

When to Localize?

A Risk-Constrained Reinforcement Learning Approach

Chak Lam Shek*, Kasra Torshizi*, Troi Williams, and Pratap Tokekar

Abstract—In a standard navigation pipeline, a robot localizes at every time step to lower navigational errors. However, in some scenarios, a robot needs to selectively localize when it is expensive to obtain observations. For example, an underwater robot surfacing to localize too often hinders it from searching for critical items underwater, such as black boxes from crashed aircraft. On the other hand, if the robot never localizes, poor state estimates cause failure to find the items due to inadvertently leaving the search area or entering hazardous, restricted areas. Motivated by these scenarios, we investigate approaches to help a robot determine “when to localize?” We formulate this as a bi-criteria optimization problem: minimize the number of localization actions while ensuring the probability of failure (due to collision or not reaching a desired goal) remains bounded. In recent work, we showed how to formulate this active localization problem as a constrained Partially Observable Markov Decision Process (POMDP), which was solved using an online POMDP solver. However, this approach is too slow and requires full knowledge of the robot transition and observation models. In this paper, we present RISKRL, a constrained Reinforcement Learning (RL) framework that overcomes these limitations. RISKRL uses particle filtering and recurrent Soft Actor-Critic network to learn a policy that minimizes the number of localizations while ensuring the probability of failure constraint is met. Our numerical experiments show that RISKRL learns a robust policy that outperforms the baseline by at least 13% while also generalizing to unseen environments.

I. INTRODUCTION

In robotics, self-localization is crucial because it enhances navigation accuracy, and situational awareness, and enables complex tasks. Typically, an autonomous robot perceives its environment and self-localizes, plans its subsequent actions, acts upon its plan phases, and repeats the cycle. However, sometimes, a robot may want to localize seldom when it is not advantageous. For example, underwater robots need to surface to localize in underwater rescue and recovery missions. Surfacing to localize too often may hinder an underwater robot from searching for critical underwater items such as black boxes from crashed aircraft. On the other hand, if the robot never localizes, it will accumulate large amounts of drift [1], which may prevent it from finding the items due to inadvertently leaving the search area or entering hazardous, restricted areas. Therefore, in such scenarios, robots must balance prolonged actions that achieve mission

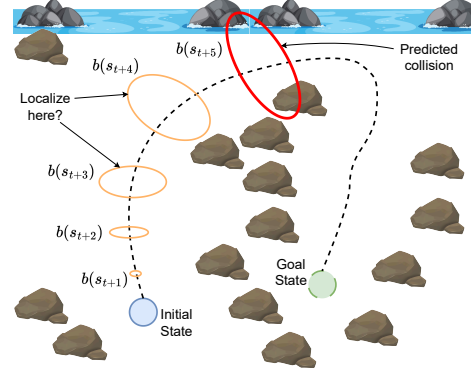


Fig. 1: Motivating example. Consider a robot that may want to seldom localize (e.g., due to resource constraints) while traveling along a path (black dashed line). Despite obstacles (rocks and water), the autonomous robot can execute a series of open-loop motions for some period. However, as the dead reckoning uncertainty (gold and red ellipses) grows, the probability of failure (such as collision) may become too large. As such the robot must localize at some point to avoid failures. Thus, **our question** is: *When should the robot localize to reduce failure probabilities?*

objectives (such as searching for critical items underwater) with localizing to improve navigation accuracy.

We explored such scenarios in our recent work [2]. Our central question was: *how can a robot plan and act for long horizons safely and only localize when necessary?* (Figure 1 discusses a general scenario to this question). We emphasize that such a question is not trivial because we have two competing objectives. The first objective is *localize often* to maximize mission safety and performance, where we can ensure the vehicle remains within the search area and out of hazardous zones. On the other hand, the second objective is to *localize infrequently* to minimize the number of times the vehicle must deviate from its mission, which in turn can reduce mission time. These two objectives are challenging to optimize via one objective, as shown with our POMCP baseline in [2]. Therefore, we addressed the question by formulating it as a constrained Partially Observable Markov Decision-making Process (POMDP), where our objective was to minimize the number of localization actions while not exceeding a given probability of failure due to collisions. Then we employed CC-POMCP [3], a cost-constrained POMDP solver, to find policies that determine when the robot should move along a given path *or* localize.

*C. Shek and K. Torshizi contributed equally and are listed alphabetically.

This research was funded in part by the National Science Foundation (NSF) Eddie Bernice Johnson INCLUDES initiative, Re-Imaging STEM Equity Utilizing Postdoc Pathways (RISE UPP), award #2217329. All authors are at the University of Maryland, College Park, MD 20742, USA. {cshek1, ktorsh, troiw, tokekar}@umd.edu

Although our prior approach produced policies that reached the goal and outperformed baselines, the approach had limitations. First, CC-POMCP was computationally expensive, requiring over 20 minutes of inference to navigate a path of 55 waypoints¹. Second, CC-POMCP requires a well-defined model of the environment, including the robot’s transition (motion) and observation models. Requiring such models may be problematic in unknown or dynamic real-world environments where it may be challenging to obtain accurate models. Finally, in our prior experiments, CC-POMCP failed to reach the goal when we increased the amount of transition noise, which limited successful experiments to low transition noises.

We propose a novel approach termed RISKRL that employs constrained Reinforcement Learning (RL) [4] and Particle Filters (PF) to overcome these limitations. Our new approach has multiple advantages over our prior work. First, although our new approach has a longer, single training time, it infers quicker during deployment, enabling real-time planning. The second was reducing the need for accurate transition and observation models of the environment. Like cost-constrained POMDPs in our prior work [2], constrained RL allows us to separate the value function and risk. In this formulation, the risk is modeled as a probability constraint, allowing us to design policies that minimize the failure probability while ensuring the robot remains within acceptable risk levels. This formulation provides greater control over the failure rate by explicitly incorporating risk constraints into the decision-making process. Furthermore, we use a PF to maintain the robot’s belief as the robot executes noisy motion commands and receives noisy measurements from the environment. We also use the PF to compute the observation for the RL robot.

We perform numerical experiments to compare RISKRL with several baselines, including standard RL (BASERL), CC-POMCP, and heuristic policies. Our main finding is that when deployed in the environment that the robot is trained in RISKRL outperforms the BASERL and CC-POMCP baselines by at least 13% in terms of the success rate while also being the only algorithm that satisfies the risk constraint. While environment-specific heuristics can match the performance, we find that RISKRL also generalizes to unseen environments, which the heuristics cannot.

II. RELATED WORK

This paper explores minimizing localization actions while not exceeding pre-defined failure probabilities. In the following subsections, we position our method within the active localization and constrained RL literature.

A. Active Localization

Active perception [5], [6] encompasses various approaches, including active localization [7], [8], active mapping [9]–[11], and active SLAM [9]. These approaches focus on finding optimal robot trajectories and observations to

achieve mission goals. Of these approaches, our current approach falls under active localization. Typically, active localization methods address *where* a robot moves and looks to localize itself [12]–[15] or a target [16], [17]. Thus, these active localization approaches generally differ from our problem because we seek to determine *when* a robot localizes. However, one exception is our prior work [2], which proposes a preceding approach to the one in this paper. This approach improves upon [2], where we now model the probability of failure in terms of risk, reduce the inference time significantly, and relax the need for well-defined noise models.

B. Constrained Reinforcement Learning

Model-free reinforcement learning promises a more scalable and general approach to solving the active localization problem since it requires less domain knowledge [18], [19]. However, many prior works applying RL to solving POMDPs, even without constraints, have gotten poorer results compared to more specialized methods [20]. A recent architecture [21] utilizing a Soft-Actor Critic (SAC) with two separate recurrent networks for both the actor and value functions has shown promise to surpass more specialized methods in select examples. Since recurrent networks also act on the history of observations, they can handle partial observability. Our RISKRL approach is based on this twin recurrent network SAC architecture [21]. However, unlike [21] we also seek to deal with risk constraints.

There is a separate line of work on constrained RL in the fully observable setting [22]. Constrained RL extends model-free RL by incorporating real-world limitations, such as safety, budget, or resource constraints, into the learning process [22]. The primary challenge in constrained RL is balancing the trade-off between maximizing cumulative rewards and satisfying these secondary constraints. A commonly used approach to tackle this problem is the primal-dual optimization method [23], which iteratively adjusts both the policy and constraint parameters. Various techniques are used to simplify and solve these problems, such as transforming the constraints into convex functions [24] or employing stochastic approximation methods to handle probabilistic constraints [4]. However, these prior works on constrained RL have only focused on the fully observable setting. In this paper, we build on these two lines of work and present RISKRL, which handles both partial observability as well as chance constraints.

III. PROBLEM STATEMENT

This paper solves the same active localization problem as in our prior work [2]. That is, a robot aims to selectively localize while navigating along a path to a pre-defined goal. When the robot believes it is opportune to localize, it uses its sensors to obtain noisy observations of its pose to mitigate failures such as collisions. Thus, we aim to generate a move-localize policy that 1) minimizes localization events and 2) avoids exceeding a failure probability threshold \hat{c} . For

¹We obtained these run times from performing the experiments on a computing cluster.

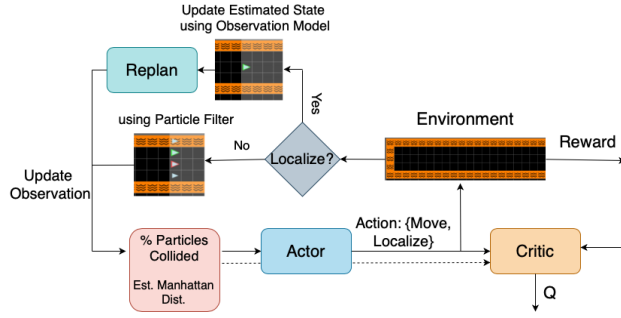


Fig. 2: This flowchart depicts a decision-making process for a robot interacting with an environment. It integrates a particle filter for state estimation and reinforcement learning components (actor-critic architecture) for decision-making.

the reader's convenience, we include the original objective function from [2]:

$$\pi^* = \arg \min_{\pi \in \Pi} \sum_{t=0}^{N_{actions}} a_t = \{\text{localize}\} \quad (1)$$

s.t. $Pr(\text{failure} | x_{1:N_{points}}, a_{1:N_{actions}}, b(s_0)) \leq \hat{c}$,

where $N_{actions}$ is the total number of actions, $a_{1:N_{actions}}$ 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 $N_{actions}$ timesteps.

IV. RISKRL ACTIVE LOCALIZATION ALGORITHM

In this section, we present our RISKRL algorithm for solving the active localization problem given in Equation 1. Within the context of RISKRL, the following subsections discuss our active localization framework (that is, the PF, low-level planner, and high-level RL planner) in Section IV-A and provide detailed descriptions of the RL algorithm in the remaining subsections.

A. Overview

Algorithm 1 RISKRL Active Localization Algorithm

```

Initialize RL planner state  $s_t$ , initial PF belief  $b(s_0)$ , path  $x_{1:N_{points}}$ 
while not in a terminal state do
  RL planner uses  $s_t$  to select  $a_t$  (move/localize)
  if  $a_t = \text{move}$  then
    Low-level planner computes motion command  $u_t$ 
    Low-level planner truncates path  $x_{2:N_{points}}$ 
    Robot executes  $u_t$ 
    PF propagates belief  $b(s_t)$  using  $u_t$ 
  end if
  if  $a_t = \text{localize}$  then
    Robot receives observation  $o_{state}$ 
    PF updates belief  $b(s_t)$  using  $o_{state}$ 
    Low-level planner replans hazard-free path to goal
  end if
  Robot receives reward  $r_t$ 
  PF computes  $o_{planner}$  for the next time step
end while

```

We now discuss our RISKRL active localization framework, which is depicted in Figure 2 and described in Algorithm 1. Our framework comprises three main components: an RL high-level planner, a PF, and a low-level planner. The high-level planner chooses which action (move or localize) the agent performs given an input state s_t . Next, we use the PF to maintain a belief of the agent's state whenever the agent selects either action and computes a state vector for the RL planner. Finally, a low-level planner determines how the agent should move (that is, move forward, turn left, or turn right) if move is chosen or recomputes hazard-free paths if localize is chosen.

B. High-Level Policy

The output from the PF is fed as input to the high-level policy. The goal of the high-level policy is to choose between two actions: $\mathbb{A}_{RLP} = \{\text{move}, \text{localize}\}$. If the policy chooses to move, then we execute the next action given by the low-level planner without taking an observation. As described earlier, we use a constrained RL policy that minimizes the number of localization actions while ensuring the risk constraint is met.

The state input s_t given to the RL policy is derived from the output of the PF. Specifically, s_t is a 2D vector containing the collision probability and the Manhattan distance from the robot to the goal. We compute the collision probability by counting the number of particles in collision at the current time step. We chose this observation representation so that our agent generalizes to unseen environments.

The reward $r_t = -1$ if the robot chooses the localize action and 0 otherwise. We can use standard RL to minimize the number of localization actions by maximizing this reward function. However, naive optimization will violate the constraints in (1). The constraint probability is difficult to estimate because it requires interaction with the environment and varies based on the policy being used, making it challenging to establish a clear relationship between the policy and the constraint. We follow the relaxation approach outlined in [4], converting the probabilistic constraint in a Chance-Constrained POMDP into a cumulative constraint. Specifically, we reformulate the optimization problem as follows. Our goal is to maximize the expected cumulative reward $V(\theta)$, defined by:

$$\max_{\theta \in \mathbb{R}^d} V(\theta) \triangleq \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \mid \pi_{\theta} \right] \quad (2)$$

where θ denotes the parameters of the policy π_{θ} , and $V(\theta)$ represents the expected reward over time. We then impose a cumulative constraint:

$$U_{\theta} : \sum_{t=0}^T \gamma^t (1 - \Pr(\text{failure} \mid x_{1:N}, a_{1:t}, b(s_0))) \geq c \quad (3)$$

Here, U_{θ} is the accumulated discounted probability of failure, and $c = \frac{(1-\hat{c})\gamma^T(1-\gamma)}{(1-\gamma)}$ represents the risk-adjusted threshold. This formulation simplifies the original problem by approximating the probabilistic constraint, thus allowing the

constraint to be evaluated based on the data generated from the rollout trajectory.

To incorporate the approximation of the probabilistic constraint into the reward function, we adjust the reward to account for the constraint violation. The new reward function is formulated as follows:

$$\hat{r}(s_t, a_t) = r(s_t, a_t) + \lambda (I(s_t \notin \text{failure}) - c(1 - \gamma)), \quad (4)$$

where $\hat{r}(s_t, a_t)$ represents the adjusted reward function that is -1 if the action is *localize* and 0 otherwise, λ is a penalty coefficient, $I(s_t \notin \text{failure})$ is an indicator function that is 1 if the state s_t is not in the failure set and 0 otherwise, and $c(1 - \gamma)$ is the threshold term derived from the relaxed constraint. The difference between the indicator function's value and the threshold can estimate the probability that the constraints are satisfied.

The algorithm [4] shown below employs the primal-dual method to optimize the expected reward while satisfying constraints. The primal component focuses on maximizing the reward function as defined in Equation (4), while the dual component adjusts the λ values to control the risk levels associated with these constraints. It iteratively updates the policy parameters and dual variables by simulating trajectories and estimating gradients. The policy can be updated by computing the policy gradient defined by the following equation:

$$\nabla_{\theta} L(\theta_k, \lambda_k) = \hat{r}(s_t, a_t) \nabla_{\theta} \log \pi_{\theta_k}(a_0 | s_0) \quad (5)$$

where $\nabla_{\theta} \log \pi_{\theta_k}(a_0 | s_0)$ is the gradient of the log-probability of the policy π_{θ_k} selecting action a_0 given state s_0 .

Algorithm 2 Primal-Dual Optimization [4]

```

Initialize  $\theta_0, \lambda_0, T, \eta_{\theta}, \eta_{\lambda}, \delta, \epsilon$ 
while not converged do
    Rollout a trajectory with the policy  $\pi_{\theta_k}(s)$ 
    Estimate primal gradient  $\nabla_{\theta} L(\theta_k, \lambda_k)$ 
    Estimate dual gradient  $U(\theta_k) - c$ 
    Update primal variable:  $\theta_{k+1} = \theta_k + \eta_{\theta} \nabla_{\theta} L(\theta_k, \lambda_k)$ 
    Update dual variable:  $\lambda_{k+1} = \lambda_k - \eta_{\lambda} (U(\theta_k) - c)$ 
end while
```

C. Handling Partial Observability

The previous section described how we can optimize the policy using the primal-dual approach. This section presents the specific architecture we use for the actual policy. Stemming from [21], we use a Soft Actor-Critic Model as it generally tends to have better sample efficiency. To deal with partial observability, Recurrent Neural Networks (RNN) have been known to mitigate the effects of a noisy observation by making decisions based on the past trajectory instead of just the current observation (or fixed sequence of observations) [25] [26]. We implement an LSTM module to help stabilize training [27]. Initial training results showed that a deterministic actor would more often converge to a uniform policy (only moving or only localizing), therefore

we decided to optimize the results with a stochastic actor. All of our embeddings are obtained with a one-layer MLP. Figure 3 provides an illustrative diagram of the architecture.

V. EXPERIMENTAL EVALUATION

In this section, we report our findings from numerical experiments comparing RISKRL with several baselines and evaluating the robustness and generalization capabilities of RISKRL. The training and testing environment are motivated by an underwater scenario introduced in our prior work [2]. In the scenarios, a robot is tasked with navigating through complex underwater environments, which include obstacles such as rocks and coral formations. The robot must perform localization and path planning while accounting for the unique challenges posed by water currents and variable visibility. Our experimental code is located on GitHub².

A. Setup

Baselines. We compare RISKRL with four types of baselines. The first type is two static policies (SP): $(M2x, L)$, and $(M3x, L)$, where for example, $(M2x, L)$ is a fixed repeating sequence of two move actions followed by a localize action. The second type is a threshold planner (TP) that localizes the robot whenever the probability of failure (computed using the PF output) exceeds a threshold. The third type is an implementation of CC-POMCP, a cost-aware, online policy planner from our prior work [2]. Finally, the last type is a standard, risk-unaware RL policy (BASERL). In the latter, we penalize every time the robot chooses a localize action (-1) and when it collides (-256). This baseline is used to assess the advantage of employing risk-aware RL.

Environments. Figure 4 shows our environmental setup. We assume the robot knows the map, the start state, and the goal region. We set the initial belief to the start state. In all of our experiments, we set the transition noise such that the robot has an 80% chance of going in the direction it is facing, a 10% chance of drifting to the left, and a 10% chance of drifting to the right. Other than the results in Section V-E, we assume no observation noise when localizing to focus more on the effect of transition noise. Figure 4 visually represents our environments.

Training Process. We trained each of the planners on the `train` environment, allotting 8 hours for BASERL and 12 hours for RISKRL to reach 200 episodes. The parameters of the BASERL robot and the proposed RISKRL robot are defined in Table I.

The following subsections compare the performance of our baselines and the proposed algorithm. Our experiments ran until the robot reached the goal or failure region. The results were based on 100 runs for each algorithm in each environment (except CC-POMCP, which we only ran 75 times in the `train` environment since it takes around 30X more time to get through a trajectory compared to the rest of the planners). All bars depict the mean values in Figures 7a and 7b.

²<https://github.com/raaslab/when-to-localize-riskrl>

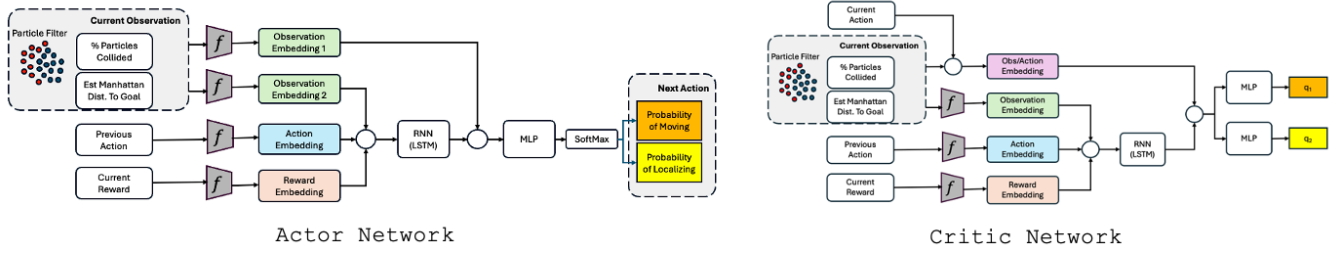


Fig. 3: A diagram of the RL architecture used to train the models

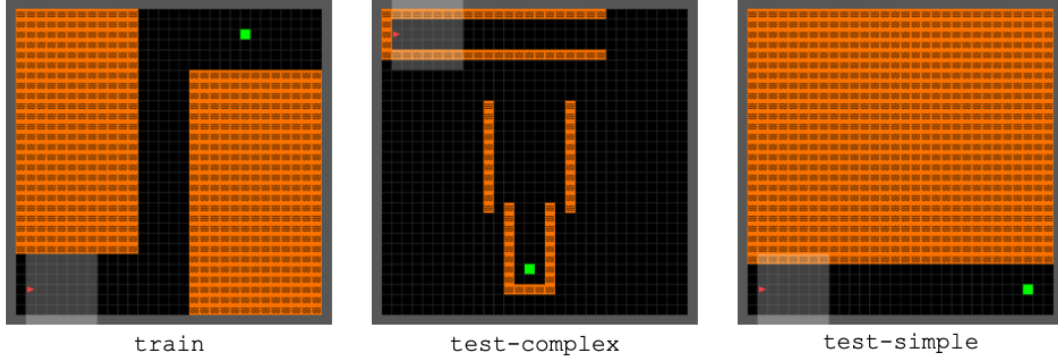


Fig. 4: This figure shows the three Minigrid environments used in our evaluations. The red triangles represent the robots, the green squares represent the goal, and the orange squares represent the failure states. We trained the RL robots using the train environment. All environments are in a 32x32 grid.

Planner	r_{goal}	r_{move}	r_{local}	r_{fail}	# Particles	ℓ_r	γ	α	τ	DQN Layers	Policy Layers	Obs Emb. Size
BASERL	0	0	-1	-256	100	0.00012	0.95	0.25	0.005	[64, 64]	[64, 64]	32
RISKRL	"	"	"	0	"	0.0001	0.9	0.5	"	[128, 128]	[128, 128]	32

TABLE I: RL Parameters. ℓ_r is the learning rate, γ is the discount factor, α is the target entropy, and τ is the soft update parameter

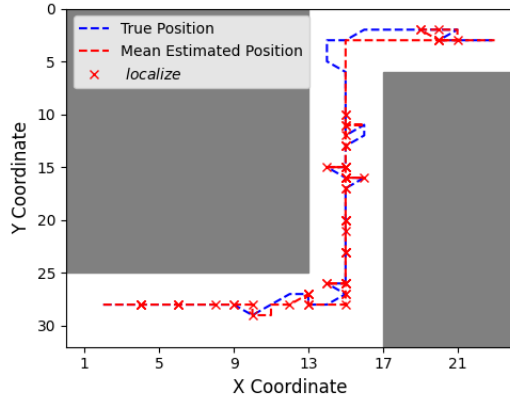


Fig. 5: A qualitative example of robot navigation: **blue** true path, **red** estimated path and localization points.

B. Qualitative Example

In Figure 5, we present a qualitative example demonstrating the robot’s behavior in a noisy and uncertain environment, characterized by (80%, 10%, 10%) transition noise and 32% observation noise, with a risk threshold set at

40%. The results show that the robot consistently applies localization in the Start Area and Middle Tunnel to mitigate the risk of early failure. Additionally, the robot increases the frequency of localization within the Middle Tunnel, where the risk of failure is higher, and noise can cause deviations from the intended path. Since the area after the Middle Tunnel is a safe zone, the robot does not perform localization there. Finally, the robot localizes upon reaching the goal to ensure precise positioning for successful task completion. This adaptive behavior highlights the robot’s strategy to navigate effectively under challenging conditions.

C. Comparing with Baseline

We compared all algorithms in terms of the number of localization actions and success rates (that is, reaching the goal). The SP, TP, and CC-POMCP results represent the average performance of each policy type. We found that we can categorize the algorithms into two main groups: (M2xL), (M3xL), and RISKRL tend to have higher success rates but also localize more often while TP, BASERL, and CC-POMCP tend to localize less but suffer from lower success rates. Our results showcase the strengths of the RISKRL planner. In our experiments, we set $c = 0.4$ for general training

and the policy successfully achieves this threshold. The success rate in the testing environments is also anticipated to be lower than the risk threshold, but not by a significant margin. Although, the RISKRL planner performs comparably worse to (M3xL) in the `train` environment as they both have a success rate of 93% with (M3xL) needing fewer localizations, the RISKRL planner generalized very well to the `test-complex` and `test-simple` environments. In the `test-complex` environment, our RISKRL planner used a similar amount of localizations to (M3xL) while having a 13% higher success rate. In the `test-simple` environment, our RISKRL planner achieved a 99% success rate, albeit with more localizations than (M3xL). One of the main strengths of RISKRL is that it can keep consistently high success rates, on par if not better than (M2xL) while having constantly lower number of localizations than (M2xL) and even on par number of localizations than (M3xL) in the `test-complex` environment.

We found that BASERL ends up performing slightly better than the TP for both metrics in `train` and `test-simple`. However, when looking at the `test-complex` environment, BASERL was able to somewhat generalize to the environment and achieve a 35% success rate. In contrast, TP was only able to achieve a 21% success rate.

Compared to CC-POMCP, our results indicate that our BASERL model performs very similarly with a slightly higher success rate but also a slightly higher number of localizations in the `train` environment. While every other planner can compute the next action nearly instantly, it takes CC-POMCP around 20 minutes to get through a trajectory.

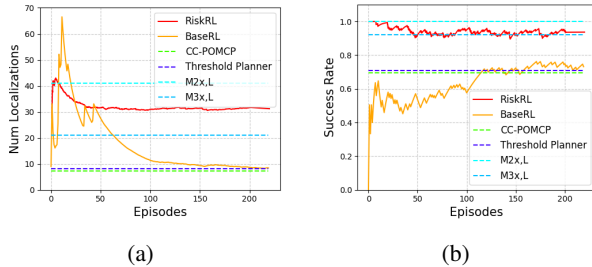


Fig. 6: (a) Training Graph showing Number of Localizations (b) Training Graph showing Success Rate. Refers to `train` environment. For RISKRL, c was set to 0.4.

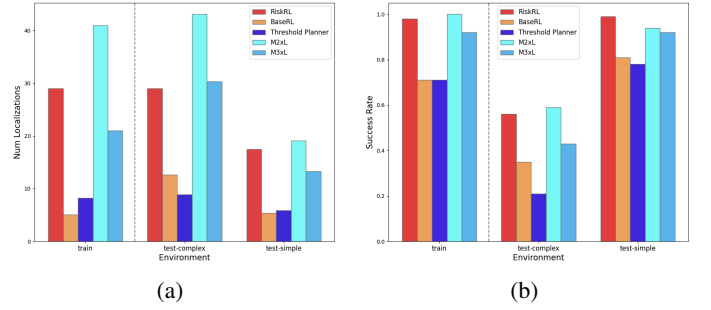


Fig. 7: (a) Number of Localizations in each environment (b) Success Rate in each environment.

D. Analysis of RL Policies

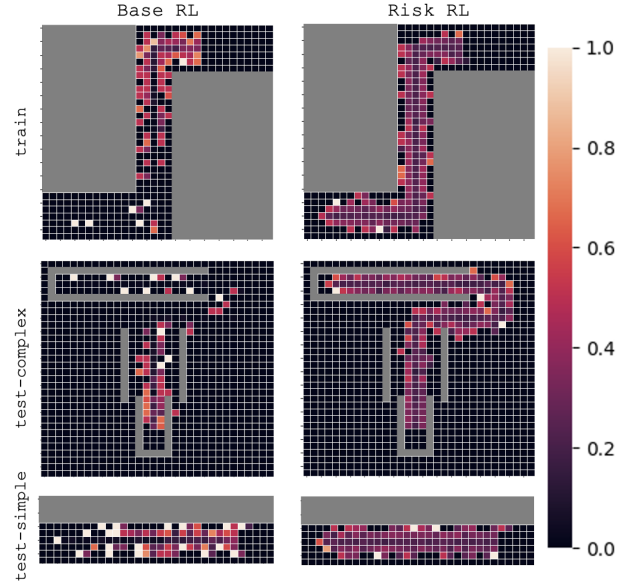


Fig. 8: Probability of localization based on the robot's estimated location.

Figure 8 showcases each of the RL planners' decision-making based on their estimated localization. In general, we see that BASERL concentrates its localizations in fewer locations that are either close to obstacles or around a turn. On the other hand, RISKRL is much more uniform where it is willing to localize at more locations along the path with higher concentrations around obstacles.

E. Effect of Transition and Observation Noise

Varying Transition Noise. Figures 9a and 9b illustrate the number of localizations and success rates for RISKRL as we vary the transition noise. In all the experiments, the allowable probability of failure was set to $c = 0.4$. Not surprisingly, the number of localization increases as the transition noise increases. But more crucially, we observe that RISKRL can respect this constraint even when the transition noise is very high, reaffirming the correctness of the algorithm.

Varying Observation Noise. In the previous results, we had no observation noise when the robot chose to obtain a

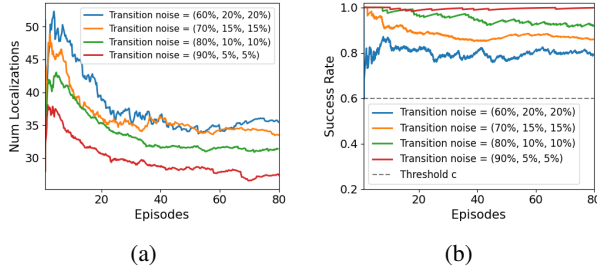


Fig. 9: (a) Number of Localizations vs. Transition Noise Levels. (b) Success Rate vs. Transition Noise Levels. The results show training performance in train.

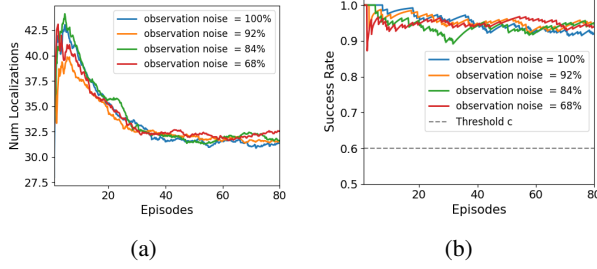


Fig. 10: (a) Number of Localizations vs. Different Observation Noise Levels. (b) Success Rate vs. Different Observation Noise Levels

measurement. Here, we study the effect of the observation noise level on the performance of RISKRL. Figures 10a and 10b demonstrate that observation noise does not impact the robot’s performance using our algorithm. This experiment draws the observation from the 3×3 region around the true position with a truncated Gaussian distribution. The value shown in the figures is at the center of the 3×3 region. The results indicate that, despite increasing observation noise, the success rates remain relatively consistent across different levels. A similar trend is observed for the number of localizations, suggesting that our algorithm maintains robust performance in the presence of observation noise.

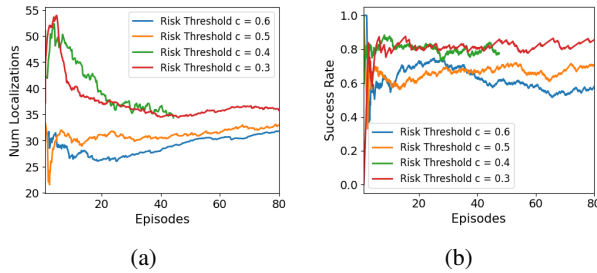


Fig. 11: (a) Number of Localizations vs. Different Threshold Levels. (b) Success Rate vs. Different Threshold Levels

F. Effect of Risk Constraint

Figures 11a and 11b depict the number of localizations and success rates under varying levels of the risk threshold. The results demonstrate that both metrics are influenced by

changes in the risk threshold. As the threshold increases, the robot adopts more conservative movement strategies to minimize risk, resulting in a higher frequency of localization and fewer failures. Conversely, with a lower risk threshold, the robot engages in more aggressive actions, leading to increased uncertainty and a higher likelihood of failure. Consequently, the success rate diminishes as the risk constraint becomes more lenient.

VI. CONCLUSION

We developed a novel active localization approach termed RISKRL, which combines a chance-constrained planner with a particle filter (PF). Our approach aims to minimize localization actions while not exceeding failure probabilities. The chance-constrained planner determines when the robot moves or localizes and was implemented using constrained reinforcement learning (RL). The PF maintains the robot’s belief by processing noisy 2D pose observations from the environment and motion commands. We also use the PF’s belief to compute the observation for our RL planner. The current approach succeeds our prior work [2], which employed an algorithm that has a slower inference time and requires well-defined transition and observation noise models, making it unusable in real-time, real-world scenarios. Our results revealed three key findings. First, RISKRL consistently outperformed the baseline by at least 13%, while maintaining a similar number of localizations. This demonstrates the robot’s ability to optimize the timing of localization actions, achieving higher rewards. Second, the robot dynamically adjusts the frequency of localizations, showcasing its adaptability to varying scenarios and environmental conditions. Finally, through bi-criteria optimization, RISKRL effectively controls risk levels while maximizing performance, ensuring the robot operates safely. Notably, RISKRL demonstrates zero-shot transfer capabilities, handling new environments without retraining, underscoring its potential for real-world deployment.

We believe that our models hold the potential for further optimization to achieve improved results. One promising direction is the implementation of *Evolving Rewards* [28] to identify a more effective set of reward values. Additionally, we aim to extend the current formulation to encompass both continuous state and action spaces, with the expectation that this will facilitate deployment in real-world scenarios.

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] T. Williams, K. Torshizi, and P. Tokekar, “When to Localize?: A POMDP Approach,” in *2024 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2024.
- [3] 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

- [4] S. Paternain, M. Calvo-Fullana, L. F. O. Chamon, and A. Ribeiro, "Safe policies for reinforcement learning via primal-dual methods," *IEEE Transactions on Automatic Control*, vol. 68, no. 3, pp. 1321–1336, 2023.
- [5] 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.
- [6] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," *Autonomous Robots*, vol. 42, pp. 177–196, 2018.
- [7] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, p. 1346–1352.
- [8] 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.
- [9] 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.
- [10] T. Sasaki, K. Otsu, R. Thakker, S. Haesaert, and A.-a. Aghamohammadi, "Where to Map? Iterative Rover-Copter Path Planning for Mars Exploration," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2123–2130, 2020.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.
- [15] 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>
- [16] 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.
- [17] 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.
- [18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: <https://arxiv.org/abs/1707.06347>
- [19] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016. [Online]. Available: <https://arxiv.org/abs/1602.01783>
- [20] L. Meng, R. Gorbet, and D. Kulić, "Memory-based deep reinforcement learning for pomdps," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 5619–5626.
- [21] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free rl can be a strong baseline for many pomdps," 2022. [Online]. Available: <https://arxiv.org/abs/2110.05038>
- [22] S. Amani, C. Thrampoulidis, and L. Yang, "Safe reinforcement learning with linear function approximation," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 243–253. [Online]. Available: <https://proceedings.mlr.press/v139/amani21a.html>
- [23] Q. Liang, F. Que, and E. Modiano, "Accelerated primal-dual policy optimization for safe reinforcement learning," 2018. [Online]. Available: <https://arxiv.org/abs/1802.06480>
- [24] M. Yu, Z. Yang, M. Kolar, and Z. Wang, "Convergent policy optimization for safe reinforcement learning," 2019. [Online]. Available: <https://arxiv.org/abs/1910.12156>
- [25] J. N. Knight and C. Anderson, "Stable reinforcement learning with recurrent neural networks," *Control Theory and Technology*, vol. 16, no. 1, pp. 65–80, 2008.
- [26] J. Ho, J. Xu, L. Sha, and Z. Qu, "Toward a brain-inspired system: Deep recurrent reinforcement learning for a simulated self-driving agent," *Frontiers in Neuroscience*, vol. 11, pp. 153–162, 2017.
- [27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [28] A. Faust, A. Francis, and D. Mehta, "Evolving rewards to automate reinforcement learning," in *6th ICML Workshop on Automated Machine Learning*, 2019. [Online]. Available: <https://arxiv.org/abs/1905.07628>