FOI-R--1241--SE
May 2004
1650-1942

Scientific report

Susanne Edlund

Driveability analysis

using a digital terrain model and map data

Command and Control Systems
Box 1165
SE-581 11 LINKÖPING
Sweden

Swedish Defence Research Agency
Command and Control Systems
Box 1165
SE-581 11 LINKÖPING
Sweden

FOI-R--1241--SE
May 2004
1650-1942

Scientific report

Susanne Edlund

Driveability analysis

using a digital terrain model and map data

Issuing organization	Report number, ISRN	Report type
Swedish Defence Research Agency Command and Control Systems Box 1165 SE-581 11 LINKÖPING Sweden	FOI-R--1241--SE	Scientific report
	Programme Areas: C4ISR	
	Month year: May 2004	Project no.: E7089
	General Research Areas: Commissioned Research	
	Subcategories: Reconnaissance and Surveillance	
Author/s (editor/s): Susanne Edlund	Project manager: Erland Jungert	
	Approved by: Martin Rantzer	
	Sponsoring agency: Swedish Armed Forces	
	Scientifically and technically responsible: Erland Jungert och Fredrik Lantz	

Report title

Driveability analysis using a digital terrain model and map data

Abstract

Driveability analysis of terrain data offers an important technique for decision support for all kinds of movements in the terrain. The work described in this report uses a high resolution digital terrain model generated from the laser radar data and further processed by the Category Viewer program, and information from the Real Estate Map. Properties of features found in a filtering process are calculated and compared with a set of rules in a knowledge base to get a driveability cost. This cost is then visualized in a graphical user interface.

An evaluation of what driveability is and what it is affected by is performed, and a general cost function is developed, which can be used even if not all relevant information is available.

The methods for property and cost calculation need to be developed further, as well as the rules in the knowledge base. However, the implemented program offers a good framework for further research in the area.

Keywords

driveability, digital terrain model, decision support

Further bibliographic information	Language
	English
ISSN: 1650-1942	Pages: 88
Distribution: By sendlist	Price: Acc. to pricelist
	Security classification: Unclassified

Utgivare	Rapportnummer, ISRN	Klassificering
Totalförsvarets forskningsinstitut Ledningssystem Box 1165 SE-581 11 LINKÖPING Sweden	FOI-R--1241--SE	Vetenskaplig rapport
	Forskningsområde: Spaning och ledning	
	Månad, år: Maj 2004	Projektnummer: E7089
	Verksamhetsgren: Uppdragsfinansierad verksamhet	
	Delområde: Spaningssensorer	
Författare/redaktör: Susanne Edlund	Projektledare: Erland Jungert	
	Godkänd av: Martin Rantzer	
	Uppdragsgivare/kundbeteckning: Försvarsmakten	
	Tekniskt och/eller vetenskapligt ansvarig: Erland Jungert och Fredrik Lantz	

Rapportens titel

Framkomlighetsanalys med hjälp av en digital terrängmodell och kartdata

Sammanfattning

Framkomlighetsanalys av terrängdata utgör en viktig teknik i militära beslutsstödshjälpmedel för bestämning av rörelser i terrängen. I det arbete som beskrivs här utnyttjas en högupplösande digital terrängmodell, genererad från laser-radardata, och data från fastighetskartan. Laser-radardata är ytterligare bearbetade med hjälp av programmet CategoryViewer för att hitta olika terrängobjekt. Objektegenskaper som hittats genom en filtreringsprocess jämförs med ett antal regler i en kunskapsdatabas för bestämning av en framkomlighetskostnad, som kan visualiseras i ett grafiskt användargränssnitt.

Arbetet omfattar också en genomgång av begreppet framkomlighet samt de objektegenskaper som påverkar denna. En generell kostnadsfunktion har utvecklats, som kan användas även då inte all nödvändig information finns tillgänglig. Metoden för egenskaps- och kostnadsberäkning behöver emellertid utvecklas vidare; liksom också den underliggande kunskapsdatabasen. Trots detta erbjuder det implementerade programmet en lämplig utgångspunkt för fortsatt forskning.

Nyckelord

framkomlighet, digital terräng modell, beslutsstöd

Övriga bibliografiska uppgifter	Språk
	Engelska
ISSN: 1650-1942	Antal sidor: 88
Distribution: Enligt missiv	Pris: Enligt prislista Sekretess: Öppen

Acknowledgements

I would like to thank the following persons for helping me and supporting me during the production of this thesis: Erland Jungert, Fredrik Lantz, Karin Silvervarg, and Tobias Horney.

Contents

1 Introduction **15**
1.1 The ISM project 15
1.2 Problem description 15
1.3 Definitions 16
1.4 Overview 17

2 Theoretical background **19**
2.1 Existing work 19
2.2 Clementini geometry 21

3 Driveability impact factors **23**
3.1 Direction 23
3.2 Environment 23
3.3 Non-geometric and non-terrain features 24
3.4 Time 24
3.5 Weather 24
3.6 Properties of vehicles and terrain features 24

4 Input data **27**
4.1 Laser radar data 27
4.2 Digital terrain model 27
4.3 Map data 28

5 Software tools **31**
5.1 Category Viewer 31
5.2 MapObjects 32

6 Driveability measures **35**
6.1 Definitions 37
6.2 Driveability knowledge base 38
6.3 Cost function 39
6.4 Restrictions 39
6.5 Data availability and reliability 39
6.6 Default reasoning 40

7 Method **41**
7.1 Test data 41
7.2 Communication with Category Viewer 41

7.3 Problems with Category Viewer . . . 43
7.4 Improving the connect algorithm . . . 44
7.5 Data formats and representations . . . 44
7.5.1 Representation of the world . . . 44
7.5.2 Representation of feature types . . . 44
7.5.3 Representation of map data . . . 45
7.6 Creating filters . . . 46
7.7 Calculating terrain feature properties . . . 46
7.7.1 Slope . . . 46
7.7.2 Direction of DTM features . . . 46
7.7.3 Direction of map features . . . 46
7.7.4 Area . . . 47
7.7.5 Width and length of DTM features . . . 47
7.7.6 Width and length of map features . . . 48
7.7.7 Centre line . . . 48
7.7.8 Boundary . . . 49
7.8 Data availability and reliability . . . 51
7.9 Cost function . . . 52

8 Results **55**
8.1 General user interface . . . 55
8.2 The postprocessing algorithm . . . 56
8.3 Terrain features . . . 56
8.4 Centre line, boundary, and width . . . 60
8.5 Driveability . . . 62
8.6 Performance . . . 64
8.7 Documentation . . . 67

9 Discussion and future research **69**
9.1 Input data . . . 69
9.2 The postprocessing algorithm . . . 70
9.3 Feature properties . . . 70
9.3.1 Width of DTM features . . . 70
9.3.2 Direction of map features . . . 72
9.3.3 Centre line . . . 72
9.4 User interface . . . 72
9.5 Data availability and reliability . . . 74
9.6 Driveability measures . . . 74
9.7 Performance . . . 75
9.8 Conclusions . . . 75

Bibliography **80**

A Vehicle properties **81**
A.1 T72 . . . 81
A.2 Volvo Valp (Pltg 903) . . . 82

B Driveability Viewer **83**
B.1 Data availability and reliability 83
B.2 The knowledge base 84
B.3 Program start up 84

C Category Viewer **87**

List of Figures

1.1 An overview of the ΣQL system. From [13]. 16

4.1 Calculation of $slope = \arctan(\Delta z/b)$. 28

4.2 The tile subareas. 28

4.3 The boundary lines of hierarchical feature types in map data features. Adapted from [2]. a) The six hierarchies in the ML theme (line layer for ground data). b) Showing only farmland results in an incomplete classification. c) Farmland together with the hierarchically superior boundary lines gives a complete classification. 30

5.1 The filter structure to be applied to the cross-section of a ditch. Adapted from [33]. 31

5.2 Overview of the data flow in Category Viewer. 33

5.3 Category Viewer using the ditch filter with a minimum norm of 0.5. 34

6.1 A partition of the area of interest into regions with uniform characteristics. 35

6.2 The possible driveability paths and their labels (compare with the DTM directions in figure 4.2). 36

6.3 Example of a cost function for slopes. 37

7.1 An overview of Driveability Viewer. 42

7.2 Width estimation using average segment length and Pythagoras' theorem. a) The feature. b) The horizontal segments, average length $l_h = 3.0$ tiles. c) The vertical segments, average length $l_v = 2.6$ tiles. This gives $l_{hyp} = \sqrt{3.0^2 + 2.6^2} = 3.97$ tiles and width $w = \frac{3.0*2.6}{3.97} = 1.96$ tiles. 48

7.3 Using the centre line of a feature to demonstrate the result of the width estimation. According to figure 9.1 the width estimation of this feature is 1.96 tiles. 48

7.4 Algorithm to determine the order of the points on a line. a) The original set of points. A starting point p_0 is arbitrarily chosen. b) p_1 is the point closest to p_0 and is therefore chosen as the next point on the line. c) p_2 is chosen as the next point on the line as it is closest to p_1. d) There is no point close enough to p_2, so the search is resumed from p_0 and p_{-1} is found. 50
7.5 Line filtering using a) distances and b) angles. 50
7.6 Calculation of boundary coordinates (marked with diamonds). The solid line corresponds to the centre line of the feature. All dashed lines have a length equal to the width estimation of the feature. 51

8.1 Result of the ditch filter using the simple connect algorithm. 56
8.2 Result of the ditch filter using the connect edge algorithm. 57
8.3 Result of the ditch filter using the connect edge algorithm and the postprocessing algorithm. 58
8.4 Colour coding of feature layers. The blue (dark) features are ditches and the yellow (light) ones are slopes. 59
8.5 Comparison of the two methods developed for steps 3 and 4 of the centre line algorithm: angle deviation (blue/dark) and distance deviation (yellow/light). 60
8.6 Calculated centre line and boundary points. 61
8.7 The driveability of T72 (all driveability paths). 62
8.8 The driveability of Volvo Valp (all driveability paths). . . . 63
8.9 The driveability of T72 (driveability path 2/north-east). . 64
8.10 The driveability of T72 (driveability path 4/north-west). . 65
8.11 Time consumption for the driveability calculation, i.e. application of every rule in the knowledge base to every tile in the area of interest, and thereby also calculation of properties. 66
8.12 Time consumption for the major parts of the data pre- and postprocessing. 66

9.1 Examples of feature types which cause unsatisfactory width estimations. a) Feature that bends. Width estimation gives 3.5 tiles. b) Feature width a large width variation. Width estimation gives 3.2 tiles. 71
9.2 Examples of rules of an alternative method for calculation of the points on the centre line. Bold-lined boxes represents horizontal centre tiles, and regular boxes represents vertical centre tiles. The first box represents the case when the horizontal and the vertical centre tile coincide. The tiles to the right show where the centre is assumed to be. . 73

A.1 T72. The picture is the property of FOI. 81

A.2 Volvo Valp. The picture is the property of FOI. 82

B.1 Simplified class diagram for Driveability Viewer. 85

C.1 Simplified class diagram for Category Viewer. 88

1. Introduction

This report is a master's thesis presented at the Department of Computer Science at Linköping University, Sweden. The work was done at the Swedish Defence Research Agency (FOI) in Linköping.

Driveability analysis (also called trafficability analysis) of terrain data offers an important technique for decision support for all kinds of movements in the terrain. This type of analysis is needed for the judgement of possible movements of targets (target tracking) and for the planning of future movements. An important data source for such an analysis is a digital terrain model (DTM) [32, 33] which in this case is a high resolution digital terrain model generated from laser radar (see section 4.1). At FOI in Linköping such a method has been developed for finding essential terrain objects such as ditches and ridges. This method needs to be improved to be used in connection with driveability analysis.

1.1 The ISM project

The ISM project (Information system for target recognition) is a recently finished project at FOI in Linköping. The main component of the information system is the ΣQL engine. ΣQL is a query language, which can be described as "a decision support tool for target recognition in a multisensor environment" [12, 13]. Sensor data fusion plays an important role in the query system, allowing the users to apply sensor independent queries, i.e., user understanding about which sensors and sensor algorithms that are used to produce the result of the query is not necessary. However, results must be calculated and presented in such a way that the user trusts them.

The ΣQL engine consists of a set of modules (see figure 1.1), one of which corresponds to a terrain query language [13] under development. The work carried out here will be part of that language.

A philosophy of the ISM project was that data should only be processed as a result of a user query. This philosophy has been regarded in this work as well (see sections 7.2 and 7.8).

1.2 Problem description

The task is to decide if a certain terrain area is driveable or not for a given type of vehicle. The result of the analysis is to be visualized on a

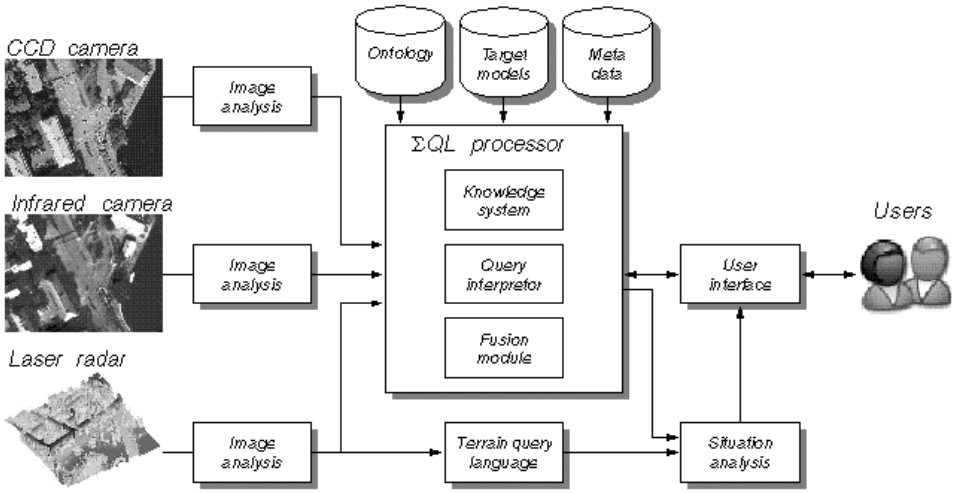

Figure 1.1: An overview of the ΣQL system. From [13].

map where the driveability is highlighted. In addition to the digital high resolution data model other information is needed as well, e.g. if a certain area can be classified as a forest. The density of the forest is required as well. Finally, information about the vehicle is needed, both its properties (e.g. if it has wheels or tracks, its weight etc.), and its capabilities (e.g. how steep slopes it can climb). This is a complex problem where various types of information must be fused and presented to enable a complete description of the driveability.

The objective of this work is to develop a method for driveability calculation and to test its usefulness as a planning tool. As this is a first attempt flexibility, extensibility, and ease of understanding are important aspects. In the future this method is to be further developed into a specific terrain query language to be used in continuation of the ISM project. Requirements for the future terrain query language are described in [13].

To be useful in real-world applications information from many sources is required. The focus in this work, however, is concerned with the overall design and basic structure of such a tool.

The implementation of the model is made in Java, using laser radar data (see section 4.1) and the Real Estate Map (see section 4.3). Category Viewer [40] (see section 5.1), and MapObjects (see section 5.2) are other software tools which are used.

1.3 Definitions

Driveability, trafficability = The capability of a vehicular movement through some region. It is a relationship between some entity (capable of movement) and the area through which it moves. Whether an

area is trafficable for the entity, and the measurement of how trafficable the area is for the entity, depend on the interaction between the entity's mobility characteristics and the terrain attributes." [10] This work will treat driveability maps, in contrast to the path planning problem.

Overlay = "diagrams that have uniform significance or characteristics relevant to the desired analysis. These diagrams . . . are produced by outlining the regions that are uniform for relevant characteristics on transparent sheets, registered to the underlying map." [10] Overlays are used to decrease the information overload resulting from field surveys and topographic maps.

Driveability map = An overlay where each region with homogeneous characteristics is assigned a driveability in one or more principal directions. Each of these directions is associated with a cost, which represents the level of driveability through that region.

Feature = A "representation of a geographic feature that has both a spatial representation referred to as a 'shape' and a set of attributes." [18]. In this work, there are two types of features: DTM *features* and *map features*, named after their sources. DTM features and map features are generally denoted *terrain features*.

1.4 Overview

Chapter 2 explores the existing work in the driveability area, and also describes a method for determination of topological relationships. Chapter 3 discusses different aspects of driveability. The used input data and software tools are described in chapters 4 and 5. How driveability can be measured is described in chapter 6. The implementation of the knowledge aquired from these chapters is described in chapter 7. The results are presented in chapter 8 and discussed in chapter 9.

The appendices contain a more detailed discussion of the used information and the implemented program.

2. Theoretical background

This chapter explores the existing work in the driveability area, and also describes a method for determination of topological relationships.

2.1 Existing work

Donlon et. al. [10] have developed a domain theory for trafficability to partition regions according to some criterion. They discuss both complex factor overlay and combined obstacle overlay as a means to do this. A complex factor overlay partitions regions according to the type of terrain and then combines them into areas with homogenous characteristics. A combined obstacle overlay characterizes the terrain according to its effect on the vehicular movement. Overlays are also discussed in [35].

Bonasso [4] presents a trafficability theory using first-order predicate calculus to enable reasoning about trafficable paths without the need for a detailed terrain model. The strength of this theory is also its weakness: only a limited set of qualitative values are used, which allows e.g. a desert to be described as having the attributes `surface-rigidity` SOFT and `forest-density` FREE. The only vehicle attribute is whether it has wheels or bands.

Much existing work on trafficability has focused on real-time applications, mainly path planning for unmanned ground vehicles (UGVs) which move in unknown terrain [7, 11, 20, 22, 9, 6]. There are a number of different approaches, some of which will be presented below.

A common technique is to use some kind of image segmentation, often together with statistical measures, confidence levels, or transition probabilities. Chaturvedi et. al. [7] use the hue information of colour images to classify terrain as road or non-road and have achieved good results under varying light conditions. However, their approach assumes a flat world terrain model. Ducksbury [11] segments images using a Bayesian network. Grunes et. al. [20] base their segmentation on texture, using a hidden Markov chain model. They identify road, grass, foliage, and brick and classify the first two as driveable and the latter two as avoidable, but do not consider other terrain attributes. Jasiobedzki [22] uses intensity images from a laser rangefinder to detect driveable floor regions in an indoor environment.

Another driveability model was developed by Digney [9]; later developed further in co-operation with Broten [6]. The authors argue that

many factors which affect trafficability cannot be represented in a geometrical model. These factors include soft ground, snow, mud, loose sand, compliant vegetation, debris hidden in vegetation, small ruts, and washboard effects. Also, terrain conditions are not constant, neither within a region nor over time. To solve these problems feedback sensors are needed together with learning algorithms. Image segmentation by the low level attributes size, shape, texture, and colour is used to characterize the terrain. For example, sand can be characterized as having the `colour` YELLOW, the `size` SMALL, the `shape` ROUND, and the `texture` SMOOTH.

Johnson et. al. [24] and Kruse et. al. [27] try to use hyperspectral or multispectral data to classify areas according to their surface composition, but with limited success. There are many problems which need to be solved before their results can be of any real use. Kruse et. al. improve the usefulness of their approach by using data fusion to identify areas which e.g. have both a high clay content and high slopes.

Sapounas et. al. [38] use an object-oriented approach and cost functions to calculate bounding regions and shortest paths. The cost functions represent the effect of the terrain on the travel speed.

To summarize, there are many possible feature types needed for determination of the driveability, using different combinations of features and feature properties:

- Donlon et. al. [10] represent the terrain using an overlay of physiography, hydrography, vegetation, and surface materials. Weather is also considered.

- Sapounas et. al. [38] consider elevation, weather, user-defined obstacles, water bodies, and threats when determining the driveability.

- Bonasso [4] considers surface rigidity, forest density and elevation.

- Chaturvedi et. al. [7] only distinguish between road and non-road features.

- Grunes et. al. [20] consider foliage and brick to be avoidable, and roads and grass to be potentially driveable.

- Kruse et. al. [27] discuss slope, soil composition, vegetation, manmade features, and drainage (e.g. streams, ditches, and lakes).

The features and feature properties used in this work are discussed in section 7.5.2.

2.2 Clementini geometry

Clementini geometry [8] is a calculus-based method for determination of topological relationships. Geographic objects are represented as points, lines, or areas, which are considered to have boundary, interior, and exterior (which may be empty). The five relationships *touch*, *in*, *cross*, *overlap*, and *disjoint* cover all possible topological relationships. There are also the two boundary operations *equal* and *contain*.

3. Driveability impact factors

Driveability is a complicated matter which does not lend itself to simple rules or simple solutions. It is affected by many factors, including the vehicle type, soil, weather, slope, etc. In this section some of these aspects will be discussed.

3.1 Direction

Driveability is direction dependent. It may e.g. be possible to drive down a slope, but not the other way around.

3.2 Environment

Most terrain features cannot be regarded in isolation. Below are some examples:

- A road barrier, a large stone, or a tree can only be seen as obstacles if it is impossible to drive around them.
- A wide ditch may be seen as an obstacle to a driver who must cross it, but not to a driver who can drive inside or around it, or use a bridge.
- A single tree may not be an obstacle, but a dense forest can be a great impediment.
- A lake is not driveable, but there might be a ferry route across it.
- A slope may not be too steep, but if the soil rigidity is too yielding it may be so. Vice versa, a mud field may be driveable on a plane area, but not at a slope.
- A ditch may not be driveable if it is filled with water, nor if trees or other obstacles are present.
- Vegatation and crops have an impact on the driveability of soils. E.g., grass and grain often improve the driveability, while vineyards decrease it. [35]

Some features or combination of features generally have a positive impact on the driveability. Among these are roads, ferry routs, bridges, etc.

3.3 Non-geometric and non-terrain features

Features exist which are not intuitively representable in a geometric model, but which may still affect the driveability. Surface roughness or vegetation may decrease the speed, damage the vehicle, or bring it to a stop. Both stickiness and sinking soil properties affect the driveability, but possibly in different ways.

Sometimes the effect is not strictly physical. Mine fields and radar installations may be examples of non-terrain features which decrease the wish to drive through an area, whereas a forest may be a good place to hide. Strictly speaking, these features should be dealt with in threat analysis instead of in driveability analysis, but they still deserve to be mentioned here.

3.4 Time

Many terrain features change their properties over time. Forests are growing denser and higher, rivers are bending, etc. During a war bridges, roads, etc., may be destroyed.

3.5 Weather

Weather properties may affect the terrain properties, thereby affecting the driveability. For example, soil rigidity is weather dependent. During a dry period a mud field may turn into a passable field. A cold winter may have turned an unpassable river into a frozen road. However, it is not sufficient to know what the weather is like right now; historical values must be taken into account as well. The lake will not be frozen simply because it is $-10°$C today; it must have been cold enough for a longer period.

Hard winds or floods may also affect the driveability.

3.6 Properties of vehicles and terrain features

Example of vehicle properties include width, length, height, override diameter, maximum gap to traverse, ground clearance, maximum step, maximum gradient, maximum tilt, specific ground pressure, and maximum straddle. Properties such as maximum speed is not of any direct interest to the driveability reasoning performed in this work, although it can be used as a basis for driveability calculations [38].

Sometimes terrain features and vehicles interact in unexpected ways. E.g., even if a vehicle can override small trees, the resulting pileup of vegetation may be a too great obstacle for it to pass. This effect is greater for wheeled vehicles than for tanks. [35]

Even if it is possible to define general terrain feature properties like width, height, length, slope, etc., they may not always be of interest to

a certain terrain feature. For example, the slope of a hill is probably of interest, whereas the slope of a river is not.

Properties may change over time and weather (see sections 3.4 and 3.5). Properties are not even constant for the same feature. A ditch, road, etc., may grow wider, or fork, which makes it close to impossible to define their width. The same is true for irregular shapes like e.g. lakes.

To start with, it may not be necessary to know the exact value of a property. It may be enough to use qualitative values (e.g. `width` = LARGE), which can later be transformed into real values, as is suggested in [4].

Due to the complexity of driveability, assumptions and simplifications must be made. The ones used in this work are described in chapter 7.

4. Input data

This chapter describes the input data sources used in this work.

4.1 Laser radar data

Terrain data has been gathered by a laser radar system called TopEye^{TM} [32, 40]. Data has been preprocessed to identify and remove buildings, trees, vehicles, etc. [3, 37] Data identified in this process can be used as a high-resolution information source. This type of information will further on be referred to as *labelled laser radar data*.

Coordinates are given in the RT90 coordinate system [14], whose main property is that the x-axis points to the north and the y-axis points to the east.

4.2 Digital terrain model

The digital terrain model [32, 33] consists of a set of templete forms, called *categories*. Each category represents some characteristic surface shape. Data from the laser radar system is divided into quadratic squares of size 2 x 2 m, here called *tiles*. A tile t is considered to belong to a category c representing surface shape s if t "resembles" s more than it resembles any other surface shape s' which is represented by category c'. A tile is replaced by its category in the terrain model.

Each category is described by its:

- Inclination. The average inclination of the tile.
- Orientation. The edge which divides the tile into two planes.
- State. Determines if the tile is concave or convex.
- Norm. A measure of the volume difference between the category and a flat tile.

The norm value of a tile can also be used to calculate the slope of the tile, see also figure 4.1:

$$slope = \arctan(\Delta z/b) \text{ and } \Delta z = \alpha_c * norm$$

where α_c is a constant that depends on the category and Δz is the difference between the highest and the lowest part of the tile.

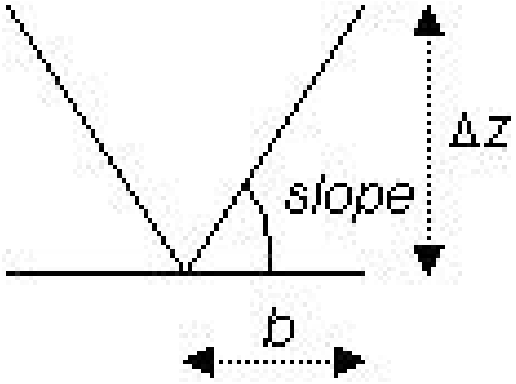

Figure 4.1: Calculation of $slope = \arctan(\Delta z/b)$.

Each tile is divided into nine subparts labelled according to figure 4.2. These integer labels will further on be referred to as DTM *points*.

4	3	2
5	0	1
6	7	8

Figure 4.2: The tile subareas.

The DTM points are used to describe inclinations and orientations. E.g., a tile which has its highest part in the north-east corner and whose orientation is in the north-west–south-east direction has inclination 2 and orientation 48 (which is to be interpreted as "an edge from DTM point 4 to DTM point 8").

4.3 Map data

Map data are commonly represented in *shapefile* data format [18]. A shapefile is a GIS data set which stores spatial information about geometric features, e.g. streets, rivers, hospitals, or country borders. A *feature* includes a number of *attributes* together with a geometric *shape* (point, line, or area) represented by vector coordinates. Similar feature types are often stored in *themes*.

Map data used in this work was taken from *the Real Estate Map* (SW. Fastighetskartan) [39, 31, 30, 2, 28], which according to the Swedish Land Survey contains the best and most detailed map information in this

scale. It consists of a number of shapefiles, representing different themes, e.g. roads, ground data, buildings, and water. Data in each theme is classified according to a certain *detail type* [31]. However, there is no further information about the features. An area might e.g. be classified as being of type VATTEN (water), but there is no information about the depth of the water. Roads, ditches, rivers, etc., are represented by their middle lines, but no width is given.

Some features are only represented by their boundary lines, e.g. nature reserves and military areas. Feature types in the same theme are hierarchically ordered, and adjacent features share boundary lines. This means that a single feature type may not give a complete classification of the area [2], see figure 4.3.

Coordinates are given in terms of the RT90 coordinate system (see section 4.1). Each shapefile is about 2.5 Mb in size and corresponds to a sheet in the economical map sheet system [29], where each sheet covers an area of 5 x 5 km. The scale is 1:10,000.

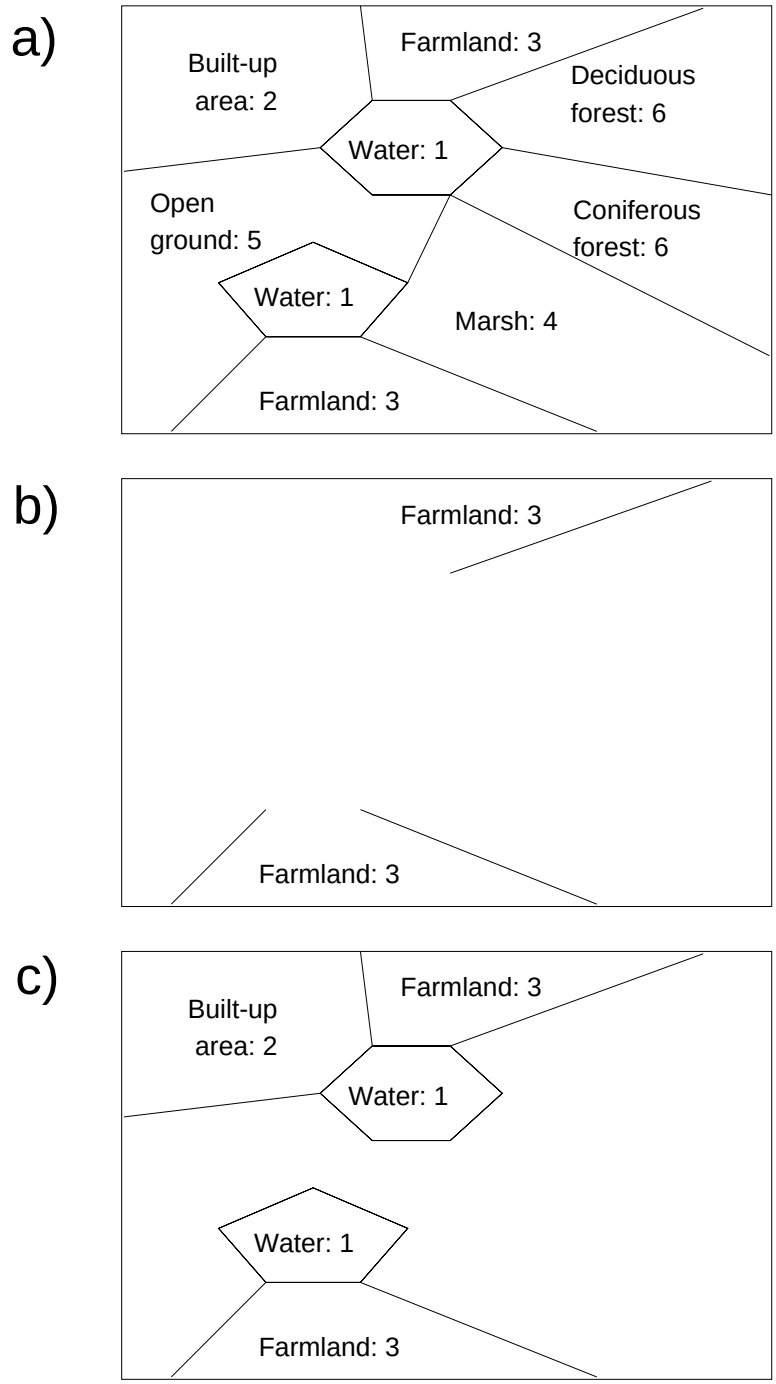

Figure 4.3: The boundary lines of hierarchical feature types in map data features. Adapted from [2]. a) The six hierarchies in the ML theme (line layer for ground data). b) Showing only farmland results in an incomplete classification. c) Farmland together with the hierarchically superior boundary lines gives a complete classification.

5. Software tools

This chapter describes the software tools used in this work.

5.1 Category Viewer

In [40], a filtering process was developed which uses the digital terrain model as input source. A syntax for description of terrain features was developed and used in the filtering process to find ditches, ridges, hills, ponds, flat areas, and roads. Each terrain feature to look for is described by a sequence of categories, called *segments*, representing its cross-section. Each segment consists of one or more subsegments, often a start segment, possibly a middle segment, and an end segment, see figure 5.1. By varying the length of the subsegments, features of different sizes can be found.

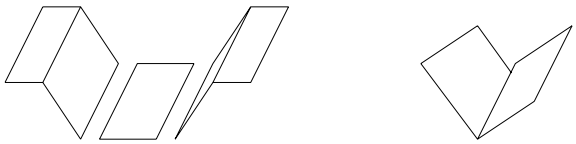

Figure 5.1: The filter structure to be applied to the cross-section of a ditch. Adapted from [33].

The segments are described in filters. The filters are applied in both horizontal and vertical directions to find features with all possible orientations. During the filtering process Category Viewer finds horizontal and vertical segments, which correspond to the segments given by the filters. The result contains false hits, which can be eliminated by applying one of three connect algorithms:

1. Simple connect, which connects adjacent tiles.
2. Connect segments, which connects adjacent segments.
3. Connect edges, similar to the simple connect algorithm, but which also considers the orientations and inclinations of the tiles.

The implementation of the filtering process is called Category Viewer. An overview of the data flow in Category Viewer is shown in figure 5.2.

The user supplies an area name and one or more filter files. This information is used to fetch the DTM data corresponding to the area and information about which tiles and tile segments to match. The DTM data and the filter information are matched, resulting in multiple segments. They are connected to features by applying a connect algorithm, the name of which is given in the filter file. The result is then visualized, see figure 5.3. The red (dark in black and write printing) squares are the tiles which were matched during the filtering process, the black lines are orientations and the green (gray) lines are inclinations.

The parameters of the connect algorithm are minimum/maximum length and minimum/maximum width of the final terrain feature. The connect segment algorithm also supports a maximum error parameter, specified in per cent.

5.2 MapObjects

MapObjects [15, 16, 17, 1] is a collection of Java GIS and mapping components, which can be used for visualization of geographic information. Queries and data retrieval can also be applied. It also has built-in Java beans e.g for zooming and panning. The version used in this work is MapObjects Java Edition v. 2.0.

MapObjects has a built-in support for Clementini geometry [15, 16] and was used in the ISM project.

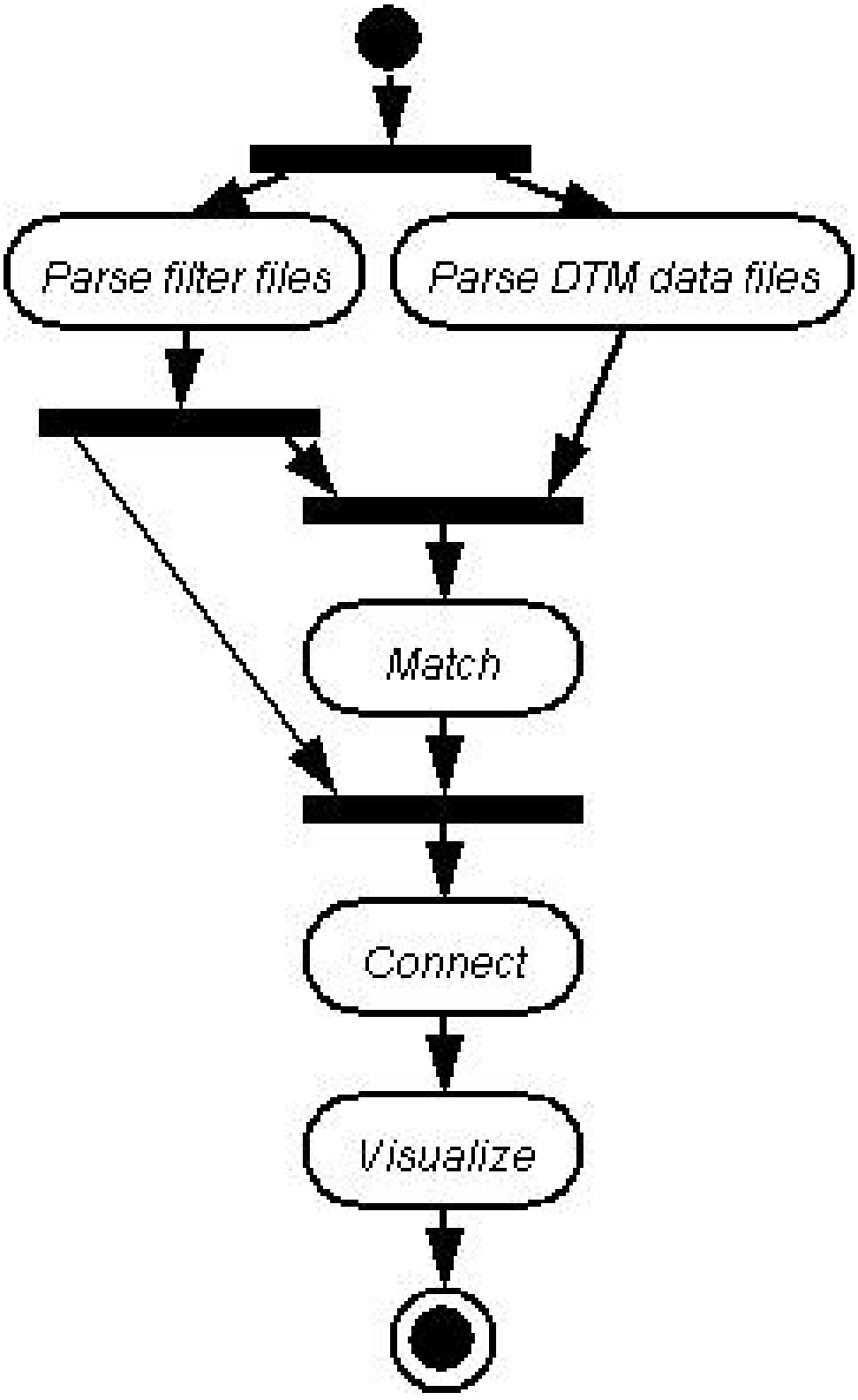

Figure 5.2: Overview of the data flow in Category Viewer.

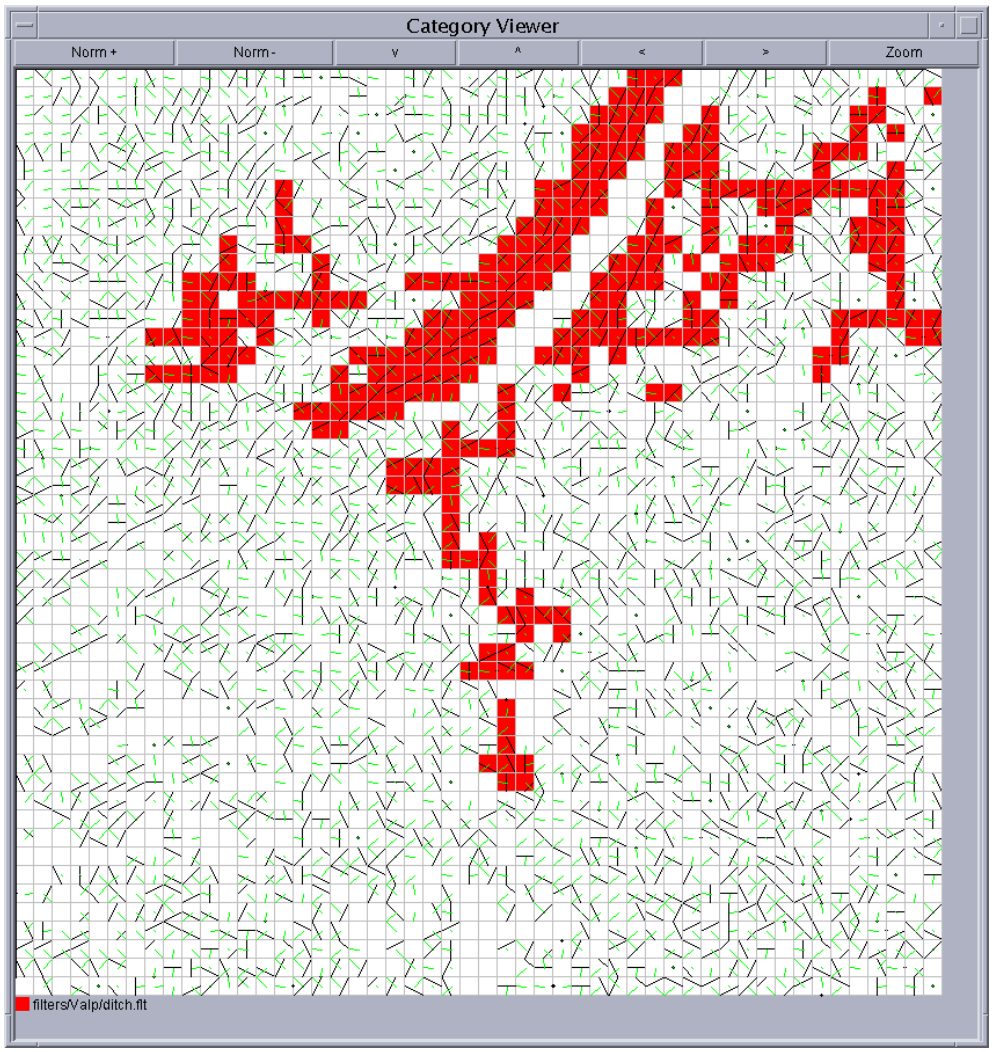

Figure 5.3: Category Viewer using the ditch filter with a minimum norm of 0.5.

6. Driveability measures

The area of interest is divided into as largely connected regions with uniform characteristics as possible, see figure 6.1. Regions 1 and 2 both consist of forest only. Region 3 consists of a ditch only, whereas region 4 consists of forest and ditch.

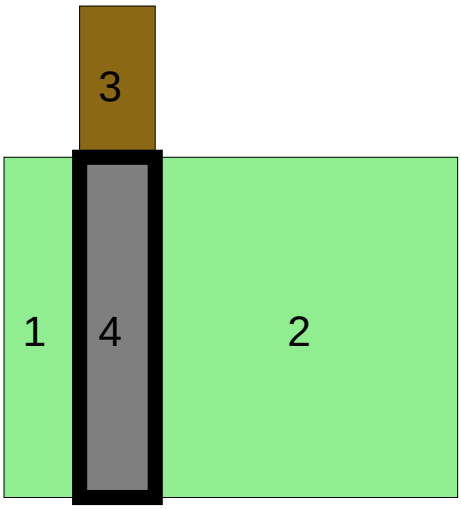

Figure 6.1: A partition of the area of interest into regions with uniform characteristics.

Each region is here considered to be potentially driveable in eight directions, called driveability paths, see figure 6.2. The driveability is the same in the whole region and depends on which impact factors (see section 6.1) that are present in the region.

Thereby, the problem of finding a driveability measure can be compared to an ordinary path planning problem, which includes finding the cost of path P from A to B, i.e.

$$\Gamma(P) = \int_P \gamma(x, y)\, dx\, dy$$

where $\gamma(x, y)$ is the cost at location (x, y).

Each driveability path in a region is assigned a cost. The total cost $\Gamma(\mathrm{R}, \mathrm{V})$ in a region R for a vehicle V is such that

$$\Gamma(R, V) \in [0, \dagger]$$

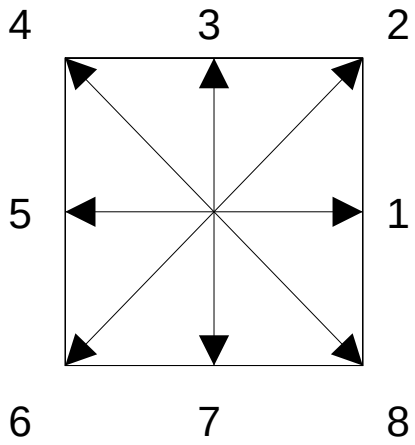

Figure 6.2: The possible driveability paths and their labels (compare with the DTM directions in figure 4.2).

where † means totally undriveable and all costs less than † mean driveable, i.e. a lower cost represents a higher driveability.

Normally, a relation $r(x, y)$ is a binary truth function. The cost function Γ can be seen as an extended relation between the region R and the vehicle. This interpretation is consistent with the definition in section 1.3.

The cost function Γ must have certain properties:

- The costs assigned by Γ can be different for all driveability paths.
- Which driveability paths that are affected often depend on the direction of the features in the region. E.g., a cost may only be assigned to the "upwards" direction of a slope.
- Γ depends on all the factors affecting driveability that were described in chapter 3, e.g. the properties of the terrain features in region R. A slope may have cost Γ_1 and the soil a cost Γ_2. Combined they have a cost Γ_3, but it is not unlikely that $\Gamma_3 \neq \Gamma_1 + \Gamma_2$.
- It must be possible to represent costs which are infinite in some sense. A mine field is undriveable regardless of what properties other features may have.
- Different vehicles have different properties, and therefore different demands on the terrain. Therefore, costs are somehow related to the vehicle type.
- It must be possible to assign different cost functions to different feature types. Some terrain features have a discrete impact on the driveability (driveable or undriveable), some have a continous impact, and some have a combination of both. For example, slopes

smaller than the maximum slope capability of the vehicle have a continuous impact, but slopes larger than this limit have a discrete impact (undriveable), regardless of their actual value. A cost function may not even be linear. In figure 6.3 an example cost function for slopes is given. Small slopes have a small impact on the driveability but at a certain point (A) considerably more engine power is needed to climb the slope and the cost increases rapidly. When the slope is too steep for the vehicle to climb the cost has reached its maximum (B).

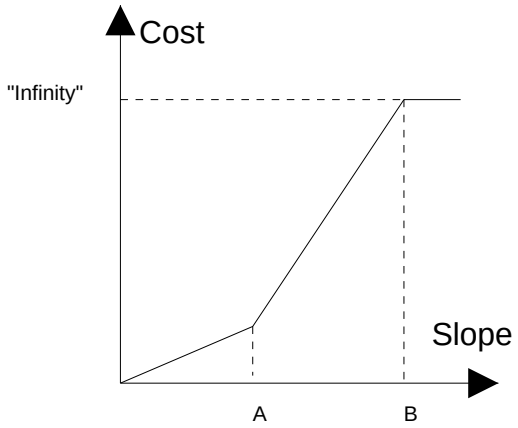

Figure 6.3: Example of a cost function for slopes.

If the calculation time is critical and the user is only interested in whether the area is driveable or not, cost calculations could be stopped if the total cost has exceeded the given limit for what is considered driveable.

6.1 Definitions

A part of the ISM project was to describe the influence of certain factors on the selection of sensors and algorithms. Thus, the concepts *impact factor*, *discrete strength value*, and *impact strength value* were defined [21]. The impact of these are given as rules in the following form:

> If an impact factor **x** has the discrete strength value **y** then the impact on the sensor/algorithm **z** is impact strength value **v** [13], where **v** $\in$ {NONE, LITTLE, MEDIUM, STRONG, COMPLETE} [21].

Example 6.1 If the impact factor **Rain** has the discrete strength value **Gentle** then the impact on recognition algorithm **GeometricFeature-Extraction** is impact strength value LITTLE [13], i.e. gentle rain has little negative impact on this particular algorithm. If the impact strength value had been COMPLETE the algorithm would have been completely useless under this condition.

Driveability impact factors can be e.g. ditch width, road quality, vehicle weight, etc, as discussed in chapter 3. Since impact factors often interact to affect the driveability in a complex way it is not enough to use the ISM rule definition. Instead, *relations* between impact factors must be allowed.

Example 6.2 If the relation ">" holds between the discrete strength values of the impact factors **angle of slope** and **vehicle slope capability** in region **R** then the impact on the driveability in R is impact strength value STRONG.

The impact strength value NONE should be interpreted as having no effect on the driveability, whereas the impact strength value COMPLETE should be interpreted as making driving in that region impossible. Here, this will be illustrated by assigning the following numbers to the impact strength values:

Impact strength value	*Assigned number*
NONE	0.0
LITTLE	0.5
MEDIUM	1.0
STRONG	1.5
COMPLETE	2.0

From now on, *impact strength value* will simply be referred to as *impact*.

6.2 Driveability knowledge base

Impact rules for a vehicle are collected in a driveability knowledge base. Each rule consists of a set of impact conditions.

Definition 6.1 An **impact condition** can be either binary or unary.

Definition 6.2 A **binary impact condition** $\kappa(f_1, f_2, rel)$ is satisfied in a region R if the relation $rel(f_1, f_2)$ holds in R, where f_1 and f_2 are impact factors and $rel \in \{<, <=, =, >=, >\}$.

Definition 6.3 A **unary impact condition** $\kappa(f_1, rel)$ is satisfied in a region R if the relation $rel(f_1)$ holds in R, where f_1 is an impact factor and $rel \in \{$`isFalse`, `isTrue`, `exists`$\}$.

Example 6.3 f_1 = angle of slope, f_2 = vehicle slope capability, and rel ="$<$".

Definition 6.4 An **impact rule** $\Upsilon(\mathrm{K}, \mathrm{P}, \mathrm{R})$ consists of a set of impact conditions K and a set $\mathrm{P} = \{(\rho_1, \iota_1), \ldots, (\rho_8, \iota_8)\}$, where (ρ_i, ι_i) means that the impact of Υ along driveability path ρ_i is ι_i if K is satisfied in region R. An impact rule can be either an AND impact rule or an OR impact rule.

Definition 6.5 An AND **impact rule** is satisfied if $\forall\kappa \in \mathrm{K} : \kappa$ is satisfied in R.

Definition 6.6 An OR **impact rule** is satisfied if $\exists\kappa \in \mathrm{K} : \kappa$ is satisfied in R.

Note that an unsatisfied impact rule may be due to an unsatisfied relation, or to lack of the information needed to check the relation.

The implemented knowledge base is described in section 7.9.

6.3 Cost function

In this work, the impact is seen as a type of cost. Thus, using the knowledge base, the cost in a region R for a vehicle V can be described as

$$\Gamma(\mathrm{R}, \mathrm{V}) = \sum_i g(\Upsilon_i(\mathrm{K}, \mathrm{P}, \mathrm{R}))$$

where

$$g(\Upsilon_i) = \begin{cases} \text{impact of } \Upsilon_i & \text{if } \Upsilon_i \text{ is satisfied in region R;} \\ \text{no impact} & \text{otherwise;} \end{cases}$$

An impact condition can be non-zero for one or more driveability paths.

6.4 Restrictions

In this work, two restrictions were made to the definitions in the previous sections:

- Each tile is assigned the same driveability properties as the region it is part of. The cost calculation is then performed per tile instead of per region.
- An impact rule has the same impact ι on a limited number of driveability paths $\mathrm{P}' = (\rho_1, \ldots, \rho_n)$ and a zero-impact on all other possible paths. Thus, an impact rule can be described as $\Upsilon(\mathrm{K}, \iota, \mathrm{P}')$.

6.5 Data availability and reliability

There are not always enough data available to calculate the costs. Data may be missing for either the vehicle or the terrain features, or both. In such cases, costs may be calculated by assuming the best case ("driveable"), the worst case ("undriveable"), or by using default reasoning. Regardless of the chosen approach, assumptions make the result less reliable than real values.

Which properties that are available does not only depend on the feature type, but also on the data source. The depth of a river is known

when acquired from DTM data, but not from map data. The resolution also differs for these data sources, affecting reliability.

Even if all needed data are available they may not be reliable. Data may have changed, due to time, weather, or human activities (e.g. destroyed bridges).

Too low data resolution also decreases reliability. However, it is not always the data with the highest resolution that has the highest reliability. There is always a balance between resolution and reliability.

Using multiple data sources does not always lead to a higher reliability, since errors from the different sources may add up to a larger total error.

The reliability notion must be clearly defined. Possible interpretations of reliability could e.g. be quality of input data, or quality of the used cost calculation method.

Due to the limited time available for this work the reliability notion has neither been defined nor used. However, it has been prepared for in the implementation, see section 7.8.

6.6 Default reasoning

Using default reasoning is problematic. Default values or assumptions may mean best cases when calculating some costs and worst cases when calculating others, with a high unreliability and confusion as a result. E.g, should default vehicle data be taken from a Japanese private car or a large military vehicle? Should a forest be assumed to be too dense to drive through? Should a vehicle be assumed to be unable to override any trees? The chosen approach will have great impact on the calculated driveability. Thus, if default values are used this must be indicated in some way, e.g. by decreased reliability.

7. Method

In this chapter the actions taken to develop a driveability calculation method and a visualization method will be discussed. The implemented program is called Driveability Viewer, and can be seen as a development of Category Viewer. An overview of Driveability Viewer is given in figure 7.1. The user supplies an area name and a vehicle type name. This information is used to determine which map data and DTM data to fetch. Map data are fetched from shapefiles and DTM data by a call to Category Viewer. As will be described in section 7.4, a postprocessing algorithm is then applied to the DTM data. During cost calculation feature properties are calculated. Finally, the result is visualized.

Appendix B contains a simplified UML class diagram for Driveability Viewer.

7.1 Test data

The test data to be used in this work were taken from laser radar flights across Kvarn, northwest of Linköping, Sweden, in August 2000. The total area covered is 800 x 800 m, divided into 64 areas of 100 x 100 m each. Each area consists of 50 x 50 tiles of size of 2 x 2 m, each tile representing a resolution of approximately 0.5 m.

Map data were taken from the Real Estate Map. There is also a map image available in TIF-format which can be used as a background for visualization. This image has a lower resolution than the shapefiles but covers a larger area.

Two vehicle types were tested: *T72* and *Volvo Valp*. Some of their properties are given in appendix A. These vehicle types were chosen since they represent vehicles of different sizes and capabilities. The T72 is a tank, whereas the Valp is a cargo/troop carrier.

The testing of the program was mainly done by colour coding and visualizing of the result of the ditch filter in different ways, using the TIF-file or one or more shapefiles as a background. The results are presented in chapter 8.

7.2 Communication with Category Viewer

As filter results from Category Viewer are to be used by Driveability Viewer, some means to communicate between the two programs had to

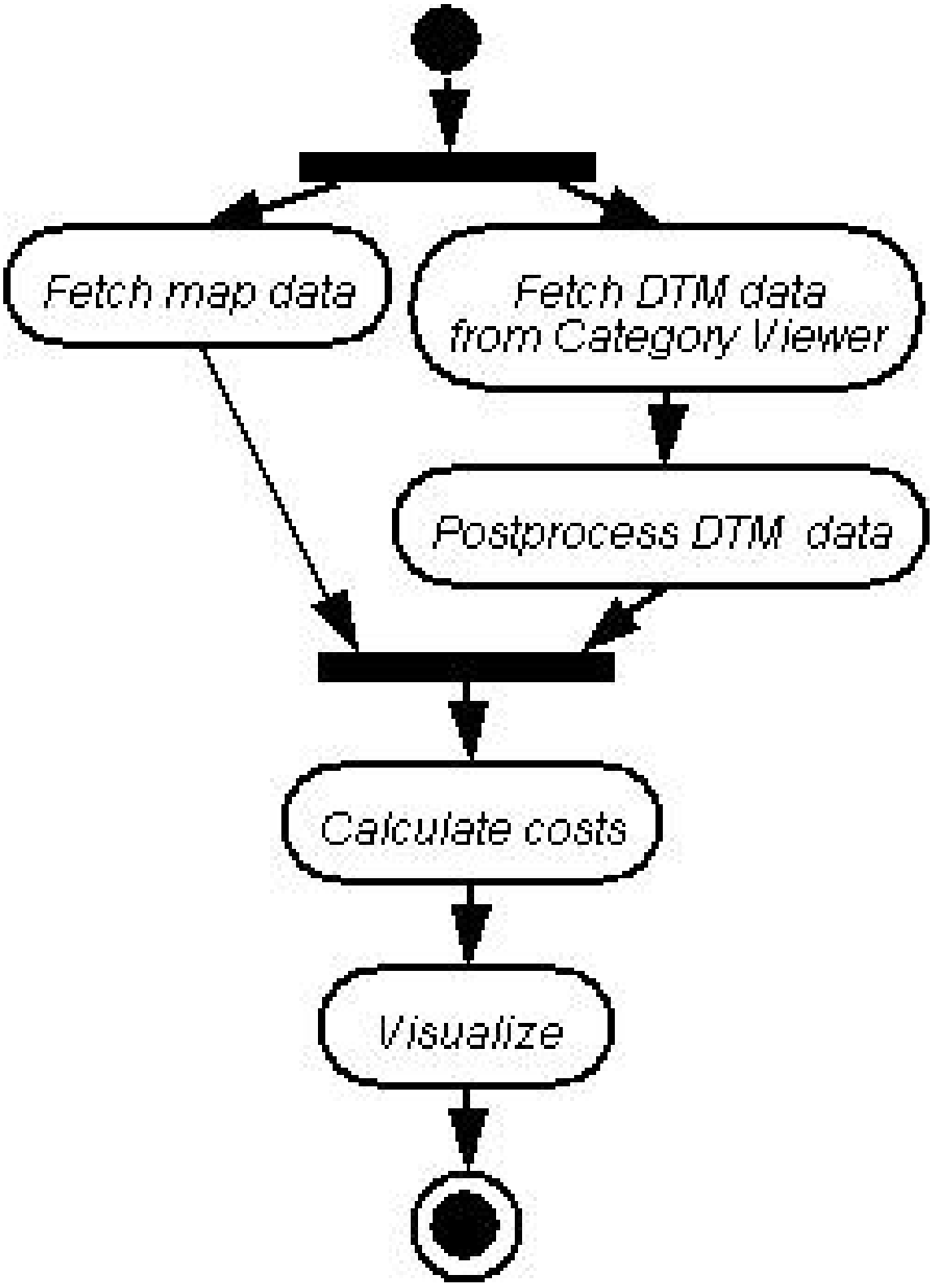

Figure 7.1: An overview of Driveability Viewer.

be added. Data are retrieved via a direct method call, guaranteeing data freshness.

To be able to distinguish the features an id number is added to each feature.

Category Viewer considers a tile to be flat when its norm is smaller than a user-defined limit, called the minimum norm. The same minimum norm is used by all the filters during the same session. To enable different minimum norm settings for different filters, an optional `norm` key word was added to the filter definitions. The syntax is

```
norm <minimum norm value> <maximum norm value>
```

The maximum norm parameter is optional and is not used, but is available for possible future use.

Appendix C contains a simplified UML class diagram for Category Viewer.

7.3 Problems with Category Viewer

When exploring the results from Category Viewer two problems concerning the world coordinates appeared. The first was that Category Viewer had switched the world coordinates. In Category Viewer this is of less importance since a local coordinate system is used for visualization. The second problem was that the preprocessing step between laser radar raw data and Category Viewer produced incorrect world coordinates. These problems have now been corrected.

As the simple connect algorithm and the connect segments algorithm do not consider the actual shape of the terrain features this work focuses on the more realistic connect edge algorithm. However, this algorithm turned out to be a flaw. This could be seen in several ways:

- Much noise was classified as features. When looking at the inclinations and the orientations of the tiles it was hard to see why they had been connected.

- Segments with no connecting tiles were sometimes considered to belong to the same feature.

- Adjacent segments were sometimes considered to belong to different features.

- Tiles were connected to features in different ways depending on how many 100 x 100 m subareas which were used as input data.

Another problem was that ditches on different sides of a road were sometimes considered to belong to a single ditch if the minimum norm was set too low.

7.4 Improving the connect algorithm

As the results of the connect edge algorithm were not sufficient a post-processing step was added to Driveability Viewer. This step overrides the grouping of tiles into features by recursively connecting all adjacent tiles. Next, all features with too few tiles are removed, and finally an id number is used to label the features. In Category Viewer terms this can be seen as the connect simple algorithm being applied immediately after the connect edge algorithm.

To solve the problem with ditches overlapping the roads an additional approach is used. The tiles determined by the ditch filter are compared with the roads from the map data. All tiles crossing a road are removed. The improved connect algorithm is then applied as usual, causing the ditches on each side of the road to be separated.

To determine the relationship between tiles and roads the built-in Clementini geometry of MapObjects is used.

7.5 Data formats and representations

In this section the used data formats and representations will be described.

7.5.1 Representation of the world The world, i.e. the tile objects, are represented using the Java `Map` interface, where the tile locations serve as keys. Thus, only feature tiles need to be represented. The used implementation was `TreeMap`, since it was necessary to be able to get a tile at a certain location without knowing the tile beforehand, and the (faster) `HashMap` implementation cannot solve this problem. The `TreeMap` implementation provides guaranteed $\log n$ time cost for the most common operations, including `get` and `put` [41].

7.5.2 Representation of feature types Features belonging to the same feature type affect the driveability in a uniform way, and can hence be grouped together. Thus, an overlay strategy is suitable to represent different feature types. In this work a number of layers were constructed for testing purposes, regarding the combinations in section 2.1 as suggestions:

1. A *surface rigidity layer* of marshes and waters bodies (from map data). All marsh types are considered equal in driveability aspect. Other possible surface rigidity types are sand, clay, etc., but they are not directly available from the map data. The purpose of this layer is to check the bearing capacity (water bodies) and frictional resistance (marshes) between the ground and the vehicle wheels, often combined with a check of slope steepness.

2. A *slope layer* of slopes (from DTM data) which may affect the driveability of the vehicle. The main slope property is the slope angle. Slopes must be compared with the surface rigidity; a small slope in a sandy area may cause as much impediment to the vehicle as a steep slope with a hard surface. Thus, rather low limits must be set in the slope filter definitions.

3. A *cavity layer* of ditches (from DTM data). Of course, ditches have slopes too, but this layer can also be used to find alternative paths, e.g. if a ditch can either be crossed or followed up and down its sides. Consequently, calculation of these feature properties differ from calculation of slope feature properties, which is why separate slope and cavity layers were constructed. Cavity properties include width, length, shape, angle of slope, slope direction, direction, etc.

4. A *forest layer* (from map data). Forest properties like tree density (stem spacing), stem size, etc. can be used to check for too dense forests.

5. A *building layer* of houses, outhouses, and churches (from map data). There are specified types for settled areas (as opposed to individual buildings) available from map data but as the Kvarn area is relatively rural these data does not often occur. Building properties include length, width, height, and material.

6. A *road layer*. A road filter is available in Category Viewer but the roads here are taken from map data. Width and quality are typical road properties, but for testing purposes all available roads are considered equal in driveability aspect.

Each layer can contain features from both DTM data and map data even though this possibility is not used here. Features can be retrieved from the same or different layers and compared.

The layers above were chosen because they include terrain feature types which may affect the driveability, both positively (e.g. roads) and negatively (e.g. water). They also affect different parts of the cost function (see section 7.9). There is no layer which contains threats as there are no mine fields, radar installations, etc. in the Kvarn area.

Of course there are more feature types which could be added (e.g., bridges and ferries are good examples of features with positive impact on the driveability) but for testing purposes the chosen feature types are sufficient.

7.5.3 Representation of map data The original map data consists of shapefiles, which is a form of vector structure. It is possible to rasterize shapefiles using ArcGIS Spatial Analyst [1], or by adjusting the method presented in [19]. However, MapObjects can present raster data but does

not allow any kind of data analysis. Therefore, the used approach was to construct vector polygons from DTM features and compare them with other vector features using Clementini geometry.

7.6 Creating filters

Category Viewer offers filters for ditches, ridges, hills, ponds, flat areas, and roads [40]. In this work an additional slope filter has been developed to be used when calculating the costs.

Filter definitions include minimum and maximum feature length and width, and have here been extended with minimum and maximum norms. To utilize Category Viewer maximally, filters should be adjusted according to the requirements of the specified vehicle types. In this way, a lot of the necessary calculations will be made in the filtering process. Hence, the filters can be viewed as a vehicle property.

As little is known about the correlation between vehicle properties and landforms the same filter defintions are used for both vehicle types.

7.7 Calculating terrain feature properties

Some properties can be determined for DTM features but not for map features, e.g. road width and slope. In the latter case the norm values of the tiles are used. Some properties cannot be calculated for any feature type but must be hard coded, e.g. soil rigidity.

Properties which cannot be calculated are set on a per feature type basis, which means that all forests are given the same density, all rivers the same current, etc.

Properties which can be calculated are set on a per feature basis. This means that a ditch has the same width everywhere, that all trees in a forest have the same trunk size and the same stem spacing, etc.

7.7.1 Slope The slope of a DTM feature is calculated as the maximum slope of any of its tiles. The slope of a tile is calculated from its norm value as described in section 5.1.

7.7.2 Direction of DTM features The most frequently occuring tile inclination of a DTM feature is considered to represent the slope direction of the feature. If more than one inclination have the same frequency the norms of all tiles with the same inclination are added and the inclination with the highest norm sum is returned.

The same method but based on tile orientations can be used to determine the direction of e.g. DTM roads.

7.7.3 Direction of map features The direction of a map road (or any other line geometry) in terms of DTM points is determined as follows:

1. Determine the straight line L between the start and the end of the map road.

2. Calculate the angle α between L and the horizontal axis.

3. Translate α to the closest DTM point.

7.7.4 Area How area calculations are performed depends on the feature data source and geometry:

- The area of a DTM feature is a multiple of the area of a tile.

- The area of a map feature which is represented by a polygon (e.g. a forest) is calculated by the `getArea` method from MapObjects.

- The area of a map feature represented by a line (e.g. a road) is unknown since there is no information about the feature width.

7.7.5 Width and length of DTM features Width and length estimations of DTM features are calculated by the connect edge algorithm in Category Viewer, but because of that implementation new measures must be calculated. The chosen method calculates the mean horizontal segment length and the mean vertical segment length and uses Pythagoras' theorem and area calculations to get a width estimation, see figure 7.2d. The area A of the triangle made up by l_h, l_v, and l_{hyp} is given either by

$$A = \frac{l_h * l_v}{2}$$

or

$$A = \frac{l_{hyp} * w}{2}$$

giving the width

$$w = \frac{l_h * l_v}{l_{hyp}}$$

To do this new segments must be identified. The new segmentation algorithm does not consider orientations or inclinations but mainly connects all adjacent tiles in the horizontal and vertical directions. This implies that every tile is part of exactly one horizontal and exactly one vertical segment, in contrast to Category Viewer, where a tile not necessarily belongs to both a horizontal and a vertical segment.

The relation between area, length, and width is roughly estimated as $area = length * width$. This will however not give a satisfactory result for irregular or circular features.

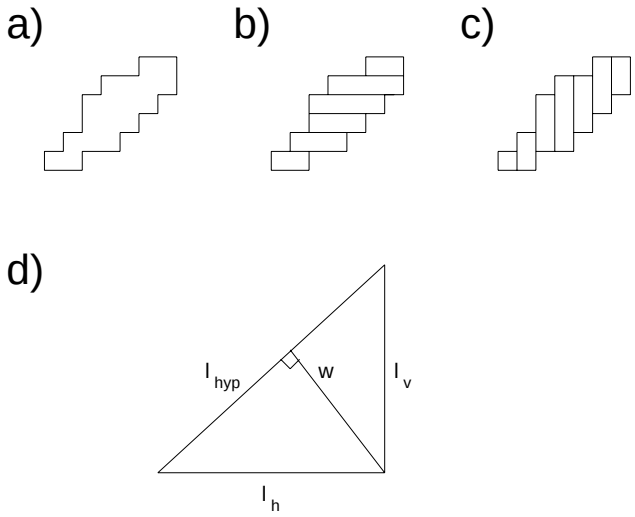

Figure 7.2: Width estimation using average segment length and Pythagoras' theorem. a) The feature. b) The horizontal segments, average length $l_h = 3.0$ tiles. c) The vertical segments, average length $l_v = 2.6$ tiles. This gives $l_{hyp} = \sqrt{3.0^2 + 2.6^2} = 3.97$ tiles and width $w = \frac{3.0*2.6}{3.97} = 1.96$ tiles.

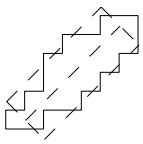

Figure 7.3: Using the centre line of a feature to demonstrate the result of the width estimation. According to figure 9.1 the width estimation of this feature is 1.96 tiles.

7.7.6 Width and length of map features When calculating width and length of map features there are two cases: the feature is represented by a polygon, e.g. forests, or by a line, e.g. roads. If all polygon shapes are assumed to be circular both width and length can be estimated by the diameter of the shape, which is calculated from its area. As line shapes only represent the centre line of the feature, the width of the feature is unknown. Its length can be estimated by MapObjects' `getPerimeter` method which returns the length of the line.

7.7.7 Centre line Finding the centre line of a DTM feature and using it together with the width estimation in section 7.7.5 as shown in figure 7.3 gives an idea of how good the estimation is. Of course, the centre line is only meaningful for some feature types, e.g. ditches and roads.

The basic approach can be described as follows:

1. Find the centre coordinate points of all horizontal and vertical segments.

2. Sort the points in an appropriate way and connect them.

3. Remove points which seem to be outliers, e.g. points which are too far away from the others.

4. Remove points that have little or no influence on the shape of the line in order to "smooth" it.

5. Remove line segments which are too short to be of any interest since they probably originate from noise.

In the first step, the "centre" means the geometric centre.

In the second step, an arbitrary point on the line is chosen as the starting point p_0, see figure 7.4. The next point, p_1, is chosen as the point closest to p_0. The search then continues from p_1 until no further point is found within a certain distance. The search then goes back to p_0 to see if there is any other point p_{-1} which can be connected to p_0. If there is no point close enough a new line is started to avoid too strange lines. Hence, a feature may have more than one centre line.

In the third step, two variations of the same methods were tried. For both variations a measure $d = f(p_1, p_2, p_3)$ is calculated where p_i and p_{i+1} are adjacent points along the line, see figure 7.5:

- In the first method, d is the distance between p_2 and p_{13}, where p_{13} is the location of the middle of the straight line between p_1 and p_3.

- In the second method, d is the angle p_1-p_2-p_3.

If d is larger than some predefined limit d_{max} all p_i:s are penalized since it is impossible to know which one of them that "caused" the limit to be exceeded. When all points on the line have been checked, points which have been penalized too many times are removed from the line. Start and end points must be treated as special cases, but no good method has yet been found.

The fourth step can be achieved in the same way as the third, replacing d_{max} with d_{min}.

7.7.8 Boundary The boundary of a DTM feature is calculated by using its centre lines and width estimation. In this case, the "boundary" is a set of coordinate points. Each coordinate is calculated by moving half the width perpendicular to the current centre line segment, see figure 7.6:

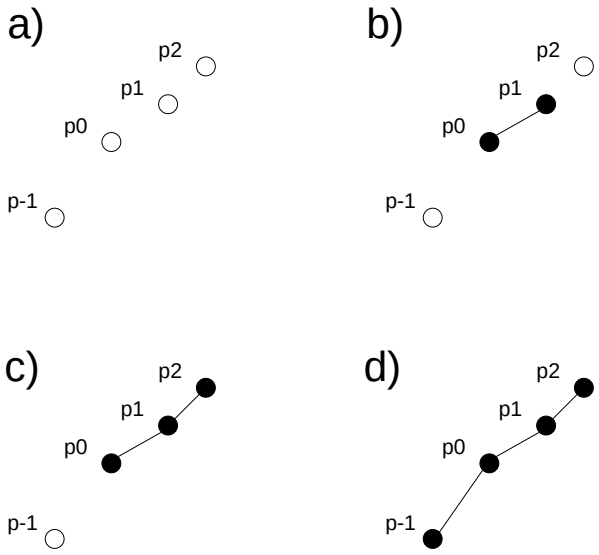

Figure 7.4: Algorithm to determine the order of the points on a line. a) The original set of points. A starting point p_0 is arbitrarily chosen. b) p_1 is the point closest to p_0 and is therefore chosen as the next point on the line. c) p_2 is chosen as the next point on the line as it is closest to p_1. d) There is no point close enough to p_2, so the search is resumed from p_0 and p_{-1} is found.

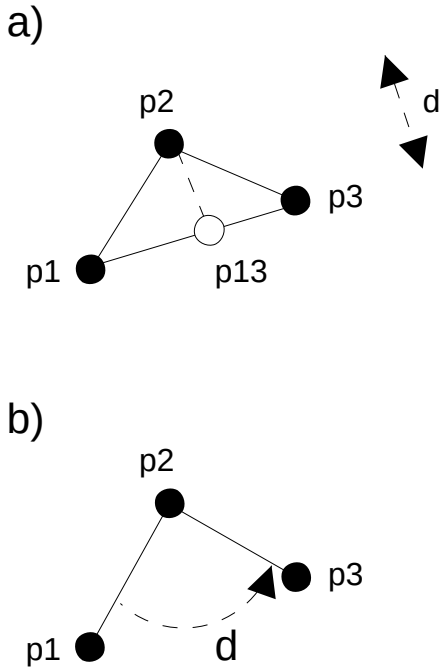

Figure 7.5: Line filtering using a) distances and b) angles.

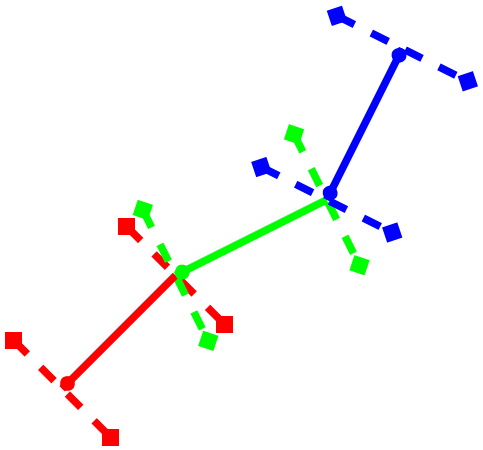

Figure 7.6: Calculation of boundary coordinates (marked with diamonds). The solid line corresponds to the centre line of the feature. All dashed lines have a length equal to the width estimation of the feature.

Let $d = 0.5 * featurewidth$
For each centre line L
  For each segment S in L between coordinates c_1 and c_2
    1. Start in c_1 and move a distance d perpendicular to S
    2. Start in c_2 and move a distance d perpendicular to S
    3. Repeat steps 1 and 2 but in the other perpendicular direction

7.8 Data availability and reliability

Each property is associated with a key, a value, and a reliability attribute. The key is used to access the corresponding property value. If the value is not present, a `null` value is returned.

Reliability is in this case a measurement of how confident a user can be of the property value. For example, the number of wheels of a vehicle is probably very trustworthy, whereas its slope capability may not. A number of discrete constants like SMALL, MEDIUM, and LARGE have been defined to represent the corresponding quantitative values. If these constants are used reliability may decrease. Property reliability is also affected by the data resolution, the property calculation method, etc.

Property calculations are not carried out until the property is requested, to follow the ISM philosophy. If a property value that is normally calculated has been set in other ways for an individual feature the calculation is not performed. The reason for this is to allow for future user input via a user interface, as discussed in section 9.4.

7.9 Cost function

A small knowledge base has been developed. The driveability is calculated tilewise, since the cost depends on all features at that location. Thus, for each tile in the area of interest each rule in the knowledge base is applied. When a rule is satisfied the cost of the tile is increased by the impact of that rule.

To illustrate that not all features have a negative impact on the driveability the cost is divided into three parts:

- An *impediment part*, which is affected by impact conditions which decrease the driveability.
- An *improver part*, which is affected by impact conditions which increase the driveability.
- A *threat part*, which is affected by threats, mainly non-terrain features such as military areas, mine fields, or radar installations. As discussed in sections 3.3 and 7.5.2, this part does not strictly belong to this work.

Thus, the impact of a rule has two parts: which part of the cost it impacts, and the size of the impact.

The feature types which are used to illustrate the cost function are given in section 7.5.2. To illustrate the driveability calculation method a number of restrictions are made:

- Two features in the same layer never overlap.
- Surface rigidity can be LIQUID, MUSHY, SOFT, YIELDING, MALLEABLE, or HARD, as suggested in [4].
- Stem spacing can be FREE, THIN, MEDIUM, or DENSE.
- Stem diameter can be THIN, MEDIUM, or THICK.

Also, some assumptions are made:

- All water bodies have surface rigidity LIQUID.
- All marshes have surface rigidity MUSHY.
- All forests have stem spacing MEDIUM and stem diameter MEDIUM.

The rules in the knowledge base are described informally below:

1. If the surface rigidity is LIQUID or MUSHY then the impediment impact is COMPLETE for all paths.
2. If the surface rigidity is SOFT and the vehicle has wheels then the impediment impact is COMPLETE for all paths.

3. If the cavity width is greater than the vehicle gap capability then the impediment impact is LITTLE for the path in the slope direction of the cavity. (There may be other ways to traverse the cavity.)

4. If the angle of the slope is greater than the vehicle slope capability then the impediment impact is STRONG for the path in the slope direction.

5. If the vehicle is a tank and the stem spacing is greater than or equal to MEDIUM and the stem diameter is greater than or equal to MEDIUM then the impediment impact is STRONG for all paths. (A tank is assumed to be a large vehicle but also capable of overriding smaller trees.)

6. If the vehicle is a carrier and the stem spacing is greater than or equal to DENSE then the impediment impact is STRONG for all paths. (A carrier is assumed to be a relatively small vehicle, thus being able to pass between the trees.)

7. If there exists a building then the impediment impact is COMPLETE for all paths.

8. If there exists a road then the improver impact is LITTLE for the path in the road direction and the opposite road direction. (Nothing is known about the road width or quality.)

More rules can easily be added.

The testing of the conditions of a rule results in either SATISFIED, UNSATISFIED, or UNKNOWN. The last-mentioned result occurs when a property that is needed for the test is unavailable. The cost is only affected if the result is SATISFIED for all conditions (AND rule) or at least one condition (OR rule) of the rule.

8. Results

This chapter describes the achieved results, which are discussed in chapter 9.

The user interface is not the main objective of this work. However, there is one issue which deserves to be mentioned: which information should be shown? The user may want to know what has affected the driveability, if any assumptions have been made, if any default values have been used, data reliability, etc. It may be interesting to see results both from best and worst case reasoning. However, information abundance may confuse more than it helps.

A focus of the ISM project was to explore the information required to make a user trust the calculated data. Research in this area is to be continued.

8.1 General user interface

A starting point was to develop a user interface similar to that of Category Viewer. Hence it is possible to show DTM tiles and features and their inclinations and orientations.

In Driveability Viewer it is possible for the user to choose a quadratic area with a side length of $100i$ m, where $i \in \{1, 2, \ldots, 8\}$, whereas $i \in \{1, 8\}$ for Category Viewer. The area of interest can be shown using colour or a grid.

Driveability Viewer enables the user to update the minimum norm settings for individual filters, see section 7.2. It is also possible to change the filter definitions and update the visualized result without restarting the program.

Visualization is made on different, semi-transparent layers, similar to the overlay approach used for terrain feature types, see section 7.5. Terrain features, driveability, and available data labellings are shown on different layers.

Zoom and pan tools have been added, and some information about the available features are shown when the mouse is dragged across the map. This information includes feature id, feature type, etc.

Most visualization properties and other settings can be set through the `Properties` class before compilation. Some settings can also be changed via menus during program execution.

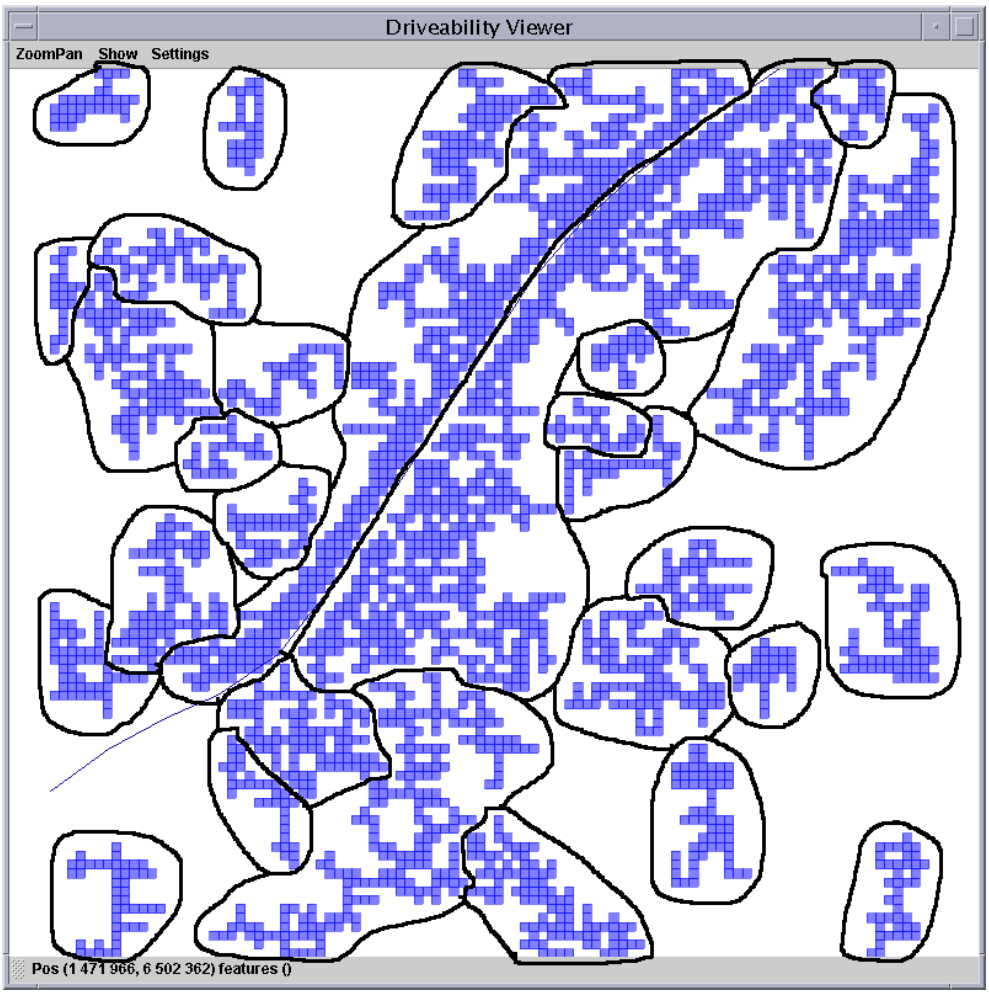

Figure 8.1: Result of the ditch filter using the simple connect algorithm.

8.2 The postprocessing algorithm

A comparison between using only the simple connect algorithm, only the connect edge algorithm, and the connect algorithm followed by the postprocessing algorithm is given in figures 8.1, 8.2, and 8.3, where the identified features have been marked. All algorithms were applied to the ditch filter, using a minimum norm of 0.5 and a minimum feature length of 20 m. The postprocessing algorithm removed all tiles crossing the road, and all features containing less than 4 tiles. The removed tiles are shown in black.

8.3 Terrain features

Each terrain feature layer is colour coded individually, so that e.g. ditches in the cavity layer gets one colour and slopes in the slope layer another colour, see figure 8.4.

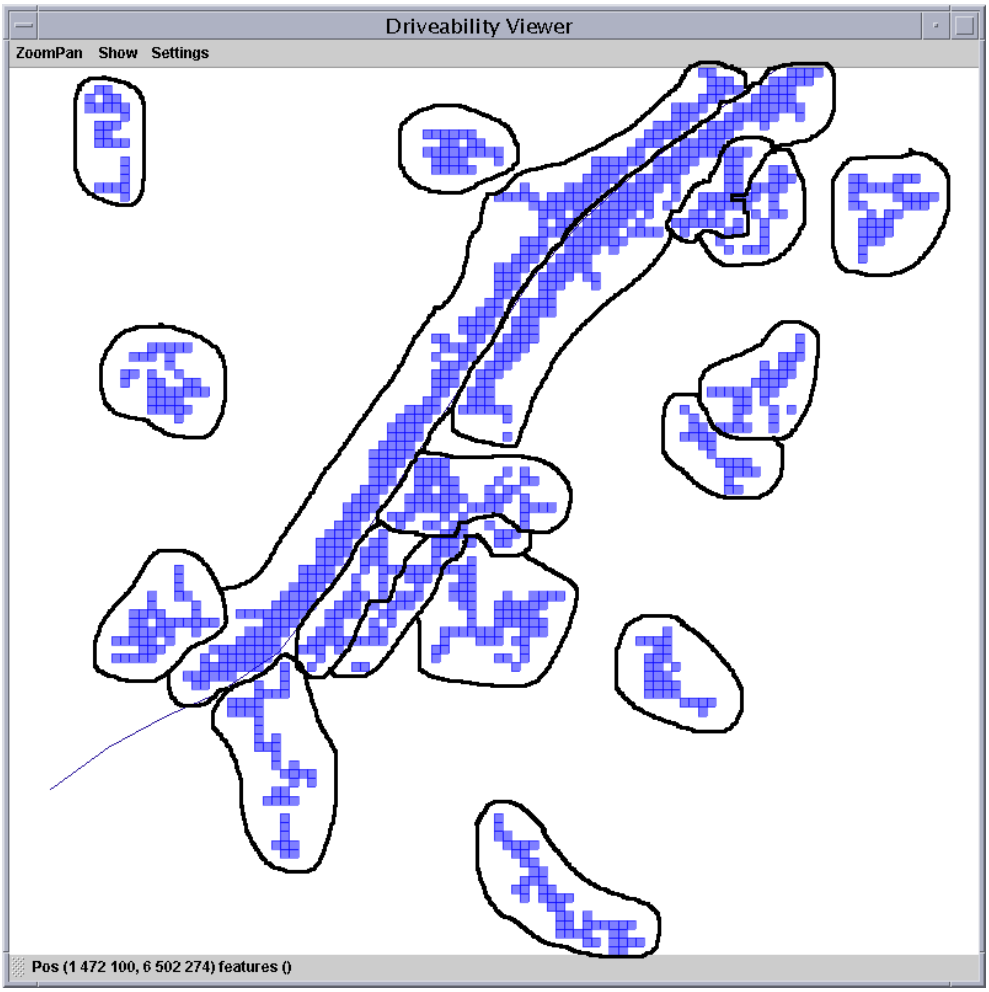

Figure 8.2: Result of the ditch filter using the connect edge algorithm.

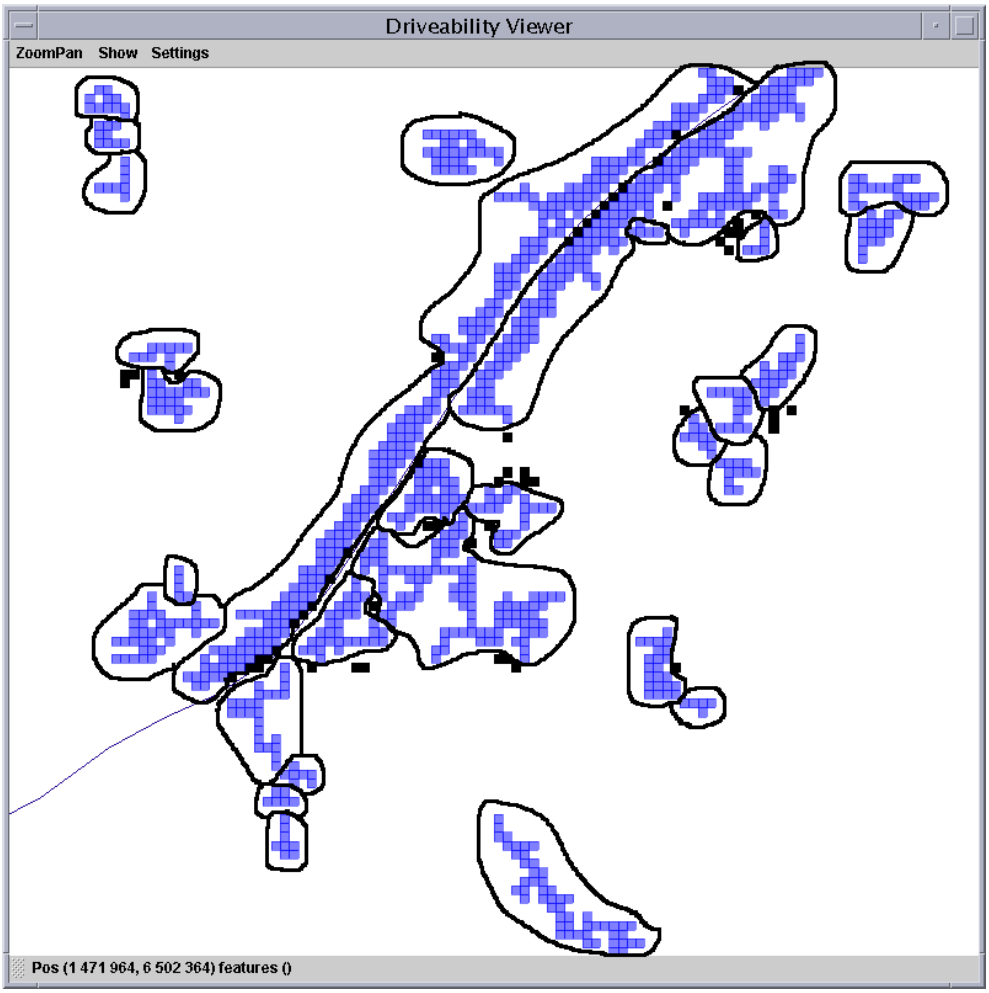

Figure 8.3: Result of the ditch filter using the connect edge algorithm and the postprocessing algorithm.

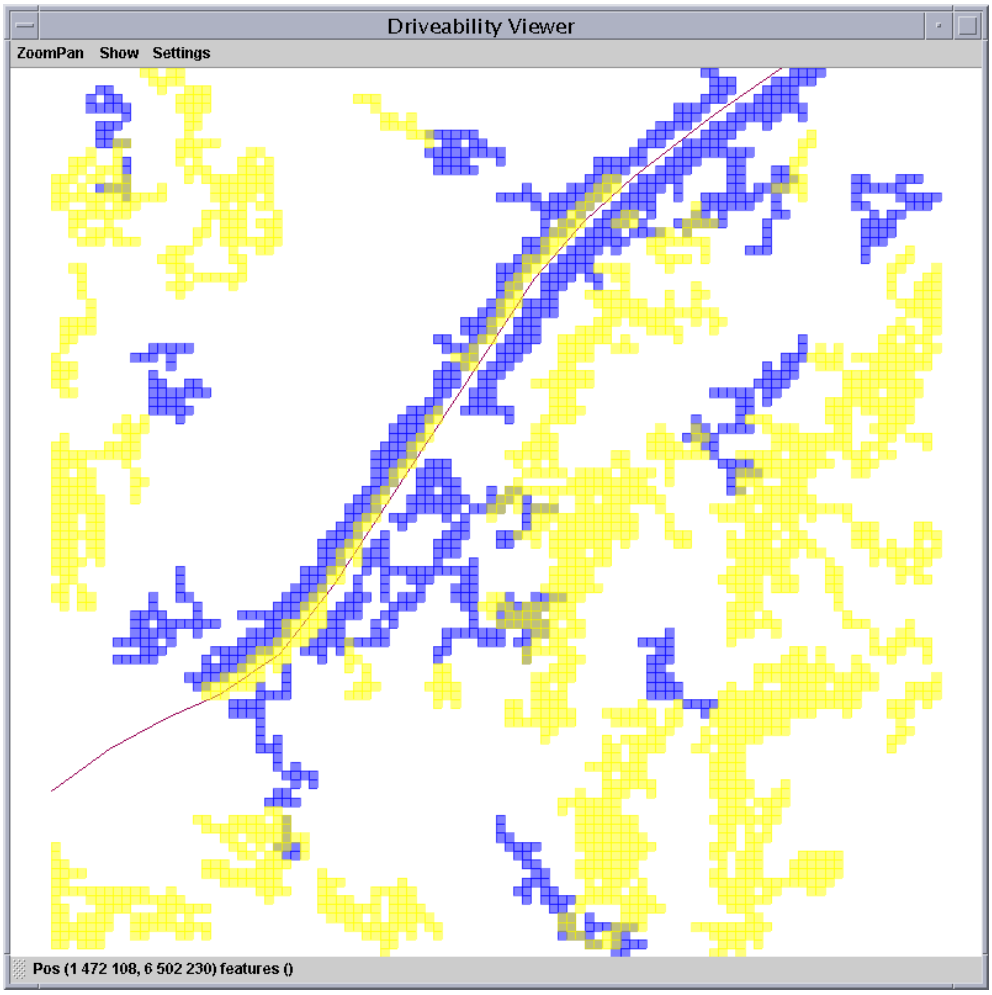

Figure 8.4: Colour coding of feature layers. The blue (dark) features are ditches and the yellow (light) ones are slopes.

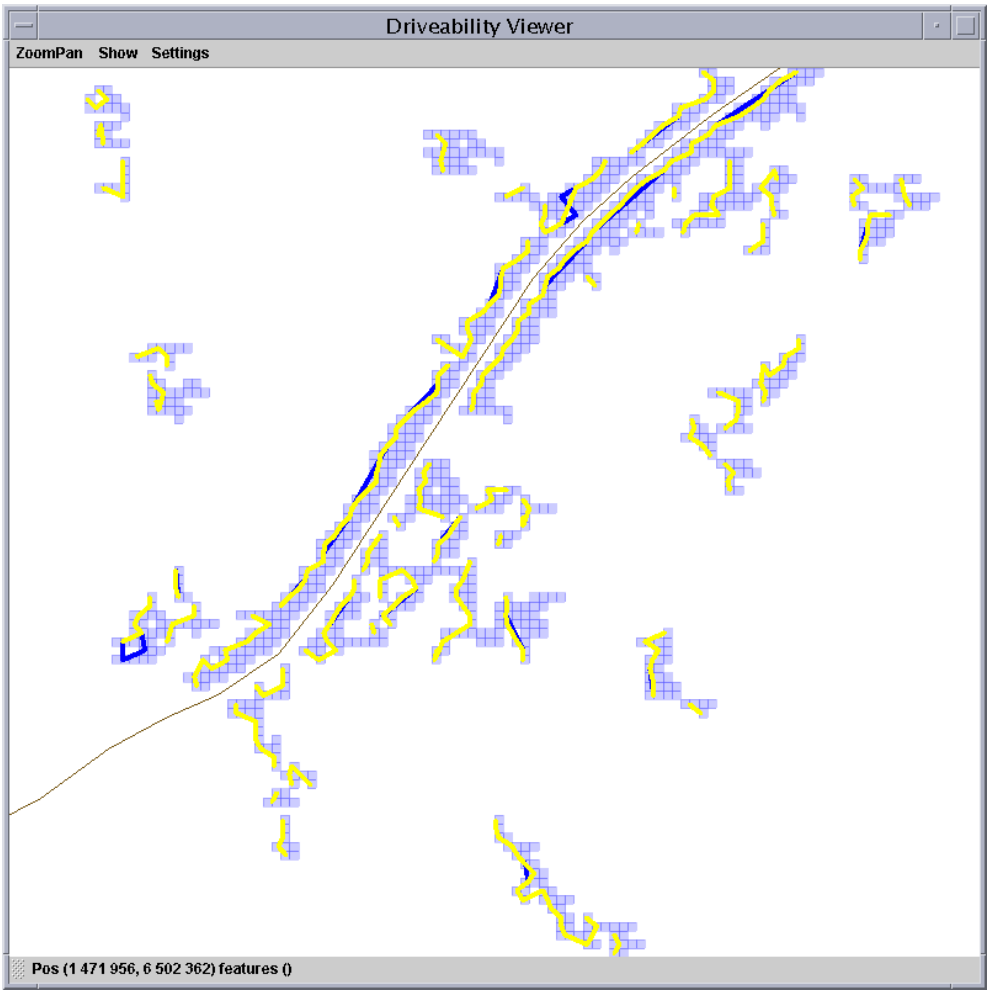

Figure 8.5: Comparison of the two methods developed for steps 3 and 4 of the centre line algorithm: angle deviation (blue/dark) and distance deviation (yellow/light).

8.4 Centre line, boundary, and width

The two methods developed for steps 3 and 4 of the centre line algorithm in setion 7.7.7 are compared in figure 8.5. The yellow (light) lines use the distance deviation method with maximum deviation 5 m and minimum deviation 3 m. The blue (dark) lines use the angle deviation method with maximum deviation 10° and minimum deviation 5°. All points with more than one penalty have been removed. No lower limit was set for the line lengths. Both methods give acceptable results if the features are "nice", but there are room for improvements, especially concerning the treatment of start points and end points.

Using the angle deviation method for the centre line, the boundary points are shown in figure 8.6, thus giving an idea of quality of the width estimation in section 7.7.5.

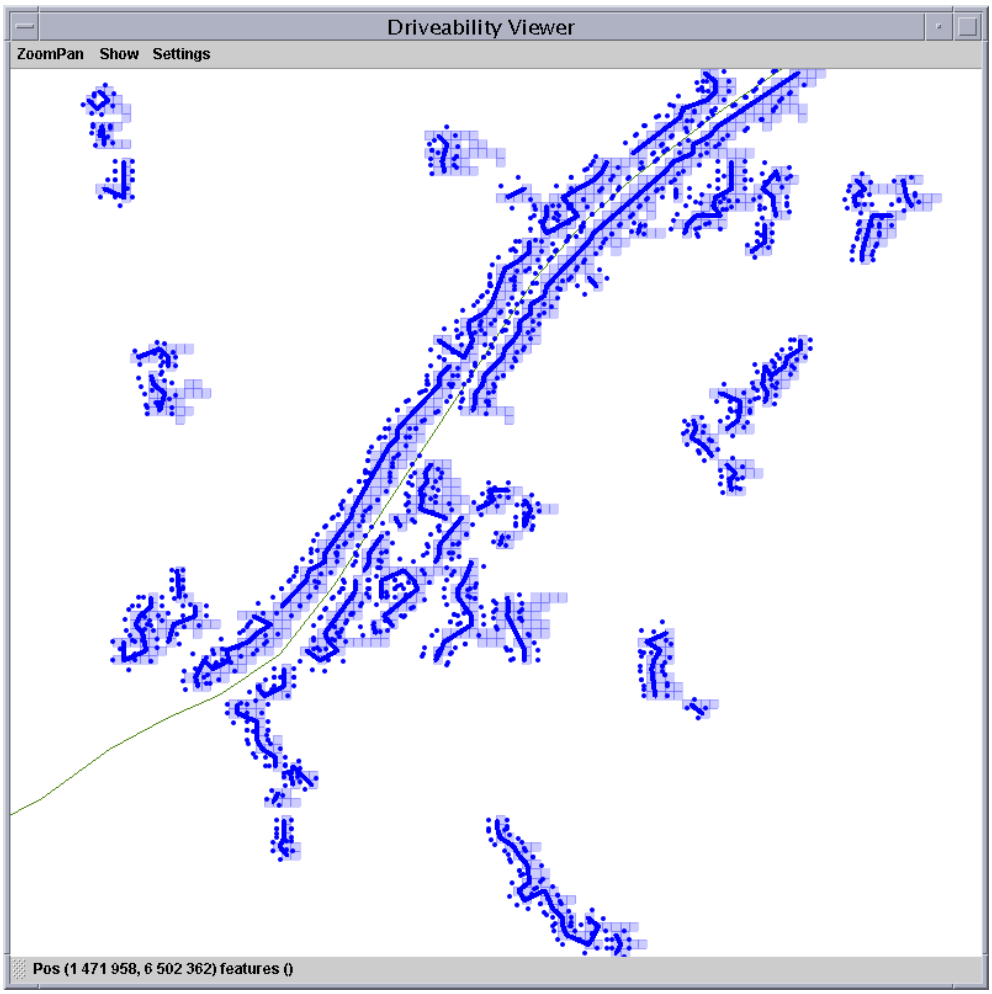

Figure 8.6: Calculated centre line and boundary points.

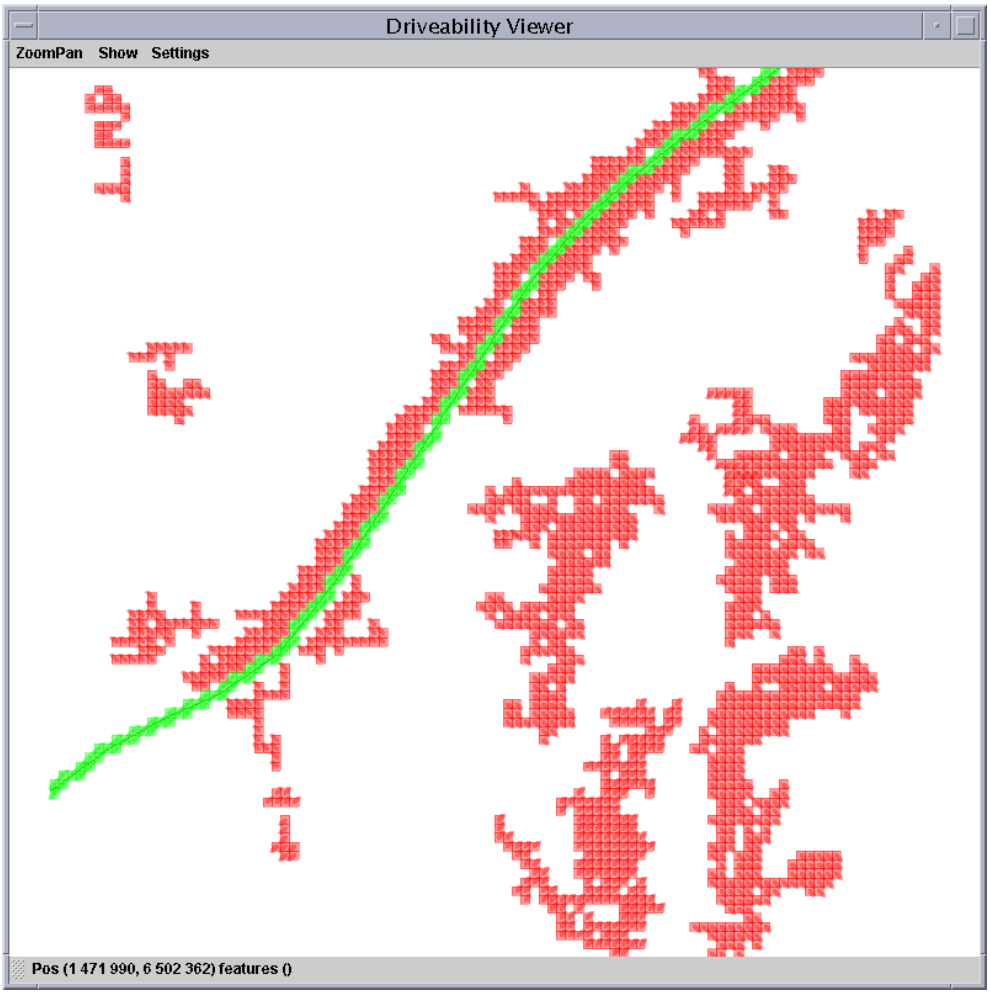

Figure 8.7: The driveability of T72 (all driveability paths).

8.5 Driveability

The basic approach of visualizating the driveability is to use colour coding. Only tiles with a total cost higher than some user-defined limit are shown. Impediments are coloured red, improvers are coloured green, and threats are coloured yellow (in black and write printing these colours correspond to dark gray, medium gray, and light gray). Cost directions are shown in the same way as inclinations. Regions where there is no cost information have no colour labels.

Using the cost function described in section 7.9 and the vehicle properties described in appendix A the driveability results shown in figures 8.7 and 8.8 are achieved. As there is more information about the capabilities of the T72, more driveability information can also be extracted for this type of vehicle. Note, however, that the white regions mean "not enough information", not "driveable". Also, the green regions mean "*some* improver factor exists", not "no impediment factors exist".

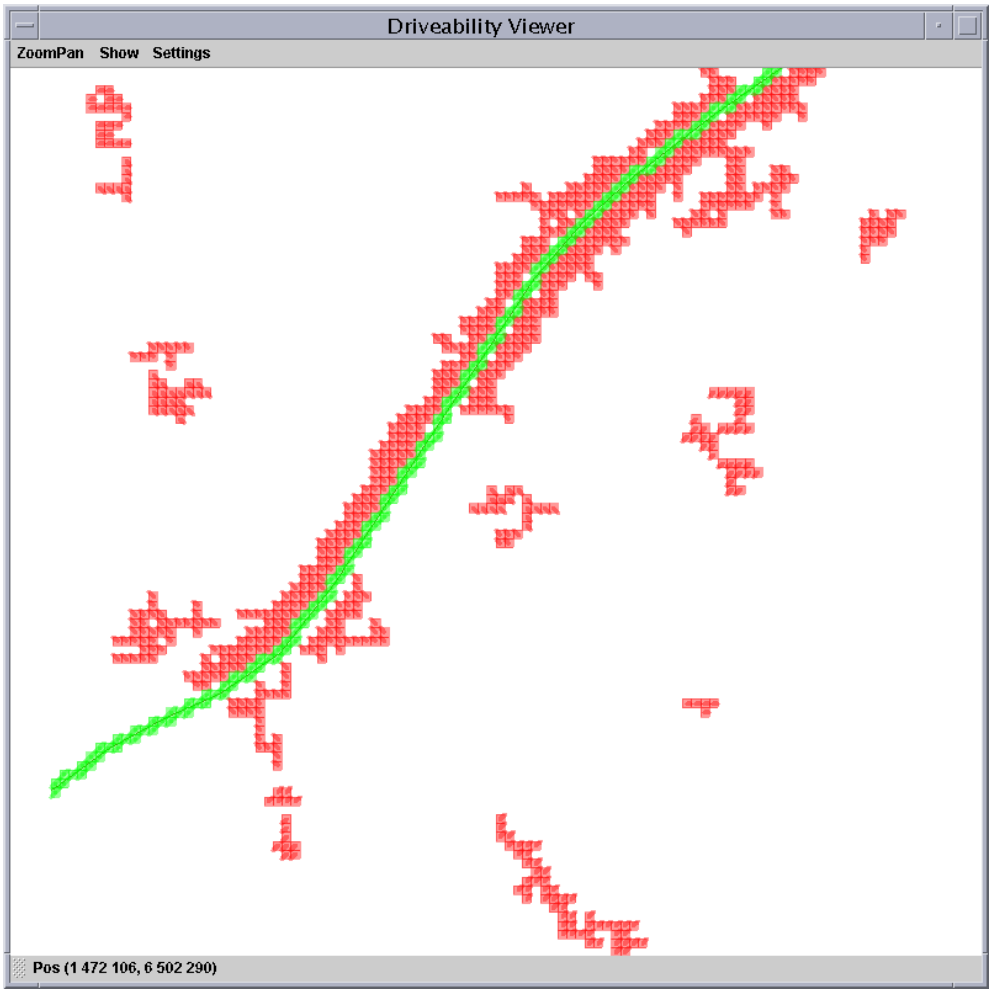

Figure 8.8: The driveability of Volvo Valp (all driveability paths).

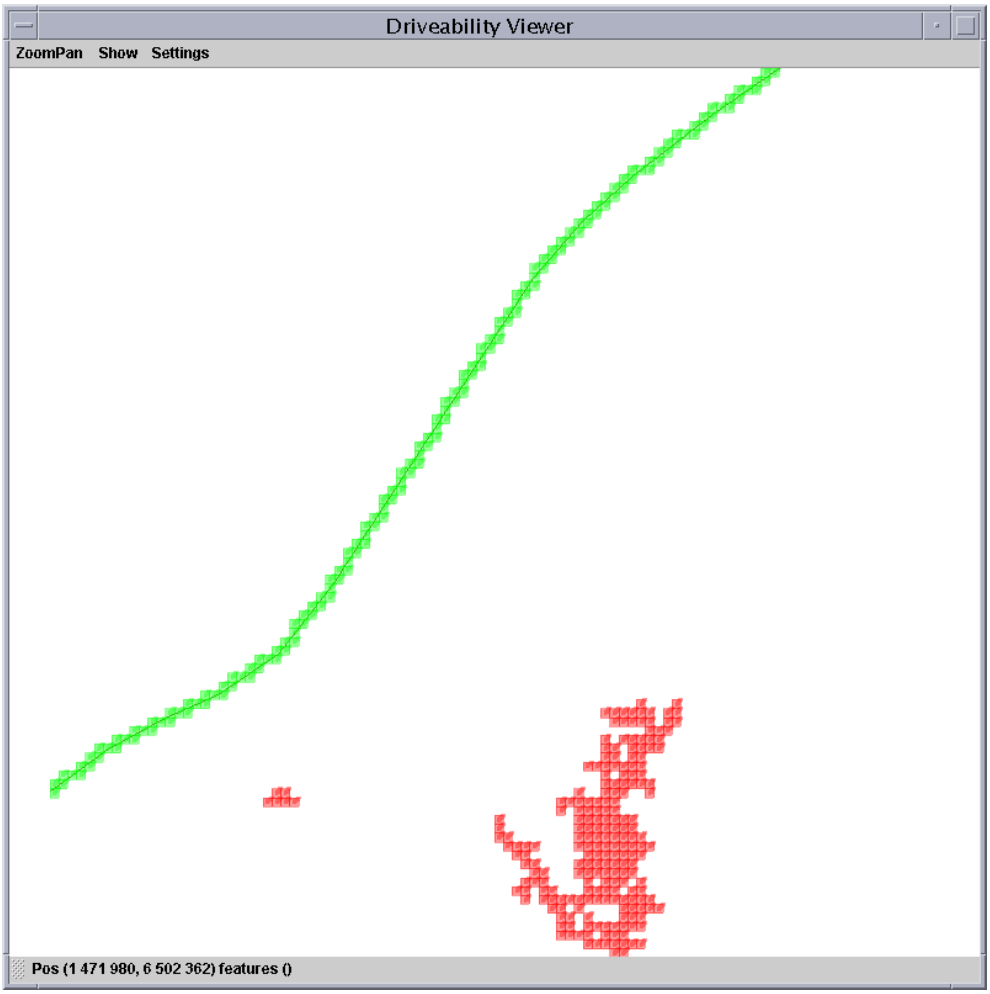

Figure 8.9: The driveability of T72 (driveability path 2/north-east).

It is also possible to show driveability in certain directions, i.e. the driveability for certain driveability paths. Figures 8.9 and 8.10 shows the driveability of T72 in driveability path 2 and 4, respectively.

8.6 Performance

It is hard to do any exact measurements of the performance of Driveability Viewer since the number of tiles and the feature types vary in the area of interest. Also, the computer caches data during the test, affecting the result. However, figure 8.11 shows that the driveability calculation is a major bottle neck of the program. As can be seen in figure 8.12, the time consumption for the postprocessing algorithm (i.e. the total time for Compare ditches and roads and Connect) is considerable compared to the calculation time of Category Viewer, especially for larger areas.

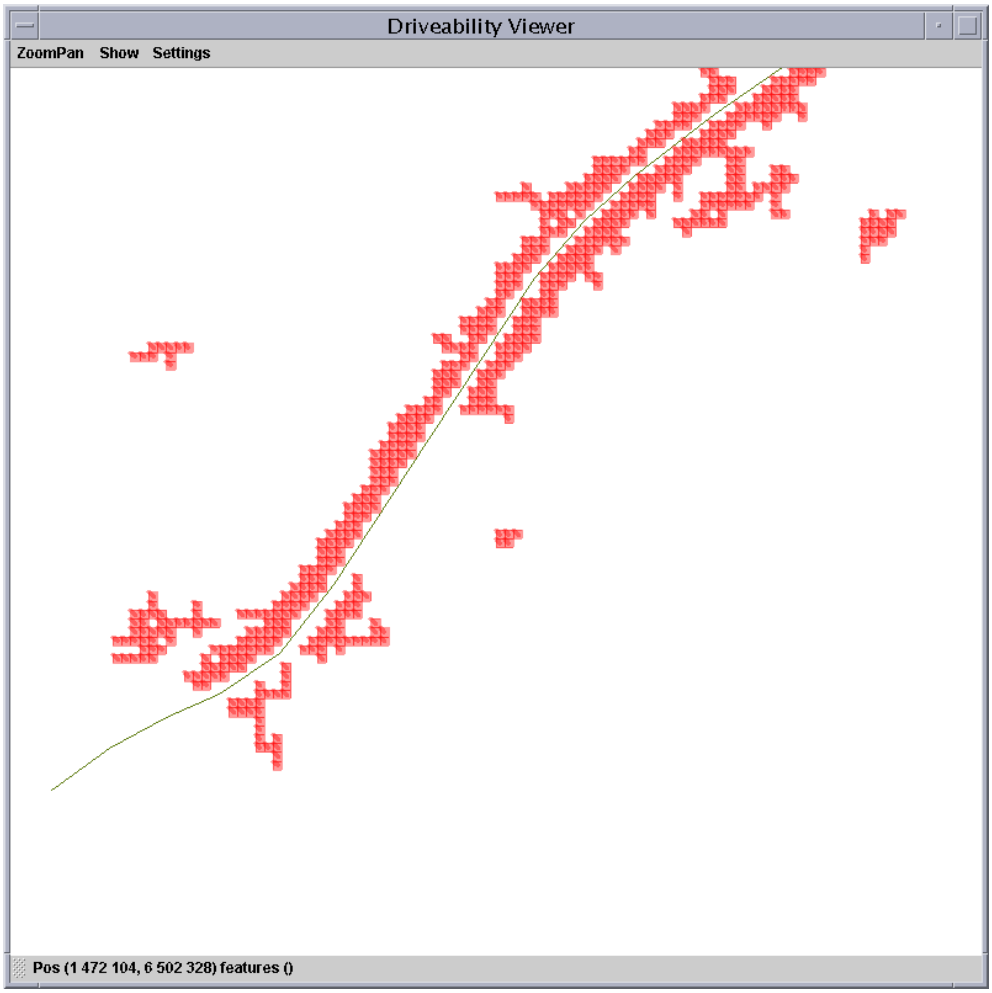

Figure 8.10: The driveability of T72 (driveability path 4/north-west).

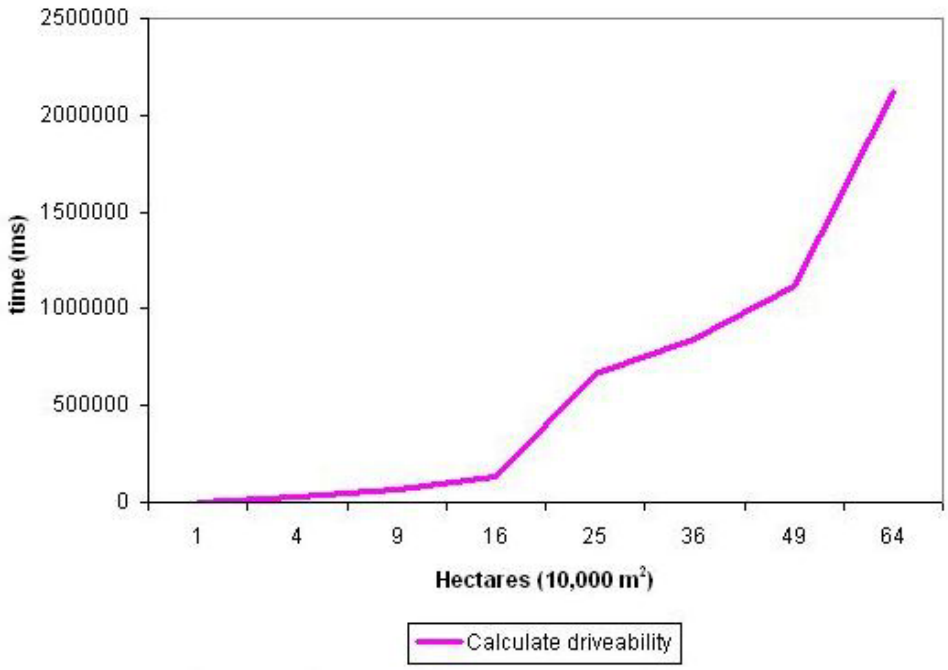

Figure 8.11: Time consumption for the driveability calculation, i.e. application of every rule in the knowledge base to every tile in the area of interest, and thereby also calculation of properties.

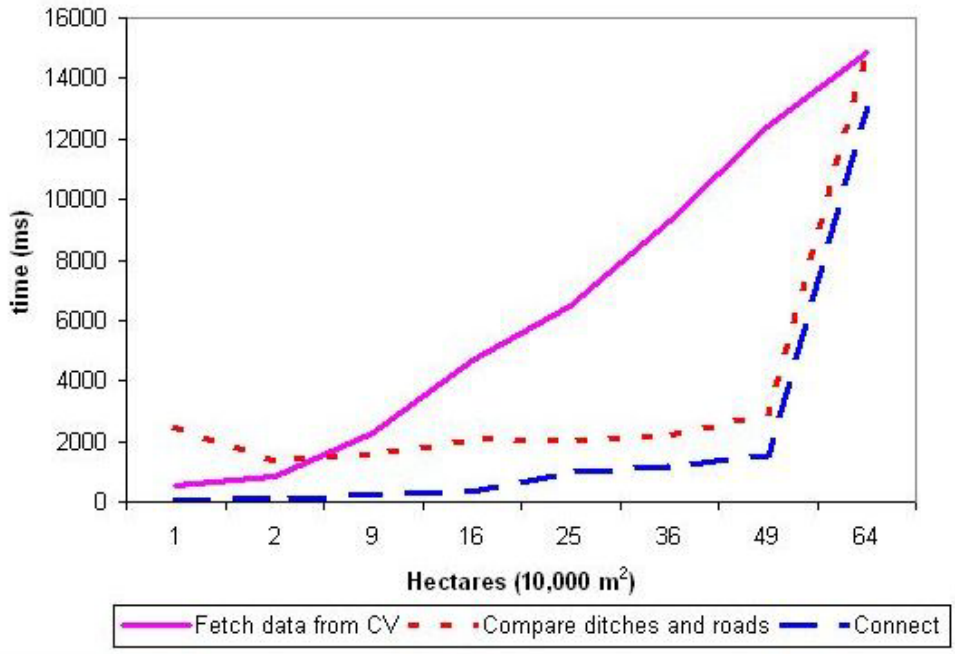

Figure 8.12: Time consumption for the major parts of the data pre- and postprocessing.

8.7 Documentation

Javadoc documentation is available for both Category Viewer and Driveability Viewer.

9. Discussion and future research

This chapter discusses the results in chapter 8, how they were achieved, and suggests some alternative solutions and possible improvements.

9.1 Input data

The more terrain feature properties that are available beforehand, the more Driveability Viewer would be able to focus on driveability issues, instead of on property calculations. There are three obvious information sources that could be utilized further: laser radar data, output from Category Viewer, and map data.

In the future it may be possible to use the intensity from the laser radar data to estimate surface rigidity, but as has been stated in [24, 27], this is not a simple task, because it cannot be guaranteed that e.g. different surface types with certain driveability properties have the same intensity. It may also be possible to calculate forest density from labelled laser radar data.

If more preprocessing of data were made in Category Viewer the filters could be utilized more efficiently. The filtering process should also be extended to store information about the exact segment start, middle, and end, facilitating the connect algorithm and e.g. the centre line calculation.

A possible future extension to the filter definitions would be to add an `angle of slope` keyword instead of a `norm` key word to the filter definition to improve its intuity. However, in order to stick to the original setup of Category Viewer this has not been implemented. Also, it is not always obvious which angle to use.

The correlation between vehicle properties and landforms should be further investigated to make better use of the filter definitions.

Information about the map data classification should be obtained to find out e.g. the width of a road of class I. Contacts with the Swedish Land Survey have been made for that purpose.

A possible way to get around the problem of hierarchical feature types in map data (section 4.3) is to rasterize the features. Raster data cannot be stored as efficiently with respect to space as vector data, but offer more efficient search methods. Set operations can efficiently be carried out such as in the query language Graqula [25]. Their storage requirements can be made more compact by using run-length encoding or other storage

structures [34, 14]. Raster data can be represented in the same resolution as laser radar data, but as the same resolution may not always be used rasterizing must be done at program start up. For further discussions on raster versus vector data, see e.g. [14].

9.2 The postprocessing algorithm

The task of Category Viewer is to detect features; it is not meant to be the last step before the driveability calculation. Instead, one or more steps are needed to describe and label the features. Ideally, these steps should reside between Category Viewer and Driveability Viewer, but presently they are a part of Driveability Viewer.

As the connect edge algorithm produces a less noisy intermediate result than the simple connect algorithm does (see figures 8.1 and 8.2), the postprocessing algorithm is mainly used to correct the flaws mentioned in section 7.3.

The postprocessing algorithm does not consider inclinations and orientations, which is not an ideal situation. Also, as it groups tiles differently into features, the filter settings of minimum and maximum feature width and length become less useful. New estimations must be done, decreasing performance.

The best solution would of course be to improve the connect edge algorithm. This attempt has previously been tried, but with limited results. A successful improvement would require more time and probably also an overhaul of the whole Category Viewer program.

To decide which side of a road a tile belongs to the λ-space concept [26] could have been used. However, as the roads in the map are only represented by their middle lines, only a limited set of tiles are removed, affecting the filter result as little as possible. Hence, the Clementini approach is acceptable.

9.3 Feature properties

Considerable simplifications and assumptions have been made to estimate feature properties and their impact on the driveability. Many of the estimations could be improved by using a divide-and-conquer approach, dividing the feature of interest into smaller sub features with homogeneous properties. However, a limit must be set somewhere; calculation of e.g. feature width cannot be performed on a per tile basis, and even if it could, performance would be dramatically decreased. Therefore, the estimations in this work are a compromise between accuracy and complexity.

9.3.1 Width of DTM features The width estimation is a great simplification. In reality, features may grow wider, or fork, which makes it

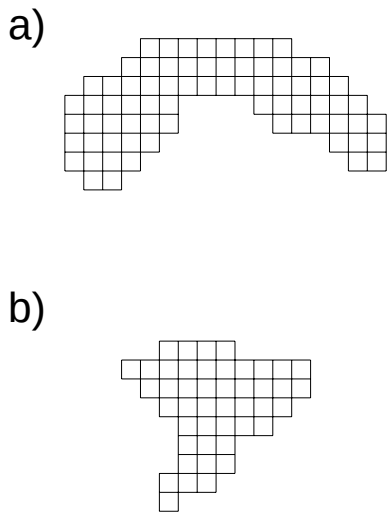

Figure 9.1: Examples of feature types which cause unsatisfactory width estimations. a) Feature that bends. Width estimation gives 3.5 tiles. b) Feature width a large width variation. Width estimation gives 3.2 tiles.

close to impossible to define their width. The same is true for irregular shapes. Some alternative width estimation methods have been proposed:

- Following inclinations and orientations to determine the feature shape. This approach was used in the connect edge algorithm in Category Viewer, and could give satisfactory results if the algorithm is improved.

- Using the bounding box of the feature to determine the width/height ratio and hence get an idea of its shape. As a feature may be curved this approach would give a very rough estimation.

- Finding the centre line of the feature and then applying the method described in section 7.7.5 to the segments which intersect the line.

However, the used method is simple and gives a satisfactory result at this stage of work. For simple, regular features this method leads to a systematic overestimation of the width, since the tiles constituting a feature cover a larger area than the feature itself. It is, however, not easy to predict the result in more complex cases. Figure 9.1 shows some features types which lead to unsatisfactory results.

To improve the estimations the proposed methods could be used recursively on different parts of the feature, but this has not yet been tried. The improvement would be greatest for features with a varying width, or for forked features.

9.3.2 Direction of map features The result of the direction estimation for e.g. a map road is derived from all available road information. This implies unsatisfactory results if the road is winding or if it forks. It would be possible to base the calculations on line segments within a smaller area (e.g. a tile) that covers the region of interest, but to use analogous methods as for other features this possibility has not been used.

9.3.3 Centre line It is possible that step 1 in the centre line algorithm can be improved by e.g. taking the deepest coordinate of a ditch as the centre one. Another suggestion is to find the centre tile of one horizontal segment at a time, calculate the centre tiles of all overlapping vertical segments and comparing these with the horizontal centre tile using a set of rules to get a better estimation of the centre, since horizontal and vertical segments are not laid out in a regular fashion. The centre of the horizontal centre tile (DTM point 0) is used as the default centre of the segment, but if there are one or more vertical centre tiles immediately adjacent to it, they are used to shift the centre to some other DTM point of the horizontal centre tile. Some examples of these rules are given in figure 9.2.

Step 2 offers several possibilities:

- Use the method recursively until no further points can be removed. If too many points are removed the line is, however, not representative for the feature.
- Start a new line using the removed points.
- Connect the line segments closest to each other using MapObjects' `squareDistanceTo` methods. This would have to be used with care to avoid unwanted line extents.
- Create new features around each centre line. It would, however, be complicated to distribute the tiles among the new features in a good way.

9.4 User interface

As the user interface has not been the main objective in this work it is easy to identify improvements:

- Add explanations to the used symbols and colours.
- Start the program without data, and let the user point out the area of interest and load as much of the data as is available.
- Enable multiple windows or tabs showing different combinations of areas and vehicle types.

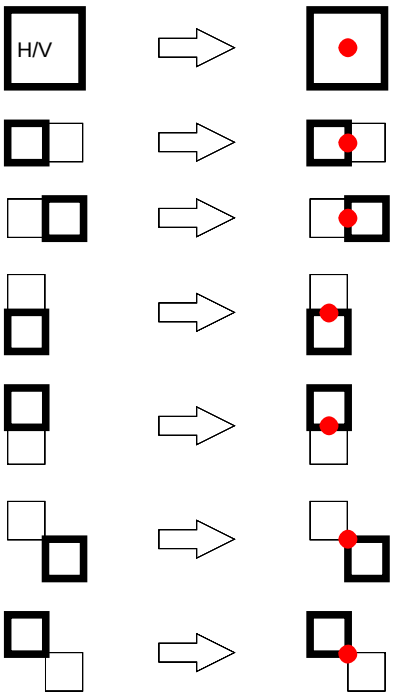

Figure 9.2: Examples of rules of an alternative method for calculation of the points on the centre line. Bold-lined boxes represents horizontal centre tiles, and regular boxes represents vertical centre tiles. The first box represents the case when the horizontal and the vertical centre tile coincide. The tiles to the right show where the centre is assumed to be.

- Show higher costs using a darker colour shade.
- Implement a graphical interface to select feature layers, set vehicle properties, set reliabilities, construct knowledge base rules, etc.

It is not obvious what type of driveabililty to use as "default": driveable or undriveable areas. However, due to the large uncertainty in the calculations, undriveable areas are preferable since you definitely know where you cannot drive, whereas unmarked areas means "not enough information to decide the driveability" or possibly "no information available at all".

More research is needed on which colours and symbols to use. E.g., red and green are not a good combination if there are colour-blind users. Since semitransparent visualization layers are used, how do the colours and symbols interact? Perception of symbols is discussed in [23].

9.5 Data availability and reliability

When calculating driveability costs property availability is checked. If a property is not available, other approaches should be taken. The chosen approach should also affect the cost reliability. Property reliability and cost reliability have been prepared for in the design of Driveability Viewer but have not been considered in the calculations, nor in the user interface.

To improve data availability and reliability it would be good to let the user select individual features in the map and set properties which are known to the user. For example, if the user knows by experience that a forest is too dense to drive through, this information could be used. This would also be preferable for experimental purposes, as would a possibility to highlight features with a certain property, e.g. type, id, width, etc, and a possibility to experiment with vehicle properties via the user interface.

A possible way of handling uncertainties is to use rough sets [36].

9.6 Driveability measures

A single cost value instead of the three-parted cost might be easier for the user to grasp, but would require more pedagogical skills to motivate. Weighing together the three parts and their reliabilities is a very complex task. The three-part cost gives the user more control over the final decision of where to drive. Different users have different needs, and a cautious user may value tiles with a high improver part in the cost higher than a more careless user.

The knowledge base rules are very simplified and do not cover all aspects of driveability. As an example, the vehicle turning radius, width, and length should be taken into account when calculating forest driveability, as well as the height to the lowest branches and the ground cover.

Another example is that surface rigiditiy should be compared with vehicle capabilities, e.g. slope capability or engine power. More research is needed to determine how different impact factors are related and how they affect each other. It is, however, easy to add new rules to the knowledge base.

Weather information and the time aspect have not been considered.

A cost estimation for the whole region R can be achieved by

$$\vec{\Gamma}_{estimation} = \min_{1 \leq i \leq 8} \Gamma_i$$

to get the most driveable path, or

$$\vec{\Gamma}_{estimation} = \max_{1 \leq i \leq 8} \Gamma_i$$

to be sure of not driving along the worst path, where Γ_i is the impact along driveability path i.

A more flexible notion of driveability path should be developed, so that different feature types can be associated with different sets of driveability paths.

9.7 Performance

As Driveability Viewer is only a first step towards a planning tool real-time performance is not required. However, to be part of the ΣQL system its performance must be improved [13]. As could be seen in figure 8.11 the main part to improve is the driveability calculation. Property calculation should not be a part of the driveability calculation. It is also possible that using larger regions than single tiles would improve performance.

Improving input data (see section 9.1) could be one way of speeding up computations, as well as only considering changes in the used data sets if the user zooms, pans, or changes the number of data areas shown. Visualization could be improved in a similar way, as well as hiding details which do not add information if the zoom level is too high.

9.8 Conclusions

This work has reached two main results: an evaluation of what driveability is and what it is affected by, and a general cost function, which can be used even if not all information is available. However, driveability is a very complex matter, and the available time is certainly not enough for much more than a discussion of the difficulties. Consequently, very few solutions are given. Some of the remaining problems include the weather factor and how to handle data availability and reliability. The user interface also needs to be developed further. The Driveability Viewer program does, however, offer a good framework for further research in the area.

Future work should also look into which terrain feature properties that are most important to a vehicle with certain capabilities. Further

vehicle types needs to be added, and maybe also the ability to calculate the driveability of different military units. Other aspects of driveability should also be considered, such as which terrain features are offering a good shelter from the enemy, or where the enemy is most likely to have placed traps. Presently, the estimations of terrain feature properties are too coarse to present driveability differences for resembling vehicles.

However, good algorithms are of no use if input data is mediocre. Therefore, the best starting point for future research would be to improve the connect edge algorithm. It would also be advantageous to extend the usefulness of the filtering process by calculating and storing more attributes of the found features so that the Driveability Viewer can concentrate more on the driveability issues.

Bibliography

[1] ESRI. `http://www.esri.com`.

[2] Allmän beskrivning: GSD - grunddata 10, November 2000. In Swedish.

[3] S. Ahlberg, C. Carlsson, and Pontus Hörling. Måligenkänning och terrängmodellering med laserradardata. Technical Report FOA-R–99-01303-408-SE, Department of Data and Information Fusion, FOA, 1999. In Swedish.

[4] R. P. Bonasso. Toward a naive theory of trafficability. In *Proceedings of the Annual AI Systems in Government Conference*. IEEE, March 1989.

[5] R. Bonds, editor. *Moderna pansarfordon*. Frank Stenvalls Förlag, 1980. In Swedish.

[6] G. S. Broten and B. L. Digney. Perception for learned trafficability models. In *Proceedings of the SPIE – The International Society for Optical Engineering*, volume 4715, pages 149–160. SPIE, 2002.

[7] P. Chaturvedi, E. Sung, A. A. Malcolm, and J. Ibañez Guzmán. Real-time identification of driveable areas in a semi-structured terrain for an autonomouts ground vehicle. In *Proceedings of the SPIE – The International Society for Optical Engineering*, volume 4364, pages 302–312. SPIE, 2001.

[8] E. Clementini and P. Di Felice. A comparison of methods for representing topological relationships. *Information Sciences*, 1995. Also available at `http://dsiaq.ing.univaq.it/hpage/eliseo/comparison.html`.

[9] B. L. Digney. Learned trafficability models. In *Proceedings of the SPIE – The International Society for Optical Engineering*, volume 4364, pages 51–60. SPIE, 2001.

[10] J. J. Donlon and K. D. Forbus. Using a geographic information system for qualitative spatial reasoning about trafficability. In *Proceedings of QR99*, volume 4364, Loch Awe, Scotland, June 1999.

[11] P. G. Ducksbury. Driveable region segmentation using a Pearl Bayes network. In *IEE Colloquium on 'Image Processing for Transport Applications' (Digest No. 1993/236)*. IEE, 1993.

[12] E. Jungert and C. Grönwall, editors. ΣQL. Ett beslutsstöd för måligenkänning i en multisensormiljö. Technical Report FOI-R-0692-SE, Department of Data and Information Fusion, FOI, 2002.

[13] E. Jungert and C. Grönwall, editors. From sensors to decision. Towards improved situation awareness in a network centric defence. Technical Report FOI-R-1041-SE, Department of Data and Information Fusion, FOI, 2003.

[14] L. Eklundh, editor. *Geografisk informationsbehandling. Metoder och tillämpningar.* Byggforskningsrådet, 1999. In Swedish.

[15] ESRI. *MapObjects - Java. Developer's Guide.*

[16] ESRI. Mapobjects javadoc.

[17] ESRI. MapObjects User Forum. `http://forums.esri.com/forums.asp?c=9`.

[18] ESRI. ESRI Shapefile Technical Description. An ESRI White Paper. `http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`, July 1998.

[19] R. Forsgren. Markspaning med flygande farkoster - omvärldsmodellen. Unpublished work at FOI, November 2003. Version 1.4. In Swedish.

[20] A. Grunes and J. F. Sherlock. Texture segmentation for defining driveable regions. In *BMVC90 Proceedings fo the British Machine Vision Conference*, pages 235–239, Oxford, UK, September 1990.

[21] T. Horney. Design of an ontological knowledge structure for a query language for multiple data sources. Master's thesis, Linkoping University, 2002.

[22] P. Jasiobedzki. Detecting driveable floor regions. In *Proceedings. 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 264–270, August 1995.

[23] A. Johansson. Symbolers synlighet mot ett informationsbärande gränssnitt. Technical Report FOA-R–00-01719-505–SE, Division of Command and Control Warfare Technology, FOA, 2000.

[24] A. J. Johnson, E. Windesheim, and J. Brockhaus. Hyperspectral imagery for trafficability analysis. In *1998 IEEE Aerospace Conference Proceedings*, volume 2, pages 21–35, Snowmass at Aspen, CO, USA, March 1998.

[25] E. Jungert. Graqula - a visual information-flow query language for a geographical information system. *Journal of Visual Languages and Computing*, 4(4):383–401, December 1993.

[26] E. Jungert. λ-space - a mapping of the observation space for spatial reasoning. In *Proceedings of The Ninth International Conference on Distributed Multimedia Systems (DMS'2003)*, Miami, Florida, USA, September 2003.

[27] F. A. Kruse, J. W. Boardman, and A. B. Lefkoff. Extraction of compositional information for trafficability mapping for hyperspectral data. In *Algorithms for Multispectral, Hyperspectral, and Ultraspectral Imagery IV*, volume 4049, pages 262–273, April 2000.

[28] Lantmäteriet. `http://www.lantmateriet.se`. In Swedish.

[29] Lantmäteriet. Kartbladsindelning, Rikets bladsystem. `http://www.lm.se/geodesi/kartprojektion/blad/bladsystem.htm`.

[30] Lantmäteriet. Produktbeskrivning: GSD - Höjdkurvor, 5 m ekvidistans (för GSD - Fastighetskartan).

[31] Lantmäteriet. Produktbeskrivning: GSD - Fastighetskartan, ytbildad med fastighetspunkter, April 2002. In Swedish.

[32] F. Lantz and E. Jungert. Dual aspects of a multi-resolution grid-based terrain data model with supplementary irregular data points. In *Proceedings of the 3rd International Conference on Information Fusion (Fusion 2000)*, Paris, France, July 2000.

[33] F. Lantz, E. Jungert, and M. Sjövall. Determination of terrain features in a terrain model from laser radar data. In *Proceedings of the ISPRS Working Group III/3 Workshop '3-D Reconstruction from Airborne Laserscanner and InSAR Data'*, Dresden, Germany, October 2003.

[34] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Academic Press, 1992.

[35] Department of the army. FM 5-33, US Army Field Manual, Terrain Analysis. `http://www.globalsecurity.org/military/library/policy/army/fm/5-33/default.htm`, July 1990.

[36] Z. Pawlak. Rough sets and data analysis. In *Soft Computing in Intelligent Systems and Information Processing, Proceedings of the 1996 Asian Fuzzy Systems Symposium*, December 1996.

[37] Å. Persson. Extration of individual trees using lasar radar data. Technical Report FOI-R–0236-SE, Department of Radar Systems, FOI, 2001.

[38] D. Sapounas, T. Kreitzberg, and M. L. Johnson IV. Terrain traffica-bility model. In *Military, Government, and Aerospace Simulation. Proceedings of the 1996 Simulation Multiconference*, pages 242–248, New Orleans, LA, USA, April 1996.

[39] Stödenhet Geografisk Information (Geo SE). Fastighetskartan över Kvarn. Cd id 1647.

[40] M. Sjövall. Object and feature recognition in a digital terrain model. Technical Report FOI-R-0499-SE, Department of Data and Information Fusion, FOI, 2002.

[41] Sun. Java 2 Platform Std. Ed. v1.4.2 javadoc. `http://java.sun.com/j2se/1.4.2/docs/api/`.

A. Vehicle properties

In this chapter some vehicle properties are given.

A.1 T72

Type:	tank
Wheel type:	six road wheels per track
Length:	6.4 m (hull); 9.02 m (including weapons)
Width:	3.375 m
Height:	2.27 m
Weight:	41,000 kg
Vertical obstacle:	0.92 m
Ditch width:	2.7 m
Ascent:	60%

Source: [5] and http://www.ebroadcast.com.au/ecars/Mil/Tanks/T72.html (20 January 2004)

Figure A.1: T72. The picture is the property of FOI.

A.2 Volvo Valp (Pltg 903)

Type:	cargo/troop carrier
Length:	4.05 m
Width:	1.65 m
Weight:	2,250 kg (total)
Height:	2.10 m
Turning radius:	11.40 m
Road clearance:	0.285 m
Wheel track:	1.34 m

Source: http://come.to/tgb-sidan (20 Januari 2004)

Figure A.2: Volvo Valp. The picture is the property of FOI.

B. Driveability Viewer

A simplified class diagram for Driveability Viewer is shown in figure B.1. Inner classes and some relations have been left out to elucidate the structure.

The `DriveabilityViewer` class is the entry point to the program. It relies on the `Vehicle` class to get information about terrain features (via a `TerrainFeatureOverlay` object) and the driveability costs (via a `DriveabilityLayer` object). `MapFeature` objects, `DTMFeature` objects, and `DriveabilityLayer` objects all keep track of their own geometries, so the `DriveabilityViewer` class only needs to supply the color (via the `Colorizer` class) and perform the drawing.

`MapFeature` objects are fetched from shapefiles using a `MapDataFetcher` object, and `DTMFeature` objects are fetched from DTM data using a `DTMDataFetcher` object, which communicates with Category Viewer. `DTMFeature` objects consist of `DTMTile` objects in a `TileCollection` structure. The `DTMTile` class can be seen as an extended wrapper class for the `Category` class in Category Viewer (see appedix C).

`DTMFeature` objects with the same properties (e.g. ditches from the same ditch filter) are stored in a `TerrainFeatureCollection` object. One or more `TerrainFeatureCollection` objects constitute the `TerrainFeatureOverlay` object.

The driveability cost for a location is calculated by comparing values from `TerrainFeatureInfo` objects, `VehicleInfo` objects, and `WeatherInfo` objects at that location. The calculated costs are stored in `DriveabilityTile` objects, which build up the `DriveabilityLayer` object.

The centre line of a `DTMFeature` object is represented by a `LineCollection` object, which is built up by one or more `Line` objects.

The `Properties`, `Util`, `AOI`, and `DTM` classes are helper classes which contain settings, helper methods, information about the area of interest, and information about the digital terrain model, respectively.

B.1 Data availability and reliability

Properties are stored in instances of the `VehicleInfo` class and the `TerrainFeatureInfo` class, which are both subclasses of the `Info` class. The `Info` class consists of a key attribute, a property value attribute, and a reliability attribute. Property values are stored as instances of the Java `Object` class, to enable storage of different data types. Key values

are available as integer constants in the `Info` sub classes, which enable the programmer to check if a certain property value is present. If not, a `null` value is returned.

The `Info` classes can be used to set default values.

B.2 The knowledge base

Each `Vehicle` object uses a `KnowledgeBase` object to calculate its driveability. The `KnowledgeBase` object consists of a number of `Rule` objects, each of which can be either `AndRule` or `OrRule` objects. Each `Rule` object also has an `Impact` object and a number of `Condition` objects. A `Condition` object consists of a `ConditionOperator` object and two `ConditionOperand` objects, one of which may be `null` to represent unary conditions. A `ConditionOperand` object may be a `RelativeConditionOperand` object, meaning that its value depends on the source, or an `AbsoluteConditionOperand`, meaning that its value is constant.

B.3 Program start up

Driveability Viewer is started from the command prompt, giving the area name (in this case *Kvarn*) and vehicle type name (in this case *T72* or *Valp*) as arguments. If the arguments are left out default settings are used. Both the area name and size, and the vehicle type can be changed during execution via menus. Three run scripts are supplied:

- *run.sh* starts the program with default settings (from the `Properties` class).
- *runt.sh* starts the program with T72.
- *runv.sh* starts the program with Volvo Valp.

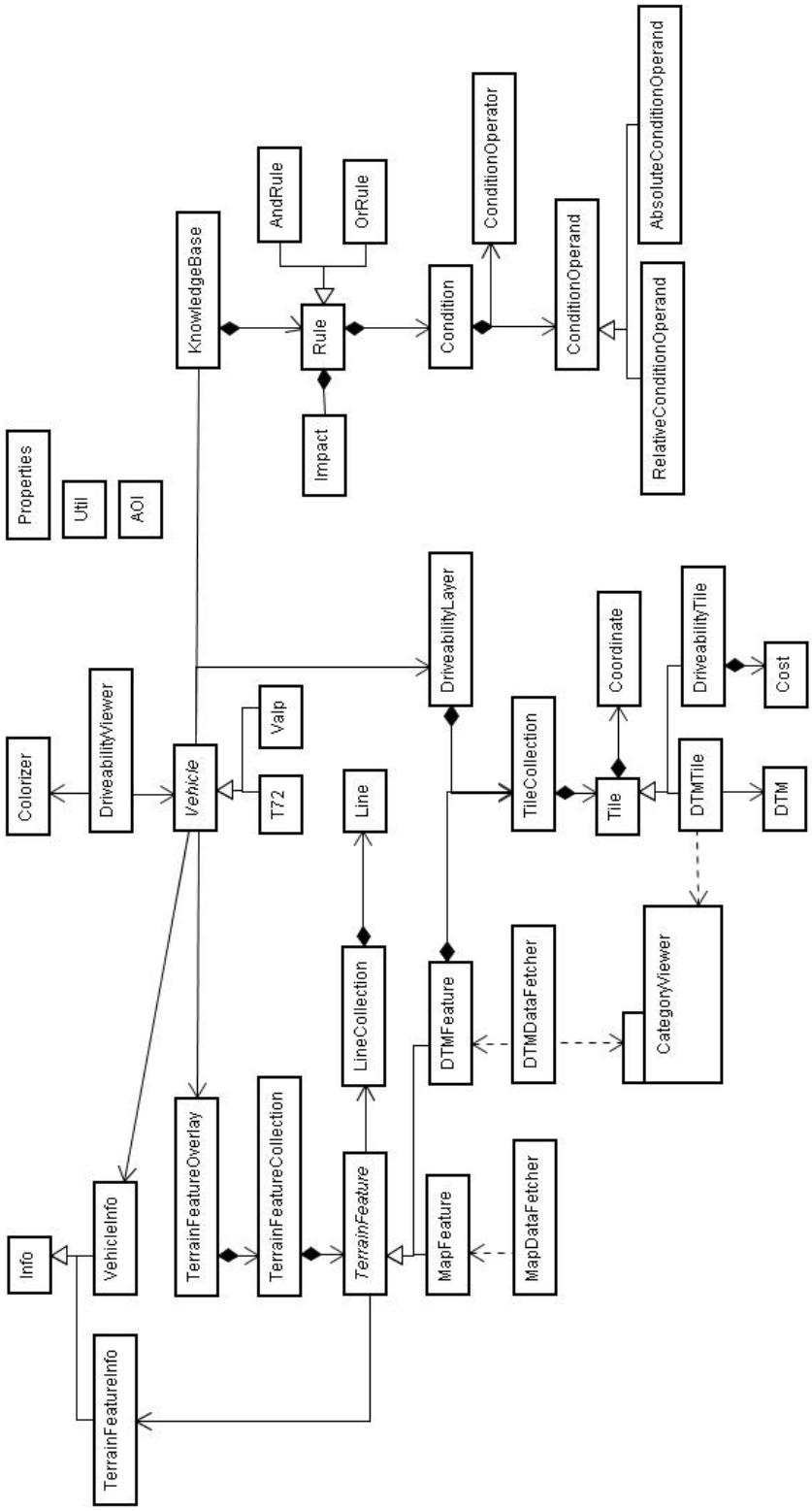

Figure B.1: Simplified class diagram for Driveability Viewer.

C. Category Viewer

Below follows a description of the classes in Category Viewer:

- `Category` represents a single tile.
- `CategoryMain` is the entry point to the program. It creates the user interface and initializes all data.
- `CategoryPanel` visualizes the result of the filtering process.
- `CategoryData` constructs `Category` objects from DTM data files.
- `CategoryFilter` applies a filter to the tiles in a `CategoryData` object.
- `FilterCategory` contains information about what tiles to search for during the filtering process.
- `FilterSubSegment` contains information about what subsegments to search for during the filtering process.
- `FilterVariables` contains variables which show the result of the filtering process.

To enable communication between Category Viewer and Driveability Viewer the following changes were made:

- A `Constants` class was added to hold some constants used in both Category Viewer and Driveability Viewer. Category Viewer has not been updated to use these constants instead of their values.
- The `Category` class was updated with attributes for feature id, width, and length, as well as accessor method for these attributes. `Get` methods were also added for the category norm and location.
- A `getIterator` method was added to `CategoryData` to enable iteration over its elements.
- `CategoryFilter` was changed to allow minimum norm settings in the filter definitions. An attribute and a corresponding `get` method was added to enable access of filtered data from Driveability Viewer. An id number is now added to each feature.

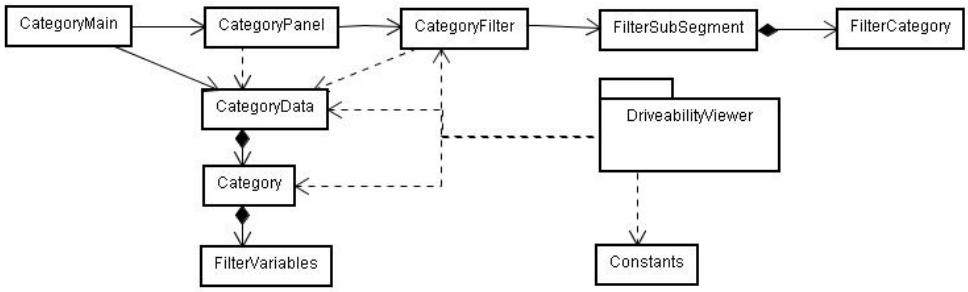

Figure C.1: Simplified class diagram for Category Viewer.

- Variables to keep track of the feature id were added to `FilterVariables`.

A simplified overview of the relations between the classes in Category Viewer is shown in figure C.1.