2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

An entropy-based hybrid local-global algorithm to navigate information-sparse environments

Bennett A. Carley Jason M. O'Kane

***Abstract*— We explore a navigation problem for a simple robot with extremely noisy sensing and significant movement uncertainty. We are particularly interested in environments containing large regions in which relatively little distinguishing sensor information is available to assist with localization. This paper proposes a navigation algorithm for this setting that strategically directs the robot through such regions when possible, but with a careful view of the need to regain relatively accurate localization at certain points in the execution. Reasoning directly about the robot's uncertainty, the approach utilizes a local entropy metric to identify regions where sensors have strong informative value. This metric informs the selection of coarse global paths that guide a more precise local planner. We discuss an implementation of this algorithm, and provide simulation results demonstrating its effectiveness in spite of large errors in both sensing and actuation.**

I. Introduction

The underwater domain presents a unique set of challenges that render many standard navigation techniques ineffective. Autonomous underwater vehicles (AUVs) such as the Aqua2 (Figure 1) typically lack access to reliable GPS signals, forcing them to surface periodically or remain within a limited operational area serviced by acoustic buoys [27]. Motion is further complicated by unpredictable currents and waves, introducing significant drift and control inaccuracies. Moreover, perception is severely hampered; even modern methods that leverage visual data, such as those using vision-language models (VLMs) [35], are unreliable underwater due to poor visibility caused by suspended particulate matter and rapid light attenuation [8]. These compounding factors necessitate navigation strategies that operate effectively under extreme uncertainty. A more comprehensive review of related literature is provided in Section II.

In this work, we address the challenge of navigating a robot from an uncertain initial state to a specified goal location under sensor and motion uncertainty. While motivated by the underwater domain, this problem formulation is broadly applicable to other scenarios, such as a ground vehicle localizing via approximate distances to landmarks or an unmanned aerial vehicle (UAV) navigating in a GPS-denied environment.

We model the robot's dynamics using a unicycle model with multiplicative, normally distributed control errors. Its perception is modeled by real valued sensors, such as depth and altitude sensors on an AUV, measuring environmental features subject to significant additive, normally distributed noise. This problem, including the formal robot and observation models, is detailed in Section III.

B. A. Carley and J. M. O'Kane ({bcarl, jokane}@tamu.edu) are with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. This material is based upon work supported by the U.S. National Science Foundation under Award No. 2313928.

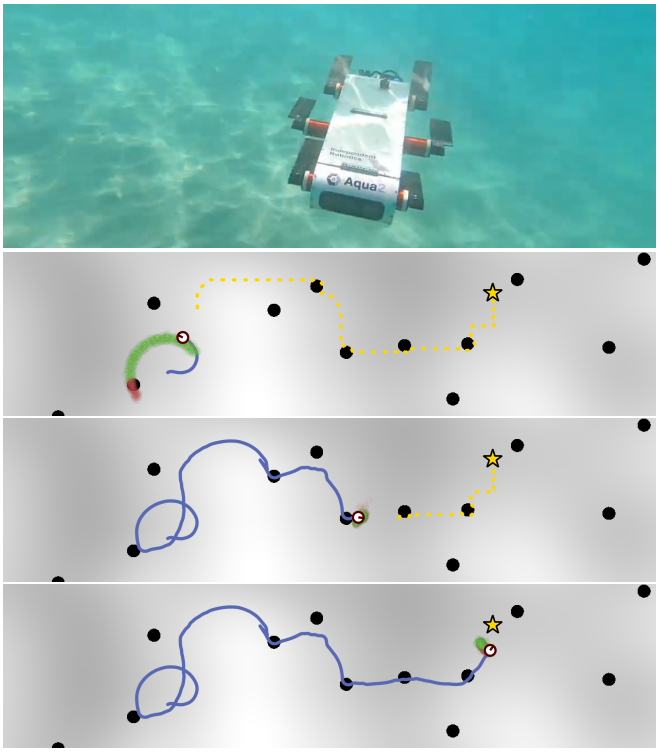

Fig. 1: [top] The Aqua2 AUV experiences significant noise in its motion due to complex dynamics and unknown ocean currents. Available sensor data is extremely sparse, even in relatively shallow clear water. The proposed algorithm provides a method for robust navigation with this platform despite these limitations. [bottom] An illustration of our algorithm completing an example trial. The robot starts within the circle (radius of 1 m) with a heading $\theta \in [0, \pi)$. Despite this initial uncertainty and notable sensing and control error, the robot first reduces its uncertainty and then follows the offline path (dashed line) to successfully reach the goal location (star).

To address this challenge, we propose a novel planning algorithm. Our approach uses a particle filter to indirectly maintain a probability distribution over possible robot states, which is updated based on sensor observations and robot movements. A key idea is leveraging entropy as a proxy for the informativeness of a set of possible sensor readings. Perhaps counterintuitively, an action with a high-entropy distribution of possible observations has strong informative value for improving the robot's localization; likewise, a low-entropy observation distribution corresponds to a likely increase in the robot's state uncertainty.

Our algorithm builds on this observation. First, we introduce a method of estimating the approximate expected

entropy of a given action. Then, we propose a global planner to find coarse paths maximizing the expected action entropy per unit distance. We introduce a local online planner to balance actions with high expected entropy with those that make progress towards the goal. We combine these two elements, using subgoals generated by the global planner as targets for the local planner. Detailed explanations of these algorithms appear in Section IV.

We evaluate the effectiveness of our proposed algorithm through simulated trials in various environments with large information sparse regions, comparing its performance against several baselines. Our experimental results, detailed in Section V, demonstrate the efficacy of our approach. A representative trial, shown in Figure 1, illustrates the algorithm successfully guiding the robot to its goal despite significant initial uncertainty along with noisy control and sensing. A characterization of the environmental properties where our algorithm can be expected to perform well is presented in Section VI. We then present limitations of our approach and outline future research directions in Section VII.

II. Related Work

A. Navigation Under Uncertainty

The problem of navigating despite environmental or control uncertainty is an active area of research with a significant amount of prior work. Early work explored how robots can succeed with minimal sensing. For instance, Erdmann developed methods for planning motions by backprojecting from goal states to find regions from which success is guaranteed [5]. Later, Erdmann and Mason demonstrated that a specific sequence of actions can orient a randomly placed object, with follow-up work by Mannam et al. showing that even a random action sequence is sufficient for some objects [6], [22]. Other foundational research explored methods to exploit the structure of the environment. Lazanas and Latombe investigated navigation in environments with distinct landmarks where the robot has perfect localization and control within an area around these landmarks but no sensing and noisy control outside of these areas [17]. Similarly, Donald and Jennings proposed using perceptual equivalence classes to partition the world into regions that are indistinguishable to the robot's sensors, forming natural landmarks for navigation [4]. A significant thread of subsequent work has continued to explore the minimum sensing requirements for localization, navigation, and coverage, demonstrating what is possible for robots that are nearly blind, equipped only with contact sensors and a compass or odometer [7], [18]–[20], [25]. We have been inspired by the types of navigation problems explored in these works and aim to characterize the types of environments where algorithms of this type succeed.

More recent work has focused on planning explicit information-gathering actions and managing uncertainty within a probabilistic framework. A key formalism for these problems is the Partially Observable Markov Decision Process (POMDP), which provides a principled mathematical foundation for planning under uncertainty [16], [31]. A variety of solution techniques and software frameworks for POMDPs have been developed [3], [15], [39]. A central concept in this area is planning in belief space or the space of probability distributions over the robot's state. Many modern planners search for a path or policy in this belief space [1], [2], [24], [38]. These methods often focus on active information acquisition and informative path planning, where a robot's path is optimized to reduce uncertainty in an underlying state or map [13], [26]. These methods have been extended to multi-robot systems for tasks like distributed environmental modeling and adaptive sampling [21]. Recent approaches have also tightly integrated semantic understanding with geometric navigation, allowing a robot to simultaneously search for and localize semantic objects in unknown environments [28].

B. Coastal Navigation

The concept of coastal navigation has been used in robotics as an analogy for navigating by keeping known, feature-rich areas within sensor range to maintain good localization. Roy et al. demonstrated this concept with a tour-guide robot in a museum, which treated walls and exhibits as a coastline to reduce positional uncertainty in an environment with dynamic obstacles like people [29].

This concept applies quite literally for marine robotics, when Unmanned Surface Vehicles (USVs) must operate in GPS-denied or spoofed environments. Han et al. propose a GPS-less coastal navigation system that uses marine radar to gather information about the surrounding coastline and matches these readings with a prior map to estimate the vehicle's position [9].

C. Underwater Navigation

Underwater navigation presents a distinct and significant set of challenges, most notably the absence of GPS signals, unreliable communication, and dynamic ocean currents. Several recent surveys provide a comprehensive overview of the state-of-the-art in Autonomous Underwater Vehicle (AUV) navigation, covering techniques from dead reckoning and acoustic localization to multi-sensor information fusion and Simultaneous Localization and Mapping (SLAM) [33], [36].

Given the harsh environment, robustness to sensor or actuator failures is critical. Zhao et al. developed a particle filter-based approach that can perform fault diagnosis and maintain a reliable state estimate for an underwater robot even when multiple faults occur simultaneously [37]. Visual sensing is particularly challenging underwater, and work has been done to develop methods for real-time image color correction [30] and to experimentally compare the performance of various visual-inertial state estimation algorithms in the underwater domain [12]. Other work has focused on leveraging AUVs to assist human divers with tracking and navigation aiding [23]. More recently, there has been a focus on tightly integrating the planning and control loops to handle environmental uncertainty. For instance, Xanthidis et al. developed ResiPlan, a motion planning framework that monitors the robot's path-following performance and adapts its path's safety margins

in real-time to account for disturbances like ocean currents and uncertainty growth, enabling safer navigation in cluttered underwater environments [34].

III. Problem Statement

A. Robot Model

We model a point robot navigating a planar environment $\mathcal{W} = \mathbb{R}^2$ using a unicycle model. The robot's configuration space is $\mathcal{C} = \mathcal{W} \times \mathbb{S}^1$ and configurations in $\mathcal{C}$ are represented as $q = (x, y, \theta)$. The component $(x, y) \in \mathbb{R}^2$ encodes the robot's position and $\theta \in \mathbb{S}^1$ encodes its heading.

We model time as a non-negative real number $t \in \mathbb{R}_{\geq 0}$. The robot selects actions at a fixed rate $\Delta t^{(u)} \in \mathbb{R}_{>0}$, the robot selects action u from the action space $U = [-v_{\max}, v_{\max}] \times [-\omega_{\max}, \omega_{\max}]$ to directly control its linear and angular velocities. We index actions with subscript $k \in \mathbb{N}_0$ to denote action $u_k = (v_k, \omega_k)$, which occurs in the time interval $t \in [k\Delta t^{(u)}, (k+1)\Delta t^{(u)})$. Simultaneously, an independent decision maker called nature selects action $\xi_k = (\alpha_k, \beta_k)$ to occur over the same time period as u_k from the nature action space $\Xi = \mathbb{R}^2$ subject to $\alpha_k \sim \mathcal{N}(1, \sigma_v^2)$ and $\beta_k \sim \mathcal{N}(1, \sigma_\omega^2)$ to multiplicatively perturb the action u_k. For any time t, we can write $t = k\Delta t^{(u)} + \delta$ for some $k \in \mathbf{N}$ and some $\delta \in [0, \Delta t^{(u)})$. We define the configuration transition function f as:

$$q(t) = f(q_k, u_k, \xi_k, t) \tag{1}$$

$$q(k\Delta t^{(u)} + \delta) = \begin{cases} \begin{bmatrix} x_k + \alpha_k v_k \delta \cos(\theta_k) \\ y_k + \alpha_k v_k \delta t \sin(\theta_k) \\ \theta_k \end{bmatrix} & \beta_k \omega_k = 0 \\ \begin{bmatrix} x_k + \frac{\alpha_k v_k}{\beta_k \omega_k}(\sin(\delta\theta_k) - \sin(\theta_k)) \\ y_k + \frac{\alpha_k v_k}{\beta_k \omega_k}(\cos(\theta_k) - \cos(\delta\theta_k) \\ (\theta_k + \beta_k \omega_k \delta) \pmod{2\pi} \end{bmatrix} & \beta_k \omega_k \neq 0 \end{cases}$$

The above definition makes use of the notation $q(k\Delta t^{(u)}) = (x_k, y_k, \theta_k)$.

B. Observation Model

The robot is equipped with n independent, real-valued sensors to observe the environment. For each sensor $i \in \{1, \ldots, n\}$, an observation function $h_i : \mathcal{W} \to \mathbb{R}$ maps the robot's true position to a noise-free measurement. Let $p(t) \in \mathcal{W}$ be the robot's position at time t. We assume observations are made at discrete time steps $t_k = k\Delta t_i$. At each step, sensor i produces a noisy observation $y_{i,k}$:

$$y_{i,k} = h_i(p(k\Delta t_i)) + \epsilon_{i,k} \tag{2}$$

The term $\epsilon_{i,k}$ represents additive white Gaussian noise, with each noise term $\epsilon_{i,k}$ being a single, independent sample drawn from the distribution $\mathcal{N}(0, \sigma_i^2)$. The variance σ_i^2 for each sensor is assumed to be known.

The sequence of observations from sensor i over a time interval $t \in [t_0, t_1]$ is collected in a history vector $H_i(t_0, t_1) = (y_{i,k} \mid t_0 \leq k\Delta t_i \leq t_1)$, in which the observations are ordered by time.

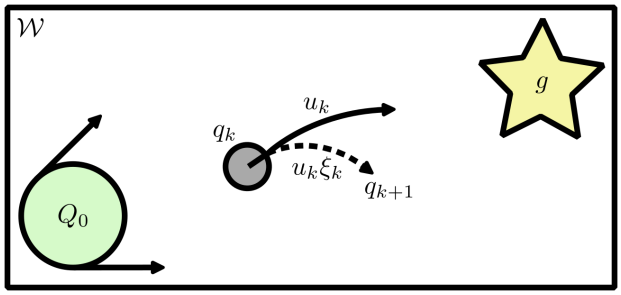

Fig. 2: The robot starts with an unknown state q_0 within the green circle with a heading in the range indicated by the arrows (Q_0). The robot at state q_k selects action u_k but is perturbed by nature's action ξ_k, resulting in state q_{k+1}.

C. Planning Objective

This paper addresses the problem of navigating this robot from an unknown initial state q_0 in a set of possible initial states Q_0 to goal location $g \in W$. We propose an algorithm that selects actions designed to minimize the robot's expected distance from the goal upon termination.

Figure 2 provides an illustrated example of some of the notation found in this section.

IV. Algorithm Description

In this section, we present an algorithm to address the problem formulated in Section III. Our approach has three primary components: (1) a computationally efficient method for approximating the entropy of Gaussian mixtures (Section IV-A), (2) an algorithm that precomputes coarse paths rich in local entropy (Section IV-B), and (3) a dynamic online algorithm that follows these paths while balancing progress against information gain (Section IV-C).

Our method assumes the robot uses a particle filter $\mathcal{P}$ to localize itself. The specific details of the particle filter are immaterial but it must be able to: (1) forward project the particle cloud given an action (ForwardProject), (2) reweigh the particles (Reweigh), (3) find the location of each particle at a given time (GetLocations).

A. Entropy

The ability to efficiently approximate the entropy of a Gaussian Mixture Model (GMM) is a cornerstone of our algorithm. We adopt the deterministic method introduced by Huber et al. to meet our dual requirements of computational efficiency and accuracy [11]. This approach is based on a component-wise Taylor-series expansion of the logarithm within the entropy integral. This method allows us to balance computational cost against accuracy by selecting the order of the Taylor-series expansion and the number of splits applied to each Gaussian component. Algorithm 1 formalizes this procedure. This algorithm calculates the approximate entropy $\mathcal{E}$ for a given GMM $\mathcal{G} = \{(w_k, \mu_k, \Sigma_k)\}_{k=1}^K$, desired Taylor-series order R, and number of component splits M.

We leverage Algorithm 1 to estimate the expected sensor entropy of a given action. To do so, we first exploit the particle filter to provide a noisy forward projection of this

Algorithm 1: APPROXIMATEENTROPY($\mathcal{G}$, R, M)

// Approximate entropy of a Gaussian mixture $\mathcal{G} = \{(w_k, \mu_k, \Sigma_k)\}_{k=1}^{K}$
$\mathcal{G}' \leftarrow \mathcal{G}$
for $m \in \{1, \ldots, M\}$ **do**
  // Split components [10]
  $\mathcal{G}' \leftarrow$ SPLITCOMPONENTS($\mathcal{G}'$)
end
$\mathcal{E} \leftarrow 0$
for $(w_k, \mu_k, \Sigma_k) \in \mathcal{G}'$ **do**
  // Component-wise Taylor-series expansion [11]
  $T_k \leftarrow$ TAYLOREXPANSION $(\log p(x), \mu_k, R)$
  $\mathcal{E} \leftarrow \mathcal{E} - w_k \int \mathcal{N}(x \mid \mu_k, \Sigma_k) T_k(x)\, dx$
end
return $\mathcal{E}$

Algorithm 2: ACTIONENTROPY($\mathcal{P}, u, t_0, t_1$)

// Estimate the expected sensor entropy of action u from t_0 to t_1
$\mathcal{P}' \leftarrow$ FORWARDPROJECT($\mathcal{P}, u, t_0$)
$\mathcal{E}_u \leftarrow 0$
for $i \in \{1, \ldots, n\}$ **do**
  $\mathcal{G} \leftarrow \varnothing$
  $\mathcal{T}_i \leftarrow \{k\Delta t_i \mid k \in \mathbb{N}_0, t_0 \le k\Delta t_i \le t_1\}$
  $\Sigma_i \leftarrow \mathrm{diag}(\sigma_i^2, \ldots, \sigma_i^2) \in \mathbb{R}^{|\mathcal{T}_i| \times |\mathcal{T}_i|}$
  for $(p, w_p) \in \mathcal{P}'$ **do**
    $\mu_p \leftarrow ()$
    for $t \in \mathcal{T}_i$ **do**
      $\mu_p \leftarrow \mu_p \oplus h_i($GETSTATE$(p, t))$
    end
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{(w_p, \mu_p, \Sigma_i)\}$
  end
  $\mathcal{E}_u \leftarrow \mathcal{E}_u +$ APPROXIMATEENTROPY$(\mathcal{G}, 2, 0)$
end
return $\mathcal{E}_u$

Algorithm 3: SCALEDLOCALENTROPY($\mathcal{A}, r_s$)

// Determine the scaled local entropy of area $\mathcal{A}$ at resolution r_s
$\Sigma \leftarrow \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$
$\mathcal{G} \leftarrow \varnothing$
for $s \in$ UNIFORMSAMPLE$(\mathcal{A}, r)$ **do**
  $\mu_s \leftarrow (h_1(s), \ldots, h_n(s))$
  $\mathcal{G} \leftarrow \mathcal{G} \cup (n^{-1}, (h_1(s), \ldots, h_n(s)), \Sigma)$
end
$\min \leftarrow \frac{1}{2}\log((2\pi e)^n \prod_1^n \sigma_i^2)$
$\max \leftarrow \min + \log(n)$
$\mathcal{E} \leftarrow$ APPROXIMATEENTROPY$(\mathcal{G}, 2, 0)$
return $(\mathcal{E} - \min)/(\max - \min)$

action. Then, for each sensor h_i and particle $p \in \mathcal{P}$, we create a vector of expected observations from $t = t_0$ to $t = t_1$. For all sensor $i \in \{1, \ldots, n\}$, we create GMM $\mathcal{G}_i = \{(w_k, \mu_k, \Sigma_k)\}_{k=1}^{|\mathcal{P}|}$ where w_k corresponds to the weight of particle k, μ_k is the particle's observation vector for sensor i, and Σ_k is a $|\mu_k| \times |\mu_k|$ diagonal matrix of all σ_i^2. The summation of the approximate entropy of each GMM (Algorithm 1) approximates the expected entropy of a given action under our assumption of sensor independence.

Furthermore, the ability to approximate sensor entropy in a given area $\mathcal{A} \subset \mathcal{W}$ is also of interest. To do so, we propose the following sampling based algorithm. First, uniformly sample the area at resolution $r_s \in \mathbb{R}_{>0}$. At each sampled location $s_i \in \{s_1, \ldots, s_m\}$, we define the observation vector as $\mu_i = \{h_1(s_i), \ldots, h_n(s_i)\}$, the observation weight as $w_i = 1/m$, and the observation covariance as $\Sigma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ under our assumption of sensor independence. We then use Algorithm 1 to approximate the entropy of $\mathcal{G} = \{(w_i, \mu_i, \Sigma_i)\}_{i=1}^{m}$. Differential entropy can be negative and so we linearly scale these values between zero and one with $\mathcal{E}_{\min}$ as the lower bound and $\mathcal{E}_{\max}$ as the upper bound. Under the assumptions of equal weights and identical diagonal covariance matrices, the minimum entropy of the above GMM is $\mathcal{E}_{\min} = \frac{1}{2}\log((2\pi e)^n \prod_1^n \sigma_i^2)$. Similarly, $\mathcal{E}_{\max} = \mathcal{E}_{\min} + \log(n)$. This scaled value is used as a proxy for the amount of distinguishing information within a given area and formalized in Algorithm 3.

B. Precomputed Paths

To find coarse paths through environments with varying levels of distinguishing sensor information, we propose the following algorithm. First, create an infinite grid of $l \times l$ cells. We then copy this grid twice and shift one copy up $l/2$ units and the other right by $l/2$ units. For each cell contained entirely in $\mathcal{W}$, determine its scaled local entropy using Algorithm 3 at resolution r_s. By performing this shifting, we ensure we avoid the case wherein one cell contains all zeros and its neighbor contains all ones so both cells have a scaled local entropy of zero and the value of this highly informative edge is lost.

We represent each of these cells as vertices in a graph and draw directed edges to both their cardinal and diagonal neighbors. The weight of said edges is calculated as the distance from the center of the source cell to the target cell divided by the scaled local entropy of the target cell. This represents the expected local informativeness per meter when traveling from one cell to another. With this graph, we find the shortest path from each vertex to the vertex whose center is closest to g. Algorithm 4 outlines this process.

Given this set of paths, we would like to find some target waypoint for our robot to make progress towards. To determine the robot's current waypoint, we find the weighted center of the particle cloud at a given time. We then find the vertex with a center nearest this weighted center, extract the path from this vertex to the goal, and find the vertex furthest along the path whose center is within distance d_w of some

Algorithm 4: GETPATHS($\mathcal{W}$, l, r_s, g)

```
// Find paths to g with the most
   local information per meter
V ← GENERATEGRIDCENTERS(W, l)
E ← ∅
E_E ← ∅
for v ∈ V do
    A_v ← SQUARE(v, l)
    E_E[v] ← SCALEDLOCALENTROPY(A_v, r_s)
end
for u ∈ V do
    for v ∈ GETNEIGHBORS(u, V) do
        d ← DISTANCE(u, v)
        w ← d / E_E[v]
        E ← E ∪ {(u, v, weight)}
    end
end
return SHORTESTPATHSTO(V, E, g)
```

Algorithm 5: GETWAYPOINT($\mathbb{P}, \mathcal{P}, t, d_w$)

```
// Along the optimal path, find the
   furthest waypoint within distance
   d_w of any particle
μ ← WEIGHTEDMEAN(P)
for w ∈ REVERSEDNEARESTPATH(P, μ) do
    for p ∈ P do
        if DISTANCE(GETSTATE(p, t), w) < d_w then
            return w
        end
    end
end
return g
```

particle in $\mathcal{P}$.

C. Local Planner

A well designed online local planner should balance informative actions with progress seeking actions to navigate the robot through areas with mild amounts of informative locations without loss in localization.

To navigate through these mildly informative areas, we propose Algorithm 7. We first sample the action space U at a given resolution $r_u \in \mathbb{N}_0$ to find a set of candidate actions $U_r = \{\pm 2^{-r} v_{\max} | r \in \{0, \dots, r_u\}\} \times \{\pm 2^{-r} \omega_{\max}, 0 | r \in \{0, \dots, r_u\}\}$. We sample actions more densely at lower speeds to ensure that low frequency sensors can capture high frequency information. This process of sampling actions is formalized in Algorithm 6.

Given a set of candidate actions U_r, we select the action that best balances information gain and progress towards the goal. To do so, we determine the action's expected sensor entropy using Algorithm 2 from times t_0 to $t_0 + \Delta t^{(\mathcal{E})}$ for some $\Delta t^{(\mathcal{E})} \geq 2\Delta t^{(u)}$. We then linearly scale the result between zero and one using the smallest and largest action entropy encountered by Algorithm 2 as the upper and lower bounds. We choose to scale between the minimum and maximum entropy encountered so far instead of a theoretical bound to decrease the likelihood of all these values being clustered together. Similarly, we find the expected progress resulting from each action by finding the weighted distance of the particle cloud to the goal before and after the action and then subtracting the distance before from the distance after. Once again, this progress value is normalized between zero and one using the maximum local progress and minimum local progress as the upper and lower bounds to ensure the algorithm values progress even when near the goal. Then we select the action maximizing $\delta\mathcal{E} + (1 - \delta)p$ where $\mathcal{E}$ is the normalized expected action information and p is the normalized progress towards the goal.

Algorithm 6: SAMPLEACTIONS($v_{\max}$, $\omega_{\max}$, r_u)

```
// Sample actions from U at
   resolution r_u
V_r, Ω_r ← ∅, {0}
for r ∈ {0, ..., r_u} do
    V_r ← V_r ∪ ({−v_max, v_max} ⊘ 2^r)
    Ω_r ← Ω_r ∪ ({−ω_max, ω_max} ⊘ 2^r)
end
return V_r × Ω_r
```

The tradeoff value δ determines the information gain versus progress tradeoff where $\delta = 1$ represents fully goal seeking behavior and $\delta = 0$ represents fully information seeking behavior. The value of δ is calculated dynamically to prioritize goal seeking behavior when the robot's state uncertainty is low, and information seeking behavior when its uncertainty is high. We use the particle cloud's average weighted distance from its weighted mean, d_μ, as a proxy for the robot's localization confidence. Therefore, we dynamically set the tuning value according to $\delta = \text{CLIP}(0.1, d_\mu / d_{\max}, 0.9)$ where $d_{\max}$ is the maximum acceptable expected distance to the mean. We bound $\delta \in [0.1, 0.9]$ to ensure the algorithm always exhibits some information seeking and some goal seeking behavior. This dynamic online algorithm is outlined in Algorithm 7.

D. Hybrid Planner

Here, we combine our precomputed paths with our local planner to enable navigation through environments with varying levels of informative density. We do this by first precomputing paths $\mathbb{P}$ using Algorithm 4. Then, at each $t \in k\Delta t^{(u)}$ we pass the precomputed paths $\mathbb{P}$ to Algorithm 5 to find subgoal g_k. We pass subgoal g_k to our local planner (Algorithm 7) and then execute the returned action. If no progress towards the goal can be made, the planner terminates.

An evaluation of each part of this algorithm can be found in Table I and these results are discussed in Section V.

Algorithm 7: GETACTION($\mathcal{P}$, U_r, g, t_0, $\Delta t^{(\mathcal{E})}$, $d_{\max}$)

```
// Select action balancing
   information gain and progress
μ ← WEIGHTEDMEAN(P, t_0)
d_μ ← WEIGHTEDDISTANCE(P, t_0, μ)
δ ← CLIP(0.1, d_μ/d_max, 0.9)
d ← WEIGHTEDDISTANCE(P, t_0, g)
Z ← ∅
for u ∈ U_r do
    P' ← FORWARDPROJECT(P, u, t_0)
    d' ← WEIGHTEDDISTANCE(P', t_0 + Δt^(u), g)
    E_u ← ACTIONENTROPY(P', u, t_0, t_0 + Δt^(E))
    Z ← Z ∪ {(u, d' − d, E)}
end
Z ← NORMALIZEENTROPYANDPROGRESS(Z)
return argmax_{(u,p,E)∈Z} (δe + (1 − δ)p)
```

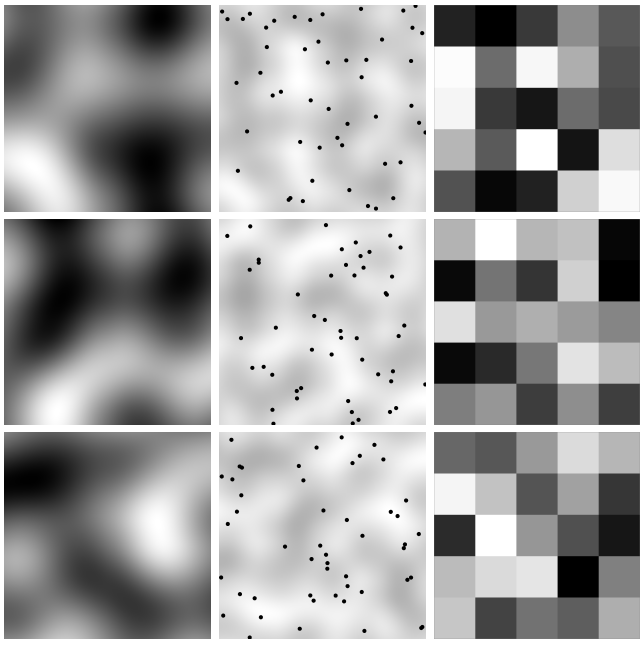

Fig. 3: All nine environment instances used in testing. The environment types appear as follows: **Perlin** (left), **Astroid** (center), and **Grid** (right).

V. Implementation and Evaluation

To validate our proposed algorithm, we conducted a series of simulated experiments. This section details the experimental setup and results of these experiments.

A. Experimental Setup

Environments. We evaluate our algorithm across three distinct types of environments, designed to present different information landscapes. For each type, we generated three unique instances and they are shown in Figure 3.

Perlin A two-dimensional 1000×1000 Perlin noise grid where each cell represents a 0.2×0.2 meter area. This creates a smooth, unstructured environment with evenly distributed information with values in $[-1, 1]$.

TABLE I: Average Distance to Goal (m)

	Naive	Static	Dynamic	Precompute	Hybrid
Perlin	8.98	7.76	3.87	5.53	**3.40**
Astroid	16.27	12.18	4.98	22.00	**4.77**
Grid	4.52	3.39	2.61	4.37	**1.43**

The average distance to goal in meters from g at termination for each algorithm and environment type. The hybrid algorithm performs best in all three environment types.

TABLE II: Average Path Length (m)

	Naive	Static	Dynamic	Precompute	Hybrid
Perlin	121.81	105.61	139.82	119.66	154.53
Astroid	138.37	150.77	122.10	129.51	128.49
Grid	152.21	103.86	118.45	185.57	183.17

The average path length from q_1 to g in meters for each algorithm and environment type. Interestingly, this illustrates the paths taken by the hybrid algorithm are longer than those taken by the static and dynamic algorithms.

Astroid This instance modifies the **Perlin** environment by reducing the background information values (dividing each by 4) and then setting the value of all points within $2m$ of 50 randomly selected locations to 1. This process creates information hot-spots in a generally less informative map.

Grid A 5×5 grid of large $40m \times 40m$ cells, where each cell is assigned a value selected uniformly at random from $[0, 1]$. In this map, information is concentrated at the edges and corners between cells.

Procedure. For each of the nine total environment instances, we selected 30 goal locations evenly spread out along the circumference of a circle centered at $(100m, 100m)$ with a radius of $75m$. The robot started at a random state $q_1 = (x, y, \theta)$ within a circle centered at $(100m, 100m)$ with a radius of $1m$ and heading $\theta \in [0, \pi)$. The simulated robot model has $\Delta t^{(u)} = 1$, $v_{\max} = 1$, $\omega_{\max} = 0.5$, $\sigma_v = 0.1$, $\sigma_\omega = 0.05$, and one sensor h_1 that returns a noisy measure of the environment value at a given location every $\Delta t_0 = 1$ seconds where $\sigma_1 = 0.4$. Furthermore, for Algorithm 1, we set $l = 1$ meters and sample at a $r_s = 0.2$ meter resolution and for Algorithm 5 $d_w = 10$ meters. Actions are sampled at resolution $r_u = 3$ and, for Algorithm 7, $d_{\max} = 5$ meters, $\Delta t^{(\mathcal{E})} = 10$ seconds, for both static and dynamic $w = g$, and for hybrid w calls Algorithm 5.

Algorithms Compared. We performed a simulated run for five different algorithms on each environment instance-goal pair: (1) naive, (2) local with static δ, (3) local with dynamic δ, (4) precomputed paths + naive, and (5) precomputed paths + local with dynamic δ. We reference these as the naive, static, dynamic, precompute, and hybrid algorithms, respectively. In total, we performed $1,350$ trials (3 types $\times$ 3 instances $\times$ 30 goals $\times$ 5 algorithms). In each trial, we recorded the robot's distance to goal at termination, the distance traveled, and the execution time.

Implementation. The entropy calculation in Algorithm 1 was implemented in Rust [14], and all other code was written in Python [32]. Our trials were run on a system with an Intel Core i7-13700KF at 3.40 GHz and 32 GB of DDR5 RAM.

TABLE III: Average Compute Time (s)

	Precompute	Local/Action	Total Local	Overall Total
Perlin	30.90	0.10	23.38	68.44
Astroid	32.53	0.10	16.63	58.13
Grid	35.39	0.09	22.46	63.89

The average compute time over all runs for each component of the hybrid algorithm. The total time is greater than the sum of precompute and total local due to simulation overhead.

B. Results and Discussion

Our results are summarized in Table I, Table II, and Table III. The results in Table I show the hybrid algorithm consistently outperforms all other algorithms in reaching the goal, achieving the lowest average distance to the goal at termination across all environment types. Table I also illustrates the value of dynamically tuning δ because while the static algorithm performed better than the naive algorithm with $\delta = 0.8$, the dynamic algorithm performed significantly better than both in all environment types. Note that the precompute algorithm performs at a level between the naive and the static algorithms, while the hybrid algorithm performs best. This suggests the precomputation algorithm generates informative paths, but these paths are most helpful when paired with a local component to effectively exploit them.

Table II reveals a trade-off. The average path length of the hybrid algorithm is the longest in the **Perlin** and **Astroid** environments and second to the precompute algorithm in the **Grid** environment. This demonstrates the dynamic algorithm exploits the information-rich precomputed paths to achieve better results despite longer travel distances.

The computational cost of our hybrid algorithm is reasonable and broken down in Table III. The path precomputation takes less than 40 seconds on average and the dynamic algorithm takes around 0.1 seconds per call. We believe the added precompute time needed for the hybrid over the dynamic algorithm is a worthwhile tradeoff because it can be done relatively quickly before deploying a robot and significantly improves results.

VI. Information Deserts

Areas with limited sensor information often pose a challenge to robot navigation. We introduce the concept of an (r, p)-information desert to characterize sensor ambiguity within a given region in $\mathcal{W}$.

Definition. Given the Mahalanobis distance

$$d(a, b) = \sqrt{\sum_{i=1}^{n} (h_i(a) - h_i(b))^2 / \sigma_i^2}$$

between $a, b \in \mathcal{W}$, an (r, p)-information desert is a circle $C(c, r) \subset \mathcal{W}$ centered at c with radius r for which

$$p \geq \max_{a,b \in C(c,r)} \left[1 - \frac{1}{2} \operatorname{erfc}\left(\frac{d(a, b)}{2\sqrt{2}} \right) \right].$$

Intuitively, p acts as an upper bound on the best-case probability of successfully distinguishing between any two locations within the circle $C(c, r)$. The derivation of this probability is based on a statistical hypothesis test. Under our observation model, any two points $a, b \in \mathcal{W}$ correspond to mean observation vectors $\mu_a = (h_1(a), \ldots, h_I(a))$ and $\mu_b = (h_1(b), \ldots, h_I(b))$, with sensor noise described by a shared covariance matrix $\Sigma = \operatorname{diag}(\sigma_1^2, \ldots, \sigma_I^2)$. If the robot is known to be at either location a or b, we can test Hypothesis H_a (an observation is from a) against Hypothesis H_b (it is from b). An optimal Bayesian classifier with equal priors ($P(H_a) = P(H_b)$) resolves this by selecting the hypothesis with the greater likelihood.

The success of this classifier depends on the distinguishability of the two sensor distributions, which we measure using the Mahalanobis distance. The probability of this optimal classifier making an error is a direct function of this distance:

$$P_{\text{error}} = \frac{1}{2} \operatorname{erfc}\left(\frac{d(a, b)}{2\sqrt{2}} \right)$$

The expression inside the definition's $\max$ operator, $1 - P_{\text{error}}$, is therefore the probability of correctly distinguishing between a and b. When comparing information deserts of the same radius, a smaller value of p corresponds to a more severe desert. Conversely, for a fixed value of p, the desert's severity increases with its radius r.

To understand how to characterize different environments, we classify each environment type discussed in this paper in terms of (r, p)-information deserts. The values of $p_{\max}$ were calculated analytically and provide an upper bound for p in the regions discussed:

Perlin A two-dimensional 1000×1000 Perlin noise grid where each cell represents a 0.2×0.2 meter area. This creates a smooth, unstructured environment with evenly distributed information with values in $[-1, 1]$.

Astroid This instance modifies the **Perlin** environment by reducing the background information values (dividing each by 4) and then setting the value of all points within $2m$ of 50 randomly selected locations to 1. This process creates information hot-spots in a generally less informative map.

Grid A 5×5 grid of large $40m \times 40m$ cells, where each cell is assigned a value selected uniformly at random from $[0, 1]$. In this map, information is concentrated at the edges and corners between cells.

This analysis illustrates why our hybrid approach excels. The online planner may opt to select a more goal seeking action over an informative one or an action providing immediately good information that leads into large information deserts. In contrast, the hybrid algorithm's offline component provides an information rich path containing, for example, many asteroids or grid corners and edges, maximizing the amount of information collected over its path to the goal, leading to a higher probability of success.

VII. Conclusion

In this paper, we formalized the idea of (r, p)-information deserts, proposed an algorithm to navigate environments with information deserts using a robot with significant movement and sensing error. The proposed algorithm was tested on three types of environments, each with different information desert characteristics and found it performed well on each of

these environment types. It overcame the extreme movement and sensor uncertainties to arrive near the goal.

Despite our algorithm's success, it does have some limitations. Our global planner finds paths maximizing the expected local entropy per meter; however, it does not consider the entropy of the entire path itself. Furthermore, we assume our sensor measurements are independent across time and consider only diagonal covariance matrices which may cause poor performance with highly correlated sensor readings.

In the future, we are interested in exploring methods for our path precomputation algorithm to efficiently consider the entropy of the entire path and using this information along with the local entropy of the path to develop more robust paths. Finally, we would like to investigate alternative methods of tuning the δ of our local algorithm because our current method, while effective, may have an analytical solution.

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, Oct. 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, Feb. 2014.

[3] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *International IEEE Conference on Intelligent Transportation Systems*, Oct. 2014, pp. 392–399.

[4] B. R. Donald and J. Jennings, "Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes," Cornell University, Tech. Rep., Dec. 1991.

[5] M. Erdmann, "Using backprojections for fine motion planning with uncertainty," in *IEEE International Conference on Robotics and Automation*, vol. 2, Mar. 1985, pp. 549–554.

[6] 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.

[7] L. H. Erickson, J. Knuth, J. M. O'Kane, and S. M. LaValle, "Probabilistic localization with a blind robot," in *IEEE International Conference on Robotics and Automation*, May 2008, pp. 1821–1827.

[8] S. P. González-Sabbagh and A. Robles-Kelly, "A Survey on Underwater Computer Vision," *ACM Comput. Surv.*, vol. 55, no. 13s, pp. 268:1–268:39, Jul. 2023.

[9] J. Han, J. Park, J. Kim, and N.-s. Son, "GPS-less Coastal Navigation using Marine Radar for USV Operation," *IFAC*, vol. 49, no. 23, pp. 598–603, Jan. 2016.

[10] U. D. Hanebeck, K. Briechle, and A. Rauh, "Progressive Bayes: a new framework for nonlinear state estimation," in *SPIE Proceedings*, B. V. Dasarathy, Ed., vol. 5099. Orlando, FL: SPIE, Apr. 2003, p. 256.

[11] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for Gaussian mixture random vectors," in *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Aug. 2008, pp. 181–188.

[12] 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 *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2019.

[13] Y. Kantaros and G. J. Pappas, "Scalable Active Information Acquisition for Multi-Robot Systems," in *IEEE International Conference on Robotics and Automation*, May 2021, pp. 7987–7993.

[14] S. Klabnik and C. Nichols, *The Rust Programming Language*, 2nd ed. San Francisco, CA: No Starch Press, Feb. 2023.

[15] H. Kurniawati, "Partially Observable Markov Decision Processes (POMDPs) and Robotics," Jul. 2021, arXiv:2107.07599 [cs].

[16] S. M. LaValle, *Planning Algorithms*. Cambridge Univ. Press, 2006.

[17] A. Lazanas and J. C. Latombe, "Landmark-Based Robot Navigation," *Algorithmica*, vol. 13, no. 5, pp. 472–501, May 1995.

[18] J. S. Lewis, D. A. Feshbach, and J. M. O'Kane, "Guaranteed Coverage with a Blind Unreliable Robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2018, pp. 7383–7390.

[19] J. S. Lewis and J. M. O'Kane, "Guaranteed navigation with an unreliable blind robot," in *IEEE International Conference on Robotics and Automation*, May 2010, pp. 5519–5524.

[20] ——, "Reliable indoor navigation with an unreliable robot: Allowing temporary uncertainty for maximum mobility," in *IEEE International Conference on Robotics and Automation*, 2012.

[21] W. Luo, C. Nam, G. Kantor, and K. Sycara, "Distributed Environmental Modeling and Adaptive Sampling for Multi-Robot Sensor Coverage," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

[22] 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.

[23] Đ. Nađ, 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.

[24] 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, Sep. 2021.

[25] J. M. O'Kane and S. M. LaValle, "Localization With Limited Sensing," *IEEE Transactions on Robotics*, vol. 23, no. 4, pp. 704–716, Aug. 2007.

[26] J. Ott, M. J. Kochenderfer, and S. Boyd, "Approximate sequential optimization for informative path planning," *Robotics and Autonomous Systems*, vol. 182, p. 104814, Dec. 2024.

[27] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV Navigation and Localization: A Review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, Jan. 2014.

[28] Z. Qian, J. Fu, and J. Xiao, "Simultaneously Search and Localize Semantic Objects in Unknown Environments," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 762–11 769, Dec. 2024.

[29] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *IEEE International Conference on Robotics and Automation*, vol. 1, May 1999, pp. 35–40 vol.1.

[30] M. Roznere and A. Q. Li, "Real-time Model-based Image Color Correction for Underwater Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2019, pp. 7191–7196.

[31] M. T. J. Spaan, "Partially Observable Markov Decision Processes," in *Reinforcement Learning: State-of-the-Art*, M. Wiering and M. van Otterlo, Eds. Berlin, Heidelberg: Springer, 2012, pp. 387–414.

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

[33] Y. Wu, X. Ta, R. Xiao, Y. Wei, D. An, and D. Li, "Survey of underwater robot positioning navigation," *Applied Ocean Research*, vol. 90, p. 101845, Sep. 2019.

[34] M. Xanthidis, E. Kelasidi, and K. Alexis, "ResiPlan: Closing the Planning-Acting Loop for Safe Underwater Navigation," *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[35] Z. Yang, C. Garrett, D. Fox, T. Lozano-Pérez, and L. P. Kaelbling, "Guiding Long-Horizon Task and Motion Planning with Vision Language Models," in *IEEE International Conference on Robotics and Automation*, May 2025, pp. 16 847–16 853.

[36] B. Zhang, D. Ji, S. Liu, X. Zhu, and W. Xu, "Autonomous Underwater Vehicle navigation: A review," *Ocean Engineering*, vol. 273, p. 113861, Apr. 2023.

[37] B. Zhao, R. Skjetne, M. Blanke, and F. Dukan, "Particle Filter for Fault Diagnosis and Robust Navigation of Underwater Robot," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 6, pp. 2399–2407, Nov. 2014.

[38] D. Zheng, J. Ridderhof, P. Tsiotras, and A.-a. Agha-mohammadi, "Belief Space Planning: a Covariance Steering Approach," in *IEEE International Conference on Robotics and Automation*, 2022.

[39] K. Zheng and S. Tellex, "pomdp_py: A Framework to Build and Solve POMDP Problems," Apr. 2020, arXiv:2004.10099 [cs].