Optimal Path Planning in Time-Varying Flows Using Adaptive Discretization

Article · October 2017

DOI: 10.1109/LRA.2017.2761939

3 authors, including:

Dhanushka Kularatne
University of Pennsylvania
13 PUBLICATIONS **39** CITATIONS

SEE PROFILE

Optimal Path Planning in Time-Varying Flows using Adaptive Discretization

Dhanushka Kularatne[1], Subhrajit Bhattacharya[2] and M. Ani Hsieh[3]

***Abstract*—Autonomous marine vehicles (AMVs) are typically deployed for long periods of time in the ocean to monitor different physical, chemical, and biological processes. Given their limited energy budgets, it makes sense to consider motion plans that leverage the dynamics of the surrounding flow field so as to minimize energy usage for these vehicles. In this paper, we present a graph search based method to compute energy optimal paths for AMVs in two-dimensional (2-D) time-varying flows. The novelty of the proposed algorithm lies in the use of an adaptive discretization scheme to construct the search graph. We demonstrate the proposed algorithm by computing optimal energy paths using an analytical time-varying flow model and using time-varying ocean flow data provided by the Regional Ocean Model System. We compare the output paths with those computed via an optimal control formulation and numerically demonstrate that the proposed method can overcome problems inherent in existing fixed discretization schemes.**

***Index Terms*—Motion and Path Planning, Marine Robotics, Field Robots, Optimization in Time-Varying Flows, Graph Based Planning**

I. INTRODUCTION

SCIENTIFIC activities such as migration tracking, characterizing the dynamics of plankton assemblages, measurement of temperature profiles, and monitoring of harmful algae blooms [1] are increasingly being automated using autonomous marine vehicles (AMVs) which can be either surface or underwater vehicles. In these applications, AMVs are often deployed over long periods while operating with limited energy budgets. As such, researchers have to design motion strategies that are energy efficient to maximize the efficacy of these autonomous platforms.

While the high inertia environment of the ocean couples the environmental dynamics to the marine vehicle dynamics, it presents a unique opportunity for vehicles to exploit the surrounding flows for more efficient navigation. As such, there is a substantial amount of recent work on determining optimal paths in flow fields. Existing work include using graph search methods to plan time [2] and energy [3, 4, 5] optimal paths in static flow fields. In [6], the authors present a tree based method to compute optimal paths in a time-varying wind field. The tree like structure, however, results in the expansion of a large number of nodes which requires significant computation resources. Eichorn also presented a graph based method, but it is limited to computing optimal time paths in time-varying flows [7]. In existing graph based techniques, optimal paths in time-varying flows are obtained using a fixed resolution in the discretization scheme for the graph construction. However, as we will show in this paper, such schemes lead to incorrect results if this discretization resolution is not chosen appropriately. In [8], the authors provide an any-time method to compute time optimal paths, that seeks to overcome such discretization errors. The approach utilizes a (N+1) dimensional grid (N=2 for 2D, N=3 for 3D) to represent the spatio-temporal workspace. Discretization errors are reduced by using a local optimizer to find intermediate points on hyper-edges (lines or/and surfaces) of the grid that minimizes the overall cost. While this method can be extended to work with an optimal energy cost function, the local optimizer will then have to search in a (space x control) space in order to find intermediate points that minimize the overall cost. Such a strategy will significantly increase the computation requirements. In contrast, our approach solves both optimal energy as well as optimal time problems and reduces discretization errors by adapting the discretization resolution locally to better match the underlying flow dynamics.

Alternatives to graph search techniques include [9, 10] for computing energy optimal paths in time-varying flows. Since these methods are based on iterative minimization techniques, they run the risk of producing paths that are only locally optimal. Lolla *et al.*[11] presented a level set expansion method to find time optimal paths in time-varying flows. This was then extended by Subramani *et al.*[12] to determine the energy optimal paths from the set of time optimal paths obtained from the level set method. Similar to many existing approaches, level set approaches require full knowledge of the flow field and require significant computational resources for the various level set expansions at each iteration.

In this paper, we present a graph based approach to compute optimal paths in time-varying flow fields. Advantages of graph based versus other techniques include: 1) the ease of incorporating both actuation constraints and holonomic

Manuscript received: May, 09, 2017; Revised August, 10, 2017; Accepted September, 19, 2017.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Science Foundation (NSF) grants IIS-1253917 and CMMI-1462825.

[1]Dhanushka Kularatne is with the Scalable Autonomous Systems Lab, Drexel University, Philadelphia, PA 19104, USA. dnk32@drexel.edu

[2]Subhrajit Bhattacharya is with the Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA 18015, USA. sub216@lehigh.edu

[2]M. Ani Hsieh is with the Department of Mechanical Engineering & Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104, USA. m.hsieh@seas.upenn.edu

 Digital Object Identifier (DOI): 10.1109/LRA.2017.2761939.

constraints imposed by static and dynamic obstacles when compared to full blown optimization schemes (e.g., optimal control formulations); 2) the ease of implementation when compared to existing energy optimal path planning strategies in time-varying flows, *i.e.,* iterative techniques, level set methods; and 3) the ease of extending graph based techniques to include topological constraints such as homotopy classes [5, 13, 14] which are relevant for environmental exploration and monitoring applications. Similar to existing graph based methods, we represent the environment as a discrete graph and use a search algorithm to find an optimal path in the graph. The fundamental difference of our method lies in the adaptive discretization scheme used in the construction of the graph. In our method, the discretization resolution is selected locally according to the flow characteristics at each point. Specifically, the discretization resolution is selected such that the flow velocity error remains bounded along an edge. Thus, if the flow changes rapidly at a point, the resolution will be made finer, and vice versa. The result is the first graph based method that uses adaptive discretization to compute optimal paths in time-varying flows. While this work focuses on optimizing the path energy cost, the presented method is general enough to handle other time-dependent cost functions.

The rest of the paper is organized as follows: the problem formulation is given in section II, our graph based solution is presented in section III, the simulation results are presented in section IV, and conclusions and directions for future work are discussed in section V.

II. PROBLEM FORMULATION

A. Environment and Flow Model

We consider a 2-D aquatic environment $\mathbb{W} \subseteq \mathbb{R}^2$, subject to a time-varying flow field $\mathbf{V_f} : \mathbb{W}_T \mapsto \mathbb{R}^2$, where $\mathbb{W}_T = \mathbb{W} \times [\underline{t},\ \bar{t}]$ and $[\underline{t},\ \bar{t}] \subset \mathbb{R}_{\geq 0}$ denotes the time interval under consideration. As such, for $\mathbf{x} \in \mathbb{W}$ and $t \in [\underline{t},\ \bar{t}]$, $\mathbf{V_f}(\mathbf{x},t) = \left[V_{fx}(\mathbf{x},t),\ V_{fy}(\mathbf{x},t)\right]^T$ denotes the flow velocity, with V_{fx} and V_{fy} denoting the components of the flow vector in the inertial frame. The speed of the flow is given by $V_f(\mathbf{x},t) = \|\mathbf{V_f}(\mathbf{x},t)\|$ and the maximum flow speed encountered in the domain is given by $V_{fm} = \max_{\mathbf{x}\in\mathbb{W},\ t\in[\underline{t},\ \bar{t}]} V_f(\mathbf{x},t)$. It is assumed that the flow description is known a priori or a reliable forecast is available. It is also assumed that, information about static and dynamic obstacles in the environment is encoded by a function $\mathbb{O} : \mathbb{W}_T \mapsto \{true, false\}$, where $\mathbb{O}(\mathbf{x},t) = true$ indicates an obstacle at $(\mathbf{x},t)$ and vice versa.

B. Vehicle Model

We assume a holonomic kinematic model for the autonomous marine vehicle (AMV). This is a reasonable assumption when the dimensions of the AMV are small when compared with the dimensions of the flow structures. Using this model, the net velocity of the vehicle with respect to the inertial frame is given by

$$\mathbf{V_{net}}(\mathbf{x},t) = \mathbf{V_f}(\mathbf{x},t) + \mathbf{V_{still}}(\mathbf{x},t), \tag{1}$$

where $\mathbf{V_{still}}$ is the velocity of the vehicle with respect to the flow, *i.e.,* $\mathbf{V_{still}}$ is the "thrust" vector of the vehicle. To achieve a given velocity $\mathbf{V_{net}}$, the AMV speed with respect to the flow needs to be

$$V_{still} = \sqrt{(V_{net} - V_f \cos\theta)^2 + (V_f \sin\theta)^2} \tag{2}$$

where $V_{net} = \|\mathbf{V_{net}}\|$, $V_f = \|\mathbf{V_f}\|$, $V_{still} = \|\mathbf{V_{still}}\|$ and θ is the angle between $\mathbf{V_f}$ and $\mathbf{V_{net}}$.We further assume that that the actuation capability of the vehicle is limited and that its maximum speed is lower than the speed of the surrounding flow *i.e.,* $V_{still}(\mathbf{x},t) \leq V_{max} < V_{fm}$.

C. Problem Statement

Given a path cost function, the objective is to find a path $\Gamma : [t_s,\ t_g] \mapsto \mathbb{W}$ that minimizes the total cost of travel between specified start and goal locations. Thus the problem addressed in this paper is finding a solution to the following optimization problem,

$$\Gamma^* = \underset{\Gamma}{\operatorname{argmin}}\ C(\Gamma) \tag{3}$$

subject to,

$$\Gamma(t_s) = \mathbf{x}_s, \quad \Gamma(t_g) = \mathbf{x}_g, \quad t_s < t_g, \quad V_{still}(\Gamma(t),t) \leq V_{max}$$

where $C(\Gamma) \geq 0$ is the given cost function, $\mathbf{x}_s$ and $\mathbf{x}_g$ are the desired start and goal positions, and $t_s, t_g \in [\underline{t},\ \bar{t}]$ are the start and end times respectively. Note that while t_s is specified explicitly, t_g is free.

The cost function $C(\Gamma)$ can be any function that can be written as

$$C(\Gamma) = \int_\Gamma dc = \int_{t_s}^{t_g} E\left(\Gamma(t),t\right) dt \tag{4}$$

where $dc = E\left(\Gamma(t),t\right) dt \geq 0$ is the cost of an infinitesimal path segment spanning a time interval dt. The incremental cost dc will be dependent on the flow velocity encountered along that path segment as well as the control V_{still} selected for that path segment. In this paper we will focus on a cost function that represents the energy consumed by the AMV. However, the graph based solution method described in section III can handle any cost function of the form given in (4).

D. Cost Function

As mentioned before, we consider a cost function that represents the energy consumption of the AMV. The total energy consumed by the AMV is considered to be $E_{total} = E_{hotel} + E_{drag}$, where E_{hotel} is the energy required to operate the vehicle's computing and sensor systems independent of propulsion [15], and E_{drag} is the energy expended to overcome drag forces exerted by the fluid. Assuming a constant power usage K_h by the computing and sensor systems gives $E_{hotel} = \int_{t_s}^{t_g} K_h dt$. The drag force F_d encountered by the AMV along a path Γ is given by $F_d(t) = K_d V_{still}^{\alpha-1}(\Gamma(t),t)$ where K_d is the drag coefficient and $\alpha \in \{2,3,...\}$. If $\alpha = 2$ the drag is linear, if $\alpha = 3$ the drag is quadratic, and so on. This leads to $E_{drag} = \int_{t_s}^{t_g} K_d V_{still}^{\alpha}(\Gamma(t),t) dt$. Thus, the cost of a path is given by

$$C(\Gamma) = \int_{t_s}^{t_g} K_h + K_d V_{still}^{\alpha}(\Gamma(t),t) dt, \tag{5}$$

and the cost of a small path segment $[dx,dy]^T$, traversed in time dt is given by,

$$dc = (K_h + K_d V_{still}^{\alpha}(\Gamma(t),t))\,dt. \tag{6}$$

where V_{still} is given by (2) with $V_{net} = \|[\frac{dx}{dt},\ \frac{dy}{dt}]^T\|$. The implicit assumption made is that the flow velocity $\mathbf{V_f}$ remains constant within dx,dy and dt. Note that, K_h and K_d can also be thought of as weighting parameters between minimum time paths and minimum energy paths. If a minimum time path is required, we could set $K_d = 0$ and proceed, and vice versa. If exact energy minimization is required, actual values for K_h and K_d should be used.

III. METHODOLOGY

A. Preliminaries

We use a graph based approach to find a solution to (3). We use a discrete graph $\mathscr{G} = (\mathscr{V},\ \mathscr{E})$ to represent the workspace $\mathbb{W}_T$, where $\mathscr{V}$ is the vertex set and $\mathscr{E}$ is the edge set. Each $v_i \in \mathscr{V}$ represents a point in $\mathbb{W}_T$ and is identified by the pair $(\mathbf{x_i}, t_i)$ where $\mathbf{x_i} \in \mathbb{W}$ and $t_i \in [\underline{t},\ \bar{t}]$. Each $e_{ij} \in \mathscr{E}$ represents a directed edge from v_i to v_j and has an associated traversal cost given by (6). In computing the the edge cost, we assume that the flow velocity along the edge remains constant at $\mathbf{V_f}(v_i)$, the flow at the base of the edge. Note that with a little abuse of notation, we sometimes represent the flow velocity at a vertex v_i with $\mathbf{V_f}(v_i)$. In this context, a path Γ_G from the start vertex $v_s = (\mathbf{x_s}, t_s)$ to the goal vertex $v_g = (\mathbf{x_g}, t_g)$ is a sequence of nodes $\{v_0, v_1, \cdots, v_N\}$ with $v_0 = v_s$ and $v_N = v_g$ and $t_i < t_j$ for $i < j$. The path cost is obtained by summing up the cost of the edges (v_i, v_{i+1}) that make up the path.

Given a start and a goal vertex in the graph, we use the A* algorithm [16] to find the shortest path in the graph connecting the vertices. In order to do this, an admissible heuristic function, $h : \mathscr{V} \to \mathbb{R}_{\geq 0}$, is required. For the optimal energy cost function given in (5), an admissible heuristic function is given by

$$h(v) = (K_h + K_d v_{still}^{\alpha})\,\frac{\|\mathbf{x_g} - \mathbf{x}\|}{v_{still} + V_{fm}}, \tag{7}$$

where V_{still} is obtained by solving, $K_d(\alpha - 1)V_{still}^{\alpha} + K_d \alpha V_{still}^{\alpha-1} - K_h = 0$. Details of this derivation are given in the Appendix.

In the literature, common practice is to discretize the workspace uniformly to construct the graph [8, 17]. However, we believe that during graph construction, the discretization should change as a function of the underlying flow.

To illustrate the need for adaptive discretization in graph based methods for planning optimal paths in time-varying flows, consider the wind-driven double gyre flow model given by $V_{fx}(x,t) = -\pi A \sin\big(\pi f(x,t)\big)\cos(\pi y)$ and $V_{fy}(x,t) = -\pi A \cos\big(\pi f(x,t)\big)\sin(\pi y)\frac{\partial f(x,t)}{\partial x}$ where $f(x,t) = \epsilon \sin(\omega t)x^2 + \big(1 - 2\epsilon\sin(\omega t)\big)x$. In this model, A determines the maximum speed of the flow field, and, ϵ and ω respectively determine the amplitude and the frequency of the time-varying oscillation of the flow field.

Lets consider three cases where we use a time extended version of the graph search method described in [5] to compute optimal energy paths. In [5], the spatio-temporal workspace is uniformly discretized using fixed intervals Δx and Δt. In the first two cases, we select the flow field parameters to be $A = 1$, $\omega = 4\pi$ and $\epsilon = 0.1$. For the third case, we set $\epsilon = 0.6$ while leaving the rest of the parameters unchanged. This results in a maximum flow speed of $V_{fm} = 3.74$m/s and $V_{fm} = 6.73$m/s for the first two cases and the third case respectively. However, in both cases the average flow speed is ≈ 2m/s. Lastly, in all three cases we set the maximum vehicle speed as $V_m = 2$m/s.

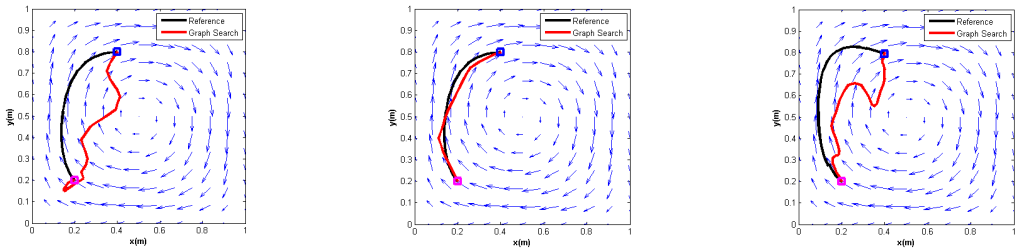
In case 1, a discretization resolution of $\Delta x = 0.01m$, and $\Delta t = 0.1s$ was used for the graph search and the results are shown in red in Fig. 1(a). It can be clearly seen that the resulting path does not match the expected path shown in black. One reason for this discrepancy is due to Δt being set too high with respect to Δx, resulting in an overly slow edge traversal speed (≈ 0.1m/s). For case 2, Δt was set to $0.01s$ so as to increase the edge traversal speed. The results are shown in Fig.1(b) which is much closer to the optimal path. Thus, it is clear that the size of Δx and Δt has to be selected according to the local flow speeds to yield better results. Case 3 further supports this assertion. While the larger ϵ value in Case 3 (0.1 vs. 0.6) does not affect the average flow speed, it results larger spatio-temporal variations within each region of the flow field. Since the encountered flow speeds are of similar magnitudes for both cases 2 and 3, one would expect similar results if the same discretization resolutions were used. However, Fig. 1(c) shows that the resulting path computed for case 3 does not match the expected path. This mismatch is a result of the large spatio-temporal variations of the flow within the volume of space-time centered at each vertex, which violates the assumption that the flow velocity remains constant along an edge of the graph. As such, appropriate selection of Δx and Δt that ensures that the local flow remains relatively constant, is crucial for obtaining good results. This is especially true in highly turbulent regions of the flow field where the discretization resolution has to be made finer to account for the larger changes over the same space-time volume. As such, in our approach, we employ an adaptive discretization scheme that explicitly considers both the local flow velocity and the local variation of the flow velocity.

B. Approach

1) Adaptive Discretization: The implicit assumption in any graph based method for computing optimal paths in time-varying flows is that the flow velocity remains constant along the edges of a graph. Thus, in order to overcome the issues highlighted in the example above, the discretization resolution should be selected to be small enough so that the actual flow velocity variation along an edge in the graph is small.

Consider an edge $e_{ij} = (v_i v_j)$ from node $v_i = (\mathbf{x_i}, t_i)$ to node $v_j = (\mathbf{x_j}, t_j)$ with $\mathbf{x_i} = [x_i,\ y_i]^T$, $\mathbf{x_j} = [x_i + dx,\ y_i + dy]$ and $t_j = t_i + dt$. We assume the flow velocity at v_j can be approximated by a first order Taylor series expansion v_i given by

$$\left[\begin{array}{c} V_{fx} \\ V_{fy} \end{array}\right]\Bigg|_{v_j} = \left[\begin{array}{c} V_{fx} \\ V_{fy} \end{array}\right]\Bigg|_{v_i} + \left[\begin{array}{ccc} \frac{\partial V_{fx}}{\partial x} & \frac{\partial V_{fx}}{\partial y} & \frac{\partial V_{fx}}{\partial t} \\ \frac{\partial V_{fy}}{\partial x} & \frac{\partial V_{fy}}{\partial y} & \frac{\partial V_{fy}}{\partial t} \end{array}\right]\Bigg|_{v_i} \left[\begin{array}{c} dx \\ dy \\ dt \end{array}\right]. \tag{8}$$



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 1. Expected optimal path (in black) vs computed optimal path (in red). Case1: $\Delta x = 0.01m$, and $\Delta t = 0.1s$, results in erroneous results as the selected discretization does not match the flow speeds encountered. Case 2: $\Delta x = 0.01m$, and $\Delta t = 0.01s$ results in a better result. Case 3: Flow changed by setting $\epsilon = 0.6$, same discretization as case 2. However, incorrect results because the discretization does not consider the spatio-temporal variation of the flow.

Since we assume the flow remains constant along an edge, the error in the velocity along an edge is given by,

$$\mathbf{V_e} = \left[\begin{array}{c} V_{fx} \\ V_{fy} \end{array} \right]\Bigg|_{v_j} - \left[\begin{array}{c} V_{fx} \\ V_{fy} \end{array} \right]\Bigg|_{v_i} = \tilde{\nabla}\mathbf{V_f}(v_i) \left[\begin{array}{c} dx \\ dy \\ dt \end{array} \right] \tag{9}$$

where $\tilde{\nabla}\mathbf{V_f}(v_i)$ is the derivative of the flow vector at v_i. Thus, the magnitude of the velocity error due to a displacement $d\mathbf{x_t} = [dx,\, dy,\, dt]^T$ is,

$$\begin{aligned} \|\mathbf{V_e}\| &= \sqrt{\mathbf{V_e}^T \mathbf{V_e}} \\ &= \sqrt{d\mathbf{x_t}^T \mathbf{H}(v_i) d\mathbf{x_t}} \end{aligned} \tag{10}$$

where $\mathbf{H}(v_i) = \tilde{\nabla}\mathbf{V_f}^T(v_i)\tilde{\nabla}\mathbf{V_f}(v_i)$ is a symmetric matrix. As such, it can be shown that,

$$\|\mathbf{V_e}\| = \sqrt{d\mathbf{x_t}^T \mathbf{H}(v_i) d\mathbf{x_t}} \leq \lambda_{max}(v_i)\|d\mathbf{x_t}\| \tag{11}$$

where $\lambda_{max} > 0$ is the maximum eigenvalue of $\mathbf{H}(v_i)$. This maximum error occurs when $d\mathbf{x_t}$ is parallel to $\mathbf{v_{max}}$ which is the direction of the eigenvector corresponding to λ_{max} (see Fig. 2(a)). Thus, to have a relatively constant flow along the edge, we require,

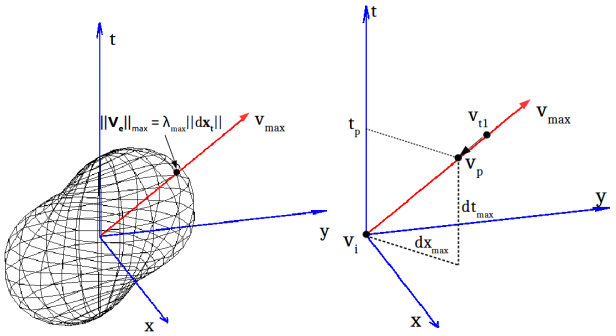
$$\|\mathbf{V_e}\| \leq \lambda_{max}(v_i)\|d\mathbf{x_t}\| \leq p\|\mathbf{V_f}(v_i)\| \tag{12}$$

where $0 < p < 1$ is a ratio that specifies the magnitude of the allowable error in terms of the flow at the base of the edge. To ensure (12) is satisfied by any edge emanating from v_i, we require

$$\|d\mathbf{x_t}\| \leq dx_{t_{max}} = \frac{p\|\mathbf{V_f}(v_i)\|}{\lambda_{max}(v_i)}. \tag{13}$$

Since the edge length limit $dx_{t_{max}}$ in (13) is computed using a first order approximation of the error, it will not be correct for large $\|d\mathbf{x_t}\|$. To address this, we use Newton's root finding method along the $\mathbf{v_{max}}$ direction, with $v_{t_1} = v_i + dx_{t_{max}}\mathbf{v_{max}}$ as the initial point, to find the point $v_p = (\mathbf{x_p}, t_p)$ along $\mathbf{v_{max}}$, that has a velocity error very close to $p\|\mathbf{V_f}(v_i)\|$ (see Fig. 2(b)). The maximum allowable spatial and temporal displacements at node v_i, dx_{max} and dt_{max} are then respectively selected as

$$dx_{max} = \|\mathbf{x_p} - \mathbf{x_i}\|, \quad dt_{max} = |t_p - t_i| \tag{14}$$

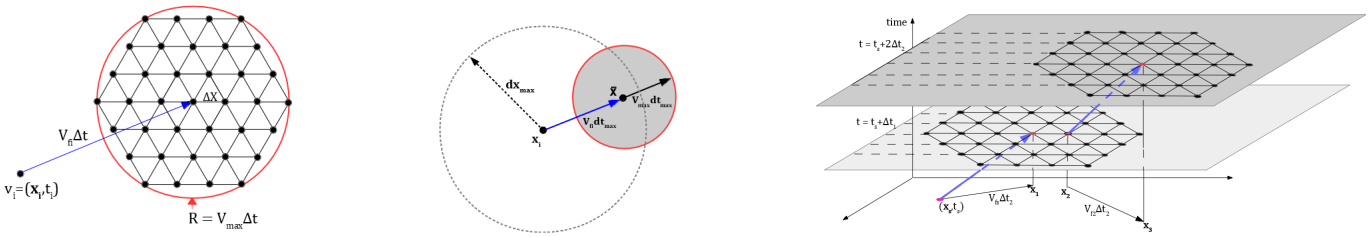


(a) (b)

Fig. 2. (a) Velocity error magnitude surface at v_i. The distance of each point on the surface from v_i represents the magnitude of the error in that direction. The maximum velocity error occurs along $\mathbf{v_{max}}$. (b) Newton's root finding method is used along the $\mathbf{v_{max}}$ direction to find the point v_p that has a velocity error of exactly $p\|\mathbf{V_f}(v_i)\|$.

2) Graph Construction: In case 1 of the illustrative example, we showed that the spatial and temporal discretization given by Δx and Δt need to be selected according to the speed of the flow. If Δx and Δt were chosen such that the resulting edge traversal speed ($\approx \Delta x/\Delta t$) is either too fast or too slow, the results would be incorrect. Here, we present the adaptive Single Time-step Search (aSTS) method which explicitly considers the local flow speeds, in addition to the limits established in (14)when selecting the discretization resolution. Our method ensures that the flow velocities remain relatively constant along edges, and that the resulting edge traversal speeds are commensurate with the underlying flow speeds.

During graph construction, the aSTS method considers the region of space that could be reached from a given node v_i in a single time step Δt, under the influence of both the vehicle actuation and the flow velocity at v_i. In 2D space, this reachable space is demarcated by a circle of radius $V_{max}\Delta t$ centered at $\bar{\mathbf{x}} = \mathbf{x_i} + \mathbf{V_f}(v_i)\Delta t$. In the aSTS graph, this reachable space is represented by a hexagonal lattice of vertices, centered at $\bar{\mathbf{x}}$ with $2n+1$ vertices along the main axis (see 3(a)). The $m = 3n^2 + 3n + 1$ number of vertices in this lattice are added to the neighbor set $\mathcal{N}(v_i)$ of v_i. For each $v_j \in \mathcal{N}(v_i)$, an edge $e_{ij} = (v_i, v_j)$ and a vertex v_j is added to the graph if $\mathbb{O}(v_j) = false$, *i.e.,* if the vertex is not obstructed by an obstacle. All the vertices in $\mathcal{N}(v_i)$ will have the same time coordinate $t_j = t_i + \Delta t$. The inter-vertex spacing of the neighbor lattice is $\Delta x = V_{max}\Delta t/n$. Note that, the $\mathbf{V_{still}}$ value required



(a) (b) (c)

Fig. 3. (a) The reachable space from v_i is a circle of radius $V_{max}\Delta t$ centered at $\bar{\mathbf{x}}$. A hexagonal lattice of vertices is used to represent this space in the graph. In this case $n=3$. (b) In this case $(V_f+V_{max})dt_{max} > dx_{max}$, as such Δt should be selected as $\Delta t = \frac{dx_{max}}{V_f+V_{max}}$ according to (15). (c) Construction of the graph using the aSTS method. All the vertices reachable within a single time step from the base node are considered as neighbors.

to reach each vertex in $\mathcal{N}(v_i)$ is the same for any v_i due to the way the graph is constructed, and thus they can be precomputed.

The time step Δt at each v_i is selected according to the conditions given in (14). The maximum spatial distance between v_i and any $v_j \in \mathcal{N}(v_i)$ is $(V_f+V_{max})\Delta t$ (at the farthest point on the reachable space). From the discussion in section III-B1, we want $\Delta t \leq dt_{max}$ and $(V_f+V_{max})\Delta t \leq dx_{max}$. Thus, for each v_i, we select Δt according to the following rule

$$\Delta t = \begin{cases} dt_{max}, & (V_f+V_{max})dt_{max} \leq dx_{max} \\ \frac{dx_{max}}{V_f+V_{max}}, & otherwise \end{cases}. \qquad (15)$$

Selection of Δt according to (15) ensures that the edges added to the graph at each node expansion, satisfy the conditions specified in (14). The graph is constructed by repeating this process at each node expansion (see Fig. 3(c)), and the node expansion is guided by the A* algorithm where the heuristic given in (7) is used for sorting nodes. Note that in highly turbulent regions of the flow, Δt will be set to a very small value, which will result in a very fine spatio-temporal discretization. The aSTS method guarantees that,

1) maximum velocity error along any edge is less than $pV_f(v_i)$,
2) the net vehicle speed along any edge is commensurate with the flow velocity at the base of the edge.

Note that, the only user selected parameters in this method are the edge velocity variation limit p and the number of neighbors along the main axis of the lattice n. All discretization levels are computed automatically based on these values during runtime.

The pseudo code for the aSTS method is given in Algorithm 1. The $getLatticeVel(V_{max}, n)$ function (line 2), precomputes the $\mathbf{V_{still}}$ values required to reach each node in the lattice. All un-expanded nodes are stored in a priority queue (Q) which is sorted using the $f = cost + heuristic$ values, and it is initialized with the start node. Parent of each node points to the vertex through which the least cost path from the start vertex arrives at the current vertex. The start vertex is initialized with a null parent, and the least cost path to the goal is constructed by successively stepping back through the parents starting from the goal node (line 9). Function $getMaxErrDir(v_i)$ (line 12) returns the maximum error direction $\mathbf{v_{max}}$ at node v_i, and $getDispLimits(v_i, \mathbf{v_{max}}, p)$ returns the point v_p along $\mathbf{v_{max}}$ which gives a velocity error of $p * V_f(v_i)$. If v_j has already been added to the graph (line 25), its cost and parent are updated if the vertex can be reached with a lower cost through the current vertex v_i (line 27). When searching the graph to find if v_j is already in the graph, any node v_k with $(t_k - t_j) < \Delta t/2$ and $\|\mathbf{x_k} - \mathbf{x_j}\| < \Delta x/2$ is considered to be the same as v_j. If v_j is not in the graph, its cost and parents are updated (line 32) and added to the graph. The processes is repeated until the goal is reached.

3) Complexity Analysis: In our analysis, we assume that the priority queue Q is implemented as a min-priority queue using a heap data structure. Thus, for a queue of length n_q, each of the $extractMin()$, $update(q_j)$, & $insert(q_j)$ operations has a worst case running time of $\mathbb{O}(\log n_q)$. We also assume that all other value assignment operations can be performed in constant time. Thus, we only consider the $extractMin()$, $update(v_j)$, & $insert(v_j)$ operations contained inside the *while* loop, in our analysis since all other operations can be done in constant time. The adaptive discretization steps in lines 12 and 13 are also considered as constant time operations since they are independent of the number of node expansions. However, it should be noted that while this step is not the dominating factor in the computational complexity, it does add a computational overhead when compared with fixed discretization methods.

Lets assume that N nodes are expanded by the algorithm to find a path Γ, and m neighbors are considered at each node, *i.e.,* $m = \|\mathcal{N}(v_i)\|$. Thus, for each node expanded, the $extractMin()$ operation is performed exactly once, and one of the $update(v_j)$ or $insert(v_j)$ operations is performed at the most m times. Furthermore, in the worst case, there are Nm number of nodes in the priority queue Q. As such, the worst case running time of the algorithm is $\mathbb{O}(Nm \log Nm)$.

Since, nodes in Q are sorted using their f values, N will depend on the efficacy of the heuristic in (7), and also on the number of vertices created during graph construction. Smaller p values will result in smaller Δt values, which will increase the number of segments/steps in a path. Furthermore, increasing n will increase the number of neighbors (m) considered at each step. Thus, decreasing p and increasing n will result in increased running times, while decreasing path error. For a given p and n, we can obtain an upper bound for the number of segments (n_p) in the path as $n_p \leq (\bar{t} - t_s)/\Delta t_{min}$, where Δt_{min} is the smallest timestep computed using (14). Thus in the worst case, where the heuristic = 0, and all nodes have to be expanded before getting to the goal, the maximum number of nodes expanded is $N = m^{n_p}$, in which case, the worst case running time is $\mathbb{O}((n_p+1)m^{n_p+1}\log(m))$.

Algorithm 1: Adaptive Single Timestep Search (aSTS)

Input : Start $v_s = (\mathbf{x}_s, t_s)$, Goal $\mathbf{x}_g$, p, n, V_{max}, Flow $\mathbf{V_f}$
Output: Optimal cost path Γ

$Q = \varnothing, \quad \mathscr{G} = \varnothing$
$\mathbf{V}_{lattice} = getLatticeVel(V_{max}, n)$
$v_s.f = 0, \quad v_s.cost = 0 \quad v_s.parent = \varnothing$
$Q.insert(v_s), \quad \mathscr{G}.addNode(v_s)$
while $(Q != \varnothing)$ **do**
  $v_i = Q.extractMin()$
  $v_i.expanded = true$
  if $(\|\mathbf{x_i} - \mathbf{x}_g\| < \delta x)$ **then** // goal is reached
    $\Gamma = retrievePath(v_i)$
    break;
  end
  $\mathbf{v_{max}} = getMaxErrDir(v_i)$
  $v_p = getDispLimits(v_i, \mathbf{v_{max}}, p)$
  $dx_{max} = \|v_i.\mathbf{x} - v_p.\mathbf{x}\|$
  $dt_{max} = |v_i.t - v_p.t|$
  $\Delta t = getDt(dx_{max}, dt_{max}, V_f, V_{max})$
  for $j = 1$ *to* m **do** // for each neighbor
    $\mathbf{x_j} = \mathbf{x_i} + [\mathbf{V_f}(v_i) + \mathbf{V}_{lattice}(j)]\Delta t$
    $t_j = t_i + \Delta t$
    $v_j = (\mathbf{x_j}, t_j)$
    if $\mathbb{O}(v_j)$ **then** // check obstacles
      continue
    end
    $cost = [K_h + K_d \|\mathbf{V}_{lattice}(j)\|^{\alpha}]\Delta t$
    if $v_j \in \mathscr{G}$ **then**
      if $(v_j.cost > v_i.cost + cost)$ **then**
        $v_j.updateVertex(v_i, cost)$
        $\mathscr{G}.addEdge(v_i v_j), Q.update(v_j)$
      end
    else
      $v_j.heuristic = \text{getHeuristic}(v_j, \mathbf{x}_g)$
      $v_j.updateVertex(v_i, cost)$
      $Q.insert(v_j), \mathscr{G}.addNode(v_j), \mathscr{G}.addEdge(v_i v_j)$
    end
  end
end
return Γ

IV. SIMULATIONS

In this section the performance of the aSTS method is verified in simulations. In all simulations, the path parameters were set as $K_h = 0.0005$, $K_d = 1$ and $\alpha = 2$ (linear drag model), and all simulations were run on a Core I-7 3.4GHz PC with 16GB of RAM.

The accuracy of the paths computed by the aSTS method was evaluated by comparing it against a path obtained by solving the corresponding optimal control problem. The optimal control problem involves minimizing the path cost given in (5), subject to the kinematic model in (1), and the constraint $V_{still} \leq V_{max}$. This results in a two point boundary value problem which was solved using the indirect *shooting method*. This approach solves the governing differential equations for a succession of initial directions until the goal position is reached. The result from the shooting method was refined using MATLAB's BVP solver. Let $\Gamma^* : [t_s,\ t_g] \mapsto \mathbb{W}$ be the reference path obtained from the optimal control formulation of the problem, and let $\Gamma : [t_s,\ t_g] \mapsto \mathbb{W}$ be the path computed by the proposed method. The mean error (mE) between Γ^* and Γ, defined by

$$mE = \int_{t_s}^{t_g} \frac{\|\Gamma^*(t) - \Gamma(t)\|}{t_g - t_s} dt, \tag{16}$$

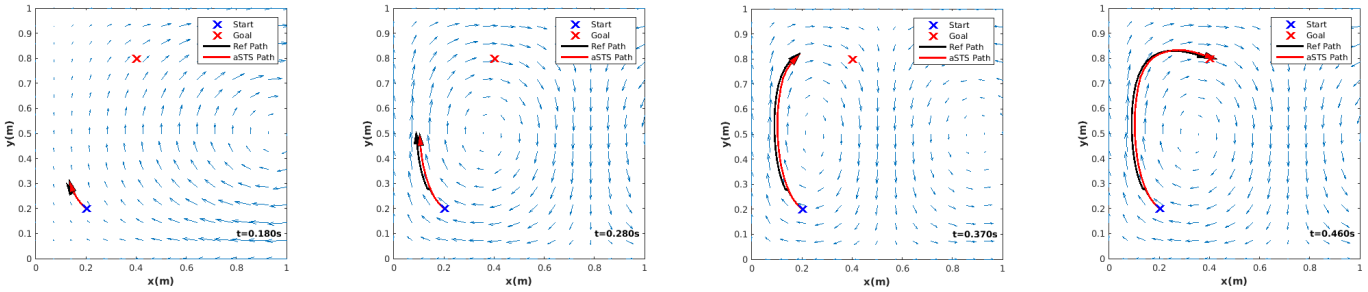
was used to evaluate the relative accuracy of computed paths. One could alternatively use the path cost to compare the results. However, the computed path cost is a function of the discretization resolution and thus does not provide a good measure for path quality. For example, a path with only two intermediate nodes might have better cost because it ignores the flow variation at intermediate points along the path.

A. Simulations using the double-gyre flow model

The aSTS method was first used to compute an optimal path in a flow field described by the double-gyre flow model. Specifically, case 3 of the illustrative example in section III-A was considered where it was required to compute an optimal path between $\mathbf{x_s} = [0.2,\ 0.2]^t$ and $\mathbf{x_g} = [0.4,\ 0.8]^t$ in a flow with parameters $A = 1$, $\omega = 4\pi$ and $\epsilon = 0.6$. The maximum flow velocity encountered in the region was $V_{fm} = 6.73$m/s, and as such the maximum vehicle speed was set at $V_{max} =$ 2m/s. The number of neighbors along the main axis of the neighbor lattice was set at $n = 3$ and velocity error threshold was set at $p = 0.1$, *i.e.,* the flow velocity variation along any edge in the graph will be less than 10%. Fig. 4 shows the time evolution of the path computed by the aSTS method against that of the reference path computed using the optimal control formulation. It can be seen that the paths match very closely, and specifically the mean error of the computed path is just 0.01m. Furthermore, the aSTS method has been able to overcome the problems associated with manual selection of discretization resolution that was seen in case 3 of section III-A. In the aSTS method only n and p has to be set by the user and the discretization levels are computed automatically to suit the underlying flow.

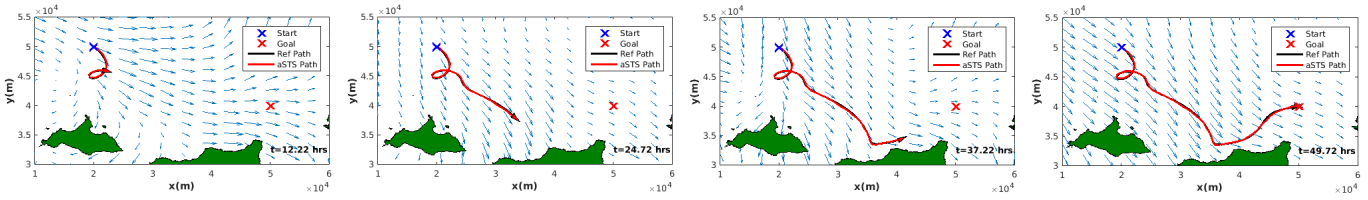
B. Simulations using ocean flow data

The aSTS method was also used to compute optimal paths in an ocean environment. Flow data generated by the Regional Ocean Model System (ROMS) for the Santa Barbara Bay area off the coast of southern California were used in these simulations. The Southern California Coastal Ocean Observing System (SCCOOS) generates these hourly ocean current forecasts everyday and each forecast is for 72 hours [18]. The data generated on July 7 and July 8 2016 were used for the simulations. The ROMS data has a $3km \times 3km \times 1hr$ spatio-temporal resolution and linear interpolation was used to obtain flow velocities at intermediate coordinates. The maximum flow speed was $V_{fm} = 0.73m/s$ and as such V_{max} was selected to be $0.5m/s$. As in the previous case, the parameters were set at $n = 3$ and $p = 0.1$. Fig. 5 shows the comparison between the reference path and the path computed using the aSTS method. The reference path computed using the optimal control formulation had a cost of 3775J and it took 12144s to converge onto the best solution. It can be seen that the paths match closely with a mean error of $mE = 169$m. The path cost obtained from the aSTS method was 4243J. As before $K_h = 0.0005$ and $K_d = 1$ were used for the simulations, and



(a) t=0.18s (b) t=0.28s (c) t=0.37s (d) t=0.46s

Fig. 4. Comparison of the optimal path computed using the aSTS method (red) with the path computed using an optimal control formulation (black), in a double-gyre flow. The parameters for the aSTS method were set at $n = 3$, $p = 0.1$. Mean path error for the aSTS path is $mE = 0.01m$.



(a) t=12.2hrs (b) t=24.7hrs (c) t=37.2hrs (d) t=49.7hrs

Fig. 5. Comparison of optimal path computed using the aSTS method (red) with the path computed using the optimal control formulation (black) in an ocean environment. The parameters for the aSTS method were set at $n = 3$, $p = 0.1$. Mean path error for the aSTS path is $mE = 169m$.

as a result, more prominence is given to reducing the energy expended to overcome drag forces. Therefore, the computed path tends to follow the flow as much as possible in order to reduce relative motion between the flow and the vehicle. This leads to the loop structure that can be observed in Fig. 5. The video for this simulation trial can be found on https://youtu.be/6R0RevaAMmY.

Table I shows the results of the aSTS method for different p and n values. It shows that increasing n while keeping p constant (paths 1-3) results in more accurate paths at the expense of running time. Similarly, decreasing p while keeping n constant (paths 5-3) also results in more accurate paths at the expense of running times. Note that the running time is only a fraction of the total path duration for each case. Table II shows the results for paths computed using a fixed-grid discretization scheme. In this fixed discretization method, the user has to specify Δx and Δt, the spatial and temporal discretization resolutions, as well as nHx and nHt, the number of spatial and temporal hops considered as neighbors at each expansion. In the results shown in Table II, all paths were computed with $\Delta x = 150m$ and $nHx = 3$ so that it has the same spatial discretization as the mean resolution obtained for path 3 of the aSTS method. It can be seen that Path 1 has a similar accuracy as path 3 of the aSTS method, and the computation is faster. However, the results obtained from the fixed discretization method depends heavily on the user defined discretization resolution. For example, a slight increase in the temporal resolution (path 2) results in a less accurate path. The results degrade further when the number of temporal neighbors considered are decreased (path3). Thus, increased accuracy cannot be guaranteed even if the discretization is made finer with such fixed discretization schemes. However, accuracy is guaranteed to improve in the proposed aSTS method by simply increasing n and decreasing p.

TABLE I
RESULTS FOR THE ASTS METHOD FOR DIFFERENT p AND n VALUES.

Path	**1**	**2**	**3**	**4**	**5**
p	0.1	0.1	0.1	0.2	0.3
n	1	2	3	3	3
mErr (m)	6426	595.3	169.2	656.3	1095
cost (J)	9200	5284	4243	4234	4179
running time (s)	385	1074	2009	687	364
path duration (s)	182800	176200	1782000	179400	180600

TABLE II
RESULTS FOR THE FIXED DISCRETIZATION METHOD.

Path	**1**	**2**	**3**
Δt	150	100	100
nXt	7	10	5
mErr (m)	226.5	326.0	1256
cost (J)	4191	4236	7195
running time (s)	1399	2641	816
path duration (s)	180000	180600	176000

V. CONCLUSIONS

In this work, a graph based approach to optimal path planning in time-varying flows was presented. The method uses an adaptive discretization scheme that is based on the spatio-temporal variation of the underlying flow field. While only optimal energy paths were considered in this paper, the method is general enough to handle other cost functions in time-varying flows. The correctness of the computed paths were verified in simulations, by comparing them with paths computed using an optimal control framework. Furthermore, it

was shown that the method was able to overcome some issues associated with the use of fixed discretization resolutions in existing methods.

Similar to existing work on path planning in flows in literature, the presented method assumes that accurate flow velocity forecasts are available for the computation of optimal paths. However, the ocean current forecasts provided by the CORDC databases, the regional ocean model systems (ROMS), and/or other numerical models are often uncertain [15]. Thus, in order for these graph search methods to be used with such uncertain forecasts, the effect of noise on the computed optimal paths need to be investigated, and this is a direction for future work.

Furthermore, it seems plausible that optimal paths computed with a higher p value could be re-used to compute optimal paths with increasingly lower p values without exploring the complete environment again. Such an implementation will make the aSTS method a candidate for any time path planning [19] and this is research direction being investigated.

REFERENCES

[1] J. Yuh, G. Marani, and D. R. Blidberg, "Applications of marine robotic vehicles," *Intelligent Service Robotics*, vol. 4, no. 4, p. 221, Jul 2011.

[2] B. Garau, A. Alvarez, and G. Oliver, "Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, April 2005, pp. 194–198.

[3] T.-B. Koay and M. Chitre, "Energy-efficient path planning for fully propelled auvs in congested coastal waters," in *OCEANS - Bergen, 2013 MTS/IEEE*, 2013, pp. 1–9.

[4] D. Rao and S. B. Williams, "Large-scale path planning for underwater gliders in ocean currents," in *Australasian Conference on Robotics and Automation (ACRA)*, Sydney, 2009.

[5] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Time and energy optimal path planning in general flows," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

[6] A. Chakrabarty and J. Langelaan, "Uav flight path planning in time varying complex wind-fields," in *2013 American Control Conference*, June 2013, pp. 2568–2574.

[7] M. Eichhorn, "Optimal routing strategies for autonomous underwater vehicles in time-varying environment," *Robotics and Autonomous Systems*, vol. 67, pp. 33 – 43, 2015.

[8] M. Otte, W. Silva, and E. Frew, "Any-time path-planning: Time-varying wind field + moving obstacles," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 2575–2582.

[9] D. Kruger, R. Stolkin, A. Blum, and J. Briganti, "Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 4265–4270.

[10] J. Witt and M. Dunbabin, "Go with the flow: Optimal auv path planning in coastal environments," in *Australasian Conference on Robotics and Automation (ACRA)*, 2008.

[11] T. Lolla, P. F. J. Lermusiaux, M. P. Ueckermann, and P. J. Haley, "Time-optimal path planning in dynamic flows using level set equations: theory and schemes," *Ocean Dynamics*, vol. 64, no. 10, pp. 1373–1397, 2014.

[12] D. N. Subramani and P. F. Lermusiaux, "Energy-optimal path planning by stochastic dynamically orthogonal level-set optimization," *Ocean Modelling*, vol. 100, pp. 57 – 77, 2016.

[13] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, pp. 273–290, October 2012.

[14] S. Kim, S. Bhattacharya, and V. Kumar, "Path planning for a tethered mobile robot," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, May 31 - June 7 2014.

[15] V. Huynh, M. Dunbabin, and R. Smith, "Predictive motion planning for auvs subject to strong time-varying currents and forecasting uncertainties," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 2015, pp. 1144–1151.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, 2nd ed. MIT Press, 2001.

[17] D. R. Thompson, S. Chien, Y. Chao, P. Li, B. Cahill, J. Levin, O. Schofield, A. Balasuriya, S. Petillo, M. Arrott, and M. Meisinger, "Spatiotemporal path planning in strong, dynamic, uncertain currents," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 4778–4783.

[18] Regional ocean model system (roms) model output. [Online]. Available: http://www.sccoos.org/data/roms-3km/

[19] E. A. Hansen and R. Zhou, "Anytime heuristic search," *J. Artif. Int. Res.*, vol. 28, no. 1, pp. 267–297, Mar. 2007.

APPENDIX

To find an admissible heuristic function for the cost function given in (5), we assume that ideal flow conditions exist between the node v_i and the goal node v_g, *i.e.,* we assume that the flow is always directed from $\mathbf{x_i}$ to $\mathbf{x_g}$ and that the flow speed is at its maximum possible value V_{fm}. Thus from (2), $V_{net} = V_{still} + V_{fm}$ since $\theta = 0$, and the corresponding travel time between $\mathbf{x_i}$ and $\mathbf{x_g}$ is $\Delta t = \Delta x / V_{net}$ where $\Delta x = \|\mathbf{x_g} - \mathbf{x_i}\|$. Using (6), the cost of travel from $\mathbf{x_i}$ to $\mathbf{x}_g$ is,

$$C_{ig}(V_{still}) = (K_h + K_d V_{still}^{\alpha}) \frac{\Delta x}{V_{still} + V_{fm}}. \tag{17}$$

The derivative of (17) is given by,

$$\frac{dC_{ig}}{dV_{still}} = \frac{K_d(\alpha - 1)V_{still}^{\alpha} + K_d \alpha V_{still}^{\alpha - 1} - K_h}{(V_{fm} + V_{still})^2} \Delta x. \tag{18}$$

Thus, an extremal of C_{ig} satisfies the polynomial,

$$K_d(\alpha - 1)V_{still}^{\alpha} + K_d \alpha V_{still}^{\alpha - 1} - K_h = 0 \tag{19}$$

It can be shown that $\frac{d^2C_{ig}}{dV_{still}^2} \geq 0$. Thus, C_{ig} is minimized when V_{still} satisfies (19). Therefore, the admissible heuristic function is given by,

$$h(v_i) = (K_h + K_d v_{still}^{\alpha}) \frac{\|\mathbf{x_g} - \mathbf{x_i}\|}{v_{sel} + V_{fm}} \tag{20}$$

where V_{still} is given by the solution to (19).