Planning for Buoyancy-Controlled Ocean Exploration

Team 6: Michael Szpakowicz, Jacob Ridgway, Sohaib Kotwal

April 23, 2022

Contents

1 Introduction

It is said that we know more about the surface of the moon than we know about our own oceans. This fact remains true even today, with only around 20% of the oceans explored to a high resolution as of 2021 [1]. One barrier that stands in the way of exploring the abyssal deep is high equipment costs. To mitigate this challenge, we propose deploying propulsion-less, buoyancy-controlled exploration probes, known as gliders, allowing for wide area coverage at a much lower cost than traditional methods. For this we must create a motion planning algorithm to passively navigate the probes on ocean currents at various depths.

1.1 Objective

The goal of this project is to demonstrate the feasibility of ocean navigation using buoyancy-controlled glider-like probes by adapting classical path planning approaches to limited-actuation flow fields. If this goal is reached, we intend to expand the scope of our project to find the shortest trajectories that explore the most ocean area. Additionally, we want to demonstrate the possibility of travelling to a specified goal location or region, e.g. to send data, for recovery, or to investigate a site of interest.

2 Background

Currently, there are a wide variety of ocean exploration methods available, the most common of which is multibeam sonar mounted directly on the hulls of ships (Figure 1) [2]. A number of specialized, robot-assisted approaches also exist, such as for exploring beneath sea ice [3], deep sea exploration [4], high-quality mapping in challenging scenarios [4], and multi-agent mapping [5]. However, even limited sensing capability, such as temperature, salinity, pH, and the ocean currents themselves are of great value.

Figure 1: Multibeam Mapping. Image courtesy of NOAA. [2]

BRUIE, or the Buoyant Rover for Under-Ice Exploration, is being developed at JPL for underwater exploration in ice-covered regions on Earth, and in the icy waters of ocean worlds elsewhere in our solar system. The long-term goal is to be able to deploy BRUIE for autonomous operations in an extraterrestrial water world like Europa, where it would search for signs of life at the boundary between the ice shell and ocean. [3]

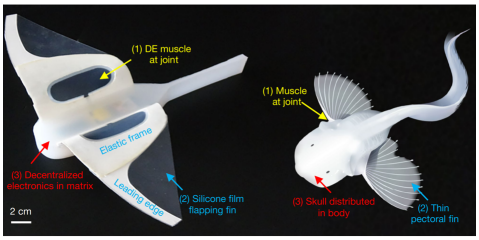

Figure 2: Snailfish Robot [4]

Similarly, a team at Zhejiang University developed an untethered self-powered soft robot for deep-sea exploration. The robot, whose shape inspired by the structure of a deep-sea snailfish, has onboard power, control, and actuation protected from pressure by integrating electronics in a silicone matrix eliminating the need for rigid vessels. This robot completed successful field tests in the Mariana Trench down to a depth of 10,900 metres and to swim freely in the South China Sea at a depth of 3,224 metres. [4]

A team from the University of Girona developed an approach to navigate unknown underwater environments while gathering information for inspections using the Sparus II, a torpedo-shaped autonomous underwater vehicle (AUV). In order to calculate a collision-free path under such constraints (i.e., incrementally and online), they use a modified version of RRT*. [6]

There are many other examples of robotic ocean exploration methods, but the vast majority of them focus on the require the use of motorized systems. Through this project we hope to shed a new light to researching simpler and more deployable methods.

3 Literature Review

This project focuses on the challenges of path planning in strong flow fields for robots with limited actuation strength. There is a large body of work that has approached this issue from many different angles. We used to work below as inspiration.

- To et al. [7] present an improved distance metric for planning in incompressible flow fields using RRT* based on streamlines from the East Australian Current.
- Hou et al. [8] apply cell decomposition with each cell having a constant flow velocity and optimizes using a bounded cost function.
- Heon Lee et al. [9] present an efficient (p-time) algorithm for time-dependent shortest path planning in time-varying flow fields based on graph theory results.
- Lolla et al. [10] employ a modified level set equation to evolve a front from the starting point of a vehicle until it reaches the desired goal location while predicting the time-optimal paths of ocean vehicles in continuous dynamic flows.
- Wang [11] presents an ensemble-based approach that is developed to conduct optimal path planning in two-dimensional unsteady ocean currents under uncertainty.

Some approaches (like the ones below) focus explicitly on planning with limited or zero actuation.

- Krishna et al. [12] explores planning for a robot with a Model Predictive Controller, mobile sensing and limited actuation within a three dimensional flow in the Gulf of Mexico.
- Hansen et al. [13] considers a spatial coverage problem in which a network of passive floating sensors is used to collect samples in a body of water. The paper models flow fields, predict trajectories and proposes and selects deployment points.
- Kularatne et al. [14] applies a graph search based method to plan time and energy optimal paths in a flow field where the kinematic actuation constraints on the vehicles are captured in their cost functions. They also present a Riemannian metric based approximation for these cost functions which provides an alternative method for computing time and energy optimal paths.

There have also been some interesting novel methods exploring this problem. For one, Gunnarson [15] proposes using a learning-based model for point-to-point navigation in vortical flow fields, and explores the impact of different flow patterns. Another interesting method is developed by Ma [5] where they implement a nature-inspired approach by using ant colony optimization algorithm to search for the optimal path. This includes using an "alarm-pheromone" that alerts the ants to infeasible areas, which saves invalid search efforts and, thus, improves the search efficiency. While our project did not consider these methods, they are still worth exploring to gain a different perspective on the problem.

4 Implementation

The related code to this project is publicly available and can be found in this github repository, and it can be easily set up and ran by following the instructions in the README file.

4.1 Environment

To help build our environment, we looked at various ocean datasets from a variety of sources like Regional Ocean Modelling System (ROMS), Ocean Reanalysis System 5 (ORAS5), and the National Oceanic and Atmospheric Administration's (NOAA) database. However, the one we was most easily manipulable came from the Simplified Ocean Data Assmilation (SODA) dataset. This dataset is a reanalysis of measured data from various periods throughout the past century and outputs time-averaged data for a variety of ocean properties. In our testing, we used the 2017 monthly data for ocean current velocity at 50 different depths for each $\frac{1}{2}^\circ$ longitude and latitude, giving us a dataset of size $(50, 330, 720)$ that started at point $(74.75\text{S}, 0.25\text{E})$. To give a rough estimate of the size of the data, 1 year's worth is about 2GB.

A dataloader class was defined with methods allowing to query the dataset directly, check whether a point exists within the dataset's bounds, get interpolated values at any given point within bounds, generate a weighted directed graph from the dataset, and draw 2D and 3D plots of currents and points of interest on a cylindrical world map projection.

4.2 Model

Unlike larger ocean gliders with wings and hydrodynamic control surfaces, the probe is assumed to exhibit symmetric drag and have negligible lift, with the only control force being an adjustable buoyancy. The ocean's density was assumed to be constant, so the probe's limited range of buoyancy adjustment corresponds directly to a limited range of vertical forces. Additionally, since the timescales in the planning problem are large, acceleration is neglected, and it is assumed that the probe can accelerate to its terminal velocity instantaneously.

Since the vertical current velocities in the ocean are minimal, they can be neglected, yielding the following continuous motion model:

$$\boldsymbol{v} = \begin{bmatrix} u \\ v \\ \pm \text{V\_TERM} \end{bmatrix} \tag{1}$$

where u and v are the horizontal components of the ocean current, and V_TERM is the terminal velocity. For continuous space planning, only synthetic flow fields were used, so V_TERM = 1 was used, in arbitrary units. Further simplifications to the model were made in order to formulate the problem as a graph problem, described in the section on the Graph Planner.

4.3 Random Exploration Planner

Continuous-space planning is advantageous because the resulting paths are more similar to feasible real-world trajectories, and the influence of the underlying grid or graph structure used for planning is eliminated. A classic method that follows this paradigm is RRT/RRT*, in which points are randomly sampled in space, and connected by a feasible path to the growing tree if possible. RRT* additionally involves some graph rewiring to ensure the algorithm converges to an optimal path.

Algorithm 1 Random Exploration

initialize $G(V, E)$ = start_node
repeat
  $G(V, E) \leftarrow$ add.random_node
  $w \leftarrow$ rand$(Up, Down)$
  $x, y, z \leftarrow$ PropogateFlowFieldDynamics(x, y, z, u, v, w)
  Collision_Check(x, y, z)
until *goal_reached*

However, unlike typical path planning problems, inverse propagation is extremely difficult. For instance, in standard RRT/RRT* you can simply connect a new sample point to a nearby node with a straight line segment, which is guaranteed to approximate a feasible trajectory as long as you do basic collision checking. However, in this case [Algorithm 1], the robot can only control its motion in the vertical direction, so unless the sampled point happens to lie directly downstream of an existing node, it is impossible to add. This event has probability zero.

In order to ensure the feasibility of added nodes, we tried starting with an existing node, picking a random actuation command, and then propagating it forward according to the flow velocity [Algorithm 2]. The resulting node was therefore guaranteed to be reachable from the start, since it was the result of a sequence of valid commands.

4.4 Graph Planner

While continuous-space planners are generally more desired, this particular problem can be easily structured as a graph traversal problem, with individual data points acting as nodes, and the veloc-

Algorithm 2 Improved Random Exploration

initialize $G(V, E)$ = start_node
repeat
  if rand(0,1) $\leq \kappa$ **then**
    $G(V, E)$.AddRandom_node
  else
    $G(V, E)$.AddLeaf_node
  end if
  $w \leftarrow$ rand$(Up, Down)$
  $x, y, z \leftarrow$ PropogateFlowFieldDynamics(x, y, z, u, v, w)
  Collision_Check(x, y, z)
until *goal_reached*

ities at those nodes acting as directed edges to neighbouring nodes, weighted by the inverse of their magnitude.

4.4.1 Generating the Graph

To find all possible routes our robot can take, we first consider the current map's 50 depth layers and decompose the area of each of these layers into cells with the uniform horizontal direction of flow, similar to [8].

Ideally, for each layer, we then create a directed graph between these cells. Afterwards, we connect the cells in each layer to the cells in the layers immediately above and below it. We are left with a 3D graph of all routes we can take.

However, given the large dataset, simplifications were made and the graph was generated using Algorithm 3. Explicit modelling of depth was eliminated, so we assumed that the robot's maximum vertical velocity is fast relative to ocean current. This seems reasonable enough considering that most ocean currents are much less than 1 m/s. We quantized the currents in ever depth column by binning them into eight directions so that they correspond directly with the neighboring nodes on the grid. The eight bins are updated with the lowest costs (highest velocities) across the 50 depths.

In this fashion, we avoided creating nearly 12 million nodes with at least double the amount of edges, and created under 240,000 nodes with up to (but usually much less than) eight times as many edges.

Algorithm 3 Generate Graph

Input: DATASET
Graph(Lats,Lons,8) $\leftarrow inf$
for $Lat \in$ DATASET.Latitudes **do**
  for $Lon \in$ DATASET.Longitudes **do**
    for $z \in$ DATASET.Depths **do**
      $u, v \leftarrow$ DATASET.Current(Lat, Lon, z)
      $angle \leftarrow$ arctan2(u, v)
      $direction \leftarrow$ get_octant$(angle)$
      $speed \leftarrow$ norm(u, v)
      $cost \leftarrow 1/speed$
      if $cost \leq$ Graph$(Lats, Lons, direction)$ **then**
        Graph$(Lats, Lons, direction) = cost$
      end if
    end for
  end for
end for

4.4.2 Modified Dijkstra's Algorithm

The graph planner used in our project is based on Dijkstra's algorithm for finding the optimal path. The algorithm allows exploring all the neighboring nodes while maintaining the search for an optimal path. As shown in Algorithm 4, nodes are dynamically initialized and the algorithm will terminate early if no new nodes are reachable. The cost used to maintain the optimal path is established in Algorithm 3 as being inversely proportional to the magnitude of the ocean current speeds at each depth.

Algorithm 4 Modified Dijkstra's Algorithm

```
Input: START_NODE, GOAL_NODE
VISITED_NODES ← {START_NODE}
REACHABLE_NODES ← find_leaves(START_NODE)
repeat
    BEST_NODE ← NONE
    for all node ∈ REACHABLE_NODES do
        BEST_NODE ← update_best_node(node)          ▷ Lowest Cost of unvisited nodes
    end for
    delete BEST_NODE from REACHABLE_NODES
    if BEST_NODE ∉ VISITED_NODES then
        REACHABLE_NODES.add(find_leaves(BEST_NODE))
    end if
until GOAL_NODE ∈ VISITED_NODES or REACHABLE_NODES = {∅}
return get_path(GOAL_NODE)
```

5 Case Studies

5.1 Experiment 1: Random Exploration

Our first experiment applied random exploration to a simplified version of the ocean model. The ocean current in the x axis was proportional to the vertical depth, in order to create a toy problem that can easily be shown to be fully reachable (at least in the x direction) with only vertical control. The results of this experiment are shown in figure 3.

Unfortunately, this performance of this method in practice was very poor. Standard RRT is rapidly exploring precisely because it does the sampling in physical space as opposed to command space, and so prospective new nodes are drawn relatively equally from the entire space being explored. In contrast, this method samples only nodes reachable in a short distance from an existing node, which results in the sampling being heavily biased towards the already explored area. The result was a tree that sampled space heavily biased in favour of existing nodes - exactly the opposite of what you want in an exploration algorithm. The branching factor was extremely high, and as a result the length of the longest chains grew very slowly as the number of nodes was increased. And finally, because inputs are selected at random, the algorithm did a very poor job of exploring areas that would require a very specific sequence of inputs to reach, like the upper left quadrant. The resulting tree can be seen on the left of the figure.

In order to improve exploration efficiency, the node selection algorithm was modified: now, most of the time (with probability $1 - \kappa$ - see Algorithm 2), a random leaf node will be selected. This has the effect of vastly reducing the branching factor and increasing the length of individual chains. Comparing the left to the right side of figure 3, it is apparent that IRE explores much farther than RE given the same number of nodes sampled.

Unfortunately, this fails to resolve some of the other issues with random exploration. Areas in the bottom right and top left quadrants are still very poorly explored because they require a highly unlikely sequence of inputs to reach. Furthermore, exploration in the vertical direction is very slow because each individual chain is essentially following a random walk, which grows as $\sim \sqrt{n}$ rather than $\sim n$.

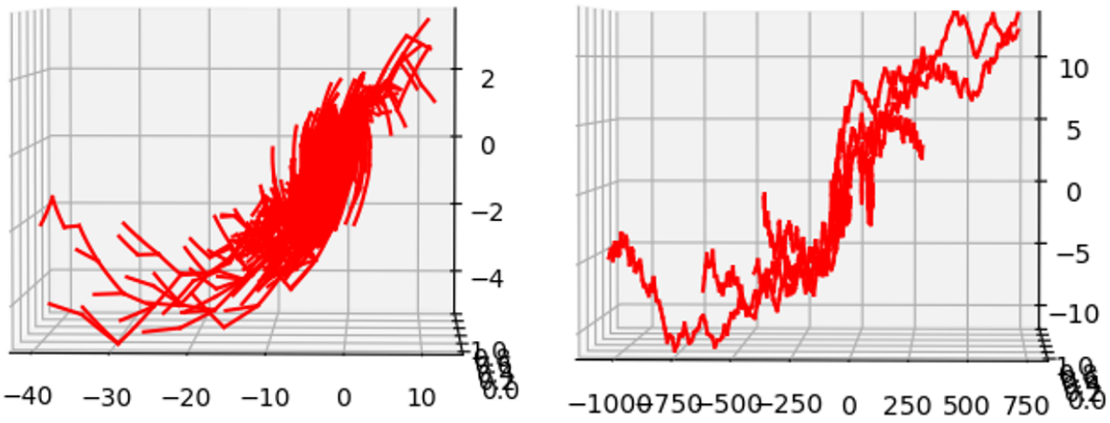

Figure 3: Left: Random Exploration — Right: Improved Random Exploration

Due to these issues, we determined that this form of random exploration was likely unsuitable for ocean exploration and reachability analysis, and instead focussed our efforts on graph based methods working with the full ocean current dataset, as detailed in the following sections.

5.2 Experiment 2: Graph Planning — Start to Goal

Our second experiment aimed to find a path using a modified version of Dijkstra's algorithm on the generated graph. Figure 4 and 5 demonstrate the paths found given a start off the coast of Miami, FL, and a goal off the coast of St. John's, NL, and vice-versa. The explored nodes are colour-coded according to their cost, with blue being low cost and yellow being high cost. The optimal path returned by Dijkstra's is marked in purple, and the endpoints in red.

In the first path (Figure 4), the trajectory stays on the surface, following the warm currents of the Gulf Stream to reach its destination with a cost of about 257. These endpoints were deliberately chosen to make them easy to get to by following a major ocean current. Of course, if the forward journey is with the current, the return will be against it (Figure 5). In order to backtrack, the probe must descend to the deeper waters where the cold Gulf Stream current flows the other way. Unfortunately, this deep current does not extend into the shallow waters off Miami, so instead the probe must take a detour through the Caribbean, surfacing between Cuba and Haiti, rejoining the warm Gulf Stream current and taking that around Cuba up to Miami. Deep ocean currents are comparatively slower and weaker than surface currents (ocean currents are driven primarily by wind), and the route length is also longer, so the cost of the return journey is 4x the cost, at 1048.

5.3 Experiment 3: Graph Planning — Coverage

To help evaluate the practicality of this mode of ocean exploration, we conducted a reachabilty analysis of the global ocean by running Dijkstra's algorithm without a defined goal. The result is that it simply spread-filled the entire ocean starting from an arbitrary start point (Miami) according to the directed edges from the graph's construction. Any node that is explored by this algorithm is reachable by a bouyancy-controlled probe released from Miami. The results are shown in Figure 6.

The connectivity graph is (mostly) complete and bidirectional, implying that it is possible to go from anywhere in the global ocean to anywhere else simply by following the currents at different depths. There are a few notable exceptions, which appear black on the map: the Hudson Bay, the Baltic sea, the Mediterranean and Black Seas, and the Persian Gulf. Although connected to the ocean, these bodies of water are comparatively isolated, and connected only via a relatively narrow strait whose width is in many cases comparable to or smaller than the grid resolution. It is likely that these areas will prove reachable with a finer grid resolution, or at different times in the year.

Regardless, this experiment demonstrates the interconnectedness of the global ocean, and the practicality of reaching arbitrary points in the ocean by launching limited-actuation probes from a single convenient location.

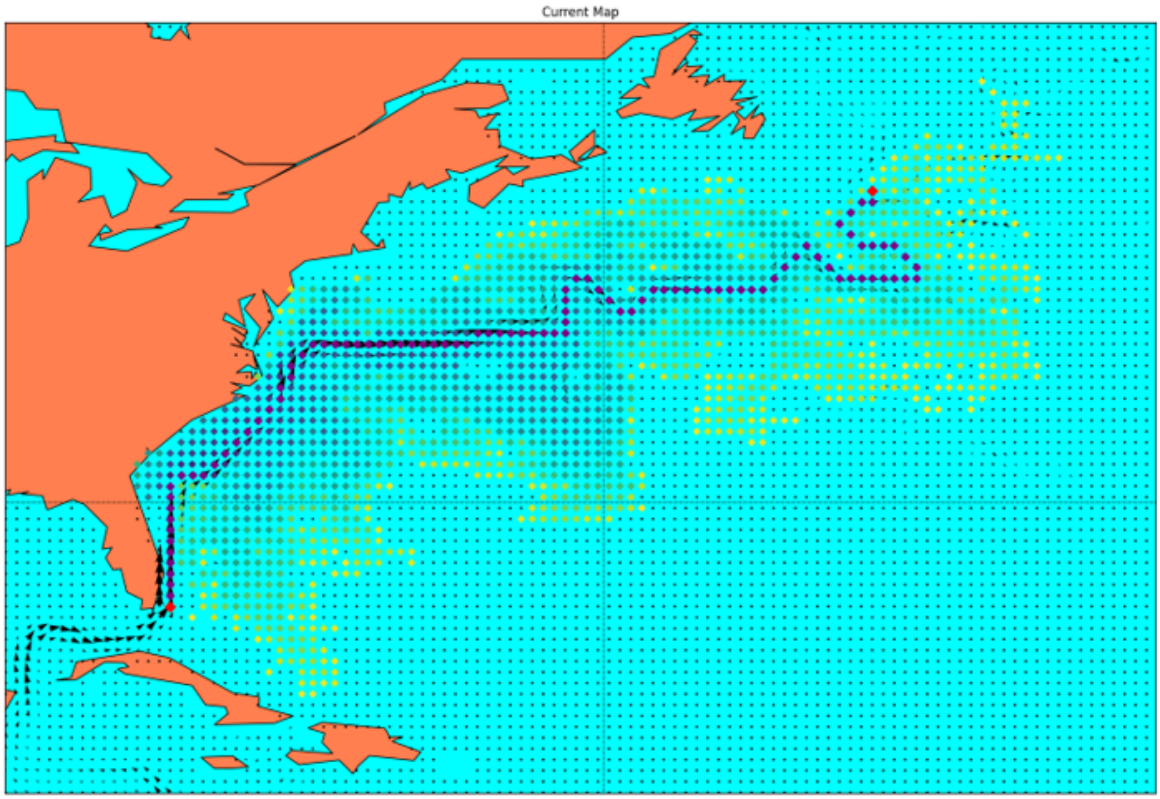

Figure 4: Path determined using modified Dijkstra's algorithm from Miami to St. John's along the warm Gulf Stream current. Cost: 256.98

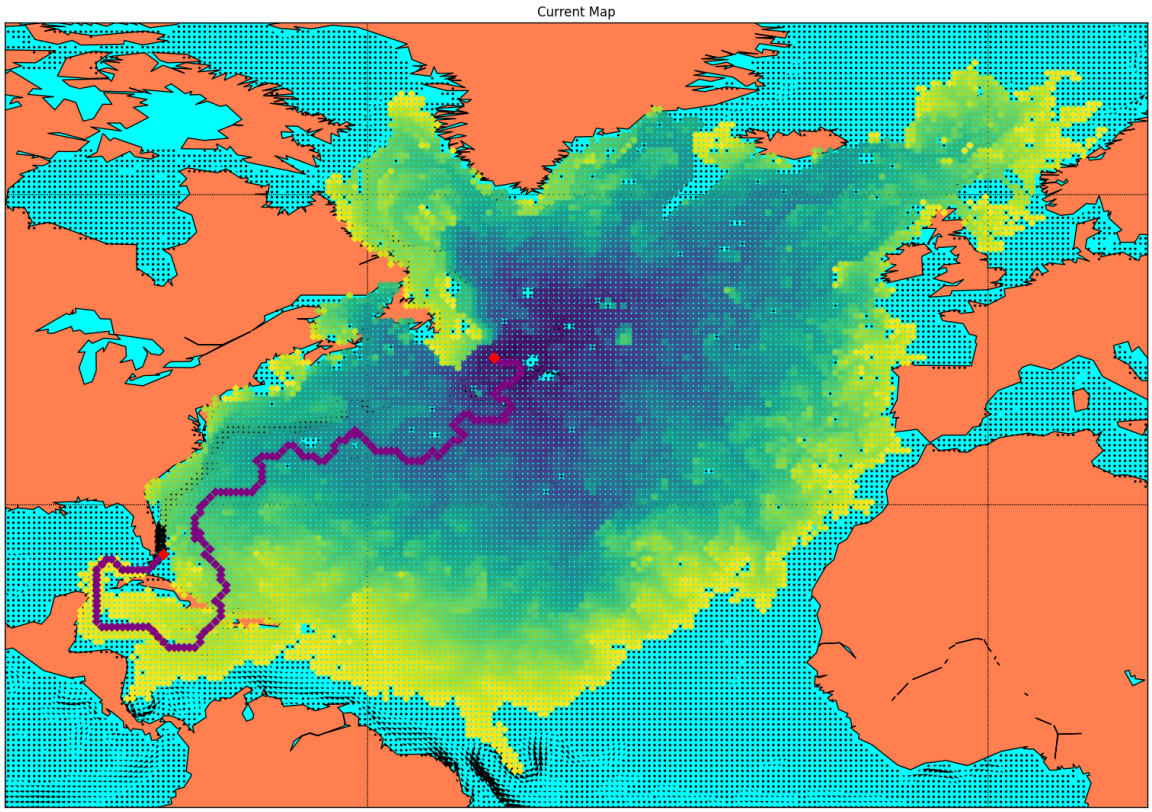

Figure 5: Path determined using modified Dijkstra's algorithm from St. John's to Miami along the cold Gulf Stream current, rejoining with the warm current in the Gulf of Mexico. Cost: 1048.13

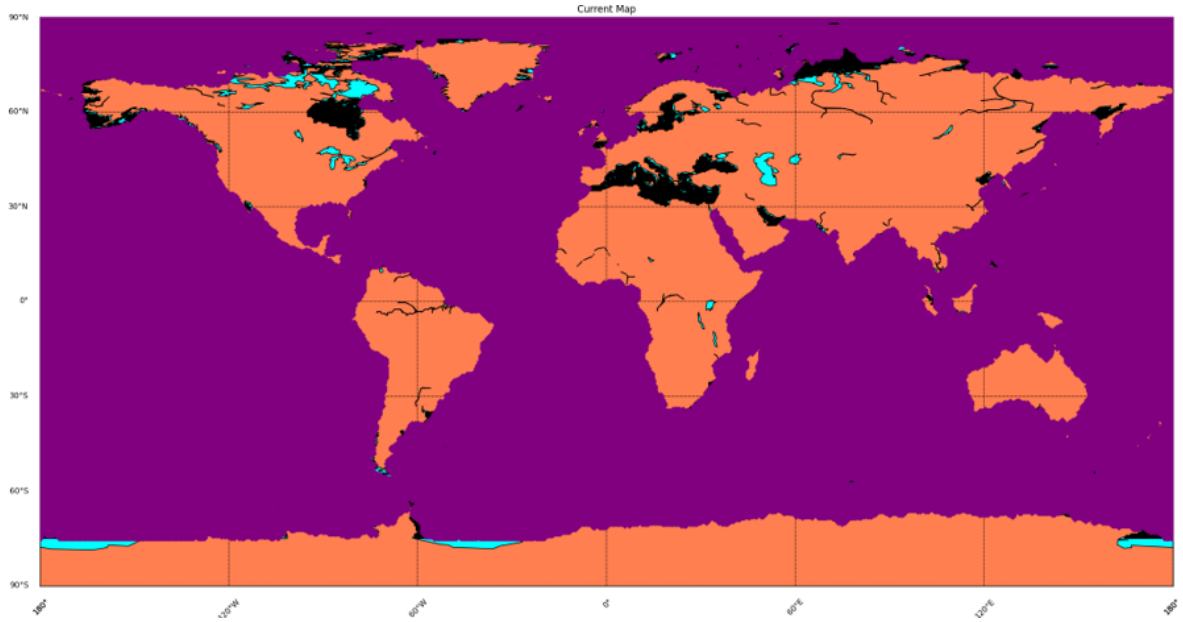

Figure 6: Possible node coverage using graph planning starting from Miami. Turquoise: No data; Purple: Reachable; Black: Unreachable

6 Conclusion

Overall, this report presents our attempt to evaluate the feasibility of using a low cost buoyancy probe with two types of algorithms: Random Exploration and Graph Planning.

One observation is that our chosen problem formulation was particularly challenging. Most works exploring flow fields with limited actuation either chose zero actuation (eliminating planning entirely), or made their limited actuation omnidirectional - sometimes directly actuated, and sometimes using gliders with hydrodynamic surfaces. In the latter case, inverse planning is possible, as the robot possesses the ability to move perpendicular to the flow field, albeit in a limited capacity. This makes a wide variety of conventional planning techniques feasible.

In contrast, our problem statement has zero ability to travel perpendicular to the flow in the horizontal plane, which renders inverse path planning and standard methods of spatial sampling impossible. This is the primary reason why our attempts at random exploration performed so poorly and we instead focused on direct graph planning.

The graph planning results were much more promising. By eliminating a direct representation of depth, we were able to increase the computational efficiency of exploration and apply deterministic and complete algorithms without sacrificing any essential aspects of the system dynamics. We were able to demonstrate both point-to-point planning as well as conduct a global reachability analysis.

The results of the graph planner are highly promising. The reachability analysis concluded that virtually all of the world's ocean is reachable from an arbitrary starting point, and most of the unreachable areas were only unreachable due to the limited resolution of the grid and dataset. This suggests that even without hydrodynamic surfaces, bouyancy-controlled probes can travel to any desired point in the ocean, thus enabling cheaper and larger-scale ocean exploration.

In addition to demonstrating the practicality of this approach, we also demonstrated a point-to-point path planner capable of quickly and efficiently planning out routes between arbitrary points on the ocean.

Overall, a surprisingly high degree of control was achieved by robots with very limited actuation ability, and suitable planning techniques were explored and demonstrated.

6.1 Future Work

Various improvements and further advancements can be made to our implementation:

- Plotting the path in 3D and keeping track of the progression of the depths travelled between each node along the path
- Trying Dijkstra's algorithm on the original 3D grid and factoring the robot's actual vertical travel dynamics to generate more realistic paths
- Factoring in time into the equation and working with multiple datasets as the robot explores through many weeks, months, or even years
- Extensively evaluating a wide combination of start and goal pairs throughout the oceans and coast
- Inclusion of static and dynamic obstacles
- Improved cost metric taking into account the true distance between nodes (while dataset is uniform on Mercator projection, the Mercator distances are not Euclidean distances; Euclidean distance should be used with velocity to make cost equal to time)
- Finer grid size and higher resolution datasets for improved reachability analysis

References

[1] "How much of the ocean has been explored?" 2021. [Online]. Available: https://oceanexplorer.noaa.gov/facts/explored.html

[2] "Multibeam sonar," 2021. [Online]. Available: https://oceanexplorer.noaa.gov/technology/sonar/multibeam.html

[3] NASA-JPL, "BRUIE - Buoyant Rover for Under-Ice Exploration," *American Geophysical Union, Fall Meeting 2012, abstract id. C13E-0655*, 2012. [Online]. Available: https://www.jpl.nasa.gov/robotics-at-jpl/bruie

[4] G. L. et al., "Self-powered soft robot in the Mariana Trench," *Nature (London) vol. 591 no. 7848*, pp. 66–71, 2021.

[5] C.-F. X. Y. G. Y.-N. Ma, Y.-J. Gong and J. Zhang, "Path Planning for Autonomous Underwater Vehicles: An Ant Colony Algorithm Incorporating Alarm Pheromone," *IEEE transactions on vehicular technology, vol. 68, no. 1,*, p. 141–154, 2019.

[6] J. D. H. et al., "Autonomous underwater navigation and optical mapping in unknown natural environments," *Sensors (Basel, Switzerland), vol. 16, no. 8*, p. 1174–1174, 2016.

[7] K. Y. C. To, C. Yoo, S. Anstee, and R. Fitch, "Streamline-Based Flow Field Planning."

[8] M. Hou, S. Cho, H. Zhou, C. R. Edwards, and F. Zhang, "Bounded Cost Path Planning for Underwater Vehicles Assisted by a Time-Invariant Partitioned Flow Field Model," *Frontiers in Robotics and AI*, vol. 8, no. July, pp. 1–17, 2021.

[9] J. J. Heon Lee, C. Yoo, S. Anstee, and R. Fitch, "Hierarchical Planning in Time-Dependent Flow Fields for Marine Robots," *Proceedings - IEEE International Conference on Robotics and Automation*, no. 885, pp. 885–891, 2020.

[10] T. Lolla, M. P. Ueckermann, K. Yiğit, P. J. Haley, and P. F. Lermusiaux, "Path planning in time dependent flow fields using level set methods," *Proceedings - IEEE International Conference on Robotics and Automation*, no. May, pp. 166–173, 2012.

[11] T. Wang, O. P. Le Maître, I. Hoteit, and O. M. Knio, "Path planning in uncertain flow fields using ensemble method," *Ocean Dynamics*, vol. 66, no. 10, pp. 1231–1251, 2016.

[12] K. Krishna, Z. Song, and S. L. Brunton, "Finite-Horizon, Energy-Optimal Trajectories in Unsteady Flows," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 478, no. 2258, feb 2021. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rspa.2021.0255http://arxiv.org/abs/2103.10556

[13] J. Hansen and G. Dudek, "Coverage Optimization with Non-Actuated, Floating Mobile Sensors using Iterative Trajectory Planning in Marine Flow Fields," *IEEE International Conference on Intelligent Robots and Systems*, pp. 1906–1912, 2018.

[14] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Time and energy optimal path planning in general flows," *Robotics: Science and Systems*, vol. 12, 2016.

[15] P. Gunnarson, I. Mandralis, G. Novati, P. Koumoutsakos, and J. O. Dabiri, "Learning efficient navigation in vortical flow fields," *Nature Communications*, vol. 12, no. 1, pp. 1–7, 2021.