When to Localize?: A POMDP Approach

Troi Williams, Kasra Torshizi, and Pratap Tokekar

***Abstract*— Robots often localize to lower navigational errors and facilitate downstream, high-level tasks. However, a robot may want to selectively localize when localization is costly (such as with resource-constrained robots) or inefficient (for example, submersibles that need to surface), especially when navigating in environments with variable numbers of hazards such as obstacles and shipping lanes. In this study, we propose a method that helps a robot determine "when to localize" to 1) minimize such actions and 2) not exceed the probability of failure (such as surfacing within high-traffic shipping lanes). We formulate our method as a Constrained Partially Observable Markov Decision Process and use the Cost-Constrained POMCP solver to plan the robot's actions. The solver simulates failure probabilities to decide if a robot moves to its goal or localizes to prevent failure. We performed numerical experiments with multiple baselines.**

I. Introduction

Self-localization is crucial in robotics, as it seeds situational awareness and enables a robot to perform downstream tasks such as object manipulation and navigation. In many real-world tasks, an autonomous robot perceives its environment (including self-localization), plans, acts, and then repeats this cycle continuously. However, a robot may seldom want to localize when localization is costly. Let us consider examples where an autonomous underwater vehicle (AUV) navigates through high-traffic shipping lanes [1] to deliver supplies, search for critical items such as black boxes from crashed aircraft, or assess post-tsunami underwater damage (Figure 1). Typically, AUVs localize accurately by surfacing to collect GPS data because underwater dead-reckoning drifts over time, and sonar-based localization is infeasible in featureless environments. However, surfacing risks collisions with ships and increases operating time.

We explore scenarios where continuous localization may be unnecessary or not required, challenging the traditional perceive-plan-act paradigm. Instead, can a robot plan and act for extended durations, localizing only when necessary? Specifically, we aim to address the question: *when should a robot localize* to avoid exceeding the probability of failing a task? We argue that this question is non-trivial. For example, what is the appropriate state uncertainty threshold to trigger localization in a given scenario? Furthermore, should the AUV localize *preemptively* before unsafe regions? If so, how far in advance (*when*)?

We propose a move-localize behavior planner based on a constrained Partially Observable Markov Decision-making Process (POMDP) to address such questions. A constrained POMDP allows us to decouple failure probabilities and rewards and reduce ad-hoc reward tuning. Our formulation proposes the probability of failure as a cost constraint with a threshold and creates a reward strategy that penalizes localization.

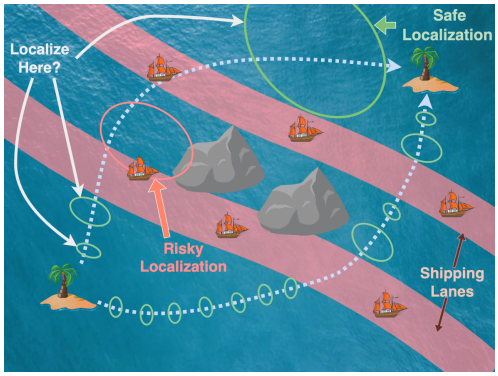

Fig. 1: An underwater vehicle follows a path to perform a task (like searching for an aircraft's black box) and must choose when to localize, which requires surfacing and poses a collision risk in shipping lanes. Continuous localization (bottom path) is inefficient yet safe. Selective localization (top path) is more efficient and desirable since the vehicle searches underwater longer. But deciding *when to localize* to avoid hazards and stay on the path is more challenging.

II. Related Work

Our work explores the idea "*when an agent should localize*", filling a research gap and potentially seeding a novel subarea. To support our assertion, we examine the questions active navigation and path planning tasks have explored and compare them to ours.

Active perception [2], [3] explores how an agent behaves to achieve one or more goals [4] and has seeded many active approaches, including active localization, active mapping, and active SLAM. Active localization [5], [6] addresses *where an agent moves* and *where it looks* to localize itself [7]–[10] or a target [11], [12]. Active mapping, or the next best view problem [4], determines *where an agent moves to map* the environment as accurately as possible (for example, [13], [14]). Finally, active SLAM determines *where an agent moves to map and localize* to reduce the uncertainty in its belief and map [4]. These methods minimize the uncertainty of the agent, its target(s), or the environment, assuming frequent or perfect localization. In contrast, we aim to minimize localization actions while avoiding task failure without guaranteeing minimal agent uncertainty.

*This research was funded in part by the National Science Foundation (NSF) Eddie Bernice Johnson INCLUDES initiative, Re-Imagining STEM Equity Utilizing Postdoc Pathways (RISE UPP), award #2217329. All authors are at the University of Maryland, College Park, MD 20742, USA. {troiw,ktorsh,tokekar}@umd.edu

Path planning seeks collision-free paths in environments with static or dynamic objects [15], [16], addressing *how an agent moves* between configurations. Path planning methods often assume localization is guaranteed or a robot localizes whenever possible [17]–[21]. Unlike these works, we explicitly plan when to localize to reach our goal.

Tasks related to path planning, such as path selection and dynamic trajectory re-planning, address *which path an agent follows* [22] or *when to deviate* [23]–[25] from a preplanned path. These tasks typically aim to minimize the state uncertainty of the agent, implying frequent localization [22]. In contrast, we aim to mitigate failure, which does not inherently require frequent localization.

Contributions. We introduce an active localization method that decides when agents should localize, contributing a) the first step towards novel methods that decide when an agent should localize, b) a hierarchical planning method based on a constrained POMDP, and c) an open-source implementation [26] of the Cost-Constrained, Partially Observable Monte-Carlo Planning (CC-POMCP) solver, a constrained POMDP solver.

III. Problem Statement

Our objective is for the robot to safely reach the goal. We seek a policy $\pi = \{a_1, \ldots, a_{N_{actions}}\}$ that 1) minimizes localization events and 2) maintains the probability of failure below a threshold $\hat{c}$ while navigating the path $\boldsymbol{X} = \{x_1, \ldots, x_{N_{points}}\}$. Here, failure includes collisions during localization or entering an unsafe region. This objective is:

$$\begin{aligned} \pi^* &= \arg\min_{\pi \in \Pi} \sum_{t=0}^{T} a_t = \{\text{localize}\} \\ \text{s.t. } & Pr(\text{failure} | x_{1:N_{points}}, a_{1:T}, b(s_0)) \leq \hat{c}, \end{aligned} \tag{1}$$

where T is the total number of actions, $a_{1:T}$ is the sequence of move and localize actions, and $b(s_0)$ is the initial belief at the start of the path. Finally, Π denotes the set of all possible action sequences over T timesteps.

We address objective (1) using a constrained POMDP. We chose this approach for two reasons: POMDPs have addressed many sequential decision problems (for example, see [27]), and constrained POMDPs allow us to separate rewards and costs, simplifying the reward model. We use a constrained POMDP solver to find an action sequence that balances feed-forward ("rarely" localize) and feedback (localize "often") actions, depending on the failure probability.

IV. When to Localize? A POMDP Formulation

We design our planner's behavior using the CC-POMCP solver [26], employing a hierarchical planning strategy like [28]. This strategy separates high-level (whether to move or localize) and low-level planning (how to move or localize). Algorithm 1 provides a general description of how our high- and low-level planners control the robot and update the belief as the robot performs actions. The remaining text describes Algorithm 1 in the context of our proposed AUV setup.

High-Level Planner (HLP). The HLP action space is defined as $\mathbb{A} = \{move,\ localize\}$. If the agent chooses to *move*, we use the transition model to propagate the belief to the next waypoint, a failure state, or a goal state. Then we skip the observation step. If the agent localizes, it surfaces to collect GPS observations, updates its belief, and submerges without moving horizontally. Since surfacing poses a collision risk with boats, the agent will enter a failure state if it localizes within a shipping lane. Finally, regardless of the action, the agent receives a reward and a cost if using CC-POMCP.

Algorithm 1 SafeNav Active Localization Algorithm

```
Initialize High-level planner (HLP) with initial belief b(s_0)
and actions a_t ∈ {move, localize}, and Low-level planner
(LLP) with path x_{1:N_points}.
t = 0
while not in a terminal state do
    HLP performs roll-out using b(s_t) to select a_t
    if a_t = move then
        LLP computes motion command u_t
        LLP truncates path, removing current waypoint
        Robot executes u_t
        HLP uses u_t to compute b(s_{t+1})
    end if
    if a_t = localize then
        Robot receives observation o_state
        HLP updates b(s_t) using o_state, producing b(s_{t+1})
        LLP uses b(s_{t+1}) to plan hazard-free path to goal
    end if
    HLP receives reward r_t and, if applicable, cost c_t
    t = t + 1
end while
```

Lower-Level Planner (LLP). The LLP determines the sequence of open-loop commands the robot will execute if the action is "move." If "localize" is chosen, the LLP makes the AUV surface, update its belief using GPS data, and submerge using no horizontal motion.

State and Measurement Spaces. We define these spaces as the agent's position within a grid $\mathbb{Z} \subseteq \mathbb{R}^2$. The goal state is located at the end of the robot's pre-defined path. Furthermore, we define the failure states as any location where the agent collides with an obstacle. In the context of an AUV, failure states include collisions with underwater obstacles or trying to surface within shipping lanes. We provide specific details of these failure states in Section V.

Reward and Cost Models. The agent receives a reward and cost after executing an action. Our reward model reflects the objective in (1): $r_{total} = r_{goal} + r_{move} + r_{local} + r_{fail}$. Here, r_{goal} is the reward for reaching the goal, and r_{move}, r_{local}, and r_{fail} are the penalties for moving, localizing, or entering the failure state, respectively.

The cost model encodes the constraint in (1), setting a cost constraint threshold $\hat{c}$ for the maximum number of particles allowed to enter a failure region. We say $\hat{c} = p \times N_p$, where N_p is the number of belief particles and p is the maximum percentage of particles allowed in the failure region.

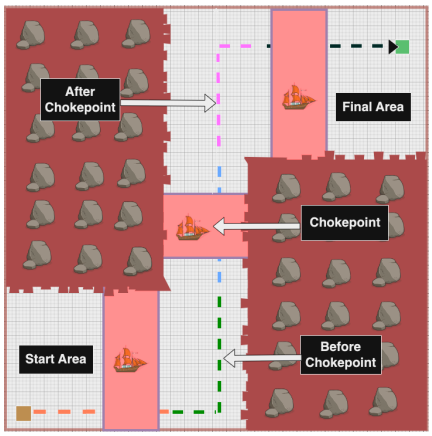

Fig. 2: This figure illustrates our training environment (ENV-TRAINING), a 30 × 30 2D grid world we used to determine the parameters for our online algorithms (shown in Table I). The salmon-colored rectangles denote shipping lanes where boats sail while the AUV navigates underwater.

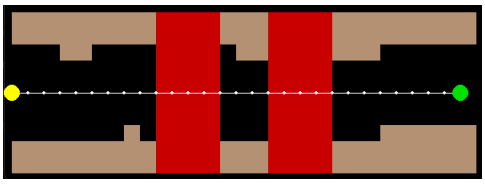

Fig. 3: This figure depicts a screenshot of ENV-TUNNEL, one of our evaluation environments. The environment contains obstacles in brown, localize hazards (such as shipping lanes) in red, the (yellow) start and (green) goal positions, and the pre-defined path the AUV should follow.

V. Numerical Evaluations

The following subsections describe our experimental setup (Section V-A) and results (Section V-B). Experiments ran until the AUV reached a goal or failure state, and the results were based on 100 runs for each HLP.

A. Setup

High-Level Planners (HLPs). We used three types of HLPs in our experiments. The first type was *offline*, *uniformed*, *cost-unaware* static planners. Here, uninformed means the planners did not use the belief to determine the next action. These static policies (SP) were (M1x,L) (that is, move once, then localize), (M2x,L), and (M3x,L), representing traditional action sequences with varying localization intervals. The second type was POMCP, an *informed*, *online*, *cost-unaware* planner. We used POMCP to assess the advantage of employing a cost constraint. Finally, the third type was CC-POMCP, which we used to implement our proposed algorithm. CC-POMCP is an *informed*, *online*, *cost-aware* planner.

POMCP and CC-POMCP Details. We used the POMCP algorithm from the pomdp_py Python library [29] and implemented the CC-POMCP algorithm[1] using the same library.

[1]Code: https://github.com/troiwill/when-to-localize-pomdp

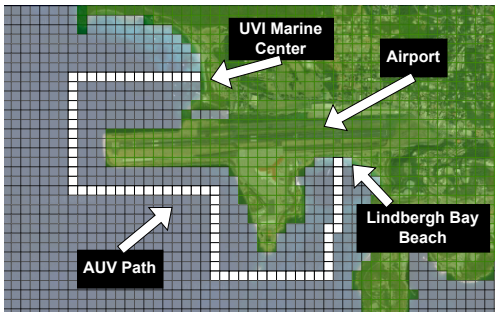

Fig. 4: This figure illustrates our real-world evaluation environment called ENV-STT. The image shows a grid overlaying a Google Maps screenshot of the southwestern portion of St. Thomas, U.S. Virgin Islands. The green cells represent obstacles (land masses), the grey cells denote open water, and the white cells represent the AUV's pre-defined path. The AUV starts at the University of the Virgin Islands' (UVI) Marine Center and ends at Lindbergh Bay Beach.

Both algorithms used the same parameters in all experiments, except for the parameters shown in Table I. These parameters were derived through a parameter search using the ENV-TRAINING environment. Furthermore, whenever the AUV moved to another grid cell (via a motion command), the AUV had a 94% chance of moving to the appropriate cell and a 6% chance of over- or undershooting the cell. We also added noise during particle reinvigoration (the update phase) for both algorithms, which helped mitigate particle deprivation. Finally, we used the GNU Parallel software [30] to run our online planning experiments in parallel due to each algorithm's high runtime during planning.

Low-Level Planner (LLP). As we mentioned in Section IV, the LLP determines the low-level motion commands that the robot should perform to go to the next waypoint based on the high-level action. Thus, if the high-level action was move, the LLP determines the next command (that is, up, down, left, or right) using the belief's mean. If the high-level action was localize, the LLP determines if the AUV is still on the path using the belief's mean. If the AUV is not on the path, the LLP finds the nearest waypoint along the path and computes a collision-free path to this waypoint using breath-first search.

Environments. We created one environment (ENV-TRAINING) to find parameters for POMCP and CC-POMCP (Figure 2) and two environments (ENV-TUNNEL and ENV-STT) for evaluation (Figures 3 and 4). The AUV knew the locations of the failure states (obstacles and shipping lanes) and goal states. Finally, we set the initial state of all particles to the start state.

B. Results

We compared the HLPs in terms of the physical failure rates, number of localization actions, and cumulative collisions. The SP results represent an ideal scenario, with no path deviations or collisions, serving as a lower-bound performance benchmark for each SP.

Planner	r_{goal}	r_{local}	r_{fail}	# Particles	α_n	γ	κ	Tree Depth	# Sims	Max % of Particles Failed
POMCP	100	-5	-10	1000	N/A	0.999	150	8	2000	N/A
CC-POMCP	”	-3	-100	”	0.001	0.9	200	”	”	10%

TABLE I: Online Planner Parameters. Here, α_n is the step size, γ is the discount factor, and κ is the explore constant.

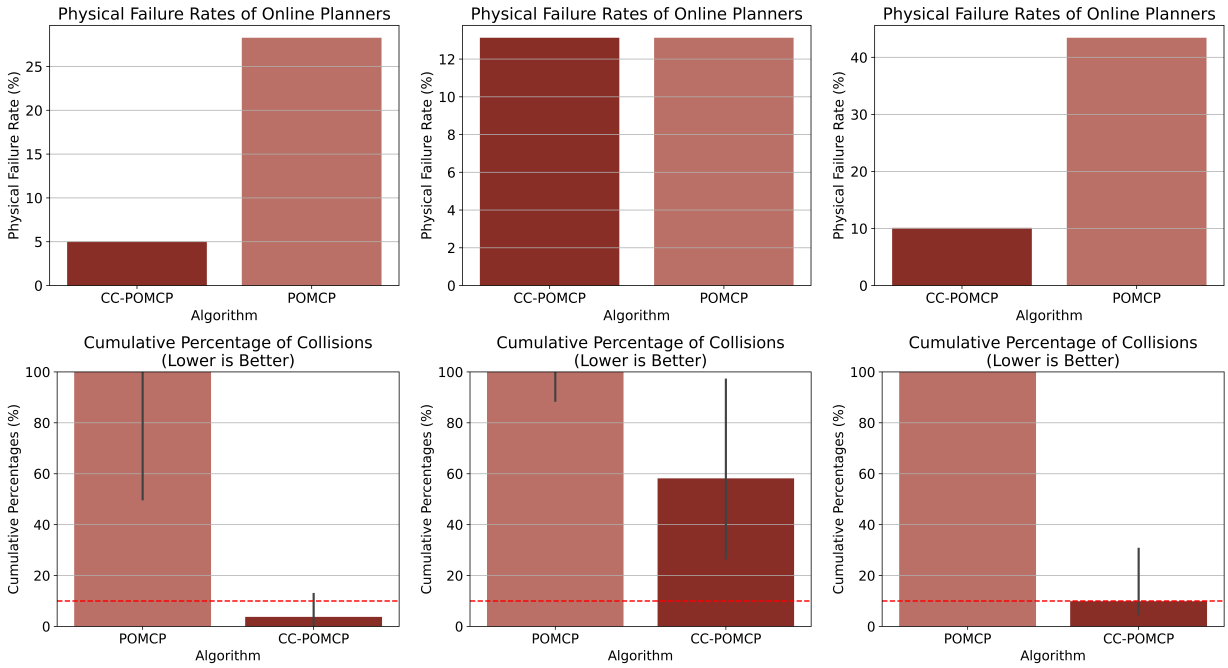

Fig. 5: **The Physical Failure Rates (top) and Cumulative Percentage of Collisions (bottom).** This figure depicts the results for the ENV-TRAINING (left), ENV-TUNNEL (middle), and ENV-STT (right) environments. Here, the failure rates refer to the number of times the robot left the map or collided with an obstacle while moving or localizing. Each graph shows the mean and standard deviation for each algorithm. Due to POMCP's poor performance, we clipped the cumulative percentages (bottom graphs) to 100%. Finally, the red dashed horizontal lines in the bottom graphs show the collision threshold of 10%.

Physical Failure Rates. Since the SPs do not consider the environment and belief when planning, they will fail due to collisions within shipping lanes in the ENV-TRAINING and ENV-TUNNEL environments (Figures 2 and 3). However, since POMCP and CC-POMCP account for the shipping lanes, no online planner failed due to localizing within a shipping lane. This means that POMCP and CC-POMCP only failed (physically) due to underwater collisions or leaving the map.

Generally, POMCP failed four times or more than CC-POMCP (Figure 5 top). We believe the disparity is due to the difference in the parameters (Table I). For example, POMCP has a small failure penalty, which may have caused POMCP to execute more risky policies (such as *never* localize) and ultimately collide with obstacles. Unfortunately, these were the only parameters that enabled POMCP to reach the goal for each environment.

In the ENV-TUNNEL environment, both algorithms had similar failure rates (Figure 5). We believe the rates are similar due to the difficulty of the environment. Figure 6 (bottom left) shows that POMCP never localized in ENV-TUNNEL, while CC-POMCP localized frequently before the first and after the last shipping lanes. Due to the arrangement of the shipping lanes, both algorithms incurred high uncertainties due to executing 12 or more consecutive motion commands without localizing, increasing the probability of the AUV colliding with underwater obstacles.

Localization Actions. Ideally, the number of localization actions for each SP planner represents the *lowest* number of times a robot would localize in a given environment. Generally, CC-POMCP varied in the number of localization actions executed based on the environment (Figure 6). For example, although the average localization counts were similar in ENV-TRAINING (Figure 6 top), their variances were lower when we performed an ablation study by removing the shipping lanes (Figure 6 top left). In another example, CC-POMCP avoids localizing between the shipping lanes in ENV-TUNNEL (Figure 6 bottom left). Instead, CC-POMCP localized prior to the first and after the second shipping lanes. We believe CC-POMCP did this because of the small gap between the shipping lanes and the high belief uncertainty due to multiple open-loop motion commands. Finally, in the ENV-STT environment, CC-POMCP typically localized along the southern part of the airport and relatively more within Lindbergh Bay (Figure 6 bottom right). We believe CC-POMCP does not localize near the Marine Center or along the western part of the airport because a) there are less number of obstacles near the AUV's path and b) the AUV's

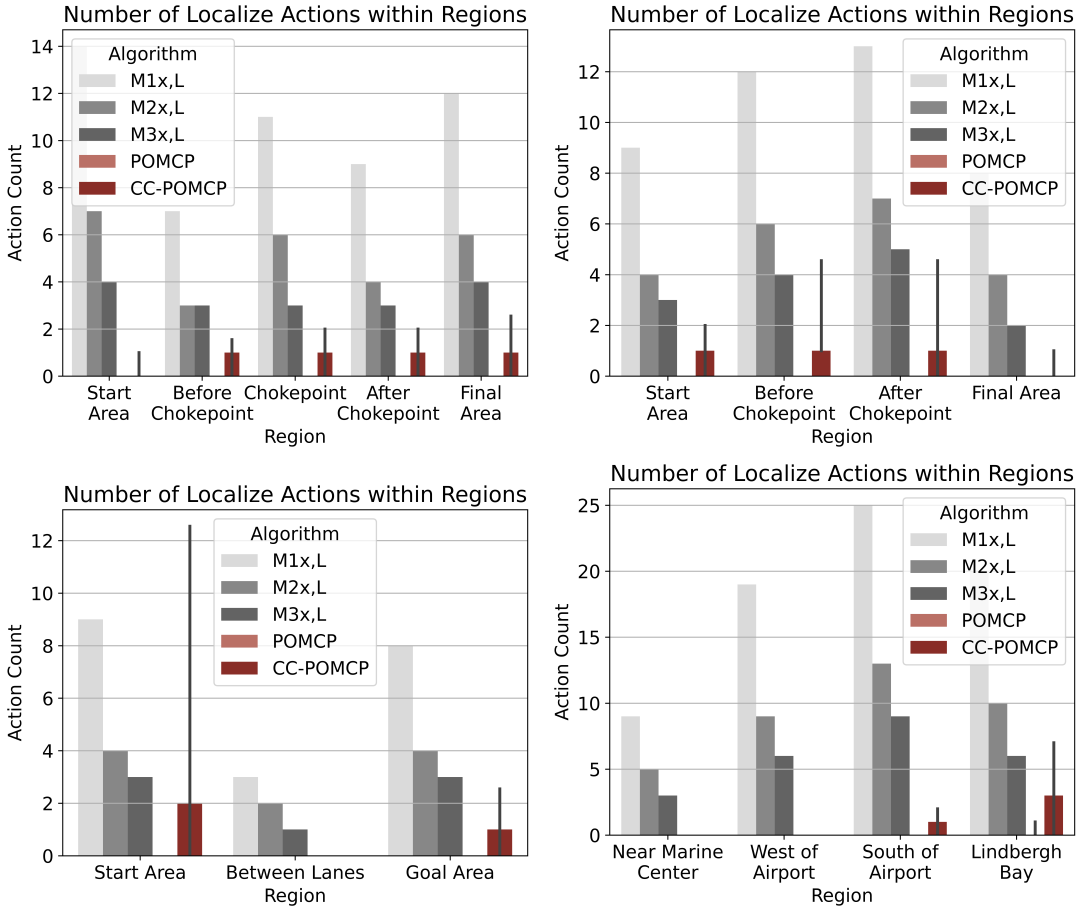

Fig. 6: **Number of Localization Actions for all Algorithms in each Environment's Regions.** The top graphs show the ENV-TRAINING environment results without shipping lanes (left) and with shipping lanes (right). Meanwhile, the bottom graphs show the ENV-TUNNEL (left) and ENV-STT environments (right). The SPs represent the ideal, lower-bound localization counts for SPs. The numbers for POMCP and CC-POMCP represent their empirical values.

uncertainty always starts small in the beginning areas. Such observations show the benefit of online planning versus static planning. That is, we localize more often to avoid collisions in the future and significantly less when collisions are either not an issue or highly probable.

Cumulative Collisions. CC-POMCP outperformed POMCP in terms of the cumulative collisions. In all three environments, POMCP exceeded the collision (cost) threshold by a significant amount (Figure 5). This result is unsurprising because POMCP is a cost-unaware algorithm. Furthermore, the failure penalty, which influences how often the AUV localizes, may be too small in magnitude. As a result, one must tune the failure penalty in an indirect, ad hoc manner to prevent POMCP from exceeding the collision threshold.

On the other hand, CC-POMCP did not exceed the collision threshold in the ENV-TRAINING (left) and ENV-STT (right) environments on average. However, their variances illustrate that the algorithm did exceed the threshold at least once for both environments. Finally, we note CC-POMCP's average cumulative percentage does exceed the threshold in the ENV-TUNNEL. Again, we believe the poor performance is due the difficulty of the environment. Here, the AUV cannot localize for an extended period, causing more particles to collide with underwater obstacles. These observations imply that a cost-aware planner is more beneficial than a cost-unaware planner.

VI. Conclusion

We developed a behavior planner for robots to determine *when to localize*, aiming to threshold probabilities of failure (such as collisions). This approach differs from traditional robotics questions (Section I). Our results showed two key findings. First, online planners allow robots to localize as needed, potentially achieving higher rewards, especially when localization is costly. Second, cost-aware planners provide a precise mechanism for setting performance thresholds like failure probabilities for collisions, reducing arbitrary reward tuning.

One limitation of our work is that CC-POMCP has a high compute time and needs for well-defined transition and observation models. We addressed these limitations by replacing CC-POMCP with a reinforcement learning approach that employs particle filtering and a recurrent Soft Actor-Critic network [33].

Future work will involve conducting more extensive experiments, including real-world experiments, and relaxing assumptions. For example, we will assume heteroscedastic measurement noise for more realistic scenarios, as in [12], [31], [32].

REFERENCES

[1] A. A. Pereira, J. Binney, G. A. Hollinger, and G. S. Sukhatme, "Risk-aware path planning for autonomous underwater vehicles using predictive ocean models," *Journal of Field Robotics*, vol. 30, no. 5, pp. 741–762, 2013.

[2] C. Cowan and P. Kovesi, "Automatic sensor placement from vision task requirements," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 407–416, 1988.

[3] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, pp. 177–196, 2018.

[4] J. A. Placed, J. Strader, H. Carrillo, N. Atanasov, V. Indelman, L. Carlone, and J. A. Castellanos, "A Survey on Active Simultaneous Localization and Mapping: State of the Art and New Frontiers," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 1686–1705, 2023.

[5] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *Proceedings of the Fifteenth International Joint Conference on Artifical Intelligence - Volume 2*, ser. IJCAI'97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 1346–1352.

[6] G. Borghi and V. Caglioti, "Minimum uncertainty explorations in the self-localization of mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 902–911, 1998.

[7] C. Mostegel, A. Wendel, and H. Bischof, "Active monocular localization: Towards autonomous monocular exploration for multirotor MAVs," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3848–3855.

[8] K. Otsu, A.-A. Agha-Mohammadi, and M. Paton, "Where to Look? Predictive Perception With Applications to Planetary Exploration," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 635–642, 2018.

[9] S. K. Gottipati, K. Seo, D. Bhatt, V. Mai, K. Murthy, and L. Paull, "Deep Active Localization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4394–4401, 2019.

[10] J. Strader, K. Otsu, and A.-a. Agha-mohammadi, "Perception-aware autonomous mast motion planning for planetary exploration rovers," *Journal of Field Robotics*, vol. 37, no. 5, pp. 812–829, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21925

[11] R. Tallamraju, N. Saini, E. Bonetto, M. Pabst, Y. T. Liu, M. J. Black, and A. Ahmad, "AirCapRL: Autonomous Aerial Human Motion Capture Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6678–6685, 2020.

[12] T. Williams, P.-L. Chen, S. Bhogavilli, V. Sanjay, and P. Tokekar, "Where Am I Now? Dynamically Finding Optimal Sensor States to Minimize Localization Uncertainty for a Perception-Denied Rover," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 207–213.

[13] T. Sasaki, K. Otsu, R. Thakker, S. Haesaert, and A.-a. Agha-mohammadi, "Where to Map? Iterative Rover-Copter Path Planning for Mars Exploration," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2123–2130, 2020.

[14] H. Dhami, V. D. Sharma, and P. Tokekar, "Pred-NBV: Prediction-Guided Next-Best-View Planning for 3D Object Reconstruction," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7149–7154.

[15] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015300671

[16] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A Survey of Path Planning Algorithms for Mobile Robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021. [Online]. Available: https://www.mdpi.com/2624-8921/3/3/27

[17] F. Tan, J. Yang, J. Huang, T. Jia, W. Chen, and J. Wang, "A navigation system for family indoor monitor mobile robot," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 5978–5983.

[18] E. Trulls, A. Corominas Murtra, J. Pérez-Ibarz, G. Ferrer, D. Vasquez, J. M. Mirats-Tur, and A. Sanfeliu, "Autonomous navigation for mobile service robots in urban pedestrian environments," *Journal of Field Robotics*, vol. 28, no. 3, pp. 329–354, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20386

[19] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1192–1198.

[20] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS '12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, p. 147–154.

[21] A. Corominas Murtra, E. Trulls, O. Sandoval, J. Pérez-Ibarz, D. Vasquez, J. M. Mirats-Tur, M. Ferrer, and A. Sanfeliu, "Autonomous navigation for urban service mobile robots," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4141–4146.

[22] A. akbar 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, pp. 268 – 304, 2014.

[23] A.-a. Agha-mohammadi, S. Agarwal, A. Mahadevan, S. Chakravorty, D. Tomkins, J. Denny, and N. M. Amato, "Robust online belief space planning in changing environments: Application to physical mobile robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 149–156.

[24] A. akbar 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, pp. 1195–1214, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52934198

[25] J. van den Berg, S. Patil, and R. Alterovitz, *Motion Planning Under Uncertainty Using Differential Dynamic Programming in Belief Space*. Cham: Springer International Publishing, 2017, pp. 473–490. [Online]. Available: https://doi.org/10.1007/978-3-319-29363-9_27

[26] J. Lee, G.-h. Kim, P. Poupart, and K.-E. Kim, "Monte-Carlo Tree Search for Constrained POMDPs," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/54c3d58c5efcf59ddeb7486b7061ea5a-Paper.pdf

[27] M. Lauri, D. Hsu, and J. Pajarinen, "Partially Observable Markov Decision Processes in Robotics: A Survey," *IEEE Transactions on Robotics*, vol. 39, no. 1, p. 21–40, Feb. 2023. [Online]. Available: http://dx.doi.org/10.1109/TRO.2022.3200138

[28] W. Sun, S. Patil, and R. Alterovitz, "High-Frequency Replanning Under Uncertainty Using Parallel Sampling-Based Motion Planning," *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[29] K. Zheng and S. Tellex, "pomdp_py: A Framework to Build and Solve POMDP Problems," in *ICAPS 2020 Workshop on Planning and Robotics (PlanRob)*, 2020, arxiv link: "https://arxiv.org/pdf/2004.10099.pdf". [Online]. Available: https://icaps20subpages.icaps-conference.org/wp-content/uploads/2020/10/14-PlanRob_2020_paper_3.pdf

[30] O. Tange, *GNU Parallel 2018*. Ole Tange, Mar 2018.

[31] T. Williams and Y. Sun, "Learning State-Dependent, Sensor Measurement Models for Localization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3090–3097.

[32] ——, "Learning State-Dependent Sensor Measurement Models with Limited Sensor Measurements," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 86–93.

[33] C. L. Shek, K. Torshizi, T. Williams, and P. Tokekar, "When to localize? a risk-constrained reinforcement learning approach," 2024. [Online]. Available: https://arxiv.org/abs/2411.02788