FFI-rapport 2014/01222

Situation dependent path planning for computer generated forces

Solveig Bruvoll

FFI Forsvarets forskningsinstitutt

Situation dependent path planning for computer generated forces

Solveig Bruvoll

Norwegian Defence Research Establishment (FFI)

2 September 2014

FFI-rapport 2014/01222

1233

P: ISBN 978-82-464-2404-0
E: ISBN 978-82-464-2405-7

Keywords

Terrenganalyse

Grafer

Vekter

Simulering

Autonome systemer

Approved by

Karsten Bråthen	Project manager
Anders Eggen	Director

English summary

We present a concept for situation dependent path planning. This concept can be applied to path planning in several applications, like autonomous vehicles and simulated entities. The idea behind the concept is to identify the main aspects that influence path planning, create a weight function for each aspect, and to combine these functions in a situation dependent weight function, where the influence of each aspect depends on the situation. We demonstrate the concept by describing in detail how it can be used to implement a path planning method for simulated main battle tank units and show examples of planned paths for several situations. This implementation is based on a standard graph search path planning method.

Sammendrag

Vi presenterer et konsept for situasjonsavhengig ruteplanlegging. Dette konseptet kan brukes til ruteplanlegging for flere anvendelser, som autonome systemer og simulerte enheter. Konseptet går ut på å identifisere hvilke aspekter som påvirker ruteplanleggingen, utvikle en vektfunksjon for hvert aspekt og å kombinere disse funksjonene i en situasjonsavhengig vektfunksjon, der hvilken innvirkning hvert aspekt har avhenger av situasjonen. Vi demonstrerer konseptet ved å beskrive i detalj hvordan det kan brukes til å implementere en ruteplanleggingsmetode for simulerte stridsvogner og viser eksempler på ruter planlagt for forskjellige situasjoner. Denne implementasjonen er basert på en standard ruteplanleggingsmetode som søker etter optimal rute gjennom en graf.

Contents

1 Introduction

This report describes a method for situation dependent path planning which is easy to adjust for different situations and tasks. This work is a continuation of the work described in [5] and is intended to extend the functionality of the simulation system described in [1, 6].

In simulations of military operations the simulated entities need to handle different tasks and situations. Many of these tasks involve movement to a new position or along an axis, and the route the entities should choose will depend on the situation and the task the entity is given. These routes should be planned based on terrain analysis, enemy positions and the requirements caused by the type of task the entity is to perform. If an entity is to move quickly through a secured area, hiding is not an issue, and if the task requires the entity to stay out of enemy sight and weapon range, a longer and more time consuming route may be preferred.

Path planning is a well studied field, however, the majority of contributions focus on optimizing the path for a single aspect, usually minimizing distance or time consumption. We want to find a path that is optimal with respect to a combination of multiple aspects. Multi-objective genetic algorithms have been used to address this problem. One example is [2], which aims to minimize path length and total difficulty by first finding short paths before comparing their risks. Another approach is used in [3], which aims to determine the shortest and safest path for an autonomous agent by optimizing two objective functions simultaneously. The two aspects distance and risk are also considered in [4], which presents a similar approach to ours, combining the two aspects in a new edge weight, which is used for graph based path planning. However, they compute the new weight based on entropy, while we want a function that handles several aspects and can be varied intuitively for different situations.

Our aim was to develop a situation dependent path planning method which can take several aspects into account simultaneously and which easily can switch between path planning for different types of situations. We have developed a concept for situation developed path planning which satisifes these needs. This concept is demonstrated by an implementation of a situation dependent path planning method for an MTB. This implementation is integrated in B-HAVE [8].

We start by describing the concept in section 2. In section 3 we describe the path planning procedure in B-HAVE. Section 4 presents the data used in our implementation of situation dependent path planning for an MTB. Thereafter we in section 5 identify five main aspects that can influence path planning for an MTB and describe the calculation of weights for each aspect based on the input data. In section 6 we present our method for combining weights in order to obtain a situation dependent path planning procedure and show results produced by our implementation of the method. Discussion of the results and ideas for further work are found in section 7, and we conclude in section 8.

2 Concept description

We have developed a concept for situation dependent path planning can be used for automatic path planning for both simulated and autonomous entities. The result is a path planning method that can be easily adjusted to different situations by personnell who are not familiar with the implementation and the details of the path planning routine. The idea is to first determine a set of aspects that are important for path planning for a specific type of entities and to determine functions for how each aspect influence path planning. Typical aspects can be related to threat and time consumption. The aspects are combined by a weighted sum, and the resulting function is used for path planning.

The key to this concept is that the coefficients of the weighted sum depend on how the different aspects are prioritized relative to each other. Therefore the adjustment to a specific situation is done by assigning a priority number to each aspect. These priority numbers can be set and modified by the user of the method and are not fixed in the implementation. This concept can therefore be used in a path planning service taking start and end point as input, together with a priority of the aspects and produces a path suitable for the situation.

For unmanned or simulated entities we can predefine sets of priority numbers for different missons or phases of a mission, for example transit to an interesting area, investigation of the area, and return to base. The priority is linked to the type of task an entity is given, and it is possible to predefine a set of priority numbers for each type of military task in a simulation system. This makes the concept suitable for use in an agent system with artificial intelligence using context based reasoning, as the one described in [1].

In this report we present the concept of situation dependent path planning and describe how we have implemented the method in B-HAVE [8] for simulated Main Battle Tank (MBT) units. We present the choices done in our implementation and mention possible adjustments and alternative choices. The MBT is used as an example, and the concept of situational dependent path planning can easily be applied to other types of entities, both simulated and unmanned. The vehicle data used in this report are not exact data for one specific type of MBT. We have aimed to show the possibilities of this path planning method, not to adjust the method to a specific type of vehicle.

3 Path planning in B-HAVE

The simulation system described in [1, 6] consists of two parts, a multi agent system which models battle management, and computer generated forces (CGF) based on the CGF platform Virtual Reality-Forces (VR-Forces) [7]. In order to demonstrate the path planning method described in this report, we have implemented it as a part of Brains for Human Activities in Virtual Environment (B-HAVE) [8], a plugin for VR-Forces. This makes it easy to use for our simulations in VR-Forces and the multi agent system can influence the path planning method by providing input parameters when tasking entities to plan and follow a path. In this section we describe the path planning procedure in standard B-HAVE and explain which parts we have modified in our implementation.

B-HAVE contains more advanced artificial intelligence that can be applied to agents in VR-Forces [7]. One of the features of B-HAVE is a path planning procedure. In VR-Forces paths must be planned manually. The functionality in VR-Forces only concerns following a path consisting of straight lines between a sequence of nodes, while using the properties of an entity to plan where to start turning when the path changes direction at a node. The path planning functionality in B-HAVE plans a path between each pair of nodes in the given node sequence. This procedure is based on Kynapse [9], a library for modeling artificial intelligence, and B-HAVE functions as a link between the simulation in VR-Forces and the tools in Kynapse. The path planning procedure in B-HAVE consists of two parts:

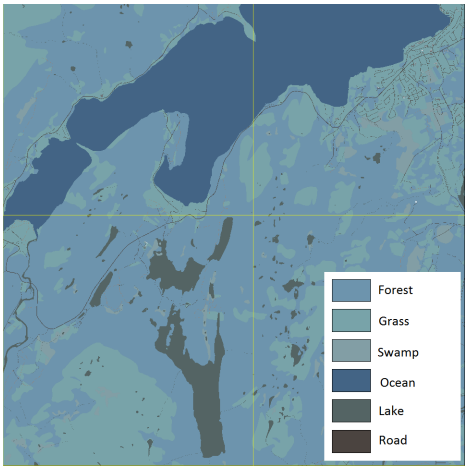
1. **Path data generation**, an automatic preprocessing of the terrain in order to structure the terrain information in a suitable way for the path planning procedure.

2. **Path planning**, a search procedure for detection of an optimal path based on the path data. The path planning is done at run time.

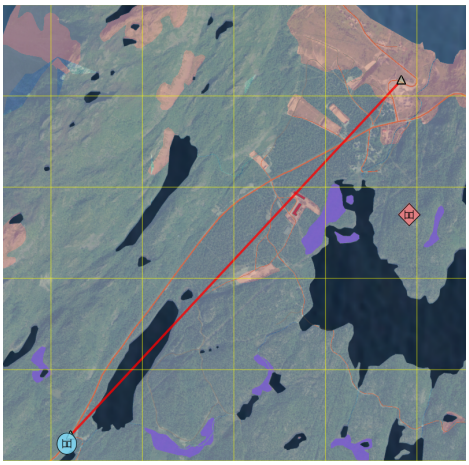
The entire path planning procedure in B-HAVE is illustrated in figure 3.1. The path data generation is illustrated in figures 3.1a–3.1c. This process is entity specific, analyzing the terrain data to determine which parts of the terrain that are accessible to a specific entity based on its properties. We have chosen to create path data for an MBT. Figure 3.1a shows the distribution of terrain types in the area. These data, together with height data and entity properties are the input data to the path data generation procedure. Figure 3.1b shows the generated navigation mesh of the area. The navigation mesh covers the accessible areas with polygons, which the entity is restricted to move within. The borders of the polygons are not shown in the figure, only the polygon colors. The colors symbolize terrain types, as these are integrated in the navigation mesh. We see that the lakes are not included in the navigation mesh, as these are not accessible to an MBT. Areas with too high slope are also excluded from the navigation mesh. Figure 3.1c shows a detail image of the graph corresponding to the navigation mesh. This graph is a discretization of the navigation mesh and will be used during path planning. The reason for path data generation is to structure the necessary terrain information in a compact and easily accessible way, i. e. a navgiation mesh and its corresponding graph, before starting the path planning procedure.

The path planning is illustrated in figures 3.1d–3.1f. In order to find the best path through the terrain, we need to weight the edges in the graph shown in figure 3.1c according to some measure for optimality. In B-HAVE distance is used as weights, enabling the path planning procedure to search for the shortest path. Finally, after weight computation, the weights are used for an A* search through the graph. This produces an optimal path based on the chosen weights. An example of a resulting path is shown in figure 3.1f.

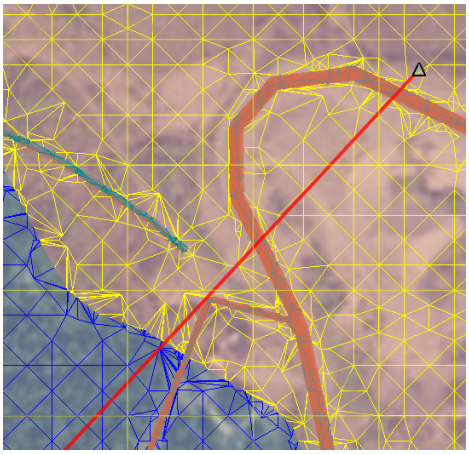
In our implementation of situation dependent path planning we create separate sets of weights for several aspects, and combine these in a new situation dependent set of weights, as shown in figure 3.1e. This step is not part of the original path planning functionality in B-HAVE, which only uses the



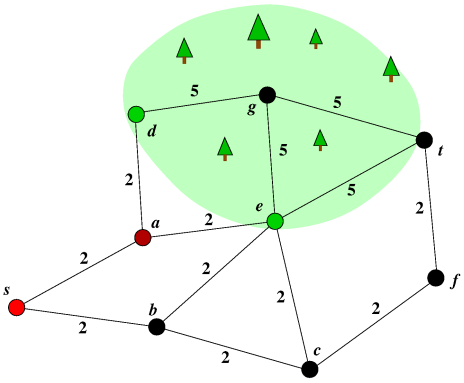
(a) The terrain data is processed with respect to the properties of the entity.



(b) The result of the preprocessing is a navigation mesh. The accessible areas are in bright colors.



(c) Details from the graph corresponding to the navigation mesh.

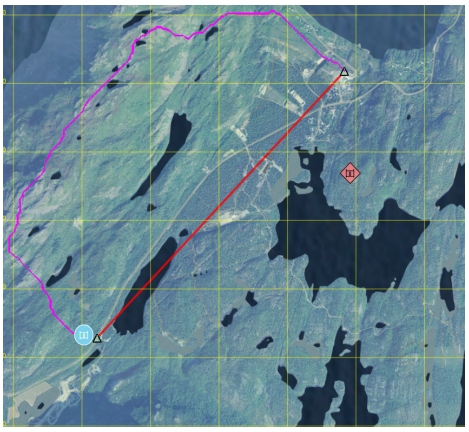


(d) Separate weights for each aspect are calculated. This is an example graph with edge weights for one aspect.

	Weights				
Edge	Time	Cover	Concealment	Accessibility	Threat
sa	4	2	1	2	4
sb	3	4	1	2	4
ad	5	8	1	2	1
ae	4	3	1	2	4
dg	9	6	5	5	1
...	...	...	...	...	...
	Priority				
	8	7	5	3	1

$$w_{sa} = 2^8 w_{sa}^{time} + 2^7 w_{sa}^{cover} + 2^5 w_{sa}^{concealment} + 2^3 w_{sa}^{accessibility} + 2^1 w_{sa}^{threat}$$

(e) A situation dependent set of weights is computed based on the aspect weights and their priority.



(f) The optimal path through the weighted graph is found by an A* search.

Figure 3.1 The path planning procedure step by step. Originally B-HAVE only computes distance weights in step d, and it skips step e. We have added weight computation for more aspects in step d, and the combination step e to obtain situation dependent path planning.

distance aspect, but is added in the implementation of our new path planning method. These new situation dependent weights are used in the A* search.

We will now explain path data generation and path planning in more detail and focus on the parts that are important for the new path planning method we propose. At the end of this section we give a brief explanation of path following in B-HAVE. Path following is strictly speaking not a part of path planning, but is included here since we need to modify the path following in B-HAVE in order to make the entities move along the planned paths.

A basic description of path planning based on a navigation mesh is found in [5], which includes background information on path planning and an overview of alternative methods.

3.1 Path data generation

The term *path data* is in Kynapse used for restructured terrain information, which is the product of preprocessing of the terrain data in combination with the properties of a specific class of entities. A set of path data consist of a navigation mesh, which contains the areas accessible to the class of entities, and a corresponding graph, which is a discretization of possible movements in the navigation mesh. The graph is used to search for paths, while the navigation mesh is used for movement along the planned paths, such that the smoothed path will lie within the navigation mesh.

The path data generation process takes as input the height data of a terrain and the terrain types in the area. This information is combined with the properties of the class of entities for which path data is to be generated. B-HAVE creates two sets of path data, one for lifeform entities and one for ground platform entities. The path data generation process is done separately for each class of entities. During the path planning process the areas with accessible terrain types and topology for one type of entity are included in the corresponding navigation mesh, which represents all accessible areas for this class of entities. Based on this navigation mesh, a graph is constructed to represent a subset of positions and movements in the navigation mesh. The nodes represent discretized positions, while the edges represent possible movements between these positions. The graph is contained in the navigation mesh, but it does not necessarily cover every little area of the mesh.

Before generating path data, the input parameters can be modified. This configuration is done separately for the two classes of entities, and it allows the user to set properties like entity height and radius, maximum slope and step it can traverse, and the desired graph density and raster precision. A list of terrain types are shown, and the user can specify for each terrain type whether it is accessible or not for each of the two classes of entities. The full list of input parameters, with the default values, are shown in table 3.1. As we see from the settings, entites are viewed as cylindrical objects with a height and a radius. The size is used to determine which locations in the terrain the entity can be positioned at. A vehicle would for example not fit in a normal doorway, and it cannot cross a bridge narrower than its own width.

The results of the path data generation is a navigation mesh and a corresponding graph for ground platforms, and mesh with corresponding graph for lifeforms. The navigation mesh and corresponding

Parameter	Lifeform	Ground platform
Entity height	1.5	2
Entity radius	0.3	2
Graph density	20	20
Slope maximum	46	46
Step maximum	0.56	0.56
Raster precision	0.1	0.2
Include road network	Yes	Yes
Include road soil type	Yes	Yes
Terrain types		
Soft soil	Yes	Yes
Swamp	Yes	Yes
Mud	Yes	Yes
Forest	Yes	Yes
Grass	Yes	Yes
Cultivated fields	Yes	Yes
Orchards	Yes	Yes
Rock	Yes	Yes
Boulder	Yes	Yes
Sand	Yes	Yes
Flimsy	Yes	Yes
Body of water		
Undefined soil type	Yes	Yes
Paved road	Yes	Yes
Dirt road	Yes	Yes
Gravel road	Yes	Yes
Muddy road	Yes	Yes
Asphalt road	Yes	Yes
US railroad	Yes	Yes
Euro railroad	Yes	Yes
Ocean		
Lake		
River		
Pond		
Stream		
Brook		
Undefined water		

Table 3.1 Default settings for path data generation

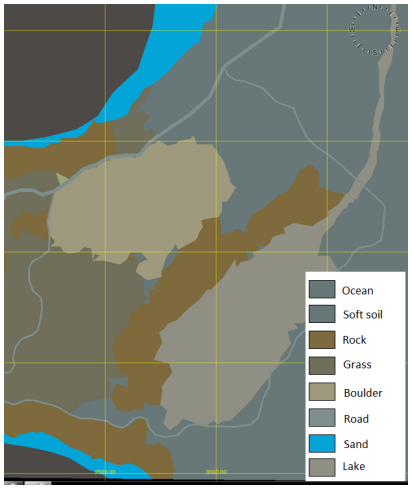


Figure 3.2 The distribution of soil types in an area.

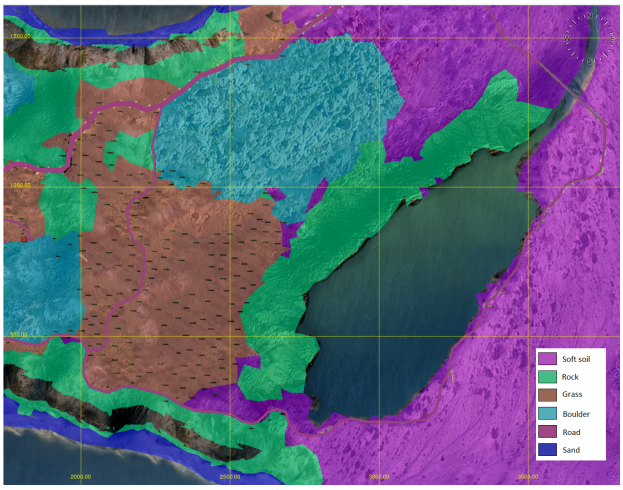


Figure 3.3 The navigation mesh in the path data for ground platforms.

graph for ground platform entities does only cover the areas that are accesible to ground platforms. Areas that are too steep or of a terrain type that is inaccessible to ground platforms are not included in the path data. For example will the ocean not be included in the path data for a ground platform, while it may be included for lifeforms if we allow them to swim. The generation also considers the size of the entity, such that path data excludes locations where the entity is too large to fit, for example narrow tunnels may be included in path data for lifeforms but not for ground platforms.

The graphs are constructed in areas with accessible terrain types, and the density is determined from the given graph density value. A dense graph means a graph with shorter distance between neighboring nodes, and therefore with shorter edges and a larger number of nodes per area. Since the graph is to be used for movement, it should not be very sparse, and there is a limit to how sparse it can be created. The default value seems to be close to or at this limit, but it is possible to create denser graphs. The graph is created such that no edge has a higher slope than the given maximum value, and no edge crosses a step higher than the maximum step for the entity type. When determining the slope and step, the given raster precision is used.

The input parameters restrict access to certain areas of the terrain. If the ground platform path data is to be used for several types of ground platforms, all areas accessible to at least one of the entities must be included. We can then only exclude areas that are too steep for all entities and areas with terrain types that are inaccessible to all entities. The access to areas that are only accessible to some of the ground platforms, must then be controlled at runtime during path planning.

An example of a distribution of soil types in an area is shown in figure 3.2. For each of the two entity types, lifeform and ground platform, their accessible soil types, maximum slope, and so on are given by the input parameters. The resulting navigation mesh for ground platforms is shown in figure 3.3. The navigation mesh is shown on top of a photography of the terrain, and the colored areas are included in the navigation mesh. The navigation mesh for each terrain type is assigned

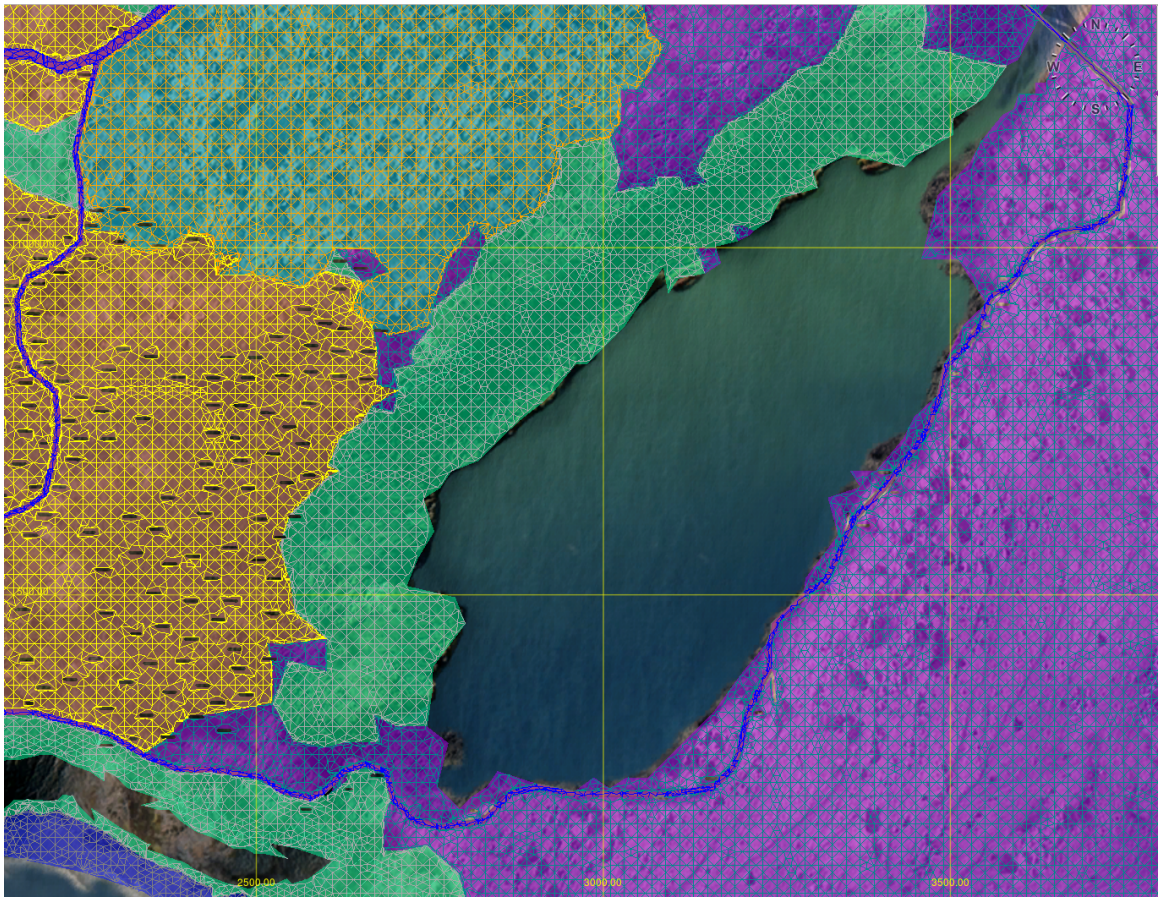


Figure 3.4 The soil types and navigation graph created for ground platforms.

a color, but these colors are not the same as in figure 3.2. However, we can recognize the shapes of the terrain type areas from this figure. We see that the lake and the ocean are not included in the navigation mesh, as well as steep areas along the shore, as these are inaccessible to the ground platforms. Several small areas in the grass covered area left of the lake are also discarded due to slope. A part of the corresponding graph for ground platforms is displayed in figure 3.4. The yellow and grey edges of the graph visualize the paths the entities can follow.

Generation of path data is time consuming. Depending on the area and the computer used, the path data generation may take from minutes to days to complete. Therefore path data are precomputed for each terrain, and the results are stored in datafiles. As mentioned above, the path data consist of navigation meshes and graphs with nodes and edges, and data associated with each node. The standard data associated with the nodes are typically terrain types and elevation, but through customization it is possible to add more types of data to each node.

The main reason for path data generation is to speed up terrain reasoning during the path planning process. The terrain data are in the navigation mesh and the graph structured in a way that allow easy access, and inaccessible, and therefore irrelevant, regions are discarded. This minimizes the areas that need to be considered, and time consumption for each check of terrain data is reduced.

3.2 Path planning

The path planning for an entity is done at runtime, based on the precomputed path data. B-HAVE plans a path from the current location of the entitiy to a given end location. If we set the "behavement mode" of an entity to *free*, each edge in the graph is assigned a weight equal to the edge length. The optimal path to the end location from the current location is determined through an A* search in the weighted graph. When the weights are based on edge lengths alone, the result of the search is the shortest path to the end location. Each edge weight is computed at runtime, every time an edge is evaluated in the A* search. See [5] for a thorough description of the A* algorithm.

The current version of B-HAVE distinguishes only between two groups of terrain types, namely roads and non-roads, and there are two "behavement modes" available in addition to *free*, namely *prefer roads* and *avoid roads*. When tasking an entity to plan a path, we can choose between these three "behavement modes". If we choose to prefer roads, terrain types in the road group are preferred, and if we choose to avoid roads, the non-road terrain types are preferred. Edges with the preferred terrain types are given weights equal to the edge length, while edges with other terrain types are multiplied with a penalty factor, resulting in an increased weight. The result is that the A* search tends to avoid these edges in the final path.

The geometry of the terrain is only used to compute distances during the path planning process. As mentioned above, the three dimensional lengths of the edges in the graphs are used as weights. The heuristic function used in the A* search to approximate the remaining distance to the end node is also based on distance calculation. The value of the heuristic function for each node is the Euclidean distance between this node and the end location.

3.3 Path following

Path planning based on an A* search through a graph results in a piecewise linear path consisting of a sequence of nodes and edges. Such a path generally has several sharp corners, and the movement of an entity that follows these edges accurately will appear jagged and unnatural. Therefore such paths are smoothed before or while entities move along them. In B-HAVE path smoothing is done dynamically while the entity is moving along the path.

The implemented path following in B-HAVE is done by checking whether the straight line from the current position to each node along the path is intersected by an obstacle. When the straight line to a node does not intersect any obstacles, the node is said to be visible. The search begins from the start node, and when the straight line to a node intersects an obstacle, the entity begins moving in a straight line towards the previous node, as shown in figure 3.5. When more nodes become visible as the entity moves, it will start moving directly towards the node that is currently the last node visible from its position, as shown in figure 3.6. The result is that if there are no obstacles between the start and the end node, the entity will discard the entire planned path and instead move in a straight line to the end node. This makes sense when the weights are based on distance alone, since the optimal

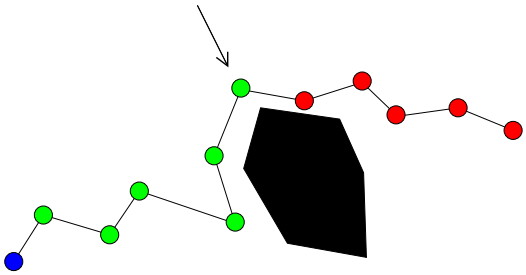


Figure 3.5 The entity starts at the blue node and moves directly to the last of the visible nodes, marked with an arrow.

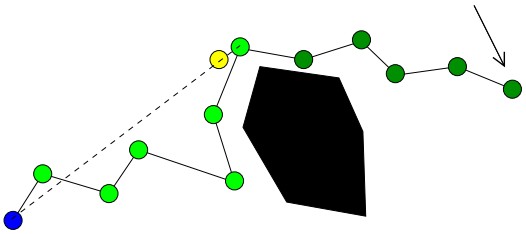


Figure 3.6 While the entity moves, it regularly checks whether more nodes become visible. The yellow dot indicates the position of the entity when all remaining nodes are visible, and it starts moving directly to the last node.

path then is the shortest path, but when more sophisticated weights are used, we need the entities to follow the planned paths more accurately.

Note that if the end node is visible from the start node, but that there are nodes along the path that are not visible from the start node, the entity will not move directly to the end node, but towards the node before the first node that is not visible, counted from the start node.

4 Data for customized weight computation

In this section we present the data that are needed for our path planning method. We start by describing the terrain data, before we present the entity specific data. Finally we present the threat model we developed and the data used in the line of sight calculations.

In order to enable the entities to do situation dependent path planning, we need to establish a dataset containing both suitable input parameters for the path data generation and parameters describing how the terrain influences the behavior of our entities. In this report we focus on path planning for ground platforms, in particular main battle tanks (MBTs), and we here describe the dataset we have established for this type of entity. These data are example data for a typical MBT, and they are not linked to a specific type of MBT. If one wants to plan paths for a specific type of unit, a corresponding data set needs to be acquired for this unit type.

4.1 Terrain data

In order to do terrain analysis during path planning, we need both terrain elevation data and terrain type data. The terrain forms a surface of polygons, where the height data are used as height values for the vertices, which are represented in a local coordinate system with z value as height. Each polygon has a terrain type, and the terrain types supported by B-HAVE are listed in table 4.1. We have used terrains on the MÄK Terrain Format (GDB) that have been built with the MÄK Terrain

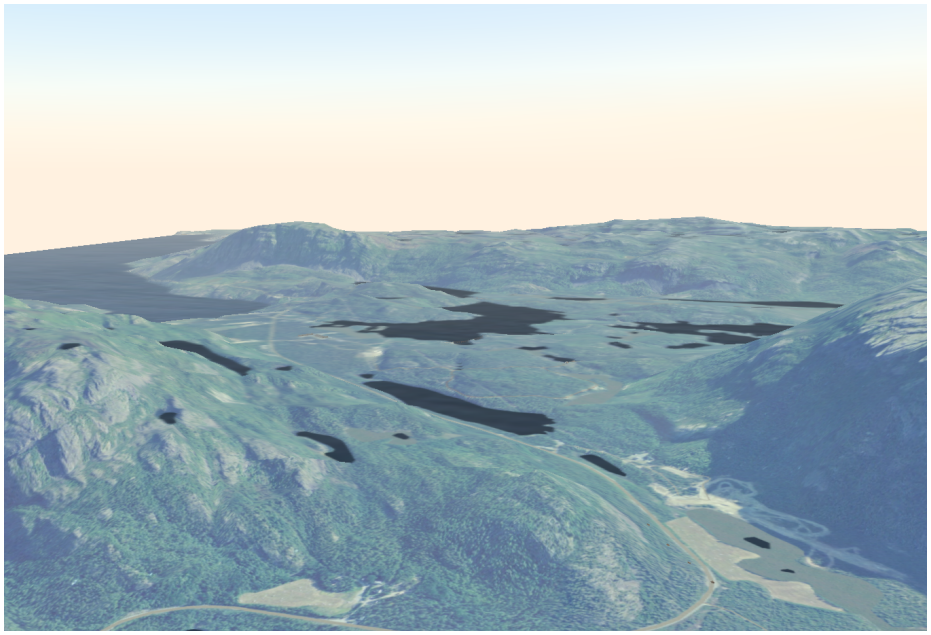


Figure 4.1 One of the terrains used for testing of the path planning method.

Database Tool [10]. A three dimensional view of the smallest of the terrains used for testing is shown in figure 4.1.

4.2 Data for MBT entities

With the help from Subject Matter Experts (SMEs) we have created tables for typical MBT units describing accessibility and speed depending on terrain types, and threat levels depending on the distance from enemies. The data for accessibility and speed are presented in table 4.1. Speed is given in km/h, and the values are valid for areas with slope smaller than 10-15 degrees. For steeper areas, the speed is 5 km/h regardless of terrain type. Accessibility is assigned an integer between 1 and 10, where 10 indicates a terrain that is difficult or almost impossible to cross, and 1 is a perfect terrain type for movement. The table is created as an example of representative values, and the values may be modified for a specific MBT. In this report the focus is to obtain plausible values in order to demonstrate the situtation dependent path planning algorithm, not to determine exact values for a specific vehicle.

Preference for terrain types is based on accessibility, risk of damage to equipment, and risk of getting stuck. In some cases a preferred terrain type requires low speed. This is the case for rock, which has low risk of damage to equipment, and low risk of getting stuck, but requires low speed because of the irregularities of the terrain and the need for continuous small scale path finding. Therefore this is a preferred, low risk terrain type, but not a terrain type chosen for fast movement.

Parameter	Preference	Speed (km/h)
Soft soil		
Swamp		
Mud		
Forest		
Grass		
Cultivated fields		
Orchards		
Rock		
Boulder		
Sand		
Flimsy		
Body of water		
Undefined soil type		
Paved road		
Dirt road		
Gravel road		
Muddy road		
Asphalt road		
US railroad		
Euro railroad		
Ocean		
Lake		
River		
Pond		
Stream		
Brook		
Undefined water		

Table 4.1 The necessary data for the behavior of MBTs on various terrain types. The column Preferences should contain a ranking of the accessibility of terrain types. The value 1 is used for the most accessible types of terrain, and 10 for the least attractive types of terrain. Inaccesible terrain types are assigned an infinitely high number. The last column should contain the average speed for the MTB on each terrain type. The exact numbers we have used are not disclosed.

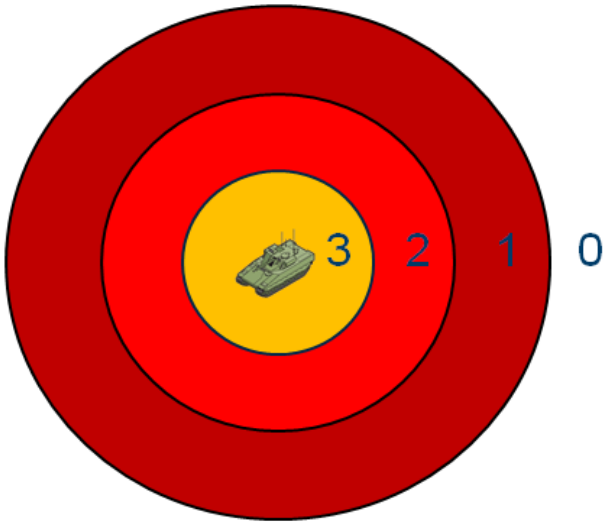


Figure 4.2 The threat model depends only on distance.

Threat value	Distance to enemy
3	< X
2	X - Y
1	Y - Z
0	> Z

Table 4.2 The distance limits for each threat level. The exact distances we have used are not disclosed.

4.3 Threats

We want to model the threat posed by being at a certain distance to opposing entities, in order to let the situation influence the acceptable threat level, or risk, corresponding to different locations. We have considered MBT enemies equipped with cannon and have chosen a basic threat model based on distance from opposing entities. The threat level is divided into discrete values; High threat (3), moderate threat (2), low threat (1) and no threat (0), as illustrated in figure 4.2. This model is based on the effect of the opposing entity's gun, which depends on distance. With increasing distance, the probability of hit decreases, as well as the potential damage. The distance limits for these cathegories are described in table 4.2.

Threat is in this model based on distance alone, and is not dependent on line of sight. This choice is based on the fact that artillery fire may hit also when the firing entity has no line of sight to the target.

Based on information from SMEs we model the influence of threats on path planning such that a

path either does not consider threats, for example based on the knowledge of enemy activities, or that it aims to avoid areas with threat value 1 or higher. This means an on/off modeling of threats based on the chosen model.

4.4 Line of sight

Line of sight calculations are based on the triangulated terrain, and they check the intervisibility between two locations in the terrain. Each entity has an observation position, which is the height from which the entity observes its surroundings. Line of sight is calculated from this position on the entity to specific positions in the terrain. If the straight line from the observation position to a point in the terrain does not intersect the terrain, the entity is said to have line of sight to this specific point.

When we consider line of sight from an entity to possible points along a path, we do not choose points on the ground along the path, but points 3 m above the path. This is because we are interested in whether enemy entities can observe friendly vehicles of a height around 2-3 m. If the enemy entity has line of sight to a point 3 m above the ground, entities at the corresponding ground position are considered within line of sight of the enemy, and the location should be avoided by friendly entities. If the point 3 m above the ground is not visible from the enemy, the location is considered safe.

In the current terrain model some triangles are assigned terrain type forest, but these triangles are not elevated above the ground, and the height of the forest is not included in the model. There is therefore no three-dimensional representation of forest, and forests will therefore not influence the line of sight calculations in the current implementation in VR-Forces. In the future we may consider to improve this aspect of the line of sight calculations.

5 Computation of weights

We want to improve the path planning procedure in B-HAVE by introducing more advanced weights for the graph edges, and to vary these weights such that different paths can be produced depending on the situation and entity. As a first step to construction of these situation dependent edge weights, we compute five separate sets of weights based on aspects that influence military path planning. These aspects are time, cover, concealment, accessibility and threat. These sets of weights will afterwards be combined in suitable ways in order to adjust the path planning procedure to different situations, as we will explain in the next section. In this section we will focus on how the weights in each category are computed.

The five weight categories should be considered possible choices, and their implementations examples of how they can be computed. In this chapter we evaluate our choice of weight categories and discuss possible improvements and alternative solutions. The aim at this point is to determine whether this approach for advanced path planning is able to produce reasonable paths for different situations. Improvements of each category and of the combination method will be left to future work.

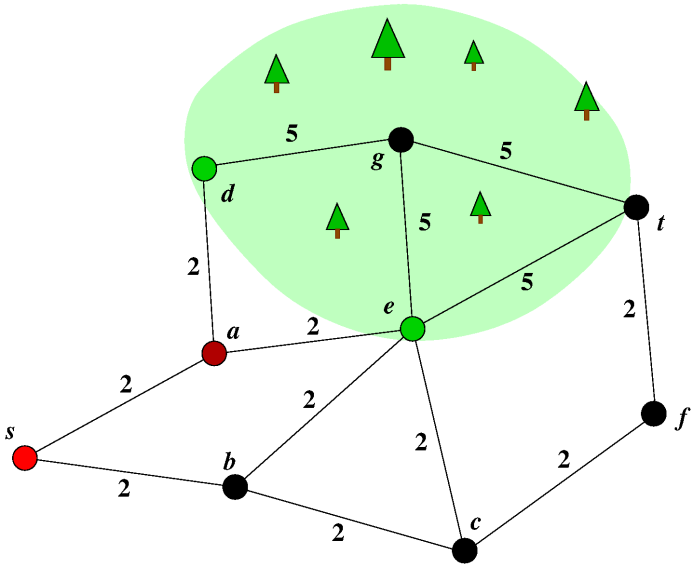


Figure 5.1 The path planning procedure.

5.1 Computation of an edge weight

We need to determine how an edge weight should correspond to the properties of the start node and end node of the edge. The edge weight can depend on one of or both nodes, and of the difference between the node properties. We mainly use the properties of the end node and the difference between the two nodes. The reason for this choice is as follows.

Let us first recall how the path planning procedure, which is explained in detail in [5], works. The weights are used to determine an optimal path from a start node s to an end node t. It starts at node s and investigates all edges from this node to its neighboring nodes, determining the cost to each of these nodes along the investigated paths. After determining the assumed cost to all neighboring nodes, the search continues with a node which has the lowest cost. If several nodes have the same cost, one is chosen according to a predefined rule, for example the one with the lowest index. All edges from this node are investigated to check the assumed cost to its neighboring nodes. An example is shown in figure 5.1, where we are investigating how a path from node s to node a should continue. The two possibilities are the nodes d and e. Since we already assume node a to be in the path and are investigating which new node to add, it makes sense to mainly use the properties of the candidates d and e to determine the edge weights. The terrain types of these nodes may influence the choice of where to plan the path. We also let the distance and slope between the current node a and each of the candidates d and e influence the weights. This strategy for weight computation results in directed edges, meaning that the weight of an edge will depend on the direction in which we move along the edge.

5.2 Computation of weights for each category

The B-HAVE plugin is a connection between VR-Forces and Kynapse, and the data used for calculation of the weights in each category is available from one or both of these. We will now explain how we have computed new weights in B-HAVE. We have not changed other parts of the path planning algorithm, but have used the existing A* implementation in B-HAVE.

We illustrate the impact of the distance and each of the five categories by showing examples of the resulting paths when an MBT is to move from its start position to a specific location, taking only one aspect into account in each example. The results are shown in figure 5.2. The start and end nodes of the path are identical in all six examples. We have used the input data described in the previous section in these examples. The start and end nodes are situated in an area mostly covered with forest, with a road from the start node through the forest to the end node. There is a hill to the left of the road and a small lake to the right of the road in the first section of the road. After about two thirds of the distance to the end node, the road moves away from the hill and into a flat area, where the end node lies. The end node lies outside the forest, in a grass covered area. The yellow grid has a resolution of 500 m, and the total distance between the start and end nodes is approximately 2.5 km.

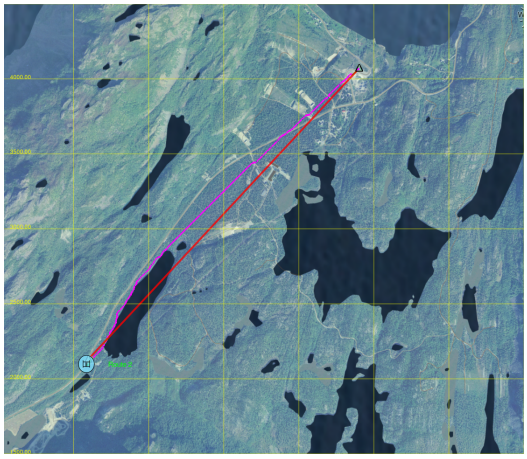
5.2.1 Distance

We do not consider distance a separate type of weight, but it influences other types of weights, and this influence needs to be explained in more detail. Without distance dependent weights, the optimality of a path would depend of the number of edges in the path. This would cause problems for graphs with varying edge lengths, like the graphs constructed in the path data generation process in B-HAVE. These graphs have high resolution along roads, which without distance dependent weights easily would cause optimal paths to avoid roads.

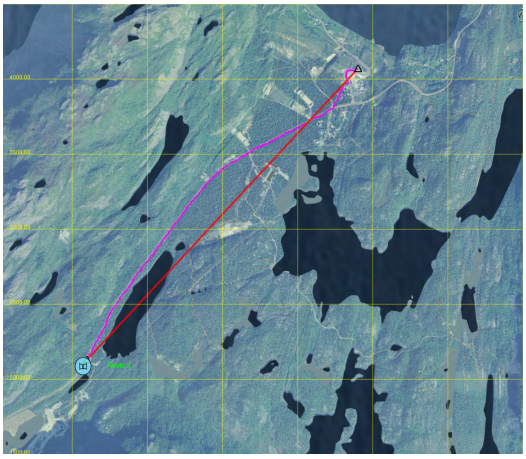
Distance dependent weights are important not only with respect to graph resolution. An optimal path may cross a difficult terrain for a short distance, for example crossing a forest at the narrowest point, in order to avoid a long detour along a road around the entire forest. In this way distances are important in weight computation, although they are normally not sufficient alone for path planning through terrain. For indoor path planning and path planning in uniform areas, like a football yard, distance alone may be a suitable option for graph weights.

We have let all the weight categories be functions of distance, in order to resolve the issues mentioned above. The distance is computed in three dimensions, meaning that the height difference along the edge is taken into account. The necessary input for distance computation of an edge is the coordinates of its start and end nodes. These coordinates are in Kynapse represented in meters in a Cartesian coordinate system, and a computation of the length of the three dimensional vector from the start node to the end node gives the distance along the edge in meters. Let $\mathbf{u} = [u_1, u_2, u_3]^T$ denote the start node and $\mathbf{v} = [v_1, v_2, v_3]^T$ denote the end node of an edge. The distance $d(\mathbf{u}, \mathbf{v})$ along this edge is then given by

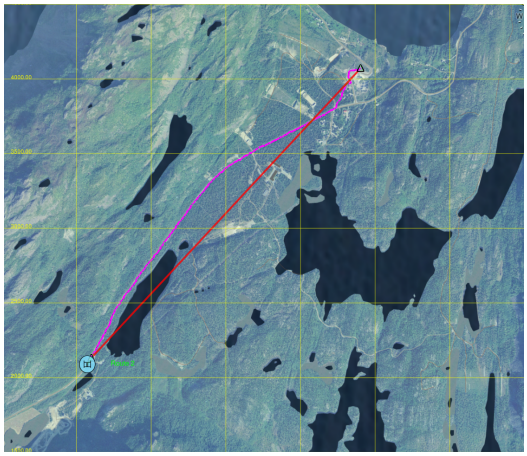
$$d(\mathbf{u}, \mathbf{v}) = \sqrt{(v_1 - u_1)^2 + (v_2 - u_2)^2 + (v_3 - u_3)^2}. \tag{5.1}$$



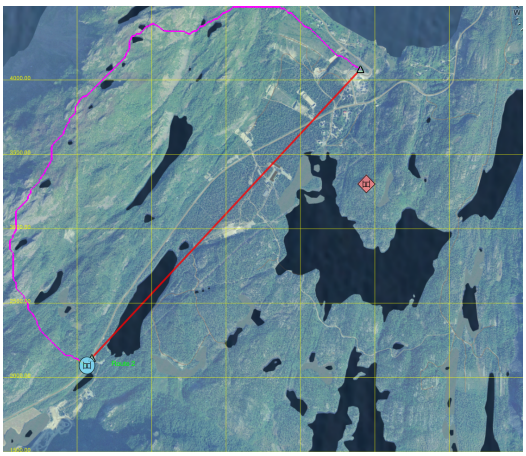
(a) Shortest distance



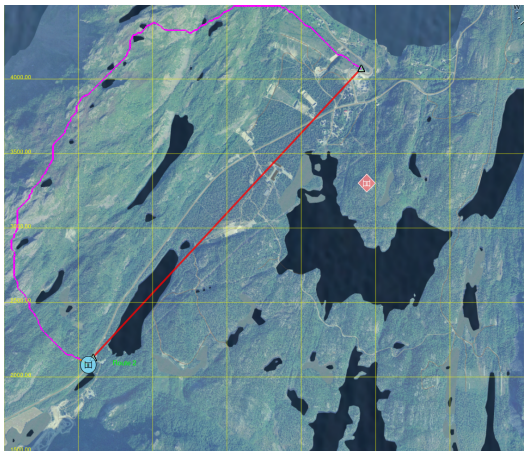
(b) Shortest time



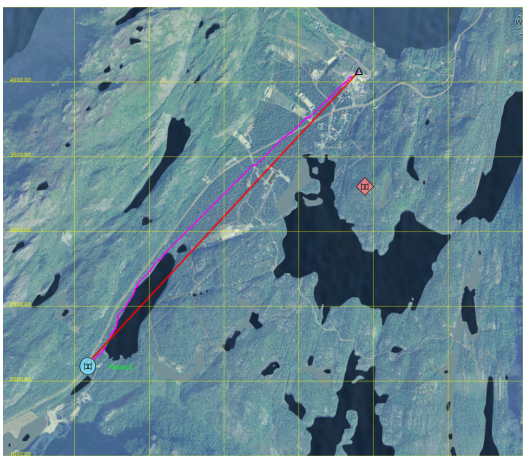
(c) Best accessibility



(d) Maximal concealment



(e) Maximal cover



(f) Minimal threat. In this example most of the area is within threat distance, and the shortest path is computed to minimize the distance traveled within enemy weapon range.

Figure 5.2 The optimal path for each aspect.

An example of an optimal path based on minimization of distance is shown in figure 5.2a. We observe that this path mostly lies relatively close to the road, but not on it, as it is slightly shorter to cut through the forest. Paths computed based on distance alone will not follow terrain type structures, as terrain type is not considered when computing the distances. These paths will therefore only distinguish between accessible terrain types, which are included in the graph, and inaccessible areas, in which there are no nodes or edges.

5.2.2 Time

Time is an important factor in military operations, andthe simulated entities will in many situations need to choose paths that require a short amount of time, in addition to satisfy other requirements. We have therefore chosen time as a weight category. The amount of time needed to move to a location depends on the distance and the terrain types along the path. Each terrain type is assigned a speed value that is considered a mean value for movement in that type of areas. We have used the values in table 4.1. The speed is also dependent on the slope, and if the slope is higher than 10 degrees, the speed is set to 5 km/h, regardless of soil type.

Time usage is computed based on the distance between the start node $\mathbf{u}$ and the end node $\mathbf{v}$, and the terrain type in the end node, which we denote by $T_{\mathbf{v}}$. The distance is computed as explained in equation 5.1, and the speed $s(T_{\mathbf{v}})$ depends on the terrain type in the end node, as explained above. The time weight w_t for this edge is given by

$$w_t(\mathbf{u}, \mathbf{v}) = \frac{d(\mathbf{u}, \mathbf{v})}{s(T_{\mathbf{v}})}. \tag{5.2}$$

The choice to use the terrain type in the end node means that we approximate by assuming that the terrain type along the entire edge is the same as in the end node. This is true if both nodes have the same terrain type, but not if the edge crosses a border between two areas with different terrain types. This approximation makes computations faster and may be accepted for several reasons. The terrain type areas are relatively large, so relatively few edges cross such borders. Also, the edges crossing the terrain type borders are quite short, as the graph is denser along these borders. Finally, the terrain type in the end node influences whether we should go in this direction at all, while the terrain type of the start node was considered in the previous path planning step. If this choice seems insufficient in an application, the function for w_t can be adjusted to account also for the terrain type in the start node and the actual edge length in each terrain type area.

Figure 5.2b shows how the path between the same pair of nodes as in figure 5.2a would be planned if we search for the fastest path instead of the shortest path. Since movement is faster along roads, we see that this path instead follows the road as much as possible. This path is longer, but requires less time to traverse.

5.2.3 Accessibility

Accessibility depends on the combination of terrain type and entity type, and table 4.1 shows the type of data used for computation of accessibility weights. Areas that are inaccessible due to slope

are not included in the navigation mesh, and we will therefore not compute accessibility weights in these areas. The accessibility values are mostly between 1 and 10, but are set higher for terrain types that are to be avoided as much as possible. For example crossing a river at a location without a bridge should only be done when it is necessary, as it is time consuming and may require special equipment, like a temporary bridge. The accessibility value is in this case set to 1000. This value should be investigated further in order to determine a suitable value to allow river crossings, but still avoid unnecessary crossings. Also, the accessibility values need to be calibrated for specific MBT types such that path planning based on the accessibility weights produces reasonable paths.

The values for computation of accessibility based weights are the distance $d(\mathbf{u}, \mathbf{v})$ between the start and end nodes, the terrain type $T_{\mathbf{v}}$ of the end node, and which set of accessibility values to use for the end node, $a_A(T_{\mathbf{v}})$ or $a_B(T_{\mathbf{v}})$. The edge weight is computed as the product of the distance and the accessibility value for the terrain type in the end node,

$$w_a(\mathbf{u}, \mathbf{v}) = d(\mathbf{u}, \mathbf{v}) a_A(T_{\mathbf{v}}), \tag{5.3}$$

here shown for the choice of set A for accessibility values.

The reasons for using only the terrain type in the end node for computation of the edge weight is as for speed weights. There are relatively few edges crossing terrain type borders. And the choice of moving to the start node was made in the previous step, while we in this step consider whether it is preferable to move in the direction of the end node, considering its properties.

For our example the two alternative sets of accessibility values result in relatively identical paths. One of the paths is shown in figure 5.2c, and the figure for the other set of values is not included, as it appears to be identical. The path follows the road, similar to the path minimizing time usage. This is as expected, as roads both are easily accessible and allow higher speed. There is often a correlation between speed and accessibility, but we have still chosen to include both categories, since some terrain types can be traversed at a reasonable speed, but the terrain type can cause damage to the equipment and will therefore be avoided when possible. This is for example the case for forests or hard rock surfaces.

5.2.4 Concealment

To be concealed means to be in a location which reduces the visibility of an entity without protecting it from fire. This is the case in forests, for example. Leaves and branches make entities difficult to spot, but projectiles will pass through.

The computation of concealment weights depends on whether the end node is located in a forest or not, such that if concealment is desired, an end node positioned in a forest is preferable in order to obtain, or sustain, reduced visibility. This reduced visibility is attained if the enemy entities have the end node within line of sight and the end node is in a forest. Nodes out of enemy line of sight are given an even lower visibility, as these nodes also provide cover.

The necessary input for computation of concealment weights is the distance between the start and end

nodes, the terrain type of the end node, and known enemy positions. We define the function $los(\mathbf{v})$, which is true if the point 3 m above the ground at node $\mathbf{v}$ is visible from a position approximately 2 m above the ground in any of the enemy positions and false if the point is not visible from any enemy position. We check intervisibility for the point 3 m above the ground, as this point lies above the highest point of the entity, so if this point is not visible, the entity will not be visible when positioned in this node. Similarly, we use a point approximately 2 m above the ground at the enemy positions, since this is a typical height for the location of the observation equipment of an entity. In addition to checking line of sight, we give a lower weight if node $\mathbf{v}$ is in the forest, compared to in open areas. We have used the function in equation (5.4) for computation of concealment weights.

$$w_{con}(\mathbf{u},\mathbf{v}) = \begin{cases} d(\mathbf{u},\mathbf{v}) & \text{if not } los(\mathbf{v}), \\ 5d(\mathbf{u},\mathbf{v}) & \text{if } los(\mathbf{v}) \text{ and } T_{\mathbf{v}} = forest), \\ 10d(\mathbf{u},\mathbf{v}) & \text{if } los(\mathbf{v}) \text{ and } T_{\mathbf{v}} \neq forest). \end{cases} \tag{5.4}$$

The resulting path when using concealment weights is shown in figure 5.2d. This path lies on the side of the hill opposite of the enemy position. The path is significantly longer than the previously computed paths, but time consumption is not a factor here. Neither is accessibility, and the path lies mostly in forests.

The implementation of the concealment weights is a topic for further investigation, mainly because of its connection to cover weights. We may choose to exclude the concealment weights from the path planning process and consider cover weights alone to be sufficient for our purpose, possibly by adding the concealment aspect to the cover weights. The concealment weights could also be implemented without considering line of sight, such that this category would encourage movement through forests also out of enemy line of sight. An argument for this approach is that this includes concealment from aircrafts and unknown enemies.

5.2.5 Cover

Cover weights are based on whether a node is visible from known enemy positions, or whether the view is blocked by terrain structures. If the view is blocked, the terrain structures will also block direct fire, which provides cover. The cover weights are therefore based on calculations of line of sight from known enemies. The cover weights are computed as concealment weights in non-forest areas.

The necessary input for cover weight computation is the distance between the start and end nodes, and enemy positions. Line of sight is checked between a point 3 m above the ground at the end node and a position approximately 2 m above the ground in each enemy position. If the end node is within line of sight of an enemy, the weight is set to ten times the distance, otherwise the weight is set equal to the distance,

$$w_{cov}(\mathbf{u},\mathbf{v}) = \begin{cases} d(\mathbf{u},\mathbf{v}) & \text{if not } los(\mathbf{v}), \\ 10d(\mathbf{u},\mathbf{v}) & \text{if } los(\mathbf{v}). \end{cases} \tag{5.5}$$

This implementation means that 100 m of movement within line of sight of an enemy is equally attractive as 1 km movement out of enemy line of sight. One would choose to move within line of sight of the enemies if the path out of sight is more than 10 times longer. Whether this factor of 10 is the best choice needs to be investigated further, but it seems to give acceptable and useful results.

The example path for cover weights is shown in figure 5.2e. This path is identical to the one for concealment weights, which is not surprising, as these weights are computed similarly, and both paths lie mostly in forests out of enemy line of sight.

5.2.6 Threat

The threat weights encourage entities to stay out of the weapon range of enemy entities and do not consider line of sight. As shown in table 4.2, the threat levels are divided into levels depending on distance to enemies. Since the distance intervals are quite large, the threat weights are most suitable for large scale path planning. In small scale path planning, the entire path will often lie in one of these intervals, which means that threat will not influence the planning of the path, as all possible edges will be considered to have the same threat. We may decide that this behavior is wanted, or we may choose to divide the threat levels into more categories. For now we have decided to keep threat weights as shown in table 4.2, and that we want these weights to only influence large scale path planning, as this in our opinion reflects the military reasoning in a suitable way.

The computation of threat weights is based on the distance between the start and end nodes, the end node coordinates, and a list of enemies with corresponding positions. Depending on the distance to the nearest enemy, denoted by $d_e(\mathbf{v})$, the threat weight is set according to the function

$$w_t(\mathbf{u},\mathbf{v}) = \begin{cases} d(\mathbf{u},\mathbf{v}) & \text{if } Z \text{ km} < d_e(\mathbf{v}), \\ 4d(\mathbf{u},\mathbf{v}) & \text{if } Y \text{ km} < d_e(\mathbf{v}) < Z \text{ km}, \\ 7d(\mathbf{u},\mathbf{v}) & \text{if } X \text{ km} < d_e(\mathbf{v}) < Y \text{ km}, \\ 10d(\mathbf{u},\mathbf{v}) & \text{if } d_e(\mathbf{v}) < X \text{ km}. \end{cases} \tag{5.6}$$

Higher weight values represent higher threat, and if more than one enemy is present, the threat value corresponding to the nearest enemy is used. It could be considered to increase the threat level based on the number of enemies in near proximity.

The example path based only on on threat weights is shown in figure 5.2f. In practice threat levels do not influence this path, as most of the path, including the end node, lie within the highest threat level. In order to show how threat could influence a path planning, we have divided the distances in function (5.6) by 5, which resembles a finer resolution near an enemy position. The resulting path

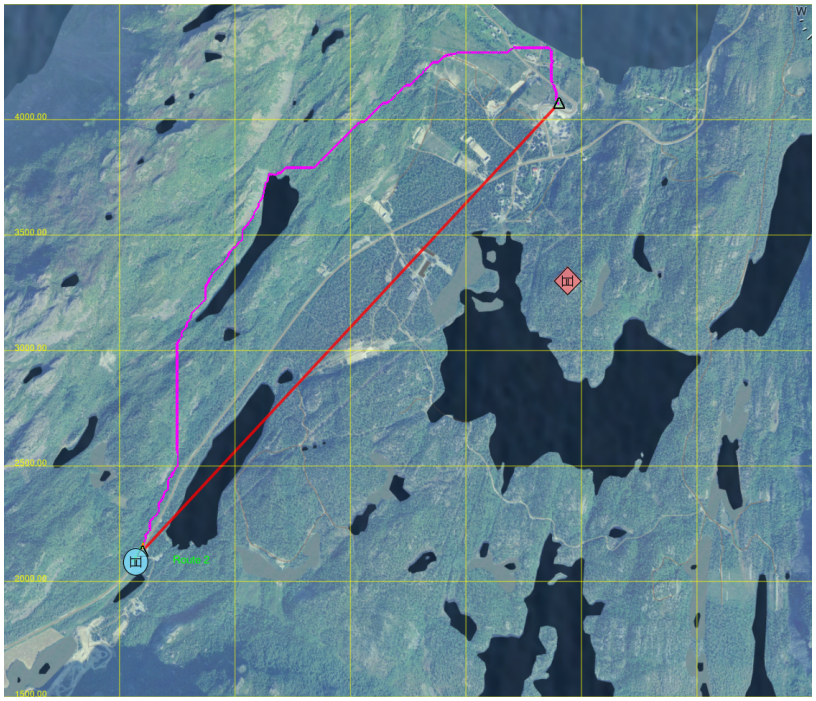


Figure 5.3 The resulting path when only threat is considered and the distances are divided by 5.

for these threat values is shown in figure 5.3. We see that the path tends to maintain a distance to the enemy in order to avoid areas with a higher threat level.

5.3 Necessary adjustments in B-HAVE

B-HAVE translates between VR-Forces terrain types and Kynapse terrain types. In the translation lists in the B-HAVE source code the terrain types "forest" and "body of water" were switched, which caused forests to be treated as water by the path planning algorithm. This is not a problem for path planning in standard B-HAVE, as the terrain type forest is not used in this implementation. It has been suggested that B-HAVE translates forest to body of water in order to make entities avoid movements in forests. However, in order to make our weight computations correct with respect to terrain types, we needed to change the translation of these terrain types.

As mentioned in section 3.3, the B-HAVE path following procedure often skips large amounts of the planned path. Instead of following the path, entities move directly to a point further down the path, often directly to the end point. We have limited the amount of route that is allowed to be skipped, such that not more than 100 m of a path can be discarded at any time. The reason for allowing entities to skip small parts of the path, is that this results in smoother movement for the entities, as they do not move directly along every little edge of the planned path, but continuously towards a node a short distance further ahead, switching to later nodes once they are within a 100 m distance and no obstacles block the direct route to this node.

In the future we may want to implement a new path following routine instead of using the modified

B-HAVE path following functionality. A proposed improved path following controller for VR-Forces is described in [11].

6 Situation dependent path planning

Situation dependent combination of the five aspect weights described in the previous section produces a more advanced set of weights which takes several aspects into account simultaneously. As only the weights are altered, standard path planning functionality for weighted graphs can be reused for the rest of the path planning process, including creation of a navigation mesh, the corresponding graph, and standard A* search algorithms. We have used the built-in functionality in B-HAVE for these parts of the process.

Each of the weight categories described in the previous section take only one aspect of path planning into account. We want a path planning algorithm that considers several aspects simultaneously and plans a path based on multiple aspects of a situation, similar to how military officers plan paths based on the situation at hand and their assigned tasks. In order to make the path planning procedure consider a set of aspects, the corresponding weights need to be combined into a new set of weights in such a way that the influence of each category resembles its importance in the current path planning.

The path planning procedure can be adjusted to produce suitable paths for different cases through situation dependent combinations of the aspect weights described in the previous section. Important factors in such combination methods and possible algorithms are described in [12]. We here describe one of these algorithms and demonstrate how it can be used for path planning for a selection of military tasks in specific situations.

6.1 An algorithm for computation of situation dependent weights

Section 5 gave a detailed description of how different types of weights are computed. We now consider how these sets of weights can be combined to form one situation dependent set of weights. The combination method takes an entity type, a terrain, the aspect weights, and a situation or context as input parameters, and it produces a suitable new set of weights based on the input.

The new combination of several types of weights, will depend on the relative magnitudes of the input weights. We therefore need to assure that the input weights are of a similar size. As we see from the description in section 5, the sets of weights are mostly restricted to the distance multiplied with a coefficient in the interval [1, 10]. This is done to simplify the combination algorithm, as it does not need to first normalize the weights. Note also that all weights are positive, which is required by the A* algorithm. This means that the combination method also needs to produce positive weights.

For weight combination we have chosen the linear combination method with exponential coefficients described in [12]. This method computes the new weights as a weighted sum of powers of 2. Let w_j be the set of weights for category j, where $j \in \{time, cover, concealment, accessibility, threat\}$, and let the category coefficients $c_j \in \{0, 1, 2, \ldots, 10\}$ symbolize the priority of category j, where

high values mean high priority. The new set of weights w' will then be computed by

$$w' = \sum_j 2^{c_j} w_j. \tag{6.1}$$

Since all input weights are positive, this function will always produce positive weights.

Note that with the category coefficient $c_j = 0$, the coefficient for w_j will be $2^0 = 1$, which means that category j will have a weak influence on the path planning even with priority 0. This choice has been made since we assume that an aspect will never be completely ignored, but will weakly influence the path planning even if other aspects are much more important. In practice this choice has little to say, as the influence in most cases will be negligible, so it is more a conceptual choice.

In order to use this path planning algorithm, one needs to establish a set of five integer category coefficients c_j, which prioritize between the aspects of path planning. The category coefficients for the aspects with highest priorities should not have the same value, such that the relative priorities are clear. This is to avoid that two sets of weights cancel each other out. The aspects with lowest priority may have equal coefficients, as these aspects are practically ignored. A suitable set of integers will depend on the type of task and is different whether an entity is to move quickly or to stay hidden, for example. The consideration of how to take enemy positions into account, is done by the weight computation of each category.

6.2 Situation dependent paths and Context-Based Reasoning

This section presents some examples of paths created for different situations. We consider a simulated MTB with artificial intelligence modeled with Context-Based Reasoning, as explained in [1] and [6]. The MTB has the possible contexts *Move, Move Cautiously, Regroup, Observe, Attack, Hasty Attack* and *Wait*. In order to enable the entity to create paths for all types of situations it needs to handle, we assign a suitable set of category coefficients to each of its possible contexts, except for the *Wait* context, which does not include movement. In this section we present several paths produced by the path planning method in order to demonstrate the variations of the produced paths and discuss whether these paths are suitable for the different contexts. However, the choice of parameters for each context will depend on the type of behavior we want the contexts to represent.

The example in figure 6.1 shows different paths between two nodes 2.5 km apart. These paths are computed with different category coefficients, reflecting different tasks and situations. The second example is shown in figure 6.2, with two alternative paths for a pair of nodes that are approximately 7 km apart, using some of the sets of category coefficients that were used for the shorter paths.

In each of the two example terrains, the paths are planned between the same two locations in the terrain and for the same entity. The only difference in input parameters is the set of category coefficients. As we see from the examples, variation in the category coefficients has great influence on the planned path. A challenge is to choose the right set of input parameters for a specific situation and task. At this point we have not calibrated the method for a specific use, but have explored the diversity of paths that can be planned through variation of the category coefficients.

6.2.1 Path planning for fast movement while taking cover

The context *Move* is used for movement through safe areas, for example transit to a location where some task is to be performed. For such movements it can be sufficient to consider time consumption. Factors like cover, concealment and threat are irrelevant if no enemies are present, so the planned path will be as in 5.2b. When moving through areas with enemy activities, the context *Move Cautiously* will be used to adjust the path depending on enemy positions. We will now consider a few possible parameter sets for this context.

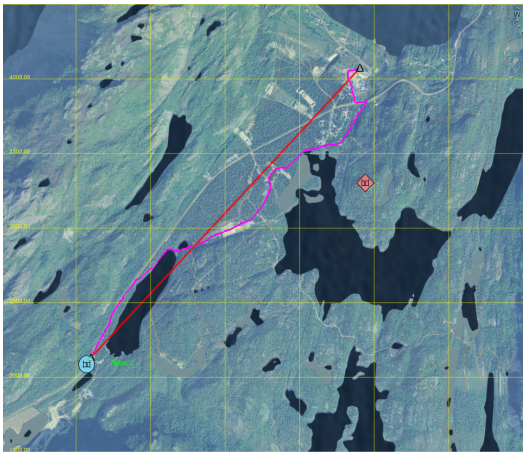
The path in figure 5.2e was planned only with cover weights. We now want a path that is fast while prioritizing cover. Such paths can be useful for example when performing a surprising attack on an enemy and may fit the *Hasty Attack* context. We have used the input parameters $\{time, cover, concealment, accessibility, threat\} = \{6, 7, 4, 1, 1\}$. Cover is given highest priority, and time is almost as important. Concealment is significantly less important, and threat and accessibility are practically ignored. The resulting path is shown in figure 6.1a.

The path in figure 5.2b for fast movement followed the road, while the path in figure 5.2e for hidden movement was at the other side of the hill. We see that the path in figure 6.1a is different from both these paths. It starts along the road and follows it as long as the road is not visible from the enemy position. Thereafter it moves away from the road into the forest in order to stay out of sight as much as possible, before it enters the road again towards the end, where the road no more is visible for the enemy. The path goes relatively directly to the end node, which means that time consumption is relatively low, even though parts of the path is through the forest.

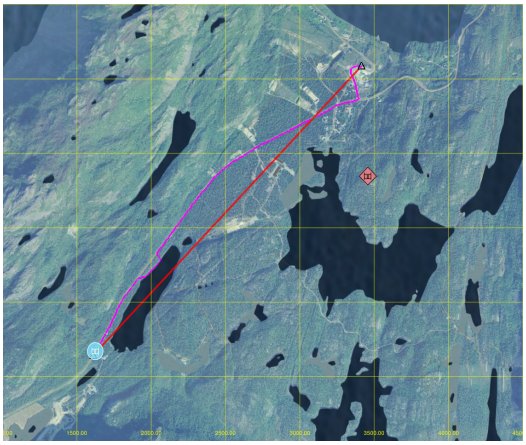
We now want the entity to pay more attention to the accessibility of the terrain. This will cause the entity to prefer roads and other easily accessible areas to a larger extent than in the previous example, but still move fast and take cover when possible. We therefore change the category coefficient for accessibility from 1 to 5, while we keep the rest of the coefficients unchanged. The resulting path is shown in figure 6.1b.

We see that the preference for roads now is so dominant that the path only deviates from the road when a small deviation is enough to obtain cover. This only happens at one place along the path, and the rest of the path follows the road. These input parameters may therefore be less suited for *Move Cautiously*, as cover seems to be sacrificed for improved speed and accessibility, which both prefer roads. However, this is a possible set of parameters for the *Move* context, as it in an area without enemies will reduce to movement prioritizing time and accessibility.

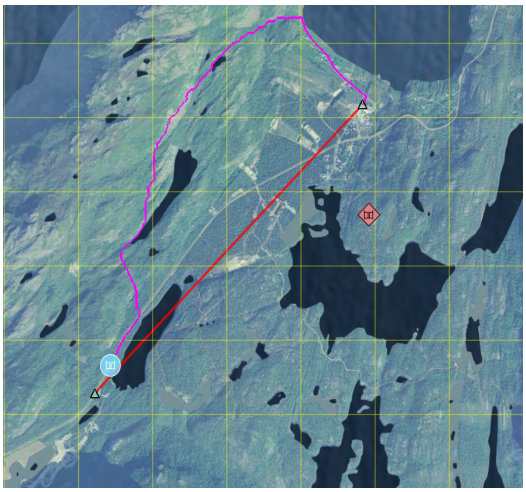
Cover has already the highest priority of the aspects, but we try to task the entity to prioritize cover higher relative to the other categories by increasing its coefficient from 7 to 9. The parameters $\{time, cover, concealment, accessibility, threat\} = \{6, 9, 4, 1, 5\}$ result in a different path, shown in figure 6.1c. We see that this path lies closer to the path in figure 5.2e, which was based only on cover weights, but that it follows the road more in the beginning and end of the path. This is a possible set of parameters for *Move Cautiously* if we want this context to value cover significantly more than time and accessibility. The parameter set could also be considered for the *Attack* context,



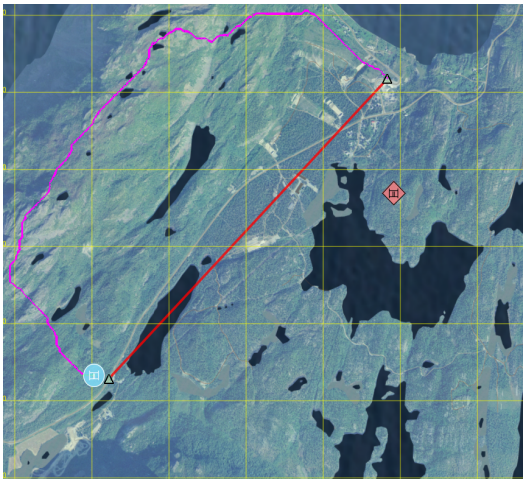
(a) Input parameters: $\{6, 7, 4, 1, 1\}$



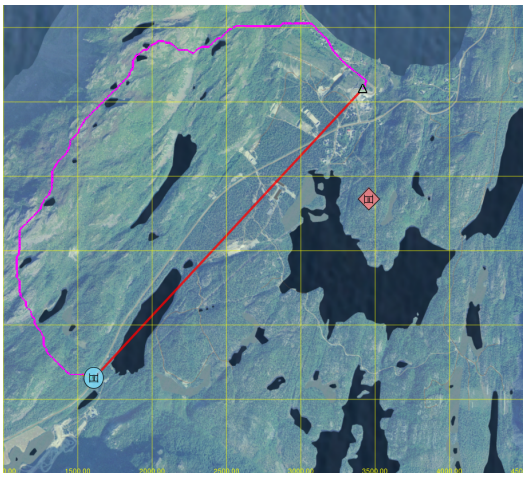
(b) Input parameters: $\{6, 7, 4, 5, 1\}$



(c) Input parameters: $\{6, 9, 4, 5, 1\}$



(d) Input parameters: $\{1, 9, 3, 1, 5\}$



(e) Input parameters: $\{1, 9, 3, 6, 5\}$

Figure 6.1 Resulting paths for different sets of input parameters $\{time, cover, concealment, accessibility, threat\}$.

depending on the type of attack this context represents. The priority of the time aspect could be lower for some types of attack.

If we use a parameter set like the above examples also for the *Move* context, the parameters for cover, concealment and threat will not influence the path when no enemies are present. However, if we also want to use the *Move* context when moving through an area with enemies, these parameters will influence the path planning. If the intention is that the entity then should move quickly, only taking cover when it does not slow the entity significantly, a parameter set like $\{6, 7, 4, 1, 5\}$, which was used in 6.1b may be suitable.

6.2.2 Path planning for hidden movement through high threat areas

In some situations it is important to stay both out of sight and at a distance from enemies. This can be the case for example in *Move Cautiously*.Also for *Observation* and reconnaissance it can be advantageous to stay out of sight and to take detours in order to observe enemy activity along an axis.

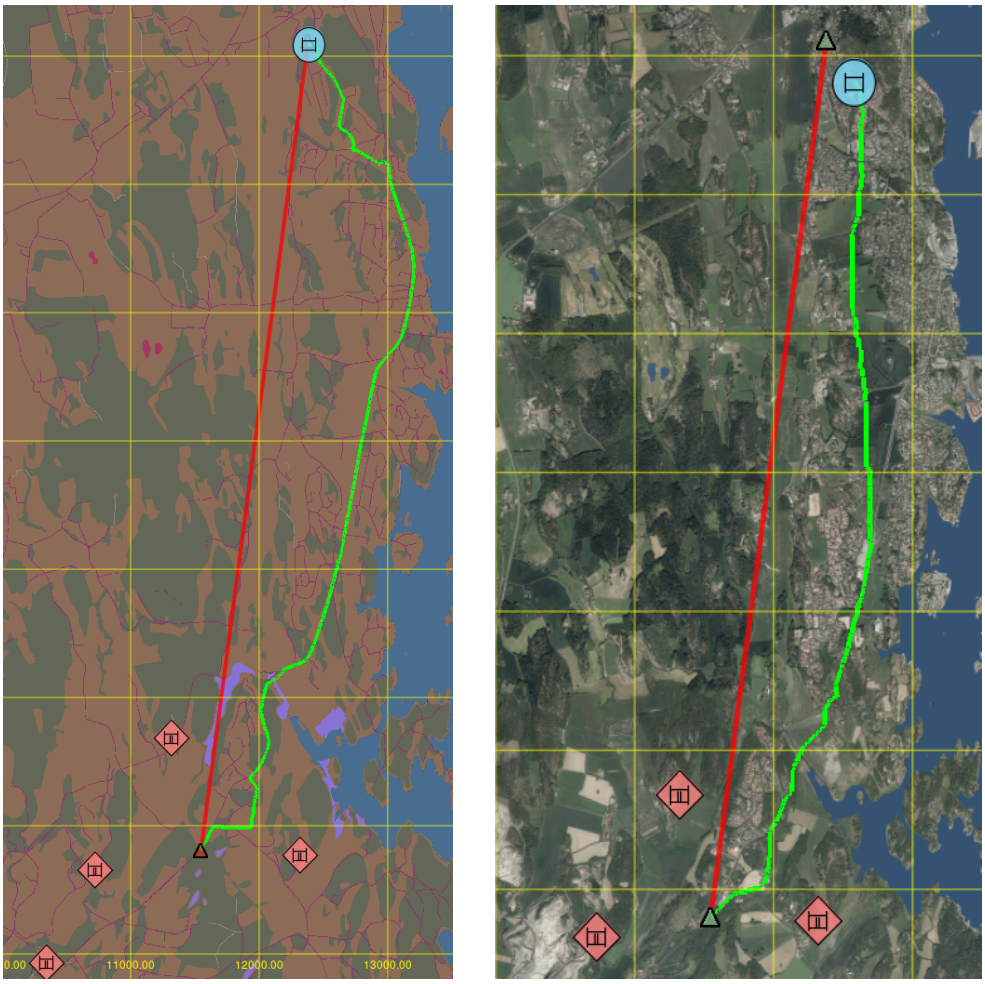
In such situations cover and threat may be prioritized, while time and accessibility may be given a low priority. An example of such a path is shown in figure 6.1d. This example path is created with the input parameters $\{time, cover, concealment, accessibility, threat\} = \{1, 9, 3, 5, 1\}$. We see that the path is practically identical to the path based on cover alone.

If we in addition prefer to move through accessible areas, we change the accessibility coefficient to 6, while keeping the rest of the parameters unchanged. The resulting path is shown in figure 6.1e. This path is very similar to the previous path, but it moves along the road the last part of the path. Such a mild preference of roads and accessible terrain will probably often be useful and seems better than to practically ignore the terrain types during path planning.

6.2.3 Situation dependent path planning over larger distances

The previous examples have shown a path planned between two points that are 2.5 km apart. The path planning algorithm can also be used to plan paths over larger distances. We will now show a few examples of planned paths with an approximate length of 7 km. Figure 6.2a shows the terrain types within the example area, while figure 6.2b shows a photograph of the area. The green areas in figure 6.2a are forest, while the orange areas denote grass. The blue areas are ocean, and the purple areas are swamps. The burgundery lines are paved roads. The areas representing houses are not assigned a specific terrain type, but are also considered grass in this terrain model.

The path in figure 6.2a is planned with priority on time and cover. It shows a path which mostly goes along roads, because of the priority of time. The choice of which roads to follow is based on the cover aspect, which can be suitable for the *Move* context. The path in figure 6.2b is planned with highest priority on cover and low threat, and we see that this path most of the time does not follow the roads. As instructed by the category coefficients, its emphasis is on cover, which makes the path suitable for the *Move Cautiously* context.



(a) $\{6, 7, 4, 1, 1\}$. (b) $\{1, 9, 3, 5, 1\}$.

Figure 6.2 Resulting paths for two different sets of input parameters $\{time, cover, concealment, accessibility, threat\}$.
The first figure shows the path in a map with the distribution of terrain types, while the second path is shown in a photography of the area.

When planning paths over large distances, we may want to vary the path planning settings depending on the distance to the target area. For example when moving cautiously over a long distance towards an attack position, we may want to use parameters like $\{6, 7, 4, 1, 1\}$, prioritizing time, cover, and concealment, when planning the first part of the path, and to reduce the priority of time when there is less than 3 km to the target area, using the parameters $\{1, 9, 3, 5, 1\}$ when planning the last part of the path. Furthermore, a third set of parameters may be chosen for the very last part of the path, when performing the actual attack. Such a variation of parameters can be done within one context based on for example remaining distance to the target area, or the parameters can be changed as a consequence of context switches during movement, or a combination of these.

7 Discussion and ideas for future work

In this report we have presented a method for situation dependent path planning. The concept is to first identify separate aspects that should be considered during path planning, then compute separate sets of weights for these aspects, and finally to combine these weights to a situation dependent set of weights based on the aspects that should be prioritized in a specific situation. We propose a set of five aspects and functions for weight computation for each of these aspects. The method is flexible, so more aspects can be added, and the functions for weight computation can be changed.

It is important that the aspects cover the important factors that should be taken into account when planning reasonable routes. We have chosen the five aspects time, cover, concealment, threat, and accessibility. These aspects cover both the need to move quickly, the necessity of being out of reach for enemy observation posts and weapons, and the need to move through areas that are suitable for movement and to avoid inaccessible areas. The weights are computed separately for each of these aspects based on input data for entity type, terrain and enemies. We have proposed functions for weight computation for each of these aspects, and have given ideas for alternative computations for some of them.

The function for situation dependent weights takes five integer category coefficients as input and constructs a new set of weights based on these coefficients and the weights for each aspect. The category coefficients define a relative priority of the aspects, such that the path is planned considering mainly the aspects with the highest coefficients. We have demonstrated our method for different sets of category coefficients, and have explained what situations these coefficients would be suitable for.

The example paths produced by the method show that the situation dependent path planning method clearly depends on the category coefficients, and that changing the priority of the categories changes the properties of the planned path. The five categories time, cover, concealment, threat and accessibility are terms that are familiar to military officers and are aspects that are considered in military planning, even though they are not prioritized as explicitly as done in our method. The example paths show types of movements that can be typical for situations in a battle, but the coefficients are not calibrated for specific tasks. However, they demonstrate that the method is able to produce routes for

different types of situations, and they show examples of how the planned paths are changed if the input parameters are altered.

An aspect that has been left for future work, is observation of enemy positions or enemy activities along an axis. To be able to observe an enemy while moving is important in many military situations. This aspect, however, is more difficult to compute. One could think that in order to observe, one should have line of sight to the enemy most of the time. But this solution would not give a route suitable for military entities, since the route will be visible from the enemy position as well. Therefore entities moving along such a route will be easy for an enemy to attack. Military units would rather move along a route which is mostly out of enemy line of sight, but from time to time the units will have the enemy within line of sight, preferably where the units are partially covered, but able to observe the enemy. An example of such a route would be along a ridge, but on the opposite side of an enemy, such that the units can observe the enemy over the hilltop, while moving out of enemy line of sight on the opposite side of the hill. Routes for observation will therefore be out of line of sight, but near the border between areas within line of sight and areas out of line of sight. This concept could also be included in the cover weights. Since the enemy entities may move in order to get a better overview of the area, it may often not be sufficient to only consider line of sight from the exact enemy position in order to ensure cover.

We will also consider to make more advanced threat weights, for example by allowing higher threat values when closing in on an enemy in order to attack, or proportionally to the distance from the start node. This is analogous to a situation where the entities want to stay safe and hidden as long as possible before becoming visible when attacking the enemy. Another possibility is to instead use a dynamic change of category coefficients depending on the distance to the enemy.

The proposed weight implementations for each aspect and the combination function should be considered demonstrations of a concept, showing that situation dependent path planning is possible in simulations, with relatively few and simple input parameters. It is possible to add more aspects, improve the computations of aspect weights, and to adjust the parameters to specific situations. As mentioned in the end of the previous section, it is also possible to use different sets of parameters when planning different parts of the path, either as a function of remaining distance, because of context switches, or both.

The path planning method presented in this report is ready to be used and can be modified as needed. It is significantly better than the alternatives to move in a straight line between the start and end node, or to move along the shortest path. The method is also easy to implement, it can be used with any A* implementation, and it is easy to adjust and improve. More interesting, it is a model for how situation dependent path planning can be achieved in general.

8 Conclusion

The concept presented in this report can be used to develop situation dependent path planning methods for several types of entities, both autonomous vehicles and simulated entities. The idea is to change

the edge weights in a standard graph for path planning, such that an optimal path through the graph is optimal with respect to a combination of several aspects of a situation. The path planning is adjusted to a specific situation simply by changing the priority of the aspects, which are given by a few integer values. It is therefore easy to use the method for path planning in different situations, without needing to change the implementation. Also, a standard graph based path planning implementation can be used as a basis for the method, only the computation of edge weights need to be changed.

We have demonstrated the concept by implementing a path planning method for MBTs as an example. The result is a path planning method able to produce paths with widely different properties suitable for different situations. The descriptions in this report aim to enable the reader to develop a suitable method for his own application.

References

[1] R. A. Løvlid *et al.*, "Modelling battle command with context-based reasoning," Forsvarets forskningsinstitutt, FFI-rapport 2013/00861, 2013.

[2] N. Sedaghat, "Mobile robot path planning by new structured multi-objective genetic algorithm," in *International Conference of Soft Computing and Pattern Recognition*. IEEE, 2011, pp. 79–83.

[3] T. Oral, "A multi-objective incremental path planning algorithm for mobile agents," in *International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2. IEEE, 2012, pp. 401–408.

[4] F. Guo, H. Wang, and Y. Tian, "Multi-objective path planning for unrestricted mobile," in *International Conference on Automation and Logistics*. IEEE, 2009, pp. 1046–1051.

[5] S. Bruvoll, "Preliminary study of terrain analysis and path planning for computer generated forces," Forsvarets forskningsinstitutt, FFI-notat 2013/00688, 2013.

[6] A. Alstad *et al.*, "Autonomous simulation of a battalion operation - seamless integration of command and control and simulation for planning and training," Forsvarets forskningsinstitutt, FFI-rapport 2013/01547, 2013.

[7] VT MÄK VR-Forces. [Online]. Available: www.mak.com/products/simulate/computer-generated-forces.html

[8] VT MÄK Technologies, *B-HAVE Users Guide*.

[9] Autodesk Kynapse. [Online]. Available: www.gameware.autodesk.com/kynapse

[10] VT MÄK Technologies, *MÄK Terrain Database Tool*.

[11] J. Borgesen, "Konfigurasjon og utvidelse av simuleringsrammeverket VR-Forces," Forsvarets forskningsinstitutt, FFI/NOTAT-2007/01939, 2007.

[12] J. H. Wiik and S. Bruvoll, "Methods for a customized path planning algorithm for use in military simulations," Forsvarets forskningsinstitutt, FFI/NOTAT-2013/02656, 2013.

Abbreviations

B-HAVE	Brains for Human Activites in Virtual Environments
CGF	Computer Generated Forces
CxBR	Context-Based Reasoning
GDB	MÄK Terrain Format
MBT	Main Battle Tank
SME	Subject Matter Expert
VR-Forces	Virtual Reality Forces