PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

OPTIMAL PATH PLANNING SYSTEM 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
phone + 01 709-772-7986; fax + 01 709-772-2462
e-mail: Mike.Eichhorn@nrc-cnrc.gc.ca

ABSTRACT

This paper presents an algorithm for path planning in a time-varying environment based on graph methods. The methods presented make it possible to find an optimal path using defined requirements in a feasible time. The task of the introduced path planning algorithm using an AUV is to find a time-optimal path from a defined start position to a goal position with consideration of the time-varying ocean current. An additional consideration discussed in this paper is 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. This paper is an abridgment of a research fellowship and has been previously published in parts in [1], [2], [3] and [4].

Index Terms **–** Path planning, geometrical graph, graph methods, time-varying environment, AUV, AUV "SLOCUM" glider, autonomous systems

1. INTRODUCTION

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 algorithm presented in this paper is designed considering the mission requirements for the AUV "SLOCUM" glider, which is a particular type of autonomous underwater vehicle (AUV). 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.

M. Eichhorn was with the National Research Council Canada, Institute for Ocean Technology, St. John's Newfoundland. He is now with the Technical University of Ilmenau, Institute of Automation and Systems Engineering, Ilmenau, 98684 Germany, P.O. Box 100565 (phone: +49 3677-691421; fax: +49 3677-691434; e-mail: mike.eichhorn@tu-ilmenau.de).

The presented search algorithm, named A*TVE (time-varying environment) algorithm, is based on a modified Dijkstra Algorithm (see [5] and [6]), 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 [6] 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 inclusion of an A* mechanism [7] in the A*TVE algorithm will be used to reduce the processing time of the search.

A symbolic wavefront expansion (SWE) technique for an Unmanned Air Vehicle (UAV) in time-varying winds was introduced in [8] to find the time optimal path and additionally the optimal departure time. The TVE uses a similar principle as is used in the SWE to calculate the time-varying cost function for the several vertices. 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 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.

2. GEOMETRICAL GRAPH

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. 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

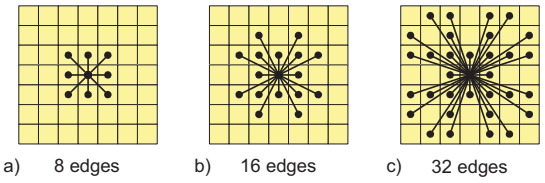

Figure 1. Rectangular grid structure a) 1-sector, b) 2-sector, c) 3-sector

an obstacle scenario with as few of the vertices and edges as possible, and, to decrease the computing time (visibility and quadtree graph [9]). 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 [10]; see Fig. 1(b). The analyses of the found paths in a current field show (see also [1] and [2]) that is it 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.

2.2. Graph-based Search-Algorithm

The described search method in this section is based on a classical A* algorithm [7] which solves the single-source shortest-paths problem on a weighted directed graph. The exact solution by using an A* 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 (nodes) and edges as a result of using small time intervals.

The A* algorithm utilizes the Dijkstra algorithm [5] 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 node u to the goal node g following a straight line based on [11] 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 ocean current velocity $v_{current\_max}$ in the operational area over the full mission time:

$$h(u) = t_{travel} = \frac{\left\| \mathbf{x}_u - \mathbf{x}_g \right\|}{v_{veh\_bf} + v_{current\_max}}. \qquad (1)$$

TABLE I shows a comparison between the A* algorithm (left column) and the A*TVE (time-varying environment) algorithm (right column). The input parameter G contains the graph structure with the vertex list and the edge list (V and E), π is a predecessor list and d is the cost list for each vertex. Q is a priority queue that supports the EXTRACT-MIN and the DECREASE-KEY operations. 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 I highlight the differences between the algorithms. There are the following three differences:

1. The new algorithm doesn't need the weight list w of the edges to begin the search. The algorithm needs a start time t_o 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 cost value of function *wfunc*.
3. The algorithm calculates the weight for the edge $w(u, v)$ in a function *wfunc* during the search. (A detailed description about these calculations will be presented in section 3.) 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 I
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 $u \in V$
$d[u] \leftarrow \infty$	$d[u] \leftarrow \infty$
$f[u] \leftarrow \infty$	$f[u] \leftarrow \infty$
$\pi[u] \leftarrow \infty$	$\pi[u] \leftarrow \infty$
$color[u] \leftarrow$ WHITE	$color[u] \leftarrow$ WHITE
$color[s] \leftarrow$ GRAY	$color[s] \leftarrow$ 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 ($Q \neq \emptyset$)
$u \leftarrow$ EXTRACT-MIN(Q,f)	$u \leftarrow$ EXTRACT-MIN(Q,f)
if ($u = g$)	if ($u = g$)
return (d, π)	return (d, π)
$color[u] \leftarrow$ BLACK	$color[u] \leftarrow$ BLACK
for each $v \in Adj[u]$	for each $v \in Adj[u]$
	if ($d[u] < d[v]$)
$d_v = w(u, v) + d[u]$	$d_v = wfunc(u, v, d[u]) + d[u]$
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 ($color[v]$ = GRAY)	if ($color[v]$ = GRAY)
DECREASE-KEY(Q,v,$f[v]$)	DECREASE-KEY(Q,v,$f[v]$)
else	else
$color[v] \leftarrow$ GRAY	$color[v] \leftarrow$ GRAY
INSERT(Q, v)	INSERT(Q, v)
return (d, π)	return (d, π)

2.3. 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 $n(n\text{-}1)$ 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 II
ALGORITHM FOR PATH SMOOTHING IN TIME-VARYING ENVIRONMENT

PATH-SMOOTHING (WP, t_0)
$TT[1] = t_0$
for (i = 2) **to** (i = length(WP))
  $TT[i] = TT[i-1]$ + TRAVELTIME($WP[i\text{-}1]$, $WP[i]$, $TT[i-1]$)
while (length(WP) ≠ length(WP_{smooth})) AND (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} = length(WP)) **II**
    $merge$ = true
    t_{travel_2} = TRAVELTIME($WP[i_{path}-1]$, $WP[i_{path}]$, $TT[i_{path}-1]$)
    t_{travel_sum} = 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} = length(WP))
        $t_{end} = t_{end}$ + 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)
    $TT[end] = TT[end-1]$ + 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

Table II 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. A 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 through 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. This above described procedure will be repeated until the number of waypoints is constant between two sequent loops (**I**). Fig. 2 shows an example for a better understanding.

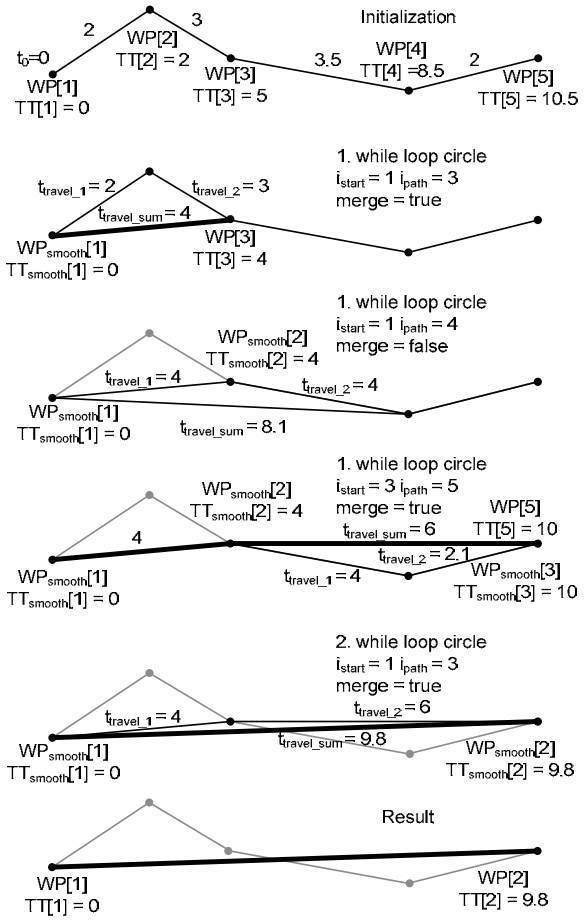

Figure 2. Several steps to smooth a path

3. CALCULATION OF THE COST VALUE

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

3.1. Speed 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.

$$t^i_{path} = \frac{s^i_{path}}{v^i_{path\_ef}} \tag{2}$$

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 $\mathbf{v}^{\mathbf{0}}_{\mathbf{path}}$. This speed can be determined by the intersection point between a line and a circle (2D) and/or sphere (3D) based on Fig. 3 according to the following relation:

$$\begin{aligned} &line\text{:}\ \ \mathbf{x}\left(v_{path\_ef}\right) = v_{path\_ef}\,\mathbf{v}^{\mathbf{0}}_{\mathbf{path}} \\ &circle\,/\,spheres\text{:}\ \ v^2_{veh\_bf} = \left\|\mathbf{x} - \mathbf{v}_{\mathbf{current}}\right\|^2 \end{aligned} \tag{3}$$

$$\begin{aligned} &disc = \left(\mathbf{v}^{\mathbf{0}}_{\mathbf{path}}{}^{\mathrm{T}} \cdot \mathbf{v}_{current}\right)^2 + v^2_{veh\_bf} - \mathbf{v}_{\mathbf{current}}{}^{\mathrm{T}} \cdot \mathbf{v}_{\mathbf{current}} \\ &\text{if } disc > 0 \quad v_{path\_ef} = \mathbf{v}^{\mathbf{0}}_{\mathbf{path}}{}^{\mathrm{T}} \cdot \mathbf{v}_{\mathbf{current}} + \sqrt{disc} \\ &\text{else} \qquad\ \ v_{path\_ef} = NaN \end{aligned} \tag{4}$$

If the discriminant in equation (4) becomes negative, it means that the vehicle can no longer be held in that path; see Fig 4.(a). If the speed v_{path_ef} is negative the vehicle is still on the path, but moving backwards; see Fig 4.(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 course.

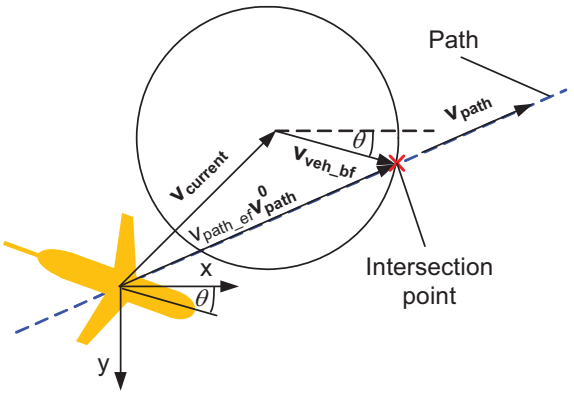

Figure 3. Definition of the velocities

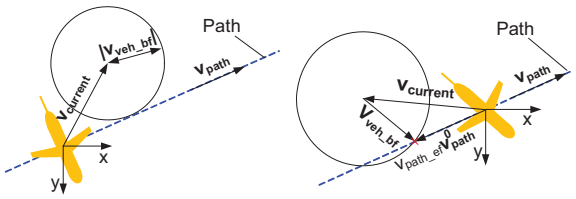

Figure 4. a) Negative discriminant b) $v_{path\_ef} < 0$

3.2. Travel time calculation in time-varying ocean flow

The determination of the travel time according to equation (2) 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 a conjunction with a location-varying ocean current, the speed v_{path_ef} will be changed depending on the current $\mathbf{v}_{\mathbf{current}}$ along the path element.

The idea of the algorithm presented below is based on a step size control for efficient calculation of numerical solutions of differential equations. The step-size h is here not the time as used in numerical solvers but is a segment of the path element. So the path element will be shared within many segments, for which equation (2) can be solved. The calculation of the travel time for a segment includes the following steps (see also TABLE III):

1. Rough approximation of the travel speed v_{path_ef} on the segment in equation (4) using only the current value $\mathbf{v}_{\mathbf{current\_start}}$ from the start point $\mathbf{x}_{\mathbf{start\_local}}$ to the time t_{start_local}.
2. Calculation of the travel time t_{rough_approx} using equation (2) with the calculated speed v_{path_ef} and the length of the segment which correspond with h s_{path}.
3. Determine the ocean current $\mathbf{v}_{\mathbf{current\_end}}$ from the endpoint $\mathbf{x}_{\mathbf{end\_local}}$ to the time $t_{end\_local} + t_{rough\_approx}$.
4. Calculation of an average ocean current $\mathbf{v}_{\mathbf{current\_mean}}$ along the segment by arithmetic mean of the two velocities $\mathbf{v}_{\mathbf{current\_start}}$ and $\mathbf{v}_{\mathbf{current\_end}}$.
5. Improved approximation of the speed v_{path_ef} on the segment used in equation (4) using the mean value $\mathbf{v}_{\mathbf{current\_mean}}$.
6. Calculation of the travel time $t_{improved\_approx}$ using equation (2) with the calculated speed v_{path_ef} and the length of the segment which correspond with h s_{path}.

Afterward occurs the calculation of the local error $error_{local}$ between the two times t_{rough_approx} and $t_{improved\_approx}$ and the determination of the new step-size h_{new} using the equation for an optimal step-size for a second order method:

$$h_{new} = \max\left\{h_{min}, \min\left\{h_{max}, \tau h \sqrt{\frac{\varepsilon}{error_{local}}}\right\}\right\} \tag{5}$$

TABLE III
PSEUDO-CODE OF THE ALGORITHM TO CALCULATE THE TRAVEL TIME

CALC-TRAVELTIME(x_{start}, x_{end}, t_{start})
defined parameters: v_{veh_bf}, h, h_{min}, h_{max}, ε, τ
if ($t_{start} = \infty$) return ($t_{travel} = \infty$)
$t_{start\_local} = t_{start}$
$x_{start\_local} = x_{start}$
$s_{path} = ||x_{end} - x_{start}||$
$\mathbf{v}^{\mathbf{0}}_{\boldsymbol{path}} = (x_{end} - x_{start})/s_{path}$
$s_{travel} \leftarrow 0$
$v_{current\_start}$ = GET-CURRENT(x_{start_local}, t_{start_local})
while ($s_{travel} < s_{path}$)
  $x_{end\_local} = x_{start\_local} + h s_{path} \mathbf{v}^{\mathbf{0}}_{\boldsymbol{path}}$
  v_{path_ef} = CALC-TRAVELVEL ($v_{current\_start}$, $\mathbf{v}^{\mathbf{0}}_{\boldsymbol{path}}$, v_{veh_bf})
  if (($v_{path\_ef} = NaN$) OR ($v_{path\_ef} \leq 0$))
    return ($t_{travel} = \infty$)
  $t_{rough\_approx} = h s_{path}/v_{path\_ef}$
  $v_{current\_end}$ = GET-CURRENT(x_{end_local}, $t_{start\_local} + t_{rough\_approx}$)
  $v_{current\_mean} = 0.5(v_{current\_start} + v_{current\_end})$
  v_{path_ef} = CALC-TRAVELVEL ($v_{current\_mean}$, $\mathbf{v}^{\mathbf{0}}_{\boldsymbol{path}}$, v_{veh_bf})
  if (($v_{path\_ef} = NaN$) OR ($v_{path\_ef} \leq 0$))
    return ($t_{travel} = \infty$)
  $t_{improved\_approx} = h s_{path}/v_{path\_ef}$
  $error_{local} = | t_{rough\_approx} - t_{improved\_approx}|$
  $h_{new} = \max(h_{min}, \min(h_{max}, \tau h \sqrt{\varepsilon / error_{local}}))$
  if (($error_{local} < \varepsilon$) OR ($h_{new} = h_{min}$))
    $v_{current\_start} = v_{current\_end}$
    $x_{start\_local\_} = x_{end\_local}$
    $t_{start\_local} = t_{start\_local} + t_{improved\_approx}$
    $s_{travel} = s_{travel} + h\, s_{path}$
  $h = h_{new}$
  if ($s_{travel} + h s_{path}) > s_{path}$
    $h = (s_{path} - s_{travel})/s_{path}$
return ($t_{travel} = t_{start\_local} - t_{start}$)

The parameter τ is a safety factor ($\tau \in (0, 1]$). Acceptance or rejection of this step will depend on the local error $error_{local}$, the defined tolerance ε the calculated step size h_{new} and the minimal step-size h_{min}.

4. 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.

4.1. 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, 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.

4.2. Algorithm

The detection of the optimal departure time occurs in three steps. Fig. 5 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 [12]. 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.

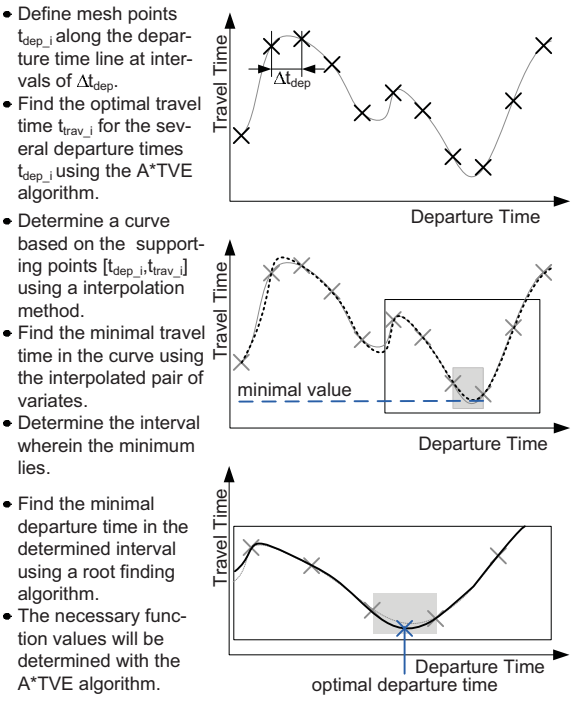

Figure 5. Steps to find the optimal departure time

Here a one-dimensional root-finding algorithm will be used to find the optimal departure time. Thereby a path search using the A*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, Fibonacci search and Brent's algorithm were tested. Brent's algorithm has the best performance and will be favoured.

4.3. Possible Modifications and critical nodes

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. These possibility result in a decrease of the number of examined vertices during the search or lead to a more rapid calculation of the cost function and thus to reduce the processing time.

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 ten 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.

5. CONCLUSION AND FUTURE WORK

In this paper an algorithm for path planning in a time-varying environment based on graph methods is 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. 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. A future research topic is the analyses of possibilities for parallelization and programmable implementation using the Task library, a component of the Boost Sandbox.

6. ACKNOWLEDGMENT

This work was financed by the German Research Foundation (DFG) within the scope of a two-year research fellowship. 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.

7. REFERENCES

[1] M. Eichhorn, "A New Concept for an Obstacle Avoidance System for the AUV "SLOCUM Glider" Operation under Ice", Oceans '09 IEEE Bremen, 2009.

[2] M. Eichhorn, "Optimal Path Planning for AUVs in Time-Varying Ocean Flows" 16th Symposium on Unmanned Untethered Submersible Technology (UUST09), 2009.

[3] M. Eichhorn et al., "A Mission Planning System for the AUV "SLOCUM Glider" for the Newfoundland and Labrador Shelf" Oceans '10 IEEE Sydney, 2010.

[4] M. Eichhorn, "Solutions for Practice-oriented Requirements for Optimal Path Planning for the AUV "SLOCUM Glider"" Oceans '10 IEEE Seattle, 2010.

[5] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, no. 1, 1959, pp. 269-271.

[6] A. Orda and R. Rom, "Shortest-Path and Minimum-Delay Algorithms in Networks with Time-Dependent Edge-Length" Journal of the Association for Computing Machinery, vol. 37, no. 3, 1990, pp. 607-625.

[7] 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. 4, no. 2, 1968, pp. 100-107.

[8] M. Soulignac, P. Taillibert and M. Rueher, "Time-minimal Path Planning in Dynamic Current Fields" IEEE International Conference on Robotics and Automation, 2009.

[9] M. Eichhorn, "An Obstacle Avoidance System for an Autonomous Underwater Vehicle" in Proc. Proceedings of 2004 International Symposium on Underwater Technology, pp. 75-82, Taipei, Taiwan, 20-23 April 2004.

[10] T. Ersson and X. Hu, "Path Planning and Navigation of Mobile Robots in Unknown Environment," in Proc. Intelligent Robots and Systems IEEE/RSJ International Conference, pp. 858-864, Maui, HI, USA, 29 Oct to 03 Nov, 2001.

[11] E. Fernández-Perdomo et al., "Path Planning for gliders using Regional Ocean Models" *Oceans '10 IEEE Sydney,* 2010.

[12] H. Akima, "A New Method of Interpolation and Smooth Curve Fitting Based on Local Procedures", Journal of the Association for Computing Machinery, vol. 17, no. 4, 1970, pp. 589-602