An algorithm for geometric navigation planning under uncertainty using terrain boundary detection

Bennett A. Carley Adeolayemi M. Bamgbelu XiMing Zhang Jason M. O'Kane

***Abstract*— We explore a navigation planning problem under uncertainty for a simple robot with extremely limited sensing. Our robot can turn subject to significant proportional error and move forward. As it moves in an environment with a known terrain map, the robot can detect changes in the terrain at its current position. Given an initial pose and a goal segment, the robot should find some sequence of actions to travel reliably from start to goal, if such a sequence exists. The resulting plan should guarantee the robot reaches the goal segment despite any movement errors experienced within some known error bound. In this paper, we propose an algorithm to find such an action sequence, implement and evaluate this algorithm, and present evidence for the feasibility of such an algorithm in an underwater navigation setting.**

I. Introduction

The problem of underwater navigation planning, where an autonomous underwater vehicle (AUV) aims to reach a goal state from an initial state despite significant uncertainty, remains an important challenge [16], [42]. Enhanced underwater navigation would facilitate diverse applications, including diver guidance [20] and coral reef inspection [19], among others. Conventional methods typically focus on minimizing localization error to reach the goal [8], [24].

However, underwater navigation presents numerous challenges. Many existing underwater path planning algorithms require intermittent surfacing for GPS fixes, resulting in time and energy inefficiencies [7]. Additionally, underwater sensors often exhibit substantial inaccuracies [13]. Camera image quality is frequently degraded by particulate matter in underwater environments, limiting visibility and overall performance [41]. A more comprehensive review of related work is provided in Section II.

This paper addresses a specific form of navigating under uncertainty, where the robot can detect terrain changes within the environment. For instance, indoor robots can observe transitions between carpet and hardwood, while marine robots can observe transitions between sand and reef. We model robots that experience significant rotational error and cannot accurately measure translational movement. The terrain transitions robots can observe are modeled as segments. A robot action is defined as a rotation subject to error, followed by forward movement until a segment is detected. Using this robot model, we aim to navigate from an initial pose to a goal segment. The formal problem definition is detailed in Section III.

B. A. Carley, A. M. Bamgbelu, X. Zhang, and J. M. O'Kane are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. {bcarl, sk3ps, ximingzhang.sam, jokane}@tamu.edu

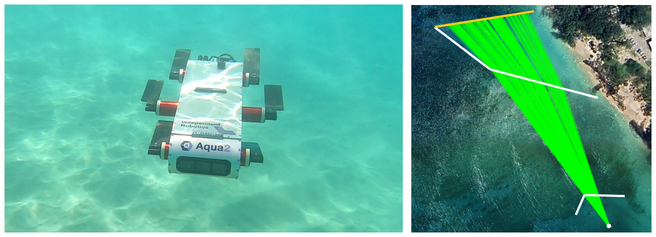
Fig. 1: [left] The Aqua2 underwater hexapod robot platform can navigate in shallow coastal waters, but its movement is generally subject to significant error, and localization without additional sensors remains a challenge. [right] A plan, generated by the proposed algorithm and realizable by Aqua2, that uses coral reefs as segment-landmarks to navigate from an initial pose to a goal segment.

Our algorithm reasons about information states (I-states), which represent sets of possible states resulting from action execution. Each I-state is defined by a line segment representing possible positions and an interval representing possible headings. First, we introduce a geometric test to determine which transition actions, if any, reliably transition between an I-state and a segment. Next, we detail a method for generating a discrete set of valid actions at a specified resolution step from an I-state, ensuring this action set becomes increasingly dense as resolution increases. Using this method with a resolution step of zero, we perform a depth-first search starting at the robot's initial pose and search for a sequence of actions to the goal segment. Finally, if no path to the goal segment is found, we increase the sampling resolution step by one and repeat this process if the current resolution step is less than the maximum resolution step. We elaborate this process in Section IV. Section V presents an implementation and evaluation of our approach.

Figure 1 illustrates an example output, designed for execution by the Aqua2 AUV, leveraging the transitions between reef and sand as distinct segments. Section VI demonstrates the Aqua2's vision system can indeed reliably detect these segments.

The results discussed in Section VII provide a roadmap for robust underwater navigation despite significant movement inaccuracies and limited sensor information.

II. Related Work

A. Planning Under Uncertainty

This paper addresses a planning problem where the robot experiences uncertainty about its current pose. The objective of forming plans that succeed in spite of uncertainty is one of the most fruitful topics in the robotics literature, approached in a wide variety of ways [1], [2], [9], [10], [21], [22], [29], [39], [40]. Some of the most closely-related work stems from problems considered by Erdmann and Mason, investigated scenarios where a robot, initially unaware of its pose, achieves a know pose after some sequence of actions [4]. More recent follow-up work demonstrated that, in certain environments, a robot's pose converges to a deterministic state given a sufficiently long action sequence [18].

Another closely related early thread of research considered navigating environments with landmarks, where the robot has perfect sensing within these landmarks and no sensing outside them [11].

For robots equipped with an environmental map, unreliable compass, and bump sensor, successful planning from start to goal [15] and guaranteed coverage of the space [14] have been demonstrated. Localization problems involving robots with only a clock and bump sensor have also been explored [23], and probabilistic methods can successfully localize such a robot [5].

B. *Underwater Navigation*

Underwater navigation has been an active research area for many years, resulting in a substantial body of literature [3], [27], [32], [33], [37], [38]. Path deviations caused by currents present a significant challenge in underwater environments, complicating navigation. Various methods have been proposed to estimate these deviations using sparse GPS data [12] and terrain structure observations [34]. The presence of obstacles further complicates underwater navigation. Research has explored 3D path optimization in these environments, both with and without known maps [36]. This work builds upon Trajopt, an optimization framework for generating robot trajectories [28]. Vidal et al. developed an online motion planning framework that accounts for the complex hydrodynamics of underwater environments and efficiently generate plans [31]. The SVIn2 framework implements a Simultaneous Localization and Mapping (SLAM) for underwater exploration, fusing sonar, visual, and inertial sensor data [24].

There has also been interest in using camera data to navigate in underwater environments. For instance, AquaVis employs one or more cameras to navigate unknown environments and avoid obstacles by tracking landmarks and dynamically adjusting the robot's movement to maintain their visibility [35]. Additionally, an onboard GPU can be leveraged by underwater robots to perform vision-based navigation while detecting and classifying objects encountered by the robot [17].

The quality of underwater images is often poor and presents a significant challenge. Fortunately, recent work has explored machine learning techniques to improve these images [6], along with established model-based methods [25]. A comprehensive review of various underwater image enhancement methods can be found in Zhou, Yang, and Zhang [41].

III. Problem Statement

In this section, we outline the problem approached in this work.

A. *Robot Model*

We consider a point robot navigating a planar environment $\mathcal{W} = \mathbb{R}^2$. The robot's configuration space is $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$, where $\mathbb{R}^2$ represents the robot's current position p, and $\mathbb{S}^1$ represents its heading θ. A state $x \in \mathcal{C}$ is defined by the tuple (p, θ).

The environment contains a set of q non-intersecting line segments S. Each segment $s \in S$, defined by endpoints $(a, b) \in \mathcal{W}^2$, is the set of points in the closed line segment between a and b. The robot detects when it crosses a segment, but not its identity, and can transverse these segments unimpeded.

We model time as discrete stages $k \in \mathbb{N}$. The robot's state at stage k is denoted as $x_k \in \mathcal{C}$. The action space is $U = [-\pi, \pi)$, representing the robot's commanded rotation. Uncertainty is modeled by an adversarial nature, which selects actions from $\Xi = [-\alpha, \alpha]$ where $\alpha \in [0, 1)$ bounds the rotational error. At each time stage k, the robot selects action u_k and nature selects action $\xi_k \in \Xi$. That is, if the robot's commanded rotation is u_k, the actual amount of rotation may be any amount in the interval $[u_k(1 - \alpha), u_k(1 + \alpha)]$.

To define the state transition function, we introduce necessary geometric machinery. Let $R(p, \theta)$ denote a ray originating at p with heading θ. The function $H_S(p, \theta)$ returns the set of all intersection points between segments in S and the ray $R(p, \theta)$, defined thusly:

$$H_S(p, \theta) = \bigcup_{s \in S} \{R(p, \theta) \cap s\} \tag{1}$$

To determine the closest intersection point to the ray's origin p, we define $h_S(p, \theta)$ as the nearest point to p on any segment $s \in S$ along the ray $R(p, \theta)$. This can be expressed as:

$$h_S(p, \theta) = \operatorname*{arg\,min}_{q \in H_S(p, \theta)} \|p - q\| \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Given the robot's state x_k, the subsequent state x_{k+1} is determined by a rotation of $u_k(1 + \xi_k)$ radians followed by forward motion until a segment $s \in S$ is detected. If no segment is detected, x_{k+1} is undefined. We define the state transition function f as:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, \xi_k) \\ &= (h_S(x_k, \theta_{x_k} + u_k(1 + \xi_k)), \theta_{x_k} + u_k(1 + \xi_k)) \end{aligned} \tag{3}$$

where p_{x_k} and θ_{x_k} represent the position and heading components of x_k, respectively. The term ξ_k represents a multiplicative rotational error, which is uncontrolled and unknown to the robot.

We further define f^m as the state transition function for a sequence of m actions:

$$\begin{aligned} x_{k+m} &= f^m(x_k, u_k, \ldots, u_{k+m-1}, \xi_k, \ldots, \xi_{k+m-1}) \\ &= f(f(\cdots f(x_k, u_k, \xi_k) \cdots), u_{k+m-1}, \xi_{k+m-1}) \end{aligned} \tag{4}$$

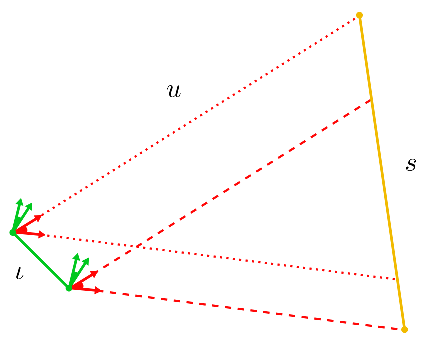

Fig. 2: Illustration of a transition from a source I-state ι (green) to a target segment s (gold). The dotted red lines represent the angle interval after performing action u on ι.

B. Planning Objective

Given this robot model, we desire plans reliably navigating the robot through $\mathcal{W}$. Specifically, given an initial state $x_I \in \mathcal{C}$, a goal segment $s_G \in S$, a set of segments S, and nature scalar α, we wish to find a sequence of actions $(u_1, \ldots, u_K)$ such that, for any nature action sequence $(\xi_1, \ldots, \xi_K)$, the following condition holds:

$$f^{K+1}(x_I, u_1, \ldots, u_K, \xi_1, \ldots, \xi_K) \in s_G \times \mathbb{S}^1 \tag{5}$$

We will henceforth refer to $s_G \times \mathbb{S}^1$ as the goal set X_G.

C. Information States

The success criterion defined in Equation 5 involves universal quantification over nature actions. Such universal quantifiers may prove challenging for algorithms, particularly for continuous domain variables. To address this challenge, we recast the problem in terms of information states (I-states), which track sets of possible states over all nature action selections. For a given action sequence $(u_1, \ldots, u_{k-1})$, we define the information state $\iota_k \in \mathcal{C}$ as the set of states the robot might have reached at stage k. This concept is illustrated in Figure 2. Starting from $\iota_1 = \{x_I\}$, these I-states are computed using the information transition function:

$$\begin{aligned} \iota_{k+1} &= F(\iota_k, u_k) \\ &= \bigcup_{x_k \in \iota_k} \bigcup_{\xi \in \Xi} \{f(x_k, u_k, \xi)\} \end{aligned} \tag{6}$$

Furthermore, we define F^m as the I-state transition function for a sequence of m actions from a given I-state:

$$\begin{aligned} \iota_{k+m} &= F^m(\iota_k, u_k, \ldots, u_{k+m-1}) \\ &= F(F(\cdots F(\iota_k, u_k) \cdots), u_{k+m-1}) \end{aligned} \tag{7}$$

The I-states reachable with these transitions can be represented as tuple $\iota = (s, [\theta, \theta'])$. Segment s denotes the set of possible robot positions, and $[\theta, \theta']$ represents an angular interval with $\theta \in [0, 2\pi)$ and $\theta' \in [\theta, 2\pi + \theta]$. The robot's state is guaranteed to lie within the set $s \times [\theta, \theta']$.

Then, using the notion of I-states, we recast the planning problem as finding some sequence of actions $(u_1, \ldots, u_K)$ for which:

$$F^K(x_I, u_1, \ldots, u_{K-1}) \subseteq X_G \tag{8}$$

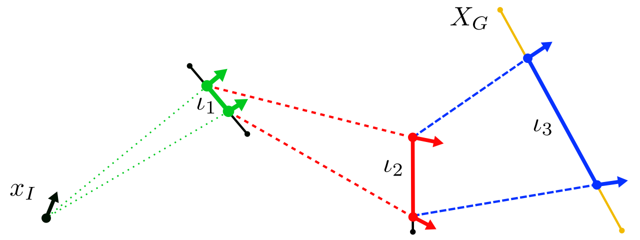

Fig. 3: An illustration of a valid action sequence from x_I to X_G. The dotted segments represent the robot's possible headings after turning.

Algorithm 1: TRANSITIONACTIONS(ι, s)

```
   // Use Equation 6 to find the set of
      all actions arriving exclusively
      at $\iota_g$ from $\iota_a$
1  return $\{u \in U \mid F(\iota, u) \subseteq s \times \mathbb{S}^1\}$
```

In other words, we wish to find some sequence of K actions such that the final I-state ι_K is entirely contained within the goal set X_G.

IV. Algorithm Description

This section details an algorithm designed to solve the problem presented in Section III. The algorithm comprises three primary components: (1) determining valid I-state to segment transition actions (Section IV-A), (2) generating a discrete subset of valid actions at an I-state for a specified resolution (Section IV-B), and (3) constructing and searching an I-state tree to produce a plan (Section IV-C).

A. I-State Transition

A fundamental component of the algorithm is the identification of transition actions between I-state $\iota = s_\iota \times [\theta, \theta']$ and segment s, if such actions exist. Algorithm 1 details this process, employing Equation 6 to determine the set of valid transition actions.

We initially determine the set of error-free rotations that orient the robot towards the target segment. Specifically, given the angle interval $[\theta_a, \theta'_a]$, Algorithm 1 calculates the set of rotations that orient the robot towards segment s using trigonometric principles. The set U_h is then computed as the actions guaranteed to achieve a rotation in the above set.

Following this, we check for occlusions between segments s_ι and s by all segments $s' \in S \setminus \{s_\iota, s\}$. For each occluding segment, the set of actions that *could* reach s' from ι is determined and accumulated into set U_o. Finally, we compute and return the set of valid transition actions $U_v = U_h \setminus U_o$. A valid transition action $u \in U_v \subseteq U$ from ι to s, as determined by Algorithm 1, is illustrated in Figure 2.

B. Sample Valid Actions

In this section, we discuss the second algorithmic component: the generation of sample actions from a given I-state in a increasingly dense manner. This process enables the construction of an increasingly dense tree, as described in Section IV-C.

Algorithm 2: SAMPLEVALID(ι, S, r)

```
   // Generate set of valid actions $U_r$
      from I-state $\iota$ to each segment $s$
      in $S$ for resolution step $r$
1  if $r = 0$ then
2      $Z_r \leftarrow \{0, 1\}$
3  else
4      $Z_r \leftarrow \{\frac{1}{2^r}, \frac{3}{2^r}, \frac{5}{2^r} \ldots, \frac{2^r-5}{2^r}, \frac{2^r-3}{2^r}, \frac{2^r-1}{2^r}\}$
5  end
6  $U_r \leftarrow \emptyset$
   // Generate the set of all valid
      actions for given $r$
7  for $s \in S$ do
8      $U_t \leftarrow$ TRANSITIONACTIONS($\iota, s$)
9      $U_F \leftarrow$ Set of maximal connected subsets of $U_t$
10     for $U_f \in U_F$ do
11         $U_r \leftarrow U_r \cup \{(\max U_f - \min U_f)z \mid z \in Z_r\}$
12     end
13 end
14 return $U_r$
```

Initially, we define a generator set Z_r for a given resolution step $r \in \mathbb{N}$, where $Z_0 = \{0, 1\}$ and $Z_r = \{\frac{1}{2^r}, \frac{3}{2^r}, \ldots, \frac{2^r-3}{2^r}, \frac{2^r-1}{2^r}\}$ for $r > 0$. Notably, as r increases, the union $\bigcup_{r'=0}^{r} Z_{r'}$ becomes increasingly dense within the interval $[0, 1]$, ensuring the resolution completeness of Algorithm 3. Subsequently, for each segment $s \in S$, we determine the set of actions U_t that facilitate transitions from the given I-state ι_i to s. Next, U_t is partitioned into the minimal set of disjoint, connected subsets U_F. This partitioning allows for the increasingly dense sampling of each subset $U_f \in U_F$.

C. Tree Construction and Search

The final algorithmic component constructs and searches an I-state tree using a sequence of breadth-first searches. Each search samples the space of valid actions with progressively finer resolution, a technique we refer to as 'iterative sharpening'. We define the set of valid actions for a given I-state ι as $U_\iota = \bigcup_{s \in S}\{u \in U \mid F(\iota, u) \subseteq s \times \mathbb{S}^1\}$. The primary data structures employed are a queue Q, which maintains the exploration frontier, and a rooted tree T, which stores the exploration history. Both are initialized with the initial I-state $\{x_I\}$. Initially, the algorithm performs a breadth-first search using the actions generated by Algorithm 2 with resolution $r = 0$. Each breadth-first search is guaranteed to terminate in finite time because the segment set S is finite, forcing the robot to eventually turn and select a non-zero rotation u_k. For each non-zero action, the angular interval of the resulting I-state expands. Once the angular interval of an I-state reaches a size of π, no further valid actions can be executed. Upon termination of a breadth-first search iteration, an iterative sharpening step is performed. The resolution r is incremented by one, provided $r + 1 < r_{\text{max}}$; otherwise, the algorithm returns failure.

Algorithm 3: CONSTRUCTANDSEARCH(x_I, X_G, S, r_{max})

```
1  $\psi \leftarrow$ empty stack
2  if $x_I \subseteq X_g$ then
3      return $\psi$
4  end
5  $T \leftarrow$ tree rooted at $\{x_I\}$
6  $Q \leftarrow$ empty queue
7  for $r \in \{0, 1, \ldots, r_{\text{max}}\}$ do
8      $Q$.ENQUEUE($\{x_I\}$)
9      while $Q$ is not empty do
10         $\iota \leftarrow Q$.DEQUEUE()
           // Determine the actions from $\iota$
              to explore
11         if $\iota$ visited then
12             $U \leftarrow$ SAMPLEVALID($\iota, S, r$)
13         else
14             Mark $\iota$ as visited
15             $U \leftarrow \bigcup_{n=0}^{r}$ SAMPLEVALID($\iota, S, n$)
16         end
17         for $u \in U$ do
               // Use Equation 6 to find
                  I-state transition
18             $\iota' \leftarrow F(\iota, u)$
19             $T$.ADDCHILD($\iota, \iota', u$)
20             if $\iota' \subseteq X_G$ then
21                 while $T$.HASPARENT($\iota'$) do
22                     $(\iota', u) \leftarrow T$.TOPARENT($\iota'$)
23                     $\psi$.PUSH($u$)
24                 end
25                 return $\psi$
26             end
27         end
28         $Q$.ENQUEUEMANY($T$.CHILDREN($\iota$))
29     end
30 end
31 return ☹
```

To optimize the search, if a node has been visited previously, Algorithm 2 is invoked only for the current resolution r, as lower resolutions have already been sampled. Conversely, if a node is encountered for the first time, Algorithm 2 is applied for all resolutions $r' \leq r$ to ensure the union $\bigcup_{r'=0}^{r} Z_{r'}$ is considered. It is important to note that the resulting I-states are enqueued after iterating through the sampled actions. This ensures that I-states discovered at previous resolution steps are included in the breadth-first search.

V. IMPLEMENTATION AND EVALUATION

We implemented this algorithm in Python [30]. The trials reported below used an Intel Core i7-13700KF at 3.40 GHz, 32 GB of DDR5 RAM, and Windows 11. Figure 1 shows an example plan completed by this implementation.

To quantitatively evaluate our algorithm, we generated

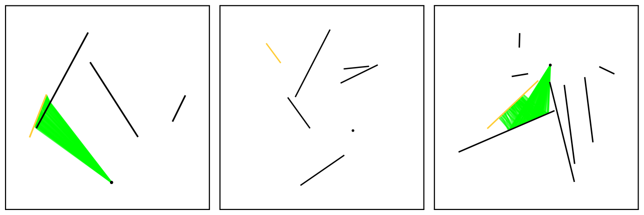

Fig. 4: Successful plans for trials where $q = 4$ (left) and $q = 8$ (right) and a trial where no plan was found for $p = 6$ (middle). For each of the three trials, $\alpha = 0.1$ and $r_{\max} = 3$. Each green line represents a plan execution. For each plan execution, nature selects a random $\xi \in \Xi$ for every robot action $u \in K$.

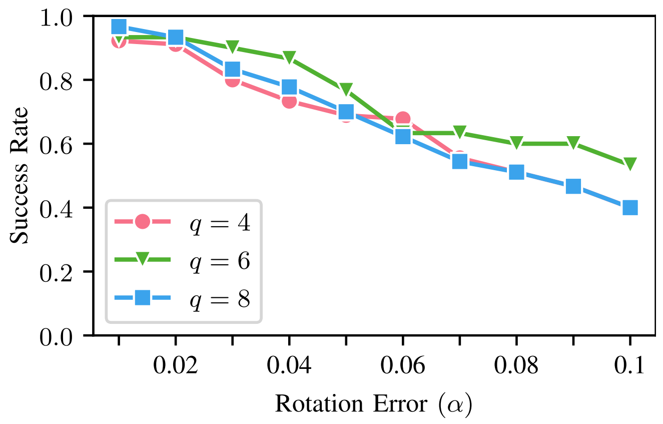

Fig. 5: The effect of α on the success rate across all maximum resolutions step values $r_{\max} \in \{0, 1, 2\}$. As α increases, the planner's success rate decreases for all $q \in \{4, 6, 8\}$.

random problem instances, each within a 50m$\times$50m environment, each consisting of q random segments with lengths of at most 30m. For each $q \in \{4, 6, 8\}$, we generated 30 random environments and executed the planner with $r_{\max} \in \{0, 1, 2\}$ and $\alpha \in \{0.01, 0.02, \ldots, 0.1\}$ for each problem instance, resulting in a total of $(3 \times 30 \times 3 \times 10) = 2700$ trials. For each trial, we recorded whether a plan was found and the trial runtime. Figure 4 illustrates two successful plans found by our algorithm for $q = 4$ and 8, and a problem instance for $q = 6$ where no path was found. The trial for $q = 6$ in Figure 4 yields no plan because the goal segment's small size and orientation, being nearly perpendicular to all neighboring segments, severely restricted the available transition actions.

The impact of varying α on the planner's success rate is depicted in Figure 5. As anticipated, the results demonstrate an inverse relationship between success rate and α. We attribute this to the fact that increasing α expands the range of possible headings following a rotation, potentially hindering the robot's ability to navigate to X_G.

Figure 6 presents the results of varying the maximum resolution step $r_{\max}$ on the success rate for problem instances where $q \in \{4, 6, 8\}$ in the same trials used for Figure 5. Mayhaps surprisingly, the results demonstrate that increasing r only marginally increases success rate. Even for small values of r, the planner often finds a plan for a given problem instance and increasing r causes existing action intervals to be more densely sampled, only resulting in new problem instance solutions when higher resolution action sampling allows for new segments in S to be explored.

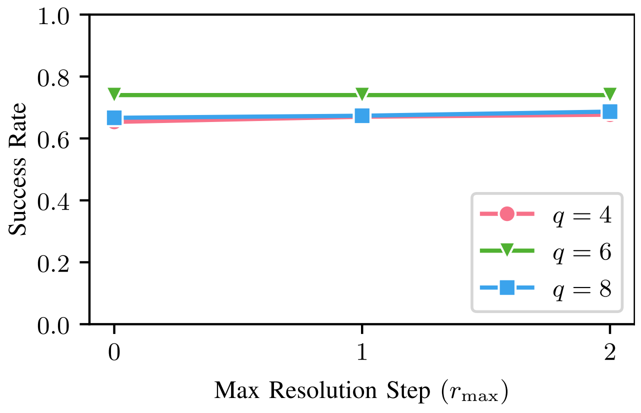

Fig. 6: The effect of maximum resolution step $r_{\max}$ on success rate across all rotational error values $\alpha \in \{0.01, 0.02, \ldots, 0.1\}$. As $r_{\max}$ increases, the planner's success rate increases for all $q \in \{4, 6, 8\}$.

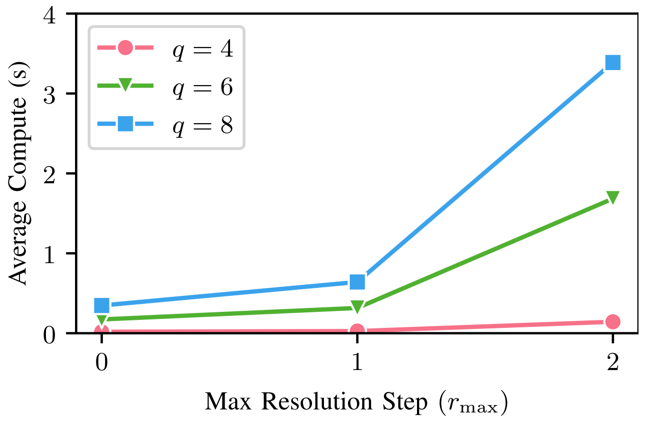

Fig. 7: The effect of the maximum resolution step $r_{\max}$ on average compute time across all rotational error values $\alpha \in \{0.01, 0.02, \ldots, 0.1\}$. As $r_{\max}$ increases, so does average compute time.

One might anticipate a monotonic relationship between success rate and segment count q. However, Figures 5 and 6 belie this expectation, as $q = 6$ generally outperforms $q \in \{4, 8\}$. This discrepancy illustrates the double-edged sword of adding segments: while new segments offer additional transition opportunities, they may also occlude existing ones. Consequently, these results suggest the existence of an optimal segment density for a given environment that maximizes success rate.

Figure 7 presents the computational time for the trials previously discussed. Increasing $r_{\max}$ may be beneficial; however, Figure 7 illustrates the algorithm's exponential dependence on $r_{\max}$. Therefore, the gains in success rate achieved by increasing $r_{\max}$ must be weighed against the corresponding increase in computational time.

VI. Feasibility of Physical Implementation

This section offers some evidence for the practical feasibility of the planning algorithm described in Section IV in the

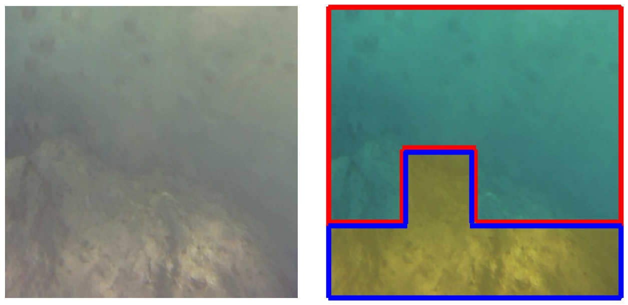

Fig. 8: [left] An image from Aqua2's downward-facing camera over Bellairs Reef, in which sand is visible at the top and a reef is visible on the bottom. [right] Our learned model correctly distinguishing reefs from sand given the captured image. The red outline indicates chunks classified as SAND; the blue outline shows chunks classified as REEF.

underwater navigation setting mentioned in Figure 1. We implemented a terrain classifier that uses RGB images captured by the downward-facing camera of an Aqua2 autonomous underwater vehicle to distinguish reefs and other rocky terrain from sandy terrain. We envision the Aqua2 using this classifier to execute plans generated by our planner, with the known boundaries between terrain types used as the segment set S.

The terrain classifier first divides the input image into 64×64 chunks. These chunks form the input to a MobileNetV2 [26] model with pre-trained weights from ImageNet. Training data were extracted from video captured by Aqua2's downward-facing camera during a deployment over Bellairs Reef off the coast of Barbados. From this hand-labeled data set, 544 reef chunks and 608 sand chunks were used for training. The resulting model accepts a 64×64 image chunk as input and produces a probability estimate that the chunk depicts a reef. Figure 8 shows an example application of this approach across a full camera image. For testing, we applied the learned model to a test set of 193 chunks, of which the model correctly classified 167, yielding an accuracy of approximately 0.87.

To classify the overall terrain of an input image, we compute the sum of reef probability for each chunk and compare this sum to half the number of chunks in the image. If the sum exceeds half of the total chunk count, the terrain classifier outputs REEF; otherwise, it outputs SAND. We evaluated this full-image terrain classifier on 54 images, including those shown in Figure 9. The classifier correctly classified all 54 images.

Preliminary tests using this classifier onboard the Aqua2 robot (Figure 1) showed two opportunities for continued improvement of the terrain change detector. First, the approach described here operates on that hardware at approximately 1Hz, which may be too slow for settings that require precise transitions at terrain boundaries. Second, the classifier showed only limited ability to generalize across variations in lighting and water conditions. Future research will investigate ways to resolve these two issues, likely by refining both the training data and the model architecture.

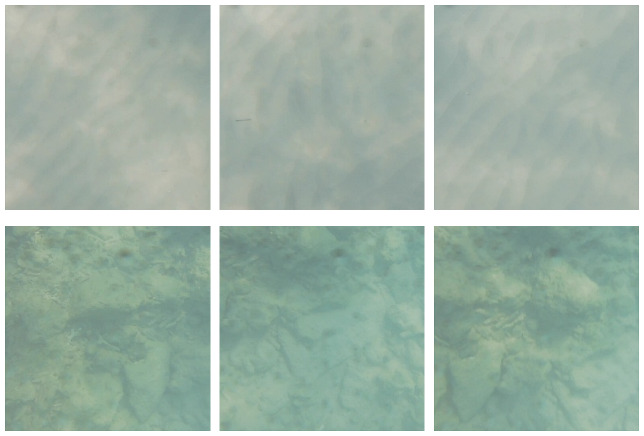

Fig. 9: Images of Bellairs Reef, captured by Aqua2's downward-facing camera. The top row depicts sand and the bottom row depicts reef. Our terrain classifier correctly classified all images.

VII. Conclusion

In this paper, we proposed a resolution complete planning algorithm for a navigation problem where the robot performs a type of action: rotate, subject to significant error, and move forward until sensing a segment. We evaluated an implementation of our algorithm and found it performed as expected, successfully overcoming the rotational error and finding plans that navigate the robot from the initial pose to the goal segment. We provided example trials to illustrate the correctness of this implementation.

Here we discuss some limitations of the current planner. Our algorithm's computational time exponentially depends on r which significantly limits r's realistic upper bound. Furthermore, we do not consider actions that could arrive at multiple segments. However, there exist environments where actions potentially arriving at multiple segments must be considered to find a successful plan.

Future work will investigate more efficient I-state tree construction methods, potentially incorporating heuristics to prioritize desirable I-states and employing a weighted queue to reflect these heuristics. Furthermore, we wish to explore extensions of this method where the robot can perform actions resulting in transitions to multiple segments, or utilize an enhanced segment detector that provides noisy readings of the robot's heading relative to the detected segment, thereby bounding its heading interval. Finally, we are interested in exploring a planning problem removing the constraint requiring a perfect sensor where, given a set of weighted candidate poses, their history, a map, and a goal, the algorithm selects an action that progresses towards that goal.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant Nos. 2313928 and 2313929. Additional support was provided by Texas A&M University.

References

[1] A.-a. Agha-mohammadi, S. Agarwal, S.-K. Kim, S. Chakravorty, and N. M. Amato, "Slap: Simultaneous localization and planning under uncertainty via dynamic replanning in belief space," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1195–1214, 2018.

[2] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[3] G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L.-A. Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Ripsman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades, "AQUA: An amphibious autonomous robot," *Computer*, pp. 46–53, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1109/MC.2007.6

[4] M. Erdmann and M. Mason, "An exploration of sensorless manipulation," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 369–379, Aug. 1988. [Online]. Available: https://ieeexplore.ieee.org/document/800

[5] L. H. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle, "Probabilistic localization with a blind robot," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1821–1827, iSSN: 1050-4729. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4543472

[6] M. J. Islam, Y. Xia, and J. Sattar, "Fast Underwater Image Enhancement for Improved Visual Perception," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3227–3234, Apr. 2020, conference Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9001231

[7] F. Jalal and F. Nasir, "Underwater Navigation, Localization and Path Planning for Autonomous Vehicles: A Review," in *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*, Jan. 2021, pp. 817–828, iSSN: 2151-1411. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9393315

[8] B. Joshi, S. Rahman, M. Kalaitzakis, B. Cain, J. Johnson, M. Xanthidis, N. Karapetyan, A. Hernandez, A. Q. Li, N. Vitzilaios, and I. Rekleitis, "Experimental Comparison of Open Source Visual-Inertial-Based State Estimation Algorithms in the Underwater Domain," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 7227–7233, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/8968049/?arnumber=8968049

[9] H. Kurniawati, "Partially observable markov decision processes and robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 253–277, 2022.

[10] M. Lauri, D. Hsu, and J. Pajarinen, "Partially observable markov decision processes in robotics: A survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 21–40, 2022.

[11] A. Lazanas and J.-C. Latombe, "Landmark-Based Robot Navigation," *Algorithmica*, vol. 13, no. 5, pp. 472–501, May 1995. [Online]. Available: https://doi.org/10.1007/BF01190850

[12] K. M. B. Lee, C. Yoo, B. Hollings, S. Anstee, S. Huang, and R. Fitch, "Online Estimation of Ocean Current from Sparse GPS Data for Underwater Vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 3443–3449, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8794308

[13] J. J. Leonard and A. Bahr, "Autonomous Underwater Vehicle Navigation," in *Springer Handbook of Ocean Engineering*, M. R. Dhanak and N. I. Xiros, Eds. Cham: Springer International Publishing, 2016, pp. 341–358. [Online]. Available: https://doi.org/10.1007/978-3-319-16649-0_14

[14] J. S. Lewis, D. A. Feshbach, and J. M. O'Kane, "Guaranteed Coverage with a Blind Unreliable Robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 7383–7390, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/8594048

[15] J. S. Lewis and J. M. O'Kane, "Planning for provably reliable navigation using an unreliable, nearly sensorless robot," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1342–1357, Sep. 2013, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364913488428

[16] T. Ma, S. Ding, Y. Li, and J. Fan, "A review of terrain aided navigation for underwater vehicles," *Ocean Engineering*, vol. 281, p. 114779, Aug. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0029801823011630

[17] T. Manderson and G. Dudek, "GPU-Assisted Learning on an Autonomous Marine Robot for Vision-Based Navigation and Image Understanding," in *OCEANS 2018 MTS/IEEE Charleston*, Oct. 2018, pp. 1–6, iSSN: 0197-7385. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8604645

[18] P. Mannam, A. Volkov Jr., R. Paolini, G. Chirikjian, and M. T. Mason, "Sensorless Pose Determination using Randomized Action Sequences," *Entropy*, vol. 21, no. 2, p. 154, Feb. 2019, arXiv:1812.01195 [cs]. [Online]. Available: http://arxiv.org/abs/1812.01195

[19] M. Modasshir and I. Rekleitis, "Enhancing Coral Reef Monitoring Utilizing a Deep Semi-Supervised Learning Approach," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 1874–1880, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9196528

[20] D. Nad, F. Mandić, and N. Mišković, "Using Autonomous Underwater Vehicles for Diver Tracking and Navigation Aiding," *Journal of Marine Science and Engineering*, vol. 8, no. 6, p. 413, Jun. 2020, number: 6 Publisher: Multidisciplinary Digital Publishing Institute. [Online]. Available: https://www.mdpi.com/2077-1312/8/6/413

[21] H. Nishimura and M. Schwager, "Sacbp: Belief space planning for continuous-time dynamical systems via stochastic sequential action control," *The International Journal of Robotics Research*, vol. 40, no. 10-11, pp. 1167–1195, 2021.

[22] M. Noseworthy, B. Tang, B. Wen, A. Handa, N. Roy, D. Fox, F. Ramos, Y. Narang, and I. Akinola, "Forge: Force-guided exploration for robust contact-rich manipulation under uncertainty," *arXiv preprint arXiv:2408.04587*, 2024.

[23] J. M. O'Kane and S. M. LaValle, "Localization with limited sensing," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 704–716, 2007.

[24] S. Rahman, A. Q. Li, and I. Rekleitis, "SVIn2: An Underwater SLAM System using Sonar, Visual, Inertial, and Depth Sensor," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 1861–1868, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8967703

[25] M. Roznere and A. Q. Li, "Real-time Model-based Image Color Correction for Underwater Robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 7191–7196, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8967557

[26] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4510–4520, iSSN: 2575-7075. [Online]. Available: https://ieeexplore.ieee.org/document/8578572

[27] J. Sattar, P. Giguere, G. Dudek, and C. Prahacs, "A visual servoing system for an aquatic swimming robot," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[28] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization," in *Robotics: Science and Systems IX*. Robotics: Science and Systems Foundation, Jun. 2013. [Online]. Available: http://www.roboticsproceedings.org/rss09/p31.pdf

[29] A. Sieverling, C. Eppner, F. Wolff, and O. Brock, "Interleaving motion in contact and in free space for planning under uncertainty," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 4011–4073.

[30] G. van Rossum, "Python tutorial," Centrum voor Wiskunde en Informatica, Amsterdam, Tech. Rep. CS-R9526, May 1995.

[31] E. Vidal, M. Moll, N. Palomeras, J. D. Hernández, M. Carreras, and L. E. Kavraki, "Online multilayered motion planning with dynamic constraints for autonomous underwater vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8936–8942.

[32] L. Whitcomb, D. Yoerger, and H. Singh, "Advances in doppler-based navigation of underwater robotic vehicles," in *Proceedings 1999 IEEE International Conference on Robotics and Automation*, 1999.

[33] L. L. Whitcomb, "Underwater robotics: out of the research laboratory and into the field," in *Proc. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, vol. 1, 2000.

[34] S. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards terrain-aided navigation for underwater robotics," *Advanced Robotics*, vol. 15, no. 5, pp. 533–549, Jan. 2001, publisher: Taylor & Francis _eprint: https://doi.org/10.1163/156855301317033559. [Online]. Available: https://doi.org/10.1163/156855301317033559

[35] M. Xanthidis, M. Kalaitzakis, N. Karapetyan, J. Johnson, N. Vitzilaios, J. M. O'Kane, and I. Rekleitis, "AquaVis: A Perception-Aware Autonomous Navigation Framework for Underwater Vehicles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2021, pp. 5410–5417, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9636124

[36] M. Xanthidis, N. Karapetyan, H. Damron, S. Rahman, J. Johnson, A. O'Connell, J. M. O'Kane, and I. Rekleitis, "Navigation in the Presence of Obstacles for an Agile Autonomous Underwater Vehicle," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 892–899, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/9197558

[37] X. Xu and S. Negahdaripour, "Vision-based motion sensing for underwater navigation and mosaicing of ocean floor images," in *Oceans '97. MTS/IEEE Conference Proceedings*, vol. 2, 1997, pp. 1412–1417 vol.2.

[38] D. R. Yoerger, "Precise control of underwater robots," in *International Advanced Robotics Programme Workshop on Mobile Robots for Subsea Environments*, 1990.

[39] D. Zheng, J. Ridderhof, P. Tsiotras, and A.-a. Agha-mohammadi, "Belief space planning: A covariance steering approach," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 051–11 057.

[40] K. Zheng and S. Tellex, "pomdp_py: A framework to build and solve pomdp problems," *arXiv preprint arXiv:2004.10099*, 2020.

[41] J. Zhou, T. Yang, and W. Zhang, "Underwater vision enhancement technologies: a comprehensive review, challenges, and recent trends," *Applied Intelligence*, vol. 53, no. 3, pp. 3594–3621, Feb. 2023. [Online]. Available: https://doi.org/10.1007/s10489-022-03767-y

[42] J. Zhu, S. Zhao, and R. Zhao, "Path Planning for Autonomous Underwater Vehicle Based on Artificial Potential Field and Modified RRT," in *2021 International Conference on Computer, Control and Robotics (ICCCR)*, Jan. 2021, pp. 21–25. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9349402