that can be drawn from this

- Similarity between 1 m and 2 m optimal paths
- Fast calculation of 10 m optimal paths

7.3.1 Combining 10 m, 2 m and 1 m terrains

A method for utilizing these effects, is to combine all three terrain resolutions. First a coarse path can be found in the 10 m terrain. From the results of 6.2, this is known to be quick to perform compared to path planning in other terrains. A hierarchical graph in the 1 m terrain can be created covering a fixed radius around this hierarchical path. This radius can be chosen to be very small as the 1 m and 2 m paths are expected to be similar. The final graph in the 1 m terrain will therefore cover a small area, and thus be fast to use for path planning.

If the statement about the closeness of the 1 m and 2 m optimal paths holds, it can be expected that this approach will yield as good results as the 10 m direct hierarchical approach. If the increased hierarchy level leads to increased time consumption, or if the proposed method is faster than the traditional approach can be tested.

The experiment is setup to use a selection of the already tested cases of experiments 1 and 2. There will be a total of $N = 10$ cases run for obtaining the results. The creation of the 2 m terrain hierarchical graph is done using the radii from experiment 2, 50 m, 100 m and 200 m. The radius of the second level in the hierarchy is chosen to be 20 m. It is more likely that any deviations from the optimal path will arise at the hierarchy level above, and thus the radius is set low to investigate how much of an improvement in computation time can be achieved.

As the distance between paths is of less importance for this experiment, this variable will not be measured. The variables measured are therefore the time consumption, which is the objective of the experiment to reduce, and the relative cost of the paths.

7.3.2 Results

The computation times of the experiments are shown in figure 7.13, and the relative improvement of the method compared to the 10 m hierarchical paths are shown in figure 7.14. The increase in performance is largest for the 200 m radius paths, with the computation time running at 40-45 % of the original time, which is a significant improvement. This improvement is not relative to the 1 m optimal path planning, and compared to that would be even larger.

The relative improvements decreases with decreasing path radius. At 100 m, running times of 55-60 % is still a significant improvement, but at 50 m radius, the running times are at 80 %, and the point of the multiple level hierarchy is starting to diminish. It is nonetheless an improvement, and it has the fastest computations of all the cases.

The costs of the multiple level paths relative to the 10 m hierarchical paths are shown in figure 7.15. As predicted, the costs of the multiple level paths are identical to the paths found with the hierarchical 10 m radius path planning. There is one outlier in the 50 m radius terrain however, but the difference in cost is small at only 0.8 %.

The results shows that the use of multiple level hierarchies as used in these tests, in fact can yield an increase in computation performance while maintaining optimality.

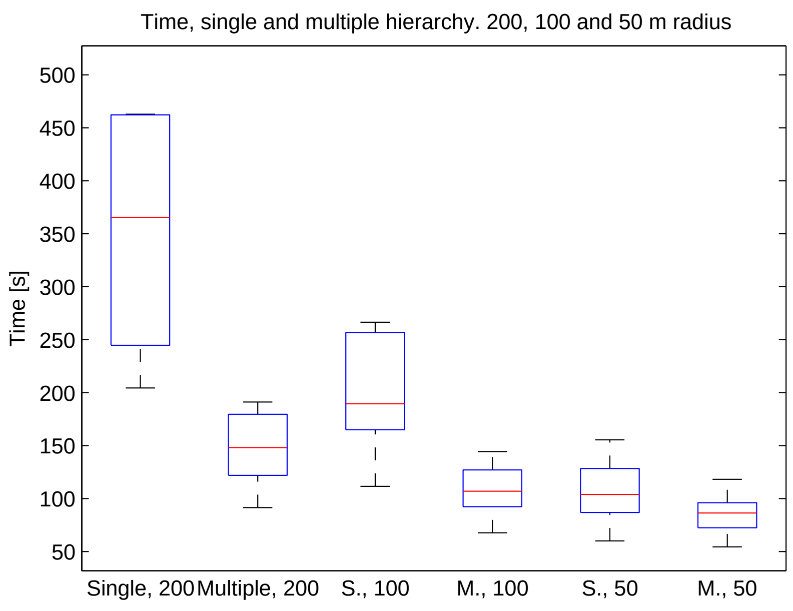


Figure 7.13: Computation time of multiple level paths and 10 m hierarchy paths. Single (S) denotes the 10 m hierarchical path, and multiple (M) denotes the paths multiple level paths.

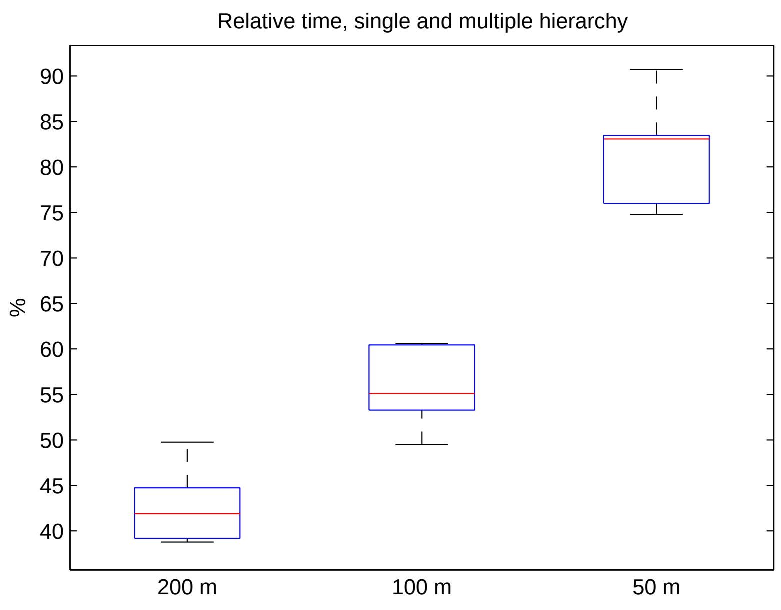


Figure 7.14: Relative computation time of multiple level paths compared to single 10 m hierarchy paths.

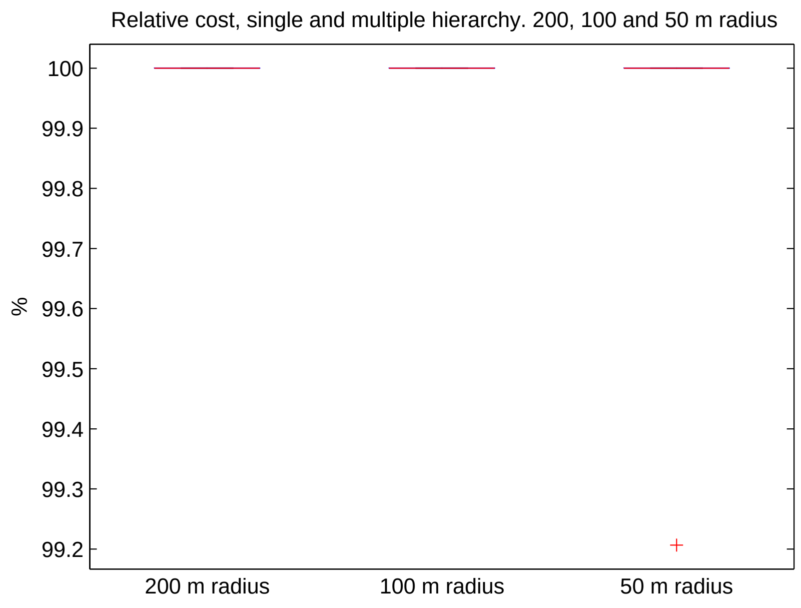


Figure 7.15: Relative cost of multiple level paths compared to 10 m hierarchical paths

Chapter 8

Discussion

In this chapter, a discussion of the results of the 5 experiments is done. It includes a review of the results and discussions about limitations of the results, and what parts can be generalized.

8.1 Hierarchical path planning

The computation times shown in experiment 1 shows how computationally expensive path planning problems can be, with the longest computation times of more than 2 hours. This shows the importance of either smaller problems or faster algorithms. For an autonomous ground vehicle that will have to perform path planning in real time, the time available for computation can be limited. By increasing the performance of the path planning, more computation time can be freed to other tasks, or the path planner can potentially deal with larger problem sizes.

8.1.1 2 m hierarchical path planning

In the terrain data given in this experiment, the closeness of the 1 m and 2 m optimal interesting, as it is the best possible case for performing hierarchical path planning where either the entire 1 m optimal path is within a neighborhood of a 2 m optimal graph.

The closeness of the 1 m and 2 m paths leads to good results with regard to optimality of paths. This is partly be due to the fact that the size difference between the 1 m and 2 m terrain is small, especially compared to the 1 m and 10 m terrains. Another point could be that the provided terrain data does not feature many details smaller than 2 m that would be lost from the 1 m terrain to the 2 m terrain. This could be due to the physical terrain which perhaps does not have many features like this, and this not have to hold for other terrain selections. Other possible error sources could be preprocessing of the terrain data which could affect features of the terrain that can't be controlled.

Despite the good results, there was an outlier in the path costs, where a path had almost 60 % increase in path cost for the hierarchical path compared to the optimal path. This is a single case, and no other cases come close to it. This illustrates that even though many trials points to a certain result, there are never any guarantees that it will always work.

A consequence of this is that any definite conclusions for example regarding path radii in general should be avoided, as this can depend on the terrain data. In these experiments, all the radii gave close to optimal paths, but for example using a 10 m radius for the path planning should not be done without knowing beforehand if it should work. The reason for the choice were the observed distances between the optimal paths, but if none such tests can be performed, larger radius paths will be a safer choice.

The reduction in computation time were not a large reduction, but all the tested radii gave expected computation of less than half of the optimal path planning, and lower for the smaller radii. Considering how safe the 2 m paths were for this terrain, the 2 m hierarchical path planning could be a good alternative. Even with the inclusion of the large deviation of the outlier, the expected value will be close to 0 increase in path cost.

8.1.2 10 m hierarchical path planning

The 10 m hierarchical paths are more interesting to analyze than the 2 m paths due to the larger deviations. Where the 2 m always are close to the 1 m paths, this is not the case for the 10 m paths. However, as seen from the results of the experiment, it is possible with large distances between paths and while they still have approximately the same costs.

For the 10 m terrain paths, the effect of the radius was also seen clearer. The computation time increase with the increase in path radius as expected like with the 2 m paths. Unlike the 2 m paths, the optimality of the 10 m paths were affected by the radius, with the path cost decreasing with increasing path radius.

All radii had paths with outliers in the cost of up to 12 % increase. For most of the cases however, the costs were within 0.5 %, 2 % and 3 % of the optimum for the 200 m, 100 m and 50 m radius respectively. These values can be considered acceptable for all practical purposes. When taking the outliers into account, the expected increase in cost will still be relatively low.

8.1.3 Infeasible hierarchical paths

An issue observed both for the 2 m hierarchical paths and the 10 m hierarchical paths was the occurrence of infeasible paths. The A* algorithm guarantees a solution if it exist[10], as the infeasible paths in reality are regular paths that contains at least one edge with weight many orders of magnitude larger than feasible edges. Any feasible path will therefore have a lower total cost, and A* will return such a path if it exists. The infeasible paths will therefore be cases where no feasible path exist in the given fixed radius neighborhood around a higher level path.

As the costs is infinite in these cases, they can be identified. In such a case, the graph can be extended to include a larger radius neighborhood, and repeat the path planning. With the A* algorithm, this can be modified to allow the existing path planning to be reused for the extended graph, and thus reducing the additional time needed.

8.1.4 Overall performance of the hierarchical path planning

From the results of experiment 1 and 2, there is a clear improvement in computation times from all the setups chosen. Largest improvement were observed using the 10 m terrain together with the 1 m terrain and using smaller radii. The computational performance should not depend much on the terrain selection used, as this depends mostly on problem size. However, given different terrains, the chosen radii or terrain resolutions might need to be changed, and this will affect the performance of the hierarchical path planning.

The results presented in experiment 1 and 2 can in general be said to be good results with regards to optimality. The 2 m hierarchical paths found are expected to find paths that is either optimal, or with marginal increases in cost. The 10 m radius, should however be avoided as it in several cases yields infeasible paths. The reason for including the 10 m path radius was the observation that in these specific cases for this specific terrain selection, the 1 m and 2 m paths were always close. This does not have to be the case in every terrain type, and there is no guarantee that this will continue to occur in this terrain selection either. In the cases where this does not occur, the paths will necessarily be not optimal. With the 50 m and 100 m, the increased terrain size leaves more room for finding the 1 m hierarchical path, and are more flexible.

The use of the 50 m and 100 m radii yielded approximately equal results, while the 100 m radius had approximately twice the computation time. The experiments also shows that all radii are at risk of having outliers with large increases on cost. It could have been interesting to also have included a 200 m radius for the experiments, but it would not have contributed significantly to the results for two reasons. First of all, all experiments points to an approximate linear relationship between problem size and computation time. A 200 m radius path would therefore be close to the 1 m optimal path planning in performance, which would mean that the optimal path planning would be preferred as it guarantees optimality. Second, any general conclusions could not have been made. There is only a single outlier that is left in the 100 m radius, and if this is removed or not with the 200 m radius, this result wouldn't be enough to generalize in any way.

Any larger radii will mean that the performance gain will In the specific cases investigated, the 50 m radius will be the better choice, but the 100 m radius could have

To summarize the results: There are two observations that can be made

- Close to optimal performance in majority of cases investigated.
- Outliers with large increase in cost, and possibly infeasible paths.

The close to optimal performance here also includes the outliers from the 10 m paths with less than 12 % increase in paths. The expected increase in costs in these cases would be around 2-3 %.

As all the results have been compared to the optimal solutions, the optimality of paths is accurately measurable. In real world applications where the A* algorithm is used, the use is often conducted with a heuristic that increases computation speeds, while returning near-optimal paths. As

The focus of this report has been on measuring the optimality of solutions, as this is the single most important measure for how well the method performs. In real life applications, the optimality is not always needed, and near-optimal paths can be good enough. Cost increases around 2-3 % or even 12 % compared to the absolute optimal path, can therefore be insignificant as such deviations from the optimum can be expected anyway. In such cases, the hierarchical path planning with the coarsest terrain resolution with lowest radii can be used, as the large computation time decrease outweighs the need for optimal paths.

In other cases, optimal paths are more critical, but in these cases, hierarchical path planning using finer terrain resolutions can give very close to optimal paths while at the same time improving the computation time.

As with the outliers, these seem to be a problem that can arise at times using hierarchical path planning, and they are not easily mitigated using larger radii. The appearance of an outlier on rare occasions can for many purposes be acceptable, as most of the cases yield overall good results. There is no way of telling when such an outlier has occurred unless it is compared with the explicit optimal cost. Using conventional near-optimal path planning in the same type of terrain, there will also be a risk of such outliers occurring, and in such cases the outliers can be acceptable.

8.2 Improved hierarchical path planning

In addition to the general tests of the hierarchical path planning, the additional experiments have contributed to the overall results of the hierarchical path planning.

8.2.1 Flat terrain

The flat terrain originated as a hypothesis that the reason for the largest deviations in path distances and costs arises due to slope details that are lost when moving from a fine to a coarse terrain resolution. Using flat terrain with a minimum of such terrain details should therefore improve the results of the regular hierarchical path planning.

The distances between the optimal paths in the 1 m and the 10 m terrain were seen to reduced compared to the random paths. Where the regular paths were below 70 m distance between the paths with the majority below 50 m, the flat terrain paths are below 12 m distance. The flat terrain does however have one outlier at 180 m distance, but the regular paths has 5 outliers between 100 m and 900 m distance. The regular hierarchical path planning does consist of 30 cases, and not 11 though, so the numbers are not directly comparable, Using these paths leads to hierarchical paths with average distances below 3 m, in addition to the 180 m outlier. The costs of the hierarchical paths are all exactly zero, except for the outlier case where the path has an increased cost of less than 1 %. The large outliers that were seen in the 2 m and the 10 m are not observed in the flat terrain cases, and could potentially arise here too. However, with the flat terrain, the areas which are inaccessible are greatly reduced, and the possibilities for such large deviations to arise decreases. With flat terrain, the regular terrain speeds dominate the overall speeds, and this ensures that consistent path costs can be expected.

These results are a large improvement on the results of the 10 m hierarchical paths. This goes to show that for the flat terrain, the hierarchical approach is a very good choice, and can be expected to return the optimal paths in most cases, or paths close to these in cost. These results are also not specific to the terrain selection used, but can be generalized to be used with any terrain selection where the terrain is flat. This can typically be farmland and mountain plateaus in Norway, and for example deserts and steppes in other parts of the world.

In cases where parts of an area is flat, the path planning can be divided into the separate areas, where for example the flat parts have a lower radius and uses a coarser terrain for hierarchical path planning than the other parts. Another possibility could also be to use hierarchical path planning for the plat terrain, and use conventional path planning for the remaining parts for increased performance.

8.2.2 8 edge graphs

The 8 edge path planning were performed in both 2 m terrain and 10 m terrain. The motivation was to investigate if the use of 8 edge graphs would decrease the distances between the optimal paths in different resolutions, and thus increasing the likelihood of the hierarchical paths being the optimal paths.

For the 2 m paths are for the most part identical with the 1 m paths, and are for that reason not very interesting. For the 10 m paths on the other hand, there are many cases where there 1 m optimal paths and the 10 m optimal paths are located far apart.

From the results, there was a clear reduction in the distance between the hierarchical paths based on the 10 m terrain with the 8 edge graphs, with the majority of the measurements below 50 m average distances, where the measurements for the 4 edge graphs was at 200 m. This is a clear improvement, but that is also the only result that shows any improvement over the 4 edge graphs. The 2 m paths in the 4 edge graphs have approximately 0 distance to the 1 m optimal paths, while the 8 edge graphs have distances of 10-20 m to the optimal 8 path. A clear degradation of the results.

The computation times for the 8 edge graphs are close to but still higher than the corresponding 4 edge results. This is an indication that the A* algorithm converges to the solution faster with the 8 edge graphs, as it contains twice the number of edges, as much as twice the computation time could have been expected.

The reduced distances for the 8 edge graphs have not improved the costs of the hierarchical paths. Where most of the regular hierarchical paths are close to optimal, the 8 edge graphs have an increase in costs of at least 10 %, with the majority centered around 20 % increase in cost. This large increase means that the method in it's current form is not suitable for path planning. The bad results can point to a systematic error in the implementation, but it can be difficult to say with certainty. The only real result here is the decreased distance for the 10 m paths which means it can be worth looking into the 8 edge graphs in the future. The result on itself however is not of much use as it does not lead to decreased computation time, nor cost.

8.2.3 Multiple level hierarchy

The use of multiple levels in the hierarchy was an idea for reducing the computaitional cost to a minimum to see how much that potentially can be gained, and to see if the resulting paths are as good as the ones from the hierarchical paths in experiment 2.

As the results showed, the computation times could be reduced to as much as 40-45% of the regular hierarchical path planning. This result was for the 200 m radius however, and the performance increase was not as high for the other terrain resolutions, but all yielded decreases from the regular hierarchical path planning.

The costs of the paths were identical to the regular hierarchical paths. This is a result of the similarity between the 1 m and 2 m paths, that are expected to be close to identical. In general, each terrain level added would contribute to the increase in total cost, but in this case, the 2 m paths are approximately equal to the 1 m paths, and the added cost is negligible compared to the cost due to the 10 m hierarchical paths.

The use of the multiple level hierarchical path planning also adds complexity to the problem, with new path radius parameters that are needed. In the cases investigated here, this was relatively simple as the 2 m hierarchical path experiments had shown that the hierarchical path planning were successful even with paths down to 10 m. Because of this, the path radius for the 2 m terrain were set to 20 m as this was expected to give good results, and so it did. When this is not the case however, care must be taken in choosing these parameters too.

Lastly, the use of multiple level hierarchy demonstrates that the method works good with adding several fine resolutions as lower levels in the hierarchy. There would in principle not be any problems with using the existing results for adding even more detailed terrain data, and finding a hierarchical path here. In a real life situation with detailed terrain data becoming available as a vehicle moves, this is a great advantage as the recalculation of the path can be computationally heavy, and using existing path planning results greatly decrease this problem.

8.3 Summary

Overall, the hierarchical path planning can be said to yield promising results. The average computation times have been reduced in all cases tested, and most of the results have been close to optimal. There is still a challenge left with large outliers, but if that risk is accepted, the hierarchical path planning can be expected to give good results.

Based on the experiments, it can be hard to draw any general conclusions to how much improvement can be expected. For a setup similar to this with the same relative problem sizes, the performance can be expected to be similar to here. However, using different terrain selections, choosing the radius can be hard, and can change much from what works here, and the performance will change. If the number of large outliers gets too high, the added computation time of the conventional path planning can be worth the cost if the hierarchical path integrity gets too low.

The specific results of these experiments can only be used with this particular terrain provided, and other terrain selections can give worse results. However, the analysis provided has a value in itself as it shows some of the potential in the hierarchical path planning. The experiments are all consistent with regards to the computation time and the size of the individual problems, with all results pointing to a linear relationship within the range of experiments tested here. This can be valuable as it makes it possible to predict how much improvement the hierarchical approach can mean in a specific case, relative to running a conventional path planning algorithm.

Another result that can be of value can be seen from the case with the 10 m hierarchical paths. Here, a fine and a coarse terrain were used in hierarchical path planning, and the optimal paths differ, causing the hierarchical paths to have increased costs. The simple observation that increased radius decreases the average cost is a result that might seem obvious, yet is still important. If this was not the case, the hierarchical path planning would be hard to predict, and if the behavior of the method can not be predicted, it will be of little value in practice. With reduced cost with increased radius, the path radius as a tuning parameter between reducing expected cost, and reducing computation time, which makes the method flexible.

Lastly, the flat terrain results gives some insight to further analysis of paths that can be conducted. With the flat terrain, the closeness and costs of the hierarchical paths was improved. This result can be directly utilized for parts of a terrain selection, with for example reduced radius for increased performance in the flat parts. The results here also helps with better understanding of why and when deviations between paths occur, and that this will not happen where the terrain is flat, but rather where there are large slopes and infeasible terrain because of it.

Chapter 9

Conclusion

The goal of this report was to develop a method for performing hierarchical path planning with terrain data of different levels of detail. The procedure includes a method for automatical terrain analysis and path planning, and a specific method for creating a hierarchy of terrain data in different resolutions, and using the hierarchy for finding a detailed path based on coarse terrain data, using a fixed radius search area. The method is flexible with regards to using existing results for performing path planning in increasingly detailed terrain dat.

Experiments with real terrain data given in different resolutions has been performed, comparing the hierarchical path planning procedure with a conventional path planning approach. For the given terrain data, the results show that the computation performance is improved using the hierarchical method, while the majority of the paths are close to optimal. The computation times were as high as 5 times faster than the conventional path planning on average, and this should be achieved with a similar setup. The paths had costs higher than the optimal paths, but due to only experimenting with one terrain no absolute conclusions can be drawn about the optimality in the general case.

There are some outliers with large deviations, which shows that the method can be vulnerable differences between the terrain levels.

Two additional improvements were investigated. Using multiple level hierarchy demonstrated that increased performance can be achieved, without degrading the path cost. Using 8 edge graphs instead of 4 edge graphs, did however not show any improvements in the experiments, neither in computation performance nor in optimality.

An investigation was also performed using only flat terrain selections. In these experiments, there were no significant differences between the hierarchical and the optimal paths showing how the flatness of terrain has an impact on the success of the hierarchical approach to path planning.

9.1 Further work

The experiments performed are all based on terrain from within a limited area, which means the variation in terrain data will be limited. To obtain more general results, the experiments should be repeated with a larger selection of different terrain types to see if the results holds, and to see how this effects the parameters chosen, such as terrain resolutions used and the path radii.

The outliers detected in some cases does appear to be present throughout all experiments, except in the experiments using the flat case. A measurement for the uncertainty of the hierarchical paths that could detect when the hierarchical path is expected to be close to the optimal, and when it is at risk of deviating would be a measurement that would make the hierarchical path planning more reliable.

As the hierarchical path planning in the flat terrain improved the optimality of the paths greatly, the flatness around a path potentially gives an indication of the integrity of the path. Another possibility is to find a direct measurement between the differences of the different terrain levels. It seems plausible that similar terrain levels can lead to more similar paths than terrain levels with large differences.

A possible method for increasing path planning performance could be to use a dynamical path radius instead of a fixed radius. This can allow sections of the terrain that are considered certain with smaller radius than the more uncertain one. For example, based on the flat terrain results, it would be possible to use a small radius in flat sections as the hierarchical paths are certain. Other examples could be roads, as these are the fastest way through terrain, and are therefore more certain than other terrain types.

Bibliography

[1] Robert A. Adams and Christopher Essex. *Calculus: A Complete Course*. Pearson, eighth edition, 2013.

[2] A. Botea, M. Müller, and J. Schaeffer. Near Optimal Hierarchical Path-Finding. *Journal of Game Development*, 1(1):7–28, 2004.

[3] Solveig Bruvoll. Preliminary study of terrain analysis and path planning for computer generated forces. Technical Report FFI-notat 2013/00688, Norwegian Defence Research Establishment (FFI), 2013.

[4] Solveig Bruvoll. Situation dependent path planning for computer generated forces. Technical Report FFI-rapport 2014/01222, Norwegian Defence Research Establishment (FFI), 2014.

[5] V. Bulitko, N. R. Sturtevant, and M. Kazakevich. Speeding Up Learning in Real-time Search via Automatic State Abstraction. *AAAI*, 2005.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, third edition, 2009.

[7] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numeriche Mathematik*, 1:269–271, 1959.

[8] J. Giesbrecht. Global Path Planning for Unnmanned ground vehicles. Technical Report DRDC Suffield TM 2004-272, Defence Research and Development Canada, 2004.

[9] D. Harabor and A. Botea. Hierarchical Path Planning for Multi-Size Agents in Heterogeneous Environments. *IEEE Symposium on Computational Intelligence and Games*, pages 258–264, 2008.

[10] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions of Systems Science and Cybernetics*, SSC-4(2):100–107, 1968.

[11] M. R. Jansen and M. Buro. HPA* Enhancements. *AIIDE*, pages 84–87, 2007.

[12] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[13] Ian Millington. *Artificial Intelligence for Games*. Morgan Kaufmann Publishers, 2006.

[14] D. Mould and M. C. Horsch. An Hierarchical Terrain Representation for Approximately Shortest Paths. *PRICAI*, 2004.

[15] Niles Ritter. GeoTIFF Revision 1.0 Specification, 2000.

[16] H. Samet. An Overview of Quadtrees, Octrees, and Related Hierarchical Data Structures. *NATO ASI Series,*, F40, 1988.

[17] N. Sturtevant and M. Buro. Partial Pathfinding Using Map Abstraction and Refinement. *AAAI*, pages 1392 –1397, 2005.

[18] Jan Henrik Wiik and Solveig Bruvoll. Methods for a customized path planning algorith, for use in military simulations. Technical Report FFI-notat 2013/02656, Norwegian Defence Research Establishment (FFI), 2013.

Appendix

Appendix A

Greens theorem in area calculations

A.1 Green's theorem

Green's theorem gives the relationship between a line integral around a simple closed curve $\mathcal{C}$ and a double integral over the plane region D bounded by $\mathcal{C}$ [1]. The theorem is as follows:

Let R be a regular, closed region in the xy-plane whose boundary, $\mathcal{C}$, consists of one or more piecewise, smooth, simple closed curves that are positively oriented with respect to R. If $\mathbf{F} = F_1(x,y)\mathbf{i} + F_2(x,y)\mathbf{j}$ is a smooth vector field then

$$\oint_{\mathcal{C}} F_1(x,y)dx + F_2(x,y)dy = \iint_R \left(\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y}\right) dA \tag{A.1}$$

A.2 Greens area formula

Green's theorem can be used in a special form for calculating the area within a simple closed curve, in a formula called Green's area formula. Using Green's theorem and defining a vector field $\mathbf{F} = \frac{1}{2}\left(-y\mathbf{i} + x\mathbf{j}\right)$, the result is

$$\frac{\partial F_2}{\partial x} - \frac{\partial F_1}{\partial y} = \frac{1}{2}\left(\frac{\partial x}{\partial x} - \frac{\partial(-y)}{\partial y}\right) = 1$$

Therefore, with the selected vector field, the form of (A.1) is

$$\frac{1}{2}\oint_{\mathcal{C}} -y\ dx + x\ dy = \iint_R dA \tag{A.2}$$

As $\iint_R dA$ is the area of the region R, (A.2) can be written as

$$\text{Area of } R = \frac{1}{2}\oint_{\mathcal{C}} -y\ dx + x\ dy \tag{A.3}$$

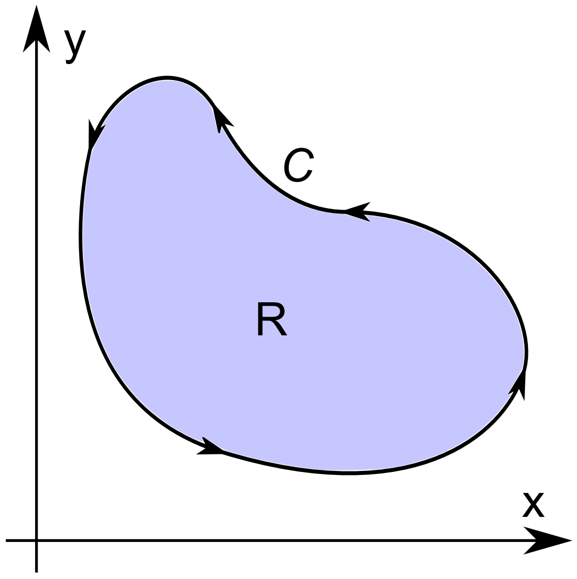


Figure A.1: An area R with boundary $\mathcal{C}$

which is known as Green's area fomula. For a closed curve in the plane, this formula will provide a way of calculating the area within the curve.

A.3 Line integral for straight edges

The line integral

$$\int_E -y\,dx + x\,dy \tag{A.4}$$

for straight edges shown in figure 4.3, can be calculated directly by a simple observation. The edges are either north-south edges, meaning $dx = 0$ and x is constant, or the edges are east-west edges and $dy = 0$ and y is constant.

The integral A.4 over a single edges is

$$\int_{E_i} -y\,dx + x\,dy = -y_0\Delta x + x_0\Delta y \tag{A.5}$$

For the edges, Δx and Δy are given as:

$$i = 1: \qquad \Delta x = 0 \qquad \Delta y = -d \tag{A.6}$$
$$i = 2: \qquad \Delta x = d \qquad \Delta y = 0 \tag{A.7}$$
$$i = 3: \qquad \Delta x = 0 \qquad \Delta y = d \tag{A.8}$$
$$i = 4: \qquad \Delta x = -d \qquad \Delta y = 0 \tag{A.9}$$

The line integral over each of the edge types are therefore given by:

$$W_i = \int_{E_i} -y \, dx + x \, dy = \begin{cases} -x_0 \cdot d & j = 1 \\ -y_0 \cdot d & j = 2 \\ x_0 \cdot d & j = 3 \\ y_0 \cdot d & j = 4 \end{cases}$$

A.4 Line integral for diagonal edges

The line integral

$$\int_E -y \, dx + x \, dy \tag{A.10}$$

for diagonal edges shown in figure 4.4, is calculated using a parametrization of the edges. With a parametrization

$$\mathbf{r}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \quad a \le t \le b, \tag{A.11}$$

an integral

$$\int_E F_1(x,y) \, dx + F_2(x,y) \, dy = \int_E \mathbf{F} \cdot d\mathbf{r} \tag{A.12}$$

can be calculated [1] as

$$\int_a^b \mathbf{F}(\mathbf{r}(t)) \cdot \frac{d\mathbf{r}}{dt} \, dt \tag{A.13}$$

With a parametrization, the integral (A.10) is therefore calculated as

$$\int_a^b \left(-y(t)\frac{dx}{dt} + x(t)\frac{dy}{dt} \right) dt \tag{A.14}$$

For each of the edge types shown in figure 4.4, a parametrization is selected

$$\mathbf{r}_5(t) = \begin{bmatrix} x_0 + t \\ y_0 - t \end{bmatrix}, \qquad \frac{d\mathbf{r}_5}{dt} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$
$$\mathbf{r}_6(t) = \begin{bmatrix} x_0 - t \\ y_0 - t \end{bmatrix}, \qquad \frac{d\mathbf{r}_6}{dt} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$
$$\mathbf{r}_7(t) = \begin{bmatrix} x_0 - t \\ y_0 + t \end{bmatrix}, \qquad \frac{d\mathbf{r}_7}{dt} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$
$$\mathbf{r}_8(t) = \begin{bmatrix} x_0 + t \\ y_0 + t \end{bmatrix}, \qquad \frac{d\mathbf{r}_8}{dt} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

for $0 \leq t \leq d$.

Evaluating the integrals:

$$\int_{E_1} -y\,dx + x\,dy = \int_0^d -(y_0 - t) + (x_0 + t)(-1)\,dt = (-x_0 - y_0)d \quad \text{(A.15)}$$

$$\int_{E_2} -y\,dx + x\,dy = \int_0^d -(y_0 - t)(-1) + (x_0 - t)(-1)\,dt = (-x_0 + y_0)d \quad \text{(A.16)}$$

$$\int_{E_3} -y\,dx + x\,dy = \int_0^d -(y_0 + t)(-1) + (x_0 - t)\,dt = (x_0 + y_0)d \quad \text{(A.17)}$$

$$\int_{E_4} -y\,dx + x\,dy = \int_0^d -(y_0 + t) + (x_0 + t)\,dt = (x_0 - y_0)d \quad \text{(A.18)}$$

Appendix B

MATLAB implementations

In this appendix, the most central implementations for

B.1 Terrain analysis

The following code is the main parts of the implementation of the automatic terrain analysis used for path planning.

Code B.1: Code for downsampling terrain types data

```
function terrainNew = downsampleTerrainTypes(terrain , N)
    [Nrows, Ncols] = size(terrain);

    % Not in use
    terrainDescriptions = {'ocean','road','sportsarena',...
    'urban','marsh','forest','farmland','quarry',...
    'river','stream','lake','industrial','open'};

    terrainTypeFromPriority = ...
        [2 10 9 11 1 12 8 4 6 5 13 7 3] - 1;
    priorities = [5 1 13 8 10 9 12 7 3 2 4 6 11];
    newNcols = floor(Ncols/N);
    newNrows = floor(Nrows/N);

    terrainNew = ones(newNrows, newNcols);
    for row = 1:Nrows/N
        for col = 1:Ncols/N
            subRows = N*(row-1)+1:N*(row-1)+N;
            subCols = N*(col-1)+1:N*(col-1)+N;
            block = terrain(subRows,subCols);
```

```
                highestPriority = min(min(priorities(block + 1)
                  ));
                terrainNew(row,col) = ...
                    terrainTypeFromPriority(highestPriority);
            end
        end
    end
```

Code B.2: Code for calculating slopes using elevation data

```
function slopes = findSlopes(heights, terrainResolution)
    [gx, gy] = gradient(heights);
    gradients = sqrt(gx.^2 + gy.^2);
    slopes = atan(gradients/terrainResolution);
end
```

Code B.3: Code for quantizing the calculated slopes to the Go, Go slow or the No-go category.

```
function slopesQuant = quantizeSlopes(slopes)
    % 1 = 0-15
    % 2 = 15-25
    % 3 = 25+
    [Nrows, Ncols] = size(slopes);
    slopesQuant = zeros(Nrows, Ncols);
    slowLimit = 15*pi/180;
    noGoLimit = 25*pi/180;

    for row=1:Nrows
        for col=1:Ncols
            if slopes(row,col) > slowLimit
                if slopes(row,col) > noGoLimit
                    slope = 3;
                else
                    slope = 2;
                end
            else
                slope = 1;
            end
            slopesQuant(row,col) = slope;
        end
    end
end
```

Code B.4: Code for creating a speed map using the quantized slopes and the terrain types

```
function speedMap = createSpeedMap(slopesQuant,
   terrainTypes)
```

```
    terrainDescriptions = {'ocean','road','sportsarena',...
        'urban','marsh','forest','farmland','quarry',...
        'river','stream','lake','industrial','open'};
    terrainSpeeds = 1/3.6*[1e-12, 30, 25, 1e-12, 15, 5, 20,
        1e-12, 1e-12, 1e-12, 1e-12, 1e-12, 15];
    terrainSpeedMap = terrainSpeeds(terrainTypes+1);

    [Nrowstt, Ncolstt] = size(terrainTypes);
    [Nrowssq, Ncolssq] = size(slopesQuant);
    Nrows = min(Nrowstt, Nrowssq);
    Ncols = min(Ncolstt, Ncolssq);
    speedMap = zeros(Nrows, Ncols);
    for row=1:Nrows
        for col=1:Ncols
            switch slopesQuant(row,col)
                case 1
                    % go
                    speed = terrainSpeedMap(row,col);
                case 2
                    % goSlow
                    speed = min(terrainSpeedMap(row,col),
                        5/3.6);
                    speed(speed == 0) = 1e-12;
                case 3
                    % noGo
                    speed = 1e-12;
                    if (terrainSpeedMap(row,col) == 30/3.6)
                        speed = 5/3.6;
                    end
            end
            speedMap(row,col) = speed;
        end
    end
end
```

B.2 A* search algorithm

A* search algorithm implementation. The algorithm uses a graph class which is a standard graph implementation of a directed graph with weighted edges, and is not included in this report. The version showed assumes a 4 edge graph structure. The algorithm uses a binary heap for the openSet which is a min priority queue. The binary heap is a common data structure often used for min priority queues. For details on implementation of graph and binary heap structures, see [6].

,

```
function shortPath = AstarShortPath(graph, startNode,
  targetNode)
    % AstarShortPath finds the approximately shortest path
    % from startNode to targetNode

    % Initializing data
    cameFrom = zeros(graph.numCols*graph.numRows, 1);
    closedSet = zeros(graph.numRows*graph.numCols, 1);
    cost = 1e15*ones(graph.numCols*graph.numRows, 1);
    openSet = BinaryHeap(graph.numRows, graph.numCols);
    openSet.insert(startNode, graph.heuristic(startNode,
       targetNode));
    cost(startNode) = 0; % minimum cost from start to start
    shortPath = -1;

    % Running search
    while openSet.size ~= 0
        currentNode = openSet.extractMin();
        if currentNode == targetNode
            shortPath = reconstructPath(cameFrom,
               targetNode);
            return
        end

        closedSet(currentNode) = 1;

        for i = 1:4
            neighbourEdge = graph.edgeLookup(currentNode,i)
               ;
            if neighbourEdge == -1
                continue;
            end

            neighbour = graph.E(neighbourEdge, 2);
            if closedSet(neighbour)
                continue
            end

            tentativeCost = cost(currentNode) + graph.E(
               neighbourEdge,3);
            if ~openSet.isMember(neighbour) || ...
                    tentativeCost < cost(neighbour)
                cameFrom(neighbour) = currentNode;
                cost(neighbour) = tentativeCost;
                futureCost = tentativeCost + graph.
                   heuristic(neighbour, ...
                    targetNode);
                if ~openSet.isMember(neighbour)
                    openSet.insert(neighbour, futureCost);
                else
                    openSet.decreaseNode(neighbour,
                       futureCost);
                end
            end
        end
    end
end
```

B.3 Path evaluation

The following code is the implementation of area between two generic paths with the same start and end points.

Code B.5: Code for calculation of area between paths. Uses code B.6 for the specific area calculation.

```
function [areaTemp, keyPoints] = areaBetweenPaths(graphA,
   graphB, pathA, pathB)
    pointsA = graphA.V(pathA,:);
    pointsB = graphB.V(pathB,:);
    terrainSize = max(graphA.terrainSize, graphB.
       terrainSize);

    if graphA.terrainSize > graphB.terrainSize
        pointsB = arrayfun(@coor1to10, pointsB);
        pointsB = unique(pointsB,'rows','stable');
    elseif graphA.terrainSize < graphB.terrainSize
        pointsA = arrayfun(@coor1to10, pointsA);
        pointsA = unique(pointsA,'rows','stable');
    end

    [keyPoints, iA, iB] = intersect(pointsA, pointsB, 'rows
       ', 'stable');
    [iA, iB] = removeCrossingSections(keyPoints, iA, iB);

    diffIB = diff(iB);
    diffIA = diff(iA);
    loopsNum = 0;
    areaTemp = 0;
    for i = 1:length(diffIB)

        if diffIA(i) == 1 && diffIB(i) == 1
            continue
        else
            loopsNum = loopsNum+1;
            fromA = iA(i);
            toA = iA(i+1);
            fromB = iB(i);
            toB = iB(i+1);

            sectionA = pointsA(fromA:toA,:);
            sectionB = flipud(pointsB(fromB:toB-1,:));
            closedLoop = [sectionA; sectionB];

            areaTemp(loopsNum) = areaFromClosedCurve(
               closedLoop(:,1), closedLoop(:,2));
        end
    end
    areaTemp = areaTemp*(terrainSize^2);
end
```

Code B.6: Implementation of Green's area formula for closed curves with straight or diagonal edges. Used by code B.5

```
function A = areaFromClosedCurve( y, x )
    % Implementation of Green's area fomula for areas with
       4 and 8 edge
    % type boundaries
    A = 0;
    if (x(1) == x(end) && y(1) == y(end))
        dx = diff(x);
        dy = diff(y);
        A = 0;
        for i = 1:length(dx)
            if (dx(i) == 0 || dy(i) == 0) % edge = 4
                W = (x(i)*dy(i) - y(i)*dx(i));
            else
                if dx(i) > 0 && dy(i) < 0 % edge = 8
                    W = -x(i) - y(i);
                elseif dx(i) < 0 && dy(i) < 0
                    W = -x(i) + y(i);
                elseif dx(i) < 0 && dy(i) > 0
                    W = x(i) + y(i);
                elseif  dx(i) > 0 && dy(i) > 0
```

```
                    W = x(i) - y(i);
                end
            end
            A = A + 0.5*W;
        end
        A = abs(A);
    end

end
```

B.4 Hierarchical path planning implementation

The following function is the essence in how the hierarchical path planning is implemented. Given a graph, a path in this graph and a desired path radius, this function creates a mask of the area within the given radius of the specified path. This mask is then used for creating a graph, only including the masked areas. This is supported directly by the graph class, and all other processes such as terrain analysis and path planning are exactly as before.

Code B.7: Code for creating a mask covering the area within a desired radius of the given path.

```
function mask = createMask(graph, path, radius)
    mask = zeros(graph.numRows, graph.numCols);
    for i = 1:length(path)
        coor = graph.V(path(i),:);
        row = coor(1);
        col = coor(2);
        rowStart = max(1, row-radius);
        rowEnd = min(graph.numRows, row+radius);
        colStart = max(1, col-radius);
        colEnd = min(graph.numCols, col+radius);
        mask(rowStart:rowEnd,colStart:colEnd) = 1;
    end
end
```