



Going with the flow: a graph based approach to optimal path planning in general flows

Dhanushka Kularatne¹ · Subhrajit Bhattacharya² · M. Ani Hsieh¹

Received: 15 February 2017 / Accepted: 2 April 2018 / Published online: 21 April 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

Autonomous surface and underwater vehicles (ASVs and AUVs) used for ocean monitoring are typically deployed for long periods of time and must operate with limited energy budgets. Coupled with the increased accessibility to ocean flow data, there has been a significant interest in developing energy efficient motion plans for these vehicles that leverage the dynamics of the surrounding flow. In this paper, we present a graph search based method to plan time and energy optimal paths in static and time-varying flow fields. We also use tools from topological path planning to generate optimal paths in different homotopy classes to facilitate simultaneous exploration of the environment by multi-robot teams. The proposed strategy is validated using analytical flow models, actual ocean data, and in experiments using an indoor laboratory testbed capable of creating flows with ocean-like features. We also present an alternative approach using a Riemannian metric based approximation for the cost functions in the static flow case for computing time and energy optimal paths. The Riemannian approximation results in smoother trajectories in contrast to the graph based strategy while requiring less computational time.

Keywords Path planning · Marine robotics · Time-varying flows

1 Introduction

Scientific activities in aquatic environments that were traditionally performed manually are increasingly being automated with autonomous marine vehicles (AMVs) that include surface and/or underwater vehicles. Examples include characterizing the dynamics of plankton assemblages (Caron et al. 2008), measurement of temperature profiles (Fiorelli

et al. 2006), and monitoring of harmful algae blooms (Smith et al. 2010). In these and similar environmental monitoring applications in the ocean, 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 use of the capabilities 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. Examples include the work by Garau et al. (2005) where the authors use a graph search method to plan time optimal paths in static flows. Graph search methods have also been proposed for computing energy optimal paths in static flow fields in Koay and Chitre (2013), Rao and Williams (2009), Kularatne et al. (2016). In Chakrabarty and Langelaan (2013), authors present a tree based method to compute optimal paths in a time-varying wind field, however, this tree like structure results in an expansion of a large number of nodes which can require significant computation resources. Eichorn (2015) and Otte et al. (2016) have also presented graph based methods that

This is one of several papers published in *Autonomous Robots* comprising the “Special Issue on Robotics Science and Systems”.

This work was supported by the National Science Foundation (NSF) Grants IIS-1253917 and CMMI-1462825.

✉ Dhanushka Kularatne
dkul@seas.upenn.edu
Subhrajit Bhattacharya
sub216@lehigh.edu
M. Ani Hsieh
m.hsieh@seas.upenn.edu

¹ Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, USA

² Department of Mechanical Engineering and Mechanics, Lehigh University, Bethlehem, PA, USA

are only capable of computing time optimal paths in time-varying flows.

Alternatives to graph search techniques include (Kruger et al. 2007; Witt and Dunbabin 2008) 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. (2012, 2014) presented a level set expansion method to find time optimal paths in time-varying flows. This was then extended by Subramani et al. (2016) 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. Thus these strategies are mostly applicable for pre-deployment planning purposes and not amenable for realtime planning purposes.

In this work, we present a graph-based path planning approach for computing time and energy optimal paths for vehicles operating in a general flows. Similar to Lolla et al. (2012, 2015), Garau et al. (2005), Kruger et al. (2007), Koay and Chitre (2013), Huynh et al. (2015), the strategy leverages the surrounding flow field in the synthesis of optimal trajectories. Different from these existing strategies, we employ graph search-based methods coupled with more accurate cost functions in computing the optimal trajectories in both static and time-varying cases. In particular, the primary contributions of our work are:

- (a) The ability to design *arbitrary* cost functions with a focus on obtaining both time and energy optimal trajectories.
- (b) The ability to plan optimal paths in *both* static and time-varying flows. As such, in contrast to existing work, our strategy is not limited in the cost functions allowed nor in the type of flows that could be considered.
- (c) The inclusion of a Riemannian approximation to the cost functions presented in static flows. The ability to model cost as Riemannian metrics enables the formulation of the optimization problem as one of solving the *geodesic equation* on a Riemannian manifold which provides additional insights into the planning problem.
- (d) The ability to impose kinematic constraints such as maximum still-water speed of the vehicle which is useful for planning optimal paths for small resource constrained vehicles.
- (e) The correctness of the computed paths are verified experimentally using autonomous surface vehicles on

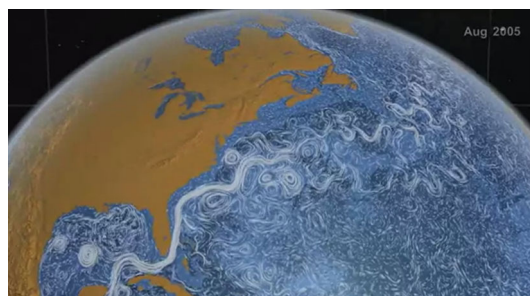


Fig. 1 Snapshot (August 2005) of visualization of ocean surface currents for June 2005 through December 2007 generated using NASA/JPLs Estimating the Circulation and Climate of the Ocean, Phase II (ECCO2) ocean model

actual flows. In addition, the results are compared against several existing methods and to “ground truth” optimal paths obtained from an optimal control formulation.

- (f) Lastly, large scale ocean circulation often exhibit significant eddy and jet structures (see Fig. 1), which gives rise to the presence of fixed points and invariant manifolds in the flow field. Coupled with the presence of obstacles, e.g., islands and archipelagos, the result is multiple topological classes of trajectories. Recent work showed the importance of reasoning about such topological classes in marine operations in the presence of obstacles (Bhattacharya et al. 2015). As such, a *fundamental advantage* of the proposed graph-based strategy over existing ones (Garau et al. 2005; Kruger et al. 2007; Koay and Chitre 2013; Huynh et al. 2015; Chakrabarty and Langelaan 2013) is the ability to use computational tools from *topological path planning* in the computation of time/energy optimal paths in different homotopy classes of the environment.

The rest of the paper is organized as follows: Sect. 2 presents the modeling preliminaries and the problem formulation. Sections 3 and 4 present the methods used to solve the problem in static and time varying flows respectively. Section 5 shows how the method is used to compute paths in different topological classes. Sections 6 and 7 describe the simulation and experimental results respectively. The paper concludes with a discussion of our findings and directions for future work in Sect. 8.

2 Problem formulation

In this section we provide a description of the environment and our assumptions, and then describe the development of our time and energy cost functions. We also provide a formal statement of our problem and outline our graph based solution approach.

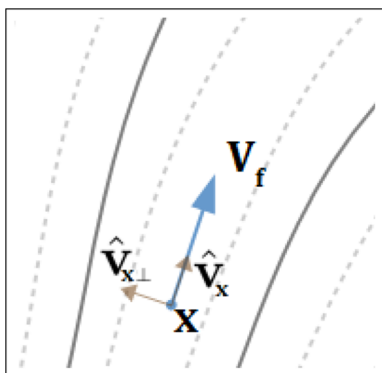


Fig. 2 The flow vector, \mathbf{V}_f at a point, \mathbf{x} , and the flow-parallel coordinate system

2.1 Flow model

In this work, 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 $\mathbb{W}_T = \mathbb{W} \times [t_s \ t_f]$ and $[t_s \ t_f] \subset \mathbb{R}_{\geq 0}$ denotes the time interval under consideration. As such, for $\mathbf{x} \in \mathbb{W}$ and $t \in [t_s \ t_f]$,

$$\dot{\mathbf{x}} = \mathbf{V}_f(\mathbf{x}, t) \tag{1}$$

denotes the flow velocity at the point \mathbf{x} . 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

$$V_{fm} = \max_{\mathbf{x} \in \mathbb{W}, t \in [t_s \ t_f]} V_f(\mathbf{x}, t).$$

It is assumed that the flow description is known a priori or a reliable forecast is available. For ocean environments, such a forecast could be obtained from, for example, the Coastal Observing Research and Development Center (CORDC) (CORDC).

We employ a *flow-parallel* coordinate representation of the vector field given by (1), such that for every point $\mathbf{x} \in \mathbb{W}$, its axes are aligned along the unit vectors $\hat{\mathbf{v}}_x = \frac{\mathbf{V}_f(\mathbf{x}, t)}{\|\mathbf{V}_f(\mathbf{x}, t)\|}$ (the “ x ” axis) and $\hat{\mathbf{v}}_{x\perp}$ (the “ y ” axis, orthogonal to the x axis – see Fig. 2). We note that this coordinate system is not “co-moving” with the flow.

Thus, in the flow-parallel coordinate system, the flow vector at \mathbf{x} can simply be written as $\mathbf{V}_f = V_f \hat{\mathbf{v}}$. With a little abuse of notation, we will also write the coordinate representation of \mathbf{V}_f in the flow-parallel coordinate system as $\mathbf{V}_f = [V_f, 0]^T$.

2.2 Vehicle model

The vehicles considered in this work are assumed to have a holonomic kinematic motion model. This is a reasonable assumption when the dimensions of the vehicle are small

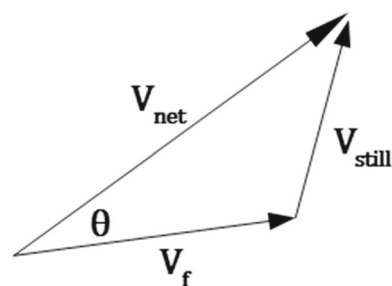


Fig. 3 The net velocity of the vehicle is the vector sum of the flow velocity and the vehicle’s still-water velocity

when compared with the dimensions of the flow structures (Garau et al. 2005). It is assumed that the lower level controllers of the vehicle are capable of following a trajectory computed using this simplified model. Using this model, the net velocity of the vehicle with respect to the inertial frame is given by

$$\mathbf{V}_{net}(\mathbf{x}, t) = \mathbf{V}_f(\mathbf{x}, t) + \mathbf{V}_{still}(\mathbf{x}, t), \tag{2}$$

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. The net velocity of a vehicle which travels infinitesimal distances, dx and dy , along $\hat{\mathbf{v}}_x$ and $\hat{\mathbf{v}}_{x\perp}$ respectively in time dt , can be written as $\mathbf{V}_{net} = [dx, dy]^T / dt$ (expressed in the components of the flow-parallel coordinates). Then, from (2), the flow-parallel coordinate representation of the vehicle thrust is $\mathbf{V}_{still} = [\frac{dx}{dt} - V_f, \frac{dy}{dt}]^T$ (see Fig. 3). Thus the speed of the vehicle in still water is given by

$$V_{still} = \|\mathbf{V}_{still}\| = \sqrt{\left(\frac{dx}{dt} - V_f\right)^2 + \left(\frac{dy}{dt}\right)^2}. \tag{3}$$

It is further assumed 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., $V_{still}(\mathbf{x}, t) \leq V_{max} < V_{fm}$.

2.3 Cost functions

We will assign (infinitesimal) costs to infinitesimal displacements of the vehicle. In general, it is a function of differentials, dx, dy and dt (where x and y are the flow-parallel coordinates), and is of the form $dc_3 = f(dx, dy, dt)$. However, for static flows (i.e., $\mathbf{V}_f(\mathbf{x}, t) = \mathbf{V}_f(\mathbf{x})$), dt can be shown to be a function of dx and dy , and as a results, the differential cost has the form $dc_2 = f(dx, dy)$.

2.3.1 Time minimizing cost

In general, the time cost of a differential element will simply be,

$$dc_{3,time} = dt. \tag{4}$$

For optimal time paths in static flows, it can be shown that dt can be expressed as a function of dx and dy (see Sect. 3.1.1), i.e., $dc_{3,time} = dc_{2,time}$ for static flows. This same expression for dt can also be used in time-varying flows, if the vehicle thrust V_{still} is able to overcome the flow velocity V_f at all times. However, since we are considering vehicles that have actuation limitations, we use the general time cost given above in time varying flows as well.

2.3.2 Energy minimizing cost

The differential cost in this case is the energy expended in achieving a differential displacement, $[dx, dy]^T$ in time dt . It is assumed that this energy expenditure can be solely attributed to the action of overcoming the drag force experienced by the vehicle. The drag force experienced by the vehicle is given by

$$\mathbf{F}_d(\mathbf{x}, t) = -\kappa V_{still}^{\alpha-1}(\mathbf{x}, t) \frac{\mathbf{V}_{still}}{V_{still}} \tag{5}$$

where κ is the drag coefficient which depends on the geometry of the vehicle and $\alpha \in \{2, 3, \dots\}$ defines the drag model considered. For example, if $\alpha = 2$ the drag model is linear and if $\alpha = 3$ it is quadratic. Thus, considering the differential displacement of the vehicle with respect to the flow $d\mathbf{x}_{still} = \mathbf{V}_{still} dt$, the differential energy cost is given by

$$\begin{aligned} dc_{3,energy} &= \mathbf{F}_d \cdot d\mathbf{x}_{still} = \kappa V_{still}^\alpha dt \\ &= \kappa \left(\left(\frac{dx}{dt} - V_f \right)^2 + \left(\frac{dy}{dt} \right)^2 \right)^{\frac{\alpha}{2}} dt \\ &= \kappa \left(\frac{dx^2 + dy^2}{dt^2} - 2V_f \frac{dx}{dt} + V_f^2 \right)^{\frac{\alpha}{2}} dt \end{aligned} \tag{6}$$

2.4 Problem statement

Given the above time and energy cost functions, the objective is to find a path $\Gamma : [t_s, t_g] \mapsto \mathbb{W}$ that minimizes the total cost. Thus the problem addressed in this paper is finding a solution to the following optimization problem,

$$\Gamma^* = \min_{\Gamma} \int_{\Gamma} dc_{\gamma,\beta}$$

subject to

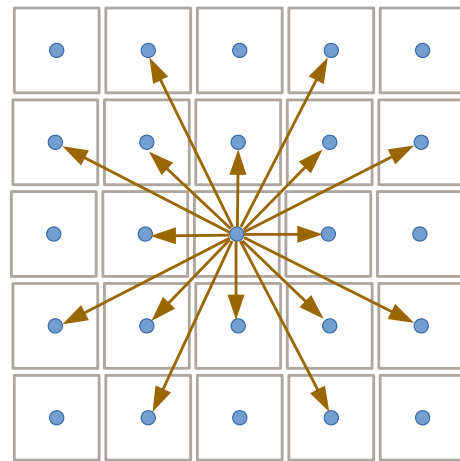


Fig. 4 The connectivity of a vertex with other neighboring vertices in the graph \mathcal{G}

$$\begin{aligned} \Gamma(t_s) &= \mathbf{x}_s, \\ \Gamma(t_g) &= \mathbf{x}_g, \\ V_{still}(\Gamma(t), t) &\leq V_{max} \end{aligned} \tag{7}$$

where \mathbf{x}_s and \mathbf{x}_g are the desired start and goal positions, γ denotes the dimension of the workspace and β denotes either *time* or *energy*. When $\gamma = 2$, the infinitesimal element belongs to \mathbb{W} and when $\gamma = 3$ it belongs to \mathbb{W}_T .

2.5 Solution approach

We use a graph based approach to find a solution to this problem. We use a discrete graph, $\mathcal{G} = (\mathcal{V}, E)$, to represent the workspace. The workspace can either be the spatial environment \mathbb{W} , or the extended space–time environment \mathbb{W}_T . The workspace that the graph represents is dependent on the type of flow considered (static vs time-varying). Vertices in this graph are centroids of cells in a uniform square/cubic discretization (in the global coordinates). As such, if the workspace is \mathbb{W} , a vertex v_i is identified by its global spatial coordinate \mathbf{x}_i , and if the workspace is \mathbb{W}_T , a vertex is identified by the pair (\mathbf{x}_i, t_i) . Edges are established between immediate neighbors as well as some n -hop neighbors (Fig. 4).

Each edge in the graph can now be approximately considered as an infinitesimal segment with projections dx and dy along $\hat{\mathbf{v}}_x$ and $\hat{\mathbf{v}}_{x\perp}$ respectively, where \mathbf{x} is the position of the vertex at the base of the edge. If the the graph is defined over \mathbb{W}_T , each edge will have a dt component as well. This lets us assign a cost to the edge using formula (4) (time optimal search) or (6) (energy optimal search). The implicit assumption is that the flow velocity remains constant along an edge.

In this approach, the optimal path would be a sequence of vertices $\{v_0, v_1, \dots, v_k\}$ where v_0 represents the start ver-

tex and v_k represents the goal vertex. If the workspace is \mathbb{W} , sometimes we use the spatial locations of the vertices $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$ to represent the path, since in this case the vertices are identified using their position coordinates alone. Furthermore, we use the notation, $\bar{\mathbf{x}}_i\bar{\mathbf{x}}_j$ to refer to a sub path of the optimal path, i.e., $\bar{\mathbf{x}}_i\bar{\mathbf{x}}_j = \{\mathbf{x}_i, \mathbf{x}_{i+1}, \dots, \mathbf{x}_j\}$

Given a start and a goal vertex in the graph, we can thus use Dijkstra’s optimal search algorithm (Cormen et al. 2001) to find the shortest path in the graph connecting the vertices. Whenever possible, we can also design admissible heuristic functions, $h : \mathcal{V} \rightarrow \mathbb{R}_{\geq 0}$, in order to run more efficient search algorithms such as A* (Cormen et al. 2001). For time minimizing search, if $[X_g, Y_g]^T$ is the coordinate of the goal vertex v_g in a global Euclidean coordinate system, and $[X_i, Y_i]^T$ is the global coordinate of a vertex $v_i \in \mathcal{V}$ then, a lower bound for the minimum time required to reach v_g from v_i is

$$h_{time}(v_i) = \frac{\sqrt{(X_g - X_i)^2 + (Y_g - Y_i)^2}}{V_{max} + V_{fm}}$$

where V_{fm} is the maximum speed of the flow over all points in \mathbb{W} . However for the *energy minimizing search*, one cannot write a reasonable heuristic function other than the trivial $h(v_i) = 0$, in which case the A* search becomes equivalent to Dijkstra’s.

3 Optimal paths in static flow fields

In this section the graph based approach is used to find optimal paths when the flow field is static, i.e., when $\mathbf{V}_f(\mathbf{x}, t) = \mathbf{V}_f(\mathbf{x})$. In such a scenario, the flow at a particular point $\mathbf{x} \in \mathbb{W}$ does not depend on the time of arrival at \mathbf{x} . As such, the cost of any path segment that starts from \mathbf{x} also does not depend on the time of arrival at \mathbf{x} . Thus, in static flows, the propagation of time along a path need not be considered in the computation of the path costs. Therefore, the graph constructed for optimal path computation in static flows only has to consider the 2D environment \mathbb{W} . In this case each vertex $v_i \in \mathcal{V}$ will be identified by its position $\mathbf{x}_i \in \mathbb{W}$. Furthermore, the optimal path will be composed of edges (path segments) which have optimal costs individually. Thus, the general costs $dc_{3,time}$ and $dc_{3,energy}$ given in (4) and (6) respectively, can now be reduced to $dc_{2,time}$ and $dc_{2,energy}$ by optimizing these costs over dt .

3.1 Cost functions for static flows

3.1.1 Time minimizing cost for static flows

In order to minimize the time cost of travel, the vehicle has to travel at its maximum speed along each edge. Thus, using $V_{still} = V_{max}$, we can solve for dt using (3) as follows,

$$\begin{aligned} & \left(\left(\frac{dx}{dt} - V_f \right)^2 + \left(\frac{dy}{dt} \right)^2 \right) = V_{max}^2 \\ & \Rightarrow dx^2 - 2V_f dx dt + (V_f^2 - V_{max}^2) dt^2 + dy^2 = 0 \\ & \Rightarrow dt = \frac{V_f}{V_f^2 - V_{max}^2} dx - \frac{\sqrt{V_{max}^2(dx^2 + dy^2) - V_f^2 dy^2}}{V_f^2 - V_{max}^2} \end{aligned} \tag{8}$$

Note that we discard the solution with the positive root since when $dy = 0$, we should have $dt = dx/(V_f + V_{max})$. Thus, the time-minimizing cost for a differential element in a static flow is

$$dc_{2,time} = \frac{V_f}{V_f^2 - V_{max}^2} dx - \frac{\sqrt{V_{max}^2(dx^2 + dy^2) - V_f^2 dy^2}}{V_f^2 - V_{max}^2} \tag{9}$$

3.1.2 Energy minimizing cost for static flows

For a given dx and dy , the differential time dt that minimizes the energy cost $dc_{3,energy}$ given in (6), can be computed by solving $\frac{\partial dc_{3,energy}}{\partial dt} = 0$. Essentially, we are letting the vehicle choose a V_{still} that minimizes $dc_{3,energy}$. It can easily be verified that this local energy cost minimization is accomplished when,

$$dt = \frac{(2 - \alpha)dx + \sqrt{\alpha^2 dx^2 + 4(\alpha - 1)dy^2}}{2V_f} \tag{10}$$

where $\alpha \in \{2, 3, \dots\}$ depends on the drag model considered. Thus the energy cost of a differential element in \mathbb{W} is

$$dc_{2,energy} = \kappa \left(\frac{dx^2 + dy^2}{dt^2} - 2V_f \frac{dx}{dt} + V_f^2 \right)^{\frac{\alpha}{2}} dt \tag{11}$$

where dt is the function of dx and dy given in (10).

3.2 An alternative approximate model for fast computation of smooth trajectories

The optimal trajectories obtained using the graph search-based approach, being constrained to the graph, are piecewise linear, and in general not smooth. In order to be able to solve the problem efficiently, we need to employ an approximation at some level, which in case of the graph search-based method is the discrete representation of the free space. In this section we present an alternative approximate model, derived from the cost functions described in Sect. 3.1, to represent the optimal trajectory in static flows in the form of a solution to an ordinary differential equation. Thus the solutions obtained using this method can be made arbitrarily smooth. Moreover

integration of a differential equation is in general faster to compute than running a graph search algorithm. In particular, we construct a Riemannian metric (Petersen 2006) that approximates the cost functions. This method is particularly useful for two reasons: (i) trajectories obtained as a solution to the geodesic differential equation are smooth, unlike paths in a graph that are restricted to the discrete graph, and (ii) computing solutions to the geodesic equation is often less computationally intensive than a full-blown graph search.

The Riemannian metric, $g(\mathbf{x})$, for every $\mathbf{x} \in \mathbb{W}$ is defined as a (symmetric, positive-definite) bilinear form over differentials dx and dy such that the cost (equivalently the “length” induced by the metric) of the differential element can be written as

$$dc = \sqrt{[dx, dy] g(\mathbf{x}) \begin{bmatrix} dx \\ dy \end{bmatrix}} \tag{12}$$

where $g(\mathbf{x})$ is a 2×2 matrix, and is called the *matrix representation of the Riemannian metric* in the flow parallel coordinate system. Using this framework, we propose the following approximate time and energy minimizing Riemannian metric modes for computing optimal paths in static flows.

Riemannian Metric Model for Time Minimization: From equation (9) one has,

$$dc_{2,time}^2 = \frac{(V_{max}^2 + V_f^2)dx^2 + (V_{max}^2 - V_f^2)dy^2}{(V_f^2 - V_{max}^2)^2} - \frac{2V_f dx \sqrt{V_{max}^2 dx^2 + (V_{max}^2 - V_f^2)dy^2}}{(V_f^2 - V_{max}^2)^2}.$$

In order to make the above quadratic in dx and dy , we use the approximation that the velocity of the vehicle is almost parallel to that of the flow, i.e., $\left| \frac{dy}{dx} \right| \ll 1$. With this approximation,

$$\begin{aligned} &\sqrt{V_{max}^2 dx^2 + (V_{max}^2 - V_f^2)dy^2} \\ &\cong V_{max} dx + \frac{1}{2} \frac{V_{max}^2 - V_f^2}{V_{max}} \frac{dy}{dx} dy. \end{aligned}$$

Thus we get,

$$dt^2 \cong \frac{1}{(v + V_{max})^2} dx^2 + \frac{1}{V_{max}(V_{max} + v)} dy^2.$$

Thus, the matrix representation of the metric tensor in the 2-dimensional flow-parallel coordinates can be written as,

$$g = \begin{bmatrix} \frac{1}{(v+V_{max})^2} & 0 \\ 0 & \frac{1}{V_{max}(V_{max}+v)} \end{bmatrix}. \tag{13}$$

Riemannian Metric Model for Energy Minimization: For the special case of $\alpha = 2$, using (10) and (11), one can write,

$$dc_{2,energy} = 2\kappa \left(\sqrt{dx^2 + dy^2} - dx \right)$$

As such,

$$dc_{2,energy}^2 = 4\kappa^2 v^2 \left(2dx^2 + dy^2 - 2dx^2 \sqrt{1 + \left(\frac{dy}{dx} \right)^2} \right).$$

Using the approximation that the velocity of the vehicle is almost parallel to that of the flow, i.e., $\left| \frac{dy}{dx} \right| \ll 1$, we could rewrite above as,

$$dc_{2,energy}^2 = 4\kappa^2 v^2 dy^2. \tag{14}$$

However, the matrix representation of the metric tensor that arises from (14) is singular and cannot be used with the geodesic equation. As such, we use the following matrix representation of the metric tensor in flow-parallel coordinates:

$$g = \begin{bmatrix} \frac{\epsilon}{v^2} & 0 \\ 0 & 4\kappa^2 v^2 \end{bmatrix} \tag{15}$$

where $\epsilon \ll 1$.

From g , one can use a coordinate transformation, $G = R^T g R$, to compute the matrix representations of the metric tensors in the global Euclidean coordinates.

$$R = \begin{bmatrix} \hat{v}_X & -\hat{v}_Y \\ \hat{v}_Y & \hat{v}_X \end{bmatrix}$$

is a suitable rotation matrix for this local-to-global coordinate transformation, where \hat{v}_X and \hat{v}_Y are the components of the flow-parallel unit vector \hat{v} expressed in the global frame. We can then solve the geodesic equation (Petersen 2006),

$$\frac{d^2 \gamma^i}{d\tau^2} + \Gamma_{jk}^i \frac{d\gamma^j}{d\tau} \frac{d\gamma^k}{d\tau} = 0$$

which is a second order ODE describing the shortest path, γ , parameterized by τ , the cumulative cost of the trajectory. Note that summation over repeated indices is assumed by Einstein notation. The quantities Γ_{jk}^i are called the Christoffel symbols, and are given by

$$\Gamma_{jk}^i = \frac{1}{2} \bar{G}^{im} \left(\frac{dG_{km}}{dX^j} + \frac{dG_{jm}}{dX^i} - \frac{dG_{jk}}{dX^m} \right)$$

with $\bar{G} = G^{-1}$.

One consequence of the Riemannian approximation is the fact that the metric is indifferent to the direction of the flow, and the vector field in effect gets converted to a line

field. Geodesics computed using the approximate Riemannian metric can thus be both in the direction of the flow and against it. However, the Riemannian model does not involve any discretization of the free space and can perform better than the graph search-based approach with a coarse discretization. This is illustrated in the example of Fig. 11, where a coarse discretization gives a path in the graph that is significantly different and of higher cost than the geodesic computed using the Riemannian metric. However, making the discretization finer (increasing connectivity of the graph) gives a path in the finer graph that matches the geodesic more closely. Other than this particular example, however, all the simulation and experimental results presented in this paper use the graph search-based method and the exact cost functions described in Sect. 3.1.

4 Optimal paths in time varying flow fields

In this section we extend the graph based optimal path planning approach to time-varying flow fields. In the static flows considered in Sect. 3, the traversal cost of a path segment starting at \mathbf{x} is independent of the time of arrival at \mathbf{x} . As such, every sub path $\bar{\mathbf{x}}_i \bar{\mathbf{x}}_j$ of an optimal path $\{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k\}$ has optimal cost. In time-varying flows however, the cost of a path segment depends explicitly on the time of arrival at the base of that path segment, as the costs are inherently tied to the flow speed. As such, in contrast to static flows, it is now possible to travel to an intermediate node in a sub-optimal manner in order to encounter a favorable current towards the destination later on that results in minimizing the overall cost of the path. Hence, greedy strategies that may be suitable for static flows are no longer valid. As such, the search space needs to take the time dimension into account explicitly, when computing optimal paths in time-varying flow fields.

An exception to this discussion can be observed when the objective function is time, if the vehicle thrust V_{still} is able to overcome the flow speed V_f at all times. In this case, it is generally possible to use the same method and cost function (given in (9)) used for static flows, without having to expand the graph along the time dimension, i.e., the graph search method used for static flows can be used for this time-varying scenario as well. However, since we are considering a more general case, we use the time-extended graph to compute optimal time paths as well.

4.1 Multi time-step (MTS) search method

In this section, we present the Multi Time-Step Search (MTS) method to find optimal paths in time-varying flows. In essence, we expand the graph along the time dimension so that we could search through different V_{still} values along

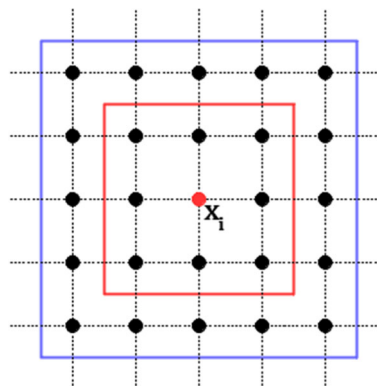


Fig. 5 The spatial distribution of neighbors considered at each node. The red square includes the spatial neighbors considered when $N_{shops} = 1$, and the blue square includes the spatial neighbors considered when $N_{shops} = 2$ (Color figure online)

each path segment, that will give a minimum overall cost. In this method, the 3-D spatio-temporal workspace \mathbb{W}_T is discretized uniformly with increments Δx , Δy and Δt in the x , y and t directions respectively. As before, the centroid of each cube in the discretized space is represented by a vertex $v_i \in \mathcal{V}$, and each vertex v_i is identified by the pair (\mathbf{x}_i, t_i) . Each v_i also has a set of tentative neighbors $\mathcal{N}(v_i)$, where each $v_j \in \mathcal{N}(v_i)$ satisfies $\max(|x_j - x_i|/\Delta x, |y_j - y_i|/\Delta y) \leq N_{sHops}$ and $0 < (t_j - t_i)/\Delta t \leq N_{tHops} \cdot N_{sHops}$. N_{sHops} defines the number of spatial hops considered as neighbors, and its value is typically set to a small integer in MTS in order to keep the average degree of the vertices in the graph low. If $N_{sHops} = 1$, only the immediate spatial neighbors are considered and if $N_{sHops} = 2$ then 2-hop spatial neighbors are also considered (see Fig. 5). Each edge of the graph is assigned a cost computed using the cost functions given in (4) (optimal time) and (6) (optimal energy).

For a given v_i , with associated flow speed $V_{f_i} = V_f(\mathbf{x}_i, t_i)$, the required still-water speed and the associated traversal cost to go to any $v_j \in \mathcal{N}(v_i)$ with a given spatial coordinate $\bar{\mathbf{x}}$ depend primarily on $dt_{ij} = t_j - t_i$. In the case of optimal energy paths, the cost to reach $\bar{\mathbf{x}}$ will decrease as dt_{ij} increases (see Fig. 6). The main idea behind the MTS method is to search through all of these path segments to find the least cost path from start to goal. Naturally, low cost paths can be obtained by setting N_{tHops} to a large value, because it allows transitions between two adjacent spatial locations, \mathbf{x}_i and \mathbf{x}_j , over a long period of time ($dt_{ij} = N_{tHops} \times \Delta t$). However, such a transition breaks the implicit assumption that the flow velocity \mathbf{V}_f remains constant between two vertices, v_i and v_j , since in practice flow conditions are likely to change over the large the transition time dt_{ij} . Thus, N_{tHops} should be selected such that the underlying flow remains relatively constant during $N_{tHops} \times \Delta t$. In this work, N_{tHops} is set such that $N_{tHops} \times \Delta t < T_s$, where T_s is the sampling period of the available flow velocity data.

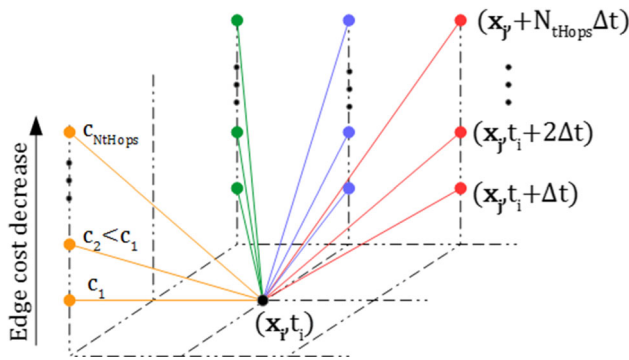


Fig. 6 The construction of the MTS graph for $N_{hops} = 1$. Only a portion of the neighbor set is shown. For each spatial neighbor \mathbf{x}_j , multiple neighbors are considered along the time axis. In the case of optimal energy paths, for each \mathbf{x}_j , the cost of the segment $\overline{\mathbf{x}_i \mathbf{x}_j}$ will decrease as $dt_{ij} = t_j - t_i$ is increased, since the V_{still} required for the traversal is reduced with increasing dt_{ij}

The pseudo code for the MTS method is given in Algorithm 1. Initially, the start vertex is added to the heap Q with zero cost. The `getHeuristic()` method computes the value of the heuristic if one is available. The heap Q is sorted using the $(heuristic + cost)$ of the vertices (A* method). Only neighbors with $V_{still} \leq V_{max}$ are added to the graph to satisfy the actuation constraint. The main highlights of this method are:

- given a node $v_i = (\mathbf{x}_i, t_i)$, for every location \mathbf{x}_j who is a neighbor of v_i in the spatial domain, multiple nodes with $t_j = [t_i + \Delta t, t_i + 2\Delta t, \dots, t_i + N_{thops} \Delta t]$ are considered as neighbors in the 3-D graph (see Fig. 6).
- while it is possible to consider a large number of nodes with the same spatial coordinate \mathbf{x} with different time coordinates as neighbors of a node, only a N_{thops} number of time hops are considered, to avoid getting incorrect results.

4.2 Extensions of the optimal path planning problem

The objective of the problem specified in (7) is to find an optimal path to the goal, with a specific start time t_s , which satisfy the actuation constraint. The termination time at the goal t_g is free. However, in practical applications, several variations of the problem are encountered. In this section, two such variations are considered and the modifications required in the MTS method to handle such variations are introduced. *Constrained time problem:* In this problem, a specific time t_g or a time window $T_G = [t_g, \bar{t}_g]$ to arrive at the goal is specified. The objective is to find an optimal path from the start to the goal that terminates at the goal \mathbf{x}_g exactly at t_g (or within T_G if a time window is specified). Such problems are applicable in data gathering scenarios where an AMV is required to reach a data gathering buoy at a specified time. To solve this

Algorithm 1: Multi Timestep Search (MTS) to compute optimal paths in time-varying flows

```

Input : Vertex set  $\mathcal{V}$ , Start vertex  $v_s = (\mathbf{x}_s, t_s)$ , Goal  $\mathbf{x}_g$ 
Output: Optimal cost path  $\Gamma$ 
1 foreach  $v_i \in \mathcal{V}$  do
2    $v_i.f = \infty, v_i.cost = \infty, v_i.parent = \emptyset$ 
3    $v_i.heuristic = \text{getHeuristic}(v_i, \mathbf{x}_g)$ 
4    $v_i.expanded = \text{false}$ 
5 end
6  $Q = \emptyset, \mathcal{G} = \emptyset$ 
7  $v_s.f = 0, v_s.cost = 0$ 
8  $Q.insert(v_s), \mathcal{G.addNode}(v_s)$ 
9 while ( $Q \neq \emptyset$ ) do
10   $v_i = Q.extractMin()$ 
11   $v_i.expanded = \text{true}$ 
12   $[V_{f_i}, \theta] = \text{getFlow}(v_i)$  // get velocity at  $v_i$ 
13  if ( $\mathbf{x}_i == \mathbf{x}_g$ ) then // goal is reached
14    while  $v_i \neq \emptyset$  do // retrieve path
15       $\Gamma.push(v_i)$ 
16       $v_i = v_i.parent$ 
17    end
18    break;
19  end
20  foreach  $v_j \in \mathcal{N}(v_i)$  do // for each neighbor
21     $V_{still} = \text{computeVstill}(v_i, v_j)$ 
22    if ( $V_{still} < V_{max}$ ) then
23      continue;
24    end
25     $cost = \text{computeCost}(v_i, v_j)$  // cost of  $\overline{v_i v_j}$ 
26    if  $v_j \in \mathcal{G}$  then
27      if ( $\neg v_j.expanded \ \& \ v_j.cost > v_i.cost + cost$ ) then
28         $v_j.cost = v_i.cost + cost$ 
29         $v_j.parent = v_i$ 
30         $v_j.f = v_j.cost + v_j.heuristic$ 
31         $Q.update(v_j)$ 
32      end
33    else
34       $v_j.cost = v_i.cost + cost$ 
35       $v_j.parent = v_i$ 
36       $v_j.f = v_j.cost + v_j.heuristic$ 
37       $Q.insert(v_j), \mathcal{G.addNode}(v_j)$ 
38    end
39  end
40 end
41 return  $\Gamma$ 

```

problem, the graph search is carried out backwards in time, starting from the goal vertex $v_g = (\mathbf{x}_g, t_g)$. If a time window T_G is specified, multiple goal vertices $v_{g_i} = (\mathbf{x}_g, t_{g_i})$ are considered, where $t_{g_i} \in T_G$ are discretized time coordinates within the specified time window. To carry out a backward time search, all flow velocity directions are reversed, and only vertices with $t_j < t_i$, are considered in the neighbor set $\mathcal{N}(v_i)$ of vertex v_i .

Bounded velocity problem: In this problem, a maximum net speed $V_{net,max}$ is specified. Thus, an additional constraint $V_{net} \leq V_{net,max}$ is added to the problem in (7). Such problems are applicable in data sampling scenarios where the AMV has to travel with a bounded speed for effective data gath-

ering. Such constraints can be easily handled with the MTS method by pruning the neighbor set $\mathcal{N}(v_i)$ so that edges with fast traversal speeds are ignored, i.e., if $\|\mathbf{x}_j - \mathbf{x}_i\|/|t_j - t_i| > V_{net,max}$ for any $v_j \in \mathcal{N}(v_i)$, that neighbor is removed from the neighbor set.

Remark 1 The MTS method presented above is transparent to the cost function used for the optimization. It is general enough to accommodate any path dependent time-varying cost function. Thus, it is possible to use a combined time-energy cost function, that considers the hotel load (energy required to run the sensing and computing systems of the vehicle), in addition to the energy required to overcome drag. Such a cost function would have the form $dc_{3,energy} = (\kappa_h + \kappa \left(\frac{dx^2+dy^2}{dt^2} - 2V_f \frac{dx}{dt} + V_f^2 \right)^{\frac{\alpha}{2}} dt)$, where κ_h is the hotel power draw. This cost function will penalize time spent and result in shorter duration paths.

5 Planning optimal trajectories in different topological classes

In multi-robot applications such as exploration or data collection in the ocean, it is vital that the team of surface vehicles can be distributed effectively across the region of interest. This task is made particularly challenging in the presence of obstacles or fluidic structures. An effective approach to this problem is topological exploration (Kim et al. 2013), where robots are assigned different topological classes of trajectories to disperse into and explore (see Fig. 7). Reasoning about topological classes of trajectories is also vital to automated surface cleaning operations using surface vehicles (Bhattacharya et al. 2015). In this section we demonstrate, how the graph based approaches presented in Sects. 3 and 4 can be modified to perform such topological path planning.

Presence of obstacles in \mathbb{W} give rise to multiple topological classes of trajectories, each of which belong to a unique homotopy class. Since we plan trajectories in a graph, our method lends itself quite naturally to *homotopy-aware path planning*, where we can compute optimal trajectories restricted to different homotopy classes. The basic algorithm for doing this is outlined in Bhattacharya et al. (2012), Kim et al. (2014). The fundamental idea is to use certain *homotopy invariants* (called *h-signature*) (Bhattacharya and Ghrist 2015) to construct a homotopy-augmented graph, $\mathcal{G}_h = (\mathcal{V}_h, E_h)$, in which every vertex is a pair of the form $(v_m, \mathfrak{w}) \in \mathcal{V}_h$, with $v_m \in \mathcal{V}$ and \mathfrak{w} is a “word” made up of letters associated with non-intersecting rays emanating from connected components of obstacles (see Fig. 8). The edge set, E_h is described incrementally as follows: If $(v_m, \mathfrak{w}) \in \mathcal{V}_h$, then for every $[v_m, v_n] \in E$ (where $v_m, v_n \in \mathcal{V}$), there exists an edge $[(v_m, \mathfrak{w}), (v_n, \mathfrak{w} \circ h(\overline{v_m v_n}))] \in E_h$ (where $\mathfrak{w} \circ h(\overline{v_m v_n})$ indicate concatenation of the words \mathfrak{w} and the *h-signature* of

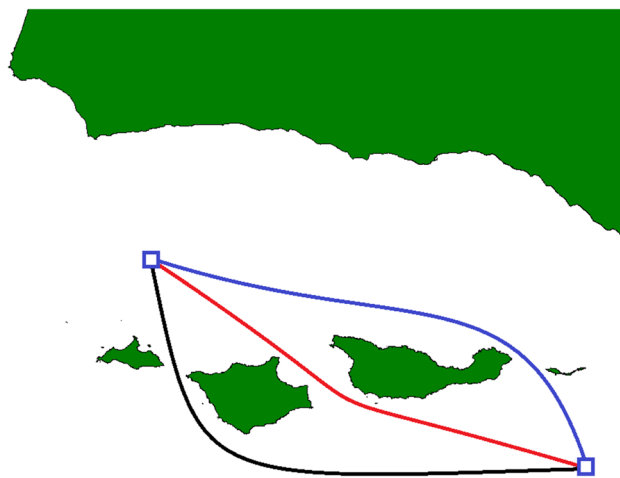


Fig. 7 Homotopy-aware path planning: Simultaneous exploration/sampling around a set of islands by a set of ASVs. Homotopy aware planning allows the generation of optimal paths in different classes of trajectories simultaneously

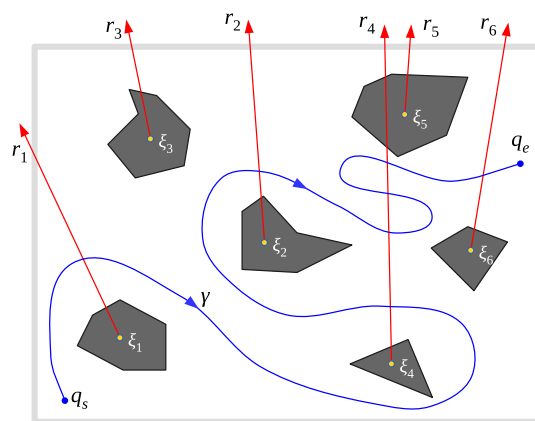


Fig. 8 Non-intersecting rays emanate from every connected component of obstacles. The homotopy invariant (*h-signature*) of a path, γ , is constructed by tracing the curve and constructing a word, in which we insert a letter or its inverse everytime we cross a ray from right-to-left or left-to-right respectively. Thus, in the figure, $h(\gamma) = “r_1^{-1}r_4 r_2^{-1}r_4^{-1}r_6^{-1}”$

the directed segment $\overline{v_m v_n}$). The cost of an edge in \mathcal{G}_h is taken to be same as the cost of the corresponding projected edges in \mathcal{G} . Executing a graph search in the *h-augmented graph*, \mathcal{G}_h , can give us optimal trajectories constrained to different homotopy classes connecting given vertices $v_s, v_e \in \mathcal{V}$.

6 Simulations

In this section we present simulation results for planning time and energy optimal paths in a flow field using the graph search methods described in the preceding sections. In these simulations we use flow velocity information obtained from an analytical model as well as from ocean current forecasts. We use the wind-driven double gyre model as our analytical model because it is often used to describe large scale recircu-

lation regions in the ocean (Veronis 1966). Furthermore, the flows in our experimental setup are also patterned according to this model. In this model, the flow velocity components in the global coordinate system are given by,

$$\dot{X} = -\pi A \sin\left(\pi \frac{X}{s}\right) \cos\left(\pi \frac{Y}{s}\right), \tag{16a}$$

$$\dot{Y} = \pi A \cos\left(\pi \frac{X}{s}\right) \sin\left(\pi \frac{Y}{s}\right), \tag{16b}$$

where A determines the amplitude of the flow velocity vector and s scales the dimensions of the gyres. In our simulations, we set $A = 0.02$ and $s = 1$ so that the flow field approximately matches the flow field generated in the experimental setup with an average flow velocity of 0.025 m/s. The maximum flow speed obtained with these parameters is $V_{fm} = 0.063$ m/s, and as such, we selected $V_{max} = 0.05$ m/s. Note that the flow field generated with (16) is static and as such it is only used for simulations relating to Sect. 3.

We also employ flow data generated by the Regional Ocean Model System (ROMS) for the Santa Barbara Bay area off the coast of southern California in these simulations. The Southern California Coastal Ocean Observing System (SCCOOS) generates these hourly ocean current forecasts on a 3 km grid everyday and each forecast is for 72 hours (SCCOOS). The the data generated on July 7 2016 was used for the simulations. The maximum flow speed observed was $V_{fm} = 0.73$ m/s and as such V_{max} was selected to be 0.5 m/s. When ocean data is used for simulations involving static flows, one time slice of the available data is used.

The accuracy of the computed paths was evaluated by comparing them against paths obtained by solving the corresponding optimal control problem. The optimal control problem involves minimizing the path cost given in (7), subject to the kinematic model in (2), and the constraint $V_{still} \leq V_{max}$. This minimization was done using the Pontryagin’s minimization principle, which resulted 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

$$mE = \int_{t_s}^{t_g} \frac{\|\Gamma^*(t) - \Gamma(t)\|}{t_g - t_s} dt \tag{17}$$

is used to evaluate the relative accuracy of a given path. To further validate the results, the cost of the path computed by the proposed method was also compared against the cost of

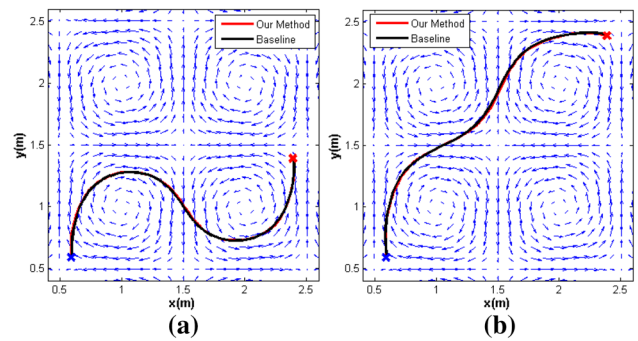


Fig. 9 Planned optimal time paths for two start-goal combinations for $V_{max} = 0.05$ m/s. In red: paths computed using the described method, in black: paths computed using the optimal control formulation. Performance parameter are given in Table 1 as **a** Path 1 and **b** Path 2 (Color figure online)

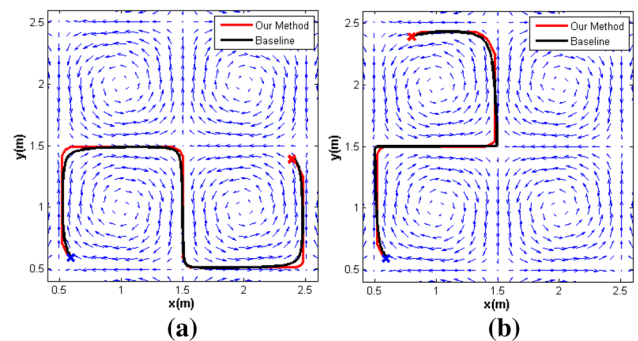


Fig. 10 Planned optimal energy paths for two start-goal combinations for $V_{max} = 0.05$ m/s. In red: paths computed using the described method, in black: paths computed using the optimal control formulation. Performance parameter are given in Table 2 as **a** Path 1 and **b** Path 2 (Color figure online)

the feasible minimum distance path between the same start and goal locations. To compute the straight line path, it was assumed that the vehicle operates at its maximum allowable speed V_{max} . In all of the proceeding simulations involving optimal energy paths, the parameters values $\alpha = 2$ and $\kappa = 1$ are used.

6.1 Optimal paths in static flow fields

Figures 9 and 10 respectively show the optimal time and energy paths planned on a static flow given by (16). In these simulations, both x and Y directions were discretized with a resolution of 0.01 m. In both figures, the traces in red are the paths computed using our method and the traces in black represent the ‘reference’ path computed using the optimal control formulation. Tables 1 and 2 provide the performance parameters for the optimal time and optimal energy paths respectively. It can be seen that paths costs are in good agreement with the reference path and that the mean error is low (approximately 0.02 m for all four paths). In addition, the computed paths have much better costs when compared with

Table 1 Performance parameters for the optimal time paths shown in Fig. 9

	Cost (s)			mErr (m)
	Graph	Reference	Min. Dist	
Path 1 (9(a))	32.92	32.87	248.5	0.019
Path 2 (9(b))	30.17	30.11	88.71	0.019

Table 2 Performance parameters for the optimal energy paths shown in Fig. 10

	Cost (J)			mErr (m)
	Graph	Reference	Min. Dist	
Path 1 (10(a))	9.44e−4	9.05e−4	6.21e−1	0.019
Path 2 (10(b))	2.05e−3	1.61e−3	3.78e−1	0.02

a straight line paths between the corresponding start and goal coordinates.

6.1.1 Approximate Riemannian metric model for path planning

The time and energy minimizing Riemannian metric tensors developed in Sect. 3.2 were also used to generate time and energy optimal paths in static flows. Finding a path from the start position to the goal position translates to solving the two point boundary value problem posed by the geodesic equation. This was achieved by the *shooting method*: integrating the geodesic equation from the start position for a succession of initial directions until the goal position was reached, i.e., the problem was solved as a series of initial value problems.

Figure 11 shows the comparison between the graph search method and the geodesic integration method, for $V_{max} = 0.01$ m/s. Figure 11a shows an instance where the graph search method considers 16 neighbors for each node in the graph. Clearly the geodesic and the graph search generated paths do not agree. The cost of the path generated by the graph search is 1.96. In Fig. 11b, where the results of the two methods are in agreement, the graph search method considers 48 neighbors for each node in the graph, and the resulting path cost is only 1.53 (which is less than the previous case). However, in this instance, the graph search method takes around 50s to compute the path while the geodesic method takes less than 2 s. This example clearly highlights three important advantages of the geodesic integration method: (i) it produces smooth trajectories (see Fig. 11a), (ii) in contrast to graph search methods, the accuracy of the generated path is not limited by the resolution of the underlying workspace, and (iii) the geodesic integration method is much faster.

However, it was observed that for some instances, the geodesic integration method gave undesirable results. Fig-

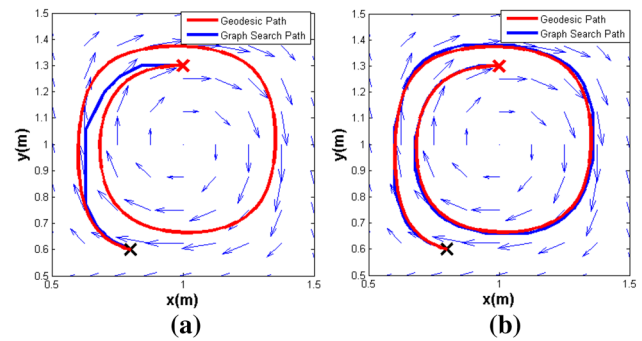


Fig. 11 Comparison of optimal energy paths generated by the graph search method (blue) with the path generated by integrating the geodesic equation (red). **a** Coarse graph constructed by connecting each vertex with 16 neighbors: Paths not in agreement and the cost of the path in graph is significantly worse, **b** Finer graph constructed by connecting each vertex with 48 neighbors: Paths in agreement and produces a lower cost path (Color figure online)

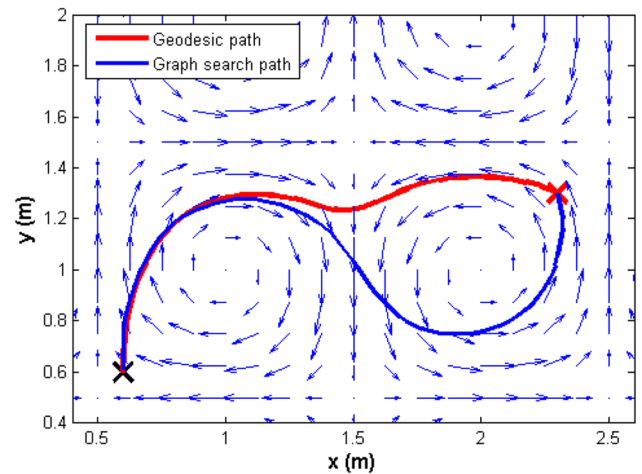


Fig. 12 Comparison of exact optimal cost paths (blue) with the paths generated by integrating the geodesic equations obtained from the approximate Riemannian metric model (red). In this instance the Riemannian metric model fails to generate a feasible path (Color figure online)

ure 12 shows the optimal time path obtained from geodesic integration for $V_{max} = 0.05$ m/s. The generated path is not necessarily a time-optimal path since parts of it fall along flow opposing directions. This occurs due to the symmetry of the Riemannian metric, where the cost heading against the flow is equal to the cost heading along the flow (see discussion in Sect. 3.2). However, a method can be developed to construct a piecewise geodesic path from the start to the goal which ignores flow opposing directions. This is a direction for future work.

6.1.2 Performance comparison with existing methods

We compared the proposed method with (i) a level set method for planning optimal time paths (Lolla et al. 2014), (ii) a

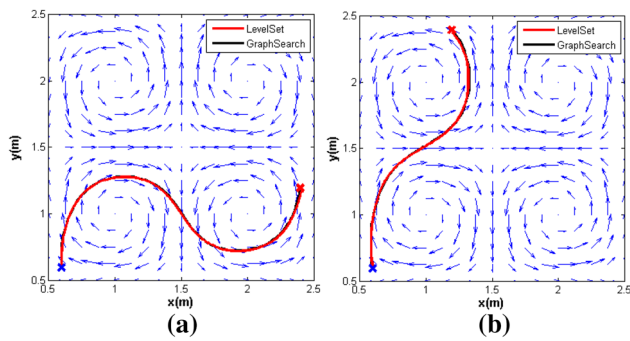


Fig. 13 Comparison of time optimal paths obtained from the level set method (red) with paths from the graph search method with 48-neighbors (black). The costs for **a** path 1 and **b** path 2 are given in Table 3, and are comparable (Color figure online)

graph search based method for planning optimal energy paths (Koay and Chitre 2013).

Level set method for optimal time paths: In this method introduced by Lolla et al., a virtual front (a zero level set of a distance function), that is initialized around the start point, is advected by the flow as well as by a velocity component perpendicular to the front, until it reaches the goal position. The point on the front that reached the goal position is then integrated backwards in time to find the optimal time path. We implemented a first order integration to compute time optimal path using the level-set method. To compute the path costs, we use the minimum-time cost function in (9), on the discretized level set path. As seen in Fig. 13 and Table 3, the paths and the corresponding path costs obtained from the two methods are almost identical.

It can be seen that the proposed method produces comparable results to those obtained from the level set method. However, for a workspace containing n grid points, the computational cost of the level set method is $\mathcal{O}(n^3)$ (Lolla et al. 2014) while for graph search method with E connected neighbors, the computational cost is only $\mathcal{O}(nE \log(n))$ in the worst case. Furthermore, the level set method periodically performs a re-initialization step, which has a computational cost of $\mathcal{O}(n^3)$. In addition to lower computational cost, a further advantage of our method is the ability to incorporate different cost functions, whereas the level set method can only generate time optimal paths.

Comparison with a graph search method for optimal energy paths: We compared the optimal energy paths generated from our method with another graph search based method presented by Koay et al. (2013) (henceforth referred to as the Koay method). Even though the approach is similar, the cost function proposed in the Koay method is approximate. In their method, the authors minimize \mathbf{v}_{still} for each path segment in the graph and use this minimized value to compute the energy cost for that path segment. In effect, the cost of each path segment is computed to be

Table 3 Comparison of path costs against the level set method for time optimal paths. The path costs obtained from the two methods are comparable

	Level Set	Graph(48-con.)	Graph(16-con.)
Path 1	30.39	30.53	31.07
Path 2	22.47	22.76	23.07

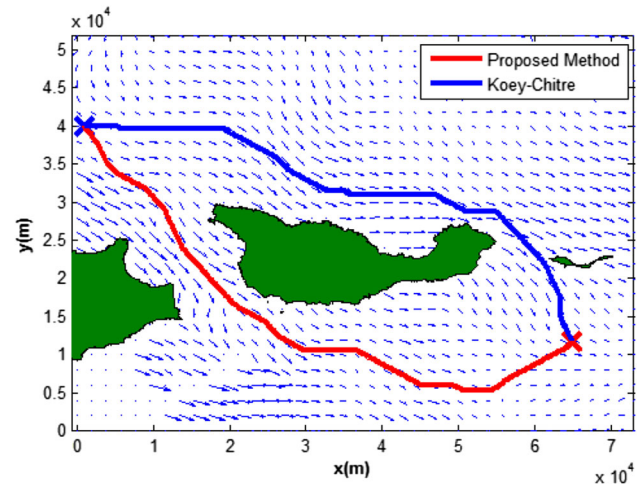


Fig. 14 Comparison of optimal energy paths generated using the proposed method and the Koay method. Path costs: our method (red path): 1802 (2097) J, Koay method (blue path): 1864 (1942) J. The values outside parenthesis are the energy values computed using the exact energy cost (11), while in parenthesis are the path costs calculated using the approximate cost metric used in the Koay method (Color figure online)

$\int \kappa \|\min(\mathbf{v}_{still})\|^2 dt$ whereas the actual energy cost should be $\min \int \kappa \|\mathbf{v}_{still}\|^2 dt$. Figure 14 shows the “optimal” energy paths computed using the two methods near the Santa Cruz island. The cost of the path generated from the Koay method, when evaluated using the cost function given in (11), is 3% higher than the actual optimal path, and it lies on the opposite side of the island. Furthermore, the cost of the actual optimal energy path (generated using the proposed method), is 16% higher when evaluated using the cost metric used in the Koay method.

6.2 Optimal paths in time-varying flow fields

In this section we present simulation results for computing optimal paths in time varying flows using the MTS method described in Sect. 4. Figure 15 shows the time evolution of an optimal time path computed using the MTS method (in red), along with the reference path $\Gamma^*(t)$ computed using the optimal control formulation (in black). In this case, since the objective is to minimize the traversal time, the path takes a direct route towards the destination, while trying to align with the flow wherever possible to get an additional ‘push’. Figure 16 shows the time evolution of an optimal energy

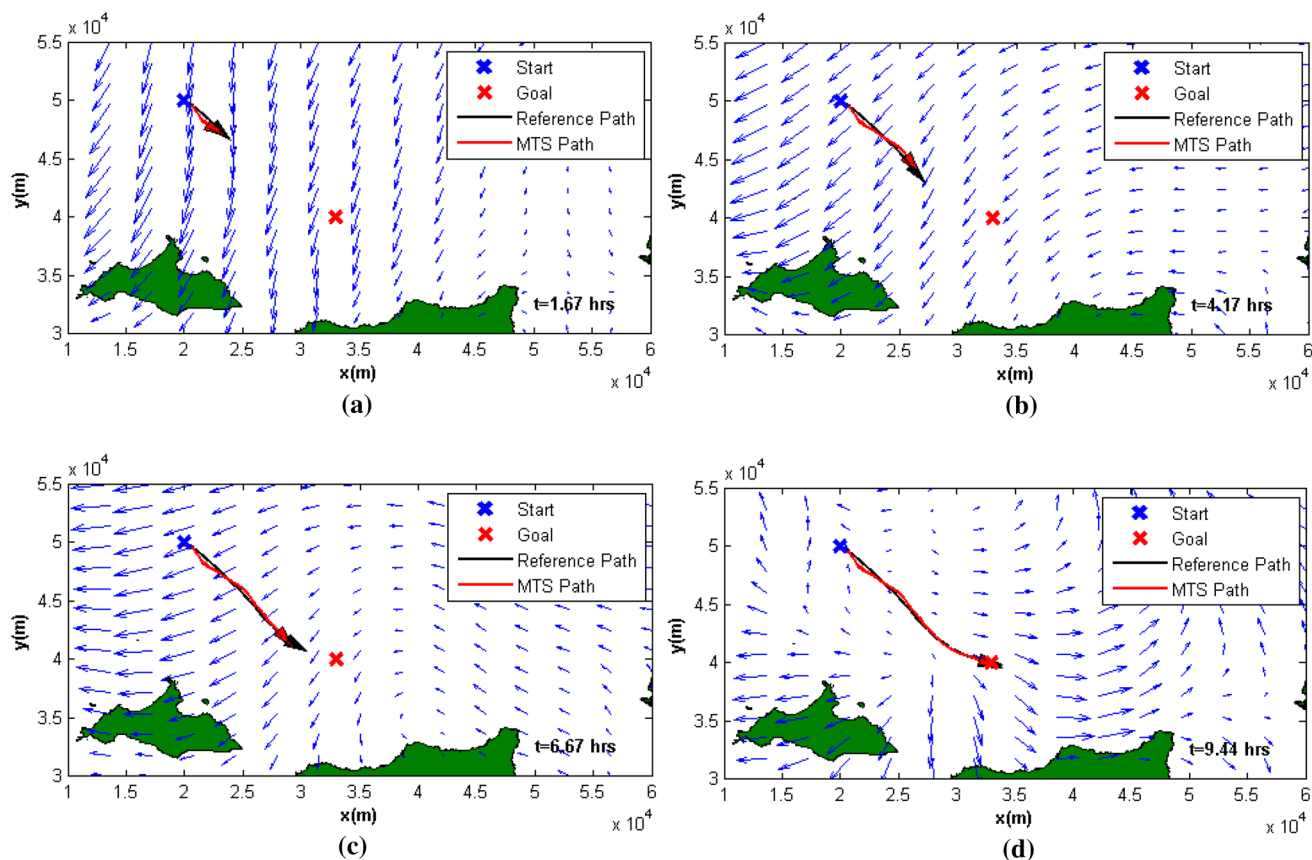


Fig. 15 Comparison of an optimal time path computed by the MTS method (red trace) against the optimal time path computed using the optimal control formulation (black trace). Paths take a more direct route

towards the goal. Refer Table 4 for performance parameters of the path. **a** $t = 1.67$ h. **b** $t = 4.17$ h. **c** $t = 6.67$ h. **d** $t = 9.44$ h (Color figure online)

path computed using the MTS method (in red), along with the corresponding reference path computed using the optimal control formulation (in black). In this case, the objective is to reduce the energy spent while traversing. As such, the path tries to reduce the drag force experienced by the vehicle, by aligning the vehicle's motion along the flow as much as possible. This results in loop structures (in space) as can be seen in the figure. The costs and the mean errors of these paths are given in Table 4.

Flow reversal is a common occurrence in marine environments. This is particularly true in channels between islands and in estuarine environments. Figure 17 shows an example of the MTS method being used to plan optimal energy paths in such a flow reversing environment. It can be seen that the flow between the two islands (Santa Rosa island and Santa Cruz island) is initially opposing the vehicle's intended travel direction. Thus the vehicle slows down and waits until it encounters a more favorable current toward the destination. This waiting can be clearly seen in the 3D time extended plot in Fig. 17f.

In Sect. 4.2, two extensions of the path planning problem in time-varying flows, the constrained time problem and the bounded velocity problem, were introduced. The solution methods proposed in that section were used to compute optimal energy paths between the same start and goal locations as before, i.e., from $\mathbf{x}_s = [20, 50]$ km to $\mathbf{x}_g = [50, 40]$ km. For the constrained time problem, a time window $T_G = [43.61, 46.94]$ h was considered, and the resulting optimal path is shown in Fig. 18. The computed path had a start time of 8.89 h and the time at goal was 46.94 h. Furthermore, it can be seen that this path better utilizes the flow to get a path cost of 1816 J, which is much less than the cost of the path starting at $t_s = 0$ h. For the bounded velocity problem, the maximum allowable speed $V_{net,max}$ was set to 0.4 m/s and the modified MTS method was used to compute an optimal energy path. The computed path had cost of 3914 J which is higher than the cost of the optimal energy path with no velocity bounds (see Table 4). This is expected because the velocity bound prevents the vehicle from approaching areas with faster currents, which are more conducive to energy efficient navigation. Furthermore the maximum net velocity

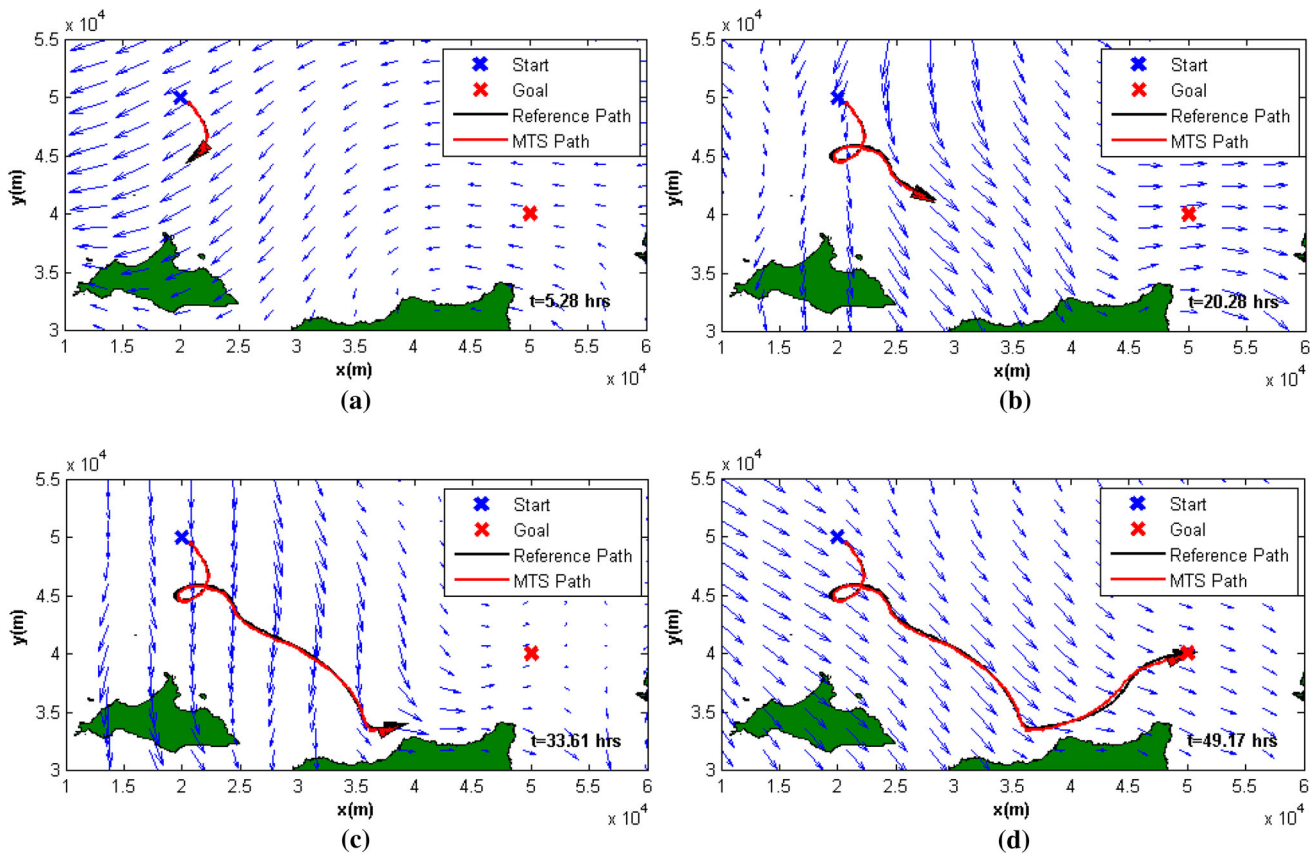


Fig. 16 Comparison of an optimal energy path computed by the MTS (red trace) method against the optimal energy path computed using the optimal control formulation (black trace). Paths try to follow the flow

at all times in order to reduce energy consumption. Refer Table 4 for performance parameters of the path. **a** $t = 5.28$ h. **b** $t = 20.28$ h. **c** $t = 33.61$ h. **d** $t = 49.17$ h (Color figure online)

Table 4 Performance parameters for the optimal time paths shown in Figs. 15 and 16

	Cost			mErr (m)
	Graph	Reference	Min. Dist.	
Opt. Time (15)	9.44 h	8.66 h	9.48 h	893
Opt. Energy (16)	3858 J	3795 J	16218 J	394

along the computed path was 0.36 m/s which is within the specified bound.

In the method described in Sect. 3 as well as in other graph based methods used to compute optimal paths in static flows (Koay and Chitre 2013; Garau et al. 2005), in which only the spatial coordinates are discretized, the accuracy of the computed paths increases as the resolution of the discretization is increased. However, it was observed that this is not generally true for time-varying flows. Table 5 shows the variation of the accuracy values for an optimal energy path computed using the MTS method, as the resolution used to

discretize the 3D space is varied. The start and goal locations for these paths are the same as the ones considered in Fig. 16. Each path in the table refers to a result from the MTS method for different discretization levels and different number of neighbors in the neighbor set. Going from Path1 to Path2 and from Path3 to Path5, where Δt is held constant, the accuracy is increased (mE is reduced) as Δx is reduced. However, for Path4 and Path5, the accuracy decreases (mE is almost doubled) as Δx is reduced (as the discretization is made finer). Thus, the relationship between the accuracy and the discretization resolution for time varying flows is not as straight forward as it is for static flows. In Path5 - Path7, both discretization resolutions are held constant while reducing N_{tHops} . It can be seen that the error is the same for Paths 5 and 7, while the error increases in Path6 when $N_{tHops} = 5$. These observations indicate that the discretization resolution needs be adapted according to the spatio-temporal scales of the underlying flow at each expansion of the graph. This is a research direction that is currently being investigated.

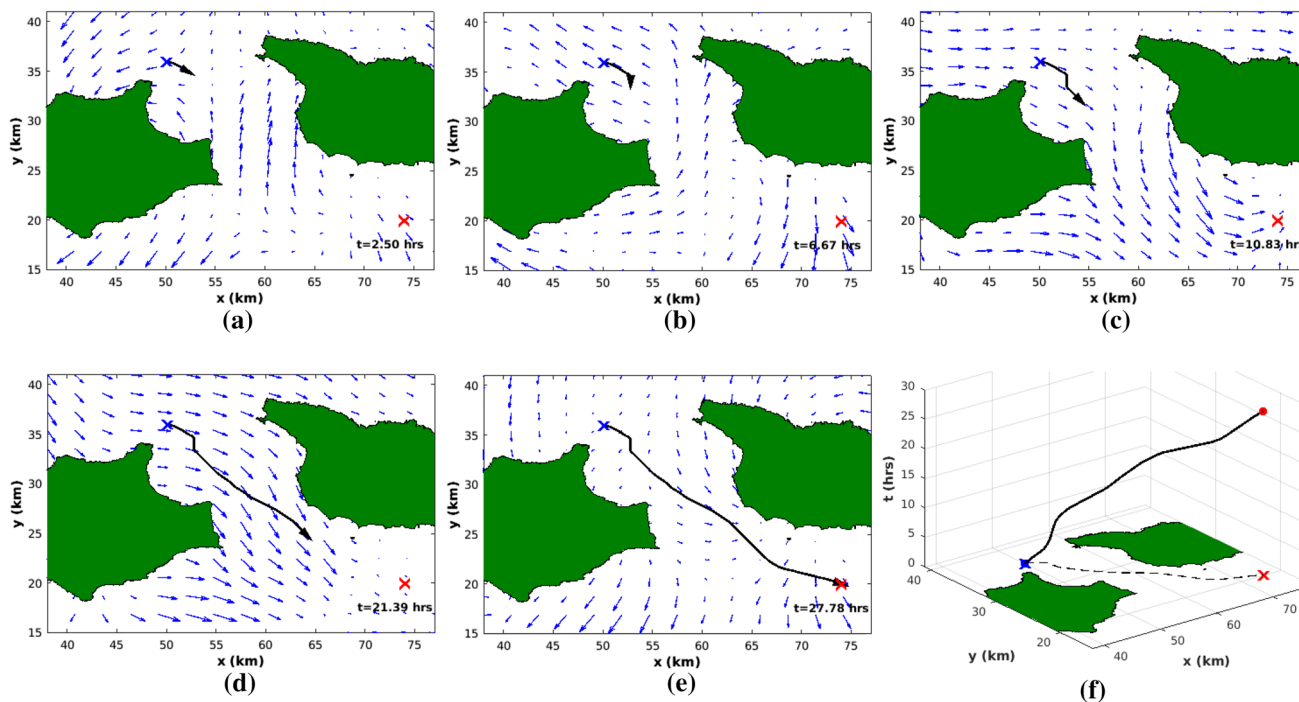


Fig. 17 Path planning in a flow reversal scenario. Initially, the flow is opposing the intended travel direction (a)–(c). The vehicle waits for until the flow is more favorable to proceed (d, e). The waiting can be seen clearly in the time extended plot (f)

6.3 Planning in multiple homotopy classes

As described in Sect. 5, the proposed path planning framework can be used to plan optimal trajectories in different homotopy classes. This approach can be used for simultaneous navigation in regions riddled with obstacles, e.g., the Philippine archipelago. Furthermore, as mentioned in Sect. 5, this approach is particularly attractive for energy efficient exploration/sampling of ocean phenomena around a set of islands using a team of ASVs. This approach generates optimal paths along each coastal segment off an island. Figure 19a shows optimal energy paths generated around the Santa Cruz island and Fig. 19b shows optimal energy paths generated around the Anacapa island, both off the coast of California. In both cases, the blue path shows the globally cost optimal path between the start and goal locations, and the red path shows a cost optimal path that is in a different homotopy class (i.e., on the opposite side of the island). For this simulation, the flow was assumed to be static (a single time slice of the available flow data was used in the computations), and the environment was discretized with a resolution of 125 m.

7 Experimental verification

We further validate the paths computed by our proposed graph search method through experiments. The experiments were conducted using the multirobot Coherent Structure

Testbed (mCoSTe) which consists of a micro Autonomous Surface Vehicle (mASV) and the $3 \times 3 \times 1 \text{ m}^3$ Multi Robot flow tank (MR tank). The mASV is a differential drive vehicle with a maximum forward speed of approximately 0.2 m/s. Localization for the vehicle is provided by an external motion capture system. Multi-gyre flows were created in the MR tank using four flow driving cylinders rotating at approximately 100 rpm. Figure 20 shows the components of the experimental setup.

The flow field generated in the tank using the flow driving cylinders, closely resembles the flow given by the double-gyre model given in (16). In fact, the parameters of the model were selected such that average flow speed of the model matches the average flow speed observed in the tank (0.25 m/s). The average flow speed in the tank was computed by logging the uncontrolled mASV speed in the flow and averaging those values. According to the model, the selected parameters result in a maximum flow speed of 0.063 m/s. Note while there are subtle disturbances to the flow, it is assumed that the flow generated in the tank is a static flow. As such we used the methodology in Sect. 3 to compute optimal paths between the start location $[0.6, 0.6]^T$ and goal location $[2.3, 1.3]^T$. The paths were computed on a graph discretized at with a resolution of 0.01 m. For the experiments, we considered three paths between the start and goal locations: the time-optimal path, the energy-optimal path and the minimum-distance path. The minimum-distance path was considered in order to provide a baseline to compare the

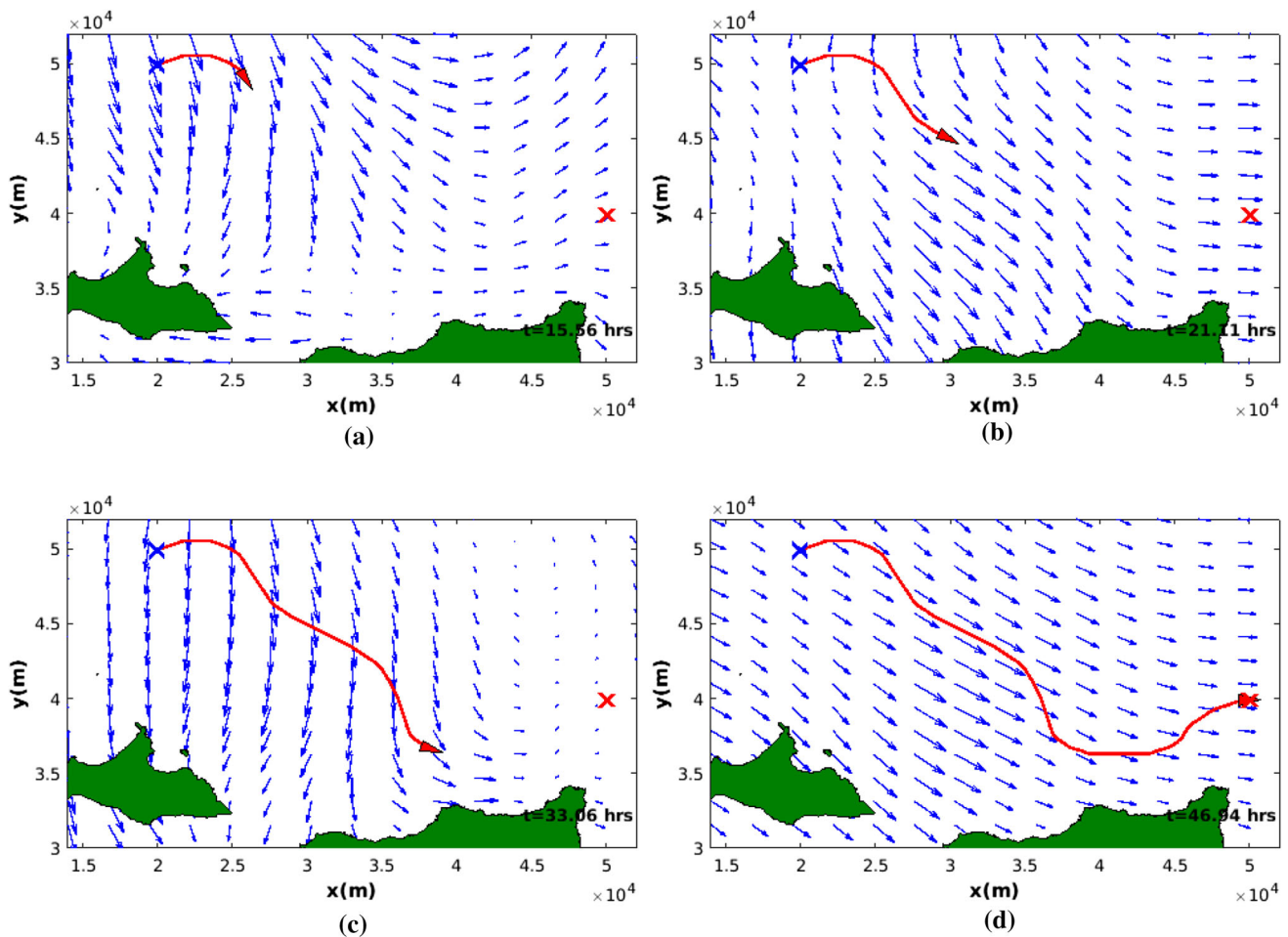


Fig. 18 Optimal energy path computed between $\mathbf{x}_s = [20, 50]$ km and $\mathbf{x}_g = [50, 40]$ km with a constrained time window $T_G = [43.6146.94]$ h at the goal. The time at goal is 46.94 h and the start time from the start location is 8.89 h. **a** $t = 15.56$ h. **b** $t = 21.11$ h. **c** $t = 33.06$ h. **d** $t = 46.94$ h

Table 5 The variation of the path accuracy with discretization resolution for the MTS method, for an optimal energy path computed from $\mathbf{x}_s = [20, 50]$ km, to $\mathbf{x}_g = [50, 40]$ km

	Path1	Path2	Path3	Path4	Path5	Path6	Path7
Δt (s)	1000	1000	500	500	500	500	500
Δx (m)	200	150	200	100	150	150	150
N_{tHops}	3	3	6	6	6	5	3
N_{sHops}	3	3	3	3	3	3	3
mE	801	660	410	747	394	521	394

results of the time-optimal and energy-optimal paths. During the experiments, the paths computed by the graph based planner was used as a reference paths for the ASV to follow. The path following was achieved using a waypoint navigation scheme. Seven runs were carried out for each case and results were recorded. The energy expended by the mASV was computed as,

$$E = \int (V_{Lmotor}^2 + V_{Rmotor}^2) dt \approx \frac{V_{in}^2}{255^2} \Delta t \sum_i (L_{cmd_i}^2 + R_{cmd_i}^2)$$

where V_{Lmotor} and V_{Rmotor} are the voltages applied to the left and right motors respectively, V_{in} is the battery voltage, L_{cmd_i} and R_{cmd_i} are the PWM signal commands sent out to the mASV during the i^{th} command cycle, and Δt is the duration of each command cycle.

Table 6 shows the average time taken and the average energy expended by the mASV to complete each path. As expected, the mASV completes the time-optimal path the quickest and expends the least amount of energy while completing the energy-optimal path. On average the time-optimal path is 19% faster than the minimum-distance path and the energy-optimal path expends 51% less energy than the minimum-distance path. Figure 21 shows the paths taken

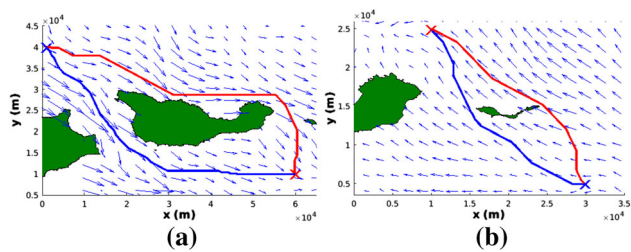


Fig. 19 **a** Optimal Energy paths around Santa Cruz island. Path costs, blue: 1607 J, red: 2871 J. **b** Optimal Energy paths around Anacapa island. Path costs, blue: 396 J, red: 1390 J. In both cases, the blue path is the globally optimal path while the red path is optimal on the opposing side of the island. The blue cross gives the start location and the red cross gives the goal location (Color figure online)

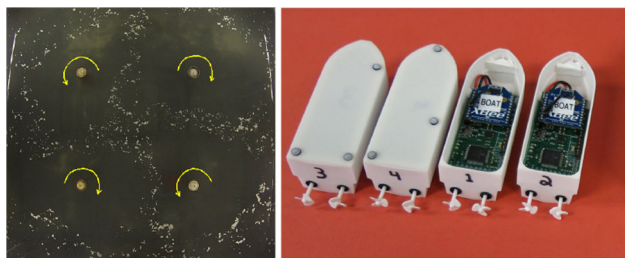


Fig. 20 Experimental setup, right: four flow driving cylinders creating a double gyre like flow in the MR tank, left: mASVs used for the experiments

Table 6 Path costs for the three paths averaged over seven trials. The standard deviations of the computed values are shown within brackets

	Time optimal	Energy optimal	Min distance
Time (s)	44.8 (1.92)	92.7 (4.64)	55.4 (8.31)
Energy (J)	250.3 (33.2)	222.3 (12.7)	449.3 (42.1)

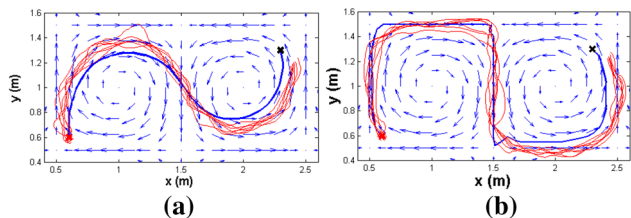


Fig. 21 Paths taken by the mASV in the MR tank (in red) and the paths planned by the graph search algorithm (in blue). **a** time-optimal path. **b** Energy-optimal path (Color figure online)

by the mASV against the path planned by the graph search method and Fig. 22 shows the path taken by the mASV overlaid on an image of the tank.

8 Conclusions and discussion

In this work we presented graph search based methods to plan optimal time and optimal energy trajectories in both static

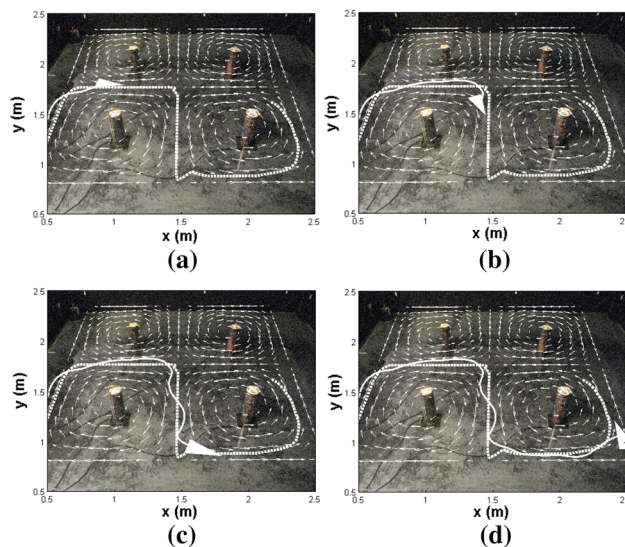


Fig. 22 Path taken by the ASV in the tank while following the optimal energy path. The dotted white line is the reference path, the solid white line shows the path taken by the ASV and the solid triangle represents the ASV. The flow in the tank is assumed to be static. **a** $t = 20$ s. **b** $t = 40$ s. **c** $t = 60$ s. **d** $t = 80$ s (Color figure online)

and time varying flow fields. We used tools from topological path planning to extend these graphs in order to compute optimal paths in different homotopy classes of the environment. We were able to verify the correctness of the computed paths in simulations, by comparing them with paths computed using an optimal control framework. Furthermore, we compared our results of our method with other methods in literature. The level-set method, although being computationally expensive and being able to compute time-optimal paths only, gave comparable time-optimal path costs; and the Koay method for energy optimal paths gave more expensive paths than ours. We also verified the efficacy of the planned paths through experiments conducted in an indoor laboratory testbed, where, despite having partial/noisy flow information, our method was successful in planning optimal trajectories.

We also presented a Riemannian approximations to the cost functions used for the graph search method in static flows. We showed how these Riemannian metrics could be used with the geodesic equation to generate approximations to time and energy optimal paths. Even though some drawbacks were observed in this method, we believe that this approach to path planning in a flow not only provides better insights into the problem but also opens up new possibilities for path planning in a flow field, and as a results warrants further investigation.

When planning optimal paths in time varying flows, it was found that the accuracy of the computed paths were dependent on the discretization used for the configuration space in a disproportionate manner. In contrast to searches in spatial coordinates alone (such as the ones used in static flow sce-

narios), increasing the discretization resolution sometimes resulted in reduced accuracy. To obtain the best accuracy, the discretization has to match the spatio-temporal scales of the underlying flow field. Finding this “sweet spot” remains a challenge and is a direction for future work.

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 (Huynh et al. 2015). 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.

Funding This study was funded by the National Science Foundation (Grant Numbers IIS-1253917 and CMMI-1462825).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

- Bhattacharya, S., & Ghrist, R. (2015). Path homotopy invariants and their application to optimal trajectory planning. In *Proceedings of IMA conference on mathematics of robotics (IMAMR)*, St Anne’s College, University of Oxford, September 9–11, 2015.
- Bhattacharya, S., Likhachev, M., & Kumar, V. (2012). Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3), 273–290. <https://doi.org/10.1007/s10514-012-9304-1>. ISSN 0929-5593.
- Bhattacharya, S., Kim, S., Heidarsson, H., Sukhatme, G., & Kumar, V. (2015). A topological approach to using cables to separate and manipulate sets of objects. *International Journal of Robotics Research*, 34(6), 799–815. <https://doi.org/10.1177/0278364914562236>.
- Caron, D., Stauffer, B., Moorthi, S., Singh, A., Batalin, M., Graham, E., et al. (2008). Macro- to fine-scale spatial and temporal distributions and dynamics of phytoplankton and their environmental driving forces in a small subalpine lake in southern California, USA. *Journal of Limnology and Oceanography*, 53(5), 2333–2349.
- Chakrabarty, A., & Langelaan, J. (June 2013). Uav flight path planning in time varying complex wind-fields. In *2013 American control conference*, pp. 2568–2574. <https://doi.org/10.1109/ACC.2013.6580221>.
- CORDC. Naitonal HF RADAR network—surface currents. URL <http://cordc.ucsd.edu/projects/mapping/maps/>.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to algorithms* (2nd ed.). Cambridge: MIT Press.
- Eichhorn, M. (2015). Optimal routing strategies for autonomous underwater vehicles in time-varying environment. *Robotics and Autonomous Systems*, 67, 33–43.
- Fiorelli, E., Leonard, N. E., Bhatta, P., Paley, D. A., Bachmayer, R., & Fratantoni, D. M. (2006). Multi-AUV control and adaptive sampling in monterey bay. *IEEE Journal of Oceanic Engineering*, 31(4), 935–948. <https://doi.org/10.1109/JOE.2006.880429>. ISSN 0364-9059.
- Garau, B., Alvarez, A., & Oliver, G. (2005, April). Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an a* approach. In *Proceedings of the 2005 IEEE international conference on robotics and automation, 2005. ICRA 2005*, pp. 194–198. <https://doi.org/10.1109/ROBOT.2005.1570118>.
- Huynh, V., Dunbabin, M., & Smith, R. (2015). Predictive motion planning for auvs subject to strong time-varying currents and forecasting uncertainties. In *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 1144–1151. <https://doi.org/10.1109/ICRA.2015.7139335>.
- Kim, S., Bhattacharya, S., Ghrist, R., & Kumar, V. (2013). Topological exploration of unknown and partially known environments. In *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, November 3–7, 2013.
- Kim, S., Bhattacharya, S., & Kumar, V. (2014). 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.
- Koay, T.-B., Chitre, M. (2013). Energy-efficient path planning for fully propelled auvs in congested coastal waters. In *OCEANS-Bergen, 2013 MTS/IEEE*, pp. 1–9. <https://doi.org/10.1109/OCEANS-Bergen.2013.6608168>.
- Kruger, D., Stolkin, R., Blum, A., & Briganti, J. (April 2007). Optimal auv path planning for extended missions in complex, fast-flowing estuarine environments. In *2007 IEEE international conference on robotics and automation*, pp. 4265–4270. <https://doi.org/10.1109/ROBOT.2007.364135>.
- Kularatne, D., Bhattacharya, S., & Hsieh, M. A. (2016, June). Time and energy optimal path planning in general flows. In *Proceedings of robotics: Science and systems*, AnnArbor, Michigan. <https://doi.org/10.15607/RSS.2016.XII.047>.
- Lolla, T., Ueckermann, M. P., Haley, P., & Lermusiaux, P. F. J. (2012, May). Path planning in time dependent flow fields using level set methods. In *Proceedings of the IEEE international conference on robotics and automation*, Minneapolis, MN, USA.
- Lolla, T., Jr H, P., & Lermusiaux, P. (2015). Path planning in multi-scale ocean flows: Coordination and dynamic obstacles. *Ocean Modelling*, 94, 46–66. <https://doi.org/10.1016/j.ocemod.2015.07.013>. ISSN 1463-5003.
- Lolla, T., Lermusiaux, P. F. J., Ueckermann, M. P., & Haley, P. J. (2014). Time-optimal path planning in dynamic flows using level set equations: theory and schemes. *Ocean Dynamics*, 64(10), 1373–1397. ISSN 1616-7228.
- Otte, M., Silva, W., & Frew, E. (2016, May). Any-time path-planning: Time-varying wind field + moving obstacles. In *2016 IEEE international conference on robotics and automation (ICRA)*, pp. 2575–2582. <https://doi.org/10.1109/ICRA.2016.7487414>.
- Petersen, P. (2006). *Riemannian Geometry*. Graduate Texts in Mathematics. Springer, ISBN 9780387294032.
- Rao, D., & Williams, S. B. (2009). Large-scale path planning for underwater gliders in ocean currents. In *Australasian conference on robotics and automation (ACRA)*, Sydney.
- SCCOOS. Regional ocean model system (roms) model output. URL <http://www.sccoos.org/data/roms-3km/>.
- Smith, R. N., Das, J., Heidarsson, H., Pereira, A., Cetinić, I., Darjany, L., et al. (2010). USC CINAPS builds bridges: Observing and monitoring the Southern California Bight. *IEEE Robotics and Automation Magazine, Special Issue on Marine Robotics Systems*, 17(1), 20–30.
- Subramani, D. N., & Lermusiaux, P. F. (2016). Energy-optimal path planning by stochastic dynamically orthogonal level-set optimization. *Ocean Modelling*, 100, 57–77.

- Veronis, G. (1966). Wind-driven ocean circulation, part i and part ii. *Deep-Sea Research*, 13(31), 17–29.
- Witt, J., & Dunbabin, M. (2008). Go with the flow: Optimal auv path planning in coastal environments. In *Australasian conference on robotics and automation (ACRA)*.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



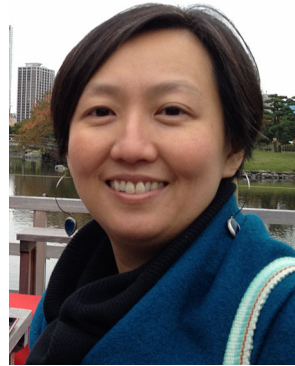
Dhanushka Kularatne received his B.S. degree in Electrical Engineering from University of Peradeniya, Sri Lanka in 2009 and his M.Sc. degree in Mechanical Engineering from the University of Central Florida in 2013. He completed his Ph.D in Mechanical Engineering at Drexel University in 2017, where he focused his research on optimal path planning for marine vehicles. Dhanushka's research interests are centered around developing optimal motion control strategies for autonomous

marine vehicles. Currently he is working on leveraging system noise for efficient navigation, which has applications in monitoring of dynamic and uncertain environments by teams of robots.



Subhrajit Bhattacharya is an assistant professor in the Department of Mechanical Engineering and Mechanics of Lehigh University. He completed Ph.D. in Mechanical Engineering and Applied Mechanics, and was a postdoctoral researcher in the Department of Mathematics at the University of Pennsylvania before he joined Lehigh. Subhrajit's research interests are centered around motion planning and control of autonomous, intelligent systems. More specifically, he is interested

in applications of topological and geometric methods (algebraic/differential topology, differential/discrete geometry) to the design and analysis of algorithms in robot motion planning, coverage, sensor networks, swarms, distributed systems and control.



M. Ani Hsieh is a Research Associate Professor in the Department of Mechanical Engineering & Applied Mechanics, University of Pennsylvania. She received a B.S. in Engineering and B.A. in Economics from Swarthmore College in 1999 and a Ph.D. in Mechanical Engineering from the University of Pennsylvania in 2007. She has over fifty publications spanning her areas of research interest including networked robotics, robot swarms, distributed control, task allocation, distributed assembly, geophysical fluid dynamics, and nonlinear dynamics & control. Her current work focuses on developing a general control and coordination framework for distributed sensing and monitoring of dynamic and uncertain environments by mobile robot teams. Dr. Hsieh is a member of IEEE and serves on editorial boards for journals including *Swarm Intelligence* (Springer), *Frontiers in Robotics and AI*, and the *IEEE Transactions on Automation Science and Engineering*.