FFI-RAPPORT

19/00229

An automatic route planning service for unmanned surface vehicles

Marius Thoresen
Solveig Bruvoll
Martin Syre Wiig

An automatic route planning service for unmanned surface vehicles

Marius Thoresen
Solveig Bruvoll
Martin Syre Wiig

Norwegian Defence Research Establishment (FFI) 8 May 2019

Keywords
Ruteplanlegging
Autonome overflatefarkoster
Autonomi

FFI-rapport
19/00229

Project number
1372

ISBN
P: 978-82-464-3170-3
E: 978-82-464-3171-0

Approvers
André Pettersen, *Director of Research*

The document is electronically approved and therefore has no handwritten signature.

(U) Summary

The Norwegian Defence Research Establishment (FFI) has developed an autonomous unmanned surface vehicle (USV), Odin, which is tasked with performing surveys and mine countermeasures missions. It is meant to navigate autonomously close to shore, and must do so in a safe manner. To do so, it requires safe and efficient routes that it can follow, and thus needs a service to provide it with such routes. In this report, we present an automatic route planning service which provides routes that are suited for route following by the Odin USV.

The service is implemented in a common framework for route planning based on open source tools and open standards. The same framework is also used for route planning for other platforms at FFI. Route planning requests are made through a web interface which implements the WPS (Web processing service) standard, which makes the service easy to integrate in a variety of applications. The planning service uses a graph based route planning method, where a graph structure is created offline based on S-57 Electronic Navigation Charts. This graph is then used for a graph-search resulting in a coarse route. This route is further refined with post processing to obtain a route suitable for direct path following by an USV.

The route planner has been used in live testing for Odin, and it has been demonstrated that the route planner is capable of providing suitable routes for littoral operations for an USV.

(U) Sammendrag

Forsvarets forskningsinstitutt (FFI) har utviklet en autonom ubemannet overflatefarkost (USV), Odin, som skal utføre minemottiltaksoperasjoner og kartleggingsoppdrag. Den skal navigere autonomt nær kysten og må gjøre det på en sikker måte. For å gjennomføre dette trenger Odin sikre og effektive ruter den kan følge, og den må derfor ha en tjeneste som genererer slike ruter. I denne rapporten presenterer vi en automatisk ruteplanleggingstjeneste som leverer ruter som er tilpasset rutefølging for USV-en Odin.

Tjenesten er implementert i et felles rammeverk for ruteplanlegging basert på åpen kildekode-verktøy og åpne standarder. Dette rammeverket er også brukt for ruteplanleggingstjenester for andre ubemannede plattformer ved FFI. Forespørsler om ruter gjøres gjennom en webtjeneste som implementerer WPS-standarden (Web Processing Service), noe som gjør tjenesten enkel å integrere i mange ulike applikasjoner. Planleggingstjenesten bruker en grafbasert ruteplanleggingsmetode hvor en grafstruktur lages offline basert på S-57 elektroniske sjøkart. Denne grafen brukes til å gjøre grafsøk som gir en grov rute. Denne ruten blir så forbedret i en etterprosessering for å få en rute som er tilpasset direkte rutefølging for en USV.

Ruteplanleggeren har blitt brukt under eksperimentelle kjøringer med Odin, og det har blitt vist at ruteplanleggeren kan lage passende ruter til en USV for kystnære operasjoner.

Contents

1 Introduction

The technological development of autonomous vehicles is moving rapidly, and autonomous units in the air, on land and at sea are becoming capable of performing increasingly complex tasks. At sea, unmanned surface vehicles (USVs) are a type of autonomous vehicles that has many potential uses for both civilian and military applications, and will soon be ready for fully autonomous operations. The types of missions that USVs will perform ranges from transportation of goods and people, to unmanned surveying and mine hunting. An important task in most missions is the transit between two locations, and for this a route suited for the given mission is required. In this report we present an automatic route planning service with the main goal of providing safe and efficient routes for USVs.

The route planning service presented here is based on a framework for automatic route planning for several types of autonomous vehicles [1]. The route planning service uses a web interface for route requests, while several modules running in the background does the route processing. The processing is based on route search in a mathematical graph structure that is made in advance. This makes it possible to include a wide variety of data types such as depth maps, weather data and terrain models for situation dependent route requests. Different aspects of the route planning can also be prioritized depending on the given mission and situation.

In this report, the development of a route planning service, and how it is fitted to the common route planning framework of [1] is described. The first part of this report describes the requirements and needs for a route planner for a USV. We specifically look at the Odin USV, which is developed by the Norwegian Defence Research Establishment (FFI). The second part describes how a graph is generated for efficient route planning for USVs, and the attributes it provides for the route planning. The third part describes how the route planning itself is performed with a graph search, and how the routes are post processed for efficient routes that is suitable for following by a USV. Lastly, a description of how the route planner can be, and has been, used for route planning for Odin.

2 The Odin USV

FFI is developing an autonomous USV, Odin [2], shown in Figure 2.1. Odin is a research platform to develop autonomy for performing surveys, executing mine countermeasures operations, and conducting surveillance. Transit between locations is required for all these purposes, and for a high level of autonomy, an automatic route planner is needed to provide routes for these transit. Odin is a small vessel designed to operate close to shore, which means Odin must be capable of performing all its missions autonomously close to shore as well. Navigation close to shore is a complex task, and it puts high demands for safety for a route planner, without it being a limiting factor for efficient operations.

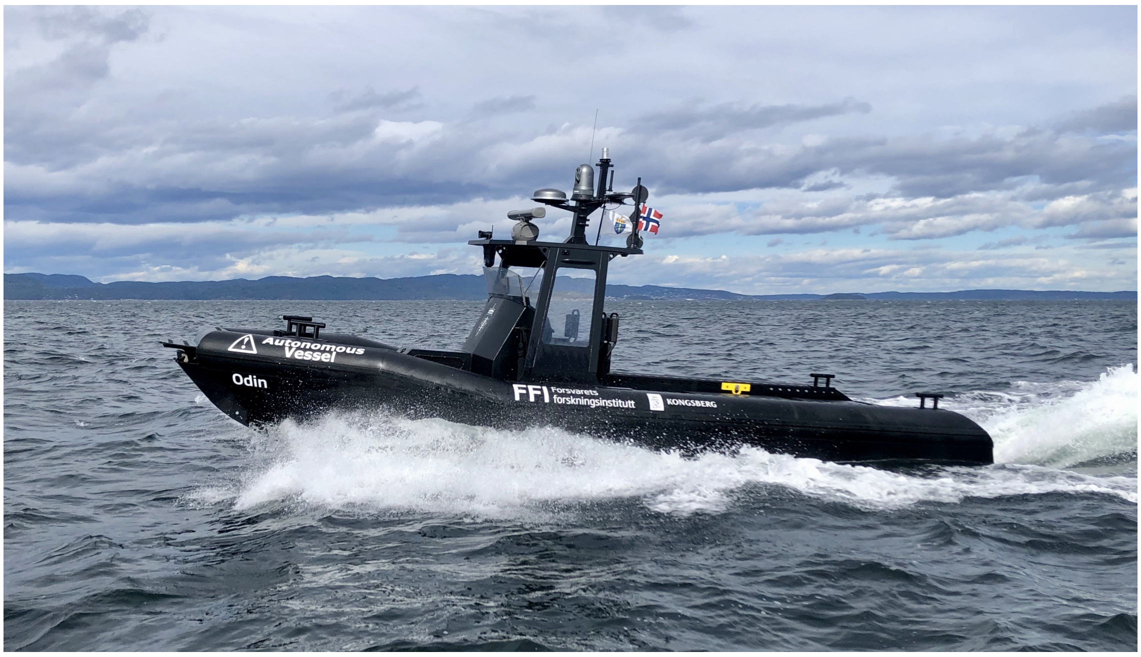
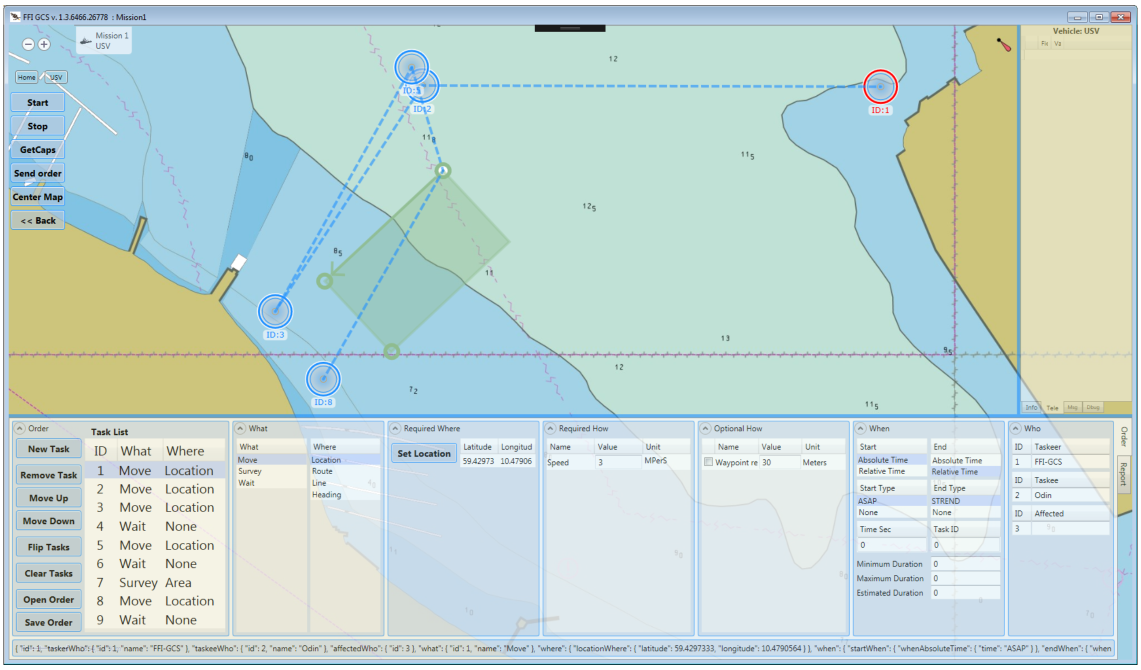


Figure 2.1 The autonomous Odin USV

2.1 Capabilities of Odin

The typical missions for Odin consist of a combination of surveying an area, patrolling along a predetermined route and simple transit to points. The missions are planned through the FFI Ground Control Station (GCS) [3], and uploaded to Odin. A mission can consist of several different types of tasks. For simple "move to location" tasks, Odin follows a route. The route is either created manually by an operator who places a sequence of waypoints, or by an automatic route planner. An operator can also use a route from an automatic route planner for decision support or modify the route before using it to create a mission. For the survey tasks, the planning is handled by the autonomy module HAL [4]. In Figure 2.2, an example of a mission planned in the control station is shown.



Task List

ID	What	Where
1	Move	Location
2	Move	Location
3	Move	Location
4	Wait	None
5	Move	Location
6	Wait	None
7	Survey	Area
8	Move	Location
9	Wait	None

What

What	Where
Move	Location
Survey	Route
Wait	Line
	Heading

Required Where

Latitude	Longitud
59.42973	10.47906

Required How

Name	Value	Unit
Speed	3	MPerS

Optional How

	Name	Value	Unit
☐	Waypoint re	30	Meters

When

Start	End
Absolute Time	Absolute Time
Relative Time	Relative Time
Start Type	End Type
ASAP	STREND
None	None
Time Sec	Task ID
0	0

Minimum Duration	0
Maximum Duration	0
Estimated Duration	0

Who

ID	Taskeer
1	FFI-GCS

ID	Taskee
2	Odin

ID	Affected
3	

{ "id": 1, "taskerWho": { "id": 1, "name": "FFI-GCS" }, "taskeeWho": { "id": 2, "name": "Odin" }, "affectedWho": { "id": 3 }, "what": { "id": 1, "name": "Move" }, "where": { "locationWhere": { "latitude": 59.4297333, "longitude": 10.4790564 } }, "when": { "startWhen": { "whenAbsoluteTime": { "time": "ASAP" } }, "endWhen": { "when

Figure 2.2 FFI GCS showing a planned mission for Odin, with planned way points and a survey area in green

2.2 Route planning for Odin

Automatic route planning is not always necessary for conducting missions autonomously. In Figure 2.2, a manually planned mission can be seen. In short missions such as this one, especially when there are few obstacles in the mission area, manually placing all waypoints is acceptable for an GCS operator. When the missions are longer, cover a larger area and takes place in more complex surroundings, manually placing every single waypoint becomes a tedious task. Both because the number of waypoints needed goes up drastically, and because it is harder to manually check that every leg is completely safe when the leg distance goes up. Route planning can also involve factors such as planning routes with limited communication coverage, routes that avoid detection or routes that has a line of sight to a location. A benefit of automatic route planning is that it can keep track of these factors, and plan routes accordingly in ways that an operator can't [5]. If the information about the world is updated during a mission, a route planner can accommodate these changes and plan updated routes based on the new information.

A route planner can also be useful as a support tool for an operator. A mission specification can be specified and sent to the route planner, and the calculated route is returned to the operator who can choose to use the route, modify individual waypoints or discard it entirely. Another benefit of using a route planner as a support tool is that meta-data about the route can be returned to the operator. This can be expected duration, distance, weather conditions along the route and other factors relevant to the mission.

The auto pilot of Odin is only capable of following a route that is a sequence of straight legs, but it handles the transition between legs itself [6]. This means that the route planner should also provide route consisting of long straight legs with no curves.

The graph-based route planning described in [1] is used as a starting point for a route planning service. This is efficient for exploring the available search space and obtaining the global optimal route. However, the route obtained will not be optimized for long straight edges, and to achieve this post processing of the routes is required. We describe this post processing procedure in Chapter 4.

3 Graph generation

The first step of performing graph based route planning is to create a graph. This graph can be created offline, and later used to perform route planning requests online. The goal of creating a graph to represent the world is that the graph is detailed enough to perform complete route planning requests without keeping the original data. To achieve this, the graph must contain all the necessary attributes in either the nodes or the edges. The attributes that are necessary depends on what criteria are used in the graph search and post-processing.

The graph is created using an S-57 Electronic navigation chart [7]. This is a vector map that contain land areas, areas with guaranteed minimum depth, rocks, bridge heights and other information relevant for navigation. When creating a graph for an area, nodes are placed everywhere except on land. Then the nodes are connected to their neighbors to form a mesh. There are two main ways of placing the nodes: structured like in a regular grid or with probabilistic placement. In our case, the nodes are placed randomly. In areas close to land or shallow areas the nodes are sampled with a random uniform distribution with a high sampling density. Further from land, the nodes are also sampled with a uniform distribution but with lower node density. This is because it's more critical with precise navigation close to land. Further from land, this is not as important, and the node density can be decreased in order to improve search speed for route planning requests. Typical node placement can be seen in Figure 3.1a.

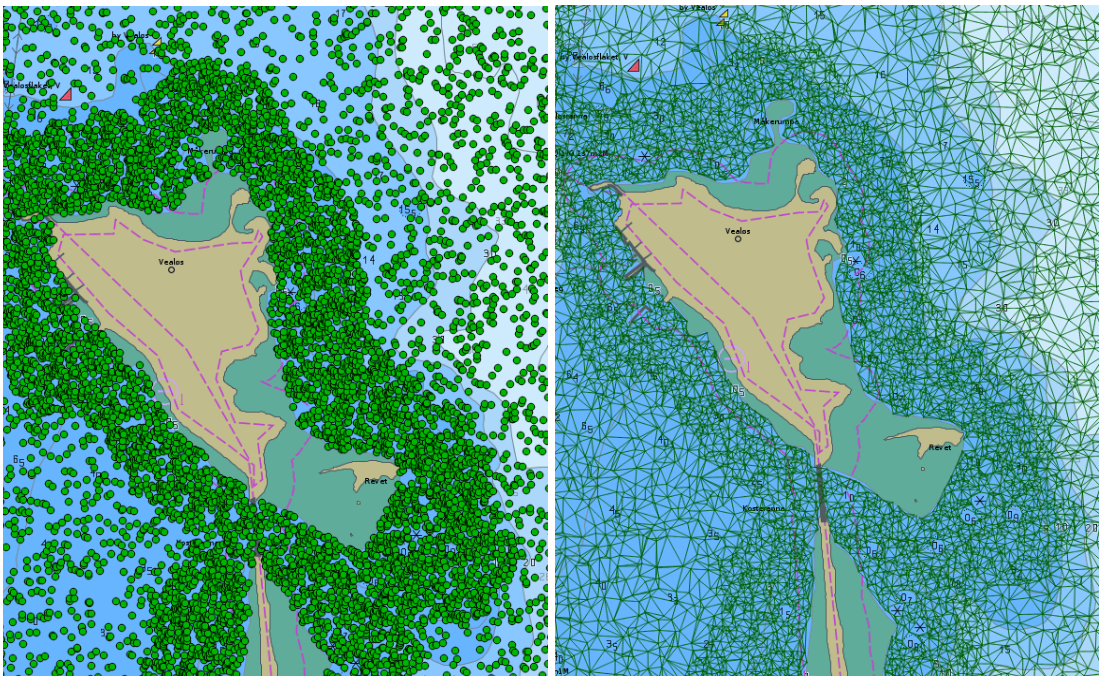

(a) Random node placement *(b) Mesh generated using Delaunay triangulation.*

Figure 3.1 Graph generation process with node placement and edge creation

With the random node placement, a mesh can be generated using a process called Delaunay triangulation [8]. This process connects each of the nodes to its closest neighbors, and these connections makes up the edges of the graph. To perform the actual graph creation, a MATLAB library developed at FFI has been used [9].

The uniform distribution of nodes means there will be local clustering of nodes. When a node density is selected for the node placement, this is based on how detailed the area should be sampled and the desired distance between nodes. With the local clustering of nodes, some of the nodes close to each other are essentially left redundant, and contribute little to the overall sampling of the area. More nodes means longer processing time for the route planning, and redundant nodes should ideally be removed. In the future, this could be avoided by using another sampling method. A more suitable option for distributing nodes evenly would be the Poisson-disk distribution [10] which guarantees a minimum distance between nodes. Efficient sampling based on this distribution is for example implemented with Bridson's algorithm [11].

3.1 Water types

For a first implementation of the route planner for Odin, providing safe navigation in the area near Horten is the goal. This was achieved with a simple cost function where what we have defined as *water types* is the main attribute needed for cost calculation. The water types reflect how safe an area is. For example being close to land should not be a problem for an USV, but it exposes the USV to hazards such as leaving limited maneuvering space during interaction with other vessels. Another risk is that errors in the navigation can have consequences such as running aground or colliding with land. In shallow waters, the risk of running aground is increased due to potential depth mapping errors, but the overall hazard can be considered lower than areas close to land. With that in mind, the following categories have been defined for the graph generation:

Land	Less than 2 m deep. Avoid completely
Near land	Closer than 50 m from $land$-category. High hazard level
Shallow	Between 2 m and 5 m deep. Moderate hazard level
Open water	Deep water and far from land. No hazard

The water types are tailored to Odin, but different depth and distance limits can be selected for other vessels, or entirely new categories as well. Here, each node sampled is assigned to one of the four water types. The water types are the basis for the edge weights which are needed in the route planning algorithm. An illustration of the different water types can be seen in Figure 3.2, where three different water types are represented. In the graph creation, the nodes in the land category are removed entirely from the graph. The actual edge weights are calculated dynamically for each query in order to handle information not available at the time of the graph creation, such as updated depth values. In addition to adding water types as attributes of the graph, the distance from each of the nodes to the closest of the other categories are included. This is used by the smoothing algorithm part of the route planning. Even though the nodes in the land category are removed, *distance to land* still exists as an attribute for the remaining nodes.

3.2 Clustering and cost function

In order to make the graph search more efficient, an hierarchical approach to graph creation is used. Groups of neighboring nodes are clustered and connected together. An example of the placement of

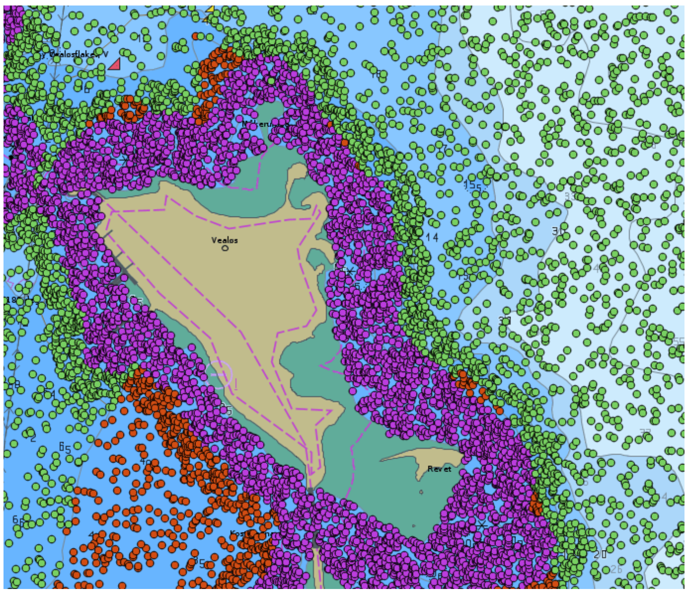


*Figure 3.2 Different water types. purple=**Near Land**, red=**Shallow**, green=**Open water***

the clusters and their connections can be seen in Figure 3.3. When a graph search is performed, a search is first done in the clusters to find which clusters the route will pass through. Then, using the resulting set of clusters, a graph search is made on the subset of nodes that is contained by the resulting clusters. This improves the overall search speed, but the resulting routes might not be optimal if the wrong clusters were selected in the first search [12]. With the area used in this particular route planning setup at Horten, the graph size is small enough that performing route planning requests on the full graph is feasible. The use of a hierarchical structure is therefore not strictly necessary, but has been included to allow for scaling of the route planner in the future.

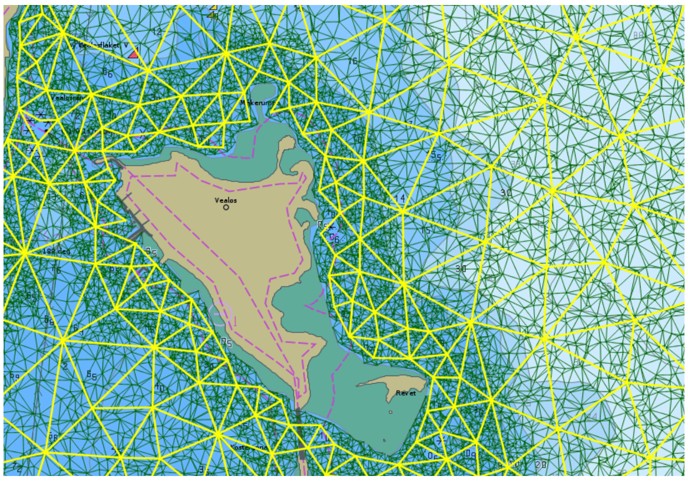


Figure 3.3 The placement of the clusters and their connections to neighbors. The intersections of the yellow lines are representing nodes in the clusters, and the yellow lines are the edges between the clusters.

4 Route planning

The route planning is performed using PostGIS and its routing extension pgRouting. This is a two-step process: First a hierarchical graph search is performed to obtain a coarse route. Secondly the coarse route is then refined with a simplification step, giving an efficient route suitable for path following by an USV.

4.1 Shortest path in the graph

As the graph generation has created a full graph, a coarse route is found by a regular shortest path algorithm. In our case, Dijkstra's algorithm [13] is used for this, as it is guaranteed to find the optimal path in the graph. The commonly used algorithm A* [14] could also be used for improved search speed. A* uses an heuristic to estimate the remaining cost to the goal. If this heuristic always underestimates the cost to the goal, A* will also be optimal. In our case, Dijkstra's algorithm is considered fast enough, and A* is not used.

4.1.1 Use of pgRouting

The graph is implemented in PostGIS database as described in [1]. meta data of minimum water depth, water type and distance to the water types are added as fields for the nodes and edges. For the graph search itself, the `pgr_dijkstra()` function of the pgRouting library is used. The function is first used on the the cluster graph to find the clusters that most likely contain the optimal path. The algorithm is then called on the subset of nodes contained within the clusters of the coarse route to find the optimal path among these nodes and then on the graph itself to find the optimal path within the clusters.

The specified start and goal positions in the request will never correspond precisely to nodes in the graph. Therefore, new temporary nodes are inserted into the graph at the start and goal points, and edges are created between the new points and their neighbors in the graph. These new nodes are then used as the start and goal nodes in the planning algorithm.

4.1.2 Cost function

The cost function determines what will be considered the best route. If any special properties is desired from a route, changing the cost function is the main way to achieve the desired behavior. This can for example be areas that are preferred or should be avoided. Factors that influence the cost can be communications coverage or areas with cover from the open ocean. As long as data to calculate costs is available for the graph there are many options for the selected cost function, and this can be changed for each route request.

In the first implementation of the route planner for Odin, safe navigation is the priority. The cost function is therefore designed so Odin stay in the safer water types as much as possible. It is based on the water types defined in the graph creation, and this is available as an attribute of the nodes. An intuitive way of thinking about the weight function is to use the traverse time for each edge as the weight. The distance d is given directly by each edge, and the speed v can be used as a parameter. In that case the weight w (i.e. traverse time) is given by the following equation:

$$w = \frac{d}{v}$$

This method has a physical interpretation in that the maximum speed selected can be set to the maximum speed of the USV, and this will represent the best case in the graph. By reducing the speed, the cost is increased as if the USV reduced speed accordingly. In our case, the selected speed values are given in table 4.1. With the given values, a edge with unit length in shallow water will have a cost of 100, and an edge in open water with the same cost would have a length of 1000.

	Speed [m/s]	**Cost**
Open water	10	$0.1 \cdot d$
Shallow	0.01	$100 \cdot d$
Near land	0.00001	$10^5 \cdot d$
Land	10^{-9}	$10^9 \cdot d$

Table 4.1 Selected speed in different water categories, and corresponding costs

As the speed reduction moving from one category to the next is large, the planner will behave almost as if it is not allowed any movement in an area with higher hazard level than it is currently in. This is not a particularly sophisticated cost function, but the relative weights assigned to each category can be easily changed, and the cost function itself can be substituted with an entirely different one. It also demonstrates that a dynamic cost calculation is possible if desired.

4.1.3 Basic route planning examples

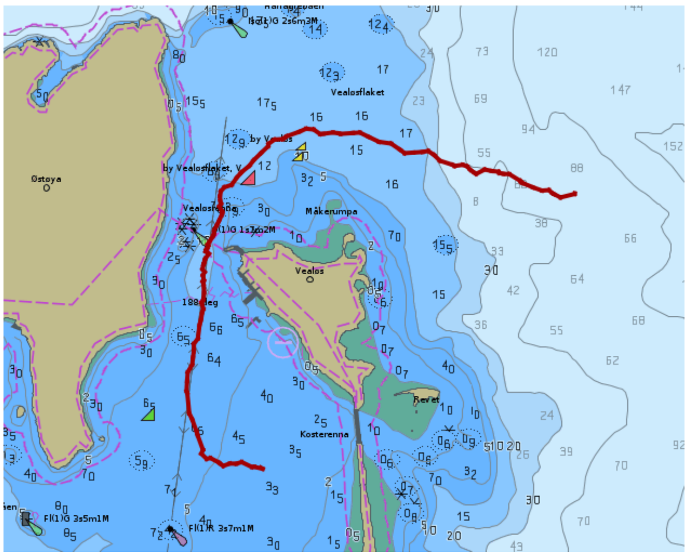


Figure 4.1 Basic planning examples

With the method described here, simple route requests can be made. An example of such a route can be seen in Figure 4.1. The route is jagged, and does in no way look like a route that a vessel would ever follow. This jaggedness can simply be attributed to how the graph is made with random node placement. The route does, however, fulfill the requirements of the routes being safe, and it doesn't move into areas of higher hazard levels than necessary. Disregarding the local jaggedness of the route, the overall shape of the route seems reasonable. The behavior seen in the example is also typical of the routes the graph search produces. In Figure 4.2, the effect of the cost function is reflected in the calculated route. The area with the red nodes is of type *shallow* and has higher cost than the area with the the green nodes, which is of type *open water*. This is reflected in the route, where a the calculated route minimizes the distance in the *shallow* area at the expense of a longer distance in *open water*.

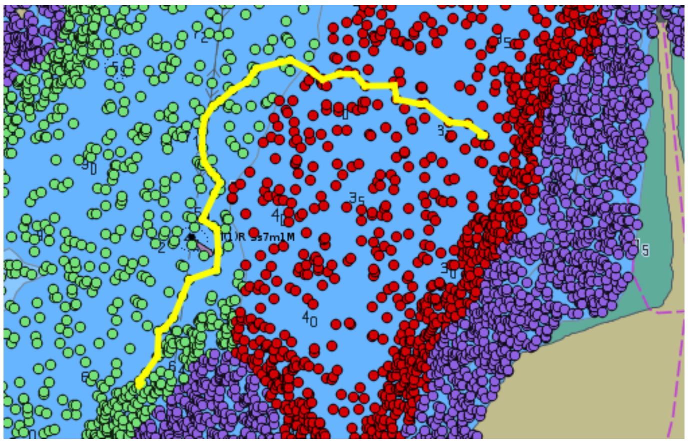


Figure 4.2 Effect of cost function for route planning

4.2 Route simplification

As mentioned earlier, the Odin's autopilot is only capable of following straight line segments [6]. This means the jagged route obtained by the graph search must be post processed so that the number of waypoints are significantly reduced, while trying to maintain the optimal cost properties of the original. This can be achieved by performing a line simplification, where redundant waypoints are removed until only the key waypoints remain.

4.2.1 Methods for line simplification

The standard method for line simplification is the Ramer-Douglas-Peucker algorithm [15]. This method removes what it considers "insignificant points", and aims to conserve the overall shape of the line. This method is also implemented in PostGIS and is easily available. This approach has been used for graph based route planning at sea [16]. However, by closer inspection of the algorithm, it becomes clear that this is not suited for our uses. Maintaining the shape of the route is actually not important for the end result, as the randomness of the graph will cause random and possibly large deviations from a true optimal route. Such deviations should ideally be removed in the post processing. We therefore want an algorithm that exploits the input route and cuts away from this as much as possible when the waters allow it, and it leads to a reduced cost.

With a more complex cost function, the shape of the route can be important to preserve. Local variations of the cost can result in seemingly strange shapes of the route which should be preserved due to the lower cost. In these cases, using the Ramer-Douglas-Peucker can be a good choice for a smoother route that preserves the optimal cost properties of the route.

4.2.2 A new method for situational aware route simplification

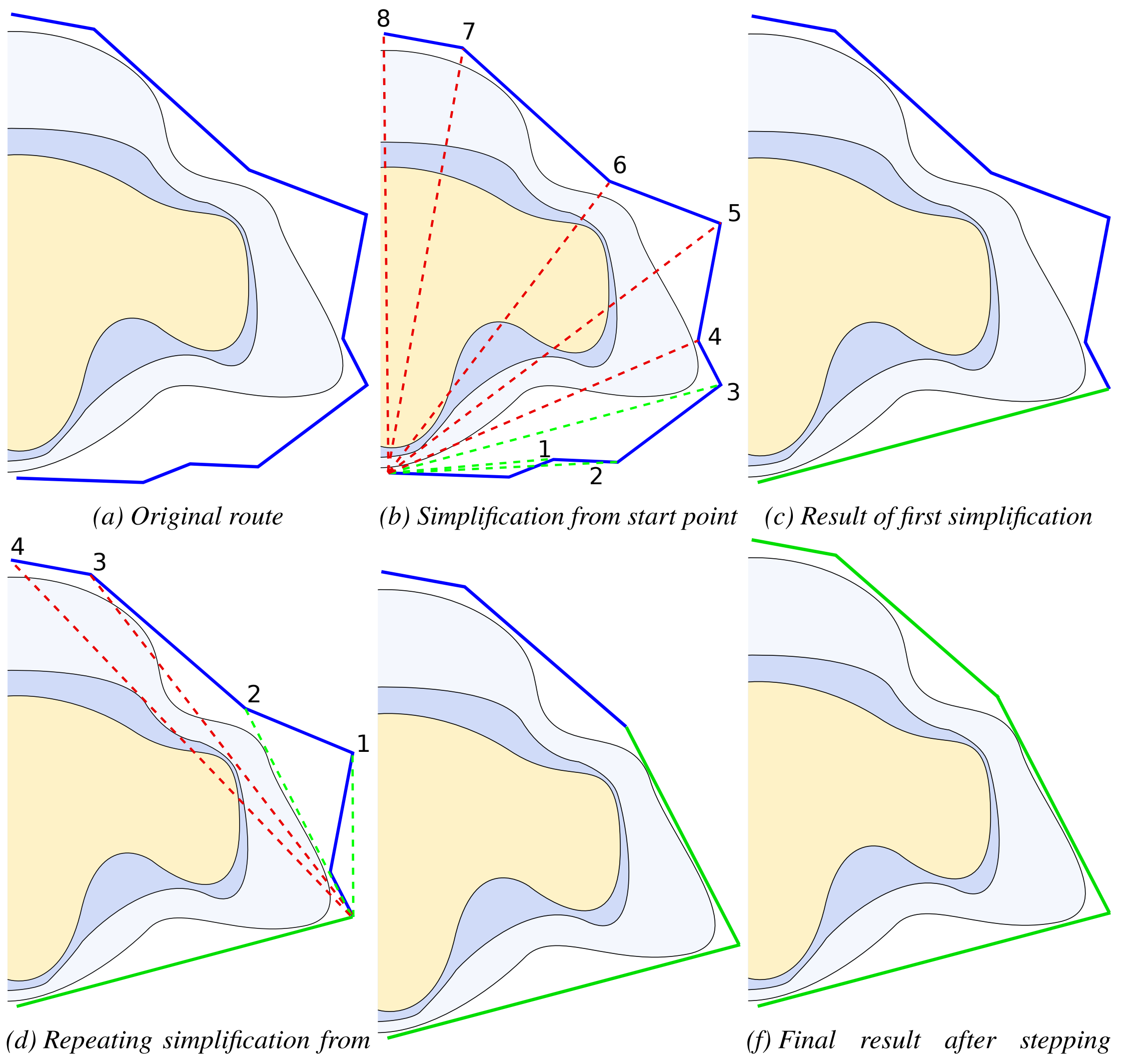
An improved algorithm for route simplification has been developed. The main idea is to start at the beginning of the coarse route, iterate forward, and remove any unnecessary points. This is achieved by creating new lines from a start point to new points on the route, and checking if these lines intersect areas with water types of a higher hazard level. It checks all possible lines forward to the end point, and selects the last valid one.

Pseudo-code for the algorithm is written in Algorithm 1. This also uses two sub-routines which can be found in the appendix.

An illustration of how the algorithm works can be seen in Figure 4.3. A coarse route is given in Figure 4.3a. In Figure 4.3b, lines are created from the starting point to all points in the route. Point 1, 2 and 3 are valid, but point 4 to 8 are not valid as the lines cuts across shallow areas or land. As line 3 was the last valid line, it is added to the simplified route as seen in Figure 4.3c. In Figure 4.3d the process is repeated from the new start point. Here, 1 and 2 are valid, while points 3 and 4 are not. Thus point 2 is selected in Figure 4.3e. No further simplifications can be made forward from this point, and the final result is shown in Figure 4.3f.

Algorithm 1 Simplify route

```
Input:  route
  current_waypoint ← route[0]
  simplified_route ← [current_waypoint]
  while current_waypoint ≠ route[end] do
    current_waypoint ← find_max_simplification(route, current_waypoint)
    simplified_route.push_back(current_waypoint)
  end while
```



(a) Original route *(b) Simplification from start point* *(c) Result of first simplification*

(d) Repeating simplification from new point *(e) Result of second simplification* *(f) Final result after stepping through route*

Figure 4.3 Route simplification of a coarse route. The new route is not allowed to cross into the shallow region near land

The simplification algorithm has been implemented in the USV route planner, and is used to post process the routes obtained from the graph search. In Figure 4.4, the simplification algorithm has been applied to a coarse route. Here, the original route is followed closely where safety margins are tight, but it is still able to take significant shortcuts elsewhere.

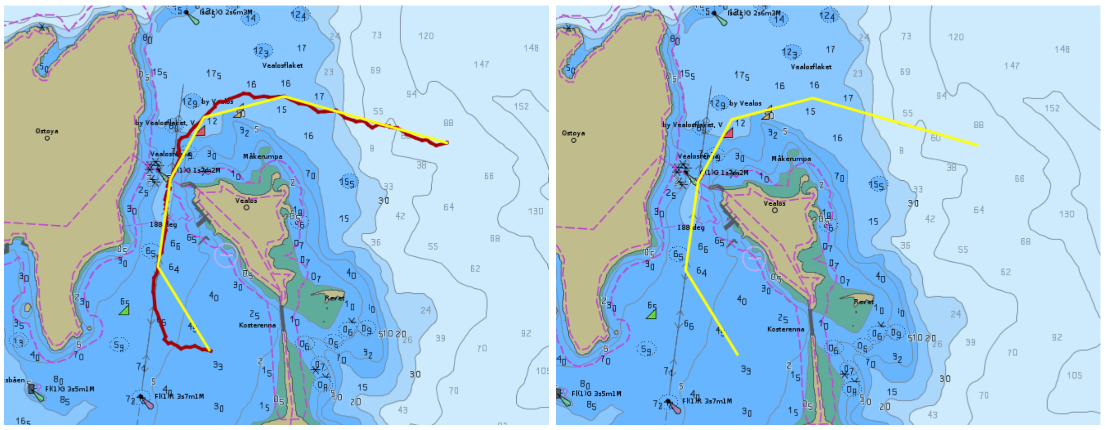

(a) Original and simplified route (b) Only simplified route

Figure 4.4 Route simplification of a coarse route

5 Using the route planner

The route planner presented here has been fully implemented, both aboard the Odin USV and in the USV control station. It is capable of providing routes for navigation in waters close to shore. In the current version, it supports an area around Horten, as this is the home port of the USV Odin. However, any desired area can be included by creating graphs according to the process described in Chapter 3 and adding them to the database.

The route planner has been tested in a real-life experiment where Odin navigated from just outside its home port at FFI in Horten, past the island Vealøs, and moving south outside Vealøs following the coast. The route obtained was close to the one shown in Figure 5.1. The current state of the route planner is that it is ready for use and experimentation by Odin.

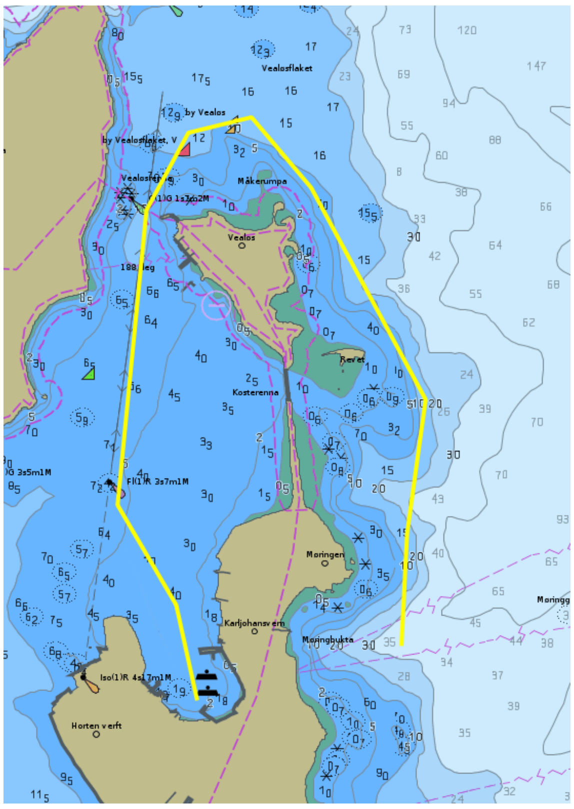


Figure 5.1 A longer route from Odin's home port

5.1 Interfaces

The main interface of the route planner is a web interface. The framework used is the ZOO-project [17], an open source project which implements the Web Processing Service (WPS) standard by the Open Geospatial Consortium (OGC). This makes the route planner accessible for a variety of tools such as web browsers, GIS software and any programs and programming languages that incorporates web request. The WPS interface is the same interface used by the route planning services for the other vehicles developed at FFI [1].

5.1.1 FFI Ground Control Station

The ground control station that is used for controlling Odin has support for calling the route planner. This allows an operator to request a route directly from the route planner, obtain route which can be reviewed before it is sent to Odin. This is especially useful before the route planner has been thoroughly tested, and it is verified that it behaves as desired. After a route is obtained, each waypoint can also be manually moved by the operator before the route is sent to the vehicle.

5.1.2 ROS interface

A ROS interface for the route planner is also implemented. With a ROS interface, the autonomy module HAL [4] on Odin can call route planner directly. This way, the operator will not be required to manually accept the route before it is started, but the process can be handled by HAL directly. Furthermore, this allows HAL to decide when it is required to call the route planner, and when an a priori planned route can be followed. This reduces the labor of the operator, and contributes to the overall autonomy and safety of Odin.

6 Conclusions and future work

In this report, an automatic route planner for USVs has been presented. The route planner includes a new method for post-processing routes so that they are suited for safe route following by an USV. The route planner satisfies the requirements of Odin, and is able to provide routes that are both safe, efficient and suitable for the path following capabilities of the vehicle. In off-line testing the route planner performs well with route requests in the vicinity of Horten. It has also been tested live in an experiment where Odin followed routes from the route planner. The route planner still needs more live testing in order to verify that the route planner does what is expected, and to get feedback on how well the routes work in a real life setting.

The route planning service is integrated in the Magellan route planning service, which also supports route planning for other autonomous vehicles [1]. The route planner supports an open standard for route requests, that has made the integration with both Odin and the GCS a simple process. The use of open standards can therefore be said to have provided a significant benefit, and we recommend that this track will be followed also in the future. The route planner has been implemented both on the USV control station, and on board the USV itself. Thus, it can be used to ease the workload of an operator in the planning phase of an operation, and also as a part of the autonomy system of the USV. The latter ensures that the USV is capable of autonomously finding efficient routes ensuring that it can maintain vehicle safety while simultaneously achieving dynamic vehicle goals.

6.1 Future work

While the route planner is working great given the ENC data used in the graph, experience show that the environment can change. For example, the port area of Horten is in the process of land reclamation, meaning that areas that were previously water are now land masses. This change is not present in the ENC data or in the graph, which leads to routes being planned across the new land areas. The new coastline is however mapped by Odin during operation, but the new map is not currently used by the route planner to update the graph and graph weights, and hence the resulting routes. In the future, updating the graph based on in situ measurements and mappings would be a useful extension of the route planner. In practice, this could be achieved by updating the water types as well as distances for the existing nodes in the graph without changing the graph structure.

The route planner is suitable for extension to new tasks and missions if the need arises. By changing the cost function, new aspects can be prioritized such as avoiding rough weather. This does require that the necessary data is available at the time of planning. A new type of task that could be useful in the future is autonomous coast following, where an USV follows a coast line at a specified distance, or within a desired distance interval. This could for example be for mapping purposes. This will require a different type of route planning, but the framework used would still be suitable.

References

[1] S. Bruvoll *et al.*, ‘Systembeskrivelse av ruteplanleggingstjenesten magellan for autonome, bemannede og simulerte systemer’, Forvarets forskningsinstitutt, Kjeller, FFI-rapport 2019/00225, Mar. 2019.

[2] E.-L. M. Ruud and J. Sandrib, ‘Usv for future maritime mine counter measures’, Forvarets forskningsinstitutt, Kjeller, FFI-rapport 2019/00470, Apr. 2019.

[3] E. Skjervold, ‘Design og implementasjon av FFI GCS - kontrollstasjon for ubemannede systemer’, Forsvarets forskningsinstitutt, Kjeller, FFI-rapport 2015/01620, Feb. 2016.

[4] T. R. Krogstad, K. Mathiassen, E.-L. M. Ruud, R. A. Seehuus, A. S. Simonsen and M. S. Wiig, ‘Hal - a decisional autonomy module for unmanned systems’, Forvarets forskningsinstitutt, Kjeller, FFI-rapport 2019/00489, Apr. 2019.

[5] S. Bruvoll, ‘Situation dependent path planning for computer generated forces’, Forvarets forskningsinstitutt, Kjeller, FFI-rapport 2014/01222, Sep. 2014.

[6] M. S. Wiig, K. Y. Pettersen, E.-L. M. Ruud and T. R. Krogstad, ‘An integral line-of-sight guidance law with a speed-dependent lookahead distance’, in *Proc. IEEE European Control Conference*, Limassol, Cyprus, 2018, pp. 1269–1276.

[7] International Hydrographic Organization, ‘IHO transfer standard for digital hydrographic data’, International Hydrographic Bureau, Monaco, Special publication No. 57, Nov. 2000.

[8] B. Delaunay, ‘Sur la sphère vide’, *Bulletin de l’Académie des Sciences de l’URSS. Classe des sciences mathématiques et naturelles*, no. 6, pp. 793–800, 1934.

[9] K. Landmark and E. Messel, ‘Tools for building path planning graphs, An annotated example’, Forsvarets forskningsinstitutt, Kjeller, FFI-rapport 2017/16168, Oct. 2017.

[10] R. L. Cook, ‘Stochastic sampling in computer graphics’, *ACM Transactions on Graphics*, vol. 5, no. 1, pp. 51–72, 1986.

[11] R. Bridson, ‘Fast poisson disk sampling in arbitrary dimensions’, in *ACM SIGGRAPH 2007 Sketches*, ser. SIGGRAPH ’07, New York, NY, USA: ACM, 2007.

[12] M. Thoresen, ‘Hierarchical path planning for ground vehicles’, Kjeller, FFI-rapport 2015/01608, Jun. 2016.

[13] E. W. Dijkstra, ‘A note on two problems in connexion with graphs.’, *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[14] P. E. Hart, N. J. Nilsson and B. Raphael, ‘A formal basis for the heuristic determination of minimum cost paths’, *IEEE Transactions on Systems Science and Cybernetics*, vol. SSC-4(2), pp. 100–107, 1968.

[15] D. Douglas and T. Peucker, ‘Algorithms for the reduction of the number of points required to represent a digitized line or its caricature’, *The Canadian Cartographer*, vol. 10, no. 2, pp. 112–122, 1973.

[16] V. Kvernelv and S. Bruvoll, ‘Ruteplanlegging og -følging for simulering av maritime operasjoner’, Forvarets forskningsinstitutt, Kjeller, FFI-rapport 2015/02231, Apr. 2016.

[17] G. Fenoy, N. Bozon and V. Raghavan, ‘Zoo-project: the open WPS platform’, English, *Applied Geomatics*, vol. 5, no. 1, pp. 19–24, 2013, ISSN: 1866-9298.

A Appendix

A.1 Algorithms

Sub-routines used by Algorithm 1

Algorithm 2 Find max simplification

```
Input: route
Input: start_node
  cleared_nodes ← []
  last_valid_node ← start_node
  for node ← start_waypoint to route[end] do
    line ← create_line(start_node, node)
    line_is_safe ← check_line(route, cleared_nodes, line)
    if line_is_safe then
      last_node ← node
    end if
    cleared_nodes.push_back(node)
    return last_valid_node
  end for
```

Algorithm 3 Check line

```
Input: route
Input: cleared_nodes
Input: line
  for all node in cleared_nodes do
    if node.type == SHALLOW then
      critical_distance ← min_distance_from_land
    else
      critical_distance ← min_distance_from_near_land
    end ifdistance_to_line ← distance_point_to_line(node, line)
    if distance_to_line < critical_distance then
      return false
    end if
  end for
  return true
```

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's MISSION

FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's VISION

FFI turns knowledge and ideas into an efficient defence.

FFI's CHARACTERISTICS

Creative, daring, broad-minded and responsible.

Om FFI

Forsvarets forskningsinstitutt ble etablert 11. april 1946. Instituttet er organisert som et forvaltningsorgan med særskilte fullmakter underlagt Forsvarsdepartementet.

FFIs FORMÅL

Forsvarets forskningsinstitutt er Forsvarets sentrale forskningsinstitusjon og har som formål å drive forskning og utvikling for Forsvarets behov. Videre er FFI rådgiver overfor Forsvarets strategiske ledelse. Spesielt skal instituttet følge opp trekk ved vitenskapelig og militærteknisk utvikling som kan påvirke forutsetningene for sikkerhetspolitikken eller forsvarsplanleggingen.

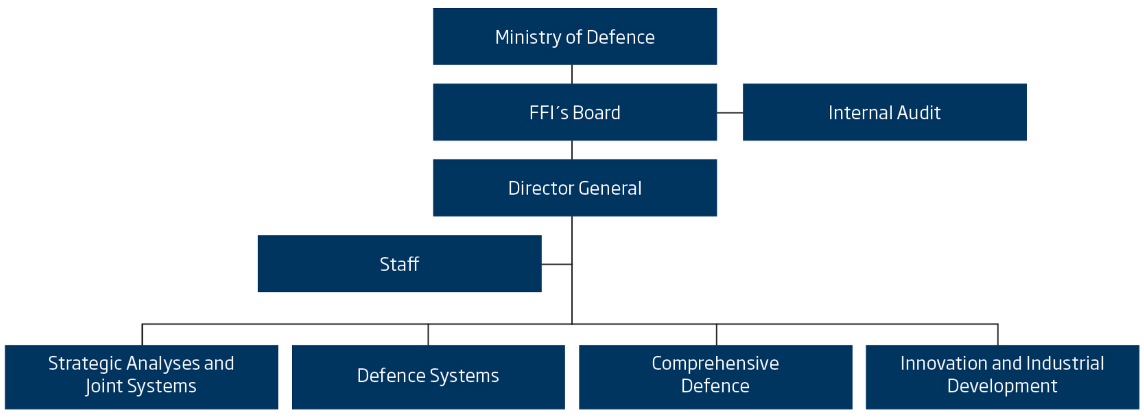
FFIs VISJON

FFI gjør kunnskap og ideer til et effektivt forsvar.

FFIs VERDIER

Skapende, drivende, vidsynt og ansvarlig.

FFI's organisation


Ministry of Defence
FFI´s Board
Internal Audit
Director General
Staff
Strategic Analyses and Joint Systems
Defence Systems
Comprehensive Defence
Innovation and Industrial Development