

Optimal routing strategies for autonomous underwater vehicles in time-varying environment

Mike Eichhorn

Institute for Ocean Technology, National Research Council Canada Arctic Avenue, P.O. Box 12093 St. John's, Newfoundland A1B 3T5, Canada

Abstract

This paper presents a mission system and the therein implemented algorithms for path planning in a time-varying environment based on graph methods. The basic task of the introduced path planning algorithms is to find a time-optimal path from a defined start position to a goal position with consideration of the time-varying ocean current for an autonomous underwater vehicle (AUV). Building on this, additional practice-oriented considerations in planning are discussed in this paper. Such points are the discussion of possible methods to accelerate the algorithms and the determination of the optimal departure time. The solutions and algorithms presented in this paper are focused on path planning requirements for the AUV “SLOCUM Glider”. These algorithms are equally applicable to other AUVs or aerial mobile autonomous systems.

Keywords: Path planning, Geometrical graph, Graph methods, Time-varying environment, AUV, AUV “SLOCUM Glider”, Autonomous systems

1. Introduction

This paper is an abridgement of a research fellowship and has been previously published in parts in [1–4]. The following sections review the important results of the study for path planning in a time-varying environment for the autonomous underwater vehicle (AUV) “SLOCUM Glider”.

Path planning represents an important characteristic of autonomous systems. It reflects the possibility for a planned behaviour during a mission using all current and future information about the area of operation. This planning task will be complicated because of the unknown, inaccurate and varying information. The path planning algorithms presented in this paper are designed considering the mission requirements for the AUV “SLOCUM Glider”. These gliders have a low cruising speed (0.2 to 0.4 m s^{-1}) in a time-varying ocean flow over a long operation range for periods up to 30 days.

There exists a variety of solutions for the path planning in a time-varying environment, especially for mobile autonomous systems. A generic algorithm was used for an AUV in [5] to find the path with minimum energy cost in a strong, time-varying and space-varying ocean current field. This approach finds a robust solution which will not necessarily correspond with the optimal solution. In [6], an adaptive genetic algorithm is presented for determining routes for a large-scale sea area under real-time re-

quirements. The defined fitness function allows the generation of a time-optimal, obstacle-free route with few waypoints. The mixed integer linear programming (MILP) will be used for handling multiple AUVs [7] or UAVs (Unmanned Aerial Vehicles) [8]. As the computing time of MILP increases exponentially with the problem size, this approach has limitations in real-sized applications. A solution with a non-linear least squares optimization technique for a path planning of an AUV mission through the Hudson River was presented in [9]. The optimization parameters are a series of changeable nodes $(x_i, y_i, z_i, \Delta t_i)$, which characterize the route. The inclusion of the time intervals Δt_i allows a variation of the vehicle speed during the mission and thus the integration of energy considerations in the optimization. This approach runs the risk of finding only a local minimum. In [10] a solution with optimal control to find the optimal trajectory for a glider in a time-varying ocean flow was presented. This approach applied the Nonlinear Trajectory Generation (NTG) algorithm including an ocean current flow B-spline model, a dynamic glider model as well as a defined cost function which is a weighted sum of a temporal and an energy cost. The inclusion of energy requirements using priori known wind information in a graph based path planning for UAVs was discussed in [11]. In [12], the level set method for time-optimal path planning in a time-varying flow field is used. In this case, a wave front, starting from the start position is generated. It consists of particles, which describe the most distant position from the vehicle, which can be achieved at a defined time. When the wave front reaches

Email address: mike.eichhorn@tu-ilmenau.de,

Mike_Eichhorn@gmx.de, mike.eichhorn@ieee.org. (Mike Eichhorn)



the target position the optimal route will be determined by backtracking the particles. The accuracy of the numerical solution found and the computing time correlate with the defined points to describe the individual wave front lines at certain discrete times. In [13] a modified A*-algorithm was used to find a time optimal path using Regional Ocean Model (ROM) data. This algorithm, called Constant-Time Surfacing A*(CTS-A*), considered the specific glider dynamics under the influence of ocean currents. An A*-algorithm was used in [14] to find a minimum risk path for gliders using historical data from the Automated Information System (AIS) for automatically identifying and locating vehicles.

The chosen pre-defined mesh structure to define the connectivity relations of the several vertices in a geometrical graph has an important influence to find the optimal path in a current field using graph algorithms. This is confirmed in [15] which is a continuing work of [16]. Both works use an A* algorithm to find an optimal path in a spatial variability and time-invariant ocean current field. The influence of the mesh structure on the determined path is discussed in Section 2.1.

The planning algorithms presented in Sections 2.2.1 - 2.2.3 are based on a modified Dijkstra Algorithm (see [17]), including the time-variant cost function in the algorithm which will be calculated during the search to determine the travel times (cost values) for the examined edges. This modification allows the determination of a time-optimal path in a time-varying environment. In [18] this principle was used to find the optimal link combination to send a message via a computer communication network with the shortest transport delay.

The presented methods to accelerate the path planning algorithms result from trying to determine real mission plans for the AUV “SLOCUM Glider” to collect oceanographic data along the Newfoundland and Labrador Shelf [3]. The number of edges in the geometrical graph ranges from one hundred thousand to one million for a real mission of duration of 10 days, whereby the sum of the cost function calculations is very time-intensive. This cost function calculations to detect the travel time for an edge are described in Section 3 in detail.

Because the required geometrical graph is not complete as not all vertices are connected by an edge within the graph, the found path has to be smoothed for a real glider mission. This path post processing is a necessary step in real applications (see [19, 20]). The path smoothing for time varying conditions will be discussed in Section 4.

A fast working algorithm is also a precondition for the detection of an optimal departure time, which is described in Section 5. A symbolic wavefront expansion (SWE) technique for a UAV in time-varying winds was introduced in [21] to find the time optimal path and additionally the optimal departure time. The path planning algorithms in this paper use a similar principle as is used in the SWE to calculate the time-varying cost function for the several vertices. This includes the arrival time at the several vertices

in the cost function calculation during the search. To find the optimal departure time, the SWE and the approach described in this paper use separate solution methods. The reasons are the accurate and fast determination of the optimal departure time, as well as the possible inclusion of uncertainties in the path planning as a result of forecast error variance, accuracy of calculation in the cost functions and a possible use of a different vehicle speed in the real mission than planned [22].

Section 6 shows the results of the presented algorithms using a simple mathematical model of the Gulf Stream and real netCDF files for a 10-day forecast. Conclusion and future research topics are in Section 7.

2. Graph Algorithm

2.1. Generation of the geometrical graph

The geometrical graph is a mathematical model for the description of the area of operation with all its characteristics. Therefore defined points (vertices) within the operational area are those passable by the vehicle. In this paper these points define positions in the 2D Euclidean space whereby the geometrical graph is planar. The passable connections between these points are recorded as edges in the graph. Every edge has a rating (cost, weight) which can be the length of the connection, the evolving costs for passing the connection or the time required for traversing the connection. There exist many approaches to describe an obstacle scenario with as few of the vertices and edges as possible, and, to decrease the computing time (visibility and quadtree graph [23]). In the case of the inclusion of an ocean current, the mesh structure of the graph will be a determining factor associated with its special change in gradient. In other words, the defined mesh structure should describe the trend of the ocean current flow in the operation area as effectively as possible. A uniform rectangular grid structure is the easiest way to define such a mesh. In the simplest case the edges are the connections between neighbouring obstacle-free sectors; see Fig. 1(a). To achieve a shorter and smoother path for mobile robots additional edges to other sectors are implemented in [24]; see Fig. 1(b). The analyses of the found paths in a current field show (see also [1, 2]) that it is important to define a great number of edges with different slopes; see Fig. 1(c). A further increase of the number of radiated edges leads to increasing lengths which is not practical to describe the change in gradient of the current flow.

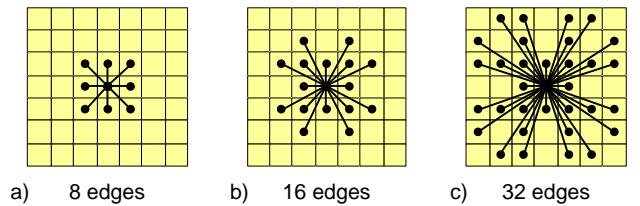


Fig. 1. Rectangular grid structure a) 1-sector, b) 2 sector, c) 3 sector

2.2. Graph-based Search-Algorithm

The developed search algorithms are all based on the Dijkstra Algorithm [17] which solves the single-source shortest paths problem on a weighted directed graph. The exact solution by using a Dijkstra algorithm in a time-varying environment requires the inclusion of the time information as an additional dimension in the graph. For instance a 2D geometrical graph acquires additional layers for each defined discrete point of time. That will lead to very large graphs with many vertices and edges as a result of using small time intervals. To get around this, the algorithms include the time-variant cost function which will be calculated during the search to determine the travel times for the examined edges. The basic algorithm, named TVE (time-varying environment) algorithm, uses only this modification. This algorithm is described in detail in [1, 2]. Its operation is similar to the A*TVE algorithm, which is presented in the next section, using the cost function $d[u]$ instead of the estimated costs $f[u]$ in the priority queue Q . Methods to accelerate the processing time of the TVE algorithm will be presented in the next sections.

2.2.1. A*TVE algorithm

A possible method to accelerate the TVE algorithm is the inclusion of an A* algorithm [25]. The A* algorithm utilizes the Dijkstra algorithm and uses a heuristic function $h(u)$ to decrease the processing time of the path search. As a heuristic function in the A*TVE algorithm, the travel time t_{travel} from the current vertex u at position \mathbf{x}_u to the goal vertex g at position \mathbf{x}_g following a straight line based on [13] will be used. Here the travel time will be calculated using the maximum possible speed, as determined by the addition of the vehicle speed through the water $v_{veh.bf}$ and the maximum current velocity $v_{current.max}$ in the operational area over the full mission time:

$$h(u) = t_{travel} = \frac{\|\mathbf{x}_u - \mathbf{x}_g\|}{v_{veh.bf} + v_{current.max}} \quad (1)$$

Table 1 shows a comparison between the A* algorithm (left column) and the A*TVE (time-varying environment) algorithm (right column). The syntax of the pseudo-code is adapted from [26]. The input parameter G contains the graph structure with the vertex list and the edge list (V and E), s and g are the source and goal vertex and t_0 includes the starting mission time. The parameter d includes the cost list for the several vertices, f includes the estimated costs from the source vertex s to the goal vertex g of the path through the several vertices using the sum of the known cost value $d[v]$ from the source s to the vertex v and the value $h(v)$ of Eq. (1), and π includes the predecessor of each vertex which is used to encode the shortest paths tree [26]. Q is a priority queue that supports the INSERT, EXTRACT-MIN and the DECREASE-KEY operations. The operation EXTRACT-MIN removes the vertex u which the least cost value $f[u]$ in the priority queue Q . The operation DECREASE-KEY assigns a new

cost value f to the vertex v in the queue Q . The *color* list defines the current state of the vertex in the priority queue Q . The allowable states are WHITE, GRAY and BLACK: WHITE indicates that the vertex has not yet been discovered, GRAY indicates that the vertex is in the priority list, and, BLACK indicates that the vertex was checked. The shaded text fields in Table 1 highlight the differences between the algorithms. There are the following three differences:

1. The new algorithm does not need the weight list w of the edges to begin the search. The algorithm needs a start time t_0 when the vehicle begins the mission.
2. The examination of the edge (u, v) is only necessary for $d[u] < d[v]$. It is clear if $d[u] \geq d[v]$ then $d_v > d[v]$, independent of the calculated weight of function $wfunc$. The parameter d_v includes the calculated cost value for vertex v by sum of cost value for vertex u and the calculated weight of function $wfunc$.
3. The algorithm calculates the weight for the edge $w(u, v)$ in a function $wfunc$ during the search. (see Section 3.2, a detailed description about these calculations will be presented in Section II.b in [2]) This function calculates the travel time to drive along the edge from a start vertex u to an end vertex v using a given start time. The start time to be used will be the current cost value $d[u]$, which describes the travel time from the source vertex s to the start vertex u .

Table 1
Pseudo-code of the A* and A*TVE algorithms

A*(G, s, g, w)	A*TVE(G, s, g, t_0)
for each vertex $u \in V$	for each vertex $u \in V$
$d[u] \leftarrow \infty$	$d[u] \leftarrow \infty$
$f[u] \leftarrow \infty$	$f[u] \leftarrow \infty$
$\pi[u] \leftarrow \infty$	$\pi[u] \leftarrow \infty$
$color[u] \leftarrow \text{WHITE}$	$color[u] \leftarrow \text{WHITE}$
$color[s] \leftarrow \text{GRAY}$	$color[s] \leftarrow \text{GRAY}$
$d[s] \leftarrow 0$	$d[s] \leftarrow t_0$
$f[s] \leftarrow h(s)$	$f[s] \leftarrow t_0 + h(s)$
INSERT(Q, s)	INSERT(Q, s)
while ($Q \neq \emptyset$)	while ($Q \neq \emptyset$)
$u \leftarrow \text{EXTRACT-MIN}(Q)$	$u \leftarrow \text{EXTRACT-MIN}(Q)$
if ($u = g$)	if ($u = g$)
return (d, π)	return (d, π)
$color[u] \leftarrow \text{BLACK}$	$color[u] \leftarrow \text{BLACK}$
for each $v \in \text{Adj}[u]$	for each $v \in \text{Adj}[u]$
$d_v = w(u, v) + d[u]$	$d_v = wfunc(u, v, d[u]) + d[u]$
if ($d_v < d[v]$)	if ($d_v < d[v]$)
$d[v] \leftarrow d_v$	$d[v] \leftarrow d_v$
$f[v] \leftarrow d_v + h(v)$	$f[v] \leftarrow d_v + h(v)$
$\pi[v] \leftarrow u$	$\pi[v] \leftarrow u$
if ($color[v] = \text{GRAY}$)	if ($color[v] = \text{GRAY}$)
DECREASE-KEY($Q, v, f[v]$)	DECREASE-KEY($Q, v, f[v]$)
else	else
$color[v] \leftarrow \text{GRAY}$	$color[v] \leftarrow \text{GRAY}$
INSERT(Q, v)	INSERT(Q, v)
return (d, π)	return (d, π)

2.2.2. Optimal navigation formula from Zermelo

The use of the TVE algorithm to find a time optimal path for the AUV “SLOCUM Glider” in time varying ocean flows allows a further possibility to reduce the computing time of the search. This approach uses the optimal navigation formula from Zermelo [27]:

$$\frac{d\theta}{dt} = -u_y \cos^2 \theta + (u_x - v_y) \cos \theta \sin \theta + v_x \sin^2 \theta \quad (2)$$

with θ as the heading and u_x , u_y , v_x and v_y as the partial derivatives of the ocean current components u and v . The idea to develop this formula came to Zermelo’s mind when the airship “Graf Zeppelin” circumnavigated the earth in August 1929 [28]. This formula describes the necessary condition for the control law of the heading θ , to steer a vehicle in a time-optimal sense through a time-varying current field. The gradient of the resulting optimal trajectory in a fixed world coordinate system is the vehicle velocity over the ground \mathbf{v}_{veh_og} . This vector is the result of a vector addition of the current vector $\mathbf{v}_{current}$ and the \mathbf{v}_{veh_bf} vector with vehicle speed through the water v_{veh_bf} as norm and heading θ as direction. The direction of this vector \mathbf{v}_{veh_og} is the course over the ground (COG) ϕ . These relationships are illustrated in Fig. 2. The idea of how to use the optimal navigation formula in the search algorithm as well as the several necessary program steps will be described subsequently.

Assuming that the search algorithm will find the time-optimal path, then the several segments (edges) of this path will match well with the optimal trajectory, which is calculated with optimal control by solving the optimal navigation formula from Zermelo. This assumption means that during the path search only vertices should be considered where the connections (edges) comply with the optimal navigation formula. This compliance is required where the transition that is the change of direction between two adjacent edges is matched with Eq. (2).

The determination of the optimal path direction ϕ_{opt} on position x_{start} required a simulation of the optimal trajectory by starting on the middle position of the previous edge by x_{start_intern} (see section II.D in [4]). The calculated path direction ϕ_{opt} will be used to select possible successor edges with the end vertex v under consideration of an angle range $\pm \Delta \phi_{max}$ (see Fig. 3). This range considers the maximal possible angle between two adjoined edges and the fact that the path direction ϕ_{opt} is only an

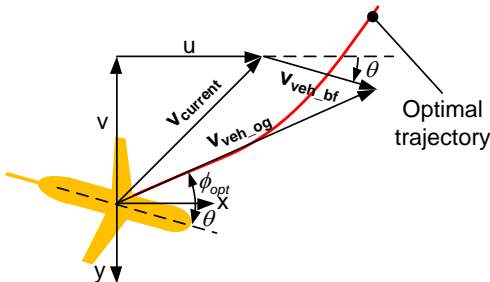


Fig. 2. Illustration of the velocities and the angles in glider steering

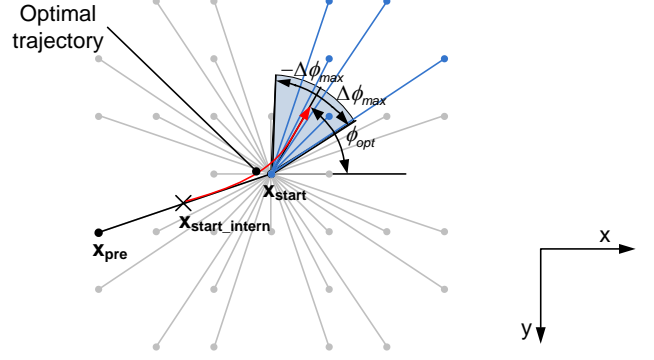


Fig. 3. Resultant angle range to define the examined successor vertices using optimal navigation formula from Zermelo

average value along the path and is predetermined through the given numbers of possible edges from the differences in slopes according to the chosen mesh structure (see Fig. 1).

This approach incorporates a pre-selection of promising successor vertices with the goal to decrease the number of cost function calls $wfunc$ during the search. Fig. 3 shows the principle idea of the approach using a 3-sector rectangular grid structure which is described in Section 2.1. By using such a structure, 31 successor vertices are possible from which the approach selects only five. This occurs in the best case (for all examined vertices v is $d[u] < d[v]$) with a resulting reduction of the called $wfunc$ to 83 % using this ZTVE algorithm.

2.2.3. The use of both methods

The use of both methods together, the A* algorithm (Section 2.2.1) and the optimal navigation formula from Zermelo (Section 2.2.2) in the TVE algorithm combines the two acceleration mechanisms and produces a larger reduction in the computing time than with either method alone. Table 2 shows this algorithm, named ZA*TVE, with a few explanations. The letters which appear in the explanation column refer to the used method (A*: A* algorithm; Z: Zermelo’s formula). According to Eq. (2), the function CAL-OPTDIR calculates the optimal path direction ϕ_{opt} on position u to the time $d[u]$ using the direction of the edge with the predecessor vertex $\pi[u]$ as start vertex and the current examined vertex u as the end vertex which should reflect the average optimal COG.

At this point additional acceleration possibilities should be discussed briefly. The first possibility includes the selective reduction of the search area to decrease the number of examined vertices during the search. To do this a first search run uses a graph with a large grid size and/or a simple grid structure (see Section 2.1). Around the found path a new geometrical graph will be generated, similar to a pipe. This graph will have a fine grid size and/or a complex grid structure and will be used in a second run to find the optimal path. A modification of the upper approach is the use of a simple cost function in the first search run and the use of an accurate glider-model in the cost function for the second run.

Table 2

Pseudo-code of the ZA*TVE algorithm

ZA*TVE($G, s, g, t_0, \Delta\phi_{max}$)	Explanations
for each vertex $u \in V$	
$d[u] \leftarrow \infty$	
$f[u] \leftarrow \infty$	A* (initialize heuristic vector)
$\pi[u] \leftarrow \infty$	
$color[u] \leftarrow \text{WHITE}$	
$color[s] \leftarrow \text{GRAY}$	
$d[s] \leftarrow t_0$	
$f[s] \leftarrow t_0 + h(s)$	A* (calculate heuristic for vertex s)
INSERT(Q, s)	discover vertex s
while ($Q \neq \emptyset$)	
$u \leftarrow \text{EXTRACT-MIN}(Q)$	A* examine vertex u
if ($u = g$)	A* (path found)
return (d, π)	A* (program termination)
$color[u] \leftarrow \text{BLACK}$	
if ($u \neq s$)	
$\phi_{opt} = \text{CALC-OPTDIR}(\pi[u], u, d[\pi[u]], d[u])$	Z (calculate optimal course)
for each $v \in \text{Adj}[u]$	examine edge (u, v)
if ($d[u] < d[v]$)	
$\phi_{path} = \text{CALC-PATHDIR}(u, v)$	Z (calculate edge direction)
if ($((u=s) \text{ OR } (\phi_{opt} - \phi_{path} < \Delta\phi_{max}))$)	Z (select possible successor edges)
$d_v = w_{func}(u, v, d[u] + d[v])$	calculate cost function
if ($d_v < d[v]$)	
$d[v] \leftarrow d_v$	
$f[v] \leftarrow d_v + h(v)$	A* (calculate heuristic function)
$\pi[v] \leftarrow u$	
if ($color[v] = \text{GRAY}$)	
DECREASE-KEY($Q, v, f[v]$)	A* (change heuristic for v in Q)
else	
$color[v] \leftarrow \text{GRAY}$	
INSERT(Q, v)	discover or reopen vertex v
return (d, π)	

3. Calculation of the cost value

This section describes the necessary equations and algorithms to determine the travel time t_{path}^i for the several path segments using information about the ocean current.

3.1. Travel time calculation

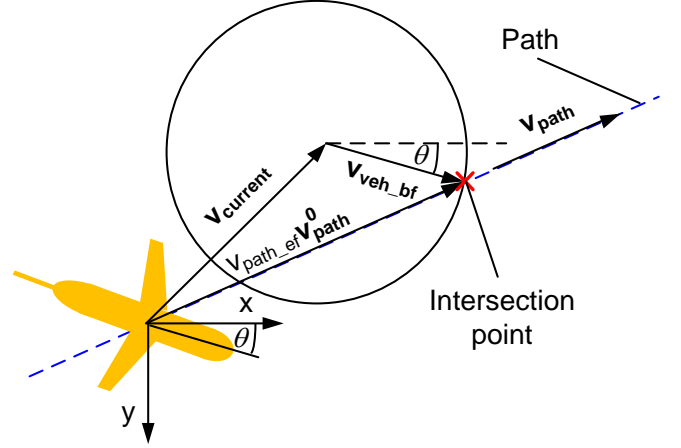
The travel time can be calculated for the i^{th} edge by formation of the quotient of the distance s_{path} and the speed v_{path_ef} , with which the vehicle travels on the path in relation to a fixed world coordinate system:

$$t_{path}^i = \frac{s_{path}^i}{v_{path_ef}^i} \quad (3)$$

This speed v_{path_ef} depends on the vehicle speed through the water v_{veh_bf} (cruising speed), the magnitude and the direction of the ocean current vector as well as the direction of the path \mathbf{v}_{path}^0 . This speed can be determined by the intersection point between a line and a circle (2D) and/or sphere (3D) [29] based on Fig. 4 according to the following relation:

$$\begin{aligned} \text{line: } \mathbf{x}(v_{path_ef}) &= v_{path_ef} \mathbf{v}_{path}^0 \\ \text{circle/spheres: } v_{veh_bf}^2 &= \|\mathbf{x} - \mathbf{v}_{current}\|^2 \end{aligned} \quad (4)$$

$$disc = \left(\mathbf{v}_{path}^0 \cdot \mathbf{v}_{current} \right)^2 + v_{veh_bf}^2 - \mathbf{v}_{current} \cdot \mathbf{v}_{current} \quad (5)$$

**Fig. 4.** Definition of the velocities

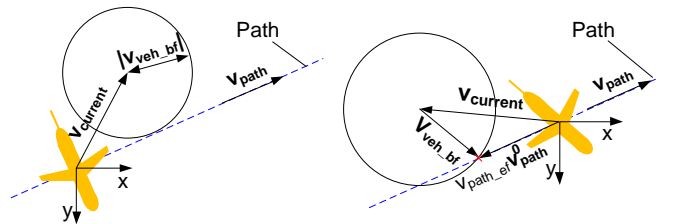
If the discriminant $disc$ in equation Eq. (5) becomes negative, v_{path_ef} has not a real solution, so that the value will be defined as NaN (Not a Number):

$$v_{path_ef} = \begin{cases} \mathbf{v}_{path}^0 \cdot \mathbf{v}_{current} + \sqrt{disc} & , \text{ for } disc > 0 \\ NaN & , \text{ otherwise.} \end{cases} \quad (6)$$

It means that the vehicle can no longer be held in that path, the path is not feasible; see Fig. 5 (a). If the speed v_{path_ef} is negative the vehicle is still on the path, but moving backwards; Fig. 5 (b). Both cases must be considered by setting a large numerical value for the edge weight. Thus, such paths are excluded in the search and it does not come to a situation that the vehicle encounters a strong backwards current and leaves the path.

3.2. Travel time calculation in time-varying ocean flow

The determination of the travel time according to equation Eq. (3) works only if the ocean current is constant along the path, or through an appropriate choice of the mesh sizes of the graph for a location-varying ocean current. In the case of a time-varying ocean current or a too coarse mesh structure used in conjunction with a location-varying ocean current, the speed v_{path_ef} will be changed depending on the current $\mathbf{v}_{current}$ along the path element. The used algorithm to solve this problem is based on a step size control for efficient calculation of numerical solutions of differential equations [30]. The step size h is here not the time as used in numerical solvers but is a segment

**Fig. 5.** a) Negative discriminant b) $v_{path_ef} < 0$

of the path element. So the path element will be shared within many segments, for which Eq. (3) using Eq. (6) can be solved. The current $\mathbf{v}_{current}$ in Eq. (6) is the arithmetic mean of the two velocities at the begin and the end of the several shared element. Detailed information about the algorithm is described in Section III.B [2].

3.3. Glider dive profile cost function

The glider dive profile is specified by its locomotion principle. By changing its buoyancy, the glider is able to descend (dive) and ascend (climb). The result is a saw-tooth vertical profile as shown schematically in Fig. 6. The exact simulation of such a dive profile is computationally time-intensive and so is impractical because of the number of edges in the geometrical graph, which range from one hundred thousand to one million. Conversely, the knowledge of the glider's behaviour in every passable depth is necessary for the planning and makes it possible that the mission planning can avoid regions with an adverse surface or seabed current.

To include the depth-varying ocean current information in the cost function (presented in Section 3.1) the path element is divided into several path segments. The number of the segments $n_{segments}$ is defined by the step size h . This number shall consider the changeable ocean current conditions along the path at every passable depth. In each segment the glider dives from the "climbto" depth $z_{climb-to}$ until the "diveup" depth $z_{dive-up}$; see Fig. 6. The calculated travel time t_{travel} for the segment will correspond approximately to the travel time which the glider needs to travel along every segment of the saw-tooth profile. Fig. 6 shows the simplified dive profile in comparison to the real saw-tooth profiles. The details of the algorithm to calculate the travel time are included in [3].

3.4. Ocean current determination

The cost function to calculate the travel time operates like a numerical simulation (see Section 3.1 and 3.2). This simulates the vehicle driving along the path from a defined start position \mathbf{x}_{start} at start time t_{start} to the end

position \mathbf{x}_{end} under consideration of the local ocean current. During such a simulation the cost function requires a large amount of local ocean current information, which is generated in the ocean current model. Thereby the Cartesian position \mathbf{x} , the depth z and the time t are passed from the cost function to the ocean current model. The model returns a two dimensional ocean current vector $\mathbf{v}_c = [u \ v]$. To determine the travel time, accurate ocean current information along the path element is required. This ocean current information will be provided in this research project through the DFO's (Department of Fisheries and Oceans) Canada-Newfoundland Operational Ocean Forecast System (C-NOOFS) in the form of netCDF files, generated in a numerical model. At present it is not possible to directly couple the search algorithm and the numerical model because of time and computational constraints.

3.4.1. Preparation of the netCDF-Files

The Network Common Data Format (NetCDF) is a binary data format for array-oriented scientific data [31] which is commonly used for climatology, meteorology and oceanography applications. The C-NOOFS provides the ocean current data at various depths (0 to 5700 m) for the entire Northwest Atlantic with a resolution of approx. 6 km in the region of interest, every 6 h for a 10-day forecast in geographical coordinates. To use these data in the ocean current model, they need to be extracted out of the region of interest in a Cartesian coordinate system as reference. These modifications can be addressed by using the FIMEX (File Interpolation, Manipulation and EXtraction) library [32].

3.4.2. Multi-dimensional interpolation scheme

Since the ocean current data coming from the forecasting system as data files will be provided only at discrete times and positions with a coarser time and length scale than is required to generate an efficient path, a multi-dimensional interpolation scheme will be utilized to generate the desired data. Fig. 7 shows the scheme for the ocean current component v in overview. The first interpolation step uses a two-dimensional interpolation function from the FIMEX library to extract the ocean current information for the several depth layers. A Nearest-Neighbour, a Bilinear and a Bicubic interpolation method are available. The interpolations via the depth and time dimensions occur separately using one-dimensional interpolation functions. Nearest-Neighbour, Linear, Cubic and Akima interpolation are possible. The first two methods require two fields (for time t) or layers (for depth z), the other methods require minimal three, optimal five fields or layers in order to generate the ocean current component v at the defined position \mathbf{x}_i , at the depth z_i and at the time t_i . The implementation of the Akima interpolation [33] can make allowance for an abrupt change of ocean current conditions in case of tides or of different depth streams.

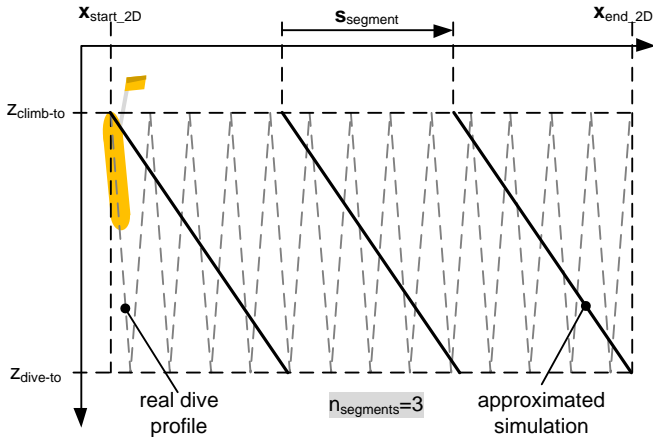


Fig. 6. Simplified dive profile along a path element

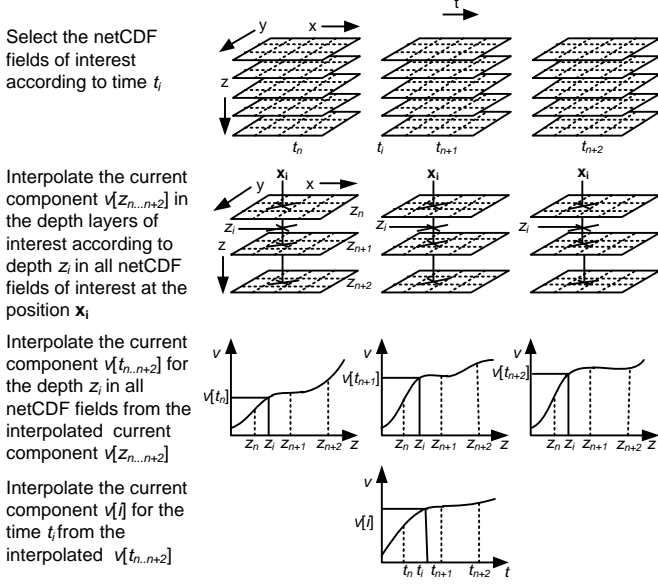


Fig. 7. Steps to interpolate an ocean current component $v[i]$ at position $x[i]$, depth z_i and time t_i from the netCDF fields

4. Path smoothing

The required geometrical graph for the search algorithm is not complete as not all vertices are connected by an edge within the graph (see Section 2.1). A complete directed graph on n vertices has $n(n-1)$ edges, which would evolve into a very large graph and a long computing time for the path search. So, the search algorithm can consider in its search only the edges which are included in the non-complete graph. The paths found are characterized by many several path segments with change of directions. The run of such a path is stair- or wiggle-shaped, which a glider cannot follow. Therefore a method to smooth the candidate path will be presented as follows. Table 3 includes the details of the algorithm to smooth the path under consideration in the time-varying environment. The candidate path is described by a list of waypoints WP . The algorithm verifies the start point $WP[i_{start}]$ of the list with the subsequent waypoints $WP[i_{path}]$ of a direct connection (III), with the goal of a quicker arrival at this point by using the several path elements (IV). Verification of the arrival time $TT[end]$ of the goal point $WP[end]$ from the tested waypoint $WP[i_{path}]$ using the existing subsequent waypoints (V) also occurs. This second verification through the time-varying environment is necessary and ensures that the merge of path elements indeed leads to a quicker arrival at a local waypoint, but leads to a later arrival time at the goal point. This is possible even though the ocean current situation is changing dynamically. The merging begins by the third waypoint (II) and will be executed until one of the two verifications is satisfied or the goal point is reached. In the case of a positive verification ($merge = false$), the present waypoint $WP[i_{path}]$ will be stored in the new waypoint list and a new merge will begin at the precedent waypoint (VI). The result is a

waypoint list WP_{smooth} with fewer waypoints in the verified waypoint list WP . If obstacles are encountered, the function TRAVELTIME also calculates the intersections of the obstacles with the several path elements. In case of a collision situation, the resulting time value has a large numerical value. The above described procedure will be repeated until the number of waypoints is constant between two sequent loops (I). Fig. 8 shows an example for a better understanding.

Table 3
Algorithm for path smoothing in time-varying environment

SMOOTHING (WP, t_0)	
$TT[1] = t_0$	
for ($i = 2$) to ($i = \text{length}(WP)$)	
$TT[i] = TT[i-1] + \text{TRAVELTIME}(WP[i-1], WP[i], TT[i-1])$	
while ($\text{length}(WP) \neq \text{length}(WP_{smooth})$) AND ($\text{length}(WP) > 2$)	I
$i_{start} = 1$	
$u_{smooth} = 1$	
$TT_{smooth} = \emptyset$	
$WP_{smooth} = \emptyset$	
$TT_{smooth}[u_{smooth}++] = TT[1]$	
$WP_{smooth}[u_{smooth}++] = WP[1]$	
$t_{travel_1} = TT[2] - TT[1]$	
for ($i_{path} = 3$) to ($i_{path} = \text{length}(WP)$)	II
$merge = true$	
$t_{travel_2} = \text{TRAVELTIME}(WP[i_{path}-1], WP[i_{path}], TT[i_{path}-1])$	
$t_{travel_sum} = \text{TRAVELTIME}(WP[i_{start}], WP[i_{path}], TT[i_{start}])$	III
if ($t_{travel_sum} = \infty$)	
$merge = false$	
else	
if ($t_{travel_1} + t_{travel_2} < t_{travel_sum}$)	IV
$merge = false$	
else	
$t_{end} = TT[i_{start}] + t_{travel_sum}$	V
for ($i_{end} = i_{path} + 1$) to ($i_{end} = \text{length}(WP)$)	
$t_{end} = t_{end} + \text{TRAVELTIME}(WP[i_{end}-1], WP[i_{end}], t_{end})$	
if ($t_{end} > TT[end]$)	
$merge = false$	
else	
$TT[end] = t_{end}$	
if ($merge = true$)	
$t_{travel_1} = t_{travel_sum}$	
$TT[i_{path}] = TT[i_{start}] + t_{travel_sum}$	
else	
$TT[i_{path}-1] = TT[i_{start}] + t_{travel_1}$	
$TT[i_{path}] = TT[i_{start}] + t_{travel_1} + t_{travel_2}$	
$i_{start} = i_{path} - 1$	
$t_{travel_1} = t_{travel_2}$	
$TT_{smooth}[u_{smooth}++] = TT[i_{start}]$	VI
$WP_{smooth}[u_{smooth}++] = WP[i_{start}]$	
if ($merge = false$)	
$TT[end] = TT[end-1] + \text{TRAVELTIME}(WP[end-1], WP[end], TT[end-1])$	
$TT_{smooth}[u_{smooth}++] = TT[end]$	
$WP_{smooth}[u_{smooth}++] = WP[end]$	
$TT = TT_{smooth}$	
$WP = WP_{smooth}$	
return WP	

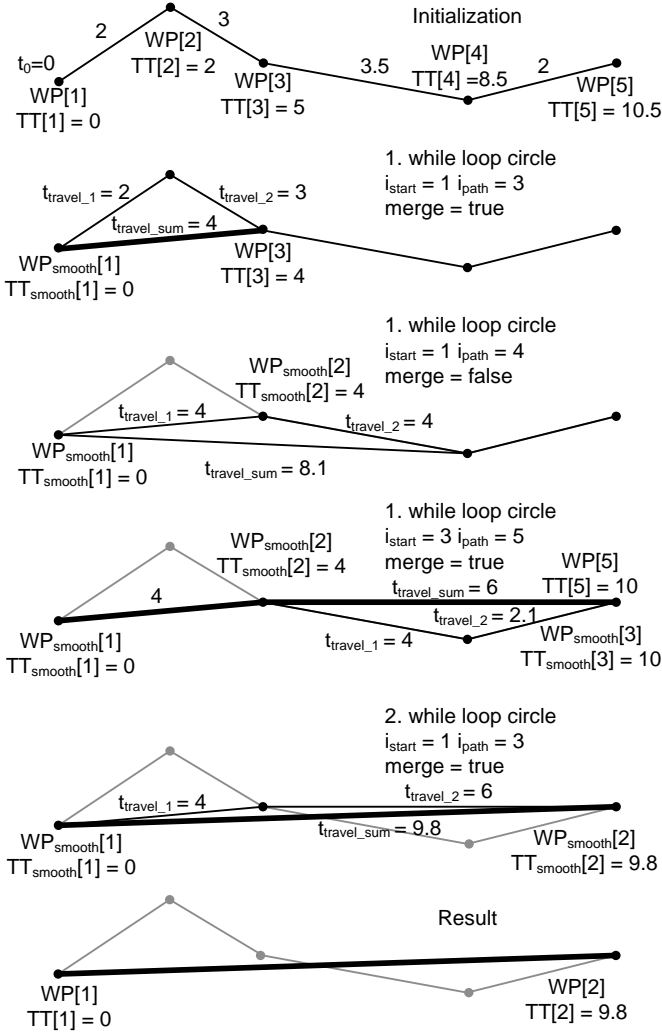


Fig. 8. Several steps to smooth a path

5. Detection of the optimal departure time

A practice-relevant requirement for optimal path planning for the AUV “SLOCUM Glider” is the determination of the optimal departure time. So is it very difficult to start a glider mission near the coast in the presence of strong tides. Through the low cruising speed (0.2 to 0.4 m s^{-1}) and a false chosen start time in combination with a strong flowing tide, it is possible that the glider will make poor forward progress or drift back to the shore. Another scenario is a bad weather situation or a temporary adverse ocean current condition in the region of interest.

5.1. Idea

The function to describe the relationship between the travel time t_{trav} and departure time t_{dep} consists of an independent single pair of variants. This means that to determine the travel time for a certain departure time, the knowledge of travel times with a lesser departure time is not necessary. Because of this, it is possible to reproduce the principle run of the curve $t_{trav} = f(t_{dep})$ using a smaller number of defined departure times $t_{dep,i}$, distributed in the time window of interest, to find the corresponding travel

times $t_{trav,i}$. In an additional step the region of the global minimum can be localized, to detect the optimal departure time using a root-finding algorithm. The algorithmic details will be described in the next section.

5.2. Algorithm

The detection of the optimal departure time occurs in three steps. Fig. 9 displays an overview of the scheme to determine the optimal departure time. The first step creates supporting points for the curve $t_{trav} = f(t_{dep})$ at intervals of Δt_{dep} . The choice of the interval width is based on the run of the curve and should reflect the positions of the local minima.

These supporting points will be provided in a second step to create the approximated run of the curve using an interpolation method. The studies in this research field favour the Akima interpolation [33]. This method provides the best fitting to the real curve and tries to avoid overshoots, which would indicate a nonexistent minimum. The determination of the interval wherein the global minimum of the approximated curve lies is the precondition for the last step.

Here a one-dimensional root-finding algorithm will be used to find the optimal departure time. Thereby a path search using the ZA*TVE algorithm will be running alongside every function call to find the travel time for the given departure time. For root-finding algorithms, root-bracketing algorithms will be used. These algorithms work without derivatives and find the root through iterative decreasing of the interval until a desired tolerance is achieved, wherein the root lies. Golden section search [34], Fibonacci search [35] and Brent’s algorithm [36] were tested. Brent’s algorithm has the best performance and will be favoured.

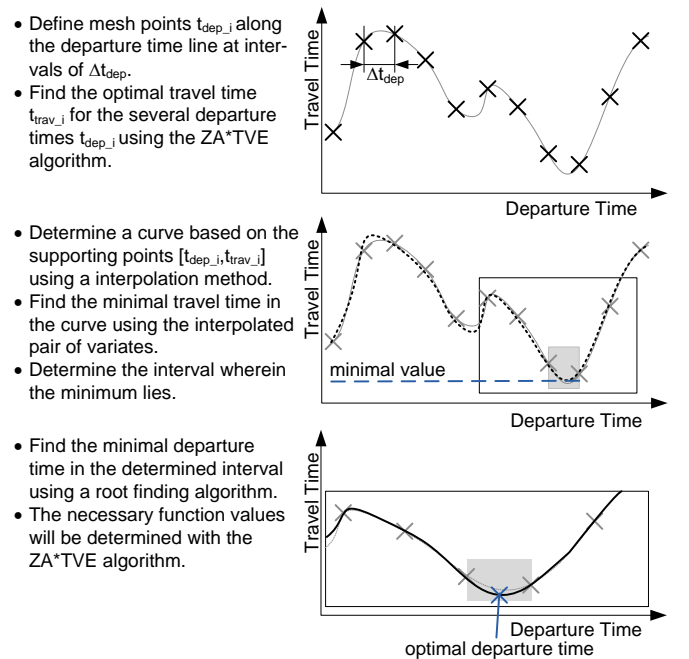


Fig. 9. Steps to find the optimal departure time

5.3. Possible Modifications and critical notes

The above described algorithm calls the search algorithm multiple times, which correlates directly with the processing time. A possibility to reduce the processing time will be discussed briefly. Because the localized global minimum in the second step represents only the rough position, the supporting points used in the interpolation do not have to be accurate. This means that in order to detect these points a graph with a larger grid size and/or a simpler grid structure can be used. The result is a generated graph with fewer edges to be examined during the search, which leads to a more rapid calculation of the approximated travel time for the given departure time.

The multiple calls of the search algorithm can be calculated independently at the same time on separate processor cores of a multi-core computer. The analyses of the possibilities for parallelization and the programmable implementation are current work fields [22].

Use of this approach in a real application requires recognition of the fact that only a limited extent of the forecast window will be available. So, the possible mission window is narrowed down to the period between the considered departure time and the forecast horizon. In the application presented in [3] the forecast window for the ocean currents is 10 days. This means that if one starts a mission on the ninth day only a one day mission can be planned. Another aspect is the delayed supply of the data of interest in the case of a later start time.

6. Results

6.1. The selected test function for a Time-Varying Ocean Flow

The function used to represent a time-varying ocean flow describes a meandering jet in the eastward direction, which is a simple mathematical model of the Gulf Stream [5, 37]. This function was applied in [2, 3] and [4] to test the TVE algorithm and its modifications and in [22] to show the influence of the methods to realize fast search algorithms and to find suboptimal paths using uncertain information. The stream function is

$$\phi(x, y) = 1 - \tanh \left(\frac{y - B(t) \cos(k(x - ct))}{\left(1 + k^2 B(t)^2 \sin^2(k(x - ct))\right)^{\frac{1}{2}}} \right) \quad (7)$$

which uses a dimensionless function of a time-dependent oscillation of the meander amplitude

$$B(t) = B_0 + \varepsilon \cos(\omega t + \theta) \quad (8)$$

and the parameter set $B_0 = 1.2$, $\varepsilon = 0.3$, $\omega = 0.4$, $\theta = \pi/2$, $k = 0.84$ and $c = 0.12$ to describe the velocity field:

$$u(x, y, t) = -\frac{\partial \phi}{\partial y} \quad v(x, y, t) = \frac{\partial \phi}{\partial x} \quad (9)$$

The dimensionless value for the body-fixed vehicle velocity $v_{veh.bf}$ is 0.5. This test function makes it possible to show very transparently how a path planning algorithm works with uncertain information. The exact time optimal solution was found by solving a boundary value problem (BVP) with a collocation method `bvp6c` [38] in MATLAB. The three ordinary differential equations (ODEs) include the two equations of motion:

$$\begin{aligned} \frac{dx}{dt} &= u + v_{veh.bf} \cos \theta \\ \frac{dy}{dt} &= v + v_{veh.bf} \sin \theta \end{aligned} \quad (10)$$

and the optimal navigation formula from Zermelo in Eq. (2).

6.2. Comparison between the methods to accelerate the TVE algorithm

This section presents the results of the methods to accelerate the TVE algorithm which are described in Section 2.2.1 - 2.2.3 using the time-varying ocean flow test function of the previous section. For the test cases, five different start positions were distributed in the whole area of operation as shown in Fig. 10. All the graph-based methods use the same graph and hence produce identical paths. Fig. 10 shows the five paths found using optimal control and the graph methods. For the graph methods, the rectangular 3-sector grid structure with a grid size of 0.4 was used (see Fig. 1(c)). Fig. 11 shows the necessary number of cost function calls (CFC) using the several methods for the five start positions. All these results are included in Table 4. The examination of the current model calls should reflect their ratio to the cost function calls, which is important in the case of computing intensive ocean current calculations. Using the A*TVE algorithm (see Section 2.2.1, the number of function calls correlates directly with the distance between the start and the goal position.

This is reasonable since the algorithm includes only a subset of the vertices in the path search, in fact, only the preferred vertices with a short distance to the goal point.

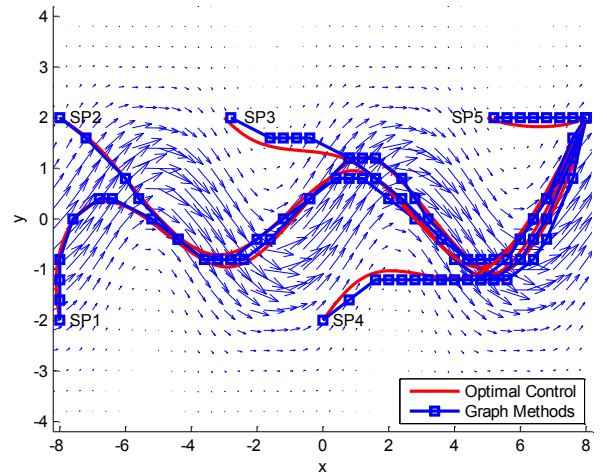


Fig. 10. Time optimal paths through a time-varying ocean field using Optimal Control and the Graph Methods for different start positions

Table 4
Results of the different search methods

Method	SP1 No. of CFC/ No. of CMC	SP2 No. of CFC/ No. of CMC	SP3 No. of CFC/ No. of CMC	SP4 No. of CFC/ No. of CMC	SP5 No. of CFC/ No. of CMC
TVE	12124/ 80838	12126/ 80726	12147/ 81635	12147/ 83886	12112/ 83757
A*TVE	7629/ 46916	6718/ 38564	4042/ 24668	2860/ 16673	638/ 5559
ZTVE	3076/ 27734	2953/ 26842	2763/ 25211	2817/ 25449	2824/ 25295
ZA*TVE	1883/ 17502	1678/ 15630	934/ 8467	627/ 5097	141/ 1409

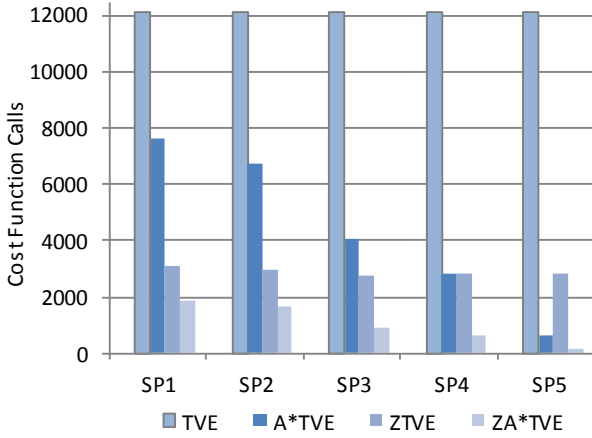


Fig. 11. Cost function calls for the various methods with different start positions

The inclusion of Zermelo's optimal navigation formula in the search algorithm (ZTVE) (see Section 2.2.2 results in a decrease of the number of cost function calls to about one quarter of the calls using the ITVE algorithm. With both methods used together (ZA*TVE), the two merged acceleration mechanisms provide a further decrease of the number of cost function and current model calls. The use of the ZA*TVE algorithm allows a decrease of the number of cost function calls (CFC) by about a factor of 5, and, by a factor of 9 for the current model calls (CMC) in comparison to the TVE algorithm. This improvement makes the practical use of the ZA*TVE algorithm possible for the case of (i) the computationally-intensive ocean current calculations, or, (ii) to determine the optimal departure time. Fig. 12 shows the visited edges (blue lines) using the several search methods started from start position SP1.

6.3. Possible missions and path smoothing algorithm

This section presents the results using the path smoothing algorithm presented in Section 4 by means of a selection of the possible missions along the Newfoundland and Labrador Shelf. Table 5 includes the results of the travel time and the length of the paths found, the smoothed path, and straight line to the goal point. Furthermore, the number of waypoints for the generated (unsmoothed) path and the smoothed path are shown. Fig. 13 shows the mission paths. The trajectories of the unsmoothed and the smoothed path of the missions are similar, so that the two lines are superimposed. The length of the straight line

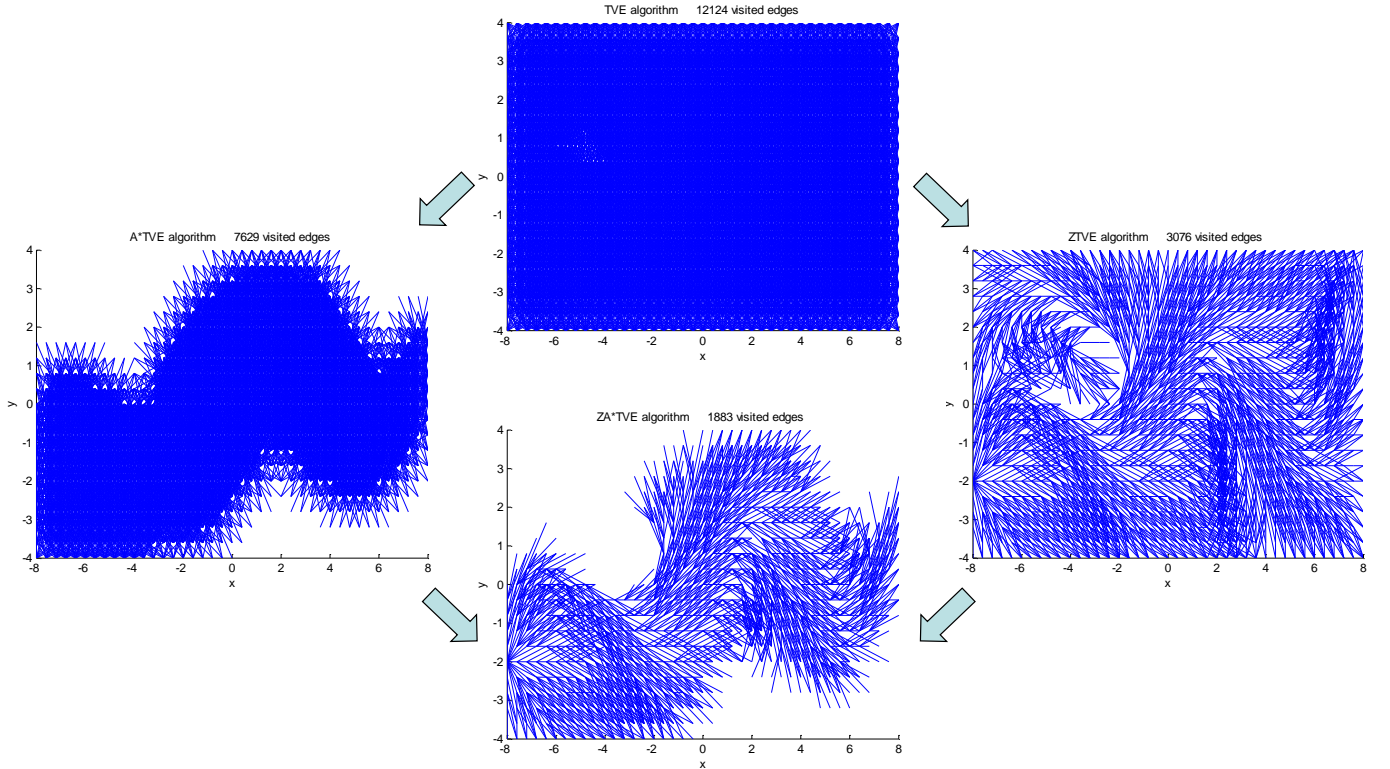


Fig. 12. Visited edges using the several methods

Table 5

Results of the path smoothing algorithm by using different missions

Mission	Travel Time Unsmoothed Path d:h:min:s	Travel Time Smoothed Path d:h:min:s	Travel Time Straight Line d:h:min:s	Travel Time Straight Line without Current d:h:min:s	Path Length Unsmoothed Path km	Path Length Smoothed Path km	Path Length Straight Line km	No. of Waypoints Unsmoothed Path	No. of Waypoints Smoothed Path
Mission 11	08:19:06:06	08:17:34:14	NaN	08:02:26:40	215.87	215.41	210.00	64	19
Mission 12	08:05:21:12	08:05:03:13	09:04:53:02	08:02:57:13	229.40	229.26	210.55	47	26
Mission 13	05:11:44:13	05:11:38:42	06:05:45:51	08:02:57:13	228.30	228.01	210.55	54	31
Mission 14	04:10:26:36	04:10:03:30	04:23:12:36	08:02:26:40	214.24	213.44	210.00	65	30
Mission 21	07:23:58:18	07:23:47:35	08:19:10:56	08:02:57:13	226.91	226.51	210.55	46	23
Mission 22	08:04:14:05	08:03:16:06	08:12:01:05	08:02:26:40	216.49	215.30	210.00	52	14
Mission 23	05:23:38:22	05:23:25:01	07:06:46:35	08:02:57:13	224.86	224.86	210.55	50	41
Mission 24	04:09:37:18	04:09:02:41	NaN	08:02:57:13	222.19	221.09	210.55	49	25
Mission 31	08:12:03:13	08:11:46:59	10:07:59:45	08:02:57:13	229.71	229.41	210.55	46	30
Mission 32	07:09:42:55	07:09:27:14	NaN	08:02:57:13	221.54	221.18	210.55	38	16
Mission 33	08:03:48:33	08:02:24:53	08:12:11:46	08:02:26:40	213.25	211.74	210.00	68	19
Mission 34	07:12:12:10	07:12:03:51	07:23:04:58	08:02:57:13	221.01	220.50	210.55	39	23

for all missions is 210 km. The utilization of the Labrador Stream in Mission M13, M14, M23 and M24 brings a remarkable decrease of the mission endurance in comparison to the other missions. The number of waypoints in the missions can be decreased on average more than half using the smoothing algorithm, which improves the resulting path with respect to travel time. This is possible because new connections (edges) will be created which were not available in the geometrical graph during the search. An additional decrease of the waypoint list is possible, when

longer travel times to the goal point are accepted (see Table 3, Marker V). The direct course to the goal point leads to longer travel times or is impassable in the case of an adverse ocean current.

7. Conclusion and future work

In this paper algorithms for path planning in a time-varying environment based on graph methods are presented. Using the ocean current information in a geometrical graph, the position of the vertices and their possible connections (edges) are very important. This choice should consider the trend of the current flow and the possibility of optimal connections from one vertex to another in a given current field. Methods to accelerate the processing time of the basic TVE algorithm are described in the first part of the paper. The algorithms of the cost function for every connection are presented in the middle part of this paper. This requires the use of a fast calculation for the precise travel time from one vertex to another. In the last part of this paper, an algorithm to detect the optimal departure time is described. Current and future research topics are the analyses of possibilities for parallelization and the inclusion of inaccuracies in path planning as a result of forecast error variance, accuracy of calculation in the cost functions and a different observed vehicle speed in the real mission than planned [22]. The presented path planning algorithms are aimed at saving time. It is also possible, however, to include the energy consumption in the cost function as shown in [5, 10]. An additional research topic is the inclusion of a glider model which simulates the energy consumption in a glider [39], to extract energy information for the cost function.

Acknowledgments

This work was financed by the German Research Foundation (DFG) within the scope of a two-year research fellowship (DFG-Number: EI 813/1-1). I would like to thank the National Research Council Canada Institute for Ocean Technology and in particular Dr. Christopher D. Williams for support during this project.

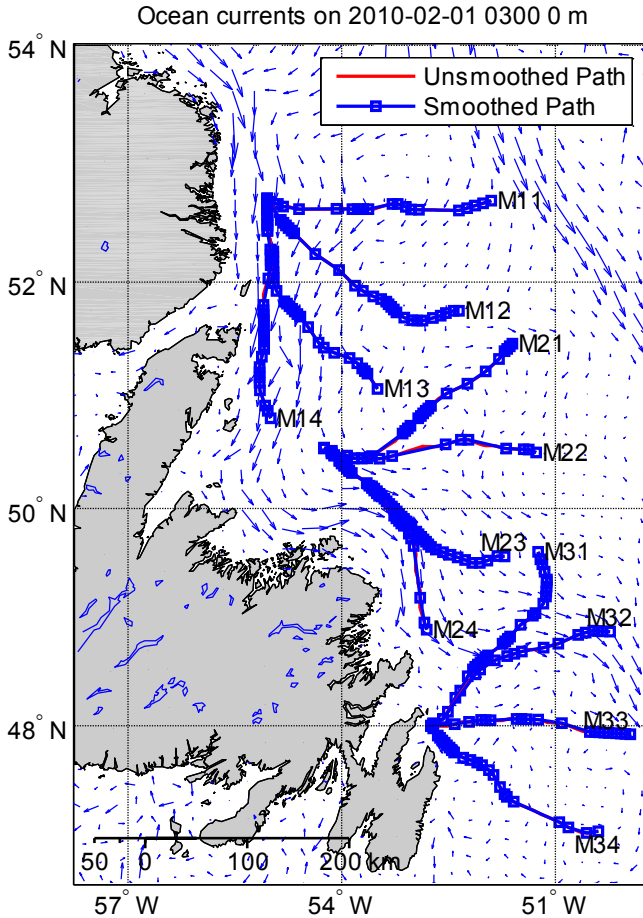


Fig. 13. Time optimal path for different missions along the Newfoundland and Labrador Shelf

References

- [1] M. Eichhorn, A New Concept for an Obstacle Avoidance System for the AUV SLOCUM Glider Operation under Ice, in: Oceans '09 IEEE, Bremen, Germany, 2009.
- [2] M. Eichhorn, Optimal Path Planning for AUVs in Time-Varying Ocean Flows, in: 16th Symposium on Unmanned Untethered Submersible Technology (UUST09), Durham, NH, USA, 2009.
- [3] M. Eichhorn, C. D. Williams, R. Bachmayer, B. deYoung, A Mission Planning System for the AUV SLOCUM Glider for the Newfoundland and Labrador Shelf, in: Oceans '10 IEEE, Sydney, Australia, 2010.
- [4] M. Eichhorn, Solutions for Practice-oriented Requirements for Optimal Path Planning for the AUV SLOCUM Glider, in: Oceans '10 IEEE, Seattle, USA, 2010.
- [5] A. Alvarez, A. Caiti, R. Onken, Evolutionary Path Planning for Autonomous Underwater Vehicles in a Variable Ocean, IEEE Journal of Oceanic Engineering 29 (2) (2004) 418–428.
- [6] H.-j. Wang, J. Zhao, X.-q. Bian, X.-c. Shi, An Improved Path Planner based on Adaptive Genetic Algorithm for Autonomous Underwater Vehicle, in: IEEE International Conference on Mechatronics & Automation, Niagara Falls, Canada, 2005.
- [7] N. K. Yilmaz, C. Evangelinos, P. F. J. Lermusiaux, N. M. Patrikalakis, Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming, IEEE Journal of Oceanic Engineering 33 (4) (2008) 522–537.
- [8] A. Richards, J. P. How, Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming, in: American Control Conference, Anchorage, Alaska, USA, 2002.
- [9] D. Kruger, R. Stolkin, A. Blum, J. Briganti, Optimal AUV path planning for extended missions in complex, fast flowing estuarine environments, in: IEEE International Conference on Robotics and Automation, Rom, Italy, 2007.
- [10] W. Zhang, T. Inanc, S. Ober-Blöbaum, J. E. Marsden, Optimal Trajectory Generation for a Glider in Time-Varying 2D Ocean Flows B-spline Model, in: IEEE International Conference on Robotics and Automation, Pasadena, CA, USA, 2008, pp. 1083–1088.
- [11] G. P. Kladis, J. T. Economou, K. Knowles, J. Lauber, T.-M. Guerra, Energy conservation based fuzzy tracking for unmanned aerial vehicle missions under a priori known wind information, Engineering Applications of Artificial Intelligence 24 (2) (2011) 278–294.
- [12] T. Lolla, M. P. Ueckermann, K. Yigit, J. P. J. Haley, P. F. J. Lermusiaux, Path Planning in Time Dependent Flow Fields using Level Set Methods, in: 2012 IEEE International Conference on Robotics and Automation, RiverCentre, Saint Paul, Minnesota, USA, 2012.
- [13] E. Fernandez-Perdomo, J. Cabrera-Gamez, D. Hernandez-Sosa, J. Isern-Gonzalez, A. C. Dominguez-Brito, A. Redondo, J. Coca, A. G. Ramos, E. A. I. Fanjul, M. Garcia, Path Planning for gliders using Regional Ocean Models: Application of Pinzon path planner with the ESEOAT model and the RU27 trans-Atlantic flight data, in: Oceans '10 IEEE, Sydney, Australia, 2010.
- [14] A. A. Pereira, J. Binney, B. H. Jones, M. Ragan, G. S. Sukhatme, Toward Risk Aware Mission Planning for Autonomous Underwater Vehicles, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, 2011.
- [15] B. Garau, M. Bonet, A. Alvarez, S. Ruiz, A. Pascual, Path Planning for Autonomous Underwater Vehicles in Realistic Oceanic Current Fields: Application to Gliders in the Western Mediterranean Sea, Journal of Maritime Research 6 (2) (2009) 5–22.
- [16] B. Garau, A. Alvarez, G. Oliver, Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach, in: 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005.
- [17] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numerische Mathematik 1 (1) (1959) 269–271.
- [18] A. Orda, R. Rom, Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length, Journal of the Association for Computing Machinery 37 (3) (1990) 607–625.
- [19] K. Yang, S. Sukkarieh, 3D Smooth Path Planning for a UAV in Cluttered Natural Environments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 2008.
- [20] H.-j. Wang, W. Xiong, Research on global path planning based on ant colony optimization for AUV, Journal of Marine Science and Application 8 (1) (2009) 58–64.
- [21] M. Soullignac, P. Taillibert, M. Rueher, Time-minimal Path Planning in Dynamic Current Fields, in: IEEE International Conference on Robotics and Automation, Kobe, Japan, 2009.
- [22] M. Eichhorn, U. Kremer, Opportunities to Parallelize Path Planning Algorithms for Autonomous Underwater Vehicles, in: Oceans '11 IEEE, Kona, HI, USA, 2011.
- [23] M. Eichhorn, An Obstacle Avoidance System for an Autonomous Underwater Vehicle - Taipei, Taiwan, in: Proceedings of 2004 International Symposium on Underwater Technology, 2004, pp. 75–82.
- [24] T. Ersson, X. Hu, Path Planning and Navigation of Mobile Robots in Unknown Environments, in: Intelligent Robots and Systems IEEE/RSJ International Conference, Vol. 2, Maui, HI, USA, 2001, pp. 858–864.
- [25] P. E. Hart, N. J. Nilsson, B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.
- [26] J. G. Siek, L.-Q. Lee, A. Lumsdaine, The Boost Graph Library - User Guide and Reference Manual, 1st Edition, Addison-Wesley, New York, 2002.
- [27] E. Zermelo, Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung, Z. Angew. Math. Mech. 11 (2) (1931) 114–124.
- [28] H.-D. Ebbinghaus, V. Peckhaus, Ernst Zermelo - an approach to his life and work, Springer, Berlin, Heidelberg, 2007.
- [29] P. J. Schneider, D. H. Eberly, Geometric Tools for 3D Graphics, Morgan Kaufmann Publishers, San Francisco, 2003.
- [30] W. Hundsdoerfer, J. Verwer, Numerical solution of time-dependent advection-diffusion-reaction equations, Springer, Berlin, Heidelberg, 2003.
- [31] Unidata, NetCDF (network Common Data Form) Webpage, 2011.
URL <http://www.unidata.ucar.edu/software/netcdf/>
- [32] N. M. Institute, FIMEX Website, 2011.
URL <https://wiki.met.no/fimex/start>
- [33] H. Akima, A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures, Journal of the Association for Computing Machinery 17 (4) (1970) 589–602.
- [34] J. Kiefer, Sequential minimax search for a maximum, Proceedings of the American Mathematical Society 4 (1953) 502–506.
- [35] D. E. Ferguson, Fibonacci searching, Communications of the ACM 3 (12) (1960) 648.
- [36] R. P. Brent, Algorithms for minimization without derivatives, Prentice-Hall, Englewood Cliffs, NJ, 1973, p. Chapter 5.
- [37] M. Cencini, G. Lacorata, A. Vulpiani, E. Zambianchi, Mixing in a Meandering Jet: A Markovian Approximation, Journal of Physical Oceanography 29 (1999) 2578–2594.
- [38] N. Hale, D. R. Moore, A Sixth-Order Extension to the MATLAB Package bvp4c of J. Kierzenka and L. Shampine - technical report, Tech. rep., Oxford University Computing Laboratory (April 2008).
- [39] H. C. Woithe, I. Chigirev, D. Aragon, M. Iqbal, Y. Shames, S. Glenn, O. Schofield, I. Seskar, U. Kremer, Slocum Glider Energy Measurement and Simulation Infrastructure, in: Oceans '10 IEEE, Sydney, Australia, 2010.